# The Numerical Solution of Parabolic Integro-differential Equations 

Lanzhen Xue BSc.

Dublin City University<br>Dr. John Carroll (Supervisor)<br>School of Mathematical Sciences

MSc. Thesis by Research
Submitted in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematical Sciences at Dublin City University, May 1993.

## DECLARATION

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of M.Sc. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Date:

Date


#### Abstract

This thesis is concerned with aspects of the numerical solution of parabolic integrodifferential equations and it consists of two parts. The first part is concerned with project preliminaries. The second part is concerned with the central theme of this thesis - the numerical solution of parabolic integro-differential equations.

The first part of the thesis (chapters 2 and 3 ) deals with aspects of background knowledge in Numerical Analysis, with emphasis on the numerical solutions of ordinary differential equations (ODEs for short) and efficient numerical solution techniques for systems of one-dimensional linear parabolic partial differential equations (PDEs for short). In particular, we use both comprehensive and sophisticated mathematical software packages and libraries to get the most reliable, robust and efficient numerical routines for solving ODEs and PDEs.

In the second part of the thesis, some numerical methods for the solution of integrodifferential equations of parabolic type are discussed, with emphasis on the methods which use time discretization schemes based on the Backward Euler and the CrankNicolson schemes. The integral term is approximated in each case by a quadrature rule with relatively high-order truncation error, so that a relatively large time step can be used for the quadrature so as to limit the storage requirements, without sacrificing the overall order of convergence.

We describe certain aspects of the numerical algorithms proposed in Le Roux and Thomee [15] and Zhang [20] and, in particular, we examine ways in which the algorithms can be implemented efficiently. The solution algorithm proposed in Zhang [20], referred to in what follows as Modified Method I, is implemented and applied to solve a number of test problems. Based closely on the ideas of Le Roux and Thomée [15], we construct a second package, referred to in what follows as Modified Method II, which implements a collection of 6 quadrature schemes (the Rectangular Rule, Trapezoidal Rules A, B and I, and Simpson's Rules A and B).

We then test the effectiveness of both packages in terms of improvements in accuracy, storage requirements and execution times by solving some integro-differential equations of parabolic type and analyzing the results.

The improved methods reduce greatly both the memory and computational expense involved in solving integro-differential equations of parabolic type. Modified Method I is shown to be very robust and efficient when solving the standard test problems. Modified Method II is more efficient than Modified Method I when solving the same type of problems.


## Acknowledgements

I wish to thank all of those who have contributed in various ways towards the completion of this thesis.

I would like to express my appreciation to my research advisor, Dr. John Carroll, for all the guidance and support that he gave to me during my graduate studies in Dublin City University. Also I am grateful to my external supervisor, Professor Vidar Thomèe, for many suggestion on my work.

I would also like to thank my own family, for their support and encouragement during the period of my research.

Lastly, thanks to my special friends - my former tutor, Dr. F. W. Liu, for his valuable assistance and many suggestions, Dr. N. Y. Zhang, G. Wen. and G. F. Sun, X. Lin for numerous stimulating discussions on my work, Paul Farrell and Kieran Arthur for carefully reading the manuscript and making a number of English corrections.

## Contents

Chapter 1 Introduction ..... 1
1.1 Opening Comments ..... 1
1.2 Thesis Outline ..... 2
Chapter 2 Ordinary Differential Equations ..... 4
2.1 Introduction ..... 4
2.2 Basic Approximation Theory of ODEs ..... 4
2.3 Sources of Software ..... 8
2.4 Numerical Experiments ..... 10
2.5 Summary ..... 18
Chapter 3 Parabolic Partial Differential Equations ..... 19
3.1 Introduction ..... 19
3.2 PDE Software ..... 20
3.3 Numerical Experiments ..... 24
3.4 Summary ..... 32
Chapter 4 Parabolic Integro-Differential Equations I ..... 33
4.1 Introduction ..... 33
4.2 Theoretical Considerations ..... 33
4.3 Time Discretization ..... 36
4.4 Quadrature Schemes ..... 37
Chapter 5 Parabolic Integro-Differential Equations II ..... 49
5.1 Introduction ..... 49
5.2 Computer Implementation ..... 49
5.3 Test Problems ..... 58
5.4 Efficiency Considerations ..... 59
5.5 Numerical Experiments ..... 60
5.6 Analysis of Results ..... 66
Chapter 6 Conclusions ..... 71
Bibliography ..... 74

## Chapter 1 <br> Introduction

### 1.1 Opening Comments

This thesis is concerned with aspects of the numerical solution of parabolic integrodifferential equations. The problem may be defined as follows. Let $\Omega$ be a bounded domain in $R^{d}$ with sufficiently smooth boundary $\partial \Omega$, and let $0<t^{0}<\infty$. A linear partial integro-differential equation of parabolic type is

$$
\begin{equation*}
\frac{\partial}{\partial t} u(x, t)+A u(x, t)=\int_{0}^{t} B(t, s) u(x, s) d s+f(x, t) \quad(x, t) \in \bar{\Omega} \quad\left(0, t^{0}\right], \tag{1.1}
\end{equation*}
$$

subject to the homogoeneous Dirichlet boundary condition

$$
\begin{equation*}
u(x, t)=0 \quad(x, t) \in \partial \Omega \quad\left(0, t^{0}\right] \tag{1.2}
\end{equation*}
$$

and the initial condition

$$
\begin{equation*}
u(x, 0)=u_{0}(x) \quad x \in \bar{\Omega} . \tag{1.3}
\end{equation*}
$$

Here A is an elliptic operator of the form

$$
A=-\sum_{i, j=1}^{d} \frac{\partial}{\partial x_{i}}\left(a_{i j}(x) \frac{\partial}{\partial x_{J}}\right)+a_{0}(x) I,
$$

Where the matrix $\left(\mathrm{a}^{\mathrm{ij}}(\mathrm{x})\right)_{\mathrm{i}, \mathrm{j}=1}$ is symmetric and uniformly positive definite, and $\mathrm{a}_{0}(\mathrm{x})$ is nonnegative on $\bar{\Omega}$. Further

$$
B(t, s)=-\sum_{i, j=1}^{d} \frac{\partial}{\partial x_{i}}\left(b_{i j}(x ; t, s) \frac{\partial}{\partial x_{j}}+\sum_{i j=1}^{d} b_{j}(x ; t, s) \frac{\partial}{\partial x_{j}}+b_{0}(x) I\right.
$$

is a partial differential operator of at most second order. It is assumed that the coefficients $\mathrm{a}_{\mathrm{ij}}(\mathrm{x}), \mathrm{a}_{0}(\mathrm{x}), \mathrm{b}_{\mathrm{ij}}(\mathrm{x} ; \mathrm{t}, \mathrm{s}), \mathrm{b}_{0}(\mathrm{x} ; \mathrm{t}, \mathrm{s})$ and $f=f(\mathrm{x}, \mathrm{t})$ are real-valued and sufficiently smooth functions.

Such problems arise in many applications, including heat conduction in materials with memory, the compression of poro-viscoelastic media, reactor dynamics, the compartment model of a double-porosity system and epidemic phenomena in biology.

### 1.2 Thesis Outline

We now give a brief outline of the main chapters of the thesis and state the aims and objectives of each chapter.

Chapters 2 and 3 deal with aspects of background knowledge in Numerical Analysis, with emphasis on the numerical solutions of ordinary differential equations (ODEs for short) and the efficient numerical solution techniques for systems of one-dimensional linear parabolic partial differential equations (PDEs for short).

Chapter 2 deals with the numerical solution of $O D E s$. It also deals with the numerical procedures used in the approximate integration of systems of ordinary differential equations. We close chapter 2 with some numerical experiments which use sophisticated mathematical software libraries to solve initial value problems (IVPs) for ODEs.

Chapter 3 is devoted to a review of efficient numerical solution techniques for systems of one-dimensional linear parabolic partial differential equations. We also consider the application of the numerical software to approximate the solutions of a number of parabolic equations frequently cited in the literature. We used three effective solvers for the parabolic PDEs, they are

■ Composite integration scheme (日BDF2) Carroll [13]

- DO3PGF [2] (in the NAG library)

PDECOL [9].

We also present some numerical results to assess the effectiveness of the above solvers.

Chapter 4 and chapter 5 are the central themes of this thesis, in which we are concerned with aspects of the numerical solution of parabolic integro-differential equations which were described briefly in section 1.1.

In chapter 4 some numerical methods for the solution of integro-differential equations of parabolic type are discussed. Emphasis is placed on two different time discretizations of an integro-differential equation of parabolic type. They are based on the Backward Euler and the Crank-Nicolson schemes. The methods reduce to the backward Euler and the Crank-Nicolson schemes if the integral term is absent. The integral term is approximated in each case by a quadrature rule with relatively high-order truncation error so that a relatively large time step can be used for the quadrature in order to reduce the memory and computational requirements of the method.

Sloan and Thomée [17], Pani, Thomee and Wahlbin [16], Roux and Thomee [15] and Zhang [20] proposed some quadrature rules which are more sparse than the standard methods, that is rules with high orders of accuracy, so that only a part of the time steps need to be used as quadrature points. They nevertheless retain the order of accuracy of the standard schemes.

In this chapter we describe certain aspects of the numerical algorithms proposed in Le Roux and Thomee [15] and Zhang [20] and, in particular, we examine ways in which the algorithms can be implemented efficiently. The solution algorithm proposed in Zhang [20], referred to in what follows as Modified Method I, is implemented and applied to solve a number of test problems. Based closely on the ideas of Le Roux and Thomée [15], we construct a second package, referred to in what follows as Modified Method II, which implements a collection of 6 quadrature schemes (the Rectangular Rule, Trapezoidal Rules A, B and I, and Simpson's Rules A and B).

We then test the effectiveness of both packages in terms of improvements in accuracy, storage requirements and execution times by solving some integro-differential equations of parabolic type and analyzing the results in the next chapter.

In chapter 5 we give the implementation details for both the modified method, which we call Modified Method II and modified quadrature schemes proposed by Zhang [20], which we call Modified Method I. We present some numerical experiments together with results analysis to measure the effectiveness of improvements in accuracy, storage requirements and execution times for both solvers. Numerical experiments employing finite differences methods demonstrate that Modified Method II not only keeps the advantages of Modified Method I but also displays more efficiency in terms of storage requirements and the CPU time.

