EXTENDED FILM THEORY FOR TRANSPIRATION BOUNDARY LAYER FLOW AT HIGH MASS TRANSFER RATES

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

MAY 2003

By
Fuminori Nakamura

Thesis Committee:

Carlos F. M. Coimbra, Chairperson
Beei-huan Chao
Michelle H. Teng

ACKNOWLEDGEMENTS

I am most grateful to my advisor, Professor Carlos F.M. Coimbra, for his profound knowledge, appropriate guidance and patience throughout my research. I gratefully acknowledge the TA-ship from Department of Mechanical Engineering.

This work is dedicated to Professor Richard M. Fand for his guidance, support and friendship throughout my graduate studies at the University of Hawai‘i. This work is also dedicated to Mr. Tong-Un Chang, who has always been encouraging me to strive for the highest.

I would like to thank my office mates, Ms. Min Zhong, and Mr. Manuel Munoz. We created a nice atmosphere in our office and worked together throughout our graduate studies at the University of Hawai‘i. I would also like to thank the faculty and the staff of Department of Mechanical Engineering for their support and guidance throughout my studies at the University of Hawai‘i.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Mass transfer is of interest to understand a wide variety of applications ranging from natural systems to technological processes, from biological metabolic rates to mass exchanger and catalytic converters, from pollution control to transpiration cooling, etc. Moreover, mass transfer processes play a critical role in the regulation of climate at both local and global levels. For example, in Oahu, trade winds pick up moisture as they blow from the ocean to the windward coast of the island. As the moist air reaches the mountain range and is forced to move up (and cool off), water vapor is condensed. The inertia of the flow is sufficient to make the condensing vapors pass the mountain ridge, delivering high rates of precipitation to the Manoa valley. The main mechanism in the rain formation in this case is a mass transfer-controlled process. The evaporation rate of water vapor from the surface of the ocean and the amount of condensation formed during the cool-off process directly impact the yearly levels of precipitation in the Manoa valley.

Despite its relevance to natural and technological processes, there are still much not known in the process of evaporation and the combined effects of convective flow, heat and mass transfer. In order to estimate mass transfer rates in boundary layer flows, two approaches are currently available:

1. To solve numerically the boundary layer equations.
2. To use appropriate graphs in textbooks to estimate the mass transfer rate, based on solutions given by the methods described in item (1).

Both methods described above have shortcomings. The first is not practical because it requires a considerable amount of effort and time to yield a comprehensive understanding of the phenomena involved for each case under study. The second is limited in accuracy and leaves the mass transfer analyst with little flexibility beyond the results available in the literature. Ideally one would like to have reliable correlations that are valid for a wide range of parameters such as blowing factors, Schmidt numbers, Reynolds numbers, etc. The goal of the present study is to provide one such correlation based on a detailed analysis of the boundary layer equations and results of film theory applied to a porous plate.

To pursue our goal for mass transfer rates in the boundary layer flow, in the present study, we will consider an inert binary mixture where one of the species is transferred (evaporated) through a porous (or wet) flat plate. A laminar boundary layer is formed over the plate, and we will focus on high mass transfer in the blowing (evaporation) regime, such as water evaporating at high temperatures (when the vapor pressure at the liquid-vapor interface is relatively high and therefore the mass fraction at the interface is not negligibly small).

We will develop the functional form for a correlation that gives the mass transfer rate as a function of the relevant parameters. As it is usual in heat and mass transfer problems, a correlation is formed by collapsing the information contained in the boundary value problem (the combination of differential governing equations and boundary conditions) into an algebraic expression that involves only the boundary conditions and the properties of the medium. In our case, the Schmidt number (the ratio of momentum to mass diffusion coefficients), the flow conditions (free stream velocity, viscosity and length scale, or the Reynolds number), and vapor concentrations at the liquid-vapor interface and at the free-stream conditions are the dependent variables. The mass transfer (or evaporation) rate is therefore correlated in terms of the Reynolds and Schmidt numbers and the boundary conditions for mass concentrations. The boundary conditions are lumped into a dimensionless number referred to as the mass transfer potential, which is given by $B_m \equiv (m_{1,s} - m_{1,e})/(1 - m_{1,s})$, where '$m_1$' is the mass fraction of species 1 and the subscripts 's' and 'e' referred to "surface" and "external".

The following steps will be taken to develop the desired correlation: in Chapter 2, we will present a film theory that is relevant to the evaporation process. The film theory will give us an approximate value for the mass transfer rate by assuming that the flow is fully developed and therefore does not vary in the longitudinal direction of the plate. We will build on the film theory concept in order to develop a functional form for our correlation. An important quantity in our problem, the normalized mass transfer rate, Y, is defined in Chapter 2. Chapter 3 presents a brief overview of boundary theory, where Y and B_m are expressed in terms of the boundary layer variables instead of the film theory parameters. The boundary layer theory gives us better estimates for mass transfer rates, and therefore we derive the boundary layer system of equations to be used in the development of our correlation. In Chapter 4, we solve the boundary layer system using a fourth-order Runge-Kutta method, and we use the solution of the boundary layer equations to determine the evaporation rates needed for our correlation.

The Runge-Kutta method was implemented in a FORTRAN program described in Appendix A and listed in Appendix B.

Chapter 2 Momentum and Mass Transfer over a Flat Plate

In this chapter we develop the appropriate governing equations that describe the mass transfer process over a porous (or wet) flat plate. We concentrate on high mass transfer rates in the blowing (evaporation) regime. To understand the underlying physics, we first discuss the physical process in section 2.1 and 2.2 to later develop a simplified film theory in sections 2.3 and 2.4. In Chapter 3, the boundary layer system of equations is presented to model the high mass transfer problem from a flat plate.

2.1 The Boundary Layer Concept

Ludwig Prandtl first introduced the concept of boundary layer in a paper presented at the Mathematical Congress in Heidelberg in 1904 (Schlichting, 1979). In the ensuing two decades after his original presentation, Prandtl's theory proved to be instrumental in solving problems that could not be solved by direct application of the Navier-Stokes equations. The boundary layer concept (which is the realization that viscous effects at high Reynolds numbers are confined to a very small region close to the surface where a no-slip condition exists) permitted to discuss intelligently the solution of viscous flow problem that retained the main characteristics of the full Navier-Stokes flow field but were amenable to solution by standard mathematical methods available at the time. Thus, the introduction of the boundary layer concept marked the beginning of a very productive era of research in fluid mechanics.

As mentioned above, through a series of experiments, scaling analysis and theoretical reasoning, Prandtl showed that in high Reynolds number flows the influence of viscosity is confined to an extremely thin region very close to the region where the no-slip condition applies, and the remainder of the flow field could be well approximated by an inviscid field. Prandtl's most fundamental contribution was to indicate that many viscous flows are amenable to analysis by dividing the flow into two regions, one close to solid boundaries where the velocity of the flow field is strongly affected by viscous effects, the other covering the rest of the flow where an inviscid approximation is justified. These two flow regions for flow over a flat plate are illustrated in Fig. 2.1.

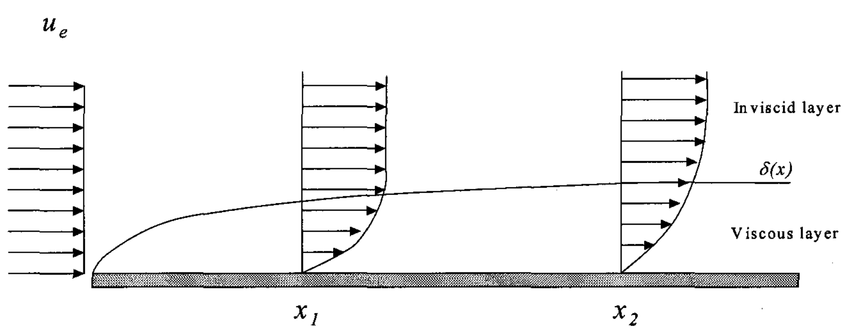

Figure. 2.1 A laminar boundary layer over a flat plate.

Inside the boundary layer, both viscous and inertia forces are important. Consequently, the Reynolds number ($\mathrm{Re}_x = u_e\, x/\nu$), which usually represents the ratio of inertia to viscous forces, has a less obvious but not less meaningful significance in boundary layer flows. Because inertia and viscous flows are both equally important in the boundary layer region, the magnitude of the Reynolds number cannot be a true estimate of the ratio of inertial to viscous forces. In boundary layer flows, the Reynolds number is more of a geometrical parameter that gives us an estimate for the boundary layer thickness in respect to the length of the plate (see e.g Bejan, 1994).

In a mass transfer problem, there are at least three boundary layers present, namely the momentum, thermal, and species boundary layers. Our major subject of study in this work is the high mass transfer problem over a flat plate, so that we focus on the momentum and species boundary layers, which are coupled by the velocity components. Throughout this work we will assume that the temperatures of the liquid-vapor interface and the free-stream flow are known so that we concentrate on the momentum and mass transfer problems. Below we consider the momentum boundary layer as an illustration to our terminology and nomenclature. We will discuss the coupling between the transfer of momentum and mass transfer in later sections of this chapter and in Chapter 3.

2.2 The Momentum Boundary Layer

We consider incompressible viscous flow over a flat plate with a free-stream velocity, u_e, as shown in Fig. 2.1. The thickness of a momentum boundary layer, $\delta(x)$, is defined as the height at which the velocity field is, by all accounts, undisturbed by the presence of the plate (earlier works placed the end of the boundary layer at $u(y) = 0.99u_e$ more modern approaches use the concepts of momentum thickness and displacement thickness, (see e.g. Schlichting, 1979).

Fluid particles in contact with the surface of a body assume the velocity of the body according to the no-slip condition. These particles act to retard the motion of particles in the adjoining fluid layer, which acts to retard the motion of particles in the next layer, and so on. The retardation effect propagates through the upper layers until the effect becomes negligible at the distance $y=\delta(x)$ from the surface. The retardation of fluid motion is associated with shear stresses, which represent the diffusion of momentum in the direction perpendicular to the fluid velocity, causing the fluid to slow down, and hence thickening the boundary layer as the distance progresses from the leading edge. The effect is shown as the difference in velocity profiles at x_1 and x_2 in Fig. 2.1, a result of both convective and diffusive effects in the boundary layer. Thus, according to Prandtl's theory, the fluid flow can be separated in two distinct regions: a thin fluid layer (the momentum boundary layer) in which velocity gradients and shear stresses are large, and a region outside the momentum boundary layer in which velocity gradients and shear stresses are negligible.

