SIMULATION OF SYSTEMS MODELLED BY CONVECTION DIFFUSION EQUATIONS

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

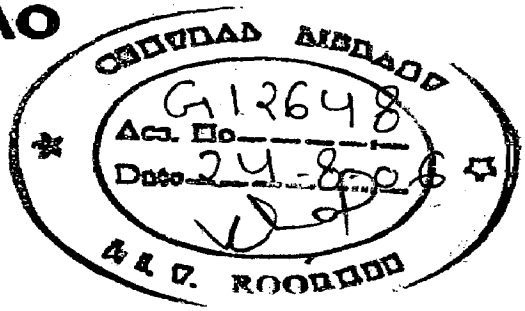
in

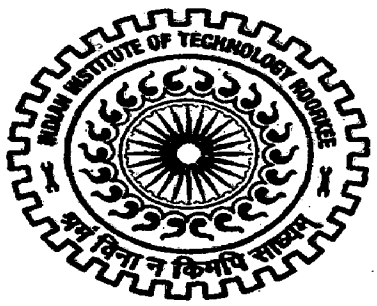
CHEMICAL ENGINEERING

(With Specialization in Computer Aided Process Plant Design)

By

J.SHASHIDHAR RAO





DEPARTMENT OF CHEMICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)
JUNE, 2006

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled **"SIMULATION OF SYSTEMS MODELLED BY CONVECTION DIFFUSION EQUATIONS"**, in partial fulfillment of the requirement for the award of the degree of Master of Technology in Chemical Engineering with specialization in **"COMPUTER AIDED PROCESS PLANT DESIGN"**, submitted in the Department of Chemical Engineering, Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during the period from July 2005 to June 2006, under the guidance of **Dr. SURENDRA KUMAR**, Professor, Department of Chemical Engineering, Indian Institute of Technology Roorkee.

The matter embodied in this project work has not been submitted for the award of any other degree.

Date: 30 June, 2006
Place: IIT Roorkee.

(J.SHASHIDHAR RAO)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

30/6/06

Dr. SURENDRA KUMAR
Professor,
Department of Chemical Engineering,
Indian Institute of Technology Roorkee.

ACKNOWLEDGEMENT

I am greatly indebted to my guide **Dr. SURENDRA KUMAR,** Professor, Department of Chemical Engineering, Indian Institute of Technology Roorkee, for his kind support and guidance during the entire course of this work. His co-operation and in-depth knowledge have made my work possible.

My special thanks are due to **Dr. (Mrs) Shashi,** Assistant Professor, Department of Chemical Engineering, IIT Roorkee, whose valuable knowledge helped me in understanding the phenomena during this dissertation work.

I would like to thank **Dr. Shrichand**, Head of Department, and Chairman, DRC, Department of Chemical Engineering, IIT Roorkee for providing various facilities during the course of this project work.

I would like to thank Mr. Rahman Jee, Mangeeram Jee, Ms. Tripta Garg for their kind co-operation.

I would like to thank Krishna, Viswanath, Shoeb, & Rahul and all my classmates for their suggestions and help in completion of this dissertation work.

Last but not the least, it is all owed to the blessings of my parents and God that I have come up with this work in due time.

Thanks are due to CAD CENTRE facilities.

(J.SHASHIDHAR RAO)

ABSTRACT

In the present scenario, it has become a characteristic feature to use fast and reliable computations in almost all branches of science and engineering by the evolution of modeling and simulation techniques. Various systems, processes, scientific and engineering principles, theories are being constantly modeled to carry out design, research and application oriented works. Coming to engineering field, these model equations provide a platform for a detailed analysis of the system to provide for safe and efficient operations and devising a control theory employing the system parameters.

Convection Diffusion equations are one such type of the equations that are frequently used in describing a variety of chemical engineering systems such as reactors, heat exchangers etc.

The solutions to these partial differential equations many a times are not possible using the regular analytical techniques and imply the use of numerical methods in finding out the solution. The development of the numerical methods which can capably represent the system is a major area of research.

Presently in literature, the method of finite differences and method of finite elements are mostly used to simulate these types of equations.

In this dissertation work, a detailed study on developing finite difference equations for the so termed Convection Diffusion equations (Linear or non linear partial differential equations) is carried out.

Computer programs are developed in the MATLAB 7.0 software for the drafted difference schemes. The programs written are quite user friendly and hold a capability to give error free results with minute tolerances. These programs are constantly checked and compared with the PDEPE toolbox in MATLAB software for their efficiency in calculations and convergence towards the exact solutions.

Finally it is concluded that the formulated difference schemes along with the computer program codes can be used effectively to carry out the simulation of a large number of Convection Diffusion equations from the fields of chemical engineering.

A brief reference is given to the future research and advancements that can be carried out in the present work to simulate even complex systems for higher dimensions.

CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

Various engineering and physical systems are characterized by the flow of mass, momentum and energy. Thus we require considering these flows in a detailed manner while modeling these systems. The detailed modelling often needs to consider both the spatial and transient nature of the system. This is very important in Dynamic Modeling.
Further; Dynamic simulation plays an important role in analyzing and predicting the spatial and transient behavior of chemical and biochemical processes and is therefore very used, especially in the last decade. But, as mentioned long time ago by Heydweiller and al. (1977) and confirmed more recently, most existing dynamic modelling and simulation tools are primarily suited to lumped parameter systems [19].
In fact, in the reality, a large number of unit operations in Chemical Engineering such as catalytic reactors, absorption column, and extraction column based on mass transfer approach and in Biochemical Engineering are intrinsically distributed in nature. It means that their properties exhibit spatial as well as temporal variations. The resulting set of equations for these types of models may be viewed as a combination of lumped and distributed parameter systems, namely described by Partial Differential Algebraic Equations (PDAE) submitted to initial conditions and boundary conditions [19].
And thus, Partial Differential equations (PDE), PDAEs arise in an enormous number of modelling applications.
In this report we study one such form of the partial differential equations namely **Convection Diffusion** equations. These are basically the equations that describe the flow (both molecular and bulk flow) of fundamental property (mass, momentum and enthalpy) and are used in a large number of systems/processes.
During the course of this report, we shall discuss in briefly about the basic derivations and models that use the Convection Diffusion equations. A discussion about modeling and simulation of a number of chemical engineering systems based on such equations is carried out.

1.1 Objectives:

We set the following objectives for the present work

- To derive the basic important models that use Convection Diffusion equations.
- To prepare finite difference analogues of these model equations.
- To develop the algorithm to solve descretized model equations.
- To develop computer programs in MATLAB and analyze their results.

1.2 Organisation of Thesis:

Here in the next section, we mention the classification of partial differential equations first and thereafter we shall study the definition of Convection Diffusion equation, its general forms and the engineering implications of the equations.

In chapter 2, we study the application of the Convection Diffusion equations to some of the chemical engineering systems and develop their respective model equations. A literature study on the recently used numerical techniques to handle these Convection Diffusion equations and their limitations is mentioned.

Further a special focus is given to the 'METHOD OF FINITE DIFFERENCES' as a numerical tool for simulation.

In the chapter 3 the very basic fundamentals, nomenclature about finite differences method and their use in tackling a number of different situations for steady state and dynamic conditions are covered.

Thereafter in the chapter 4 the difference equations for some of the chemical engineering applications are developed with due consideration given to the system constraints and boundary conditions.

In the chapter 5 the results obtained as a result of application of the finite differences schemes developed in previous chapters through the implementation by MATLAB software codes are mentioned.

Finally in the concluding section, a brief mention of the various works carried out in this dissertation work is done along with the conclusions drawn from the simulation work.

A reference for the scope of future work that can be carried out in this field is given at the end.

A detailed Appendix covering the various MATLAB codes written for this dissertation work is attached to the report.

1.3 Classification of Partial Differential Equations

When considering the partial differential equations we can classify them by considering a general form given as [19, 16];

$$A\frac{\partial^2\Phi}{\partial x^2} + 2B\frac{\partial^2\Phi}{\partial x\partial y} + C\frac{\partial^2\Phi}{\partial y^2} + D\frac{\partial\Phi}{\partial x} + E\frac{\partial\Phi}{\partial y} + F\Phi = g(x,y) \qquad 1.3\text{-}1$$

We define a variable d as $d := AC - B^2$

One defines the differential equation as parabolic, hyperbolic or elliptic depending on the values taken by'd'.

The equation is:

Parabolic Differential equation: $d = 0$ (one parameter characteristics)

Hyperbolic differential equation: $d < 0$ (two parameter characteristics)

Elliptic Differential equation: $d > 0$ (no real characteristics)

This classification of the differential equation is oriented to the equations for parabolas, hyperbolas and ellipses of the plane geometry in which the equation

$$ax^2 + 2bxy + cy^2 + dx + ey + f = 0 \qquad 1.3\text{-}2$$

describes parabolas ($ac\text{-}b^2 = 0$), hyperbolas ($ac\text{-}b^2 < 0$), and ellipse ($ac\text{-}b^2 > 0$).

Accordingly for the differential equations discussed in the text we have;

Diffusion equation: $\frac{\partial U}{\partial t} = \nu\frac{\partial^2 U}{\partial x^2}$ i.e. it holds $A = \nu$, $B = C = 0$ and thus $D = 0$. The Diffusion equation has parabolic properties.

Wave equation: $\frac{\partial^2 U}{\partial t^2} = c^2\frac{\partial^2 U}{\partial x^2}$ i.e. it holds $A = C^2$, $B = 0$, $C = -1$ and thus $d = -c^2 < 0$. The Wave equation is hyperbolic.

Potential equation: $\frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = 0$ i.e. $A = C = 1$, $B = 0$ and thus $d > 0$. The potential equation shows elliptic behavior.

For parabolic equations, they have a property which is important for the numerical solution. The solution of the differential equation determined at a certain point of flow does not depend upon the boundary conditions that lie downstream. This makes it possible to find the solution for the entire flow field via forward integration. i.e the solution can be computed in a certain level alone from the values of the preceding level. This characteristic behavior is not found in the elliptic equations [19].

Further the equations which we are going to consider are called Convection Diffusion equations which are basically Diffusion equations. But they contain an additional term to account for the convective process in the equation which entertains some hyperbolic characteristics to the parabolic nature of the Diffusion equation. Thus the numerical solution of such an equation is a considerable part of study and research [16,15].

In the subsequent chapters of this work we study the various details of the "CONVECTION DIFFUSION" Equations along with a comparative study of the recent methods used to solve them. We start here with the basic definition and derivation of the equation.

1.4 Definition & Theory:

CONVECTION DIFFUSION EQUATIONS:

Convection Diffusion Equations can be defined as the partial differential equations resulting from basic conservation laws of mass, momentum & energy transfer considering both molecular (Diffusive) and bulk (Convective) flows [2].

A general form of Convection Diffusion equation (for mass flow) is shown below [1, 2];

$$\frac{\partial C}{\partial t} + V_X \frac{\partial C}{\partial Y} + V_Y \frac{\partial C}{\partial Y} + V_Z \frac{\partial C}{\partial Z} = D\left(\frac{\partial^2 C}{\partial Z^2} + \frac{\partial^2 C}{\partial Y^2} + \frac{\partial^2 C}{\partial X^2}\right) \qquad 1.4\text{-}1$$

Where;

C- Concentration of the component species in flow

Vx, Vy, Vz are the components of velocity in the spatial directions *X, Y, & Z.*

In a similar fashion we can represent the momentum and energy balances.

The above equation (1.4-1) considers both spatial and temporal variation of the concentration. A detailed method of deriving the above Convection Diffusion equation is given in Appendix A1.

The above equation can be written in a compact form as;

$$D\underline{\nabla}^2 C - \underline{V}.\underline{\nabla}C = \frac{\partial C}{\partial t} \qquad 1.4\text{-}2$$

Equation (1.4-2) represents the general form of Convection Diffusion equation for mass transfer.

Further if there is a volume source of solute within volume 'V' of strength Q_0 (mass/volume/time) then Q_o is added to the LHS of above equation.

For small $\underline{V}$, diffusion is dominant & the convection term may be neglected.

If D is small then Convection dominates almost everywhere. However the term cannot be neglected or omitted as a first approximation unless a boundary condition is also omitted. This is the phenomena of boundary layer.

Next, we study the engineering implications of the Convection Diffusion equations.

The concentration satisfies the Convection Diffusion equation at steady state as;

$$D\underline{\nabla}^2 C - \underline{V}.\underline{\nabla}C = 0 \qquad 1.4\text{-}3$$

Further let; L is the characteristic Length

& U_0 is the initial fluid velocity.

Then the above Convection Diffusion equation can be efficiently written as;

$$\frac{1}{Pe}\nabla^2 C - \underline{V}.\underline{\nabla}C = 0 \qquad 1.4\text{-}4$$

Where Peclet number, $Pe = U_0 L / D$

Since D is usually small; Pe is nearly always large; which means that mass transport is dominated by Convection except in boundary layers [2].

As can be seen from the basic Convection Diffusion equation we require knowledge of the velocity field. The governing equation for velocity distribution is also a Convection Diffusion equation which is derived same as shown above considering the momentum fluxes instead of the mass fluxes.

These governing equations are also called as Navier-Stokes equations [2].

For an incompressible, Newtonian fluid with a density ρ and subject to a body force F, the velocity field $\underline{V}$ and the pressure field P satisfy the following equation;

$$\frac{\partial \underline{V}}{\partial t} + (\underline{V}.\underline{\nabla})\underline{V} = -\frac{1}{\rho}\underline{\nabla}P + \underline{F} + \nu\nabla^2\underline{V} \qquad 1.4\text{-}5$$

where;

ν = Kinematic Viscosity

and the Continuity Equation $= \underline{\nabla}.\underline{V} = 0$

Also, Reynolds number $= \mathrm{Re} = \frac{U_0 L}{\nu}$. This determines the relative importance of the Convection & Diffusion fluxes.

For the steady flow with zero body force ($\underline{F} = 0$) the Navier Stokes equation becomes;

$$\frac{\partial \underline{V}}{\partial t} + (\underline{V}.\underline{\nabla})\underline{V} = -\frac{1}{\rho}\underline{\nabla}P + \nu\nabla^2\underline{V} \qquad 1.4\text{-}6$$

this can be easily written as;

$$\frac{1}{\mathrm{Re}}\nabla^2\underline{V} - (\underline{V}.\underline{\nabla})\underline{V} = \frac{1}{\rho U_0{}^2}.\underline{\nabla}P \qquad 1.4\text{-}7$$

For small Reynolds number: Viscous (Diffusive) forces dominate

For large Reynolds number: Inertial (Convective) forces dominate

Thus the Reynolds number indicates the relative importance of the momentum convection over momentum diffusion in a fluid. While the, Peclet number indicates the importance of the mass convection over mass diffusion.

As we have seen till now, the Convection Diffusion equations for mass and momentum balances, similar way we can derive for the energy balances considering energy fluxes. However in the energy equation we need to consider one more term to account for the conduction occurring along the boundaries of the system.

1.5 Generalized forms of the Convection Diffusion equation [2, 6]:

The Convection Diffusion equation in its very basic form as derived in Appendix A1 is given as;

$$D\underline{\nabla}^2 C - \underline{V}.\underline{\nabla}C = \frac{\partial C}{\partial t} \qquad 1.5\text{-}1$$

writing in a more precise manner we have;

$$\frac{\partial C}{\partial t} + V_X \frac{\partial C}{\partial Y} + V_Y \frac{\partial C}{\partial Y} + V_Z \frac{\partial C}{\partial Z} = D\left(\frac{\partial^2 C}{\partial Z^2} + \frac{\partial^2 C}{\partial Y^2} + \frac{\partial^2 C}{\partial X^2} \right) \qquad 1.5\text{-}2$$

This equation can be taken as the most general form of the "Convection Diffusion" equation in terms of rectangular co-ordinates. Here we have not considered the source or sink term which can be added or subtracted from the above generalized form.

The equations are supplemented with some initial and boundary conditions to solve analytically or numerically and determine the distribution of the dependant variable(C, V and T) spatially and temporally.These set of conditions work out many times for a preliminary solution of the system.

It is generally found that the systems evaluated are not essentially easily described by the rectangular co-ordinate system. Further the systems many a times do have complex geometries. For such cases we find it easy to describe the system in terms of spherical and cylindrical co-ordinates.

Here in this section we present the general forms of the Convection Diffusion equation in these co-ordinate systems.

Once again we recall the Convection Diffusion equation in terms of the dimensionless form as;

$$\frac{\partial C}{\partial t} + Pe(V.\nabla)C = \Delta C \qquad 1.5\text{-}3$$

In terms of **Spherical Co-ordinates**:

$$(V.\nabla)C = V_\omega \frac{\partial C}{\partial \omega} + V_Z \frac{\partial C}{\partial z} + V_\psi \frac{\partial C}{\partial \psi} \quad \text{and ;} \qquad 1.5\text{-}4$$

$$\Delta C = \frac{1}{\omega}\frac{\partial}{\partial \omega}\left(\omega \frac{\partial C}{\partial \omega}\right) + \frac{\partial^2 C}{\partial z^2} + \frac{1}{\omega^2}\frac{\partial^2 C}{\partial \psi^2} \quad \text{where,} \qquad 1.5\text{-}5$$

$$\omega = \sqrt{x^2 + y^2}$$

In terms of **cylindrical co-ordinates;**

$$(V.\nabla)C = V_r \frac{\partial C}{\partial r} + \frac{V_\theta}{r}\frac{\partial C}{\partial \theta} + \frac{V\psi}{r.\sin\theta}\frac{\partial C}{\partial \psi} \quad \text{and;} \qquad 1.5\text{-}6$$

$$\Delta C = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial C}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial C}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\left(\frac{\partial^2 C}{\partial \psi^2}\right) \quad \text{where,}$$

1.5-7

$$r = \sqrt{x^2 + y^2 + z^2}$$

Any of these general forms of the Convection Diffusion equations can be used to describe the system; however the boundary and the initial conditions are essentially required to be considered in the same co-ordinate system.

CHAPTER 2

LITERATURE REVIEW

2.1 Applications of Convection Diffusion Equations:

Convection Diffusion Equations have a wide range of applications in various fields of engineering, physical sciences and financial analysis [8].

Recently the equations are also applied to various biological processes to describe the fluid phenomena in cellular reactions.As the equations represent the spatial and transient characteristics of the basic laws of conservation of energy, we can use them describing various unit operations that are carried out in the branch of chemical engineering.

Various operations such as catalytic reactors, absorption column, extraction column, adsorption column, fluidized bed reactors, packed bed reactors etc... involves these equations as model equations in different forms.

Further the equations are also used to model the processes such as flow in porous media. The dusty gas model based on Convection Diffusion equations is used to describe this particular flow.

In the field of physics, we use the equations in the problems of non linear heat transfer problems, transport of charged species in semiconductor and plasmas. Recently the water transport through a polymer electrolyte membranes (PEM) cell has been proved to obey the equation. This problem is of large interest since PEM is a key component to of fuel cells, which are expected to perform revolution in mobile power sources including vehicles [13].

Next, in the field of hydrology, the equations are used to describe the flood waves after an approximation using the kinematical wave equation.

In the following discussion we list out some of the chemical engineering applications of the Convection Diffusion equations.

2.1A Isothermal & Adiabatic Fixed Bed Reactors [4, 17]:

The plug flow model may be employed to the fixed bed catalytic reactor; provided plug flow behavior is a valid assumption.

Here we consider a small volume element of radius 'r' width Δr and height Δz, through which reaction mixture flows isothermally.

Suppose that radial & axial mass transfer can be expressed by Fick's law, with D_{er} and D_{el} as effective diffusivities, based on the total (void and non void) area perpendicular to the direction of diffusion.

The volume of the element = $2\Pi r \Delta r \Delta z$

Concentration in the fluid phase is constant within element.

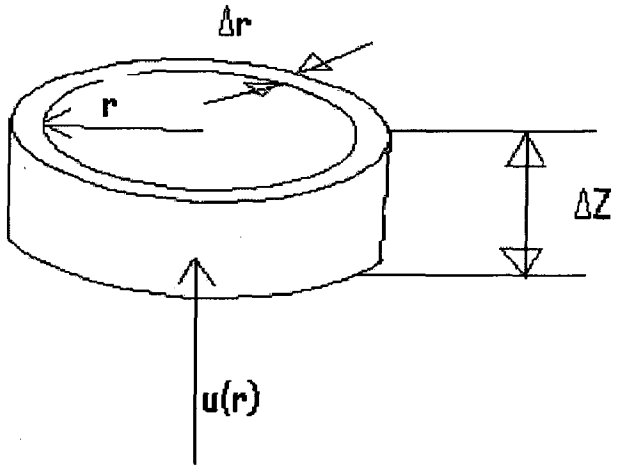


Fig. 2.1 Volume element of reaction mixture

Mass Balance equation:

The steady state differential equation for mass balance is given as;

$$\frac{\partial}{\partial r}\left(r(D_{er})\frac{\partial C}{\partial r}\right)+r\frac{\partial}{\partial z}\left(-uC+D_{eL}\frac{\partial C}{\partial z}\right)-r_P\rho_B.r=0 \qquad 2.1\text{-}1$$

Where;

r_p = Global rate of disappearance of reactant per unit mass of catalyst.

ρ_B = density of catalyst in the bed.

u = Superficial velocity in the axial direction.

Further; if the diffusivities are not sensitive to r; or to z, the velocity is not a function of z then we write above equation as;

$$(D_e)_r\left[\frac{1}{r}\frac{\partial C}{\partial r}+\frac{\partial^2 C}{\partial r^2}\right]-u\frac{\partial C}{\partial z}+(D_e)_L\frac{\partial^2 C}{\partial z^2}-r_P\rho_B=0 \qquad 2.1\text{-}2$$

If the velocities varies with z ;(due to changes in temperature or number of moles in a gaseous reaction), then the previous equation is to be used.

Assumptions:

Concentration entering the reactor = Co

& there is no axial dispersion in the feed line; the boundary conditions are;

$$\frac{\partial C}{\partial z}=0; \qquad -(D_e)_L\left(\frac{\partial C}{\partial z}\right)_{>0}+u(c)_{>0} \qquad \text{at } z = 0 \text{ for all } 'r'$$

and;

$$\frac{\partial C}{\partial r}=0 \quad \text{for } r = r_0 \text{ and } r = 0 \text{ at all } z \qquad 2.1\text{-}3$$

Next using dimensionless variables;

$$x=\frac{C_0-C}{C_0} \quad ; \quad r^*=\frac{r}{d_P} \quad ; \quad z^*=\frac{z}{d_P} \qquad 2.1\text{-}4$$

The above equation reduces to;

$$-\frac{1}{Pe_r}\left[\frac{1}{r^*}\frac{\partial x}{\partial r^*}+\frac{\partial^2 x}{\partial r^{*2}}\right]+\frac{\partial x}{\partial z^*}-\frac{1}{Pe_L}\frac{\partial^2 x}{\partial z^{*2}}-\frac{r_P\rho_B d_P}{C_o u}=0 \qquad 2.1\text{-}5$$

where; $Pe_r=\dfrac{ud_P}{(D_e)_r}$

and $Pe_L=\dfrac{ud_P}{(D_e)_L}$ 2.1-6

Above differential equation shows that conversion depends upon the dimensionless reaction rate group; $\tau_P\rho_B d_P / C_0 u$ and the radial & axial Peclet numbers.

The typical profile is flat in the center of tube, increases slowly until a maximum velocity is reached about one pellet diameter from the wall & then decreases sharply to zero at the wall.

The numerical solutions of this model are developed in the subsequent chapters of the report assuming a one dimensional dynamic equation where the radial terms are neglected.

2.1B Catalytic micro reactor [5]:

A model for the analysis of the fluid dynamics and heat transfer in catalytic micro reactor systems for the decomposition of the ammonia over a monolayer Ni non porous catalyst are discussed in literature (Ammonia being a potential source of Hydrogen for a number of fuel cell applications for small scale power generation).

The overall model consists of a flow model, mass transport model, an energy conservation model & a reaction kinetics model for ammonia decomposition.

The chemical reaction occurring is;

$$2NH_3 \longrightarrow N_2 + 3H_2 \qquad \Delta H = 11\,\text{kcal/mol.} \qquad \text{2.1.-7}$$

The small scale micro reactor devices generally have high heat transfer coefficients and mass transfer coefficients, enhanced surface to volume ratios for reactions & smallest occupied volumes for convenient end use in small scale application.

The small dimensions of the micro reactor systems reveal that the transport processes are strongly dominated by the Diffusion mechanism.

Assuming the gas to be an ideal & incompressible homogenous fluid, we get following equations;

Continuity equation:

$$\frac{1}{\rho C_S^2}\frac{\partial P}{\partial t} + \frac{\partial V_X}{\partial x} + \frac{\partial V_Y}{\partial y} = 0 \qquad \text{2.1-8}$$

Where;

P = Hydrostatic pressure

Cs = speed of sound in gaseous fluid.

The term in the braces accounts for the fluid to be slightly compressible.

Conservation of momentum:

(Navier –Stokes equation) [2, 6]:

The contribution of convective and diffusive mechanisms to momentum transport can be evaluated by the value of Reynolds number.

In absence of body forces the conservation of momentum for the creeping incompressible Newtonian fluid flow is expressed by the Stokes equation [2].

In two dimensional co- ordinate system;

$$\rho\frac{\partial V_X}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left(2\eta\frac{\partial V_X}{\partial x}\right) + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V_X}{\partial y} + \frac{\partial V_Y}{\partial x}\right)\right] \qquad 2.1\text{-}9$$

$$\rho\frac{\partial V_Y}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial}{\partial y}\left(2\eta\frac{\partial V_Y}{\partial y}\right) + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial V_Y}{\partial x} + \frac{\partial V_X}{\partial y}\right)\right] \qquad 2.1\text{-}10$$

where, η = Fluid viscosity.

Mass Transport equation:

Convection Diffusion equation gives;

$$\frac{\partial C_i}{\partial t} + V_X\frac{\partial C_i}{\partial x} + V_Y\frac{\partial C_i}{\partial y} = \frac{\partial}{\partial x}\left(D_{Xi}\frac{\partial C_i}{\partial x}\right) + \frac{\partial}{\partial y}\left(D_{Yi}\frac{\partial C_i}{\partial y}\right) + Sc \qquad 2.1\text{-}11$$

$$i = 1, 2...N$$

where;

D_{xi} & D_{yi} = mass diffusion coefficients in X & Y directions respectively.

and N = Total number of components in gaseous reaction.

Sc = Source / Sink term.

Energy Transport equation:

Solid walls of the domain are considered to be insulated so that there is no heat exchange with surroundings.

Assume negligible effects of boiling & condensation.

Two dimensional transient energy balance equation is as follows;

$$\rho C_P\frac{\partial T}{\partial t} + \rho C_P\left(V_x\frac{\partial T}{\partial x} + V_y\frac{\partial T}{\partial y}\right) = \frac{\partial}{\partial x}\left(k_x\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y\frac{\partial T}{\partial y}\right) + Sh \qquad 2.1\text{-}12$$

where, T = temperature term

C_p = Specific heat capacity

k_x & k_y are thermal conductivity of gaseous fluids.

Sh = Heat source/ sink.

Initial & Boundary conditions:

At the inlet of the catalytic micro reactors a plug flow velocity field is specified.

On the impermeable boundaries of the micro reactors, no slip wall boundary conditions are imposed.

The condition of the entering gases is specified at the inlet of the reactor.

Different constitutive relationships are required for fluid property evaluation.

The rate of ammonia decomposition can considered to be 1st order with respect to ammonia's partial pressure. The small dimensions of the micro chemical reactor systems reveal that the transport processes are strongly dominated by the diffusion mechanism.

The numerical solution of the governing equations can be Stiff in temporal direction due to diffusive terms and steep in spatial direction due to convection terms.

In literature a standard Galerkin Finite Element Technique has been employed for solution of the flow equations mentioned.

2.1C Adsorber Design [3]:

We discuss the **Thermal Swing Adsorption** type of adsorption as this type of adsorption is the most commonly used to prevent contamination of air by organic solvents of low concentration and to dehumidify gases.

Drying of air by TSA is one of the major commercial gas separation processes.

In practical adsorption, a one dimensional model serves to be enough for design & operation of the system, where radial temperature and concentration distributions are neglected.

The thermal properties of the adsorbates and adsorbents, radial temperature gradient and heat transfer through the column wall can affect the column dynamics due to high temperature of purge gas. The system is a packed bed with porous spherical adsorbent particles.

Mathematical model:

Assumptions:

1. The flow patterns of gas phase in the packed bed can be described by an axially dispersed plug flow model.

2. Gas phase behaves as an ideal gas mixture.
3. Frictional pressure drop is neglected.
4. Velocity distribution & second order concentration gradient in radial direction is neglected.

Because, only axial dispersion is considered both in one & in two dimensional models, the same component and overall mass balance equations for the gas phase were used.

5. Contribution of Nitrogen to total adsorption is neglected.
6. Thermal equilibrium is assumed between gas and solid phases.
7. Axial conduction in the column wall can be neglected but heat loss through column wall and heat accumulation in the wall cannot be neglected.

The balance equations are as follows:

Component mass balance:

$$-D_Z \frac{\partial^2 C}{\partial z^2} + \frac{\partial (uC)}{\partial z} + \frac{\partial C}{\partial t} + \rho_P \left(\frac{1-\varepsilon_B}{\varepsilon_B} \right) \frac{\partial \bar{q}}{\partial t} = 0 \qquad 2.1\text{-}13$$

Overall mass balance:

$$\frac{\partial (u/\ T)}{\partial z} - \frac{1}{T^2} \frac{\partial T}{\partial t} + \rho_P \frac{R}{P} \left(\frac{1-\varepsilon_B}{\varepsilon_B} \right) \frac{\partial \bar{q}}{\partial t} = 0 \qquad 2.1\text{-}14$$

Energy balance in bed:

$$\left(\alpha \rho_g C_{Pg} + \rho_B C_{Pg} + \rho_B \bar{q} C_{Pa} M_a\right) \frac{\partial T}{\partial t} + \frac{\partial}{\partial z} \left(\varepsilon_B u \rho_g C_{Pg} T\right) -$$

$$\rho_B \left(-\Delta H_S\right) \frac{\partial \bar{q}}{\partial t} - k_Z \frac{\partial^2 T}{\partial z^2} - k_r \left[\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) \right] = 0 \qquad 2.1\text{-}15$$

Energy balance in wall:

$$\rho_W C_{PW} A_W \frac{\partial T_W}{\partial t} = 2\Pi R_{Ri} \alpha_W \left(T \big|_{r=Rw} - T_W \right) - 2\Pi R_{B0} h_0 \left(T_W - T_{atm} \right) \qquad 2.1\text{-}16$$

In axial directions we need the Danckwert's boundary conditions for both energy and mass balances.Clean bed conditions and saturated bed conditions were used as initial conditions for adsorption breakthrough and regeneration breakthrough.

2.1D Fixed Bed catalytic reactors [7]:

Here we mention the Convection Diffusion equations required to represent the transport model of a fixed bed catalytic reactor.

The term "fixed" means that the catalyst pellets are held in place & do not move with respects to fixed reference frame.

Material and energy balances are required for both the fluid & catalyst particles.

Essentially all reactions occur within catalyst particles.

This step to consider;

1) Transport of reactants and energy from external surface
2) Transport of reactants & energy from external surface into porous pellet
3) Adsorption, chemical reaction and desorption of products at the catalytic sites.
4) Transport of products from catalyst interior to the external surface of the pellet
5) Transport of the products into the bulk fluid.

Next we take the rate limiting steps as below;

The system is intra particle transport controlled if the step 2 is the slow process (referred to as diffusion limited).

For kinetics or reaction control step 3 is the slowest process.

For the step 1 as the slowest process, the reaction is said to be externally transport controlled.

Let the effective diffusivity coefficient is D_j and sold density is ρ_s

Pellet void fraction $(\varepsilon) = \rho_p . V_g$

Let;

V_j = velocity of species j giving rise to molar flux N_j

$N_j = C_j V_j$

E = Total Energy within the volume element

e = flux of total energy through the boundary surface due to all mechanisms of transport.

The conservation of mass and energy is given by following Convection Diffusion equations;

$$\frac{\partial C_j}{\partial t} = -\nabla N_j + R_j \qquad j = 1, 2, \ldots N \qquad \text{2.1-17}$$

$$\frac{\partial E}{\partial t} = -\nabla e \qquad \text{2.1-18}$$

where;

R_j = production of species 'j' due to chemical reactions.

$$N_j = -D_j \nabla^2 C_j + R_j \qquad j = 1, 2 \ldots N \qquad \text{2.1-19}$$

$$e = -k\nabla^2 T - \sum_j N_j H_j \qquad \text{2.1-20}$$

Boundary equations are given as;

$$C_A = C_{AS} \quad where \quad r = R \qquad \text{2.1-21}$$

The symmetry of pellet implies the vanishing of derivative at the centre of the pellet.

$$\frac{\partial C_A}{\partial r} = 0 \quad \text{at} \quad r = 0 \qquad \text{2.1-22}$$

Next in terms of the characteristic length for sphere;

$$a = \frac{V_P}{S_p} = \frac{\frac{4\Pi R^3}{3}}{4\Pi R^2} = \frac{R}{3} \quad ; \quad \bar{r} = \frac{r}{a} \;\&\; \overline{C} = \frac{C_A}{C_{AS}} \qquad \text{2.1-23}$$

Above equation becomes;

$$\frac{1}{\bar{r}^2}\frac{d}{d\bar{r}}\left(\overline{r^2 . \frac{d\overline{C}}{d\bar{r}}}\right) - \Phi^2 . \overline{C} = 0 \qquad \text{2.1-24}$$

subject to;

$$\frac{d\overline{C}}{d\bar{r}} = 0 \text{ for } \overline{r = 0} \qquad \text{2.1-25}$$

The numerical simulation of this equation considering steady state conditions has been done in this report.

In a similar way we can formulate the Convection Diffusion equations for other types of reactors. Even the Combustion process can be modeled by such equations.

As we have seen that the convection diffusion equations consider both the temporal and spatial variations in the systems, the resulting partial differential equations are not easy to solve using the analytical techniques and thus require a variety of numerical methods to solve these equations.

In the next section we discuss some of the numerical methods that are in recent use to solve these Convection Diffusion Equations along with their applications.

2.2 Numerical methods to Solve Convection Diffusion Equations

In the previous section we have described the importance of the Convection Diffusion equations in characterizing flows in various fields of Science and Engineering.The equation is a very useful in predicting the dynamic characteristics of a number of systems in the field of Chemical Engineering.

In the present section we shall discuss briefly the various numerical methods for obtaining the solution of these Convection Diffusion equations with as much as practical accuracy as possible.

Several methods are applied to the Convective Diffusion equation because it has a sharp front but is a linear problem with an exact solution. However due to the hyperbolic behavior of the equation induced by the term of Convection we need to handle the equation cautiously.

While investigating the interpolation error when we use some general numerical methods such as finite difference or finite element method to interpolate the exact solution, we find that there comes a restriction on the maximum number of interval that are considered to reduce the possible error [15].

We write the Convection Diffusion in the mathematical form as follows [2];

$$\frac{\partial C}{\partial t} + Pe\frac{\partial C}{\partial x} = \frac{\partial^2 C}{\partial x^2} \qquad 2.2\text{-}1$$

$$\text{subject to } C(x,0) = 0; C(0,t) = 1; \frac{\partial C}{\partial x}(1,t) = 0 \qquad 2.2\text{-}2$$

A variety of traditional numerical methods including method of finite differences, finite element method, Galerkin finite element method ,Orthogonal collocation, method of lines(MOL),Weighted Residual Methods, Finite Volume methods, Adaptive Grid

Methods & Moving Grid Methods are presently used in dealing with the solution of these equations. We discuss some of these methods in the following sections.

2.2A Numerical Solution of One-Dimensional Convection-Diffusion Equation (finite differences method) [8]:

Here several finite difference schemes for solving the one-dimensional convection-diffusion equation with constant coefficients are discussed.

In the research carried out by **Mehdi Dehghan** [8] the use of Modified Equivalent Partial Differential Equation (MEPDE) as a means of estimating the order of accuracy of a given finite difference technique is emphasized.

This approach can unify the deduction of arbitrary techniques for the numerical solution of convection-diffusion equation. It is also used to develop new methods of high accuracy. This approach allows simple estimation and comparison of the errors associated with the partial differential equation. Various difference approximations are derived for the one-dimensional constant coefficient convection-diffusion equation.

Computational aspects are also studied in the research work.

When comparing the explicit finite difference techniques described in the report, it was found that the most accurate method is the fourth-order explicit formula.

2.2B Finite element Approximations of Convection Diffusion Equations Using Graded Meshes [18]:

As is well known, the numerical approximation of convection-diffusion equations requires some special treatment in order to obtain good results when the problem is convection dominated due to the presence of boundary or interior layers. A lot of work has been done in this direction. There are in principle two ways to proceed: to use some kind of upwind or to use adapted meshes appropriately refined. **Ricardo G. Duran and Ariel L. Lombardi** [18] proved that, using appropriate graded meshes, the solution is well approximated by the standard piecewise bilinear finite element method.

The numerical experiments showed that no oscillations appear in the numerical solution and the predicted order of convergence is observed.

2.2C Finite-Element Solution of Convection Diffusion with reaction [11]

The techniques are based on the modified method of characteristics, in some cases including streamline or velocity-weighted numerical diffusion to suppress numerical oscillations.A effective scheme is mentioned in the paper authored by **Biyue Liu, Myron B. Allen, Hristo Kojouharov, Benito Chen** [11].

The finite element method discussed involve the dividing the volume of the system into small elements (fine gridding) and are analysed each of these elements for solution. The equations obtained are expressed in terms of matrices and are thus solved by integrating over the volume.

The study is focused on Finite Element scheme for the two-dimensional, advective reaction-diffusion equation.

The general form of such a equation with reaction is given as;

$$\Phi\frac{\partial C}{\partial t}+V.\nabla C-\nabla.(D\nabla C)=f(C) \quad \text{for concentration } C(x,\ t) \qquad 2.2\text{-}3$$

What makes the above equation numerically challenging is the advective term $V.\nabla C$

One can measure the degree of advection dominance via the dimensionless Peclet number. The numerical solution requires considering fine gridding and is expensive.

A common alternative to fine gridding is to suppress the oscillations by adding numerical diffusion of various types.

The amount of such numerical diffusion to be added is devised in the work.

2.2D Fourth-order difference schemes on rotated grid for two-dimensional convection–diffusion equation [10]:

A fourth order difference scheme on the rotated grid for the two-dimensional Convection– Diffusion equation is also developed in the literature.

A comparison of the fourth-order schemes with the nine point scheme obtained from the second-order central difference scheme has shown remarkable accuracy and sharp error reduction. In literature a fourth-order compact finite difference approximation for above Convection Diffusion equation is already derived by Gupta et al. The idea behind the method is to utilize the Taylor's series expansions of all the functions involved in above equation. In the paper authored by, **Jun Zhang, Jules Kouatchou & Lixin Gea** [10] a

derivation of a family of fourth-order finite difference schemes of above equation on the rotated grid is given.

Besides the numerical techniques discussed above a large number of methods are currently in research and applications such as Sinc-Galerkin method for convection diffusion equation with mixed boundary conditions [14] , Broyden scheme[20], High resolution FEM-FCT scheme, Finite volume Descretization scheme, method of approximation etc...

However it is seen that the most commonly used and widely accepted methods are the methods of finite elements and the method of finite differences [15, 16].

This is because, these are easy to formulate, less expensive when carried out with accepted limits of error, easily converged, stable, accurate for designing and using in the computational Software applications with short computational times.

From the literature review carried out on CONVECTION DIFFUSION equations it is clear that a vast number of engineering & physical systems can be modeled using the Convection Diffusion equations. The recent applications include the use of the equation in biochemical engineering, fuel cell modeling, modeling & modeling of the micro reactors. The equations have a great importance in describing the flow & conservation of mass, momentum, & energy in various unit processes and operations from the field of Chemical Engineering.

The numerical solution of these equations determines transport models of the systems which is a very good tool in writing the computer algorithms for studying and determining the fluid properties.

From the field of the numerical techniques, we find that the 'Method of the Finite Differences is largely used in handling the convection diffusion equations which has many advantages over other classical techniques in the view of convergence, accuracy and error reduction.

CHAPTER 3

METHOD OF FINITE DIFFERENCES

Finite Difference Method [15, 16, 21]:

Here in this chapter, we first discuss the basics of the method of finite differences, the different approximations meant for the partial derivatives along with the order of the errors associated with their use.

Subsequently we derive the difference equations for the steady state and dynamic Convection Diffusion equations. Consequently we include the addition of different complexities such as non linearity, complex boundary conditions (moving boundary type), addition of source terms, purely hyperbolic nature, solution for simultaneous differential equations etc.

Based on these difference schemes we carry out the descretization of some of the engineering problems in the next chapter.

