NUMERICAL METHODS FOR SOLVING ONE DIMENSIONAL PROBLEMS WITH A MOVING BOUNDARY

BY

ABDELLATIF-BOUREGHDA

A THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN THE DEPARTMENT OF COMPUTING SCIENCE AT THE UNIVERSITY OF GLASGOW

SEPTEMBER 1988

-SCOTLAND-

Acknowledgements

I am deeply indebted to my supervisors Professor D.C.Gilles and Dr. R.D.Mills for suggesting these problems, their guidance, interest, and many helpful discussions during this period of this research.

I am also grateful to other members of staff in the department for their assistance and help.

A.Boureghda

Contents

		page
Summary		1
Chapter one :	**Introduction**	2
	1.1. Moving boundary problems	2
	1.2. Formulation of the problem	3
	1.3. Generalizations of the classical problem	6
	1.4. Other moving boundary problems	7
Chapter two :	**Finite differences**	9
	2.1. Introduction	9
	2.2. Boundary conditions	11
	2.3. Parabolic equation	12
	2.4. Methods of numerical solution	13
	2.5. Finite difference grids	13
	2.6. Irregular boundaries	16
	2.7. Explicit and implicit numerical methods	18
	2.8. Explicit formula	18
	2.9. The Dufort-Frankel explicit approximation	19
	2.10. Implicit formula	21
	2.11. Douglas formula	22
	2.12. Convergence,stability & consistency	22

2.13. Non-Linear equation 23

Chapter three: Modified variable time step method for solving Ice melting problem 25

3.1. One phase moving boundary 25

3.2. Problem 26

3.3. Methods of solution 26

3.4. Method of Douglas and Gallie 27

3.5. Method of Gooding and Khader 29

3.6. Method of Gupta and Kumar 30

3.7. Boundary conditions melting ice 30

3.8. Solution of ice melting using Douglas implicit in replacement of $u_t=u_{xx}$ 36

3.9. Numerical solutions of cartesian problem 38

3.10. Cylindrical problem 39

3.11. The two cylindrical problems 41

3.12. Numerical solutions of cylindrical problem 46

Chapter four : Solution of ice melting problem using fixed domain method with moving boundary 48

4.1. Several methods 48

4.2. Problem 49

4.3. Numerical results 55

Chapter five : Oxygen diffusion problem 56

5.1. Flow example 56

5.2. Cartesian coordinates :Statement of problem 56

5.3. Approximate Analytical solution 58

5.4. Numerical method 60

5.5. The Du Fort-Frankel difference scheme 66

5.6. Berger et al problem 68

5.7. Cylindrical problem 69

5.8. Oxygen diffusion inside problem 72

5.9. Approximate analytical solution 72

5.10. Oxygen diffusion outside problem 76

5.11. Numerical solution of cylindrical problems 78

Chapter six : Conclusion and numerical results 81

6.1. Summary 82

6.2. Numerical results for ice melting problem 84

6.3. Numerical results for oxygen diffusion problem 105

Refferences : 116

Appendices : 121

Summary

This thesis describes some methods which were employed to compute approximate solutions to partial differential equations with moving boundaries_verifying the existing numerical solutions, modifying the techniques and applying them to new problems. The problems considered were the determination of the temperature in melting ice and of the concentration of oxygen diffusion in both one dimensional cartesian and axially symmetric cylindrical coordinates.

For the melting ice problem the methods studied included variable time step methods with difference formulae and different methods of calculating the variable time step, and also a transformation to fix the moving boundary using a conventional finite difference technique on the known domain. The diffusion problem had a singularity on the initial boundary. The singularity was treated by using an approximate analytical solution and the numerical solution found by a finite difference method with fixed time and space steps and a Lagrange-type formula near the moving boundary.

Chapter-1-

Introduction

1.1. Moving boundary problems

Boundary-value problems are problems involving the solution of a differential equation satisfying certain conditions on the boundary of a prescribed domain . Moving boundary-value problems (MBP's), on the other hand, are associated with time dependent problems and the position of the boundary has to be determined as function of space and time.

The practical applications arise, for example in metal, glass, plastics and oil industries, space vehicle design, chemical diffusion, biological processes, statiscal decision theory, meteorology, shock waves in gas dynamics, crack problems in solid mechanics and many others (see Ockendon & Hodgkins 1975, Furzeland 1976).

Heat flow or diffusion involving phase changes from solid, liquid or vapor states constitute a large class of moving boundary problems. MBP's are often called Stefan problems with reference to early work of J.Stefan who around 1890, was interested in the most common example which is the melting or freezing of water.

The numerical solution of such problems has either concentrated on many different methods for solving the classical

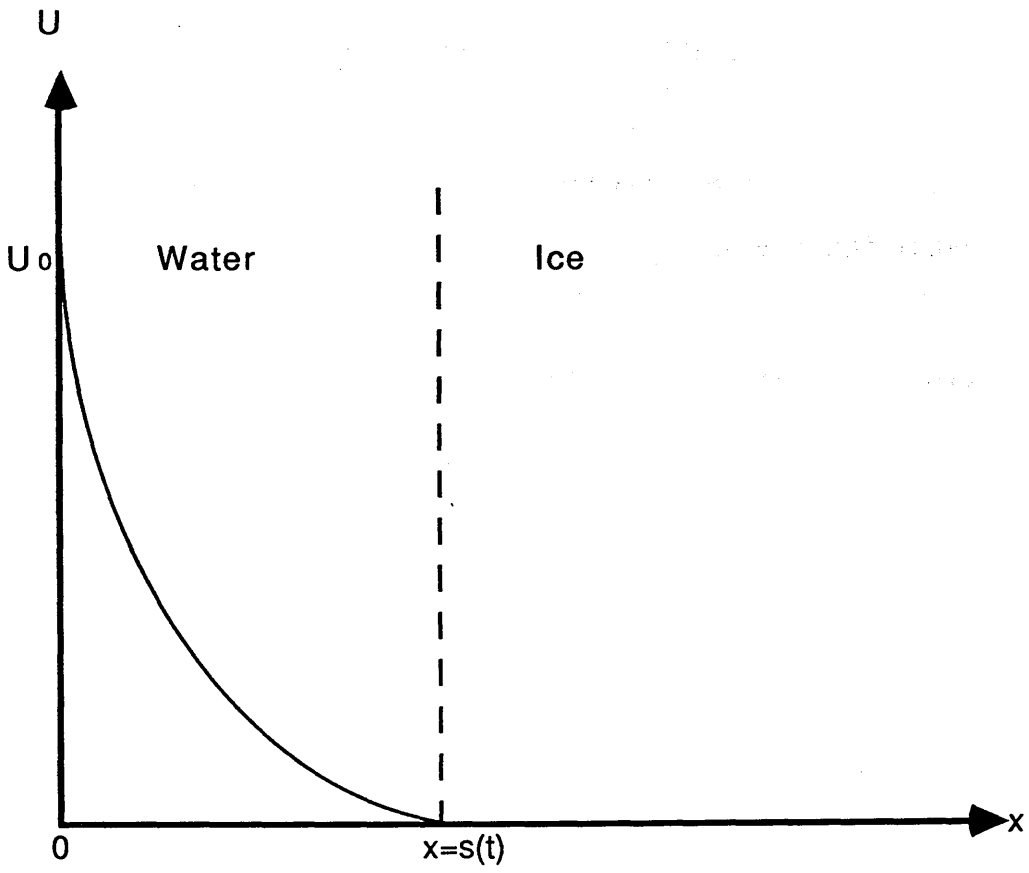
problems or one-off methods for more general types of MBP (see Fox in Ockendon & Hodgkins 1975).

Successive authors using numerical methods have referred to Stefan's publications. Earlier surveys have mainly concentrated on the one space dimensional problem and have been written by Bankoff (1964), Muelhbaurer and Sunderland (1965), Rubinstein (1971), Boley (1972), Crank (1975). More recent surveys which included multi-dimensional applications are given by Crank and Fox in Ockendon & Hodginks (1975), Meyer (1975a, 1976), Hoffman (1977 vol 1-3), Furzeland (1977a), Fox (1979) and Crank (1981) all with useful bibliographies.

Reports on several conferences (Ockendon & Hodgkins 1975), Wilson, Solomon and Boggs (1978), Furzeland (1979), Magnes (1980), Albbrecht, Collatz and Hoffman (1980), Fasano and Premicerio (1983), are most comprehensive and contain a lot of practical applications.

1.2. Formulation of the problem

A simple version of the Stefan problem is the one phase, one dimensional ice problem, which consists of finding the pair of unknowns $\{u(x,t),s(t)\}$, where $u(x,t)$ is the temperature distribution and $x=s(t)$ the position of the ice/melted water interface (the moving boundary or MB). See Fig(1.1).



Fig(1.1). Simple Stefan problem

The governing partial differential equation is

$$\partial u/\partial t=\partial^2 u/\partial x^2 \quad (0<x<s(t),\ t>0), \tag{1.1}$$

the fixed boundary condition is

$$u=u_0 \qquad \text{on } x=0,\ t>0 \tag{1.2}$$

the initial conditions are

when t=0

$$u=0 \tag{1.3}$$

$$s=0 \tag{1.4}$$

and the moving boundary conditions are

on the MB $x=s(t)$ $t>0$

$$u=0 \tag{1.5}$$

$$\partial u/\partial x=-ds/dt \tag{1.6}$$

Equation (1.6) is known as the Stefan condition.

The problem is one phase because we assume that in the semi infinite block of ice $x>s(t)$ the temperature $u(x,t)$ remains at $u=0$.

The corresponding two phase problem is to find the triple of unknowns $\{u_1(x,t),u_2(x,t),s(t)\}$, where u_1 and u_2 denote temperatures in water and ice phases respectively. A typical example is a finite sheet of ice occupying the space $0\leq s(t)\leq x\leq 1$ where $u_2(x,t)$ is not constant in this case and there is one governing equation for each phase,

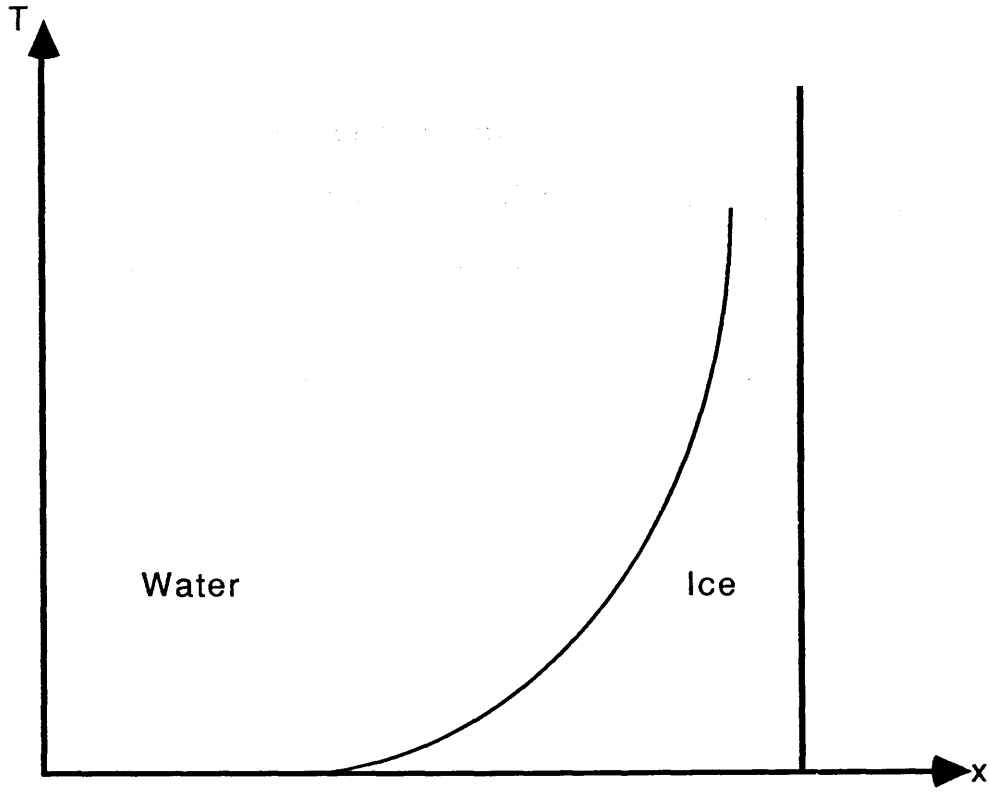
$$\partial u_i/\partial t=\partial^2 u_i/\partial x^2 \qquad i=1,2 \tag{1.7}$$

$i=1$ refers to water phase, $0<x<s(t)$ and $i=2$ refers to ice phase $s(t)<x<1$. It is assumed that the water and ice phases together always occupy the space $0\leq x\leq 1$.see Fig(1.2),

The MB conditions or Stefan conditions on $x=s(t)$ are :

$$u_i=0 \qquad i=1,2 \tag{1.8}$$

$$(\partial u/\partial x)_2-(\partial u/\partial x)_1=-ds/dt \tag{1.9}$$



Fig(1.2). Path of melting interface

1.3. Generalizations of the classical Stefan problem

These two phase problems posed in **1.2** can be generalized to multi-phase, multi-MBP's with different phase change temperature on each moving boundary. Other more general formulations of this classical problem are based on :

1.3.1. Non-linear governing equations

1.3.2. Time dependent or non-linear fixed boundary condition

1.3.3. Space and time dependent phase change temperatures.

(for more details see Crank 1984).

1.4. Other moving boundary problems

There are a number of other physical problems which are governed by a parabolic differential equation in conjunction with moving boundary conditions. Examples are water flooding and gravity drainage, the progress of temperature dependent chemical reaction through a solid, and the evaporation of droplets. All these problems have been considered under the name of Stefan's problem. (see Douglas and Gallie1955, Meyer 1976).

In one dimension, these problems involving time t as one independent variable lead usually to parabolic or hyperbolic equations. The simplest parabolic equation

$$u_t = k u_{xx} \tag{1.10}$$

derives from the theory of heat conduction. In cylindrical coordinates with axial symmetry this equation is

$$u_t = k(u_{xx} + x^{-1} u_x) \tag{1.11}$$

Numerical methods for solving problem with parabolic partial differential equations of any complexity generally involve a great deal of computation. It is usual therefore to arrange, whenever possible for one solution to represent wide variety of similar problems and this can be done by transforming all equations in terms of non-dimensional variables.

In this thesis we have solved one-dimensional Stefan problems, firstly the melting ice problem (cartesian and cylindrical coordinates) using variable time step methods which

have been suggested by some authors. Secondly, oxygen diffusion problems have been solved using both cartesian and cylindrical coordinate systems . The results computed from various methods are compared with each other and are found to be in very good agreement.

Chapter-2-

Finite differences

2.1. Introduction

Consider a partial differential equation (PDE) in which the independent variables are denoted by x, y, z . . . . . and the dependent variables by u, v, w, ...

Direct functionality is often written in the form

$$u=u(x, y, z) \tag{2.1}$$

which, designates u as a function of the independent variables x, y, and z .

The partial derivatives are denoted as follows :

$$u_x=\partial u/\partial x, \quad u_y=\partial u/\partial y, \quad u_{xx}=\partial^2 u/\partial x^2, \quad u_{xy}=\partial^2 u/\partial x\partial y \text{ etc.} \tag{2.2}$$

Using the definition of (2.1) and (2.2) we can represent a partial differential equation in the general form

$$F(x, y, u, u_x, u_y, u_{xx}, u_{yy}, u_{xy}, \ldots .)=0, \tag{2.3}$$

where F is a given function of the independent variables x, y, . . . and the "unknown" function u and of a finite number of its partial derivatives . We call u a solution of (2.3).

Examples :

$$u_{xx}+u_{yy}=0 \tag{2.4a}$$

$$u_x=u+x^2+y^2 \tag{2.4b}$$

$$u_{xxx}=u_{yy}+u^2 \tag{2.4c}$$

$$(u_x)^2+(u_y)^2=\exp(u) \tag{2.4d}$$

The order of a PDE is defined by the highest-order derivative in

the equation

$u_x - a u_y = 0$ is of first order (2.5a)

$u_{xx} + u_y = 0$ is of second order (2.5b)

$u_{xxx} + u_{yyy} = 0$ is of third order (2.5c)

In the solution of partial differential equation the proprety of linearity plays a particularly important role.

Consider the first order equation

$$a(\cdot)u_x + b(\cdot)u_y = c(\cdot) \tag{2.6}$$

The linearity of the above equation is established by the functionality of the coefficients $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$.

2.1.1. If the coefficient are constant or functions of the independent variable only $[(\cdot) \equiv (x, y)]$, the PDE is linear.

Example :

$$u_x + b u_y = 0 \tag{2.7}$$

2.1.2. If the coefficients are also functions of the dependent variable $[(\cdot) \equiv (x, y, u)]$, the PDE is quasi-linear.

Example :

$$u_x + u u_y = x^2 \tag{2.8}$$

2.1.3. If the coefficients are functions of the first derivative $[(\cdot) \equiv (x, y, u, u_x, u_y)]$, the PDE is non-linear, e.g.

$$u_x + (u_y)^2 = 0 \tag{2.9}$$

In general when the coefficients of an nth order PDE depend upon mth-order derivatives, $m<n$, the equation is quasi-linear.

The analytical solution of a PDE may be written

$$u=u(x,\ y) \qquad (2.10)$$

When one discusses the solution of a PDE, it is necessary to consider appropriate auxiliary initial and boundary conditions which arise from the physical problem itself, for example:
The transient temperature distribution in a homogenous rod of finite length with insulated sides is described by the following system.

$$u_x=u_{yy} \qquad x>0,\ 0<y<1 \qquad (P\ D\ E) \qquad (2.11a)$$

$$u(0,y)=f(y) \qquad x>0,\ 0<y<1 \quad \text{(initial condition)} \qquad (2.11b)$$

$$u(x,\ 0)=\phi(x) \qquad x\geq 0,\quad y=0 \qquad (2.11c)$$

$$u(x,\ 1)=\theta(x) \qquad x\geq 0,\quad y=1 \quad \text{(boundary condition)} \qquad (2.11d)$$

(see Lapidus, L. & Pinder, G.F 1982)

2.1.4. Classification of PDE

PDE may be classified in term of their transpose forms such as elliptic, hyperbolic and parabolic or in terms of type of problems to which they apply, namely

Equilibrum problems

Eigenvalue problems

Propagation problems.

2.2. Boundary Conditions

The solution of a PDE has to be made to satisfy the boundary conditions which arise from the problem formulation.

2.2.1. First boundary value problem (the Dirichlet problem) where the solution u has to satisfy the given values

$$(u)_s=\phi \qquad \text{on the boundary } s$$

if $\phi=0$ the problem is homogeneous Dirichlet.

2.2.2. The Neumann problem satisfies the normal derivatives

$(u_v)_s=\psi$ on the boundary of the region

2.2.3. Mixed problem satisfies a combination

$[u_v+hu]=\psi$ on s

2.2.4. Periodic boundary problem satisfies the periodicity conditions, for example

$(u)_x=(u)_{x+p}$, $(u_v)_x=(u_v)_{x+P}$ where p is called the period.

2.3. Parabolic equation

Many problems in physics and engineering requiring numerical solution involve special cases of the linear parabolic partial differential equation.

The problem of flow of heat along a rod whose temperature depends only on the co-ordinate x and on the time t leads to the equation

$$c\rho u_t=(ku_x)_x +f(x,t) \tag{2.12}$$

Linear parabolic equations are often written as

$$u_t=f(t, x, u_x, u_{xx}) \tag{2.13}$$

Parabolic PDEs can serve as test vehicles for the numerical algorithms to be presented. Our model second-order parabolic PDE takes the form

$$u_t=L(u) \tag{2.14}$$

where L(u) is a linear or non-linear operator involving partial derivatives and possibly other conditions. We now turn to the development of finite difference methods for parabolic equations.

To explain the basic ideas and illustrate the process of developing improved difference equations for a given problem, the simple dimensionless diffusion equation

$$u_t = u_{xx} \tag{2.15}$$

will be used as the initial model.

2.4. Methods of Numerical Solution

There are, in general two different main techniques for solving PDEs, namely finite difference and finite element method. Finite difference methods have been used extensively for the numerical solution of moving boundary problems in partial differential equations applicable to linear and non-linear problems. In recent years the finite element technique has been introduced.

The application of these techniques to time independent (elliptic) and time dependent or propagation problems (parabolic and hyperbolic) differ greatly. Here only time dependent problems will be considered; the problems to be studied satisfy parabolic equations.

2.5. Finite Difference Grids

In the finite difference method the whole region is covered by a grid, the values of the unknowns being found at the nodes of the grid. These grids may be applied in different ways.

2.5.1. *Fixed finite difference grid*

In this method the whole region for example ($0 \leq x \leq 1$) must be subdivided into a finite number of equal intervals and a suitable time step is chosen before using a numerical method. Furthermore the grid sizes of the space and the time are kept constant during the process (see Gupta 1974).

At any time the interface moves through the medium and at any time it will usually be located between two neighbouring grid points. This can be allowed for by using variable finite difference formulae which incorporate unequal space intervals near the moving boundary. Interpolation formulae of Lagrangian type can be used (see Crank 1957a).

2.5.2. *Modified grids*

There are various ways of varying the grid with the aim of avoiding the increased complication and loss of accuracy associated with unequal space intervals near the moving boundary (see Gupta 1974).

2.5.2.a. Variable time step

Douglas and Gallie (1955) subdivided the whole region into a fixed space grid for all times but chose each time step such that the moving boundary coincides with a grid line in space at each time level. They treated the following problem

$$u_t = u_{xx} \qquad 0<x<s(t) \quad t>0 \qquad (2.16a)$$

$$u_x = -1 \qquad x=0 \quad t>0 \qquad (2.16b)$$

$$dx/dt = -u_x \qquad u=0 \quad x=s(t) \qquad (2.16c)$$

$$s(0)=0 \qquad (2.16d)$$

This applied at the fixed surface x=0.

By integrating (2.16a) with respect boundary condition (2.16a) t using (2.16b) they deduced the alternative form of boundary condition (2.16c),

$$s(t)= t - \int_0^{s(t)} u(x,t)dt \; . \qquad (2.17)$$

This equation was discretized as

$$\Delta t_j^{(r+1)} = \left(j+1 + \sum u_{i,j+1}^{(r)} \right).\Delta x - t_j \qquad (2.18)$$

where $u_{i,j}$ denotes $u(x_i,t_i)$, and r is an iteration index .

The solution proceeds iteratively by using (2.18) to correct the assumed time step.

To avoid the iteration, they suggested using $u_{i,j}$ in place of $u_{i,j+1}^{(r)}$ in (2.18).

2.5.2.b. Variable space grid

This method, due to Murray and Landis, uses a variable space mesh, choosing a finite time.

They keep the number of space intervals, between x=0 and x=s(t) i.e between a fixed and a moving boundary, constant and equal to n say, for all time see Fig.(2.1) . The space step $\Delta x = s(t)/n$ is different at each time step.

The moving boundary is always on the nth grid line.

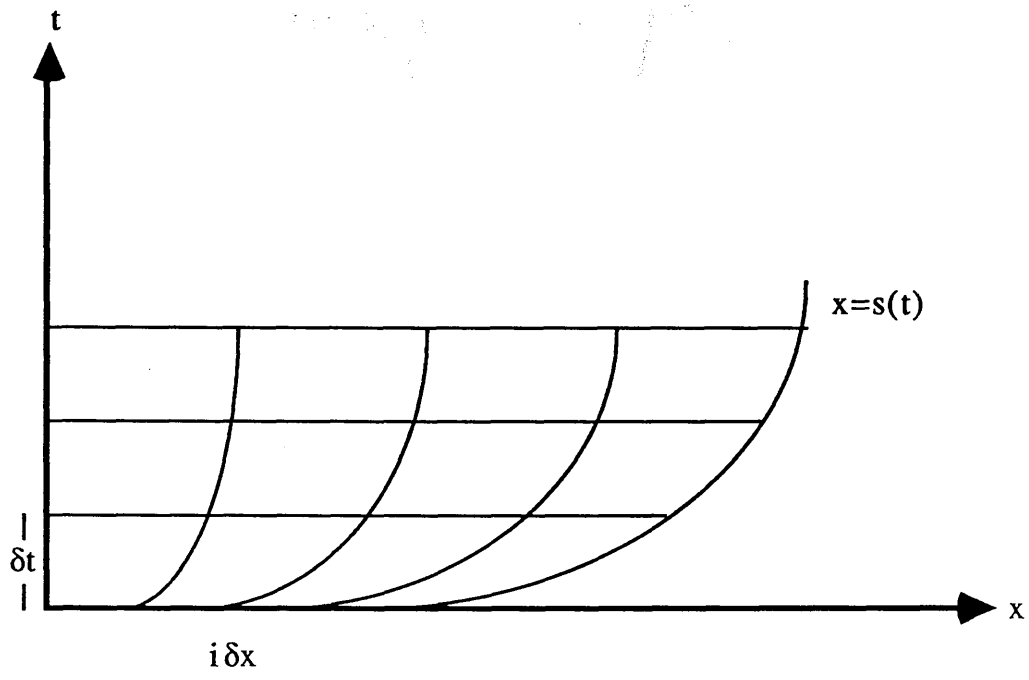


Fig.(2.1) Murray and Landis variable grid

2.6. Irregular Boundaries

Many physical problems have irregular boundaries, that is the boundary does not coincide with a mesh line, so the distance from at least one line of points adjacent to the boundary will be smaller than the standard interval length in that direction. In this case we can still find similar formulae for points near the boundary, but the truncation error is likely to be of lower order. To illustrate the construction of irregular molecules we consider the situation in Fig.(2.2) and obtain approximations for u_x, u_{xx}

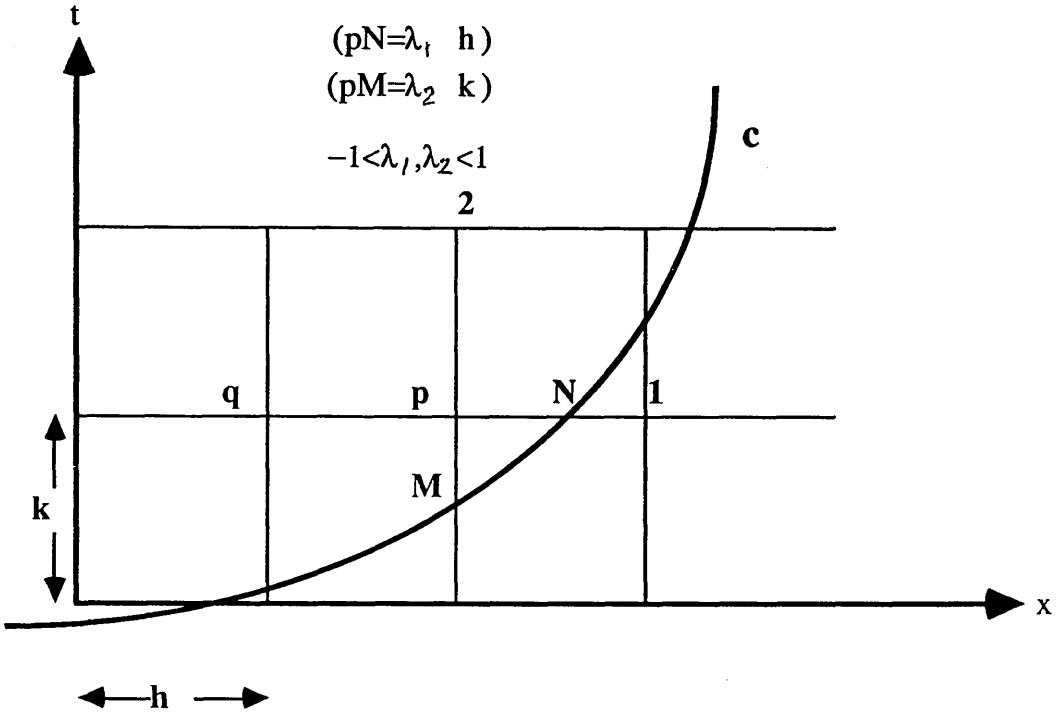


Fig.(2.2) geometry of an irregular point p near the moving boundary

The values of u are assumed to be known, N and M are boundary points and the distances pN and pM are fractions of the standard interval lengths h and k respectively.

