FUNDAMENTAL STUDY OF EVAPORATION MODEL IN MICRON PORE

A Dissertation

by

RYOJI OINUMA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2004

Major Subject : Nuclear Engineering

FUNDAMENTAL STUDY OF EVAPORATION MODEL IN MICRON PORE

A Dissertation

by

RYOJI OINUMA

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Frederick R. Best (Chair of Committee)	Marvin L. Adams (Member)
Yassin A. Hassan (Member)	Sai Lau (Member)
William E. Burchill (Head of Department)	

August 2004

Major Subject: Nuclear Engineering

ABSTRACT

Fundamental Study of Evaporation Model in Micron Pore. (August 2004)

Ryoji Oinuma, B.S., Tokai University;

M.S., Tokai University

Chair of Advisory Committee: Dr. Frederick R. Best

As the demand for high performance small electronic devices has increased, heat removal from these devices for space use is approaching critical limits. A heat pipe is a promising device to enhance the heat removal performance due to the phase change phenomena for space thermal management system. Even though a heat pipe has a big potential to remove the thermal energy from a high heat flux source, the heat removal performance of heat pipes cannot be predicted well since the first principle of evaporation has not been established. The purpose of this study is to establish a method to apply the evaporation model based on the statistical rate theory for engineering application including vapor-liquid-structure intermolecular effect. The evaporation model is applied to the heat pipe performance analysis through a pressure balance and an energy balance in the loop heat pipe.

ACKNOWLEDGEMENTS

I would like to express my special thanks to my dissertation advisor, Dr. Frederick R. Best, who has given me advice and who has supported me not only financially, but also spiritually, with his great patience. I would also like to thank Cable Kurwitz, manager of the Interphase Transport Phenomena Laboratory at Texas A&M University, for his help and advice on my work and dissertation. I would like to acknowledge funding for this project by the NASA Commercial Space Center and the Center for Space Power within the Texas Engineering Experimental Station at Texas A&M University. Finally, I would like to thank my parents, Toshiaki Oinuma and Eiko Oinuma, for their priceless affection.

TABLE OF CONTENTS

Page

CHAPTER Page

NOMENCLATURE

Symbol	Description
a	Length of Triangle
A_{evap}	Evaporative Area on Liquid-Vapor Interface
A_{pore}	Pore Area $[m^2]$
A_{source}	Heat Source Area $[m^2]$
b	Post Width $[m]$
B	Hamaker Constant $[J]$
C	Atom-atom Air Potential Coefficient
C_b	Post Width to Unit Cell Length Ratio (=b/w)
C_f	Friction Factor
C_p	Post Pitch to Pore Diameter Ratio (=P/d)
C_v^{α}	Specific Heat Capacity for Constant Volume $[J/kg/K]$
d	Pore Diameter $[m]$
d_j	Oscillator Strength
D_l	Liquid Transport Line Diameter $[m]$
D_v	Vapor Transport Line Diameter $[m]$
F	Intermolecular Force $[N]$
g	Bandwidth
h	Conduction Post Height $[m]$

h_l	Liquid Enthalpy $[J/kg]$
h_v	Vapor Enthalpy $[J/kg]$
H	Gap between Surfaces
h_{fg}	Latent Heat $[J/kg/K]$
I	Product of Principal Moments of Inertia of Molecule
J_{LV}	Evaporation Flux $[kg/s]$
k	Boltzmanns Constant
k_{wall}	Wall Thermal Conductivity $[W/m/K]$
K_{bottom}	Bottom Curvature of Pore $[1/m]$
l	Transport Line Length $[m]$
L	Evaporator Length $[m]$
$\dot{m}_{cell}$	Evaporation Rate per Unit Cell $[kg/\text{sec}]$
$\dot{m}_{pore}$	Evaporation Rate per pore $[kg/\text{sec}]$
M	Molecular Mass [kg]
n	Number of Pore in Unit Cell
n'	Number of Vibrational Degree of Freedom
N_A	Avogadro's Number
N^{α}	Number of Molecular for Phase α
p	Integration Variable for Intermolecular Force
P	Conduction Post Pitch $[m]$
ΔP_{cap}	Capillary Pressure Difference $[Pa]$

ΔP_{liq}^{line}	Liquid Pressure Drop in Transport Line $[Pa]$
ΔP_{liq}^{wick}	Liquid Pressure Drop in Evaporator Wick $[Pa]$
P_{sat}	Saturation Pressure $[Pa]$
P_{ve}	Equilibrium Pressure $[Pa]$
ΔP_{vap}^{line}	Vapor pressure Drop in Transport Line $[Pa]$
ΔP_{vap}^{wick}	Vapor Pressure Drop in Evaporator Vapor Channel $[Pa]$
Pr	Prandtl Number
q_{elec}	Electronic partition function
q''_{evap}	Heat Flux due to Evaporation $[W/m^2]$
q_{rot}	Rotational Partition Function
q''_{source}	Heat Flux of Heat Source $[W/m^2]$
q_{vib}	Vibrational function
R_c	Pore Radius $[m]$
Re_{Dh}	Reynolds Number for Hydraulic Diameter
ΔS	Entropy Difference between Liquid and Vapor
T_l	Liquid Temperature $[K]$
T_{li}	Liquid Interface Temperature $[K]$
T_v	Vapor Temperature $[K]$
T_{vi}	Vapor Interface Temperature $[K]$
U^{α}	Internal Energy of Phase α $[J]$

v_l	Liquid Specific Volume $[m^3/kg]$
v_v	Vapor Specific Volume $[m^3/kg]$
v_p	Liquid Velocity in Pore $[m/s]$
v_v	Specific Volume per Unit Molecule $[m^3]$
$\lvert V_{v\varepsilon} \rvert$	Matrix Element for a Change from Quantum State of λ_j to λ_k
w	Unit Cell Length $[m]$

Greek Symbol	Description
δ	Gap between mediums or Film Thickness
δt	Interval Time $[s]$
ε	Dielectric Constant
ζ	Energy Density of States within Energy Uncertainty $[J/kg]$
θ	Angle of Interface with respect to Pore Wall
λ_j	Molecular Distribution
μ_l	Liquid Viscosity $[N \cdot s/m^2]$
μ_{li}	Liquid Chemical Potential $[J/kg]$
μ_v	Vapor Viscosity $[N \cdot s/m^2]$
μ_{vi}	Vapor Chemical Potential $[J/kg]$
ρ_1	Number Density of Atoms $[1/kg]$

ρ_l	Liquid Density $[kg/m^3]$
ρ_v	Vapor Density $[kg/m^3]$
σ	Surface Tension $[N/m]$
σ_s	Symmetry factor of Vibrational Orientation
τ	Resonant Frequency $[s]$
Π	Disjoining Pressure $[Pa]$
Θ_l	Characteristic Temperature for Vibration
$\Omega(\lambda_j)$	Number of Quantum States available to System

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

For decades, air has been the preferred fluid for cooling electronic devices due to the simplicity, low cost, ease of maintenance and high reliability. High thermal performance requirements for integrated circuits and power-semiconductor markets as well as the advanced space platforms are driving the need for high heat flux and advanced thermal management devices. Thus, due to the limits of single phase cooling, there is growing interest in cooling by phase change phenomena because of its high potential heat flux capability.

Phase change heat transfer coefficients are normally 10-1000 times larger than typical forced vapor or liquid convection. Many heat pipes' heat removal capability is high enough to remove waste heat flux generated in devices. However, the conventional heat pipe performance analysis cannot accurately predict the performance due to the inaccuracy of the evaporation model. In past studies, temperature was assumed to be uniform in the entire evaporator and controlling phenomena in the evaporator operating on the order of micron meter were not considered. Although studies of evaporation phenomena in wicks have been conducted by many researchers, these studies were de-coupled from overall heat pipe performance.

This dissertation follows the style of *Nuclear Technology*.

In addition, the conventional evaporation model based on kinetic theory contains empirical parameters, which are usually determined by experiment and are strongly affected by conditions in the vicinity of the interface such as the shape of the liquid-vapor interface, pressure, and temperature. Due to this, the design of heat pipes has depended on experiments to establish performance parameters. Thus, even though a heat pipe has a high potential to remove thermal energy from a high heat flux source, the performance of heat pipes cannot be predicted well analytically due to inaccurate assumptions and the requirement for knowledge of empirical parameters.

Until now, the statistical rate theory has not been applied to evaluate the evaporation rate in heat pipes since the model is not intrinsically designed to use in the engineering application, relying on a precise knowledge of temperatures in the immediate vicinity of the liquid-vapor interface. The purpose of this dissertation is to establish a method to apply the evaporation model based on the statistical rate theory for engineering application including vapor-liquid-structure intermolecular effect.

CHAPTER II
LITERATURE REVIEW

Summary of This Chapter

The bottom line of Chapter II is that, by reviewing past studies, the state-of-the-art heat pipe performance calculation does not predict the real heat pipe performance and analysis of micron-scale phenomena needs to be included to enhance the heat pipe performance modeling. In this chapter, there are six main parts: (1) overview of cooling for electronic devices, (2) experimental investigation of micro heat pipe, (3) macroscopic model development of micro heat pipe, (4) microscopic model development of micro heat pipe, (5) phase change phenomena in a pore, (6) evaporation model development. First, the history of cooling system for electronic devices is explained and shows the need for a heat pipe in this region. In the next section, the performance of current heat pipes and model development to calculate the performance are shown. These sections also show that current heat pipe performance modeling as well as demonstrating will not meet the demand for handling high heat source in the near future and mismatch between the real heat performance and the performance calculation. Sections (5) and (6) give a history and a basic idea of the evaporation models and shows how to enhance the entire heat pipe performance.

Overview of Cooling for Electronic Devices

Currently the most common fluid for cooling integrated circuits(IC) is air. A typical heat

transfer coefficient to air convection is around $10-200[W/m^2K]$[1] which corresponds to a heat flux of less than $1.0\times10^3[W/m^2]$ or $0.1[W/cm^2]$ under realistic temperature conditions for such devices. Air's low thermal conductivity ($\approx10^{-3}[W/mK]$) and Prandtl number ($\mathrm{Pr}=C_p\mu/k\approx0.7$) hinder heat removal by convection and its low density and specific heat minimize its ability to transfer thermal energy without incurring an extremely high temperature rise.

Due to the recent development of integrated circuits with denser components making faster processing and increasing power dissipation, indirect or direct liquid cooling has been used. Indirect liquid cooling involves the attachment of a cold plate to the IC module. Thermal energy is transferred through an intermediate structure to the cold plate. The thermal resistance between the cold plate and the module may be eliminated by making the cold plate an integral part of the module. Also to increase the effective area per unit volume, the micro channel heat sink with length scales as small as $0.1\mu m$ has been developed. By this concept, the convection heat transfer coefficient reaches $2000-3000[W/m^2K]$[2]. The constraint of this concept is the pressure drop in the micro-channel.

Liquid jet impingement cooling is a good option for dissipating large heat fluxes. There are two basic types of impinging jet: a free surface or a submerged jet. A free-surface jet is discharged into an ambient gas, whereas a submerged jet is discharged into a liquid of the same type. Because of the thin hydrodynamic and thermal boundary layers which

form at the impingement surface, convection heat transfer coefficients are large$(\approx 1000 - 10000(W/m^2K))$.

Several studies have considered the means by which heat transfer may be enhanced for liquids flowing through channels. Enhancement may be achieved by increasing the surface area for convection and disrupting the thermal boundary layer. The averaged heat transfer coefficient may be $10^4(W/m^2K)$ for water as a working fluid.

However, thermal energy dissipated from electronic devices is increasing and will reach over $100(W/cm^2)$ within several years. Due to the limit of single phase cooling, there is growing interest in cooling by phase change phenomena due to the high potential heat removal capability. In heat pipes, thermal energy is transferred by the heat of vaporization from evaporator to the condenser. These systems are passive and do not require external power to operate. In a heat pipe liquid is returned to the evaporator by surface tension or capillary forces acting in a porous or grooved material. A heat pipe is also a promising device to handle high heat flux and it is not limited by the gravitational environment.

Heat Pipe Performance Investigation

Experimental Investigation of Micro Heat Pipe

Cotter(1984)[3] proposed a micro heat pipe concept for cooling of electronic devices. The micro heat pipe was defined as a heat pipe in which the mean curvature of the liquid-vapor interface is the same order of magnitude as the radius of the flow channel

shown as

$K \propto 1/r$. (2-1)

A flat heat pipe with grooved edge inside a tube designed by Cotter was reported to transport 0.3W which corresponds to $1(W/cm^2)$. Babin et al. (1989)[4] designed a trapezoidal channel heat pipe. The material of the heat pipe was copper and the working fluid was water. This heat pipe transported 0.5W which corresponded to $1(W/cm^2)$. Wu and Peterson (1990)[5] tested a flat plate heat pipe and obtained $5(W/cm^2)$. Plesch et al.(1991)[6] compared two types of flat heat pipes. One has grooved edges transversing the vapor flow and another has grooved edges parallel to the vapor flow. These heat pipes obtained $35(W/cm^2)$. The main difference between previous designs and Plesch's design is the shape of the grooves: the former design has a triangle groove shape and the latter has a rectangular shape. This performance difference can be explained by the difference in effective evaporative area. Plesch's design has larger evaporative area per unit extended surface area.

Zhou et al.[7] studied the operational characteristics of a miniature heat pipe with copper and acetone. They found that the maximum operating power of the heat pipe was limited by the capillary limit which is mostly affected by the temperature of air. Li et al. (1992)[8] tested cylindrical miniature heat pipes with an inner diameter of 1.2mm. They concluded that the influence of entrainment on the capillary limit is more important than a conventional performance analysis. Khrustalev and Faghri (1994)[9] designed a miniature heat pipe with triangular cross section and evaluated its characteristics. The working

fluid flows at the corner of the triangle in such a design.

Macroscopic Model

The first steady-state model for micro heat pipe was presented by Cotter (1984)[3]. In this study the heat pipe performance was evaluated based on the capillary limit. The result was given as:

$$Q_{\max} = \left(\frac{0.16\beta\sqrt{K_l^+ K_v^-}}{8\pi H(1)} \right) \left(\frac{\sigma h_{fg}}{\nu_l} \right) \left(\frac{\nu_l}{\nu_v} \right)^{1/2} \left(\frac{A_t^{0.75}}{L_t} \right), \qquad (2\text{-}2)$$

where K_l^+ and K_v^- are the flow shape factors, $H(1)$ is an integral of the fraction of the total heat transport over the length, L_t, of the pipe. A_t is the total cross section area of the heat pipe and β is a geometrical factor. σ is the surface tension. h_{fg} is the latent heat. ν_l and ν_v is the specific volume for liquid and vapor respectively.

Babin et al. (1990)[10] used the capillary pressure limitation to evaluate heat pipe performance also, but viscous pressure drops occurring in the vapor and the liquid were calculated separately. Pressure drops occurring in the heat pipe loop cannot exceed the capillary pressure rise. This relationship can be expressed as

$$\Delta P_{capillary} \geq \Delta P_+ + \Delta P_{\|} + \Delta P_l + \Delta P_v \qquad (2\text{-}3)$$

where $\Delta P_{capillary}$ is the capillary pressure. ΔP_+ and $\Delta P_{\|}$ are pressure drops of radial hydrostatic and axial hydrostatic due to the gravity respectively. ΔP_l and ΔP_v are liquid and vapor pressure drop. They expressed all of the pressure drops occurring in a

heat pipe by the viscous pressure drop as a factor of the heat transfer rate and obtained a maximum heat transfer rate from the capillary limitation. The significance of the vapor pressure losses was pointed out.

Gerner et al.(1992)[11] included a interfacial shear stress at the vapor-liquid interface due to the vapor flow and calculated a maximum heat flux from the capillary pressure limit. Gerner estimated the average film thickness to be approximately one-fourth the hydraulic radius. The final result becomes

$$Q_{\max} = \frac{3\pi}{2048}\frac{\sigma h_{fg} D^3}{v_v L} \qquad (2\text{-}4)$$

where σ is the surface tension, h_{fg} is the latent heat of vaporization. D is the diameter of the vapor path. v_v is the vapor kinetic viscosity and L is the length of the heat pipe.

Kaya et al. (1999)[12] developed a mathematical model of a loop heat pipe to predict heat pipe performance using a thermodynamic approach. The analytical model is applied to two separate LHP designs at different sink temperatures and different elevations. The calculations agreed with the experimental measurement between 0.1% and 3%. However, the input heat flux or heat density was not shown, only the total heat inputs were given. For heat removal at low temperature, the input heat flux was not so high condition and the heat pipe works easily, but this situation does not tell the performance limit for higher heat transfer.

Microscopic Model

In a micro heat pipe, the wick interface meniscus is so small that evaporation and condensation are considered to occur mostly in the thin film area. For micro heat pipes with diameters less than $100\mu m$ a majority of the liquid-vapor interface exists where the liquid would be considered in the thin film regime and the fluid behavior may begin to deviate from bulk liquid behavior. As a result, accurate prediction of the thermal behavior of micro heat pipes smaller than $100\mu m$ in diameter will require development of models which utilize knowledge of the thin film behavior and include the effects of the wetting angle, the disjoining pressure, and van der Waals forces.

Potash and Wayner (1972)[13] have proposed a two dimension model to evaluate the evaporation in a thin liquid film with kinetic theory. Mirzamoghadam et al.(1988)[14] investigated a physical model of the evaporating meniscus for a inclined plate and calculated a heat transfer coefficient, a velocity profile and evaporation rate in the meniscus. The result showed that the heat transfer coefficient between 20 and 30 degrees was a factor of seven relative to the horizontal angle. Xu and Carey (1990)[15] investigated a mechanism of film evaporation in a V-shaped groove channel.

Swanson and Peterson (1994)[16] investigated fluid flow and heat transfer in a V-shaped micron-scale groove. In their investigation, a mathematical model including the momentum equation, the continuity equation, the energy equation with an interfacial condition of thermal energy balance was developed to determine the fluid flow and heat

transfer. This investigation showed that the magnitude of the surface tension effect was much less than that of the disjoining pressure in the thin film region less than 500Å thick and the evaporation rate in the thin film region is significantly higher than the other regions. They calculated the averaged Nusselt number and found that Nusselt number increased as the wedge half-angle decreased.

Thermodynamic Approaches

Richter and Gottschlich (1994)[17] first investigated heat pipe performance from the view point of a thermodynamic cycle rather than operating limitations such as the capillary, sonic, boiling and entrainment limitations since the experimental performance was very far from the predicted performance by these limitations. They concluded that the fluid circulation in a heat pipe is the result of a thermodynamic cycle that operates over the internal temperature difference. The rate at which the working fluid can be circulated is determined by the internal temperature difference that can be achieved and the absolute operating temperature.

H. Khalkhali et al.[18] inspired by Richter and Gottshclchs'[17] analysis sought to minimize the lost work by minimizing entropy losses in a heat pipe cycle. They derived equations to calculate the entropy generation of a typical heat pipe and tried to achieve the minimum entropy generation of a heat pipe from the view point of thermodynamics. They concluded: (1) the vapor temperature must be greater than a certain value in order for the heat pipe to function: (2) a condenser temperature should be close to an optimum

value($T_{L,opt} = \sqrt{T_v T_\infty}$) corresponding to $\frac{dS_{gen}}{dT_L} = 0$ to minimize the entropy generation: (3) both the condenser temperature and the convection heat transfer coefficient must be adjusted to obtain a minimum entropy generation: (4) the evaporator should be designed to reduce the entropy generation due to friction: (5) vapor transport line should be insulated efficiently to avoid entropy generation due to the heat transfer between the vapor and the ambient.

Phase-change Phenomena in Meniscus

Dejaguin et al. (1965)[19] demonstrated that liquid flow in a thin film on the walls of a capillary tube can be enhanced by several times the rate of evaporation of moisture from a capillary. The liquid flow effect due to long-range intermolecular forces was analyzed by using the disjoining pressure concept for the pressure gradient in a thin film.

Wayner et al. (1972)[20] experimentally demonstrated the stability of an evaporating meniscus formed on a vertical flat plate immersed in a pool of saturated liquid. Potash and Wayner (1972)[13] analyzed evaporation from a 2-D extended meniscus in which the pressure gradient for liquid flow was due to both the capillary pressure and the disjoining pressure, balancing the mass flow rate with the evaporation rate. The presence of an adsorbed superheated thin film gave a smooth transition between evaporating and non-evaporating portions of the extended meniscus. They defined three regions based on their characteristic: a non-evaporating region, a transient region or thin film region and an intrinsic region (Fig. 2-1). In the non-evaporating region, the intermolecular

dispersion force (Van der Waals force) between liquid molecules and wall molecules is strong enough to prevent evaporation from the liquid-vapor interface. The intermolecular force is also known as the disjoining pressure. In the thin film region, the surface tension is larger than the disjoining pressure, but still on the same order, so evaporation is occurring while the liquid is held close to the wall. They pointed out that the interfacial temperature which controls the evaporation rate is higher than the temperature in the intrinsic region and the evaporation rate per unit area becomes higher than other regions. The liquid thickness of this region is normally about a nano-meter. In the intrinsic region, the surface tension is dominant and the meniscus is formed. The evaporation rate per unit area is relatively smaller than in the thin film region.

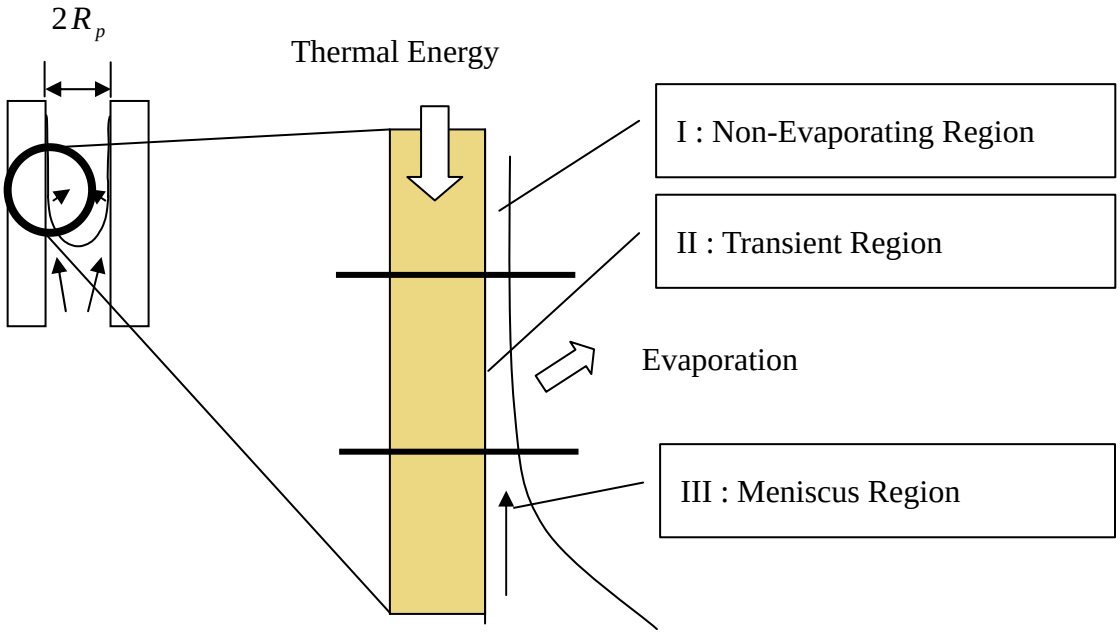

Fig. 2-1. Classification of Evaporating Region

Wayner et al. (1976)[20] developed a one-dimensional model to estimate the average evaporation heat transfer coefficient for a liquid thin film heated in a flat, micron wide slit and gave the result:

$$\bar{h} = h^{id}\left[1 - 0.5\left(\eta^{-1} + \eta^{-2}\right)\right] \qquad (2\text{-}5)$$

h^{id} is the ideal heat transfer coefficient which is defined the evaporation heat transfer coefficient neglecting the adsorption forces. η is a non-dimensionized liquid film thickness. Moosman and Hosmy[21] described the flow in the thin film region by lubrication theory. This concept has been applied to enhance the heat removal performance of micro devices.

Cotter(1984)[3] used these concepts in proposing a micro heat pipe with triangular cross-section and side thickness between 10 and 500 micro meters. Wayner and Schonberg(1991)[22] demonstrated the reduction of the evaporation rate due to the intermolecular force between solid and liquid by comparing with a evaporation rate from a liquid-vapor interface between two slots in which the effect of the intermolecular force between solid and liquid is relatively much smaller than in a capillary tube.

Stephan et al. (1990)[23] and Swanson et al. (1992)[24] analyzed an evaporating meniscus in triangular grooves and circular pores. They developed a two-dimensional model and both results show the existence of the sharp peak in the evaporation profile in the transition region. There were some questionable assumptions in their model. Wayner and Schonberg (1991)[22] found experimentally that the curvature within the thin film

changes with increasing evaporation rate rather than the arbitrary value assumed by Swanson et al.[24]

Chebaro et al. (1992)[25] pointed out that it is not good assumption that the liquid flow in the meniscus region could be described by a Hagen-Poiseuille flow and the boundary condition in the meniscus region should be $\bar{t} \cdot \overline{\overline{T}} \cdot \bar{n} = -\nabla\sigma \cdot \bar{t}$ rather than $\mu_l \frac{\partial u_l}{\partial r} = \frac{d\sigma}{dx}$ which can be applied in the thin film region. $\bar{t}$ and $\bar{n}$ are the tangential and the normal vector on the liquid-vapor interface respectively. $\overline{\overline{T}}$ is a stress tensor. Also they indicated that the meniscus shape, the film length and the absorbed film thickness changed as the evaporation rate changed.

Sujanani and Wayner (1992)[26] measured the thickness profile of 1,1,2-Trichlorotrifluoroethane using microcomputer based image processing of interferomic images. They found that the curvature increased very rapidly from zero at the absorbed film to a large value in the intrinsic meniscus, which is consistent with theoretical models. The phase change heat transfer and fluid flow in thin films are strongly coupled by their common dependence on the intermolecular force field and gravity. Stephan and Hammer (1994)[23] used interfacial models to evaluate the influence of meniscus shape, adhension forces, interfacial thermal resistance, and wall thermal resistance on nucleate boiling heat transfer with R114 and R-12 on a copper plate. For a single bubble and microlayer between the bubble and the copper plate, they calculated temperature and heat flux profiles for boiling heat transfer which agreed with

macroscopic experimental results. DasGupta, Schonberg and Wayner (1993)[27] and DasGupta, Kim and Wayner(1994)[28] compared the measured meniscus profile with a calculated profile. The thickness and curvature profiles formed by using interfacial transport phenomena models agreed with measured results.

Analysis of the interfacial phenomena on the micron order was applied to evaluate micro heat pipe performance. The micro heat pipe has been extensively studied by Wu and Peterson (1991)[29] and by Khrustalev and Faghri (1993)[30], Ha and Peterson (1998)[31]. Schonberg, DasGupta and Wayner (1995)[32] investigated evaporation in a micro-channel which could be applied to high heat flux source cooling. They designed an inverted meniscus evaporator to improve the thermal resistance between the evaporator and the heat source. They obtained a heat flux of $(1.3-1.6)\times 10^6 (W/m^2)$ for heptane. They pointed out the importance of the apparent contact angle which has a strong effect on the area of the meniscus and the capillary suction of the working fluid. Swanson and Peterson (1995)[33] add the effect of an axial temperature difference, changes in local interfacial curvature, Marangoni surface effect to the conventional method. They calculated the maximum heat transfer rate for a miniature heat pipe with triangular wick combining Clausius-Claperyron and the liquid and vapor pressure difference by calculating liquid and vapor pressure drop in the system.

Evaporation Model

Evaporation Model Based on Kinetic Theory

Evaporation phenomena have been studied for a long time. Schrage (1953)[34] used the Maxwell-Boltzmann velocity distribution to calculate the rate of mass transfer across an evaporating flat interface and obtained

$$\dot{m} = \frac{1}{\sqrt{2\pi Mk}}\left(\frac{\sigma_e}{1-0.5\sigma_e}\frac{P_e(T_{li})}{\sqrt{T_{li}}} - \frac{\sigma_c}{1-0.5\sigma_c}\frac{P_v}{\sqrt{T_{vi}}}\right) \tag{2-6}$$

where $\dot{m}$ is the mass flux, M is the molecular weight, k is Boltzmann's constant, σ_e and σ_c are the evaporation and condensation coefficients, P_e is the equilibrium pressure which is given as:

$$P_{ve} = P_{sat}(T_l)\exp\left\{\frac{v_l\left[P_{ve} - P_{sat}(T_l) - \sigma K + \Pi\right]}{RT_l}\right\} \tag{2-7}$$

where P_v is the vapor pressure, T_{li} and T_{vi} are the liquid and vapor interfacial temperatures. This model has been modified by Potash and Wayner(1972)[13] and Wayner(1991)[35] using the Clausius-Clapeyron equation, Young-Laplace equation for $(p_v - p_l) << 1[\mathrm{N/m^2}]$, $(T_v - T_l) << 1[\mathrm{K}]$ and the disjoining pressure to obtain the evaporation rate for a curved interface:

$$\dot{m} = a\left(T_{li} - T_{vi}\right) - b(\Pi + \sigma K) \tag{2-8}$$

where

$$a = \frac{2C}{2-C}\left(\frac{M}{2\pi R_{gas}T_{li}}\right)^{1/2}\left(\frac{P_v M h_{fg}}{R_{gas}T_{vi}T_{li}}\right), \tag{2-9}$$

$$b = a\left(\frac{T_{vi}V_l}{Mh_{fg}}\right), \mathrm{C} = \sigma_{\mathrm{e}} = \sigma_c, \tag{2-10}$$

and

$$\Pi = \begin{cases} -\frac{A_{slv}}{\delta^3} \text{ for } \delta < 10\text{nm} \\ -\frac{B_{slv}}{\delta^4} \text{ for } \delta > 15\text{nm} \end{cases} \tag{2-11}$$

A_{slv} is the Hamaker constant for a nonretarded dispersion force and B_{slv} is the retarded dispersion force constant. M is the molecular weight, h_{fg} is the latent heat, V_l is the specific volume of liquid and R_{gas} is the gas constant, K is the curvature of the interface and Π is the disjoining pressure.

The difficulty of using the evaporation model based on kinetic theory is that it is dependent on free parameters, i.e., evaporation (σ_e) and condensation (σ_c) coefficients. σ_e is the fraction of the liquid molecules crossing the interface due to the evaporation. The remaining fraction, $1-\sigma_e$, is due to the reflection of vapor molecules that strike the interface but do not condense. σ_c is similar but for condensation from the vapor phase. We can not predict by kinetic theory how many vapor molecules striking the liquid side become liquid or bounce back to the vapor side. Further, these coefficients have been shown by many researches to scatter widely depending on the condition of the interface[36].

Cao and Faghri (1993)[37] used kinetic theory to simulate the startup behavior of a liquid

metal heat pipe from a frozen condition and they had good agreement with experiments for an accommodation coefficient equal to unity. However, for a low temperature working fluid, no researcher has succeeded in obtaining a consistent result with experiments using kinetic theory. In addition, the temperature difference between the liquid and vapor near the interface predicted by kinetic theory is not consistent with the recently measured temperature profile[38,39] as described in the next section. If we applied this measured temperature profile to Eq. (2-6), the evaporation rate by kinetic theory would be negative although the net evaporation rate is known to be positive.

Evaporation Model Based on Statistical Rate Theory

The classical evaporation model based on kinetic theory predicts that liquid could evaporate only if the vapor temperature in the vicinity of the liquid-vapor interface is less than the liquid interfacial temperature (Aoki and Cercignami, 1983)[40]. However, Ward and Fang (1999)[41] measured the temperature profile normal to a liquid-vapor interface with a movable 25.4μm thermocouple and found the liquid was evaporating under the opposite temperature gradient. Fig. 2-2 shows their experimental apparatus and the typical vertical temperature profile across the liquid-vapor interface. T_{li} and T_{vi} are the liquid and vapor temperature near the liquid-vapor interface. In their experiment the vapor temperature was always measured to be greater than the liquid temperature contradicting kinetic theory.

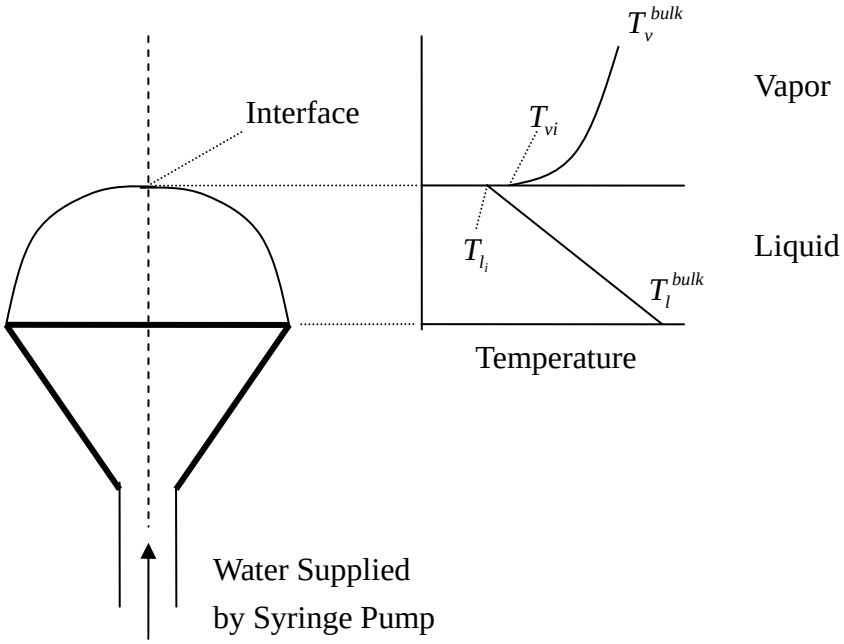

Fig. 2-2. Experimental Apparatus to Measure Vertical Temperature Profile Across Liquid-Vapor Interface

They investigated the possibility of predicting evaporation phenomena using statistical rate theory which is based on the transition probability concept of quantum mechanics, and used the Boltzmann definition of entropy to introduce a thermodynamic description of the evaporating system. They developed a relationship:

$$\dot{m} = \frac{M}{N_A}\frac{P_{ve}}{\sqrt{2mkT_{li}}}\left\{\exp\frac{\Delta S}{k} - \exp\frac{-\Delta S}{k}\right\}, \tag{2-12}$$

where

$$P_{ve} = P_{sat}(T_l)\exp\left\{\frac{v_l\left[P_{ve} - P_{sat}(T_l) - \sigma K + \Pi\right]}{RT_l}\right\}, \tag{2-13}$$

$$\frac{\Delta S}{k} = \frac{1}{k}\exp\left[\left(\frac{\mu_l}{T_{li}} - \frac{\mu_v}{T_{vi}}\right) + h_v\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\right]$$

$$= 4\left(1 - \frac{T_{vi}}{T_{li}}\right) + \left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\sum_{l=1}^{3}\left(\frac{\Theta_l}{2} + \frac{\Theta_l}{\exp\left(\Theta_l / T_{vi}\right) - 1}\right) + \frac{\nu_{l\infty}}{kT_{li}}\left(P_v - \sigma K - P_{sat}(T_{li}) - \Pi\right),$$

$$+ \ln\left[\left(\frac{T_{vi}}{T_{li}}\right)^4 \frac{P_{sat}(T_{li})}{P_v}\right] + \ln\left(\frac{q_{vib}(T_{vi})}{q_{vib}(T_{li})}\right)$$

$$q_{vib}(T) = \prod_{l=1}^{3} \frac{\exp\left(-\Theta_l / 2T\right)}{1 - \exp\left(-\Theta_l / T\right)} \qquad (2\text{-}14)$$

CHAPTER III
BASE THEORY

Summary of This Chapter

As mentioned in Chapter II, phase change phenomena in the evaporating thin film are important to enhance the evaporator performance. The phenomena in the thin film region are governed by Intermolecular forces. In this Chapter, the basic ideas of intermolecular forces are reviewed. In the next step, the evaporation model including the effects of intermolecular forces is derived. Finally the liquid flow model in a micro pore is developed.

Intermolecular Forces

Origin of Intermolecular Forces

Intermolecular and surface forces are the result of the electronic structure of atoms and molecules and all intermolecular forces are electrostatic in origin. If the spatial distribution of the electron clouds is determined, the intermolecular forces may be calculated using classical electrostatics (Hellman-Feynman theorem). However, exact solutions of the Schrödinger equation are not easy to obtain. Usually it is useful to classify intermolecular interactions into a number of different categories even though they all have the same fundamental origin. Some intermolecular forces such as covalent or coulomb interactions and polar forces arise from straightforward electrostatic interactions involving inherently charged or dipole forces. Others such as

charge-fluctuation forces, electrodynamics forces and induced-dipole forces are is different. Normally, the intermolecular interactions can be classified as follows: (1) electrodynamic van der Waals forces, (2) polar forces such as hydrogen bonds, and (3) purely electrostatic forces. The van der Waals force, so called the London dispersion force, is always present and controls transport processes in very thin films. These dispersion forces are long-range forces, vary inversely with the film thickness raised to a power, and can be effective over a large distance(approximately 100nm). Their features may be summarized as follows: (1) they are long-range forces and can be effective from 0.2nm (Interatomic Spacing) to 100nm, (2) these forces may be repulsive or attractive and in general do not follow a simple power law, (3) Dispersion forces not only bring molecules together but also tend to align them, (4) the dispersion interaction of two bodies is affected by the presence of other bodies nearby.

Dispersion forces can include quantum mechanics effect and are amenable to a wide range of theoretical treatments of varying complexity.

Hamaker Constant

To obtain a relation of forces between solids and evaporating liquid molecules, the Hamaker constant plays an important role.

Between atoms, a potential of van der Waals forces is given as

$$w(r) = -C / r^6 . \qquad (3\text{-}1)$$

If we integrate the potentials of all atoms for two flat surfaces, the potential will normally be given as

$$W = -A/12\pi D^2 \tag{3-2}$$

The constant, A, in this equation is called the Hamaker constant. The Hamaker constant is originally defined between two atoms as

$$A = \pi C \rho_1 \rho_2 \tag{3-3}$$

where ρ_1 and ρ_2 are the number density of atoms in the two materials and C is the coefficient in the atom-atom pair potential which can be found in the potential between atoms. The Hamaker constant is on the order of 10^{-19} J between typical solid and liquids across vacuum. For instance, the Hamaker constant between water molecules is around 1.5×10^{-19} J for given $\rho = 3.3\times10^{28}[1/m^3]$ and $C = 140\times10^{-79}[Jm^6]$. However, the equation does not include an influence from neighboring atoms. Similar to a well-known multi-body kinetic, the Hamaker constant for multi-atoms is more complicated to determine. This difficulty could be solved using the Lifshitz theory in which the forces between atoms are considered as continuous media and the interaction between the bodies is treated as the effect of the fluctuational electromagnetic field, which is derived in terms of bulk properties with their dielectric constants and refractive indices.

The theoretical formula for the attractive force per unit area between medium 1 and 2 in a vacuum per unit area was derived by Lifshitz[42] as:

$$\Pi(\delta) = \frac{kT}{\pi c^3}\sum_{N=0}^{\infty}{}'\int_1^{\infty}\left\{\left[\frac{(s_1+p)(s_2+p)}{(s_1-p)(s_2-p)}\exp\left(\frac{2p\xi_N\delta}{c}\right)-1\right]^{-1}+\left[\frac{(s_1+\varepsilon_1 p)(s_2+\varepsilon_2 p)}{(s_1-\varepsilon_1 p)(s_2-\varepsilon_2 p)}\exp\left(\frac{2p\xi_N H}{c}\right)-1\right]^{-1}\right\}dp \tag{3-4}$$

where k is the Boltzmann constant, T is the absolute temperature, p is the

integration variable, $\xi_N = 4\pi^2 kTN/h$, where N is a positive integer, δ is the gap between the two medium, and $\varepsilon = \varepsilon(i\xi)$ is a dielectric constant. Subscripts show frequency regions.

s is defined by

$$s = \left(\varepsilon - 1 + p^2\right)^{1/2} \tag{3-5}$$

In this equation the dielectric constant is an unknown parameter. The dielectric constant for water is give as

$$\varepsilon(i\xi) = 1 + \sum_j \frac{d_j}{1 + \xi\tau_j} + \sum_j \frac{f_j}{\omega_j^2 + \xi^2 + g_j\xi} \tag{3-6}$$

where $d_j, \tau_j, f_j, \omega_j, g_j$ are oscillator strengths, resonant frequencies and bandwidth terms. Substituting Eqs.(3-5) and (3-6) into Eq.(3-4), the intermolecular forces can be calculated. In TABLEs(3-1) through (3-3) the spectral parameter for three frequency regions are provided.

TABLE 3-1 Spectral Parameters for Water in Microwave Region.

D(eV)	1/τ(eV)
74.8	6.5×10^{-5}

TABLE 3-2 Spectral Parameters for Water in Infrared Frequencies.

ω_j (eV)	f_j (eV2)	g_j (eV)
2.07×10^{-2}	6.25×10^{-4}	1.5×10^{-2}
6.9×10^{-2}	3.5×10^{-3}	3.8×10^{-2}
9.2×10^{-2}	1.28×10^{-3}	2.8×10^{-2}
2.0×10^{-1}	5.69×10^{-3}	2.5×10^{-2}
4.2×10^{-1}	1.35×10^{-2}	5.6×10^{-2}

TABLE 3-3 Spectral Parameters for Water in Ultraviolet Frequencies.

ω_j (eV)	f_j (eV2)	g_j (eV)
8.25	2.68	0.51
10.0	5.67	0.88
11.4	12.0	1.54
13.0	26.3	2.05
14.9	33.8	2.96
18.5	92.8	6.26

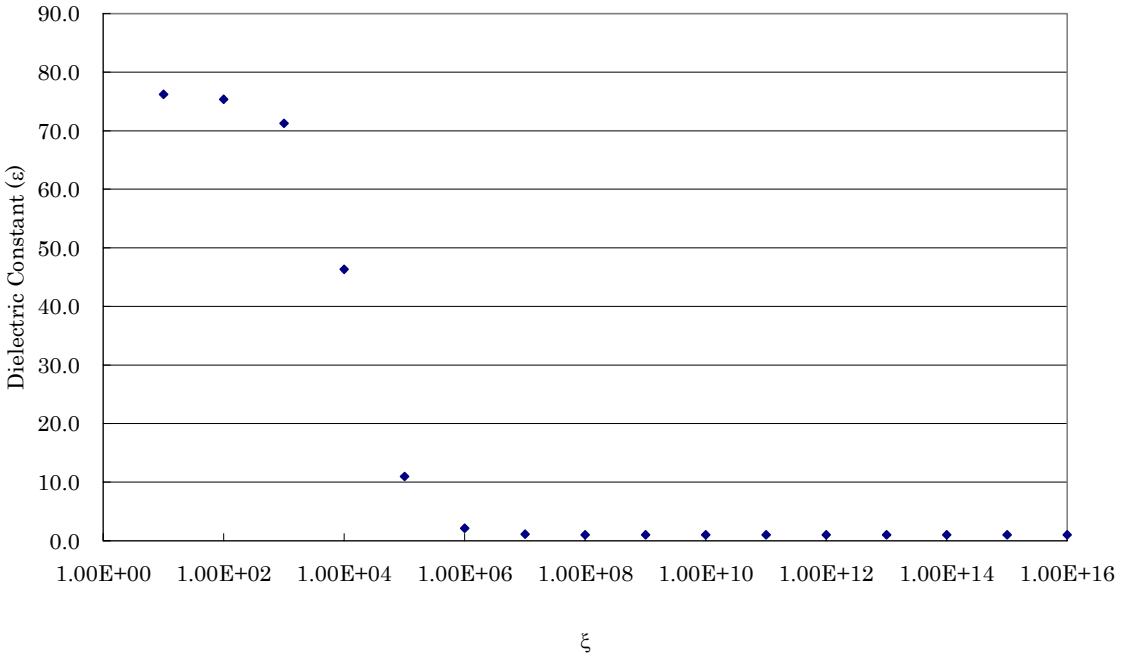

Fig. 3-1. Dielectric Constant for Water.

Substituting parameters from TABLE 3-1 into the second term of Eq. (3-6) and on TABLEs (3-2) and (3-3) into the third terms, the electric constants can be obtained (Fig. 3-1).

Applying the results in Fig. 3-1 into Eq. (3-4), the intermolecular forces between solid silicon and water molecules were obtained. For a non-polar liquid whose intermolecular forces are only dependent on the dispersion forces, the intermolecular force can be expressed with proper approximations as

$$F = \frac{B}{H^4} \tag{3-7}$$

where B is constant and H is the gap between surfaces. Although the working fluid for a heat pipe such as the water discussed in this paper is usually not a non-polar liquid and Eq.(3-6) is not valid for them, it is convenient to define B for a narrow region, i, as

$$B_i = \frac{F_i}{H_i} \tag{3-8}$$

Fig. 3-2 shows Hamaker constant, B_i obtained by Eq. (3-8) for entire region.

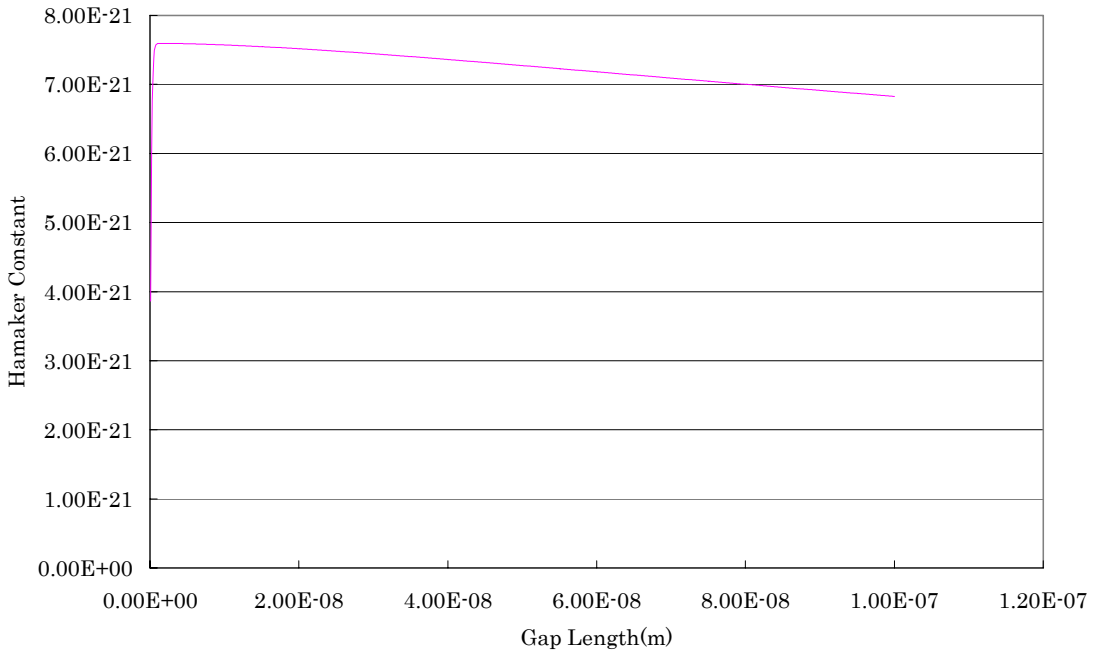

Fig. 3-2. Hamaker Constant between Silicon and Water Molecules.

Applying Intermolecular Forces to Evaporation Model

In Chapter II, a new evaporation model based on statistical rate theory developed by Ward and Fang7 was introduced. Since the original equation does not include the effect of intermolecular force and is derived for convex liquid-vapor interface, the modified evaporation model in this chapter will be derived including the van der waals intermolecular force for the concave interface which is formed in an evaporative meniscus. The derivation is an extension of the work done by Ward and Fang[40].

Consider a small, isolated volume that contains both vapor and liquid phase. Each phases' intensive properties are uniform. Assume that thermodynamic equilibrium does not exist between phases, but each phase may be assumed to be under local equilibrium, thus, each system is considered a canonical ensemble system. If the internal energy of phase α were denoted as U^{α}, the temperature as T^{α}, the specific heat for constant volume as C_v^{α}, the energy of the possible quantum states of each phase would be within the energy range

$$U^{\alpha} \pm T^{\alpha}\sqrt{kC_v^{\alpha}}\,, \qquad (3\text{-}9)$$

where $\alpha = l$ or v.

If the number of molecules of each phase at t is N^L and N^V and the distribution is denoted as λ_j, then the number of quantum states available to the system is denoted as $\Omega(\lambda_j)$. If one molecular evaporates in the time interval δt, the molecular distribution would change to λ_k. From the transition probability concept of quantum mechanics and

the entropy definition in statistical thermodynamics, the probability of this event occurring at any instant in the time interval δt can be expressed as:

$$\tau(\lambda_j, \lambda_k) = \frac{|V_{v\varepsilon}|\zeta}{h} \exp\left(\frac{S(\lambda_j) - S(\lambda_k)}{k}\right) \tag{3-10}$$

where $|V_{v\varepsilon}|$ is the matrix element for a change from a quantum state of λ_j to λ_k and the energy density of the states within the energy uncertainty as ζ. h is Plank's constant and k is Boltzmanns constant. For an isolated system the change in entropy may be expressed in terms of the chemical potential and the enthalpy as:

$$S(\lambda_k) - S(\lambda_j) = \left(\frac{\mu_{li}}{T_{li}} - \frac{\mu_{vi}}{T_{vi}}\right) + h_v\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right) \tag{3-11}$$

The number of molecules that evaporate during δt would be:

$$\delta N = \tau(\lambda_{j,}\lambda_k)\delta t \tag{3-12}$$

Hence the unidirectional evaporation flux will be expressed as:

$$J_{LV} = \frac{|V_{v\varepsilon}|\zeta_{LV}}{h} \exp\left[\frac{1}{k}\left(\frac{\mu_{li}}{T_{li}} - \frac{\mu_{vi}}{T_{vi}}\right) + \frac{h^V}{k}\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\right] \tag{3-13}$$

Similarly, the condensation flux from vapor to liquid can be expressed as:

$$J_{VL} = \frac{|V_{v\varepsilon}|\zeta_{VL}}{h} \exp\left[-\frac{1}{k}\left(\frac{\mu_{li}}{T_{li}} - \frac{\mu_{vi}}{T_{vi}}\right) - \frac{h^V}{k}\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\right] \tag{3-14}$$

Since it has been assumed that $|V_{v\varepsilon}|\zeta$ is constant for transitions between states that are within the energy uncertainty. When the system is at equilibrium, temperature, chemical potential in both phases and number of molecules exchanging between phases are the

same, thus:

$$\mu_l = \mu_v,\ T_{li} = T_{vi} \text{ and } J_{LV} = J_{VL} \tag{3-15}$$

Therefore, we have the relation:

$$J_{LV} = J_{VL} \Leftrightarrow \frac{|V_{v\varepsilon}|\zeta_{LV}}{h} = \frac{|V_{v\varepsilon}|\zeta_{VL}}{h} \equiv K_e \tag{3-16}$$

Under equilibrium conditions, the molecular mass flux is given as:

$$K_e = \frac{P_{ve}}{\sqrt{2\pi mkT_l}} \tag{3-17}$$

Finally the net mass flux will be:

$$j = \frac{P_{ve}}{\sqrt{2\pi mkT_{li}}}\left(\exp\left(\frac{\Delta S}{k}\right) - \exp\left(-\frac{\Delta S}{k}\right)\right), \tag{3-18}$$

where

$$\Delta S = \left(\frac{\mu_{li}}{T_{li}} - \frac{\mu_{vi}}{T_{vi}}\right) + h^V\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right) \tag{3-19}$$

Expression for Chemical Potential and Enthalpy by Statistical Thermodynamics

The modified Young-Laplace equation relates the vapor and liquid pressure in the vicinity of a pore interface as:

$$P_v = P_{li} + 2\sigma K + \Pi \tag{3-20}$$

The Gibbs-Duhem equation yields a relation between chemical potential and pressure as

$$d\mu = -sdT + vdP \tag{3-21}$$

If the temperature does not change, thus, $dT = 0$, integration of this equation at the constant temperature from $P = P_{sat}(T_l)$ to $P = P_{li}$ yields

$$\mu_{li} - \mu_{sat} = \int_{P_{sat}}^{P_{li}} v dP \tag{3-22}$$

For liquid, change of specific volume with pressure can be considered very small.

$$\mu_{li} - \mu_{sat} = v_{\infty} \int_{P_{sat}}^{P_{li}} dP \tag{3-23}$$

v_{∞} is a specific volume under an equilibrium condition.