2.3 The Classic Film Theory for an Infinite Plate

A thin film model, which is based on a fully developed flow between two constant fluid velocities, is introduced as a working model for the actual momentum boundary layer profile. An illustration of the two profiles is shown in Fig. 2.2. A film (or Couette model) will be used to establish a simplified relationship between the flow conditions and evaporation rates. This simplified theory is referred in this work as CFT (Classic Film Theory). We will base our starting point for the sought correlation on an extension of the CFT hitherto called as Extended Film Theory (EFT).

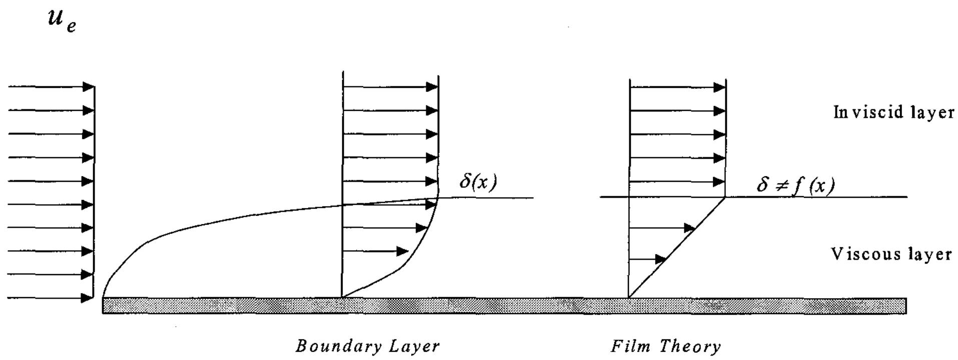

Figure. 2.2 Boundary layer and thin film profiles.

As shown in Fig. 2.2, the CFT model deviates from the actual momentum boundary layer profile since the plate is assumed to be infinite. Therefore, the derived correlation using the CFT needs to be adjusted to accommodate these differences. The EFT will be discus`sed in section 2.4.

Given the geometry of our problem is only appropriate to use Cartesian coordinates to describe the continuity, momentum and species conservation equations. The general form of the mass conservation equation in Cartesian coordinates is:

$$\frac{\partial \rho}{\partial t}+\left(\frac{\partial n_x}{\partial x}+\frac{\partial n_y}{\partial y}+\frac{\partial n_z}{\partial z}\right)=0 \tag{2.1}$$

where the subscripts, x, y, and z, indicate the vector components of the total mass flux, n, in the x, y, and z-directions correspondingly. For a steady and two-dimensional flow, Eq. 2.1 is reduced to

$$\frac{\partial n_x}{\partial x} + \frac{\partial n_y}{\partial y} = 0 . \tag{2.2}$$

In the CFT model, all variables are independent of x, so that $\frac{\partial n_x}{\partial x} = 0$, and Eq. 2.2 is reduced to

$$\frac{\partial n_y}{\partial y} = 0 \text{, or} \qquad n_y = \text{constant.} \tag{2.3}$$

Since the mass flux is only in the y-direction the subscript y is dropped. The total mass flux n is therefore constant in the CFT.

In our analysis we are concerned with one species in a binary mixture. In this case, the total mass flux n is equal to the mass flux of species 1, n_1. Since there is no variation of the total mass flux with y, we equate the total mass flux n as $n =$ constant.

This means that the total mass flux is constant regardless the value of y, or the height from the surface of the porous flat plate. In particular, at the s-surface, $\dot{m}'' = n_{1,s} = n$. The total flux of species 1 is the sum of the diffusive and convective fluxes, i.e., $n_1 = j_1 + m_1 n$ (Mills, 2001), so that, after applying Fick's law of diffusion

$$j_1 = - \rho D_{12} \frac{dm_1}{dy}, \tag{2.4}$$

we arrive at

$$(1 - m_1)\, \dot{m}'' = - \rho D_{12} \frac{dm_1}{dy} . \tag{2.5}$$

We can rearrange Eq. 2.5 and integrate dm_1 from $m_{1,s}$ at $y=0$ to m_1 at a generic height y. Then,

$$\int_{m_{1,s}}^{m_1} \frac{dm}{m_1 - 1} = \int_0^y \frac{\dot{m}''}{\rho D_{12}} dy \tag{2.6}$$

We assume that ρD_{12} is constant, so that the above integration becomes

$$\frac{m_1 - 1}{m_{1,s} - 1} = \exp\left(\frac{\dot{m}'' y}{\rho D_{12}}\right), \tag{2.7}$$

or after rearranging the terms,

$$m_1 = \exp\left(\frac{\dot{m}'' y}{\rho D_{12}}\right)(m_{1,s} - 1) + 1. \tag{2.8}$$

Adding $1 - \frac{m_{1,s} - 1}{m_{1,s} - 1}$ to the left side of Eq. 2.8 gives

$$1 + \frac{m_1 - m_{1,s}}{m_{1,s} - 1} = \exp\left(\frac{\dot{m}'' y}{\rho D_{12}}\right), \tag{2.9}$$

and with the boundary condition $m_1 = m_{1,e}$ at $y = \delta_m$ we arrive at

$$1 + \frac{m_{1,e} - m_{1,s}}{m_{1,s} - 1} = \exp\left(\frac{\dot{m}'' \delta_m}{\rho D_{12}}\right). \tag{2.10}$$

The mass transfer conductance g is defined as

$$g \equiv \frac{j_{1,s}}{m_{1,s} - m_{1,e}}, \tag{2.11}$$

where $j_{1,s}$ can be expressed as

$$j_{1,s} = n_{1,s} - m_{1,s} n_s. \tag{2.12}$$

From the mass, and species conservation identities, $n_s = n_{1,s} = \dot{m}''$, Eq. 2.12 becomes

$$j_{1,s} = \dot{m}''\left(1 - m_{1,s}\right), \tag{2.13}$$

which, after rearranging, yields

$$\dot{m}'' = g\left(\frac{m_{1,s} - m_{1,e}}{1 - m_{1,s}}\right). \tag{2.14}$$

We define the mass transfer potential, B_m, as

$$B_m \equiv \left(\frac{m_{1,s} - m_{1,e}}{1 - m_{1,s}}\right), \tag{2.15}$$

so that (Mills, 2001)

$$\dot{m}'' = gB_m. \tag{2.16}$$

Eq. 2.16 gives the mass flux of species 1 (the total mass flux since only one species is being transferred) in terms of the boundary conditions (B_m) and the convective mass transfer conductance g. The difficulty here is that g is modified as high mass transfer rates are considered. Thus we must find a way to correct g since only the "dry" value (or zero mass transfer rate) of the conductance g^* is well known.

We now convert Eq. 2.16 into a non-dimensional form by introducing the dry-condition mass transfer conductance, g^*, and the normalized mass transfer rate, Y. The dry condition indicates that the convective component of the mass transfer rate goes to zero, so that $\dot{m}'' \to 0$ (Mills, 2001). Taking the logarithm of both sides of Eq. 2.10 and rearranging the terms, we get an expression for the mass transfer rate in terms of the species boundary layer thickness:

$$\dot{m}'' = \frac{\rho D_{12}}{\delta_m} \ln(1 + B_m), \tag{2.17}$$

and multiplying B_m / B_m to the right side of Eq. 2.17 we arrive at

$$\dot{m}'' = \frac{\rho D_{12}}{\delta_m} \frac{\ln(1 + B_m)}{B_m} B_m. \tag{2.18}$$

Comparing Eq.2.16 and Eq. 2.18, we recognize that

$$g = \frac{\rho D_{12}}{\delta_m} \frac{\ln(1 + B_m)}{B_m}. \tag{2.19}$$

For dry conditions, $B_m \to 0$, and $g \to g^*$. Then, taking the limit

$$\lim_{B_m \to 0} \frac{\ln(1 + B_m)}{B_m} = 1 \tag{2.20}$$

We thus define the dry-condition mass transfer conductance from Eq. 2.19 as

$$g^* \equiv \frac{\rho D_{12}}{\delta_m^*}, \tag{2.21}$$

and we introduce the normalized mass transfer rate

$$Y \equiv \frac{\dot{m}''}{g^*}, \tag{2.22}$$

resulting in

$$\ln(1 + B_m) = \frac{\delta_m}{\delta_m^*} Y, \tag{2.23}$$

which is the non-dimensional form of Eq. 2.16.

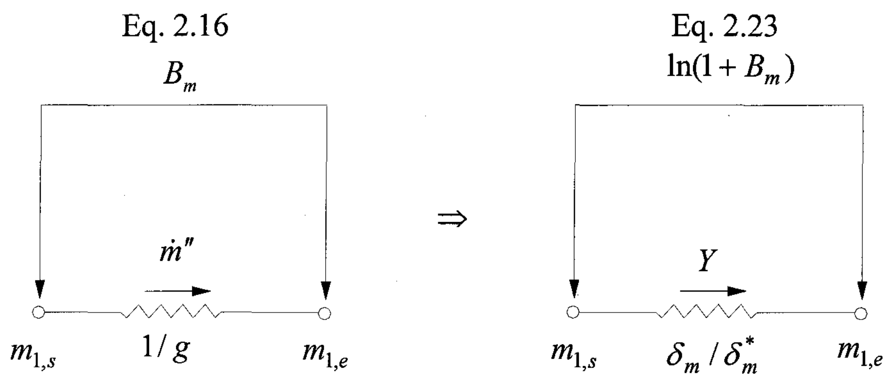

Figure. 2.3 A graphic representation of Eqs. 2.16 and Eq. 2.23.

For the dry-condition, $\dot{m}'' \to 0$ and $\delta_m \to \delta_m^*$, and

$$Y = \ln(1+B_m), \tag{2.24}$$

which is the main result of the CFT (Mills, 2001). An obvious limitation of the CFT is that the theory only applies to low mass transfer rates where the vaporization flux in the y-direction does not disturb both the gradients at the wall and the boundary layer thickness. This limitation comes from Eq. 2.20 and from the fact that good estimates for the boundary layer thickness δ_m and for the wet (high mass transfer rate) conductance g are not available. We will develop an original theory next where some of the shortcomings of the CFT are circumvented. This is accomplished by taking a different limit than indicated in Eq. 2.20 as explained in the next section.

2.4 The Extended Film Theory (EFT)

Our objective in this section is to seek for an appropriate functional form for the relationship between the boundary layer thickness at dry and wet conditions. We introduce a function h, such that $h(\Delta) = \frac{\delta_m}{\delta_m^*}$, and analyze the function $h(\Delta)$ (Coimbra, 2002).