If p is assumed to be the origin then by expanding u in terms of p we get in Taylor series

$$(u)_N=(u)_p+h\lambda_1(u_x)_p+0.5(h\lambda_1)^2(u_{xx})_p+o(h^3) \qquad (2.19)$$

$$(u)_q=(u)_p-h(u_x)_p+0.5h^2(u_{xx})_p+o(h^3) \qquad (2.20)$$

Elimination of $(u_{xx})_p$ gives

$$(u_x)_p=(1/h)\{[1/\lambda_1(1+\lambda_1)](u)_N-[(1-\lambda_1)/\lambda_1](u)_p-[\lambda_1/(1+\lambda_1)](u)_q\}$$
$$+o(h^2) \qquad (2.21)$$

Similarly the elimination of $(u_x)_p$ leads to

$$(u_{xx})_p=(1/h^2)\{[2/\lambda_1(1+\lambda_1)](u)_N+[2/(1+\lambda_1)](u)_q-(2/\lambda_1)(u)_p\}+o(h) \qquad (2.22)$$

Note that all finite difference formula are based upon polynomial approximation, that is they give exact results when operating upon a polynomial of suitable degree. In other cases the formulae are approximations. Since only a finite number of terms can of necessity be used, the truncation error is of concern.

2.7. Explicit and implicit numerical methods

Fully explicit finite difference methods have been used repeatedly with good success for a number of melting problems. However, the explicit solution of diffusion equations imposes severe stability restrictions on the space-time resolution of the problem (semi implicit technique). Explicit methods will break down when s(t) moves rapidly (see Attey 1974). Implicit methods require the solution of simultaneous linear equations.

2.8. Explicit formula

Explicit methods for parabolic PDE's use explicit formulae which compute directly the solution one point at time at grid points on an advanced time level $t = (j+1)\delta t$ from the solution on previous time levels. Explicit difference approximations are now derived for various forms of the difference equation with special cases of the equation.

Case constant coefficients

$$u_t = u_{xx} \qquad (2.23)$$

The resulting difference equation is

$$u_{i,j+1} = r u_{i-1,j} + (1-2r)u_{i,j} + r u_{i+1,j} \qquad i=1, \ldots \qquad (2.24)$$

where $r=(\delta t/h^2)=\Delta t/(\Delta x)^2$, $x= i\Delta x$, and $t = j\Delta t$.

The computational molecule for (2.24) is illustrated in Fig.(2.3)

The explicit relation (2.24) is often called the forward difference equation (see Ames 1977).

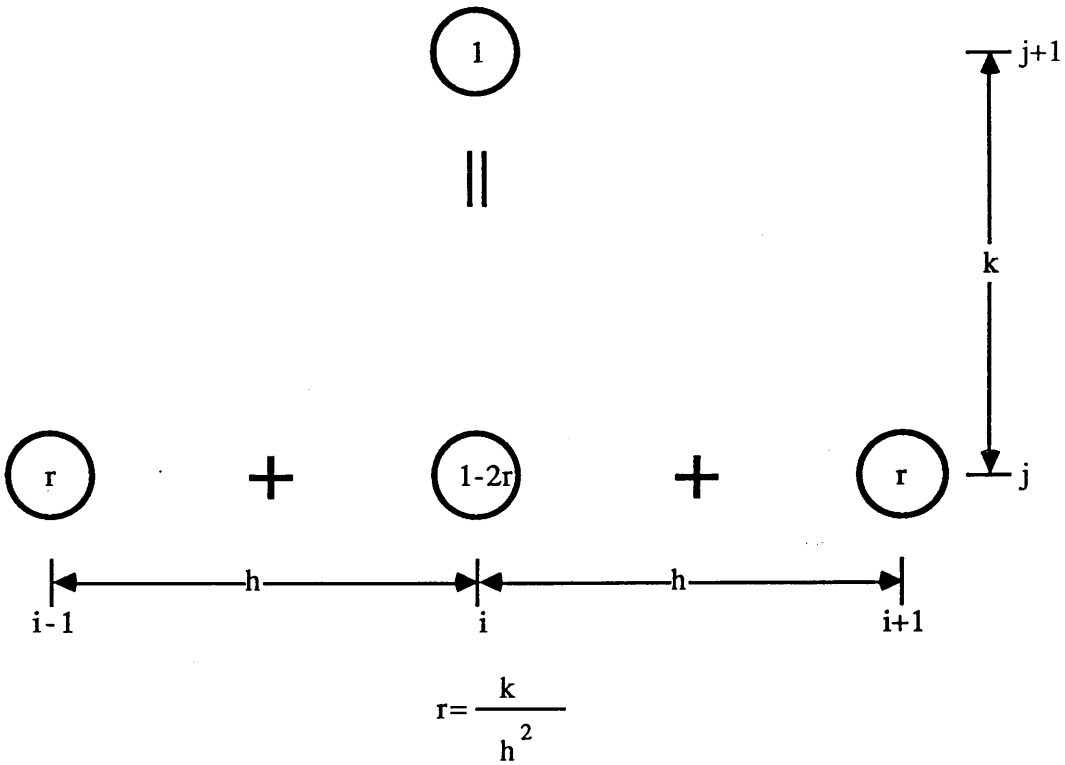


Fig.(2.3). Computational molecule for explicit difference approximation of $\partial u/\partial t=\partial(\partial u/\partial x)\partial x$

2.9. The Dufort-Frankel Explicit approximation

The simplest three-level scheme for the solution of the

diffusion equation

$u_t=u_{xx}$ at point (ih, jk) is (2.25)

$$(u_{i,j+1} - u_{i,j-1})/2k=(1/h^2)(u_{i-1,,j} + -2u_{i,j} + u_{i+1,j}) \quad (2.26)$$

Dufort-Frankel replace $2u_{i,j}$ in the equation by $u_{i,j-1} + u_{i,j+1}$ thereby generating the three time level formula

$$(1+2r)u_{i,j+1}=2r(u_{i-1,j} + u_{i+1,j}) + (1-2r)u_{i,j-1} \quad (2.27)$$

where $r=k/h^2$,

The computational molecule for [Eqn (2.27)] is illustrated in Fig.(2.4). (see Gottlieb, D. & Gustafsson, B. 1976).

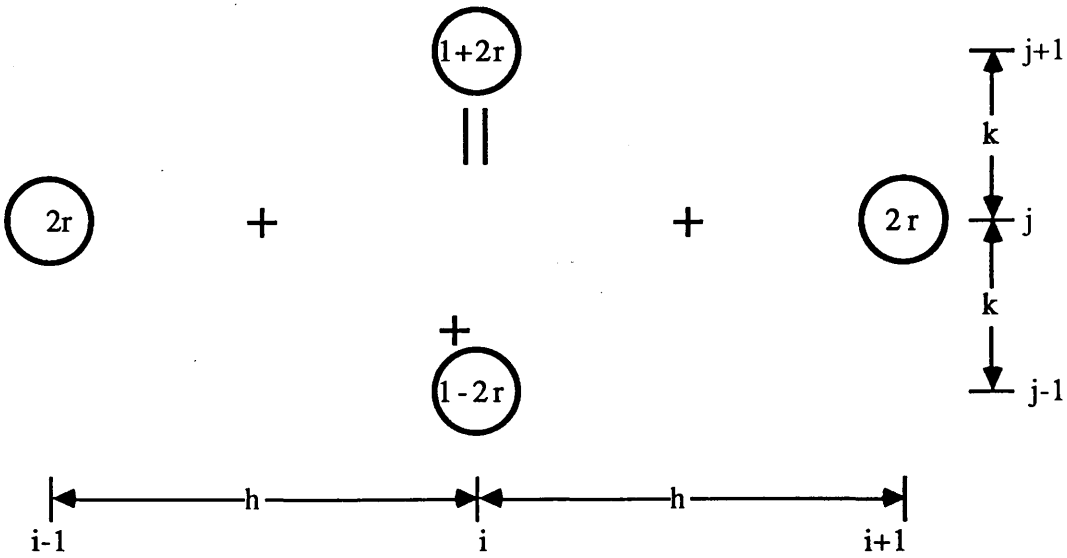


Fig.(2.4). Computational molecule for Dufort-Frankel explicit difference approximation of $\partial u/\partial t=\partial(\partial u/\partial x)/\partial x$

2.10. Implicit formula

An implicit formula is one with two or more unknown values in the j+1 row.

The simplest implicit method is that suggested first by O'Brien et al upon approximation of the derivative u_{xx} of of the following equation

$$u_t=u_{xx}, \quad 0<x<1, \quad 0<t<T \tag{2.28a}$$

$$u(x,0)=f(x,x), \quad 0<x<1 \tag{2.28b}$$

$$u(0,t)=g(t), \quad 0<t\leq T \tag{2.28c}$$

$$u(1,t)=h(t), \quad 0<t\leq T \tag{2.28d}$$

In the j+1 row instead of the j row we obtain

$$-ru_{i-1,j+1} + (1+2r)u_{i,j+1} -ru_{i+1,j+1}=u_{i,j} . \tag{2.29}$$

Crank and Nicolson used an average of approximations in the j and j+1 row. More generally one can introduce a weighting factor λ and replace (2.29) by

$$u_{i,j+1}-u_{i,j}=r\{\lambda[u_{i-1,j+1}-2u_{i,j+1}+u_{i+1,j+1}]+(1-\lambda)[u_{i-1,j}-2u_{i,j}+u_{i+1,j}]\} \tag{2.30}$$

with $0\leq \lambda \leq 1$ then

$$-r\lambda u_{i-1,j+1}+(1+2r\lambda)u_{i,j+1}-r\lambda u_{i+1,j+1}=$$

$$r(1-\lambda)u_{i-1,j}+[1-2r(1-\lambda)]u_{i,j}+r(1-\lambda)u_{i+1,j} . \tag{2.31}$$

(see the computational molecule form in Fig (2.5))

Special cases are

if $\lambda=1$ eqn(2.31) becomes the O'Brien et al form, and

if $\lambda=0.5$ eqn(2.31) becomes the Crank-Nicolson formula.

On the other hand. If $\lambda=0$ the explicit relation eqn(2.24) is recovered.

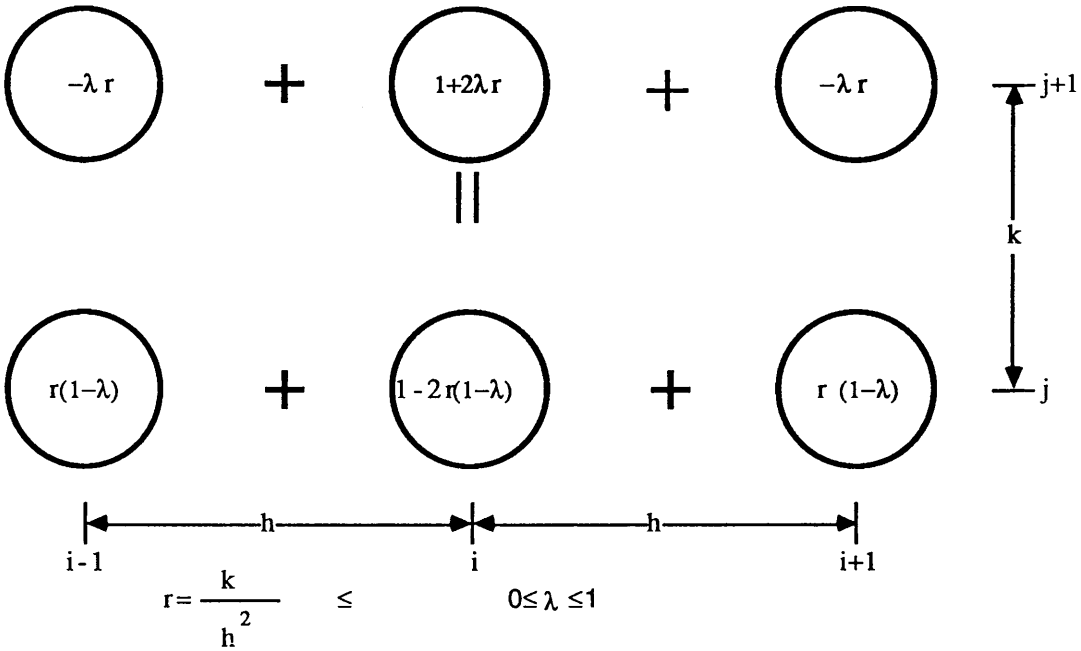


Fig.(2.5). Computational molecule for implicit finite difference approximation for $\partial u/\partial t=\partial(\partial u/\partial x)/\partial x$

2.11. Douglas Formula

The formula of highest possible accuracy based on the same six grid points as the Crank-Nicolson formula.

The resulting difference equation is

$$u_{i,j+1}-0.5(r-1/6)[u_{i-1,j+1}-2u_{i,j+1}+u_{i+1,j+1}]=$$
$$u_{i,j}+0.5(r+1/6)[u_{i-1,j}-2u_{i,j}+u_{i+1,j}] \qquad (2.32)$$

with a local truncation error that is $O(k^2+h^4)$

2.12.Convergence, stability and consistency

2.12.1. Convergence is concerned with the conditions that must be satisfied in order for the discretization error to tend to zero at a fixed point or fixed time level t=jk as k (and h) tends to zero i.e as the number of time-levels of calculations tends to infinity.

2.12.2. Stability: Two standard ways of investigating the boundness of the solution of the finite difference equations exist. In the first we use a finite Fourier series, whereas in the second we express the difference equations in matrix form and examine the eigenvalues of an associated matrix. (See the diffusion oxygen problem in chapter-5- and the matrix stability in appendix B.) The Fourier series method is the easier of the two in that it requires no knowledge of matrix algebra but it is the less rigorous because it neglects the boundary conditions. There is also the energy method (see Mitchell 1980 p44).

2.12.3. Consistency ensures that the difference equation converges to the solution of correct differential equation as the grid spacing (h,k) tends to zero. A difference approximation to a parabolic equation is constant if the truncation error tends to zero as h,k tend to zero.

2.13. Non-Linear Equations

We have considered only parabolic equations but many mathematical formulations of natural processes are non-linear or quasi-linear.

Many numerical methods and techniques of proof for linear equations with constant coefficients carry over to non-linear equations. Questions of stability and convergence are more complicated (see Ames[1977]). Richtmyer[1967] considered the

non-linear problem $u_t=(u^5)_{xx}$ and Douglas considered the quasi-linear parabolic equation.

$$u_{xx}=F(x, t, u)u_t+G(x, t, u) \qquad F\geq a>0 \qquad (2.33)$$

For stability and more details see Ames [1977], Smith [1978], and Mitchell [1980].

Chapter-3-

Modified variable time step method for solving the ice melting problem

3.1. One phase moving boundary problem

There are a number of problems in heat conduction in which one material is transformed into another with generation or absorbtion of heat. Examples are the melting or freezing of a solid and the progress of a temperature-dependent chemical reaction through a solid.

$u_t=u_{xx}$ $0<x<s(t)$ $0<t<T$ (3.1)

$u(x,0)=h(x)$ $0<x<b$ $(b=s(0),\ h(b)=0)$ (3.2)

$h(x)>0$ if $x\neq b$

$u(0,t)=f(t)$ $0<t<T$ $(f>0)$ (3.3)

$u(s(t),t)=0$ $0<t<T$ (3.4)

$u_x(s(t),t)=s'(t)$ $0<t<T$ (3.5)

The curve $x=s(t)$ is called the moving boundary, or the free boundary and there are several methods to construct a solution:

3.1.1. Reduce the problem to an integral equation for $u_x(s(t),t)$

3.1.2. Finite differences

3.1.3. Other approximation techniques

3.1.4. Weak solution

There are numerous other problems which gives rise to free

boundary problems, in which some conditions (3.3)-(3.5) are different.

3.2. Problem

One such problem is the melting of a semi-infinite slab of ice which initially is at uniform temperature and subsequently is subjected to a constant heat flux at the surface.

Mathematically, the problem is defined as follows:

$$u_t = u_{xx} \qquad (0<x<s(t),\ t>0) \qquad (3.6)$$

$$u_x = -1 \qquad (x=0,\ t>0) \qquad (3.7)$$

$$u = 0 \qquad (x \geq s(t),\ t \geq 0) \qquad (3.8)$$

$$dx/dt = -u_x \qquad (x=s(t),\ t>0) \qquad (3.9)$$

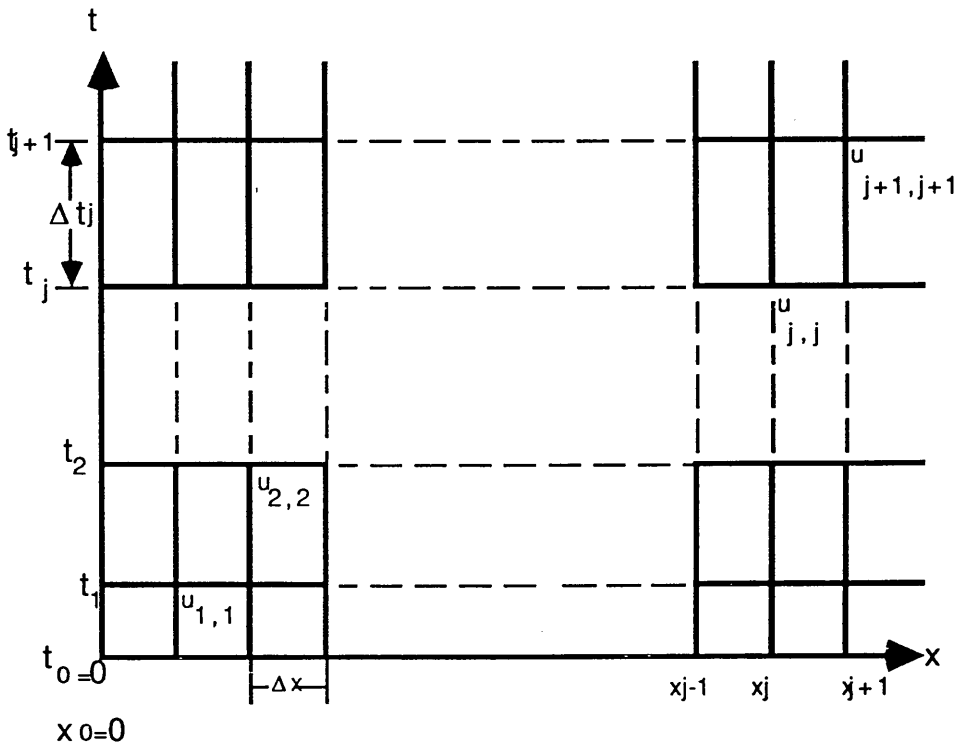
$$s(0) = 0 \qquad (3.10)$$

where u(x,t) is the temperature at a distance x from the fixed surface at time t, and s(t) is the position of the moving boundary. Condition (3.9) arises from the fact that part of the heat supplied at the moving boundary is used in changing the phase, and the condition (3.10) means the process of melting has not started yet.

3.3. Methods of solution

In a variable time method (see 2.5.2a) let us suppose (see Fig(3.1)) that during time t_j the boundary has moved a distance $j\Delta x$ in successive time steps of Δt_r r=0,1,2, ... j-1, Δt_r being the time taken by the boundary to move a distance Δx from its position $x_r = r\Delta x$ to $x_{r+1} = (r+1)\Delta x$.

We assume that the values $u(x_i,t_j)$ are known. We wish to find Δt_j where $t_{j+1}=t_j+\Delta t_j$ such that the boundary moves a distance Δx during that time. Finally $u(x_i,t_{j+1})$ may be computed by standard finite difference methods.



Fig(3.1). Grid system showing position of moving boundary at different times.

3.4. Method of Douglas and Gallie

The differential equation (3.6) is discretiszed at (x_i,t_{j+1}) by O'Brien et al. formulae (See (2.28) and (2.29)) giving

$$-r^{(k)}u^{(k)}_{i-1,j+1}+(1+2r^{(k)})u^{(k)}_{i,j+1}-r^{(k)}u^{(k)}_{i+1,j+1}=u^{(k)}_{i,j} \qquad (3.11)$$

$$i=1,2,3 \ldots\ldots\ldots j$$

where $r^{(k)}=\Delta t^{(k)}_j/(\Delta x)^2$ and the superscript k denotes the kth iteration. The boundary condition (3.7) is replaced by a forward difference, giving

$$u^{(k)}_{1,j+1}-u^{(k)}_{0,j+1}=-\Delta x \qquad (3.12)$$

Having chosen $\Delta t^{(k)}_j$, and hence r, the system of equations given by (3.11) and (3.12) is solved, using the Thomas algorithm (see appendix A, and for more details Atkinson(1983)),

for $u^{(k)}_{i,j+1}$ $i=0,1,2, \ldots\ldots j$ remembering that $u_{j+1,j+1}=0$ from the boundary condition (3.8).

After the values of u have been computed at the kth iteration, the assumed time step $\Delta t^{(k)}_j$ is modified by the formula.

$$\Delta t^{(k)}_j=\{(j+1)+\sum_{i}^{j} u^{(k)}_{i,j+1}\}\Delta x-t_j \qquad (3.13)$$

in order to obtain (3.13). Equation[3.6] is integrated with respect to x from 0 to s(t) and with respect to t from 0 to t giving

$$t=s(t)+\int_0^{s(t)} u\,dx \qquad (3.14)$$

The moving boundary condition (3.9) has been used in deriving (3.14) and relation (3.13) is simply a finite difference replacement of [3.14] (see (2.17) and (2.18) in chapter-2). Using the improved value of Δt_j from (3.13) the system of equations given by (3.11) and (3.12) is again solved by the Thomas algorithm using new value of r. The process is repeated until Δt_j is obtained to within desired accuracy.

3.5. Method of Gooding and Khader

These authors obtained equations [3.11] and [3.12] for a known value

$^{(k)}$
of Δt_j and the equations given by [3.11] (j in number) contain j+1

$^{(k)}$
unknowns $u_{i,j+1}$, i=0,1,2,3, ... j. The authors used (3.9) instead of using (3.12) which gives

$$-(u_{j+1,j+1}-u_{j,j+1})/\Delta x=\Delta x/\Delta t_j \qquad (3.15)$$

with the boundary condition (3.8) becoming

$$u_{j,j+1}=(\Delta x)^2/\Delta t_j, \qquad (3.16)$$

and in terms of the kth iterate

$$\Delta t_j=(\Delta x)^2/\ u_{j,j+1}^{(k)} \qquad (3.17)$$

Choosing an arbitrary value $u_{j,j+1}^{(k)}$, the value of $\Delta t_j^{(k)}$ is fixed from

(3.17) which subsequently fixes the value of $r^{(k)}$.

The values $u^{(k)}_{i,j+1}$ i=0,1,2, ...j-1 are determined from (3.11) by the process of back-substitution. Equation (3.12) is used to check the

error in the choice of $u^{(k)}_{j,j+1}$, but if (3.12) is not satisfied (with some reasonable limit), $u_{j,j+1}$ is adjusted accordingly and a new value of Δt_j is obtained from (3.17).

The process is repeated until the desired accuracy in (3.12) is obtained.

3.6. Method of Gupta and Kumar

In this method the replacement of (3.6) is made as in methods **3.4** and **3.5** leading to (3.11). The tridiagonal set of equations given by (3.11) and (3.12) is then solved by the method of Douglas and Gallie, and Δt_j is improved using the finite difference replacements of (3.9) given by (3.17) and not by (3.13).

When an improved value of Δt_j was obtained, the system (3.11),(3.12) is solved again to obtain $u_{i,j+1}$, i=0,1,2, . . . ,j which is used to compute a new value of Δt_j from (3.17).

This process is repeated until the desired accuracy in Δt_j is achieved.