We close, in chapter 6, with some conclusions and suggestions for future work.

## Chapter 2 <br> Ordinary Differential Equations

### 2.1 Introduction

This chapter deals with the numerical solutions of ordinary differential equations (ODEs) and also with numerical procedures for the approximate integration of systems of ODEs. We conclude by using sophisticated mathematical software libraries to solve IVPs for ODEs.

Numerical analysis of ODEs is a vast subject, and the software for solving ODEs has been highly developed. Numerical methods for ODEs have been studied in many books such as Lambert [6], Gear [4], Hairer and Wanner [5], Shampine and Gordon [7] and many references quoted there.

We do not attempt to introduce numerical methods and software for ODEs in detail here, but focus on using sophisticated mathematical software libraries such as the NAG library [2]. We discuss how to choose efficient numerical routines for ODE problems. A brief introduction to the numerical solution of ODEs and characteristics of the mathematical software libraries are presented.

We show some numerical results to illustrate and compare a variety of performances of of the state-of-the-art routines which are used in a number of test problems.

### 2.2 Basic Approximation Theory of ODEs

The most important mathematical model for physical phenomena is the differential equation. The motion of objects and fluid, heat flow, bending and cracking of materials, vibration, chemical reactions, and nuclear reactions are all modeled by differential equations. If a differential equation has one independent variable then it is an ordinary differential equation. Examples of such equations are

$$
\begin{align*}
& \frac{d y}{d x}=x+y,  \tag{2.1}\\
& y^{\prime}=x^{2}+y^{2}, \\
& y^{\prime \prime}+\cos \left(x y^{\prime}-3 y=\sin (2 x),\right.
\end{align*}
$$

The notation $y^{\prime}=d y / d x$ is used.

The ordinary differential equation problem needs more than a differential equation. To solve $y^{\prime \prime \prime}=0$ is not a well formulated problem since $y(x)=x^{2}+2, y(x)=3 x^{2}-2 x+4$, and $y(x)=1-5 x$ all satisfy this equation. Generally, an equation of order $n$ (that is, the highest derivative appearing is the nth) requires $n$ additional conditions in order to have a unique solution. In principle, these conditions can be of any type, for example:

$$
\begin{align*}
& y(3.6)=6.3 \\
& y^{\prime}(2)=2.2  \tag{2.2}\\
& y(2)+3 y^{\prime}(2)=6, \quad \text { etc. }
\end{align*}
$$

If all the conditions occur at one point, then we have an initial value problem (IVP) for example:

$$
\begin{array}{ll}
y_{1}^{\prime}=y_{2} y_{3} & y_{1}(0)=0 \\
y_{2}^{\prime}=-y_{1} y_{3} & y_{2}(0)=1  \tag{2.3}\\
y_{3}^{\prime}=-.51 y_{1} y_{2} & y_{3}(0)=1
\end{array}
$$

The problem in (2.2) is called an IVP for ordinary differential equations (ODEs) and it is one of the main topics in this chapter.

The general form of the IVP for a first-order ODE is

$$
\begin{equation*}
y^{\prime}(t)=f(t, y(t)) \quad a \leq t \leq b, \quad y(a)=y_{0} \tag{2.4}
\end{equation*}
$$

while for a system of first-order ODEs it is

$$
\begin{array}{cc}
y_{1}^{\prime}(t)=f_{1}\left(t, y_{1}(t), y_{2}(t), \ldots y_{n}(t)\right) & y_{1}(a)=y_{10} \\
y_{2}^{\prime}(t)=f_{2}\left(t, y_{1}(t), y_{2}(t), \ldots y_{n}(t)\right. & y_{2}(a)=y_{20}  \tag{2.5}\\
\cdot & \\
\cdot & \\
y_{n}^{\prime}(t)=f_{n}\left(t, y_{1}(t), y_{2}(t), \ldots y_{n}(t)\right) & y_{n}(a)=y_{n 0}
\end{array}
$$

and for an nth-order ODE it is

$$
\begin{gather*}
y^{(n)}(t)=f\left(t, y(t), y^{\prime}(t), \ldots, y^{(n-1)}(t)\right) \quad y^{(i)}=y_{i 0}, \quad i=0,1, \ldots, n-1 \\
\left(\text { Note: } \quad y^{(n)}=\frac{d^{n} y}{d t^{n}}\right) \tag{2.6}
\end{gather*}
$$

A higher order ODE can almost always be reduced to the first-order form by introducing new variables. For example, suppose we have the third-order equation:

$$
\begin{equation*}
z^{\prime \prime \prime}+z z^{\prime \prime}+k\left(1-z^{\prime 2}\right)=0 \tag{2.7}
\end{equation*}
$$

We write $\mathrm{y}_{1}=\mathrm{z}, \mathrm{y}_{2}=\mathrm{z}^{\prime}, \mathrm{y}_{3}=\mathrm{z}^{\prime \prime}$, and the equation (2.7) may then be written as the system

$$
\left\{\begin{array}{l}
y_{1}^{\prime}=y_{2}  \tag{2.8}\\
y_{2}^{\prime}=y_{3} \\
y_{3}^{\prime}=-y_{1} y_{3}-k\left(1-y_{2}^{2}\right)
\end{array}\right.
$$

thus, the original third-order equation (2.7) can easily be reformulated as a system of 3 first-order equations (2.8).

It is easy to see that we can use the technique described above to reformulate a differential equation of order $n$ as a system of $n$ first-order differential equations. The reformulated problem greatly simplifies the development, use and analysis of methods for IVPs.

Finding accurate and efficient solution procedures for solving differential equations has long been a problem of importance. However, in many practical situations, an analytical solution is either impossible to find or extremely difficult to evaluate. In recent years, numerical solution procedures for approximating solutions have become increasingly popular, thanks in large part to the power of modern, high-speed computers.

The numerical methods for ODEs have been studied in many books such as Lambert [6], Gear [4], Hairer and Wanner [5], Shampine and Gordon [7] and many references quoted there. The software for solving ODEs using numerical methods has been highly developed. We will outline basic concepts of the numerical analysis and discuss some commonly used routines.

Some authors, such as Rice [1], classify the numerical methods for ODEs in the following:

- The first class consists of simple methods that are reasonably easy to understand and analyze.

■ The second class includes the first class. It obtains greater accuracy and efficiency. Also, it is the starting point for the third class.

- The third class combines integration formulas of the second class with error control, starting procedures, printing control, etc..

In this section we briefly present here several of the more common, basic and important numerical methods for IVPs only. Some well know third class numerical methods for IVPs will be mentioned in the next section.

Before considering the methods, we write the equation for a single first-order IVP as :

$$
\begin{align*}
& y^{\prime}(t)=f(t, y) \quad t>a  \tag{2.9}\\
& y(a)=\eta
\end{align*}
$$

where $f$ is some known function and $\eta$ defines an initial condition. Each method we describe for the numerical solution of (2.9) attempts to find an approximation $y_{i}$ to the true solution $y\left(t_{i}\right)$ at a set of discrete points $\left\{\mathrm{t}_{\mathrm{i}} \mid \mathrm{i}=0,1, \ldots\right\}$ where $\mathrm{a}=\mathrm{t}_{0}<\mathrm{t}_{1}<\ldots$... This is done in a step-by-step fashion; that is starting with the initial value $\mathrm{y}_{0}=\eta$ we compute $\mathrm{y}_{1}$, then $\mathrm{y}_{2}$, and so on until some appropriate criterion is satisfied (say, we have obtained an approximation at $t=T$ ). For the moment we shall assume that the distance between successive discretization points, called mesh points is a constant, $h$, called the integration step size that is $\mathrm{t}_{\mathrm{i}+1}-\mathrm{t}_{\mathrm{i}}=\mathrm{h}$ for $\mathrm{i}=0,1, \ldots$.

The Taylor series expansion for $y\left(t_{i}+h\right)$ about $t=t_{i}$ may be written

$$
\begin{gather*}
y\left(t_{i}+h\right)=y\left(t_{i}\right)+h y^{\prime}\left(t_{i}\right)+\frac{h^{2}}{2!} y^{\prime \prime}\left(t_{i}\right)+\ldots+\frac{h^{\gamma}}{\gamma!} y^{(\gamma)}\left(t_{i}\right)+R_{\gamma}  \tag{2.10}\\
\text { where } \quad R_{\gamma}=\frac{h^{\gamma+1}}{(\gamma+1)!} y^{(\gamma+1)}(\xi) \quad \xi \in\left(t_{i}, t_{i+1}\right),
\end{gather*}
$$

If we have available approximations to $y$ and its first $\gamma$ derivatives (assuming they exist) at $t=t_{i}$, the Taylor series (2.10) may be truncated at the remainder term $R_{\gamma}$. If we denote these approximations by $y_{i}, y_{i}^{\prime}, y_{i}{ }^{\prime \prime}$, etc., we have the scheme

$$
\begin{equation*}
y_{i+1}=y_{i}+h y_{i}^{\prime}+\frac{h^{2}}{2!} y_{i}^{\prime \prime}+\ldots+\frac{h^{\gamma}}{\gamma!} y_{i}^{(\gamma)} \tag{2.11}
\end{equation*}
$$

The local truncation error (LTE) of any numerical scheme is defined to be the difference between $y\left(t_{i+1}\right)$ and $y_{i+1}$ under the assumption that all values used in the calculation of $y_{i+1}$
are exact. (This definition assumes the absence of rounding error). Further, the scheme is said to be of order $\gamma$ if the LTE is $\mathrm{O}\left(\mathrm{h}^{\gamma+1}\right)$. For the Taylor series scheme just outlined this means that if $y_{i}{ }^{\prime}=y^{\prime}\left(t_{i}\right)$, etc., the LTE is just $R_{r}$, and the method is of order $\gamma$.

The simplest form of a Taylor series method is derived by setting $\gamma=1$. In (2.11) we obtain the one step Euler method:

$$
\begin{equation*}
y_{i+1}=y_{i}+h y_{i}^{\prime}=y_{i}+h f_{i} \tag{2.12}
\end{equation*}
$$

where $f_{\mathrm{i}}=f\left(\mathrm{t}_{\mathrm{i}}, \mathrm{y}_{\mathrm{i}}\right)$. Assuming that an analytic form for $f$ is available, we may easily compute $\mathrm{y}_{1}, \mathrm{y}_{2}, \ldots$ in turn. Obviously, the equation (2.12) is a first-order method.