The Finite difference method is concerned with specific points in the domain called grids. The domain is divided into a number of equidistant intervals. We then use the Taylor series expansion to deduce the difference formulae for first and the second derivatives of the given differential equation.

Using these approximations of the derivatives we obtain an interpolation formula.

Further we obtain the value of a parameter at a grid point using the linear difference equations and carry out similar scheme for other grid points.

Thus we see that the **Finite Differences method** has the advantage that the method is easy to formulate, although it may need a large number of grid points for high accuracy.

Derivatives must be correctly evaluated in order not to destroy the accuracy that has been achieved. While handling the partial differentials we express the spatial derivatives too in the same way as shown in the previous equations.

The local truncation error is defined as the difference of the differential equation and the finite difference scheme. The finite difference scheme is called consistent if the limit of the local truncation error is zero as Δx *and/or* Δt *approach zero.*

While obtaining the higher order derivatives we use the derivatives interpolations obtained by the difference equation of lower order derivatives.

A different range of the finite difference schemes are available based on the consideration of the number of grid points in the estimation of the values at a particular grid point.

In practice we use 2-point, 3-point, 5-point and 9-point schemes.

The finite differences schemes are also classified based on the truncations done in the Taylor approximations. These are called first order, second order, third order & fourth order descretizing techniques.

3.1 Steps involved in Finite Differences method

A Finite difference method typically involves the following steps:

1. Generate a grid, where we want to find an approximate solution.
2. Substitute the derivatives in an ODE/PDE or an ODE/PDE system of equations with finite difference schemes. The ODE/PDE then become a linear/non-linear system of algebraic equations.
3. Solve the system of algebraic equations using some standard iterative techniques.
4. Implement and debug the computer code.
5. Do the error analysis, both analytically and numerically.

In the following section we study the difference equations for steady state Convection Diffusion equations.

3.2 Finite Difference methods for steady state Convection diffusion Equations:

In this part of study, we shall study the method of formulation of various finite difference schemes that can be used to approximate the steady state Convection Diffusion equations. To understand the approximation procedure we consider a differential equation and subject it to different situations and conditions for analysis [16, 21, 23].

Here as an example, we consider the differential equation used to describe the heat conduction in an un-insulated, tapered rod which is thin enough that a one dimensional analysis can be used.

$$\frac{d^2u}{dx^2}+\frac{2}{q+x}\frac{du}{dx}-\frac{2p}{q+x}u=0 \qquad 3.2\text{-}1$$

The length variable, x and the temperature variable, u, are dimensionless and normalized. The constants 'q' and 'p' are dimensionless combination of parameters which describe the geometry and heat transfer characteristics of the rod. These are defined as;

$$p=\frac{hL}{k}\sqrt{1+\frac{4}{f^2}} \qquad \text{and} \qquad q=\frac{D_0}{fL} \qquad 3.2\text{-}2$$

where L is the length of the rod.

h is the coefficient of heat transfer between the rod and surroundings.

k is the thermal conductivity of the rod.

f and D_0 define the diameter, D, of the rod by $D = D0 + fLx$.

Consider the simplest boundary conditions as:

$$u(0) = 0 \quad \text{and} \quad u(1)=1 \qquad 3.2\text{-}3$$

The above differential equation is a steady state form of Convection Diffusion equation. The first term accounts for the diffusion of heat and the second term results as a result of convection.

Now we convert this continuous differential equation into discrete finite difference equation that is an approximation with some error.

Using the Taylor series expansion, the adjacent points that are separated by finite difference, Δx (i.e. step size), are given as;

$$u(x+dx)=u(x)+\frac{du}{dx}(x).\Delta x+\frac{d^2u}{dx^2}(x).\frac{(\Delta x)^2}{2!}+\frac{d^3u}{dx^3}(x).\frac{(\Delta x)^3}{3!}+.....+ \qquad 3.2\text{-}4$$

Notation for the discrete variables:

It is very convenient to use a special notation for the discrete variables that are spaced between 0 and 1 as shown in figure 3.1

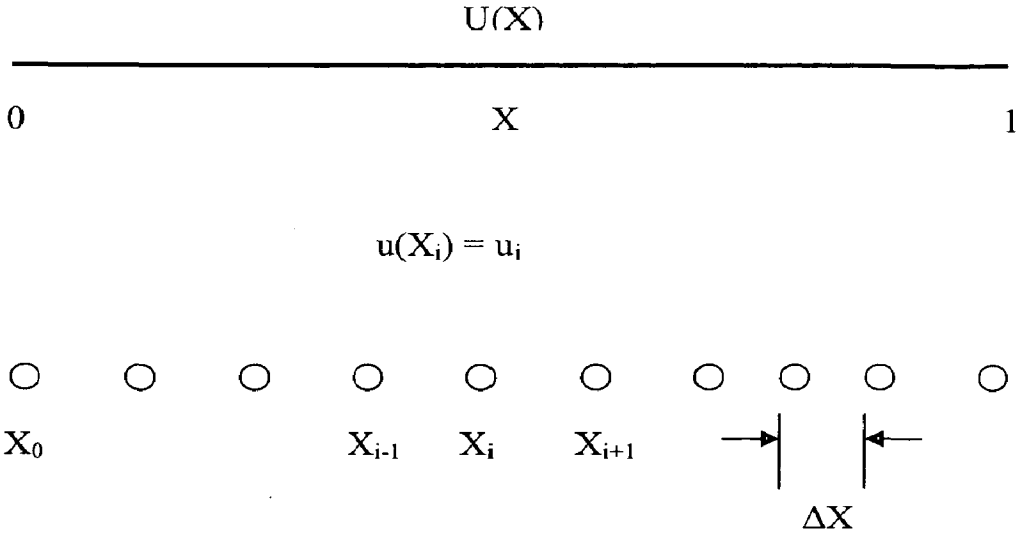


Fig. 3.1 Discrete and continuous variables

We denote the value of the discrete variable at each of these points by the subscript '*i*' and define it as;

$$x_i = i(\Delta x) \qquad 3.2\text{-}5$$

The index '*i*' takes the values from 0 to *R*, where *R* is the total number of increments in the complete interval between 0 and 1.

Further the adjacent values are given as;

$$x_{i+1} = x_i + \Delta x$$

$$x_{i-1} = x_i - \Delta x \qquad 3.2\text{-}6$$

Now the values of dependant variable *u* at x_i are denoted by u_i

We follow this nomenclature through out this work.

In the nomenclature given above, the Taylor series appears as;

$$u_{i+1} = u_i + \left(\frac{du}{dx}\right)_i \Delta x + \left(\frac{d^2u}{dx^2}\right)_i \frac{(\Delta x)^2}{2!} + \left(\frac{d^3u}{dx^3}\right)_i \frac{(\Delta x)^3}{3!} + \left(\frac{d^4u}{dx^4}\right)_i \frac{(\Delta x)^4}{4!} + \ldots\ldots + \qquad 3.2\text{-}7$$

Similarly the value u_{i-1} at x_{i-1} is ;

$$u_{i-1} = u_i - \left(\frac{du}{dx}\right)_i \Delta x + \left(\frac{d^2u}{dx^2}\right)_i \frac{(\Delta x)^2}{2!} - \left(\frac{d^3u}{dx^3}\right)_i \frac{(\Delta x)^3}{3!} + \left(\frac{d^4u}{dx^4}\right)_i \frac{(\Delta x)^4}{4!} + \ldots\ldots + \qquad 3.2\text{-}8$$

Analogs for the derivatives:

a) Analog for the first derivative:

When equation 3.2-7 is solved for the first derivative; it takes the form;

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1}-u_i}{\Delta x} - \left(\frac{d^2u}{dx^2}\right)_i \frac{\Delta x}{2!} - \left(\frac{d^3u}{dx^3}\right)_i \frac{(\Delta x)^3}{3!} - \ldots. - \qquad 3.2\text{-}9$$

The error in using this analog is of the order of the first term which is truncated $(d^2u/dx^2)_i(\Delta x/2!)$. *This term contains* Δx, *so the truncation error is said to be first order, and the analog is said to be first order correct.*

b) Analog for second derivative:

A finite difference analog to the second order derivative is obtained by adding equations and 3.2-7 and 3.2-8. The result is;

$$u_{i+1} + u_{i-1} = 2u_i + \left(\frac{d^2u}{dx^2}\right)_i (\Delta x)^2 + 2\left(\frac{d^4u}{dx^4}\right)_i \frac{(\Delta x)^4}{4!} + \ldots. + \qquad 3.2\text{-}10$$

On writing the equation explicitly for second derivative, we have;

$$\left(\frac{d^2u}{dx^2}\right)_i = \frac{u_{i+1}-2u_i+u_{i-1}}{(\Delta x)^2} - \left(\frac{d^4u}{dx^4}\right)_i \frac{(\Delta x)^2}{12} - \ldots. - \qquad 3.2\text{-}11$$

This approximation is second order correct, since the first term dropped contains $(\Delta x)^2$.

An improved analog for the first derivative is obtained by subtracting equation 3.2-8 from equation 3.2-7. The resulting equation, written explicitly for the first derivative is;

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1}-u_i}{2(\Delta x)} - \left(\frac{d^3u}{dx^3}\right)_i \frac{(\Delta x)^2}{6} - \ldots.. - \qquad 3.2\text{-}12$$

This approximation is second order correct and possesses a good degree of precision.

Finite Difference Equations:

On substituting the respective difference equations for first and second derivatives into above equations we obtain;

$$\frac{u_{i+1}-2u_i+u_{i-1}}{(\Delta x)^2} + \frac{2}{q+i(\Delta x)} \cdot \frac{u_{i+1}-u_{i-1}}{2(\Delta x)} - \frac{2p}{q+i(\Delta x)} u_i = 0 \qquad 3.2\text{-}13$$

When rearranged, it becomes;

$$\left[1-\frac{\Delta x}{q+i(\Delta x)}\right]u_{i-1}+\left[-2-\frac{2p(\Delta x)^2}{q+i(\Delta x)}\right]u_i+\left[1+\frac{\Delta x}{q+i(\Delta x)}\right]u_{i+1}=0 \qquad 3.2\text{-}14$$

Boundary conditions:

The finite difference equations written about the points x_1 and x_R are different from the above equations and contain the terms involving the boundary points that are known from the boundary conditions. Putting these values we get the following equations for the boundary points.

$$\left[-2-\frac{2p(\Delta x)^2}{q+\Delta x}\right]u_1+\left[1+\frac{\Delta x}{q+\Delta x}\right]u_2=0 \qquad 3.2\text{-}15$$

$$\left[1-\frac{\Delta x}{q+(R-1)(\Delta x)}\right]u_{R-2}+\left[-2-\frac{2p(\Delta x)^2}{q+(R-1)(\Delta x)}\right]u_{R-1}=-\left[1+\frac{\Delta x}{q+(R-1)(\Delta x)}\right] \qquad 3.2\text{-}16$$

Hence we have converted the given non linear differential equation into a set of linear algebraic equations. This is the essence of using the finite differences method.

These equations are to be solved simultaneously for determining the values of dimensionless dependant variable *'u'*.

The equations above can be written in the following tridiagonal form;

$$b_1u_1 + c_1u_2 + 0 + \ldots\ldots\ldots\ldots + 0 = d_1$$
$$a_2u_1 + b_2u_2 + c_2u_3 + 0 + \ldots\ldots\ldots + 0 = d_2$$

$$0 + \ldots a_iu_{i-1} + b_iu_i + c_iu_{i+1} + \ldots\ldots\ldots + 0 = d_i \qquad 3.2\text{-}17$$

$$0 + \ldots\ldots + 0 + a_{R-2}u_{R-3} + b_{R-2}u_{R-2} + c_{R-2}u_{R-1} = d_{R-2}$$
$$0 + \ldots\ldots + 0 + a_{R-1}u_{R-2} + b_{R-1}u_{R-1} = d_{R-1}$$

A method for solving a set of equations of this type (3.2-17) has been developed by Thomas. An algorithm to implement this method is given in Appendix A2.

The computer program for this entire difference method is also given in Appendix A4.

The results and discussions obtained are discussed in detail in the chapter 5.

Other types of Boundary conditions:

We now apply the method of finite differences to the same differential equation, but for some complex boundary conditions.

We consider the following types of boundary conditions:

$$\text{a)} \quad \frac{du}{dx}(0) = g \qquad \text{3.2-18}$$

$$u(1) = 1$$

These conditions arise when the rate of heat transferred to the rod is specified at one end. The boundary condition at $x = 1$ is unchanged. However the boundary condition specified at $x=0$ is that the first derivative, du/dx, is equal to the constant value g. To specify the value of the first derivative at the boundary, a fictitious point, $x_{-1,}$, outside the region is used. The second-order-correct analog to $(du/dx)_0$, given by equation 3.2-12 with $i= 0$, is then set equal to g. Thus,

$$u_{-1} = u_1 - 2g(\Delta x) \qquad \text{3.2-19}$$

The boundary equation is obtained by substituting this value for u_{-1} into 'equation 3.2-14 for $i = 0$. The resulting equation is

$$\left[-2 - \frac{2p(\Delta x)^2}{q}\right]u_0 + 2u_1 = 2g(\Delta x)(1 - \frac{\Delta x}{q}) \qquad \text{3.2-20}$$

This equation is included in the formulation of tridiagonal system of equations and is solved using Thomas Algorithm (Appendix A2).The computer code is given in Appendix A4.The results and discussions are given in the chapter 5.

b) Now we consider a more general boundary condition that results when the rate of heat transferred to an end of the rod is given by Newton's law of convection. In this case the boundary condition at $x = 0$ can be expressed as

$$\frac{du}{dx}(0) - Hu(0) = -g \qquad \text{3.2-21}$$

$$\text{where} \quad H = \frac{hL}{k}$$

The condition of equation 3.2-21 for $x = 0$ is handled similarly to that for equation 3.2-18 A fictitious point is used, and the dependent variable at this point, u_{-1}, is eliminated from

the finite difference equation for $x = 0$ by the finite difference analog of the boundary condition. This is;

$$\frac{u_1 - u_{-1}}{2(\Delta x)} - Hu_0 = -g \qquad 3.2\text{-}22$$

$$u_{-1} = u_i - 2H(\Delta x)u_0 + 2g(\Delta x) \qquad 3.2\text{-}23$$

The resulting finite difference equation is

$$\left[-2 - 2H(\Delta x) - \frac{2(p-H)(\Delta x)^2}{q}\right]u_0 + 2u_1 = -2g(\Delta x)(1 - \frac{\Delta x}{q}) \qquad 3.2\text{-}24$$

The equations for this problem also are of the tridiagonal form and can be readily solved by the Thomas algorithm (given in Appendix A2).

A computer program for this problem is also given in Appendix A4.

Many a times, it is desirable to develop a single set of equations that can handle a variety of boundary conditions. This objective can be accomplished by spacing the first point at half increment from the boundary. The points are arranged as shown in the following figure 3.2 and the value of the independent variable at each point is given by

$$x_i = (i - \frac{1}{2})\Delta x \qquad 3.2\text{-}25$$

The complete set of difference equations for the difference equations are given in the following chapter.

The program to implement this technique is given in Appendix A4 and the results that are obtained with exactness are discussed in the chapter 5.

One disadvantage of the arrangement of points as mentioned above is that the temperature at the boundaries is not computed. However, each of these can be evaluated as the average of the exterior value and the last interior value. The exterior value can be obtained from the appropriate boundary condition analog.

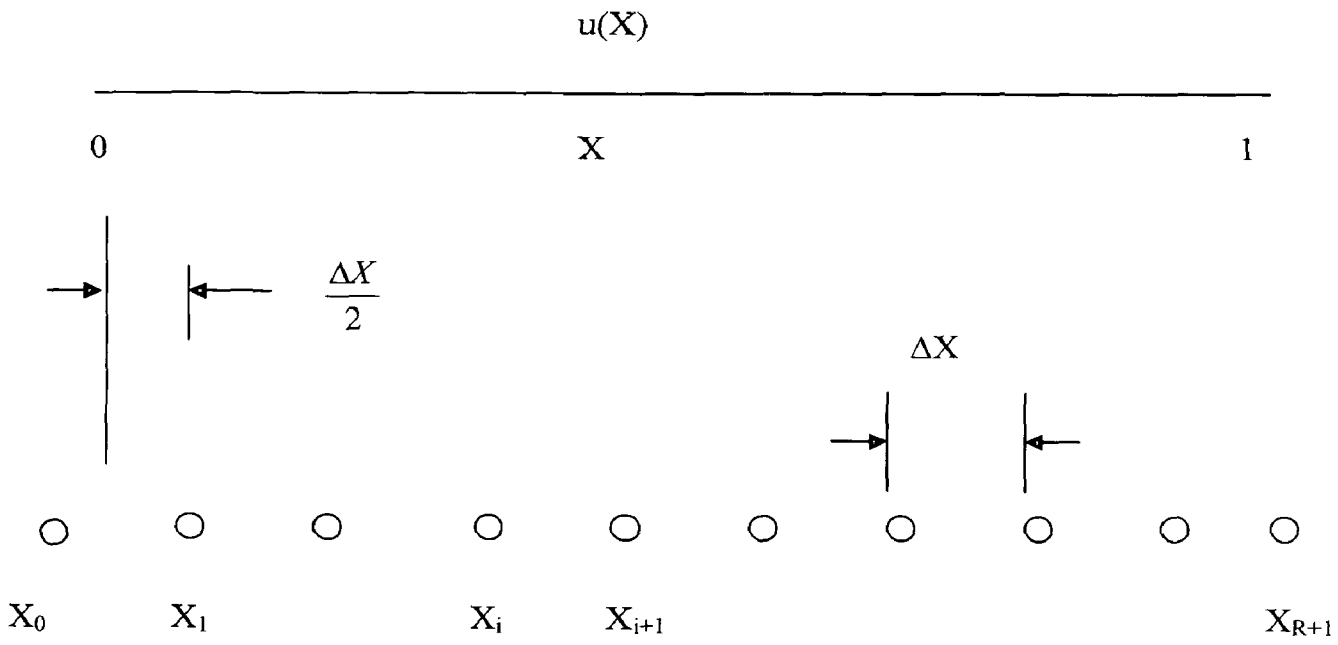


Fig 3.2 Grid points shifted from boundaries.

The arrangement of points shown in Figure 3.2 is also convenient to use when the problem is described by radial coordinates. The differential equation for heat conduction in radial coordinates is similar to the equation for a tapered rod which goes to a point at one end. This latter equation is obtained by setting the parameter *q* in equation to zero. In fact, when the parameter '*p,* is also set to zero so that there is no heat loss from the surface of the rod by convection, equation 3.2-1 becomes the equation for heat conduction in a sphere. The equation for heat conduction in a cylinder is similar. Although numerical methods can be used to great advantage for unsteady state heat conduction in a cylinder or sphere, the steady state problems for these geometries have trivial solutions.

3.3 Finite differences for unsteady state Convection Diffusion Equations:

In the Introduction chapter it was stated that most of the applications involving Convection Diffusion equations are dynamic in nature (behavior varying with time).

In this section we shall see the treatment of such applications. The equations arising from such analysis are mostly partial differential and possess either parabolic or hyperbolic nature.

We first consider the most general form of such equation where the term interpreting the convection phenomenon is neglected. This simplification gives a good understanding of the derivation of difference equations[16, 21, 23].

Consider the one dimensional diffusion equation.

The equation is purely parabolic.

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}1$$

the boundary conditions are given as,

$$\left.\begin{aligned} u(0,t) &= 0 \\ u(1,t) &= 1 \end{aligned}\right\} all\ t \qquad 3.3\text{-}2$$

With an initial condition given as;

$$u(x,0) = 0; \quad x < 1 \qquad 3.3\text{-}3$$

Since the time domain has to be included in the difference equations, we do this by adding *'n'* as subscript for *'u'* in the difference equations. We take the time step as Δt.

The descretization grid is as shown in the following figure.

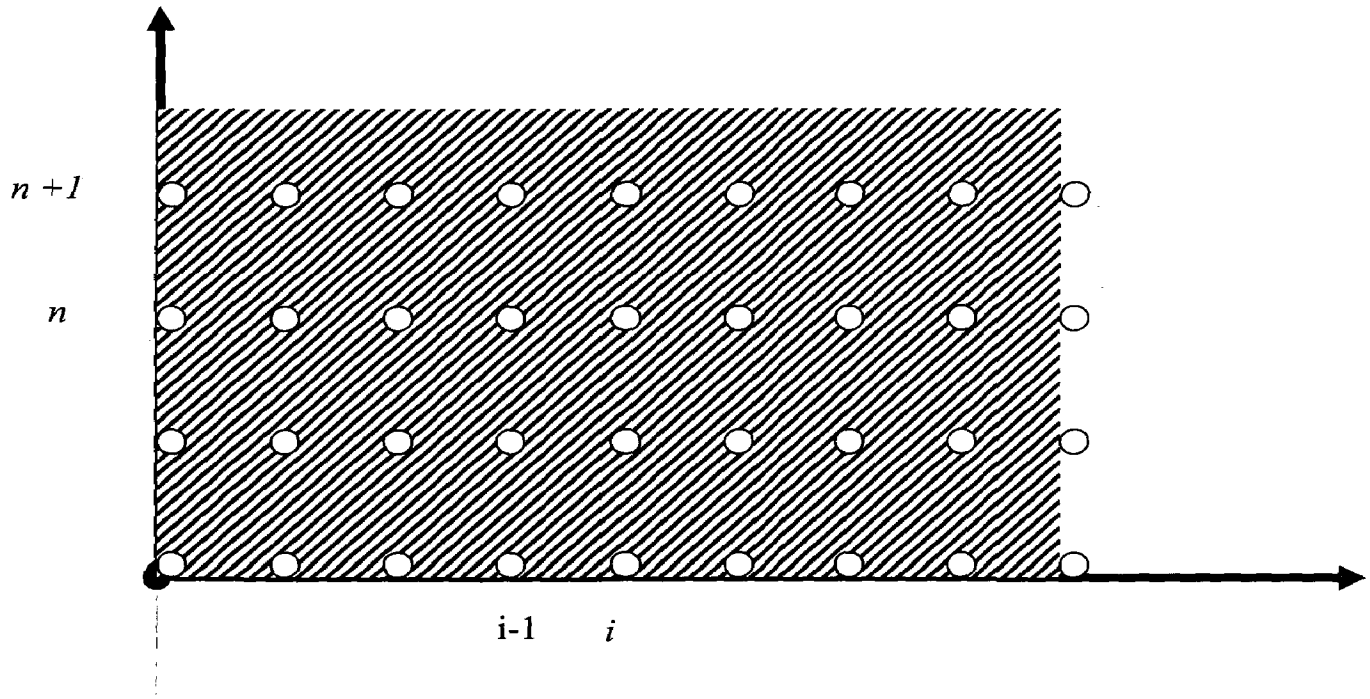


Fig. 3.3 Grid points for unsteady state problem.

For many numerical solutions, it will be desirable to increase the size of the time step as the solution progress. However in this entire simulation work we shall consider the step size to remain constant along the time axis.

The dependant variable is now a function of two independent variables, x and t.

3.3-1 General procedure for solving Parabolic Equations:

For the problem described by above equations (3.3-1), the value of the dependant variable is unknown at a row of points at each time level and there are actually an unlimited number of time intervals. It is not feasible to solve for all the unknown vales of u simultaneously even when a limited number of time intervals are considered.

Consequently the technique employed is to solve for the unknown values of u at one time interval, using the known values of u at the previous time level.

The values of u at the initial time level, where $n = 0$; are given by the initial conditions.

These values are used to determine the unknown values of u at the next time level for which $n = 1$ and so on. This procedure is continued for as many time increments as desired. Therefore the finite difference equations are so formulated that they contain values of u at two consecutive time levels.

3.3-1a Forward Difference equation.

The forward finite difference equation is an explicit method and is probably the most well known, although it is the least efficient of all the possible equations which can be used.

In developing the forward finite difference equation, we write an analog for $\partial^2 u / \partial x^2$ at the known time level which is indexed by n.

The relation thus used is,

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,n} \approx \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{(\Delta x)^2} \qquad 3.3\text{-}4$$

This analog is second order correct in the variable x.

The analog to time derivative is obtained from Taylor series in time about the point x_i, t_n

This is given as;

$$\left(\frac{\partial u}{\partial t}\right)_{i,n} = \frac{u_{i,n+1} - u_{i,n}}{\Delta t} - \left(\frac{\partial^2 u}{\partial t^2}\right)_{i,n} \frac{\Delta t}{\lfloor 2} \cdots \qquad 3.3\text{-}5$$

The analog resulting from truncations of this series after the first term is first order correct in t.

When the analogs defined as above are substituted in to the equation; we get the difference equation as;

$$u_{i,n+1} = \frac{\Delta t}{(\Delta x)^2}(u_{i+1,n} + u_{i-1,n}) + \left[1 - \frac{2\Delta t}{(\Delta x)^2}\right] u_{i,n} \qquad 3.3\text{-}6$$

This equation certainly contains only one unknown value and is written explicitly.

The numerical computations of the dependant variable are thus made one point at a time.

Even though it appears that the forward difference equation is an easy way to estimate the dependant variable, there lies a problem with its convergence.

For a numerical solution to be of any value, its solution must converge to solution of the corresponding differential equation when the finite increments are Δx ***and*** Δt ***are decreased.***

Analysis has shown that a very restrictive relationship between the size of Δx and that of Δt must be satisfied in order for the solution of the equation to approach that of the equation. The restriction requires that the ratio of Δt to Δx must remain less than or equal to ½.

This restriction is rather a serious one, because in order to minimize the truncation errors in the x analogs, the size of Δx has to be small. Thus for the forward equation, the size of Δt must remain on the order of Δx for the solution to be stable and hence a very small value of Δt must be used for stability even when a much larger value could be used without causing truncation error.

A difference equation which does not need this restriction is therefore a much better one to use as an analog to the diffusion equation.

3.3-1b Backward Differences:

In searching for a new finite difference equation which does not have a restriction on the size of Δt for stability, we might write the finite difference analogs for the $\partial^2 u / \partial x^2$ at the new or unknown time interval which is indexed by $n+1$. This backward difference is;

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,n+1} = \frac{u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{(\Delta x)^2} \qquad 3.3\text{-}7$$

The time derivative analog is obtained from a Taylor series in time about point x_i, t_{n+1}

The series is;

$$\left(\frac{\partial u}{\partial t}\right)_{i,n+1} = \frac{u_{i,n+1} - u_{i,n}}{\Delta t} + \left(\frac{\partial^2 u}{\partial t^2}\right)_{i,n+1} \frac{\Delta t}{2} - \ldots - \qquad 3.3\text{-}8$$

The first term on the right side of the equation is a first order correct analog to the time derivative.

On substitution we arrive at the following implicit equation;

$$u_{i-1,n+1} + \left[-2 - \frac{(\Delta x)^2}{\Delta t}\right] u_{i,n+1} + u_{i+1,n+1} = -\frac{(\Delta x)^2}{\Delta t} u_{i,n} \qquad 3.3\text{-}9$$

The equation is implicit because it contains three values of the dependant variable u at the unknown time interval.

If we consider R increments along x direction then; we have to write down the separate equations for the boundary conditions for $i=1$ and $i=(R-1)$.

For the given boundary conditions we get the following difference equations for these values of i.

$$\left[-2 - \frac{(\Delta x)^2}{\Delta t}\right] u_{R-1,n+1} + u_{2,n+1} = \left[-\frac{(\Delta x)^2}{\Delta t}\right] u_{1,n} \qquad 3.3\text{-}10$$

$$u_{R-2,n+1} + \left[-2 - \frac{(\Delta x)^2}{\Delta t}\right] + u_{R-1,n+1} = \left[-\frac{(\Delta x)^2}{\Delta t}\right] u_{R-1,n} - 1 \qquad 3.3\text{-}11$$

Finally, we get a set of equations in the form of; 3.2-17. In terms of matrix, we have a coefficient matrix that is tridiagonal. Thus the equations can be readily solved using the Thomas Algorithm (Appendix A2). After determining the values at one time level we

can proceed for the next time interval with the same type of equations and the values found in the previous time level.

The difference equation suffers no restriction on time step for stability. However we need to use a small time step in order to reduce the truncation error of the Taylor series in time. *The backward difference is an efficient one, and is simple to use. However, it is only first order correct in time.*

It is thus desirable to find out second order correct analog to this derivative.

3.3-1c Crank-Nicholson equation:

The desired second order correct analog is called the Crank-Nicholson equation.

For this equation all the finite difference equations are written about the point $x_i, t_{n+1/2}$ which is half way between the known and the unknown time levels.

The second order correct analog for the time derivative at this point is given by;

$$\left(\frac{\partial u}{\partial t}\right)_{i,n+1/2} = \frac{u_{i,n+1} - u_{i,n}}{(\Delta t)} - \left(\frac{\partial^3 u}{\partial t^3}\right)_{i,n+1/2} \frac{(\Delta t)^2}{24} - \ldots - \qquad 3.3\text{-}12$$

The second order spatial derivative is approximated about the point x by the arithmetic average of its finite difference analogs at the point x_i, t_n and x_i, t_{n+1}. The resulting analog is the average of the forward and backward analogs and is given as;

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,n+1/2} \approx \frac{1}{2}\left[\frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{(\Delta x)^2} + \frac{u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{(\Delta x)^2}\right] \qquad 3.3\text{-}13$$

Further on substituting these approximations in the differential equations we get the following difference equation;

$$u_{i-1,n+1} + [-2 - \frac{2(\Delta x)^2}{\Delta t}]u_{i,n+1} + u_{i+1,n+1} = -u_{i-1,n} + [2 - \frac{2(\Delta x)^2}{\Delta t}]u_{i,n} - u_{i+1,n} \qquad 3.3\text{-}14$$

The boundary condition equations can then be derived from the above difference equation in a similar way as was done in the backward difference scheme.

The set of equations thus obtained are readily solved in the tridiagonal form employing the Thomas Algorithm (Appendix A2).

Here a large value of time step can be effectively used since the time derivative analog is second order correct. *Thus the Crank-Nicholson equation is more efficient than the backward difference method and is thus the preferred method for obtaining the numerical solutions for parabolic differential equations.*

The Crank-Nicholson equation, like the backward difference equation is stable for all ratios of Δx and Δt.

For other types of the boundary conditions:

The difference equation developed in the previous sections(3.2) can be used with other types of complicated boundary conditions with approximations for the derivative terms in the boundary condition equations and including them to approximate the difference equations for the values of $i=1$ and $i=R-1$;

However it is observed sometimes that the coefficients in the boundary conditions may bring oscillations in the solutions which cause the smaller step size selection inevitable. The smaller step size serves the purpose by compensating these oscillations.

3.3-2 Parabolic equation with small Dispersion Coefficient:

Consider the following Convection Diffusion equation. As expressed in detail in the previous chapters, the first term here considers the diffusion of the dependant variable which may be as such concentration, Velocity, temperature etc. The second term is the convective transfer term which considers the bulk transport of the dependant variable.

The differential equation to be considered is;

$$\frac{\partial^2 u}{\partial x^2} - b\frac{\partial u}{\partial x} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}15$$

The above equation describes the one dimensional flow of a fluid with dispersion or diffusion. Here the parameter *b* is the ratio of the velocity of flow to the dispersion coefficient.

When the dispersion coefficient is small, or the parameter b is very large, the adjoining term becomes the controlling term on the left side. Under such conditions the numerical solutions at a given time interval oscillate around the true curve. The extent of these oscillations depends upon the relative values of b and Δx .Price et al have shown that for the Crank Nicholson equation there will be no oscillations when;

$$\left(\frac{b\Delta x}{2}\right) < 1 \quad \text{for } 0 \le x \le 1. \qquad 3.3\text{-}16$$

As a result, for large values of 'b', small values of Δx are to be used to eliminate this oscillation. For problems in one space dimension, it should be possible to use a Δx, small enough to eliminate the oscillation. When the dispersion coefficient becomes zero so that 'b' becomes unbounded and the equation is no longer parabolic, it is impossible to use small enough time increment. Consequently, the crank Nicholson and backward difference equations are not applicable to the solution of differential equations of such a type.

The difference equations that are developed by the three methods: Forward Differences, Backward Differences, Central Differences (Crank Nicholson scheme) are applied to the one dimensional parabolic equation as shown and the results are analyzed to check for their accuracy and stability in the chapter 5.
A difference method to solve for the parabolic equation with small dispersion coefficient is also developed and analyzed in the next chapter.
MATLAB programs for all these equations are given in Appendix A4.

3.3-3 Linear Hyperbolic partial differential equations:

One-dimensional hyperbolic differential equations arise from pure convection problems. The simplest equation describes the flow of a fluid through a tube with no transfer of the quantity conserved and with no generation or consumption. This equation is

$$-b\frac{\partial u}{\partial x} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}17$$

The centered difference equation is used for its numerical solution.

The Centered Difference Equation:

For the centered difference equation, the finite difference analogs are centered in both space and time with respect to the grid points at which the values of the dependent variable are determined. In Figure 3.4, two successive time levels of grid points are shown in space.

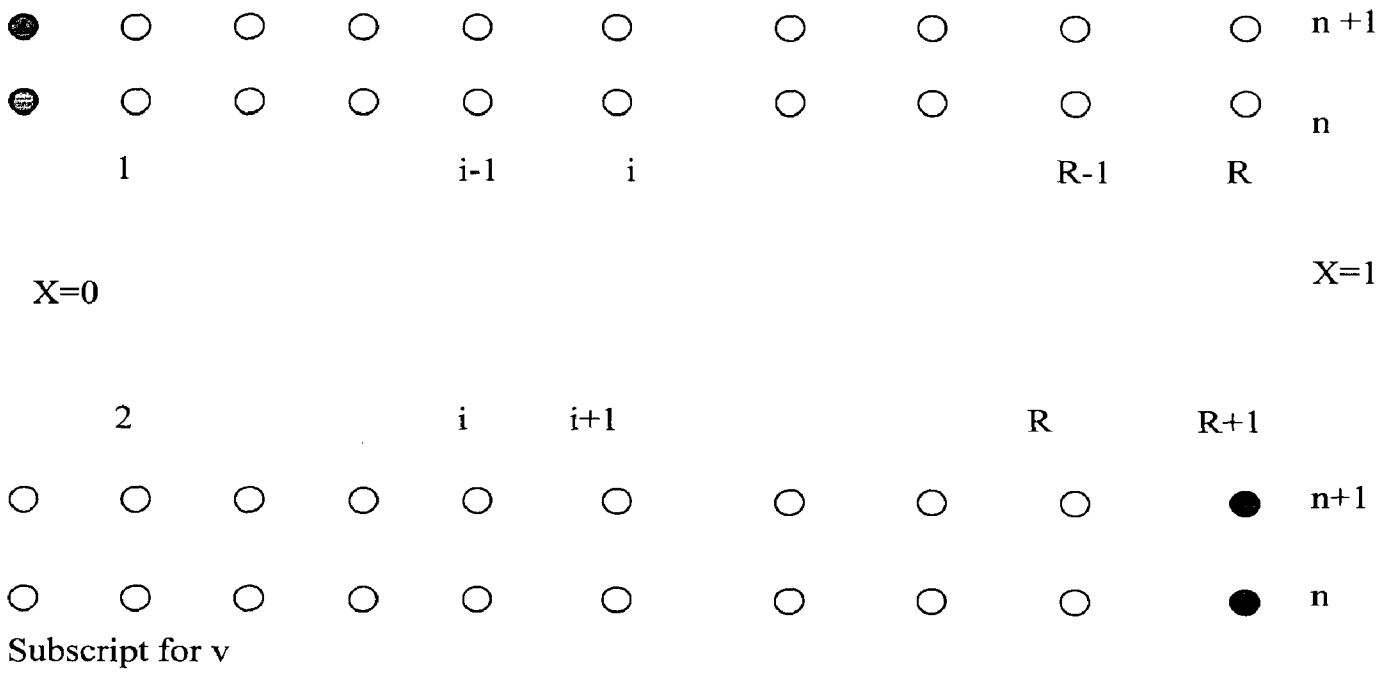


Fig. 3.4 Split points for coupled hyperbolic equations.

A single, representative point about which the finite difference analogs are written is shown; this point is designated by a cross. The coordinate of this point is $x_{i-1/2}, t_{n+1/2}$. As in the Crank-Nicholson equation, the space derivative at the n + ½ time level is approximated as the average of the space derivatives at the time levels for t_n and t_{n+1}. This analog is;

$$\left(\frac{\partial u}{\partial x}\right)_{i-1/2,n+1/2} \approx \frac{1}{2}\left(\frac{u_{i,n+1}-u_{i-1,n+1}}{\Delta x}+\frac{u_{i,n}-u_{i-1,n}}{\Delta x}\right) \qquad 3.3\text{-}18$$

This analog is second-order correct. In a similar manner, the time derivative at the *(i -1/2)* space position is approximated as the average of the time derivatives at the space positions for x_i and x_{i-1}. This derivative is also second-order correct, and it is given as;

$$\left(\frac{\partial u}{\partial t}\right)_{i-1/2,n+1/2} \approx \frac{1}{2}\left(\frac{u_{i,n+1}-u_{i,n}}{\Delta t}+\frac{u_{i-1,n+1}-u_{i-1,n}}{\Delta t}\right) \qquad 3.3\text{-}19$$

Both of these analogs contain values of the dependent variable at the same four points in the grid. Consequently, when they are substituted into equation 3.3-17, the resulting finite difference equation will contain values of u at these four points. This equation is

$$\left(\frac{b}{\Delta x}+\frac{1}{\Delta t}\right)u_{i,n+1}=\left(\frac{b}{\Delta x}-\frac{1}{\Delta t}\right)\left(u_{i-1,n+1}-u_{i,n}\right)+\left(\frac{b}{\Delta x}+\frac{1}{\Delta t}\right)u_{i-1,n} \qquad 3.3\text{-}20$$

This equation is stable for any ratio of $\Delta x / \Delta t$.

The centered difference analog of equation (3.3-17) is explicit. Although the centered difference equation is stable for any ratio of increments, the truncation error can be minimized by a proper choice of this ratio. In fact, there is no truncation error at all in equation (3.3-20) for the proper value of the ratio $\Delta x / \Delta t$. This value is b, the velocity of the fluid.

Numerical methods are not required to solve hyperbolic equations as simple as equation (3.3-17). For many physical problems, however, the material conserved is being added to or removed from the stream by a mechanism such as adsorption, heat transfer, or chemical reaction. For these equations a numerical solution is often desired, and the centered difference equation is very satisfactory.

A large number of physical problems are described by two or more coupled hyperbolic equations. Some of these arise from fluid-to-fluid heat exchangers, while others describe fixed-bed adsorbers. In these latter cases, there is no spatial derivative in the equation for the solid, since it is not moving. The centered difference equations have proved excellent for these problems also.

This is demonstrated by the simulation of a counter current heat exchanger where we experience two or three partial differential equations as model equations in the chapter 4.

3.4-1 Nonlinear Parabolic Equations:

One of the most important applications of numerical methods is solution to nonlinear, partial differential equations. Several methods for solving quasi-linear equations have been developed that result in linear finite difference equations which can be solved by existing algorithms and which do not involve excessive iteration. These methods are used

with finite difference analogs which are centered midway in time between the old and the new time steps. The general quasi-linear parabolic equation is

$$[a(u)]\frac{\partial^2 u}{\partial x^2}+[b(u)]\frac{\partial u}{\partial x}+[c(u)]u=\frac{\partial u}{\partial t} \qquad 3.4\text{-}1$$

In order that the equation to be quasi-linear, these coefficients must be functions of 'u'only and not of its derivatives. The general finite difference equation to be used for solving (3.4-1) is the Crank-Nicholson equation, and various methods of handling the nonlinear coefficients are used in conjunction with this method. For many quasi-linear equations, the boundary conditions are linear and can be handled by methods discussed previously.

In order to study the various methods to solve the quasi-linear parabolic equations, it is sufficient to consider the following partial differential equations with some non linear coefficients. All other non-linearities can be handled in a similar way.

$$[a(u)]\frac{\partial^2 u}{\partial x^2}=\frac{\partial u}{\partial t} \qquad 3.4\text{-}2$$

Following are some of the efficient methods to solve the above equation numerically.