3.7. Boundary conditions for melting ice

On the unknown boundary x=s(t),

$$u=0 \tag{3.18a}$$

$$dx/dt=-u_x \tag{3.18b}$$

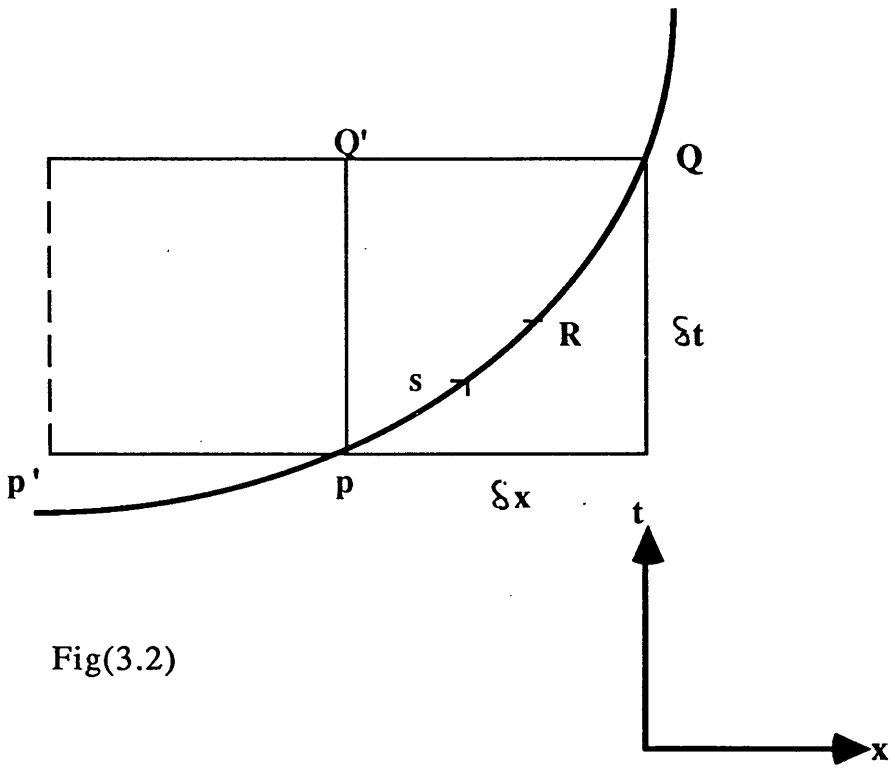
Also $\Delta u=u_x\Delta x + u_t\Delta t=0$ (3.18c)

i.e $u_x(dx/dt) + u_t=0$ (3.18d)

and so $u_t=(u_x)^2$ (3.18e)

3.7.1. Variable grid method

(see Fig(3.2) where δx is fixed and δt is chosen so that Q is a boundary point).



Fig(3.2)

At a boundary point, for x and t on the boundary

$$t=t_0 + (x-x_0)(dt/dx)_0 + (1/2!)(x-x_0)^2(d^2/dx^2)_0 + \ldots\ldots \quad (3.19)$$

whence $\delta t=\delta x(dt/dx)_0 + (1/2!)(\delta x)^2(d^2t/dx^2)_0 + \ldots\ldots\ldots (3.20)$

and so the ratio

$k=\delta t/(\delta x)^2=1/\delta x(dt/dx)_0 + (1/2!)(d^2t/dx^2)_0 + (1/3!)\delta x(d^3t/dx^3)_0+\ldots$

$=-1/\delta x(u_x)_0 + (1/2)(d^2t/dx^2)_0 +(\delta x/3!)(d^3t/dx^3)_0 + \ldots\ldots\ldots (3.21)$

Now $\delta x(u_x)_Q=u_Q-u_{Q'} +((\delta x)^2/2!)(u_{xx})_Q + \ldots$ and $u_Q=0$ (3.22)

so $k=1/u_{Q'} + (1/2)(d^2t/dx^2)_Q + O(\delta x)$ (3.23)

where the term (d^2t/dx^2) has been neglected

Alternatively, there exists an intermediate point R, between p and Q, for which $(dt/dx)=(\delta t/\delta x)$.

Let this be at $s_R=s_p+\theta h \quad 0<\theta<1$ and denote u_x by u'. Then along the boundary, we have

$u'= u'_p +(s-s_p)(du'/ds)_p+(1/2!)(s-s_p)^2(d^2u'/ds^2) + \ldots\ldots$ (3.24)

and so $u'_Q=u'_p+h(du'/ds)_p+(h^2/2)(d^2u'/ds^2)_p + \ldots\{h=s_Q-s_p\}$ (3.25)

and $u'_R=u'(s_p+\theta h)=u'_p +\theta h(du'/ds)_p + (\theta^2h^2/2)(d^2u'/ds^2)_p + \ldots$ (3.26)

Thus $u'_R - (1/2)u'_Q = (1/2)u'_p + (\theta-1/2)h(du'/ds)_p+$

$((\theta^2-1/2)h^2/2)/(d^2u/ds^2)_p + \ldots\ldots$ (3.27)

i.e $u'_R=(u'_p+u'_Q)/2 + (\theta-1/2)h(du'/ds)_p +$

$((\theta^2-1/2)h^2/2!)(d^2u'/ds^2)_p + \ldots\ldots\ldots$ (3.28)

Now $h=\int_0^h ds = \int_0^h \sqrt{((dx)^2+(dt)^2)}$

$$= \int_0^{\delta x} \sqrt{\left(1+(dt/dx)^2\right)}\, dx$$

Thus $h=k\delta x + O(\delta x)^2$ where $k=\sqrt{\left(1+(dt/dx)^2_p\right)}$ (3.29)

and so $u'_R=(u'_p+u'_Q)/2 + O(\delta x)$, (3.30)

which gives

$$(u_x)_R= \{(u_x)_p + (u_x)_Q\}/2 + O(\delta x) \quad (3.31)$$

Further $\delta x(u_x)_Q = -u_{Q'} + ((\delta x)^2/2!)(u_{xx})_Q$ (3.32)

and similarly for $\delta x(u_x)_p$,

so that $dt/dx=-1/u_x$ becomes, at R

$$(\delta t/\delta x)=-1/(u_x)_R \quad (3.33)$$

or $k= \delta t/(\delta x)^2 = -1/\delta x(u_x)_R$

$$=-1/[(1/2)\{(u_x)_p + (u_x)_Q\}+O(\delta x^2)]$$

$$=1/2.[(u_{p'}+u_{Q'})]+O(\delta x^2)$$

$$=2/(u_{p'}+u_{Q'}) + O(\delta x^2) \quad (3.34)$$

3.7.2. Approximation to u_x

At a point on the moving boundary

$u=0$

$dx/dt=-u_x$

These conditions give

$$u_t=(u_x)^2 \quad (3.35)$$

and from the equation $(u_x)^2=u_{xx}$, (3.36)

assuming expansion in powers of x about a boundary point x_0,

$h=x-x_0$

$$u(x)=u_0 + h(u_x)_0 +(h^2/2!)(u_{xx})_0 +(h^3/3!)(u_{xxx})_0 + \ldots\ldots, \quad (3.37)$$

and applying conditions

$$u(x)= h(u_x)_0 + (h^2/2!)(u_x)^2_0 +(h^3/3!)(u_{xxx})_0+ \ldots \quad (3.38)$$

then for the point immediately inside boundary u_{-1}

$h=-\delta x$

and so $u_{-1}= -\delta x(u_x)_0 + \{\delta x(u_x)_0\}^2/2 + \ldots .$ (3.39)

Solving for $\delta x(u_x)_0$ we get

$$\delta x(u_x)_0= 1 \pm \sqrt{(1+2u_{-1})} \quad (3.40)$$

$$=1- \sqrt{(1+2u_{-1})} \qquad \text{since } u_x<0 \quad (3.41)$$

$$=1-(1+2u_{-1})^{1/2}$$

$$=1-\{1+(1/2).2u_{-1}+[(1/2).(1/2)/2!].4u^2_1+ \ldots .\}$$

$$=-u_{-1} +u^2_{-1}/2 -u^3_{-1}/2 + \ldots .$$

From the above we can obtain the possible boundary condition methods, i.e approximations to $dx/dt=-u_x$ giving formula for $k=\delta t/(\delta x)^2$

For the calculation at row $j+1$ see Fig(3.3),

we get

$k=1/u[j,j+1]$ or $1/k=u[j,j+1]$ (3.42)

Applying $(u_x)^2=u_{xx}$ on the boundary

$1/k=-1+(1 + 2u)^{1/2}$ $u\equiv u[j,j+1]$ (3.43)

$=u-u^2/2 + u^3/2$

and taking the tangent,

$1/k=\{u[j-1,j]+u[j,j+1]\}/2.$ (3.44)

Taking the tangent with condition $(u_x)^2=u_{xx}$,

$1/k=(1/k_1 + 1/k_2)$ (3.45)

where $1/k_1 =u-u^2/2 + \ldots .$ with $u=u[j-1,j]$

and $1/k_2=u-u^2/2+\ldots\ldots$ with $u=u[j,j+1]$

Finally we obtain four different formulae to calculate the time,

$$t=(\delta x)^2/u_{j,j+1}, \tag{3.46}$$

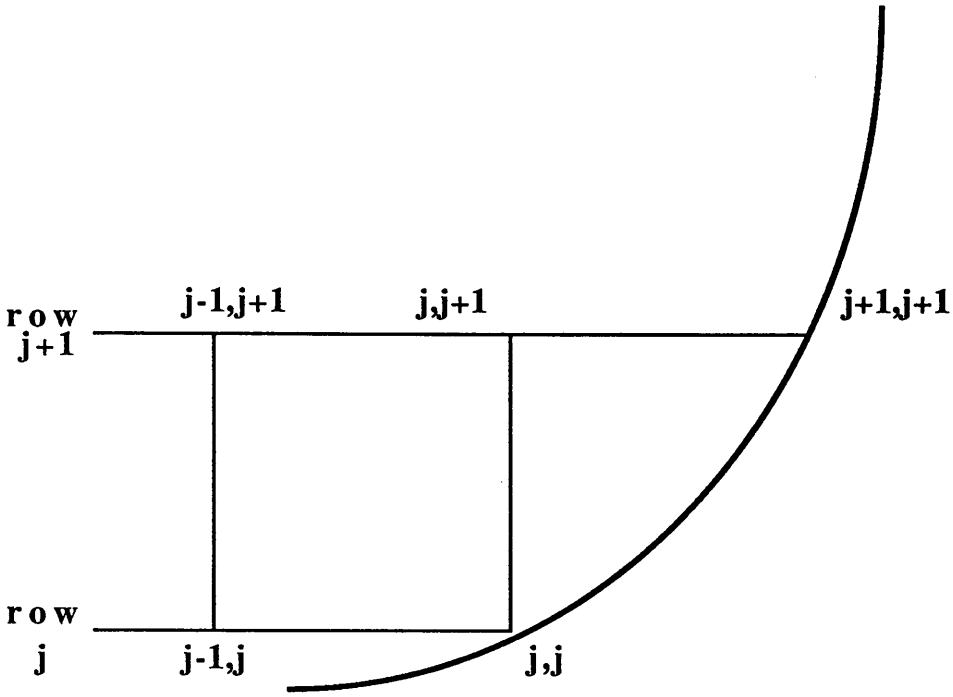
(the formula (3.46) was used by Gooding and Kahder (see(3.17)) and by Gupta and Kumar (see section **3.6**).

$$t=2(\delta x)^2/(u_{j-1,j}+u_{j,j+1}), \tag{3.47}$$

$$t=(\delta x)^2/[-1+\sqrt{(1+2u_{j,j+1})}], \tag{3.48}$$

and $$t=2(\delta x)^2/[-2+\sqrt{(1+2u_{j-1,j})}+\sqrt{(1+2u_{j,j+1})}] \tag{3.49}$$

We use the above formulae to calculate the time in some methods mentioned in this chapter. See tables in chapter-6.



Fig(3.3)

3.8. Solution of ice melting using the Douglas implicit formula in the replacement of equation (3.6)

The Douglas formula is shown previously (see (2.32) section **2.11** in chapter 2). Using notation shown in Fig(3.4), the extrapolated value at (j+1,j) is required

On the boundary,

$$u = 0 \qquad (3.50a)$$

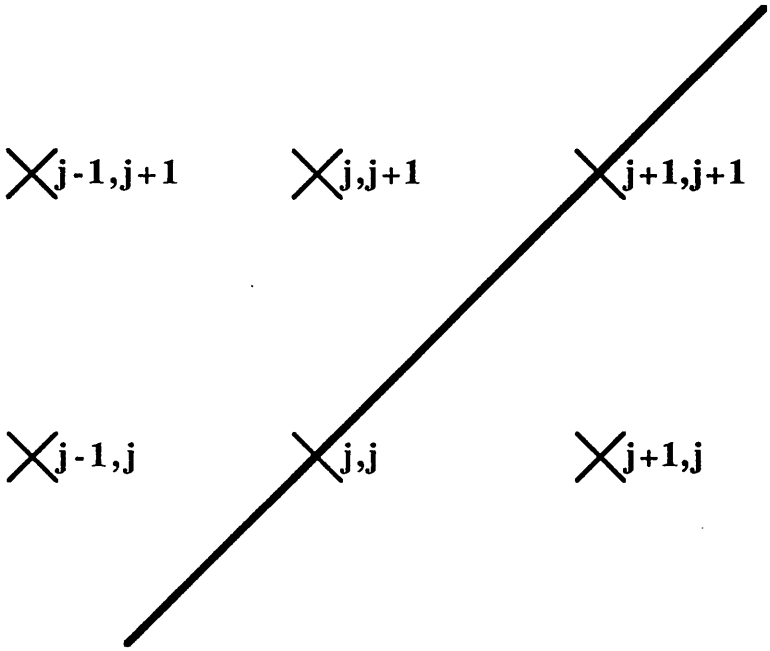
and $$dx/dt = -u_x \qquad (3.50b)$$

the latter gives

$$u_t = (u_x)^2 \qquad (3.50c)$$

and so we have

$$(u_x)^2=u_{xx} \qquad (3.50d)$$



Fig(3.4)

The finite difference approximation is { taking points as -1, 0, 1}

$$[(u_1-u_{-1})^2/2\delta x]^2=(u_{-1}-2u_0+u_1)/(\delta x)^2 \quad \text{and } u_0=0$$

so $(u_1-u_{-1})^2=4(u_1+u_{-1})$

This gives the quadratic equation for u_1

$$u^2_1-(2u_{-1}+4)u_1+u^2_{-1}-4u_{-1}=0$$

with solution

$$u_1=u_{-1}+2\pm2\sqrt{(2u_{-1}+1)}$$

Now plotting $(u_1-u_{-1})^2=4(u_1+u_{-1})$ see Fig(3.5), we see that for small u_1, u_{-1} , $|u_1,u_{-1}|< 4$ when $u_1>0$ $u_{-1}<0$ and vice versa, choosing the negative sign we have

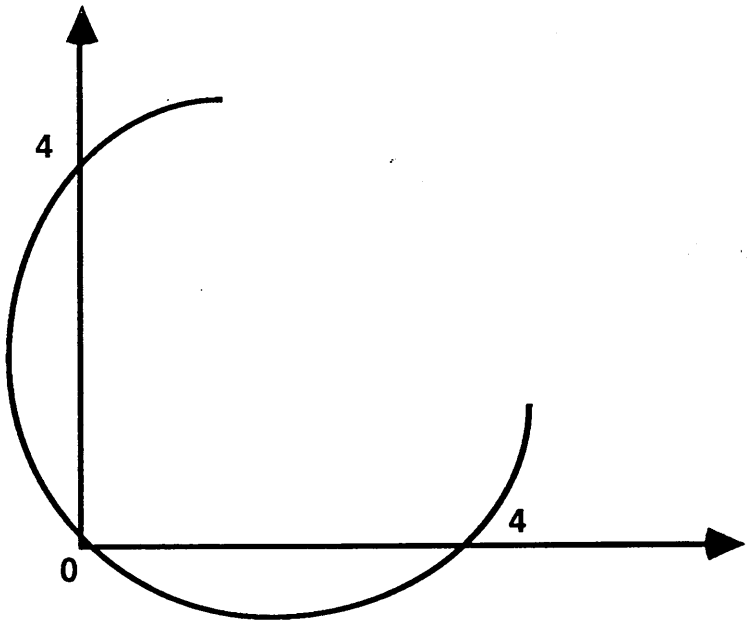
$$u_1=u_{-1}+2-2\sqrt{(1+2u_{-1})}$$

This is the approximation required at (j+1,j)

i.e $u[j+1,j] =u[j-1,j]+2-2\sqrt{(1+2u[j-1,j])}$ (3.51)

$=-u + u^2 +u^3 + \ldots\ldots$

where $u=u[j-1,j]$



Fig(3.5)

3.9. Numerical Solutions of the Cartesian Problem

We have performed computations using the methods described in (**3.4**), (**3.6**) and (**3.8**). In the case of the methods of (**3.6**) and (**3.8**) we have calculated different solutions using each of the forms of the boundary condition on the moving boundary derived in (**3.7**).

Each computation has been made until the boundary x=1 is reached. In methods (**3.4**) and (**3.6**) an error of 0.5% has been allowed in the calculation of Δt i.e we iterate until two successive values of Δt differ by not more than 0.5%. In method (**3.5**) until the left hand side of (3.12) agrees within 0.5% with its right hand side has been taken equal to Δx.

Comparative results for methods (**3.4**) and (**3.6**) are given in tables (6.1)--(6.5). Fig(3.11) shows the position of the moving boundary as calculated by the method (**3.6**) and $\delta x=\Delta x=h=0.01$. It is clear that the velocity of the moving boundary s'(t) gets slower as time proceeds. Fig(3.12) shows the temperature at the fixed surface x=0 increases with time as expected and $\delta x=\Delta x=h=0.01$.

Various results are given in table (6.6)--(6.11) using the method of Gupta and Kumar (section **3.6**) and the new finite difference replacement of equation (3.6) given in section **3.8** i.e using method **3.8.**

3.10. The cylindrical problem

The non-dimensional form of the heat conduction equation in three dimensions is $u_t = \nabla^2 u$ which, in cylindrical polar coordinates (x,θ,z) is

$$u_t = u_{xx} + (1/x)u_x + (1/x^2)u_{\theta\theta} + u_{zz} \qquad (3.52)$$

For simplicity, assuming that u is independent of z, this reduces to the two-dimensional equation.

$$u_t = u_{xx} + (1/x)u_x + (1/x^2)u_{\theta\theta} \qquad (3.53)$$

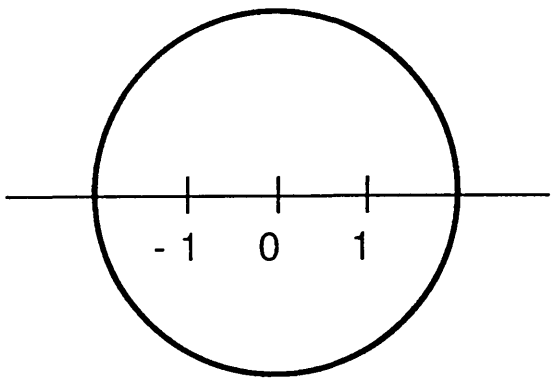
Finally assuming that u is independent of θ, this leads to the simplest equation with cylindrical symmetry

$$u_t = u_{xx} + x^{-1}u_x \qquad (3.54)$$

where x is the radial space variable. There may be difficulties at the point x=0, which we shall consider

3.10.1. Finite difference representation of the equation (3.54)

There is an apparent difficulty in representing the term $x^{-1}u_x$ at x=0, but this is eliminated by noting that there is symmetry about the line x=0 i.e u(x)=u(-x) see Fig(3.6)



Fig(3.6)

Hence we can assume an expansion about x=0 of the form

$$u=u_0 + a_2x^2 + a_4x^4 + \ldots\ldots$$

i.e $u_x= 2a_2x + 4a_4x^3 + \ldots.$

$$u_{xx}=2a_2 + 12a_4x^2 + \ldots\ldots\ldots$$

Thus at x=0 $x^{-1}u_x=u_{xx}$, and so

the equation is

$$u_t=2u_{xx}$$

The backward finite difference representation is

$$u_{0,t} -u_{0,t-\delta t} = 2k(u_{1,t} -2u_{0,t} + u_{-1,t})$$

and $u_{1,t}\equiv u_{-1,t}$ where $k=\delta t/(\delta x)^2$

This leads to

$$u_{0,t}-u_{0,t-\delta t}=4k(u_{1,t} - u_{0,t})$$

or $(1+4k)u_{0,t}-4ku_{1,t}=u_{0,t-\delta t}$

3.10.2. Singularity

$$u_{xx} + x^{-1}u_x=u_t \qquad (3.55)$$

This has a singularity when x=0 provided du/dx $\neq$ 0 at x=0

Taking the case $u_t=c$ (constant)

$$(d^2u/dx^2) +x^{-1}(du/dx) =c$$

$$x^{-1}d[x(du/dx)]=c$$

$$d[x(du/dx)]dx=cx$$

with solution

$$u=cx^2/4 +b*\ln(x) + a$$ where a, b are constants.

Hence $u\rightarrow\infty$ as $x\rightarrow 0$ and this can cause a difficulty, with inaccurate approximations for x small.

3.11.Two cylindrical problems

We consider a unit cylinder at a temperature greater than zero, and the ice either outside or inside this cylinder. We thus have two distinct problems.

3.11.1. Ice outside cylinder

The equation is

$$u_{xx}+x^{-1}u_x=u_t$$

with initial conditions

(t=0) u=0 x>1

and boundary conditions

on the fixed boundary (x=1)

$$u_x=-1 \qquad t>0$$

on the moving boundary (x=s(t)>1)

$u=0$, $dx/dt = -u_x$

In the solution of a cylindrical problem we use the same conditions as in the cartesian problem (see (3.6)—(3.10) section (3.2)). Hence we use the following implicit formula, see Fig(3.7)

$$(1/\delta t)\nabla_t u_{0,t}=[1/(\delta x)^2]\delta_x^2 u_{0,t} +(x\delta_x)^{-1}\mu\delta_x u_{0,t}$$

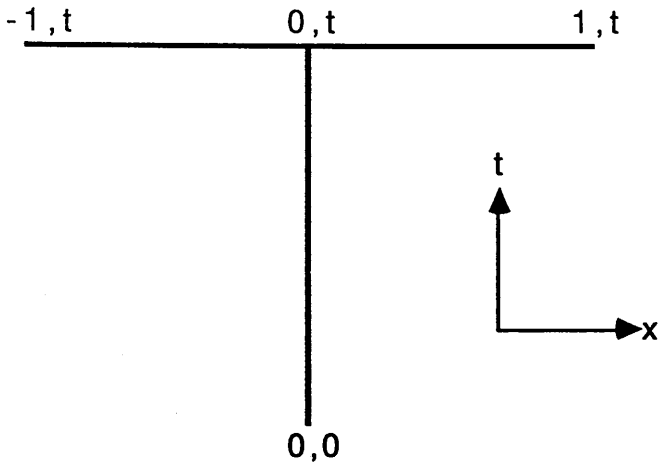
i.e $u_{0,t}-u_{0,0} = [\delta t/(\delta x)^2]\{\delta_x^2 + \delta_x\mu\delta_x\}u_{0,t}$

or $u_{0,t} -u_{0,0}= k \{u_{1,t} -2u_{0,t} +u_{-1,t} +(\delta_x/2x)(u_{1,t} -u_{-1,t})\}$

$$(1+2k)u_{0,t} -k(1+ \delta_x/2x)u_{1,t} -k(1-\delta_x/2x)u_{-1,t}=u_{0,0}$$

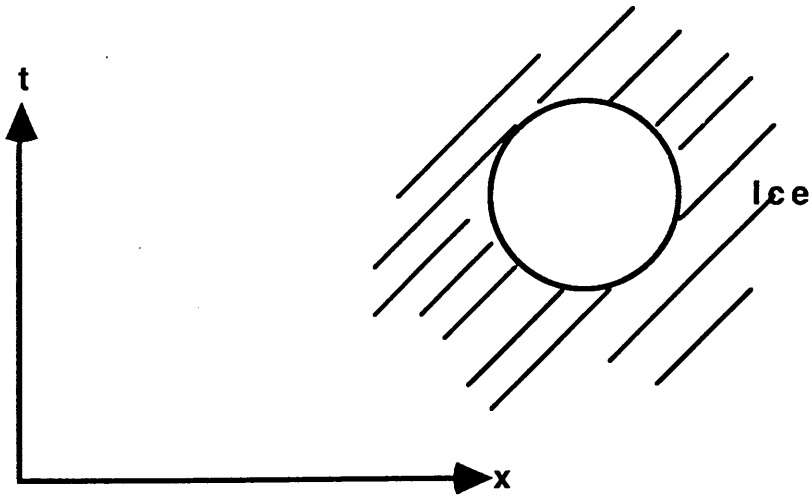
i.e

$$(1+2k)u[i,j+1]-k(1+\delta_x/2x)u[i+1,j+1]-k(1-\delta_x/2x)u[i-1,j+1]=u[i,j] \qquad (3.56)$$



Fig(3.7)

Melting outside the unit cylinder $1 \leq x$ then



Fig(3.8)

Point 0,0 corresponds to t=0, x=1

Point i,j corresponds to x=1+iδx, t=jδt

The equation is

$$(1+2k)\ u[i,j+1]\ -k(1+0.5/(1/\delta x+i))\ u[i+1,j+1]$$

$$-k(1-0.5/(1/\delta x\ +i))u[i-1,j+1]=u[i,j] \qquad (3.57)$$

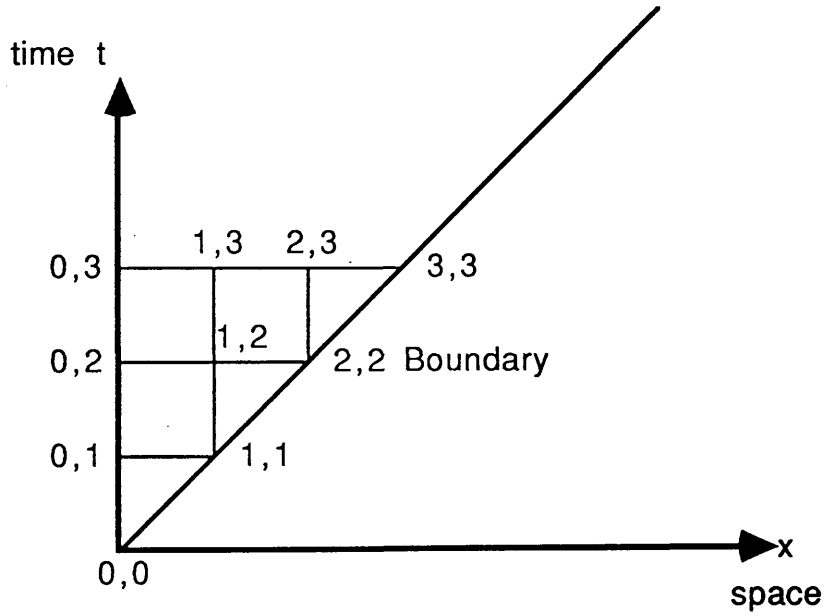
$$u[0,j+1]=u[1,j+1]+\delta x$$

On the boundary, $u=0$

and so $u[0,0]=0$

$u[1,1]=0$ etc see Fig(3.9)

Also $u[0,1]=\delta x$



Fig(3.9)

3.11.2. Ice inside the cylinder

The equation is

$$u_{xx}+x^{-1}u_x=u_t$$

with initial conditions

(t=0) u=0 x<1

and boundary conditions

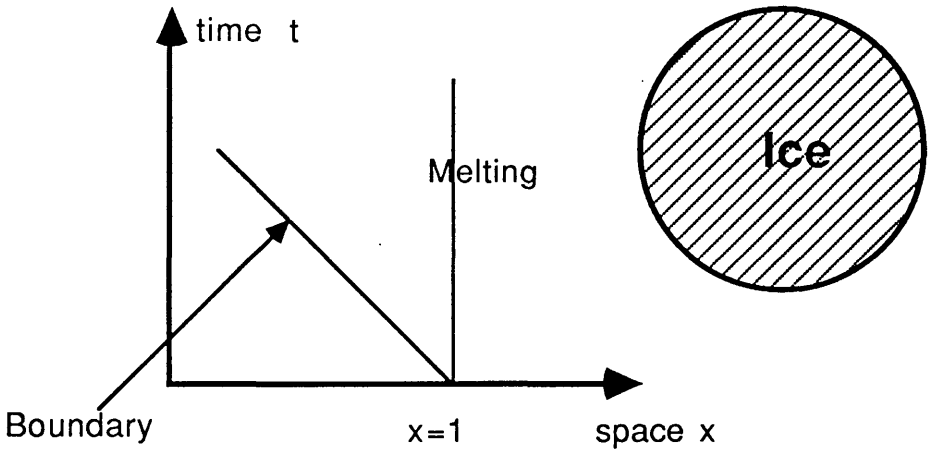
on the fixed boundary (x=1)

$u_x=-1$ t>0

on the moving boundary (x=s(t)<1)

u=0, dx/dt = u_x

Melting ice inside unit cylinder see Fig(3.10)



Fig(3.10)

Point 0,0 corresponds to x=1, t=0

Point i,j corresponds to $x=1-i\delta x$, $t=j\delta t$. Equation is

$$-k(1+0.5/(1/\delta x-i))u[i-1,j+1]+(1+2k)u[i,j+1]-$$

$$k(1-0.5/(1/\delta x-i))u[i+1,j+1]=u[i,j] \qquad (3.58)$$

$i\neq 0$, $1/\delta x$ changes slope when $i=1/\delta x$, i.e $j=1/\delta x$ -1

Same conditions for the initial value as for outside the cylinder.