We can use the $\theta$ scheme, $0 \leq \theta \leq 1$, to provide a general formula for one-step methods as follows :

$$
\begin{equation*}
y_{i+1}=y_{i}+h\left\{(1-\theta) f_{i}+\theta f_{i+1}\right\} \quad i \geq 0 \tag{2.13}
\end{equation*}
$$

with $y_{0}=y(a)$, and the following are three well-known examples:
(a) the explicit Euler method $(\theta=0)$ given by formula (2.12).
(b) the implicit Trapezoidal rule $(\theta=1 / 2)$ given by

$$
\begin{equation*}
y_{i+1}=y_{i}+\frac{h}{2}\left(f_{i}+f_{i+1}\right) \tag{2.14}
\end{equation*}
$$

(c) the implicit Backward Euler scheme $(\theta=1)$ given by

$$
\begin{equation*}
y_{i+1}=y_{i}+h f_{i+1} \tag{2.15}
\end{equation*}
$$

### 2.3 Sources of Software

Software for solving ODEs has developed rapidly owing to the nature of the inherently more complicated problems.

We restrict our discussions to the following two aspects:

> ■ We briefly describe the characteristics of the comprehensive numerical software, in particular, the NAG Library.

> ■ We briefly introduce several popular numerical methods for IVPs in the Library which are to be used in the next section of this chapter.

As we mentioned in above, there are a lot of programs for solving ODEs. Generally, they are included in several software sources, such as:

Individual programs (perhaps with a few subprograms) The programs carry out one specific computation.

A software package It is a set of programs for a particular problem area and is usually narrowly focused.

A software library It is a much larger set of programs to support general numerical computation and has well-organized documentation with on-line information.

Hence, we selected the software library as the main sources of software for the numerical experiments for ODEs in this thesis for the following reasons:

■ It is especially helpful to students of numerical analysis.

- Provides a broad range of reliable, robust and efficient numerical routines.

■ For an intelligent user, routines of a library should allow quite difficult problems to be solved with a minimum of programming effort.

It was recognized early as 1951 that software libraries are important in computing [1]. The motivation for using a library program is simple: one avoids writing and debugging a new program. A numerical software library is a much larger set of programs to support general numerical computation. About 500 programs are required for good support and the library must have well-organized documentation and on-line information. The programs are convenient and reliable. Presently, there are three general comprehensive libraries for numerical computation that are widely used in scientific and engineering fields.They are:

- IMSL—IMSL.Inc.
- NAG - Numerical Algorithms Group. Oxford University.
- SL/MATH-IBM Corporation.

As the ideal software library selected, the NAG Library includes many routines for the numerical solution of ODEs. The majority of the routines available can be classified as:

Merson's method routines The routines make use of the Runge-Kutta-Merson error control technique and are named D02BxF (where x is $\mathrm{A}, \mathrm{N}, \mathrm{D}, \mathrm{G}$ and H ) [2]. They are the best routines for simple problems with low accuracy requirements, that is problems on a short range of integration, with derivative functions $f_{i}$ which are inexpensive to calculate and where only a few correct figures are required.

Adams' method routines The routines make use of Adams-Moulton and AdamsBashforth formula and are named D02CxF (where x is $\mathrm{A}, \mathrm{B}, \mathrm{G}$ and H ) in the Library. For larger problems, over long ranges or with high accuracy requirements the variableorder, variable-step routines should usually be preferred.

BDF (or Gear) routines The routines make use of Gear's Backward Differentiation Formula (BDF) and are named D02ExF (where x is $\mathrm{A}, \mathrm{B}, \mathrm{G}$, and H ). In describing the purpose of D02Exf we briefly introduce a term, namely, stiffness. A stable differential equation is called stiff when it has a decaying exponential particular solution with a time constant which is very small relative to the interval over which it is being solved. When stiffness is present in an ODE system, standard methods may fail to compute an accurate solution, or may require excessive amounts of computation because they need small steplengths to avoid numerical instability. Gear's variable-order variable-step routines should be used as special purpose algorithms for solving stiff systems.

In order to test the performance of various solvers described above and to show how suitable selections of various routines enable us to obtain an accurate solution for ODEs, several numerical experiments for IVPs are given in the next section.

### 2.4 Numerical Experiments

In this section three routines for three algorithms (Runge-Kutta-Merson's method, Adams' method and Gear's method ) contained in the NAG Library will be applied to the solution of example IVPs and the performance of the three routines will be compared.

## ODE Example 1

Bessel's equation of order $1 / 2$ with the origin shifted one unit to left, written as a first-
order system:

$$
\begin{array}{lll}
y_{1}^{\prime}=y_{2} & x \in[0,10] & y_{1}(0)=.6713967071418030 \\
y_{2}^{\prime}=\left(\frac{1}{4(x+1)^{2}}-1\right) y_{1}-\frac{y_{2}}{x+1} & y_{2}(0)=.09540051444747446 \tag{2.16}
\end{array}
$$

An accurate numerical solution is used as a reference solution.

Source: Taken from [4A]

## ODE Example 2

Euler's equations of motion for a rigid body without external forces:

$$
\begin{array}{ll}
y_{1}^{\prime}=y_{2} y_{3} & y_{1}(0)=0 \\
y_{2}^{\prime}=-y_{1} y_{3} & y_{2}(0)=1  \tag{2.17}\\
y_{3}^{\prime}=.51 y_{1} y_{2} & y_{3}(0)=1
\end{array} \quad x \in[0,10]
$$

An accurate numerical solution is used as a reference solution.

Source: Taken from [4A]

## ODE Example 3

A radioactive decay chain problem:

$$
\begin{array}{lll}
y_{1}^{\prime}=-y_{1} & y_{1}(0)=1 & x \in[0,10] \\
y_{i}^{\prime}=(i-1) y_{i-1}-i y_{i} & y_{i}(0)=0 & i=2,3,4, \ldots, 9 \\
y_{10}^{\prime}=9 y_{9} & y_{10}(0)=0 & \tag{2.18}
\end{array}
$$

An accurate numerical solution is used as a reference solution.

Source: Taken from [4A]

## ODE Example 4

The Robertson chemical kinetics problem:

$$
\begin{array}{lll}
y_{1}^{\prime}=-.04 y_{1}+.01 y_{2} y_{3} & y_{1}(0)=1 \\
y_{2}^{\prime}=400 y_{1}-100 y_{2} y_{3}-3000 y_{2}^{2} & y_{2}(0)=0  \tag{2.19}\\
y_{3}^{\prime}=30 y_{2}^{2} \quad x \in[0,10] & y_{3}(0)=0
\end{array}
$$

An accurate numerical solution is used as a reference solution.

Source: Taken from [4A]

## ODE Example 5

$$
\begin{align*}
y^{\prime \prime}=-.5-y^{\prime}-\left(\frac{\cos (x)+y^{\prime}}{y}\right), & x \in[-.43,0],  \tag{2.20}\\
& y(0)=y^{\prime}(0)=1
\end{align*}
$$

An accurate numerical solution is used as a reference solution.

ODE Example 6

$$
\begin{array}{r}
y^{\prime}=\frac{2 x^{3}-2 x}{y}, \quad x \in[0,2],  \tag{2.21}\\
y(0)=1
\end{array}
$$

An accurate numerical solution is used as a reference solution.

The following tables show the numerical results for the three methods, Runge-KuttaMerson, Adams and Gear methods, applied to the solution of the six example ODE problems above. For each problem the performances of the three methods are compared in terms of accuracy and efficiency. The accuracy of each algorithm is illustrated by the global error at the endpoint of each example. The efficiency of each method is expressed using, NFE, the number of function evaluations used and the CPU-seconds taken throughout the integration. The notation used and statistics collected include:

TOL a scalar value for error tolerance used in time integration. The value TOL is supplied by the user to the NAG D02 routines, it specifies a local error request and is not a global error bound. The idea of tolerance proportionality is used here, i.e., for some problems, the global error is roughly proportional to TOL. However, the relation between TOL and the accuracy is not guaranteed. The user is recommended to call D02 with more than one value for TOL and to compare the
results obtained to estimate their accuracy.

Erx (where x is the number of equations in the system of ODEs) the global error at the endpoint for each problem.

NFE the number of function evaluations.

CPU the execution time in seconds on a DEC VAX/6230.

Examples 1 and 2 are non-stiff systems. Accurate reference solutions obtained by solving the problem with the stringent tolerance of $\mathrm{TOL}=10^{-12}$ are used as reference solutions for comparison with the computed solution which have different values of $\mathrm{TOL}=10^{-4}$, $10^{-5}, \ldots, 10^{-9}$. Tables 2.1 and 2.2 show several integration statistics for solutions obtained for Examples 1 and 2 using the three methods respectively and it is easy to see that:

The Runge-Kutta method exhibits a higher degree of tolerance proportionality than the others. If parameter TOL $>10^{-6}$ for ODE problem 1 and $\mathrm{TOL}>10^{-7}$ for ODE problem 2 the solution cannot be computed using this method.

Gear's method is more expensive than the others because it makes more function calls.

The Adams routine makes less function calls but the overall time is more than that for the Runge-Kutta method.

Comparing the accuracy of the three methods in terms of global error measurements, the Runge-Kutta method is the best one.