3.4-2 Iteration using old value:

The Crank-Nicholson analogs to the derivatives are centered about the time level $t_{n+1/2}$.An analog to the nonlinear coefficient, *a(u)*, is required at this time level; and, if the resulting finite difference equations are to be linear, analog must not contain values of u at the time level t_{n+1}. The simplest analog is obtained by evaluating *a(u)* at the old time level and using for $a(u_{n+1/2})$. If the function *a(u)* does not change very rapidly with u, the solution to the resulting finite difference equations should be fairly near correct values. These values can be improved by next evaluating $a(u_{n+1/2})$ as $a\left[\left(u_n+u_{n+1}^{(1)}\right)/2\right]$ where u_{n+1}^{1} is the value obtained when $a(u_n)$ was used for $a(u_{n+1/2})$. The result of a continuation of this procedure is the following iterative equations:

$$\left[a\left(\frac{u_{i,n}+u_{i,n+1}^{(m)}}{2}\right)\right]\frac{1}{2}\Delta_x^2\left(u_{i,n}+u_{i,n+1}^{(m+1)}\right)=\frac{u_{i,n+1}^{(m+1)}-u_{i,n}}{\Delta t} \qquad 3.4\text{-}3$$

where ; $$\Delta_x^2 u_{i,n} = \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta x^2} \qquad 3.4\text{-}4$$

and $$u_{i,n+1}^{(0)} = u_{i,n} \qquad 3.4\text{-}5$$

Iteration is continued until $u_{i,n+1}^{(m+1)} = u_{i,n+1}^{(m)}$ within a predetermined tolerance. The resulting finite difference equations are linear in $u_{i,n+1}^{(m+1)}$ with the coefficient matrix being tridiagonal so that the Thomas algorithm can be used for the solution. This method should converge in three or four iterations. Notice, however, that the first analog to $a(u_{n+1/2})$ is completely forward, so that there will be some limitations on the size of the time increment to ensure stability.

3.4-3 Forward projection of Coefficient to Half Level in Time:

Douglas has devised a method for projecting the value of u to the half-time level for use in the nonlinear coefficients. This method has less stringent restrictions on the time-step size for stability and converges more rapidly than the method described above.

For this method the value of the dependent variable at the half level in time is obtained from a truncated Taylor series as follows:

$$u_{i,n+1/2} = u_{i,n} + \left(\frac{\partial u}{\partial t}\right)_{i,n}\left(\frac{\Delta t}{2}\right) + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,n}\frac{1}{2!}\left(\frac{\Delta t}{2}\right)^2 + \ldots + \qquad 3.4\text{-}6$$

The series in equation (3.4-6) is truncated after the second term to obtain a second-order-correct analog to $u_{i,n+1/2}$.The time derivative in this analog is then obtained from equation (3.4-2). The resulting finite difference analog for to be used in the nonlinear coefficient is;

$$u_{i,n+1/2} = u_{i,n} + \frac{\Delta t}{2}\left[a(u_{i,n})\right]\Delta_x^2 u_{i,n} \qquad 3.4\text{-}7$$

This value is then used in evaluating *a(u)* for use in the Crank-Nicolson analog to (3.4-2). The resulting finite difference equation can be written as;

$$\left\{a\left[u_{i,n} + \frac{\Delta t}{2}a(u_{i,n})\Delta_x^2 u_{i,n}\right]\right\}\frac{1}{2}\Delta_x^2\left(u_{i,n} + u_{i,n+1}\right) = \frac{u_{i,n+1} - u_{i,n}}{\Delta t} \qquad 3.4\text{-}8$$

The values of, $u_{i,n+1}$ which result from the application of equation (3.4-8), can be corrected by an iteration procedure similar to that described in the previous section. The need for such a correction can be determined experimentally and usually is found unnecessary. It might be more advisable to decrease the time step rather than to iterate.
The set of equations thus obtained from equation (3.4-8) are solved by the Thomas algorithm. Such procedure has been proved to be very efficient for the numerical solution of a number of quasi-linear, partial differential equations.

3.4-4 Backward Taylor series Projection:

An analog to $u_{i,n+1/2}$ for use in the nonlinear coefficients can also be obtained from a truncated Taylor series written about the time level $t_{n+1/2}$. In this case, the Taylor series is

$$u_{i,n} = u_{i,n+1/2} - \left(\frac{\partial u}{\partial t}\right)_{i,n+1/2} \frac{\Delta t}{2} + \left(\frac{\partial^2 u}{\partial t^2}\right)_{i,n+1/2} \frac{1}{2!}\left(\frac{\Delta t}{2}\right)^2 - \ldots - \qquad 3.4\text{-}9$$

Again, the Taylor series is truncated after the second term to obtain a second-order-correct analog, and the time derivative is obtained from (3.4-2).The resulting finite difference equations can be written as

$$[a(u_{i,n})]\Delta_x^2 u_{i,n+1/2} = \frac{u_{i,n+1/2} - u_{i,n}}{\Delta t/2} \qquad 3.4\text{-}10$$

This equation is not explicit, but the resulting coefficient matrix is tridiagonal. The values of $u_{i,n+1/2}$ can thus be readily obtained, and these are used in a Crank-Nicholson analog to (3.4-2), which is

$$[a(u_{i,n+1/2})]\frac{1}{2}\Delta_x^2(u_{i,n} + u_{i,n+1}) = \frac{u_{i,n+1} - u_{i,n}}{\Delta t} \qquad 3.4\text{-}11$$

The values obtained can be refined using the iteration scheme given in section 3.4-2

3.4-5 Centered Taylor Series Projection:

The centered Taylor series projection for $u_{i,n+1/2}$ can be obtained from the Taylor series for $u_{i,n+1/2}$ and $u_{i,n}$ written about the level $t_{n+1/2}$. In this case $a(u_{i,n})$ is used for $a(u_{i,n+1/2})$, and the resulting finite difference equation is;

$$[a(u_{i,n})]\frac{1}{2}\Delta_x^2(u_{i,n}+u_{i,n+1/2})=\frac{u_{i,n+1/2}-u_{i,n}}{\Delta t/2} \qquad 3.4\text{-}12$$

The values of $u_{i,n+1/2}$ obtained from this implicit equation are then used in (3.4-11) to obtain the values of $u_{i,n+1}$ .This method is called a predictor-corrector method by Douglas.

All the above discussed methods are implemented using MATLAB. The codes generated are used for developing the results. The results are given in the chapter 5 of this report.
The programs for the above methods are given in Appendix A4.
Further, all these methods discussed above can be used for the solution of non linear coupled (more than one) partial differential equations.
These equations are successfully used to simulate for the enthalpy and material balances describing a packed bed reactor. The equations can be easily coupled with the reaction term also. This is included in the simulation of an isothermal packed bed reactor with second order reaction as demonstrated in chapter 4. Computer program is given in the Appendix A4.

3.5 Nonlinear Hyperbolic Equations

Most of the methods described in the previous sections can be adapted for use with quasi-linear, hyperbolic equations. As one of the more important applications of numerical methods of solution of hyperbolic equations is to those with split boundary conditions, the methods will be illustrated on this system. Consider the equations

$$-b_1\left(\frac{\partial u}{\partial x}\right)-[c_1(u,v)](u-v)=\frac{\partial u}{\partial t}$$

$$+b_2\left(\frac{\partial v}{\partial x}\right)+[c_2(u,v)](u-v)=\frac{\partial v}{\partial t} \qquad 3.5\text{-}1$$

More generally, the coefficients of the space derivatives could also be functions of u and v, but the methods of handling the nonlinear terms would be the same.
The nomenclature and indexing for the centered difference method applied to a split boundary value problem is illustrated in Figure (). The derivatives are centered about the point $x_{i-1/2},t_{n+1/2}$ for the variable u and about the point $x_{i+1/2},t_{n+1/2}$ for the variable v.
Actually, this is the same point in space, but the x index is shifted for the two variables.

It is necessary then to evaluate $u_{i-1/2,n+1/2}$ and $v_{i+1/2,n+1/2}$ for determining the $c_1(u,v)$ and $c_2(u,v)$ in the centered difference equation.

A method of straight iteration using old values similar to that described in the previous section can be used. The average of the two values at the level t_n must be used for the starting value; as a result, the starting value of the nonlinear coefficient $c_1(u,v)$ is

$$c_1^0(u_{i-1/2,n+1/2}, v_{i+1/2,n+1/2}) = c_1\left[\frac{1}{2}(u_{i-1,n} + u_{i,n}), \frac{1}{2}(v_{i,n} + v_{i+1,n})\right] \qquad 3.5\text{-}2$$

The starting value for $c_2(u,v)$ is defined similarly. With c_1 and c_2 being known, the centered difference equations, such as (3.3-20) can be used to compute the values of $u_{i,n+1}$ and $v_{i,n+1}$ by one of the algorithms. These values can be further improved by an iterative process similar to that described in previous sections.

Forward Projection:

The forward projection method described in Section 3.4-3 is also readily adaptable to hyperbolic systems. The Taylor series of equation (3.4-6) is written about the point $x_{i-1/2}, t_n$ for the variable u, and it is also truncated after the second term of the series. The time derivative is evaluated by equation (3.5-1). The values of the variables $u_{i-1/2,n}$ and $v_{i+1/2,n}$ are evaluated as the average of the values at the two grid points on either side. These values are used both for evaluating the nonlinear coefficients and in the term $(u-v)$. The finite difference analog of the space derivative is second-order correct and is;

$$\left(\frac{\partial u}{\partial x}\right)_{i-1/2,n} = \frac{u_{i,n} - u_{i-1.n}}{\Delta x} \qquad 3.5\text{-}3$$

The resulting relation for $u_{i-1/2,n+1/2}$ is;

$$u_{i-1/2,n+1/2} = u_{i-1/2,n} - \frac{\Delta t}{2}\{b_1(u_{i,n} - u_{i-1,n})/\Delta x + [c_1(u_{i-1/2,n}, v_{i+1/2,n})](u_{i-1/2,n} - v_{i+1/2,n})\} \qquad 3.5\text{-}4$$

where

$$u_{i-1/2,n} = \frac{1}{2}(u_{i,n} + u_{i-1,n}) \qquad 3.5\text{-}5$$

Similar equation can be written for the dependant variable $v_{i+1/2,n}$.

Using the value from equation (3.5-4), the values of the nonlinear coefficients, $c_1(u,v)$ and $c_2(u,v)$, can be computed. These can then be used in the finite difference equations, and, and the values at the new time level, $u_{i,n+1}$ and $v_{i,n+1}$, can be computed.

The difference schemes for some of the examples involving the forward difference equations for non linear hyperbolic equations are discussed in the next chapter.

The program for such an example is given in Appendix A4.

Results and discussions are given for the particular example in the subsequent chapter.

3.6 Nonlinear Boundary Conditions

All the second-order methods described in the previous sections have been applied satisfactorily to all types of linear boundary conditions without any complications in the use of existing solution algorithms. These algorithms can be used on problems with nonlinear boundary conditions with little additional complication and with a negligible increase in computer time. The method of handling nonlinear boundary conditions can be described easily for a single parabolic equation so that the resulting finite difference equations are tridiagonal and can be solved by the Thomas algorithm.

Consider heat conduction in an insulated rod, which is described by

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}1$$

Let one end be held at a constant temperature and the other end receives heat by radiation from a constant-temperature source. These boundary conditions are

$$u = u_0$$

$$s\left(1-u^4\right) - \frac{\partial u}{\partial x} = 0 \qquad 3.6\text{-}1$$

with the initial condition

$$u = u_0 \qquad 3.6\text{-}2$$

In this case the temperature has been divided by the absolute temperature of the source of radiation in the normalization process. The parameter 's' in equation (3.6-1) is dimensionless and contains the cube of the source temperature, the Stephan-Boltzmann constant, and the thermal conductivity and length of the rod.

The Crank-Nicholson equation can be used for equation (3.3-1) as shown in the previous sections, and the boundary condition at $x = 0$ can be handled readily as done previously. Consider that the grid points are arranged as shown in the figure; 3.3.The backward analog to should be used at $x=1$ to ensure against oscillation of the computed values of u_R. This value can be obtained from equation (3.3-9) as

$$u_{R-1,n+1} + \left[-2 - \frac{(\Delta x^2)}{\Delta t}\right] u_{R,n+1} + u_{R+1,n+1} = \left[-\frac{(\Delta x)^2}{\Delta t}\right] u_{R,n} \qquad 3.6\text{-}3$$

The discrete analog to the boundary condition of equation is ;

$$s\left(1 - u_{R,n+1}^4\right) - \frac{u_{R+1,n+1} - u_{R-1,n+1}}{2(\Delta x)} = 0 \qquad 3.6\text{-}4$$

Or

$$u_{R+1,n+1} = u_{R-1,n+1} + 2s(\Delta x)\left(1 - u_{R,n+1}^4\right) \qquad 3.6\text{-}5$$

When the value of u from equation is substituted into equation as was done with the linear boundary condition, the resulting equation is nonlinear in $u_{R,n+1}$ and this is one of the unknowns to be evaluated.

If this equation were linear, it would have been written as the last equation of the tridiagonal system and would be

$$a_R u_{R-1} + b_R u_R = d_R \qquad 3.6\text{-}6$$

where a_R, b_R, and d_R are known constants. Since this equation is nonlinear, we write it as;

$$a_R u_{R-1} + b_R u_R = h + g u_R^4 \qquad 3.6\text{-}7$$

where a_R, b_R, h and g are constants. Now we write this as the last equation of the tridiagonal set of the difference equations. All the rest of the equations are linear. Thus the first half of the Thomas Algorithm, as given in the appendix, can be applied exactly as shown except for the computation of γ_R, which is also $u_{R,n+1}$. In place of this computation, coefficients in a nonlinear equation containing only $u_{R,n+1}$ are computed. This equation is;

$$u_{R,n+1} = \gamma_R = \frac{h - a_R \gamma_{R-1}}{\beta_R} + \frac{g}{\beta_R} u_{R,n+1}^4 \qquad 3.6\text{-}8$$

or

$$u_{R,n+1} = p + qu_{R,n+1}^4 \qquad 3.6\text{-}9$$

where p and q are constants as defined in equation. This equation can be solved by a number of techniques. Once the value of is $u_{R,n+1}$ determined, the back half of the Thomas algorithm can be used to obtain the rest of $u_{i,n+1}$.

Hence the non linear boundary can be handled with a negligible increase in computation time. Here we determine the value of $u_{R,n+1}$ from non linear relationship given by equation using the method of false position, or ***regula falsi*** method. This method gives assured convergence. The algorithm for the *regula falsi* method is given in Appendix A5. The entire difference equations are developed and implemented using the program code given in Appendix A4.

Similarly we can handle the situations with non linear boundary conditions at both ends. In this case however, we need to first consider initial assumptions at one boundary and carry the forward calculations as was done with the case of single non linear boundary conditions and further the back calculations are carried to refine the initial assumptions using a similar analysis as shown above.

The results obtained are discussed in the chapter 5.

CHAPTER 4

ENGINEERING APPLICATIONS

Applications to Engineering Problems:

In this chapter we apply the numerical schemes developed in the previous chapters to a number of engineering problems. The examples considered are quite general, but they can be used to simulate a number of systems where such types of equations are found. The fundamental difference equations are developed to treat a particular type of system with a set of initial and boundary conditions. The computer programs given in Appendix A4 are based on these numerical schemes. A reference can be taken from the Appendix for various algorithms used to solve the linear system of equations developed here in this chapter.

The results obtained from these schemes are discussed in the next chapter.

4.1 Heat Flow in Insulated Rod:

The heat flow in an insulated rod is modeled by Diffusion equation as given below. The equation may be having constant parameters multiplied on both sides of the equation.

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}1$$

$$\left.\begin{aligned} u(0,t) &= 0 \\ u(1,t) &= 1 \end{aligned}\right\} all\ \text{t}$$

$$u(x,0) = 0 \ \ all\ x.$$

This equation is in dimensionless form and all the variables are normalized.

Following are the resulting difference equations when the respective analogs for the partial differentials are substituted.

for $2 \le i \le R-2$

$$u_{i-1,n+1} + \left[-2 - 2\frac{(\Delta x)^2}{\Delta t}\right] u_{i,n+1} + u_{i+1,n+1} = -u_{i-1,n} - u_{i+1,n} + \left[2 - 2\frac{(\Delta x)^2}{\Delta t}\right] u_{i,n} \qquad 4.1\text{-}1$$

for $i = 1$

$$\left[-2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{1,n+1}+u_{2,n+1}=\left[2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{1,n}-u_{2,n} \qquad 4.1\text{-}2$$

for $i = R-1$

$$u_{R-2,n+1}+\left[-2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{R-1,n+1}=-u_{R-2,n}+\left[2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{R-1,n}-2 \qquad 4.1\text{-}3$$

These result in a set of linear algebraic equations that can be readily solved using Thomas Algorithm A2 for the estimation of dependant variable.

4.2 Heat Flow in Tapered Rod

Here we consider the heat flow in a tapered rod with some moving boundary conditions. The equation consists of both Diffusive and Convective terms. Here the term 'p' is dimensionless combination of parameters that describe geometry and heat transfer characteristics.

$$\frac{\partial^2 u}{\partial x^2}+\frac{2}{x}\left(\frac{\partial u}{\partial x}-pu\right)=\frac{\partial u}{\partial t} \qquad 4.2\text{-}1$$

subject to;

$u(1,t)=1$

$$\frac{\partial u}{\partial x}-pu=0 \quad \text{at } x=0 \text{ all } t \qquad 4.2\text{-}2$$

$u(x,0)=0$ all x

We define: $x_i=(i-1/2)\Delta x$ and the step size is given as; $\Delta x = 1/R$

Next we use the Crank-Nicholson method to derive the difference equations as given below.

for $2 \le i \le (R-1)$

$$\left(\frac{2i-3}{2i-1}\right)u_{i-1,n+1}+\left[-2-\frac{4p(\Delta x)}{2i-1}-\frac{2(\Delta x)^2}{\Delta t}\right]u_{i,n+1}+\left(\frac{2i+1}{2i-1}\right)u_{i+1,n+1}=$$

$$-\left(\frac{2i-3}{2i-1}\right)u_{i-1,n}-\left(\frac{2i+1}{2i-1}\right)u_{i+1,n}+\left[2+\frac{4p(\Delta x)}{2i-1}-\frac{2(\Delta x)^2}{\Delta t}\right]u_{i,n} \qquad 4.2\text{-}3$$

for $i=1$

$$\left[-2-\frac{4p(\Delta x)}{2i-1}-\frac{2-p(\Delta x)}{2+p(\Delta x)}-\frac{2(\Delta x)^2}{\Delta t}\right]u1_{1,n+1}+3u_{2,n+1}=$$

$$\left[2+\frac{4p(\Delta x)}{2+p(\Delta x)}-\frac{2-p(\Delta x)}{2+p(\Delta x)}-\frac{2(\Delta x)^2}{\Delta t}\right]u_{1,n}-3u_{2,n}$$

4.2-4

for $i=R$

$$\left(\frac{2R-3}{2R-1}\right)u_{R-1,n+1}+\left[-2-\frac{4p(\Delta x)}{2R-1}-\frac{2(\Delta x)^2}{\Delta t}-\frac{2R+1}{2R-1}\right]u_{R,n+1}=$$

$$-\left(\frac{2R-3}{2R-1}\right)u_{R-1,n}-4\left(\frac{2R+1}{2R-1}\right)+\left[2+\frac{4p(\Delta x)}{2R-1}+\frac{2R+1}{2R-1}-\frac{2(\Delta x)^2}{\Delta t}\right]u_{R,n}$$

4.2-5

This set of equations again constructs a tridiagonal system of linear equations that is readily solved using Thomas Algorithm as given in Appendix A2. The results obtained are explained in chapter 5.

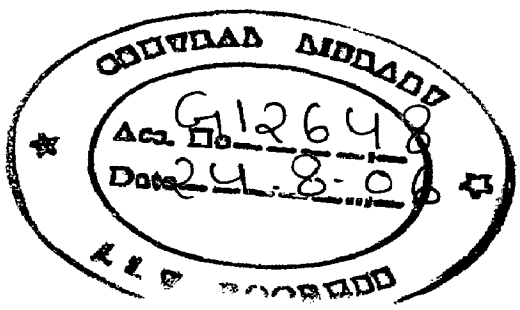
4.3 Countercurrent Heat Exchanger

The equations that model the transfer of heat in an unsteady state countercurrent heat exchanger are purely convective as given below. These equations are a result of heat balance of fluids (cold and hot) flowing in the countercurrent heat exchanger. Here 'u' is the dimensionless temperature. The parameters b_1 and b_2 are dimensionless velocities of the two streams and the parameters c_1 and c_2 contain the coefficient of heat transfer between the two streams.

$$-b_1\left(\frac{\partial u}{\partial x}\right)-c_1(u-v)=\left(\frac{\partial u}{\partial t}\right)$$
$$b_2\left(\frac{\partial v}{\partial x}\right)+c_2(u-v)=\left(\frac{\partial v}{\partial t}\right) \qquad 4.3\text{-}1$$

Subject to;

$$u(x,0)=0 \quad x>0 \qquad 4.3\text{-}2$$



$$\begin{aligned} &u(0,0)=1 \\ &v(x,0)=0 \\ &u(0,t)=1 \\ &v(1,t)=0 \end{aligned} \qquad all\ t \qquad 4.3\text{-}3$$

We define $x_i = i(\Delta x)$ for u and $x_i = (i-1)(\Delta x)$ for v.

The difference equations generated as a result of application of difference analogs given in the previous chapter (section 3.3-3) are;

for $2 \le i \le R-1$

$$\left(-\frac{b_1}{\Delta x}+\frac{c_1}{2}+\frac{1}{\Delta t}\right)u_{i-1,n+1}+\left(\frac{b_1}{\Delta x}+\frac{c_1}{2}+\frac{1}{\Delta t}\right)u_{i,n+1}+\left(-\frac{c_1}{2}\right)v_{i,n+1}+\left(-\frac{c_1}{2}\right)v_{i+1,n+1}=$$

$$\left(\frac{b_1}{\Delta x}-\frac{c_1}{2}+\frac{1}{\Delta t}\right)u_{i-1,n}+\left(-\frac{b_1}{\Delta x}-\frac{c_1}{2}+\frac{1}{\Delta t}\right)u_{i,n}+\frac{c_1}{2}\left(v_{i+1,n}+v_{i,n}\right) \qquad 4.3\text{-}4$$

and

$$\left(\frac{c_2}{2}\right)u_{i-1,n,n+1}+\left(\frac{c_2}{2}\right)u_{i,n+1}+\left(-\frac{b_2}{\Delta x}-\frac{c_2}{2}-\frac{1}{\Delta t}\right)v_{i,n+1}+\left(\frac{b_2}{\Delta x}-\frac{c_2}{2}-\frac{1}{\Delta t}\right)v_{i+1,n+1}=$$

$$-\frac{c_2}{2}\left(u_{i,n}+u_{i-1,n}\right)+\left(\frac{b_2}{\Delta x}+\frac{c_2}{2}-\frac{1}{\Delta t}\right)v_{i,n}+\left(-\frac{b_2}{\Delta x}+\frac{c_2}{2}-\frac{1}{\Delta t}\right)v_{i+1,n} \qquad 4.3\text{-}5$$

The boundary condition equations are found by substituting their values in equations for *i=1* and *i=R*. The resulting equations form a bi-tridiagonal coefficient matrix that is solved using the Algorithm given in Appendix A3. Results are discussed in chapter 5.

4.4 Isothermal flow reactor

The model equation given below is a Convection Diffusion equation usually describing the flow of mass in a variety of reactors such as packed bed catalytic reactor and Adsorber etc.The equation also contains a source term to account for the reaction. The dependant variable u, is the dimensionless concentration.

$$\frac{\partial^2 u}{\partial x^2}-s\frac{\partial u}{\partial x}-ru^2=\frac{\partial u}{\partial t} \qquad 4.4\text{-}1$$

Subject to the boundary conditions;

$$u(x,0) = 0 \ \textit{for all } x \qquad 4.4\text{-}2$$

$$\frac{\partial u}{\partial x} = 0; \ \textit{at } x=1 \ , \ \textit{all } t \qquad 4.4\text{-}3$$

$$\frac{\partial u}{\partial x} + s(1-u) = 0 \ \textit{ at } x=0, \ \textit{all } t \qquad 4.4\text{-}4$$

The equation is a non linear partial differential equation. We apply the Crank-Nicholson equation. The resulting finite difference equations are;

for $2 \le i \le R-1$

$$\left[1+\frac{s(\Delta x)}{2}\right]u_{i-1,n+1} + \left[-2-r(\Delta x)^2 u_{i,n+1/2} - \frac{2(\Delta x)^2}{\Delta t}\right]u_{i,n+1} + \left[1-\frac{s(\Delta x)}{2}\right]u_{i+1,n+1} =$$

$$-\left[1+\frac{s(\Delta x)}{2}\right]u_{i-1,n} - \left[1-\frac{s(\Delta x)}{2}\right]u_{i+1,n} + \left[2+r(\Delta x)^2 u_{i,n+1/2} - \frac{2(\Delta x)^2}{\Delta t}\right]u_{i,n}$$

4.4-5

With

$$u_{i,n+1/2} = u_{i,n} + \frac{\Delta t}{2(\Delta x)^2}\left\{\left[1+\frac{s(\Delta x)}{2}\right]u_{i-1,n} + \left[1-\frac{s(\Delta x)}{2}\right]u_{i+1,n} + \left[2+r(\Delta x)^2 u_{i,n}\right]u_{i,n}\right\}$$

4.4-6

for $i=1$

$$\left\{-r(\Delta x)^2 u_{1,n+1/2} - \frac{2(\Delta x)^2}{\Delta t} - \left[1+\frac{s(\Delta x)}{2}\right]\right\}u_{1,n+1} + \left[1-\frac{s(\Delta x)}{2}\right]u_{2,n+1} =$$

$$-\left[1-\frac{s(\Delta x)}{2}\right]u_{2,n} + \left\{r(\Delta x)^2 u_{1,n+1/2} - \frac{2(\Delta x)^2}{\Delta t} + \left[\left[1+\frac{s(\Delta x)}{2}\right]\right]\right\}u_{1,n} - 2s(\Delta x)$$

4.4-7

with

$$u_{1,n+1/2} = u_{1,n} + \frac{\Delta t}{2\Delta x^2}\left\{\left[1-\frac{s(\Delta x)}{2}\right]u_{2,n} + s(\Delta x) - \left[1+\frac{s(\Delta x)}{2} + r(\Delta x)^2 u_{1,n}\right]u_{1,n}\right\}$$

4.4-8

For $i=R$

$$\left[1+\frac{s(\Delta x)}{2}\right]u_{R-1,n}+\left\{-r(\Delta x)^2 u_{R,n+1/2}-\frac{2(\Delta x)^2}{\Delta t}-\left[1+\frac{s(\Delta x)}{2}\right]\right\}u_{R,n+1}=$$
$$-\left[1+\frac{s(\Delta x)}{2}\right]u_{R-1,n}+\left\{r(\Delta x)^2 u_{R,n+1/2}-\frac{2(\Delta x)^2}{\Delta t}+\left[1+\frac{s(\Delta x)}{2}\right]\right\}u_{R,n+1} \qquad 4.4\text{-}9$$

With;

$$u_{R,n+1/2}=u_{R,n}+\frac{\Delta t}{2(\Delta x)^2}\left\{\left[1+\frac{s(\Delta x)}{2}\right]u_{R-1,n}-\left[1+\frac{s(\Delta x)}{2}+r(\Delta x)^2 u_{R,n}\right]u_{R,n}\right\} \qquad 4.4\text{-}10$$

The resulting equations here again are tridiagonal in nature and are readily solved. This example illustrates the simplification of a number of reactor models. The results and the concentration profiles are discussed in chapter 5.

4.5 Parabolic equation with small dispersion coefficient

This problem illustrates the effect of dispersion coefficient on a Convection Diffusion equation and shows the numerical instability introduced by the Convective term

The differential equation to be considered is;

$$\frac{\partial^2 u}{\partial x^2}-b\frac{\partial u}{\partial x}=\frac{\partial u}{\partial t} \qquad 4.5\text{-}1$$

The above equation describes the one dimensional flow of a fluid with dispersion or diffusion. Here the parameter b is the ratio of the velocity of flow to the dispersion coefficient.

When the dispersion coefficient is small, or the parameter b is very large, the adjoining term becomes the controlling term on the left side. Under such conditions the numerical solutions at a given time interval oscillate around the true curve. We firstly develop the difference analogues for the partial differential equation.

By Crank Nicholson analog;

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,n+1/2}\approx\frac{1}{2}\left[\frac{u_{i+1,n}-2u_{i,n}+u_{i-1,n}}{(\Delta x)^2}+\frac{u_{i+1,n+1}-2u_{i,n+1}+u_{i-1,n+1}}{(\Delta x)^2}\right] \qquad 4.5\text{-}2$$

and $$\left(\frac{\partial u}{\partial t}\right)_{i,n+1/2} = \frac{u_{i,n+1} - u_{i,n}}{(\Delta t)} - \left(\frac{\partial^3 u}{\partial t^3}\right)_{i,n+1/2} \frac{(\Delta t)^2}{24} - \ldots - \qquad 4.5\text{-}3$$

$$\left(\frac{\partial u}{\partial x}\right)_{i-1/2,n+1/2} \approx \frac{1}{2}\left(\frac{u_{i,n+1} - u_{i-1,n+1}}{\Delta x} + \frac{u_{i,n} - u_{i-1,n}}{\Delta x}\right) \qquad 4.5\text{-}4$$

Substituting these analogues in the equation 4.5-1 we get the following difference equation

$$(1 + b\Delta x)u_{i-1,n+1} + \left(-2 - b\Delta x - \frac{2\Delta x^2}{\Delta t}\right)u_{i,n+1} + u_{i+1,n+1} =$$

$$(-b\Delta x - 1)u_{i-1,n} + \left(2 + b\Delta x - \frac{2\Delta x^2}{\Delta t}\right)u_{i,n} - u_{i+1,n} \qquad 4.5\text{-}5$$

This numerical scheme can be tested for different values of b and Δx to study the convective effect. The results obtained for such a system, as a result of implementation of above scheme are given in chapter 5.

4.6 Heat flow in a rod with heat received by radiation at one end

This application explains the methods to handle the non linear boundary conditions. For understanding purpose we may consider the simple Diffusion equation with non linear boundary arising as a result of radiative heat transfer. The equation is;

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 3.3\text{-}1$$

Subject to;

$u(x,0) = u_0$ *all* x.

$u(0,t) = u_0$ *all* x

$$s(1 - u^4) - \frac{\partial u}{\partial x} = 0 \quad \text{at } x=1 \text{ and } t \qquad 4.6\text{-}1$$

Let $x_i = i(\Delta x)$

The resulting finite difference equations are;

$$u_{i-1,n+1} + \left[-2 - 2\frac{\Delta x^2}{\Delta t}\right]u_{i,n+1} + u_{i+1,n+1} = -u_{i-1,n} - u_{i+1,n} + \left[2 - 2\frac{(\Delta x)^2}{\Delta t}\right]u_{i,n} \qquad 4.6\text{-}2$$

for $i=1$

$$\left[-2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{1,n+1}+u_{2,n+1}=-u_{2,n}-2u_0+\left[2-2\frac{(\Delta x)^2}{\Delta t}\right]u_{1,n} \qquad 4.6\text{-}3$$

For $i=R$

Here we use the backward difference equation to prevent oscillation in the values of the dependant variable.

$$2u_{19,n+1}+\left[-2-\frac{(\Delta x)^2}{\Delta t}\right]u_{20,n+1}=\left[-\frac{(\Delta x)^2}{\Delta t}u_{20,n}-2s(\Delta x)\right]+2s(\Delta x)u_{20,n+1}^4 \qquad 4.6\text{-}4$$

The first half of the Thomas Algorithm is implemented till the computation of β_{R-1} and γ_{R-1}. Hereafter we follow the method discussed in previous chapter (section 3.6).

$$\beta_R=-2-\frac{(\Delta x)^2}{\Delta t}-2/\beta_{R-1} \qquad 4.6\text{-}5$$

then, we have;

$$u_{R,n+1}=\gamma_R=(d_R-a_R\gamma_{R=1})/\beta_R \qquad 4.6\text{-}6$$

$$=\left[-\frac{(\Delta x)^2}{\Delta t}u_{R,n}-2s(\Delta x)-a_R\gamma_{R-1}\right]/\beta_R+\left[2s(\Delta x)/\beta_R\right]u_{R,n+1}^4 \qquad 4.6\text{-}7$$

Comparing equation with equation , we get;

$$p=\left[-\frac{(\Delta x)^2}{\Delta t}u_{R,n}-2s(\Delta x)-a_R\gamma_{R-1}\right]\Big/\beta_R \quad \text{and;} \qquad 4.6\text{-}8$$

$$q=2s(\Delta x)/\beta_R \qquad 4.6\text{-}9$$

The function to determine the value of $u_{R,n+1}$ using *regula falsi* method is;

$$f=p+qu_{R,n+1}^4-u_{R,n+1}=0 \qquad 4.6\text{-}10$$

Thus we obtain the values of $u_{R,n+1}$. Thereafter we can implement the remaining half of the Thomas Algorithm for the estimation of the all the remaining values of $u_{i,n+1}$.

Thus the nonlinear boundary conditions are easily handled and their solutions are numerically comparable to that obtained by FEM (Finite Element Method) as will be shown in the next chapter.

4.7 Shrinking core model:

The model equations developed by this method are used for the concentration analysis of

Catalytic packed bed reactors, slurry reactors, and bubbling fluidized bed reactors.

The model equation is given as follows. The equation holds true for first order reaction.

$$\frac{d^2\Psi}{d\lambda^2}+\frac{2}{\lambda}\frac{d\Psi}{d\lambda}-\Theta^2\Psi=0 \qquad 4.7\text{-}1$$

with the boundary conditions;

$$\Psi=1 \quad \text{at } \lambda=1$$

and $$\Psi=finite \quad \text{at } \lambda=0 \qquad 4.7\text{-}2$$

here $\Theta=R\sqrt{\frac{k_1\rho_c S_a}{D_e}}$ = Thieles modulus and is a dimensionless term.

The equation is derived for steady state conditions. Also Ψ is the dimensionless concentration. We derive the difference equations for this differential.

Using the difference approximations for second and first derivatives as mentioned in the previous chapter, the following difference equations are formulated.

$$\frac{u_{i+1}-2u_i+u_{i-1}}{\Delta x^2}+\frac{2}{i(\Delta x)}\frac{u_{i+1}-u_{i-1}}{2\Delta x}-\Theta^2u_i=0 \qquad 4.7\text{-}3$$

This equation can be rearranged as;

$$\left[1-\frac{1}{i}\right]u_{i-1}+\left[-2-\Theta^2\left(\Delta x^2\right)\right]u_i+\left[1+\frac{1}{i}\right]u_{i+1}=0 \qquad 4.7\text{-}4$$

The above difference equation is implemented on MATLAB and the results obtained are compared with the theoretical result which is given by;

$$\psi=\frac{1}{\lambda}\left(\sinh\Theta\lambda/\sinh\Theta\right) \qquad 4.7\text{-}5$$

The detailed discussions are carried in the next chapter.

CHAPTER 5

RESULTS AND DISCUSSIONS

In this chapter, the numerical results obtained as a result of implementation of the various finite difference equations for a number of situations are presented. The finite difference equations so developed in the previous chapters are to be tested with a number of alterations to prove their efficiency and accuracy in handling the Convection Diffusion equations.

All the problems discussed assume that the equations derived are normalized and dimensionless.

The results are obtained from the different MATLAB codes generated in cohesion with the formulated difference equations (as given in the Appendix A4).

The results are checked for a number of parameters such as Convergence, Accuracy, Sensitivity, Stability and Error analysis. The various engineering applications that are discussed in the previous chapter are also included for result analysis in a serial-wise manner. The program codes are so written that they can be run for a number of variations in step sizes and other parameters. Most of the times, the accuracy of the results are compared with those obtained from 'PDEPE tool box' of the MATLAB software.

This tool box is based on the 'Finite Element Method' for partial differential equations.

The programs are written in such a way that the user can enter the step sizes and ascertain the boundary and initial conditions. The script files written in MATLAB simulate the given problem with these inputs and calculate the results. The files are so written that the user can watch the value of the dependant variable at a particular instant of time and space. There is provision of plotting the dependant variable against the time and space as variables at the user prompt. At the back end, the step sizes and parameters so entered are passed to the programs written with PDEPE tool box and thereafter are compared with those obtained from the simulation of difference equation. This allows to carry out the error examinations.

5.1 Heat conduction in uninsulated tapered rod

Consider the problem of heat conduction in an uninsulated, tapered rod. The differential equation (Convection Diffusion) for steady state conditions is;

$$\frac{d^2u}{dx^2} + \frac{2}{q+x}\frac{du}{dx} - \frac{2p}{q+x}u = 0 \qquad 5.1\text{-}1$$

The length variable, x and the temperature variable, u, are dimensionless and normalized.

The constants 'q' and 'p' are dimensionless combination of parameters which describe the geometry and heat transfer characteristics of the rod. These are defined as;

$$p = \frac{hL}{k}\sqrt{1+\frac{4}{f^2}} \qquad \text{and} \qquad q = \frac{D_0}{fL}$$

where L is the length of the rod.

h is the coefficient of heat transfer between the rod and surroundings.

k is the thermal conductivity of the rod.

f and D_0 define the diameter, D, of the rod by $D = D0 + fLx$.

Consider the simplest boundary conditions as:

$$u(0) = 0 \quad \text{and} \quad u(1)=1 \qquad 5.1\text{-}2$$

The finite difference approximations for this equation are developed in the section 3.2

On implementing the program written in the Appendix A4 we get the following results.

For $q=1$; $p=80$ and step size of $\Delta x = 0.1$, we get the following plot as shown in figure 5.1

The figures show that the results are converging towards 1 and are not sensitive to the step size (no oscillations). For error analysis we compare the numerical value of the dependant variable with the analytical value at $x=0.9$.

$$\%error = \frac{Analyticalvalue - Numericalvalue}{Analyticalvalue} \times 100 \qquad 5.1\text{-}3$$

$$\%error = \frac{0.419208 - 0.18934849}{0.419208} \times 100 = 54.83\%$$

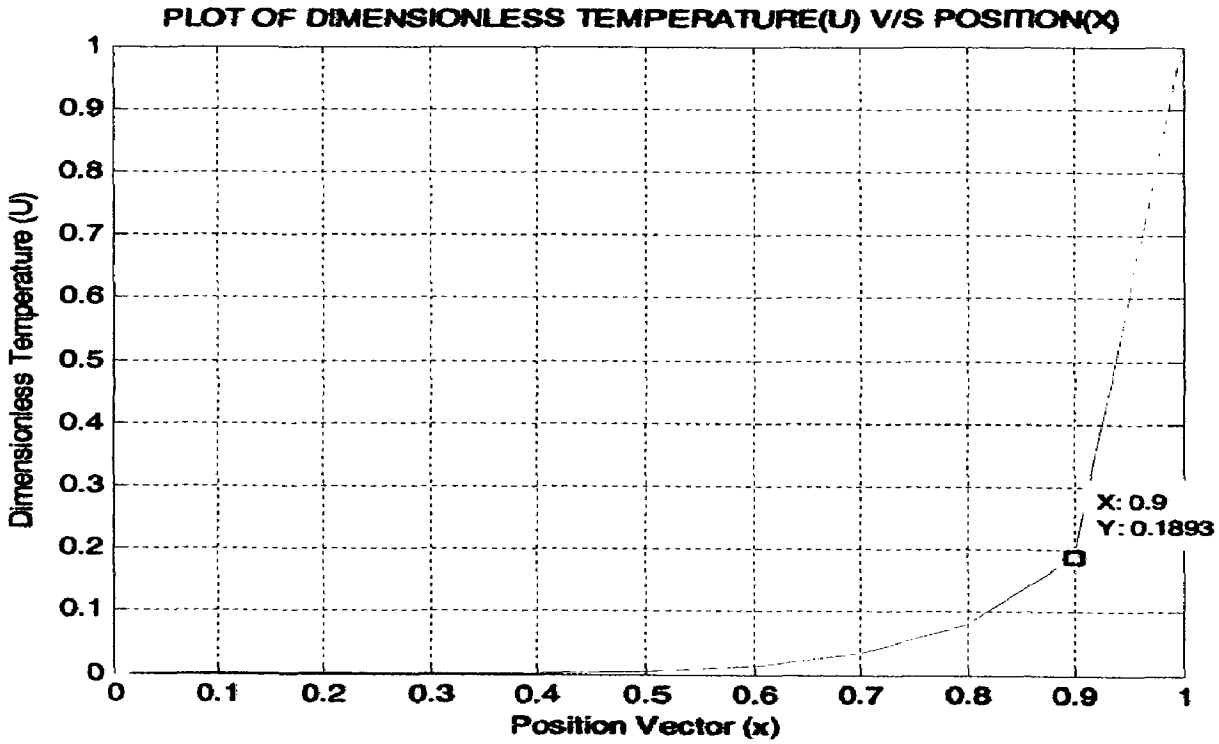


Figure: 5.1 Heat conduction in an uninsulated rod Δx =0.1

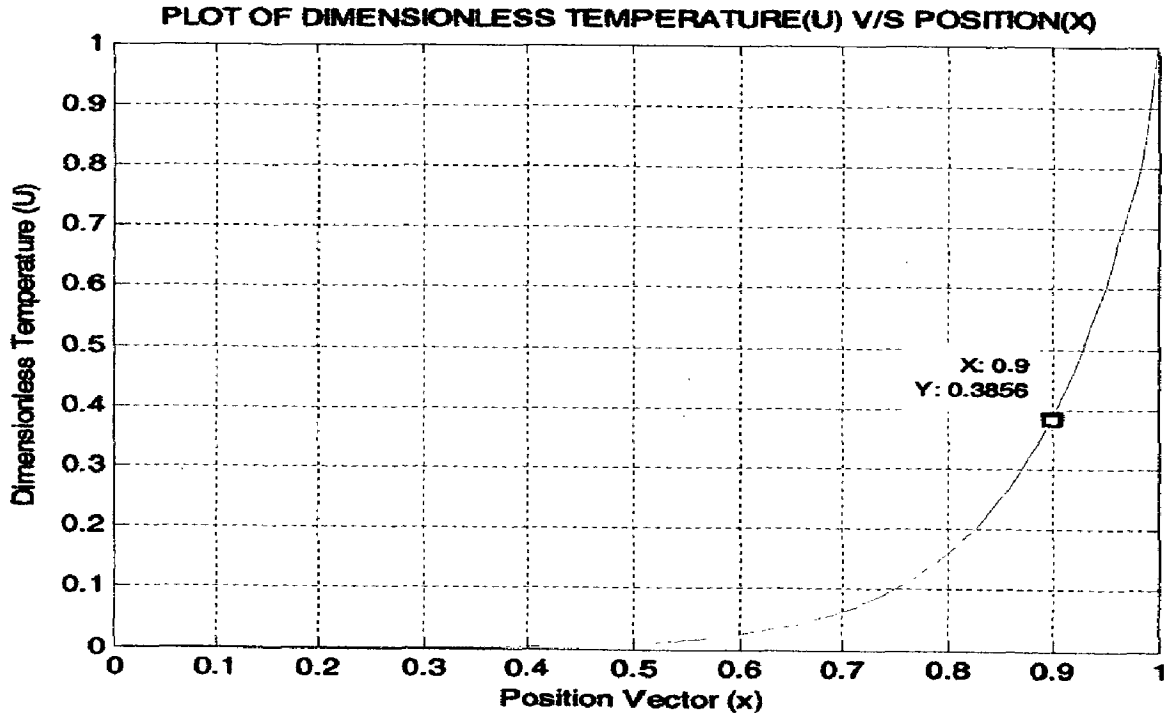


Figure: 5.2 Heat conduction in an uninsulated rod. Δx =0.01

The analytical value for $x = 0.9$ *is* 0.419208.