3.11.3. Other numerical methods

Another technique which has been used to solve the cylindrical problem is to apply the Douglas formula (see (2.32) section **2.11**) with central differences for the space derivatives,

and a forward difference for the time derivative. The finite difference replacement of (3.55) may then be written as

Outside cylinder

$$[-0.5(r-1/6)+0.5r/((1/h)+i)]u_{i-1,j+1} + (r+5/6)u_{i,j+1} +[-0.5(r-1/6)-0.5r/((1/h)+i)]u_{i+1,j+1}=0.5(r+1/6)u_{i-1,j} -(r-5/6)u_{i,j} +0.5(r+1/6)u_{i+1,j} \quad (3.59)$$

and inside the cylinder,

$$[-0.5(r-1/6)-0.5r/((1/h)-i)]u_{1-1,j+1}+(r+5/6)u_{i,j+1}u_{i+1,j}+[-0.5(r-1/6)+0.5r/((1/h)-i)]u_{i+1,j+1}=0.5(r+1/6)u_{i-1,j}-(r-5/6)u_{i,j}+0.5(r+1/6)u_{i+1,j} \quad (3.60)$$

3.12. Numerical Solution of the Cylindrical Problem

The computation and the permitted error are the same as in the cartesian problem. (See section **3.9.**) Various results are given in tables (6.12)—(6.19) using the method of Gupta & Kumar and the new finite difference replacement of (3.55) is given as in (3.56)—(3.60), section **3.11.** Furthermore in all our methods we solve the problems by using one of the four formulae Δt_j (3.46)—(3.49).

Fig-3.11- Position of moving boundary vers us time

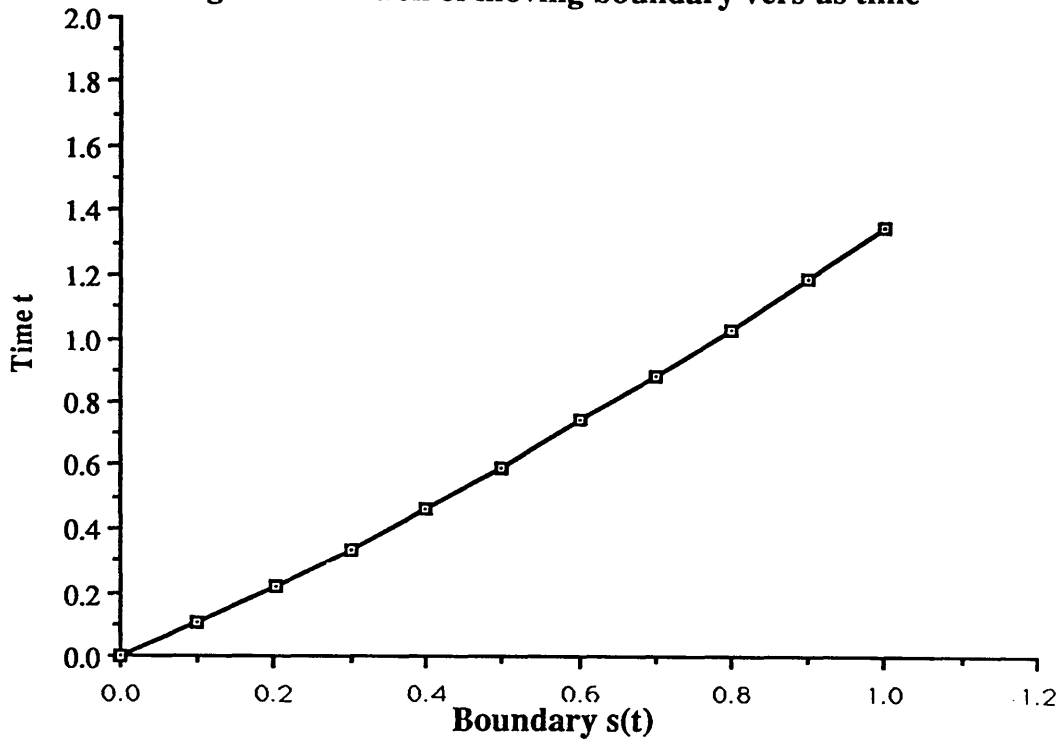
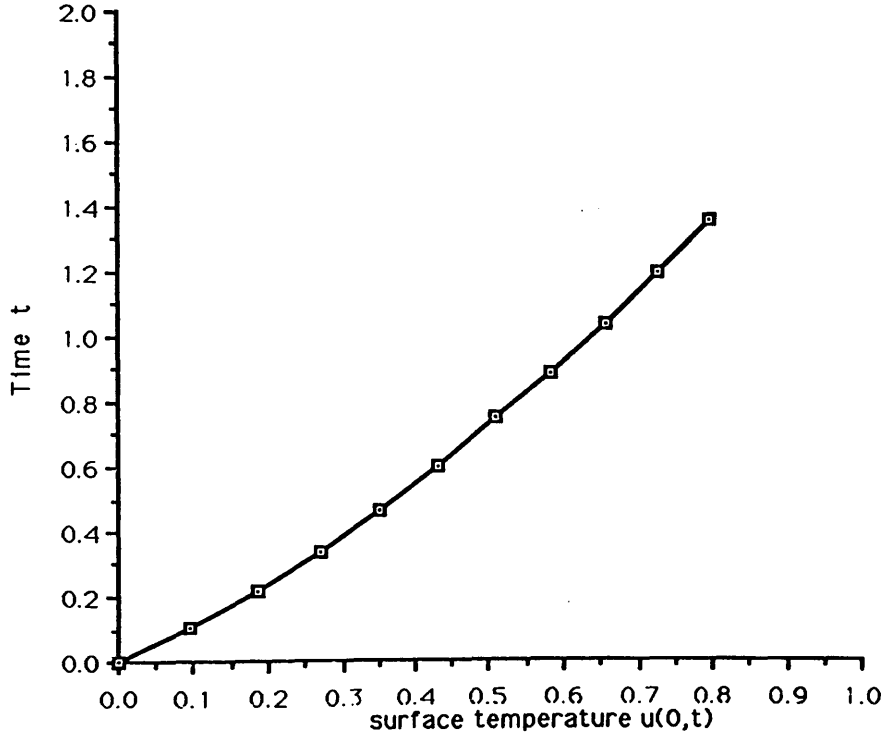


Fig-3.12- Temperature at fixed surface vers us time



Chapter 4

Solution of an ice melting problem using a fixed domain method with a moving boundary

4.1. Several Methods

A variety of diffusion problems with moving boundaries are described in Ockendon and Hodginks (1975) where we observe that analytical solutions are possible for some problems and arise when there is a singularity in the region of solution or on the boundary e.g Fox in (Ockendon and Hodginks 1975 p228) mentions the need for a short time analytical solution in the case of discontinuous agreement of initial and boundary conditions.

But the general approach should be necessarily numerical. Crank in (Ockendon and Hodginks 1975) discusses finite difference methods for the classical one dimensional diffusion problem, which include variable space steps and variable time steps and a change of space variable to fix the moving boundary.

Other methods for the numerical solution of the one dimensional problem with a moving boundary are compared by Furzeland (1980).

4.2. Problem

This is the problem mentioned in section 3.5 , viz

$u_t=u_{xx}$ $(0<x<s(t),\ t>0)$ (4.1)

$u_x=-1$ $(x=0,\ t>0)$ (4.2)

$u=0$ $(x\geq s(t),\ t\geq 0)$ (4.3)

$dx/dt=-u_x$ $(x=s(t),\ t>0)$ (4.4)

$s(0)=0$ (4.5)

In a paper which is reviewed in detail here, Morland introduces another transformation to fix the moving boundary by replacing the time by the boundary position as the time variable. Hence we obtain a non-linear equation on a known domain.

This method introduces the boundary velocity expressed in terms of the boundary position, which will be updated at each step. Here the iteration to update the extrapolated boundary velocity is required. An essential feature of the transformation is the monotonicity of the boundary position with respect to time over the interval considered. From now on, we concentrate on Morland's procedure which is now outlined. We know that s(t) is monotonic and, so s can replace t as an independent variable. Defining

$u(x,t)=c(x,s)$ (4.6)

$s'(t)=¥(s)\geq 0$ (4.7)

the diffusion equation (4.1) becomes

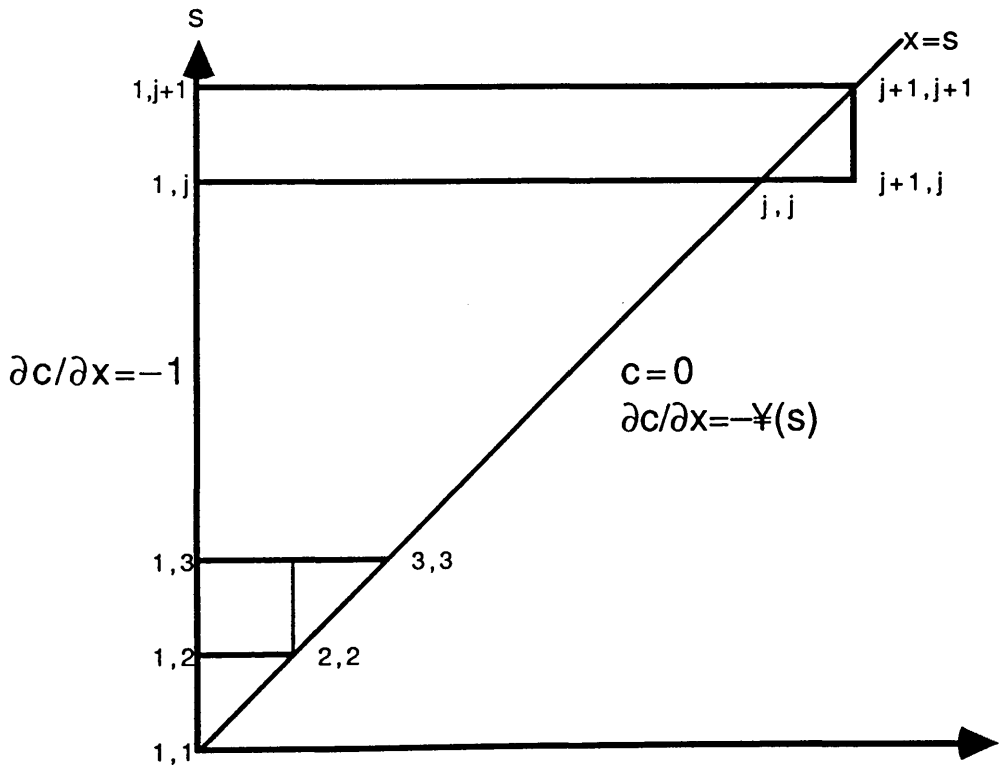
$¥c_s=c_{xx}$ $0<x<s$ (4.8)

where the time at each s is calculated by

$$t=\int_0^s ds'/¥(s') \qquad (4.9)$$

The boundary velocity ¥(s) is a new dependent variable to be determined simultaneously with c(x,s) from equation (4.8) subject to initial and boundary conditions (4.3)—(4.5).

Now we seek a finite difference replacement of eqn (4.8). The region to be examined is the domain 0<x<s which is covered by a rectilinear grid, with grid points (j,j) lying on the boundary x=s see Fig(4.1).



Fig(4.1)

Let the step length be h and introduce the following notation ;

$$u_{i,j}=c(x_i,s_j) \qquad ¥_j=¥(s_j) \tag{4.10}$$

where $x_j=(i-1)h$, $s_j=(j-1)h$ (i,j=1,2,3,4, ...)

Following Ames (1977) the Crank-Nicolson implicit finite difference approximation for equation (4.8) with arbitrary weighting $(0\le r\le 1)$ between the j+1 and j levels of s is

$$¥c_s=¥_j(c_{i,j+1}-c_{i,j})/\delta s \tag{4.11}$$

$$c_{xx}=(r/(\delta x)^2)(c_{i-1,j+1}-2c_{i,j+1}+c_{i+1,j+1})+ ((1-r)/(\delta x)^2)(c_{i-1,j}-2c_{i,j}+c_{i+1,j}) \tag{4.12}$$

with $\delta x=\delta s=h$

In order to obtain a more accurate representation we introduce another weight $k(0\le k\le 1)$, and instead of $¥_j$ we use the variable

$$ß_j=kh¥_{j+1}+(1-k)h¥_j \ , \tag{4.13}$$

which leads to the following equation

$$rc_{i-1,j+1}-(2r+ß_j)c_{i,j+1}+rc_{i+1,j+1}=-(1-r)c_{i-1,j}+[2(1-r)-ß_j]c_{i,j}-(1-r)c_{i+1,j} \tag{4.14}$$

The boundary condition (4.2) gives

$$c_{1,j+1}-c_{2,j+1}=h \tag{4.15}$$

and the moving boundary condition (4.3) gives

$$c_{j+1,j+1}=0 \tag{4.16}$$

From the above we deduce that (4.14),(4.15) with (i=2,3, ...j), (j=3,4,5, ...) constitute a tridiagonal linear system of equations for the j unknowns $c_{i,j+1}$(i=1,2,3, ...j) which can be solved by a Thomas algorithm (see appendix-A-). The exterior value $c_{j+1,j}$ may be estimated by the Taylor series

$$c_{j+1,j+1}=c_{j,j}+hc_x|_j+(1/2)h^2c_{xx}|_j+ \ldots \ldots \quad (4.17)$$

where $c_x|_j$ is the first partial derivative of c with respect to x at level j on x=s. We know that c=0 and c_x=-¥, and so

$$c_{j,j}=0 \text{ and } c_x|_j=-¥_j \quad (4.18)$$

In order to find c_{xx} we recall that

$$dc=c_x dx+c_s ds$$

and since x=s, (dx/ds)=1, and (dc/ds)=0, we deduce that

$$c_s=¥ \quad (4.19)$$

and from (4.8) we have

$$(c_{xx})=¥c_s=¥^2 \quad (4.20)$$

From (4.17), (4.18), and (4.20) we deduce that

$$c_{j+1,j}=-h¥_j+(1/2)h^2¥_j \quad (4.21)$$

and similarly we may approximate $¥_{j+1}$ (boundary velocity) at j+1 using the Taylor series,

$$c_{j,j+1}=c_{j+1,j+1}-hc_x|_{j+1}+(1/2)h^2(c_{xx})|_{j+1}+ \ldots \ldots$$

$$=h¥_{j+1}+(1/2)h^2¥_{j+1} \quad (4.22)$$

From (4.21) we have

$$¥_{j+1}=(1+\sqrt{(1+2c_{j,j+1})})/h \quad (4.23)$$

which will be used to update $¥_{j+1}$ once $c_{j,j+1}$is determined from the above linear system. An iteration is started with $ß_j=h¥_j$

The time at each s is calculated by the integral (4.9) and in particular t_{j+1}, which is required in the boundary condition (4.15), is calculated from

$$t_1=0,\ t_2=(h/3)[(1/¥_2)+(4/¥_{3/2})+(1/¥_1)] \quad (4.24)$$

$$t_{j+1}=t_{j-1}+(h/3)[(1/¥_{j+1})+(4/¥_j)+(1/¥_{j-1})],\ (j=2,3,4, \ldots \ldots) \quad (4.25)$$

It remains to use the starting values for the temperature and the boundary velocity.

Alternatively we can use Taylor series expansions in (x,s) about (0,0) which will give better accuracy $O(h^3)$ in starting values for (4.14) than the finite difference approximation. Firstly from condition (4.2) and(4.4) we see that the initial velocity ¥=1 at x=s=0. In addition from (4.3) we deduce that the initial value of the temperature is zero leading to $c_{11}= 0$.

In order to obtain expressions for $c_{1,2}$, $c_{1,3}$, $c_{2,3}$ we need the first three partial derivatives of c with respect to s calculated at (0,0). From (4.19) if we use $\lim(x,s)\rightarrow(0,0)$ we deduce that

$$c_s(0,0)=1 \qquad (4.26)$$

Considering the second total derivative and using (4.3) we have

$$d^2c/ds^2=c_{ss}+2c_{xs}+c_{xx}=0 \ . \qquad (4.27)$$

But $2c_{xs}(0,0)=0$ (from condition 4.5) (4.28)

and equation (4.8) calculated at (0,0) gives $c_{xx}(0,0)=1$ because $c_s(0,0)=1$ and ¥=1.

Therefore from (4.27) we have $c_{ss}(0,0)=-1$.

Also $d^3c/ds^3=c_{sss}+3c_{xss}+3c_{xxs}+c_{xxx}=0$ (4.29)

with $c_{xss}=0$ (from 4.5) . (4.30)

We also have above that $c_{xx}(0,0)=1$, and by differentiation of (4.8),

$$c_{xxx}(0,0)=0 \ . \qquad (4.31)$$

Differentiation of (4.8) with respect to s gives

Differentiation of (4.8) with respect to s gives

$$c_{sxx}=¥c_{ss}+c_s(d¥/ds) \qquad (4.32)$$

and so (4.19) which is valid on x=s, gives

$$(d¥/ds)(0)=-1 \ ,$$

and so $c_{xss}(0,0)=-2$ (4.33)

We also deduce that $c_{sss}(0,0)=-3c_{xxs}(0,0)=6$, and (4.34)

$c_{1,2}$, $c_{1,3}$, and $c_{2,3}$ are

$c_{1,2}=c(0,h)=c(0,0)+hc_s(0,0)+(h^2/2)c_{ss}(0,0)+(h^3/6)c_{sss}(0,0)+ \ldots$

$\ldots .=h-(h^2/2)+h^3$,

$c_{1,3}=c(0,2h)=2h-2h^2+8h^3$,

and

$c_{2,3}=c(h,2h)=h-(3h^2/2)+6h^3$, respectively.

Similarly we obtain expressions for $(¥_{3/2})$, $¥_2$, $¥_3$ (boundary velocities)

On x=s from (4.19), we obtain

$d^2¥/ds^2=c_{sss}+2c_{xss}+c_{xxs}$

$¥"(0)=d^2¥/ds^2=4$

$(¥_{3/2})=¥(h/2)=¥(0)+(h/2)¥'(0)+(h^2/8)¥"(0)=1-(h/2)+(h^2/2)$

$¥_2=¥(h)=1-h+2h^2$

$¥_3=¥(2h)=1-2h+8h^2$

and the values of time are deduced from (4.24), (4.25) as

$t_1=0$

$t_2=h+(h^2/2)-(h^3/3)$

$t_3=2h+2h^2-8h^3/3$

Finally we can start the scheme with j=3 in (4.13)-(4.16), and (4.21), and (4.23).

In the first these the tridiagonal system (4.14), (4.15) is solved with $ß_j=h¥_j$ using the Gaussian elimination. Hence we calculate the value of $c_{j,j+1}$, i=1,2,3, $\ldots$.,j, for a fixed j and we can update the boundary velocity from relation (4.23).

After that the system is solved using

$ß_j=hk¥_{j+1}+(1-k)h¥_j$ and we imposed the convergence test

$$|¥^{(N)}_{j+1} - ¥^{(N-1)}_{j+1}| < h|¥^{(N-1)}_{j+1}| \qquad (4.35)$$

If the current value of $¥_{j+1}$ satisfies the halt criterion , it is employed to calculate the value of t_{j+1}, otherwise we perform another iteration in order to find a more accurate value for $¥_{j+1}$. For that reason we solve again the system with updated velocity and we repeat the procedure until (4.35) is satisfied. Superscript N is the iteration index.

4.3. Numerical Results

The calculations were performed with

a - r=k=0.5 and h=0.01 for $0<s\leq 5$

b- r=k=0.5 and h=0.02 for $0<s\leq 10$

as suggested in Morland's paper, the results agreeing with these found by Morland but the number of iterations being different.

(see results in chapter-6- Tables 6.20 & 6.21)

Chapter-5-

The Oxygen Diffusion Problem

5.1. Flow Example

A problem arises from the diffusion of oxygen in a medium where some of the oxygen is absorbed and thereby removed from the diffusion process. A moving boundary is an essential feature of this problem but the conditions which determine its movements are different, the concentration of oxygen always being zero at the boundary and no oxygen diffusing across the boundary at any time (for more details see Crank 1972).

5.2. Cartesian Coordinates:Statement of Problem

In one dimension, the diffusion with absorption process is represented by the parabolic partial differential equation.

$$C_T=D(C_{XX}) - m$$

where C(X,T) denotes the concentration of oxygen free to diffuse at distance X from the outer surface of the medium at time T, D is a constant diffusion coefficient and m, a constant rate of consuption of oxygen per unit volume.

This problem has two parts:

5.2.1. *Steady State Solution*

When oxygen is entering through the surface we have

$C=C_0$, $X=0$, $T\geq 0$ with C_0 a constant (5.1)

The steady state is defined by a solution of $C_T=0$ viz

$$D(d^2C/dX^2) - m=0 \qquad (5.2)$$

which satisfies the conditions (i.e no oxygen can diffuse beyond this point)

$$C=C_X=0 \qquad X\geq X_0 \qquad (5.3)$$

where X_0 is the innermost extent of oxygen penetration, and on the outer surface X=0 $C = C_0$ = constant.

The required solution is readily seen to be

$$C=(m/2D)(X-X_0)^2 \qquad (5.4)$$

where $$X_0=\sqrt{(2DC_0)}/m \qquad (5.5)$$

5.2.2. *The Moving Boundary Problem*

After the surface X=0 has been sealed, the position of the receding boundary is denoted by $X_0(T)$ and the problem can be expressed by the equation

$$C_T=D(C_{xx}) - m \qquad 0\leq X\leq X_0(T) \qquad (5.6)$$

with the conditions

$$C_x=0 \qquad X=0 \qquad T\geq 0 \qquad (5.7)$$

$$C=C_x=0 \qquad X=X_0(T) \qquad T\geq 0 \qquad (5.8)$$

and $$C=(m/2D)(X-X_0)^2 \qquad 0\leq X\leq X_0 \qquad T=0 \qquad (5.9)$$

where T=0 is the time when the surface is sealed.

By making the change of variables

$$x=X/X_0, \qquad t=D(T/X_0^2), \qquad c=D/(mX_0^2)=C/2C_0$$

and denoting by s(t) the value of x corresponding to $X_0(t)$, the above system is reduced to the following non-dimensional form:

$$c_t= c_{xx}-1 \qquad 0\leq x\leq s(t) \qquad (5.10)$$

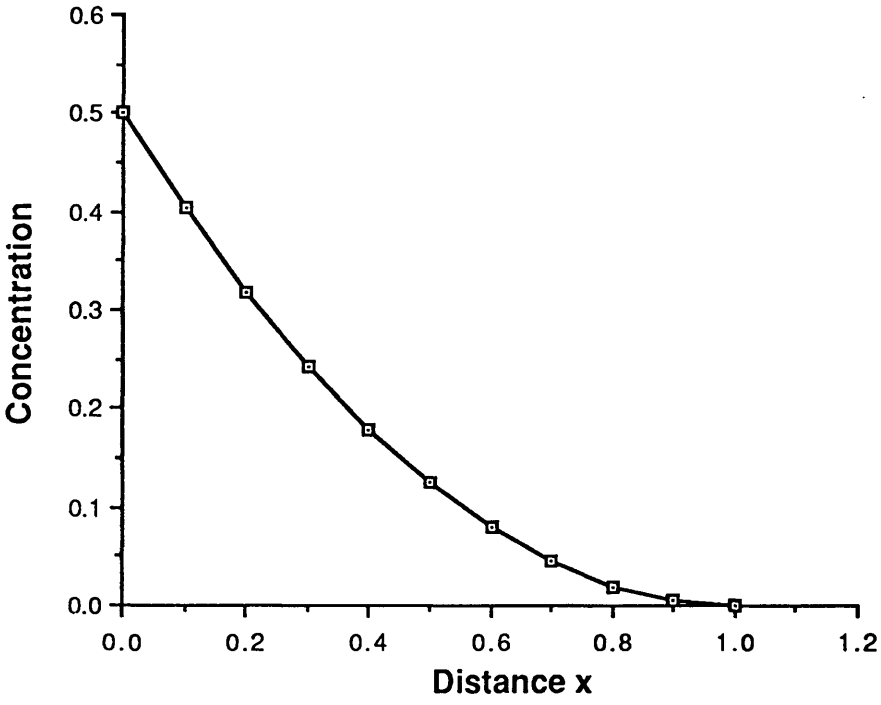
with $$c_x=0 \qquad x=0 \qquad t\geq 0 \qquad (5.11)$$

$$c=c_x=0 \qquad x=s(t) \qquad t\geq 0 \qquad (5.12)$$

and $c=0.5(1-x)^2$ $0\leq x\leq 1$ $t=0$ (5.13)

(see Fig-5.1-)

Fig-5.1-Concentration distributions for the steady-state



where c is the concentration of the oxygen free to diffuse. It is the absence of s_t in (5.12) that renders this problem implicit.