Table 2.1

## ODE Example 1

| METHODS | Adams |  |  | Gear |  |  | Runge-Kutta |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TOL | Er1 Er2 | NFE | CPU | Er1 Er2 | NFE | CPU | Er1 Er2 | NFE | CPU |
| .1D-3 | .4D-4 .1D-3 | 29 | 0.2 | .3D-4 .1D-4 | 50 | 0.4 |  |  |  |
| .1D-4 | .1D-4 .5D-5 | 36 | 0.3 | .6D-5 .1D-5 | 68 | 0.5 |  |  |  |
| .1D-5 | .2D-5 .1D-5 | 54 | 0.3 | .3D-6 .5D-6 | 80 | 0.6 | .5D-6 .1D-5 | 37 | 0.1 |
| .1D-6 | .3D-6 .1D-6 | 62 | 0.4 | .6D-8 .3D-7 | 98 | 0.7 | .9D-7 .2D-6 | 52 | 0.2 |
| .1D-7 | .4D-7 .1D-7 | 72 | 0.4 | .3D-8 .2D-9 | 105 | 0.8 | .1D-7 .3D-7 | 71 | 0.2 |
| .1D-8 | .6D-8 .3D-8 | 80 | 0.5 | .1D-8 .4D-9 | 130 | 1 | .2D-8 .6D-8 | 101 | 0.3 |

Table 2.2

ODE Example 2

| METH | Adams |  |  | Gear |  |  | Runge-Kutta |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TOL | Er1 Er2 Er3 | NFE | CPU | Er1 Er2 Er3 | NFE | CPU | Er1 $\begin{array}{lll}\text { Er2 } & \text { Er3 }\end{array}$ | NFE | CPU |
| .1D-3 | . $5 \mathrm{D}-3.3 \mathrm{D}-3.9 \mathrm{D}-4$ | 39 | . 4 | .3D-3 .3D-3 .3D-3 | 60 | 0.6 |  |  |  |
| .1D-4 | .5D-4 .6D-4 .6D-4 | 47 | . 4 | .8D-6 .1D-4 .1D-4 | 68 | 0.7 |  |  |  |
| .1D-5 | .2D-5 .2D-5 .9D-6 | 60 | . 5 | .1D-5 .4D-5 .3D-5 | 84 | 0.8 |  |  |  |
| .1D-6 | .9D-7 .5D-7 .2D-6 | 70 | . 6 | .2D-6 .1D-6 .9D-7 | 99 | 1 | .2D-7 .3D-7 .1D-7 | 83 | . 4 |
| .1D-7 | .2D-7 .1D-7 .3D-7 | 74 | . 7 | .4D-7 .2D-7 .2D-7 | 123 | 1.2 | .2D-8 .3D-8 .1D-8 | 138 | . 5 |
| .1D-8 | .7D-9 .4D-8 .4D-8 | 86 | . 8 | .9D-8 .1D-8 .2D-8 | 148 | 1.4 | .2D-9 .3D-9 .1D-9 | 233 | . 6 |

Example 3 is a mildly stiff system. The comparison of the performances in Table 2.3 is similar to those for Examples 1 and 2.

Table 2.2

ODE Example 3

| Runge-Kutta Method |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TOL | NFE | CPU | Er1 | Er2 | Er3 | Er4 | Er5 | Er6 | Er7 | Era | Er9 | Er 10 |
| .1D-7 | 29 | 0.3 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .5D-9 | .4D-8 |
| .1D-8 | 38 | 0.4 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .1D-9 | .9D-9 |
| Adams' Method |  |  |  |  |  |  |  |  |  |  |  |  |
| TOL | NFE | CPU | Er1 | Er 2 | Er3 | Er4 | Er5 | Er6 | Er7 | Er8 | Er9 | Er10 |
| .1D-3 | 38 | 0.5 | .7D-7 | .7D-7 | .7D-7 | .7D-7 | .7D-7 | .7D-7 | .7D-7 | .7D-7 | .8D-5 | .7D-5 |
| .1D-4 | 28 | 0.4 | .3D-6 | .3D-6 | .3D-6 | .3D-6 | .3D-6 | .3D-6 | 3D-6 | .4D-6 | SD-6 | .2D-5 |
| .1D-5 | 33 | 0.5 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-6 | .1D-5 |
| .10-6 | 36 | 0.6 | 1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .10-7 | .1D-7 | .1D-6 |
| .1D-7 | 36 | 0.8 | .3D-10 | .3D-10 | 3D-10 | . 3 D-10 | .35-10 | .3D-10 | .3D-10 | 3D-10 | .3D-10 | .3D-9 |
| .1D-8 | 55 | 1 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .9D-10 | .8D-9 |
| Gear's Method |  |  |  |  |  |  |  |  |  |  |  |  |
| TOL | NEF | CPU | Er1 | Er2 | Er3 | Ers | Er5 | Er6 | Er7 | Era | Er9 | Er10 |
| .1D-3 | 69 | 0.9 | .1D-5 | .1D-5 | .1D-5 | .1D-5 | .1D-5 | .1D-5 | .10.5 | .1D-5 | .1D-5 | .1D-4 |
| .1D-4 | 85 | 1.1 | .5D-6 | .5D-6 | .5D-6 | 5D-6 | .5D-6 | .5D-6 | .5D-6 | .5D-6 | .5D-6 | .5D-5 |
| .1D-5 | 100 | 1.4 | .4D-7 | .4D-7 | .4D-7 | .4D-7 | .4D-7 | .4D-7 | .4D-7 | .4D-7 | . $4 \mathrm{D}-7$ | .3D-6 |
| .1D-6 | 111 | 1.7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-7 | .1D-6 |
| .1D-7 | 130 | 2 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | .4D-9 | . $4 \mathrm{D}-8$ |
| .1D-8 | 161 | 2.5 | 2D-9 | , 2D-9 | ,212-9 | .2D-9 | .2D-9 | .2D-9 | 2D-9 | 2D-9 | .2D-9 | 20-8 |

Problem 4 is quite a stiff system. For the problem, the performances of the three routines are:
Runge-Kutta method requires excessive amounts of computation because of the need for small step-lengths which causes more expense because of the resulting high NFE calls and CPU time.

Gear's method is the most efficient for the stiff system.

The Adams routine requires excessive amounts of computation because of the need for small step-lengths which causes more expense because of the resulting high NFE calls
and CPU time.

Table 2.4

ODE Example 4

| Meth | Adams |  |  | Gear |  |  | Runge-Kutta |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TOL | Er1 Er2 Er3 | NFE | C $\mathbf{P}$ $\mathbf{U}$ | Er1 Er2 Er3 | NFE | C $\mathbf{P}$ $\mathbf{U}$ | Er1 Er2 Er3 | NFE | C <br> $\mathbf{P}$ <br> $\mathbf{U}$ |
| .1D-3 | .1D-6 .3D-4 .1D-4 | 5419 | 33 | . $5 \mathrm{D}-5.4 \mathrm{D}-5.5 \mathrm{D}-3$ | 35 | 1.4 |  |  |  |
| .1D-4 | .1D-6 .1D-4 .1D-4 | 5778 | 34 | .2D-5 .1D-5 .2D-3 | 43 | 1.5 |  |  |  |
| .1D-5 | .1D-6 .1D-6 .1D-4 | 5565 | 34 | .1D-6 .1D-6 .1D-4 | 56 | 1.6 |  |  |  |
| .1D-6 | .2D-6 .2D-6 .2D-4 | 5080 | 35 | .1D-7 .8D-8 .1D-5 | 55 | 1.7 | .2D-11 .4D-7 .2D-9 | 3978 | 7.8 |
| .1D-7 | .1D-6 .1D-6 .1D-4 | 6222 | 39 | .8D-9 .6D-9 .8D-7 | 83 | 2 | .4D-11 .6D-7 .2D-9 | 3989 | 7.8 |
| .1D-8 | .2D-7 .2D-7 .2D-5 | 6201 | 39 | .2D-9 .1D-9 .2D-7 | 79 | 2.1 | .2D-12 .3D-8 .1D-10 | 3992 | 8.4 |

Example 5 is solved on the interval $[-.43,0]$. For this example, the comparison of the performances of the three routines is similar to that for Examples 1 and 2.

Table 2.5

ODE Example 5


Example 6 is solved on the interval [0,2]. For this example, the three methods need a higher degree of tolerance proportionality, TOL $=10^{-9}, 10^{-10}$ and $10^{-11}$.

Table 2.6

## ODE Example 6

| METHOD | Runge-Kutta |  |  | Gear |  |  | Adams |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TOL | Er1 | NFE | CPU | Er1 | NFE | CPU | Er1 | NFE | CPU |
| $.1 D-08$ | $.6 D-08$ | 51 | 2 | $.1 D-11$ | 77 | 1.3 |  |  |  |
| $.1 D-09$ | $.5 D-09$ | 57 | 2.2 | $.6 \mathrm{D}-13$ | 122 | 1.5 | $.3 D-09$ | 43 | 1.5 |
| $.1 D-10$ | $.2 D-10$ | 44 | 2.5 | $.3 D-14$ | 206 | 1.7 | $.2 D-10$ | 50 | 1.5 |

### 2.5 Summary

In this chapter of the thesis we have considered a number of routines from the NAG D02 chapter of the library for ODEs. We are concerned only with methods for initial value problems, although some of the codes to solve boundary value problems make use of similar techniques. Three basic methods are employed; a fixed-order Runge-Kutta-Merson method, a variable-order Adams method and a variable-order BDF method.

The comparison of the performances of the three solvers is described in section 2.4 and demonstrated the more useful experiences for determining a best approximation. For example, the BDF routines are particularly well suited to the solution of stiff problems. For non-stiff problems the choice between the Merson and Adams methods will largely be governed by the cost evaluations of the functions appearing in the differential equation. If these are relatively cheap, the use of a Merson routine is likely to prove the more efficient.

In the next chapter, we address the problem of partial differential equations (PDE). It will be shown that systems of ODEs occur naturally in the application of approximation techniques to more complex problems. For example, the parabolic PDE,

$$
\begin{equation*}
\frac{\partial u(x, t)}{\partial t}=\frac{\partial^{2} u(x, t)}{\partial^{2} x} \quad 0<x<1, \quad t>0, \tag{2.22}
\end{equation*}
$$

with appropriate initial and boundary conditions which ensure a unique solution, may be approximated using the method of lines (see Carroll [13] and the references quoted therein), which will be mentioned in the next chapter, to yield the system

$$
\mathrm{u}^{\prime}(\mathrm{t})=\mathrm{Au}(\mathrm{t})+\mathrm{b}(\mathrm{t}),
$$

where $A$ is a tridiagonal matrix, $u^{T}=\left(u_{1}(t), u_{1}(t), \ldots, u_{n}(t)\right)$, and $u_{i}(t)$ represents an approximation to $u\left(x_{i}, t\right)$ for some $x_{i} \in[0,1]$ and $t>0$.

## Chapter 3

## Parabolic Partial Differential Equations

### 3.1 Introduction

In this, the second phase of the project, we review efficient numerical solution techniques for systems of one-dimensional linear parabolic partial differential equations (PDEs for short). We also consider the application of the numerical PDE software to approximate the solutions of a number of test problems frequently cited in the literature.