For $\Delta x = 0.01$

$$\%error = \frac{0.419208 - 0.3856}{0.419208} \times 100 = 8.0153\%$$

Further on running the code for $\Delta x = 0.001$ and 0.0001 we get 0.69 % and 0.0954% error respectively. Thus the difference scheme is efficiently representing the given differential equation.

b) Consider the same differential equation discussed above with a changed boundary condition as;

$$\frac{du}{dx}(0) = g \qquad 5.1\text{-}4$$

We take trials for $g = -10$. The resulting figures are shown in the figures 5.3 and 5.4 We see that the solution is converging towards 1 even with this case. It is noticed that the solutions are converging and the values for $\Delta x = 0.01$ and $\Delta x = 0.001$ are comparable and thus the results obtained are accurate with a truncation error.

c) Consider the moving type of boundary conditions of the type for the same differential equation

$$\frac{du}{dx}(0) - Hu(0) = -g \qquad 5.1\text{-}5$$

where; $H = \frac{hL}{k} = Nusseltnumber$

The figure obtained on inputting the values $q=2.5\ p=80; g=10; H=300$ for $\Delta x = 0.001$ is shown in figure 5.5 , for $\Delta x = 0.005$ in figure 5.6. The obtained results are compared at $x=0.6$ and found to be satisfactory. Thus the developed schemes are working well with moving boundary conditions and give comparable values of the dependant variable u.

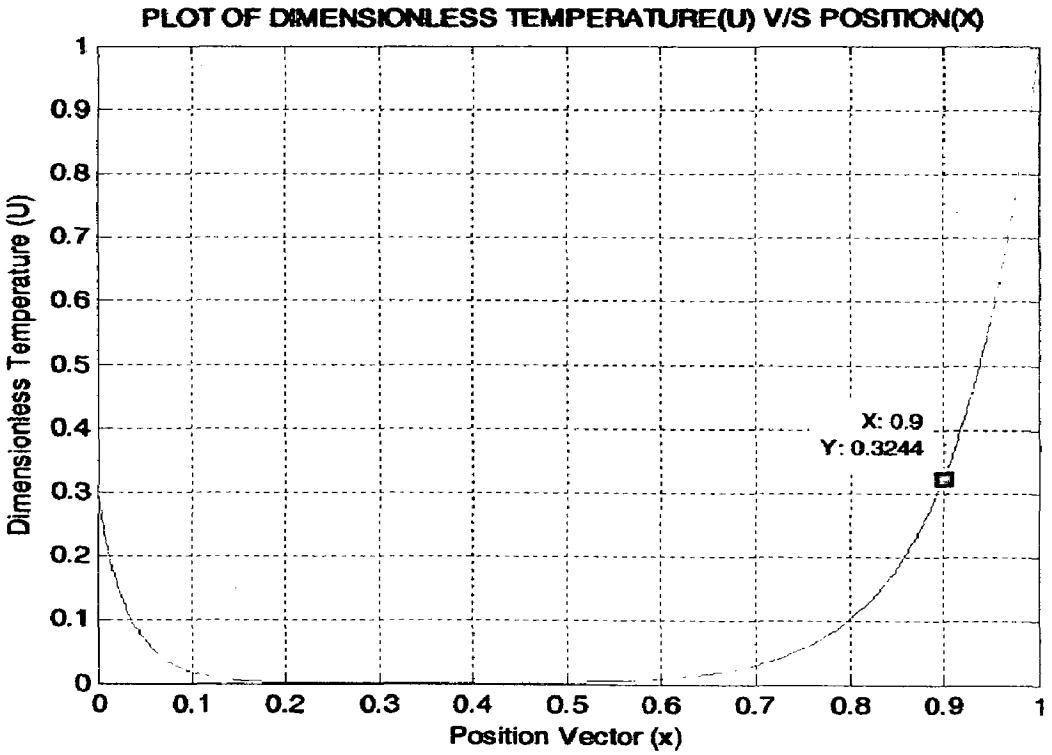


Figure: 5.3 Heat conduction in an uninsulated rod Δx =0.01

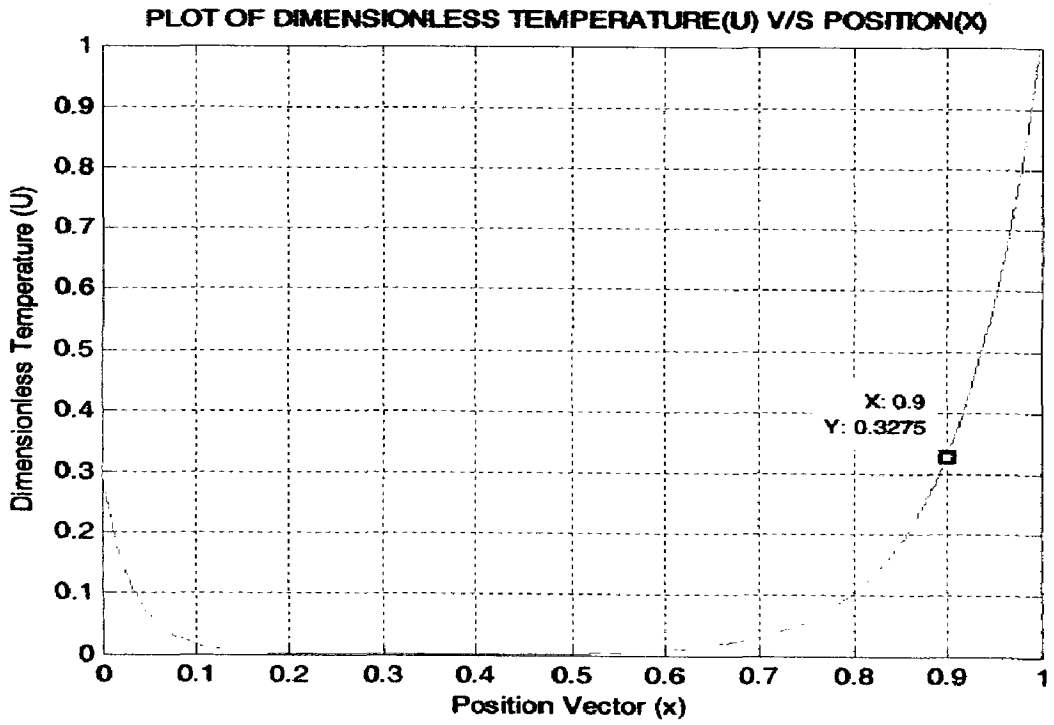


Figure: 5.4 Heat conduction in an uninsulated rod Δx =0.001

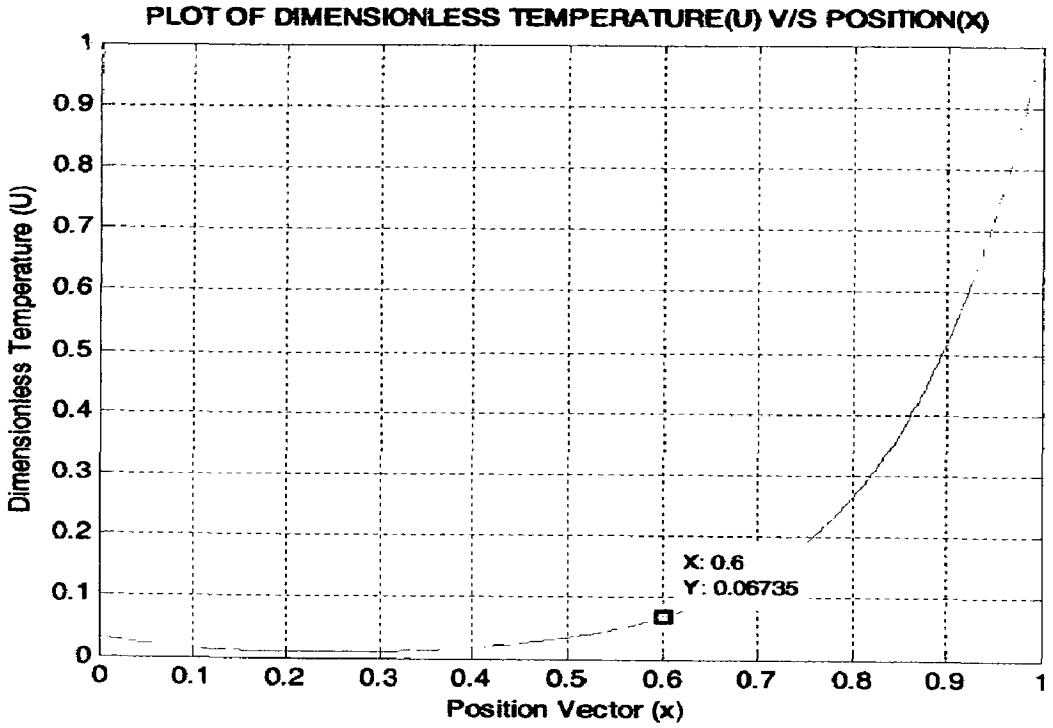


Figure: 5.5 Heat conduction in an uninsulated rod Δx =0.001

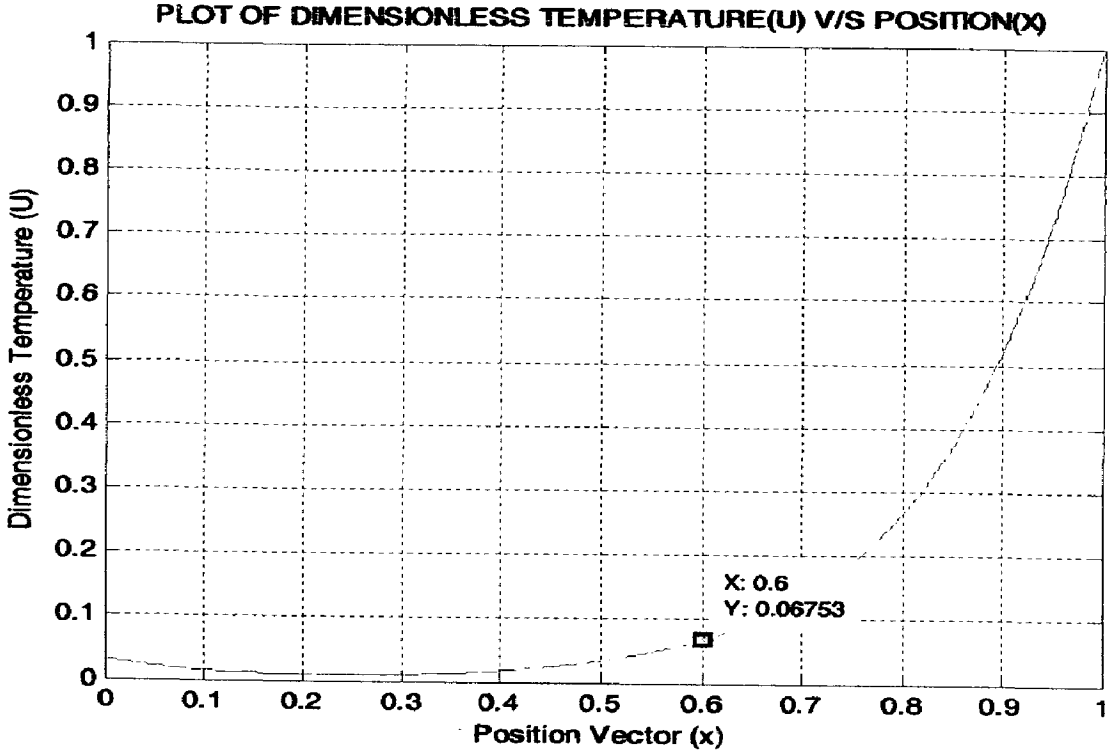


Figure: 5.6 Heat conduction in an uninsulated rod Δx =0.005

5.2 Diffusion Equation

Consider the one dimensional diffusion equation. The equation is purely parabolic.

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 5.2\text{-}1$$

the boundary conditions are given as,

$$\left.\begin{aligned} u(0,t) &= 0 \\ u(1,t) &= 1 \end{aligned}\right\} all\ t \qquad 5.2\text{-}2$$

With an initial condition given as;

$$u(x,0) = 0;\quad x < 1 \qquad 5.2\text{-}3$$

We have already developed the difference equations for the above Diffusion equation via three methods. We present them serially.

a) Forward Differences:

The difference equations using this method are given in sections 3.3-1a. As already mentioned this is an explicit method and imposes a criterion on step sizes for stability of the solution. The stability criterion is $\Delta t / \Delta x^2 = 0.5$

A plot to illustrate violated stability condition is shown in figure 5.7. Here we used $\Delta x = 0.1 = \Delta t$.

Next we use the step size along time dimension that is well-matched with the stability criterion. For a step size of $\Delta x = 0.1$, this comes out to be $\Delta t = 0.005$.

The subsequent plot obtained is shown in figure 5.8.

The overall percent error obtained in the method when compared with the PDEPE tool box solution is: 2.682149%. We use the same formula for evaluating the percent error as given in equation 5.1-3 except for the analytical term is replaced by the one obtained from PDEPE tool box. The error initially is too high, because the value of the dependant variable changes very slowly, i.e. at successive grid points for every time interval. The error can be further reduced, but at the expense of huge computational times.

The figure 5.9 is obtained using the PDEPE tool box of MATLAB and figure 5.10 is the result from finite difference schemes. The profiles from the figures are comparable.

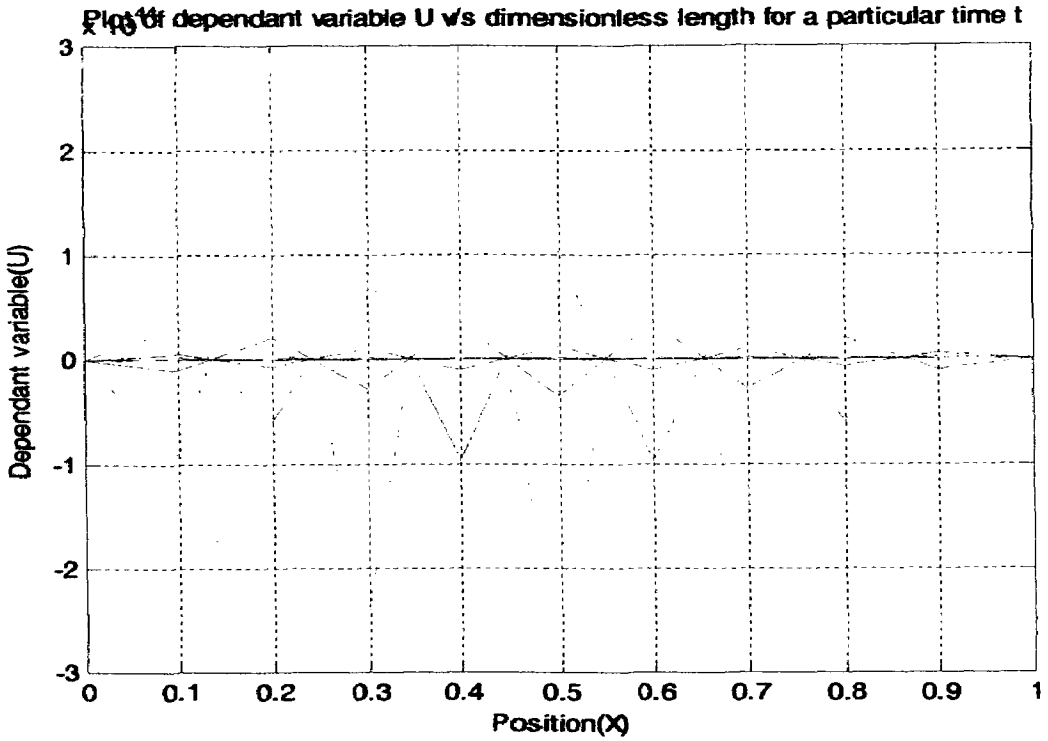


Figure: 5.7 Diffusion equation (Forward Differences) $\Delta x = \Delta t = 0.1$

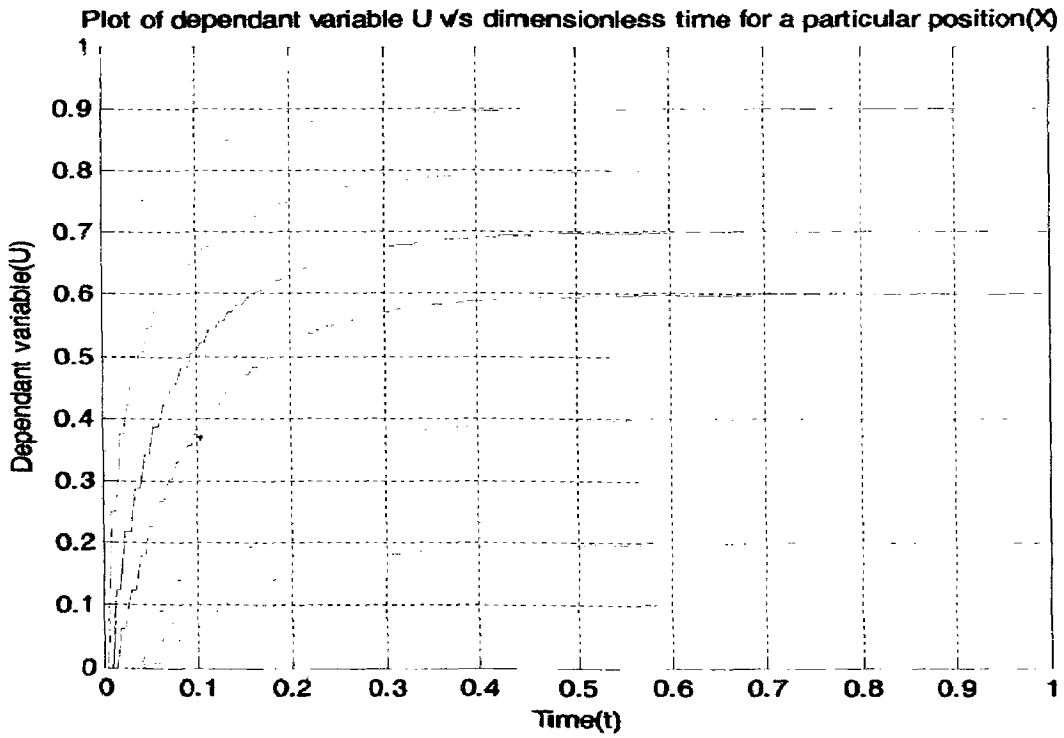


Figure: 5.8 Diffusion equation (Forward Differences) $\Delta x = 0.1$, $\Delta t = 0.005$

b) Backward Differences:

The difference equations for this method are described in section: 3.3-1b.

The system does not suffer any stability criterion and reduction in step size entirely depends on truncation errors. The error obtained for step size $\Delta x = \Delta t = 0.01$ is: 2.563%.

Further the error resulting from $\Delta x = 0.1$ and $\Delta t = 0.01$ is 1.9135 %. A table showing the error values for different step sizes is shown in Table: 1

The respective plots obtained are shown in figure 5.11, 5.12, 5.13, 5.14.

c) Crank Nicolson method:

In this method we use the central difference equations to approximate the given partial differential as is described in the section 3.3-1c.The method is stable for all the ratios of $\Delta x / \Delta t$. Further the method gives the most accurate results for $\Delta t < \Delta x$ step sizes. One such trial for $\Delta x = 0.1$ and $\Delta t = 0.01$ is run and the percent error is found to be 0.60528%. The respective plots obtained for these values of step size are shown in figure

Finally we present a tabulated form of the percent error found in each of the methods (Forward, Backward & Central Differences) for different step sizes along the length and time dimension. This is given in Table1.

From these tabulated error analysis, one may easily conclude that the Crank Nicholson (Central differences) equation is the most effective method as compared to others.

The error obtained is too low to be considered and thus the equations depict the real situation of the physical problem.

Δx	Δt	Percentage Error			*Remarks*
		Forward Differences	**Backward Differences**	**Crank-Nicolson**	
0.1	0.1	Not applicable	4.81672	4.6961523	Crank-Nicolson is better
0.1	0.01	Not applicable	1.9135264	0.6052854	Crank-Nicolson is better
0.1	0.05	Not applicable	3.39369	1.650292	Crank-Nicolson is better
0.1	0.025	Not applicable	2.59934	0.8888068	Crank-Nicolson is better
0.01	0.01	Not applicable	2.56344	1.5665114	Crank-Nicolson is better
0.01	0.001	Not applicable	1.41275	0.6609212	Crank-Nicolson is better
0.01	0.005	2.682149	2.14429	1.1033498	Crank-Nicolson is better

Table:1 Comparison of difference methods for Diffusion equation

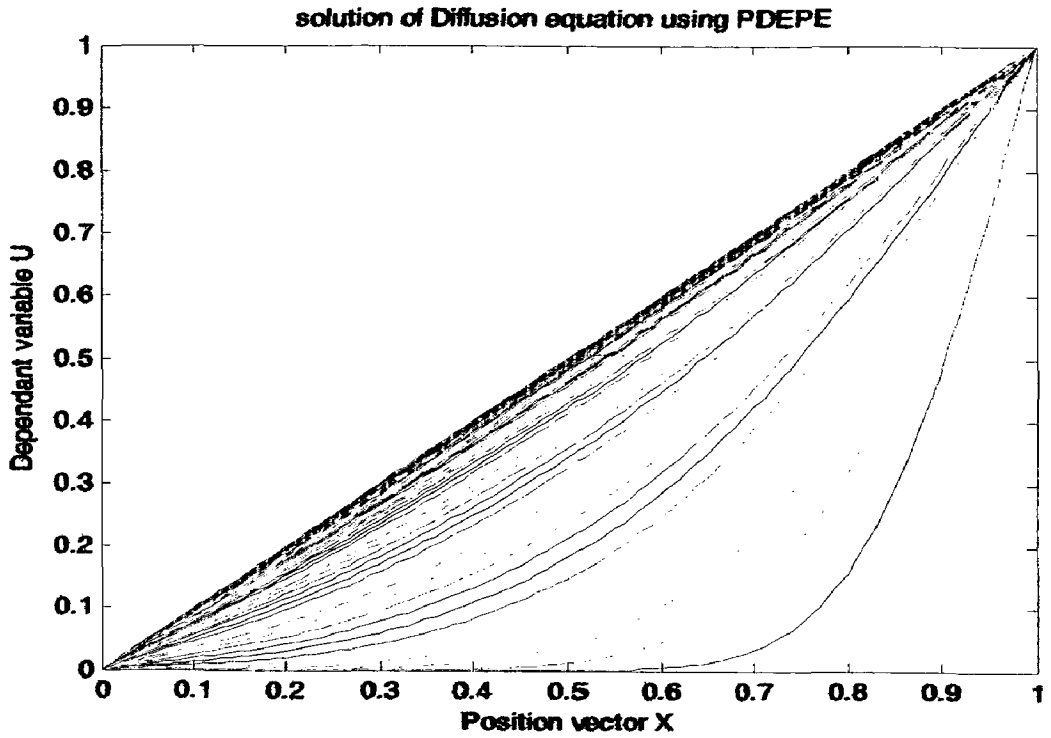


Figure 5.11 Diffusion equation (PDEPE Tool Box) $\Delta x = 0.01, \Delta t = 0.01$

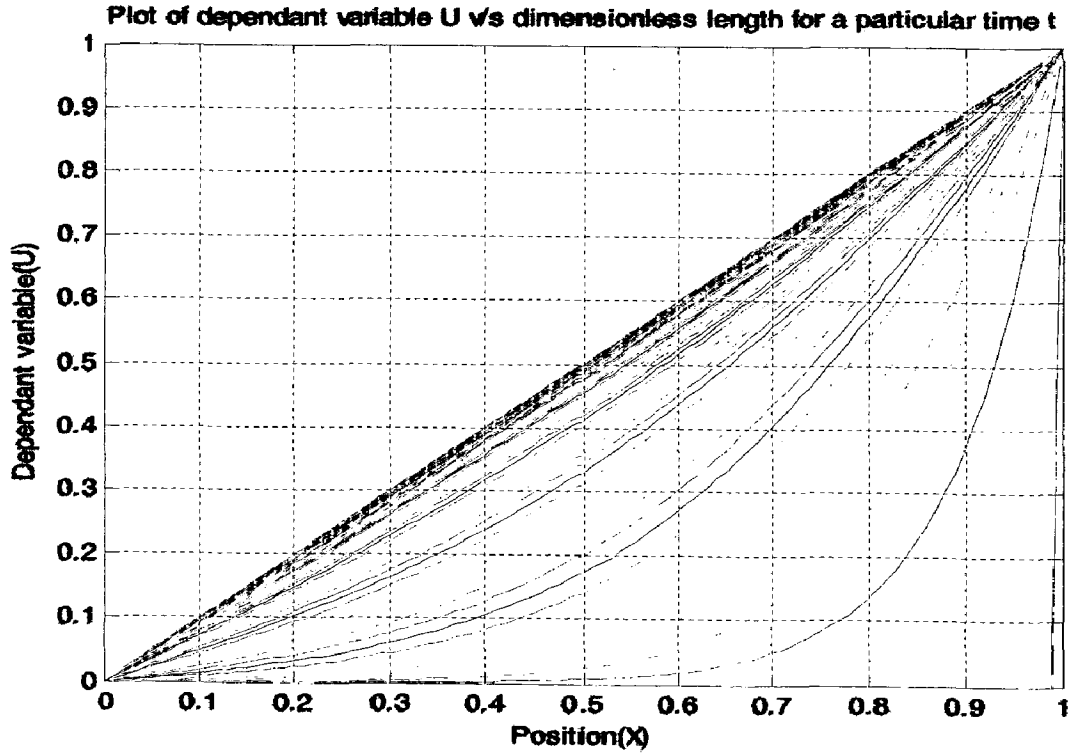


Figure:5.12 Diffusion equation (Backward Differences) $\Delta x = 0.01$, $\Delta t = 0.01$

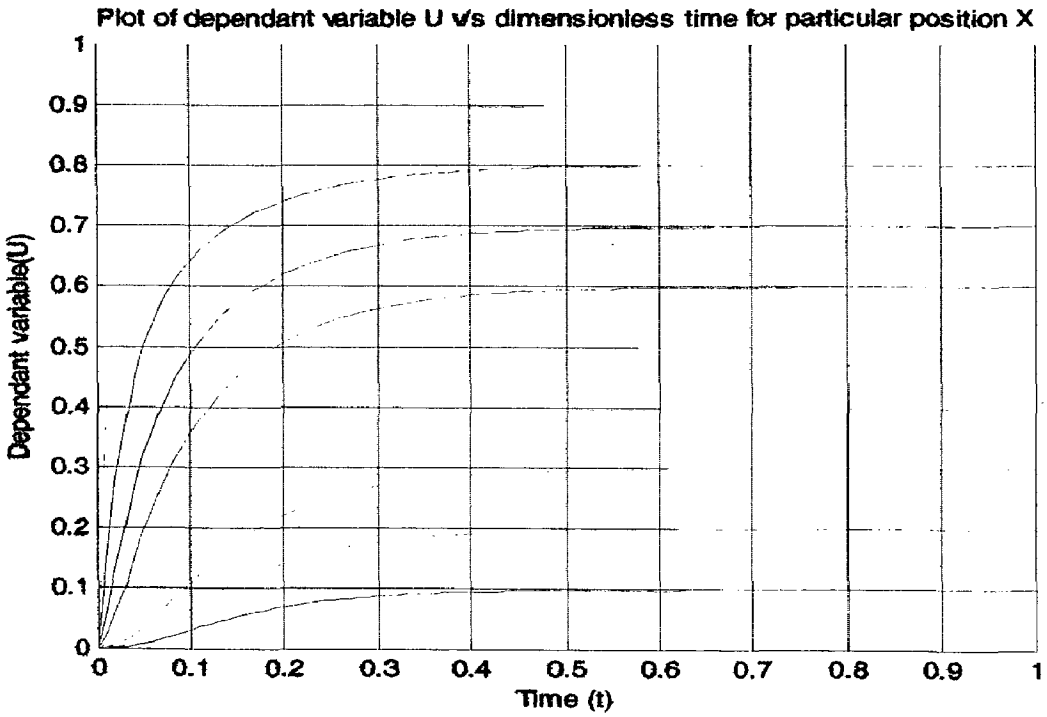


Figure: 5.13 Diffusion equation (Backward Differences) $\Delta x = 0.1, \Delta t = 0.01$

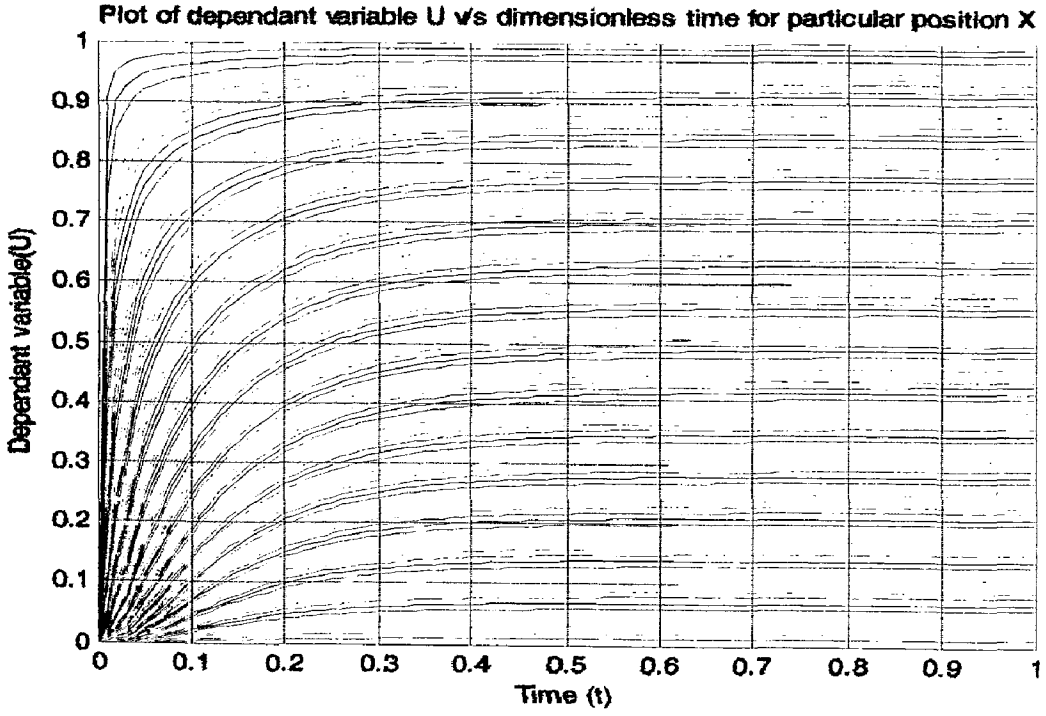


Figure: 5.14 Diffusion equation (Backward Differences) $\Delta x = 0.01, \Delta t = 0.01$

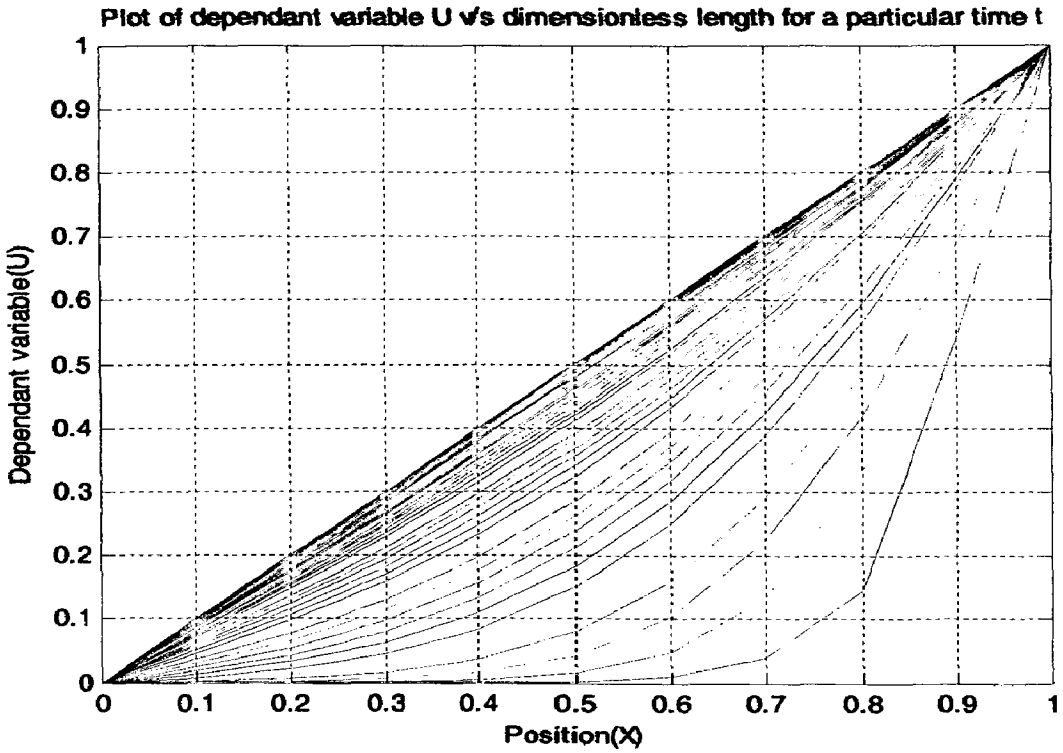


Figure: 5.15 Diffusion equation (PDEPE Tool Box). $\Delta x = 0.1$, $\Delta t = 0.01$

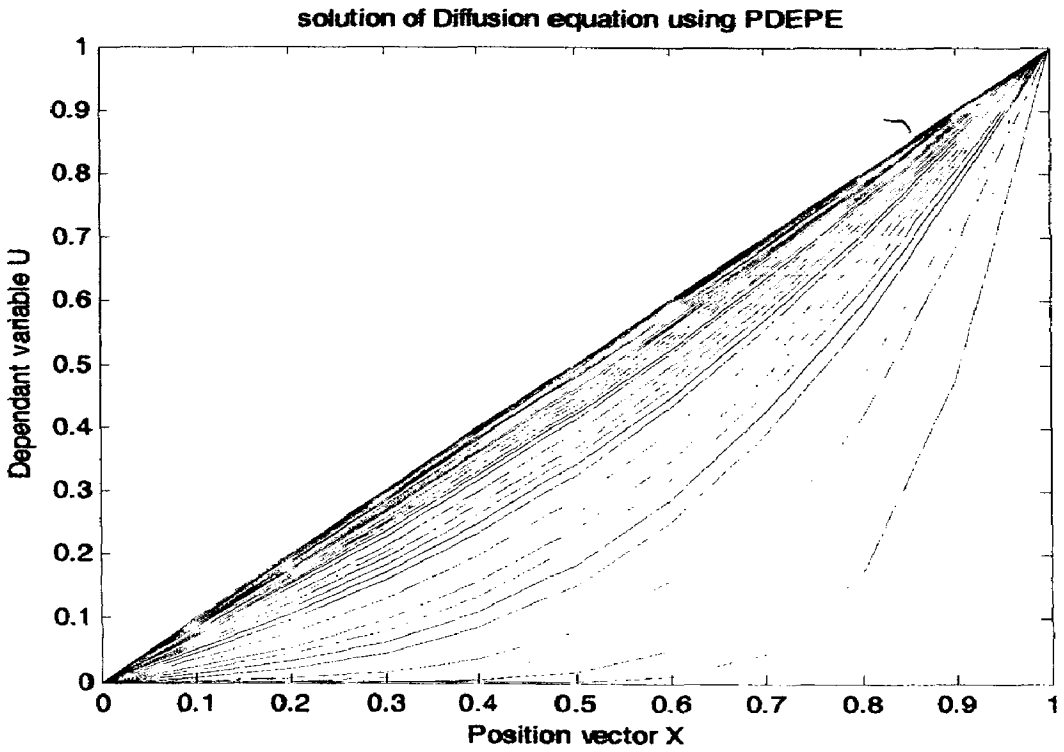


Figure: 5.16 Diffusion equation (Crank Nicholson). $\Delta x = 0.1$, $\Delta t = 0.01$

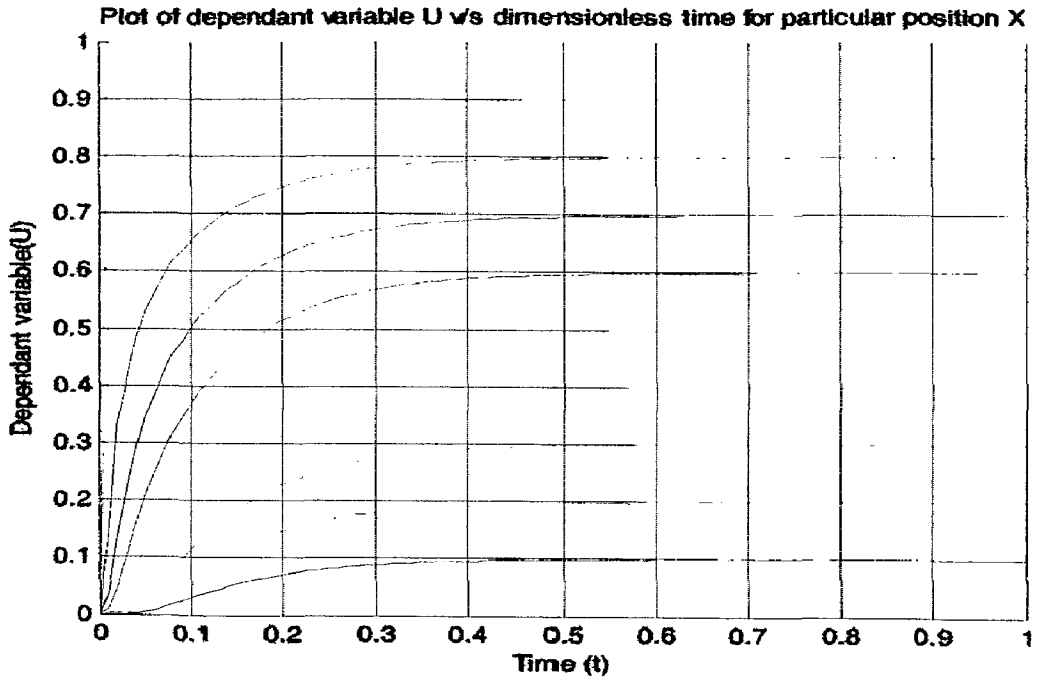


Figure: 5.17 Diffusion equation (Crank Nicholson). $\Delta x = 0.1$, $\Delta t = 0.01$

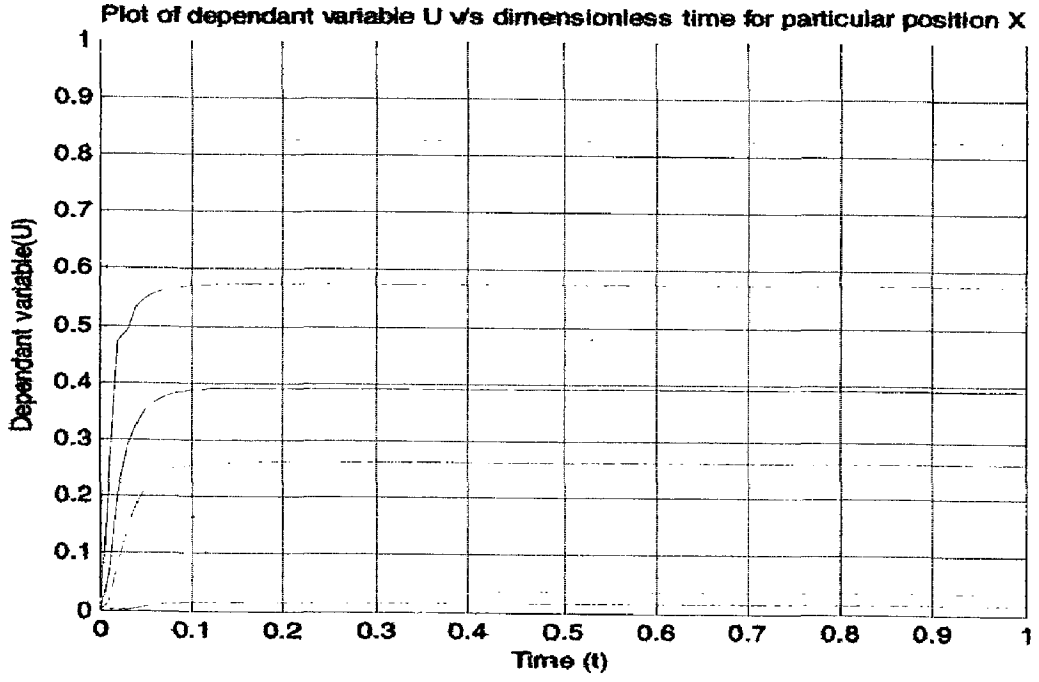


Figure: 5.18 Heat flow in a tapered rod (Crank Nicholson). $\Delta x = 0.1$, $\Delta t = 0.01$

5.3 Heat Flow in a Tapered Rod (Unsteady state):

The governing equations are

$$\frac{\partial^2 u}{\partial x^2} + \frac{2}{x}\left(\frac{\partial u}{\partial x} - pu\right) = \frac{\partial u}{\partial t} \qquad 5.3\text{-}1$$

subject to;

$$u(1,t) = 1$$

$$\frac{\partial u}{\partial x} - pu = 0 \quad \text{at } x = 0 \text{ all } t \qquad 5.3\text{-}2$$

$$u(x,0) = 0 \text{ all } x$$

The numerical difference equations are developed using Crank Nicholson method and are mentioned in the section 4.2

Here we present the results. When the difference scheme is run for $\Delta x = 0.1$ and $\Delta t = 0.01$, we get the profiles as shown in figure 5.19 and 5.20. Here we observe that the profiles do match to some extent, but the percentage error calculated achieves a greater value of 28.01%. The high error can be devoted to the oscillations brought in the numerical scheme by the convective term. This erratic profile can be stated from figure 5.18. The convective term is multiplied by the reciprocal of x. This extends the effect caused by Convective term. The equation no more possesses the parabolic characteristics and develops hyperbolic nature. The PDEPE tool box which we are using for error analysis do not support for the solution of hyperbolic equations. Hence the tool box cannot be used to handle such situations. This it can be presumed that the difference equations are simulating the situation more powerfully than the PDEPE tool box.