This solution also has a singularity at x=0, t=0 due to the instantaneous sealing of the surface. In mathematical terms this is noted by the conditions

$c=0.5(1-x)^2$ $0\leq x\leq 1$,

and $c_x=0$ $x=0$ when $t = 0$.

5.3. Approximate Analytical Solution

An approximate analytical solution can be used to move

away from the singularity at x=0, t=0. Thus in the range $0 \le t \le 0.020$, the expression

$$c(x,t)=0.5(1-x)^2-2\sqrt{(t/\pi)}\exp\{-(x/2\sqrt{t})^2\}+x\,\mathrm{erfc}(x/2\sqrt{t}) \quad 0 \le x \le 1 \quad (5.14)$$

is sufficiently accurate for most purposes (see Crank 1972).

This analytical solution can also be obtained by using Fourier expansion. This is relevant to the cylindrical coordinate case (see Sec 5.7). The cosine expansion for a function f(x) over the range $0<x<1$ is

$$f(x)=\sum_{s=0}^{\infty} A_s \cos(s\pi x) .$$

For this expansion, $f_x(0)=0$, thus satisfying the derivative condition $c_x=0$, $t \ge 0$. For the function $f(x)=x^2$,

$$x^2=(1/3)-(4/\pi^2)\sum_{s=1}^{\infty}\left[(-1)^{s-1}\cos(s\pi x)\right]/s^2$$

so that

$$c[x,0]=(1/6)-(2/\pi^2)\sum_{s=1}^{\infty}\left[(-1)^{s-1}\cos(s\pi(1-x))\right]/s^2 , \quad 0<x \le 1 .$$

Note that $$c_x[x,0]=(2/\pi)\sum_{s=1}^{\infty}\left[(-1)^{s-1}\sin(s\pi(1-x))\right]/s$$

and $c_x=0$, when $x=0$

Assume $$c=a_0(t) + \sum_{s=1}^{\infty} a_s(t)\cos(s\pi(1-x))$$

then $$c_t=da_0/dt+\sum_{s=1}^{\infty}(da_s/dt)\cos(s\pi(1-x))$$

and $$c_{xx}=-\sum_{s=1}^{\infty}a_s\, s^2\pi^2\cos(s\pi(1-x))$$

Hence the equation becomes

$$(da_0/dt)+\sum_{s=1}^{\infty}da_s/dt)\cos(s\pi(1-x))=-\sum_{s=1}^{\infty}a_s\, s^2\pi^2\cos(s\pi(1-x))-1$$

Equating coefficients of $\cos(s\pi(1-x))$, we get

$$da_0/dt=-1,$$

and $$da_s/dt=-s^2\pi^2a_s\ , \qquad s\geq 1$$

giving $$a_0=A_0-t$$

and $$a_s=A_se^{-s^2\pi^2t} \quad \text{where } A_s=\text{constant}$$

Finally because of the initial condition t=0 , we get

$$c[x,t]=(1/6)-t\ -(2/\pi^2)\sum_{s=1}^{\infty}\left[(-1)^{s-1}e^{-s^2\pi^2t}\cos(s\pi(1-x))\right]/s^2 \qquad (5.15)$$

5.4. Numerical Method

Once the moving boundary has started to move we resort to numerical methods of solution. Several methods have been proposed (see Douglas & Gallie (1955), Murray & Landis (1959), Ehlric (1958), Loktin (1960), Koh Et Al (1969), Saitoh (1972), Bonnerot & Janet (1974), Meyer (1976)). Crank (1957) suggested a three point Lagrangian Interpolation Formula near the moving boundary, where the concentrations at the intermediate points

between the two boundaries have been calculated by using simple explicit finite difference formulae and the location of the moving point itself is determined by Taylor series.

Confining attention to three-point formulae, we have

$$f(x)=\sum_{j=0}^{2} l_j(x)f(a_j) \qquad (5.16)$$

where $l_j(x)=p_2(x)/(x-a_j)p'_2(a_j)$ (5.17)

with $p_2(x)=(x-a_0)(x-a_1)(x-a_2)$ (5.18)

and $p'_2(a_j)$ is its derivative with respect to x at $x=a_j$. This leads to

$$d^2f(x)/2dx^2=[f(a_0)/(a_0-a_1)(a_0-a_2)]+[f(a_1)/(a_1-a_0)(a_1-a_2)]+$$
$$[f(a_2)/(a_2-a_0)(a_2-a_1)] \qquad (5.19)$$

and $df/dx=l'_0(x)f(a_0)+l'_1(x)f(a_1)+l'_2f(a_2)$ (5.20)

where $l'_0(x)=[(x-a_1)+(x-a_2)]/(a_0-a_1)(a_0-a_2)$

or (5.21)

$l_0(x)=[(x-a_1)(x-a_2)]/(a_0-a_1)(a_0-a_2)$

Similarly for $l'_1(x)$, and $l'_2(x)$.

We apply these formula in the neighbourhood of the moving boundary at time $t=j\delta t$ when there is a fractional distance $p\delta x$ between the grid lines $i\delta x$ and $(i+1)\delta x$. The points a_0, a_1, and a_2 are identified with the grid lines $(i-1)\delta x$, $i\delta x$ and the moving boundary itself and correspondingly $f(a_0)$, $f(a_1)$, and $f(a_2)$ with $c_{i-1,j}$ $c_{i,j}$ and $c\dagger$ on the boundary. Then for $x<s(t)$ we identify (see Fig-5.2-)

$$f(a_0)=c_{i-1}, \qquad f(a_1)=c_i, \qquad f(a_2)=c\dagger \qquad (5.22)$$

Then (5.19) and (5.20) become

$$c_{xx}=(2/(\delta x)^2)[(c_{i-1}/p+1)-(c_i/p)+(c\dagger/p(p+1))],\quad x=i\delta x \qquad (5.23)$$

and

$$c_x=(1/\delta x)[(pc_{i-1}/p+1)-((p+1)c_i/p)+((2p+1)c\dagger/p(p+1))],\quad x=s(t) \qquad (5.24)$$

and similarly for $x>s(t)$ we have

$$c_{xx}=(2/(\delta x)^2)[(c\dagger/(1-p)(2-p))-(c_{i+1}/(1-p))+(c_{i+2}/(2-p))], \qquad (5.25)$$

$x=(i+1)\delta x$

and

$$c_x=(1/\delta x)[((2p-3)c\dagger/(1-p)(2-p))+((2-p))c_{i+1}/(1-p))- \qquad (5.26)$$
$$((1-p)c_{i+2}/(2-p)),\qquad x=s(t)$$

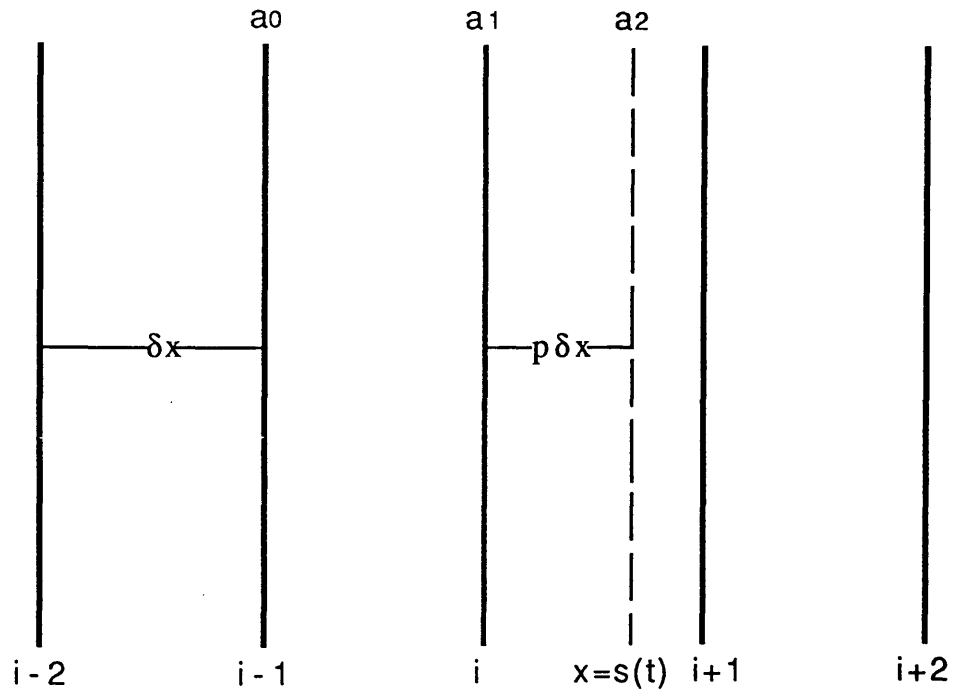


Fig-5.2-

We use these formula for the space derivatives in conjunction with the usual explicit or implicit replacement of the time derivative in the heat flow equation and in the conditions on the moving boundary $x=s(t)$. For points other than $i\delta x$, $s(t)$ and

$(i+1)\delta x$, we use the usual finite difference formula for equal space intervals including any conditions on $x=0$.

The numerical solution of the one-phase problem defined by equation (1.1_1.6) provides simple equations at the values δx, $2\delta x$, $3\delta x$, ... ,$n\delta x$, ... and $(i-1)\delta x$

Equation (1.1) is replaced by the simple explicit formula

$$c_{n,j+1}=c_{n,j}+(\delta t/(\delta x)^2)(c_{n-1,j}-2c_{n,j}+c_{n+1,j}) \quad n=1,2,3, \ . \ . \ (i-1) \qquad (5.27)$$

and from (1.2) $c_{o,j}=1$ at the point $i\delta x$. Instead of (5.27) we write, using (5.23),

$$c_{i,j+1}=c_{i,j}+(2\delta t/(\delta x)^2)[c_{i-1,j}/(p_j+1)-c_{i,j}/p_j], \qquad (5.28)$$

since $c\dagger=0$ from (1.5). Similarly substitution of (5.24) into (1.6) and writing $s_j=(i+p_j)\delta x$ gives

$$p_{j+1}=p_j+(\delta t/(\delta x)^2)((p_jc_{i-1,j})/p_j+1)-((p_j+1)c_{i,j}/p_j) \quad . \qquad (5.29)$$

Here we assume that the concentrations at each of the grid points, at the jth time level are known and $s=[(i-1)+p_j]\delta x$. As mentioned above p_j is positive and usually less than one. Hence the concentration at the (j+1)th time level up to and including the mesh point i-2 can be calculated using the well known explicit formulae (see Fig-5.3-)

$$(c_{0,j+1}-c_{0,j})/\delta t=(2/(\delta x)^2)(c_{1,j}-c_{0,j})-1 \qquad (5.30)$$

$$(c_{n,j+1}-c_{n,j})/\delta t=[(c_{n-1,j}-2c_{n,j}+c_{n+1,j})/(\delta x)^2]-1 \qquad (5.31)$$

$$n=1,2,3,4, \ . \ . \ . \ . \ (i-2)$$

So from the above we have

$$c_{xx}=(2/(\delta x)^2)[(c_{i-2}/(1+p))-(c_{i-1,j}/p)]-1 \qquad (5.32)$$

and the appropriate finite difference replacement at the point $(i-1)\delta x$ leads to

$$(c_{i-1,j+1}-c_{i-1,j})/\delta t=(2/(\delta x)^2)[(c_{i-2,j}/(1+p_j))-(c_{i-1,j}/p_j)]-1, \qquad (5.33)$$

an explicit expressions for $c_{i-1,j+1}$

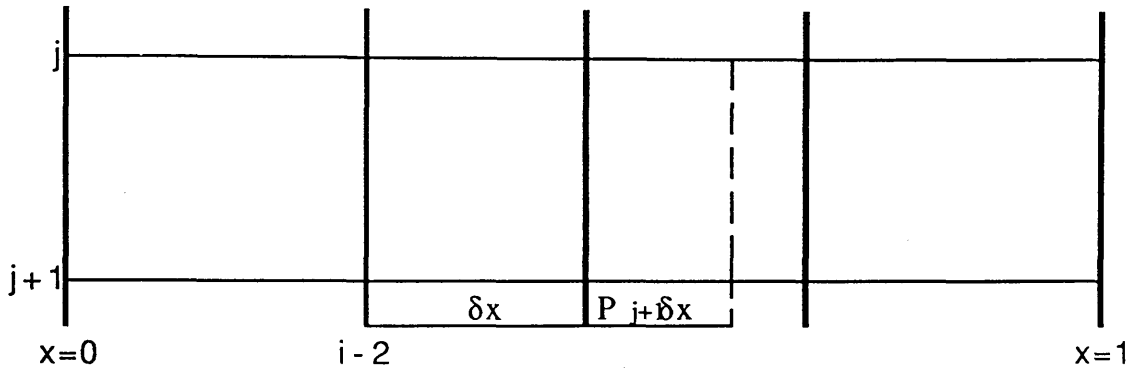


Fig-5.3-

5.4.1. Position of The Moving Boundary

Crank and Gupta (1972) in the absence of an explicit expression such as the Stefan condition, for the velocity of the moving boundary, used the two boundary condition (5.12) to deduce higher space derivatives for substitution into the Taylor Series.

Differentiation of the first of (5.12) with respect to t gives

$$dc/dt=(c_x)_{x=s}(ds/dt)+(c_t)_{x=s}=0 \qquad (5.34)$$

By using (5.10) and the second of (5.12) in (5.34) we deduce that

$$c_{xx}=1 \qquad x=s \qquad (5.35)$$

Differentiation of (5.10) with respect to x, gives

$$c_{xt}=c_{xxx} \qquad (5.36)$$

Again differentiating the second of (5.12) gives

$$d(c_x)/dt=(c_{xx})_{x=s}(ds/dt)+(c_{tx})_{x=s}=0 \qquad (5.37)$$

and hence using (5.35) and (5.36) in the above and assuming that the order of differentiation with respect to x and t can be interchanged, we obtain

$$c_{xxx}=-ds/dt, \qquad x=s \qquad (5.38)$$

Similarly

$$c_{xxxx}=(ds/dt)^2, \qquad c_{xxxxx}=-(d^2s/dt^2)-(ds/dt)^3 \quad \text{etc} \qquad (5.39)$$

Now, the Taylor series for c_{i-1}, the oxygen concentration at the grid point $(i-1)\delta x$ obtained by expanding about the moving boundary point can be written as (see Fig-5.3-).

$$c_{i-1}=c(s)-p\delta x(c_x)_{x=s}+((p\delta x)^2/2)(c_{xx})_{x=s}-((p\delta x)^3/6)(c_{xxx})_{x=s}+\ldots$$

$$=((p\delta x)^2/2)+((p\delta x)^3/6)ds/dt+\ldots\ldots \qquad (5.40)$$

Provided the boundary is not moving too quickly, the first term of the series provides a reasonable approximation and gives

$$p=(\sqrt{2}c_{i-1})/\delta x \qquad (5.41)$$

We shall see later that the boundary moves faster towards the end of the process and we then replace the finite difference solution by an analytical expression. When $c_{i-1,j+1}$ has been calculated from (5.33), the relation (5.41) gives the position of the moving point at the (j+1)th time level.

5.4.2. Moving Boundary Crossing A Mesh Line

If c_{i-1} goes on decreasing, we have

$$\text{(a)-} \quad c_{i-1,j+1}\leq 0 \qquad \text{or} \qquad \text{(b)-} \quad c_{i-1,j+1}>c_{i-1,j} \qquad (5.42)$$

where (a) is physically impossible, and (b) could be caused by instability (for a stability analysis see Appendix B). If the latter this happens at a point adjacent to the boundary, the (i-1)th mesh point is given up at the (j-1)th time level. The Lagrange Formula is then applied to $c_{i-2,,j}$ using $p_{j-1}=p_{j-1}+1$. The calculations were performed with:

(a)- t=0.001 and h=0.1

and

(b)- t=0.001 and h=0.05

as suggested in Crank's & Gupta's paper, (see numerical results in chapter-6-).

5.5. The Du Fort-Frankel Difference Scheme

We now solve the the problem posed by Crank using a new technique of applying the Du Fort-Frankel scheme, and following Crank's steps in solving the problem. This scheme has the advantage of being explicit and yet unconditionally stable. However the consistency condition requires that δt goes to zero faster than δx does, but the requirement is in practice not too severe if the coefficient of the second derivative is small. The simplest three level explicit scheme for the solution of the diffusion equation

$$c_t=c_{xx} \tag{5.43}$$

is

$$(c_{i,j+1}-c_{i,j-1})/2\delta t=(c_{i+1,j}-2c_{i,j}+c_{i-1,j})/\delta x^2 \tag{5.44}$$

which can be written in the form

$$c_{i,j+1}=2r(\delta x)^2 c_{i,j}+c_{i,j-1} \qquad r=\delta t/(\delta x)^2 \tag{5.45}$$

Du Fort-Frankel (1953) proposed that $(\delta x)^2 c_{i,j}$ be written as $(c_{i-1,j}-2c_{i,j}+c_{i+1,j})$ in (5.45) and $c_{i,j}$ altered to give $(c_{i,j+1}+c_{i,j-1})/2$. This leads to the scheme

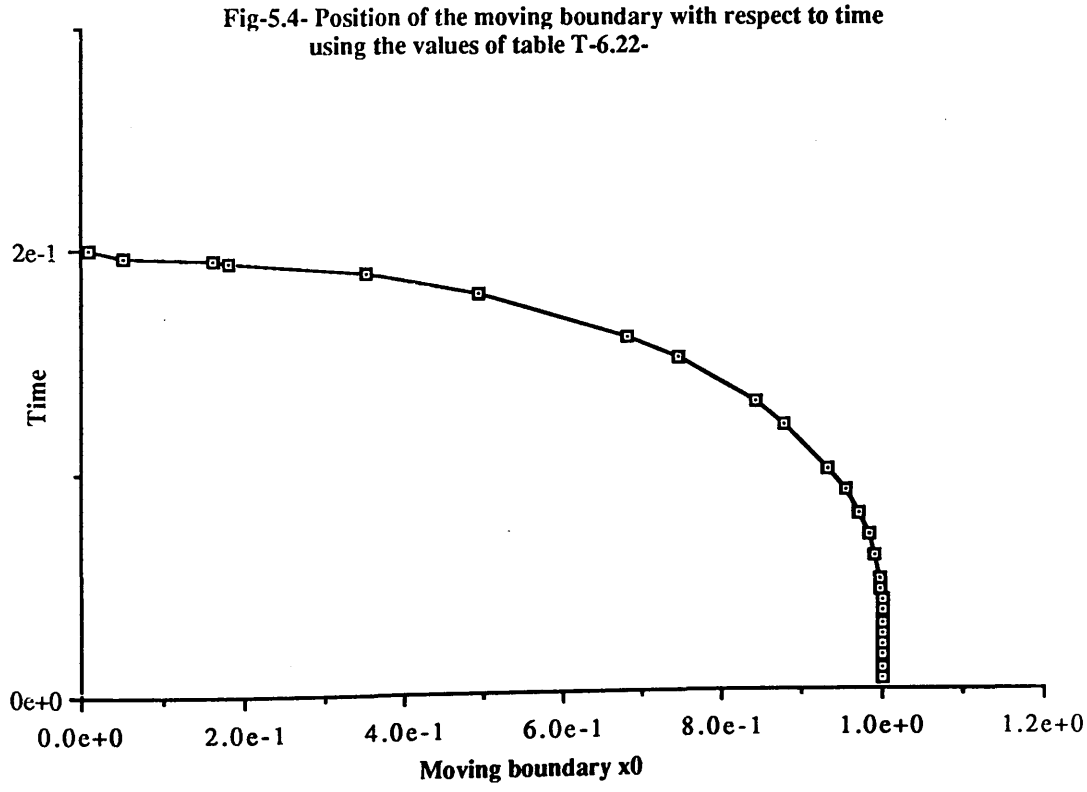
$$(1+2r)c_{i,j+1}=2r(c_{i-1,j}+c_{i+1,j})+(1-2r)c_{i,j-1} \ . \qquad (5.46)$$

Returning to our problem, in order to solve the first set of equations for $c_{i,2}$, we calculate the solution along the first time-level by using the explicit scheme

$$(c_{i,j+1}-c_{i,j})/\delta t=(c_{i-1,j}-2c_{i,j}+c_{i+1,j})/(\delta x)^2 .$$

The initial data along t=0 are known and so we calculate the concentration when x=0 by using the analytical solution (5.14), where at x=0 we have $c(0,t)=0.5-2\sqrt{(t/\pi)}$ (see tables of results in chapter-6-).

Fig-5.4- Position of the moving boundary with respect to time using the values of table T-6.22-



5.6. The Berger et al Problem.

Berger et al (1975) proved satisfactory graphical results for the Crank Gupta oxygen problem. They also examined a similar oxygen comsuption problem but modified the initial and fixed boundary conditions.

i.e they obtained a solution of $c_t=c_{xx}-1$

satisfying $c=c_x=0$ for $x=s(t)$. For the same $s(t)$ look for a solution of the form $c=f(x+t-1)=f(z)$

taking $x=1-t \equiv s(t)$.

Then $c_t=f'=df/dz$

$c_{xx}=f''=d^2f/dz^2$

and so $df/dz=(d^2f/dz^2) -1$

If $df/dz=p$ then $p=(dp/dz)-1$,

i.e $p=Ae^z-1$

or $df/dz=Ae^z-1$

i.e $f=Ae^z-z+B$

or $c=Ae^{(x+t-1)}-x-t+1+B$

leading to

$c_x=Ae^{(x+t-1)}-1$

When $x=1-t$ $c=A+B$,

and $c_x=A-1$.

Hence $A=1$, $B=-1$, and $c=e^{(x+t-1)}-x-t$

leading to $c(x,0)=e^{(x-1)}-x$

and $c(0,t)=e^{(t-1)}-t$.

Finally we can solve the problem using the numerical method mentioned above.

5.7. The Cylindrical Problem

A general version of the oxygen problem in cylindrically shaped sections of tissue is described by Galib, Bruch and Soloss (1981). They assumed the following analytic expression for c which satisfies the tissue boundary conditions and which is zero and has a zero normal flux on the moving boundary

$$c(x,\theta,t)=0.5[\rho(\theta,t)-r]^2-[\rho(\theta,t)-r]^3/(3[\rho(\theta,t)-r_i])$$

and in non-dimensional terms the mathematical problem in two dimensions (r, and θ) and time, in cylindrical coordinates is defined by the equation

$$c_t=c_{xx} + (c_x/x) + (c_{\theta\theta}/x^2) + f(x,\theta,t,c) \qquad (5.47)$$

where $c(x,\theta,t)$ is the oxygen concentration in the tissue, x is the radial coordinate, θ is the angular coordinate, t is time and $f(x,\theta,t,c)$ is the rate of absorption of oxygen. $\rho(\theta,t)$ is the position of the moving boundary, r_i the sealed surface radius and r_0 the outer tissue radius. The boundary radius conditions are

$$c(\rho,\theta,t)=0 \qquad (5.48a)$$

$$c_r(\rho,\theta,t)=0 \qquad (5.48b)$$

$$c_\theta(\rho,\theta,t)=0 \qquad (5.48c)$$

$$c_r(r_i,\theta,t)=0, \qquad (5.48d)$$

and

$$c(r_0,\theta,t)=0. \qquad (5.48e)$$

The absorption function is found by substituting this expression for c into the partial differential equation (5.47). For numerical solution, the finite difference equations are formulated using the

Crank-Nicolson method (details are given by Galib et al (1981)).

The one dimensional oxygen diffusion problem solved by Crank and Gupta can be posed in terms of cylindrical coordinates.