Substituting Eq. (3-20) into Eq. (3-23) for the liquid chemical potential gives:

$$\frac{\mu_{li}}{T_{li}} = \frac{\mu_{sat}(T_{li})}{T_{li}} + \frac{v_{\infty}}{T_{li}}\left(P_v - 2\sigma K + \Pi\right) \tag{3-24}$$

If we assume that the vapor is an ideal gas, the vapor chemical potential can be expressed from statistical thermodynamics as[43]:

$$\frac{\mu_{vi}}{T_{vi}} = -k \ln\left[\left(\frac{2\pi m}{h^2}\right)^{3/2} \frac{\left(kT_{vi}\right)^{5/2}}{P_v}\right] - k \ln\left(q_{vib} q_{rot} q_{elec}\right) \tag{3-25}$$

where q_{vib}, q_{rot} q_{elec} is the vibrational function, the rotational partition functions and the electronic partition function respectively.

The vibrational and the rotational partition functions for ideal polyatomic molecules are expressed as:

$$q_{vib} = \prod_{l=1}^{n'} \frac{\exp(-\Theta_l / 2T_{vi})}{1 - \exp(-\Theta_l / T_{vi})} \tag{3-26}$$

and

$$q_{rot} = \left(\frac{4\pi k T_v}{h^2}\right)^{1/2} \frac{(\pi I)^{1/2}}{\sigma_s}, \tag{3-27}$$

where Θ_l is a characteristic temperature for vibration, n' is the number of vibrational degrees of freedom, I is the product of principal moments of inertia of the molecule, and σ_s is the symmetry factor of the vibrational orientation. The electronic partion

function is given by

$$q_{elec} = g_e \exp\left(\frac{D_e}{kT_v}\right), \tag{3-28}$$

where g_e and D_e are the degeneracy of the state and the reference potential minimum, respectively.

TABLE 3-4 Statistical Thermophyical Constants for Water[42].

Constant	Value
Θ_1	2290K
Θ_2	5160K
Θ_3	5360K
σ_s	2
D_e	$1.613\times10^{-18}\, J/molecule$
g_e	1

Substituting Eqs. (3-26), (3-27), and (3-28) into Eq. (3-25) for ΔS gives:

$$\Delta S = \frac{\mu_{li}}{T_{li}} - \frac{\mu_{vi}}{T_{vi}} = \frac{v_\infty}{T_{li}}\left(P_v - 2\sigma K + \Pi - P_{ve}\right) - D_e\left(\frac{1}{T_{li}} - \frac{1}{T_{vi}}\right) + k\ln\left[\left(\frac{T_{vi}}{T_{li}}\right)^4 \frac{P_{ve}}{P_v}\right] + k\ln\left(\frac{q_{vib}(T_{vi})}{q_{vib}(T_{li})}\right) \tag{3-29}$$

All of constants used in Eqs (3-26), (3-27), and (3-28) are given in Table 3-4. The enthalpy can be obtained from statistical thermodynamics as:

$$h^v = 4kT_{vi} - D_e + k\sum_{l=1}^{3}\frac{\Theta_l}{2} + k\sum_{l=1}^{3}\frac{\Theta_l}{\exp(\Theta_l / T_{vi}) - 1} \tag{3-30}$$

Therefore,

$$\frac{\Delta S}{k} = \frac{1}{k}\exp\left[\left(\frac{\mu_l}{T_{li}} - \frac{\mu_v}{T_{vi}}\right) + h_v\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\right]$$
$$= 4\left(1 - \frac{T_{vi}}{T_{li}}\right) + \left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\sum_{l=1}^{3}\left(\frac{\Theta_l}{2} + \frac{\Theta_l}{\exp(\Theta_l / T_{vi}) - 1}\right) + \frac{v_{l\infty}}{kT_{li}}\left(P_v - 2\sigma K - P_{sat}(T_{li}) + \Pi\right).$$
$$+ \ln\left[\left(\frac{T_{vi}}{T_{li}}\right)^4 \frac{P_{sat}(T_{li})}{P_v}\right] + \ln\left(\frac{q_{vib}(T_{vi})}{q_{vib}(T_{li})}\right)$$

(3-31)

Equilibrium Pressure

The saturation pressure is one of the important parameters determining the evaporation rate. However, the saturation pressure for a curved interface is different from one for a flat interface which could be normally found in steam tables because the chemical potential of a curved liquid interface is not equal to the chemical potential of the flat liquid interface. The liquid pressure in the liquid phase can be expressed by the modified Young-Laplace equation.

$$P_v = P_l + 2\sigma K - \Pi \qquad (3\text{-}32)$$

When the liquid and vapor are at equilibrium, both chemical potential and temperature should be the same in the two phases, thus,

$$\mu_l = \mu_v \qquad (3\text{-}33)$$

To obtain the chemical potential for the curved interface we use the Gibbs-Duhem equation,

$$d\mu = -sdT + vdP \qquad (3\text{-}34)$$

Integrating Eq. (3-34) at constant temperature from $P = P_{sat}(Tl)$ to $P = P_e$ yields

$$\mu_{ve} - \mu_{sat} = \int_{P_{sat}}^{P_e} v dP \tag{3-35}$$

For an ideal gas, the specific volume can be obtained as

$$v = \frac{RT_l}{P} \tag{3-36}$$

Substituting Eq. (3-36) into Eq. (3-35) and integrations give

$$\mu_{ve} = \mu_{sat,v} + RT_l \ln\left[\frac{P_{ve}}{P_{sat}(T_l)}\right] \tag{3-37}$$

For liquid phase, if we assume that the liquid is incompressible, the specific volume is equal to the specific volume at the saturation condition.

$$\mu_l = \mu_{sat,l} + v_l\left[P_l - P_{sat}(T_l)\right] \tag{3-38}$$

Since $\mu_{ve} = \mu_l$ and $\mu_{sat,v} = \mu_{sat,l}$, the equilibrium vapor pressure can be given by

$$P_{ve} = P_{sat}(T_l)\exp\left\{\frac{v_l\left[P_l - P_{sat}(T_l)\right]}{RT_l}\right\} \tag{3-39}$$

Substituting the modified Young-Laplace equation, we obtain

$$P_{ve} = P_{sat}(T_l)\exp\left\{\frac{v_l\left[P_{ve} - P_{sat}(T_l) - \sigma K + \Pi\right]}{RT_l}\right\} \tag{3-40}$$

With a numerical method such as the Newton method, the equilibrium pressure for a curved interface can be obtained.

Validity of Navier-Stokes Equations for Liquid Flow in Microscale

We have to discuss the validity of the Navier-Stokes equations and the no-slip-boundary on the wall since the liquid flow discussed in this chapter is smaller

than in the order of micro scale. For vapor flow, the Knudsen number explains well the phenomena in the micro scale, while it does not work for liquids. At this moment, there is no well-explained molecular based theory for liquids as dilute gases since the vapor is less dense and the intermolecular forces which play a main role of small scale liquid phenomena ,but negligible for vapor, makes calculations much more complicated.

The points are that whether liquid flows in a micron film can be considered as a continuum, the bulk viscosity is valid in a micro scale, and Fick's law for thermal energy diffusion can be applied. The average distance between molecules in the gas is one order magnitude higher than the diameter of its molecules. On the other hand, its distance for liquid molecules is in the order of the molecular diameter. For instance, the water molecules spacing is about 0.3nm.

There are some experiments to measure viscosity in micro scale. Pfahler et al.[44] summarize the relevant literature and also measured the friction coefficient in a silicon etched micro channel. They conclude that the results would be in agreement with experimental observations, but the viscosity in a micro scale is consistently smaller than the bulk viscosity. Israelachvili[45] found that the bulk viscosity is appropriate for the thin film flow if the film thickness is larger than 10 molecular layers.

Thompson and Troian[46] suggested a universal boundary condition and conducted several experiments to measure a slip length as a function of shear rate in a Couette

flow. According to these results, the limiting value of slip length is approximately 5.0×10^{-9} [m] for water and it is smaller mostly than the flow scale in a pore.

The linearity of the temperature gradient could be broken by the effects of internal heating due to viscous dissipation. But, since the liquid has a large heat capacity and the liquid is dense even in a micro scale, these effects can be considered less significant.

In conclusion, the Navier-Stokes equation can be used to describe liquid flows under circumstance applicable to geometries on the order of microns in size.

Pore Evaporation Model Development

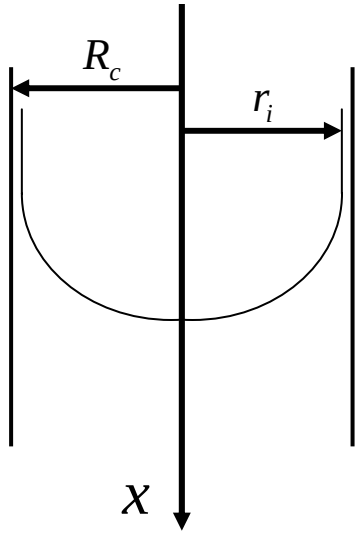

Fig. 3-3. Evaporating Meniscus in a Capillary Tube

The mathematical model developed in this chapter incorporates the fluid mechanics, heat transfer and interfacial phenomena controlling evaporation from the meniscus of a capillary tube. Fig. 3-3 shows the geometry of the evaporating meniscus. The geometry of the pore is cylindrical and all variables are axially symmetric. The momentum equation for axial direction is given as

$$\left(\vec{V}\cdot\vec{\nabla}\right)u = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\vec{\nabla}^2 u \tag{3-41}$$

If it is assumed that all of velocity components except axial direction are negligible, Eq.(3-41) can be expressed as

$$u\frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial P_l}{\partial x} + \nu_l \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) \tag{3-42}$$

Comparing the friction term and inertia term,

$$\rho u\frac{\partial u}{\partial x} << \mu_l \frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) \tag{3-43}$$

This is approximately

$$\rho u\frac{u}{L} << \mu_l\left(\frac{u}{r^2}\right) \tag{3-44}$$

The mass flow rate is on the order of $10^{-12} - 10^{-11}$ [kg/sec] in a pore, the velocity is

$$u = \frac{M_0}{\rho A_{pore}} \approx 10^{-5} - 10^{-4}\ [\text{m/sec}] \tag{3-45}$$

The orders of other variables are

$$\rho \approx 10^3\ [\text{kg/m}^3],\ \ \mu_l \approx 10^{-4}\ [\text{N}\cdot\text{s/m}^2],\ \ L \approx 10^{-6}\ [\text{m}],\ \ r \approx 10^{-6}\ [\text{m}] \tag{3-46}$$

The first term and second term in Eq. (3-44) give

$$\rho u\frac{u}{L} \approx 10^1, \mu_l\left(\frac{u}{r^2}\right) \approx 10^4 \tag{3-47}$$

Therfore,

$$\rho u \frac{u}{L} << \mu_l \left(\frac{u}{r^2} \right) \tag{3-48}$$

The momentum equation of the thin liquid film is given by applying lubrication theory as:

$$\frac{\mu_l}{r} \left\{ \left(\frac{\partial}{\partial r} u(r) \right) + r \left(\frac{\partial^2}{\partial r^2} u(r) \right) \right\} = \frac{\partial}{\partial x} P_l \tag{3-49}$$

The boundary condition at the wall is given as

$$u_l = 0 \text{ at } r = R_c, \tag{3-50}$$

where u_l is fluid velocity. The stress balance under evaporation at the interface is given by Burelbach et al.[47] for the components of the normal and tangential stress as

$$-\frac{j^2}{\rho_v} - \bar{\bar{T}} \cdot \bar{n} \cdot \bar{n} = 2K\sigma + \Pi \tag{3-51}$$

$$\bar{\bar{T}} \cdot \bar{n} \cdot \bar{t} = \nabla \sigma \cdot \bar{t} \tag{3-52}$$

$\bar{n}$ and $\bar{t}$ are the normal vector and the tangential vector on the interface respectively and are assumed as

$$\bar{n} = \begin{pmatrix} n_r \\ n_\theta \\ n_z \end{pmatrix} \tag{3-53}$$

$$\bar{t} = \begin{pmatrix} t_r \\ t_\theta \\ t_z \end{pmatrix} \tag{3-54}$$

$\bar{\bar{T}}$ is the stress tensor and simplified by

$$\bar{\bar{T}} = \begin{pmatrix} -p & 0 & \mu\dfrac{\partial u_l}{\partial r} \\ 0 & -p & 0 \\ \mu\dfrac{\partial u_l}{\partial r} & 0 & -p \end{pmatrix} \tag{3-55}$$

In Eq. (3-51), the second term in the left hand side is given as

$$\bar{\bar{T}} \cdot \bar{n} \cdot \bar{n} = -pn_r^2 + 2\mu\frac{\partial u_l}{\partial r} n_r n_z - pn_z^2 = -p + 2\mu\frac{\partial u_l}{\partial r} n_r n_z \tag{3-56}$$

Therefore, substituting Eq. (3-56) into Eq. (3-51) will be

$$-\frac{j^2}{\rho_v} + p = 2K\sigma + \Pi \tag{3-57}$$

In Eq. (3-52), the left hand side will be

$$\begin{aligned} \bar{\bar{T}} \cdot \bar{n} \cdot \bar{t} &= -pn_r t_r + \mu\frac{\partial u_l}{\partial r} n_z t_r + \mu\frac{\partial u_l}{\partial r} n_r t_z - pn_z t_r \\ &= -p(n_r t_r + n_z t_r) + \mu\frac{\partial u_l}{\partial r}(n_z t_r + n_r t_z) \end{aligned} \quad . \tag{3-58}$$

The inner product of the normal vector and the tangential vector is zero, giving

$$\bar{n} \cdot \bar{t} = n_r t_r + n_z t_z = 0 \tag{3-59}$$

Eq. (3-58) becomes

$$\bar{\bar{T}} \cdot \bar{n} \cdot \bar{t} = \mu\frac{\partial u_l}{\partial r}(n_z t_r + n_r t_z) \tag{3-60}$$

The right hand side term are expressed by

$$\nabla\sigma \cdot \bar{t} = \frac{\partial\sigma}{\partial z} t_z + \frac{\partial\sigma}{\partial r} t_r \tag{3-61}$$

The derivatives of surface tension, σ, can be expressed as

$$\frac{\partial\sigma}{\partial z} = \frac{\partial\sigma}{\partial T}\frac{\partial T}{\partial z} \tag{3-62}$$

$$\frac{\partial \sigma}{\partial r} = \frac{\partial \sigma}{\partial T}\frac{\partial T}{\partial r} \tag{3-63}$$

The surface tension for water can be expressed as a function of temperature as

$\sigma = 7.583\times10^{-2} + 1.477\times10^{-2} T$ [N/m],

so

$$\frac{\partial \sigma}{\partial T} = 1.477\times10^{-2} \text{ [N/m/T]} \tag{3-64}$$

To simplify the problem, the temperature gradients in Eqs. (3-62) and (3-63) are assumed to be negligible.

Thus, Eq. (3-52) becomes

$$-\mu_l \frac{\partial u}{\partial r} = 0 \text{ at } r = r_i \tag{3-65}$$

The first boundary condition(3-50) is the velocity of liquid at the wall and the second boundary condition(3-65) is the shear stress at the interface. The solution for this equation is:

$$u_l(r) = -\frac{1}{4\mu_l}\left(\frac{dP_l}{dx}\right)\left(R_c^2 - r^2 + 2r_i^2 \ln\left(\frac{r}{R_c}\right)\right). \tag{3-66}$$

The total mass flow rate, Γ [kg/sec], in the x-axis-direction for flow between the liquid-vapor interface and the wall at position x is given by integrating the velocity from the interface to the wall as:

$$\Gamma = \rho\int_{r=r_i}^{r=R_c} 2\pi r u_l(r) dr \tag{3-67}$$

Substituting Eq. 3-4 for u_l into Eq. 3-5 and integrating gives:

$$\Gamma = F_1 \frac{dP_l}{dx} \tag{3-68}$$

where

$$F_1 = \frac{\rho\pi(4R_c^2 r_i^2 - R_c^4 + 4r_i^4 \ln(r_i / R_c) - 3r_i^4)}{8\mu}. \quad (3\text{-}69)$$

The liquid pressure can be expressed by the modified Young-Laplace equation as

$$P_l = P_v - \sigma K - \Pi . \quad (3\text{-}70)$$

Substituting Eq. (3-56) into Eq. (3-54), the mass flow rate becomes

$$\Gamma = F_1 \frac{d}{dx}\left[-\sigma K - \Pi\right]. \quad (3\text{-}71)$$

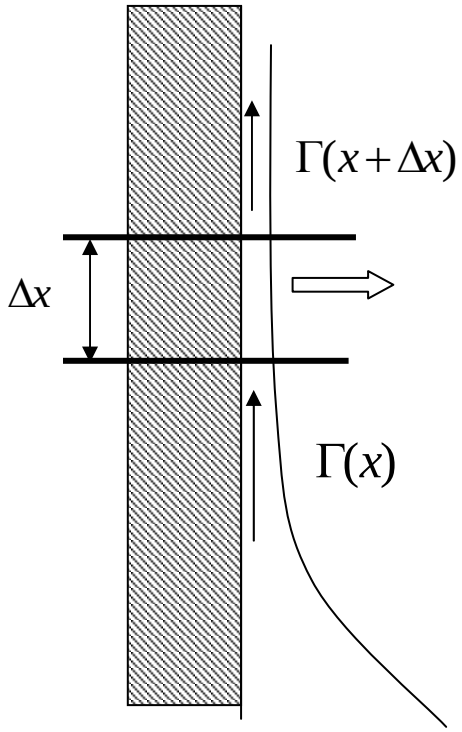

Fig. 3-4. Mass Balance in Liquid Thin Film

To satisfy the mass balance, the amount of the vapor leaving from the liquid interface must equal the evaporation rate at the interface (Fig. 3-4). Thus,

$$\Gamma(x)-\Gamma(x+\Delta x)=\frac{M}{N_A}\frac{\eta P_{sat}(T_{li})}{\sqrt{2mkT_{li}}}\left\{\exp\frac{\Delta S}{k}-\exp\frac{-\Delta S}{k}\right\}A_{evap}. \tag{3-72}$$

The left hand side expresses the net flow balance and the right hand side is the evaporation rate. $\Gamma(x)$ is a mass flow rate at x and A_{evap} is an evaporative surface area between x and x+dx which is givens as

$$A_{evap}=\frac{2\pi R_c\Delta x}{\cos\theta} \tag{3-73}$$

where

$$\theta=\text{atan}\left(-h'_{n+1/2}\right)\approx\frac{h_n-h_{n+1}}{\Delta x}. \tag{3-74}$$

In the right hand side $\Delta S/k$ is given as

$$\begin{aligned}\frac{\Delta S}{k}&=\frac{1}{k}\exp\left[\left(\frac{\mu_l}{T_{li}}-\frac{\mu_v}{T_{vi}}\right)+h_v\left(\frac{1}{T_{vi}}-\frac{1}{T_{li}}\right)\right]\\&=4\left(1-\frac{T_{vi}}{T_{li}}\right)+\left(\frac{1}{T_{vi}}-\frac{1}{T_{li}}\right)\sum_{l=1}^{3}\left(\frac{\Theta_l}{2}+\frac{\Theta_l}{\exp(\Theta_l/T_{vi})-1}\right)+\\&\frac{v_{l\infty}}{kT_{li}}\left(P_v-\sigma K-P_{sat}(T_{li})-\Pi\right)+\ln\left[\left(\frac{T_{vi}}{T_{li}}\right)^4\frac{P_{sat}(T_{li})}{P_v}\right]+\ln\left(\frac{q_{vib}(T_{vi})}{q_{vib}(T_{li})}\right)\end{aligned} \tag{3-75}$$

$$q_{vib}(T)=\prod_{l=1}^{3}\frac{\exp(-\Theta_l/2T)}{1-\exp(-\Theta_l/T)}. \tag{3-76}$$

The disjoining pressure, Π, is expressed as

$$\Pi=\begin{cases}-\dfrac{\overline{A}}{\delta^3}\ \text{for}\ \delta<10\text{nm}\\ -\dfrac{B}{\delta^4}\ \text{for}\ \delta>15\text{nm}\end{cases} \tag{3-77}$$

A is the non-retarded Hamaker constant and B is the retarded Hamaker constant. $v_{l\infty}$ is the specific volume per unit molecule. k is the Boltzmann constant. P_v and

$P_{sat}(T_{li})$ are vapor pressure and saturation pressure at T_{li} respectively. The Hamaker constant is a function of film thickness, δ, and described in the beginning of this Chapter including values.

The definition of mean curvature for three dimensions[48] for $z = h(x, y)$ is

$$K = \frac{(1+h_y^2)h_{zz} - 2h_z h_y h_{zy} + (1+h_z^2)h_{yy}}{2(1+h_z^2+h_y^2)^{3/2}}. \tag{3-78}$$

The thermal energy entering into and exiting from the liquid-vapor interface is balanced with the evaporation rate at the interface.

$$(E(x+\Delta x) - E(x)) + \left(-k_l \left.\frac{\partial T}{\partial r}\right|_{r=R_c} A_{cond} - \frac{M}{N_A}\frac{\eta P_{sat}(T_{li})}{\sqrt{2mkT_{li}}}\left\{\exp\frac{\Delta S}{k} - \exp\frac{-\Delta S}{k}\right\} h_{fg} A_{evap} \right) = 0 \tag{3-79}$$

where A_{cond} is an thermal conducting area through thin liquid film which is given as

$$A_{cond} = 2\pi R_c \Delta x \tag{3-80}$$

Liquid energy conservation can be expressed as

$$u\frac{\partial T}{\partial x} = \frac{\alpha}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) \tag{3-81}$$

with boundary conditions of

$$\begin{aligned} &T = T_i(x) \text{ at } r = r_i \\ &T = T_w \text{ at } r = R_c \end{aligned}. \tag{3-82}$$

The first boundary condition is the interfacial temperature $T_i(x)$, and the second boundary condition applies at the wall temperature which is assumed constant in x. Assuming that thermal energy transfer is only due to conduction, the temperature at r,

$T(r)$, is given as

$$T(r) = \frac{\{T_i \ln(R_c / r) - T_w \ln(r_i / r)\}}{\ln(R_c / r_i)}. \tag{3-83}$$

Substituting Eq. (3-83) into Eq. (3-79) gives

$$(E(x+\Delta x) - E(x)) + \left(-k_l \frac{T_w - T_i}{\ln(R_c / r_i) R_c} A_{cond} - \frac{M}{N_A} \frac{\eta P_{sat}(T_{li})}{\sqrt{2mkT_{li}}} \left\{ \exp\frac{\Delta S}{k} - \exp\frac{-\Delta S}{k} \right\} h_{fg} A_{evap} \right) = 0 \tag{3-84}$$

Numerical Calculation

To conduct numerical calculations, mass balance and thermal energy balance equations need to be descretized. For a control-volume at $x = x_{n+1/2}$, a net mass flow balance is equal to the evaporation rate from the surface area.

$$\Gamma_n - \Gamma_{n+1} = -\left(f(h_{n+1}) \frac{dP_l}{dx}\bigg|_{x=x_{n+1}} - f(h_n) \frac{dP_l}{dx}\bigg|_{x=x_n} \right) = \frac{2M}{N} \frac{P_{sat}(T_n)}{\sqrt{2\pi mkT_n}} \sinh\left(\frac{\Delta S}{k} \right) A_{evap} \tag{3-85}$$

The energy balance can be expressed by the net thermal energy entering into and exiting from the control volume.

$$(E_n - E_{n+1}) + \left(-k_l \frac{T_w - T_n}{\ln(R_c / r_i) R_c} A_{cond} - \frac{M}{N_A} \frac{\eta P_{sat}(T_n)}{\sqrt{2mkT_n}} \left\{ \exp\frac{\Delta S}{k} - \exp\frac{-\Delta S}{k} \right\} h_{fg} A_{evap} \right) = 0 \tag{3-86}$$

where

$$A_n = 2\pi R_c \Delta x\,,\ \ A_{evap} = \frac{2\pi r_n \Delta x}{\cos\theta}\,, \tag{3-87}$$

$$r_{n+1/2} = R_c - h_{n+1/2}\,, \tag{3-88}$$

$$h_{n+1/2} = (h_n + h_{n+1})/2\,, \tag{3-89}$$

$$\theta = a\tan\left(-\frac{dh}{dx}\right) = a\tan\left(\frac{h_{n-1} - h_{n+1}}{2\Delta x}\right) \tag{3-90}$$

In the $\Delta S / k$ term in Eq. (3-72), curvature, K and disjoining pressure need to be discretized for the control-volume.

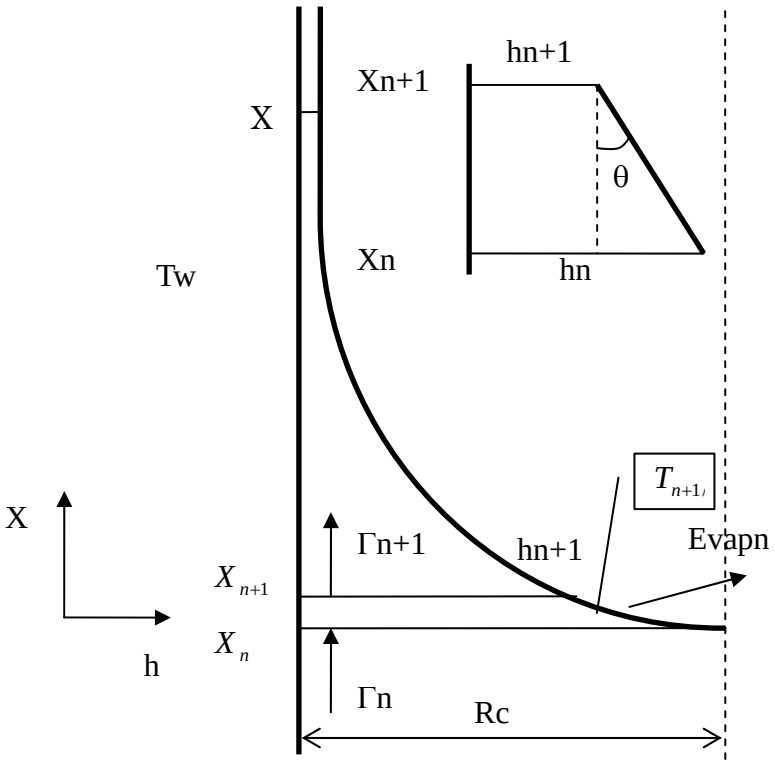

Fig. 3-5: Control Volume to Calculate Mass and Thermal Energy Balances

The definition of mean curvature[47] for three dimensions for $z = h(x, y)$ gives

$$K = \frac{(1+h_y^2)h_{zz} - 2h_z h_y h_{zy} + (1+h_z^2)h_{yy}}{2(1+h_z^2+h_y^2)^{3/2}} \tag{3-91}$$

With the neighboring two points of the control-volume, using the Lagrange interpolating polynomial, a function of these three points can be expressed as

$$x = \frac{(y-y_n)(y-y_{n+1})}{(y_{n-1}-y_n)(y_{n-1}-y_{n+1})}x_{n-1} + \frac{(y-y_{n-1})(y-y_{n+1})}{(y_n-y_{n-1})(y_n-y_{n+1})}x_n + \frac{(y-y_{n-1})(y-y_n)}{(y_{n+1}-y_{n-1})(y_{n+1}-y_n)}x_{n+1} \tag{3-92}$$

It can be transformed with respect to y as

$$\begin{aligned} x = &\left(\frac{x_n}{(y_n-y_{n-1})(y_n-y_{n+1})} + \frac{x_{n-1}}{(y_{n-1}-y_n)(y_{n-1}-y_{n+1})} + \frac{x_{n+1}}{(y_{n+1}-y_{n-1})(y_{n+1}-y_n)}\right)y^2 \\ &+\left(\frac{-x_{n-1}y_n - x_{n-1}y_{n+1}}{(y_{n-1}-y_n)(y_{n-1}-y_{n+1})} + \frac{-x_n y_{n-1} - x_n y_{n+1}}{(y_n-y_{n-1})(y_n-y_{n+1})} + \frac{-x_{n+1}y_{n-1} - x_{n+1}y_n}{(y_{n+1}-y_{n-1})(y_{n+1}-y_n)}\right)y \\ &+\left(\frac{x_{n-1}y_n y_{n+1}}{(y_{n-1}-y_n)(y_{n-1}-y_{n+1})} + \frac{x_{n+1}y_{n-1}y_n}{(y_{n+1}-y_{n-1})(y_{n+1}-y_n)} + \frac{x_n y_{n-1}y_{n+1}}{(y_n-y_{n-1})(y_n-y_{n+1})}\right) \end{aligned} \tag{3-93}$$

Now we set an equation like

$$x = a_n d^2 + b_n d + c_n \tag{3-94}$$

d is a value of y when z=0 and can be given as

$$d = \frac{-b_n + \sqrt{b_n^2 - 4a_n(c_n - x)}}{2a_n} \tag{3-95}$$

The surface equation can be expressed as

$$y^2 + z^2 = d^2$$
$$= \left(\frac{-b_n + \sqrt{b_n^2 - 4a_n(c_n - x)}}{2a_n} \right)^2 \tag{3-96}$$

Therefore,

$$x = \frac{2a_n c_n + 2y^2 a_n^2 + 2z^2 a_n^2 + 2b_n \sqrt{y^2 a_n^2 + z^2 a_n^2}}{2a_n} \tag{3-97}$$

Derivatives of function h are given

$$h_z = \frac{4za_n^2 + 2b_n z a_n^2 / \sqrt{y^2 a_n^2 + z^2 a_n^2}}{2a}$$
$$h_{zz} = \frac{4a_n^2 - 2b_n z^2 a_n^4 / (y^2 a_n^2 + z^2 a_n^2)^{3/2} + 2b_n a_n^2 / \sqrt{y^2 a_n^2 + z^2 a_n^2}}{2a}$$
$$h_y = \frac{4ya_n^2 + 2bya_n^2 / \sqrt{y^2 a_n^2 + z^2 a_n^2}}{2a} \tag{3-98}$$
$$h_{yy} = \frac{4a_n^2 - 2b_n y^2 a_n^4 \Big/ (y^2 a_n^2)^{(3/2)} + 2b_n a_n^2 / \sqrt{y^2 a_n^2}}{2a}$$
$$h_{zy} = -\frac{b_n z a_n^3 y}{(y^2 a_n^2 + z^2 a_n^2)^{3/2}}$$

Since h is not dependent on the z value due to symmetry of interface along x axis, z is set to zero. Derivatives will be expressed as

$$h_z = 0$$
$$h_{zz} = 2a_n + b_n / y$$
$$h_y = 2ya_n + b_n \tag{3-99}$$
$$h_{yy} = 2a_n - b_n / y + a_n b_n / y$$
$$h_{zy} = 0$$

The mean curvature at $y = y_n$ will be given as

$$K = \frac{(1+h_y^2)h_{zz} - 2h_z h_y h_{zy} + (1+h_z^2)h_{yy}}{2(1+h_z^2+h_y^2)^{3/2}}$$
$$= \frac{(1+h_y^2)h_{zz} + h_{yy}}{2(1+h_y^2)^{3/2}} = \frac{2a_n + (1+(2a_n y + b)^2)(2a_n + b_n / y)}{2(1+(2a_n y + b_n)^2)^{3/2}} \quad (3\text{-}100)$$

A net thermal energy balance of the control-volume at $x = x_{n+1/2}$ will be expressed as

$$E_n - E_{n+1} + \left(-k \frac{dT}{dr} A_n \right) - evap_n A_{evap} h_{fg} = 0 . \quad (3\text{-}101)$$

The first and second terms are the thermal energy entering and exiting due to mass flow respectively. The third term is the thermal energy due to the heat conduction entering from the wall into liquid film. The fourth term is leaving energy due to the evaporation.

E_n is the thermal energy due to the mass flow and can be given as

$$E_n = \rho C_p \int_{r=r_i}^{r=Rc} 2\pi u_l T_l(r) r dr$$
$$= \frac{\rho \pi Cp}{32\mu \ln(Rc/r_i)} \left(\frac{dP_l}{dx} \right) \{ 4R_c^4 \ln(r_i) T_w + 12R_c^2 r_i^2 T_n + 8R_c^2 r_i^2 \ln(R_c) T_w - 12R_c^2 r_i^2 T_w$$
$$+ 3R_c^4 T_w - 4R_c^4 T_w \ln(R_c) - 8R_c^2 r_i^2 \ln(r_i) T_w - 3R_c^4 T_n + 8r_i^2 R_c^2 T_n \ln(R_c)$$
$$- 16r_i^4 (\ln(R_c))^2 T_n - 8r_i^4 \ln(r_i) T_w + 8r_i^4 \ln(R_c) T_w - 9r_i^4 T_n - 20r_i^4 T_n \ln(R_c)$$
$$- 8r_i^2 \ln(r_i) R_c^2 T_n + 32r_i^4 T_n \ln(R_c) \ln(r_i) + 9r_i^4 T_w - 16r_i^4 T_n (\ln(r_i))^2 + 20r_i^4 T_n \ln(r_i) \} \quad (3\text{-}102)$$

The numerical calculations are conducted by setting the curvature at the bottom of meniscus:

1: Set the initial condition.

K_0 and $h_0 = R_c$

2: Assume the total mass flow rate in a pore.

3: Determine h_n from Eq. (3-100)

$$K_{n-1} = \frac{2a_{n-1} + (1 + (2a_{n-1}y + b_{n-1})^2)(2a_{n-1} + b_{n-1}/y)}{2(1 + (2a_{n-1}y + b_{n-1})^2)^{3/2}} \tag{3-100}$$

Since $K_{n-1}, x_{n-2}, x_{n-1}, x_n, h_{n-2}, h_{n-1}$ are known in Eq. (3-100), h_n can be determined from this equation.

4: Determine K_n and T_n from Eqs. (3-85) and (3-86).

Recalling Eqs. (3-85) and (3-86)

$$\frac{2M}{N}\frac{P_{sat}(T_n)}{\sqrt{2\pi mkT_n}}\sinh\left(\frac{\Delta S}{k}\right)A_{in} = -(M_{n+1} - M_n)$$

$$\frac{\Delta S}{k} = \frac{1}{k}\exp\left[\left(\frac{\mu_l}{T_{li}} - \frac{\mu_v}{T_{vi}}\right) + h_v\left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\right]$$
$$= 4\left(1 - \frac{T_{vi}}{T_{li}}\right) + \left(\frac{1}{T_{vi}} - \frac{1}{T_{li}}\right)\sum_{l=1}^{3}\left(\frac{\Theta_l}{2} + \frac{\Theta_l}{\exp(\Theta_l/T_{vi}) - 1}\right) + \frac{\nu_{l\infty}}{kT_{li}}\left(P_v - 2\sigma K_n - P_{sat}(T_{li}) + \Pi_n\right)$$
$$+ \ln\left[\left(\frac{T_{vi}}{T_{li}}\right)^4 \frac{P_{sat}(T_{li})}{P_v}\right] + \ln\left(\frac{q_{vib}(T_{vi})}{q_{vib}(T_{li})}\right) \tag{3-85}$$

$$(E_n - E_{n+1}) +$$
$$\left(-k_l \frac{T_w - T_n}{\ln(R_c/r_i)R_c}A_{cond} - \frac{M}{N_A}\frac{\eta P_{sat}(T_n)}{\sqrt{2mkT_n}}\left\{\exp\frac{\Delta S}{k} - \exp\frac{-\Delta S}{k}\right\}h_{fg}A_{evap}\right) = 0 \tag{3-86}$$

In these equations, only K_n and T_n are unknown and can be determined from the above equations.

5: Repeat from 2 to 4 until the curvature reaches the minimum curvature of the pore wall.

6: If the total mass flow rate does not match the total evaporation rate, modify the total evaporation rate and repeat from 2.

CHAPTER IV

RESULT AND APPLICATION

Summary of This Chapter

Based on the method developed for the liquid flow in a pore, evaporation rates, meniscus profiles and the total mass flow rate are calculated for applicable combinations of wall temperature, initial curvature at the bottom of meniscus, and the vapor pressure. These results are applied to optimize the loop heat pipe design.

Evaporation Rate and Meniscus Profile in Pore

The evaporation rate in a pore is determined by the conditions of its environment such as the pore wall temperature, the liquid-vapor interface temperature, and the vapor pressure above the interface. However, all of these values are not know in advance since all of these values are dependent on geometric shape and operational conditions in a closed system such as a heat pipe. Therefore, we assume the wall temperature, the initial curvature at the bottom of meniscus and the vapor pressure and make calculations for a range of these parameters. The results of these calculations are shown in Fig. 4-1 through Fig. 4-24. The wall temperature range and the curvature at the bottom of meniscus are set to from 363.15 to 423.15K and from $2/R_c - 40.0\times10^3$ to $2/R_c$. These are tabulated in Table 4-1.

TABLE 4-1 Figures for Local Evaporation Rate, Meniscus Profile and Temperature Profile in a Pore for a Given Pore Wall Temperature

Pore Wall Temperature(K)	Evaporation Profile	Meniscus Profile	Temperature Profile
353.15	Fig. 4-1	Fig. 4-2	Fig. 4-3
363.15	Fig. 4-4	Fig. 4-5	Fig. 4-6
373.15	Fig. 4-7	Fig. 4-8	Fig. 4-9
383.15	Fig. 4-10	Fig. 4-11	Fig. 4-12
393.15	Fig. 4-13	Fig. 4-14	Fig. 4-15
403.15	Fig. 4-16	Fig. 4-17	Fig. 4-18
413.15	Fig. 4-19	Fig. 4-20	Fig. 4-21
423.15	Fig. 4-22	Fig. 4-23	Fig. 4-24

The bottom curvature, the pore wall temperature and the vapor pressure above the meniscus are known for each plot as boundary conditions. The curvature at the bottom of the meniscus relates the pressure difference between liquid and vapor for the operating condition of the heat pipe system. The pore wall temperature and the vapor pressure are assumed different for each plot. The local evaporation rate, meniscus profile and liquid interface temperature are calculated by starting from the bottom of the meniscus and, proceeding along up a pore wall, ending at the non-evaporating region. At the bottom of the meniscus, the evaporation rate is low since the interface temperature is lower than at other locations and the thermal energy tends to conduct to the upper region. The meniscus profile sets the thickness of the liquid film. The profile starts at the center of pore and approaches the pore wall. In each position, the liquid film thickness, the liquid interface temperature and the evaporation rate satisfy the mass, momentum and the energy balances. The evaporation rate is integrated from the meniscus bottom to the non-evaporating point to provide the total evaporation rate per a pore.

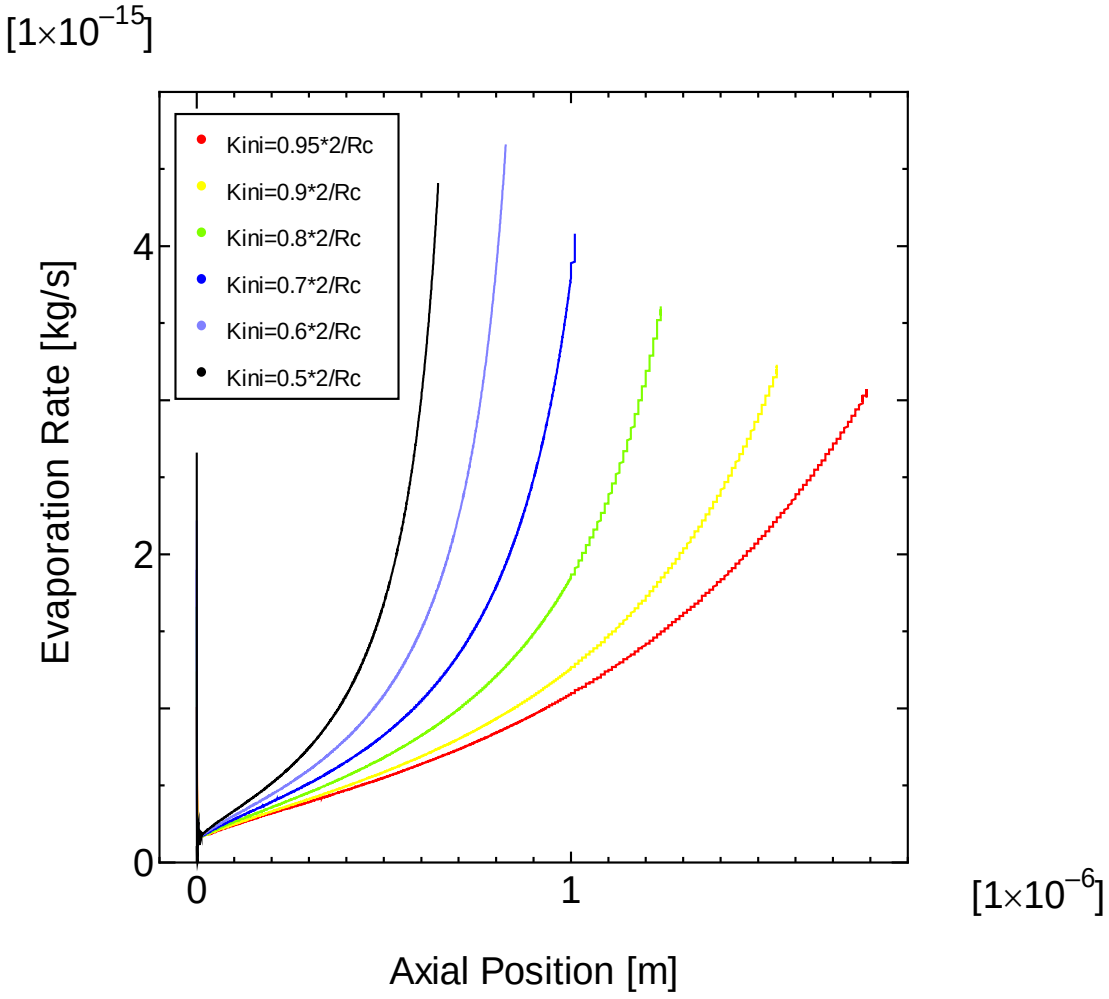

Fig. 4-1. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 353.15[K], Vapor Pressure: $P_{sat}(353.15K)$

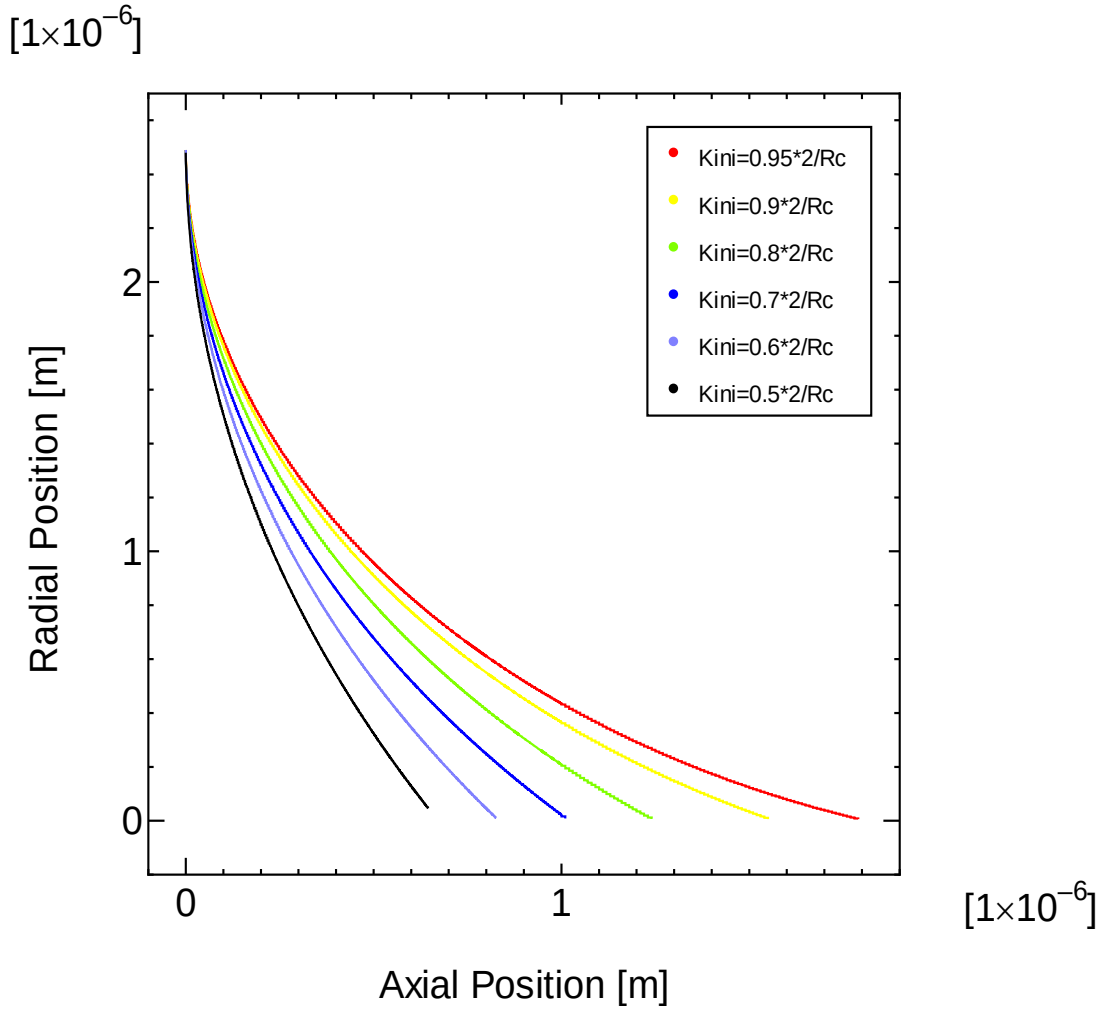

Fig. 4-2. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 353.15[K], Vapor Pressure: $P_{sat}(353.15K)$.

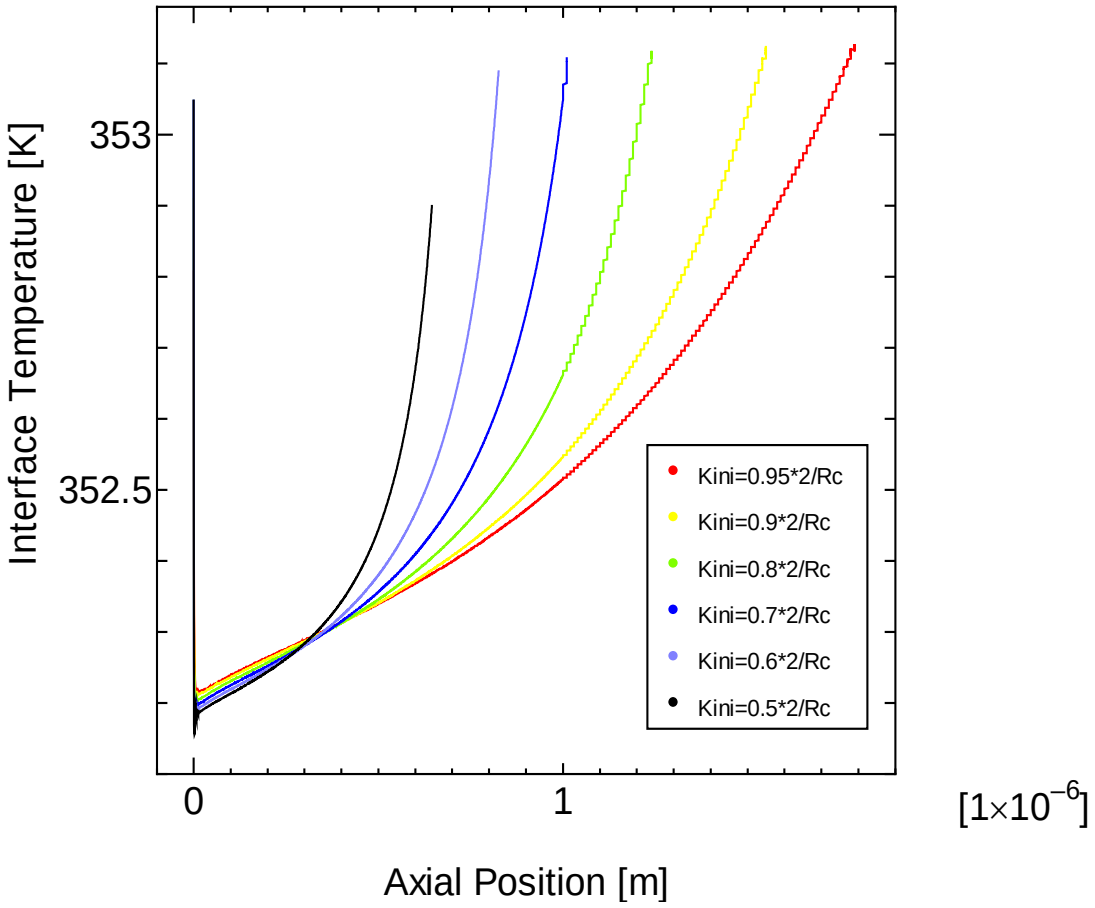

Fig. 4-3. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: $353.15[\mathrm{K}]$, Vapor Pressure: $P_{sat}(353.15K)$.