Partial differential equations (PDEs) occur widely in science and engineering and have led to the development of a large body of numerical methods. The most successful PDE computer codes, for example PDECOL [9], and D03PxF (where $x$ is A, B or G) [2] (NAG library) have involved the discretisation of the problem in the space dimension (semidiscretisation) and its reduction to a system of ODEs which can be routinely solved using available high quality ODE integrators. When semidiscretisation is performed using Finite Differences the procedure is known as the "Method of Lines" (MOL).

We briefly outline a numerical solution procedure for the approximate integration of systems of one-dimensional parabolic PDEs using the composite integration scheme ( $\because B D F 2$ ). This applies the numerical method of lines to approximate the solution of parabolic PDEs in one space dimension. More generally, we approximate the spatial derivatives by discrete values of the solution at a set of mesh points in space. We can use these approximations to represent the partial differential equations at each mesh point, giving a semidiscrete system of ODEs in the time direction. The resulting ODE system is integrated numerically using a second-order L-stable composite integration scheme [8] using variable stepsize sequences, Carroll [13].

We present some numerical experiments to evaluate the performance and effectiveness of three software sources, DO3PGF [2], PDECOL [9] and $\theta$ BDF2 [13].

### 3.2 PDE Software

Partial differential equations (PDEs) occur widely in science and engineering and have led to the development of a large body of numerical methods. One of the most popular approaches in the numerical solution of PDEs is the Method of Lines (MOL). The MOL approach involves semidiscretizing the PDE system - replacing the spatial derivative terms with finite difference approximations to yield a system of time dependent ODEs.

In this section, we are interested in applying basic solution techniques to systems of onedimensional parabolic PDEs, using PDE software such as PDECOL [9], $\theta$-BDF2 [13] and NAG Routine D03PGF [2].

NAG D03PGF: This routine is designed to solve a general system of N parabolic equations of the form

$$
\begin{equation*}
C_{i} \frac{\partial u_{i}}{\partial t}=\sum_{j=1}^{N} \frac{\partial}{\partial x}\left(g_{i j} \frac{\partial U_{i}}{\partial x}\right)+f_{i} \quad i=1,2, \ldots, N \tag{3.1}
\end{equation*}
$$

subject to the general boundary conditions

$$
\begin{equation*}
p_{i}(t) U_{i}+q_{i}(t) \frac{\partial U_{i}}{\partial x}=r\left(U_{i}, t\right) \quad i=1,2, \ldots, N \tag{3.2}
\end{equation*}
$$

in either Cartesian, polar or spherical polar coordinates.

The method of lines approach is used in which the spatial terms are discretised using second order centred Finite Differences. If the boundary conditions are such that q is non-zero then the accuracy at the boundaries is of order one. Discontinuities are permitted between the initial and boundary values and the user may choose from a limited number of fixed non-uniform grids. The ODE integrator for this package is the GEARIB variable order/variable step code. The initial time step is automatically chosen by the program and subsequent steps are chosen so that a user specified accuracy in the time integration is maintained. The code is robust and allows automatic resetting of the integration in the case of rejected time steps.

PDECOL: This routine by Madsen and Sincovec [9] solves the general system of N
partial differential equations

$$
\begin{equation*}
\frac{\partial U_{i}}{\partial t}=f_{i}\left(t, x, U, U_{x}, U_{x x}\right) \quad i=1,2, \ldots, N \tag{3.3}
\end{equation*}
$$

Since the system incorporates ODEs and the three standard types of PDEs then for each equation of the system zero, one or two boundary conditions may be needed. They must be of the form

$$
\begin{equation*}
b_{i}\left(U, U_{x}\right)=z_{i}(t) \tag{3.4}
\end{equation*}
$$

and most be consistent with the initial conditions. The program semidiscretises in space using a Finite Element collocation procedure with piecewise polynomial test functions. The degree of these polynomials is required to be higher than the degree of the PDE(s) being solved. The user specifies the numerical grid and the result of the automatic semidiscretisation is the ODE system

$$
\begin{equation*}
A \frac{d U}{d t}=g(t, U) \tag{3.5}
\end{equation*}
$$

The main restrictions in PDECOL are the requirements for continuity between the initial and boundary conditions which limits its applicability. Also, the nonconservative nature of collocation methods in general makes them inappropriate for problems where a conservation law must be satisfied.

日BDF2: This routine by Carroll [13] solves the general numerical integration of systems of PDEs of size NPDE

$$
\begin{equation*}
U_{t}=\left[A(x, t, u) U_{x}\right]_{x}+g\left(x, t, u, U_{x}\right) \tag{3.6}
\end{equation*}
$$

for $(x, t) \in[a, b] *\left[0, t^{0}\right]$, subject to appropriate initial and boundary conditions, where $\mathrm{U}=\left[\mathrm{u}_{1}, \mathrm{u}_{2}, \ldots, \mathrm{u}_{\text {NPDE }}\right]^{\mathrm{T}}, \mathrm{A}$ is an NPDE * NPDE uniformly positive definite matrix whose coefficients $\left\{\mathrm{a}_{\mathrm{ij}}\right\}$ are uniformly Lipschitz continuous as functions of $\mathbf{u}, \mathbf{x}$, and $t$. We assume that g , together with the prescribed initial conditions, $\mathrm{U}(\mathrm{x}, 0)=\mathrm{U}_{0}(\mathrm{t})$, are uniformly Lipschitz continuous on $[0,1] *[0, T]$ and $[0,1]$ respectively.

The numerical method of lines is used to solve a system of PDEs in one space dimension. By approximationg the spatial derivatives by discrete values of the solution at a set of mesh points in space, then using these approximations to represent the PDEs
at each mesh point, gives a semidiscreste system of ODEs in the time direction. The resulting ODE system is integrated numerically using a second-order L -stable composite integration scheme [8] using variable stepsize integration. The main numerical solution procedures in routine $\theta \mathrm{BDF} 2$ is:

Numerical Method of Lines [13]. We consider the application of the MOL when applied to approximate the solution of the single parabolic equation

$$
\begin{equation*}
u_{t}=k u_{x x}+g\left(x, t, u, u_{x}\right) \quad x \in[0,1], \quad k>0 \tag{3.7}
\end{equation*}
$$

subject to the boundary conditions

$$
\begin{aligned}
& \mathrm{a}_{0}(\mathrm{t}) \mathrm{u}+\mathrm{b}_{0}(\mathrm{t}) \mathrm{u}_{\mathrm{x}}=\mathrm{c}_{0}(\mathrm{t}) \text { at } \mathrm{x}=0, \\
& \mathrm{a}_{1}(\mathrm{t}) \mathrm{u}+\mathrm{b}_{1}(\mathrm{t}) \mathrm{u}_{\mathrm{x}}=\mathrm{c}_{1}(\mathrm{t}) \text { at } \mathrm{x}=1,
\end{aligned}
$$

and the initial condition

$$
\mathrm{u}(\mathrm{x}, 0)=\mathrm{u}_{0}(\mathrm{t}) \quad 0 \leq \mathrm{x} \leq 1,
$$

It gives a boundary value problem in $x$ and an initial value problem in $t$. We represent the partial derivatives on the right-hand side of (3.7) by appropriate difference approximations using a fixed uniform or non-uniform mesh on the interval [0,1]. Using these approximations at each mesh point reduces the problem to solving a system of ODEs in time which we subsequently solve using a temporal integration scheme. By generating the mesh

$$
\left\{x_{i}\right\}, i=1,2, \ldots, N, \quad x_{i+1}=x_{i}+h_{i}, \quad 1 \leq i \leq N-1,
$$

with $\mathrm{x}_{1}=0$ and $\mathrm{x}_{\mathrm{N}}=1$, a possible semidiscretization of (3.7) is

$$
\begin{equation*}
u_{i}^{\prime}=k D_{+} D_{-} u_{i}+g\left(x, t, u, D_{0} u_{i}\right) \quad 2 \leq i \leq N-1, \tag{3.8}
\end{equation*}
$$

where $D_{.}, D_{0}$ and $D_{+}$are respectively the backward, central and forward difference operators:

$$
\begin{equation*}
D_{-} u_{i}=\frac{u_{i}-u_{i-1}}{h_{i-1}}, \quad D_{0} u_{i}=\frac{u_{i+1}+u_{i-1}}{h_{i-1}+h_{i}} \quad D_{+} u_{i}=\frac{u_{i+1}-u_{i}}{h_{i}}, \tag{3.9}
\end{equation*}
$$

so that the standard three point difference approximation to the second derivative is give by

$$
\begin{equation*}
u_{x x} \approx D_{+} D_{-} u_{i}=\frac{2}{h_{i-1} h_{i}\left(h_{i-1}+h_{i}\right)}\left[h_{i} u_{i-1}-\left(h_{i-1}+h_{i}\right) u_{i}+h_{i-1} u_{i+1}\right] \tag{3.10}
\end{equation*}
$$

The case $\mathrm{i}=1$ and $\mathrm{i}=\mathrm{N}$ are treated separately and their forms depend on the type of boundary conditions (see Carroll [13]).

Temporal Integration: The semidiscretization employed in the numerical method of lines leads to the system of ODEs

$$
\begin{equation*}
u^{\prime}=f(t, u) \quad u(0)=u_{0} \tag{3.11}
\end{equation*}
$$

where $u=\left[u_{1}, u_{2}, \ldots u_{M}\right]$ where $M=N *$ NPDE. To solve the system (3.11) numerically, we step from $t_{n}$ to $t_{n}+\Delta t$ by the application of a composite integration formula [13]. For some $0<\gamma<1$ we approximate $u\left(x, t_{n}+\gamma \Delta t\right)$ using the $\theta$-scheme

$$
\begin{equation*}
u_{n+\gamma}=u_{n}+\gamma \Delta t\left[(1-\theta) f_{n}+\theta f_{n+\gamma}\right] \quad 0<\theta \leq 1 \tag{3.12}
\end{equation*}
$$

and the solution is used in a 2-step backward differentiation type formula to approximate $u\left(x, t_{n+1}\right)$ as follows:

$$
\begin{equation*}
\alpha_{0} u_{n}+\alpha_{1} u_{n+\gamma}+\alpha_{2} u_{n+1}=\Delta t f_{n+1} \tag{3.13}
\end{equation*}
$$

The coefficients $\gamma, \alpha_{0}, \alpha_{1}$ and $\alpha_{2}$ which were presented in [8, 13] are generated by requiring that the overall scheme is second-order accurate in time and also that both formulae share a common iteration matrix:

$$
\begin{equation*}
B=I-\theta \gamma \Delta t J, \quad J=\frac{\partial f}{\partial u}, \tag{3.14}
\end{equation*}
$$

in a modified Newton iteration scheme which was described in [13]. An estimation of local truncation error (LTE) at each time step was considered in [8].