5.4 Shrinking Core model

The difference equations that are derived in the section: 4.7 are simulated. The results are compared with those obtained by analytical solution for error analysis. The dimensionless concentration profiles are shown in figure 5.21 and 5.22. The error obtained for $\Delta x = 0.001$ is 0.2045%. The results are converging towards 1. Hence the developed numerical scheme is efficiently simulating the given steady state system.

This difference scheme thus can be used to simulate the model equations of a number of reactors under steady state conditions.

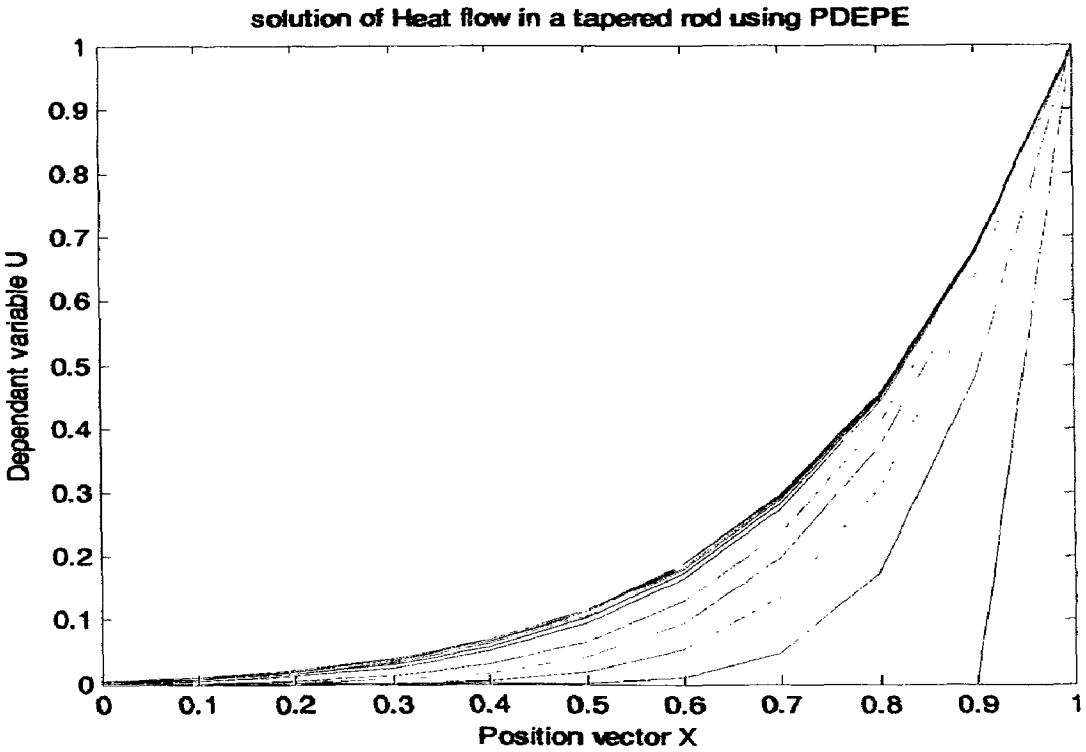


Figure:5.19 Heat flow in a tapered rod (PDEPE Tool Box) $\Delta x = 0.1$ $\Delta t = 0.01$

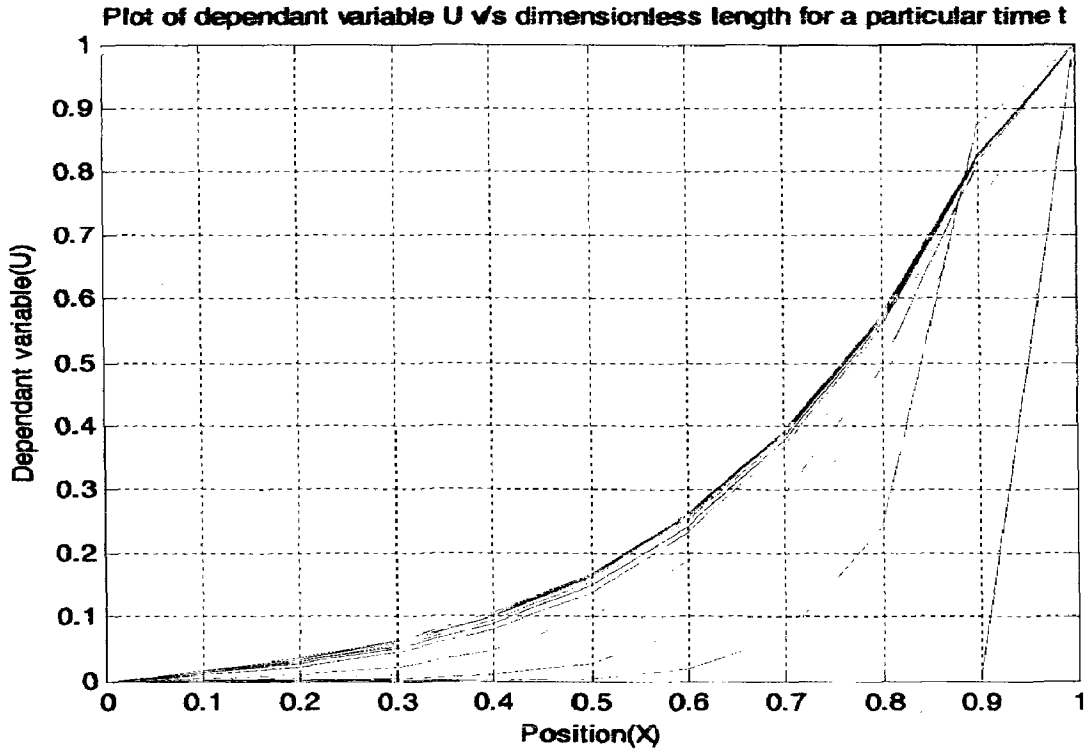


Figure: 5.20 Heat flow in a tapered rod (Crank Nicholson). $\Delta x = 0.1$, $\Delta t = 0.01$

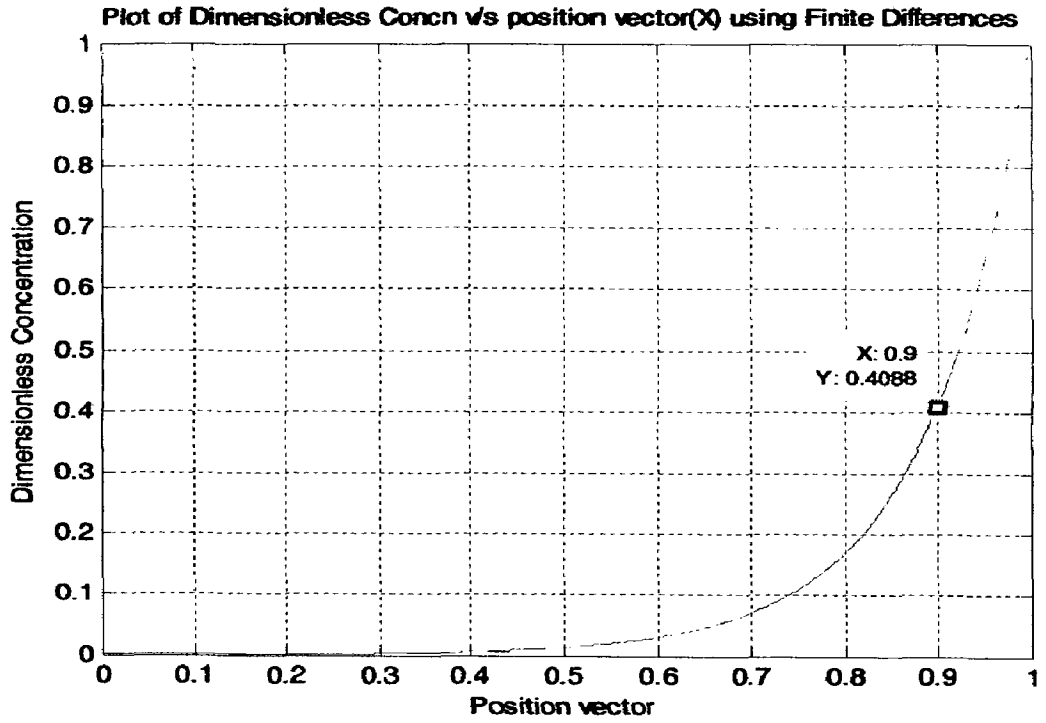


Figure: 5.21 Shrinking core model (Difference equations) $\Delta x = 0.001$

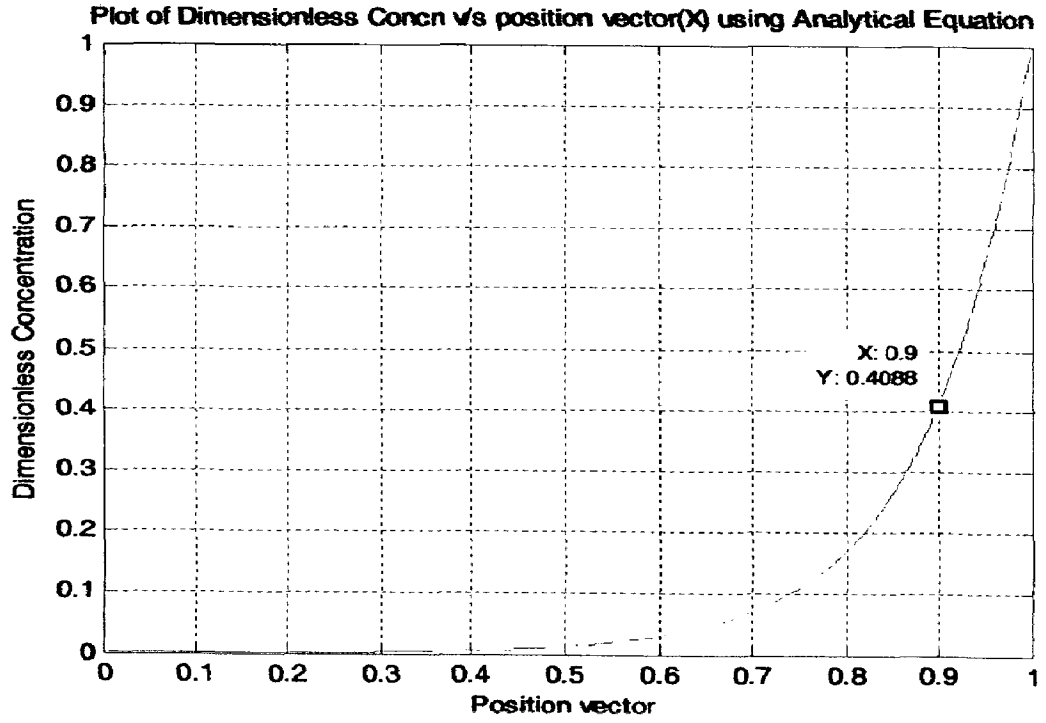


Figure: 5.22 Shrinking core model (Analytic Solution) $\Delta x = 0.001$

5.5 Isothermal Flow Reactor with a Second order Reaction:

The flow equation describing the concentration in an Isothermal flow reactor with a second order reaction can be depicted by a second order non linear partial differential equation (Convection Diffusion equation). This equation is as follows

$$\frac{\partial^2 u}{\partial x^2} - s\frac{\partial u}{\partial x} - ru^2 = \frac{\partial u}{\partial t} \qquad 5.5\text{-}1$$

Subject to the boundary conditions;

$$u(x,0) = 0 \ \textit{for all } x$$

$$\frac{\partial u}{\partial x} = 0; \ \textit{at } x=1 \ , \ \textit{all } t \qquad 5.5\text{-}2$$

The second order partial differential accounts for the molecular diffusion. The single partial differential with respect to space is representing the convective or bulk flow of mass where as the term ru^2 is the source term accounting for the reaction occurring.

The difference equations for this non linear partial differential are developed in the section: 4.4

Here we shall discuss the results obtained after simulation of this system.

For the step size of $\Delta x = \Delta t = 0.1$ we get the profiles as shown in figure 5.23.

There is some sort of oscillation in the initial time period. This is because the non linear terms are dominant and the linearization of this term is insufficient with these step sizes.The percent error in the calculation is 4.1596618%.

When the step size is reduced to $\Delta x = \Delta t = 0.01$, the error is 1.360%.

Further for the step size of $\Delta x = 0.01$ and $\Delta t = 0.001$, the error is 1.04714%

The profiles for both PDEPE tool box and the difference schemes are exactly matching as can be seen in the figure: 5.24, 5.25, 5.26, and 5.27.

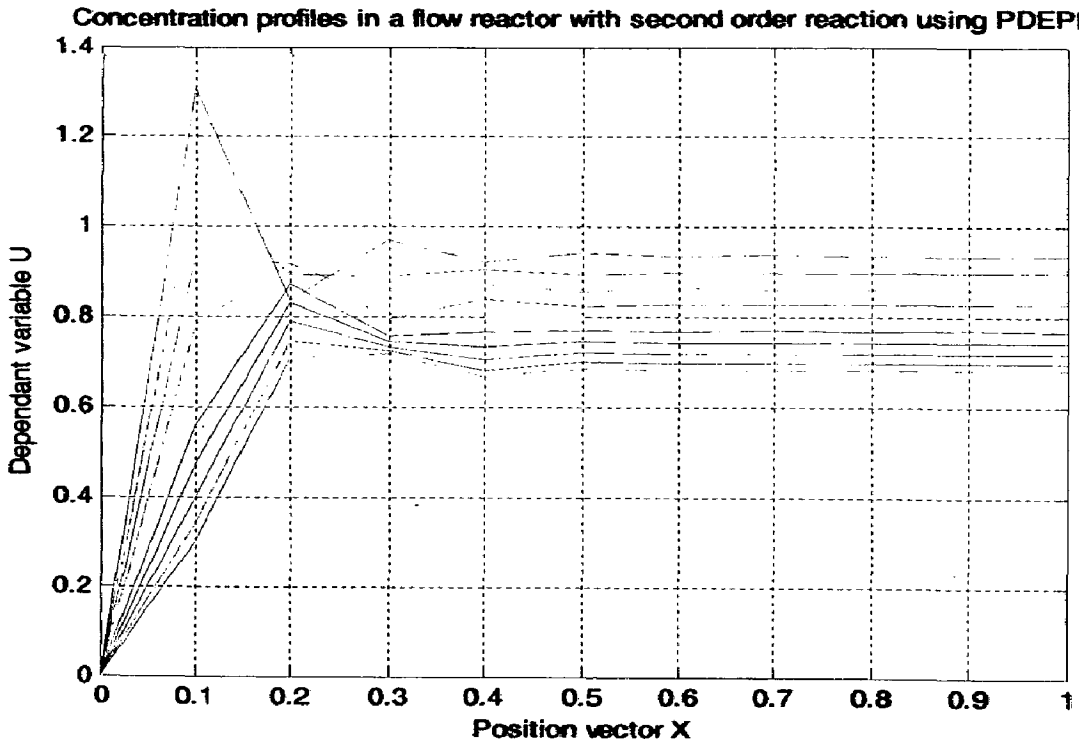


Figure: 5.23 Isothermal flow reactor $\Delta x = \Delta t = 0.1$

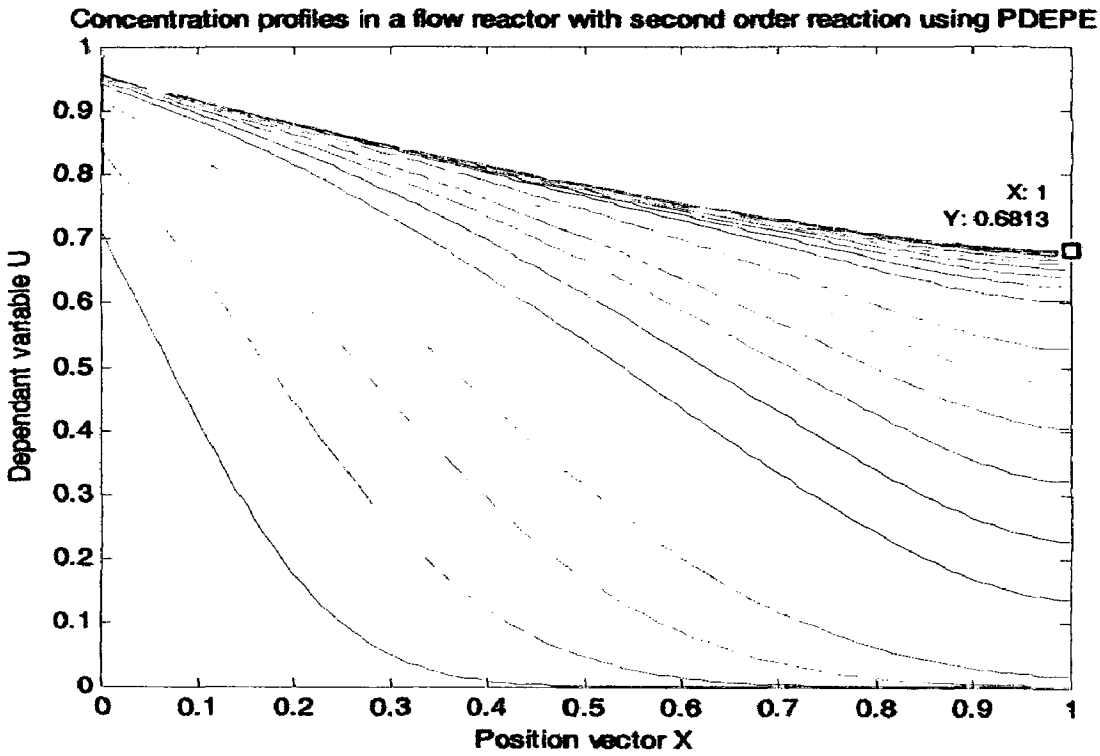


Figure: 5.24 Isothermal Flow Reactor (PDEPE Tool Box) $\Delta x = \Delta t = 0.01$

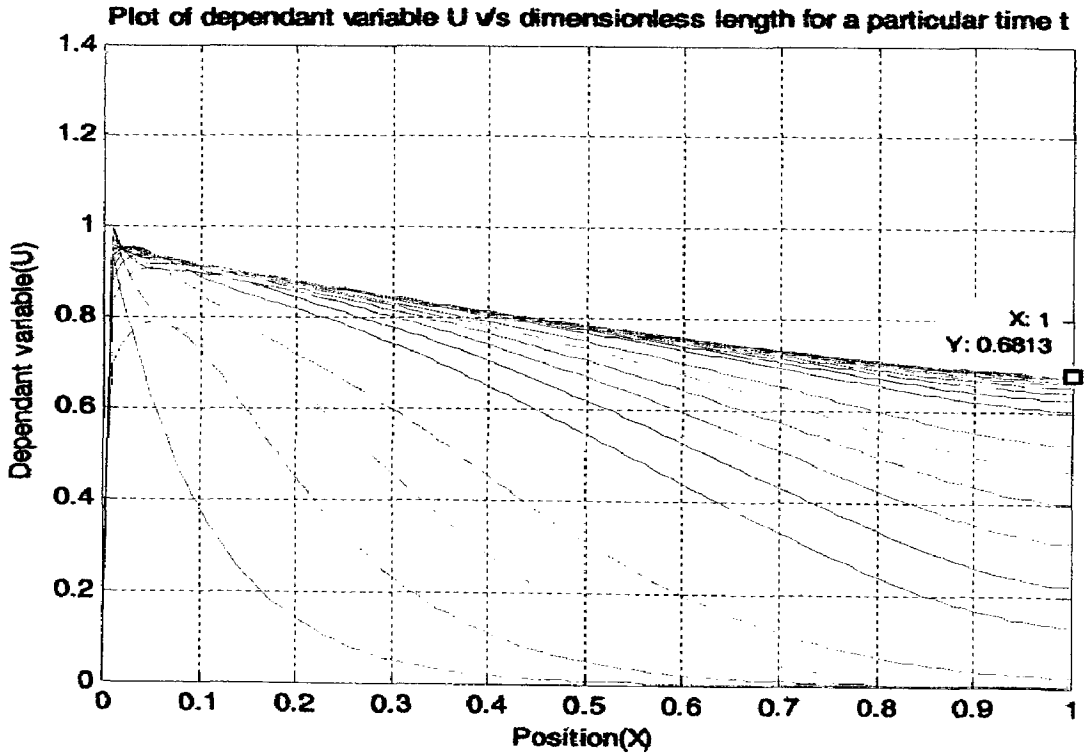


Figure: 5.25 Isothermal Flow Reactor (Differences scheme) $\Delta x = \Delta t = 0.01$

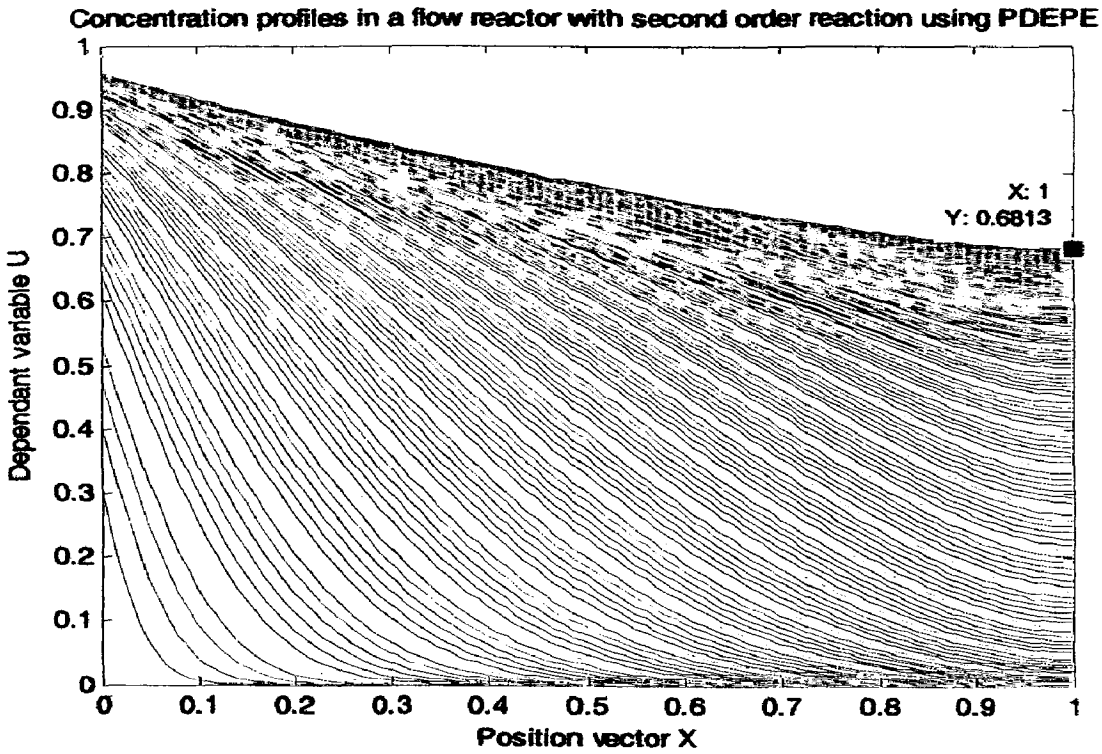


Figure: 5.26 Isothermal Flow Reactor (PDEPE Tool Box) $\Delta x = 0.01$ $\Delta t = 0.001$

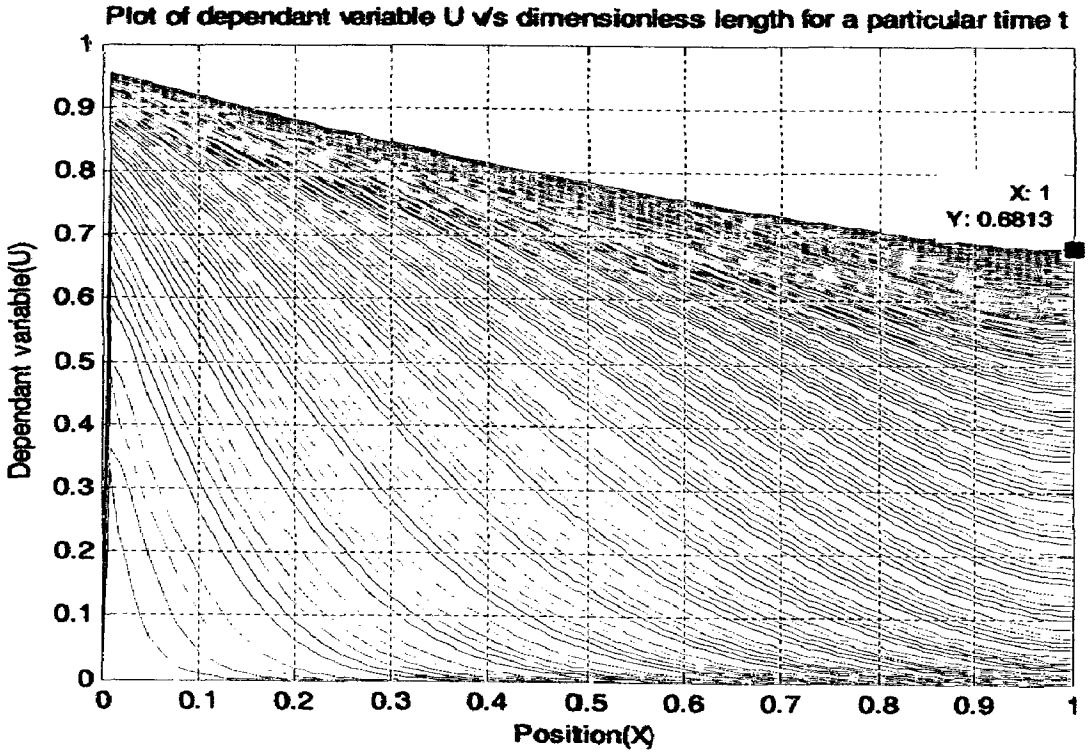


Figure: 5.27 Isothermal Flow Reactor (Difference scheme) $\Delta x = 0.01$ $\Delta t = 0.001$

5.6 Countercurrent Heat exchanger

The equations describing the flow of hot and cold streams are purely convective. These are represented by hyperbolic equations.

The equations are given as;

$$-b_1\left(\frac{\partial u}{\partial x}\right)-c_1(u-v)=\left(\frac{\partial u}{\partial t}\right)$$
$$b_2\left(\frac{\partial v}{\partial x}\right)+c_2(u-v)=\left(\frac{\partial v}{\partial t}\right) \qquad 5.6\text{-}1$$

Subject to; $u(x,0)=0 \quad x>0$; $\begin{matrix} u(0,0)=1 \\ v(x,0)=0 \\ u(0,t)=1 \\ v(1,t)=0 \end{matrix}$ *all t* 5.6-2

The difference approximations for these equations are covered in the section: 4.3

The simulation results are shown here.

Since the equations are purely hyperbolic, the PDEPE tool from MATLAB is not applicable for this particular case.Thus only the profiles obtained by the simulations using the difference schemes developed are presented here (see Figure 5.28, 5.29). Further the difference scheme can be said to be converging and accurate as it is producing the same value of the dependant variables u and v for different step sizes (Figure: 5.28, 5.29, 5.30, and 5.31).

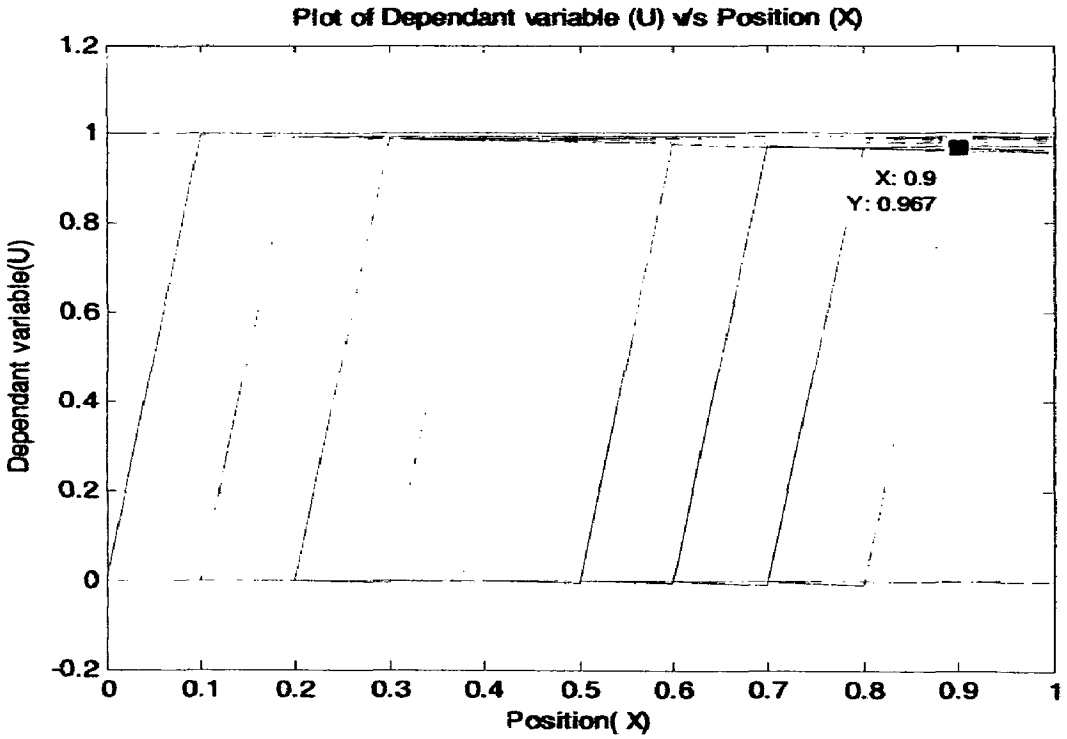


Figure: 5.28 Countercurrent Heat Exchanger $\Delta x = \Delta t = 0.1$

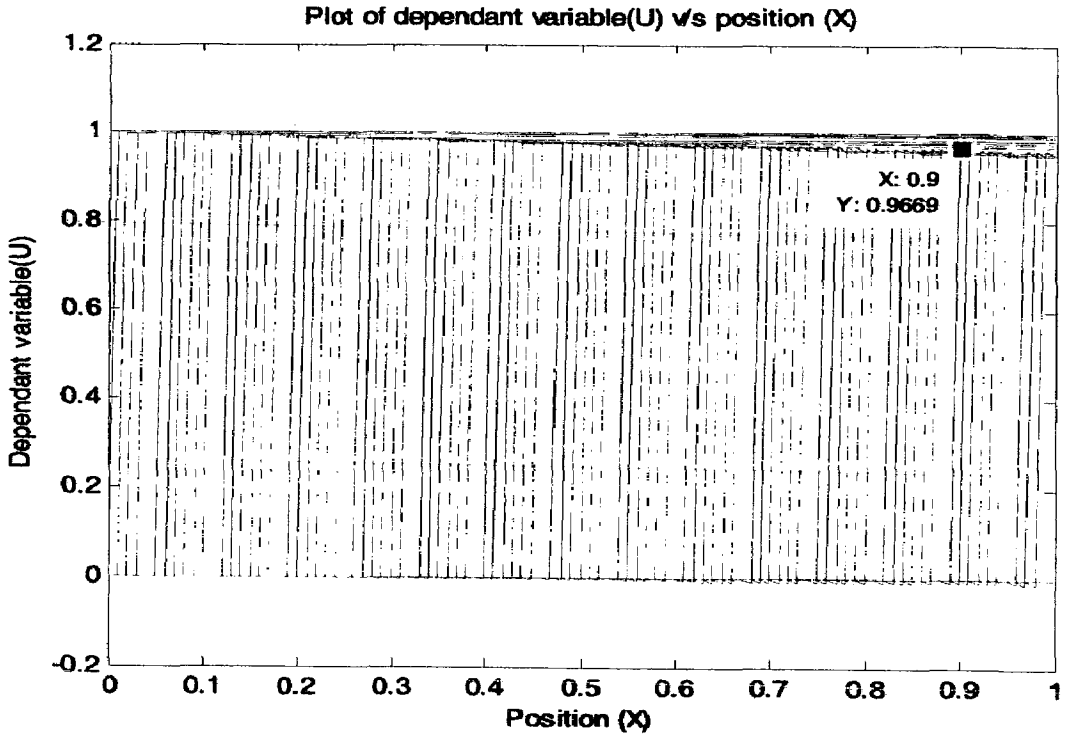


Figure: 5.29 Countercurrent Heat Exchanger $\Delta x = \Delta t = 0.01$

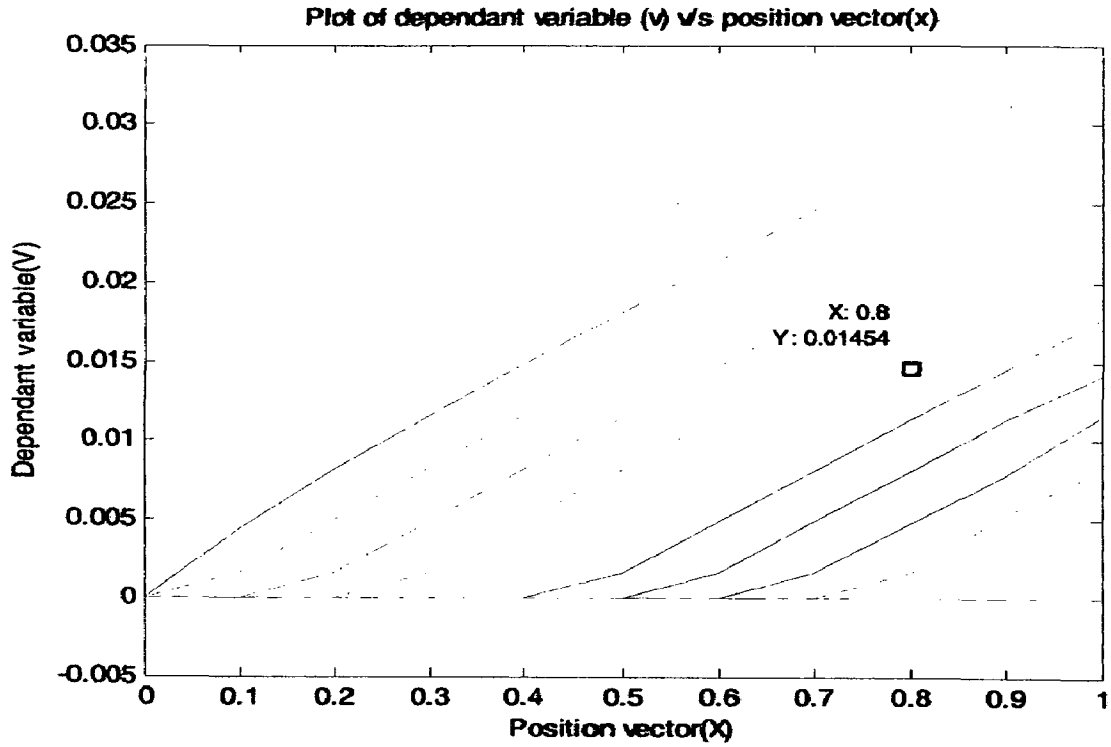


Figure: 5.30 Countercurrent Heat Exchanger $\Delta x = \Delta t = 0.1$

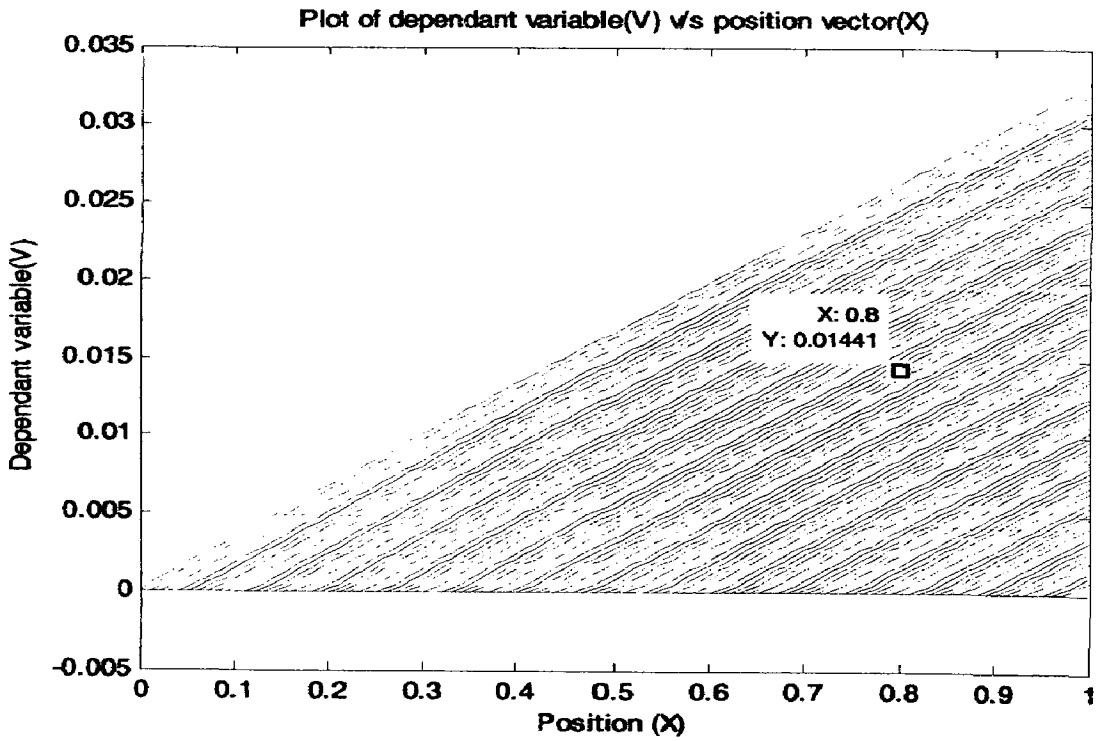


Figure: 5.31 Countercurrent Heat Exchanger $\Delta x = \Delta t = 0.01$

5.7 Heat flow in a rod with heat received by radiation at one end

This application explains the methods to handle the non linear boundary conditions. We may consider the simple Diffusion equation with non linear boundary arising as a result of radiative heat transfer. The equation is;

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad 5.7\text{-}1$$

Subject to;

$$u(x,0) = u_0 \ \ all\ x.$$

$$u(0,t) = u_0 \ \ all\ x \qquad 5.7\text{-}2$$

$$s(1-u^4) - \frac{\partial u}{\partial x} = 0 \ \ at\ x=1\ and\ t$$

The difference schemes are developed in the section: 4.6. This is clear that the non linear boundary conditions increases the computational times. A functional relationship is derived for the grid point along the boundary, which is solved by using the regula falsi method as is described in section: 4.6. A similar treatment can be prescribed for solutions involving boundary conditions at both the system boundaries.