Assuming that the concentration c is independent of θ, the equation is

$$c_t=c_{xx} + x^{-1}c_x -1. \qquad (5.49a)$$

Taking the boundary to be the unit cylinder (x=1)

the boundary conditions are:

at the boundary of the cylinder (x=1)

$$c_x=0 \qquad t\geq 0 \qquad (5.49b)$$

and on the moving boundary (x=s(t))

$$c=c_x=0 \qquad t\geq 0. \qquad (5.49c)$$

Initially (t=0) the moving boundary is at $x=x_1=s(0)$,

and c satisfies

$$d^2c/dx^2 + (dc/dx)x -1=0$$

This has the solution

$$c=0.25x^2 +A \ln(x) +B$$

and the boundary conditions ($x=1$, $x=x_1$) give

$$A=-0.5\ x_1^2 \qquad B=0.5(x_1^2)\ln(x_1)-0.25x_1^2$$

so that

$$c=0.25(x^2-x_1^2)-0.5x_1^2(\ln(x)-\ln(x_1)) \qquad (5.49d)$$

Note that from this initial solution ,

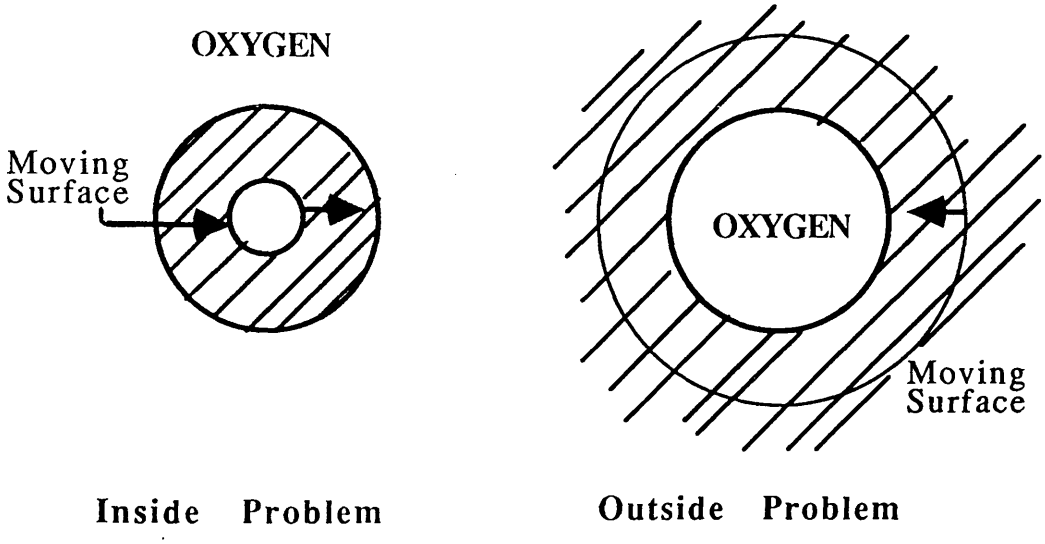
$$c_x=0.5x-0.5(x_1^2/x)$$

at the surface of the cylinder (x=1)

$$c=0.25(1-x_1^2) + 0.5x_1^2 \ln(x1)$$

and $c_x=0.5\ (1-x_1^2)$

Because of the condition $c_x=0$, $x=1$, $t>0$ the solution has therefore a singularity at the point $x=1$, $t=0$, as well as a moving boundary.



Inside Problem **Outside Problem**

Fig-5.5-

Finally there are two different problems to consider. The oxygen may be outside the cylinder and diffusing into the medium inside the cylinder where it is absorbed. Thus the equations hold inside the cylinder. Alternatively, the oxygen may be inside the cylinder and diffusing into the surounding medium as in Galib et al (1981). In this case the problem is solved outside the cylinder. These two cases will be considered separately.

5.8. Oxygen Diffusion Inside The Cylinder.

The problem involves the solution of the equation

$$u_t=u_{xx}+ (u_x/x)-1 \quad t\geq 0 \quad s(t)\leq x<1 \tag{5.50a}$$

with

$$u=u_x=0, \qquad x=s(t), \qquad t\geq 0, \quad s(0)=x_1 \tag{5.50b}$$

$$u_x=0 \qquad x=1 \qquad t\geq 0, \tag{5.50c}$$

and

$$u=0.25(x^2-x_1^2)-0.5x_1^2(\ln(x)-\ln(x_1)), \quad x_1\leq x<1, t=0 \tag{5.50d}$$

(see Fig-5.6-)

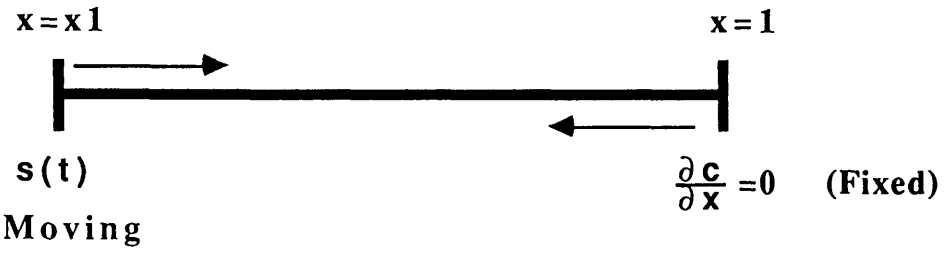


Fig-5.6-

The problem has a singularity at the point (x=1, t=0) so that an approximate analytical solution must be found for the first values of t.

5.9. Approximate Analytical Solution

It is known (see e.g Gray, Matthews and MacRobert 1952) that solutions of the homogenous partial differential equation

$$c_t=c_{xx} + x^{-1}c_x \tag{5.51}$$

are of the form

$$c= e^{-\lambda^2 t} (CJ_0(\lambda x) +DY_0(\lambda x))$$

where $J_0(\lambda x)$, $Y_0(\lambda x)$ are Bessel function of order zero. This suggests that we should look for a series solution of the form

$$c=a_0(t) + \sum_s a_s(t)\,(CJ_0(\lambda_s x) + DY_0(\lambda_s x)) .$$

Writing

$$u(x)= C\,J_0(\lambda x) + DY_0(\lambda x) ,$$

we note that u satisfies

$$u_{xx} + x^{-1}u_x + \lambda^2 u=0 .$$

Hence

$$c_t=da_0/dt + \sum_s a_s\,(u_{xx} + x^{-1}u_x)=\sum_s a_s\,\lambda_s^2\,u_s .$$

Equating coefficients, we get

$$da_0/dt= -1 ,$$

$$da_s/dt= -a_s\lambda_s^2 \qquad s\geq 1$$

giving $a_0=A_0-t$ $\qquad a_s=A_s e^{-\lambda_s^2 t}$

and $$c=A_0-t + \sum_s A_s e^{-\lambda_s^2 t}\,(J_0(\lambda_s x) + k_s Y_0(\lambda_s x)) .$$

When t=0, we have

$$c(0,x)=0.25\,(x^2-x_1^2)-0.5x_1(\ln(x))-\ln(x_1))\equiv f(x)$$

so that we require the Fourier Bessel Expansion

$$f(x)= A_0 + \sum_s A_s\,(CJ_0(\lambda_s x) + DY_0(\lambda_s x)) .$$

For the inside problem, c remains finite as $x\to 0$ so that $k_s=0$ and we can consider the expansion

$$f(x)=A_0 + \sum_s A_s\,J_0(\lambda_s x)$$

5.9.1. Expansion in terms of Bessel Functions

If λ,μ are constant then

$$(\mu^2-\lambda^2)\int_0^1 xJ_0(\lambda x)J_0(\mu x)dx=\lambda J_0'(\lambda)J_0(\mu)-\mu J_0'(\mu)J_0(\lambda)$$

and

$$\int_0^1 xJ_0^2(\lambda x)dx=0.5[J_0^2(\lambda)+J_0'^2(\lambda)] .$$

Hence, for the expansion

$$f(x)=A_0+\sum_s A_s J_0(\lambda_s x)$$

on multiplying by x, if s=0 or $xJ_0(\lambda_s x)$ otherwise and integrating over the range 0,1

if $J_0(\lambda)=0$, (the usual case)

we get $A_0=0$,

$$A_s=(\int_0^1 xf(x)J_0(\lambda_s x)dx)/(\int_0^1 xJ_0^2(\lambda_s x)dx)$$

$$=2(\int_0^1 xf(x)J_0(\lambda_s x)dx)/J_1^2(\lambda_s) \qquad s\geq 1 ,$$

whereas if $J_0'(\lambda)=0$,

we get $$A_0=(\int_0^1 xf(x)dx)/(\int_0^1 xdx)$$

$$A_s=(\int_0^1 xf(x)J_0(\lambda_s x)dx)/(\int_0^1 xJ_0^2(\lambda_s x)dx)$$

$$=2(\int_0^1 xf(x)J_0(\lambda_s x)dx)/J_0^2(\lambda_s) \qquad s\geq 1 .$$

We require to find the Fourier Bessel expansion of

$$f(x)=0.25(x^2-x_1^2)-0.5x_1^2(\ln(x)-\ln(x_1))$$

5.9.2. Bessel Function Integrals.

$$\int x^m J_0(x)dx=x^m J_1(x)-(m-1)\int x^{(m-1)}J_1(x)dx$$
$$=x^m J_1(x)+(m-1)\int x^{(m-1)}J_0'(x)dx$$
$$=x^m J_1(x)+(m-1)x^{(m-1)}J_0(x)-(m-1)^2\int x^{(m-2)}J_0(x)dx$$

$$\int xJ_0(x)dx=xJ_1(x)$$

so that

$$\int x^3 J_0(x)dx=x^3 J_1(x)+2x^2 J_0(x)-4xJ_1(x)$$
$$=(x^3-4x)J_1(x)+2x^2 J_0(x)$$

$$\int \ln(x)xJ0(x)dx=x\ln(x)J_1(x)+J_0(x)$$

5.9.3. After some algebra, we find that

{λ is root of $J_1(\lambda)=0$}

$$\lambda^2\int_0^1 xJ_0(x)dx=\lambda J_1(\lambda) ,$$

$$\lambda^4 \int_0^1 x^3 J_0(x)dx=(\lambda^3-4\lambda)J_1(\lambda) +2\lambda^2 J_0(\lambda) ,$$

and

$$\lambda^2 \int_0^1 \ln(x)xJ_0(x)dx=J_0(\lambda) -1,$$

so that

$$A_s= [(1-x_1^2)J_0(\lambda_s)+x_1^2]/\lambda_s^2 J_0^2(\lambda_s) \quad (5.52)$$

and $$A_0=0.125+0.5x_1^2 \ln(x_1) \quad (5.53)$$

5.10. Oxygen Diffusion Outside The cylinder.

The problem is to solve

$$u_t=u_{xx}+(u_x/x)-1, t\geq 0, (1\leq x\leq x_1) \quad s(t)\leq x_1 \quad (5.54a)$$

with $$u=u_x=0, \quad x=s(t), \ t\geq 0, \ s(0)=x_1=2 \quad (5.54b)$$

$$u_x=0 \quad x=1 \quad t\geq 0 \quad (5.54c)$$

$$u=0.25(x^2-x_1^2)-0.5x_1^2(\ln(x)-\ln(x_1)), \ 1\leq x\leq x_1=2, \ t=0. \quad (5.54d)$$

5.10.1. Approximate Analytical Solution

We assume a series solution of the following form

$$u(x,t)=a_0(t)+\sum_s a_s(t)C_sJ_0(\lambda_s x)+D_sY_0(\lambda_s x) \quad (5.55)$$

where $J_0(\lambda_s x)$, $Y_0(\lambda_s x)$ are Bessel functions of order zero

The solution has to satisfy the boundary conditions. In order to satisfy the boundary condition at x=1, we require

$$C_sJ_0'(\lambda_s) +D_sY'(\lambda_s)=0, \quad (5.56)$$

i.e assume the expansion of the following form

$$u(x,t)=a_0(t)+\sum_s a_s(t)\{J_0(\lambda_s x)/J_0'(\lambda_s x) - Y_0(\lambda_s x)/Y_0'(\lambda_s)\} \qquad (5.57)$$

To satisfy the boundary condition at $x=x_1$, the values of λ_s are given by

$$J_0(\lambda_s x_1)/J_0'(\lambda_s) - Y_0(\lambda_s x_1)/Y_0'(\lambda_s)=0 \qquad (5.58)$$

The coefficients $a_s(t)$ are given as before, i.e

$$a_0(t)=A_0-t$$

and $a_s(t)=A_s e^{-\lambda_s^2 t}$

In this case $A_0=0$,

$$A_s=P_s/Q_s \qquad s\geq 1$$

where $$P_s=\int_1^{x_1} xf(x)\{J_0(\lambda_s x)/J0'(\lambda_s) - Y_0(\lambda_s x)/Y_0'(\lambda_s)\}^2 dx$$

$$Q_s=\int_1^{x_1} x\{J_0(\lambda_s x)/J0'(\lambda_s) - Y_0(\lambda_s x)/Y_0'(\lambda_s)\}^2 dx=0.5\{x_1^2[J_0'(\lambda_s x_1)/J_0'(\lambda_s)] - Y_0'(\lambda_s x_1)/Y_0'(\lambda_s)]^2-[J_0(\lambda_s)/J_0'(\lambda_s) - Y_0(\lambda_s)/Y_0'(\lambda_s)]^2\}. \qquad (5.59)$$

Using the expressions for the integrals in sec *5.9.2.* which are also true for Y_0,

we get

$$A_s=\{-x_1[J_1(\lambda_s x_1)/J_0'(\lambda_s)-y_1(\lambda_s x_1)/y_0'(\lambda_s)]/\lambda_s^3+0.5(x_1^2-1)[J_0(\lambda_s)/J_0'(\lambda_s) - y_0(\lambda_s)/y_0'(\lambda_s)]/\lambda_s^2\}/Q_s \qquad s\geq 1 \qquad (5.60)$$

For the analytical and the numerical solutions, we use the same methods as mentioned for the inside problem, but the boundary conditions are different, (see Fig-5.8-).

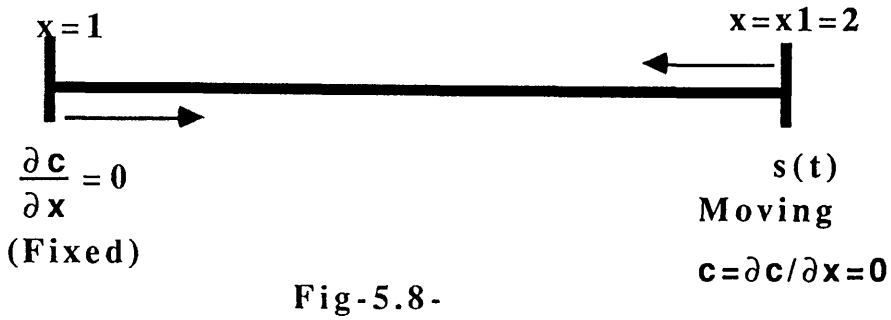


Fig-5.8-

Here in the numerical solution, for the interval 0.1, we have solved the problem using the analytical solution for the first three steps, but for the interval 0.05 we get the initial values from the formula (5.54d). See numerical results in tables (6.30, 6.31, 6.32) in chapter-6-.

5.11. Numerical Solution of Cylindrical Problems

For the numerical solution of the problem (5.49) we use the development of Taylor series near the moving boundary in the space direction where Crank used Lagrange type formula in the cartesian problem. (see Fig-5.7-)

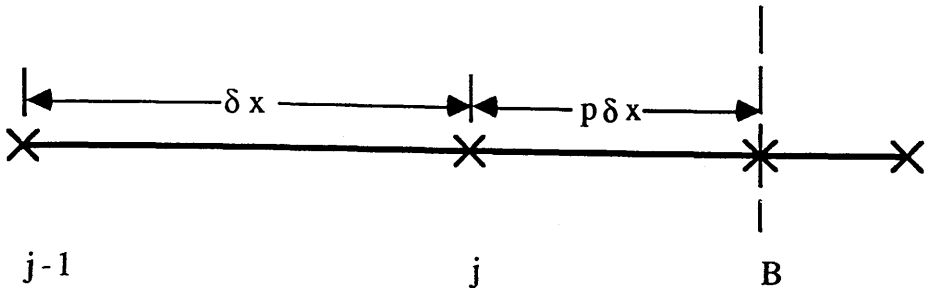


Fig-5.7-

We get

$$u_{xx}=(2/(\delta x)^2)[u_{j-1}/(p+1)-u_j/p+u_B/(p(p+1)) \qquad u_B=0 \qquad (5.61)$$

and

$$u_x=(1/\delta x)[-pu_{j-1}/(p+1)+(p-1)u_j/p+u_B/(p(p+1)) \quad u_B=0 \qquad (5.62)$$

Applying finite differences to (5.49a) we calculate the concentration from the following equations

$$u_{0,j+1}=2r(u_{1,j}-u_{0,j})+u_{0,j}-\delta t \qquad r=\delta t/(\delta x)^2 \quad (5.63)$$

$$u_{n,j+1}=r(1-0.5(1/((1/\delta x)+n)))u_{n-1,j}+(1-2r)u_{n,j}+r(1+0.5(1/((1/\delta x)+n)))u_{n,+1,j}+\delta t$$

$$n=1,2,3,\ldots.\ (i-2) \qquad (5.64)$$

and

$$u_{i-1,j+1}=u_{i-1,j}+2r(u_{i-2,j}/(p_j+1)-u_{i-1,j}/p_j)+((\delta t/\delta x)((-p_ju_{i-2,j})/(p_j+1)+$$
$$(p_j-1)u_{i-1,j}/p_j))/((i-1)\delta x)+\delta t \qquad (5.65)$$

For the position of the moving boundary and conditions crossing a mesh line, we use the same methods as in (5.4.1.) & (5.4.2.). (see numerical results in chapter-6-).

Finally the concentration at the intermediate points between the two boundaries has been calculated by equations (5.63) and

(5.64), but near the moving boundary by (5.65). The location of the moving boundary point and conditions crossing the mesh line are the same as in the cartesian problem.

Chapter-6-

Conclusion & Numerical Results

6.1. Summary

In general finite difference equations are used to solve initial value and boundary-value problems. It is required that finite difference equations be everywhere reasonably accurate, that is the local truncation error is small and the step by step process in t direction is stable.

In chapter Three the results are agree quite well where using different methods of calculations of time step and specially in cartesian problem are agree with those obtained by Douglas & Gallie and Gupta & Kumar where in every method, the implicit formula was used. For The simplest explicit scheme the results are not close to the results given when using the implicit scheme, may be because of the process is stable only for $0<\delta t/(\delta x)^2 \leq d$ where d is a number depending on which formula is used (0.5 for simple explicit).(tables 6.1__6.11 for cartesian problem). The method of Gupta and Kumar has been applied to similar problem in cylindrical coordinates using other techniques and finding the solution both inside and outside cylinder.(tables 6.12, 6.13, 6.16, 6.17 for outside cylinder and tables 6.14, 6.15, 6.18, 6.19 for inside cylinder).

In chapter four we have presented an alternative fixed domain transformation by replacing time by boundary position, appropriate one moving boundary in one space dimension and the

results are agree quite well with those found by Morland(1982) but the number of iteration is different.(see tables 6.20 and 6.21).

In chapter five, agreement for cartesianproblem with results of Crank and Gupta. Emphasis the importance of using the approximate analytical solution for the first few steps. We note that as the mesh size decrease, the position of the moving boundary as given by Crank and Gupta changes significantly. We therefore consider that the values as given by them are too small (tables 6.22__6.26 for cartesian problem). The method has been applied to a similar problem in cylindrical coordinates, finding the solution both inside and outside the cylinder. We obtained an approximate analytical solution in each case to deal with singularity. The results (tables 6.27___6.29 for the inside problem and tables 6.30___6.32 for the outside Problem) show that an accurate solution can be obtained.

Finally for more details about Fourier-Bessel expanssions and the roots of $J_0(ls), Y_0(ls)$ etc see (Bickley(1953), Watson(1944), N.B.S of Appl.Math series55 (1964), and Appendix C in thesis).

6.2. Numerical Results for Ice Melting Problem

Comparison of time step Δt, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.4**, the middle to **3.6**, and the lower to **3.8**. In **3.6** and **3.8**, $\Delta t_j^{(k)}$ is calculated by (3.46) . $\Delta x=0.01$

s(t)	Δt	t	k	u(0,t)
0.2	0.012	0.2172	2	0.1858
	0.012	0.2171	2	0.1858
	0.012	0.2170	2	0.1859
0.3	0.012	0.3378	2	0.2709
	0.012	0.3378	2	0.2709
	0.012	0.3375	2	0.2710
0.4	0.013	0.4655	2	0.3525
	0.013	0.4655	2	0.3525
	0.013	0.4652	2	0.3527
0.5	0.014	0.5999	2	0.4313
	0.014	0.5999	2	0.4314
	0.014	0.5994	2	0.4316
0.6	0.014	0.7405	2	0.5078
	0.014	0.7405	2	0.5080
	0.014	0.7400	2	0.5082
0.8	0.015	1.0396	2	0.6553
	0.015	1.0396	2	0.6555
	0.015	1.0388	2	0.6558
1.0	0.017	1.3608	1	0.7967
	0.017	1.3609	2	0.7971
	0.017	1.3598	2	0.7974

Table-6.1-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.4**, the middle to **3.6**, and the lower to **3.8**. In **3.6** and **3.8**, $\Delta t_j^{(k)}$ is calculated by **(3.46)**. $\Delta x=0.0125$

s(t)	Δt	t	k	u(0,t)
	0.015	0.2169	2	0.1859
0.2	0.015	0.2169	2	0.1860
	0.015	0.2168	2	0.1860
	0.016	0.4651	2	0.3527
0.4	0.016	0.4651	2	0.3528
	0.016	0.4647	2	0.3529
	0.018	0.7400	2	0.5081
0.6	0.018	0.7399	2	0.5082
	0.018	0.7392	2	0.5085
	0.019	1.0389	2	0.6556
0.8	0.019	1.0388	2	0.6558
	0.019	1.0378	2	0.6561
	0.021	1.3600	2	0.7971
1.0	0.021	1.3600	2	0.7974
	0.021	1.3586	2	0.7977

Table-6.2-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.4**, the middle to **3.6** and the lower to **3.8.** In **3.6** and **3.8**, $\Delta t_j^{(k)}$ is calculated by **(3.46)**. $\Delta x=0.025$.

s(t)	Δ t	t	k	u(0,t)
	0.029	0.2158	2	0.1867
0.2	0.029	0.2158	2	0.1867
	0.029	0.2155	2	0.1869
	0.032	0.4631	2	0.3537
0.4	0.032	0.4630	2	0.3538
	0.032	0.4621	2	0.3542
	0.036	0.7371	2	0.5093
0.6	0.036	0.7369	2	0.5095
	0.036	0.7355	2	0.5100
	0.039	1.0351	2	0.6569
0.8	0.039	1.0349	2	0.6572
	0.038	1.0328	2	0.6578
	0.041	1.3555	2	0.7984
1.0	0.041	1.3552	2	0.7989
	0.041	1.3525	2	0.7996

Table-6.3-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.4** the middle to **3.6** and the lower to **3.8. In 3.6** and **3.8**, $\Delta t_j^{(k)}$ is calculated by **(3.46)**. $\Delta x=0.05$.

s(t)	Δ t	t	k	u(0,t)
	0.057	0.2136	2	0.1882
0.2	0.057	0.2136	3	0.1882
	0.056	0.2132	3	0.1885
	0.064	0.4590	2	0.3559
0.4	0.064	0.4590	3	0.3559
	0.064	0.4577	3	0.3566
	0.070	0.7312	2	0.5117
0.6	0.070	0.7314	3	0.5119
	0.070	0.7290	3	0.5128
	0.076	1.0278	3	0.6596
0.8	0.076	1.0280	3	0.6598
	0.076	1.0245	3	0.6609
	0.082	1.3466	3	0.8015
1.0	0.082	1.3469	3	0.8016
	0.081	1.3422	3	0.8029

Table-6.4-

Comparison of time step Δt, t time, radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.4** the middle to **3.6** and the lower to **3.8. In 3.6** and **3.8**, $\Delta t_j^{(k)}$ is calculated by **(3.46)**. $\Delta x=0.1$

s(t)	Δt	t	k	u(0,t)
	0.109	0.2092	2	0.1916
0.2	0.109	0.2092	3	0.1916
	0.109	0.2086	3	0.1921
	0.124	0.4508	2	0.3604
0.4	0.125	0.4509	3	0.3604
	0.123	0.4483	4	0.3619
	0.138	0.7197	3	0.5171
0.6	0.138	0.7199	3	0.5170
	0.136	0.7149	4	0.5192
	0.150	1.0129	3	0.6653
0.8	0.150	1.0134	3	0.6653
	0.148	1.0057	4	0.6680
	0.161	1.3286	3	0.8074
1.0	0.161	1.3294	3	0.8073
	0.159	1.3190	4	0.8105

Table-6.5-

Comparison of time stepΔt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and Δt_j is calculated by (3.47), $\Delta x=0.01$

s(t)	Δt	t	k	u(0,t)
0.2	0.012	0.2164	2	0.1858
	0.012	0.2163	2	0.1858
0.4	0.013	0.4641	2	0.3524
	0.013	0.4638	2	0.3526
0.6	0.014	0.7387	2	0.5078
	0.014	0.7382	2	0.5080
0.8	0.015	1.0374	2	0.6553
	0.015	1.0366	2	0.6556
1.0	0.017	1.3583	2	0.7969
	0.017	1.3573	2	0.7972

Table-6.6-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and $\Delta t_j^{(k)}$ is calculated by (3.47), $\Delta x=0.1$

s(t)	Δ t	t	k	u(0,t)
0.2	0.105	0.2046	2	0.1913
	0.104	0.2044	2	0.1916
0.4	0.121	0.4394	3	0.3596
	0.120	0.4377	3	0.3610
0.6	0.135	0.7030	3	0.5158
	0.134	0.6994	3	0.5178
0.8	0.147	0.9919	3	0.6638
	0.146	0.9861	3	0.6663
1.0	0.159	1.3038	3	0.8057
	0.158	1.2958	3	0.8085

Table-6.7-

Comparison of time step Δt, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and Δt_j is calculated by (3.48), $\Delta x=0.01$

s(t)	Δt	t	k	u(0,t)
0.2	0.012	0.2180	2	0.1859
	0.012	0.2179	2	0.1859
0.4	0.013	0.4672	2	0.3527
	0.013	0.4669	2	0.3528
0.6	0.014	0.7430	2	0.5082
	0.014	0.7424	2	0.5084
0.8	0.016	1.0428	2	0.6558
	0.016	1.0420	2	0.6560
1.0	0.017	1.3648	2	0.7974
	0.017	1.3637	2	0.7977

Table-6.8-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and Δt_j is calculated by (3.48), $\Delta x=0.1$

s(t)	Δ t	t	k	u(0,t)
0.2	0.114 0.113	0.2136 0.2127	3 3	0.1919 0.1926
0.4	0.129 0.127	0.4635 0.4605	3 4	0.3615 0.3630
0.6	0.141 0.140	0.7402 0.7346	3 4	0.5188 0.5210
0.8	0.153 0.152	1.0408 1.0325	3 4	0.6677 0.6704
1.0	0.164 0.163	1.3637 1.3526	3 4	0.8104 0.8134

Table-6.9-

Comparison of time stepΔt_j, time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and $\Delta t_j^{(k)}$ is calculated by (3.49), $\Delta x=0.01$

s(t)	Δ t	t	k	u(0,t)
0.2	0.012 0.012	0.2173 0.2172	2 2	0.1858 0.1859
0.4	0.013 0.013	0.4658 0.4655	2 2	0.3526 0.3527
0.6	0.014 0.014	0.7412 0.7406	2 2	0.5080 0.5082
0.8	0.016 0.015	1.0406 1.0398	2 2	0.6556 0.6558
1.0	0.017 0.017	1.3622 1.3612	2 2	0.7972 0.7975