### 3.3 Numerical Experiments

In this section, we consider the application of the three solvers that were mentioned in the last section to approximate the solutions of a number of parabolic equations frequently cited in the literature. We present a comparison of the results of the numerical experiments for the three solvers, OBDF2 [13], DO3PGF [2] and PDECOL [9].

The notation used and the statistics collected for the experiments include :

NSTEP the number of the number of integration steps.
NFE the number of function evaluations.
NJE the number of Jacobian evaluations.
NFCALLS the number of calls to the function $\left[A(x, t, u) u_{x}\right] x+g\left(x, t, u, u_{x}\right)$.
ERRGLO the maximum absolute component of error over the integration interval.
CPU the execution time in seconds on a DEC VAX/6230.
TOL a scalar value of ATOL.

## PDE Problem 1

P1: This is the first example taken from [13]:

$$
\begin{aligned}
& U_{t}=U_{x x}+\pi^{2} \sin (\pi x), \\
& u(0, t)=u(1, t)=1 ; \quad t \geq 0, \quad u(x, 0)=1, \quad 0 \leq x \leq 1,<1, \\
& \text { whose exact solution is } \\
& \qquad u(x, t)=1+\left[1-e^{-\pi^{2}}\right] \sin (\pi x) .
\end{aligned}
$$

Table 2.11

The results for P1 using $\operatorname{\theta BDF} 2$ the TOL between $10^{-2}$ and $10^{-5}$ with a uniform mesh ( $\mathrm{N}=21$ )

| TOL | $\mathbf{0 . 1 D - 0 1}$ | $\mathbf{0 . 1 D - 0 2}$ | $\mathbf{0 . 1 D - 0 3}$ | $\mathbf{0 . 1 D - 0 4}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 11 | 20 | 35 | 64 |
| NFE | 34 | 69 | 150 | 301 |
| NJE | 4 | 5 | 7 | 8 |
| ERRGLO | $.95 \mathrm{D}-2$ | $.36 \mathrm{D}-2$ | $.23 \mathrm{D}-2$ | $.21 \mathrm{D}-2$ |
| NfCALLS | 646 | 1311 | 2850 | 5719 |
| CPU | 0.32 | 0.60 | 1.29 | 2.58 |

## Table 2.12

The results for P1 using $\theta$ BDF2
the mesh points between $\mathrm{N}=21$ and 81 with a fixed $\mathrm{TOL}=\left(10^{-3}\right)$

| $\mathbf{N}$ | $\mathbf{2 1}$ | $\mathbf{4 1}$ | $\mathbf{6 1}$ | $\mathbf{8 1}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 20 | 20 | 20 | 20 |
| NFE | 69 | 69 | 69 | 69 |
| NJE | 5 | 5 | 5 | 5 |
| ERRGLO | $.36 \mathrm{D}-2$ | $.27 \mathrm{D}-2$ | $.23 \mathrm{D}-2$ | $.23 \mathrm{D}-2$ |
| NfCALLS | 1311 | 2691 | 4071 | 5451 |
| CPU | 0.62 | 1.17 | 1.74 | 2.3 |

Table 2.13

The results for P1 using D日3PGF (NAG Routine) the mesh points between $\mathrm{N}=21$ and 81 with a fixed $\mathrm{TOL}=\left(10^{-3}\right)$

| $\mathbf{N}$ | $\mathbf{2 1}$ | $\mathbf{4 1}$ | $\mathbf{6 1}$ | $\mathbf{8 1}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 44 | 46 | 45 | 45 |
| NFE | 135 | 259 | 296 | 376 |
| NJE | 4 | 5 | 4 | 4 |
| ERRGLO | $0.5 \mathrm{D}-02$ | $0.41 \mathrm{D}-02$ | $0.4 \mathrm{D}-02$ | $0.4 \mathrm{D}-02$ |
| NfCALLS | 1344 | 2870 | 3965 | 5265 |
| CPU | 0.83 | 1.66 | 2.32 | 3.06 |

The results for P1 using PDECOL Routine the mesh points between $\mathrm{N}=21$ and 81 with a fixed $\mathrm{TOL}=\left(10^{-3}\right)$

| $\mathbf{N}$ | $\mathbf{2 1}$ | $\mathbf{4 1}$ | $\mathbf{6 1}$ | $\mathbf{8 1}$ |
| ---: | ---: | ---: | ---: | ---: |
| STEPS | 31 | 30 | 31 | 32 |
| NFE | 41 | 41 | 47 | 46 |
| NJE | 7 | 8 | 8 | 8 |
| NfCALLS | 1764 | 3444 | 5856 | 7614 |
| CPU | 0.77 | 1.50 | 2.34 | 3.18 |

We first apply the composite scheme, $\theta \mathrm{BDF} 2$, to solve $\mathbf{P 1}$ over a range of tolerance values between $10^{-2}$ and $10^{-5}$ using a fixed uniform mesh of width $1 / 20$. The results are presented in Table 2.11.

As a contrast, we also apply the scheme to solve the problem over a range of spatial mesh densities using a fixed tolerance of integration $\left(10^{-3}\right)$ and we compare the results of $\theta$ BDF2 with the corresponding statistics for NAG routine D03PGF and PDECOL. The details for these are given in Tables 2.12, 2.13 and 2.14 respectively. It is clear that the $\theta B D F 2$ routine is the most efficient of the three routines in terms of the CPU time used, D03PGF is more efficient (in terms of computation overhead) and PDECOL takes the largest CPU time.

## PDE Problem 2

P2: This is the second example taken from [13]:

$$
\begin{equation*}
U_{t}=\sigma U_{x x}+f(x, t) \quad 0<x<1, \quad 0<t \leq 1, \tag{3.16}
\end{equation*}
$$

where the function $f(\mathrm{x}, \mathrm{t})$, the Dirichlet boundary conditions and initial conditions are chosen so that the exact solution is

$$
u(x, t)=\tanh [5(x+t-1)]=10
$$

Table 2.15

The results for P2 using $\theta$ BDF2
the TOL between $10^{-2}$ and $10^{-5}$ with a uniform mesh ( $\mathrm{N}=41$ )

| TOL | $\mathbf{0 . 1 D - 0 1}$ | $\mathbf{0 . 1 D - 0 2}$ | $\mathbf{0 . 1 D - 0 3}$ | $\mathbf{0 . 1 D - 0 4}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 13 | 25 | 52 | 92 |
| NFE | 65 | 123 | 254 | 474 |
| NJE | 3 | 4 | 8 | 10 |
| ERRGLO | $.31 D-1$ | $.72 \mathrm{D}-2$ | $.18 \mathrm{D}-2$ | $.48 \mathrm{D}-3$ |
| NfCALLS | 2535 | 4797 | 9906 | 18486 |
| CPU | 1.17 | 2.16 | 4.44 | 8.26 |

## Table 2.16

The results for $\mathbf{P 2}$ using $\theta$ BDF2
the mesh points between $\mathrm{N}=21$ and 81 with a fixed $\mathrm{TOL}=\left(10^{-3}\right)$

| $\mathbf{N}$ | $\mathbf{2 1}$ | $\mathbf{4 1}$ | $\mathbf{6 1}$ | $\mathbf{8 1}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 25 | 25 | 25 | 26 |
| NFE | 115 | 123 | 123 | 122 |
| NJE | 4 | 5 | 5 | 5 |
| NfCALLS | 2185 | 4797 | 7257 | 9638 |
| ERRGLO | $.73 \mathrm{D}-2$ | $.72 \mathrm{D}-2$ | $.72 \mathrm{D}-2$ | $.72 \mathrm{D}-2$ |
| CPU | 1.05 | 2.16 | 3.23 | 4.25 |

Table 2.17

The results for $\mathbf{P 2}$ using D03PGF(NAG)
the mesh points between $\mathrm{N}=21$ and 81 with a fixed $\mathrm{TOL}=\left(10^{-3}\right)$

| $\mathbf{N}$ | $\mathbf{2 1}$ | $\mathbf{4 1}$ | $\mathbf{6 1}$ | $\mathbf{8 1}$ |
| ---: | ---: | ---: | ---: | ---: |
| NSTEP | 54 | 54 | 48 | 48 |
| NFE | 146 | 142 | 240 | 296 |
| NJE | 4 | 2 | 3 | 3 |
| NfCALLS | 1575 | 2747 | 4087 | 5103 |
| ERRGLO | $0.11 \mathrm{D}-01$ | $0.95 \mathrm{D}-02$ | $0.90 \mathrm{D}-02$ | $0.87 \mathrm{D}-02$ |
| CPU | 0.93 | 1.61 | 2.29 | 2.89 |

We take $\sigma=10^{-2}$. On a uniform mesh of width $1 / 40$, we solve this equation using $\operatorname{\theta BDF} 2$ for four tolerance values and the results are given in Table 2.15. As before, we also apply $\theta$ BDF2 to solve the problem over a range of spatial mesh densities between $\mathrm{N}=21$ and $\mathrm{N}=81$ using a fixed tolerance of integration $\left(10^{-3}\right)$ and compare the results with the corresponding statistics for NAG Routine D03PGF. The details are presented in Tables 2.16 and 2.17 respectively. The composite integration scheme OBDF2 compares favourably with NAG Routine D03PGF in terms of accuracy, but it is less efficient in terms of NfCALLS (and hence CPU).