The thickness of the species boundary layer, δ_m, is mainly dependent on two quantities, namely the normalized evaporative mass flux Y and the Sc number (ν/D_{12}). We combine these two parameters in a "similarity" variable $\Delta = Y / A(Sc)$ and look for physical constraints that would allow us to estimate the behavior of $h(\Delta)$.

The function $h(\Delta) = \frac{\delta_m}{\delta_m^*}$ must, as minimum, satisfy the following constraints:

1. $\lim_{\Delta \to 0} h(\Delta) = 1$, since $\Delta \to 0$ implies $\dot{m}'' \to 0$ and $\delta_m \to \delta_m^*$

2. The Domain (D) and the Image (I) of $h(\Delta)$ for blowing/evaporation are

$$\mathrm{D}\{\, h(\Delta) \,\} = [0, \infty)$$

$$\mathrm{I}\{\, h(\Delta) \,\} = [1, \infty)$$

3. $h(\Delta)$ must be a monotonically increasing, positive function of Δ.

Note that Δ assumes negative values for strong suction (or condensation), but here we are concerned only with the evaporation regime. In any case, the function chosen should also cover, at least in principle, the condensation regime. Two trial functions for $h(\Delta)$ that satisfy all the conditions above are

$$h(\Delta)=\frac{e^{\Delta}-1}{\Delta} \quad \text{and} \quad h(\Delta) = e^{\Delta}, \tag{2.25}$$

but the first option is more convenient algebraically (Coimbra, 2002). We chose our trial function to $h(\Delta)=\left(e^{\Delta}-1\right)/\Delta$, and we substitute this expression back in equation Eq. 2.23:

$$\ln(1+B_m)=\frac{e^{\Delta}-1}{\Delta}\cdot Y. \tag{2.26}$$

Substituting $\Delta = Y / A(Sc)$ in Eq. 2.26 and rearranging the terms we arrive at

$$Y_A \equiv A(Sc)\cdot \ln\left(1+\frac{\ln(1+B_m)}{A(Sc)}\right) \tag{2.27}$$

In general the form of $A(Sc)$ is not known, but since the similarity variable has to work in the dry regime too, and since the conductance in the dry regime varies with $Sc^{1/3}$, we assume a functional form for $A(Sc)$ such that

$$A(Sc)=\frac{Sc^{1/3}}{a} \tag{2.28}$$

where a is a constant to be determined.

With the procedure outlined above we extended the CFT to allow for variations of the boundary layer thickness with the evaporation rate. The functional form chosen for $h(\Delta)$ and $A(Sc)$ are the simplest possible that also satisfy the physical constraints of the problem. Only one parameter (the constant a) is left to be determined, and we will determine this parameter by direct comparison of Eq. 2.27 with the numerical results given by the solution of the boundary layer system. If a constant a is sufficient to predict a wide range of boundary layer behaviors (different B_m and Sc), the procedure used to derive Eq. 2.27 is validated and a working correlation for mass transfer is found.

Chapter 3 Boundary Layer Model

In this chapter we approach the same high mass transfer rate problem treated in Chapter 2 using a boundary layer analysis. The general form of the continuity, the x-momentum, the species, and the thermal energy conservation equations are introduced in section 3.2. The general equations will be simplified to the Prandtl system of the equations by means of a scaling analysis in section 3.3. The concept of similarity is introduced in section 3.4. Then, in section 3.5, we determine the normalized mass transfer flux, Y, and the mass transfer potential, B_m, which were introduced in chapter 2, in terms of the boundary layer variables.

3.1 Governing Equations

We consider a two-dimensional laminar flow over a porous (or wet) flat plate. We focus our attention on a steady, incompressible flow of a fluid that has thermal properties that are well approximated to be invariant during the process. We also assume negligible effect of body forces.

There are three principles that need to be satisfied by the flow:

1. conservation of mass
2. conservation of linear momentum
3. conservation of individual species

The governing equations are based on the continuum assumption of the flow field, and the three conservation principles are implemented into corresponding partial differential equations through a differential volume formulation. In two-dimensional space, the vector equation for the momentum principle has two scalar components. We thus arrive at a system of four partial differential equations:

The continuity equation: $$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, \tag{3.1}$$

The x-momentum equation: $$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right), \tag{3.2}$$

The y-momentum equation: $$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right), \tag{3.3}$$

The species conservation equation: $$u\frac{\partial m_1}{\partial x}+v\frac{\partial m_1}{\partial y}= D_{12}\left(\frac{\partial^2 m_1}{\partial x^2}+\frac{\partial^2 m_1}{\partial y^2}\right). \tag{3.4}$$

3.2 Boundary Layer Equations

Eqs. 3.1 to 3.4 describe the physical nature of two-dimensional laminar flow over the porous flat plate. The solution of the four equations is not an easy task, and so substantial simplification is sought by applying the concept of boundary layer. By invoking the principle that viscous effects are confined to a boundary layer (BL), which is very thin compared with the characteristic length of the plate for large Reynolds numbers, we can substantially simplify the system of equations to (Schilichting, 1979; Mills, 2001):

The continuity equation: $$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{3.5}$$

The x- and y- BL momentum equation: $$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\frac{\partial^2 u}{\partial y^2} \tag{3.6}$$

The BL species conservation equation: $$u\frac{\partial m_1}{\partial x}+v\frac{\partial m_1}{\partial y}=D_{12}\frac{\partial^2 m_1}{\partial y^2} \tag{3.7}$$

3.3 Similarity Formulation

Our goal is to determine the mass transfer rate from a porous flat plate in the high mass transfer regime. In order to solve the system Eq. 3.5 ~ Eq. 3.7, we resort to a similarity approach in order to transform the boundary value problem to an ordinary differential equation. Prandtl's group first introduced the concept of similarity solution of boundary layer equations through the work of H. Blasius in 1908 (see e.g., Schlichting, 1979). The method is basically the well-know method of combination of variables used to reduce PDEs to ODEs.

Blasius (1908) introduced the similarity variable

$$\eta=y\sqrt{\frac{u_e}{2\nu x}} \tag{3.8}$$

and formulated the BL momentum equation in terms of the streamfunctions defined as

$$u=\frac{\partial\psi}{\partial y},\quad v=-\frac{\partial\psi}{\partial x}, \tag{3.9}$$

to arrive at the celebrated Blasius equation:

$$\frac{d^3 f}{d\eta^3}+f\frac{d^2 f}{d\eta^2}=0 \tag{3.10}$$

or

$$f'''+ff''=0, \tag{3.11}$$

where the primes indicate derivatives in respect to η and

$$f(\eta)=\frac{\psi}{\sqrt{2\nu x u_e}} \tag{3.12}$$

is the dimensionless stream function. Since $f'(\eta)=u/u_e$ and the absolute value of the stream function is immaterial, we have two simple boundary conditions for f at zero and for the velocity at infinity:

$$f'(0)=0, \quad \text{and} \quad f'(\infty)=1. \tag{3.13}$$

A similar approach can be used to generate the similarity boundary layer species equation by defining a dimensionless scalar, $\phi(\eta)$, as

$$\phi(\eta)\equiv\frac{m_1-m_{1,e}}{m_{1,s}-m_{1,e}}, \tag{3.14}$$

and by rewriting the BL species equation in terms of $m_1=(m_{1,s}-m_{1,e})\phi+m_{1,e}$.

The non-dimensional form of the BL species conservation equation is thus

$$\phi''+Scf\phi=0, \tag{3.15}$$

subjected to the following boundary conditions (see e.g., Mills, 2001):

$$\phi(0)=1, \quad \text{and} \quad \phi(\infty)=0 \tag{3.16}$$

Since the Blasius equation is a third-order, non-linear equation, we still need one more boundary condition in addition to the two previously mentioned (Mills, 2001). In order to determine this third boundary condition, we look into the y-component of the velocity at the wall in terms of $f(0)$ (the x-component is zero at the wall by the no-slip condition):

$$v_s=-\sqrt{\frac{u_e\nu}{2x}}f(0). \tag{3.17}$$

For blowing or evaporation ($v_s > 0$), $f(0)$ must be a constant defined between zero and -0.875 (blown-off condition when the shear stress at the wall is zero). This is the third boundary condition for Eq. 3.11, so that at $\eta=0$,

$$-0.875 < f(0) = \text{Constant} < 0 \text{ (blowing/evaporation)} \tag{3.18}$$

At $f(0) = -0.875$, $f''(0)$ becomes zero, and a wall shear stress becomes zero. This condition is said to be "blown-off the wall", and the boundary layer equations no longer hold.

The mass transfer conductance, g, and the dry condition mass transfer conductance, g^*, can be expressed in terms of the Stanton mass number as

$$g = \rho u_e St_{mx}, \quad \text{and} \quad g^* = \rho u_e St_{mx}, \tag{3.19}$$

where the Stanton number and the dry Stanton number are defined as

$$St_{mx} = \frac{-\phi'(0)}{\sqrt{2\mathrm{Re}_x Sc}} \quad \text{and} \quad St_{mx}^* = \frac{-\phi_0'(0)}{\sqrt{2\mathrm{Re}_x Sc}}, \tag{3.20}$$

and $\phi_0'(0) \equiv \phi'(0)|_{f(0)=0}$. Then, the dry condition mass conductance becomes

$$g^* = -\frac{\rho u_e \phi_0'(0)}{\sqrt{2\mathrm{Re}_x Sc}} \tag{3.21}$$

Also, the normalized mass conductance becomes

$$\frac{g}{g^*} = \frac{St_{mx}}{St_{mx}^*} = \frac{\phi'(0)}{\phi'(0)|_{f(0)=0}} = \frac{\phi'(0)}{\phi_0'(0)} \tag{3.22}$$

From Eq 3.17, we express the mass flux at the wall as

$$\dot{m}'' = \rho v_s = -\rho\sqrt{\frac{u_e \nu}{2x}} f(0), \tag{3.23}$$

which allows us to finally reach the objective of writing the normalized mass transfer flux Y, and the mass transfer potential B_m in terms of the boundary layer solution values

$$Y = \frac{\dot{m}''}{g^*} = \frac{Sc\, f(0)}{\phi_0'(0)} \tag{3.24}$$

$$B_m = \frac{g^*}{g} Y = \frac{Sc\, f(0)}{\phi'(0)} \tag{3.25}$$

The final step in our analysis is to use the numerical solution of the boundary layer equations to find the parameter a needed to complete correlation Eq. 2.27.

Chapter4 Numerical Solution for the Governing Equations

In this chapter, we determine the value of coefficient a in Eq. 2.27 by numerically solving the BL momentum and species equations.