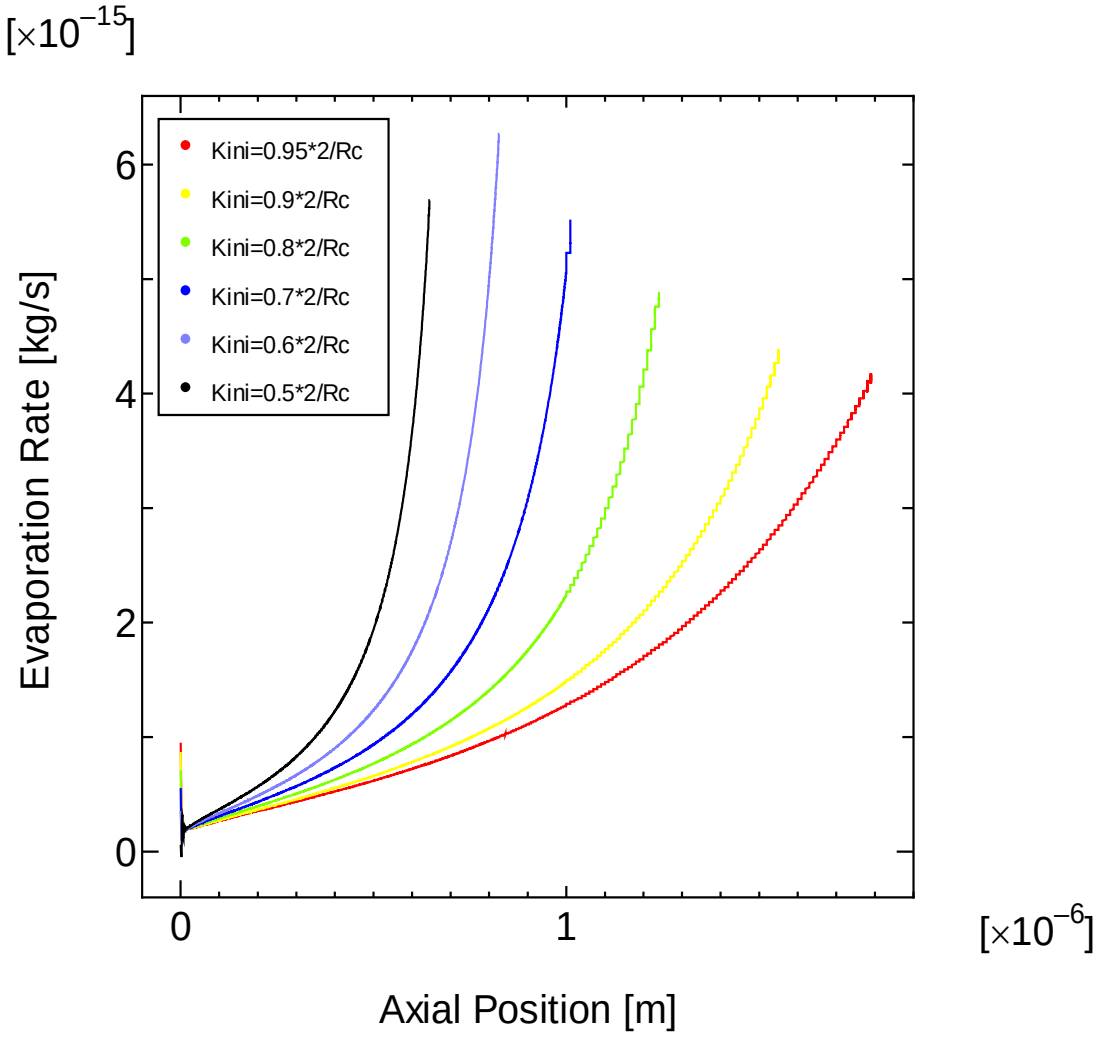

Fig. 4-4. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 363.15[K], Vapor Pressure: $P_{sat}(363.15K)$.

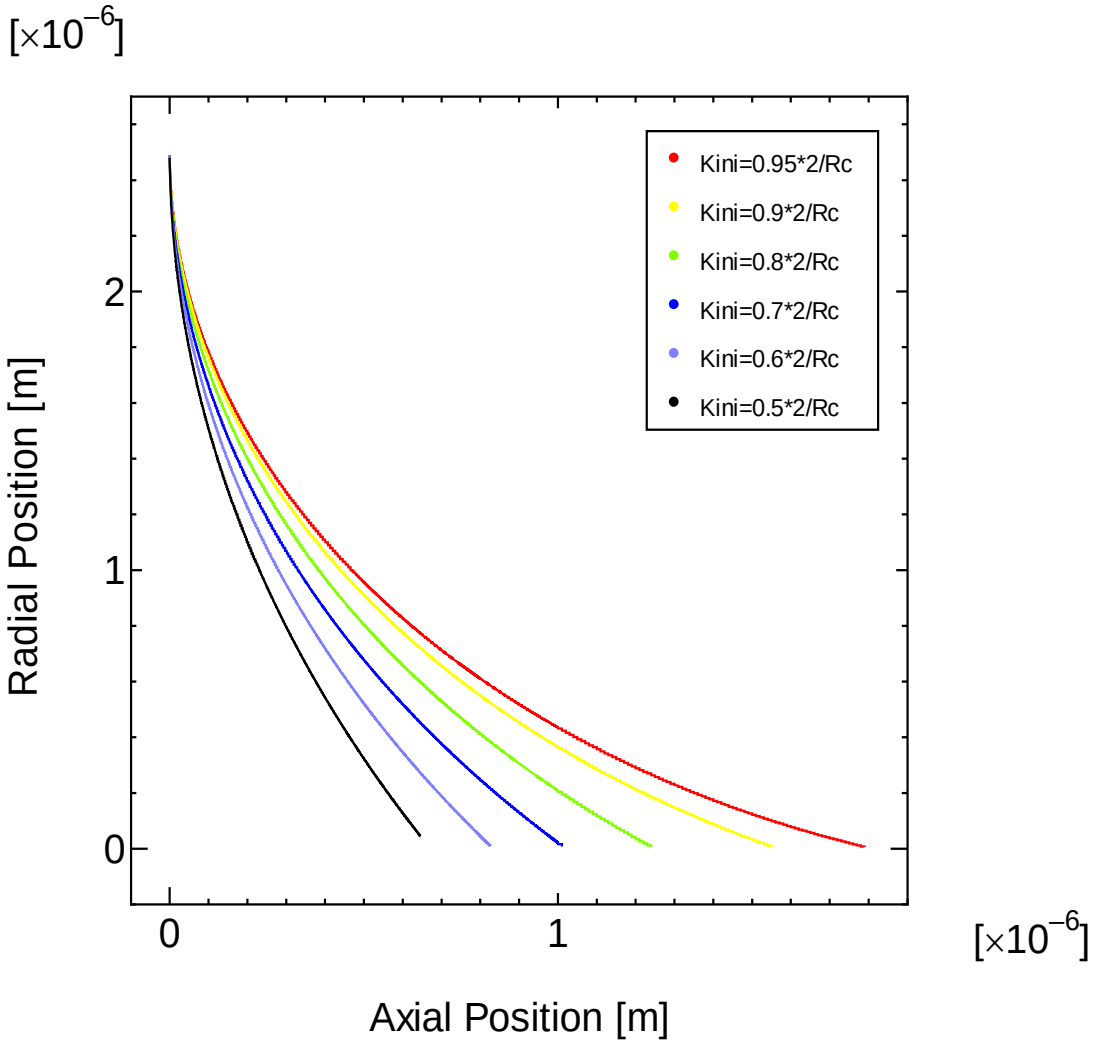

Fig. 4-5. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 363.15[K], Vapor Pressure: $P_{sat}(363.15K)$.

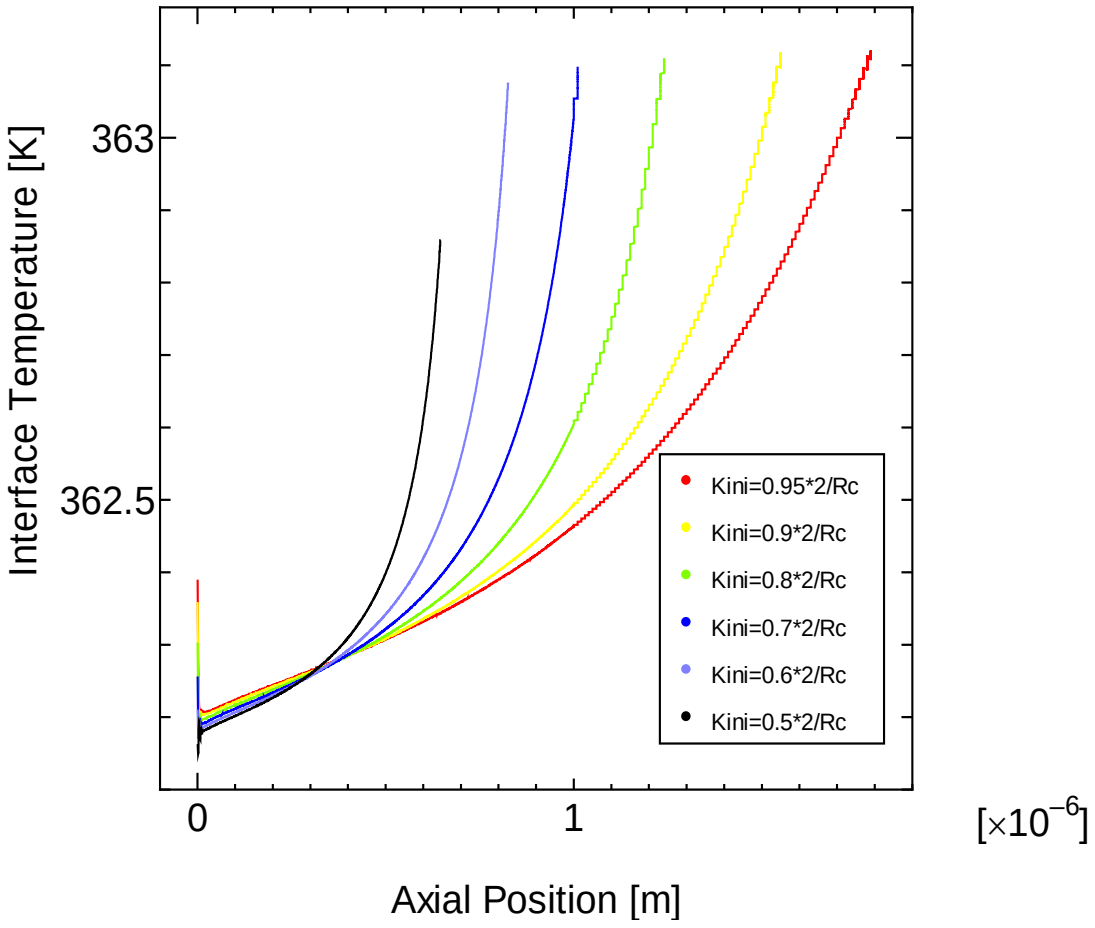

Fig. 4-6. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: $363.15[\mathrm{K}]$, Vapor Pressure: $P_{sat}(363.15K)$.

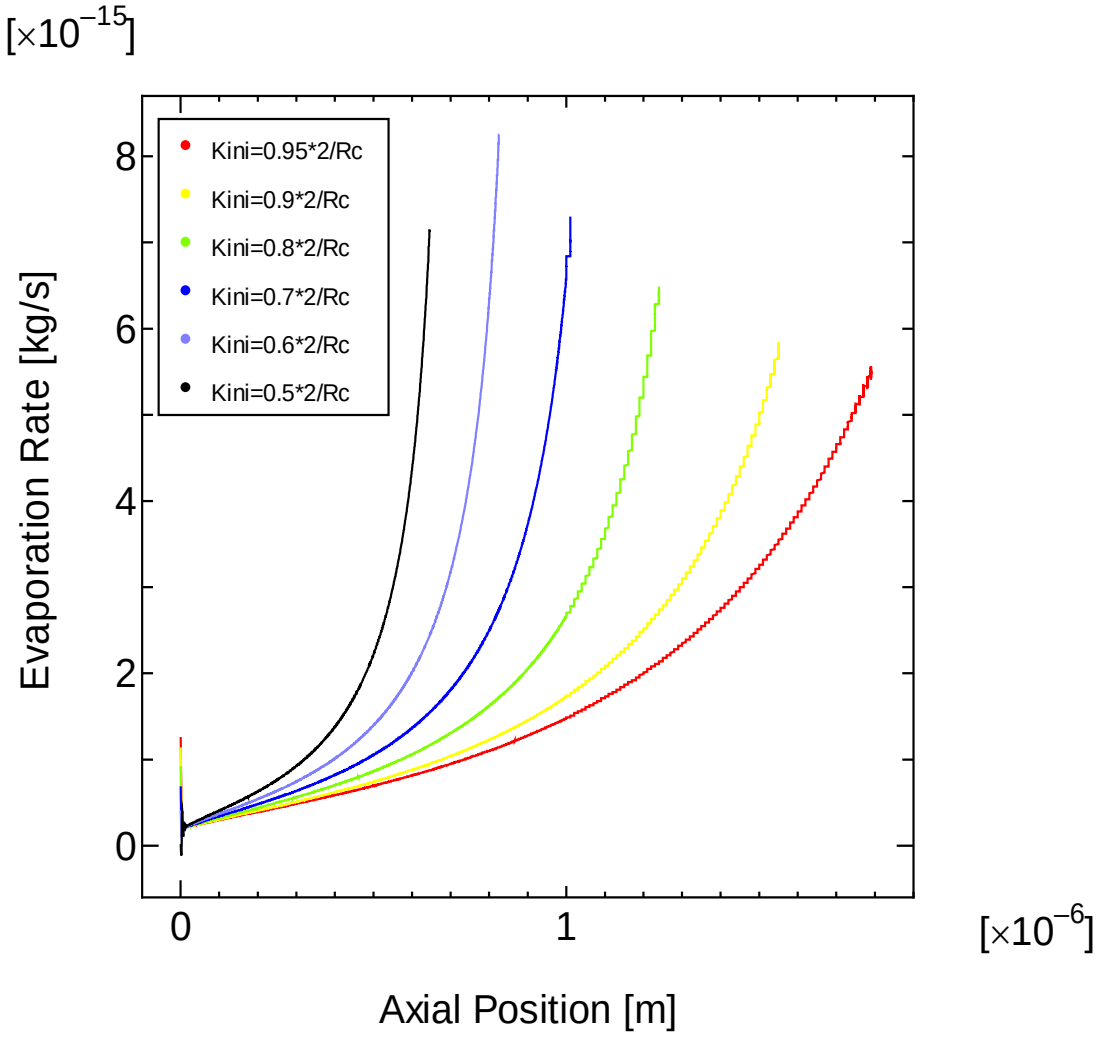

Fig. 4-7. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 373.15[K], Vapor Pressure: $P_{sat}(373.15K)$.

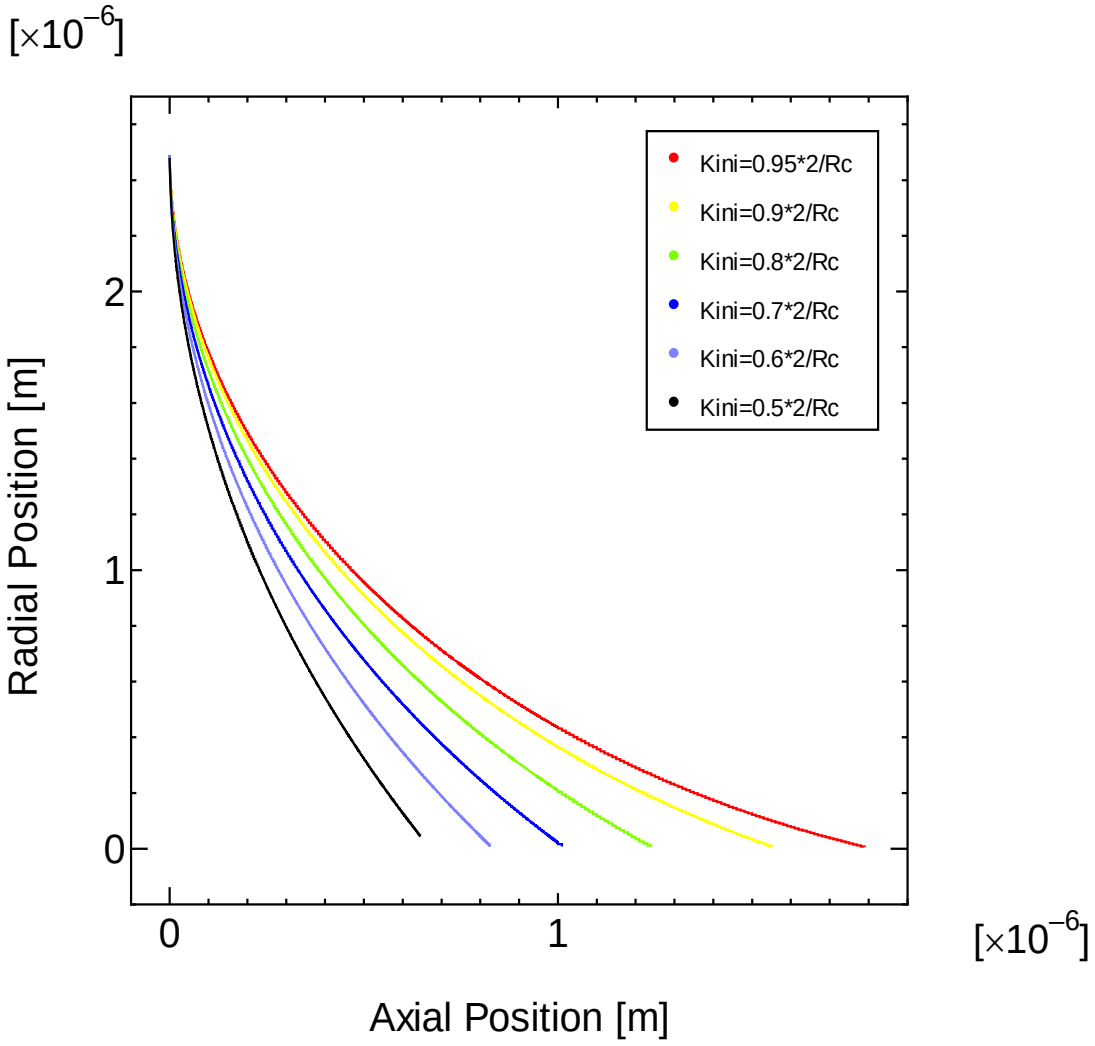

Fig. 4-8. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 373.15[K], Vapor Pressure: $P_{sat}(373.15K)$.

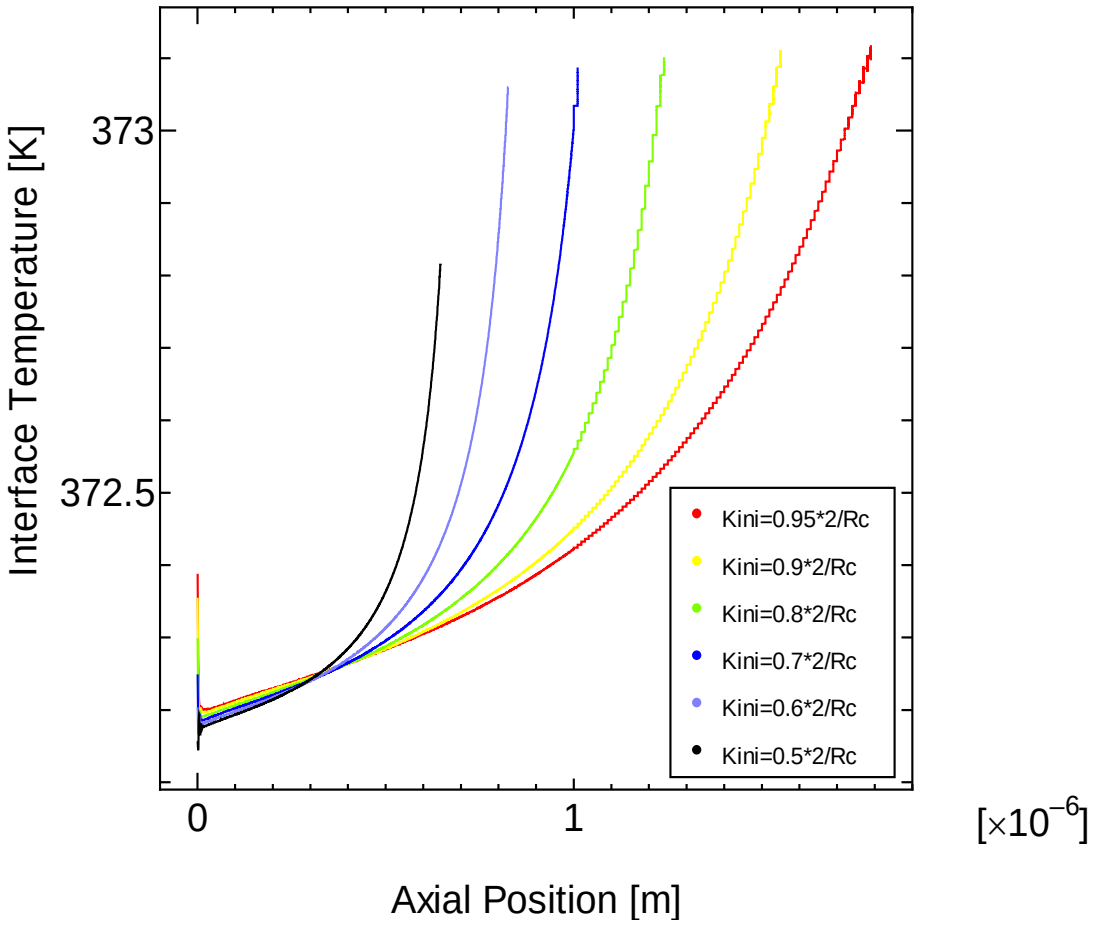

Fig. 4-9. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 373.15[K], Vapor Pressure: $P_{sat}(373.15K)$.

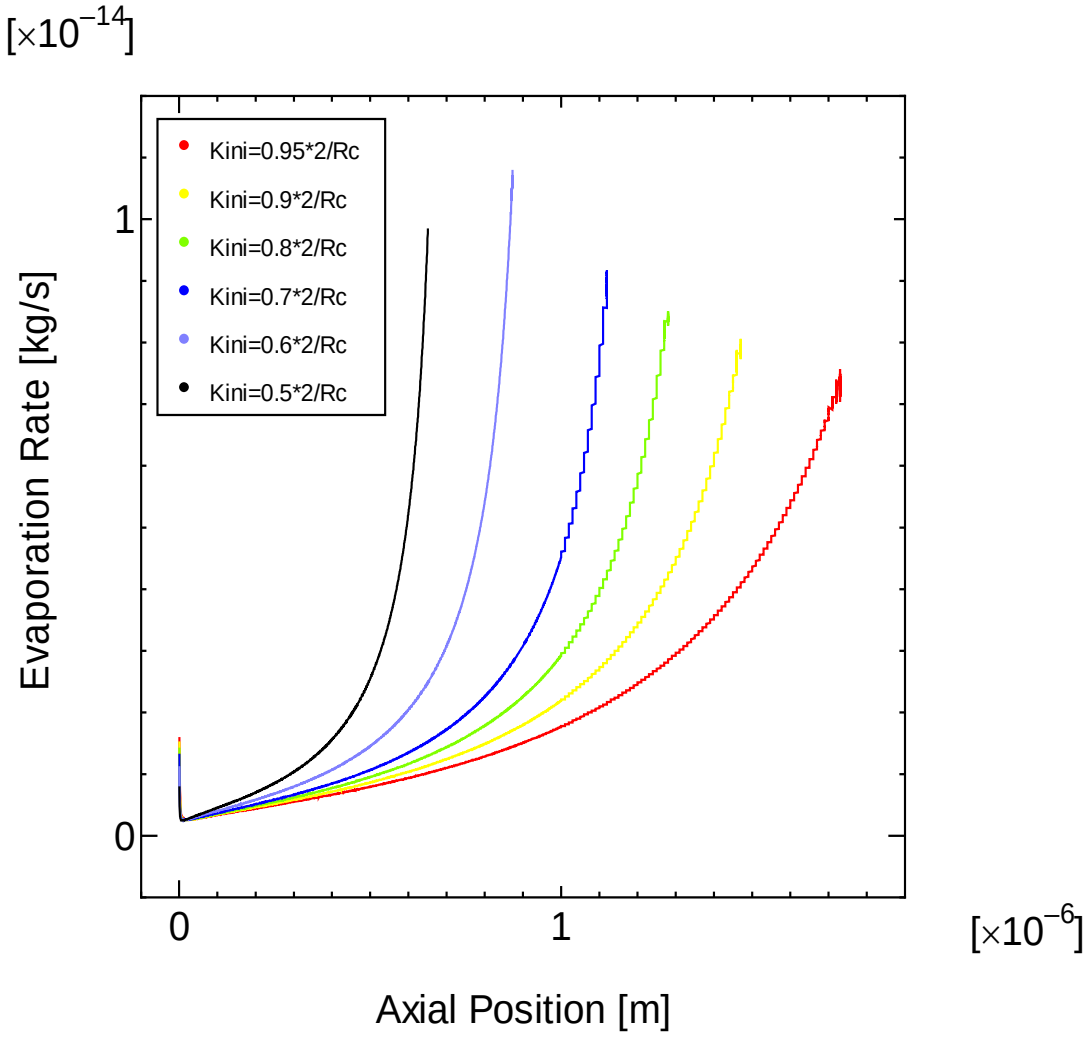

Fig. 4-10. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: $383.15[\mathrm{K}]$, Vapor Pressure: $P_{sat}(383.15K)$.

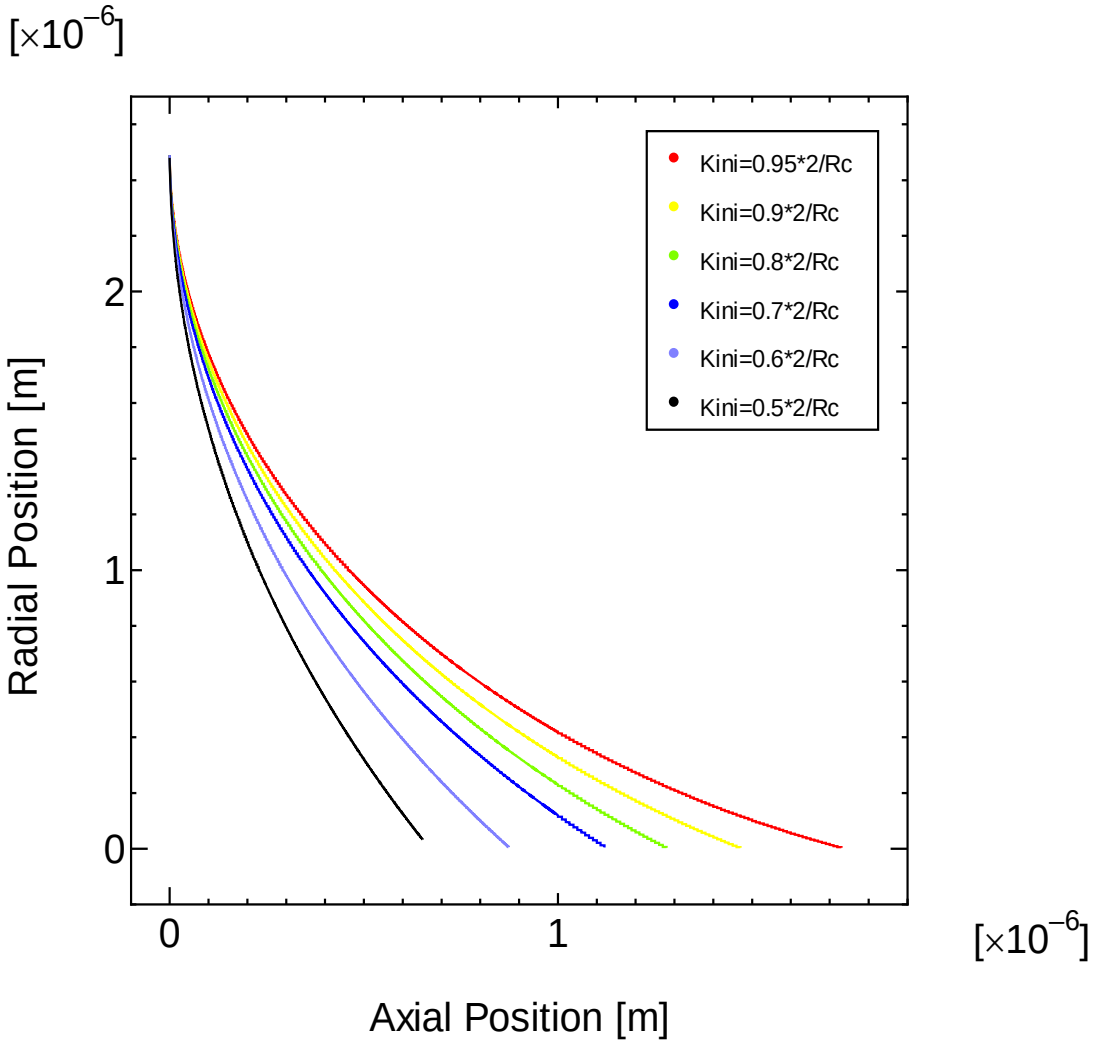

Fig. 4-11. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 383.15[K], Vapor Pressure: $P_{sat}(383.15K)$.

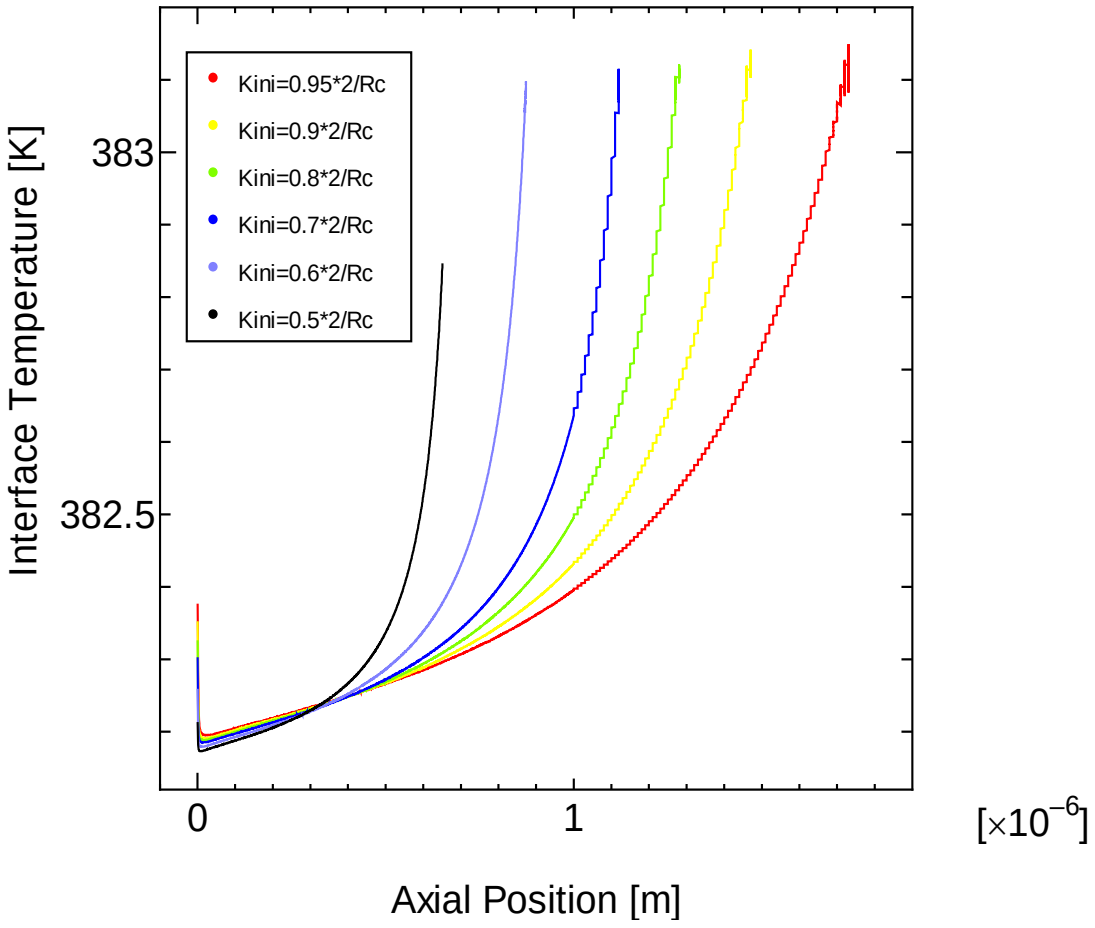

Fig. 4-12. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: $383.15[\mathrm{K}]$, Vapor Pressure: $P_{sat}(383.15K)$.

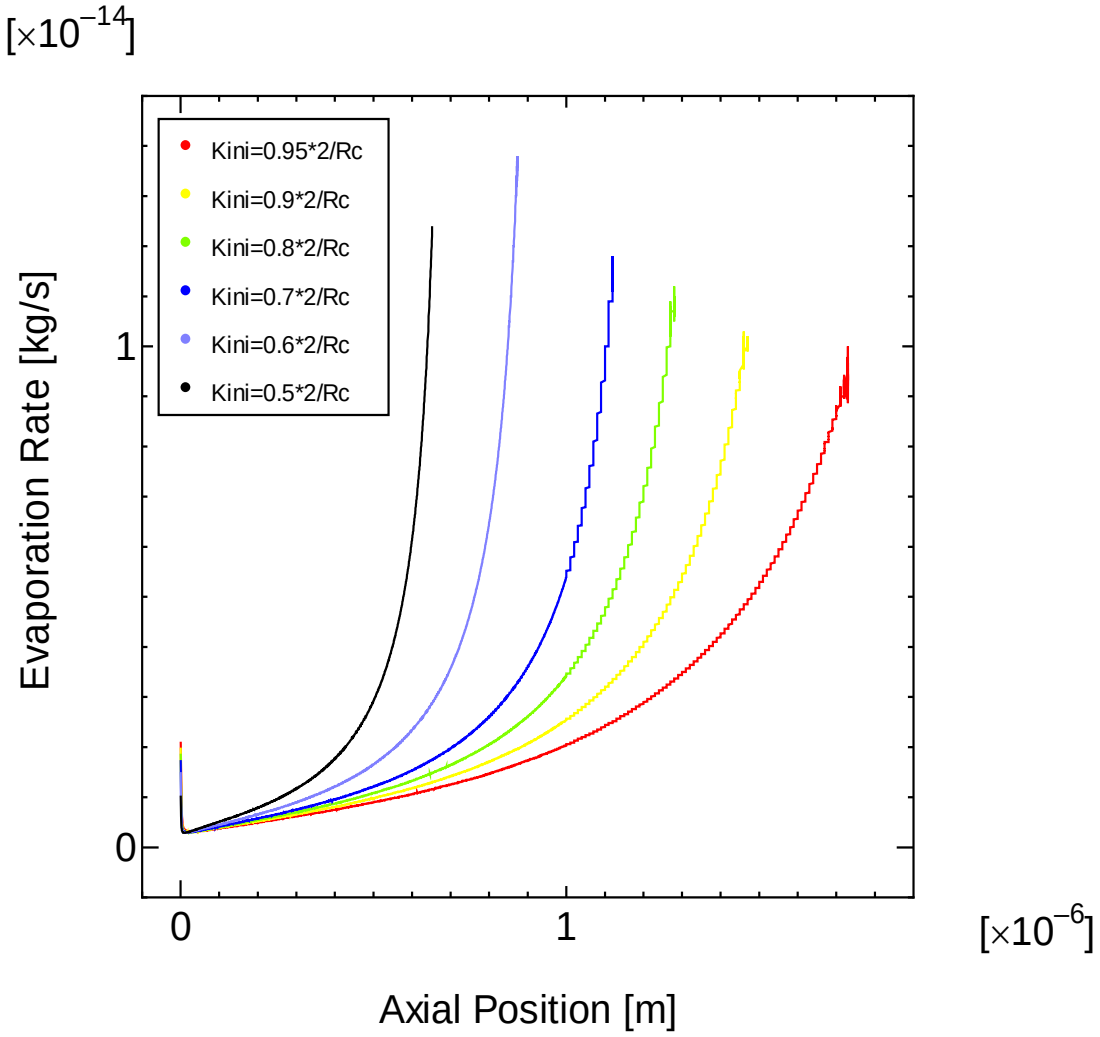

Fig. 4-13. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 393.15[K], Vapor Pressure: $P_{sat}(393.15K)$.

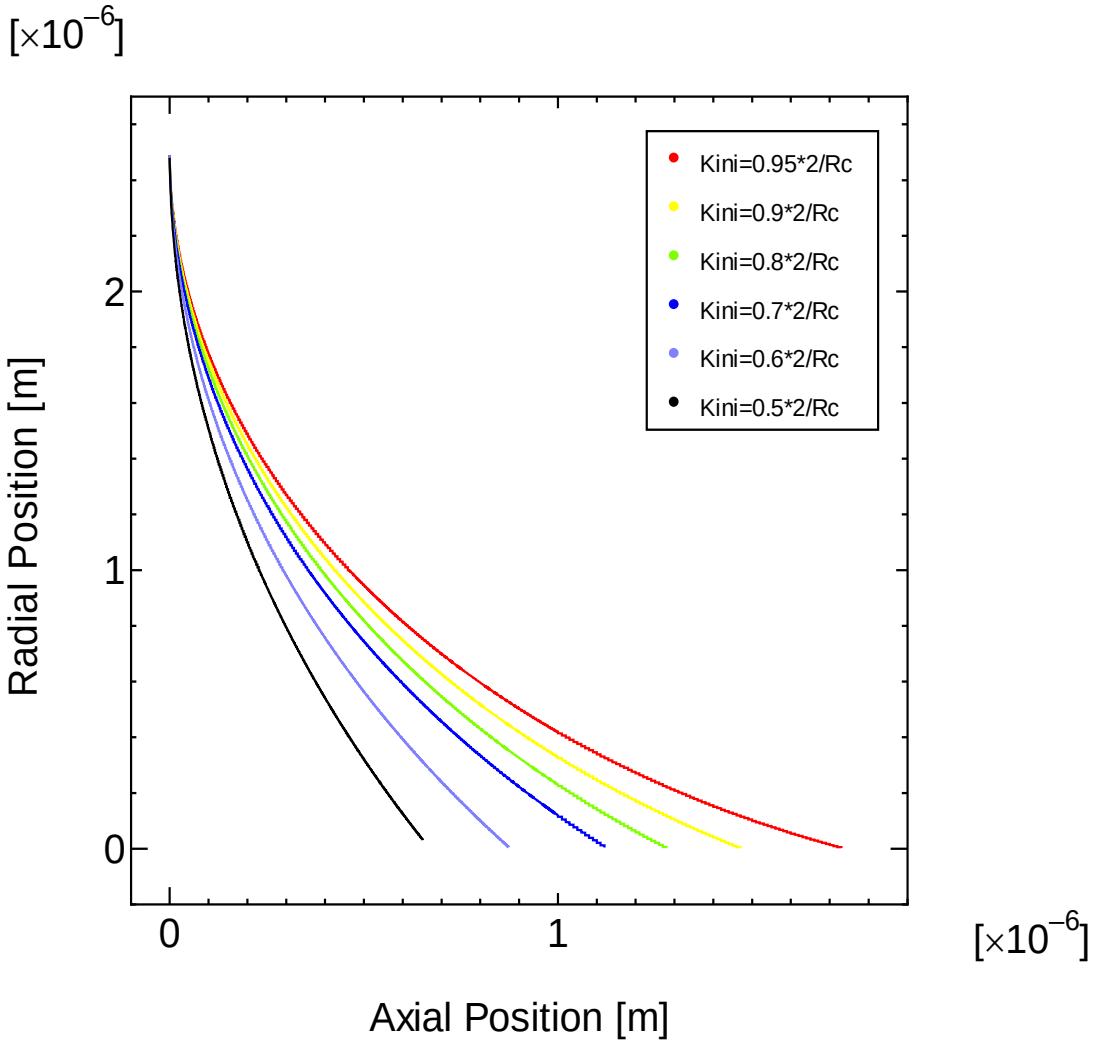

Fig. 4-14. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 393.15[K], Vapor Pressure: $P_{sat}(393.15K)$.

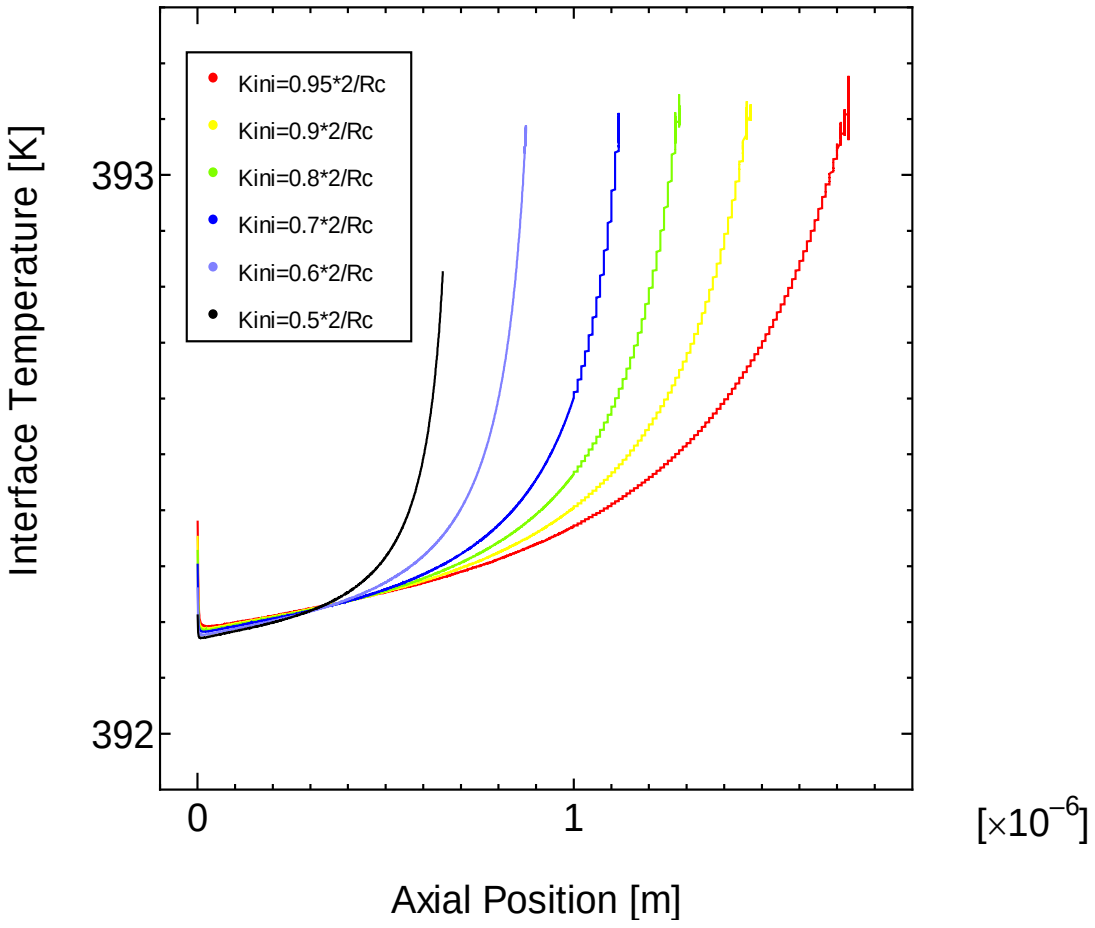

Fig. 4-15. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: $393.15[\mathrm{K}]$, Vapor Pressure: $P_{sat}(393.15K)$.

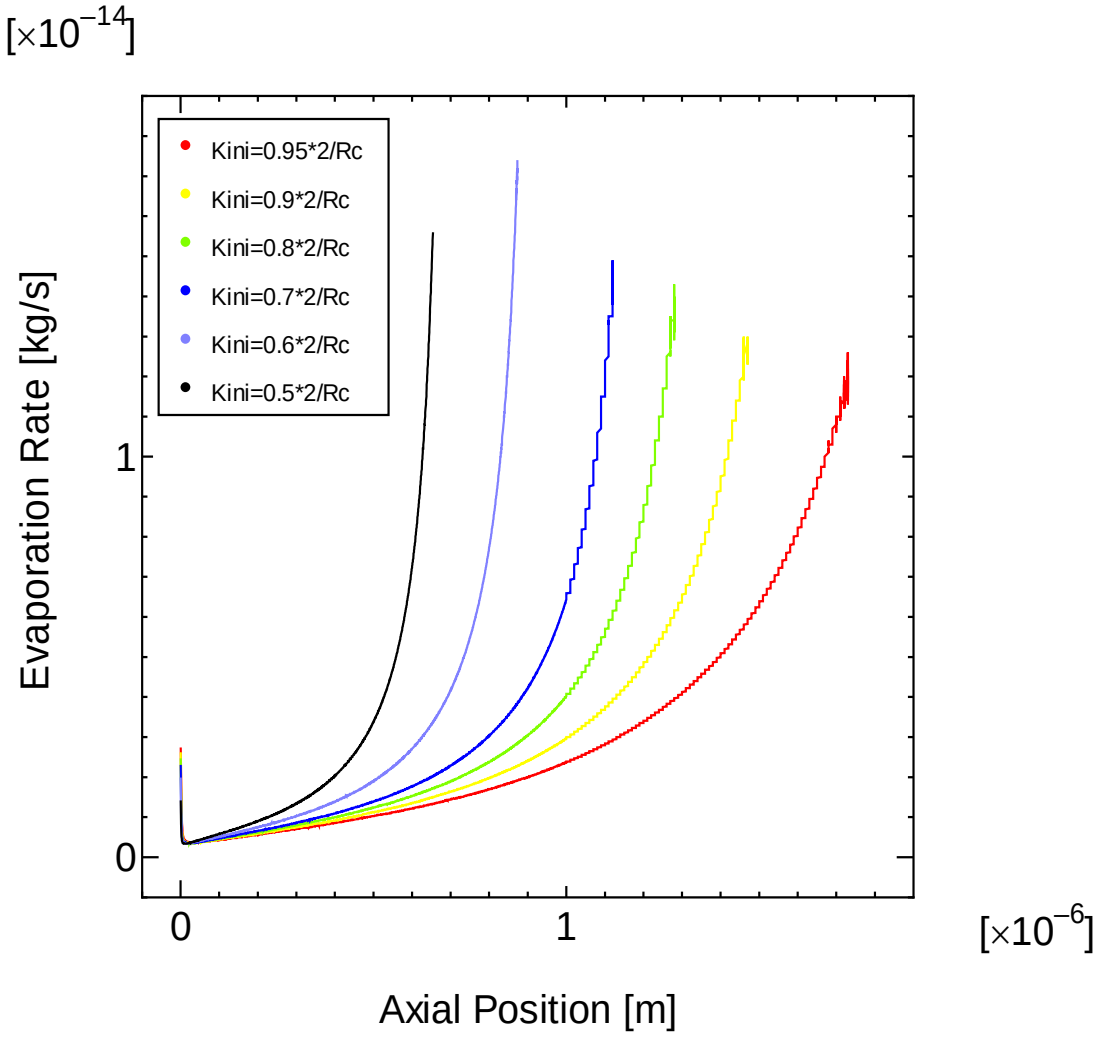

Fig. 4-16. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 403.15[K], Vapor Pressure: $P_{sat}(403.15K)$.

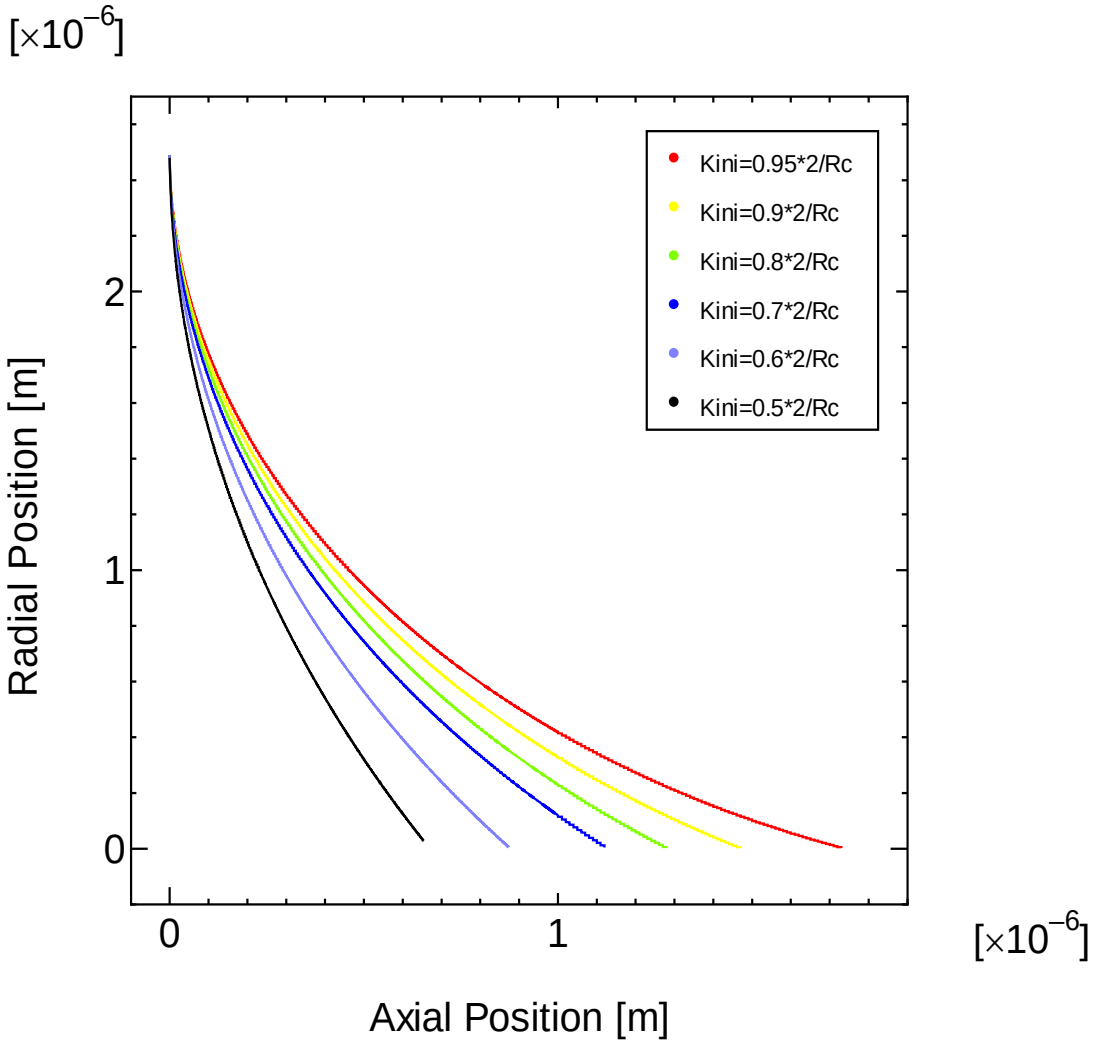

Fig. 4-17. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 403.15[K], Vapor Pressure: $P_{sat}(403.15K)$.