Table-6.10-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t). The upper entry corresponds to method **3.6** and the lower to **3.8** and Δt_j is calculated by (3.49), $\Delta x=0.1$

s(t)	Δ t	t	k	u(0,t)
0.2	0.109	0.2092	2	0.1916
	0.109	0.2089	3	0.1922
0.4	0.125	0.4523	3	0.3607
	0.124	0.4503	3	0.3621
0.6	0.139	0.7236	3	0.5177
	0.138	0.7194	3	0.5196
0.8	0.151	1.0197	3	0.6663
	0.150	1.0132	3	0.6687
1.0	0.162	1.3385	3	0.8088
	0.161	1.3296	3	0.8115

Table-6.11-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.1 in solving cylindrical problem. The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49),$\Delta x=0.01$

s(t)	Δ t		k	u(0,t)
0.2	0.013	0.2354	2	0.1731
	0.013	0.2338	2	0.1731
	0.014	0.2363	2	0.1731
	0.013	0.2347	2	0.1731
0.4	0.017	0.5398	2	0.3118
	0.017	0.5368	2	0.3117
	0.017	0.5416	2	0.3119
	0.017	0.5385	2	0.3118
0.6	0.020	0.9089	2	0.4306
	0.020	0.9045	2	0.4305
	0.020	0.9114	2	0.4307
	0.020	0.9071	2	0.4305
0.8	0.023	1.3405	2	0.5355
	0.023	1.3349	2	0.5353
	0.023	1.3439	2	0.5355
	0.023	1.3383	2	0.5354
1.0	0.026	1.8334	2	0.6298
	0.026	1.8267	2	0.6296
	0.026	1.8376	2	0.6298
	0.026	1.8308	2	0.6297

Table-6.12-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.1 in solving cylindrical problem. The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.1$

s(t)	Δ t	t	k	u(0,t)
0.2	0.118	0.2184	3	0.1845
	0.109	0.2087	2	0.1839
	0.123	0.2229	3	0.1847
	0.113	0.2134	3	0.1842
0.4	0.152	0.5055	3	0.3286
	0.144	0.4802	3	0.3275
	0.156	0.5185	3	0.3292
	0.1.48	0.4934	3	0.3281
0.6	0.183	0.8562	4	0.4516
	0.176	0.8171	3	0.4502
	0.187	0.8774	3	0.4523
	0.180	0.8385	3	0.4510
0.8	0.213	1.2680	4	0.5602
	0.207	1.2162	3	0.5586
	0.217	1.2973	3	0.5610
	0.211	1.2455	3	0.5595
1.0	0.243	1.7396	3	0.6577
	0.237	1.6758	3	0.6561
	0.247	1.7769	3	0.6587
	0.241	1.7127	3	0.6572

Table-6.13-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.2 in solving cylindrical problem. The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.01$

s(t)	Δ t	t	k	u(0,t)
0.2	0.010	0.2010	1	0.1992
	0.010	0.2009	1	0.1992
	0.010	0.2019	1	0.1993
	0.010	0.2018	1	0.1992
0.4	0.010	0.4065	1	0.3958
	0.010	0.4063	1	0.3957
	0.010	0.4081	1	0.3960
	0.010	0.4079	1	0.3959
0.6	0.011	0.6182	1	0.5900
	0.011	0.6179	1	0.5900
	0.011	0.6205	1	0.5904
	0.011	0.6202	1	0.5904
0.8	0.011	0.8369	1	0.7827
	0.011	0.8364	1	0.7826
	0.011	0.8399	1	0.7833
	0.011	0.8395	1	0.7832
1.0	0.011	1.0629	1	0.9744
	0.011	1.0624	1	0.9742
	0.012	1.0665	1	0.9751
	0.012	1.0660	1	0.9750

Table-6.14-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.2 in solving cylindrical problem. The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.1$

s(t)	Δ t	t	k	u(0,t)
0.2	0.101	0.2008	2	0.1992
	0.100	0.2004	1	0.1991
	0.105	0.2052	3	0.1996
	0.105	0.2050	2	0.1996
0.4	0.103	0.4061	2	0.3950
	0.103	0.4047	2	0.3947
	0.107	0.4184	2	0.3967
	0.107	0.4174	2	0.3964
0.6	0.107	0.6179	3	0.5874
	0.106	0.6151	2	0.5871
	0.110	0.6372	3	0.5910
	0.110	0.6350	2	0.5906
0.8	0.110	0.8367	3	0.7779
	0.110	0.8324	2	0.7774
	0.114	0.8625	3	0.7832
	0.113	0.8589	2	0.7827
1.0	0.114	1.0629	3	0.9671
	0.113	1.0571	2	0.9664
	0.117	1.0948	3	0.9741
	0.116	1.0898	2	0.9735

Table-6.15-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.3 in solving cylindrical problem(outside cylinder). The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.01$

s(t)	Δ t	t	k	u(0,t)
0.2	0.013	0.2362	2	0.1732
	0.013	0.2345	2	0.1732
	0.014	0.2371	2	0.1732
	0.013	0.2354	2	0.1732
0.4	0.017	0.5412	2	0.3120
	0.017	0.5382	2	0.3119
	0.017	0.5430	2	0.3121
	0.017	0.5399	2	0.3120
0.6	0.020	0.9109	2	0.4309
	0.020	0.9066	2	0.4307
	0.020	0.9135	2	0.4310
	0.020	0.9091	2	0.4308
0.8	0.023	1.3431	2	0.5358
	0.023	1.3376	2	0.5356
	0.023	1.3465	2	0.5359
	0.023	1.3409	2	0.5357
1.0	0.026	1.8367	2	0.6301
	0.026	1.8299	2	0.6299
	0.026	1.8408	2	0.6302
	0.026	1.8341	2	0.6300

Table-6.16-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.3 in solving cylindrical problem(outside cylinder). The upper entry corresponds to Δt_j calculated by (3.46, the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.1$

s(t)	Δt	t	k	u(0,t)
0.2	0.125	0.2252	4	0.1799
	0.112	0.2120	3	0.1786
	0.129	0.2295	4	0.1802
	0.117	0.2165	3	0.1791
0.4	0.154	0.5174	4	0.3283
	0.147	0.4910	3	0.3267
	0.158	0.5301	4	0.3288
	0.151	0.5039	3	0.3274
0.6	0.186	0.8744	4	0.4531
	0.180	0.8346	3	0.4518
	0.190	0.8952	4	0.4539
	0.184	0.8556	3	0.4526
0.8	0.216	1.2920	4	0.5624
	0.210	1.2397	3	0.5609
	0.220	1.3207	4	0.5633
	0.214	1.2685	3	0.5618
1.0	0.246	1.7695	4	0.6605
	0.240	1.7054	3	0.6589
	0.250	1.8060	4	0.6614
	0.244	1.7418	3	0.6599

Table-6.17-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.3 in solving cylindrical problem(inside cylinder). The upper entry corresponds to Δt_j calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.01$

s(t)	Δt	t	k	u(0,t)
0.2	0.010	0.1978	1	0.2009
	0.010	0.1979	1	0.2009
	0.010	0.1987	1	0.2010
	0.010	0.1988	1	0.2010
0.4	0.009	0.3888	1	0.4051
	0.009	0.3891	1	0.4051
	0.009	0.3904	1	0.4053
	0.009	0.3907	1	0.4054
0.6	0.008	0.5661	2	0.6104
	0.008	0.5667	2	0.6107
	0.008	0.5684	2	0.6110
	0.008	0.5690	2	0.6113
0.8	0.007	0.7196	2	0.8061
	0.007	0.7207	2	0.8070
	0.007	0.7225	2	0.8071
	0.007	0.7236	2	0.8081
1.0	0.002	0.8178	4	0.9451
	0.002	0.8205	3	0.9483
	0.002	0.8212	4	0.9467
	0.002	0.8240	3	0.9500

Table-6.18-

Comparison of time step Δt_j, t time, r radius, k iteration number and the surface temperature u(0,t) using the method **3.6** and the techniques described in 3.11.3 in solving cylindrical problem(inside cylinder). The upper entry corresponds to Δt_j, calculated by (3.46), the second by (3.47), the third by (3.48) and the lower by (3.49), $\Delta x=0.1$

s(t)	Δt	t	k	u(0,t)
0.2	0.092	0.1917	3	0.2090
	0.095	0.1954	2	0.2096
	0.096	0.1958	3	0.2099
	0.100	0.1997	1	0.2107
0.4	0.091	0.3764	4	0.4106
	0.092	0.3806	2	0.4126
	0.094	0.3879	4	0.4133
	0.096	0.3926	2	0.4155
0.6	0.083	0.5470	5	0.6151
	0.084	0.5533	3	0.6175
	0.086	0.5651	4	0.6200
	0.088	0.5719	3	0.6230
0.8	0.071	0.6953	6	0.8120
	0.073	0.7055	4	0.8182
	0.073	0.7189	5	0.8219
	0.075	0.7298	3	0.8284
1.0	0.045	0.8011	8	0.9669
	0.049	0.8180	5	0.9834
	0.047	0.8293	7	0.9831
	0.052	0.8477	4	0.9989

Table-6.19-

Comparison of boundary velocity ¥, t time, N iteration number and the surface temperature c(0,s)

s(t)	¥	t	N	c(0,s)
0.00	1.00000	0.00000	6	0.00000
0.20	0.85547	0.21728	6	0.18587
0.30	0.80459	0.33792	6	0.27102
0.40	0.76247	0.46568	6	0.35270
0.50	0.72666	0.60009	6	0.43161
0.60	0.69563	0.74080	6	0.50821
0.70	0.66832	0.88752	6	0.58286
1.00	0.60233	1.36140	6	0.79747
1.20	0.56765	1.70367	6	0.93436
1.37	0.54229	2.01021	5	1.04762
3.00	0.39588	5.59289	5	2.04103
4.00	0.34634	8.30208	5	2.59902
4.50	0.32702	9.78863	5	2.86848
4.80	0.31670	10.72100	5	3.02762
5.00	0.31026	11.35908	5	3.13276

r=k=0.5 and h=0.01

Table-6.20-

Comparison of boundary velocity ¥, t time, N iteration number and the surface temperature c(0,s).

s(t)	¥	t	N	c(0,s)
0.20	0.8583	0.2168	5	0.1865
0.26	0.8262	0.2881	6	0.2382
0.34	0.7890	0.3872	5	0.3050
0.40	0.7644	0.4644	5	0.3537
0.80	0.6452	1.0375	5	0.6572
2.00	0.4707	3.2572	5	1.4502
3.00	0.3962	5.5854	5	2.0432
4.00	0.3466	8.2927	5	2.6013
5.00	0.3104	11.3479	5	3.1352
6.00	0.2826	14.7291	5	3.6507
6.40	0.2732	16.1691	5	3.8527
7.00	0.2604	18.4197	5	4.1417
8.00	0.2421	22.4065	5	4.6408
10.00	0.2135	31.2271	5	5.5900

r=k=0.5 and h=0.02

Table-6.21-

6.3. Numerical Results for Oxygen Diffusion Problem

The values of $10^6 c$ and the position of the moving boundary using the simplest explicit formula

$\delta t=0.001$ and $\delta x=0.1$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	391612	244916	125000	45000	5000	1.00000
0.010	369057	243388	124980	45000	5000	1.00000
0.015	348136	239577	124785	44996	5000	1.00000
0.025	312219	227335	123070	44880	4996	0.99996
0.045	254960	196340	113442	43024	4779	0.99770
0.060	219368	172360	102444	39543	4123	0.99080
0.100	141031	112108	066643	23345	0546	0.93304
0.130	091927	071106	038379	08763	0000	0.84083
0.150	062278	045694	020360	01049	0000	0.74581
0.160	048230	033595	012036	00000	0000	0.68089
0.180	021597	011010	000000	00000	0000	0.49257
0.190	009082	001353	000000	00000	0000	0.35201
0.195	003138	000000	000000	00000	0000	0.17922
0.196	001809	000000	000000	00000	0000	0.16015

Table-6.22-

The values of $10^6 c$ and the position of the moving boundary using the simplest explicit formula

$\delta t=0.001$ and $\delta x=0.005$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	389128	245000	125000	45000	5000	1.00000
0.010	365668	243726	125000	45000	5000	1.00000
0.015	344810	239672	124891	45000	5000	1.00000
0.025	309328	226854	123264	44931	4999	1.00000
0.045	252653	195347	113417	43159	4823	1.00000
0.060	217330	171251	102227	39645	4186	0.99220
0.100	139414	110966	066112	23232	0619	0.93518
0.130	090492	069995	037728	08541	0000	0.84189
0.150	060928	044602	019668	01007	0000	0.74488
0.160	046912	032511	011346	00000	0000	0.68128
0.180	020328	009950	000000	00000	0000	0.49607
0.190	007827	000750	000000	00000	0000	0.33873
0.195	001909	000000	000000	00000	0000	0.16128
0.196	000000	000000	000000	00000	0000	0.10287

Table-6.23-

The values of $10^6 c$ and the position of the moving boundary using The Du Fort-Frankel Scheme

$\delta t=0.001$ and $\delta x=0.1$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	387269	244766	125000	45000	5000	1.00000
0.010	364732	242620	124943	44999	5000	1.00000
0.020	325779	232458	123881	44941	4998	0.99998
0.030	293454	218218	120766	44553	4969	0.99969
0.040	265352	202550	115673	43520	4841	0.99840
0.060	217187	170650	101397	39065	3996	0.98940
0.100	139339	110653	065792	00000	0000	0.89640
0.130	090560	070345	000000	00000	0000	0.67509
0.150	059415	000000	000000	00000	0000	0.44472
0.160	043921	000000	000000	00000	0000	0.39638
0.180	016485	000000	000000	00000	0000	0.28158
0.180	011721	000000	000000	00000	0000	0.28158

Table-6.24-

The values of $10^6 c$ and the position of the moving boundary using the Du Fort-Frankel Scheme

$\delta t=0.001$ and $\delta x=0.05$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	387917	245000	125000	45000	5000	1.00000
0.010	365754	243730	125000	45000	5000	1.00000
0.015	344315	239288	124848	45000	5000	1.00000
0.020	326167	233766	124350	44989	5000	1.00000
0.025	309018	226497	123130	00000	0000	0.99962
0.030	293670	219270	000000	00000	0000	0.98458
0.035	279000	000000	000000	00000	0000	0.88484
0.038	000000	000000	000000	00000	0000	0.79777

Table-6.25-

Comparison between analytical (5.15) and numerical solutions using the simplest explicit formula for small times. The upper entry corresponds to the analytical solution and the lower entry to the numerical solutions. Values of $10^6 c$ and $\delta x=0.05$.

t \ x	0.0	0.2	0.4	0.6	0.8
0.001	464317 460000	320000 320000	180000 180000	080000 080000	020000 020000
0.002	449537 452000	319973 320000	180000 180000	080000 080000	020000 020000
0.003	438196 437600	319760 320000	180000 180000	080000 080000	020000 020000
0.004	428635 429600	319212 320000	180000 180000	080000 080000	020000 020000
0.005	420211 420320	318302 318976	179999 180000	080000 080000	020000 020000
0.010	387162 387497	309949 310719	179804 179927	079999 080000	020000 020000
0.015	361802 362071	298690 299304	178766 179040	079977 079994	020000 020000
0.020	340423 340661	286674 287180	176604 176960	079847 079905	019997 019999
0.040	274324 274497	240143 240440	159898 160269	076551 076767	019607 019672
0.045	260635 260798	229328 229598	154563 154917	074905 075140	019377 019377
0.050	247687 247842	218841 219090	148990 149327	072960 073208	018834 018955

Table-6.26-

The values of $10^6 c$ and the position of the moving boundary for inside cylindrical problem

$\delta t=0.001$ and $\delta x=0.1$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	188965	116893	058499	019410	000399	0.07176
0.010	172151	110761	054487	016398	000000	0.10196
0.015	156537	103834	050423	013533	000000	0.12371
0.025	129101	088634	041966	008604	000000	0.16425
0.030	116860	080831	037592	006532	000000	0.18388
0.035	105393	073081	033174	004576	000000	0.20434
0.040	094593	065457	028750	002798	000000	0.22520
0.045	084378	058007	024350	001388	000000	0.24731
0.050	074685	050759	020034	000394	000000	0.27729
0.060	056675	036942	011819	000000	000000	0.32937
0.070	040266	024103	004445	000000	000000	0.40571
0.080	025266	012369	000000	000000	000000	0.50460
0.095	005143	000000	000000	000000	000000	0.74950
0.097	002166	000000	000000	000000	000000	0.83418

Table-6.27-

The values of $10^6 c$ and the position of the moving boundary for inside cylindrical problem

$\delta t=0.001$ and $\delta x=0.05$

t \ x	0.1	0.3	0.5	0.7	0.9	Moving boundary
0.005	183472	114633	057492	019100	000503	0.06828
0.010	162652	106403	052518	015758	000000	0.09876
0.015	144169	097367	047548	012638	000000	0.12390
0.025	112728	078666	037416	007251	000000	0.17126
0.030	099105	069536	032299	004989	000000	0.19551
0.035	086592	060721	027227	003011	000000	0.21977
0.040	075041	052271	022261	001396	000000	0.24715
0.045	064334	044215	017461	000284	000000	0.27618
0.050	054381	036561	012883	000000	000000	0.30960
0.060	036446	022476	004746	000000	000000	0.39234
0.070	020781	010075	000000	000000	000000	0.50390
0.080	007097	000000	000000	000000	000000	0.68798
0.083	003371	000000	000000	000000	000000	0.78603
0.085	000000	000000	000000	000000	000000	0.89570

Table-6.28-

Comparison between analytical and numerical solutions inside cylindrical problem for small times. The upper entry correspondons to the analytical solution and the lower entry to the numerical solutions. Values of 10^6c and $\delta x=0.05$.

t \ x	0.0	0.2	0.4	0.6	0.8
0.001	244333 247950	160000 159799	090000 089875	040000 039918	010000 009941
0.002	241971 246041	160000 159599	090000 089749	040000 039837	010000 009883
0.003	240152 244256	160000 159399	090000 089624	040000 039755	010000 009824
0.004	238615 242581	160000 159198	090000 089498	040000 039673	010000 009765
0.005	237258 241004	160000 158998	090000 089373	040000 039592	010000 009707
0.010	231904 234238	160000 157997	090000 088747	040000 039184	010000 009413
0.015	227766 228744	159998 156993	090000 088121	040000 038776	010000 009120
0.020	224256 224023	159985 155979	090000 087496	040000 038368	010000 008827
0.040	213280 208949	159552 151697	090000 085001	040000 036739	010000 007679
0.045	210983 205717	159318 150557	090000 084378	040000 036332	010000 007400
0.050	208803 202623	159033 149390	089999 083756	040000 035925	010000 007127

Table-6.29-

The values of $10^6 c$ and the position of the moving boundary for outside cylindrical problem

$\delta t=0.001$ and $\delta x=0.1$

t \ x	1.1	1.3	1.5	1.7	1.9	Moving boundary
0.005	471557	282352	137285	047410	004823	1.99822
0.010	434437	275990	135720	047008	004748	1.99744
0.020	373069	255949	131267	045902	004545	1.99534
0.030	323192	231920	124215	044192	004244	1.99213
0.040	280962	207395	114948	041602	003785	1.98701
0.060	211857	161118	092749	033707	002254	1.96714
0.100	111109	084151	046006	012106	000000	1.86426
0.130	055685	038639	015154	000000	000000	1.70042
0.150	025339	013452	000000	000000	000000	1.50876
0.160	011865	002493	000000	000000	000000	1.37061
0.166	004149	000000	000000	000000	000000	1.19109
0.167	002522	000000	000000	000000	000000	1.17102

Table-6.30-

The values of $10^6 c$ and the position of the moving boundary for outside cylindrical problem

$\delta t=0.001$ and $\delta x=0.05$

t \ x	1.1	1.3	1.5	1.7	1.9	Moving boundary
0.005	458655	274706	134108	046507	004982	1.99909
0.010	412439	263751	130166	045303	004797	1.99719
0.020	337164	234323	121139	042421	004232	1.99163
0.030	278037	202458	109634	038814	003471	1.98327
0.040	229662	172028	096329	034289	002545	1.97152
0.060	154565	118797	068140	022948	000480	1.93098
0.080	098936	075615	041432	010578	000000	1.85896
0.100	056408	040872	018170	000827	000000	1.74066
0.120	023287	013161	000882	000000	000000	1.54201
0.130	009524	002131	000000	000000	000000	1.36528
0.135	003293	000000	000000	000000	000000	1.21389
0.136	001933	000000	000000	000000	000000	1.16218

Table-6.31-

Comparison between analytical and numerical solutions outside cylindrical problem for small times. The upper entry corresponds to the analytical solution and the lower entry to the numerical solutions. Values of 10^6c and $\delta x=0.05$.

t \ x	1.0	1.2	1.4	1.6	1.8
0.001	583507 577730	381651 378896	203350 202144	086287 085879	020721 020657
0.002	562063 562222	381614 376137	203350 200931	086287 085463	020720 020586
0.003	545771 538795	381326 373373	203350 199711	086287 085039	020715 020510
0.004	532144 523928	380586 370605	203349 198483	086287 084608	020697 020425
0.005	520219 507742	379363 366595	203348 197248	086287 084170	020660 020334
0.010	474140 447050	368248 343224	203106 190888	086277 081870	020122 019795
0.015	439528 399404	353467 317353	201822 183507	086185 079391	019042 019130
0.020	410833 359475	337887 292060	199164 174854	085852 076670	017539 018353
0.040	324780 242033	279016 205337	178729 134174	079816 062196	008489 014106
0.045	307433 219463	265605 187192	172186 123788	077011 057792	005552 012728
0.050	291164 198802	252678 170266	165330 113602	073745 053193	002357 011233

Table-6.32-

References

[01] Albrecht, J., Collatz, L., and Hoffman, K.H.(eds)(1980). Proceedings of The Oberwolfach Conference on Free Boundary Problems.

[02] Ames, W.F.(1977). Numerical Methods for Partial Differential Equations, Academic Press, Thomas Nelson & Sons.

[03] Atkinson, L.V. & Harley, P.J.(1983). An Introduction to Numerical Methods With Pascal, international Computer series.

[04] Atthey, D.R.(1974). J.Inst.Math.Appl.**13**, 353-66.

[05] Bankff, S.G.(1964). Adv.Chem.Engng **5**, 75-150.

[06] Berger A.E., Ciment M. & Rogers J.C.W., Numerical solution of a diffusion consumption problem with a free boundary, SIAM, J.Num.Anal.12,646-672 (1975).

[07] Bickely,W.G. (1953). Bessel Functions and Formulae. Published for the Royal Society at the University Press Cambridge.

[08] Boley, B.A.(1972). Nucl.Engng.Des. **18**, 377-99.

[09] Carnahan, B., Luther, H.A. & Wilkes, J.O.(1969). Applied Numerical Methods, John Wiley & Sons.

[10] Crank, J.(1957). Q.Jl.Mech.Math **10**, 220-231.

[11] _____.(1975a). The Mathematics of Diffusion. Clarendon Press, Oxford.

[12] _____.(1975b).In Ockendon & Hoginkins(1975).pp192-207

[13] _____.(1981). In Numerical Methods in Heat Transfer.(ed.R.W.Lewis, K.Morgan & O.C.Zienkiewicz, Wiley New York.

[14] _____.(1984). Free & Moving Boundary Problems, Clarendon Press, Oxford.

[15] _____ & Gupta, R.S.(1972).J.Inst.Math.Appl.**10**, 19-33.

[16] Douglas, J. & Gallie, J.M.(1955). Duke.Math.J.**22**,557-71.

[17] Fasano, A. & Primicerio, M.(eds)(1983). Free Boundary Problems Theory and Applications. Vols I & II Pitam.London

[18] Fox, L.(1975). In Ockendon & Hodgkins.(1975).pp.210-41.

[19] _____.(1979). In Gladwell, I. & Wait.(1979).

[20] Furzeland, R.M.(1977a).Brunel University.Math.Report TR/76

[21] _____.(1980). J.Inst.Math.Appl. **26**, 411-29.

[22] Galib, J.A. Burch, J.C. & Sloss,J.M.(1981) Int.J.Bio-Med.Comput **12**, 157-80.

[23] Gladwell, I. & Wait, R.(eds)(1979). A Survey of Numerical methods for Partial Differential Equations. Clarendon Press Oxford.

[24] Goodling, J.S. & Khader, M.S.(1974).A.F.S. Cast Metals. Res.J.(Mar 1974). 26-29.

[25] Gottlieb, D. & Gustafsson, B.(1976). Siam.Numer.Anal.**13**, Nº1,129-44.

[26] Gray, A. & Macrobert, T.M.(1952). Bessel Functions and their Applications to Physics. Second Edition.

[27] Gupta, R.S., Moving Grid Methods Without Interpolations. Computer.Meths.Appl.Mech.Eng **4**,(1974). 143-152.

[28] _____ & Kumar, D.(1980). Computer.Meths.Appl.Mech. Eng. 23, 101-109.

[29] Hoffman, K.H.(ed)(1977). Frei Universitat (Berlin) Fachebereich Mathematik Volumes (I-III).

[30] Lapidus, L. & Pinder, G.F (1982).Numerical Solution of Partial Differential Equations in Science and Engineering, John Wiley & Sons.

[31] Magenes, E.(eds)(1980). Free Boundary Problems Vols I & II Proceeding of Seminar in Pavia.(1979). Institute Nationale di. Alta Matematica Francesco Severi, Rome.

[32] Meyer, G.H.(1975). Inst.J.Numer.Meth.Engng. **9**, 669-78.

[33] _____.(1976). Brunel University Math Report TR/62.

[34] Morland, L.W.(1982). J.Engg.Math, **16**,259-269.

[35] Mitchell, A.R. & Griffiths, D.F.(1980). The Finite Difference Method in Partial Differential Equations , John Wiley & Sons.

[36] Muehlbauer, J.C. & Sunderland, J.E.(1965).Appl.Mech.Rev. **8**, pp 951-9.

[37] Murray, W.D. & Landis, F.(1959).J.Heat.Transfer, Trans.ASME(c) 8 106-12.