First, we select relevant values of Schmidt number. Then, for each Schmidt number, we solve the boundary layer equations, Eq. 3.11 and Eq. 3.15 using a fourth-order Runge-Kutta numerical method and the appropriate boundary conditions:

$$f''' + ff'' = 0, \tag{3.11}$$

$$\phi'' + Scf\phi' = 0. \tag{3.15}$$

The calculated results, $\phi'(0)$ and $\phi'_\theta(0)$, are then converted to Y and B_m in our correlation using Eq. 3.24 and Eq. 3.25, which allows us to create a set of data points, $(\ln(1+B_m), Y)$, for each Schmidt number. The parameter a then determined by the least squares, best-fitting method. The procedure is repeated for all selected Schmidt numbers, and the calculated values of a are adjusted to yield the best fit for $A(Sc)$ in order to complete correlation Eq. 2.27.

4.1 Numerical Solution of the Boundary Layer Equations

For our computations we select the following values for the Schmidt numbers: Sc = 0.5, 0.7, 1.0, 10.0, and 100.0. We vary the value of $f(0)$ from zero to -0.875 by an appropriate step size. Then, for each value of $f(0)$, we calculate the values of $\phi'(0)$, and $\phi'_0(0)$ by solving the boundary layer equations, which are coupled and solved simultaneously with their appropriate boundary conditions. The fourth-order Runge-Kutta method used in the computations is described in Appendix A, and the FORTRAN program is listed in Appendix B.

Fig. 4.1 summarizes the behavior of the boundary layer solutions for different Sc.

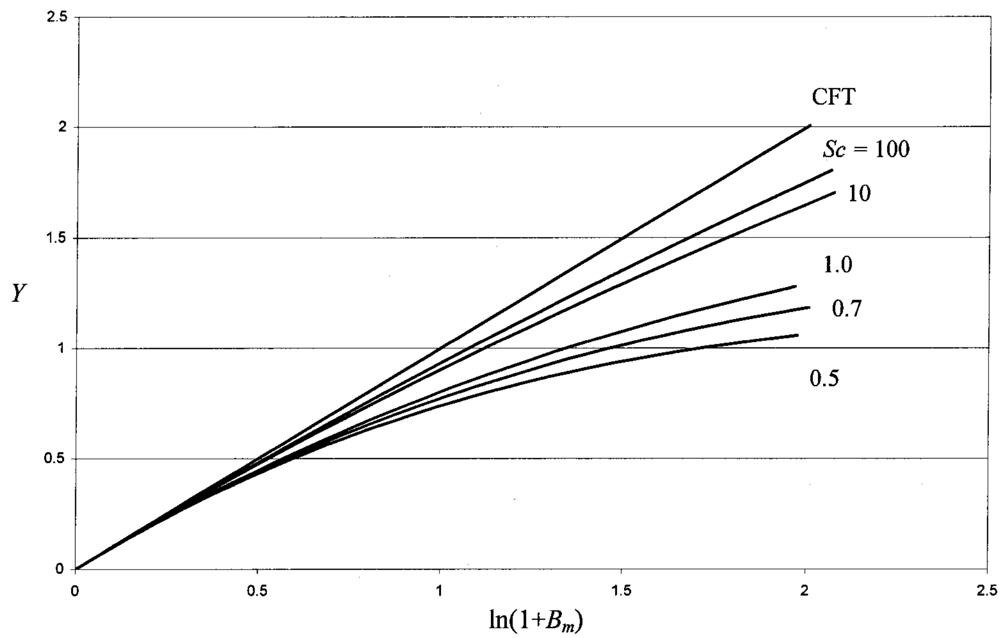

Figure 4.1 Normalized mass flux Y as a function of $\ln(1+B_m)$ for different Sc.

4.2 Determination of the coefficient *a*

We use a least squares method to determine the best value of a, which gives us the minimum deviation between Y calculated by the boundary layer equations and Y_A, the mass flux calculated by correlation Eq. 2.27. By repeating the procedure, we can determine the values of a for all selected values of Schmidt number.

Table 4.1 The coefficient a for selected values of Sc.

		A	B	$\lvert B\text{-}A \rvert / A$
Sc	a	$Sc^{1/3} / a$	$1.820Sc^{1/3}$	% Error
0.5	0.5586	1.421	1.445	1.689
0.7	0.5523	1.608	1.616	0.4975
1	0.5439	1.839	1.820	1.033
10	0.5064	4.254	3.921	7.828
100	0.4736	9.801	8.448	13.80

Table 4.1 shows the Schmidt numbers and the determined values of a. The calculated values of $A(Sc)$ from Eq. 2.28 are listed on the third column of the table. After averaging the values, we select a = 0.5495, which gives $A(Sc) = 1.82Sc^{1/3}$, and our correlation Eq. 2.27 becomes

$$Y_A = 1.820Sc^{1/3}\ln\left(1 + \frac{\ln(1+B_m)}{1.820Sc^{1/3}}\right) \tag{4.1}$$

Eq. 4.1 is the final product of the extended film theory, and the fourth column of Table 4.1 shows the calculated values of $A(Sc)$ with the selected value of a = 0.5495. Fig. 4.2 shows a graphical representation of the third and fourth columns of Table 4.1.

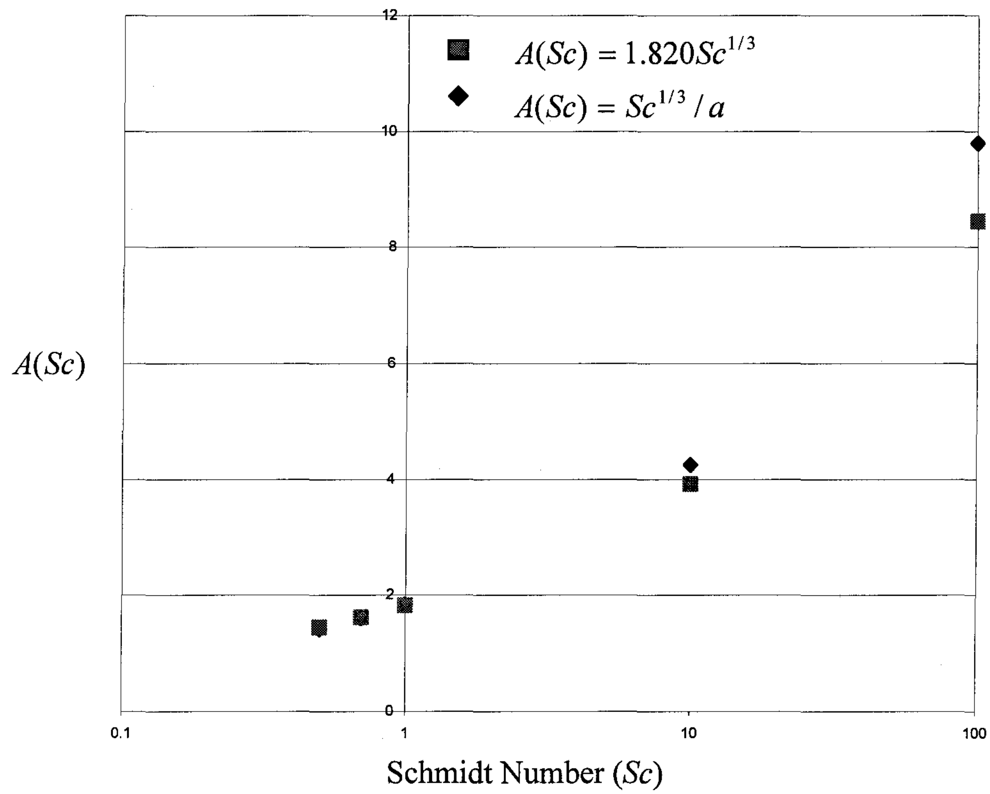

Figure 4.2 Calculated values of $A(Sc) = Sc^{1/3} / a$, and $A(Sc) = 1.820Sc^{1/3}$.

4.3 Predictive Value of the Extended Film Theory

The normalized mass flux given by the extended film theory Y_A in Eq. 4.1 is the correlated value, while Y from Eq. 3.24 is the "exact" value from the boundary layer equations. We can now compare graphically the two mass fluxes Y_A with Y for each value of the selected values of Schmidt number. The graphs are shown in Fig. 4.3 through Fig. 4.7. These graphs reveal that the deviations of Y_A from Y are minimum for all selected Schmidt numbers for a very wide range of mass transfer potentials.

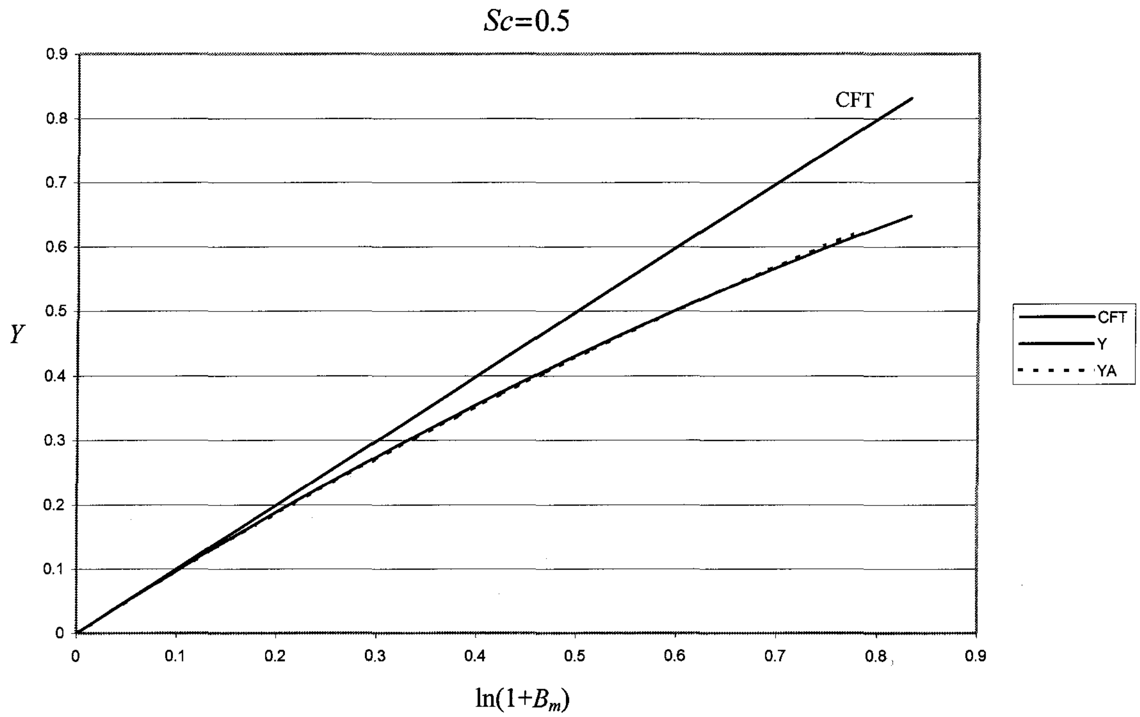

Figure 4.3 Y and Y_A for Sc = 0.5.