Here we analyze the results. For a step size of $\Delta x = \Delta t = 0.1$, the error is 0.3362%.For $\Delta x = \Delta t = 0.01$ error: 0.045248% and for $\Delta x = 0.1$ and $\Delta t = 0.01$, the error is 0.0478 % (see Figure: 5.32, 5.33, 5.34, 5.35).

We observe that the percent error in the simulation results is very less. This accuracy can be devoted to the method of iterations used in refining the values till the tolerance level is achieved.

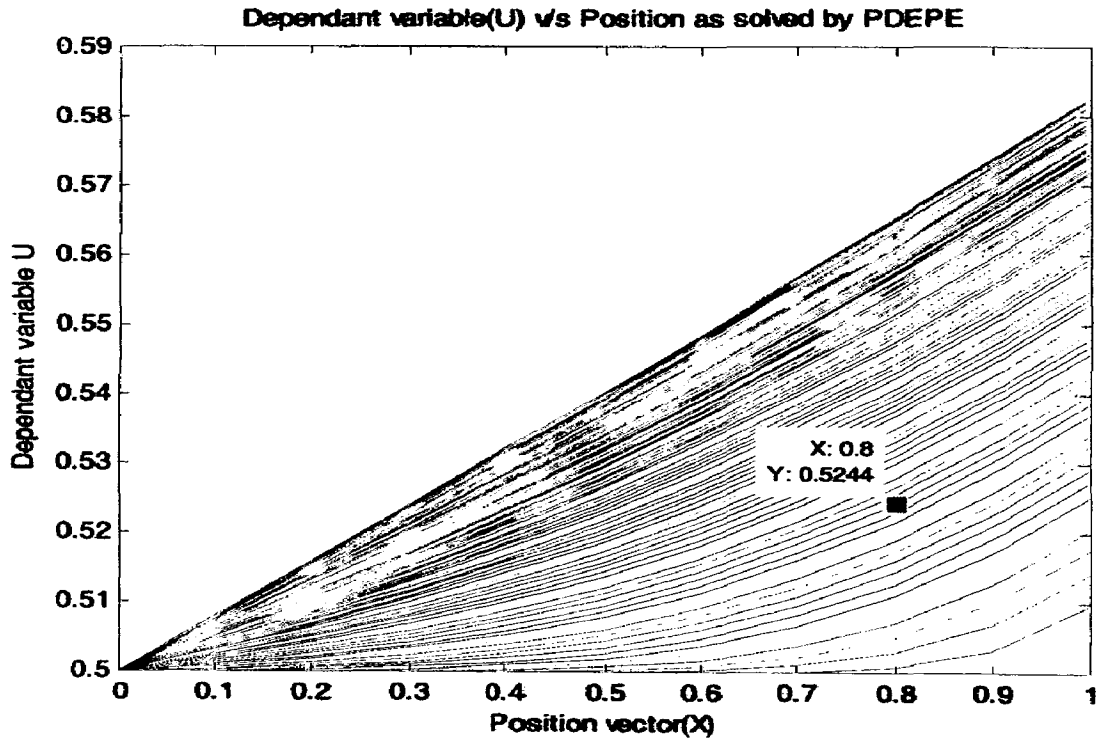


Figure 5.32 Heat received by radiation(PDEPE Tool Box) Δx =0.1 Δt =0.01

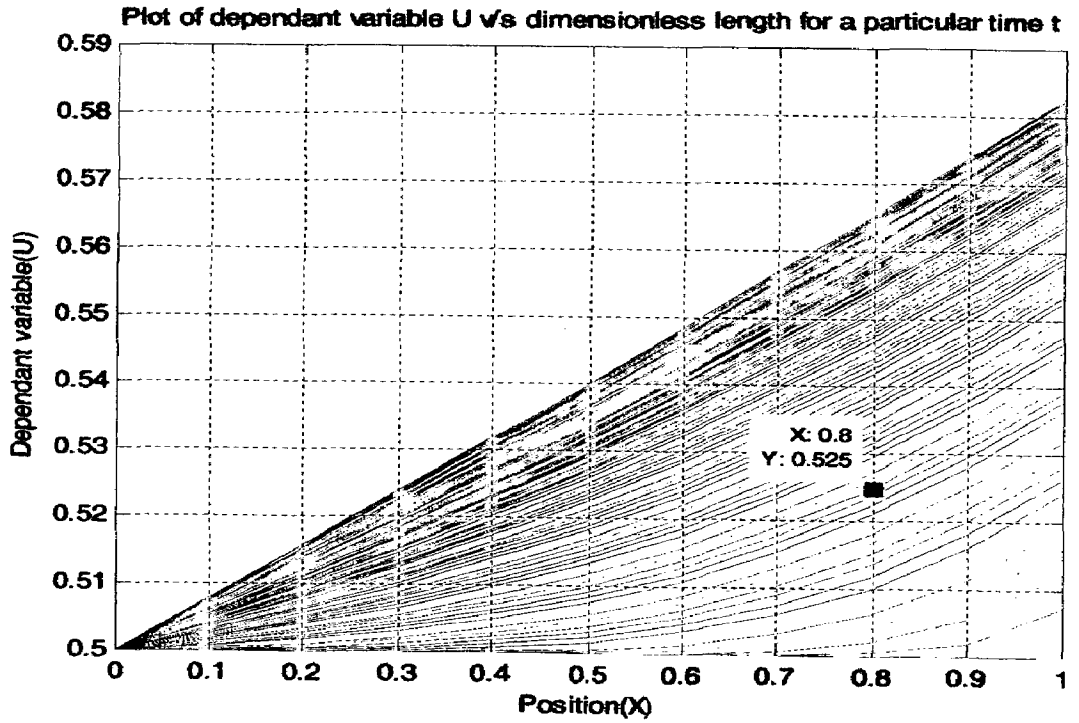


Figure:5.33 Heat received by radiation (Difference scheme) Δx =0.1 Δt =0.01

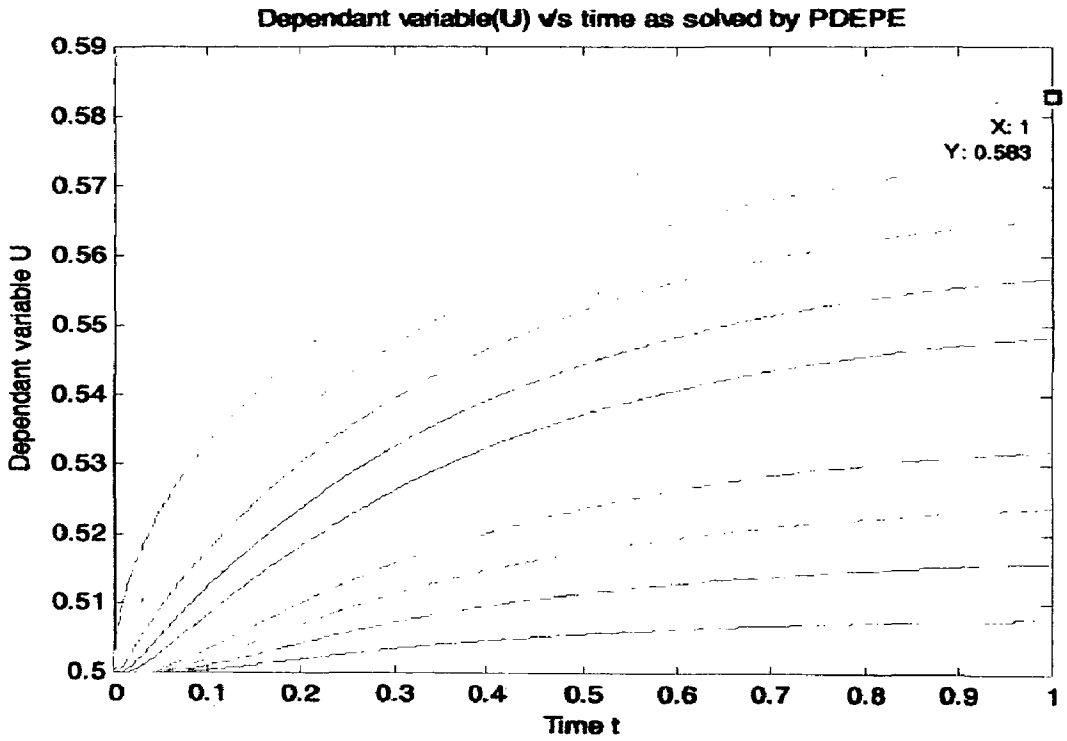


Figure: 5.34 Heat received by radiation (PDEPE Tool Box) Δx **=0.1** Δt **=0.01**

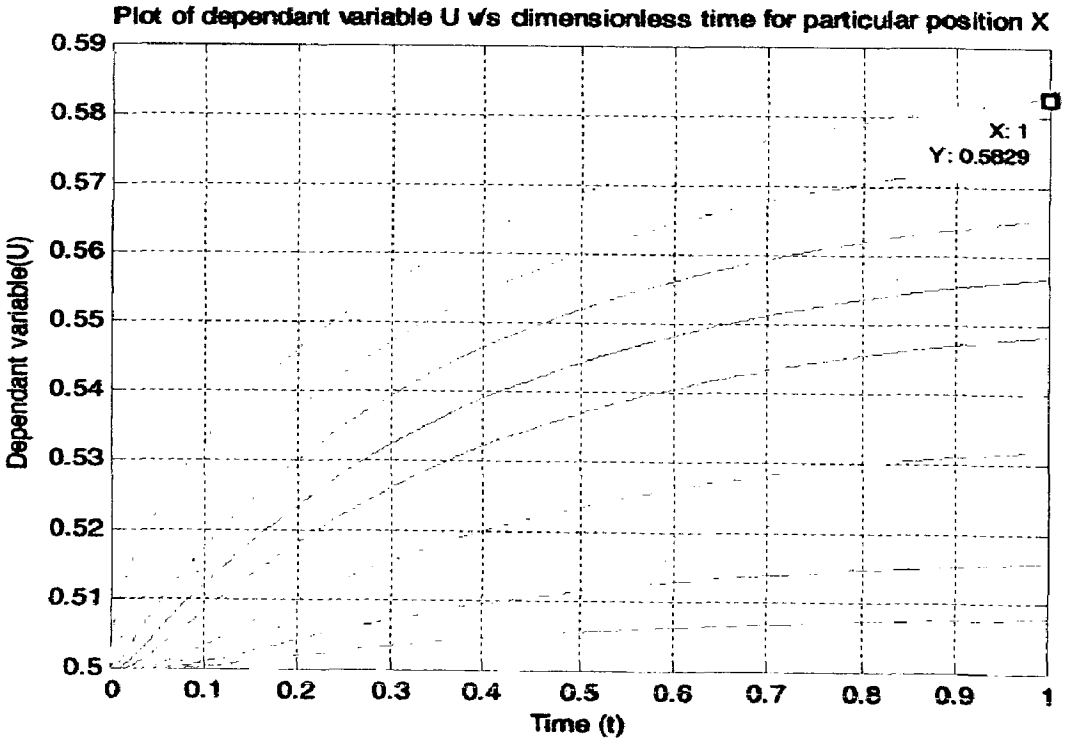


Figure: 5.35 Heat received by radiation (Difference scheme) Δx **=0.1** Δt **=0.01**

5.8 Parabolic equation with small dispersion coefficient

The difference equations for this case are derived in the section: 4.5. Here we check the dependence of the solution for the varying values of '*b*' .i.e for the effect of convection dominance in the differential equation.

The condition given in the literature is; $\frac{b\Delta x}{2} < 1$. Thus if we choose the value of $b=20$, and $\Delta x = 0.1$, the condition is violated. The plot obtained shows severe oscillations (see Figure: 5.36, 5.37). Next if we take the value of b as 0.005, $\Delta x = 0.1$ and $\Delta t = 0.01$, the profiles obtained are shown in Figure: 5.38, 5.39 and the condition is not violated. The error in the calculation is 2.303%. Thus for higher values of b one cannot use the regular difference equations meant for parabolic equations.

From the results discussed in the various sections of this chapter, we find that the most of the systems modelled by Convection Diffusion equations are simulated and the solutions represent the exact characteristics of the system to a considerable extent. However it becomes sometimes necessary to analyze the difference equations for stability and convergence owing to step size restrictions.

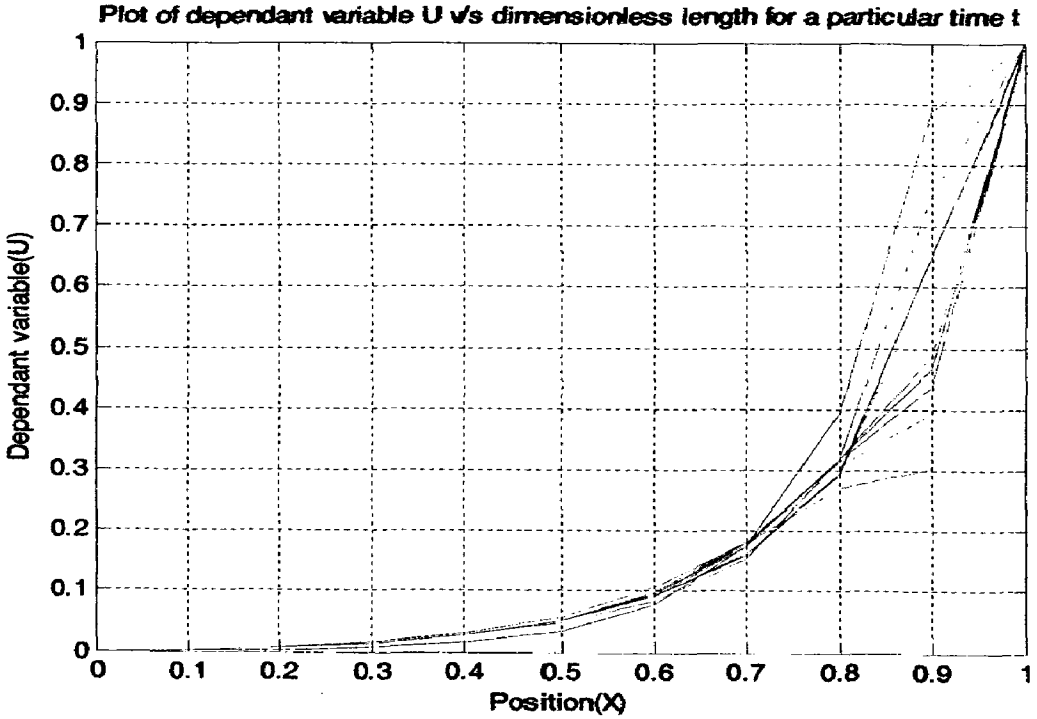


Figure:5.36 Parabolic equation with small dispersion coefficient $\Delta x = \Delta t = 0.1$

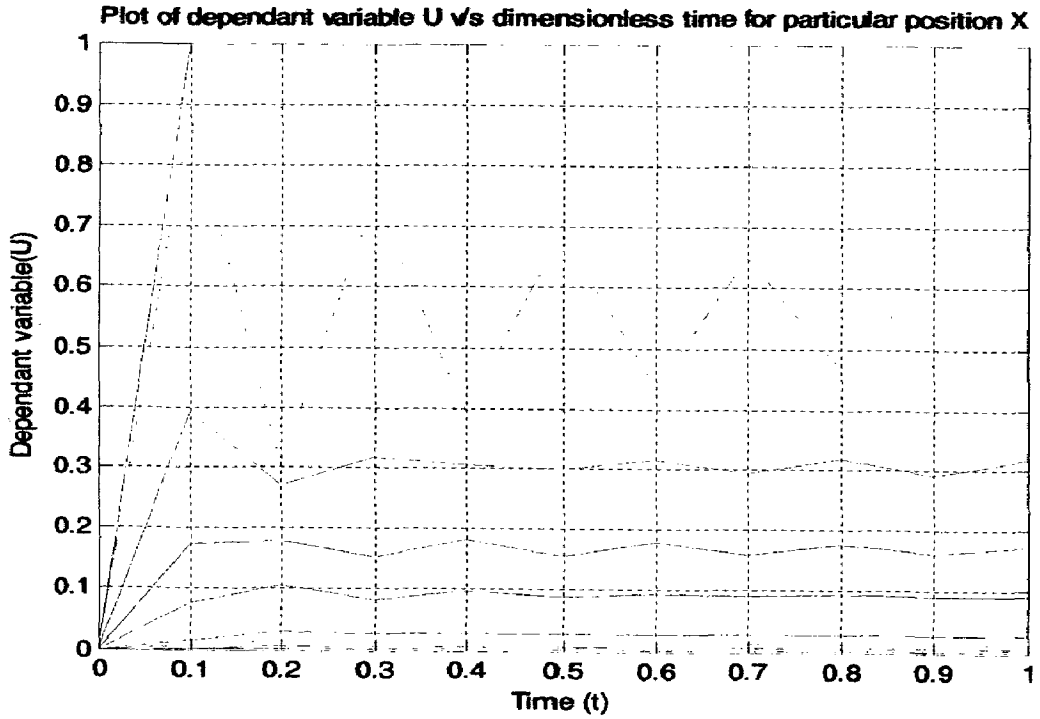


Figure:5.37 Parabolic equation with small dispersion coefficient $\Delta x = \Delta t = 0.1$

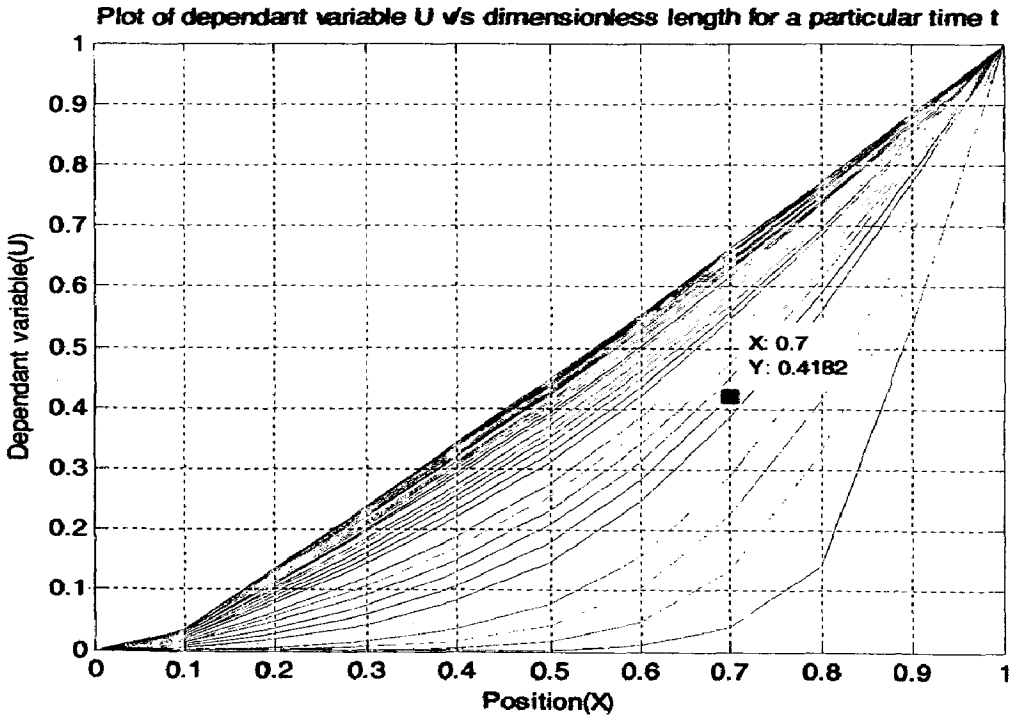


Figure:5.38 Parabolic equation for small dispersion Coefficient Δx =0.1, Δt =0.01

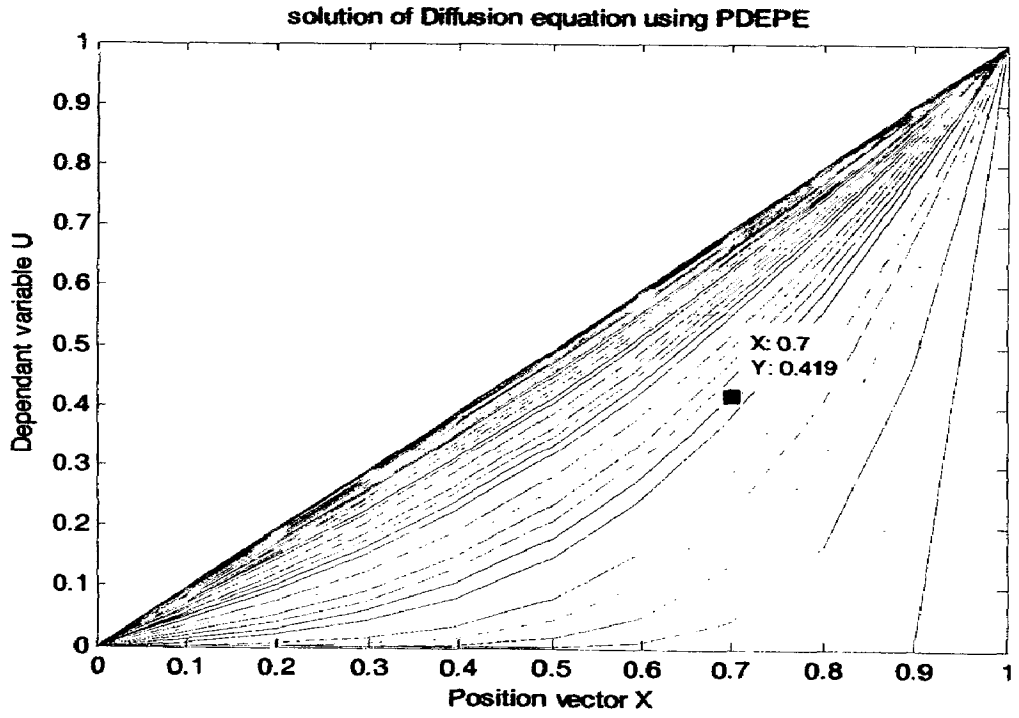


Figure:5.39 Parabolic equation for small dispersion coefficient Δx =0.1, Δt =0.01 (using PDEPE Tool Box)

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

Conclusions:

In the present dissertation work, we focused on the simulation of Convection Diffusion equations using method of Finite Differences. As a result of this analysis, we may write the following conclusions.

- Convection Diffusion equations are widely used in many chemical engineering applications to characterize the flow properties of fundamental modes of transfer
  i.e. mass, momentum and energy.
- The Convection Diffusion equations are mostly unsolved by regular analytical techniques as these are basically non linear partial differential equations. Thus it becomes mandatory to use several numerical techniques to simulate them.
- The most widely used numerical techniques in handling the Convection Diffusion equations are: Method of Finite Differences and Method of Finite Elements.
- The finite difference approximations are easy to formulate and can be utilized in the development of computer algorithms for large simulation works.
- The method of Finite Differences is an effective and reliable method, when the difference equations are formulated with appropriate understanding of the system.
- In this work, the very basics of the method of Finite differences are studied. Thereby a number of finite difference schemes are developed for various types of changes involved in the Convection Diffusion equations, such as non linearity, moving boundary conditions etc.
- The finite difference approximations can be successfully used to discretize the partial differential equations into a set of linear algebraic equations that can be solved easily for solutions. This method gives significant results for a number of applications in the field of engineering. Further, the error analysis shows that the stipulated accuracy can be achieved by simply varying the step size of descretization. Knowledge of the stability of the method is essential to avoid

erratic results. This needs a considerable study of the physical behavior of the system under examination.

- The results obtained on application of finite difference schemes to various engineering problems are comparable to those affirmed by the PDEPE tool box of MATLAB software (based on Method of Finite Elements).
- Thus the objective of simulation of Convection Diffusion equations using Method of Finite Differences is realized.

Recommendations for future work:

The conclusions in the previous section imply the capability of Method of Finite Differences as a simulation tool. The future developments that can be suggested in this work are;

- Development of various new models that involve the application of Convection Diffusion equations.
- The difference equations formulated in this work are suitable for handling one dimensional Convection Diffusion equations. Consequently, a search for new difference methods pertinent to multidimensional Convection Diffusion equations can be recommended.
- The finite difference schemes for coupled partial differential equations can be formulated for simultaneous simulations of engineering systems.
- A considerable study can be done in the field of error reduction and accuracy.
- A new user friendly software can be devised that considers the formulated difference schemes for fast and reliable computations.

REFERENCES

[1] Bush,A.W.,Gibson, R.D.,**1982**. Proceedings of Polymodel4, the fourth Annual Conference of the North East Polytechnics Mathematical Modeling & Computer Simulation group- Sunderland. Numerical Modeling in Diffusion Convection.Pentech Press, London.

[2] Polyanin, A.D., Kutepov, A.M., Vyazmin, A.V., Kazenin, D.A.,**2002**. Hydrodynamics Mass & Heat Transfer in Chemical Engineering. Vol 14. Taylor & Francis.

[3] Warnatz, J.**1987**. Physical chemical Applications. Modeling of ChemicalReaction Systems. Springer series in Chemical Physics. Springer-Verlag New York.

[4] Smith, J.M., **1992**.Design of Heterogeneous Catalytic Reactor. Chemical Engineering Kinetics.3 rd edition. Mc Graw Hill International Book Company.

[5] Waghode, A.N., Hanspal, N.S., Shigidi, I.M.T.A., Nasselli, V., Hellgard, K., **2005**. Computer Modeling and numerical analysis of Hydrodynamics and heat transfer in a non porous Catalytic reactor for the decomposition of ammonia Chemical Engg Science 60, 5862-5877.

[6] Carberry, James. J., **1976**. Conservation Equations for Reactors. Chemical Catalytic Reaction Engineering. Mc Graw Hill Co.

[7] WilHelm, H.Richard., **1977**. Steady state operation of Fixed Bed Reactors & Monolith Structures. Chemical reactor Theory: A Review. Prentice Hall, Inc.

[8] Dehghan, Mehdi., **2005**.On the numerical solution of the one dimensional Convection Diffusion Equation. Mathematical Problems in Engineering .Hindawi Publishing Corporation.

[9] Kulikovsky, A.A., **2001**. Simple and accurate Scheme for Nonlinear Convection Diffusion equation. Journal of Computational Physics 173, 716-729.

[10] Zhang, Jun., Kouatchou, Jules., Ge, Lixin., **2002**. A family of fourth order Differences schemes on rotated grid for two-dimensional convection-diffusion equation. Mathematics and Computers in Simulation 54, 413-429.

[11] Liu, Biyue., Myron B.Allen., Hristo, Kojouharov., Benito, Chen., **1998**. Finite Element solution of Reaction Difusion equations with Advection. www.prosim.net/pdf/13_1CL_1998.

[12] Chantasiriwan, Somchart., **2004.** Cartesian grid methods using radial basis functions for solving Poisson Helmholtz, and Diffusion-Convection equations. Engineering Analysis with Boundary Elements 28, 1417-1425.

[13] Cockburn, Bernardo., Shu, Chi-Wang., The Local Discontinuous Galerkin Method for time dependant Convection Diffusion systems. University of Minnesota. www.uta.edu/math/faculty/hristo/research/paper/cmwr96.pdf2005.

[14] Mueller, J.L., Shores, T.S., **2004**. A new Sinc-Galerkin method for Convection Diffusion Equations with mixed boundary conditions. An International Journal of Computers & Mathematics with applications 47, 803-822.

[15] Finlayson, Bruce A., **1980**. Non-linear analysis in Chemical Engineering. Mc Graw Hill series.

[16] Jain, M.K., **2002**. Numerical Solution Of Differential Equations-second edition. New Age International (P) Limited Publishers.

[17] Iordanidis, A.A., Annaland, M.Van Sint., 2004. A numerical method for the solution of the wave model and convection dominated diffusion type models for catalytic packed bed reactors. Computers & Chemical Engineering 28, 237- 249.

[18] Duran, G.Ricardo., Lombardi, Ariel L., Finite Element Approximation of Convection Diffusion problems using Graded meshes.Mathematics Subject classification. 65N30.

[19] J.M. Le Lann, A. Sargousse, P. Sere Peyrigain, X.Joulia., **2005**. Dynamic Simulation of partial Differential Algebraic Systems: Application to some Chemical Engineering problems. www.simulink/dynamic/modeling.pdf

[20] Ivan Cimrak., Numerical Solution of Degenerate Convection-Diffusion Problem using Broyden Scheme. Conference on Scientific Computing, 14-22, Proceedings of ALGORITMY.

[21] Taggart, N.Mc., Zagorski, R., Finite Difference Scheme for Solving general 3D convection diffusion equation. Computer Physics Communications 164, 318-329.

[22] Fogler, H.Scott., **2002**. Elements of Chemical Reaction Engineering 3 edition. Prentice Hall of India.

[23] Rosenberg, Dale.,Von U.,**1969.** Methods for the numerical solution of Partial Differential equations.,American Elsevier Publishing Company, Inc. New York.

[24] Bird.,Byron. R., Stewart.,E Warren, Lightfoot.,Edwin N., **2002**. Transport Phenomena second edition., John Wiley & sons, Inc.

[25] Mariano, Pinto., Adriano.,Vasco de Toledo.,Coselli.,Eduardo.,Da silva., F.,Jose Marcos.,Maria regina.,W.,Maciel., **2005**. Development of a software for simulation analysis of the phenomenon of the phase change of three phase catalytic slurry reactor. Computers and Chemical Engineering 29, 1639-1378.

[26] Pellegrini.,L.,Ranzi.,G. Biardi., **1989.** Dynamic Model of Packed Bed Tubular Reactors. Computers Chemical Engineering 13, 511-518.

[27] Fink., D. Kurtis., Mathews., H. John.,**2004** Numerical Methods using MATLAB, 4 edition. Prentice Hall of India Private Limited New Delhi.

[28] Pushpavanam.,S. **1998**, Mathematical methods in Chemical Engineering. Prentice Hall of India Private Limited New Delhi.

APPENDIX

Appendix A1

Derivation of Convection Diffusion Equation [1]:

To derive the governing equation we consider the convective diffusion of a solute in a liquid.

Let C = Concentration (Mass/Volume) is a function of the space variables x, y, z and time t.

Concentration Differences give raise to mass transfer by the process of molecular diffusion.

The flux due to diffusion, $\underline{j_D}$ is given by

$$\underline{j_D} = -D\underline{\nabla C} \qquad 1.2\text{-}2$$

Where $\underline{j_D}$ has magnitude equal to the mass of the solute crossing unit area in unit time.

Direction of $\underline{j_D}$ is the direction of flow.

The diffusion constant 'D' (length^2/time) varies with the solute & liquid involved; but is typically about 10^{-9} m^2/s (H_20 & NaCl).

'D' varies with 'C' and Temperature but we shall assume that the changes in these two quantities are sufficiently small to treat D as a constant.

The flux due to convection, $\underline{j_C}$ is given by ;

$$\underline{j_C} = C\underline{V} \qquad 1.2\text{-}3$$

where ; $\underline{V}$ is the liquid velocity.

The total flux of matter ; j is the sum of the equations 1.2-2 & 1.2-3

i.e.

$$\underline{j_C} = C\underline{V} - D\underline{\nabla C} \qquad 1.2\text{-}4$$

The convection Diffusion equation for mass transport is an expression of conservation of matter.

Consider an arbitrary fixed volume V of space with surface S through which the solute is transported.

Thus the rate of flow of solute out of 'V' is;

$$\iint_S \underline{j}.\underline{n}.ds \qquad 1.2\text{-}5$$

Where; $\underline{n}$ is outward normal.

This is equal to the rate of loss of solute within 'V' giving the equation;

$$\iint \underline{j}.\underline{n}.ds = -\frac{\partial}{\partial t}\iiint_V C.dV \qquad 1.2\text{-}6$$

The Left hand side can be changed to a volume integral using the Divergence theorem and time derivative on Right hand side can be taken inside the integral since V is fixed & therefore ;

$$\iiint_V \left(\underline{\nabla}.\underline{j} + \frac{\partial C}{\partial t} \right) dV = 0 \qquad 1.2\text{-}7$$

Since the volume V is arbitrary the integrand is zero; we get;

$$\underline{\nabla}.\underline{j} + \frac{\partial C}{\partial t} = 0 \qquad 1.2\text{-}8$$

Substituting for $\underline{j}$ from equation and assuming that the fluid is incompressible; so that the $\underline{\nabla}.\underline{V} = 0$; we obtain the final form of **Convection Diffusion Equation** as ;

$$\underline{\nabla}.[C\underline{V} - D\underline{\nabla C}] + \frac{\partial C}{\partial t} = 0 \qquad 1.2\text{-}9$$

$$\underline{\nabla} C\underline{V} - D\underline{\nabla}^2 C + \frac{\partial C}{\partial t} = 0 \qquad 1.2\text{-}10$$

$$D\underline{\nabla}^2 C - \underline{V}.\underline{\nabla} C = \frac{\partial C}{\partial t} \qquad 1.2\text{-}11$$

The above equations are the general form of a CONVECTION DIFFUSION equation. Here the fundamental property considered is mass. Similar equations can be formulated for energy and momentum.

The different forms and the applications of these equations are discussed in the chapter of **Introduction.**

Appendix A2

Thomas Algorithm for Tri-diagonal Matrix

The given the linear algebraic equations are written in the following form.

$$b_1u_1 + c_1u_2 + 0 + \ldots\ldots\ldots\ldots + 0 = d_1$$
$$a_2u_1 + b_2u_2 + c_2u_3 + 0 + \ldots\ldots\ldots\ldots + 0 = d_2$$

$$0 + \ldots a_iu_{i-1} + b_iu_i + c_iu_{i+1} + \ldots\ldots\ldots\ldots + 0 = d_i$$

$$0 + \ldots\ldots\ldots + 0 + a_{R-2}u_{R-3} + b_{R-2}u_{R-2} + c_{R-2}u_{R-1} = d_{R-2}$$
$$0 + \ldots\ldots\ldots + 0 + a_{R-1}u_{R-2} + b_{R-1}u_{R-1} = d_{R-1}$$

The equations can be represented in general as;

$$a_iu_{i-1} + b_iu_i + c_iu_{i+1} = d_i \qquad \text{for; } 1 \le i \le R$$

with $a_1 = c_R = 0$

The algorithm is as follows;

First Compute;

$$\beta_i = b_i - \frac{a_ic_{i-1}}{\beta_{i-1}} \quad \text{with } \beta_1 = b_1$$

and;

$$\gamma_i = \frac{d_i - a_i\gamma_{i-1}}{\beta_i} \quad \text{with } \gamma_1 = \frac{d_1}{b_1}$$

The values of the dependant variable are then computed by back substituted from;

$$u_R = \gamma_R \text{ and } u_i = \gamma_i - \frac{c_iu_{i+1}}{\beta_i}$$

The algorithm however suffers a drawback that, if the coefficients b_i ***are assuming very small values, then the algorithm gives erratic results. Further the algorithm is not applicable for the conditions when*** $b_i = 0$