## PDE Problem 3

P3: We take our third example from [13] and it is a coupled system of two equations.

$$
\left\{\begin{array}{l}
u_{t}=\left[v^{2} u_{x}\right]_{x}-u v-u^{2}+10  \tag{3.17}\\
v_{t}=\left[u^{2} v_{x}\right]_{x}+u_{x x}+u v-v^{2}
\end{array}\right.
$$

where $0 \leq \mathrm{x} \leq 1$ and $0<\mathrm{t} \leq 1$, the boundary conditions are

$$
\begin{gathered}
u(0, t)=1 / 2, \\
u_{x}(1, t)=1 / 2-\sin (u v), \quad v_{x}(1, t)=1+\cos (u v),
\end{gathered}
$$

and the initial conditions are

$$
u(x, 0)=1 / 2(x+1), \quad v(x, 0)=\Pi \quad 0 \leq x \leq 1 .
$$

Table 2.18

The results for P3 using 0 BDF2
the TOL between $10^{-2}$ and $10^{-5}$ with a uniform mesh ( $\mathrm{N}=31$ )

| TOL | $\mathbf{0 . 1 D - 0 1}$ | 0.1D-02 | 0.1D-03 | $\mathbf{0 . 1 D - 0 4}$ |
| ---: | ---: | ---: | :---: | :---: |
| STEPS | 29 | 37 | 69 | 116 |
| NFE | 132 | 192 | 444 | 835 |
| NJE | 10 | 13 | 16 | 19 |
| NfCALLS | 3960 | 5760 | 13320 | 25050 |
| CPU | 2.14 | 3.05 | 6.71 | 12.48 |

Table 2.19

The results for P3 using NAG routine D03PGF the TOL between $10^{-2}$ and $10^{-5}$ with a uniform mesh ( $\mathrm{N}=31$ )

| TOL | $\mathbf{0 . 1 D - 0 1}$ | $\mathbf{0 . 1 D - 0 2}$ | $\mathbf{0 . 1 D - 0 3}$ | $\mathbf{0 . 1 D - 0 4}$ |
| ---: | ---: | ---: | :---: | :---: |
| STEPS | 86 | 83 | 120 | 180 |
| NFE | 468 | 588 | 951 | 1201 |
| NJE | 6 | 8 | 13 | 16 |
| NfCALLS | 4309 | 4619 | 7347 | 9982 |
| CPU | 2.97 | 3.14 | 4.98 | 6.98 |

Table 2.20

> The results for P3 using PDECOL routine the TOL between $10^{-2}$ and $10^{-5}$ with a uniform mesh $(\mathrm{N}=31)$

| TOL | 0.1D-01 | 0.1D-02 | 0.1D-03 | 0.1D-04 |
| ---: | ---: | ---: | ---: | ---: |
| steps | 28 | 47 | 76 | 115 |
| NFE | 43 | 68 | 108 | 159 |
| NJE | 12 | 12 | 15 | 22 |
| NfCALLS | 2728 | 4278 | 6758 | 9920 |
| CPU | 2.19 | 2.89 | 4.34 | 6.40 |

We compare the three routines, $\theta$ BDF2, D03PGF and PDECOL by applying them to approximate the solution of the problem over a range of tolerance values on a fixed uniform mesh of size $1 / 30$. The results for the three methods are presented in Tables 2.18, 2.19 and 2.20.

Of the three approaches, PDECOL appears to be slightly faster than D03PGF both having comparable CPU times. The $\begin{aligned} & \text { BDF2 scheme is the least competitive for this problem and }\end{aligned}$ is particularly inefficient for the smaller tolerance values.

### 3.4 Summary

In this chapter we have discussed three sources of PDEs, PDECOL, D03PGF and $\theta B D F 2$, including the main solution procedures. We introduced some solution methods for PDEs such as a discretization in both space and or time, using finite differences and forming an approximation using the $\theta$-scheme. The above methods have been incorporated into the numerical procedure for the approximate solution of integro-differential equations. It is this which will be addressed in the following chapters. The use of such state-of-the-art routines provides a foundation and model for the main goal of this thesis, i.e., to implement an efficient solver to approximate the solution of integro-differential equations.

## Chapter 4

## Parabolic Integro-Differential Equations I

### 4.1 Introduction

In this chapter we shall turn to the main subject of the thesis: the numerical methods for the solution of integro-differential equations of parabolic type.

We consider two different time discretizations of an integro-differential equation of parabolic type based on the Backward Euler and the Crank-Nicolson schemes, respectively, namely, the methods reduce to the backward Euler and the Crank-Nicolson schemes if the integral term is absent. The integral term is approximated in each case by a quadrature rule with relatively high-order truncation error, so that a relatively large time step can be used for the quadrature, in order to reduce the memory and computational requirements of the method (see Sloan and Thomee [17]).

Many authors suggested some quadrature rules which are more sparse than the standard methods, but nevertheless retain the order of accuracy of these time discretization schemes. Roux and Thomee [15], Sloan and Thomee [17] and Zhang [20] suggested such kinds of methods which we collect and implement. In this chapter we describe certain aspects of the numerical algorithms which were suggested by the above authors. We give a more detailed description of the collected methods and present the quadrature schemes.

### 4.2 Theoretical Considerations

In this section we give a brief introduction to the theoretical background, fields of applications and recent developments of methods in the numerical solution of integrodifferential equations.

We shall consider equations of the form

$$
\begin{equation*}
\frac{\partial}{\partial t} u(x, t)+A u(x, t)=\int_{0}^{1} B(t, s) u(x, s) d s+f(x, t) \quad(x, t) \in \bar{\Omega} \quad\left(0, t^{0}\right], \tag{4.1}
\end{equation*}
$$

subject to the homogeneous Dirichlet boundary condition

$$
u(x, t)=0 \quad(x, t) \in \partial \Omega \quad\left(0, t^{0}\right]
$$

and the initial condition

$$
u(x, 0)=u_{0}(x) \quad x \in \bar{\Omega} .
$$

Here A is an elliptic operator of the form

$$
A=-\sum_{i, j=1}^{d} \frac{\partial}{\partial x_{i}}\left(a_{i j}(x) \frac{\partial}{\partial x_{J}}\right)+a_{0}(x) I,
$$

Where the matrix $\left(\mathrm{a}^{\mathrm{ij}}(\mathrm{x})\right)_{\mathrm{i}, \mathrm{j}=1}$ is symmetric and uniformly positive definite, and $\mathrm{a}_{0}(\mathrm{x})$ is nonnegative on $\bar{\Omega}$. Further

$$
\begin{equation*}
B(t, s)=-\sum_{i j=1}^{d} \frac{\partial}{\partial x_{i}}\left(b_{i j}(x ; t, s) \frac{\partial}{\partial x_{j}}\right)+\sum_{i, j=1}^{d} b_{j}(x ; t, s) \frac{\partial}{\partial x_{j}}+b_{0}(x) I \tag{4.5}
\end{equation*}
$$

is a partial differential operator of at most second order. It is assumed that the coefficients $\mathrm{a}_{\mathrm{ij}}(\mathrm{x}), \mathrm{a}_{0}(\mathrm{x}), \mathrm{b}_{\mathrm{ij}}(\mathrm{x} ; \mathrm{t}, \mathrm{s}), \mathrm{b}_{\mathrm{j}}(\mathrm{x} ; \mathrm{t}, \mathrm{s}), \mathrm{b}_{0}(\mathrm{x} ; \mathrm{t}, \mathrm{s})$ and $f=f(\mathrm{x}, \mathrm{t}), \mathrm{u}=\mathrm{u}(\mathrm{x}, \mathrm{t})$ are sufficiently smooth functions and real-valued.

Such problems and variants of them arise in many applications, such as in heat conduction in materials with memory, compression of poro-viscoelastic media, nuclear reactor dynamics, etc., and have been extensively investigated in the mathematical literature, see, e.g., the references quoted in Thomee [18] and Zhang [20].

The numerical solution of problems like that in (4.1) by means of finite differences has been studied by many authors such as Douglas and Jones (1962), Habetler and Shciffman (1970), Pavlov (1968), Rektorys (1963), Tavernini (1977) and Thompson (1973). Recently finite element methods have been studied by Greenwell, Yanik and Fairweather (1986), Roux and Thomee (1986), Thomèe and Zhang (1989), Cannon and Lin (1987) and etc. Howevere, special attention to the time stepping has been studied by Sloan and

Thomée [17], Zhang [20], Thomee [18], [19], and Le Roux and Thomée [15]. The present work is based on implementing and extending some of the methods mentioned by the above authors.

Zhang [20] considered the time discretization of the equation (4.1)

$$
\begin{align*}
u_{t}+A u=\int_{0}^{t} B(t, s) u(s) d s+f(t) & \equiv \tilde{B} u(t)+f(t) \quad 0<t \leq t^{0}  \tag{4.6}\\
u(0) & =u_{0}
\end{align*}
$$

In applications, A will often be a second order elliptic operator, and $\mathrm{B}(\mathrm{t}, \mathrm{s})$ an arbitrary second order partial differential operator of order $\beta \leq 2$. Let $\mathrm{k}>0$ be the step-size and $\mathrm{t}_{\mathrm{n}}=\mathrm{nk}$, let $\sigma_{\mathrm{n}}(\phi)$ be a quadrature rule with weight, $\left\{\omega_{\mathrm{nj}}\right\}$, such that, for $\phi \in \mathrm{C}\left(\left[0, \mathrm{t}^{0}\right]\right)$ and $\phi_{\mathrm{j}}=\phi\left(\mathrm{t}_{\mathrm{j}}\right)$, we have

$$
\begin{equation*}
\sigma_{n}(\phi)=\sum_{j=0}^{n-1} \omega_{n j} \phi_{j} \approx \int_{0}^{1} \phi(s) d s \tag{4.7}
\end{equation*}
$$

The time discrete versions of (4.1) we shall study are of the form

$$
\begin{align*}
& \bar{\partial}_{t} U_{n}+A U_{n}=\sum_{j=0}^{n-1} \omega_{n j} B\left(t_{n}, t_{j}\right) U_{j}+f_{n} \\
& \equiv \sigma_{n}\left(B\left(t_{n}\right) U\right)+f_{n}  \tag{4.8}\\
& \equiv \sigma_{n}(B U)+f_{n}, \quad 0<t_{n} \leq t^{0} \\
& U_{0}=u_{0}
\end{align*}
$$

For an example, we approximate (4.6) by a Backward Euler (B.E.) type scheme defined by

$$
\begin{equation*}
\frac{U_{n}-U_{n-1}}{K}+A U_{n}=\sigma_{n}\left(B\left(t_{n}\right) U\right)+f\left(t_{n}\right) \quad 0<t_{n} \leq t^{0} \tag{4.9}
\end{equation*}
$$

It is easy to see that the simplest quadrature rule that is consistent with the $\mathrm{O}(\mathrm{k})$ accuracy of the Backward-Euler scheme is the rectangular rule, that is the rule with weights

$$
\omega_{\mathrm{nj}}=\mathrm{k} \quad \text { for } 0 \leq \mathrm{j} \leq \mathrm{n}-1
$$

However, to calculate $u_{n}$ by this scheme using the standard rectangular rule, we have to use, and thus store, all the previous values of the solution $\mathrm{u}_{0}, \ldots, \mathrm{u}_{\mathrm{n}-1}$, and hence, a huge amount of memory will be occupied for the calculation. In particular, to compute $\mathrm{u}_{\mathrm{n}}$, $0<\mathrm{t}_{n} \leq \mathrm{t}^{0}$, the solution needs to be stored at $\mathrm{t}^{0} / \mathrm{k}$ time levels. Obviously, this is undesirable in practice. Thus the number of time levels used in the quadrature will be one of the key criteria in choosing quadrature rules. this will be the main topic in our later discussion.