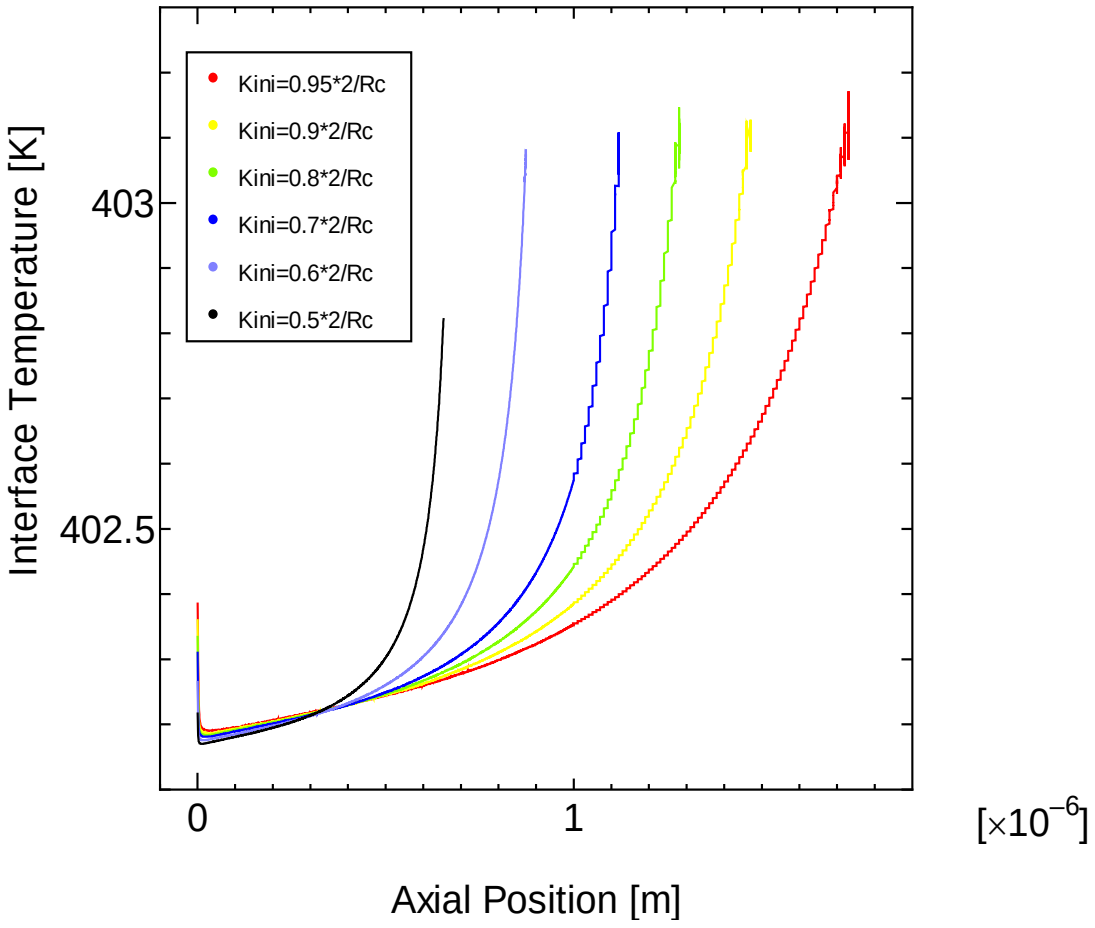

Fig. 4-18. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 403.15[K], Vapor Pressure: $P_{sat}(403.15K)$.

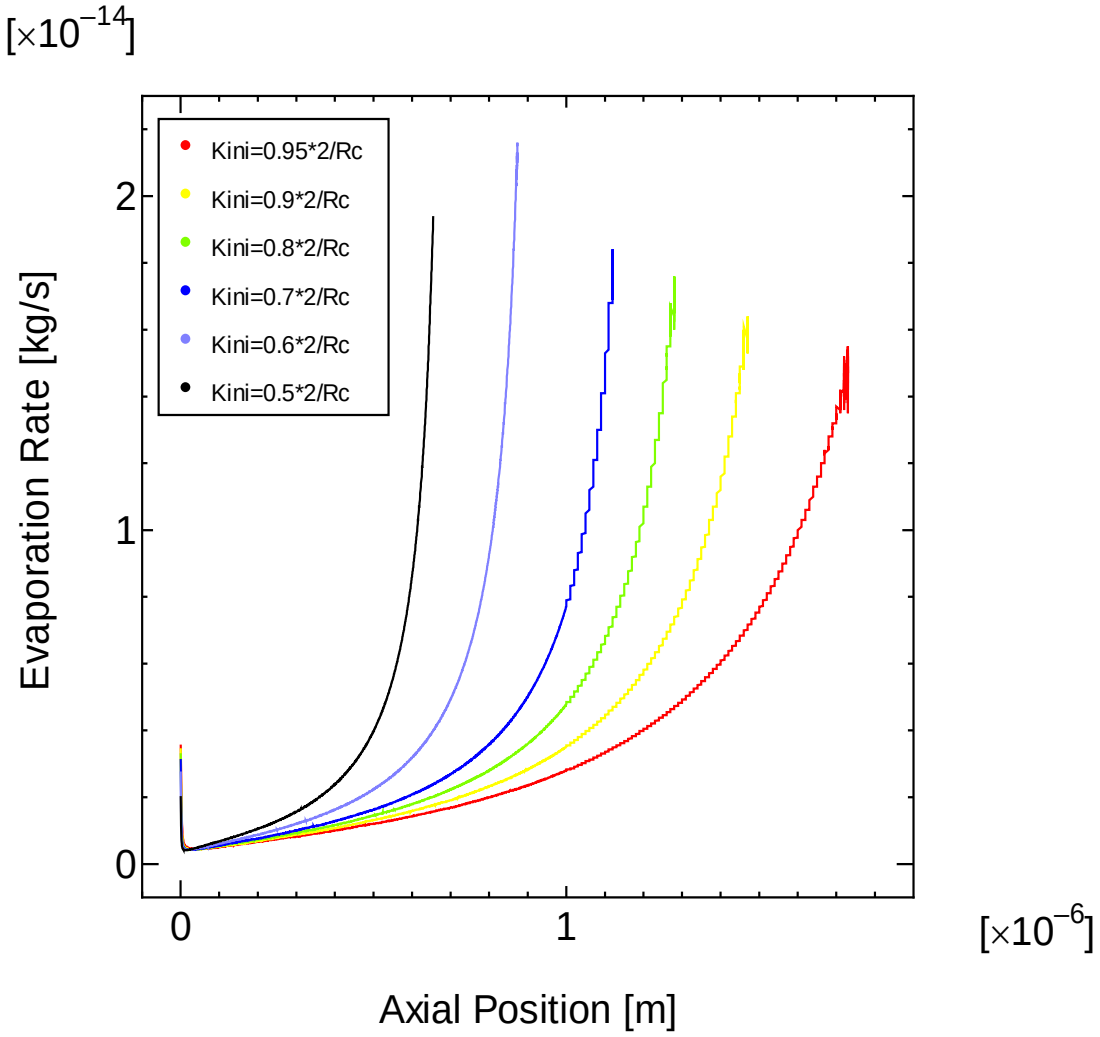

Fig. 4-19. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 413.15[K], Vapor Pressure: $P_{sat}(413.15K)$.

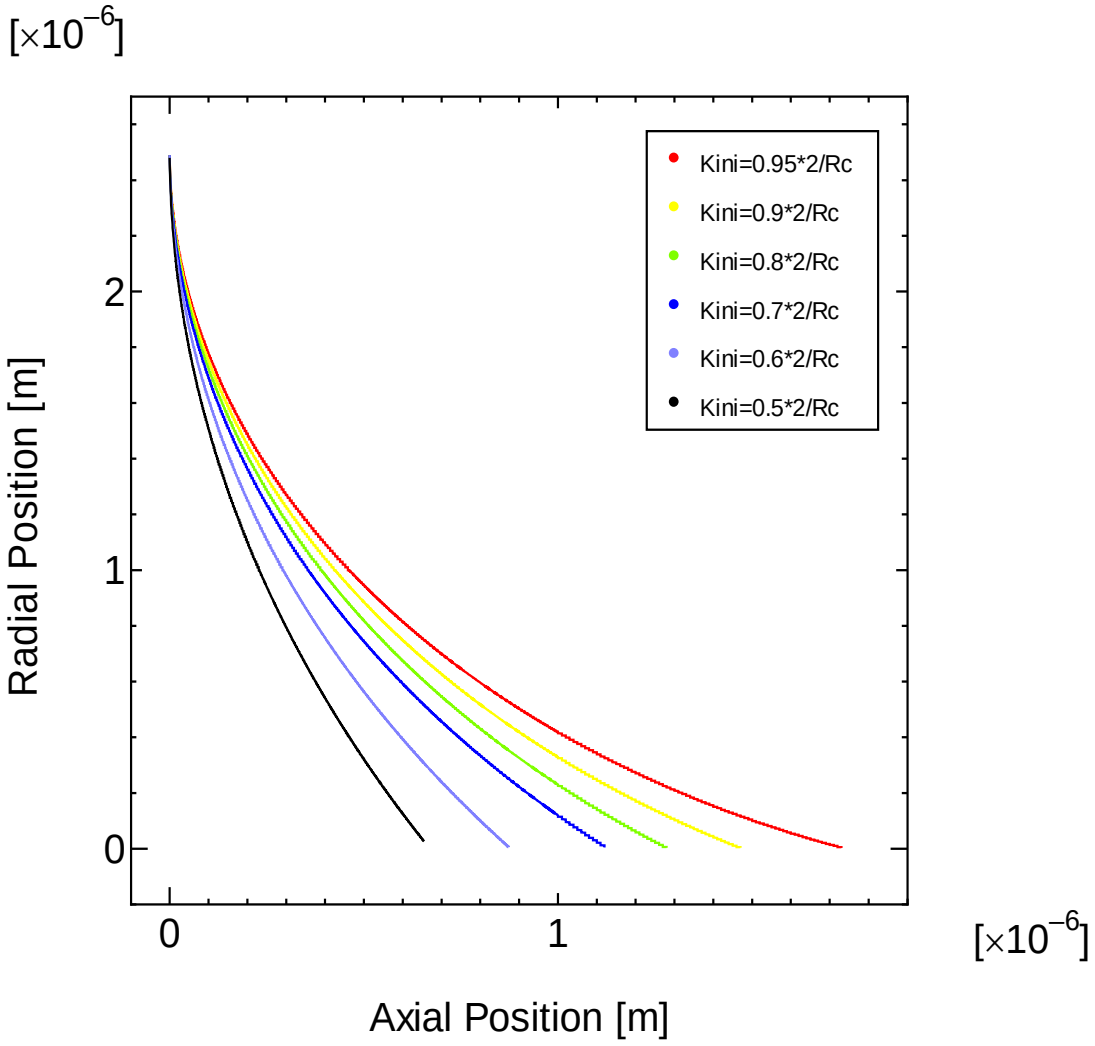

Fig. 4-20. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 413.15[K], Vapor Pressure: $P_{sat}(413.15K)$.

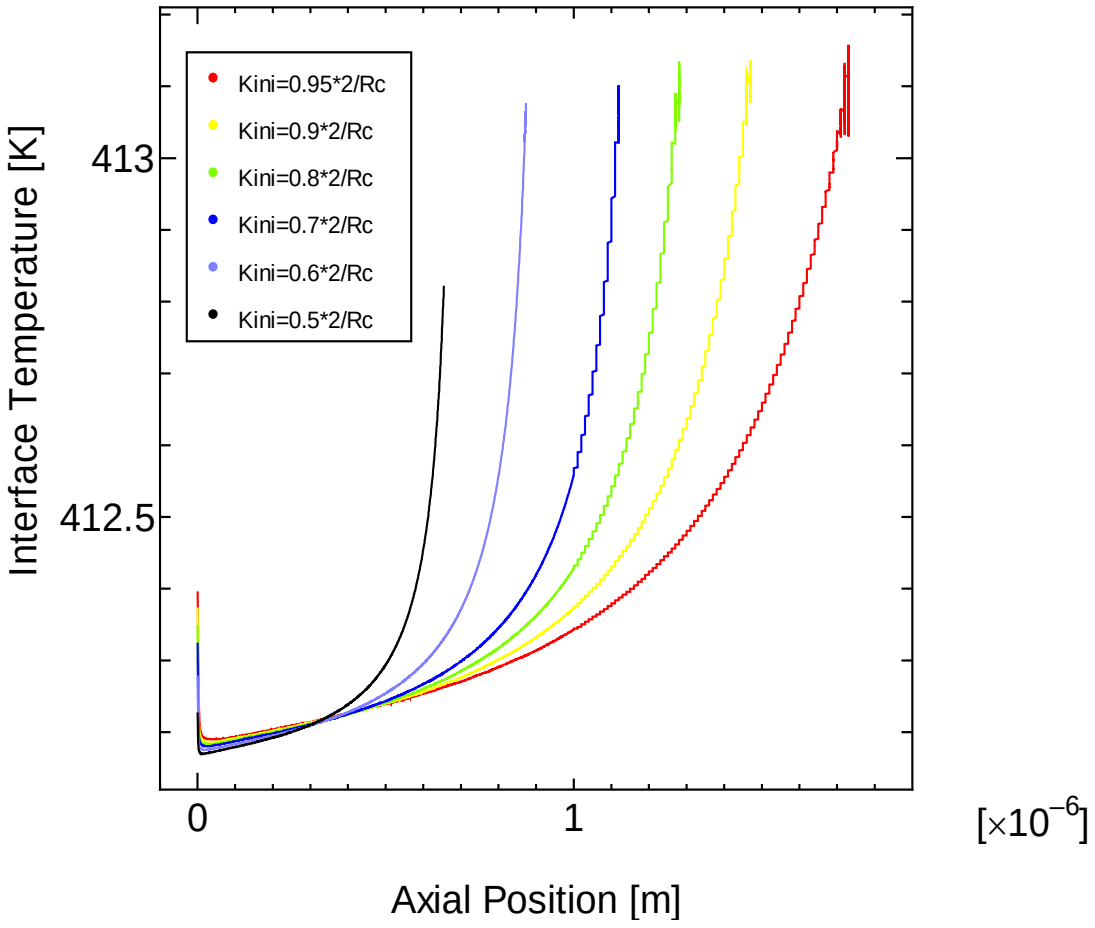

Fig. 4-21. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 413.15[K], Vapor Pressure: $P_{sat}(413.15K)$.

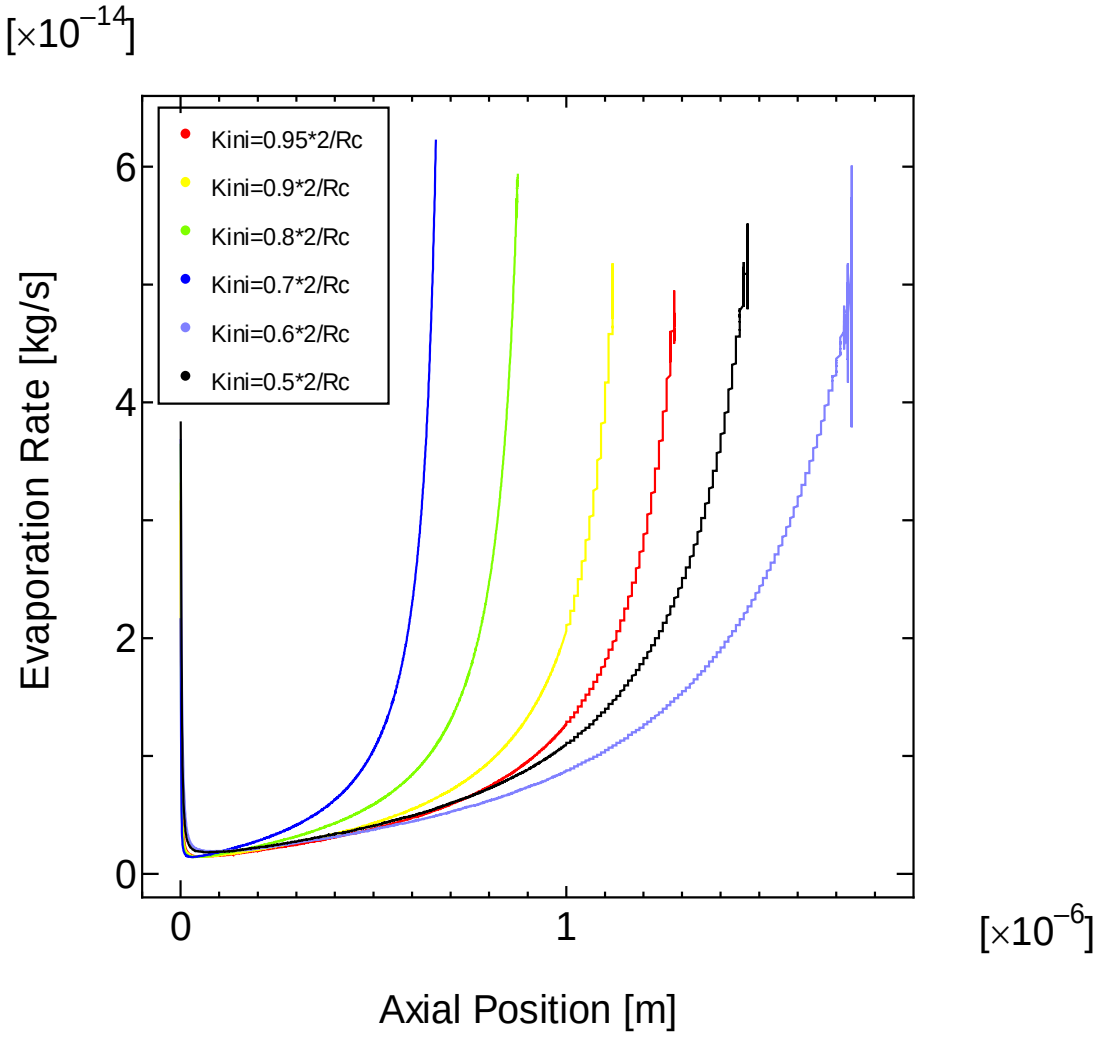

Fig. 4-22. Evaporation Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 423.15[K], Vapor Pressure: $P_{sat}(423.15K)$.

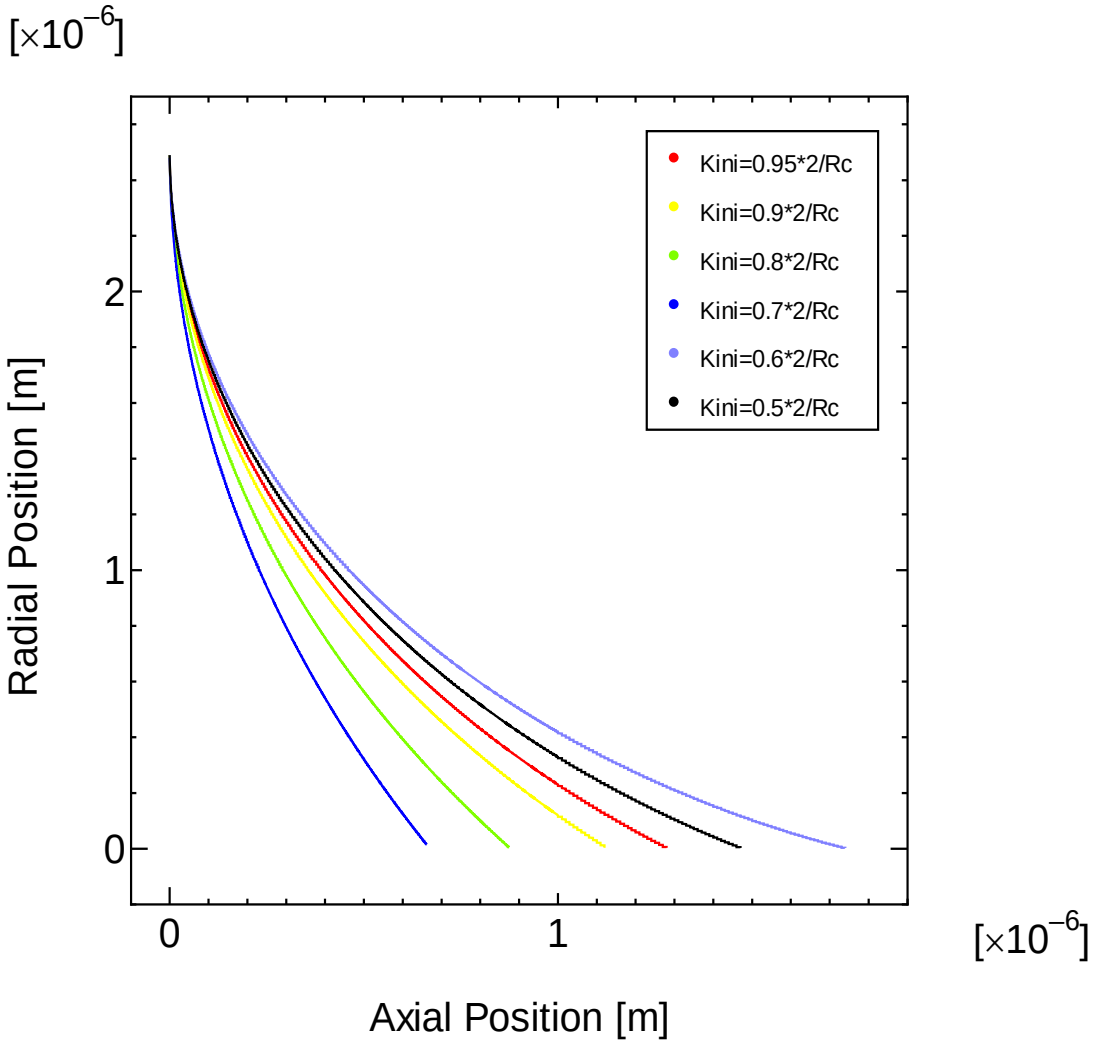

Fig. 4-23. Meniscus Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 423.15[K], Vapor Pressure: $P_{sat}(423.15K)$.

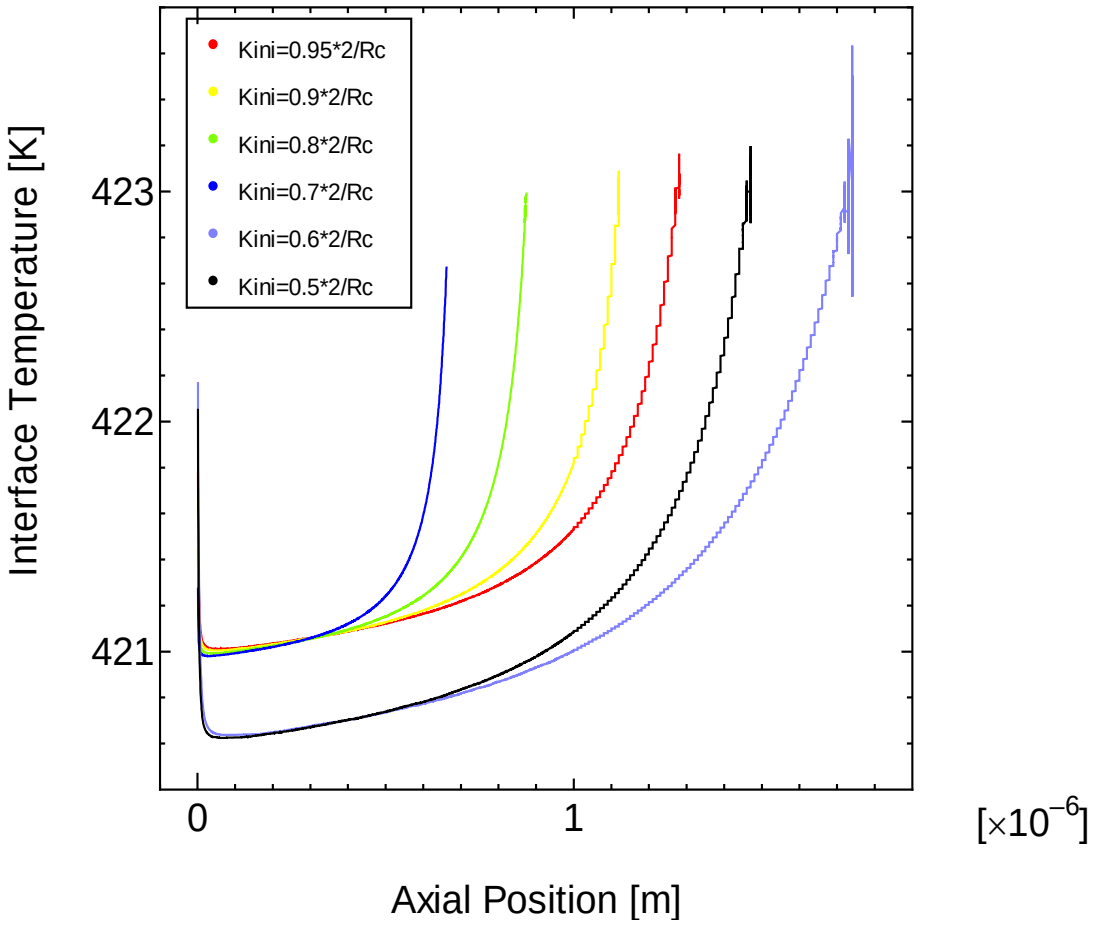

Fig. 4-24. Interface Temperature Profile in Pore for Pore Diameter: 5.0×10^{-6} [m], Wall Temperature: 423.15[K], Vapor Pressure: $P_{sat}(423.15K)$.

Total Evaporation Rate in Pore

The total evaporation rates per pore for given temperature and initial curvature at the pore diameter of 5.0×10^{-6} [m] are obtained to integrate evaporation rate along the pore axis position. The results are plotted through Fig. 4-26 to Fig. 4-33. Fig. 4-25 shows evaporation rate per pore under saturation pressure at several pore wall temperature in the evaporator.

TABLE 4-2 Total Evaporation Rate in a Pore for a Given Pore Wall Temperaturefor Pore Diameter=1.0×10^{-6} [m]

Temperature(K)	Fig. #
353.15	Fig. 4-26
363.15	Fig. 4-27
373.15	Fig. 4-28
383.15	Fig. 4-29
393.15	Fig. 4-30
400.15	Fig. 4-31
413.15	Fig. 4-32
423.15	Fig. 4-33

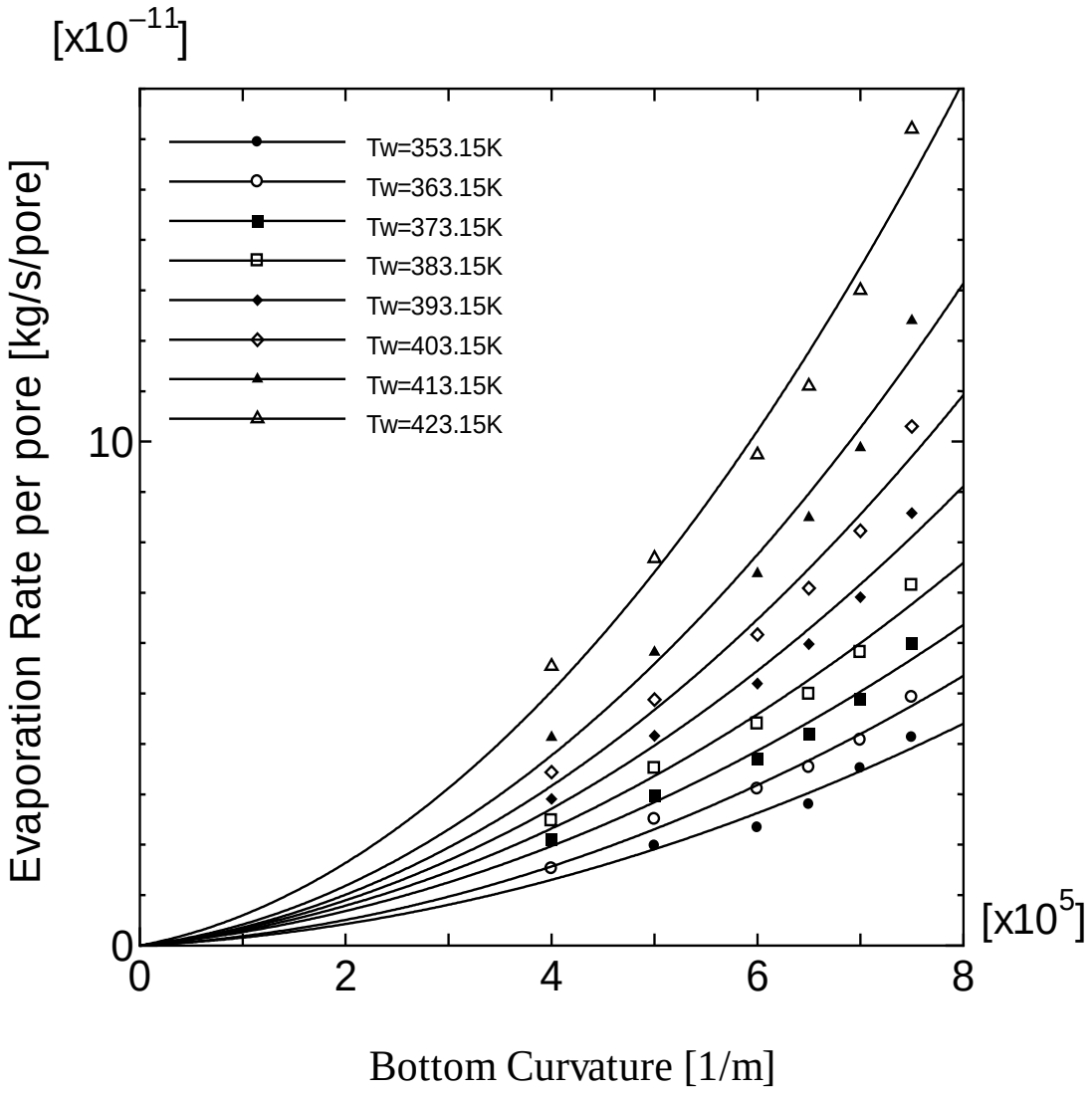

Fig. 4-25. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: $353.15-423.15$[K], Vapor Pressure: $P_{sat}(423.15K)$.

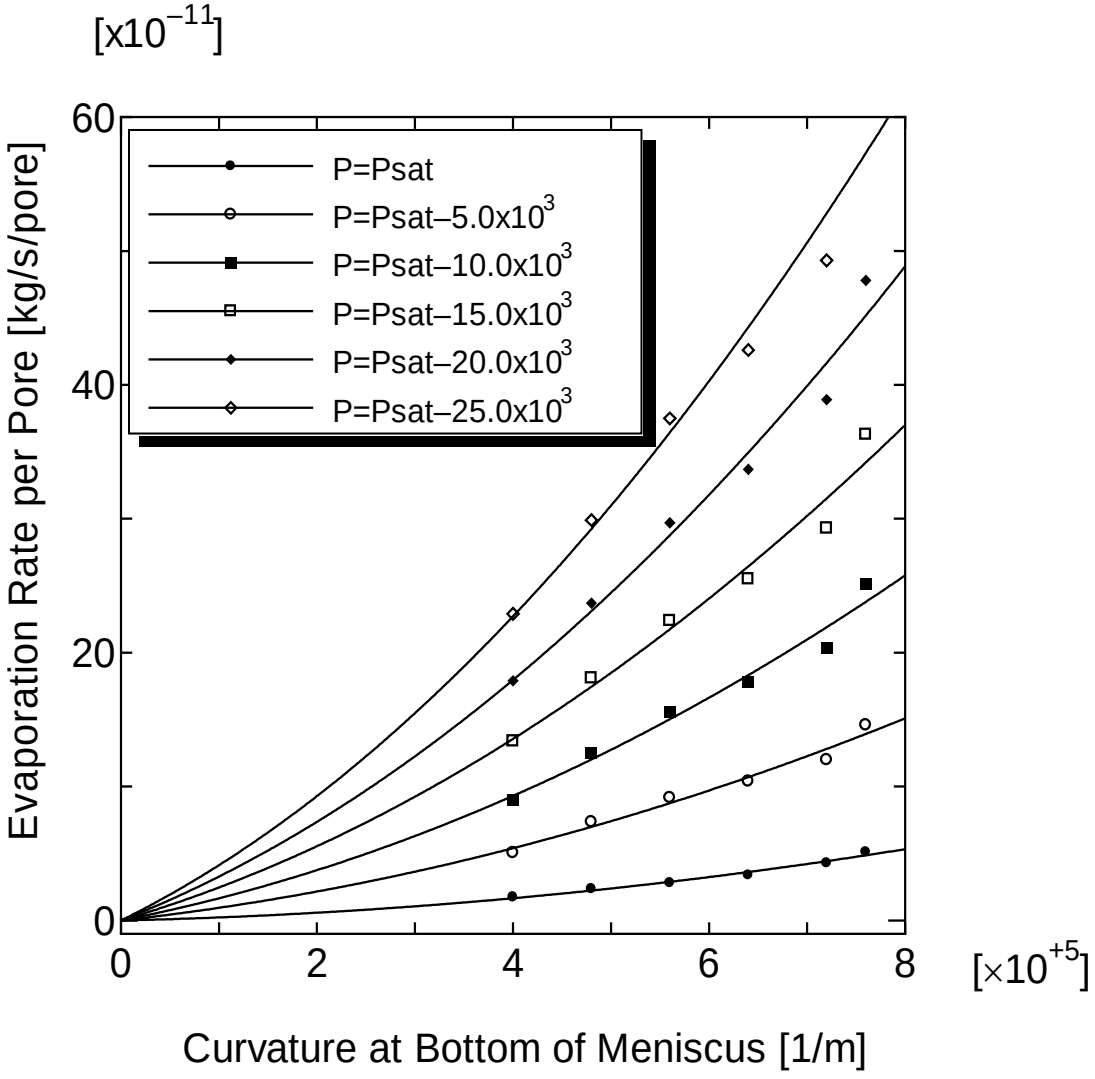

Fig. 4-26. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 353.15[K], Vapor Pressure: $P_{sat}(353.15K)$ to $P_{sat}(353.15K)-25\times10^{3}$ [Pa].

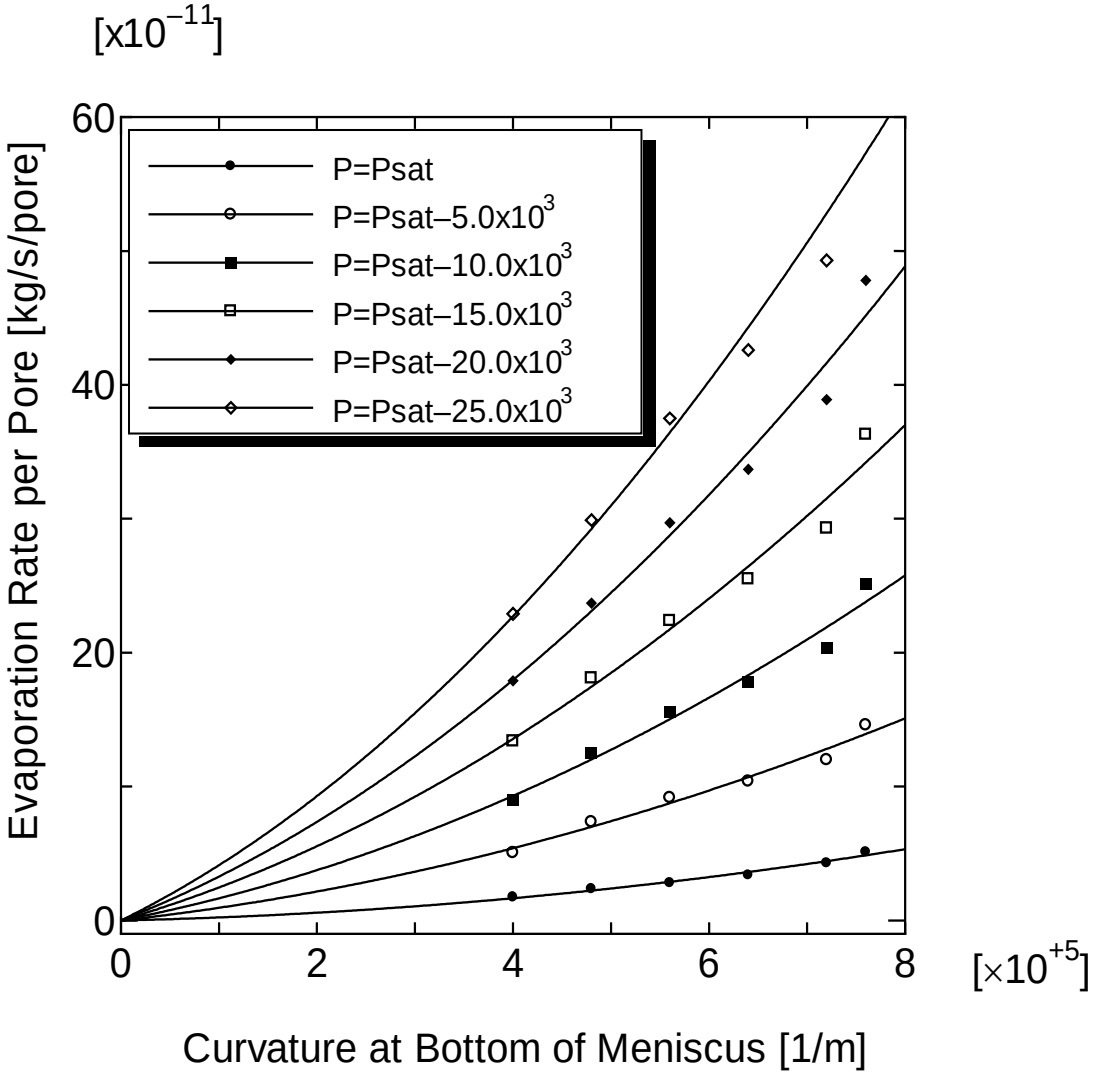

Fig. 4-27. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 363.15[K], Vapor Pressure: $P_{sat}(363.15K)$ to $P_{sat}(363.15K)-25\times10^{3}$ [Pa].

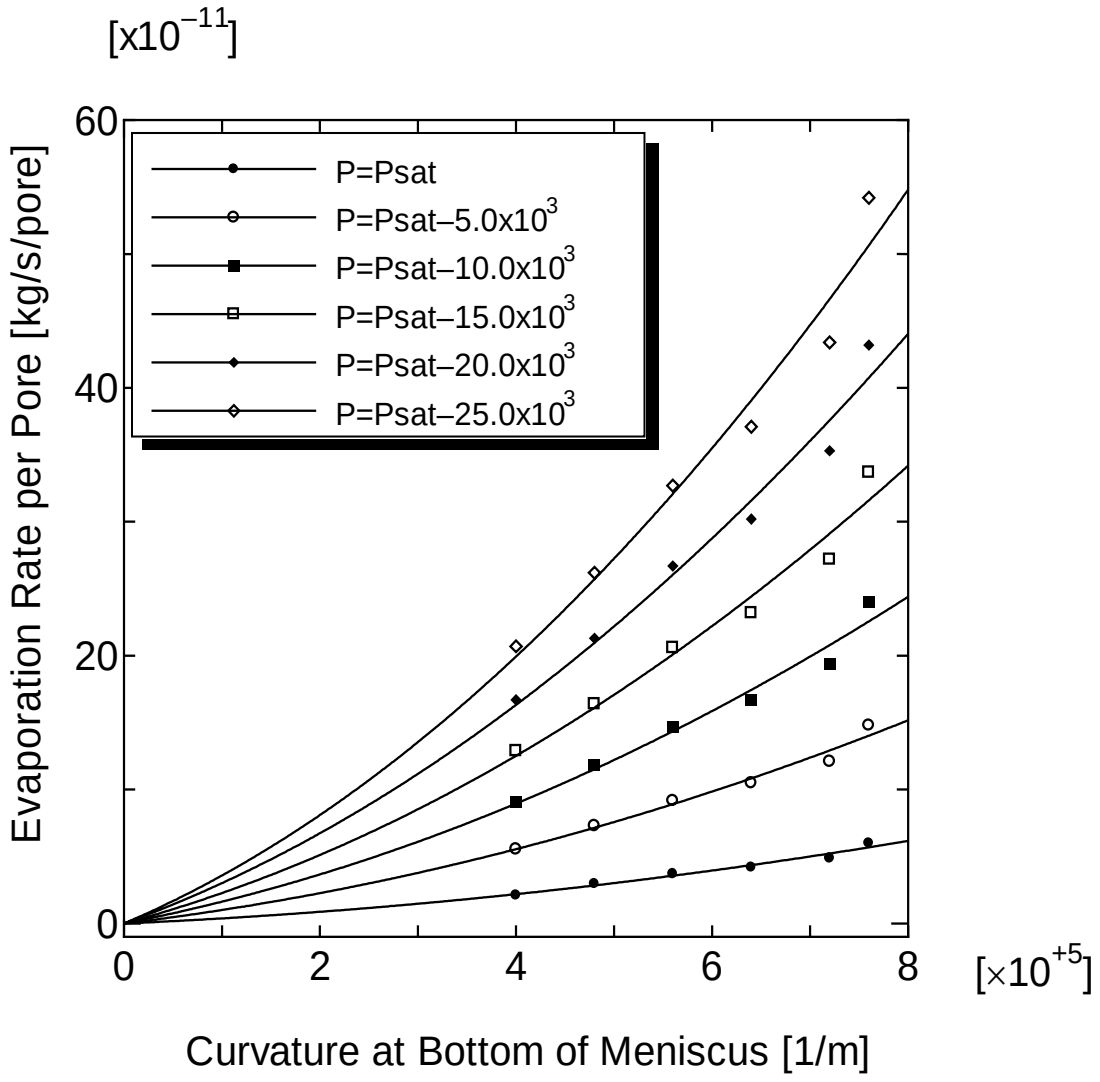

Fig. 4-28. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 373.15[K], Vapor Pressure: $P_{sat}(373.15K)$ to $P_{sat}(373.15K)-25\times10^{3}$ [Pa].

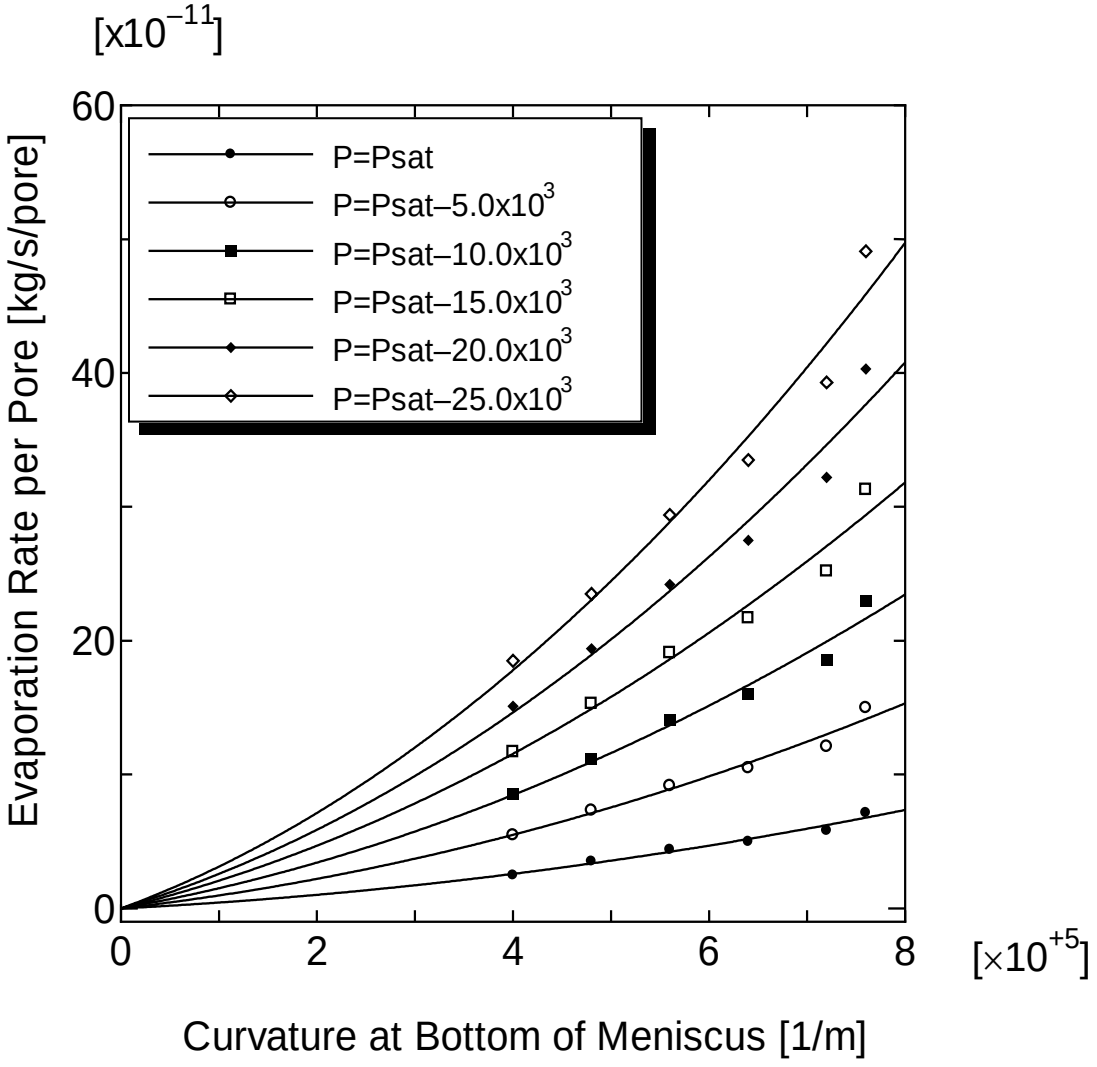

Fig. 4-29. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 383.15 [K], Vapor Pressure: $P_{sat}(383.15K)$ to $P_{sat}(383.15K)-25\times10^{3}$ [Pa].

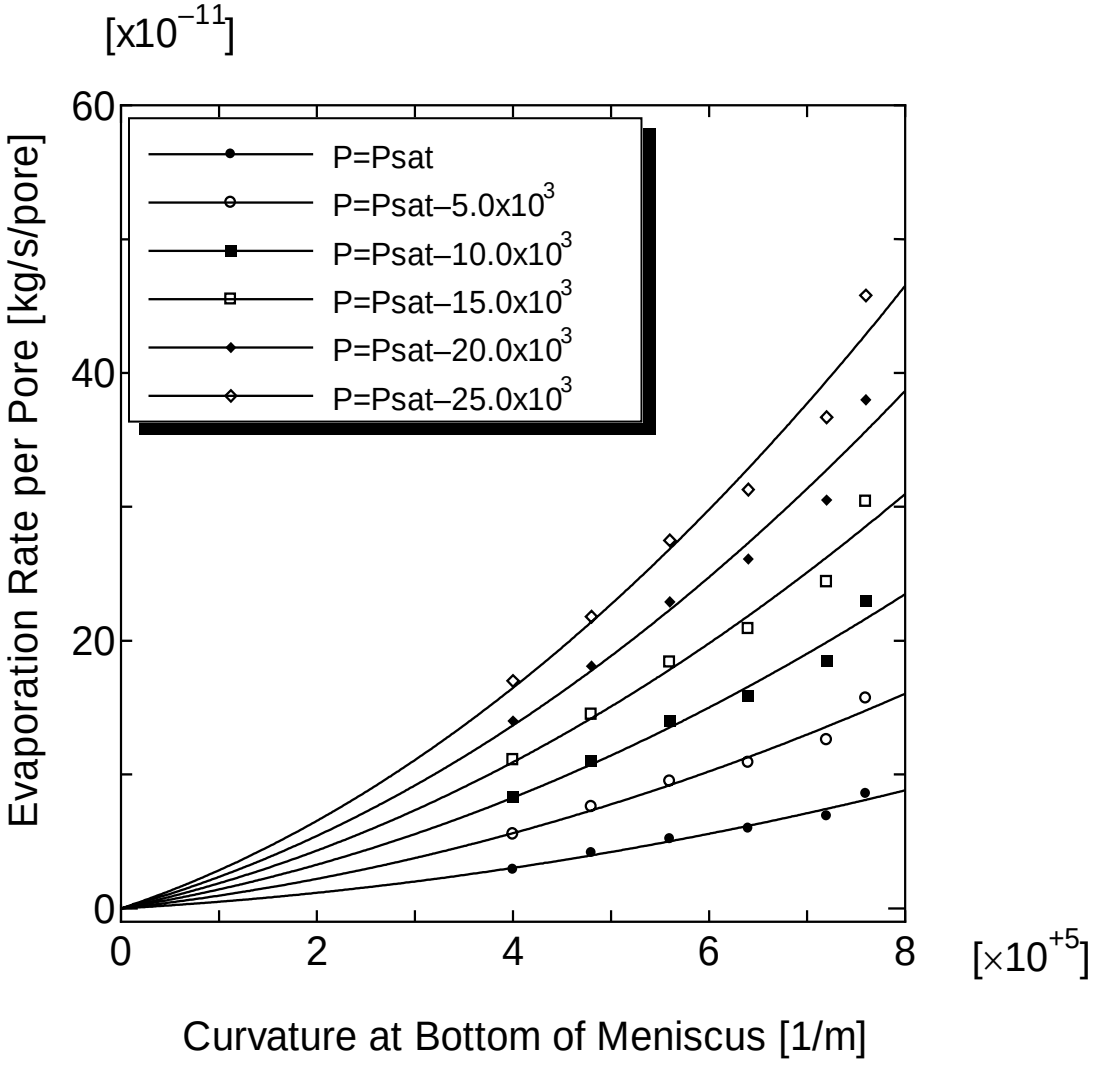

Fig. 4-30. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 393.15[K], Vapor Pressure: $P_{sat}(393.15K)$ to $P_{sat}(393.15K)-25\times10^{3}$ [Pa].