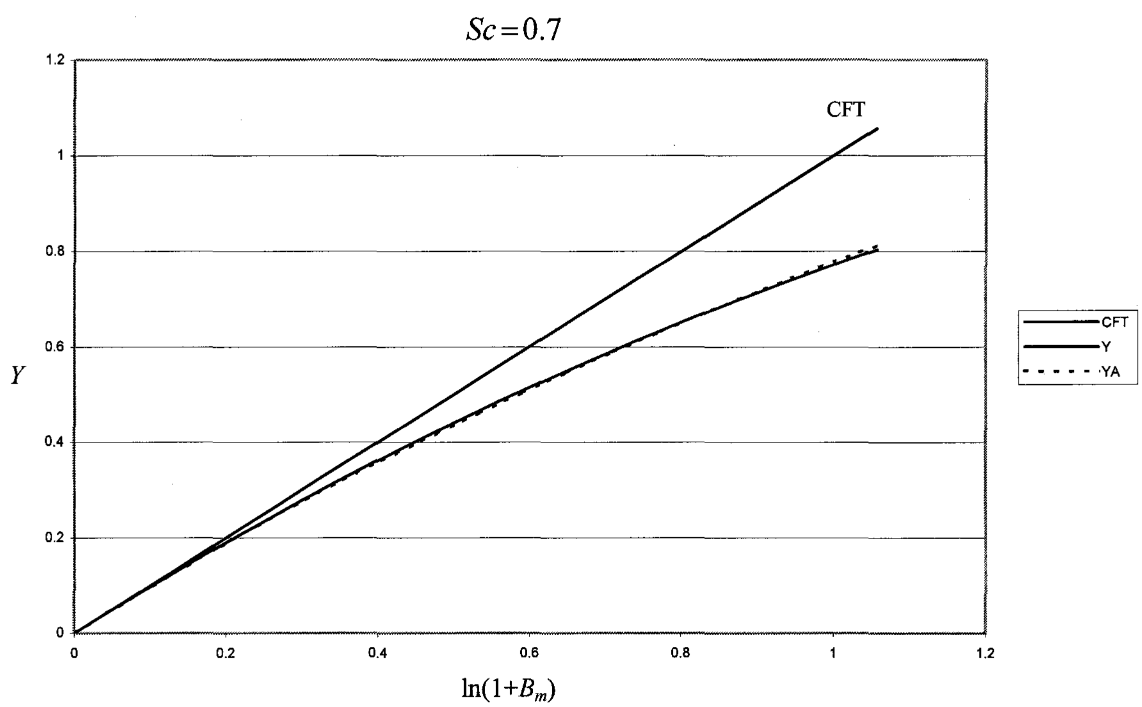

Figure 4.4 Y and Y_A for Sc = 0.7.

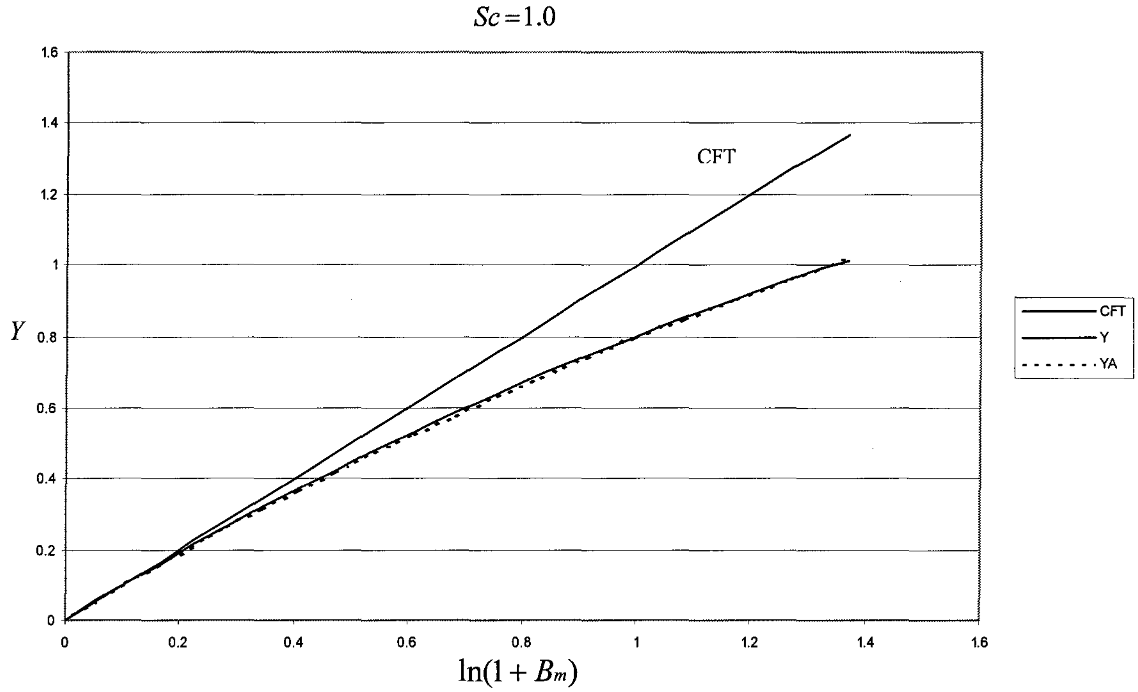

Figure 4.5 Y and Y_A for Sc = 1.0.

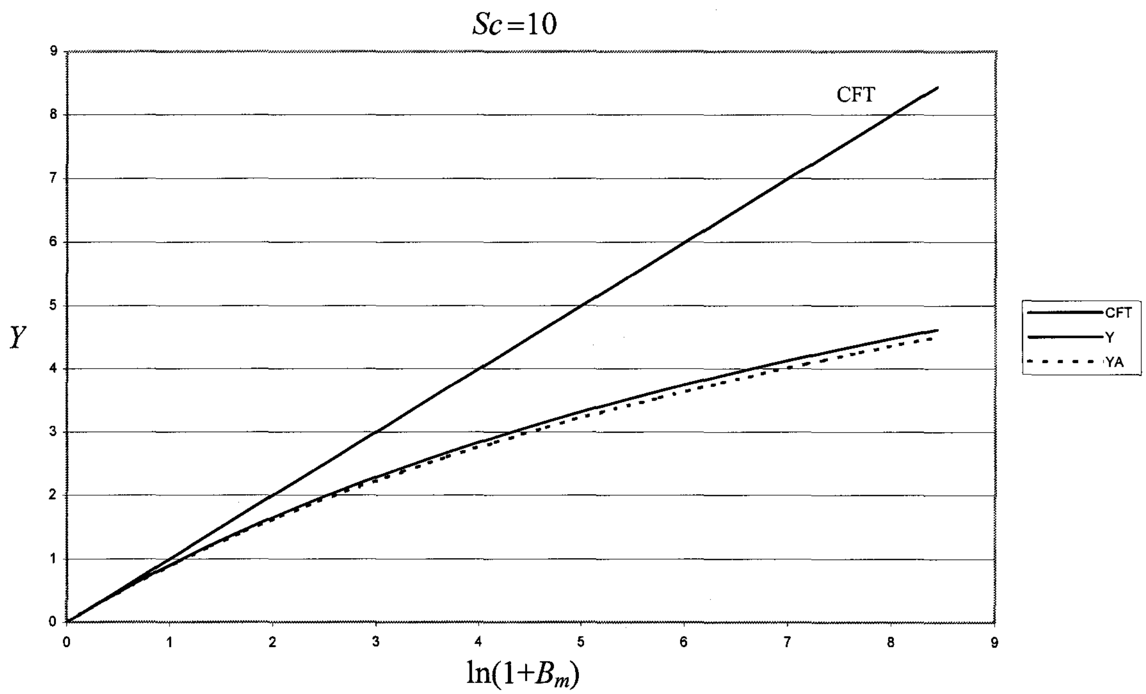

Figure 4.6 Y and Y_A for Sc = 10.

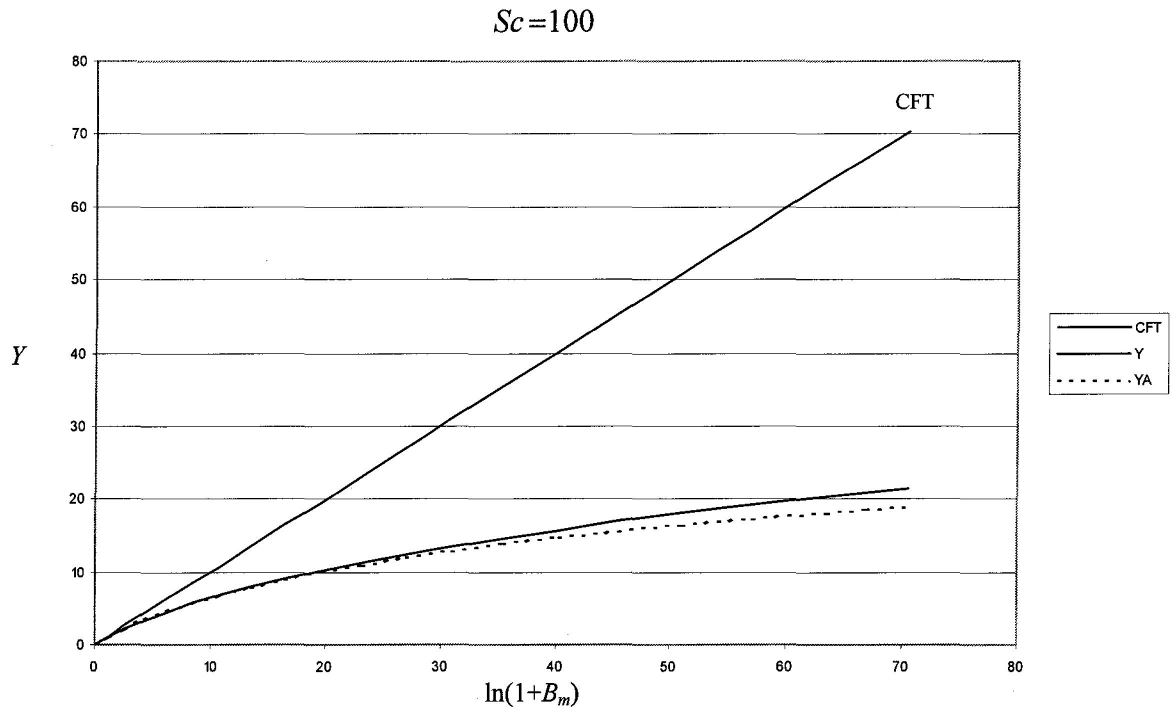

Figure 4.7 Y and Y_A for Sc = 100.