[38] N.B.S. Appl.Math.Serie 55 (1964). Hand book of

Math functions with Formulas, Graphs, & Math Tables

[39] Richtmyer, R.D.(1967).Difference Methods for Initial Value Problems John Wiley & Sons.

[40] Rubinstein,L.I.(1971). Trans. Math. Monographs 27.Am.Math. Soc.Providence, R.I.

[41] Smith, G.D.(1978).Numerical Solution of Partial Differential Equations, Second Edition, Oxford University Press.

[42] Watson (1944) Theory of Bessel Functions., Second Edition Cambridge at the University Press.

[43] Wilson, D.G., Slomon, A.D., & Boggs, P.T.(eda)(1978).Moving Boundary Problems, Academic Press, inc.New York.

Appendix-A-

Thomas algorithm

Note that equation (3.11) is tridiagonal for all $r \neq 0$, using (3.11) and (3.12) in section 3.7 as model, let us write the general tridiagonal system of j+1=M equations in the form :

$$
\begin{aligned}
d_1 u_1 + a_1 u_2 &= c_1 \\
b_2 u_1 + d_2 u_2 + a_2 u_3 &= c_2 \\
\bullet \quad \bullet \quad \bullet &= \bullet \\
\bullet \quad \bullet \quad \bullet &= \bullet \\
\bullet \quad \bullet \quad \bullet &= \bullet \\
\bullet \quad \bullet \quad \bullet &= \bullet \\
\bullet \quad \bullet \quad \bullet &= \bullet \\
b_i u_{i-1} + d_i u_i + a_i u_{i+1} &= c_i \\
b_{M-1} u_{M-2} + d_{M-1} u_{M-1} &= c_{M-1}
\end{aligned}
$$

Where a's, b's, c's, and d's are known u_M are known from the boundary conditions. The name tridiagonal system arises from the fact that the matrix is tridiagonal.

As we shall see, this system can be solved explicitly for the unknowns, thereby eliminating any matrix operations. The method described here was discovered by many and has been

called Thomas algorithm by young (see Ames 1977).

The above system is readily solved by a Gaussian elimination method, with a maximum of three variables per equation, the solution can be expressed very concisely. The first equation of the system can be used to eliminate u_1 from the second equation, the new second equation used to eliminate u_2 from the third equation and so on, until finally the new last but one equation can be used to eliminate u_{M-2} from the last equation giving one equation with only one unknown u_{M-1}.

The unknowns $u_{M-2}, u_{M-3}, \ldots\ldots\ldots, u_3, u_2, u_1$ can then be found in turn back substitution. Noting that the coefficient " a " in each new equation is the same as in the corresponding old equation, assume that the following stage of the elimination has been reached

$$ß_{i-1}u_{i-1} + a_{i-1}u_i = ¥_{i-1}$$

$$b_iu_{i-1} + d_iu_i + a_iu_{i+1} = c_i$$

where $ß_1 = d_1$, $¥_1 = c_1$

Elimination of u_{i-1} leads to

$$(d_i - b_ia_{i-1}/ß_{i-1})u_i + a_iu_{i+1} = c_i - b_i¥_{i-1}/ß_{i-1}$$

$$\text{i.e } ß_iu_i + a_iu_{i+1} = ¥_i \qquad (1)$$

where $ß_i = (d_i - b_ia_i/ß_{i-1})$ and $¥_i = c_i - b_i¥_{i-1}/ß_{i-1}$ (i=2, 3, . . . M-1).

The last pair of simultaneous equations are

$$ß_{M-2}u_{M-2} + a_{M-2}u_{M-1} = ¥_{M-2}$$

and

$$b_{M-1}u_{M-2} + d_{M-1}u_{M-1} = c_{M-1}$$

Elimination of u_{M-2} gives

$$(d_{M-1} - b_{M-1}a_{M-2}/ß_{M-2})u_{M-1} = c_{M-1} - b_{M-1}¥_{M-2}/ß_{M-2}$$

$$ß_{M-1} = (d_{M-1} - b_{M-1}a_{M-2}/ß_{M-2}) = ¥_{M-1}$$

$$\text{i.e } ß_{M-1}u_{M-1} = ¥_{M-1} \qquad (2)$$

Equation (1) and (2) show that the solution can be calculated from

$$u_{M-1} = ¥_{M-1}/ß_{M-1} \qquad (3)$$

$$u_i = (¥_i - a_i u_{i+1})/ß_i \qquad (i=M-2, M-3, M-4, \ldots\ldots .3, 2, 1\ )$$

where ß's and ¥'s are given recursively by

$$ß_1 = d_1, \quad ß_i = d_i - (b_i/ß_i)a_{i-1} \qquad (4.1)$$

$$¥_1 = c_1, \quad ¥_i = c_i - (b_i/ß_i)¥_{i-1} \qquad (i=2, 3, 4, \ldots\ldots M-1)\} \quad (4.2)$$

In many problems $ß_i$ and $b_i/b_i ß_{i-1}$ are independent of time and need only be calculated once, irrespective of the number of time-steps. If the coefficients are time dependent more calculation is required. The above method is highly efficient for automatic computation of tridiagonal systems.

While this method is equivalent to Gaussian elimination, it avoids the error growth associated with back substitution in the elimination method, and also minimises the storage in the machine computation. In fact, the solution of a problem by this method in conjunction with an implicit difference scheme is more efficient than the solution using an explicit difference scheme where the solution is calculated directly. (For more details and examples see Ames 1977, Smith 1978 and Mitchell 1980).

Appendix-B-

Stability Analysis

The matrix method for analyzising stability will automatically include the effects of the boundaries. To illustrate the procedure we utilize

$$(c_{i,j+1}-c_{i,j})/\delta t=(c_{i-1,j}-2c_{i,j}+c_{i+1,j})/(\delta x)^2-1 \qquad i=0,1,\ldots(r-2)$$

$$(c_{r-1,j+1}-c_{r-1,j})/\delta t=(2/(\delta x)^2)(c_{r-2,j}/(1+p_j)-c_{r-1,j}/p_j)-1$$

The two above formulas can be written in the following matrix form where $r=\delta t/(\delta x)^2$ and $c_{-1,j}=c_{1,j}$

$$\begin{bmatrix} c_{0,j+1} \\ c_{1,j+1} \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ c_{N-2,j+} \\ c_{N-1,j+} \end{bmatrix} = \begin{bmatrix} 1-2r & 2r & 0 & & & \\ r & 1-2r & r & 0 & & \\ & & * & & & \\ & & & * & & \\ & & & & * & \\ & & & & & * \\ & & 0 & r & 1-2r & r \\ & & & & 2r/(1+pj) & 1-2r/pj \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ c_{N-2,j} \\ c_{N-1,j} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ 1 \\ 1 \end{bmatrix} \delta t \qquad (1)$$

$$c_{j+1}=A_j c_j-U\delta t \qquad (2)$$

Where A_j is a square matrix of order N

U is column vector

In order to make analysis possible we replace p_j by a constant

value p. Then equation (2) is written as

$$c_{j+1}=Ac_j-u\delta t \qquad (3)$$

Let's denote the computed values by ψ so that we have actually solved the equations

$$\psi_{j+1}=A\psi_j+U\delta t \qquad (4)$$

the computational error is

$$c_{j+1}-\psi_{j+1}=A(c_j-\psi_j)$$

i.e $$e_{j+1}=Ae_j \qquad (5)$$

(The error introduced at the kth step is denoted by the vector e_k). The recurrence relation (5) gives

$$e_{j+1}=A_{j+1}e_0 \qquad (6)$$

e_0 is an error vector for starting values

Let us express e_0 as

$$e_0=\sum_{s=1}^{N} a_s v_s$$

v_s is an eigenvector of A corresponding to the eigenvalue λ_s and a's are constant

$$e_n=\sum_{s=1}^{N} a_s \lambda_s^{\,n} v_s$$

For e_n to tend to zero, as n increases, it follows that the largest of $|\lambda_1|, |\lambda_2|, \ldots\ldots |\lambda_N|$ must be less than unity.

The Q_s is the sum of the moduli of terms along the sth row exluding the diagonal term a_{ss} in matrix A then by Braurer's theorem every eigenvalue of a lies inside or on the boundary of at least one of circles $|\lambda-a_{ss}|=Q_s$.

As we are interested in the bounds of p, applying Braurer's theorem to the last row of A that contains p, we have

$$Q_s=2p/(1+p), \quad a_{ss}=1-2r/p$$

so that $$|\lambda-(1-2r/p)|\leq 2r/(1+p)$$

the bounds of λ are given by

$$\lambda_1=1-2r/p(1+p), \quad \lambda_2=2r(1+2p)/p(1+p) \quad -1$$

For stability we require $|\lambda_1|\leq 1$, $|\lambda_2|\leq 1$, and hence

$$-1 \leq 1-2r/p(1+p)\leq 1 \quad \text{giving} \quad r/p(1+p)\leq 1$$

Since p is always positive, the condition for stability is given by the second inequality because the first one is then satisfied automatically. Therefore for overall stability

$$p^2+(1-2r)p-r\geq 0$$

Since $r\leq 0.5$ for the stability of the simple explicit scheme used at the intermediate points, it can be shown that

$$p\geq r-0.5+\sqrt{(0.25+r^2)} \qquad (7)$$

For $\delta x=0.1$, $\delta t=0.001$ we get the stability condition $p\geq 0.11$ and for $\delta x=0.05$, $\delta t=0.001$ we have $p\geq 0.54$. This suggests that an instability may arise when the moving points is nearer than 0.11 to the neighbouring mesh point in the first case and 0.027 in the second case ($\delta x=0.05$). This confirms the need for the stability check described in moving boundary crossing a mesh line.

Appendix-C-

Programs

```
program Ice.M1 (input,output);
{Ice melting cartesian problem using method 3.9}
label 999;
type
subs=0. .300;
rows=record
a,b,c,d: real;
end;{row}
rowsvectors=array[subs] of rows;
matrix=array[subs,subs] of real;
vectors=array[subs] of real;
var
dt,h,r,m: real;
k,n,i,j: integer;
eqn: rowsvectors;
x,t: matrix;
s :vectors;
procedure TriDiag(var Ceqn : rowsvectors; var x: mtrix);
const
assumedzero=1E-20;
var
i:subs;
pivot,mult, ci,bi: real;
begin{TriDiag}
eqn:=Ceqn;
for i:=0 to n-1 do
begin
with eqn[i] do
begin
pivot:=d; ci:=c; bi:=b
end;
with eqn[i+1] do
begin
mult:=a/pivot;
if abs(mult)>assumedzero then
begin
a:=mult; d:=d-mult*ci; b:=b-mult*bi; c:=c
end
else
a:=0
```

```
end
end;
with eqn[n] do
x[n,j+1]:=b/d;
for i:=n-1 downto 0 do
with eqn[i] do
x[i,j+1]:=(b-c*x[i+1,j+1])/d
end;{TriDiag}
begin
write('value of dt,n');
read(dt,n);
while n>0 do
begin
h:=1/n;
x[0,0]:=0;x[1,1]:=0;x[0,1]:=h;t[0,0]:=h;j:=1;r:=1/h;s[0]:=h;t[1,0]=h;
write(r:5:2,' ':3);write(j:3,' ':2);write(t[0,0]:4:3,' ':3);
write(s[0]:6:4,' ':3);write(0:2,' ':3);write(x[0,1]:6:4);writeln;
for j:=1 to n-1 do
begin
x[j,j]:=0;x[j+1,j+1]:=0;
k:=1;
while k≤10 do
begin
with eqn[0] do
begin
a:=0; d:=-1; c:=1; b:=-h;
end;
for i:=1 to j do
begin
with eqn[i] do
begin
a:=-r; d:=(1+2*r); c:=-r; b:=x[i,j];
end;
end;
n:=j;
TriDiag(eqn,x);
t[j,k]:=(h*h)/x[j,j+1];
m:=t[j,k]-t[j,k-1];
if abs(m)≤dt then
begin
r:=t[j,k]/(h*h);
goto 999;
end
else
r:=t[j,k]/(h*h);
```

```
k:=k+1;
end;
999;
t[j+1,0]:=t[j,k];
s[j]:=s[j-1]+t[j,k]
write(r:5:2,' ':3);write((j+1):3,' ':2);write(t[j,k]:4:3,' ':3);
write(s[j]:6:4,' ':3);write(k:2,' ':3);write(x[0,j+1]:6:4);writeln;
end;
write('values of dt,n);
read(dt,n)
end
end.
```

```
program Ice.M2 (input,output);
{Ice melting cartesian problem using method described in
chapter-4-}.
label 999;
const
k:=0.5
r:=0.5;
m:=100;
h:=0.01;
type
subs=1..501;
rows=record;
a,b,cd: real;
end;{rows}
rowsvectors=array[subs] of rows;
matrix=array[subs,subs] of real;
vectors=array[subs] of real;
var
z: real;
e,n,i,j: integer;
eqn: rowsvectors;
x:matrix;
s,y,t :vectors;
procedure TriDiag{ as mentioned in program Ice.M1}
begin
y[1]:=1;y[2]:=1-h+2*h*h;y[3]:=1-2*h+8*h*h;
t[1]:=0;t[2]:=h+0.5*(h*h)-(h*h*h)/3; t[3]:=2*h+2*h*h-(8*h*h*h)/3;
x[1,1]:=0; x[1,2]:=h-0.5*h*h+(h*h*h); x[1,3]:=2*h-2*h*h+8*h*h*h;
x[2,3]:=h-((3*h*h)/2)+6*h*h*h; x[3,3]:=0; x[4,4]:=0; s[3]:=h*y[3];
x[4,3]:=-h*y[3]+0.5*h*h*y[3]*y[3];
with eqn[1] do
begin
```

```
a:=r; d:=1; c:=-1; b:=h;
end;
for i:=2 to 3 do
begin
with eqn[i] do
begin
a:=r; d:=-(2*r+s[3]); c:=r;
b:=-(1-r)*x[i-1,j]+(2*(1-r)-s[j])*x[i,j]-(1-r)*x[i+1,j];
end;
end;
z:=h*(1-2*h+8*h*h);
j:=3;
while j≤m do
begin
x[j,j]:=0; x[j+1,j+1]:=0;
x[j+1,j]:=-h*y[j]+0.5*h*h*y[j]*y[j];
while e≥1 do
begin
n:=j;
TriDiag(eqn,x);
y[j+1]:=(-1+sqrt(1+2*x[j,j+1]))/h;
if abs (y[j+1]-z)<h*h*abs(z) then
begin
write('j:=',j:=1,' ':4);
t[j+1]:=t[j-1]+(h/3)*((1/y[j+1])+(4/y[j])+(1/y[j-1]));
write('e:=',e:1,' ':1); write('t=',t[j+1]:6:4);
write(' ':3);write('x='x[1,j+1]:6:4,' ':6);write('y=',y[j+1]:6:4);writeln;
goto 999;
end else
z:=y[j+1]; s[j]:=h*k*y[j+1]+(1-k)*h*y[j];
with eqn[1] do
begin
a:=r; d:=1; c:=-1; b:=h;
end;
for i:=2 to j do
begin
with eqn[i] do
begin
a:=r;d;=-(2*r+s[j]);c:=r;
b:=-(1-r)*x[i-1,j]+(2*(1-r)-s[j])*x[i,j]-(1-r)*x[i+1,j];
end;
end;
end;
999:
```

```
j:=j+1;
s[j]:=h*k*y[j]+(1-k)*h*y[j-1];
with eqn[1] do
begin
a:=r; d:=1; c:=-1; b:=h;
end;
for i:=2 to j do
begin
with eqn[i] do
begin
a:=r;d:=-(2*r+s[j]);c:=r;
b:=-(1-r)*x[i-1,j]+(2*(1-r)-s[j])*x[i,j]-(1-r)*x[i+1,j];
end;
end;
end;
end.

program diffusion1 (input, output);
{Analytical solution of oxygen diffusion(inside cylinder)}
const
eps=0.00000001;
pi=3.14159265359;

function j0 (x:real):real;
var u,term,s0,s1,k:real;
r: integer;
begin
x:=abs(x);
if x≤8 then
begin
u:=-0.25*x*x;
s0:=1;
term:=1;
r:=0;
repeat
r:=r+1;
term:=term*u/(r*r);
s0:=s0+term;
until abs (term)<eps;
j0:=s0;
end
else
begin
s1:=sqrt(2/(pi*x));
u:=-0.015625/(x*x);
```

```
term:=s1;
s0:=term;
r:=0;
repeat
r:=r+2;
k:=(2*r-1)*(2*r-3);
term:=term*u*k*k/(r*(r-1));
s0:=s0+term
until abs(term)<eps;
term:=s1*0.125/x;
s1:=term;
r:=1;
repeat
r:=r+2;
K:=(2*r-1)*(2*r-3);
term:=term*u*k*k/(r*(r-1));
s1:=s1+term;
until abs(term)<eps;
j0:=s0*cos(x-0.25*pi)+s1*sin(x-0.25*pi);
end;
end;
function zeroj1 (s: integer): real;
var x,u,sum: real;
begin
if s=1 then zeroj1:=3.831706
else
begin
x:=(s+0.25)*pi;
u:=1/(x*x);
sum:=(((24.39137388*u-1.701319231)*u+0.2302734375)*u-
                                                0.0234375)*u+0.375;
zeroj1:=x-sum/x;
end
end;
var z,n,w,i,r: integer;
  dt,y,t,x,x1,j,k,sum,last,term: real;
begin
write('values of dt,n,z');
read(dt,n,z);
while n>0 do
begin
y:=1/n;
x1:=0;
for w:=0 to z do
begin
```

```
writeln(w);
t:=w*dt;
for i:=0 to n do
begin
x:=i*y;
if x1:=0 then sum:=0.125-t else
sum:=0.125-t+0.5*x1*x1*ln(x1);
r:=0;
last:=1;
repeat
r:=r+1
k:=zeroj1(r);
j:=j0(k);
last:=abs(term);
term:=((1-x1*x1)*j+x1*x1)/(k*k*j*j)*(j0(k*x)*exp(-k*k*t));
sum:=sum+term;
until (abs(term),eps) and (last<eps);
if t:=0 then
if x1:=0 then term :=0.25*x*x
else
term:=0.25*(x*x-x1*x1)-0.5*x1*x1*(ln(x)-ln(x1));
writeln('terms', r:6,' sum',sum:12:6,' true', term:12:6);
end;
writeln;
end;
write('values of dt, n,z');
read(dt,n,z);
end
end.

program diffusion2 (input,output);
{Numerical solution of Oxygen diffusion (inside cylinder)}
var
n,m,i,k,j: integer;
u:array[0. . 40,0. . 1000] of real;
t,y,f,r,x: real;
p:array[0. . 1000] of real;
begin
write('values of t,n,m');
read(t,n,m);
while n>0 do
begin
y:=1/n;
r:=t/(y*y);
k:=2;
```

```
f:=-t;
p[0]:=1;
for i:=0 to n do
begin
x:=1-i*0.1;
u[i,0]:=0.25*x*x;
end;
for j:=0 to (m-1) do
begin
u[0,j+1]:=2*r*(u[1,j]-u[0,j])+u[0,j]-t;
for i:=1 to n-k do
begin
u[i,j+1]:=r*(1-0.5*(1/((1/y)+i)))*u[i-1,j]+(1-2*r)*u[i,j]+
                              r(1+0.5*(1/((1/y)+i)))*u[i+1,j]+f
end;
u[n-k+1,j+1]:=u[n-k+1,j]+2*r*(u[n-k,j]/(p[j]+1)-u[n-k+1,j]/p[j])+
((t/y)*((-p[j]*u[n-k,j])/(p[j]+1)+(p[j]-1)*u[n-k+1,j]/p[j]))/((n-
k+1)*y)+f;
if (u[n-k+1,j+1]≤0) or (u[n-k+1,j+1>(u[n-k+1,j])) then
begin
k:=k+1;
p[j-1]:=p[j-1]+1;
u[n-k+1,j]:=u[n-k+1,j-1]+2*r*(u[n-k,j-1]/(p[j-1]+1)-u[n-k+1,j-1]/p[j-
1])+((t/y)*(-p[j-1]*u[n-k,j-1]/(p[j-1]+1)+(p[j-1]-1)*u[n-k+1,j-1]/p[j-
1]))/((n-k+1)*y)+f;
p[j]:=sqrt(2*u[n-k+1,j])/y;
u[n-k,j+1]:=(r-t/(2*y+2*(n-k)*y*y))*u[n-k-1,j]+(1-2*r)*u[n-
k,j]+(r+t/(2*y+2*(n-k)*y*y))*u[n-k+1,j]+f;
u[n-k+1,j+1]:=u[n-k+1,j]+2*r*(u[n-k,j]/(p[j]+1)-u[n-
k+1,j]/p[j])+((t/y)*((-p[j]*u[n-k,j])/(p[j]+1)+(p[j]-1)*u[n-
k+1,j]/p[j]))/((n-k+1)*y)+f;
p[j+1]:=sqrt(2*u[n-k+1,j+1)/y;
end
else
p[j+1]:=sqrt(2*u[n-k+1,j+1])/y;
x:=1-((n-k+1)+p[j])*y;
write(j,' ':1);
for i:=0 to n-k+1 do
begin
write(u[i,j]:8:6,' ':1);
end;
write(x:7:5);
writeln;
end;
```

```
write('values of t,n,m');
read(t,n,m);
end
end.

program diffusion3 (input,output);
{Analytical solution of oxygen diffusion (outside cylinder)}
const
x1=2;
pi=3.14159265359;
count=20;

function approxzerojy (s:integer):real;
var sum,a,b,p,q,r,term1,term2,term3:real;
begin
if s:=1 then approxzerojy:=1.794007;
   else begin
    a:=((s-0.5)*pi)/(x1-1);
   b:=1/a;
    p:=(3*x1+1)/(8*x1*(x1-1));
     q:=(-63*x1*x1*x1-25)/(384*x1*x1*x1*(x1-1));
   r:=(1899*x1*x1*x1*x1*x1*x1+1073)/(5120*x1*x1*x1*x1*x1*x1*(x1-1));
  term1:=p;
   term2:=q-p*p;
   term3:=r-4*p*q+2*p*p*p;
    sum:=((term3*b*b+term2)*b*b+term1)*b+a;
   approxzerojy:=sum;
  end;
end;

function f0 (x:real):real;
var z,f:real;
begin
z:=3/x;
f:=(((0.00014476*z-0.00072805)*z+0.00137237)*z-0.00009512)*z
      -0.0055274;
f0:=(f*z-0.00000077)*z+0.79788456;
end;

function g0  (x:real):real;
var z,g:real;
begin
z:=3/x;
g:=(((0.00013558*z-0.00029333)*z-0.00054125)*z+
         0.00262573)*z-0.00003954;
```

```
g0:=(g*z-0.04166397)*z-0.78539816+x;
end;

function j0 (x:real):real;
var z,j:real;
begin
x:=abs(x);
z:=x*x/9;
if x<=3 then
   begin
j:=(((0.00021*z-0.0039444)*z+0.0444479)*z-0.3163866)*z+
     1.2656208;
j0:=(j*z-2.2499997)*z+1;
end
else   j0:=f0(x)*cos(g0(x))/sqrt(x);
end;

function y0 (x:real):real;v
var z,y:real;
begin
x:=abs(x);
z:=x*x/9;
if x<=3 then
begin
y:=(((-0.00024846*z+0.00427916)*z-0.04261214)*z+
      0.25300117)*z-0.74350384;
y0:=(y*z+0.60559366)*z+0.36746691+(2/pi)*ln(0.5*x)*j0(x);
end
else  y0:=f0(x)*sin(g0(x))/sqrt(x);
end;

function f1 (x:real):real;
var z,f:real;
begin
z:=3/x;
f:=(((-0.00020033*z+0.00113653)*z-0.00249511)*z+
      0.00017105)*z+0.01659667;
f1:=(f*z+0.00000156)*z+0.79788456;
end;

function g1 (x:real):real;
var z,g:real;
begin
z:=3/x;
g:=(((-0.00029166*z+0.000798224)*z+0.00074348)*z-
```

```
        0.00637879)*z+0.00005650;
g1:=(g*z+0.12499612)*z-2.35619449+x;
end;

function j1 (x:real):real;
var z,j:real;
begin
 x:=abs(x);
z:=x*x/9;
if x<=3 then
begin
j:=(((0.00001109*z-0.00031761)*z+0.00443319)*z-
      0.03954289)*z+0.21093573;
j1:=((j*z-0.56249985)*z+0.5)*x;
end
else  j1:=f1(x)*cos(g1(x))/sqrt(x);
end;

function y1 (x:real):real;
var z,y:real;
begin
x:=abs(x);
z:=x*x/9;
if x<=3 then
begin
y:=(((0.0027873*z-0.0400976)*z+0.3123951)*z-1.3164827)*z+
      2.1682709;
y1:=((y*z+0.2212091)*z-0.6366198)/x+(2/pi)*ln(0.5*x)*j1(x);
end
else
   y1:=f1(x)*sin(g1(x))/sqrt(x);
end;

function eqn (x:real):real;
begin
   eqn:=j0(x*x1)*y1(x)-y0(x1*x)*j1(x);
end;

function root   (a,b:real):real;
var  fb,mid,fa,fmid,:real;
       i:integer;
begin
fa:=eqn(a);
fb:=eqn(b);
for i:=1 to count do
```

```
 begin
 mid:=0.5*(b+a);
  fmid:=eqn(mid);
 if fa*fmid>0
     then begin a:=mid; fa:=fmid; end
else begin b:=mid; fb:=fmid; end;
end;
root:=0.5*(a+b);
end;

procedure locate (var a,b:real; db:real);
var fa,fb:real;
begin
fa:=eqn(a);
fb:=eqn(b);
while fa*fb>0 do
     begin
     a:=b;
     fa:=fb;
     b:=b+db;
      fb:=eqn(b);
     end;
end;

function zerojy (s:integer):real;
var xa,xb,dx:real;
begin
dx:=0.01;
xa:=approxzerojy(s);
xb:=xa+dx;
locate (xa,xb,dx);
zerojy:=root (xa,xb);
end;

function jy1 (x,k:real):real;
begin
jy1:=j1(x*k)*y1(k)-y1(x*k)*j1(k);
end;

function jy0 (x,k:real):real;
begin
jy0=j0(x*k)*y1(k)-y0(x*k)*j1(k)
end;
```