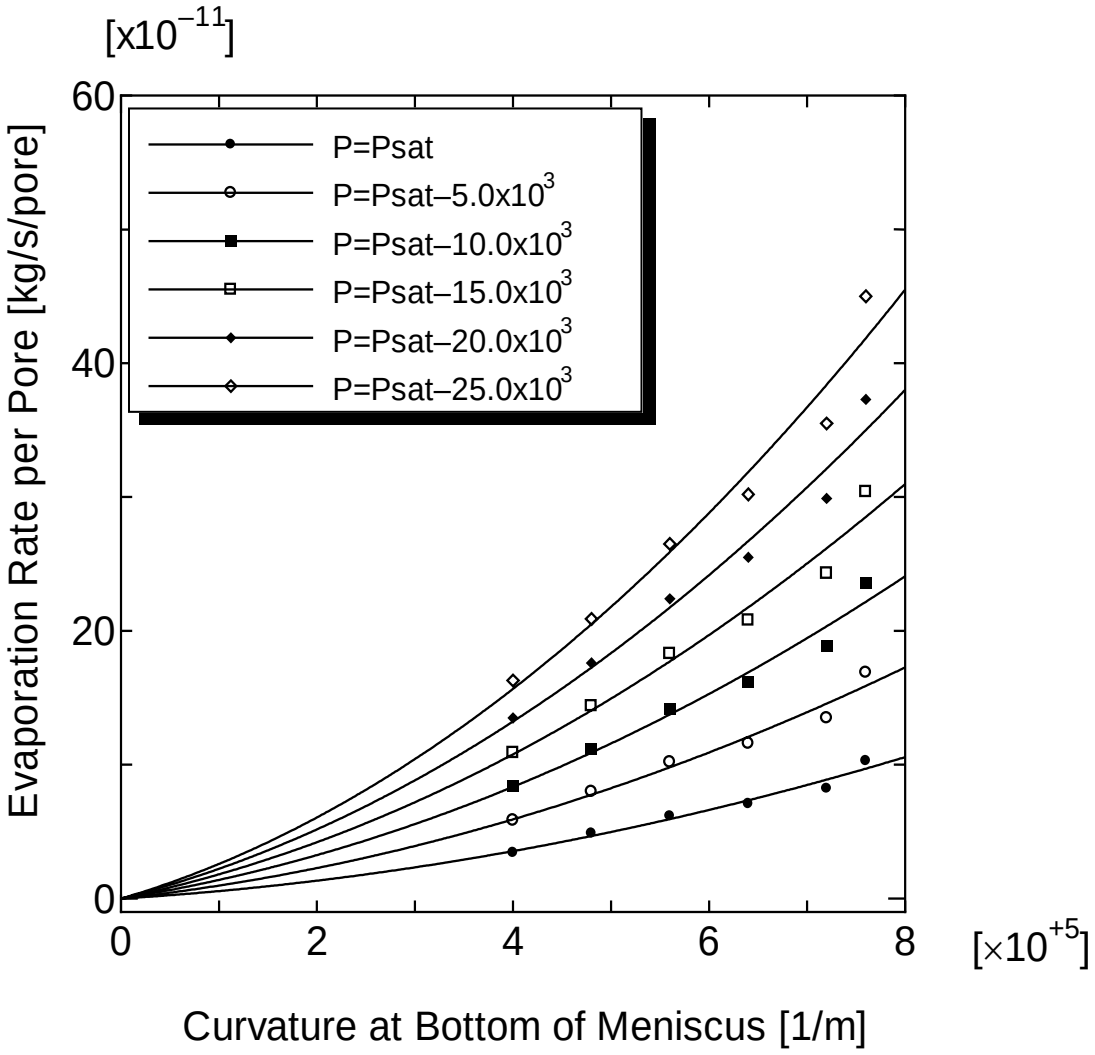

Fig. 4-31. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 403.15[K], Vapor Pressure: $P_{sat}(403.15K)$ to $P_{sat}(403.15K)-25\times10^{3}$ [Pa].

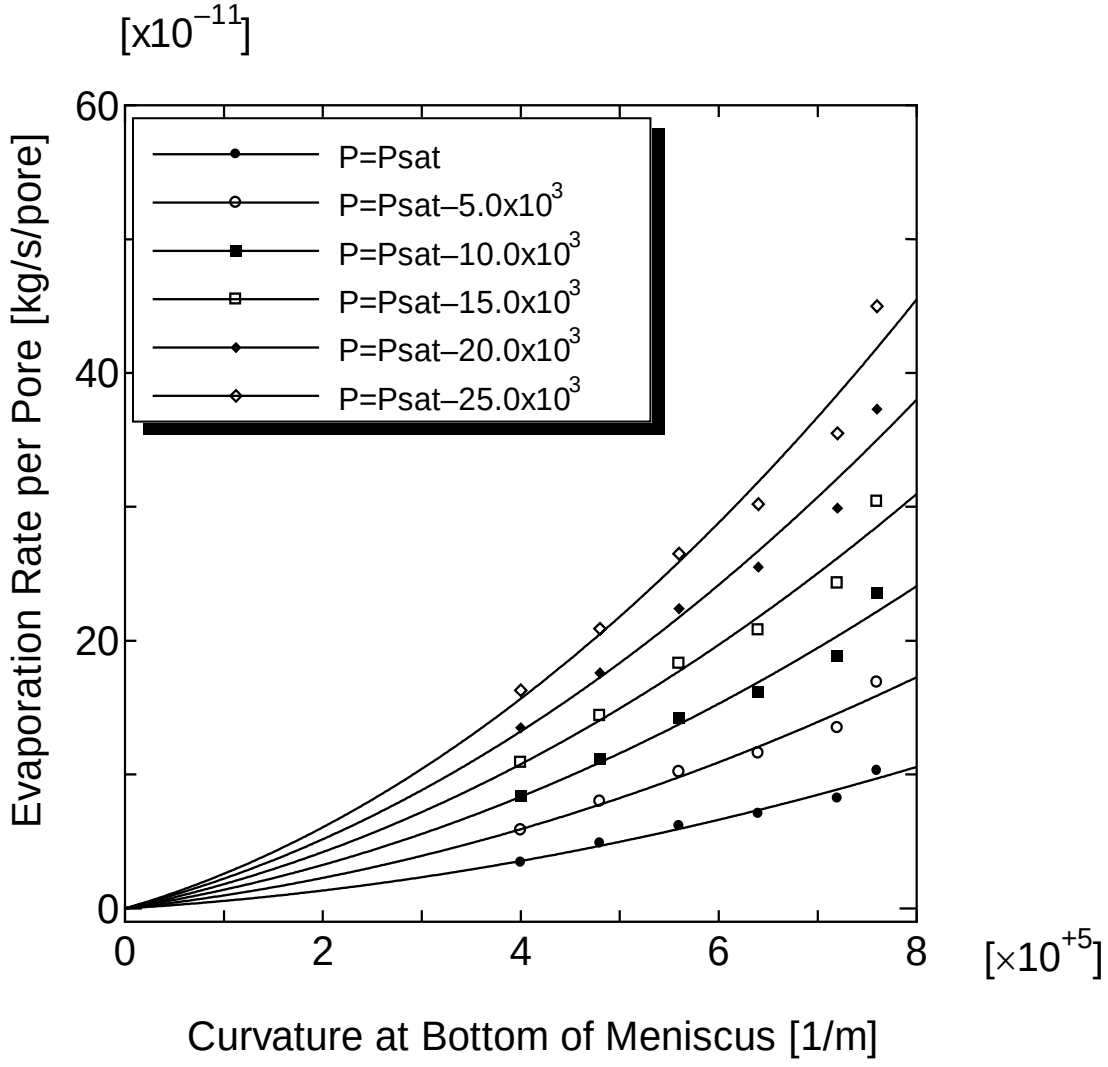

Fig. 4-32. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 413.15[K], Vapor Pressure: $P_{sat}(413.15K)$ to $P_{sat}(413.15K)-25\times10^{3}$ [Pa].

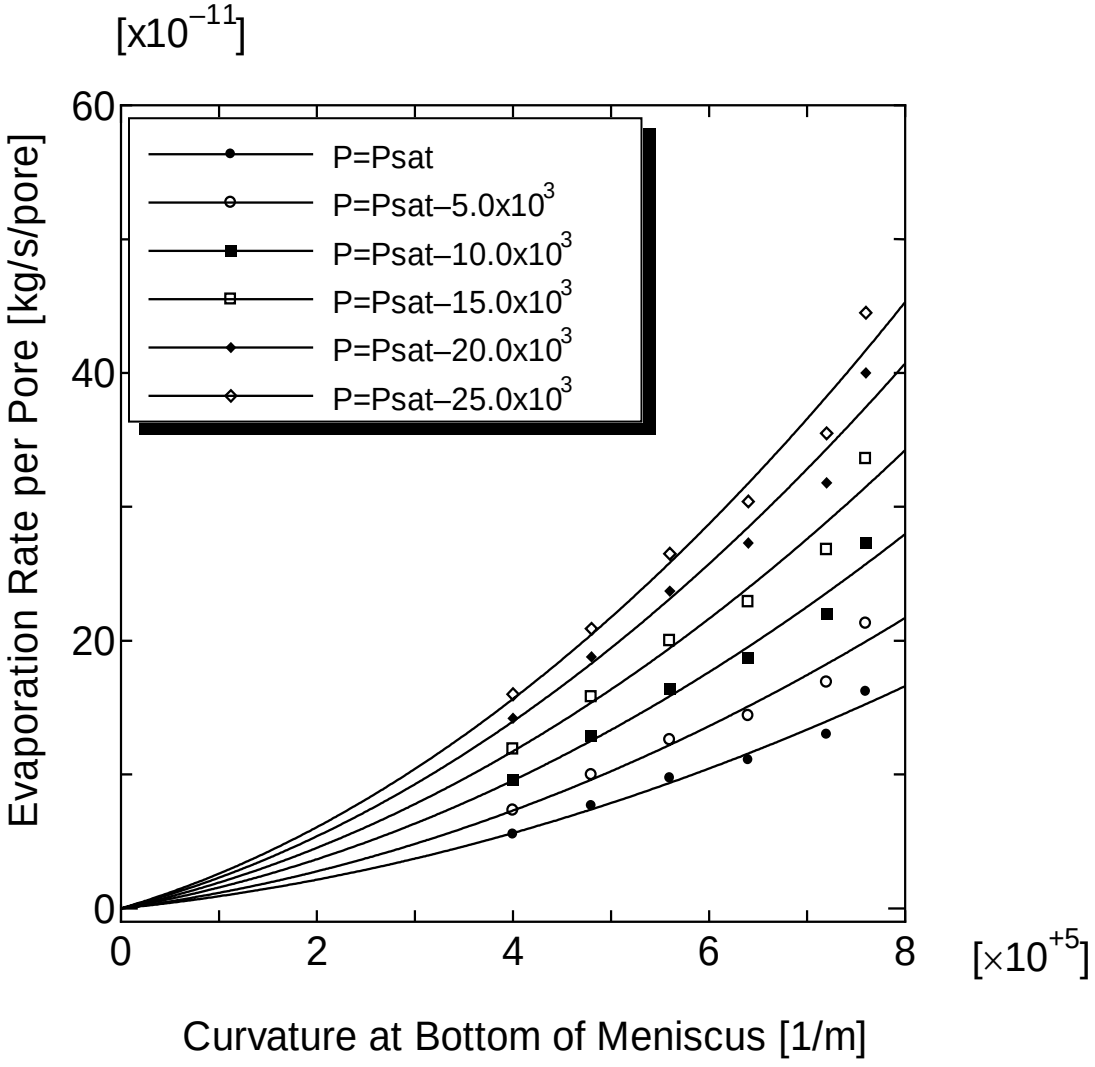

Fig. 4-33. Evaporation Rate per Pore: 5.0×10^{-6} [m], Wall Temperature: 423.15[K], Vapor Pressure: $P_{sat}(423.15K)$ to $P_{sat}(423.15K)-25\times10^{3}$ [Pa].

The total evaporation rates per pore for a given temperature and initial curvature at the pore diameter of 1.0×10^{-6} [m] are calculated and plotted through Fig. 4-34 to Fig. 4-42.

TABLE 4-3 Total Evaporation Rate in a Pore for a Given Pore Wall Temperaturefor Pore Diameter=1.0×10^{-6} [m]

Temperature(K)	Fig. #
353.15	Fig. 4-35
363.15	Fig. 4-36
373.15	Fig. 4-37
383.15	Fig. 4-38
393.15	Fig. 4-39
400.15	Fig. 4-40
413.15	Fig. 4-41
423.15	Fig. 4-42

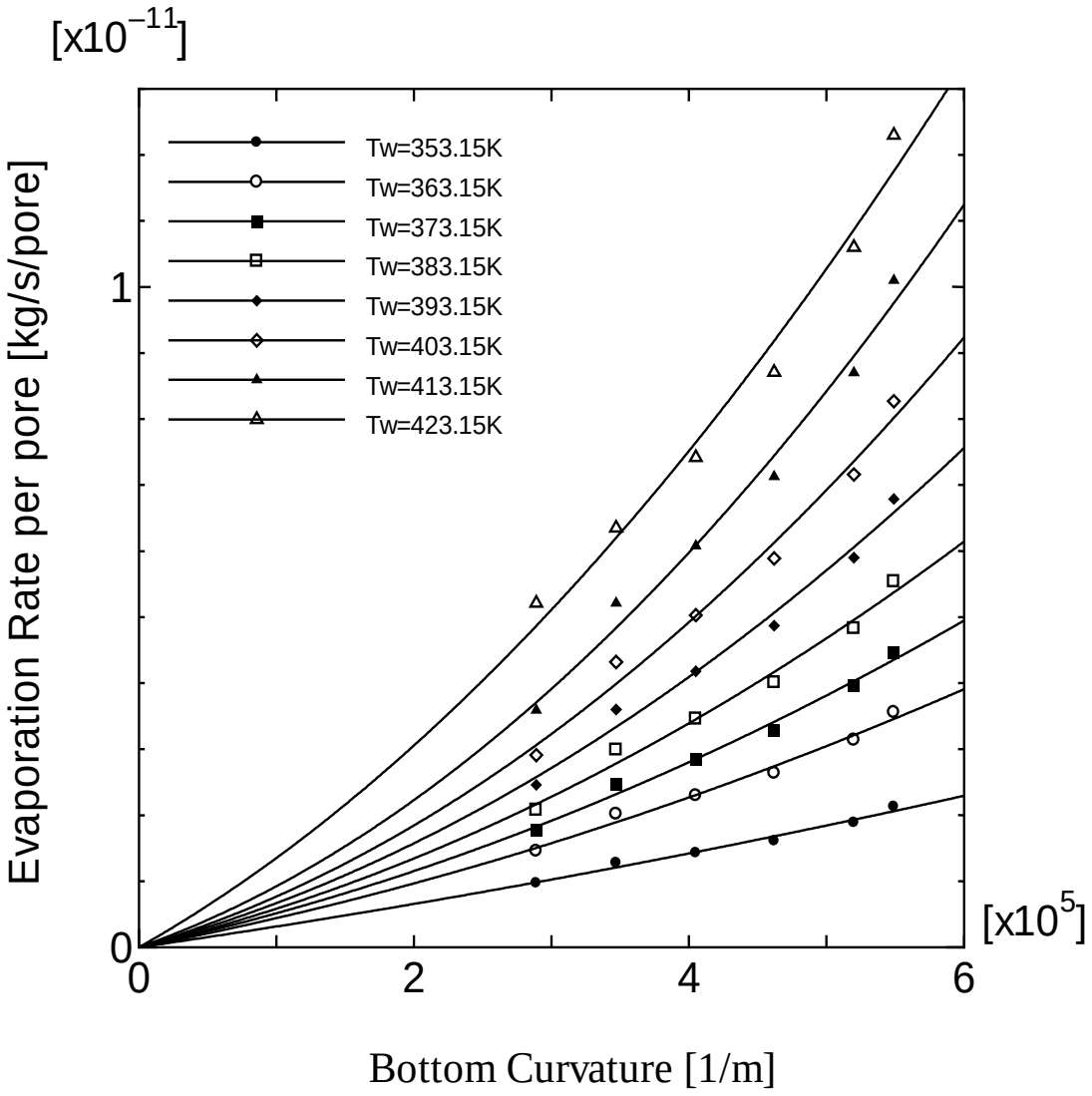

Fig. 4-34. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: $353.15-423.15$[K], Vapor Pressure: $P_{sat}(423.15K)$.

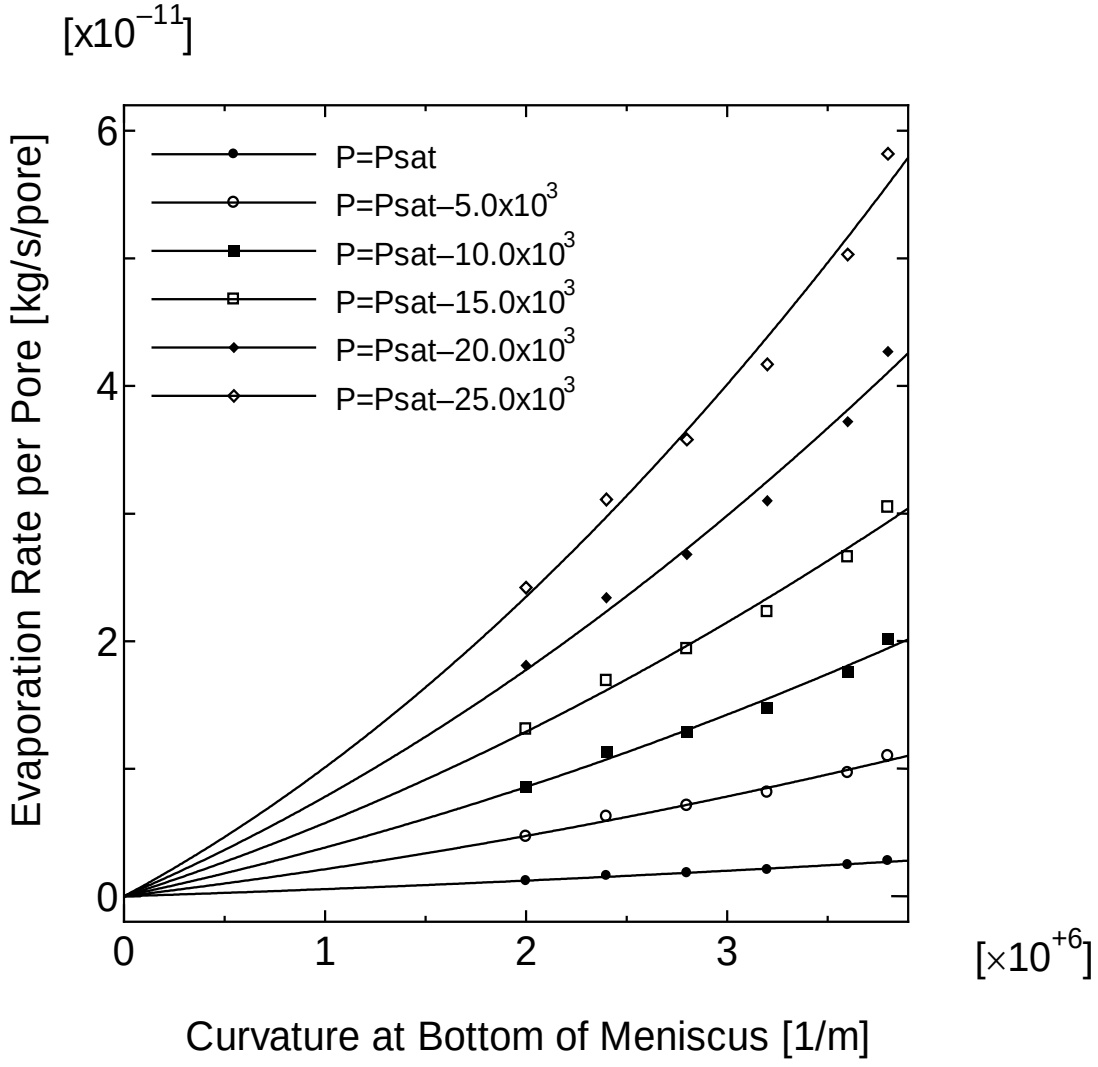

Fig. 4-35. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 353.15[K], Vapor Pressure: $P_{sat}(353.15K)$ to $P_{sat}(353.15K)-25\times10^{3}$ [Pa].

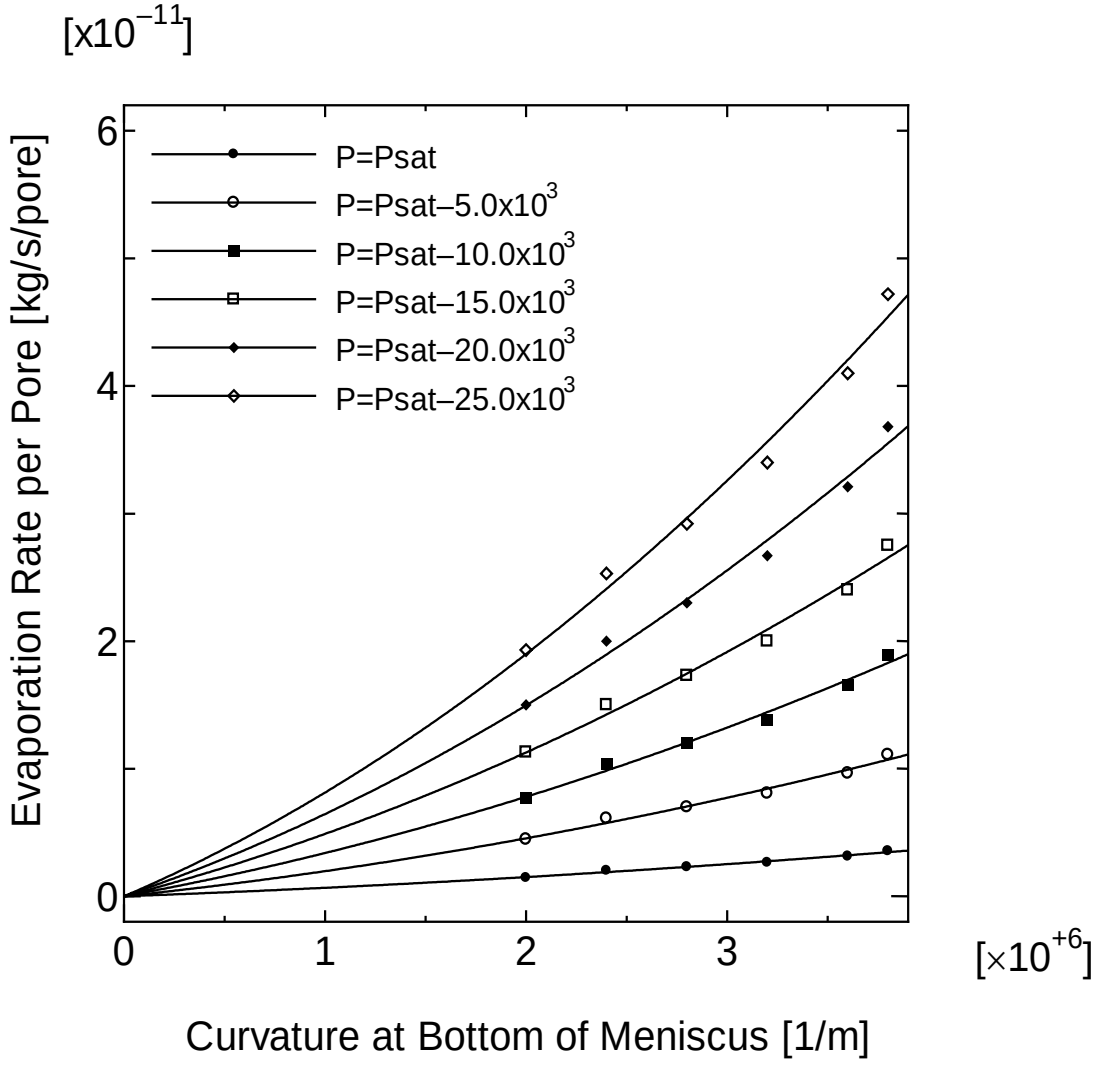

Fig. 4-36. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 363.15[K], Vapor Pressure: $P_{sat}(363.15K)$ to $P_{sat}(363.15K)-25\times10^{3}$ [Pa].

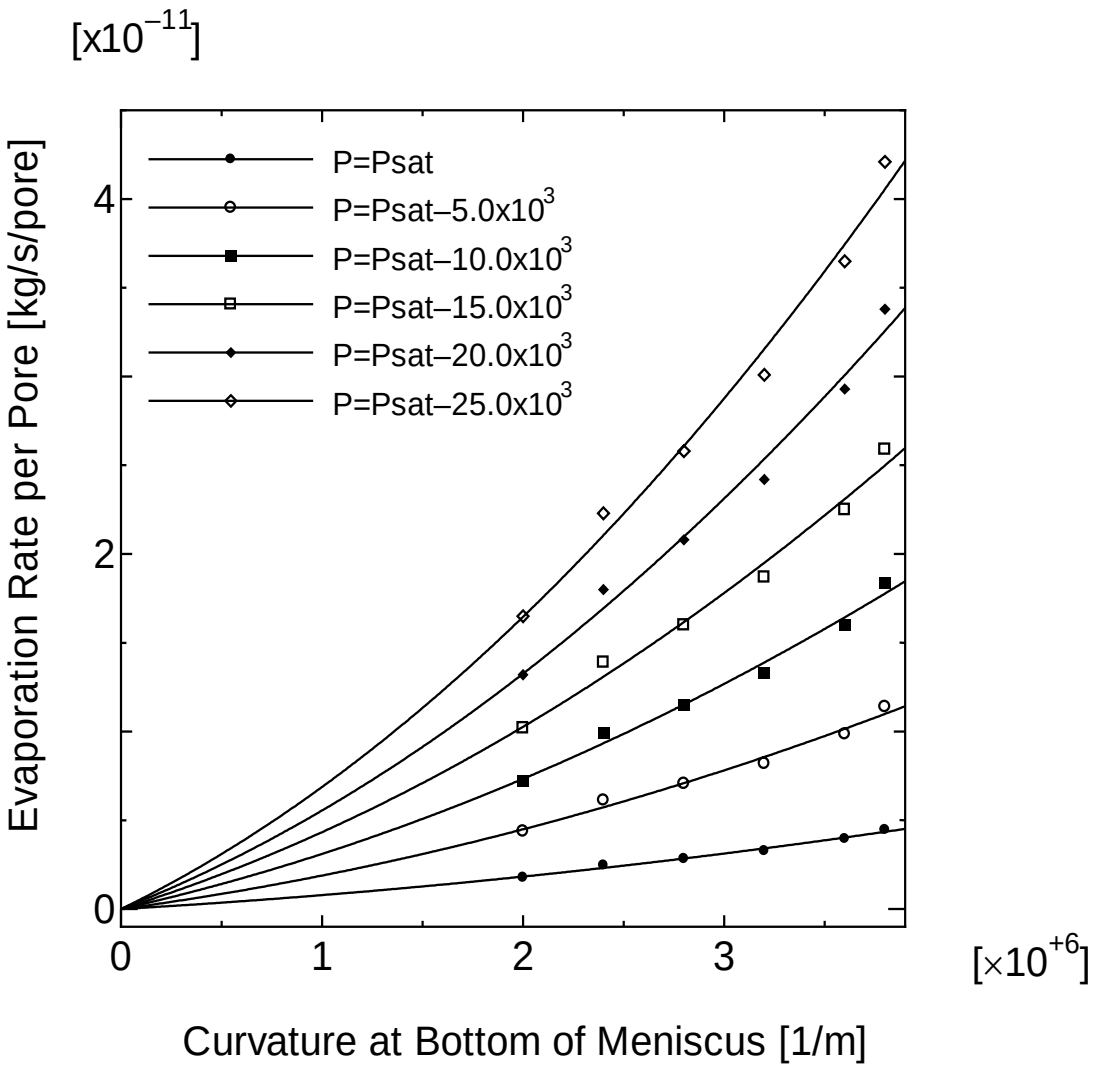

Fig. 4-37. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 373.15[K], Vapor Pressure: $P_{sat}(373.15K)$ to $P_{sat}(373.15K)-25\times10^{3}$ [Pa].

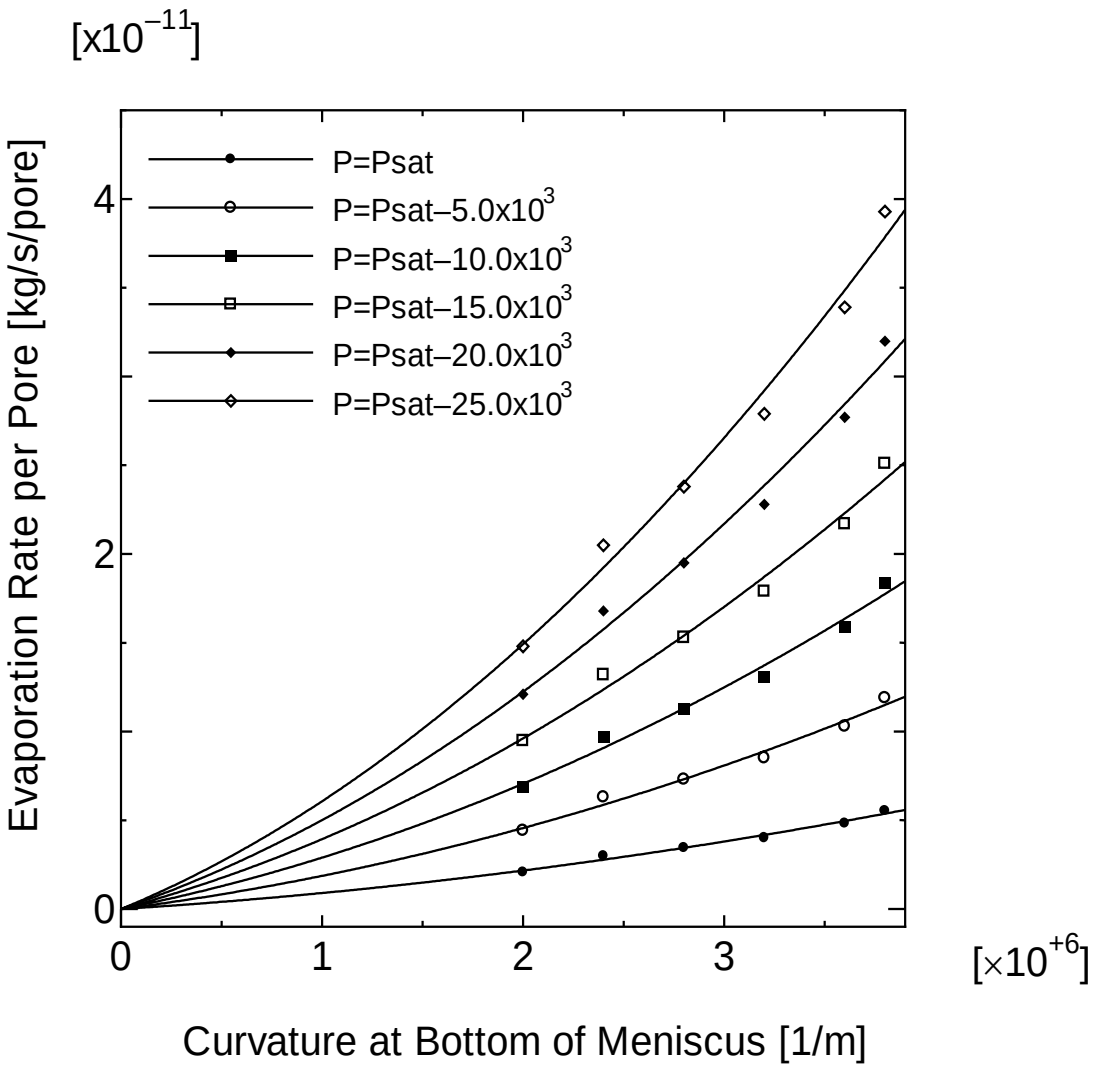

Fig. 4-38. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 383.15[K], Vapor Pressure: $P_{sat}(383.15K)$ to $P_{sat}(383.15K)-25\times10^{3}$ [Pa].

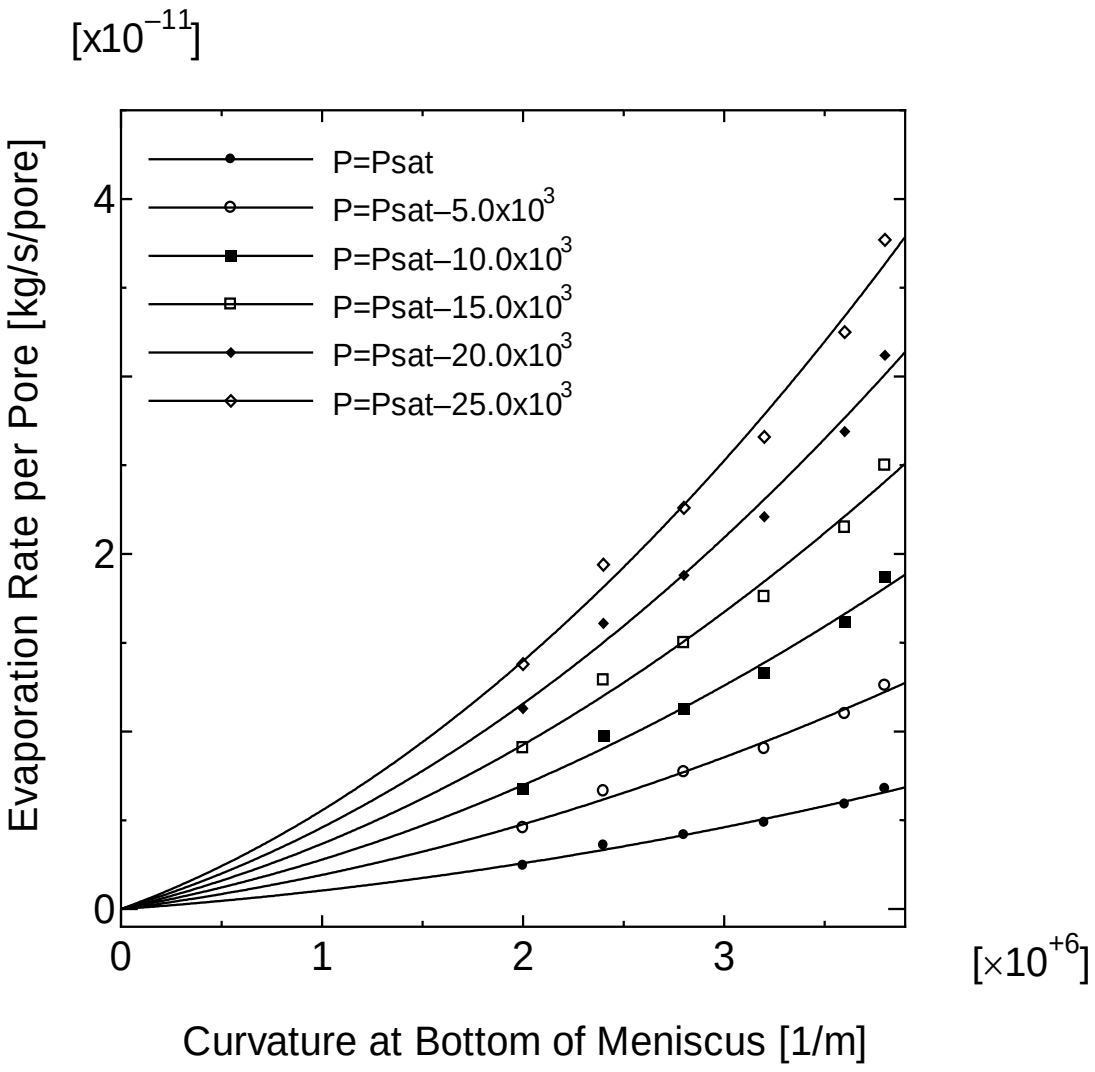

Fig. 4-39. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 393.15[K], Vapor Pressure: $P_{sat}(393.15K)$ to $P_{sat}(393.15K)-25\times10^{3}$ [Pa].

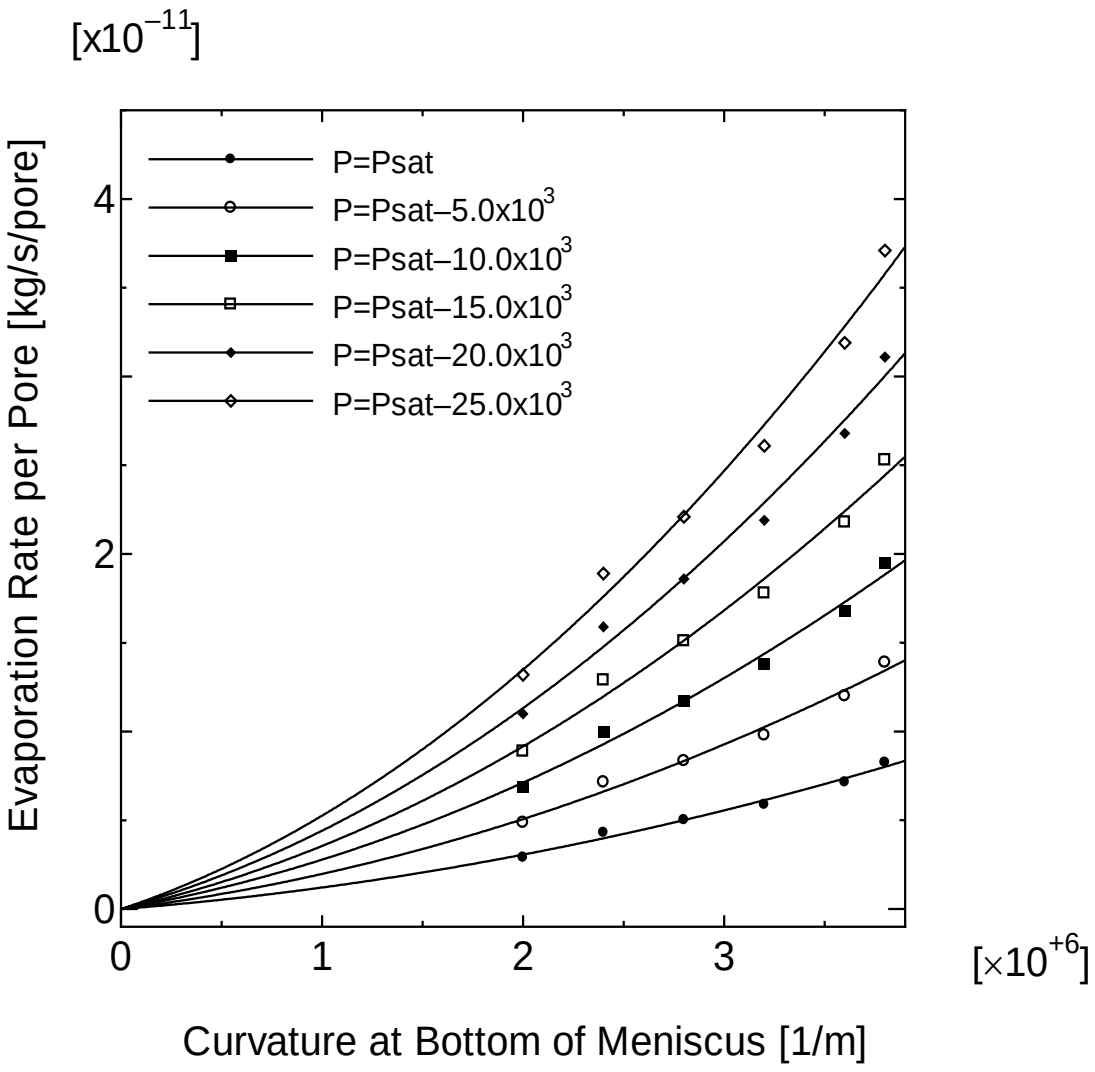

Fig. 4-40. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 403.15[K], Vapor Pressure: $P_{sat}(403.15K)$ to $P_{sat}(403.15K)-25\times10^{3}$ [Pa].

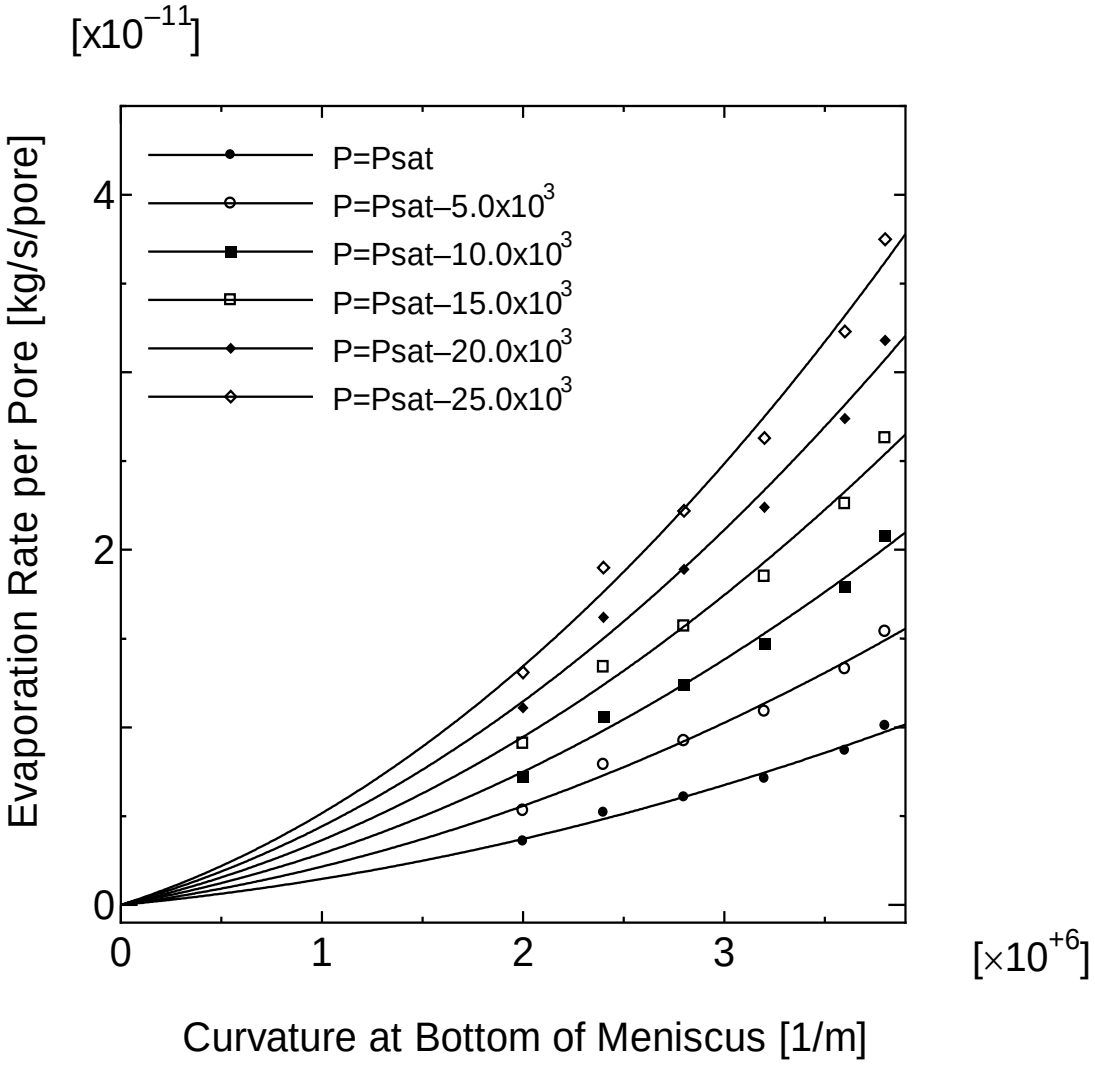

Fig. 4-41. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 413.15[K], Vapor Pressure: $P_{sat}(413.15K)$ to $P_{sat}(413.15K)-25\times10^{3}$ [Pa].

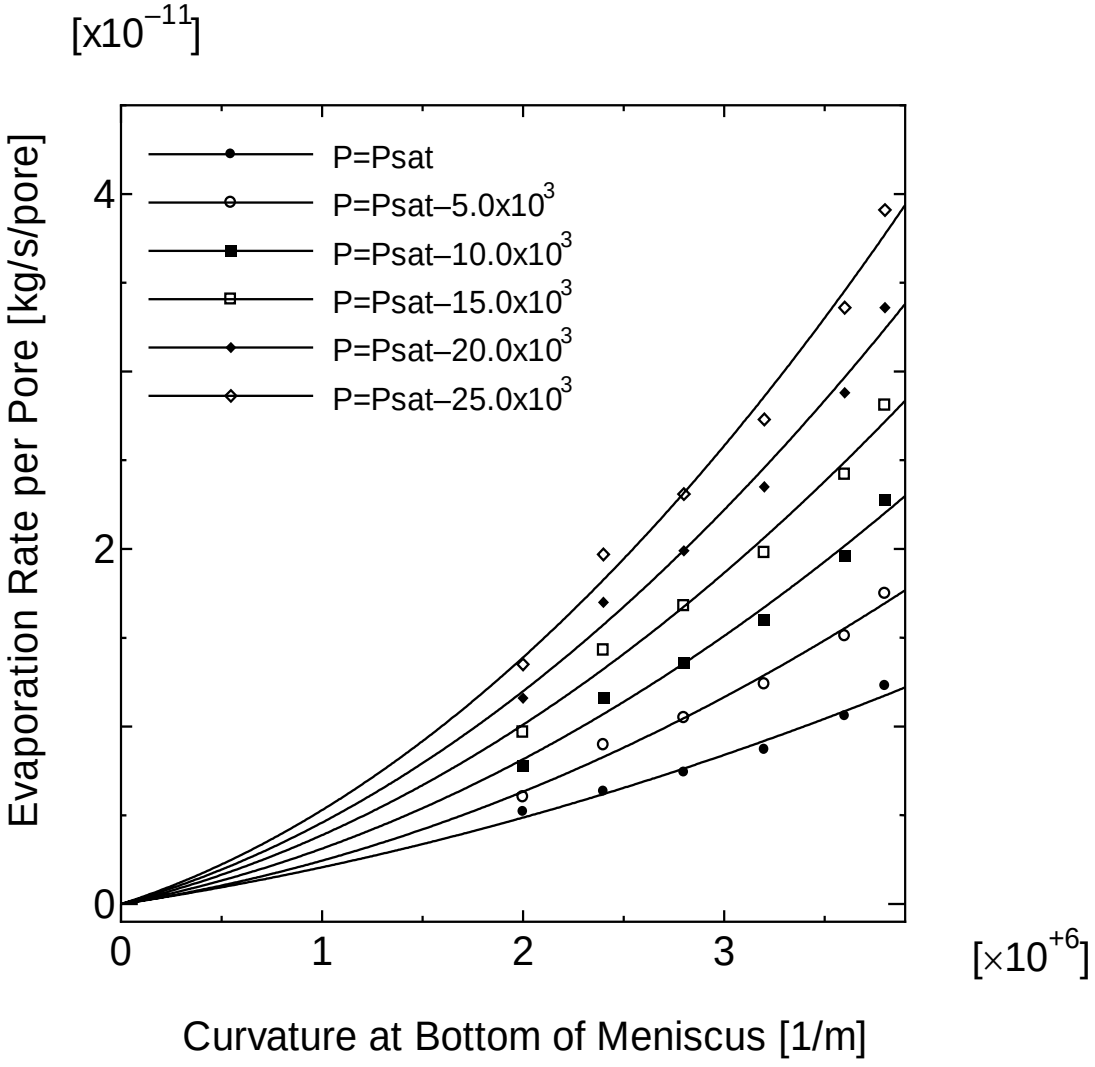

Fig. 4-42. Evaporation Rate per Pore: 1.0×10^{-6} [m], Wall Temperature: 423.15[K], Vapor Pressure: $P_{sat}(423.15K)$ to $P_{sat}(423.15K)-25\times10^{3}$ [Pa].

Application of Evaporation Model to Heat Pipe Performance Analysis

Coherent Porous Evaporator

Although heat pipes are very efficient heat transfer devices, they are subject to a number of heat transfer limitations. These limitations determine the maximum heat transfer rate. The type of limitation that restricts the operation of the heat pipe is determined by which limitation has the lowest value at a specific heat pipe working temperature. The possible limitations are shown below: 1. Continum flow limit, 2. Frozen startup limit, 3. Viscous limit, 4. Sonic limit, 5. Entrainment limit, 6. Capillary limit, 7. Condenser limit, 7. Boiling limit. The heat pipe performance limitations shown above express only the physical possibility and do not guarantee the achievable performance.

As an application of the evaporation model development, focus was on designing an optimized heat pipe to handle the high heat flux source. Past studies based on experience showed the heat removal capability of a heat pipe with a stochastic porous media could reach up to $100(W/cm^2)$ limited by the bubble growth which plugs the liquid flow in a porous media. On the other hand, the heat flux generated in recent electronic devices will be over $100(W/cm^2)$, which implies a design optimization by analytical or numerical methods is required to achieve this heat removal capability. To efficiently optimize the heat pipe design, some problems should be overcome. One is to avoid using stochastic porous media and another is to use a non-empirical evaporation model.

Since the porous material used for current heat pipes is usually stochastic in structure, bubble growth in the media ultimately hinders the performance of the heat pipe. In

addition, it is difficult to apply analytical or numerical methods for design optimization because of the stochastic nature of the material and models. A loop heat pipe with coherent pores integrated into the heat source is considered as a calculation model in this paper (Fig. 4-43). The advantage of this design is to have accurate micron-order diameter pores which will give large areas of evaporative thin film versus that in conventional heat pipes. This approach is based on the modeling idea that most evaporation occurs in the thin film area of a pore as explained in detail in Chapter II.

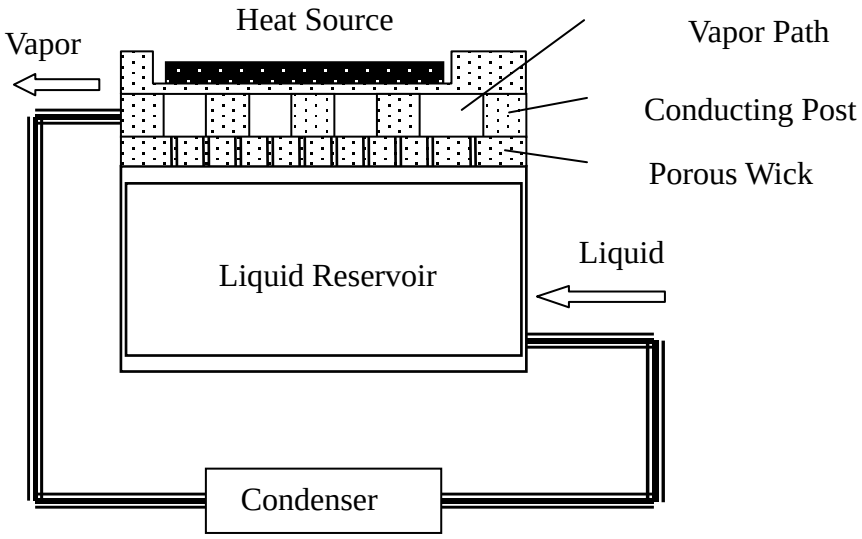

Fig. 4-43. Loop Heat Pipe with Coherent Porous Wick

Therefore, evaporators should be designed to optimize the thin film region. Also an additional benefit of the coherent pore structure offered is that it makes it easy to calculate pressure drop and evaporation rate in the wick compared with the evaluation of the heat pipe performance with the stochastic wick, which brings us confidence in operating limit

calculation, and potentially the ultra high capillary pressures without a corresponding pressure penalty such as entrainment of the liquid due to the fast vapor flow.

Definition of Geometric Parameters

Fig. 4-44 shows the top view of the coherent porous evaporator. The evaporator consists of a number of unit cells which have a unit length of w, pitch of P, and number of pores, n, pore diameter of d. The width of conducting posts connecting the heat source and the evaporator is b.