Chapter 5 Conclusions

In this study, we used a combination of physical insight, theoretical development and numerical experiments to develop a working correlation for the mass flux in a boundary layer flow over a porous (wet) plate. The correlation developed (Eq. 4.1) captures the main physics of the problem while maintaining a simple algebraic form that required only one parameter to be adjusted by numerical experiments.

The working correlation develop in this work represents very well the boundary layer solutions over a wide range of Schmidt numbers (0.5-100) and mass transfer potentials, B_m therefore validating the assumptions made to develop the functional form of the Y dependence on B_m. We find that our correlation compares favorably to others attempts to solve this or related problems in mass transfer (see e.g. Mills and Wortman, 1972; Wortman and Mills, 1974; Landis and Mills, 1972; Landis, 1971).

References

Bejan, A., Convective Heat Transfer, 2nd ed., New York: John Wiley & Sons, Inc. (1995).

Coimbra, C.F.M., Private communication. (2002).

Fox, R.W. and McDonald, A.T., Introduction to Fluid Mechanics, 5th ed., New York: John Wiley & Sons, Inc. (1998).

Hornbeck, R.W., Numerical Methods, New Jersey: Prentice-Hall, Inc. (1975).

Landis, R.B., and Mills, A.F., "The calculation of turbulent boundary layers with foreign gas injection," Int. J. Heat Mass Transfer, 15, 1905-1932 (1972).

Landis, R.B., "Numerical solution of variable property turbulent boundary layers with foreign gas injection," Ph. D. dissertation, School of Engineering and Applied Science, University of California, Los Angeles (1971).

Mills, A.F., Mass Transfer, New Jersey: Prentice Hall, Inc. (2001).

Schlichting, H., Boundary-Layer theory, 7th ed., New York: McGraw-Hill (1979).

White, F.M., Viscous Fluid Flow, 2nd ed., New York: McGraw-Hill (1991)

Wortman, A., "Mass transfer in self-similar boundary layer flows," Ph.D. dissertation, School of Engineering and Applied Science, University of California, Los Angeles (1969).

Wortman, A., and Wortman, A.F., "Accelerating compressible laminar boundary layers of binary gas mixtures," Archiwum Mechaniki Stosowanej, 26, 479-497 (1974).

Appendix A The Runge-Kutta and Shooting Methods

A1. The Fourth-Order Runge-Kutta Method

The basic format of the fourth-order Runge-Kutta method is the following. For a given differential equation, $\frac{dy}{dx} = R(x, y)$, and a given step size, h, the $i+1^{\text{th}}$ term solution is given by

$$y_{i+1} = y_i + 1/6(k_1 + 2k_2 + 2k_3 + k_4)h \tag{a1}$$

where

$$k_1 = R(x_i, y_i)$$

$$k_2 = R(x_i + 1/2h, y_i + 1/2k_1h)$$

$$k_3 = R(x_i + 1/2h, y_i + 1/2k_2h)$$

$$k_4 = R(x_i + h, y_i + k_3h)$$

The boundary layer equations and their boundary conditions with the initial guessed values are

$$\text{DE:} \quad f''' + ff'' = 0 \tag{3.11}$$

$$\text{BC:} \quad f'(0) = 0, \text{ and } f'(\infty) = 1 \tag{3.13}$$

$$-0.875 < f(0) = \text{Constant} < 0 \text{ (blowing/evaporation)} \tag{3.18}$$

$$\text{Initial guessed value: } f''(0) = 1 \tag{a2}$$

$$\text{DE:} \quad \phi'' + Scf\phi' = 0 \tag{3.15}$$

$$\text{BC:} \quad \phi(0) = 1, \text{ and } \phi(\infty) = 0 \tag{a3}$$

$$\text{Initial guessed value: } \phi'(0) = \text{-}1 \tag{a4}$$

The Blasius equation, Eq. 3.11, is decomposed into three coupled differential equations. They are

$$\frac{df''}{d\eta} = -ff'' = R_1(f'', f', f, \eta) \tag{a5}$$

$$\frac{df'}{d\eta} = f'' = R_2(f'', f', f, \eta) \tag{a6}$$

$$\frac{df}{d\eta} = f' = R_3(f'', f', f, \eta) \tag{a7}$$

To solve Eq. a5, Eq. a1 must be converted to

$$f''(\eta + h) = f''(\eta) + 1/6(k_1 + 2k_2 + 2k_3 + k_4)h$$

where

$$k_1 = R_1(f''(\eta), f'(\eta), f(\eta), \eta)$$

$$= -f(\eta)f''(\eta)$$

$$k_2 = R_1(f''(\eta) + 1/2k_1h, f'(\eta) + 1/2k_1h, f(\eta) + 1/2k_1h, \eta + 1/2h)$$

$$= -(f(\eta) + 1/2k_1h)(f''(\eta) + 1/2k_1h)$$

$$k_3 = R_1(f''(\eta) + 1/2k_2h, f'(\eta) + 1/2k_2h, f(\eta) + 1/2k_2h, \eta + 1/2h)$$

$$= -(f(\eta) + 1/2k_2h)(f''(\eta) + 1/2k_2h)$$

$$k_4 = R_1(f''(\eta) + k_3h, f'(\eta) + k_3h, f(\eta) + k_3h, \eta + h)$$

$$= -(f(\eta) + k_3h)(f''(\eta) + k_3h)$$

Eq. a6 and Eq. a7 are solved in the similar manner.

To solve Eq. a5 for f'', f must be known. Similarly, to solve Eq. a6 for f', f'' must be known, and to solve Eq. a7 for f, f' must be known. Therefore, these three differential equations must be coupled and solves simultaneously with their boundary conditions.

In Eq. a2, the value of $f''(0)$ is initially guessed. The appropriate value of $f''(0)$ for a given value of $f(0)$ is determined by the shooting method (section A2). Once the appropriate value of $f''(0)$ is found, the Runge-Kutta method is repeated once more with the found value of $f''(0)$ to determine $f(\eta)$ for all values of η. Then, the determined values of $f(\eta)$ are used to solve the species conservation equation, Eq. 3.15, which is decomposed into two coupled differential equations. They are

$$\frac{d\phi'}{d\eta} = -Sc\, f\, \phi' \tag{a8}$$

$$\frac{d\phi}{d\eta} = \phi' \tag{a9}$$

As in the case of the Blasius equation, two coupled differential equations and their boundary conditions are solved simultaneously through the fourth-order Runge-Kutta method. In Eq. a4, the appropriate value of $\phi'(0)$ for a given value of $f(0)$ is determined by the shooting method (section A2).

A2. The Shooting Method

To solve the Blasius equation, Eq. 3.11, we use the fourth-order Runge-Kutta method with the boundary condition, Eq. 3.13, along with the initial guessed value of $f''(0)$. At the end of each Runge-Kutta routine, we get the calculated values of $f(\infty)$, $f'(\infty)$, and $f(\infty)$. Then, we can compare the calculated value of $f'(\infty)$ with the BC:

$f'(\infty) = 1$. If the calculated value of $f'(\infty)$ is larger than one, we subtract the correction factor, $\Delta f''$, from the initial guessed value of $f''(0)$. We repeat the Runge_Kutta routine by subtracting the correction factor, $\Delta f''$, from $f''(0)$ until $f'(\infty) < 1$ is reached. Once $f'(\infty) < 1$ is reached, we divide the correction factor by two, or $\Delta f''/2$, and repeat the Runge_Kutta routine.

Oppositely, if the first Runge_Kutta routine yields $f'(\infty) < 1$, we add the preset correction factor, $\Delta f''$, to the initial guessed value of $f''(0)$. We repeat the Runge-Kutta routine by adding the correction factor, $\Delta f''$, to $f''(0)$ until $f'(\infty) > 1$ is reached. Once $f'(\infty) > 1$ is reached, the correction factor is divided by two, or $\Delta f''/2$, and we repeat the Runge-Kutta routine.

In this manner, The Runge-Kutta routine is repeated until the calculated value of $f'(\infty)$ converges to one within the preset tolerance.

We perform the same procedure for the species conservation equation, Eq. 3.15. This time, the guessed value is $\phi'(0)$, and the boundary condition to be checked is $\phi(\infty) = 0$. After each Runge-Kutta routine, the calculated value of $\phi(\infty)$ is compared with zero. Then, the guessed value of $\phi'(0)$, and the preset correction factor, $\Delta\phi'$, are adjusted accordingly. The Runge-Kutta routine is repeated until the calculated value of $\phi(\infty)$ converges to zero within the preset tolerance.

The examples of the shooting technique are shown in Fig. A1 and A2. We set Sc=1.0 and $f(0)$=0, and we obtained the following values: $f''(0) = 0.4696$, and $\phi(0) = -0.46915$.