Appendix A4

MATLAB PROGRAMS FOR THE SIMULATION OF CONVECTION DIFFUSION EQUATIONS

```
% A program to calculate the Solution of Convection Diffusion Equations
% for Steady State Conditions.
% Here mention the values of the specified variables as required
% The program simulates the Dependent variable using Finite Differences
Schemes
% Under Steady state conditions the profiles have been generated.
%-------------------------------------------------------------------
% The differential equation used to describe the heat conduction in an
uninsulated, tapered rod which is thin  enough that a one dimensional
analysis can be used.
% The independent variable is x,and the dependant variable is the
temperature i.e. 'u'
% All the variables are normalized and dimensionless.
%-------------------------------------------------------------------
% The differential Equation is basically given as:
% D^2U/Dx^2  +  (2/(q+x))Du/Dx - 2p/(q+x)u=0
% The parameters q = DO/fL
% and  p = hL/k(sqroot(1 + 4/f^2))
% where L is the Length of the rod, H, the heat transfer coefficient ,
k ,the thermal conductivity
% and f and D define the diameter.
% The boundary conditions we assume are the simplest i.e u(0)=0 and
u(1)=1;
%-------------------------------------------------------------------
function steady()
clc;
clf;

q=3;p=7;
delx=input('Input the Required grid size along the x direction  ');
R=1/delx;
for i=1:R

    if(i==1)
     A(i)=(0);
      B(i)=(-2-(2*p*(delx)^2)/(q+delx));
      C(i)= (1+(delx/(q+delx)));
      D(i)=0;
     elseif(i>1 && i<R)

               A(i)= (1-(delx/(q+i*delx)));
               B(i)= (-2-(2*p*(delx)^2)/(q+i*delx));
               C(i)= (1+(delx/(q+i*delx)));
               D(i)=0;
     else
              A(i)=(1-(delx/(q+(R)*delx)));
              B(i)= (-2-(2*p*(delx)^2)/(q+(R)*delx));
              C(i)= 0;
              D(i)=-(1+(delx/(q+((R)*delx))));
    end
end

    for i=1:R
        if(i==1)
            beeta(i)=B(1);
            gamma(i)=D(1)/B(1);
```

```
        else
        beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
        gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
        end
    end

    format long
    x=0:delx:1;
    for i=R:-1:1
        if(i==R)
            U(R)=gamma(R);
        else
        U(i)=(gamma(i)-(C(i)*U(i+1)/beeta(i)));
        end
    end

X=[0,U(1:R-1),1];
%  plot(x,X);
 for(ctr=1:10)

       message1='Enter a choice:';
       choice=input('press "A" to see the Solution Matrix, \n B to see
the specific value of the U at some X value, \n C to see the plot for
entire U along entire X\n Press Q to Quit\n  ','s');
       if(choice=='Q')
           clc;
           break;
       else
       switch(choice)
        case 'A'
             clc;clf;
             disp(X);
        case 'B'
             clc;clf;
         message='  To see the value of the Solution U at a particular
instant of time\n  :';
         time=input('    Enter the value of time :')
         disp(U(time));
        case 'C'
             clc;clf;
         message='  You entered the option to view the plot of U versus
x \n';
         plot(x,X);
       end
       end
 end
disp('You came out of the program')
```

```
% program to calculate the Solution of Convection Diffusion
Equations for Steady State Conditions
% Here mention the values of the specified variables as required
% The program simulates the Dependent variable using Finite
Differences Schemes
% Under Steady state conditions the profiles have been generated.

% The differential equation used to describe the heat conduction in
an uninsulated , tapered rod which is thin enough that a one
dimensional analysis can be used.
% The independent variable is x,and the dependant variable is the
temperature i.e. 'u'
% All the variables are normalized and dimensionless
% The differential Equations is basically given as:
% D^2U/Dx^2  +  (2/(q+x))Du/Dx - 2p/(q+x)u=0
% The parameters q = DO/fL
% and  p = hL/k(sqroot(1 + 4/f^2))
% where L is the Length of the rod, H, the heat transfe coefficient
, k , the thermal conductivity
% and f and D define te diameter.
% The boundary conditions we assume are the simplest i.e du/dx(0)=g
and u(1)=1;

function steady1()
clc;
q=0.2;p=80;
g=input('Enter the value of "g",for the boundary value of the type
du/dx(0)=g \n');
delx=input('Input the Required grid size along the x direction \n  ');
R=1/delx;

for i=1:(R+1)
    if(i==1)
        A(i)=(0);
        B(i)=(-2-(2*p*(delx)^2)/(q));
        C(i)= 2;
        D(i)=2*g*delx*(1-delx/q);

    elseif(i>1 && i<(R+1))

        A(i)= (1-(delx/(q+i*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+i*delx));
        C(i)= (1+(delx/(q+i*delx)));
        D(i)=0;
    else
        A(i)=(1-(delx/(q+i*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+i*delx));
        C(i)= 0;
        D(i)=-(1+(delx/(q+(R*delx))));
    end
end

for i=1:(R+1)
    if(i==1)
        beeta(i)=B(1);
        gamma(i)=D(1)/B(1);
    else
```

```
        beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
        gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
    end
end

format long
x=0:delx:1;
for i=(R+1):-1:1
    if(i==(R+1))
        U(R+1)=gamma(R+1);
    else
        U(i)=(gamma(i)-(C(i)*U(i+1)/beeta(i)));
    end
end
% plot(x,U)
for(ctr=1:10)

    message1='Enter a choice:';
    choice=input('press "A" to see the Solution Matrix, \n B to see the
specific value of the U at some X value, \n C to see the plot for
entire U along entire X\n Press Q to Quit\n  ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp(U);
            case 'B'
                clc;clf;
                message='  To see the value of the Solution U at a
particular instant of time\n  :';
                message='   Enter the particular instant of time you
wish to see the value of Dependant variable:';
                time=input('    Enter the value of time :')
                disp(U(time));
            case 'C'

                message='  You entered the option to view the plot of U
versus x \n';
                figure
                plot(x,U);
                grid on;
                title('PLOT OF DIMENSIONLESS TEMPERATURE(U) V/S
POSITION(X) ');
                xlabel('Position Vector (x)');
                ylabel('Dimensionless Temperature (U)');
        end
    end
end
disp('You came out of the program');
```

```
% A program to calculate the Solution of Convection Diffusion Equation
% for steady state conditions
% Here mention the values of the specified variables as required
% The program simulates the dependent variable using Finite Differences
% Schemes.
% Under Steady state conditions the profiles have been generated.
%-------------------------------------------------------------------
%
% The differential equation used to describe the heat conduction in an
uninsulated, tapered rod which is thin enough that a one dimensional
analysis can be used.
% The independent variable is x,and the dependant variable is the
% temperature i.e. 'U'
% All the variables are normalized and dimensionless.
%-------------------------------------------------------------------
%
% The differntial Equation is basically given as:
% D^2U/Dx^2  +  (2/(q+x))Du/Dx - 2p/(q+x)u=0
% The parameters q = DO/fL
% and  p = hL/k(sqroot(1 + 4/f^2))
% where L is the Length of the rod, H,the heat transfer
coefficient,k,the thermal conductivity
% and f and D define te diameter.
% The boundary conditions we assume are the simplest i.e du/dx(0) -
H*U(0)= -g,
% where H is the Nusselt number and is given as H=hL/k and U(1)=1;
%-------------------------------------------------------------------
function steady2()
q=2.5;p=80;clc;clf;
g=input('Enter the value of "g", in the boundary value of the type
"du/dx(0)-Hu(0)=-g "\n');
H=input('Enter a Value for Nusselt Number i.e H as pr the problem
norms\n');
delx=input('Input the Required grid size along the x direction\n');
R=1/delx;

for i=1:(R+1)
    if(i==1)
        A(i)=(0);
        B(i)= (-2 - (2*H*delx)- (2*(p-H)*delx^2)/q);
        C(i)= 2;
        D(i)= -2*g*delx*(1-delx/q);

    elseif(i>1 && i<(R+1))

        A(i)= (1-(delx/(q+i*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+i*delx));
        C(i)= (1+(delx/(q+i*delx)));
        D(i)=0;
    else
        A(i)=(1-(delx/(q+i*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+i*delx));
        C(i)= 0;
        D(i)=-(1+(delx/(q+(R*delx))));
    end
end
```

```
 for i=1:(R+1)
     if(i==1)
         beeta(i)=B(1);
         gamma(i)=D(1)/B(1);
     else
         beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
         gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
     end
 end

 format long
 x=0:delx:1;
 for i=(R+1):-1:1
     if(i==(R+1))
         U(R+1)=gamma(R+1);
     else
         U(i)=(gamma(i)-(C(i)*U(i+1)/beeta(i)));
     end
 end
 for(ctr=1:10)
     message1='Enter a choice:';
     choice=input('press "A" to see the Solution Matrix, \n B to see the
 specific value of the U at some X value, \n C to see the plot for
 entire U along entire X\n Press Q to Quit\n  ','s');
     if(choice=='Q')
         clc;
         break;
     else
         switch(choice)
             case 'A'
                 disp(U);
             case 'B'
                 clc;clf;
                 message='  To see the value of the Solution U at a
particular instant of time\n  :';
                 message='   Enter the particular instant of time you
wish to see the value of Dependant variable:';
                 time=input('    Enter the value of time :')
                 disp(U(time));
             case 'C'
                 clc;
                 message='  You entered the option to view the plot of U
versus x \n';
                 figure
                 plot(x,U);
                 grid on;
                 title('PLOT OF DIMENSIONLESS TEMPERATURE(U) V/S
POSITION(X) ');
                 xlabel('Position Vector (x)');
                 ylabel('Dimensionless Temperature (U)');
         end
     end
end
disp('You came out of the program');
```

```
% program to calculate the Solution of Convection Diffusion Equation
% for steady state conditions.
% Here mention the values of the specified variables as required.
% The program simulates the dependent variable using Finite
Differences Schemes
% Under Steady state conditions the profiles have been generated.
%-----------------------------------------------------------------
% The differential equation used to describe the heat conduction in
an uninsulated,
% tapered rod which is thin enough that a one dimensional analysis
can be used.
% The independent variable is x,and the dependant variable is the
temperature i.e. 'U'
% All the variables are normalized and dimensionless.
%-----------------------------------------------------------------
% The differential equation is basically given as:
% D^2U/Dx^2  +  (2/(q+x))Du/Dx - 2p/(q+x)u=0
% The parameters q = DO/fL
% and  p = hL/k(sqroot(1 + 4/f^2))
% where L is the Length of the rod,h,the heat transfer
coefficient,k,the thermal conductivity
% and f and D define the diameter.
% The boundary conditions we assume are the simplest i.e du/dx(0) -
H*U(0)= -g,
% where H is the Nusselt number and is given as H=hL/k
% and u(1)=1;
%-----------------------------------------------------------------
function steady3()
q=2.5;p=80;clc;clf;
g=input('Enter the value of "g", the boundary value of the type
du/dx(0)-H*U(0)=-g \n');
H=input('Enter a Value for Nusselt Number i.e H as pr the problem
norms\n');
delx=input('Input the Required grid size along the x direction\n');
R=1/delx;
format long
for i=1:(R)
    format long
    if(i==1)
        B(i)= -(delx/(q+(0.5*delx)))*((2-
H*delx)/(2+H*delx)+2*p*delx)-((2+(3*H*delx))/(2+(H*delx)));
        C(i)= (1+(delx/(q+(0.5*delx))));
        A(i)= 0;
        D(i)=-(1-(delx/(q+0.5*delx)))*((2*g*delx)/(2+(H*delx)));

    elseif(i==R)
        A(i)= 1-(delx/(q+(R-0.5)*delx));
        B(i)= (-3-(((2*p)*(delx)^2)+delx)/(q+(R-0.5)*delx));
        C(i)= 0;
        D(i)= -(2*(1+(delx/(q+(R-0.5)*delx))));
    else
        A(i)= (1-(delx/(q+(i-0.5)*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+(i-0.5)*delx));
        C(i)= (1+(delx/(q+(i-0.5)*delx)));
        D(i)=0;

    end
```

```
 end
 for i=1:R
     format long
     if(i==1)
         beeta(i)=B(1);
         gamma(i)=D(1)/B(1);
     else
         beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
         gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
     end
 end
 format long
 x=0.001:delx:1;
 for i=R:-1:1
     if(i==(R))
         U(R)=gamma(R);
     else
         U(i)=gamma(i)-(C(i)*U(i+1)/beeta(i));
     end
 end
 for(ctr=1:10)

     message1='Enter a choice:';
     choice=input('press "A" to see the Solution Matrix, \n B to see
 the specific value of the U at some X value, \n C to see the plot
 for entire U along entire X\n Press Q to Quit\n  ','s');
     if(choice=='Q')
         clc;
         break;
     else
         switch(choice)
             case 'A'
                 clf;
                 disp(U);
             case 'B'
                 clc;clf;
                 message='  To see the value of the Solution U at a
particular instant of time\n  :';
                 message='   Enter the particular instant of time you
wish to see the value of Dependant variable:';
                 time=input('    Enter the value of time :')
                 disp(U(time));
             case 'C'
                 clc;
                 message='  You entered the option to view the plot
of U versus x \n';
                 figure
                 plot(x,U);
                 grid on;
                 title('PLOT OF DIMENSIONLESS TEMPERATURE(U) V/S
POSITION(X) ');
                 xlabel('Position Vector (x)');
                 ylabel('Dimensionless Temperature (U)');
         end
     end
end
disp('You came out of the program')
```

```
% A program to calculate the Solution of Convection Diffusion Equations
for steady state conditions.
% Here mention the values of the specified variables as required.
% The program simulates the Dependent variable using Finite Differences
Schemes
% Under Steady state conditions the profiles have been generated.
%-----------------------------------------------------------------------
% The differential equation for heat conduction in radial co-ordinates
is similar to that for a tapered rod which goes to a point at one end.
% This is obtained by setting the parameter q=0 in the equation given
below
% The differential Equation is basically given as:
% D^2U/Dx^2  +  (2/(q+x))Du/Dx - 2p/(q+x)u=0
% The parameters q = DO/fL
% and  p = hL/k(sqroot(1 + 4/f^2))
%-----------------------------------------------------------------------
% Further when the parameter "p" is also set to zero so that there is
no heat loss from the surface of the rod by convection, then the above
equation may be treated as the equation of heat conduction in a Sphere.
% Also the heat conduction in a cylinder is similar.
%-----------------------------------------------------------------------
% The independent variable is x,and the dependant variable is the
% temperature i.e.'U'
% All the variables are normalized and dimensionless
%-----------------------------------------------------------------------
%  Here  L  is  the  Length  of  the  rod,  H,the  heat  transfer
coefficient,k,the thermal conductivity
% and f and D define the diameter.
% The boundary conditions we assume are the simplest i.e du/dx(0) -
H*U(0)= -g,
% where H is the Nusselt number and is given as H=hL/k.
% and u(1)=1;
%-----------------------------------------------------------------------

function steady4()
clc;clf;
q=0;p=12;
g=input('Enter  the  value  of  "g",  the  boundary  value  of  the  type
du/dx(0) - H*U(0)= -g \n');
H= p;
delx=input('Input the Required grid size along the x direction\n   ');
R=1/delx;
format long
for i=1:(R)
    format long
    if(i==1)
        B(i)=   -(delx/(q+(0.5*delx)))*((2-H*delx)/(2+H*delx)+2*p*delx)-
((2+(3*H*delx))/(2+(H*delx)));
        C(i)= (1+(delx/(q+(0.5*delx))));
        A(i)= 0;
        D(i)=-(1-(delx/(q+0.5*delx)))*((2*g*delx)/(2+(H*delx)));

    elseif(i==R)
        A(i)= 1-(delx/(q+(R-0.5)*delx));
        B(i)= (-3-(((2*p)*(delx)^2)+delx)/(q+(R-0.5)*delx));
        C(i)= 0;
        D(i)= -(2*(1+(delx/(q+(R-0.5)*delx))));
```

```
    else
        A(i)= (1-(delx/(q+(i-0.5)*delx)));
        B(i)= (-2-(2*p*(delx)^2)/(q+(i-0.5)*delx));
        C(i)= (1+(delx/(q+(i-0.5)*delx)));
        D(i)=0;

    end
end

for i=1:R
    format long
    if(i==1)
        beeta(i)=B(1);
        gamma(i)=D(1)/B(1);
    else
        beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
        gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
    end
end

format long
x=0.001:delx:1;
for i=R:-1:1
    if(i==(R))
        U(R)=gamma(R);
    else
        U(i)=gamma(i)-(C(i)*U(i+1)/beeta(i));
    end
end
X=[U];

for(ctr=1:10)

 message1='Enter a choice:';
 choice=input('press "A" to see the Solution Matrix, \n B to see the
specific value of the U at some X value, \n C to see the plot for
entire U along entire X\n Press Q to Quit\n  ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp(U);
            case 'B'
                clc;clf;
                message='   To see the value of the Solution U at a
particular instant of time\n  :';
                message='    Enter the particular instant of time you
wish to see the value of Dependant variable:';
                time=input('    Enter the value of time :')
                disp(U(time));
            case 'C'
                clc;
                message='  You entered the option to view the plot of U
versus x \n';
```

```
                figure
                plot(x,U);
                grid on;
                title('PLOT   OF   DIMENSIONLESS   TEMPERATURE(U)  . V/S
POSITION(X) ');
                xlabel('Position Vector (x)');
                ylabel('Dimensionless Temperature (U)');

        end
    end
end
disp('You came out of the program');
```

```
% Numerical simulation of Heat(Diffusion) Equation.
%-----------------------------------------------------------------------
% The equation is basically a second order partial differential
equation of parabolic nature. i.e
%
% d^2U/dx^2=du/dt
%
% subject to boundary conditions U(0,t)=0 and U(1,t)=1 for all t;
% we further need a initial condition i.e U(x,0)= 0 for all x<1
%-----------------------------------------------------------------------
% We adopt the method of Finite Differences where by we substitute the
various differential terms in the equation with thier respective
difference equations
% The space and time domain is replaced by respective grid points of
significant step size which has to comply with the stability and
truncation error conditions.
%-----------------------------------------------------------------------
% Here we use the forward differences equations for the analogs of the
partial diferential terms.
%-----------------------------------------------------------------------
function unsteadyforward()
clc;clf;
disp('     Program for simulation of Heat equation          ');
delx=input('\n\nEnter the step size along spatial direction:     ');
delt=input('Enter the step size along the time domain:     ');
format long
for(i=1:10)

    temp=(delt/(delx^2));
    temp1=(delx^2)*.5;
    if(temp>0.5)
        disp('The time step is too large for computation');
        delt=input('Enter a new step size which satsfies the stability
condition');
    else
        break; end;
end

R=1/delx;
N=1/delt;
x=[0:delx:1];
t=[0:delt:1];
for(n=1:N+1)
    U(1,n)=0;
    U(R+1,n)=1;
end
for(i=1:(R+1))
    U(i,1)=0;
end
U(R+1,1)=1;
for(n=2:N+1)
    for(i=2:(R))
U(i,n)=(delt/(delx^2))*(U(i+1,n-1)+U(i-1,n-1))+(1-
(2*delt/(delx^2)))*U(i,n-1);
    end
end
U=U';
```

```
for(ctr=1:10)
    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :   ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
                position=input('Enter the value of exact position :
');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
            case 'C'
                clc;
                disp('You choose to see the values of the U at a
particular instant(t) along entire X');
                time=input('Enter the value of time:    ');
                tp=N*(time)+1;
                for(i=1:R+1)
                    Uinstantaneous(i)=U(tp,i);
                end
                disp(Uinstantaneous);
                figure
                plot(x,Uinstantaneous);
            case 'D'
                clc;
                disp('You entered the option to view the plots of U ');
                st=input('Press 1 to view U v/s X and press 2 to view U
v/s T   ');
                if(st==1)
                    figure
                    plot(x,U);
                    grid on;
                    title('Plot   of   dependant   variable   U   v/s
dimensionless length for a particular time t');
                    xlabel('Position(X)');ylabel('Dependant
variable(U)');
```

```
                elseif(st==2)
                    figure
                    plot(t,U');
                    grid on;
                    title('Plot of dependant variable U v/s
dimensionless time for a particular position(X)')
                    xlabel('Time(t)');ylabel('Dependant variable(U)');
                end
            end
        end
    end
    [ans]= linearparabolicpde(delx,delt);
    for(n=2:N+1)
        for(i=2:R)
            percenterror(n,i)=abs(((ans(n,i)-U(n,i)))/(ans(n,i)))*100;
        end
    end
    disp(percenterror);
    Avgerror=0;
    for(n=2:N)
        for(i=2:R)
            Avgerror=Avgerror+percenterror(n,i);
        end
    end
    disp(Avgerror/((R)*(N)));

    disp('You came out of the program');
```

```
% Simulation of Diffusion Equation using the PDEPE tool box
% ------------------------------------------------------------------
% Here we give the partial differential equation in terms of parameters
% ------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ------------------------------------------------------------------
function [ans]=linearparabolicpde(delx,delt)
clc;
format long;
% x=linspace(0,1,10);
% t=linspace(0,1,10);
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex7pde,@pdex7ic,@pdex7bc,x,t);
u=sol(:,:,1);
disp(u');
figure
plot(x,u');
title('solution of Diffusion equation using PDEPE');
xlabel('Position vector X');
ylabel('Dependant variable U')
```

```
% figure
% surf(x,t,u);
% title('Surface plot for U')
% xlabel('Position vector X');
% ylabel('Time t');
ans=u;

% --------------------------------------------------------------------------
% function [c,f,s]=pdex7pde(x,t,u,DuDx)
% c=1;
% f=DuDx;
% s=0;
%---------------------------------------------------------------------------
% function uo=pdex7ic(x)
% uo=0;
%---------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex7bc(xl,ul,xr,ur,t)
% pl=ul;
% ql=0;
% pr=[ur-1];
% qr=(sin (pi));
%---------------------------------------------------------------------------
```

```
% Numerical simulation of Heat(Diffusion Equation)
%-------------------------------------------------------------------------
% The equation is basically of the type of a second order partial
differential equation of parabolic nature. i.e
% d^2U/dx^2=du/dt
% subject to boundary conditions U(0,t)=0 and U(1,t)=1 for all t;
% we further need a initial condition i.e U(x,0)= 0 for all x<1
%-------------------------------------------------------------------------
% The space and time domain is replaced by respective grid points of
significant step size which has to comply with the stability and
truncation error conditions.
% Here we use the backward differences equations for the analogs of the
%partial diferential terms.
%The backward differences scheme do not impose any stability conditions
and is free to choose any of the value of step size.
%-------------------------------------------------------------------------
function unsteadybackward1()
clc;clf;
delx=input('Enter the step size along spatial direction\n');
delt=input('Enter the step size along the time domain\n');

R=1/delx;
N=1/delt;
x=[0:delx:1];
t=[0:delt:1];
for(n=1:N+1)
    U(1,n)=0;
    U(R+1,n)=1;
end
for(i=1:R)
    U(i,1)=0;
end
%      Formation of tri diagonal system of equations
for(n=2:N+1)
    for(i=2:R)
        if(i==2)
            A(i,n)=0;
            B(i,n)=(-2-(delx^2)/delt);
            C(i,n)=1;
            D(i,n)=(-(delx^2)/delt)*U(i,n-1);
        elseif(i==R)
            A(i,n)=1;
            B(i,n)=(-2-((delx^2))/delt);
            C(i,n)=0;
            D(i,n)=(-(delx^2)/delt)*U(R,n-1)-1;
        else
            A(i,n)=1;
            B(i,n)=(-2-((delx^2))/delt);
            C(i,n)=1;
            D(i,n)=(-(delx^2)/delt)*U(i,n-1);
        end
    end

    for(i=2:R)
        if(i==2)
            beeta(i,n)=B(i,n);
            gamma(i,n)=D(i,n)/B(i,n);
```

```
        else
            beeta(i,n)=(B(i,n)-(A(i,n)*C(i-1,n)/beeta(i-1,n)));
            gamma(i,n)=(D(i,n)-A(i,n)*gamma(i-1,n))/(beeta(i,n));
        end
    end

    format long
    for i=R:-1:2
        if(i==R)
            U(R,n)=gamma(R,n);
        else
            U(i,n)=(gamma(i,n)-((C(i,n)*U(i+1,n))/beeta(i,n)));
        end
    end
end
U=U';
for(ctr=1:100)

    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :   ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
               position=input('Enter the value of exact position :  ');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
          . case 'C'
                clc;
                disp('You choose to see the values of the U at a
particular instant(t) along entire X');
                time=input('Enter the value of time:    ');
                tp=N*(time)+1;
                for(i=1:R+1)
                    Uinstantaneous(i)=U(tp,i);
                end
```

```
                disp(Uinstantaneous);
                plot(x,Uinstantaneous');
            case 'D'
                clc;
                disp('You entered the option to view the plots of
 dependant variable U ');
                st=input('Press 1 to view U v/s X and press 2 to view U
 v/s T   ');
                if(st==2)
                    figure
                    plot(t,U);
                    grid on;
                    title('Plot of dependant variable U v/s
 dimensionless time for particular position X');
                    xlabel('Time (t)');ylabel('Dependant variable(U)');
                elseif(st==1)
                    figure
                    plot(x,U');
                    grid on;
                    title('Plot of dependant variable U v/s
 dimensionless length for a particular time t')
                    xlabel('Position(X)');ylabel('Dependant
 variable(U)');
                end
        end
    end
end
[ans]= linearparabolicpde(delx,delt);
for(n=2:N+1)
    for(i=2:R)
        error(i,n)=abs(((U(n,i)-ans(n,i)))/(U(n,i)))*100;
    end
end
disp(error);
Avgerror=0;
for(n=2:N)
    for(i=2:R)
        Avgerror=Avgerror+error(i,n);
    end
end
disp(Avgerror/((R+1)*(N+1)));

disp('You came out of the program');


% Simulation of Diffusion Equation using the PDEPE tool box
% ---------------------------------------------------------------------
% Here we give the partial differential equation in terms of parameters
% ---------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial
% and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ---------------------------------------------------------------------
function [ans]=linearparabolicpde(delx,delt)
clc;
```

```
format long;
% x=linspace(0,1,10);
% t=linspace(0,1,10);
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex7pde,@pdex7ic,@pdex7bc,x,t);
u=sol(:,:,1);
disp(u');
figure
plot(x,u');
title('solution of Diffusion equation using PDEPE');
xlabel('Position vector X');
ylabel('Dependant variable U')

% figure
% surf(x,t,u);
% title('Surface plot for U')
% xlabel('Position vector X');
% ylabel('Time t');
ans=u;

%--------------------------------------------------------------------------
% function [c,f,s]=pdex7pde(x,t,u,DuDx)
% c=1;
% f=DuDx;
% s=0;
%--------------------------------------------------------------------------
% function uo=pdex7ic(x)
% uo=0;
%--------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex7bc(xl,ul,xr,ur,t)
% pl=ul;
% ql=0;
% pr=[ur-1];
% qr=(sin (pi));
%--------------------------------------------------------------------------
```

```
% Simulation of the Heat Equation using the Crank Nicolson Method
%-------------------------------------------------------------------
% Here the differential Equation is replaced by Difference equation
written about a point half way between two Successive time steps i.e we
consider U(i,n+1/2)
% The method is as such a combination of Forward and Backward
Differences.
% The method do not establish any stablity criteria on the time step
considered.
%However we use a smaller step size for time dimension than the space
%direction in order to prevent heavy oscillations.
%-------------------------------------------------------------------
% Hence we simulate the equation:
% D^2(U)/Dx^2=DU/Dt
% subject to  U(x,0)=0 for all x;
%             U(0,t)=0 for all t;
%             U(1,t)=1 for all t;
% -------------------------------------------------------------------
function cranknicolson1()
clc;clf;
delx=input('Enter the step size along the Spatial Direction x:     ');
delt=input('Enter the step size along the time Domain t:      ');

R=1/delx;
N=1/delt;
x=[0:delx:1];t=[0:delt:1];

for(n=1:N+1)
    U(1,n)=0;
    U(R+1,n)=1;
end
for(i=2:R+1)
    U(i,1)=0;
end
%     Formation of tri diagonal system of equations
for(n=2:N+1)

    for(i=2:R)
        if(i==2)
            A(i,n)=0;
            B(i,n)=(-2-(2*(delx^2)/delt));
            C(i,n)=1;
            D(i,n)=(2-(2*(delx^2)/delt))*U(2,n-1)-U(i+1,n-1);
        elseif(i==R)
            A(i,n)=1;
            B(i,n)=(-2-(2*(delx^2))/delt);
            C(i,n)=0;
            D(i,n)= -U(i-1,n-1)+(2-(2*(delx^2))/delt)*U(R,n-1)-2;
        else
            A(i,n)=1;
            B(i,n)=(-2-(2*(delx^2))/delt);
            C(i,n)=1;
        D(i,n)=-U(i-1,n-1)-U(i+1,n-1)+((2-(2*(delx^2))/delt)*U(i,n-1));
        end
    end
    for(i=2:R)
        if(i==2)
```

```
            beeta(i,n)=B(2,n);
            gamma(i,n)=D(2,n)/B(2,n);
        else
            beeta(i,n)=(B(i,n)-(1/beeta(i-1,n)));
            gamma(i,n)=(D(i,n)-gamma(i-1,n))/(beeta(i,n));
        end
    end
    format long
    for i=R:-1:2
        if(i==R)
            U(R,n)=gamma(R,n);
        else
            U(i,n)=(gamma(i,n)-((C(i,n)*U(i+1,n))/beeta(i,n)));
        end
    end
end
U=U';
for(ctr=1:100)

    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :   ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
               position=input('Enter the value of exact position :  ');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
            case 'C'
                clc;
                disp('You choose to see the values of the U at a
particular instant(t) along entire X');
                time=input('Enter the value of time:    ');
                tp=N*(time)+1;
                for(i=1:R+1)
                    Uinstantaneous(i)=U(tp,i);
```

```
                end
                disp(Uinstantaneous);
                plot(x,Uinstantaneous');
            case 'D'
                clc;
                disp('You entered the option to view the plots of
dependant variable U ');
                st=input('Press 1 to view U v/s X and press 2 to view U
v/s T   ');
                if(st==2)
                    figure
                    plot(t,U);
                    grid on;
                    title('Plot of dependant variable U v/s
dimensionless time for particular position X');
                    xlabel('Time (t)');ylabel('Dependant variable(U)');
                elseif(st==1)
                    figure
                    plot(x,U');
                    grid on;
                    title('Plot of dependant variable U v/s
dimensionless length for a particular time t')
                    xlabel('Position(X)');ylabel('Dependant
variable(U)');
                end
        end
    end
end
[ans]= linearparabolicpde(delx,delt);
for(n=2:N+1)
    for(i=2:R)
        percenterror(n,i)=abs(((U(n,i)-ans(n,i)))/(U(n,i)))*100;
    end
end
disp(percenterror);
Avgerror=0;
for(n=2:N)
    for(i=2:R)
        Avgerror=Avgerror+percenterror(n,i);
    end
end
disp(Avgerror/((R+1)*(N+1)));
%       surf(x,t,U);
disp('You came out of the program');
```

```
% Simulation of Diffusion Equation using the PDEPE tool box
% ---------------------------------------------------------------------
% Here we give the partial differential equation in terms of parameters
% ---------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ---------------------------------------------------------------------
```

```
function [ans]=linearparabolicpde(delx,delt)
clc;
format long;
% x=linspace(0,1,10);
% t=linspace(0,1,10);
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex7pde,@pdex7ic,@pdex7bc,x,t);
u=sol(:,:,1);
disp(u');
figure
plot(x,u');
title('solution of Diffusion equation using PDEPE');
xlabel('Position vector X');
ylabel('Dependant variable U')

% figure
% surf(x,t,u);
% title('Surface plot for U')
% xlabel('Position vector X');
% ylabel('Time t');
ans=u;

% --------------------------------------------------------------------------
% function [c,f,s]=pdex7pde(x,t,u,DuDx)
% c=1;
% f=DuDx;
% s=0;
%--------------------------------------------------------------------------
% function uo=pdex7ic(x)
% uo=0;
%--------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex7bc(xl,ul,xr,ur,t)
% pl=ul;
% ql=0;
% pr=[ur-1];
% qr=(sin (pi));
%--------------------------------------------------------------------------
```

```
% Simulation of the Heat Equation in a tapered Rod using the Crank
Nicolson Method
%-------------------------------------------------------------------
% Here the differential Equation is replaced by Difference equation
written
% about a point half way between two successive values of X i.e
successive
%time steps i.e we consider U(i,n+1/2).
%-------------------------------------------------------------------
%we use the crank nicolson method for descretization
% The method is as such a combination of Forward and Backward
Differences.
% The method do not require to fulfil any stablity criteria on the time
step considered.
%-------------------------------------------------------------------
% Hence we simulate the equation:
%              D^2(U)/Dx^2 + (2/x)[Du/Dx-pU]=DU/Dt
% subject to  U(x,0)=0 for all x;
%            Du/Dx - pU =0 for all t;
%             U(1,t)=1 for all t;
%-------------------------------------------------------------------
function cranknicolson2()
p=10;clc;clf;
delx=input('Enter the step size along the Spatial Direction x:     ');
delt=input('Enter the step size along the time Domain t:      ');
R=1/delx;
N=1/delt;
x=[0:delx:1];t=[0:delt:1]
for(n=1:N+1)
    U(R+1,n)=1;
end
for(i=1:R)
    U(i,1)=0;
end
%      Formation of tri diagonal system of equations
for(n=2:N+1)
    for(i=2:R)
        if(i==2)
            A(i,n)=0;
            B(i,n)=(-2-(4*p*delx)-((2-p*delx)/(2+p*delx))-
((2*(delx^2))/delt));
            C(i,n)=(3);
            D(i,n)=-3*U(i+1,n-1)+(2+(4*p*delx)+((2-p*delx)/(2+p*delx))-
((2*(delx^2))/delt))*U(i,n-1);
        elseif(i==R)
            A(i,n)=((2*R-3)/(2*R-1));
            B(i,n)=(-2-((4*p*delx)/(2*R-1))-((2*R+1)/(2*R-1))-
((2*(delx^2))/delt));
            C(i,n)=0;
            D(i,n)=   -((2*R-3)/(2*R-1))*U(R-1,n-1)+(2+((4*p*delx)/(2*R-
1))+((2*R+1)/(2*R-1))-((2*(delx^2))/delt))*U(R,n-1)-4*((2*R+1)/(2*R-1))
;
        else
            A(i,n)=((2*i-3)/(2*i-1));
            B(i,n)=(-2-((4*p*delx)/(2*i-1))-((2*(delx^2))/delt));
            C(i,n)=(2*i+1)/(2*i-1);
```

```
            D(i,n)=-((2*i-3)/(2*i-1))*U(i-1,n-1)+(2+((4*p*delx)/(2*i-
1))-((2*(delx^2))/delt))*U(i,n-1)-((2*i+1)/(2*i-1))*U(i+1,n-1);
        end
    end

    for(i=2:R)
        if(i==2)
            beeta(i,n)=B(2,n);
            gamma(i,n)=D(2,n)/B(2,n);
        else
            beeta(i,n)=(B(i,n)-(1/beeta(i-1,n)));
            gamma(i,n)=(D(i,n)-gamma(i-1,n))/(beeta(i,n));
        end
    end

    format long
    for i=R:-1:2
        if(i==R)
            U(R,n)=gamma(R,n);
        else
            U(i,n)=(gamma(i,n)-((C(i,n)*U(i+1,n))/beeta(i,n)));
        end
    end
end
U=U';clf;
for(ctr=1:100)
    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :   ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
               position=input('Enter the value of exact position :  ');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
```

```
                case 'C'
                clc;
                disp('You choose to see the values of the U at a
 particular instant(t) along entire X');
                time=input('Enter the value of time:    ');
                tp=N*(time)+1;
                for(i=1:R+1)
                    Uinstantaneous(i)=U(tp,i);
                end
                disp(Uinstantaneous);
                plot(x,Uinstantaneous');
            case 'D'
                clc;
                disp('You entered the option to view the plots of
 dependant variable U ');
    st=input('Press 1 to view U v/s X and press 2 to view U v/s T   ');
                if(st==2)
                    figure
                    plot(t,U);
                    grid on;
                    title('Plot of dependant variable U v/s
 dimensionless time for particular position X');
                    xlabel('Time (t)');ylabel('Dependant variable(U)');
                    elseif(st==1)
                    figure
                    plot(x,U');
                    grid on;
 title('Plot of dependant variable U v/s dimensionless length for a
 particular time t')
 xlabel('Position(X)');ylabel('Dependant variable(U)');
                end
        end
    end
 end
 % disp(unew);
 [ans]= linearparabolicpdel(delx,delt);
 %Error Analysis
 for(n=2:N+1)
    for(i=2:R)
        error(i,n)=abs(((U(n,i)-ans(n,i)))/(U(n,i)))*100;
    end
 end
 disp(error);
 Avgerror=0;
 for(n=2:N+1)
    for(i=2:R)
        Avgerror=Avgerror+error(i,n);
    end
 end
 disp(Avgerror/((R)*(N+1)));
 disp('You came out of the program');


 % Simulation of Diffusion Equation using the PDEPE tool box
 % ------------------------------------------------------------------
 % Here we give the partial differential equation in terms of parameters
```

```
% ------------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ------------------------------------------------------------------------
function [ans]=linearparabolicpdel(delx,delt)
clc;
format long;
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex8pde,@pdex8ic,@pdex8bc,x,t);
u=sol(:,:,1);
disp(u);
figure
plot(x,u);
title('solution of Heat flow in a tapered rod using PDEPE');
xlabel('Position vector X');
ylabel('Dependant variable U');

figure
surf(x,t,u);
xlabel('Position vector X');
ylabel('Time t');
ans=[u];
% ------------------------------------------------------------------------
% function [c,f,s]=pdex8pde(x,t,u,DuDx)
% c=1;p=10;
% f=DuDx;
% s= (2/x)*(DuDx-p*u);
%------------------------------------------------------------------------
% function uo=pdex8ic(x)
% uo=0;
%------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex8bc(xl,ul,xr,ur,t)
% p=10;
% pl=(-p*ul);
% ql=1;
% pr=[ur-1];
% qr=0;
%------------------------------------------------------------------------
```

```
% Program to Simulate the Differential equations describing the
Diffusion & Reaction on the surface of Catalyst pellet under Steady
State conditions (Shrinking core model)
%-------------------------------------------------------------------
% Here the differential equation is given in terms of dimensionless
form
% D^2U/Dx^2 + (2/x)* DU/Dx - phi^2* U =0;
% subject to  BC (1) U=1          for x=1;
%             BC (2) U=finite     for x=0;
%-------------------------------------------------------------------
% The descretization is done using Finite Differences
% The computed solutions are then compared with the analytical results
for error analysis.
% The average percentage error is calculated and is found to be very
close to zero.
%-------------------------------------------------------------------
function shrinkingcore()
clc;clf;
delx=input('Input the Required grid size along the x direction\n');
R=1/delx;phi=10.0;
U(1)=0;
U(R+1)=1;
for i=1:(R)
    if(i==1)
        A(i)=(0);
        B(i)=(-2-(phi^2)*(delx^2));
        C(i)= (1+(1/(i+1)));
        D(i)= 0;
    elseif(i>2 && i<(R))
        A(i)=(1-(1/(i+1)));
        B(i)= (-2-(phi^2)*(delx^2));
        C(i)= (1+(1/(i+1)));
        D(i)=0;
    else
        A(i)=(1-(1/(i+1)));
        B(i)= (-2-(phi^2)*(delx^2));
        C(i)= 0;
        D(i)= -(1+(1/(i+1)))*U(i+1);
    end
end
for i=2:(R)
    if(i==2)
        beeta(i)=B(2);
        gamma(i)=D(2)/B(2);
    else
        beeta(i)=(B(i)-(A(i)*C(i-1))/beeta(i-1));
        gamma(i)=(D(i)-A(i)*gamma(i-1))/(beeta(i));
    end
end

format long
x=0:delx:1;
for i=(R):-1:2
    if(i==(R))
        U(R)=gamma(R);
    else
        U(i)=(gamma(i)-(C(i)*U(i+1)/beeta(i)));
```

```
    end
end

%Analytical solution
Uanalytical(1)=U(1);
Uanalytical(R+1)=U(R+1);
y=delx:delx:1;

for(i=1:(R))
    A(i)=sinh (phi*(i*delx));
    B=sinh (phi);
    C(i)=(A(i)/B);
    Uanalytical(i)= (1/(i*delx))*C(i);
end
plot(x,Uanalytical);
error=(Uanalytical-U);
disp(error);
for(i=2:R+1)
    Avgerror(i)=error(i)/Uanalytical(i)*100;
end
disp(Avgerror);

for(ctr=1:10)
    message1='Enter a choice:';
    choice=input('press "A" to see the Solution Matrix, \n B to see the
specific value of the U at some X value, \n C to see the plot for
entire U along entire X\n Press Q to Quit\n  ','s');
    if(choice=='Q')
        clc;clf;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp(U);
            case 'B'
                clc;clf;
                message='   To see the value of the Solution U at a
particular instant of time\n  :';
                message='    Enter the particular instant of time you
wish to see the value of Dependant variable:';
                time=input('    Enter the value of time :')
                disp(U(time));
            case 'C'
                clc;
                message='  You entered the option to view the plot of U
versus x \n';
                plot(x,U);
          end
    end
end
disp('You came out of the program');
```

```
%Simulation of an Isothermal Flow Reactor with Second order Reaction
%-------------------------------------------------------------------------
%program to calculate the solution fo quasi linear partial differential
equations
% the general form of the equation is :
% [a(U)]D^2U/Dx^2 + [b(U)]DU/Dx  +  [c(U)]U  =  DU/Dt;
%-------------------------------------------------------------------------
% As an example we solve the Isothermal Flow Reactor with Second Order
Reaction:
%   The model equation for this system is given as :
%   D^2U/Dx^2 - s*DU/Dx - rU^2 = DU/Dt;
%    subject to the boundary Conditions :
%    U(x,0)= 0 for all x;
%        DU/Dx =0 at x=1 and all t;
%        DU/Dx + s*(1-U)= 0 at x=0 all t;
%     Here we shall use the Crank-Nicolson scheme for developing the
difference
% equations.
%-------------------------------------------------------------------------
%The simulation gives the dimensionless Concentration as a function of
time and space(both dimensionless).
%-------------------------------------------------------------------------
function flowreactor()
clc;
disp('                  Simulation of Flow Reactor       ')
delx= input('\nEnter the step size along the X direction:       ');
delt= input('Enter the step size along the time direction:      ');
S=10;r=5;x=(0:delx:1);
format long;
R=(1/delx);
N=(1/delt);
U=zeros(R+1,N+1);
for(i=1:R+1)
    U(i,1)=0;
end
for(n=2:N+1)
    for(i=2:R+1)
        if(i==2)
            Upre(i,n)=U(i,n-1)+(delt/(2*delx^2))*((1-S*delx/2)*U(i+1,n-
1)+S*delx -(1+S*delx/2+r*(delx^2)*U(2,n-1))*U(2,n-1));
       A(i,n)= 0;
       B(i,n)=(-r*(delx^2)*Upre(i,n)-((2*(delx)^2)/delt)-(1+S*delx/2));
       C(i,n)= (1-(S*delx)/2);
       D(i,n)=-(1-(S*delx)/2)*U(i+1,n-1)+(r*(delx^2)*Upre(i,n)-
((2*delx^2)/delt)+(1+S*delx/2))*U(i,n-1)-(2*S*delx);

        elseif(i>2 && i<(R+1))
            Upre(i,n)=U(i,n-1)+(delt/(2*delx^2))*((1+S*delx/2)*U(i-1,n-
1)+(1-S*delx/2)*U(i+1,n-1)-(2+r*(delx^2)*U(i,n-1))*U(i,n-1));
            A(i,n)= (1+(S*delx)/2);
            B(i,n)= (-2-r*(delx^2)*Upre(i,n)-(2*(delx^2)/delt));
            C(i,n)= (1-(S*delx)/2);
            D(i,n)= -(1+(S*delx)/2)*U(i-1,n-1)- (1-(S*delx)/2)*U(i+1,n-
1)+ (2+r*(delx^2)*Upre(i,n)-((2*delx^2)/delt))*U(i,n-1);

        else
```

```
            Upre(i,n)=U(i,n-1)+(delt/(2*(delx^2)))*((1+S*delx/2)*U(i-
1,n-1)-(1+S*delx/2+(r*delx^2)*U(i,n-1))*U(i,n-1));
       A(i,n)=(1+(S*delx)/2);
       B(i,n)=(-r*(delx^2)*Upre(i,n)-((2*(delx)^2)/delt)-(1+S*delx/2));
       C(i,n)= 0;
       D(i,n)=-(1+(S*delx)/2)*U(i-1,n-1)+(r*(delx^2)*Upre(i,n)-
((2*delx^2)/delt)+(1+S*delx/2))*U(i,n-1);
        end
    end
    for(i=2:R+1)
        if(i==2)
            beta(i,n)=B(2,n);
            gamma(i,n)=D(2,n)/B(2,n);
        else
            beta(i,n)=B(i,n)-(A(i,n)*C(i-1,n)/beta(i-1,n));
            gamma(i,n)=(D(i,n)-A(i,n)*gamma(i-1,n))/beta(i,n);
        end
    end
    for(i=(R+1):-1:2)
        if(i==(R+1))
            U(R+1,n)=gamma(R+1,n);
        else
            U(i,n)=gamma(i,n)-(C(i,n)*U(i+1,n)/beta(i,n));
        end
    end
end
t=(0:delt:1);
U=U';
for(ctr=1:100)

    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :   ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
               position=input('Enter the value of exact position :  ');
                xp= R*(position)+1;
```

```
                    tp=N*(time)+1;
                    disp(U(tp,xp));
                case 'C'
                    clc;
                    disp('You choose to see the values of the U at a
  particular instant(t) along entire X');
                    time=input('Enter the value of time:    ');
                    tp=N*(time)+1;
                    for(i=1:R+1)
                        Uinstantaneous(i)=U(tp,i);
                    end
                    disp(Uinstantaneous);
                    plot(x,Uinstantaneous');
                case 'D'
                    clc;
                    disp('You entered the option to view the plots of
 dependant variable U ');
   st=input('Press 1 to view U v/s X and press 2 to view U v/s T   ');
                    if(st==2)
                        figure
                        plot(t,U);
                        grid on;
    title('Plot of dependant variable U v/s dimensionless time for
 particular position X');
                        xlabel('Time (t)');ylabel('Dependant variable(U)');
                    elseif(st==1)
                        figure
                        plot(x,U');
                        grid on;
  title('Plot of dependant variable U v/s dimensionless length for a
 particular time t');
                        xlabel('Position(X)');ylabel('Dependant
 variable(U)');
                    end
            end
        end
    end
    disp('You came out of the program');
    [ans]= nonlinearparabolicpdel(delx,delt);
    %Error Analysis
    for(n=2:N+1)
        for(i=2:R+1)
            error(i,n)=abs(((U(n,i)-ans(n,i)))/(U(n,i)))*100;
        end
    end
    disp(error);
    totalerror=0;
    for(n=2:N+1)
        for(i=2:R+1)
            totalerror=totalerror+error(i,n);
        end
    end
    Avgerror=(totalerror/((R+1)*(N+1)));
    disp(Avgerror);
```

```
% Simulation of Isothermal Flow Reactor(with a second order reaction)
% using the PDEPE tool box
% ------------------------------------------------------------------------
% Here we give the partial differntial equation in terms of parameters
% ------------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ------------------------------------------------------------------------
function [ans]= nonlinearparabolicpde1(delx,delt)
clc;
format long;
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u=sol(:,:,1);
disp(u);
figure
plot(x,u');
title('Concentration profiles in a flow reactor with second order
reaction using PDEPE');
xlabel('Position vector X');
ylabel('Dependant variable U')
ans=[u];
% figure
% surf(x,t,u);
% xlabel('Position vector X');
% ylabel('Time t');
% ------------------------------------------------------------------------
% function [c,f,s]=pdex1pde(x,t,u,DuDx)
% c=1;
% a=10;r=5;
% f=DuDx;
% s=(-r*(u^2))-a*DuDx;
%------------------------------------------------------------------------
% function uo=pdex1ic(x)
% uo=0;
%------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex1bc(xl,ul,xr,ur,t)
% a=10;
% pl=a*(1-ul);
% ql=1;
% pr=0;
% qr=1;
%------------------------------------------------------------------------
```

```
% Countercurrent Heat Exchanger
%-------------------------------------------------------------------------
% The heat balance equations give a set of Hyperbolic Equations which
are to be solved using Central finite Differences
%-------------------------------------------------------------------------
% Here we consider a unsteady state Countercurrent Heat Exchanger
% The Equations describing the problem are:
%     -b1(DU/Dx)-c1(U-V)=(Du/Dt)
%     b2(Dv/Dx)+c2(U-V)=(Dv/Dt)
%     subject to following boundary Conditions
%        U(x,0)=0    for x>0;
%        U(0,0)=1;
%        v(x,0)=0;
%        U(0,t)=1;
%        V(1,t)=0;
%-------------------------------------------------------------------------
%For simplicity we take same step size along time and space domain
%-------------------------------------------------------------------------
function CountercurrentHE()
clc;clf;
delx=input('Input the step Size along the spatial Direction:     ');
delt=delx;
R=(1/delx);
N=(1/delt);
x=[0:delx:1];
for(n=1:N+1)
    U(1,n)=1;
end
for(n=1:N+1)
    V(R+1,n)=0;
end
for(i=1:R+1)
    U(i,1)=0;
    V(i,1)=0;
end
U(1,1)=1;
format long
U(1,1)=1;
bone=1;
btwo=0.5;
cone=0.05;
ctwo=0.05;
for(n=2:(N+1))
    for(i=2:(R))
        if(i==2)
            Aone(i,n)=0;
            Atwo(i,n)=0;
            Bone(i,n)=((bone/delx)+(cone/2)+(1/delt));
            Btwo(i,n)=(-cone/2);
            Cone(i,n)=(0);
            Ctwo(i,n)=(-cone/2);
            Done(i,n)=((bone/delx)-(cone/2)+(1/delt))*U(i-1,n-1)+((-
bone/delx)-(cone/2)+(1/delt))*U(i,n-1)+(cone/2)*(V(i+1,n-1)+V(i,n-1))-
((-bone/delx)+(cone/2)+(1/delt))*U(i-1,n);
            Athree(i,n)=0;
            Afour(i,n)=0;
            Bthree(i,n)=(ctwo/2);
```

```
            Bfour(i,n)=(-btwo/delx)-(ctwo/2)-(1/delt);
            Cthree(i,n)=0;
            Cfour(i,n)=((btwo/delx)-(ctwo/2)-(1/delt));
            Dtwo(i,n)=                          (-ctwo/2)*(U(i,n-1)+U(i-1,n-
1))+((btwo/delx)+(ctwo/2)-(1/delt))*V(i,n-1)+((-btwo/delx)+(ctwo/2)-
(1/delt))*V(i+1,n-1)-(ctwo/2)*U(i-1,n);
        elseif(i==R)
            Aone(i,n)=(-bone/delx+(cone/2)+(1/delt));
            Atwo(i,n)=0;
            Bone(i,n)=((bone/delx)+(cone/2)+(1/delt));
            Btwo(i,n)=(-cone/2);
            Cone(i,n)=(0);
            Ctwo(i,n)=0;
            Done(i,n)=((bone/delx)-(cone/2)+(1/delt))*U(i-1,n-1)+((-
bone/delx)-(cone/2)+(1/delt))*U(i,n-1)+(cone/2)*(V(i+1,n-1)+V(i,n-1))-
(-cone/2)*V(i+1,n);
            Athree(i,n)=(ctwo/2);
            Afour(i,n)=0;
            Bthree(i,n)=(ctwo/2);
            Bfour(i,n)=((-btwo/delx)-(ctwo/2)-(1/delt));
            Cthree(i,n)=0;
            Cfour(i,n)=0;
            Dtwo(i,n)=(-ctwo/2)*(U(i,n-1)+U(i-1,n-
1))+((btwo/delx)+(ctwo/2)-(1/delt))*V(i,n-1)+((-btwo/delx)+(ctwo/2)-
(1/delt))*V(i+1,n-1)-((btwo/delx)-(ctwo/2)-(1/delt))*V(i+1,n);
        else
            Aone(i,n)=(-bone/delx+(cone/2)+(1/delt));
            Atwo(i,n)=0;
            Bone(i,n)=((bone/delx)+(cone/2)+(1/delt));
            Btwo(i,n)=(-cone/2);
            Cone(i,n)=(0);
            Ctwo(i,n)=(-cone/2);
            Done(i,n)=((bone/delx)-(cone/2)+(1/delt))*U(i-1,n-1)+((-
bone/delx)-(cone/2)+(1/delt))*U(i,n-1)+(cone/2)*(V(i+1,n-1)+V(i,n-1));
            Athree(i,n)=(ctwo/2);
            Afour(i,n)=0;
            Bthree(i,n)=(ctwo/2);
            Bfour(i,n)=((-btwo/delx)-(ctwo/2)-(1/delt));
            Cthree(i,n)=0;
            Cfour(i,n)=((btwo/delx)-(ctwo/2)-(1/delt));
            Dtwo(i,n)=                          (-ctwo/2)*(U(i,n-1)+U(i-1,n-
1))+((btwo/delx)+(ctwo/2)-(1/delt))*V(i,n-1)+((-btwo/delx)+(ctwo/2)-
(1/delt))*V(i+1,n-1);
        end
    end
    for(i=2:(R))
        if(i==2)
            betaone(i,n)=Bone(i,n);
            betatwo(i,n)=Btwo(i,n);
            betathree(i,n)=Bthree(i,n);
            betafour(i,n)=Bfour(i,n);
            mu(i,n)=betaone(i,n)*betafour(i,n)-
betatwo(i,n)*betathree(i,n);

            deltaone(i,n)=Done(i,n);
            deltatwo(i,n)=Dtwo(i,n);
```

```
            lambdaone(i,n)=((betafour(i,n)*Cone(i,n))-
(betatwo(i,n)*Cthree(i,n)))/(mu(i,n));
            lambdatwo(i,n)=((betafour(i,n)*Ctwo(i,n))-
(betatwo(i,n)*Cfour(i,n)))/(mu(i,n));
            lambdathree(i,n)=((betaone(i,n)*Cthree(i,n))-
(betathree(i,n)*Cone(i,n)))/(mu(i,n));
            lambdafour(i,n)=((betaone(i,n)*Cfour(i,n))-
(betathree(i,n)*Ctwo(i,n)))/(mu(i,n));

            gammaone(i,n)=((betafour(i,n)*deltaone(i,n))-
(betatwo(i,n)*deltatwo(i,n)))/(mu(i,n));
            gammatwo(i,n)=((betaone(i,n)*deltatwo(i,n))-
(betathree(i,n)*deltaone(i,n)))/(mu(i,n));
        else
            betaone(i,n)=(Bone(i,n)-(Aone(i,n)*lambdaone(i-1,n))-
(Atwo(i,n)*lambdathree(i-1,n)));
            betatwo(i,n)=(Btwo(i,n)-(Aone(i,n)*lambdatwo(i-1,n))-
(Atwo(i,n)*lambdafour(i-1,n)));
            betathree(i,n)=(Bthree(i,n)-(Athree(i,n)*lambdaone(i-1,n))-
(Afour(i,n)*lambdathree(i-1,n)));
            betafour(i,n)=(Bfour(i,n)-(Athree(i,n)*lambdatwo(i-1,n))-
(Afour(i,n)*lambdafour(i-1,n)));

            mu(i,n)=betaone(i,n)*betafour(i,n)-
betatwo(i,n)*betathree(i,n);

            deltaone(i,n)=(Done(i,n)-(Aone(i,n)*gammaone(i-1,n))-
(Atwo(i,n)*gammatwo(i-1,n)));
            deltatwo(i,n)=(Dtwo(i,n)-(Athree(i,n)*gammaone(i-1,n))-
(Afour(i,n)*gammatwo(i-1,n)));

            lambdaone(i,n)=((betafour(i,n)*Cone(i,n))-
(betatwo(i,n)*Cthree(i,n)))/(mu(i,n));
            lambdatwo(i,n)=((betafour(i,n)*Ctwo(i,n))-
(betatwo(i,n)*Cfour(i,n)))/(mu(i,n));
            lambdathree(i,n)=((betaone(i,n)*Cthree(i,n))-
(betathree(i,n)*Cone(i,n)))/(mu(i,n));
            lambdafour(i,n)=((betaone(i,n)*Cfour(i,n))-
(betathree(i,n)*Ctwo(i,n)))/(mu(i,n));

            gammaone(i,n)=((betafour(i,n)*deltaone(i,n))-
(betatwo(i,n)*deltatwo(i,n)))/(mu(i,n));
            gammatwo(i,n)=((betaone(i,n)*deltatwo(i,n))-
(betathree(i,n)*deltaone(i,n)))/(mu(i,n));

        end
    end
    for(i=(R):-1:2)
        if(i==(R))
            U(R,n)=gammaone(R,n);
            V(R,n)=gammatwo(R,n);
        else
            U(i,n)=(gammaone(i,n)-(lambdaone(i,n)*U(i+1,n))-
(lambdatwo(i,n)*V(i+1,n)));
            V(i,n)=(gammatwo(i,n)-(lambdathree(i,n)*U(i+1,n))-
(lambdafour(i,n)*V(i+1,n)));
        end
```

```
    end
end

U=U';V=V';
for(ctr=1:100)

    disp('press "A" to see the entire Solution Matrix for U and V.')
    disp('press B to see the specific value of the U and at some X & T
values');
    disp('press C to see the values of the dependant variables at a
particular instant U(T) and V(T) for all x');
    disp('press D to see the plot for entire U or V along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:    ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
                disp('The solution matrix V is :: ');
                disp(V);
            case 'B'
                clc;clf;
                disp('    You choose to see the value of the U and V at
a particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variables:    ');
                time=input('Enter the value of time :    ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
                position=input('Enter the value of exact position :
');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
                disp(V(tp,xp));
            case 'C'
                clc;
                disp('You choose to see the values of the U at a
particular instant(t) along entire X');
                time=input('Enter the value of time:     ');
                %                U=U';
                tp=N*(time)+1;
                option=input('Press 1 for getting values of U and press
2 for getting values of V::  ');
                if(option==1)
                    for(i=1:R+1)
                        Uinstantaneous(i)=U(tp,i);
                    end
                    disp(Uinstantaneous);
                    plot(x,Uinstantaneous);
                end
                if(option==2)
```

```
                        for(i=1:R+1)
                            Vinstantaneous(i)=V(tp,i);
                        end
                        disp(Vinstantaneous);
                        plot(x,Vinstantaneous);
                    end
                case 'D'
                    clc;
                    disp('You entered the option to view the plot of U
 versus entire X and Time ');
                    option=input('Press 1 to view the plot for U and press
 2 for viewing plot for V::    ' );
                    if(option==1)
                        figure
                        plot(x,U);
                    end
                    if(option==2)
                        figure
                        plot(x,V);
                    end
        end
    end
end
disp('You came out of the Program');
```

```
% program to solve the parabolic and hyperbolic equations with non
% linear boundary conditions.
%-------------------------------------------------------------------------
% The resulting differential equations are converted to difference
equations
% that are converted to tridiagonal system ,which is there by solved
using Thomas Algorithm.
%-------------------------------------------------------------------------
% As an example we solve the heat equation
% D^2U/Dx^2 = DU/Dt;
% subject to the boundary conditions :
% U=Uo      at x = 0 for all t;
% s(1-U^4) - DU/Dx = 0 at x =1; all t;
% and U = Uo at t = 0 for all x;
%-------------------------------------------------------------------------
% The parameter s in the equation is dimensionless and contain the cube
of the source temperature, the Stephen Boltzmann constant and thermal
conductivity and length of the rod.
%-------------------------------------------------------------------------
% We use the crank Nicolson equation for the partial differential
equation
%-------------------------------------------------------------------------
function nonlinearboundary()
clc;clf;
delx=input('Enter a specific step size along X direction:      ');
delt=input('Enter a specific step size along T direction:      ');
format long;
R=(1/delx);
N=(1/delt);s=0.1;
U=zeros(R+1,N+1);
Uo=input('Enter a particular initial value of the variable U at x=0 and
for all t:    ');
for(n=1:N+1)
    U(1,n)=Uo;
end
for(i=1:R+1)
    U(i,1)= Uo;
end
for(n=2:N+1)
    for(ctr=1:100)
        for(i=2:R+1)
            if(i==2)
                A(i,n)= 0;
                B(i,n)= (-2-(2*(delx^2))/delt);
                C(i,n)= 1;
                D(i,n)= (2-(2*(delx^2))/delt)*U(i,n-1)-U(i+1,n-1)-2*Uo;
                beta(i,n)=B(2,n);
                gamma(i,n)=D(2,n)/B(2,n);
            elseif(i>2 && i<(R+1))
                A(i,n)= 1;
                B(i,n)= (-2-(2*(delx^2))/delt);
                C(i,n)= 1;
                D(i,n)=-U(i-1,n-1)+(2-(2*(delx^2))/delt)*U(i,n-1)-
U(i+1,n-1);
                beta(i,n)=B(i,n)-(A(i,n)*C(i-1,n)/beta(i-1,n));
                gamma(i,n)=(D(i,n)-A(i,n)*gamma(i-1,n))/beta(i,n);
            else
```

```
                    A(i,n)= 2;
                    B(i,n)=(-2-(delx^2)/delt);
                    C(i,n)= 0;
                    h= -((delx^2/delt*U(i,n-1))+2*s*delx);
                    g=(2*s*delx);
                    beta(i,n)=B(i,n)-(A(i,n)*C(i-1,n)/beta(i-1,n));
                    p= ((h-A(i,n)*gamma(i-1,n))/beta(i,n));
                    q= g/beta(i,n);
                    if(n==2)
                        U1=U(i,n-1);
                        disp(U1);
                        U2=U(i,n-1)+2*(U(i,n-1)-0);
                    else
                        U1=U(i,n-1);
                        U2=U(i,n-1)+2*(U(i,n-1)-U(i,n-2));
                    end
                    U(i,n)=regulafalsi(p,q,U1,U2);
                end
            end
            for(i=(R):-1:2)
                U(i,n)=gamma(i,n)-(C(i,n)*U(i+1,n)/beta(i,n));
            end
            for(i=2:R+1)
                Um(i,ctr)=U(i,n);
            end
            if(ctr==1)
                continue;
            else
                if(Um(i,ctr)-Um(i,(ctr-1))>=0.0000000005)
                    break;
                else
                    continue;
                end
            end
        end
    end
end
x=(0:delx:1); t=(0:delt:1);
U=U';
for(ctr=1:100)
    disp('press "A" to see the entire Solution Matrix.')
    disp('press B to see the specific value of the U at some X & T
values');
    disp('press C to see the values of the dependant variable at a
particular instant U(T)for all x');
    disp('press D to see the plot for entire U along entire X');
    disp('Press Q to Quit  ');
    choice= input('Enter a choice:   ','s');
    if(choice=='Q')
        clc;
        break;
    else
        switch(choice)
            case 'A'
                clf;
                disp('The Solution matrix U is :: ');
                disp(U);
            case 'B'
```

```
                clc;clf;
                disp('    You choose to see the value of the U at a
particular instant of time and position     ');
                disp('Enter the particular instant of time you wish to
see the value of Dependant variable:   ');
                time=input('Enter the value of time :    ');
                disp(' Enter the position along the X direction, you
wish to see the value of Dependant variable:');
                position=input('Enter the value of exact position :
');
                xp= R*(position)+1;
                tp=N*(time)+1;
                disp(U(tp,xp));
            case 'C'
                clc;
                disp('You choose to see the values of the U at a
particular instant(t) along entire X');
                time=input('Enter the value of time:    ');
                tp=N*(time)+1;
                for(i=1:R+1)
                    Uinstantaneous(i)=U(tp,i);
                end
                disp(Uinstantaneous);
                plot(x,Uinstantaneous');
            case 'D'
                clc;
                disp('You entered the option to view the plots of
dependant variable U ');
                st=input('Press 1 to view U v/s X and press 2 to view U
v/s T   ');
                if(st==2)
                    figure
                    plot(t,U');
                    grid on;
                    title('Plot   of   dependant   variable   U   v/s
dimensionless time for particular position X');
                    xlabel('Time (t)');ylabel('Dependant variable(U)');
                elseif(st==1)
                    figure
                    plot(x,U);
                    grid on;
                    title('Plot   of   dependant   variable   U   v/s
dimensionless length for a particular time t');
                    xlabel('Position(X)');ylabel('Dependant
variable(U)');
                end
        end
    end
end
 [ans]= radiationdiffusion(delx,delt);
%Error Analysis
for(n=1:N+1)
    for(i=1:R+1)
        error(i,n)=abs((ans(i,n)-U(n,i))/(ans(i,n)))*100;
    end
end
disp(error);
```

```
Avgerror=0;
for(n=1:N+1)
    for(i=1:R+1)
        Avgerror=Avgerror+error(i,n);
    end
end
disp(Avgerror/((R+1)*(N+1)));
disp('You came out of the program');



% Simulation of Heat flow in a rod(Conduction accompanied by Radiation
along Boundary) using the PDEPE tool box
% ------------------------------------------------------------------------
% Here we give the partial differntial equation in terms of parameters
% ------------------------------------------------------------------------
% Functions are written for the differential equation as well as for
the initial and boundary conditions.
% The tool box simulates the solution using thre Finite Element
Analysis.
% ------------------------------------------------------------------------
function [ans]= radiationdiffusion(delx,delt)
clc;
format long;
x=[0:delx:1];
t=[0:delt:1];
m=0;
sol = pdepe(m,@pdex12pde,@pdex12ic,@pdex12bc,x,t);
u=sol(:,:,1);
disp(u);
figure
plot(x,u);
title('Dependant variable(U) v/s Position as solved by PDEPE');
xlabel('Position vector(X)');
ylabel('Dependant variable U');
[ans]=u';
% figure
% surf(x,t,u);
% xlabel('Position vector X');
% ylabel('Time t');
% ------------------------------------------------------------------------
% function [c,f,s]=pdex12pde(x,t,u,DuDx)
% c=1;
% f=DuDx;
% s=0;
%------------------------------------------------------------------------
% function uo=pdex12ic(x)
% uo=0.50;
%------------------------------------------------------------------------
% function [pl,ql,pr,qr]=pdex12bc(xl,ul,xr,ur,t)
% uo=0.50;s=0.1;
% pl=ul-uo;
% ql=0;
% pr=s*(1-(ur)^4);
% qr=-1;
%------------------------------------------------------------------------
```