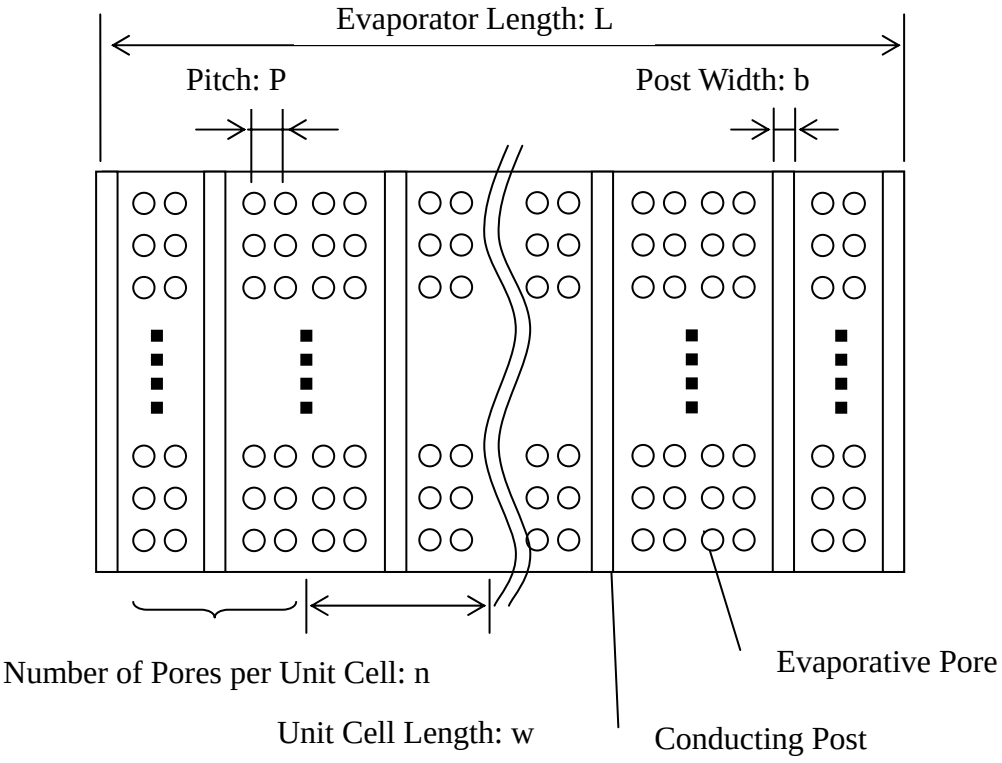

Fig. 4-44. Top View of Coherent Porous Evaporator

These parameters have relations:

$$w = b + nP$$
$$h = \frac{\sqrt{3}}{2} nP \tag{4-1}$$

To simplify the problem, the following parameters are defined

$$C_p \equiv P / d$$
$$C_b \equiv b / w \tag{4-2}$$

and parameters will be given again as

$$b = \frac{nC_p d}{(1/C_b - 1)}$$
$$h = \frac{\sqrt{3}}{2} nP = \frac{\sqrt{3}}{2} nC_p \tag{4-3}$$

In the calculation, n and d are fixed and C_b, C_p will be varied to change the geometry.

Estimation of Heat Pipe Performance

Fig. 4-45 shows the operating cycle of a Loop Heat Pipe. The heavy line is the saturation curve for the working fluid. Point 1 corresponds to the vapor condition just above the evaporating meniscus surface and Point 2 corresponds to the bulk vapor condition in the vapor path. Point 3 corresponds to the vapor pressure at the exit of the vapor path and the Point 4 corresponds to the vapor in the condenser. Point 5 corresponds to the liquid state in the condenser and Point 6 corresponds to the superheated liquid just below the meniscus interface. The heat pipe must satisfy the following pressure balance in order to operate.

$$P2 - P6 = \Delta P_{liq}^{wick} + \Delta P_{vap}^{wick} + \Delta P_{liq}^{line} + \Delta P_{vap}^{line} \tag{4-4}$$

$\Delta P_{liq}^{wick}, \Delta P_{vap}^{wick}$ are the liquid pressure and vapor pressure drops in the evaporator.

$\Delta P_{liq}^{line}, \Delta P_{vap}^{line}$ are the liquid and vapor pressure drops in the transport line. All of pressure drops are supposed to be a function of a heat flux generated by the heat source and geometric parameters such as w, b, n, P in Fig. 4-44. Since the evaporation rate in a pore can be determined by the pressure and temperature of liquid and vapor near the interface as described in Chapter III, the evaporation rate in a pore is a function of these values.

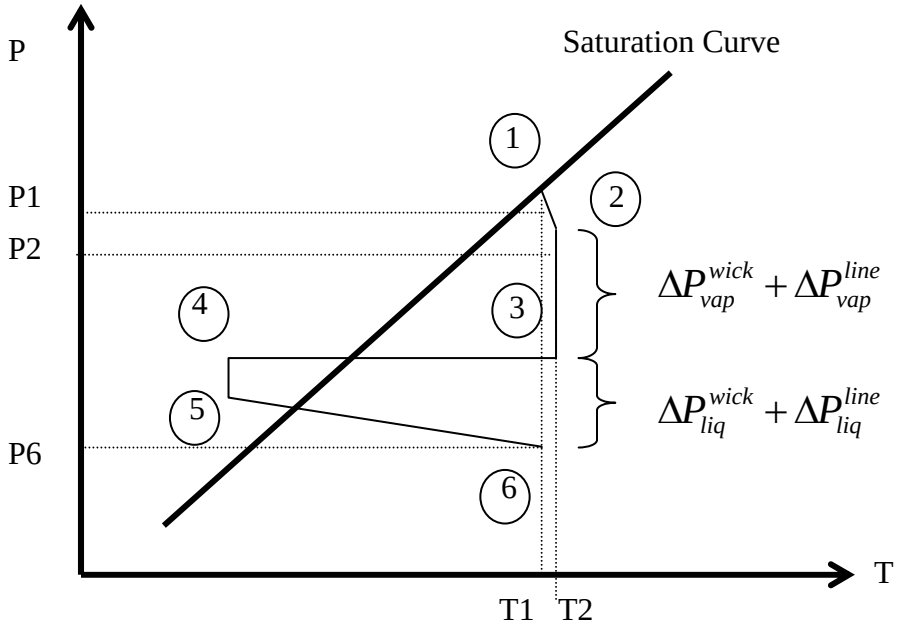

Fig. 4-45. Operating Cycle of Loop Heat Pipe

Define the effective evaporation rate per unit pore area, q''_{evap} gives the relationship:

$$q''_{source} A_{source} = q''_{evap}(P2, T2, P6, T6) N(w, b, n, P) A_{pore}^{eff}(d) \ . \qquad (4\text{-}5)$$

q''_{source} is heat flux density generated by a heat source. A_{source} and A_{pore} are the area of the heat source and a pore. N is the total number of pores in an evaporator and will be given as

$$N = \frac{L^2}{wP} n \qquad (4\text{-}6)$$

The area of a pore is a function of the diameter of the pore. The number of pores is a function of the geometry parameter of evaporator (Fig. 4-44), N varies with evaporator geometry. It will be assumed that the temperature increase(T2-T1) of the steam in the vapor path due to the heat transfer from the conducting wall or the heat source is small so that $T1 = T6 \approx T2$. We set T2=T6=constant. P6 is assumed to be equal to the saturation pressure at the temperature of condenser(T5). This is based on the fact that Point 5 could not be far from the saturation curve and the pressure drop from Point 5 to 6 is not large compared with the saturation pressure at T5. P2 will be obtained for a provided geometry to satisfy Eqs. (4-4) and (4-2). This chapter derives the way to calculate the effective evaporation rate per unit pore area(q''_{evap}) and the pressure drops(P2-P6) to solve Eqs. (4-4) and (4-5).

Limitation Due to Pressure Drop in System

The total pressure drop in the system must be lower than the available capillary pressure to make the heat pipe work. The total pressure drop in the system is given by the sum of the pressure drops of the liquid in the pore, the vapor in the exiting path over the evaporator, and the liquid and vapor transportation lines between the evaporator and the condenser.

The liquid is moved due to the capillary force in a pore from the bottom liquid reservoir to the top of the pore. To compensate for the liquid loss by evaporation from the liquid-vapor, the mass flow rate of the liquid in the pore must balance the evaporation rate. The liquid pressure drop of the capillary tube in the evaporator for laminar flow is

$$\Delta P_{liquid} = f \frac{t}{d} \rho_l \frac{v_p^2}{2} \tag{4-7}$$

$$\mathrm{Re} = \rho_l v_p \frac{d}{\mu} \tag{4-8}$$

The mass flow rate in a unit cell of the evaporator is given by the energy balance as

$$q''_{chip} Lw = h_{fg} \dot{m}_{cell} \tag{4-9}$$

The geometry of Fig. 4-44 shows that the number of pores in a unit cell is

$$n = \frac{L(w-b)}{P^2} \tag{4-10}$$

From the relationship between the mass flow rate per unit cell and pore, the mass flow in the pore is shown as

$$\dot{m}_{cell} = n\dot{m}_{pore} \tag{4-11}$$

$$\therefore \dot{m}_{pore} = \left(\frac{Lw - bL}{P^2} \right)^{-1} \frac{q''_{chip} LW}{h_{fg}} = \frac{P^2}{(w-b)} \frac{q''_{chip} Lw}{h_{fg}} \tag{4-12}$$

The velocity in the pore becomes

$$v_p = \frac{4\dot{m}_p}{\rho_l \pi d^2} = \frac{4P^2}{\rho_l \pi d^2 (w-b)} \frac{q''_{chip} w}{h_{fg}} \tag{4-13}$$

Reynolds's number is given by

$$\mathrm{Re} = \frac{\rho_l d}{\mu} \frac{4P^2}{\rho_l \pi d^2 (w-b)} \frac{q''_{chip} w}{h_{fg}} = \frac{4P^2 q''_{chip} w}{\mu_l \pi d (w-b) h_{fg}} \tag{4-14}$$

For laminar flow, the friction factor is

$$f = \frac{16}{\mathrm{Re}} \tag{4-15}$$

Substituting Eqs 4-7, 4-12, 4-13 and 4-14 into Eq. 4-6, the liquid pressure drop is

$$\Delta P_{liquid} = \frac{16}{\mathrm{Re}} \frac{t}{d} \rho_l \frac{v_p^2}{2} = \cdots = \frac{32\mu_l}{\rho_l h_{fg} \pi} \frac{P^2 wt}{d^4 (w-b)} q''_{chip} \tag{4-16}$$

Vapor Pressure Drop in Evaporator

As shown in Fig. 4-46 and Fig. 4-47, the cross sectional shape of the vapor path in the evaporator is triangle, which is due to constraints on the current lithographic technology. The flow rate is given as

$$\dot{m}_v = \rho_v V_v A = \rho_v V_v \frac{w-b}{2} h \tag{4-17}$$

Therefore, the vapor velocity is

$$V_v = \frac{\dot{m}_v}{\rho_v A} = \frac{\dot{m}_v}{\rho_v} \frac{2}{(w-b)h} = \frac{2}{\rho_v (w-b)h} \frac{q''_{chip} Lw}{h_{fg}} \tag{4-18}$$

It will be checked whether the vapor velocity exceeds the speed of sound or not when the maximum heat removal ability is determined. If it exceeds, the vapor velocity is set to the sound of speed and the maximum heat removal ability of the evaporator is calculated with it as an upper limit.

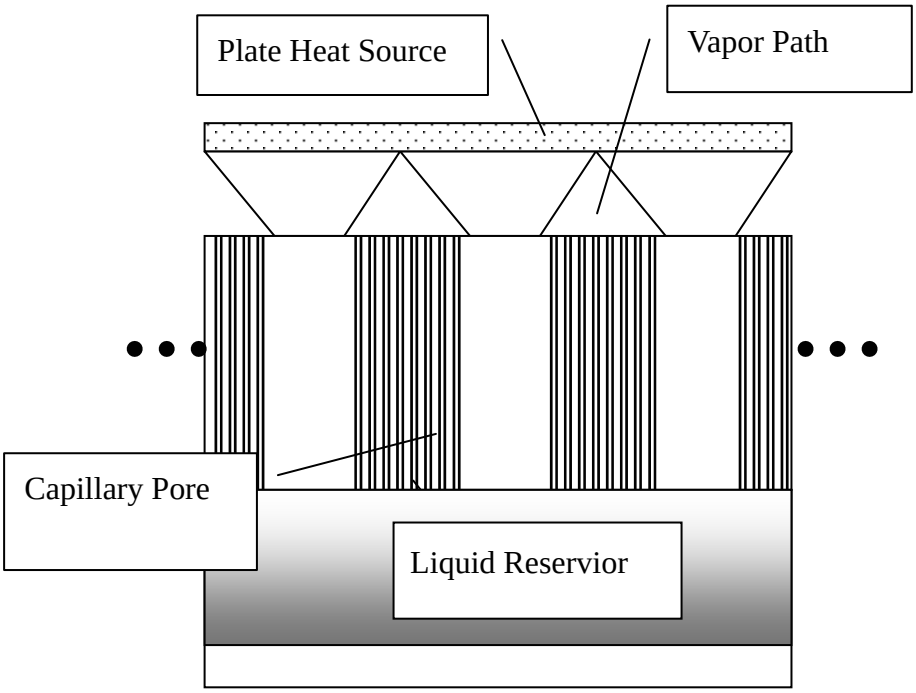

Fig. 4-46. Crossectional Shape of Evaporator

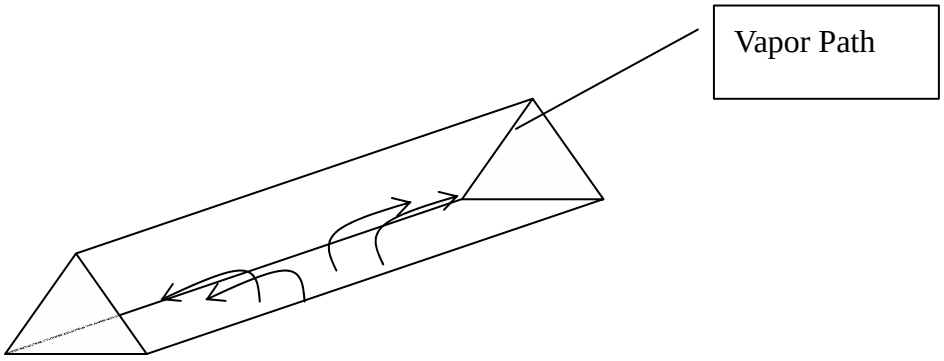

Fig. 4-47. Vapor Path Liquid Pressure Drop in Evaporator

The friction factor for the equilateral triangle (White, 1991)[49] is

$$C_f = \frac{13.333}{\mathrm{Re}_{Dh}} = \frac{13.333}{\rho V_v D_h / \mu_v} = \frac{13.333 \mu_v}{\rho_v D_h} \frac{\rho_v (w-b) h}{2} \frac{h_{fg}}{q''_{chip} L w} \tag{4-19}$$

where

$$D_h = \frac{a}{\sqrt{3}} = \frac{2}{3} h \tag{4-20}$$

Actually, vapor is generated along the evaporator and the pressure drop is dependent on the distribution of vapor generation. Since we want to know the performance limitation due to pressure drop and the sonic limitation, the maximum pressure drop should be considered to evaluate the heat pipe performance. As a simplification, all vapor is considered to be generated in the middle of the evaporator. The path length for this vapor is $L/2$ and the hydraulic diameter for the equilateral triangle is given as $D_h = a/\sqrt{3} = 2h/3$

By using Eqs. 4-17, 4-18 and 4-19, the vapor pressure drop is given as

$$\Delta P_{vapor} = C_f \frac{L/2}{D_h} \rho_v \frac{V_v^2}{2} = \ldots = \frac{30}{h^3} \frac{L^2 w^2}{\rho_v (w-b) h_{fg}} q''_{chip} \tag{4-21}$$

The Reynolds number is

$$\mathrm{Re} = \frac{\rho_V V_v D_h}{\mu_v} \tag{4-22}$$

Vapor and Liquid Pressure Drop in Transport Line

The vapor and liquid pressure drop in the transport line are given as pressure drop in a tube,

$$\Delta P_{vap}^{line} = \frac{32\mu_v l}{D_v^4 \rho_v \pi h_{fg}} q''_{chip} \tag{4-23}$$

$$\Delta P_{liq}^{line} = \frac{32\mu_l l}{D_l^4 \rho_l \pi h_{fg}} q''_{chip} \tag{4-24}$$

Relationship Between Capillary Pressure and Pressure Drop in the System

Since the total pressure drop in the system should be less than the capillary pressure, we will have a relationship as

$$\Delta P_{cap} > \Delta P_{liq}^{wick} + \Delta P_{vap}^{wick} + \Delta P_{liq}^{line} + \Delta P_{vap}^{line} \tag{4-25}$$

Substituting Eqs. 4-13, 4-18, 4-20 and 4-20 into Eq. 4-22 yields

$$\frac{2\sigma}{r} > \frac{32\mu_l P^2 wt}{\rho_l h_{fg} \pi d^4 (w-b)} q''_{chip} + \frac{30}{h^3} \frac{L^2 w^2}{\rho_v (w-b) h_{fg}} q''_{chip} + \frac{32\mu_v l}{D_v^4 \rho_v \pi h_{fg}} q''_{chip} + \frac{32\mu_l l}{D_l^4 \rho_l \pi h_{fg}} q''_{chip} \tag{4-26}$$

Therefore,

$$q''_{chip} < \frac{2\sigma}{r} \left[\frac{32\mu_l P^2 wt}{\rho_l h_{fg} \pi d^4 (w-b)} + \frac{30\mu_v L^2 w^2}{h^3 \rho_v (w-b) h_{fg}} + \frac{32\mu_v l}{D_v^4 \rho_v \pi h_{fg}} + \frac{32\mu_l l}{D_l^4 \rho_l \pi h_{fg}} \right]^{-1} \tag{4-27}$$

This equation gives a maximum heat flux of chip due to the capillary pumping limitation.

Temperature Difference Between Heat Source and Evaporator

Since the shape of vapor path is the right triangle, the height(h) can be determined if other parameters such as P, n, d, b are known. Therefore, the shape of the conducting post will be determined as well (Fig. 4-44 and Fig. 4-46). The areas at the top and the bottom are

given as

$$A_{top} = \left(b + \frac{2h}{\sqrt{3}} \right) L , \tag{4-28}$$

$$A_{bottom} = bL \tag{4-29}$$

Since the area of the post is proportional to position x, the area at x is given as

$$A(x) = \left(b + \frac{2}{\sqrt{3}} (h - x) \right) L \tag{4-30}$$

If a heat flux, q''_{source} is applied to the top part and all thermal energy goes through this post, the thermal energy flow at position x is given as

$$q''_{source} A_{top} = q''(x) A(x) \tag{4-31}$$

From Fick's law,

$$-k \frac{dT}{dx} A(x) = q''_{source} \left(b + \frac{2h}{\sqrt{3}} \right) L \tag{4-32}$$

Therefore,

$$\frac{dT}{dx} = -\frac{q''_{source} A_{top}}{kL \left(b + \frac{2}{\sqrt{3}} (h - x) \right)} \tag{4-33}$$

Integrating from x=0 to x=h gives,

$$\begin{aligned} &\int_{T(0)}^{T(h)} dT = -\int_0^h \frac{q''_{source} A_{top}}{kL \left(b + \frac{2}{\sqrt{3}} (h - x) \right)} dx \\ &\Leftrightarrow T(h) - T(0) = \frac{1}{2} \frac{q''_{source} A_{top}}{kL} \left\{ -\ln(b) - \ln(3) + \ln(3b + 2\sqrt{3}h) \right\} \end{aligned} \tag{4-34}$$

The temperature difference between heat source and evaporator due to this condition is given as

$$\Delta T_{wall} = 0.5 q''_{source} (b + \frac{2h}{\sqrt{3}}) \frac{\sqrt{3}}{k_{wall} L} \left\{ \ln \left(\frac{3b + 2\sqrt{3h}}{3b} \right) \right\}. \tag{4-35}$$

Optimizing Heat Pipe Design

The purpose of heat pipe performance analysis is to optimize a heat pipe design. If a heat pipe is designed to remove heat from the heat source on a satellite, the size of the heat pipe is a critical issue and needs to be small. If the integrated chip has a temperature limit, the heat source temperature needs to be lower. In this section, the heat pipe design is optimized for the heat source to be as low as possible.

Recalling Eqs. (4-4) and (4-5), Eq. (4-4) shows a pressure balance in the system and Eq. (4-5) shows the energy balance between input energy and evaporation.

$$P2 - P6 = \Delta P_{liq}^{wick} + \Delta P_{vap}^{wick} + \Delta P_{liq}^{line} + \Delta P_{vap}^{line} \quad (4\text{-}4)$$

$$q''_{source} A_{source} = q''_{evap}(P2, T2, P6, T6) N(w, b, n, P) A_{pore}^{eff}(d). \quad (4\text{-}5)$$

The energy loss to exterior is neglected.

Assumed that evaporator geometry is known and the dimension and the heat flux of the heat source are known. The unknown parameters are $P2, T2, P6, T6$. P2, T2 are vapor pressure and vapor temperature. P6 and T6 are liquid pressure and temperature in the vicinity of the liquid-vapor interface respectively. P2 and P6 can be related by the modified Young-Laplace equation.

$$P_v - P_l = P2 - P6 = \sigma K_{bottom} + \Pi. \quad (4\text{-}36)$$

If it is assumed the vapor temperature is nearly equal to the saturation pressure at T2, the vapor pressure, P2, is given as

$$P2 \approx P_{sat}(T2). \quad (4\text{-}37)$$

In addition, $T2$ can be considered to be close to $T1$ and $T6$. Therefore, K_{bottom} and T6

are considered as parameters to determine $q''_{evap}(P2,T6,P6,T6)$. The curvature at the bottom of meniscus can be also expressed by

$$K_{bottom} = \left(P2 - P6\right)/\sigma_{bottom} = (\Delta P_{liq}^{wick} + \Delta P_{vap}^{wick} + \Delta P_{liq}^{line} + \Delta P_{vap}^{line})/\sigma_{bottom}. \quad (4\text{-}38)$$

Therefore,

$$q''_{evap}(P2,T1,P6,T6) \Rightarrow q''_{evap}(T6,K_{bottom}) \quad (4\text{-}39)$$

The evaporation rate per pore can be obtained by

$$\dot{m}_{pore} = \frac{P^2}{(w-b)}\frac{q''_{chip}Lw}{h_{fg}} = \frac{\left(C_p d\right)^2}{\left(1-C_b\right)}\frac{q''_{chip}Lw}{h_{fg}}. \quad (4\text{-}40)$$

q''_{evap} can be converted to $\dot{m}_{pore}$ with the latent heat, h_{fg} by

$$q''_{evap} = \dot{m}_{pore} h_{fg}.$$

The heat source temperature can be calculated with Eq. (4-35) by

$$\Delta T_{wall} = 0.5 q''_{source}(b + \frac{2h}{\sqrt{3}})\frac{\sqrt{3}}{k_{wall}L}\left\{\ln\left(\frac{3b + 2\sqrt{3h}}{3b}\right)\right\}. \quad (4\text{-}35)$$

$$T_{source} = T_{wall}^{pore} + \Delta T_{wall} \quad (4\text{-}41)$$

If geometries of heat source, evaporator, other conditions are given as

$$\begin{aligned}
&q''_{source} = 100\,[W/cm^2] \\
&C_p = (P/d) = Arbitarity \\
&C_b = 0.01 \\
&n = 10, 20 \quad , \\
&L = 1.0\times10^{-2}\,m \\
&d = 5.0\times10^{-6}\,m \\
&D_l = D_v = 1.0\times10^{-2}\,m \\
&l = 1.0\times10^{-1}\,m
\end{aligned} \quad (4\text{-}42)$$

The bottom curvature and the evaporation rate as a function of Cp are tabulated in Table 4-4 for n=10 and Table 4-5 for n=20.

If these values are imposed on Fig. 4-25, Fig. 4-48 is obtained. The points inside the thick solid line satisfy the thermophysical condition of the loop heat pipe for given condition. The minimum bottom curvature is $1/R_c = 4.0\times10^5$ which is equal to the curvature at a pore wall and the maximum bottom curvature can not exceed $2/R_c = 8.0\times10^5$ due to the size of pore diameter. Cp can be only taken from 2.43 to 2.78 for n=10 and from 1.56 to 1.7 for n=20. The lowest wall temperatures which satisfy the operating condition are approximately 373.15K of the pore wall temperature and Cp=2.43 for n=10 and 353.15K of the wall temperature and Cp=1.4 for n=20. The heat source temperature can be calculated by Eq. (4-41) and the temperature difference between pore wall and heat source can be obtained from TABLE 4-4 for n=10 and TABLE 4-5 n=10. The heat source temperature for the optimized design is T_{source} =376.35K for n=10 and T_{source} =359.15K for n=20. Also, these results suggest that the lower temperature can be achieved when n=20, but the range of Cps are more narrow than for n=10.

TABLE 4-4 Bottom Curvature and Evaporation Rate per Pore (n=10)

Cp	K_{bottom} [1/m]	$\dot{m}_{pore}[kg/s]$	$\Delta T_{wall}[K]$	$\Delta P_{total}[Pa]$
2.10	1.24E+06	5.22E-11	2.45	8.77E+04
2.20	1.08E+06	5.73E-11	2.57	7.64E+04
2.30	9.48E+05	6.26E-11	2.69	6.70E+04
2.40	8.36E+05	6.82E-11	2.80	5.92E+04
2.50	7.42E+05	7.40E-11	2.92	5.25E+04
2.60	6.61E+05	8.00E-11	3.04	4.68E+04
2.70	5.92E+05	8.63E-11	3.15	4.19E+04
2.80	5.33E+05	9.28E-11	3.27	3.77E+04
2.90	4.81E+05	9.95E-11	3.39	3.40E+04
3.00	4.37E+05	1.07E-10	3.50	3.09E+04

Conditions are given in (4-42).

TABLE 4-5 Bottom Curvature and Evaporation Rate per Pore (n=20)

Cp	K_{bottom} [1/m]	$\dot{m}_{pore}[kg/s]$	$\Delta T_{wall}[K]$	$\Delta P_{total}[Pa]$
1.10	1.08E+06	1.43E-11	2.57	7.64E+04
1.20	8.36E+05	1.70E-11	2.80	5.91E+04
1.30	6.61E+05	2.00E-11	3.04	4.67E+04
1.40	5.32E+05	2.32E-11	3.27	3.76E+04
1.50	4.36E+05	2.66E-11	3.50	3.08E+04
1.60	3.62E+05	3.03E-11	3.74	2.56E+04
1.70	3.04E+05	3.42E-11	3.97	2.15E+04
1.80	2.59E+05	3.84E-11	4.20	1.83E+04
1.90	2.23E+05	4.27E-11	4.44	1.57E+04
2.00	1.93E+05	4.73E-11	4.67	1.37E+04

Conditions are given in (4-42).

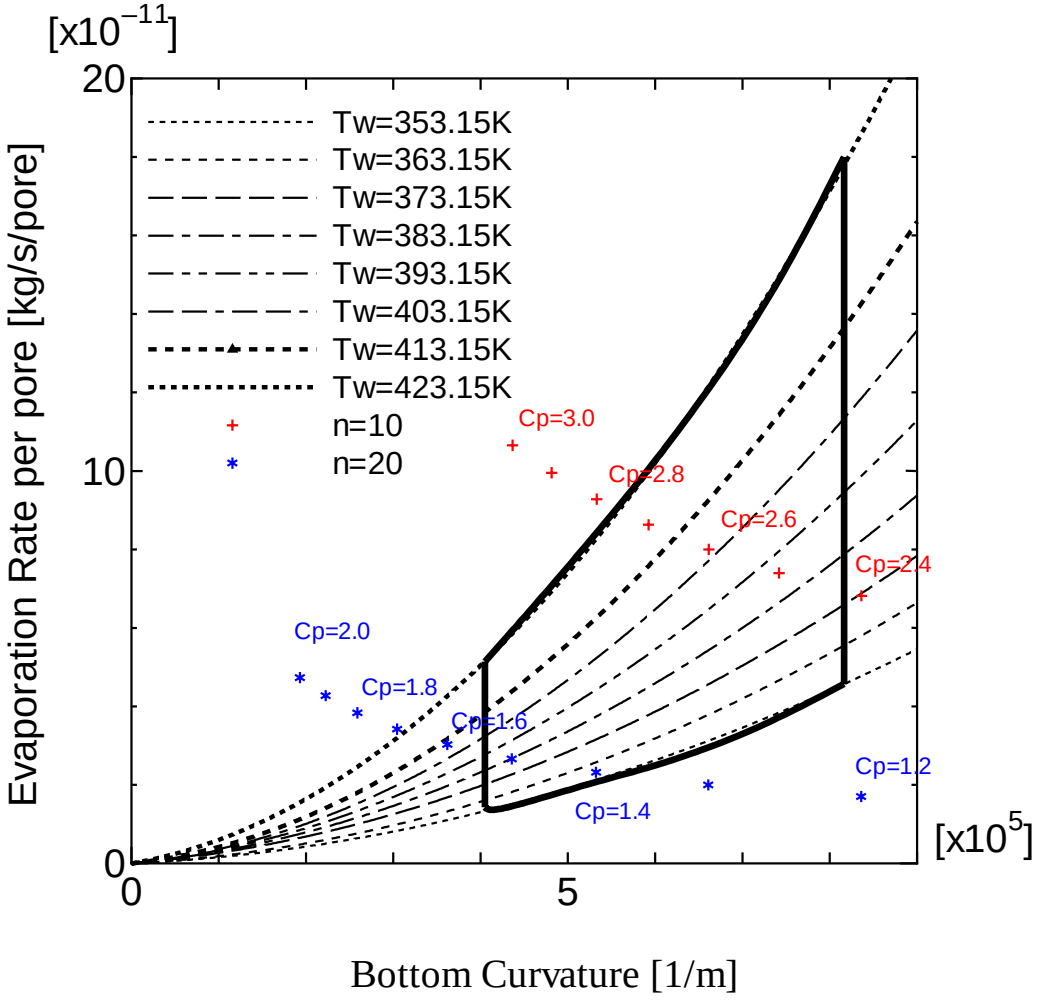

Fig. 4-48: Evaporation Rate vs. Curvature Satisfying Thermophysical Condition.

The given conditions are

$q''_{source} = 100\,[W/cm^2]$

$C_p = (P/d) = Arbitarity$

$C_b = 0.01$

$n = 10, 20$

$L = 1.0 \times 10^{-2}\,m$

$d = 5.0 \times 10^{-6}\,m$

$D_l = D_v = 1.0 \times 10^{-2}\,m$

$l = 1.0 \times 10^{-1}\,m$

CHAPTER V
CONCLUSION

The major object of this dissertation is to link a relationship between the evaporation model based on the statistical rate including vapor-liquid-structure intermolecular forces and engineering applications such as a heat pipe performance analysis. It is found that the evaporation rate in a pore can be determined from the pore wall temperature, vapor pressure and the bottom curvature of meniscus. By finding these dependent parameters, the general evaporation model is successfully related to engineering application.

Another interesting important finding is that the higher curvature at the bottom of meniscus gives the larger evaporation rate per pore. This means the large evaporation rate, thus the high heat pipe performance, requires the high pressure drop since the curvature is proportional to the total pressure drop in the system. As seen in Fig. 4-48, the lower heat source temperature is achieved at the higher total pressure drop in a system for a given heat flux source. This result contradicts to the conventional heat pipe design which makes effort to reduce the total pressure drop in the system. The coherent pore structure approach de-couples the pore pumping power from the wick parametric pressure drop.

In this dissertation, the wall temperature of the pore and the vapor pressure above the liquid interface are assumed to be constant anywhere in the evaporator. This condition simplifies the evaporation model in the pore and the pore evaporation rate is independent

of global location in the wick. In practice, this is not true because vapor pressure decreases along the vapor channel in the heat pipe evaporator due to friction losses. If the vapor pressure above the liquid interface changes, the evaporation rate will be affected. Another important simplification is that at the liquid interface there is no friction loss. Thus the marangoni effect due the interfacial liquid temperature change along the interface is neglected. Since the liquid interfacial temperature obtained in chapar IV is not constant in any case and has sometimes steep change, the marangoni effect may not be negligible at the interface boundary. Study of this effect is left for the future work.

Our group has a research collaboration with University of Cincinnati which is responsible for a fabrication of evaporator part by lithographical method and Therma Core, Inc. which will construct the entire system of loop heat pipe. The design suggested in this dissertation was provided to University of Cincinnati and they have started to fabricate the evaporator. The following pictures are the proto type of the evaporator provided by University of Cincinnati.

Fig. 5-1 Prototype of Evaporator Lithographically Etched by University of Cincinnati

REFERENCES

1) R. A. Wirtz, and R. Sohal, H. Wang, “Thermal Performance of Pin-Fin-Fan-Sink Assembles”, *HTD* **319**, 67-75 (1995).

2) F.J. Kelecy, S. Ramadhyani, and F. P. Incropera, “Effect of Shrouded Pin Fins on Forced Convection Cooling of Discrete Heat Sources by Direct Liquid Immersion”, *Proc. 2nd ASME-JSME Thermal Eng. Joint Conf.* **3**, 387-394, August (1987).

3) T. P. Cotter, “Principles and Prospects for Micro Heat Pipes”, *Proc. 5 th Int. Heat Pipe Conf.*, Tukuba, Japan, 328-335, May 14-17 (1984).

4) B. R. Babin, G. P. Peterson, and D. Wu, “Analysis and Testing of a Micro Heat Pipe During Steady-State Operation”, *Proc. ASME/AIChE National Heat Transfer Conf.* **89-HT-17,** Philadelphia, August (1989).

5) D. Wu, and G. P. Peterson, "Experimental Investigation of the Transient Behavior of Micro Heat Pipes", *Proc. AIAA/ASME 5th Joint Thermophysics and Heat Transfer Conference*, Seattle WA, June 18-20 (1990).

6) D. Plesch, W. Bier, D. Seidel and K. Schubert, “Miniature Heat Pipes for Heat Removal from Microelectronic Circuits”, *Proc. ASME Annual Meeting*, Atlanta, GA, Dec. 1-6 (1991).

7) J. Zhou, Z. Yao, and J. Zhu, “Experimental Investigation of the Application Characters of Micro Heat Pipe”, *Proc. 8th Int. Heat Pipe Conf.*, 421-424, Beijing, China, September (1992).

8) T. Li, L. Cao, and L. Xiang, “Research and Application for the Heat Transfer Performance of Small Heat Pipes”, *Proc. 8th Int. Heat Pipe Conf.*, 413-415, Beijing, China, September (1992).

9) D. Khrustalev, and A. Faghri, “Thermal Analysis of a Micro Heat Pipe”, *J. Heat Transfer* **116** (1), 189-198 (1994).

10) B. R. Babin, G. P. Peterson, and D. Wu, "Steady State Modeling and Testing of a Micro Heat Pipe," *J. of Heat Transfer* **112** vol. 3, 595-601 (1990).

11) F. M. Gerner, J. P. Longtin; H. T. Henderson; W. M. Hsieh; P. Ramadas; and W. S. Chang, “Flow and Heat Transfer Limitations in Micro Heat Pipes”, *HTD* **206-3**, 99-104 (1992).

12) T. Kaya, and T. T. Hoang, “Mathematical Modeling of Loop Heat Pipe”, *AIAA Paper* **99-0477** (1999).

13) M. Potash, Jr., and P.C. Wayner, Jr., “Evaporation from a Two-Dimensional Extened Meniscus”, *Int. J. Heat. Mass Transfer* **15**, 1851-1863 (1972).

14) Mirzamoghadam, and I. Catton, “A Physical Model of the Evaporating Meniscus”, *J. Heat Transfer* **110**, 201-207 (1988).

15) X. Xu, and V. P. Carey, “Film Evaporation from a Micro-grooved Surface and an Approximate Heat Transfer Model and its Comparison with Experimental Data”, *J. Thermophysics and Heat Transfer* **4(4)**, 512-520 (1990).

16) L.W. Swanson, and G. P. Peterson, “Evaporating Extended Meniscus in a V-shaped Channel”, *J. of Thermophysics and Heat Transfer* **8(1)**, 172-180 (1994).

17) R. Richter, and M. Gottschlich, “Thermodynamic Aspects of Heat Pipe Operation”, *J. Thermophysics and Heat Transfer* **8(2)**, 334-340 (1994).

18) H. Khalkhali, A. Faghri, and Z. J. Zuo, “Entropy Generation in a Heat Pipe System”, *Applied Thermal Engineering* **19**, 1027-1043 (1999).

19) B. V. Derjaguin, S. V. Nerpin; and N. V. Churaev, “Effect of Flm Transfer Upon Evaporation of Liquids From Capillaries”, *Bulletin Rilem* **29**, 93-98 (1965).

20) P. C. Wayner, Jr., Y. K. Kao, and L. V. LaCroix, “The Interline Heat Transfer Coefficient of an Evaporating Wetting Film”, *Int. J. Heat Mass Transfer* **19**, 487-492 (1976).

21) S. J. Moosman, and G. M. Homsy, “Evaporating Menisci of Wetting Fluids”, *J. Colloid and Interface Sci.* **73**, 212 (1980).

22) P. C. Wayner, Jr., and J. Schonberg, “Heat Transfer and Fluid Flow in an Evaporating Extended Meniscus”, *Proc. Int. Heat Transfer Conference*, Jerusalem, Israel, 229-234, August 19-24 (1990).

23) P. Stephan, and J. Hammer, “A New Model for Nucleate Boiling Heat Transfer”, *Wärme-und Stoffübertsagung* **30**, 119-125 (1994).

24) L. W. Swanson, and G. C. Herdt, “Model of the Evaporating Meniscus in a Capillary Tube”, *J. Heat Transfer* **114**, 434-441 (1992).

25) H. C. Chebaro, K. P. Hallinan, and S. J. Kim, “Evaporation form a Porous Wick Heat Pipe for Isothermal Interfacial Condition”, *HTD* **221**, 23-28 (1992).

26) M. Sujanani, and P. C. Wayner Jr., “Transport Process and Interfacial Phenomena in an Evaporating Meniscus”, *Chem. Eng. Comm.* **118**, 89-110 (1992).

27) S. DasGupta, J. A. Schonberg, and P. C. Wayner Jr., “Investigation of an Evaporating Extended Meniscus Based on the Augmented Young-Laplace Equation”, *J. Heat Transfer* **115**, 201-208 (1993).

28) S. DasGupta, I. Y. Kim, and P. C. Wayner Jr., “Use of the Kelvin-Clapeyron Equation to Model an Evaporating Curved Microfilm”, *J Heat Transfer* **116**, 1007-1015 (1994).

29) D. Wu, and G. P. Peterson, “Investigation of the Transient Characteristics of a Micro Heat Pipe”, *J. Thermophys.* **5**, 129 (1991).

30) D. Khrustalev, and A. Faghri, “Thermal Analysis of a Micro Heat Pipe”, *HTD* **236**, 19 (1993).

31) J. M. Ha, and G. P. Peterson, “Capillary Performance of Evaporating Flow in Micro Grooves: An Analytical Approach for Very Small Tilt”, *J. of Heat Transfer* **120**, 452 (1998).

32) J. A. Schonberg, S. DasGupta, and P. C. Wayner Jr., “An Augmented Young-Laplace Model of an Evaporating Meniscus in a Microchannel with High Heat Flux”, *Experimental Thermal and Fluid Science* **10**, 163-170 (1995).

33) L. W. Swanson, and G. P. Peterson, “The Interfacial Thermodynamics of Micro Heat Pipes”, *J. Heat Transfer* **117**, 195-201 (1995).

34) R. W. Schrage, *A Theoretical Study of Interphase Mass Transfer*, Columbia University Press, New York (1953).

35) P. C. Wayner Jr., “The Effect of Interfacial Mass Transport on Flow in Thin Liquid Films”, *Colloids and Surfaces* **52**, 71-84 (1991).

36) R. Marek, and J. Straub, “Analysis of the Evaporation Coefficient and the Condensation Coefficient of Water.”, *Int. J. Heat Mass Transfer* **44**, 39-53 (2001).

37) Y. Cao, and A. Faghri, “Simulation of the Early Startup Period of High-Temperature Heat Pipes from the Frozen State by Rarefied Vapor Self-Diffusion Model”, *J. Heat Transfer* **115**, 239-245 (1993).

38) G. Fang, and C. A. Ward, “Temperature Measured Close to the Interface of an Evaporating Liquid”, *Physical Review E* **59**, 417-428 (1999).

39) G. Fang, and C. A. Ward, “Examination of the Statistical Rate Theory Expression for Liquid Evaporation Rates”, *Physical Review E* **59**, 441-453 (1999).

40) K. Aoki, and C. Cercignani, “Evaporation and Condensation on Two Parallel Plates at Finite Reynolds Numbers”, *Phys Fluids* **26**, 1163-1164 (1983).

41) C. A. Ward, and G. Fang, “Expression for Predicting Liquid Evaporation Flux: Statistical Rate Theory Approach”, *Physical Review E* **59**, 429-440 (1999).

42) E. M. Lifshitz, “The Theory of Molecular Attractive Forces between Solids”, *Soviet Physics* **2(1)**, 73-83 (1956).

43) V. P. Carey, *Statistical Thermodynamics and Microscale Thermophysics*, Cambridge University Press, New York (1999).

44) J. Pfahler, J. Harley, and H. Bau, “Liquid Transport in Micro and Submicron Channels”, *Sensors and Actuators* **A21-A23**, 431-434 (1990).

45) J.N. Israelachvili, “Measurement of the Viscosity of Liquids in Very Thin Films”, *J.Coll. Interface Sci.* **110**, 263-271 (1986).

46) P. A. Thompson, and S. M. Toian, “A General Boundary Condition for Liquid Flow at Solid Surface”, *Phy. Rev. Lett.* **63**, 766-769 (1997).

47) J. P. Burelbach, S. G. Bankoff, and S. H. Davis, “Nonlinear Stability of Evaporationg/Condensing Liquid Films”, *J. Fluid Mech.* **195**, 463-494 (1988).

48) Gray, Modern Differential Geometry of Curves and Surfaces with Mathematica, CRC Press, New York (1999).

49) F. M. White, *Viscous Fluid Flow*, McGraw-Hill, Inc., New York (1991).

50) *ASME Steam Properties for Industrial Use*, American Society of Mechanical Engineers, New York (1998).

APPENDIX

PROGRAM TO OBTAIN THE LOCAL AND TOTAL EVAPORATION RATE IN A PORE

The program consists of three parts: (1) Main Program, (2) Functions and (3) Subroutines to calculate water properties. The main program and the functions are shown in this appendix. The subroutine to calculate water properties by the International Association for the Properties of Water and Steam(IAPWS-IF97)[50] is used.