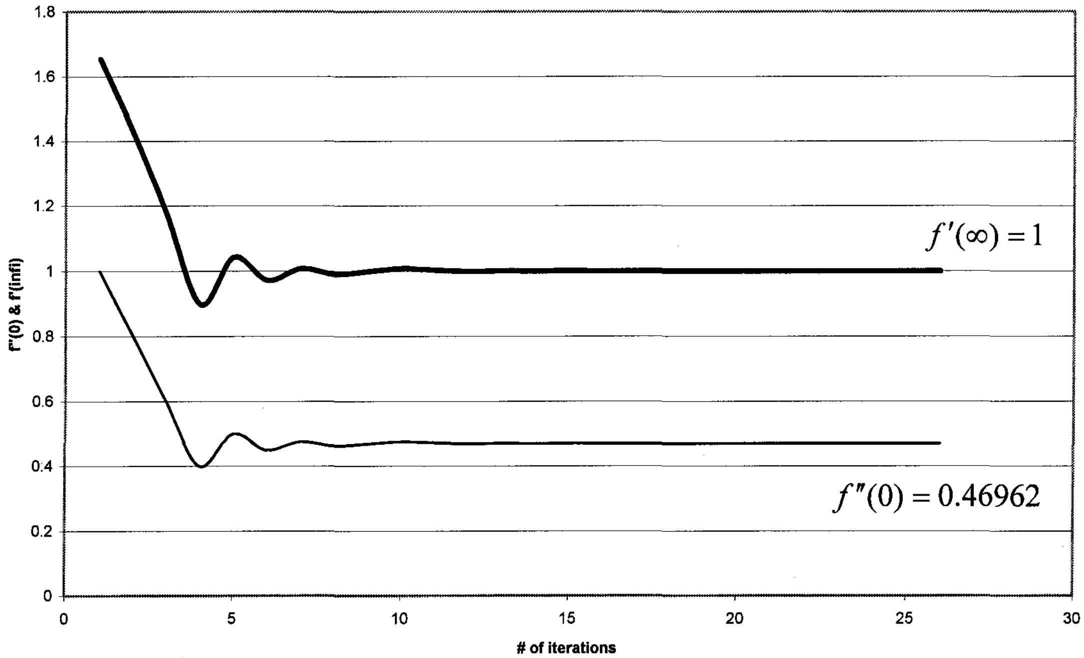

Figure A1. The convergence of $f''(0)$ for the condition $f'(\infty)=1$

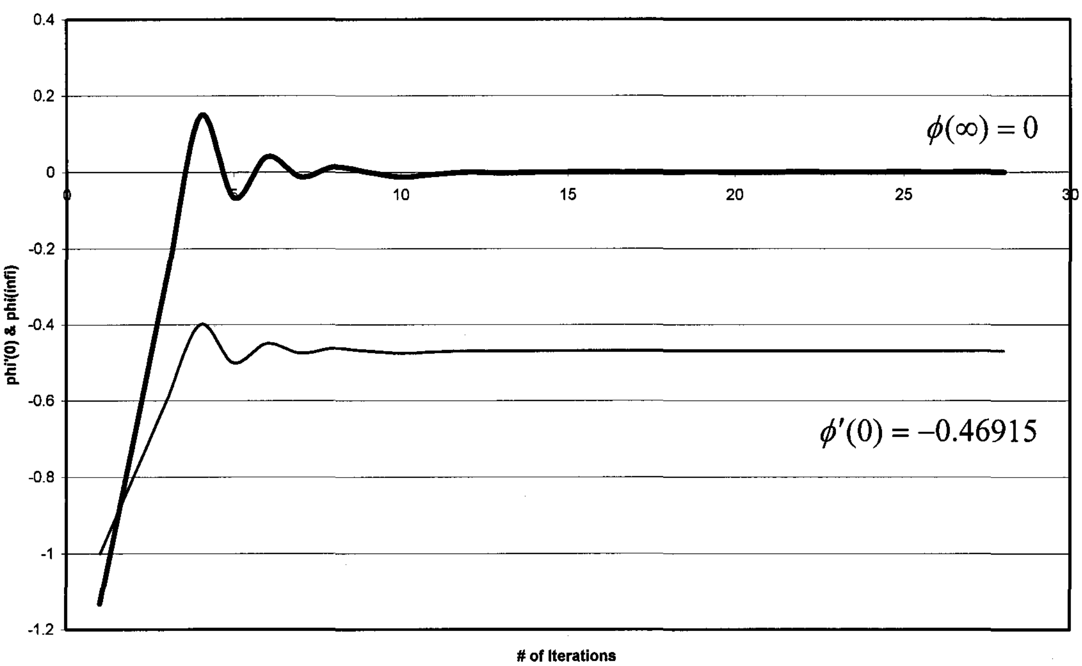

Figure A2. The convergence of $\phi'(0)$ for the condition $\phi(\infty)=0$

Appendix B CODE Listing: Numerical Solutions for Boundary Layer Equations

```
program thesis
!*********************************************************************
! RK100.f                                              August 20, 2002
!
! Programmed by: Fuminori Nakamura
!
! Purpose: This program performs the numerical integration of the Blasius
!          equation with mass transfer.
!
! The program solves the system of first order ordinary differential equations
! resulting from a self-similar laminar flat plate boundary layer with normal
! transpiration. The five first order equations are integrated through the four
! stage Runge-Kutta method.
!
! The following variables are used in the program.
!
! y(i)   Main variables
! ------------------------------------------------------------
! y(1)   f"   : nondimensional acceleration.
! y(2)   f'   : nondimensional velocity, u/Ue.
! y(3)   f    : nondimensional stream function.
! y(4)   phi' : derivative of normalized mass concentration.
! y(5)   phi  : normalized mass concentration.
! ------------------------------------------------------------
!
! Output file: RKOUT100
!
! Note: The output file, RKOUT100 will be used by Y_Bm.f program to create a
!       table, which contains the following data: Sc(Schmidt number), f(0),
! phisubzero, phi', ln(1+Bm), and Y. The output data from Y_Bm.f is transferred
! to Excel to crate graphs of ln(1+Bm) vs Y lines.
!
! Modification: RK100.f program deals with a wide range of Schmidt numbers from
!               0.01 to 1000.0. Therefore, to create graphs in Excel with a
! per-selected domain, 0<ln(1+Bm)<2.0, RK100.f program needs to be modified
! for larger Schmidt numbers to produce the appropriate data ranges. To do so,
! Schmidt-do-loop, and f0-do-loop at the beginning of the program need to be
! modified. Schmidt-do-loop selects Schmidt numbers, and f0-do-loop sets the
! corresponding step size.
!
!*********************************************************************
implicit doubleprecision(A-H,O-Z)
```

```
doubleprecision y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)
doubleprecision f(20005),schmidt(10)

OPEN(6,FILE='RKOUT100',STATUS='UNKNOWN')
write(*,*)" Sc      phi_psub0(0) phi_p(0)       f(0)"
write(*,*)"-------------------------------------------"

write(6,*)" Sc      phi_psub0(0) phi_p(0)       f(0)"
write(6,*)"-------------------------------------------"

Call Initialize_Sc(schmidt)
Call Get_Sc(schmidt)

do I=1,4  ! Schmidt loop
  Sc=schmidt(I)

do f0=0,-0.875D00,-0.025D00  ! f0 loop
  fpp=1.0D00     ! initialize f''(0)

Call Ak(Ak1,Ak2,Ak3,Ak4)  ! initialize k(I)=0

y(1)=fpp    ! f''(0)=1  guessed initial value. It will be calculated.
y(2)=0      ! f'(0)=0  u/Ue=0, no-slip condition.
y(3)=f0     ! f(0)=f0,  preset value, a negative constant, -0.875 ~ 0.
y(4)=-1     ! phi'(0)=-1 guessed initial value. It will be calculated.
y(5)=1      ! phi(0)=1, preset condition.

step=0.001D00   ! step size
eta_max=20.0D00 ! Max range of eta
L=eta_max/step  ! Number of loop turns.
d_fpp=0.2D00    ! correction of f''(0)
 tole_fp=1.0D-6  ! tolerance of f'(@)-1<tole_fp.  @ --- infinity.

!*********************************************************************
!
! y(1)=f''(0) is the guessed value. We determine the appropriate y(1)=f''(0)
! value by checking the condition: y(1)@=f''(@)=1. @ --- infinity.
!
!*********************************************************************
prev_fp=1.5D00 ! previous f'(@)value  @ --- infinity.
Call Get_f(y,Ak1,Ak2,Ak3,Ak4,step,L)

y(1)=fpp  ! After Get_f, y(1) contains f''(@). We need to reassign f''(0).

do while (abs(y(2)-1)>tole_fp)
```

```
  if ((prev_fp-1)*(y(2)-1)<0)then  ! if two consecutive f'(@) values are
    d_fpp=0.5D00*d_fpp             ! located between 1, then decrement
  endif                            ! the correction of f''(0), d_fpp, by
                                   ! a half.
     if (y(2)>1) then     ! if f'(@)>1 then
    y(1)=y(1)-d_fpp       !   set f''(0)= decrement by d_fpp
  else                    ! else
    y(1)=y(1)+d_fpp       !   increment by d_fpp
  endif

  fpp=y(1)                ! Keep f''(0) value in fpp before get into Get_f
                          ! Get_f erases the value of f''(0)

  prev_fp=y(2)            ! Hold f'(@) value into prev_fp,before getting
                          ! the next f'(@) value in Get_f
  Call Ak(Ak1,Ak2,Ak3,Ak4)  ! initialize k(I)=0

  y(2)=0                  ! Initialize f'(0) and f(0) before getting into
  y(3)=f0                 ! Get_f.

  Call Get_f(y,Ak1,Ak2,Ak3,Ak4,step,L)

   y(1)=fpp               ! Assign back the value of f''(0) into y(0)

   enddo                  ! do while loop for f''(0)

!*******************************************************************
!
! After appropriate f''(0) value is found for the preset Schmidt number, Runge-
! Kutta routine is performed once more to make f(eta) array of L=eat_max/step
! elements. f(eta) array is used for finding phi'(0) value.
!
!*******************************************************************
Call Ak(Ak1,Ak2,Ak3,Ak4)   ! initialize k(I)=0

y(1)=fpp                   ! initialize variables
y(2)=0
y(3)=f0

Call Make_f_arrey(f,y,Ak1,Ak2,Ak3,Ak4,step,L,f0)

!*******************************************************************
!
! Use the similar method as finding f''(0), the following procedure finds phi'(0).
```

```
!
!*********************************************************************
d_phi=0.2D00          ! correction of phi'(0)
phi0=1.0              ! initial set phi(0)=1. Condition: phi(@) goes to zero.
phi_p0=-1.0           ! guessed value phi'(0)=-1.0
prev_phi=-1.0         ! set the previous value of phi(0)=y(5)=-1.0
tole_phi=1.0D-6       ! tolerance of phi(@)=y(5)@. phi(@) must converge to zero.

Call Ak(Ak1,Ak2,Ak3,Ak4)  ! initialize k(I)=0

y(4)=phi_p0           ! y(4)0=phi'(0)=-1.0, a guessed value.
y(5)=phi0             ! y(5)0=phi(0)=1.0, an initial condition.

!Get the first value of phi(@)=y(5)
Call Get_phi(Sc,f,y,Ak1,Ak2,Ak3,Ak4,step,L)

y(4)=phi_p0           ! reassign the guessed phi'(0) value to y(4)
                      ! At this point, y(5)is y(5)@ value.

do while (abs(y(5))>tole_phi) ! do while y(5)@=phi(@)>tolerance

  if (prev_phi*y(5)<0) then   ! If zero is located between the previous
    d_phi=0.5D00*d_phi        ! phi and current phi values, then decrement
  endif                       ! the correction d_phi by half.

  if (y(5)>0) then            ! If phi(@)>0, reduce the guessed phi'(0) value by
    y(4)=y(4)-d_phi           !   d_phi.
  else                        ! else increment phi'(0) by d_phi.
    y(4)=y(4)+d_phi
  endif

  prev_phi=y(5)       ! keep the current phi(@) value in prev_phi.
  phi_p0=y(4)         ! keep the current phi'(0) value in phi_p0

  ! reinitialize
  Call Ak(Ak1,Ak2,Ak3,Ak4)  ! initialize k(I)=0
  y(5)=phi0           ! phi(0)=y(5)= phi0=1.0
                      ! phi'(0)=y(4) --- adjusted by d_phi

  Call Get_phi(Sc,f,y,Ak1,Ak2,Ak3,Ak4,step,L)

  y(4)=phi_p0         ! y(4) and phi_p0 hold phi'(0) values after the loop

enddo                 ! do while loop for determining phi'(0)
```

```
    if (f0==0) then
      phi_p_sub0=y(4) ! assign phi_p(0) to phi_p_subzero when f(0)=0,
    endif                    !( dry condition)

    write(*,20) Sc,phi_p_sub0,y(4),f0