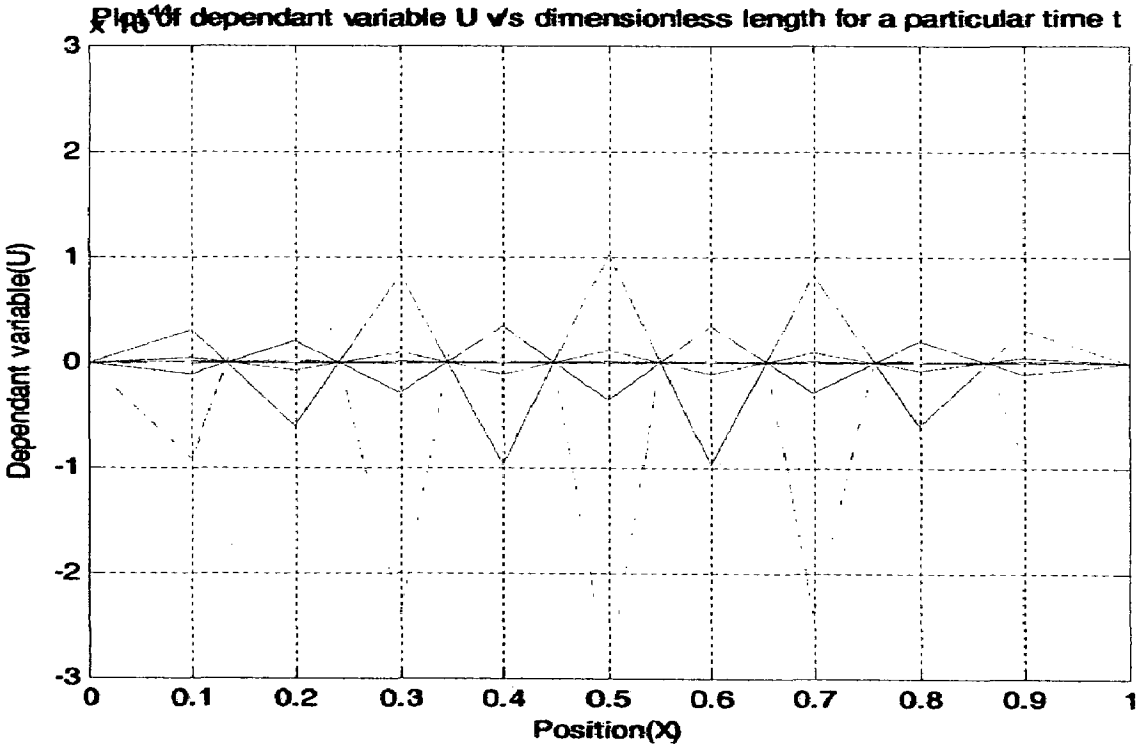


Figure: 5.7 Diffusion equation (Forward Differences) $\Delta x = \Delta t = 0.1$

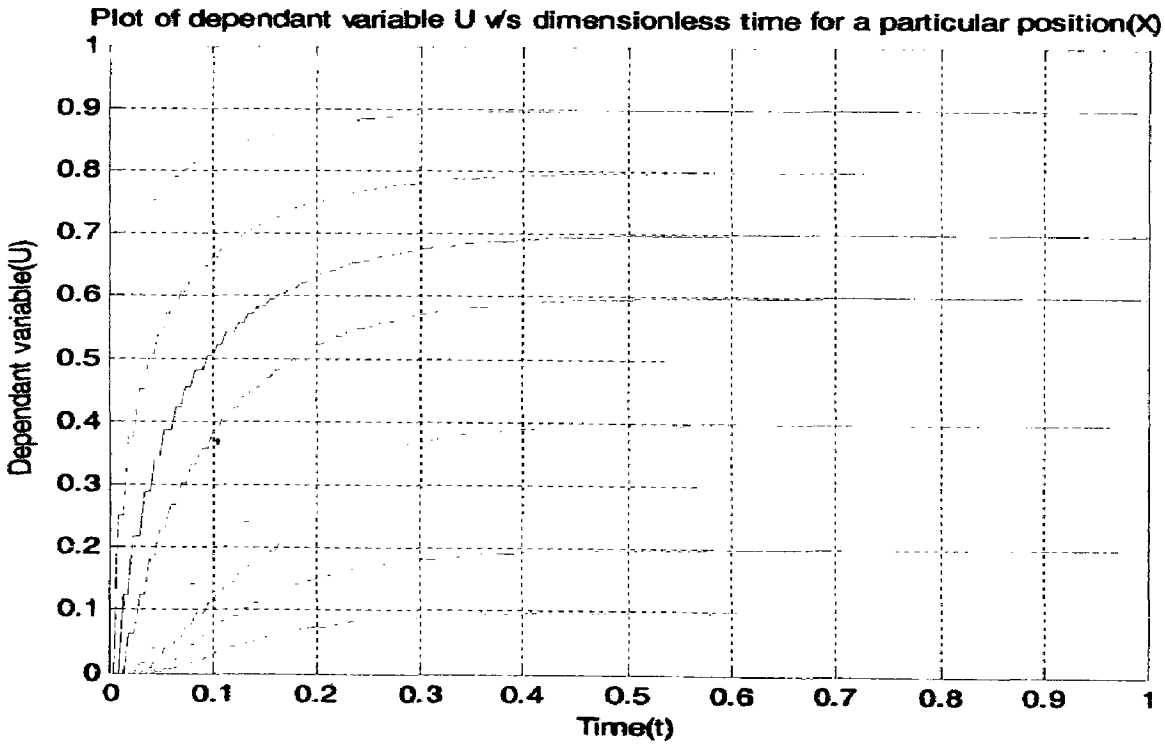


Figure: 5.8 Diffusion equation (Forward Differences) $\Delta x = 0.1$, $\Delta t = 0.005$

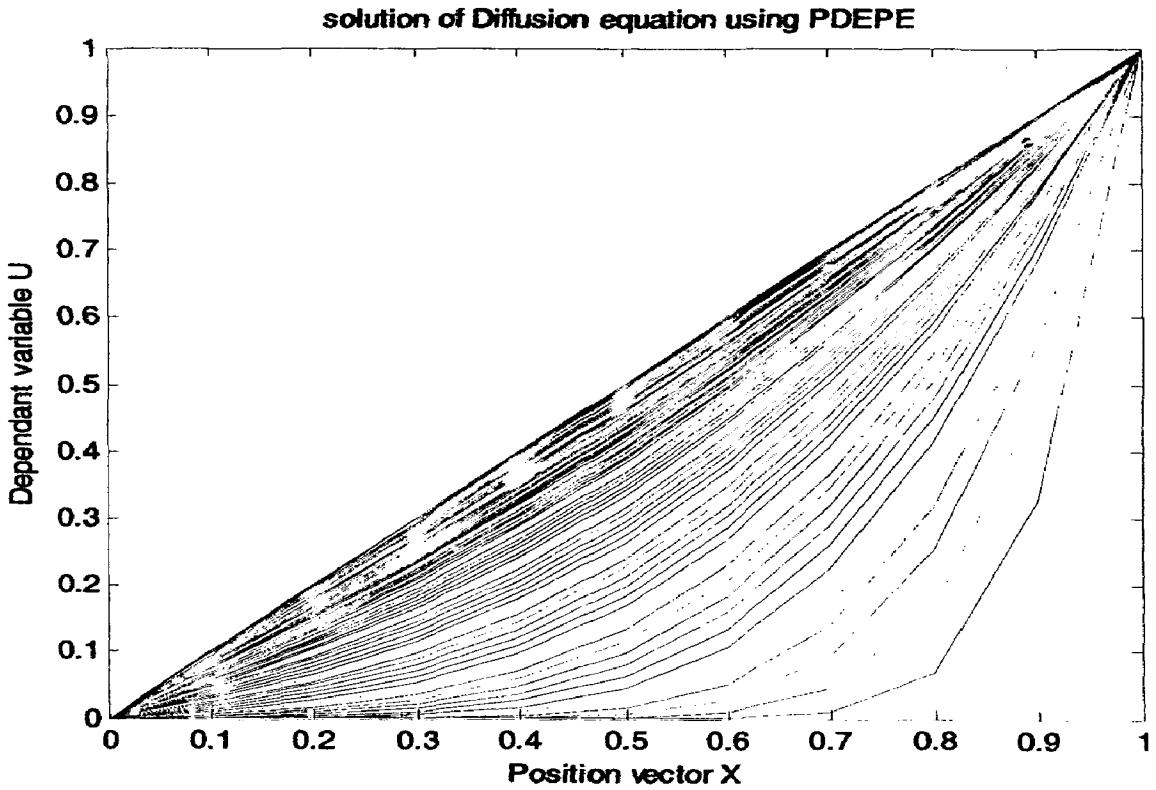


Figure: 5.9 Diffusion equation (PDEPE Tool Box) $\Delta x = 0.1, \Delta t = 0.005$

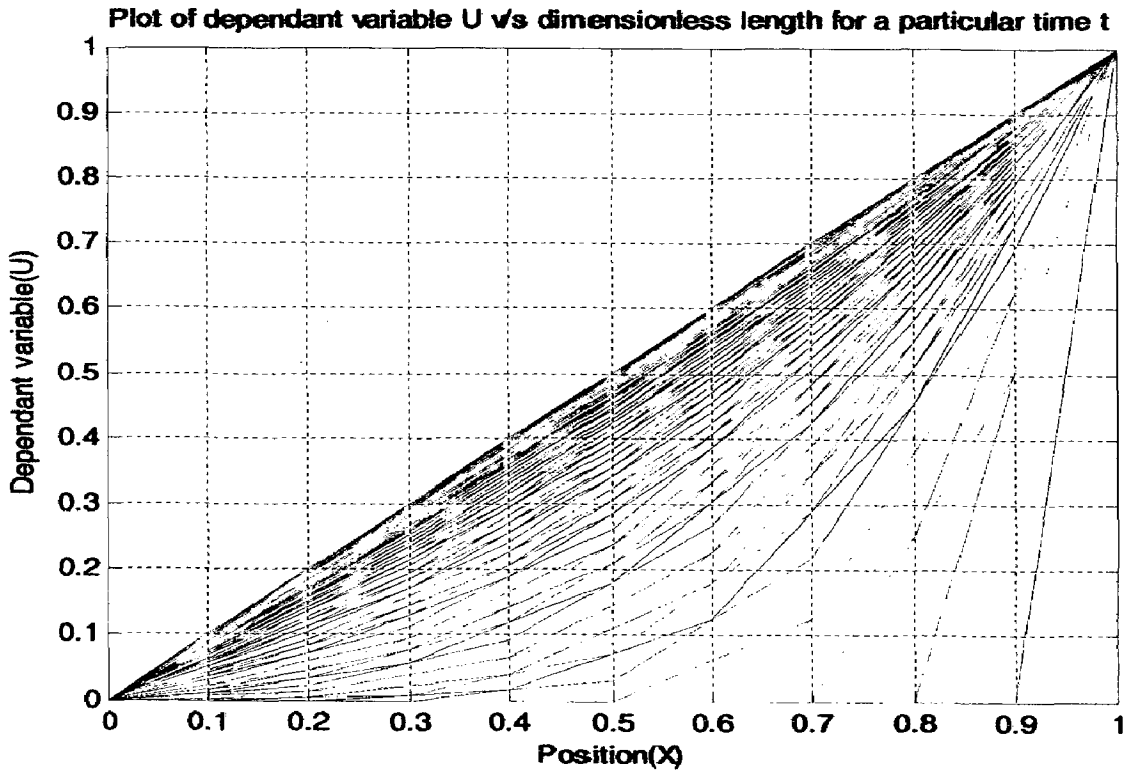


Figure 5.10 Diffusion equation (Forward Differences) $\Delta x = 0.1$ $\Delta t = 0.005$

Appendix A2

Thomas Algorithm for Tri-diagonal Matrix

The given the linear algebraic equations are written in the following form.

$$b_1u_1 + c_1u_2 + 0 + \ldots\ldots\ldots\ldots + 0 = d_1$$
$$a_2u_1 + b_2u_2 + c_2u_3 + 0 + \ldots\ldots\ldots\ldots + 0 = d_2$$

$$0 + \ldots a_iu_{i-1} + b_iu_i + c_iu_{i+1} + \ldots\ldots\ldots\ldots + 0 = d_i$$

$$0 + \ldots\ldots\ldots + 0 + a_{R-2}u_{R-3} + b_{R-2}u_{R-2} + c_{R-2}u_{R-1} = d_{R-2}$$
$$0 + \ldots\ldots\ldots + 0 + a_{R-1}u_{R-2} + b_{R-1}u_{R-1} = d_{R-1}$$

The equations can be represented in general as;

$$a_iu_{i-1} + b_iu_i + c_iu_{i+1} = d_i \qquad \text{for; } 1 \le i \le R$$

with $a_1 = c_R = 0$

The algorithm is as follows;

First Compute;

$$\beta_i = b_i - \frac{a_ic_{i-1}}{\beta_{i-1}} \quad \text{with } \beta_1 = b_1$$

and;

$$\gamma_i = \frac{d_i - a_i\gamma_{i-1}}{\beta_i} \quad \text{with } \gamma_1 = \frac{d_1}{b_1}$$

The values of the dependant variable are then computed by back substituted from;

$$u_R = \gamma_R \text{ and } u_i = \gamma_i - \frac{c_iu_{i+1}}{\beta_i}$$

The algorithm however suffers a drawback that, if the coefficients b_i ***are assuming very small values, then the algorithm gives erratic results. Further the algorithm is not applicable for the conditions when*** $b_i = 0$

Appendix A4

MATLAB PROGRAMS FOR THE SIMULATION OF CONVECTION DIFFUSION EQUATIONS

Appendix A2

Thomas Algorithm for Tri-diagonal Matrix

The given the linear algebraic equations are written in the following form.

$$b_1u_1 + c_1u_2 + 0 + \ldots\ldots\ldots\ldots + 0 = d_1$$
$$a_2u_1 + b_2u_2 + c_2u_3 + 0 + \ldots\ldots\ldots\ldots + 0 = d_2$$

$$0 + \ldots a_iu_{i-1} + b_iu_i + c_iu_{i+1} + \ldots\ldots\ldots\ldots + 0 = d_i$$

$$0 + \ldots\ldots\ldots + 0 + a_{R-2}u_{R-3} + b_{R-2}u_{R-2} + c_{R-2}u_{R-1} = d_{R-2}$$
$$0 + \ldots\ldots\ldots + 0 + a_{R-1}u_{R-2} + b_{R-1}u_{R-1} = d_{R-1}$$

The equations can be represented in general as;

$$a_iu_{i-1} + b_iu_i + c_iu_{i+1} = d_i \qquad \text{for; } 1 \le i \le R$$

with $a_1 = c_R = 0$

The algorithm is as follows;

First Compute;

$$\beta_i = b_i - \frac{a_ic_{i-1}}{\beta_{i-1}} \quad \text{with } \beta_1 = b_1$$

and;

$$\gamma_i = \frac{d_i - a_i\gamma_{i-1}}{\beta_i} \quad \text{with } \gamma_1 = \frac{d_1}{b_1}$$

The values of the dependant variable are then computed by back substituted from;

$$u_R = \gamma_R \text{ and } u_i = \gamma_i - \frac{c_iu_{i+1}}{\beta_i}$$

The algorithm however suffers a drawback that, if the coefficients b_i ***are assuming very small values, then the algorithm gives erratic results. Further the algorithm is not applicable for the conditions when*** $b_i = 0$

Appendix A3

Algorithm for Bi-Tridiagonal Matrix

The two equations are written in the following general form as;

$$a_i^{(1)}u_{i-1} + a_i^{(2)}v_{i-1} + b_i^{(1)}u_i + b_i^{(2)}v_i + c_i^{(1)}u_{i+1} + c_i^{(2)}v_{i+1} = d_i^{(1)}$$

and

$$a_i^{(3)}u_{i-1} + a_i^{(4)}v_{i-1} + b_i^{(3)}u_i + b_i^{(4)}v_i + c_i^{(3)}u_{i+1} + c_i^{(4)}v_{i+1} = d_i^{(2)} \quad \text{for} \quad 1 \le i \le R$$

with $a_1^{(m)} = c_R^{(m)} = 0 \quad \text{for } 1 \le m \le 4$

The algorithm is as follows;

First compute;

$$\beta_i^{(1)} = b_i^{(1)} - a_i^{(1)}\lambda_{i-1}^{(1)} - a_i^{(2)}\lambda_{i-1}^{(3)}$$

$$\beta_i^{(2)} = b_i^{(2)} - a_i^{(1)}\lambda_{i-1}^{(2)} - a_i^{(2)}\lambda_{i-1}^{(4)}$$

$$\beta_i^{(3)} = b_i^{(3)} - a_i^{(3)}\lambda_{i-1}^{(1)} - a_i^{(4)}\lambda_{i-1}^{(3)}$$

$$\beta_i^{(4)} = b_i^{(4)} - a_i^{(3)}\lambda_{i-1}^{(2)} - a_i^{(4)}\lambda_{i-1}^{(3)}$$

with; $\beta_i^{(m)} = b_i^{(m)} \quad \text{for } 1 \le m \le 4$

and;

$$\delta_i^{(1)} = d_i^{(1)} - a_i^{(1)}\gamma_{i-1}^{(1)} - a_i^{(2)}\gamma_{i-1}^{(2)}$$

$$\delta_i^{(2)} = d_i^{(2)} - a_i^{(3)}\gamma_{i-1}^{(1)} - a_i^{(4)}\gamma_{i-1}^{(2)}$$

with;

$$\delta_i^{(1)} = d_1^{(1)} \text{ and } \delta_i^{(2)} = d_1^{(2)}$$

and;

$$\mu_i = \beta_i^{(1)}\beta_i^{(4)} - \beta_i^{(2)}\beta_i^{(3)}$$

The $\beta_i^{(m)}$, $\delta_i^{(m)}$ and μ_i are computed to aid in the computation of the following functions and need not be stored after the computation of;

$$\lambda_i^{(1)} = \left(\beta_i^{(4)}c_i^{(1)} - \beta_i^{(2)}c_i^{(3)}\right)/\mu_i$$

$$\lambda_i^{(2)} = \left(\beta_i^{(4)}c_i^{(2)} - \beta_i^{(2)}c_i^{(4)}\right)/\mu_i$$

$$\lambda_i^{(3)} = \left(\beta_i^{(1)} c_i^{(3)} - \beta_i^{(3)} c_i^{(1)}\right) / \mu_i$$

$$\lambda_i^{(4)} = \left(\beta_i^{(1)} c_i^{(4)} - \beta_i^{(3)} c_i^{(2)}\right) / \mu_i$$

and;

$$\gamma_i^{(1)} = \left(\beta_i^{(4)} \delta_i^{(1)} - \beta_i^{(2)} \delta_i^{(2)}\right) / \mu_i$$

$$\gamma_i^{(2)} = \left(\beta_i^{(1)} \delta_i^{(2)} - \beta_i^{(3)} \delta_i^{(1)}\right) / \mu_i$$

The values of $\lambda_i^{(m)}$ and $\gamma_i^{(m)}$ must be stored, as they are used in the back solution. This is;

$$u_R = \gamma_R^{(1)}$$

$$v_R = \gamma_R^{(2)}$$

and;

$$u_i = \gamma_i^{(1)} - \lambda_i^{(1)} u_{i+1} - \lambda_i^{(2)} v_{i+1}$$

$$v_i = \gamma_i^{(2)} - \lambda_i^{(3)} u_{i+1} - \lambda_i^{(4)} v_{i+1}$$

$$\text{for } (R-1) \geq i \geq 1$$

Appendix A4

MATLAB PROGRAMS FOR THE SIMULATION OF CONVECTION DIFFUSION EQUATIONS