Main Program and Functions

```
program main
!
!           Constants
!
            double precision, parameter :: d=5.00d-6        ! Pore Diameter
            double precision, parameter :: Rc=d/2.0d0       ! Pore Radius
            double precision, parameter :: ck=1.0d-10       !Coefficient Normalization
            double precision, parameter :: cRc=Rc/ck        ! Normalized Pore Radius
            double precision, parameter :: pi=3.141592654d0         ! PI
            double precision, parameter :: dx=5.0d-11       ! Spacing between nodes
            integer, parameter:: n_max=int(Rc/dx)           ! Maximum Node Number
!
            integer ::ircnt
            character(Len=60):: cright
            character(len=40):: title1,title2
            character(len=10):: version
            character(len=4):: year
!
!           Pressure
!
            double precision :: Psat_amb            ! Psat_amb = Saturation Pressure
            double precision :: P5                  ! Pressure at point 5
            double precision :: Psat_1              ! Pressure at point 1
            double precision :: Pv                  ! Vapor Pressure
```

```
!
!       Pressure Drop
!
        double precision :: dPl                 ! Total Pressure Drop in Liquid
        double precision :: dPl_tran ! Liquid Pressure Drop in Transport Line
        double precision :: dPl_pore ! Liquid Pressure Drop in Pore
        double precision :: dPv_tran ! Vapor Pressure Drop in Transport Line
        double precision :: dPv_evap            ! Vapor Pressure Drop in Evaporator
        double precision :: dP                  ! Total Pressure Drop in System
!
!       Temperature
!
        double precision :: Tamb                ! Tamb = Ambient Temperature
        double precision :: Tc                  ! Temp (Celcius)
        double precision :: Tf                  ! Temp (Farenheit)
        double precision :: Tv                  ! Vapor Temperature
        double precision :: Tl                  ! Liquid Temperature
        double precision :: Tw                  ! Wall Temperature
!
!       Energy
!
        double precision :: q_source            ! Heat Flux of Source
!
!       Property
!
        double precision :: sigma               ! Surface Tension
        double precision :: mu_l                ! Liquid Viscosity
        double precision :: mu_v                ! Vapor Viscosity
        double precision :: rho_l               ! Liquid Density
        double precision :: vl                  ! Liquid Specific Volume
        double precision :: vv                  ! Vapor Specific Volume
        double precision :: Hamaker             ! Hamaker Constant
        double precision :: Avgadro             ! Avogadro #
        double precision :: Mass                ! Molecular Mass
        double precision :: Amass               ! Atomice Mass
        double precision :: Boltzmann           ! Boltzmann Constant
        double precision :: Conduct             ! Conductivity
!
!       Gemoetry
!
        double precision :: width               ! Width of unit cell
        double precision :: b                   ! width of post
        double precision :: thick               ! Thickness of Evaporator
!       double precision :: d                   ! Diameter of Pore
        double precision :: Pitch               ! Pitch between pores
        double precision :: height              ! Height of Post
        double precision :: L                   ! Length of Evaporator
        integer :: n_cell                       ! Number of Pores pre unit cell
        double precision :: Cp                  ! =Pitch/d
        double precision :: Cb                  ! =b/width
        double precision :: Dv                  ! Diameter for Vapor Transport Line
```

```
      double Precision :: Dl              ! Diameter in Liquid Transport Line
      double precision :: l_tran_l        ! Length of Liquid Transport Line
      double precision :: l_tranv         ! Length of Vapor Transport Line
      double precision :: n_p             ! Total number of pores
!
!     Meniscus
!
      double precision :: K0              ! Curvature in Previous Iteration
      double precision :: K1              ! Previous Curvature in the next iteration
      double precision :: K2              ! New Curvature in the next iteration
      double precision :: PD0             ! Disjoining Pressure at Bottom
!
!     Constants
!
!
!     Iterations
!
      double precision :: dtheta          ! spacing between angles
!     double precision :: n_max           ! Maximum number for Calculation
      double precision :: x_x(n_max+2)    ! Position of Node n
      double precision :: xt(n_max+2)     ! Interface Temperature on Node n
      double precision :: xh(n_max+2)     ! Film Thickness of Node n
      double precision :: xM0(n_max+2)    ! Liquid Mass Flow in Node n
      double precision :: xxeq_f1(n_max+2)        ! eq_f1 in Node n
      double precision :: xxeq_f2(n_max+2)        ! eq_f2 in Node n
      double precision :: xxK2(n_max+2)   ! Curvature in Node n
      double precision :: xevap(n_max+2)  ! Evaporation Rate in Node n
      double precision :: x_1,x0,x1,x2    ! Position of Neighborhoods
      double precision :: x_2
      double precision :: h_1,h0,h1,h2    ! Film Thickness of Neighborhoods
      double precision :: h_2
      double precision :: y               ! Increment in Newton's method
      double precision :: Curvature0      ! Curvature for K2_p
      double precision :: Curvature1      ! Curvature for K2_n
      double precision :: Curvaturem      ! Curvature for K2_m
      double precision :: new_h, old_h    ! Storage for h
      double precision :: h1_p,h1_n,h1_m  ! Storage for Bisection Method
      double precision :: h2_p,h2_n,h2_m  ! Storage for Bisection Method
      double precision :: h_1_n,h_1_m     ! Storage for Bisection Method

      double precision :: M0,M1           ! Mass Flow Rate &
      &M0=Mn-1, M1=Mn
      double precision :: theta           ! Angle
      double precision :: Ain             ! Evaporative Area
      double precision :: An              ! Area for Heat Transfer at Wall
      double precision :: A_ratio         ! =(Ain/An)
      double precision :: eq_f1,eq_f2     ! eq_f1, eq_f2
      double precision :: old_K2          ! New Curvature

      double precision :: K2_p,K2_n,K2_m  ! Curvature for Bisection Method
```

```
      double precision :: eq_f1_m,M1_m    ! eq_f1 and M1_m for dK_m
      double precision :: Tl_p,Tl_n,Tl_m  ! Liquid Temperature &
      &for Bisection Method
      double precision :: eq_f2_m         ! eq_f2 for Tl_m
      double precision :: xTl             ! Liquid Temperature in eq_f2 Loop
      double precision :: xeq_f2          ! eq_f2 in eq_f2 loop
      double precision :: sc              ! Constant Part in Evaporation Model
      double precision :: dK,dK_m,old_dK  ! dK=Kn-Kn-1
      double precision :: dK_n,dK_p,ddK   ! dK for Bisection Method

      double precision :: K_ini           ! Curvature at Bottom Meniscus,
      double precision :: Tw_ini          ! Twall at the first node
      double precision :: Tl_ini          ! Liquid Temperature at the first node
      double precision :: M_ini           ! Liquid Temperature

      double precision :: p0,p1,pm        ! Film Thickness Storage &
      &for Bisection Method
      double precision :: h
      double precision :: old_curvature   ! Curvature in Previous Iteration
      double precision :: oldh1           !
      double precision :: y_1,y0,y1,y2    ! Interface Position of Neighborhoods
      double precision :: y_2             ! Interface Position of Neighborhoods
      double precision :: z0              ! z position
      double precision :: y1_n,y1_p       ! Interface Position for Bisection Method
      double precision :: y1_m            ! Interface Position for Bisection Method
      double precision :: y_1_n,y_1_p     ! Interface Position for Bisection Method
      double precision :: y_1_m   ! Interface Position for Bisection Method
      double precision :: cca,ccb,ccc     ! Fitting Parameter x=ay^2+by+c
      double precision :: oldTl           ! Liquid Temp in Iteration
      double precision :: prev_Tl  ! Liquid Temp in previous Iteration
      double precision :: prev_dK ! Curvature drop in Previous Iteration
      double precision :: prev_vl  ! Specific Volume in Previous Iterration
      double precision :: d_En0
      double precision :: old_eq_f2       ! eq_f2 in Previous Loop
      double precision :: prev_En         ! En in Previous Iteration = En-1
      double precision :: old_eq_f1       ! eq_f1 in Previous Loop

      double precision :: old_xTl         !
      double precision :: xeq_f20,dkm     !

      integer :: ll,llend
!
!*************************************************!
!     Declaring File Names
!*************************************************
!
      open(1,file='eq1.dat')
      open(10,file='eq1_1.dat')
      open(2,file='eq2.dat')
      open(3,file='result.dat')
```

```
          open(4,file='h1.dat')
          open(11,file='force.dat')
          open(12,file='h1_1.dat')
          open(13,file='fitting.dat')
          open(14,file='eq2_1.dat')
          open(15,file='final.dat')
!
!
!
          call spinit(ircnt,cright,title1,title2,version,year)
          print *,ircnt,cright,title1,title2,version,year
!
!*****************************************************!
!         Setting Constant Values
!*****************************************************
!
          Avgadro=6.02d23
          Mass=18.0d-3
          Amass=Mass/Avgadro
          Boltzmann=1.3805d-23
!
          pi=3.141592654d0
          Tamb=390.50d0
          q_source=100.0d4

          Hamaker=1.28d-21
          z0=0.0d0

!         cRc=Rc/ck
!*****************************************************
!         Determine Inintial Values of Nodes
!*****************************************************
!         n_max=4000
          dtheta=0.3d0*pi/180.0d0
          h0=0.0d0
          do kk=1,n_max+2
            xt(kk)=373.0d0
            x_x(kk)=dx*float(kk)
            xh(kk)=Rc-sqrt(2.0d0*Rc*x_x(kk)-x_x(kk)**2.0d0)
            if (kk>n_max*2.0d0/3.0d0) then
              xh(kk)=4.0e-9
            end if
            print *,kk,x_x(kk),xt(kk),xh(kk)
          end do
!
          Tw_ini=353.15d0
          M_ini=15.50d-11
          Tv_ini=Tw_ini-1.0d-5
          M0=M_ini
!
!0000000000000000000000000000000000000000000000000000000
```

```
!           Starting Program
!00000000000000000000000000000000000000000000000000000000
!
!11111111111111111111111111111111111111111111111111111111
!           Loop of Wall Temperature from 353.15K to 423.15K
!11111111111111111111111111111111111111111111111111111111
!
           do itw=0,7,1
              Tw_ini=353.15d0+float(itw)*10.0
              Tv_ini=Tw_ini-1.0d-5
              M_ini=M_ini*3.0d0
              M0=M_ini
!
!22222222222222222222222222222222222222222222222222222222
!              Loop for Changing Initial Curvature in Pore
!22222222222222222222222222222222222222222222222222222222
!
              do mmm=1,6
                 if (mmm>1) then
                    M_ini=M_ini*0.9d0
                 end if
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!                Determing Bottom Curvature
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                 select case (mmm)
                 case (1)
                    dkm=0.05*1.0d0/(d/2.0d0)*2.0d0
                 case (2)
                    dkm=0.1*1.0d0/(d/2.0d0)*2.0d0
                 case (3)
                    dkm=0.2*1.0d0/(d/2.0d0)*2.0d0
                 case (4)
                    dkm=0.3*1.0d0/(d/2.0d0)*2.0d0
                 case (5)
                    dkm=0.4*1.0d0/(d/2.0d0)*2.0d0
                 case default
                    dkm=0.499*1.0d0/(d/2.0d0)*2.0d0
                 end select
!
!33333333333333333333333333333333333333333333333333333333
!                Loop for Determing Flow Rate=Total Evap. Rate in Pore
!33333333333333333333333333333333333333333333333333333333
!
            do mm=1,10,1
!

                   do nn=1,nmax
                      x_x(nn)=0.0d0
                  xt(nn)=0.0d0
                      xh(nn)=0.0d0
```

```
                    xM0(nn)=0.0d0
                    xxeq_f1(nn)=0.0d0
                    xxeq_f2(nn)=0.0d0
                    xevap(nn)=0.0d0
                    xxK2(nn)=0.0d0
                  end do
!
                  if (mm>1) then
                    if (M0>0) then
                      M_ini=M_ini-M0*0.99d0
                    else
                      M_ini=M_ini-M0*1.2d0
                    end if
                  end if

!
                  K0=1.0d0/(d/2.0d0)*2.0d0-dkm
                  M0=M_ini
                dK=1.0e-2
                  prev_TI=Tw_ini-0.12d0
                  Tw=Tw_ini
                  TI=Tw_ini-0.1d0
                  Tv=Tv_ini
                  Pv=psattn(Tv)*1.0d6-25.0d3
!
!4444444444444444444444444444444444444444444444444444
!               Main Loop for Determing Local Film Thickness
!         & Interface Temperature for each Nodes
!4444444444444444444444444444444444444444444444444444
!b

                  do ll=0,n_max
                    select case (ll)
                      case (0)
                        x_1=x_x(ll+1)
                        x0=0.0d0
                        x1=x_x(ll+1)
                        x2=x_x(ll+2)
                        h_1=2.0d0*Rc-xh(ll+1)
                        h0=Rc
                        h1=xh(ll+1)
                        h2=h1
                        y_1=Rc-h_1
                        y0=Rc-h0
                        y1=Rc-h1
                        y2=Rc-h2
                        old_h=xh(ll+1)
                      case (1)
!                       x_2=x_x(ll)
                        x_1=0.0d0
```

```
                    x0=x_x(ll)
                    x1=x_x(ll+1)
                    x2=x_x(ll+2)
                    h_1=Rc
                    h0=xh(ll)
                    h1=xh(ll+1)
                    h2=h1
                    y_1=Rc-h_1
                    y0=Rc-h0
                    y1=Rc-h1
                    y2=Rc-h2
                    old_h=xh(ll+1)
                  case default
!                   x_2=x_x(ll-2)
                    x_1=x_x(ll-1)
                    x0=x_x(ll)
                    x1=x_x(ll+1)
                    x2=x_x(ll+2)
!                   h_2=xh(ll-2)
                    h_1=xh(ll-1)
                    h0=xh(ll)
                    h1=xh(ll+1)
                    h2=h1
!                   y_2=Rc-h_2
                  y_1=Rc-h_1
                    y0=Rc-h0
                  y1=Rc-h1
                    y2=Rc-h2
                    old_h=xh(ll+1)
                end select
!

                if (ll>1) then
                  if ((ccb**2-4.0*cca*(ccc-x1))<0) then
                    y1=Rc-h1
                  else
                    y1=(-ccb+sqrt(ccb**2-4.0*cca*(ccc-x1)))/2.0/cca
                  end if
                end if
!
!
                x_1=x_1/ck
                x0=x0/ck
                x1=x1/ck
                x2=x2/ck
                h_1=h_1/ck
                h0=h0/ck
                h1=h1/ck
                h2=h2/ck
                y_1=y_1/ck
```

```
                y0=y0/ck
                y1=y1/ck
                y2=y2/ck
                old_h1=h1
!
                  print *,'Beginning of',ll
                  Curvature0=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!               Check if y1>y0
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                do iii=1,1000000
                  y1=y1+1.0d-11/ck
                  h1=cRc-y1
                  Curvature0=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
                  if (y1>y0) then
                    print *,ll,iii,y0,y1,Curvature0
                    exit
                  end if
                end do
!
!5555555555555555555555555555555555555555555555555
!               Determining Local Film Thickness for a given Curvature and Temperature
!5555555555555555555555555555555555555555555555555
!
                do ii=1,200000
!
                  old_curvature=Curvature0
!
                  if (Curvature0>=0.0d0) then
                    oldh1=h1
                    oldy1=y1
                    y1=y1+1.0d-12/ck
                    h1=cRc-y1
                    if (ll==0) then
                      y_1=-y1
                    end if
                    Curvature0=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
                    if (Curvature0<0.0d0) then
                      y1_p=y1-1.0d-12/ck
                      y1_n=y1
                      h1_p=cRc-y1_p
                      h1_n=cRc-y1_n
                   write(12,'(3I10,20E22.10)') mm,ll,ii,oldh1,&
                   &h1_n,h1_p,old_curvature&
                   &,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,oldy1,ck)*2.0d0,K0,&
                   &x_1,x0,x1,y_1,y0,oldy1
                      write(12,'(3I10,20E22.10)') mm,ll,ii,h1,h1_n,h1_p,&
                   &Curvature0,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0,&
```

```
                &K0,x_1,x0,x1,y_1,y0,y1
                    exit
                    end if
                  end if
!
                  if (Curvature0<0.0d0) then
                    oldh1=h1
                   oldy1=y1
                    y1=y1-1.0d-12/ck
                   h1=cRc-y1
                    if (ll==0) then
                      y_1=-y1
                    end if
                    Curvature0=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
                    if (Curvature0>=0.0d0) then
                      y1_n=y1
                      y1_p=y1+1.0d-12/ck
                      h1_p=cRc-y1_p
                      h1_n=cRc-y1_n
                      write(12,'(3I10,20E22.10)') mm,ll,ii,oldh1,h1_n,h1_p,&
                  &old_curvature,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,&
                  &oldy1,ck)*2.0d0&
                  &,K0,x_1,x0,x1,y_1,y0,oldy1
                      write(12,'(3I10,20E22.10)') mm,ll,ii,h1,h1_n,h1_p,Curvature0&
                     &,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0,K0&
                  &,x_1,x0,x1,y_1,y0,y1
                        exit
                   end if
                  end if
!
                end do
!
!
!
                y1=y1_p
                if (ll==0) then
                  y_1=-y1
                end if
                Curvature1=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
!
                y1=y1_n
                if (ll==0) then
                  y_1=-y1
                end if
                Curvature0=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0-K0
!
                do ii=1,200
                y1_m=y1_p-Curvature1*(y1_p-y1_n)/(Curvature1-Curvature0)
```

```
                    if (ll==0) then
                       y_1=-y1_m
                    end if
                    Curvaturem=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1_m,ck)*2.0d0-K0
!
                    if (abs((y1_m-y1_p)/y1_m)<1.0d-8) then
                       if (abs(Curvature0)<1.0d-12) then
                          exit
                       end if
                    end if
!
                    if ((Curvaturem*Curvature1)<0.0d0) then
                       y1_n=y1_p
                       Curvature0=Curvature1
                    end if
                    y1_p=y1_m
                    Curvature1=Curvaturem
                    if (ll==38533) then
                       if (MOD(ii,10)==0) then
                       write(4,'(3I10,10E22.10)')mm,ll,ii,h1_n,h1_p,Curvature0,&
                    Curvature1&
                       &,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1_m,ck)*2.0d0,K0
                    end if
                 end do
!
!5555555555555555555555555555555555555555555555555
!                 End of Determing Local Film Thickness
!5555555555555555555555555555555555555555555555555
!
                 if (MOD(ll,1)==0) then
                    write(4,'(3I10,10E22.10)') mm,ll,ii,h1_n,h1_p,Curvature0,Curvature1&
                    &,cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1_m,ck)*2.0d0,K0
                 end if
!
!            pause
!
                 y1=y1_m
                 h1=cRc-y1
                 if (ll==0) then
                    y_1=-y1
                 end if
                 K1=cH_lang(y0,z0,x_1,x0,x1,y_1,y0,y1,ck)*2.0d0
                 K2=cH_lang(y1,z0,x0,x1,x2,y0,y1,y2,ck)*2.0d0
!
!
                 K1=K0
                 K2=K1
!
                 x_1=x_1*ck
                 x0=x0*ck
```

```
                x1=x1*ck
                x2=x2*ck
                h_1=h_1*ck
                h0=h0*ck
                h1=h1*ck
                h2=h2*ck
                y_1=y_1*ck
                y0=y0*ck
                y1=y1*ck
                y2=y2*ck
                old_h1=h1
                xh(ll+1)=h1
!
                call P_coeff(x_1,x0,x1,y_1,y0,y1,cca,ccb,ccc)
                write(13,'(2I10,15E22.10)') mm,ll,x_1,x0,x1,y_1,y0,y1,cca,ccb,ccc
!
!5555555555555555555555555555555555555555555555555
!               Loop for Finding Interface Temperature
!               which satisfies eq_f1=0 and eq_f2=0.
!5555555555555555555555555555555555555555555555555
!
                do itemp=1,20000
!
                  M1=C_F1(h1,Rc,Tl)*dpdxdK(dK,h0,h1,h2,dx,Tl,Rc)
                  Psat=psattn(Tl)*1.0d6
                  sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*Tl)
                  vl=VFTSI(Tl)
                  vl=vl*Mass/Avgadro
                  evap1=sc*sinh(dSndK(Tl,Tv,vl,Pv,Rc,K0,dK,(h1+h0)/2.0d0,dx))
                  theta=atan((h0-h1)/dx)
                  Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                  An=2.0d0*pi*Rc*dx
                  eq_f1=(M0-M1)/Ain-evap1
                  if (ll>30000) then
                    ddK=abs(dK/100.0d0)
                  else
                    ddK=abs(dK/100.0d0)
                  end if
!
!6666666666666666666666666666666666666666666666666
!               Finding dK to satisfy eq_f1=0 by assuming Tl
!6666666666666666666666666666666666666666666666666
!
                  do ii=1,100000
                    old_eq_f1=eq_f1
                    theta=atan((h0-h1)/dx)
                    Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                    An=2.0d0*pi*Rc*dx
                    Psat=psattn(Tl)*1.0d6
                    sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*Tl)
                    vl=VFTSI(Tl)
```

```
                vl=vl*Mass/Avgadro
!
                if (eq_f1>0.0d0) then
                  old_dK=dK
                  if (ll>1000) then
                    dK=dK+ddK
                  else
                    dK=dK+ddK
                  end if
                  M1=C_F1(h1,Rc,Tl)*dpdxdK(dK,h0,h1,h2,dx,Tl,Rc)
                  evap1=sc*sinh(dSndK(Tl,Tv,vl,Pv,Rc,K0,dK,(h1+h0)/2.0d0,dx))
                  eq_f1=(M0-M1)/Ain-evap1
                  if (eq_f1<=0) then
                    dK_p=old_dK
                    dK_n=dK
                    exit
                  end if
!                 end if
!
                if (eq_f1<=0.0d0) then
                  old_dK=dK
                  if (ll>1000) then
                    dK=dK-ddK
                  else
                    dK=dK-ddK
                  end if
                  M1=C_F1(h1,Rc,Tl)*dpdxdK(dK,h0,h1,h2,dx,Tl,Rc)
                  evap1=sc*sinh(dSndK(Tl,Tv,vl,Pv,Rc,K0,dK,(h1+h0)/2.0d0,dx))
                  eq_f1=(M0-M1)/Ain-evap1
                  if (eq_f1>0) then
                    dK_p=dK
                    dK_n=old_dK
                    exit
                  end if
                end if
              end do

              if (MOD(ll,10)==0) then
                write(10,'(4I10,10E22.10)') mm,ll,ii,itemp,Tl,dK,eq_f1,M0,M1,&
             &evap1*Ain,C_F1(h1,Rc,Tl),dpdxdK(dK,h0,h1,h2,dx,Tl,Rc)
              end if
!
!
!
!
!
              do ii=1,200
                theta=atan((h0-h1)/dx)
                Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                An=2.0d0*pi*Rc*dx
```

```
                    Psat=psattn(Tl)*1.0d6
                    sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*Tl)
                    vl=VFTSI(Tl)
                    vl=vl*Mass/Avgadro
!
                    dK_m=(dK_p+dK_n)/2.0d0
                    M1_m=C_F1(h1,Rc,Tl)*dpdxdK(dK_m,h0,h1,h2,dx,Tl,Rc)
                    evap1_m=sc*sinh(dSndK(Tl,Tv,vl,Pv,Rc,K0,&
                  &dK_m,(h1+h0)/2.0d0,dx))
                    eq_f1_m=(M0-M1_m)/Ain-evap1_m
!
                    M1=C_F1(h1,Rc,Tl)*dpdxdK(dK_n,h0,h1,h2,dx,Tl,Rc)
                    evap1=sc*sinh(dSndK(Tl,Tv,vl,Pv,Rc,K0,dK_m,(h1+h0)/2.0d0,dx))
                    eq_f1=(M0-M1)/Ain-evap1

!
                    if (abs((dK_m-dK_p)/dK_m)<1.0d-8) then
                      if (abs(eq_f1)<1.0d-12) then
                        exit
                      end if
                    end if
!
                    if ((eq_f1_m*eq_f1)<0.0d0) then
                      dK_p=dK_m
                      dK_n=dK_n
                    else
                      dK_p=dK_p
                      dK_n=dK_m
                    end if
                  end do
!
!6666666666666666666666666666666666666666666666666
!                 End of Finding dK to satisfy eq_f1=0
!6666666666666666666666666666666666666666666666666
!
                  if (MOD(ll,1)==0) then
                    write(1,'(4I10,10E22.10)') mm,ll,ii,itemp,eq_f1,dK_m,dK_p,Tl
                  end if
!
                  dK=dK_m
                K2=K1+dK
                  oldTl=Tl

!6666666666666666666666666666666666666666666666666
!                 Finding Tl_p and Tl_n to satisfy
!                  eq_f2>0 and eq_f2<0 for a given dK
!6666666666666666666666666666666666666666666666666
                  xTl=Tl
                  vl=VFTSI(xTl)
                  Conduct=CONVT(vl,xTl)
```

```
      vl=vl*Mass/Avgadro
      Psat=psattn(xTl)*1.0d6
      sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*xTl)
      evap1=sc*sinh(dSn1K(xTl,Tv,vl,Pv,Rc,K1,K2,(h1+h0)/2.0d0,dx))
      theta=atan((h0-h1)/dx)
      Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
      An=2.0d0*pi*Rc*dx
      A_ratio=(Rc-(h0+h1)/2.0d0)/cos(theta)/Rc
      hfg=(HGTSI(xTl)-HFTSI(xTl))*1.0d3
     xeq_f20=-Conduct*(Tw-xTl)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc&
     &-evap1*hfg*A_ratio
      if (ll==0) then
         prev_vl=VFTSI(prev_Tl)
         Cp=CPVT3N(prev_vl,prev_Tl)*1.0d-3
       Cp=4.2246d3
         En0=Cp*M0*prev_Tl
       xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-En0)/An
      else
         xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-&
         &prev_En)/An
      end if
!
      if (xeq_f2<0) then
      old_xeq_f2=xeq_f2
      old_xTl=xTl
         If (abs(xeq_f2)>1.0d8) then
            xTl=xTl-1.0e-1
         else
        xTl=xTl-1.0e-3
      end if
         vl=VFTSI(xTl)
         Conduct=CONVT(vl,xTl)
         vl=vl*Mass/Avgadro
         Psat=psattn(xTl)*1.0d6
         sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*xTl)
         evap1=sc*sinh(dSn1K(xTl,Tv,vl,Pv,Rc,K1,K2,(h1+h0)/2.0d0,dx))
         theta=atan((h0-h1)/dx)
         Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
         An=2.0d0*pi*Rc*dx
         A_ratio=(Rc-(h0+h1)/2.0d0)/cos(theta)/Rc
        hfg=(HGTSI(xTl)-HFTSI(xTl))*1.0d3
        xeq_f20=-Conduct*(Tw-xTl)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc&
     &-evap1*hfg*A_ratio
         if (ll==0) then
            prev_vl=VFTSI(prev_Tl)
            Cp=CPVT3N(prev_vl,prev_Tl)*1.0d-3
            Cp=4.2246d3
            En0=Cp*M0*prev_Tl
            xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-En0)/An
```

```
                else
                  xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-&
                  &prev_En)/An
                end if
!
                if (xeq_f2>0) then
                  Tl_p=xTl
                  Tl_n=old_xTl
                  write(14,'(4I10,10E22.10)') mm,ll,ii,itemp,old_xTl,old_xeq_f2,&
                  &Enxmass(Rc,Tw,h1,old_xTl,dx,dK,h0,h2),prev_En
                  write(14,'(4I10,20E22.10)') mm,ll,ii,itemp,xTl,xeq_f2,&
                  &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2),prev_En,&
                  &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-prev_En,&
                 &xeq_f20*An,&
                  &dpdxdKw(dK,h0,h1,h2,dx,xTl,Rc,ll),Rc,Tw,&
                 &xTl,dx,dK,h0,h1,h2
                  exit
              end if
            end if
!
!
             if (xeq_f2>0) then
                old_xeq_f2=xeq_f2
                old_xTl=xTl
                if (abs(xeq_f2)>1.0d8) then
                  xTl=xTl+1.0e-1
                else
                  xTl=xTl+1.0e-3
                end if
                vl=VFTSI(xTl)
                Conduct=CONVT(vl,xTl)
                vl=vl*Mass/Avgadro
                Psat=psattn(xTl)*1.0d6
                sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*xTl)
                evap1=sc*sinh(dSn1K(xTl,Tv,vl,Pv,Rc,K1,K2,(h1+h0)/2.0d0,dx))
                theta=atan((h0-h1)/dx)
                Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                An=2.0d0*pi*Rc*dx
                A_ratio=(Rc-(h0+h1)/2.0d0)/cos(theta)/Rc
                hfg=(HGTSI(xTl)-HFTSI(xTl))*1.0d3
                eq_f20=-Conduct*(Tw-xTl)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc&
              &-evap1*hfg*A_ratio
                if (ll==0) then
                  prev_vl=VFTSI(prev_Tl)
                  Cp=CPVT3N(prev_vl,prev_Tl)*1.0d-3
                 Cp=4.2246d3
                 En0=Cp*M0*prev_Tl
                 xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-En0)/An
               else
```

```
                        xeq_f2=xeq_f20-(Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-&
                        &prev_En)/An
                      end if

                        if (xeq_f2<0) then
                          Tl_p=old_xTl
                          Tl_n=xTl
                          write(14,'(4I10,10E22.10)') mm,ll,ii,itemp,old_xTl,old_xeq_f2,&
                          &Enxmass(Rc,Tw,h1,old_xTl,dx,dK,h0,h2),prev_En
                          write(14,'(4I10,20E22.10)') mm,ll,ii,itemp,xTl,xeq_f2,&
                          &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2),prev_En,&
                          &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-prev_En,xeq_f20*An,&
                          &dpdxdKw(dK,h0,h1,h2,dx,xTl,Rc,ll),Rc,Tw,xTl,dx,dK,h0,h1,h2
                          exit
                        end if
                      end if
                      if (MOD(itemp,10)==0) then
                        write(14,'(4I10,20E22.10)') mm,ll,ii,itemp,xTl,xeq_f2,&
                        &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2),prev_En,&
                        &Enxmass(Rc,Tw,h1,xTl,dx,dK,h0,h2)-prev_En,xeq_f20*An,&
                        &dpdxdKw(dK,h0,h1,h2,dx,xTl,Rc,ll),Rc,Tw,xTl,dx,dK,h0,h1,h2
                      end if

                      Tl=xTl
!6666666666666666666666666666666666666666666666666
!                     Finding Tl_p and Tl_n to satisfy
!               eq_f2>0 and eq_f2<0 for a given dK
!6666666666666666666666666666666666666666666666666
!
!                 end do
!
!


                  do ii=1,200
                    Tl_m=(Tl_p+Tl_n)/2.0d0
                    vl=VFTSI(Tl_m)
                    Conduct=CONVT(vl,Tl_m)
                    vl=vl*Mass/Avgadro
                    Psat=psattn(Tl_m)*1.0d6
                    sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*Tl_m)
                    evap1=sc*sinh(dSn1K(Tl_m,Tv,vl,Pv,Rc,K1,K2,(h1+h0)/2.0d0,dx))
!
                    theta=atan((h0-h1)/dx)
                    Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                    An=2.0d0*pi*Rc*dx
                    A_ratio=(Rc-(h0+h1)/2.0d0)/cos(theta)/Rc
                    hfg=(HGTSI(Tl_m)-HFTSI(Tl_m))*1.0d3
                    eq_f2_m=-Conduct*(Tw-Tl_m)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc&
                  &-evap1*hfg*A_ratio
                    if (ll==0) then
```

```
                    prev_vl=VFTSI(prev_Tl)
                    Cp=CPVT3N(prev_vl,prev_Tl)*1.0d-3
                    Cp=4.2246d3
                    En0=Cp*M0*prev_Tl
                    eq_f2_m=eq_f2_m-(Enxmass(Rc,Tw,h1,Tl_m,dx,dK,h0,h2)-En0)/An
                  else
                    eq_f2_m=eq_f2_m-(Enxmass(Rc,Tw,h1,Tl_m,dx,dK,h0,h2)-&
                    &prev_En)/An
                  end if
!
!
                  vl=VFTSI(Tl_p)
                  Conduct=CONVT(vl,Tl_p)
                  vl=vl*Mass/Avgadro
                  Psat=psattn(Tl_p)*1.0d6
                  sc=2.0d0*Mass/Avgadro*Psat/sqrt(2.0d0*pi*Amass*Boltzmann*Tl_p)
                  evap1=sc*sinh(dSn1K(Tl_p,Tv,vl,Pv,Rc,K1,K2,(h1+h0)/2.0d0,dx))
!
                  theta=atan((h0-h1)/dx)
                  Ain=2.0d0*pi*(Rc-(h0+h1)/2.0d0)/cos(theta)*dx
                  An=2.0d0*pi*Rc*dx
                  A_ratio=(Rc-(h0+h1)/2.0d0)/cos(theta)/Rc
                  hfg=(HGTSI(Tl_p)-HFTSI(Tl_p))*1.0d3
                  eq_f2=-Conduct*(Tw-Tl_p)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc&
                &-evap1*hfg*A_ratio
                 if (ll==0) then
                    prev_vl=VFTSI(prev_Tl)
                    Cp=CPVT3N(prev_vl,prev_Tl)*1.0d-3
                    Cp=4.2246d3
                      En0=Cp*M0*prev_Tl
                      eq_f2=eq_f2-(Enxmass(Rc,Tw,h1,Tl_p,dx,dK,h0,h2)-En0)/An
                    else
                      eq_f2=eq_f2-(Enxmass(Rc,Tw,h1,Tl_p,dx,dK,h0,h2)-&
                      &prev_En)/An
                    end if
!
                  if (abs((Tl_m-Tl_p)/Tl_m)<1.0d-8) then
                      exit
                    end if
!
                  if ((eq_f2_m*eq_f2)<0.0d0) then
                      Tl_p=Tl_p
                      Tl_n=Tl_m
                  else
                      Tl_p=Tl_m
                      Tl_n=Tl_n
                  end if
                 end do
!
!55555555555555555555555555555555555555555555555555
```

```
!                  Loop for Finding Interface Temperature
!                  which satisfies eq_f1=0 and eq_f2=0.
!555555555555555555555555555555555555555555555555
!
                   Tl=Tl_m
                   write(2,'(3I10,10E22.10)') mm,ll,ii,Tl,eq_f2
!              pause
!
!
                   xt(ll+1)=Tl
                   xh(ll+1)=h1
                   if (ll<>0) then
                     xM0(ll)=M0
                   end if
                   xxeq_f1(ll+1)=eq_f1
                   xxeq_f2(ll+1)=eq_f2
                   xevap(ll+1)=evap1*Ain
                   xxK2(ll+1)=K2
                   if (MOD(ll,10)==0) then
                     write(3,'(3I10,15E22.10)') mmm,mm,ll,x_x(ll+1),K2,h1&
                  &,Tl,M0,M1,&
                  &evap1*Ain,-Conduct*(Tw-Tl)/log((Rc-(h0+h1)/&
                  &2.0d0)/Rc)/Rc*An,eq_f1,eq_f2,theta
                     print *,mm,ll,x_x(ll+1),K2,h1,Tl,M0,M1,evap1*Ain&
                     &,-Conduct*(Tw-Tl)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc*An,eq_f1,eq_f2
                   elseif (h1<5.0d-8) then
                     write(3,'(3I10,15E22.10)')&
                  &mmm,mm,ll,x_x(ll+1),K2,h1,Tl,M0,M1,evap1*Ain&
                     &,-Conduct*(Tw-Tl)/log((Rc-(h0+h1)/2.0d0)/Rc)/Rc*An,&
                  &eq_f1,eq_f2,theta
end if
                   prev_En=Enxmass(Rc,Tw,h1,Tl,dx,dK,h0,h2)
                   dK=K1-K2
                   K0=K2
                   M0=M1
                   prev_Tl=Tl
                   prev_dK=dK
                   print *,'end of',ll
!
                   if (K0<(2.0/d)) then
                     llend=ll
                     exit
                   end if
                 end do          ! end of ll
!
!444444444444444444444444444444444444444444444444444
!                End of Main Loop for Determing Local Film Thickness
!          & Interface Temperature for a Node
```

```
!444444444444444444444444444444444444444444444444444444
!
                write(1,*)
                write(2,*)
                write(3,*)
                write(4,*)
                write(10,*)
                if (M0>0) then
                  if (M0<(M_ini/100d0)) then
                    print *,mm,M_ini,M0
                    print *,'end of calculation'
                    exit
                  end if
                end if
!             pause
              end do ! end of mm
!
!333333333333333333333333333333333333333333333333333333
!             End of Loop for Determing Flow Rate=Total Evap. Rate in Pore
!333333333333333333333333333333333333333333333333333333
!
            do nll=0,llend
              h1=xh(nll+1)
              Tl=xt(nll+1)
              if (nll==0) then
                M0=M_ini
              else
                M0=xM0(nll)
              end if
              M1=xM0(nll+1)
              eq_f1=xxeq_f1(nll+1)
              eq_f2=xxeq_f2(nll+1)
              K2=xxK2(nll+1)
              evap1=xevap(nll+1)
!
             if (MOD(nll,10)==0) then
                write(15,'(3I10,15E22.10)') &
               &iTw,mmm,nll,x_x(nll+1),K2,h1,Tl,M0,M1,evap1,eq_f1,eq_f2
              elseif (h1<5.0d-8) then
               write(15,'(3I10,15E22.10)')&
               &iTw,mmm,nll,x_x(nll+1),K2,h1,Tl,M0,M1,evap1,eq_f1,eq_f2
             end if
!
              end do

           end do
!
!222222222222222222222222222222222222222222222222222222
```

```
!           End of Loop for Changing Initial Curvature in Pore
!22222222222222222222222222222222222222222222222222222
!
!
            end do
!
!1111111111111111111111111111111111111111111111111111
!           End of Loop of Wall Temperature from 353.15K to 423.15K
!1111111111111111111111111111111111111111111111111111
!
!
!0000000000000000000000000000000000000000000000000000
!           End of Program
!0000000000000000000000000000000000000000000000000000
!
end program

contains

Program Fs
            function C_F1(sh,sRc,sTemp)
               double precision:: svl,sTemp,sRc,sr,sTc,spi
               double precision:: sterm1,ssr,C_F1,sh
               spi=3.141592654d0
               svl=VFTSI(sTemp)
               sTc=sTemp-273.15
               sr=sRc-sh
!
               sterm1=-4.0d0*sRc**2.0d0*sr**2.0d0+sRc**4.0d0+&
               &4.0d0*sr**4.0d0*log(sRc/sr)+3.0d0*sr**4.0d0
               C_F1=-sterm1*spi/(VISCOS(svl,sTc)*1.0d-6)/svl/8.0d0
!              print *,'C_F1,sterm1= '
!              print *,C_F1,sterm1
!              pause
            end function
!
            function C_F2(sh,sRc,sTemp,sa,sb,sc)
               double precision:: svl,sTemp,sRc,sr,sTc,spi
               double precision:: sterm1,ssr,C_F1,sh
               double precision:: snx,sny,stx,sty,ssC
               double precision:: sa,sb,sc
               spi=3.141592654d0
               svl=VFTSI(sTemp)
               sTc=sTemp-273.15
               sr=sRc-sh
               snx=-1.0/sqrt((2.0*sa*sr+sb)**2+1.0)
!              sny=2.0*sa*sr/sqrt((2.0*sa*sr+sb)**2+1.0)-sb/sqrt((2.0*sa*sr+sb)**2+1.0)
               sny=sqrt(1.0-snx**2)
               stx=(2.0*sa*sr+sb)/sqrt((2.0*sa*sr+sb)**2+1.0)
               sty=1.0/sqrt((2.0*sa*sr+sb)**2+1.0)
```

```
!               ssC=stx/(VISCOS(svl,sTc)*1.0d-6)/(snx*sty+sny*stx)
                ssC=1.0/(snx*sty+sny*stx)
!
                sterm1=4.0*sRc**2*sr+8.0*sr**3*log(sr)-8.0*sr**3*log(sRc)-4.0*sr**3
                C_F2=-ssC*sterm1*spi/(VISCOS(svl,sTc)*1.0d-6)/svl/8.0d0
!               print *,'C_F1,sterm1= '
!               print *,C_F1,sterm1
!               pause
              end function
!
              function Pe(sTl,sh,shd,shh,sRc)
                double precision :: sp0,sp1,sTc,sTl,sTf,ssigma,sRg
                double precision :: sK,sPsat,sdiff0,sdiff1,soldp1
                double precision :: sh,shd,shh,sRc,shamaker,Pe
                integer:: iis
!
!               shamaker=-1.28d-20
                shamaker=-1.28d-21
                shamaker=f_Hamaker(sh)
                sPsat=psattn(sTl)*1.0d6
                svl=VFTSI(sTl)
                sp0=sPsat
                sp1=sPsat-100.0d0
                sRg=462.0d0
!               print *,sTl,sh,shd,shh,sRc,sp0,sp1
!
                do iis=1,100
                  sTc=sTl-273.15d0
                  sTf=9.0d0/5.0d0*sTc+32.0d0
                  ssigma=SURFT(sTf)*4.448d0/0.305d0
                  sK=0.5d0*(1.0d0/(sRc-sh)/sqrt(1.0d0+(shd**2))+shh/&
                 &(1.0d0+shd**2)**(3.0d0/2.0d0))
!                 sK=0.0d0
!                 print *,'sK,ssigma= ',sk,ssigma
                  sdiff0=sp0-sPsat*exp(svl*(sp0-ssigma*sK+&
                  &shamaker/sh**3-sPsat)/sRg/sTl)
                  sdiff1=sp1-sPsat*exp(svl*(sp1-ssigma*sK+&
                  &shamaker/sh**3-sPsat)/sRg/sTl)
                  soldp1=sp1
                  sp1=sp1-sdiff1*(sp1-sp0)/(sdiff1-sdiff0)
                  sp0=soldp1
!                 print *,iis,sp0,sp1
                  if (abs((sdiff0-sdiff1)/sdiff0)<1.0d-7) then
!                   print *,sp0,sdiff0,sp1,sdiff1
                        Pe=sp0
!                       write(1,'(E17.5)') sp0
                    exit
                  end if
                end do
```

```
      end function
!
!
!
      function eq_K1(sRc,sx_1,sx0,sx1,sh_1,sh0,sh1)
        double precision :: FFA,FFB,FFC,FFD,FFE,FFF
        double precision :: sh_1,sh0,sh1,sRc
        double precision :: sx_1,sx0,sx1
        double precision :: sx_1_2,sx1_2
        double precision :: sh_1_2,sh1_2
        sx_1_2=(sx0+sx_1)/2.0d0
        sx1_2=(sx0+sx1)/2.0d0
        sh_1_2=(sh0+sh_1)/2.0d0
        sh1_2=(sh0+sh1)/2.0d0
        FFA=(sx1-sx0)/sqrt((sx1-sx0)**2.0d0+(sh1-sh0)**2.0d0)
        FFB=(sx0-sx_1)/sqrt((sx0-sx_1)**2.0d0+(sh0-sh_1)**2.0d0)
        FFD=(sh1-sh0)/sqrt((sx1-sx0)**2.0d0+(sh1-sh0)**2.0d0)
        FFC=sqrt((sx0-sx_1)**2.0d0+(sh0-sh_1)**2.0d0)
        FFE=(sh0-sh_1)/sqrt((sx0-sx_1)**2.0d0+(sh0-sh_1)**2.0d0)
        FFF=sqrt((sx1_2-sx_1_2)**2.0d0+(sh1_2-sh_1_2)**2.0d0)
        eq_K1=sqrt((FFA-FFB)**2.0d0+(FFD-FFE)**2.0d0)/FFF+1.0d0/sRc
      end function
!
!
      function dcur1(sx_1,sx,sx1,sx2,sh_1,sh,sh1,sh2,sRc)
        double precision:: sx_1,sx,sx1,sx2
        double precision:: sh_1,sh,sh1,sh2,sRc
        dcur1=eq_K1(sRc,sx,sx1,sx2,sh,sh1,sh2)-&
        &eq_K1(sRc,sx_1,sx,sx1,sh_1,sh,sh1)
      end function
!
      function dSn1(sTl,sTv,svl,sPv,sRc,sx_1,sx,sx1,sx2,sh_1,sh,sh1,sh2,sdx)
        double precision, dimension(3)::C_theta
        double precision :: sterm1
        double precision :: sterm2,sterm21,sterm22
        double precision :: sterm3
        double precision :: sterm4
        double precision :: sterm5,sterm51,sterm52
        double precision :: sTl,sTv,svl,sPv,sPsat
        double precision :: sTc,sTf
        double precision :: hamaker,sK,Boltzmann,dsn1
        double precision :: sRc,sdx,Kn
        double precision :: shd,shd2
        double precision :: sx_1,sx,sx1,sx2
        double precision :: sh_1,sh,sh1,sh2
!
        Boltzmann=1.3805d-23
!       hamaker=-1.28d-20
        hamaker=-1.28d-21
        hamaker=f_Hamaker(sh)
        sTc=sTl-273.15d0
```

```
      sTf=9.0d0/5.0d0*sTc+32.0d0
      ssigma=SURFT(sTf)*4.448d0/0.305d0
      C_theta(1)=2290.0d0
      C_theta(2)=5160.0d0
      C_theta(3)=5360.0d0
      sPsat=psattn(sTl)*1.0d6
!
      sterm1=4.0d0*(1.0d0-sTv/sTl)
!
      sterm21=(1.0d0/sTv-1.0d0/sTl)
      sterm22=0.0d0
      do is=1,3
        sterm22=sterm22+C_theta(is)/2.0d0+C_theta(is)/&
     &(exp(C_theta(is)/sTv)-1.0d0)
!       print *,is,C_theta(is)/2.0d0,exp(C_theta(is)/sTv),sTv
      end do
!     print *,'sterm21,sterm22= '
!     print *,sterm21,sterm22
      sterm2=sterm21*sterm22
!
!     sK=0.5d0*(1.0d0/(sRc-sh)/sqrt(1.0d0+(shd(sh_1,sh1,sdx)**2.0d0))&
!     &+shd2(sh_1,sh,sh1,sdx)/(1.0d0+shd(sh_1,sh1,sdx)**2.0d0)&
!     &**(3.0d0/2.0d0))
!     sK=cur(sx_2,sx_1,sx,sx1,sx2,sh_2,sh_1,sh,sh1,sh2,sRc,sdx)
      sK=(eq_K1(sRc,sx,sx1,sx2,sh,sh1,sh2)&
      &+eq_K1(sRc,sx_1,sx,sx1,sh_1,sh,sh1))/2.0d0
!
      sterm3=svl/Boltzmann/sTl*(sPv-ssigma*sK+hamaker/sh**3.0d0-sPsat)
      sterm4=log((sTv/sTl)**4.0d0*sPsat/sPv)
!
      sterm51=1.0d0
      sterm52=1.0d0
      do is=1,3
        sterm51=sterm51*exp(-C_theta(is)/2.0d0/sTv)/&
     &(1.0d0-exp(-C_theta(is)/sTv))
        sterm52=sterm52*exp(-C_theta(is)/2.0d0/sTl)/&
     &(1.0d0-exp(-C_theta(is)/sTl))
      end do
      sterm5=log(sterm51/sterm52)
!
      dSn1=sterm1+sterm2+sterm3+sterm4+sterm5
!     print *,'sterm1,sterm2,sterm3,sterm4,sterm5= '
!     print *,sterm1,sterm2,sterm3,sterm4,sterm5
    end function
!
!
    function dSn1K(sTl,sTv,svl,sPv,sRc,sK0,sK1,sh,sdx)
      double precision, dimension(3)::C_theta
      double precision :: sterm1
```

```
      double precision :: sterm2,sterm21,sterm22
      double precision :: sterm3
      double precision :: sterm4
      double precision :: sterm5,sterm51,sterm52
      double precision :: sTl,sTv,svl,sPv,sPsat
      double precision :: sTc,sTf
      double precision :: hamaker,sK,Boltzmann,dsn1
      double precision :: sRc,sdx,Kn
      double precision :: shd,shd2
      double precision :: sK0,sK1,sh
!
      Boltzmann=1.3805d-23
!     hamaker=-1.28d-20
!     hamaker=-1.28d-21
      hamaker=-f_Hamaker(sh)
      sTc=sTl-273.15d0
      sTf=9.0d0/5.0d0*sTc+32.0d0
      ssigma=SURFT(sTf)*4.448d0/0.305d0
      C_theta(1)=2290.0d0
      C_theta(2)=5160.0d0
      C_theta(3)=5360.0d0
      sPsat=psattn(sTl)*1.0d6
!
      sterm1=4.0d0*(1.0d0-sTv/sTl)
!
      sterm21=(1.0d0/sTv-1.0d0/sTl)
      sterm22=0.0d0
      do is=1,3
        sterm22=sterm22+C_theta(is)/2.0d0+C_theta(is)/&
     &(exp(C_theta(is)/sTv)-1.0d0)
!       print *,is,C_theta(is)/2.0d0,exp(C_theta(is)/sTv),sTv
      end do
!     print *,'sterm21,sterm22= '
!     print *,sterm21,sterm22
      sterm2=sterm21*sterm22
!
!     sK=0.5d0*(1.0d0/(sRc-sh)/sqrt(1.0d0+(shd(sh_1,sh1,sdx)**2.0d0))&
!     &+shd2(sh_1,sh,sh1,sdx)/(1.0d0+shd(sh_1,sh1,sdx)**2.0d0)&
!     &**(3.0d0/2.0d0))
!     sK=cur(sx_2,sx_1,sx,sx1,sx2,sh_2,sh_1,sh,sh1,sh2,sRc,sdx)
      sK=(sK1+sK0)/2.0d0
!
      sterm3=svl/Boltzmann/sTl*(sPv-ssigma*sK+hamaker/sh**3.0d0-sPsat)
      sterm4=log((sTv/sTl)**4.0d0*sPsat/sPv)
!
      sterm51=1.0d0
      sterm52=1.0d0
      do is=1,3
        sterm51=sterm51*exp(-C_theta(is)/2.0d0/sTv)/&
     &(1.0d0-exp(-C_theta(is)/sTv))
```

```
            sterm52=sterm52*exp(-C_theta(is)/2.0d0/sTl)/&
           &(1.0d0-exp(-C_theta(is)/sTl))
            end do
            sterm5=log(sterm51/sterm52)
!
            dSn1K=sterm1+sterm2+sterm3+sterm4+sterm5
!           print *,'sterm1,sterm2,sterm3,sterm4,sterm5= '
!           print *,sterm1,sterm2,sterm3,sterm4,sterm5
          end function
!
!
!
!
          function eq_f1_K1(sTl,sTv,svl,sPv,sRc,sK0,sK1,sh,sdx,ssc,sAin)
            double precision:: sTl,sTv,svl,sPv,sRc,sK0,sK1,sh,sdx,ssc
            double precision:: sterm1,sterm2