    write(6,20) Sc,phi_p_sub0,y(4),f0

  enddo   ! f0 loop

  enddo   ! Schmidt loop


   write(*,20) -99.9D00,-99.9D00,-99.9D00,-99.9D00  !  a trailer data.
   write(6,20) -99.9D00,-99.9D00,-99.9D00,-99.9D00

 20    format(F8.2,3X,F10.6,3X,D14.6,3X,F8.3)
close(6)
END

!********************************************************************
! SUBROUTINE Get_phi(Sc,f,y,Ak1,Ak2,Ak3,Ak4,step,L)
!
! Purpose:  This subroutine is called in by the main procedure after the
!           determination of f''(0) value and creation of f(I) array. This
! subroutine determines phi' and phi values, which will be stored in y(4) and
! y(5) correspondingly. Two subroutines are called in to perform the Runge-
! Kutta method, and calculations of phi' and phi values.
!
!********************************************************************
SUBROUTINE Get_phi(Sc,f,y,Ak1,Ak2,Ak3,Ak4,step,L)
implicit doubleprecision(A-H,O-Z)
doubleprecision f,y,Ak1,Ak2,Ak3,Ak4

do I=1,L
   Call RK2(Sc,f,y,Ak1,Ak2,Ak3,Ak4,step,I,L)
   Call Next_y_phi(y,Ak1,Ak2,Ak3,Ak4,step)
enddo
return
end

!********************************************************************
! SUBROUTINE Next_y_phi(y,Ak1,Ak2,Ak3,Ak4,h)
!
! Purpose:  This subroutine calculates phi' and phi vales, which will be stored
!           in y(4), and y(5).
```

```
!
! Called by: Get_phi
!
!***********************************************************************
SUBROUTINE Next_y_phi(y,Ak1,Ak2,Ak3,Ak4,h)
implicit doubleprecision(A-H,O-Z)
doubleprecision y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)
do I=4,5
  y(I)=y(I)+(1.0/6.0)*(Ak1(I)+2.0*Ak2(I)+2.0*Ak3(I)+Ak4(I))*h
enddo

return
end

!***********************************************************************
! SUBROUTINE RK2(Sc,f,y,Ak1,Ak2,Ak3,Ak4,h,I,L)
!
! Purpose:  This subroutine calculates the new k1, k2, k3 and k4 values for the
!           Runge-Kutta method. The calculated values will be stored in AK1,
! AK(2), Ak(3), and Ak(4) arrays.
!
! Called by: Get_phi
!
!***********************************************************************
SUBROUTINE RK2(Sc,f,y,Ak1,Ak2,Ak3,Ak4,h,I,L)
implicit doubleprecision(A-H,O-Z)
doubleprecision f(L),y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

Ak1(4)=-Sc*f(I)*y(4)
Ak1(5)=y(4)

Ak2(4)=-Sc*f(I)*(y(4)+0.5D00*Ak1(4)*h)
Ak2(5)=y(4)+0.5D00*Ak1(5)*h

Ak3(4)=-Sc*f(I)*(y(4)+0.5D00*Ak2(4)*h)
Ak3(5)=y(4)+0.5D00*Ak2(5)*h

Ak4(4)=-Sc*f(I)*(y(4)+Ak3(4)*h)
Ak4(5)=y(4)+Ak3(5)*h

return
end

!***********************************************************************
! SUBROUTINE Make_f_arrey(f,y,Ak1,Ak2,Ak3,Ak4,step,L,f0)
```

```
!
! Purpose:  This subroutine is called in by the main procedure after the
!           determination of f''(0) value for the preset value of Schmidt number.
! This subroutine creates f(I) array which contains f values. ( f(I) array will
! be used to determine phi'(0) value. )
!
!*********************************************************************

SUBROUTINE Make_f_arrey(f,y,Ak1,Ak2,Ak3,Ak4,step,L,f0)
implicit doubleprecision(A-H,O-Z)
doubleprecision f(L),y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

f(1)=f0 ! Array f(1) contains f0, the initial setting value.

do I=1,L
  Call RK1(y,Ak1,Ak2,Ak3,Ak4,step)
  Call Next_y_f(f,I,y,Ak1,Ak2,Ak3,Ak4,step)
enddo

return
end

!*********************************************************************
! SUBROUTINE Next_y_f(f,I,y,Ak1,Ak2,Ak3,Ak4,h)
!
! Purpose:  This subroutine calculates the values of f'', f', and f, which are
!           stored in y(1),, y(2), and y(3). Values of f are assigned in f(I) array.
!
! Called by: Make_f_array
!
!*********************************************************************
SUBROUTINE Next_y_f(f,I,y,Ak1,Ak2,Ak3,Ak4,h)
implicit doubleprecision(A-H,O-Z)
doubleprecision f(20005),y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

do J=1,3
  y(J)=y(J)+(1.0/6.0)*(Ak1(J)+2.0D00*Ak2(J)+2.0D00*Ak3(J)+Ak4(J))*h
enddo

f(I+1)=y(3)  ! f(1) contailns f0 value. f(2) contains the next value of f.

return
end

!*********************************************************************
```

```
! SUBROUTINE Get_f(y,Ak1,Ak2,Ak3,Ak4,step,L)
!
! Purpose:  This subroutine calculates the new f'', f', and f values, which will
!           be stored in y(1), y(2), and y(3) arrays. Two subroutines, RK1,
! and Next_y, are called to process the Runge-Kutta method.
!
!*********************************************************************
SUBROUTINE Get_f(y,Ak1,Ak2,Ak3,Ak4,step,L)
implicit doubleprecision(A-H,O-Z)
doubleprecision y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

do I=1,L
  Call RK1(y,Ak1,Ak2,Ak3,Ak4,step)
  Call Next_y(y,Ak1,Ak2,Ak3,Ak4,step)
enddo

return
end

!*********************************************************************
! SUBROUTINE Next_y(y,Ak1,Ak2,Ak3,Ak4,h)
!
! Purpose:  This subroutine calculates the new f'', f', and f values, which will
!           be stored in y(1), y(2), and y(3) correspondingly. k1, k2, and k3
! values of the Runge-Kutta method are stored in Ak1, Ak2, and Ak3 arrays
! correspondingly.
!
! Called by: Get_f
!
!*********************************************************************
SUBROUTINE Next_y(y,Ak1,Ak2,Ak3,Ak4,h)
implicit doubleprecision(A-H,O-Z)
doubleprecision y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

do I=1,3
  y(I)=y(I)+(1.0/6.0)*(Ak1(I)+2.0D00*Ak2(I)+2.0D00*Ak3(I)+Ak4(I))*h
enddo

return
end

!*********************************************************************
! SUBROUTINE RK1(y,Ak1,Ak2,Ak3,Ak4,h)
!
! Purpose:  This subroutine calculates new k1, k2, k3 and k4 values for the Runge-
```

```
!          Kutta method. The calculated values will be stored in AK1, Ak(2),
! Ak(3), and Ak(4) arrays.
!
! Called by: Get_f, Make_f_array
!
!*******************************************************************
SUBROUTINE RK1(y,Ak1,Ak2,Ak3,Ak4,h)
implicit doubleprecision(A-H,O-Z)
doubleprecision y(5),Ak1(5),Ak2(5),Ak3(5),Ak4(5)

Ak1(1)=-y(3)*y(1)
Ak1(2)=y(1)
Ak1(3)=y(2)

Ak2(1)=-(y(3)+0.5D00*Ak1(3)*h)*(y(1)+0.5D00*Ak1(1)*h)
Ak2(2)=y(1)+0.5D00*Ak1(1)*h
Ak2(3)=y(2)+0.5D00*Ak1(2)*h

Ak3(1)=-(y(3)+0.5D00*Ak2(3)*h)*(y(1)+0.5D00*Ak2(1)*h)
Ak3(2)=y(1)+0.5D00*Ak2(1)*h
Ak3(3)=y(2)+0.5D00*Ak2(2)*h

Ak4(1)=-(y(3)+Ak3(3)*h)*(y(1)+Ak3(1)*h)
Ak4(2)=y(1)+Ak3(1)*h
Ak4(3)=y(2)+Ak3(2)*h

return
end

!*******************************************************************
! SUBROUTINE Ak(Ak1,Ak2,Ak3,Ak4)
!
! Purpose:  This subroutine initializes AK(I) array to zeros.
!
!*******************************************************************
SUBROUTINE Ak(Ak1,Ak2,Ak3,Ak4)
implicit doubleprecision(A-H,O-Z)
doubleprecision Ak1(5),Ak2(5),Ak3(5),Ak4(5)

do I=1,5
  Ak1(I)=0
  Ak2(I)=0
  Ak3(I)=0
  Ak4(I)=0
enddo
```

```
return
end

!*******************************************************************
! SUBROUTINE Initialize_Sc(Schmidt)
!
! Purpose:  This subroutine initializes schmidt(I) array to zeros.

!!*******************************************************************
SUBROUTINE Initialize_SC(schmidt)
implicit doubleprecision(A-H,O-Z)
doubleprecision schmidt(10)

do I=1,10
  schmidt(I)=0
enddo

return
end

!*******************************************************************
! SUBROUTINE Get_SC(schmidt)
!
! Purpose:  This subroutine assigns Schmidt numbers to schmidt(I) array.

!*******************************************************************
SUBROUTINE Get_SC(schmidt)
implicit doubleprecision(A-H,O-Z)
doubleprecision schmidt(10)

schmidt(1)=0.02D00
schmidt(2)=0.5D00
schmidt(3)=0.7D00
schmidt(4)=1.0D00
schmidt(5)=10.0D00
schmidt(6)=100.0D00
schmidt(7)=1000.0D00

return
end
```