            double precision:: Boltzmann,Hamaker,sTc,sTf,ssigma,sAin
            Boltzmann=1.3805d-23
!           Hamaker=-1.28d-20
            Hamaker=-1.28d-21
            Hamaker=f_Hamaker(sh)
            sTc=sTl-273.15d0
            sTf=9.0d0/5.0d0*sTc+32.0d0
            ssigma=SURFT(sTf)*4.448d0/0.305d0
            sterm1=C_F1(sh,sRc,sTl)*ssigma/sdx
!
            sterm2=ssc*cosh(-dSn1K(sTl,sTv,svl,sPv,sRc,sK0,sK1,sh,sdx))&
            &*svl*ssigma*sAin/Boltzmann/sTl/2.0d0
!
            eq_f1_K1=sterm1+sterm2
          end function
!
          function gd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)
            double precision:: sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1
            double precision:: sterm1,sterm2,sterm3,sterm4,sterm5,sterm6
            sterm1=(sui-sui1)*sgx_1/(sui_1-sui0)/(sui_1-sui1)
            sterm2=(sui-sui0)*sgx_1/(sui_1-sui0)/(sui_1-sui1)
            sterm3=(sui-sui1)*sgx0/(sui0-sui_1)/(sui0-sui1)
            sterm4=(sui-sui_1)*sgx0/(sui0-sui_1)/(sui0-sui1)
            sterm5=(sui-sui0)*sgx1/(sui1-sui_1)/(sui1-sui0)
            sterm6=(sui-sui_1)*sgx1/(sui1-sui_1)/(sui1-sui0)
            gd=sterm1+sterm2+sterm3+sterm4+sterm5+sterm6
          end function
!
          function hd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1)
            double precision:: sui_1,sui0,sui1,sui,shy_1,shy0,shy1
            double precision:: sterm1,sterm2,sterm3,sterm4,sterm5,sterm6
            sterm1=(sui-sui1)*shy_1/(sui_1-sui0)/(sui_1-sui1)
            sterm2=(sui-sui0)*shy_1/(sui_1-sui0)/(sui_1-sui1)
            sterm3=(sui-sui1)*shy0/(sui0-sui_1)/(sui0-sui1)
```

```
      sterm4=(sui-sui_1)*shy0/(sui0-sui_1)/(sui0-sui1)
      sterm5=(sui-sui0)*shy1/(sui1-sui_1)/(sui1-sui0)
      sterm6=(sui-sui_1)*shy1/(sui1-sui_1)/(sui1-sui0)
      hd=sterm1+sterm2+sterm3+sterm4+sterm5+sterm6
    end function
!
    function gdd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)
      double precision:: sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1
      double precision:: sterm1,sterm2,sterm3
      sterm1=sgx_1/(sui_1-sui0)/(sui_1-sui1)
      sterm2=sgx0/(sui0-sui_1)/(sui0-sui1)
      sterm3=sgx1/(sui1-sui_1)/(sui1-sui0)
      gdd=2.0d0*(sterm1+sterm2+sterm3)
    end function
!
    function hdd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1)
      double precision:: sui_1,sui0,sui1,sui,shy_1,shy0,shy1
      double precision:: sterm1,sterm2,sterm3
      sterm1=shy_1/(sui_1-sui0)/(sui_1-sui1)
      sterm2=shy0/(sui0-sui_1)/(sui0-sui1)
      sterm3=shy1/(sui1-sui_1)/(sui1-sui0)
      hdd=2.0d0*(sterm1+sterm2+sterm3)
    end function
!
    function eq_K_Lag(sRc,sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1,shy_1,shy0,shy1)
      double precision:: sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1,shy_1,shy0,shy1
      double precision:: sterm1,sterm2,sterm3,sterm4,sRc
!     eq_K_Lag=(gd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)&
!     &*hdd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1)&
!     &-gdd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)&
!     &*hd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1))&
!     &/(gd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)&
!     &+hd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1))**(3.0d0/2.0d0)
      sterm1=gd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)
      sterm2=hdd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1)
      sterm3=hd(sui_1,sui0,sui1,sui,shy_1,shy0,shy1)
      sterm4=gdd(sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1)
!     print *,'sterm1,sterm2,sterm3,sterm4= ',sterm1,sterm2,sterm3,sterm4
      eq_K_Lag=(sterm1*sterm2-sterm3*sterm4)&
      &/(sterm1**2.0d0+sterm3**2.0d0)**(3.0d0/2.0d0)+(1.0d0/sRc)
    end function
!
    function eq_K_Lag_d(sRc,sui_1,sui0,sui1,sui,sgx_1,sgx0,&
    &sgx1,shy_1,shy0,shy1)
      double precision:: sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1,shy_1,shy0,shy1
      double precision:: sterm1,sterm2,sterm3,sterm4,sRc
      double precision:: sterm5,sterm6,sterm7,sterm8,sterm9
      double precision:: sterm10,sterm11,sterm12,sterm13,sterm14,sterm15
      double precision:: FFA,FFB,FFC,FFD,FFE,FFF,FFG,FFH,FFI,FFJ
      double precision:: FFA_1,FFC_1,FFK,FFL,FFM,FFE_1,FFA1,FFF1,FFO
```

```
      double precision:: FFP,FFB_1,FFG_1,FFB1,FFH1,FFQ,FFI1
      double precision:: FFI_1,FFJ1
!
      FFA_1=sgx_1/(sui_1-sui0)/(sui_1-sui1)
      FFA=sgx0/(sui0-sui_1)/(sui0-sui1)
      FFA1=sgx1/(sui1-sui0)/(sui1-sui_1)
      FFB_1=shy_1/(sui_1-sui0)/(sui_1-sui1)
      FFB=shy0/(sui0-sui_1)/(sui0-sui1)
      FFB1=shy1/(sui1-sui0)/(sui1-sui_1)
      FFE=FFA*(sui-sui1)
      FFF=FFA*(sui-sui_1)
      FFG=FFB*(sui-sui1)
      FFH=FFB*(sui-sui_1)
!
!
      FFM=(shy_1/(sui_1-sui0)/(sui_1-sui1)+shy0/(sui0-sui_1)/(sui0-sui1)&
      &+shy1/(sui1-sui_1)/(sui1-sui0))*2.0d0
      FFE_1=FFA_1*(sui-sui1)
      FFF1=FFA1*(sui-sui_1)
      FFG_1=FFB_1*(sui-sui1)
      FFH1=FFB1*(sui-sui_1)
      FFQ=2.0d0*sgx_1/(sui_1-sui0)/(sui_1-sui1)+&
     &2.0d0*sgx0/(sui0-sui_1)/(sui0-sui1)&
      &+2.0d0*sgx1/(sui1-sui_1)/(sui1-sui0)
!
!
!
!
      sterm1=2.0d0*(FFE_1+FFE+FFF+FFF1)/(sui1-sui_1)/(sui1-sui0)
      sterm2=FFQ*(sui-sui_1)/(sui1-sui_1)/(sui1-sui0)
      sterm3=(FFE_1+FFE+FFF+FFF1)**2.0d0
      sterm4=(FFG_1+FFG+FFH+FFH1)**2.0d0
      sterm5=(FFE_1+FFE+FFF+FFF1)*FFM
      sterm6=FFQ*(FFG_1+FFG+FFH+FFH1)
      sterm7=(FFG_1+FFG+FFH+FFH1)*(sui-sui_1)/(sui1-sui_1)/(sui1-sui0)
      sterm8=(FFE_1+FFE+FFF+FFF1)**2.0d0
      sterm9=(FFG_1+FFG+FFH+FFH1)**2.0d0
!
      eq_K_Lag_d=(sterm1-sterm2)/(sterm3+sterm4)**(3.0d0/2.0d0)&
      &-3.0d0*(sterm5-sterm6)*sterm7/(sterm8+sterm9)**(5.0d0/2.0d0)
!
!     print *,FFA_1,FFA,FFA1,FFB_1,FFB,FFB1
!     print *,FFE_1,FFE,FFF,FFF1,FFG_1,FFG,FFH,FFH1
!     print *,FFM,FFQ
!     print *,'sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7,sterm8,sterm9= '&
!     &,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7,sterm8,sterm9
    end function
!
    function eq_K_Lag_dold(sRc,sui_1,sui0,sui1,sui,sgx_1,sgx0&
    &,sgx1,shy_1,shy0,shy1)
```

```
      double precision:: sui_1,sui0,sui1,sui,sgx_1,sgx0,sgx1,shy_1,shy0,shy1
      double precision:: sterm1,sterm2,sterm3,sterm4,sRc
      double precision:: sterm5,sterm6,sterm7,sterm8,sterm9,sterm25
      double precision:: FFA,FFB,FFC,FFD,FFE,FFF,FFG,FFH,FFI,FFJ
      FFA=sgx0/(sui0-sui_1)/(sui0-sui1)
      FFB=shy0/(sui0-sui_1)/(sui0-sui1)
      FFC=FFA*(sui-sui1)/(sui0-sui1)
      FFD=FFA*(sui-sui_1)/(sui0-sui1)
      FFE=FFA*(sui-sui1)
      FFF=FFA*(sui-sui_1)
      FFG=FFB*(sui-sui1)
      FFH=FFB*(sui-sui_1)
      FFI=FFB*(sui-sui1)/(sui0-sui1)
      FFJ=FFB*(sui-sui_1)/(sui0-sui1)
!
      sterm1=2.0d0*(-FFA+FFC+FFD)*shy0/(sui0-sui_1)/(sui0-sui1)
      sterm2=2.0d0*(FFE+FFF)*shy0/(sui0-sui_1)/(sui0-sui1)/(sui0-sui1)
      sterm25=2.0d0*sgx0*(FFG+FFH)/(sui0-sui_1)/(sui0-sui1)/(sui0-sui1)
      sterm3=2.0d0*sgx0*(-FFB+FFI+FFJ)/(sui0-sui_1)/(sui0-sui1)
      sterm4=(FFE+FFF)**2.0d0+(FFG+FFH)**2.0d0
      sterm5=2.0d0*(FFE+FFF)*shy0/(sui0-sui_1)/(sui0-sui1)
      sterm6=2.0d0*(FFG+FFH)/(sui0-sui_1)/(sui0-sui1)
      sterm7=2.0d0*(FFE+FFF)*(-FFA+FFC+FFD)
      sterm8=2.0d0*(FFG+FFH)*(-FFB+FFI+FFJ)
      sterm9=2.0d0*((FFE+FFF)**2.0d0+(FFG+FFH)**2.0d0)**(5.0d0/2.0d0)
!     print *,'FFA,FFB,FFC,FFD,FFE,FFF,FFG,FFH,FFI,FFJ= '&
!     ,FFA,FFB,FFC,FFD,FFE,FFF,FFG,FFH,FFI,FFJ
!     print *,'sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7,sterm8,sterm9= '&
!     &,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7,sterm8,sterm9
      eq_K_Lag_dold=(sterm1+sterm2-sterm25-sterm3)/sterm4&
     &-3.0d0*(sterm5-sterm6)&
      &*(2.0d0*sterm7+2.0d0*sterm8)/sterm9
    end function
!
    function dpdxdK(sdK,sh,sh1,sh2,sdx,sTl,sRc)
      double precision:: sdK
      double precision:: sh_1,sh,sh1,sh2,sdx,sTl
      double precision:: sTc,sTf,ssigma,shamaker,sRc,sterm1,sterm2
!     shamaker=1.28d-20
      shamaker=1.28d-21
      shamaker=f_Hamaker(sh)
      sTc=sTl-273.15d0
      sTf=9.0d0/5.0d0*sTc+32.0d0
      ssigma=SURFT(sTf)*4.448d0/0.305d0
      sterm1=ssigma*sdK/sdx
      sterm2=shamaker*(1.0d0/sh1**3.0d0-1.0d0/sh**3.0d0)/sdx
!     dpdx=-ssigma*DK(sh_2,sh_1,sh,sh1,sh2,sRc,sdx)-&
!     &3.0d0*shamaker*shd(sh_1,sh1,sdx)/sh**4.0d0
      dpdxdK=-(sterm1+sterm2)
!     dpdxdK=-sterm1
```

```
      end function
!
      function dSndK(sTl,sTv,svl,sPv,sRc,sK0,sdK,sh,sdx)
        double precision, dimension(3)::C_theta
        double precision :: sterm1
        double precision :: sterm2,sterm21,sterm22
        double precision :: sterm3
        double precision :: sterm4
        double precision :: sterm5,sterm51,sterm52
        double precision :: sTl,sTv,svl,sPv,sPsat
        double precision :: sTc,sTf
        double precision :: hamaker,sK,Boltzmann,dsn1
        double precision :: sRc,sdx,Kn
        double precision :: shd,shd2
        double precision :: sK0,sdK,sh
!
        Boltzmann=1.3805d-23
!       hamaker=-1.28d-20
        hamaker=-1.28d-21
        hamaker=f_Hamaker(sh)
        sTc=sTl-273.15d0
        sTf=9.0d0/5.0d0*sTc+32.0d0
        ssigma=SURFT(sTf)*4.448d0/0.305d0
        C_theta(1)=2290.0d0
        C_theta(2)=5160.0d0
        C_theta(3)=5360.0d0
        sPsat=psattn(sTl)*1.0d6
!
        sterm1=4.0d0*(1.0d0-sTv/sTl)
!
        sterm21=(1.0d0/sTv-1.0d0/sTl)
        sterm22=0.0d0
        do is=1,3
          sterm22=sterm22+C_theta(is)/2.0d0+&
       &C_theta(is)/(exp(C_theta(is)/sTv)-1.0d0)
!         print *,is,C_theta(is)/2.0d0,exp(C_theta(is)/sTv),sTv
        end do
!       print *,'sterm21,sterm22= '
!       print *,sterm21,sterm22
        sterm2=sterm21*sterm22
!
!       sK=0.5d0*(1.0d0/(sRc-sh)/sqrt(1.0d0+(shd(sh_1,sh1,sdx)**2.0d0))&
!       &+shd2(sh_1,sh,sh1,sdx)/(1.0d0+shd(sh_1,sh1,sdx)**2.0d0)&
!       &**(3.0d0/2.0d0))
!       sK=cur(sx_2,sx_1,sx,sx1,sx2,sh_2,sh_1,sh,sh1,sh2,sRc,sdx)
        sK=(sdK+sK0*2.0d0)/2.0d0
!
        sterm3=svl/Boltzmann/sTl*(sPv-ssigma*sK+hamaker/sh**3.0d0-sPsat)
!       print *,sPv,-ssigma*sK,+hamaker/sh**3.0d0,-sPsat
        sterm4=log((sTv/sTl)**4.0d0*sPsat/sPv)
```

```
!
              sterm51=1.0d0
              sterm52=1.0d0
              do is=1,3
                 sterm51=sterm51*exp(-C_theta(is)/2.0d0/sTv)/&
             (1.0d0-exp(-C_theta(is)/sTv))
                 sterm52=sterm52*exp(-C_theta(is)/2.0d0/sTl)/&
             &(1.0d0-exp(-C_theta(is)/sTl))
              end do
              sterm5=log(sterm51/sterm52)
!
              dSndK=sterm1+sterm2+sterm3+sterm4+sterm5
!             print *,'sterm1,sterm2,sterm3,sterm4,sterm5= '
!             print *,sterm1,sterm2,sterm3,sterm4,sterm5
            end function
!
            function dpdxdKw(sdK,sh,sh1,sh2,sdx,sTl,sRc,sll)
              double precision:: sdK
              double precision:: sh_1,sh,sh1,sh2,sdx,sTl
              double precision:: sTc,sTf,ssigma,shamaker,sRc,sterm1,sterm2
              integer :: sll
!             shamaker=1.28d-20
              shamaker=1.28d-21
              shameker=f_Hamaker(sh)
              sTc=sTl-273.15d0
              sTf=9.0d0/5.0d0*sTc+32.0d0
              ssigma=SURFT(sTf)*4.448d0/0.305d0
              sterm1=ssigma*sdK/sdx
              sterm2=shamaker*(1.0d0/sh1**3.0d0-1.0d0/sh**3.0d0)/sdx
!             dpdx=-ssigma*DK(sh_2,sh_1,sh,sh1,sh2,sRc,sdx)-&
!             &3.0d0*shamaker*shd(sh_1,sh1,sdx)/sh**4.0d0
              dpdxdKw=-(sterm1+sterm2)
!             dpdxdK=-sterm1
              write(11,'(1I10,10E22.10)') sll,dpdxdKw,-sterm1,-sterm2
            end function
!
            subroutine fKend_1(sh0,sKend,sKend_1,sdx,sTl,sRc&
            &,sMass,sAvgadro,sAmass,sBoltzmann,sTv,sPv)
              double precision :: sh0,sKend,sKend_1,sdx,sTl
              double precision :: sRc,sMass,sAvgadro,sAmass
              double precision :: sBoltzmann,sTv,sPv
              double precision :: sTc,sTf,ssigma,sHamaker
              double precision :: seq,old_seq,seq0,seq1,seqm
              double precision :: oldsh_1
              double precision :: sh_1,sh_1_n,sh_1_p,sh_1_m
              double precision :: sfd,ssc,sevap1,stheta,spi
              double precision :: sAin,sAn,sdK,sM0,svl
              integer :: iis
!
              sTc=sTl-273.15d0
```

```
      sTf=9.0d0/5.0d0*sTc+32.0d0
      ssigma=SURFT(sTf)*4.448d0/0.305d0
      sHamaker=1.28d-21
      sHamaker=f_Hamaker(sh0)
      spi=3.141592654d0
!
!
      sh_1=sh0+1.0d-10
      fd=(sh0-sh_1)/sdx
      sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/sRc
      sdK=sKend-sKend_1
      sPi0=sHamaker/sh0**3.0d0
      sPi_1=sHamaker/sh_1**3.0d0
!
      sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/sdx-(sPi0-sPi_1)/sdx)
      sPsat=psattn(sTl)*1.0d6
      ssc=2.0d0*sMass/sAvgadro*sPsat/sqrt(2.0d0*spi*sAmass*sBoltzmann*sTl)
      svl=VFTSI(sTl)
      svl=svl*sMass/sAvgadro
      sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,&
     &sdK,(sh_1+sh0)/2.0d0,sdx))
      stheta=atan((sh0-sh_1)/sdx)
      sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
      sAn=2.0d0*spi*sRc*sdx
      seq=sM0/sAin-sevap1
!
!
      do iis=1,1000000
        old_seq=seq
        if (seq>=0.0d0) then
          oldsh_1=sh_1
              sh_1=sh_1-0.1d-11
!         h1=h1-Rc/100000.0d0
          fd=(sh0-sh_1)/sdx
          sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/sRc
          sdK=sKend-sKend_1
          sPi_1=sHamaker/sh_1**3.0d0
!
          sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/sdx&
        -(sPi0-sPi_1)/sdx)
          sPsat=psattn(sTl)*1.0d6
          sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,sdK,&
        &(sh_1+sh0)/2.0d0,sdx))
          stheta=atan((sh0-sh_1)/sdx)
          sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
          seq=sM0/sAin-sevap1
!
          if (seq<0.0d0) then
            sh_1_p=oldsh_1
```

```
                sh_1_n=sh_1
                print *,'1'
                    exit
                end if
            end if
            if (seq<0.0d0) then
              oldsh_1=sh_1
!             h1=h1+Rc/100000.0d0
                  sh_1=sh_1+0.1d-11
              fd=(sh0-sh_1)/sdx
              sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/sRc
              sdK=sKend-sKend_1
              sPi0=sHamaker/sh0**3.0d0
              sPi_1=sHamaker/sh_1**3.0d0
!
              sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/
            &sdx-(sPi0-sPi_1)/sdx)
              sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,sdK,&
            &(sh_1+sh0)/2.0d0,sdx))
              stheta=atan((sh0-sh_1)/sdx)
              sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
              seq=sM0/sAin-sevap1
              if (seq>=0.0d0) then
                sh_1_n=sh_1
                sh_1_p=oldsh_1
                print *,'2'
                exit
              end if
            end if
                print *,'iis,sh_1,seq,eq_K,sKend= '
                print *,iis,sh_1,seq,eq_K,sKend
!               pause
          end do
          print *,'iis,sh_1,seq,eq_K,sKend= '
          print *,iis,sh_1,seq,eq_K,sKend
!
          sh_1=sh_1_n
          fd=(sh0-sh_1)/sdx
          sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/sRc
          sdK=sKend-sKend_1
          sPi_1=sHamaker/sh_1**3.0d0
!
          sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/sdx-(sPi0-sPi_1)/sdx)
          sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,&
          &sdK,(sh_1+sh0)/2.0d0,sdx))
          stheta=atan((sh0-sh_1)/sdx)
          sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
          seq0=sM0/sAin-sevap1
!
```

```
!
!
              sh_1=sh_1_p
              fd=(sh0-sh_1)/sdx
              sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/sRc
              sdK=sKend-sKend_1
              sPi_1=sHamaker/sh_1**3.0d0
!
              sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/sdx-(sPi0-sPi_1)/sdx)
              sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,sdK,&
            &(sh_1+sh0)/2.0d0,sdx))
              stheta=atan((sh0-sh_1)/sdx)
              sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
              seq1=sM0/sAin-sevap1
!
!   print *,'sh_1_n,sh_1_p,seq0,seq1= '
!   print *,sh_1_n,sh_1_p,seq0,seq1
!
!             pause
!
              do iis=1,200
                    sh_1_m=sh_1_p-seq1*(sh_1_p-sh_1_n)/(seq1-seq0)
                    sh_1=sh_1_m
                fd=(sh0-sh_1)/sdx
                sKend_1=-fd/(1.0d0+fd*fd)**(2.0d0/3.0d0)+1.0/(sRc-sh_1)
                sdK=sKend-sKend_1
                sPi_1=sHamaker/sh_1**3.0d0
!
                sM0=C_F1(sh_1,sRc,sTl)*(-ssigma*(sKend-sKend_1)/&
              &sdx-(sPi0-sPi_1)/sdx)
                sevap1=ssc*sinh(dSndK(sTl,sTv,svl,sPv,sRc,sKend,sdK,&
            &(sh_1+sh0)/2.0d0,sdx))
                stheta=atan((sh0-sh_1)/sdx)
                sAin=2.0d0*spi*(sRc-(sh0+sh_1)/2.0d0)/cos(stheta)*sdx
                seqm=sM0/sAin-sevap1
!
!
                if (abs((sh_1_m-sh_1_p)/sh_1_m)<1.0d-8) then
                  if (abs(seqm)<1.0d-12) then
                    exit
                  end if
                end if
!
                if ((seqm*seq1)<0.0d0) then
                  h1_n=h1_p
                  seq0=seq1
                    end if
                sh_1_p=sh_1_m
                seq1=seqm
```

```
!      print *,'iis,sh_1_m,seqm= '
!           print *,iis,sh_1_m,seqm,sM0/sAin,sevap1
!                      pause
              end do
!   print *,'iis,sh_1_m,seqm= '
!   print *,iis,sh_1_m,seqm
!             pause
            end subroutine
!
            function Enxmass2(sRc,sTw,sh,sTl,sdx,sdK,sh0,sh2)
              double precision :: sRc,sTw,sh,sTl,sr,smu,sTc,spi,sCp,svl
              double precision :: sdx,sdK,sh0,sh2
              double precision :: sterm1,sterm2,sterm3,sterm4
              double precision :: sterm5,sterm6,sterm7,sterm8,sterm9,sterm10
!             print *,'Enxmass stars'
              spi=3.141592654d0
              svl=VFTSI(sTl)
              sTc=sTl-273.15
              sr=sRc-sh
              sCp=CPVT3N(svl,sTl)*1.0d-3
              smu=VISCOS(svl,sTc)*1.0d-6
!             print *,sCp,smu
!             write(6,*) sCp,smu
!             pause
              sterm1=4.0*sRc**4*log(sr)*sTw-4.0*sRc**4*sTw*log(sRc)+&
              &8.0*sRc**2*sr**2*log(sRc)*sTw
              sterm2=12.0*sRc**2*sr**2*sTl-12.0*sRc**2*sr**2*sTw-&
              &3.0*sRc**4*sTl+3.0*sRc**4*sTw
              sterm3=-8.0*sRc**2*sr**2*log(sr)*sTw-16.0*sr**4*(log(sRc))**2*sTl
              sterm4=8.0*sr**2*sRc**2*sTl*log(sRc)-16.0*sr**4*sTl*(log(sr))**2-&
              &9.0*sr**4*sTl
              sterm5=9.0*sr**4*sTw+8.0*sr**4*log(sRc)*sTw-8.0*sr**4*log(sr)*sTw
              sterm6=-20.0*sr**4*sTl*log(sRc)+32.0*sr**4*sTl*log(sRc)*log(sr)
              sterm7=20.0*sr**4*sTl*log(sr)-8.0*sr**2*log(sr)*sRc**2*sTl
              sterm8=1.0/32.0*spi/svl/smu/log(sRc/sr)*sCp*&
              &(sterm1+sterm2+sterm3+sterm4+sterm5+sterm6+sterm7)
              sterm9=dpdxdK(sdK,sh0,sh,sh2,sdx,sTl,sRc)
              sterm10=sterm8*sterm9
              write(20,'(11E22.10)')   sTl,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6&
              &,sterm7,sterm8,sterm9,sterm10
!             pause
!
!             print *,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6
!             write(6,*) sterm10,sterm11,sterm12
              Enxmass2=sterm10
!             write(6,*) 'Enxmass ends'
!             pause
            end function
!
            function d_Enxmass(sRc,sTw,sh,sTl,sdx,sdK,sh0,sh2)
              double precision :: sRc,sTw,sh,sTl,sr,smu,sTc,spi,sCp,svl
```

```
      double precision :: sdx,sdK,sh0,sh2
      double precision :: sterm1,sterm2,sterm3,sterm4
      double precision :: sterm5,sterm6,sterm7
      double precision :: sterm10,sterm11,sterm12
      spi=3.141592654d0
      svl=VFTSI(sTl)
      sTc=sTl-273.15
      sr=sRc-sh
      sCp=CPVT3N(svl,sTl)*1.0d-3
      smu=VISCOS(svl,sTc)*1.0d-6
!
      sterm1=sRc**4*log(sRc)-4.0*sRc**2*log(sRc)*sr**2+3.0*sr**4*log(sRc)
      sterm2=4.0*sr**4*(log(sRc))**2-4.0*sr**4*log(sRc)*log(sr)
      sterm3=-1.0/8.0*sCp*spi/smu/svl/log(sr/sRc)*&
     &dpdxdK(sdK,sh0,sh,sh2,sdx,sTl,sRc)
      sterm4=sterm3*(sterm1+sterm2)
      write(6,*)   sterm1,sterm2,sterm3,sterm4
!
      sterm5=-sRc+sRc*log(sRc)+sr-sr*log(sr)
      sterm6=-1.0*sCp/svl/log(sr/sRc)
      sterm7=sterm5*sterm6
      write(6,*) sterm5,sterm6,sterm7
!
      d_Enxmass=sterm4+sterm7
    end function
!
    function Enxmass(sRc,sTw,sh,sTl,sdx,sdK,sh0,sh2)
      double precision :: sRc,sTw,sh,sTl,sr,smu,sTc,spi,sCp,svl
      double precision :: sdx,sdK,sh0,sh2
      double precision :: sterm1,sterm2,sterm3,sterm4,sterm5
      double precision :: sterm6,sterm7,sterm8,sterm9,sterm10,sterm11
!     print *,'Enxmass stars'
      spi=3.141592654d0
      svl=VFTSI(sTl)
      sTc=sTl-273.15
      sr=sRc-sh
!     sCp=CPVT3N(svl,sTw-1.0d0)*1.0d-3
      sCp=4.2246d3
      smu=VISCOS(svl,sTc)*1.0d-6
!     print *,sCp,smu
!     write(6,*) sCp,smu
!     pause
      sterm1=4.0*sRc**4*log(sr)*sTw+12.0*sRc**2*sr**2*sTl+&
      &8.0*sRc**2*sr**2*log(sRc)*sTw
      sterm2=-12.0*sRc**2*sr**2*sTw+3.0*sRc**4*sTw-&
      4.0*sRc**4*sTw*log(sRc)
      sterm3=-8.0*sRc**2*sr**2*log(sr)*sTw-3.0*sRc**4*sTl
      sterm4=8.0*sr**2*sRc**2*sTl*log(sRc)-16.0*sr**4*sTl*(log(sRc))**2
      sterm5=-8.0*sr**4*log(sr)*sTw+8.0*sr**4*log(sRc)*sTw-9.0*sr**4*sTl
      sterm6=-20.0*sr**4*sTl*log(sRc)-8.0*sr**2*log(sr)*sRc**2*sTl
```

```
            sterm7=32.0*sr**4*sTl*log(sRc)*log(sr)+9.0*sr**4*sTw
            sterm8=-16.0*sr**4*sTl*(log(sr))**2+20.0*sr**4*sTl*log(sr)
            sterm9=1.0/32.0*spi/svl/smu/log(sRc/sr)*sCp*&
            &(sterm1+sterm2+sterm3+sterm4+sterm5+sterm6+sterm7+sterm8)
            sterm10=dpdxdK(sdK,sh0,sh,sh2,sdx,sTl,sRc)
            sterm11=sterm9*sterm10
!           write(20,'(20E22.10)')  sTl,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6&
!           &,sterm7,sterm8,sterm9,sterm10,sterm11,svl,smu,sCp
!           pause
!
!           print *,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6
!           write(6,*) sterm10,sterm11,sterm12
            Enxmass=sterm11
!           write(6,*) 'Enxmass ends'
!           pause
          end function
!
          function Enxmassp(sRc,sTw,sh,sTl,sdx,sdK,sh0,sh2)
            double precision :: sRc,sTw,sh,sTl,sr,smu,sTc,spi,sCp,svl
            double precision :: sdx,sdK,sh0,sh2
            double precision :: sterm1,sterm2,sterm3,sterm4,sterm5
            double precision :: sterm6,sterm7,sterm8,sterm9,sterm10,sterm11
!           print *,'Enxmass stars'
            spi=3.141592654d0
            svl=VFTSI(sTl)
            sTc=sTl-273.15
            sr=sRc-sh
!           sCp=CPVT3N(svl,sTw-1.0d0)*1.0d-3
            sCp=4.2246d3
            smu=VISCOS(svl,sTc)*1.0d-6
!           print *,sCp,smu
!           write(6,*) sCp,smu
!           pause
!           sterm1=4.0*sRc**4*log(sr)*sTw-32.0*sr**4*sTl*log(sRc)*log(sr)-&
!           &8.0*sRc**2*sr**2*log(sr)*sTl
!           sterm2=8.0*sr**2*sRc**2*sTl*log(sRc)-17.0*sr**4*sTl-&
!           36.0*sr**4*sTl*log(sRc)
!           sterm3=36.0*sr**4*log(sr)*sTl-32.0*sr**4*sTl*(log(sr))**2
!           sterm4=-16.0*sr**4*log(sr)*sTw+16.0*sr**4*sTw*log(sRc)
!           sterm5=-32.0*sr**4*(log(sRc))**2*sTl-16.0*sRc**2*sr**2*log(sr)*sTw&
!           &+16.0*sRc**2*sr**2*sTw*log(sRc)
!           sterm6=-4.0*sRc**4*sTw*log(sRc)-20.0*sRc**2*sr**2*sTw
!           sterm7=20.0*sRc**2*sr**2*sTl+3.0*sRc**4*sTl
!           sterm8=-3.0*sRc**4*sTl+17.0*sr**4*sTw
!
            sterm1=-32.0*sr**4*log(sRc)*sTl*log(sr)+3.0*sRc**4*(sTl-sTw)
            sterm2=9.0*sr**4*(sTl-sTw)+4.0*sRc**4*sTw*&
           &log(sRc)-12.0*sRc**2*sr**2*sTl
            sterm3=12.0*sRc**2*sr**2*sTw-4.0*sRc**4*sTw*log(sr)&
           &+16.0*sr**4*(log(sr))**2*sTl
```

```
      sterm4=-20.0*sr**4*log(sr)*sTl+20.0*sr**4*log(sRc)*sTl&
     &+16.0*sr**4*(log(sRc))**2*sTl
      sterm5=8.0*sr**4*sTw*log(sr)-8.0*sr**4*log(sRc)*sTw&
     &+8.0*sRc**2*sr**2*sTw*log(sr)
      sterm6=-8.0*sRc**2*sr**2*log(sRc)*sTw+8.0*sr**2*log(sr)*sRc**2*sTl
      sterm7=-8.0*sr**2*sRc**2*log(sRc)*sTl
      sterm9=1.0/32.0*spi/svl/smu/log(sr/sRc)*sCp*&
      &(sterm1+sterm2+sterm3+sterm4+sterm5+sterm6+sterm7)
!
      sterm10=dpdxdK(sdK,sh0,sh,sh2,sdx,sTl,sRc)
      sterm11=sterm9*sterm10
!     write(20,'(20E22.10)')  sterm1,sterm2,sterm3,sterm4,sterm5,sterm6&
!     &,sterm7,sterm9,sterm10,sterm11
!     pause
      Enxmassp=sterm11
!     write(6,*) 'Enxmass ends'
!     pause
    end function
!
    function Enxmasss(sRc,sTw,sh,sTl,sdx,sdK,sh0,sh2)
      double precision :: sRc,sTw,sh,sTl,sr,smu,sTc,spi,sCp,svl
      double precision :: sdx,sdK,sh0,sh2
      double precision :: sterm20,sterm21,sterm22,sterm23,sterm24,sterm25
      double precision :: sterm26
!     print *,'Enxmass stars'
      spi=3.141592654d0
      svl=VFTSI(sTl)
      sTc=sTl-273.15
      sr=sRc-sh
!     sCp=CPVT3N(svl,sTw-1.0d0)*1.0d-3
      sCp=4.2246d3
      smu=VISCOS(svl,sTc)*1.0d-6
!     pause
!
!     sterm20=2.0*sr**3*(log(sRc))**2*sTl+2.0*sr**3*sTl*(log(sr))**2
!     sterm21=-2.0*sr**3*sTl*log(sr)-4.0*sr**3*sTl*log(sRc)*log(sr)
!     sterm22=sr**3*log(sr)*sTw-sr**3*log(sRc)*sTw
!     sterm23=2.0*sr**3*sTl*log(sRc)+sr**3*sTl
!     sterm24=-sr**3*sTw-sRc**2*sr*log(sRc)*sTw
!     sterm25=sRc**2*sr*sTw-sRc**2*sr*sTl+sRc**2*sr*log(sr)*sTw
!     sterm26=-1.0/2.0*spi/svl/smu/log(sr/sRc)*sCp*&
!     &(sterm20+sterm21+sterm22+sterm23+sterm24+sterm25)
!
      sterm20=16.0*sRc**2*(sr*sTl-sr*sTw+sr*log(sRc)*sTw-sr*sTw*log(sr))
      sterm21=2.0*log(sr)*sTl-sTl+sTw-2.0*(log(sr))**2*sTl
      sterm22=4.0*log(sRc)*log(sr)*sTl-2.0*log(sRc)*sTl
      sterm23=-2.0*(log(sRc))**2*sTl+log(sRc)*sTw-log(sr)*sTw
      sterm24=16.0*sr**3*(sterm21+sterm22+sterm23)
      sterm26=1.0/32.0*spi/svl/smu/log(sr/sRc)*sCp*&
      &(sterm20+sterm24)
```

```
!               print *,sterm21,sterm21,sterm22,sterm23,sterm24,sterm26
!               write(6,*) sterm10,sterm11,sterm12
!               write(21,'(20E22.10)')   sterm20,sterm21,sterm22,sterm23,sterm24,sterm26
                Enxmasss=sterm26
!               write(6,*) 'Enxmass ends'
!               pause
              end function
!
              function f_Hamaker(sh)
                double precision:: sh,ff
                if (sh>1.0d-9) then
                  ff=5.09926d-1*sh**3-9.86698d-8*sh**2-2.98631d-15*sh+&
                       &7.66877d-21
                else if ((sh<=1.0d-9)*(sh>5.0d-10)) then
                  ff=-3.19318d15*sh**4+1.22765d7*sh**3-1.78584d-2*sh**2+&
                       &+1.17050d-11*sh+4.66154d-21
                else
                  ff=-2.67125d17*sh**4+4.10258d8*sh**3-2.42859d-1*sh**2+&
                       &6.90134d-11*sh-9.94700d-22
                end if
                f_Hamaker=ff
              end function
!
              function f_evap(sK,sPv,sT)
                double precision :: sK,sPv,sT
                double precision :: sTb,sTe,sTw(7),sPvb(7),sPve(7)
                double precision :: sevap1,sevap2,sevap3,sevap4
                double precision :: sevap0,sevap5,sevap6
                double precision :: sa1,sa2,sa3
                double precision :: sb1,sb2,sb3
                integer :: num_Tb,num_Te,num_Pvb,num_Pve
!
!               print *,sK,sPv,sT
!               pause
!
                do iis=1,7
                  sTw(iis)=423.15d0-float(iis-1)*10.0d0
!                       print *,sTw(iis)
                end do
!
!               if (sT<sTw(1)) then
                  if (sT>sTw(2)) then
                          num_Tb=1
                          num_Te=2
                        else if (sT>sTw(3)) then
                          num_Tb=2
                          num_Te=3
                  else if (sT>sTw(4)) then
                          num_Tb=3
                          num_Te=4
                        else if (sT>sTw(5)) then
```

```
                    num_Tb=4
                    num_Te=5
                  else if (sT>sTw(6)) then
                    num_Tb=5
                    num_Te=6
                  else if (sT>sTw(7)) then
                    num_Tb=6
                    num_Te=7
                  end if
!                 print *,'damepo!1'
!               end if
!
!               print *,num_Tb,num_Te
!               pause
!
                do iis=1,6
                  sPvb(iis)=psattn(sTw(num_Tb))*1.0d6-float(iis-1)*5.0d3
!                 print *,sPvb(iis)
                end do
!
!               if (sPv<sPvb(1)) then
                  if (sPv>sPvb(2)) then
                    num_Pvb=1
                  else if (sPv>sPvb(3)) then
                    num_Pvb=2
                  else if (sPv>sPvb(4)) then
                    num_Pvb=3
                  else if (sPv>sPvb(5)) then
                    num_Pvb=4
                  else if (sPv>sPvb(6)) then
                    num_Pvb=5
                  end if
!                 print *,'damepo!2'
!               end if
!
!               print *,num_Pvb
!
                do iis=1,6
                  sPve(iis)=psattn(sTw(num_Te))*1.0d6-float(iis-1)*5.0d3
!                 print *,sPve(iis)
                end do
!
!               if (sPv<sPve(1)) then
                  if (sPv>sPve(2)) then
                    num_Pve=1
                  else if (sPv>sPve(3)) then
                    num_Pve=2
                  else if (sPv>sPve(4)) then
                    num_Pve=3
                  else if (sPv>sPve(5)) then
                    num_Pve=4
```

```
            else if (sPv>sPve(6)) then
                num_Pve=5
            end if
!           print *,'damepo!2'
!         end if
!         print *,num_Pve
!         pause
!
          sevap1=fK_evap(num_Tb,num_Pvb,sK)
          sevap2=fK_evap(num_Tb,num_Pvb+1,sK)
          sevap3=fK_evap(num_Te,num_Pve,sK)
          sevap4=fK_evap(num_Te,num_Pve+1,sK)
!         print *,sevap1,sevap2,sevap3,sevap4
!         pause
!
          call f_P(sPvb(num_Pvb),sevap1,sPvb(num_Pvb+1),sevap2,sa1,sb1)
          call f_P(sPve(num_Pve),sevap3,sPve(num_Pve+1),sevap4,sa2,sb2)
!
          sevap5=sa1*sPv+sb1
          sevap6=sa2*sPv+sb2
!         print *,sevap5,sevap6
!         pause
!
          call f_P(sTw(num_Tb),sevap5,sTw(num_Te),sevap6,sa3,sb3)
          sevap0=sa3*sT+sb3
!         print *,sTw(num_Tb),sevap5,sTw(num_Te),sevap6,sa3,sb3
!         print *,sevap0
!         pause
!
          f_evap=sevap0
!
        end function
!
!
        subroutine f_P(sx0,sy0,sx1,sy1,sa,sb)
          double precision :: sx0,sy0,sx1,sy1,sa,sb
          sa=(sy0-sy1)/(sx0-sx1)
          sb=-sy0*sx1/(sx0-sx1)-sy1*sx0/(sx1-sx0)
        end subroutine
!
        function fK_evap(num_T,num_Pv,sK)
          double precision :: sK
          integer :: num_T,num_Pv
          if (num_T==1) then
            select case (num_Pv)
                case (1)
                  sff=2.204e-11*sK**2+3.80d-6*sK
                case (2)
                  sff=2.901e-11*sK**2+4.782d-6*sK
                case (3)
                  sff=3.646e-11*sK**2+6.895d-6*sK
```

```
      case (4)
              sff=4.432e-11*sK**2+8.680d-6*sK
            case (5)
              sff=5.260e-11*sK**2+10.39d-6*sK
            case default
              sff=5.778e-11*sK**2+12.15d-6*sK
          end select
else if(num_T==2) then
  select case (num_Pv)
            case (1)
              sff=2.203e-11*sK**2+2.411d-7*sK
            case (2)
              sff=3.422e-11*sK**2-1.749d-6*sK
            case (3)
              sff=4.043e-11*sK**2+1.418d-6*sK
      case (4)
              sff=5.116e-11*sK**2+1.386d-6*sK
            case (5)
              sff=5.081e-11*sK**2+9.395d-6*sK
            case default
              sff=6.028e-11*sK**2+10.55d-6*sK
          end select
else if(num_T==3) then
  select case (num_Pv)
            case (1)
              sff=1.645e-11*sK**2+4.625d-7*sK
            case (2)
              sff=2.254e-11*sK**2+4.260d-6*sK
            case (3)
              sff=3.066e-11*sK**2+6.524d-6*sK
      case (4)
              sff=3.909e-11*sK**2+8.642d-6*sK
            case (5)
              sff=4.785e-11*sK**2+10.69d-6*sK
            case default
              sff=5.821e-11*sK**2+12.04d-6*sK
          end select
else if(num_T==4) then
  select case (num_Pv)
            case (1)
              sff=1.352e-11*sK**2+6.077d-7*sK
            case (2)
              sff=2.008e-11*sK**2+4.632d-6*sK
            case (3)
              sff=2.900e-11*sK**2+7.072d-6*sK
      case (4)
              sff=3.802e-11*sK**2+9.474d-6*sK
            case (5)
              sff=4.719e-11*sK**2+12.04d-6*sK
            case default
              sff=5.638e-11*sK**2+14.77d-6*sK
```

```
            end select
        else if(num_T==5) then
          select case (num_Pv)
                case (1)
                  sff=1.103e-11*sK**2+2.503d-6*sK
                case (2)
                  sff=1.826e-11*sK**2+5.135d-6*sK
                case (3)
                  sff=2.750e-11*sK**2+8.210d-6*sK
             case (4)
                  sff=3.686e-11*sK**2+11.47d-6*sK
                case (5)
                  sff=4.818e-11*sK**2+13.95d-6*sK
                case default
                  sff=5.890e-11*sK**2+16.85d-6*sK
               end select
        else if(num_T==6) then
          select case (num_Pv)
                case (1)
                  sff=7.088e-12*sK**2+2.535d-6*sK
                case (2)
                  sff=1.716e-11*sK**2+5.431d-6*sK
                case (3)
                  sff=2.760e-11*sK**2+9.323d-6*sK
             case (4)
                  sff=3.869e-11*sK**2+13.02d-6*sK
                case (5)
                  sff=4.898e-11*sK**2+18.19d-6*sK
                case default
                  sff=6.283e-11*sK**2+20.22d-6*sK
               end select
        else
          select case (num_Pv)
                case (1)
                  sff=0.6904e-11*sK**2+1.165d-6*sK
                case (2)
                  sff=1.835e-11*sK**2+4.817d-6*sK
                case (3)
                  sff=3.053e-11*sK**2+8.811d-6*sK
             case (4)
                  sff=4.227e-11*sK**2+13.85d-6*sK
                case (5)
                  sff=5.535e-11*sK**2+18.65d-6*sK
                case default
                  sff=7.056e-11*sK**2+23.38d-6*sK
               end select
        end if
        fK_evap=sff
      end function
!
      function P_lang(sy,sgx_1,sgx0,sgx1,shy_1,shy0,shy1)
```

```
      double precision :: sy
      double precision :: sgx_1,sgx0,sgx1,shy_1,shy0,shy1
      double precision :: sterm1,sterm2,sterm3
      print *,sgx_1,sgx0,sgx1,shy_1,shy0,shy1

      sterm1=sgx_1*(sy-sgx0)*(sy-shy1)/(shy_1-shy0)/(shy_1-shy1)
      sterm2=sgx0*(sy-shy_1)*(sy-shy1)/(shy0-shy_1)/(shy0-shy1)
     sterm3=sgx1*(sy-shy_1)*(sy-shy0)/(shy1-shy_1)/(shy1-shy0)
!
!     sterm1=shy_1*(sx-sgx0)*(sx-sgx1)/(sgx_1-sgx0)/(sgx_1-sgx1)
!     sterm2=shy0*(sx-sgx_1)*(sx-sgx1)/(sgx0-sgx_1)/(sgx0-sgx1)
!     sterm3=shy1*(sx-sgx_1)*(sx-sgx0)/(sgx1-sgx_1)/(sgx1-sgx0)
      P_lang=sterm1+sterm2+sterm3
!     print *,'P_Lang='
!     print *,P_lang,sterm1,sterm2,sterm3
    end function
!
    subroutine P_coeff(sgx_1,sgx0,sgx1,shy_1,shy0,shy1,sa,sb,sc)
      double precision :: sa,sb,sc
      double precision :: shy_1,shy0,shy1,sgx_1,sgx0,sgx1
!     print *,shy_1,shy0,shy1,sgx_1,sgx0,sgx1
      sa=sgx0/(shy0-shy_1)/(shy0-shy1)+sgx_1/(shy_1-shy0)/(shy_1-shy1)&
      &+sgx1/(shy1-shy_1)/(shy1-shy0)
      sb=(-sgx_1*shy0-sgx_1*shy1)/(shy_1-shy0)/(shy_1-shy1)&
      &+(-sgx0*shy_1-sgx0*shy1)/(shy0-shy_1)/(shy0-shy1)&
      &+(-sgx1*shy_1-sgx1*shy0)/(shy1-shy_1)/(shy1-shy0)
      sc=sgx_1*shy0*shy1/(shy_1-shy0)/(shy_1-shy1)&
      &+sgx1*shy_1*shy0/(shy1-shy_1)/(shy1-shy0)&
      &+sgx0*shy_1*shy1/(shy0-shy_1)/(shy0-shy1)
    end subroutine
!
    function P_lang_coeff(sy,sgx_1,sgx0,sgx1,shy_1,shy0,shy1,sa,sb,sc)
      double precision :: sa,sb,sc,sy
      double precision :: shy_1,shy0,shy1,sgx_1,sgx0,sgx1
!     print *,shy_1,shy0,shy1,sgx_1,sgx0,sgx1
      P_lang_coeff=sa*sy**2.0d0+sb*sy+sc
    end function
!
    function H_lang(sy,sz,sgx_1,sgx0,sgx1,shy_1,shy0,shy1)
      double precision :: sy,sz,sgx_1,sgx0,sgx1,shy_1,shy0,shy1
      double precision :: sterm1,sterm2,sterm3,sterm4,sa,sb,sc
      double precision :: sterm11,sterm12
      double precision :: shy,shz,shyz,shyy,shzz
      call P_coeff(sgx_1,sgx0,sgx1,shy_1,shy0,shy1,sa,sb,sc)
!     print *,'sy,sa,sb,sc= '
!     print *,sy,sa,sb,sc
      sterm1=4.0d0*sz*sa**2.0d0+(2.0d0*sb*sa**2.0d0*sz)/&
     &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
      sterm2=4.0d0*sa**2.0d0
      sterm3=2.0d0*sb*sz**2.0d0*sa**4.0d0/(sqrt((sa*sy)**2.0d0&
```

```
     &+(sz*sa)**2.0d0))**3.0d0
        sterm4=2.0d0*sb*sa**2.0d0/sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
        sterm5=4.0d0*sy*sa**2.0d0+(2.0d0*sb*sa**2.0d0*sy)/&
     &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
        sterm6=2.0d0*sb*sy**2.0d0*sa**4.0d0/(sqrt((sa*sy)**2.0d0&
     &+(sz*sa)**2.0d0))**3.0d0
        sterm7=sb*sz*sa**3.0d0*sy/(sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0))**3.0d0
!
!
!       print *,'sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7= '
!       print *,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7
!       pause
!
        shz=sterm1/2.0d0/sa
        shzz=(sterm2-sterm3+sterm4)/2.0d0/sa
        shy=sterm5/2.0d0/sa
        shyy=(sterm2-sterm6+sterm4)/2.0d0/sa
        shyz=-sterm7
!
!       In case of x<0
!
!       shzz=(sterm2+sterm3-sterm4)/2.0d0/sa
!       shyy=(sterm2+sterm6-sterm4)/2.0d0/sa
!       shyz=sterm7
!       sterm1=4.0d0*sz*sa**2.0d0-(2.0d0*sb*sa**2.0d0*sz)/&
     &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
!       sterm5=4.0d0*sy*sa**2.0d0-(2.0d0*sb*sa**2.0d0*sy)/
     &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
!
        sterm11=(1.0d0+shy**2.0d0)*shzz-2.0d0*shz*shy*shyz+&
     &(1.0d0+shz**2.0d0)*shyy
        sterm12=2.0d0*(1.0d0+shz**2.0d0+shy**2.0d0)**(3.0d0/2.0d0)
!       print *,sterm11,sterm12
        H_lang=sterm11/sterm12
     end function
!
     function cH_lang(sy,sz,sgx_1,sgx0,sgx1,shy_1,shy0,shy1,ck)
        double precision :: sy,sz,sgx_1,sgx0,sgx1,shy_1,shy0,shy1
        double precision :: sterm1,sterm2,sterm3,sterm4,sa,sb,sc
        double precision :: sterm11,sterm12
        double precision :: shy,shz,shyz,shyy,shzz,ck
        call P_coeff(sgx_1,sgx0,sgx1,shy_1,shy0,shy1,sa,sb,sc)
!
        if (sy==0.0) then
          shy=2.0*sa*sy/ck+b
              shyy=2.0*sa/ck
              shz=0.0
              shzz=0.0
```

```
                shyz=0.0
            else
              shz=0.0
              shzz=(2.0*sy*sa+sb)/ck/sy
              shy=2.0*sy*sa+sb
              shyy=2.0*sa/ck
              shyz=0.0
            end if
!
!
        sterm11=(1.0d0+shy**2.0d0)*shzz-2.0d0*shz*shy*shyz+&
        &(1.0d0+shz**2.0d0)*shyy
          sterm12=2.0d0*(1.0d0+shz**2.0d0+shy**2.0d0)**(3.0d0/2.0d0)
!         print *,sterm11,sterm12
          cH_lang=sterm11/sterm12
        end function
!
        subroutine Seid(N,A,X1)
!
          double precision :: A(N,N+1),X1(N),S
          double precision :: err
          integer :: kk,max
          max=100000
          kk=0
          err=1.0d-5
          do kks=1,N
            print *,A(kks,1),A(kks,2),A(kks,3),A(kks,4)
                X1(kks)=1.0d0
          end do
!         pause
!
          do while (max>kk)
            DO I=1,N
          S = 0.0d0
!         DO-LOOP COMPUTED THE SUMMATION
          DO J=1,N
            S = S-A(I,J)*X1(J)
                  end do
          S = (S+A(I,N+1))/A(I,I)
          X1(I) = X1(I)+S
          IF(ABS(S).GT.ERR)   ERR=ABS(S)
            end do
        WRITE(6,'(I10,10E22.10)') KK,ERR,(X1(I),I=1,N)
!
        IF(ERR.LE.TOL) THEN
          WRITE(6,'(I10,10E22.10)') K,TOL
          exit
        END IF
        kk = kk+1
      end do
```

```
      end subroutine

!
      subroutine Least_Coeff(ns,nps,sa,sx,sy)
        double precision :: sa(ns,ns+1),sx(nps),sy(nps)
!
        do is=1,ns
          do js=1,ns+1
                sa(is,js)=0.0d0
                do ks=1,nps
                  if (js==ns+1) then
                        sa(is,js)=sa(is,js)+sx(ks)*sy(ks)**float(is-1)
                    else
                   sa(is,js)=sa(is,js)+sy(ks)**float(is+js-2)
                    end if
                end do
          end do
        end do
!
      end subroutine
!
      subroutine gauss(N,M,A,DELT)
        double precision:: A(N,M+N),PIV
        if (N>1) then
          do K=1,N-1
                PIV=abs(A(K,K))
                KK=K+1
                IN=K
                do I=KK,N
                  if (abs(A(I,K))>PIV) then
                        PIV=abs(A(I,K))
                        IN=I
                    end if
                end do
                if (K<>IN) then
                  do J=K,M+N
                        X=A(K,J)
                        A(K,J)=A(IN,J)
                        A(IN,J)=X
                    end do
                end if
                if (PIV<DELT) then
                  print *,'Error!'
                        pause
                  return
                end if
                do I=KK,N
                  do J=KK,M+N
                        A(I,J)=A(I,J)-A(I,K)*A(K,J)/A(K,K)
                    end do
                end do
```

```
                end do
!
            if (abs(A(N,N))<DELT) then
               print *,'Error!'
!                    pause
            end if
!
            do K=1,M
               A(N,K+N)=A(N,K+N)/A(N,N)
                  do IE=1,N-1
                     I=N-IE
                     IX=I+1
                     do J=IX,N
                        A(I,K+N)=A(I,K+N)-A(J,K+N)*A(I,J)
                     end do
                     A(I,K+N)=A(I,K+N)/A(I,I)
                  end do
            end do
                  return
         else if (abs(A(1,1))<DELT) then
            print *,'Error!'
         end if
!
         do J=1,M
            A(1,N+J)=A(1,N+J)/A(1,1)
         end do
         return
      end subroutine
!
      function H_Least(sy,sz,sa,sb,sc)
         double precision :: sy,sz
         double precision :: sterm1,sterm2,sterm3,sterm4,sa,sb,sc
         double precision :: sterm11,sterm12
         double precision :: shy,shz,shyz,shyy,shzz
!        call P_coeff(sgx_1,sgx0,sgx1,shy_1,shy0,shy1,sa,sb,sc)
!        print *,'sy,sa,sb,sc= '
!        print *,sy,sa,sb,sc
         sterm1=4.0d0*sz*sa**2.0d0+(2.0d0*sb*sa**2.0d0*sz)/&
      &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
         sterm2=4.0d0*sa**2.0d0
         sterm3=2.0d0*sb*sz**2.0d0*sa**4.0d0/&
      &(sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0))**3.0d0
         sterm4=2.0d0*sb*sa**2.0d0/sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
         sterm5=4.0d0*sy*sa**2.0d0+(2.0d0*sb*sa**2.0d0*sy)/&
      &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
         sterm6=2.0d0*sb*sy**2.0d0*sa**4.0d0/&
      &(sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0))**3.0d0
         sterm7=sb*sz*sa**3.0d0*sy/(sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0))**3.0d0
!
```

```
!
!             print *,'sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7= '
!             print *,sterm1,sterm2,sterm3,sterm4,sterm5,sterm6,sterm7
!             pause
!
              shz=sterm1/2.0d0/sa
              shzz=(sterm2-sterm3+sterm4)/2.0d0/sa
              shy=sterm5/2.0d0/sa
              shyy=(sterm2-sterm6+sterm4)/2.0d0/sa
              shyz=-sterm7
!
!             In case of x<0
!
!             shzz=(sterm2+sterm3-sterm4)/2.0d0/sa
!             shyy=(sterm2+sterm6-sterm4)/2.0d0/sa
!             shyz=sterm7
!             sterm1=4.0d0*sz*sa**2.0d0-(2.0d0*sb*sa**2.0d0*sz)/&
            &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
!             sterm5=4.0d0*sy*sa**2.0d0-(2.0d0*sb*sa**2.0d0*sy)/&
            &sqrt((sa*sy)**2.0d0+(sz*sa)**2.0d0)
!
              sterm11=(1.0d0+shy**2.0d0)*shzz-2.0d0*shz*shy*shyz&
            &+(1.0d0+shz**2.0d0)*shyy
              sterm12=2.0d0*(1.0d0+shz**2.0d0+shy**2.0d0)**(3.0d0/2.0d0)
!             print *,sterm11,sterm12
              H_Least=sterm11/sterm12
            end function
!
            subroutine least_sq(ms,sx,sy,sa0,sa1)
              integer :: ms,iis
              double precision :: sx(10),sy(10)
              double precision :: sa0,sa1
              double precision :: sterm1,sterm2,sterm3,sterm4
!
              sterm1=0.0
              sterm2=0.0
              sterm3=0.0
              sterm4=0.0
              do iis=1,10
                sterm1=sterm1+sx(iis)*sx(iis)
                     sterm2=sterm2+sy(iis)
                     sterm3=sterm3+sx(iis)*sy(iis)
                     sterm4=sterm4+sx(iis)
!                    print *,sx(iis),sy(iis)
              end do
!             print *,sterm1,sterm2,sterm3,sterm4
!             print *,sterm1*sterm2,sterm3*sterm4,(float(10)*sterm1-sterm4*sterm4)
!             print *,float(10)*sterm3,sterm4*sterm2,(float(10)*sterm1-sterm4*sterm4)
              sa0=(sterm1*sterm2-sterm3*sterm4)/(float(ms)*sterm1-sterm4*sterm4)
```

```
      sa1=(float(ms)*sterm3-sterm4*sterm2)/(float(ms)*sterm1-sterm4*sterm4)
!     print *,sa0,sa1
    end subroutine
!
    function K1st(sh0,sh1,sRc,sdx)
      double precision :: sh0,sh1,sRc,sdx
      double precision :: sterm1,sterm2
      sterm1=(sh0+sh1)/2.0
      sterm2=sqrt(1.0-((sh1-sh0)/sdx)**2)
      print *,sterm1,sterm2
      K1st=1.0/(sRc-sterm1)/sterm2
    end function
!
    function hddd(sdx,sRc,su1,su2,su3,su4)
      double precision :: sdx,sRc,su1,su2,su3,su4,sTl
      double precision :: ssigma,shameker,sTc,sTf
      double precision :: sterm1,sterm2,sterm3,sterm4
      double precision :: sterm5,sterm6,sterm7,sterm8
      sterm1=su2**2/(sRc-su1)**3/sqrt(1.0+su2**2)
      sterm2=-su2**2*su3/(sRc-su1)**2/(1.0+su2**2)**(3.0/2.0)
      sterm3=su3/(sRc-su1)**2/sqrt(1.0+su2**2)/2.0
      sterm4=3.0/2.0*(su2**2*su3**2)/(sRc-su1)/(1.0+su2**2)**(5.0/2.0)
      sterm5=-1.0/2.0*su3**2/(sRc-su1)/(1.0+su2**2)**(3.0/2.0)
      sterm6=-1.0/2.0*su2*su4/(sRc-su1)/(1.0+su2**2)**(3.0/2.0)
      sterm7=-9.0/2.0*su4*su2*su3/(1.0+su2**2)**(5.0/2.0)
      sterm8=15.0/2.0*su3**3*su2**2/(1.0+su2**2)**(7.0/2.0)
      sterm9=-3.0/2.0*su3**3/(1.0+su2**2)**(5.0/2.0)
      hddd=sterm1+sterm2+sterm3+sterm4+sterm4+&
      &sterm5+sterm6+sterm7+sterm8+sterm9
    end function
!
    function Curva(sRc,su1,su2,su3,su4)
      double precision :: sRc,su1,su2,su3,su4
      double precision :: sterm1,sterm2
      sterm1=1.0/(sRc-su1)/sqrt(1.0+su2**2)
      sterm2=su3/(1.0+su2**2)**(3.0/2.0)
      Curva=sterm1+sterm2
    end function
```

VITA

Ryoji Oinuma was born on June 4, 1970 in Tokyo, Japan. After graduating in 1989 from Seijo High School in Shinjuku, Tokyo, Japan, he entered the nuclear engineering program at Tokai University in Hiratsuka, Kanagawa, Japan. He received a B.S. degree in March 1994 and earned M. S. degree in March 1996. He came to the United States in August 1997 and studied at Texas A&M University, College Station, majoring in nuclear engineering. He received his Ph.D. degree in August 2004. He can be reached at 3-7-8 Minamisenzoku, Ota-ku, Tokyo 145-0063, Japan.