In order to reduce the storage requirement, Sloan and Thomée [17], Pani, Thomée and Wahlbin [16], Le Roux, Thomėe [15] and Zhang [20]. proposed some quadrature rules which are more sparse than the standard methods, but nevertheless retain the order of accuracy of these time discretization schemes.

The present work is based on some proposed methods which were suggested by Le Roux and Thomée [15] and Zhang [20], and may be considered as an implementation of existing solvers. A detailed discussion of the methods shall be presented in the rest of this thesis.

### 4.3 Time Discretization

This section deals with the time discretization of integro-differential equations of parabolic type (4.1), based on the Backward-Euler and Crank-Nicolson discretization schemes [17].

Firstly, we briefly consider the concrete situation of a partial integro-differential equation of parabolic type in space and time.

Subdivide the $x$-t plane into sets of equal rectangles of side $\Delta x=h, \Delta t=k$, and let the co-ordinates ( $\mathrm{x}_{\mathrm{i}}, \mathrm{t}_{\mathrm{j}}$ ) of the representative mesh point be

$$
\mathrm{x}_{\mathrm{i}}=\mathrm{ih}, \mathrm{t}_{\mathrm{i}}=\mathrm{jk}, \quad \mathrm{i}=0,1, \ldots, \mathrm{M} ; \quad \mathrm{j}=0,1, \ldots, \mathrm{~N},
$$

where $\mathrm{i}, \mathrm{j}, \mathrm{M}$ and N are integers, $\mathrm{h}=1 / \mathrm{M}$ and $\mathrm{k}=\mathrm{t}^{0} / \mathrm{N}$.

In this work we restrict our attention to the time discretization and less attention will be
paid to the problem of space discretization. However, in practice the space and time discretization must be carried out simultaneously.

More generally, we consider a time discretization based on the " $\theta$-scheme", where $\theta$ is some value between 0 and 1 . We discretize the parabolic integro-differential equation using the " $\theta$-scheme" and replace the integral term by a quadrature formula, $\sigma^{\text {n }}$. The result is the scheme,

$$
\begin{equation*}
\frac{u_{n}-u_{n-1}}{k}+A\left(\theta u_{n}+(1-\theta) u_{n-1}\right)=\sigma_{n}\left(B\left(\theta t_{n}+(1-\theta) t_{n-1}\right) u\right)+f\left(\theta t_{n}+(1-\theta) t_{n-1}\right) \tag{4.10}
\end{equation*}
$$

where

$$
\begin{equation*}
\sigma_{n}\left(B\left(\theta t_{n}+(1-\theta) t_{n-1}\right) u\right)=\sum_{j=0}^{n-1} \omega_{n j} B\left(\theta t_{n}+(1-\theta) t_{n-1}, t_{j}\right) u_{j} \tag{4.11}
\end{equation*}
$$

and the quadrature weight $\left\{\omega_{\mathrm{nj}}\right\}$ are chosen so that for $\varphi \in \mathrm{c}\left[0, \mathrm{t}^{0}\right]$ and $\varphi_{\mathrm{j}}=\varphi\left(\mathrm{t}_{\mathrm{j}}\right)$ we have

$$
\begin{equation*}
\sigma_{n}(\varphi)=\sum_{j=0}^{n-1} \omega_{n j} \varphi_{j} \approx \int_{0}^{\theta t_{n}+(1-\theta) t_{n-1}} \varphi(s) d s . \tag{4.12}
\end{equation*}
$$

The backward Euler scheme and the Crank-Nicolson scheme, which were defined in Zhang [20], are the special cases of this $\theta$-scheme when we choose $\theta=1$ and $\theta=1 / 2$, respectively. The precise choice of the quadrature formulas will be explained in the next section.

### 4.4 Quadrature Schemes

We consider some effective methods for the numerical solution of problem (4.1) and our particular concern in this work will be a class of quadrature rules whose quadrature weights $\left\{\omega_{\mathrm{nj}}\right\}$ are defied by,

$$
\begin{equation*}
\sum_{j=0}^{n-1} \omega_{n j} \phi_{j} \approx \int_{0}^{t_{n}} \phi(s) d s \tag{4.13}
\end{equation*}
$$

This class contains not only the natural rectangular rule, whose quadrature weights are $\omega_{\mathrm{nj}}=\mathrm{k}$, but also other rules with relatively high-order truncation error, so that a relatively large time step can be used for the quadrature, in order to reduce the memory and computational requirements of the methods.

In this section we shall focus our attention on the discussion of quadrature schemes based on the trapezoidal and Simpson's rules that have, as far as possible, the maximum time step or minimum number of quadrature points, without losing the order of accuracy of the whole scheme for problem (4.1) and present the amended quadrature schemes respectively based on the proposed methods developed by Sloan, Thomee [17] and Zhang [20].

Before considering further details of various quadrature schemes, we introduce in more detail one of the methods called, Modified Trapezoidal Rule I, proposed by Zhang [20]:

Using the trapezoidal rule based on larger subintervals, but with second order truncation error, we may then approximate the integral employing relatively fewer, $\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)$, nodal points, while retaining the $\mathrm{O}(\mathrm{k})$ convergence rate so that is consistent with the $O(\mathrm{k})$ accuracy of the backward-Euler scheme.

More precisely, we use a trapezoidal quadrature rule with second-order truncation error, but applied on larger intervals of length $\mathrm{k}_{1}=\mathrm{m}_{1} \mathrm{k}$, where $\mathrm{m}_{1}=\left[\mathrm{k}^{-1 / 2}\right]$. ([.] denoting integral part), so that $\mathrm{k}_{1}=\mathrm{O}\left(\mathrm{k}^{1 / 2}\right)$, ie., $\mathrm{k}_{1}{ }^{2}=\mathrm{O}(\mathrm{k})$. We set $\overline{\mathrm{t}}_{\mathrm{j}}=\mathrm{j} \mathrm{k}_{1}$ and define $\mathrm{j}_{\mathrm{n}}$ to be the largest integer such that $\overline{\mathrm{j}}_{\mathrm{in}}<\mathrm{t}_{\mathrm{n}},\left(0 \leq \mathrm{j} \leq \mathrm{n}, 0<\mathrm{t}_{\mathrm{n}} \leq \mathrm{t}^{0}\right)$. Thus we can also divide the integration interval $\left[0, \mathrm{t}_{\mathrm{n}}\right]$ as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{j-1}, \bar{t}_{j}\right] \bigcup\left[\bar{t}_{j_{j, n}} t_{n}\right] \tag{4.14}
\end{equation*}
$$

We shall apply the trapezoidal rule with stepsize $\mathrm{k}_{1}$ on $\left[0, \overline{\mathrm{t}}_{\mathrm{jn}}\right]$ and then use the rectangular rule with steplength k for the second subinterval $\left[\mathrm{t}_{\mathrm{n}}, \mathrm{t}_{\mathrm{n}}\right]$.

The quadrature approximation was defined by

$$
\begin{equation*}
\int_{0}^{t_{n}} \phi(s) d s \approx \sigma^{n}(\phi)=\frac{k_{1}}{2} \sum_{j=1}^{j_{n}}\left(\phi\left(\bar{t}_{j}\right)+\phi\left(\bar{t}_{j-1}\right)\right)+k \sum_{j_{n} m_{1}}^{n-1} \phi\left(t_{j}\right)=\sigma_{2}^{n}+\sigma_{1}^{n} \tag{4.15}
\end{equation*}
$$

Denote $S_{\max }$ as an upper bound for the maximal number of time levels of the solution stored during the calculation. The upper bound of the storage for this rule is given by [20]

$$
\begin{align*}
& S_{\max } \leq \frac{t^{0}}{\left(m_{1} k\right)}+m_{1} \\
& \text { since } m_{1}=O\left(k^{-1 / 2}\right)  \tag{4.16}\\
& S_{\max }=O\left(k^{-1 / 2}\right)
\end{align*}
$$

We next give a description of the slight differences in ideas between the methods of [15] and those proposed by [20], as follows:

In practice, based on the above method, one can consider a slight modification for the previous rule to make it more computationally effective. Following the ideas of [15], we can suggest

$$
\begin{equation*}
\bar{m}_{1}=n-m_{1} j_{n}-1, \quad \bar{k}_{1}=\bar{m}_{1} k \tag{4.17}
\end{equation*}
$$

on the remaining subinterval $\left[\mathrm{t}_{\mathrm{jn}}, \mathrm{t}_{\mathrm{n}}\right]$, whose length is at most $\mathrm{k}_{1}$ we apply once again the trapezoidal rule with stepsize $\bar{k}_{1}$ on $\left[\overline{\mathrm{t}}_{\mathrm{j},}, \mathrm{t}_{\mathrm{n}-1}\right]$ and the rectangular rule with step size k on $\left[\mathrm{t}_{\mathrm{n}-1}, \mathrm{t}_{\mathrm{n}}\right]$. The most obvious advantage of such an amended method is that it is similar to the earlier proposed method and has the same accuracy for the quadrature formulas but requires the minimum number of quadrature points to be stored on the remaining subinterval $\left[\mathrm{t}_{\mathrm{jn}}, \mathrm{t}_{\mathrm{n}}\right]$.

For a contrast, we also write the interval of integration as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{j-1,} \bar{t}_{j}\right] \bigcup\left[\bar{t}_{j_{m}} t_{n-1}\right] \bigcup\left[t_{n-1}, t_{n}\right] \tag{4.18}
\end{equation*}
$$

We thus define the quadrature approximation by

$$
\begin{equation*}
\sigma^{n}(\phi)=\frac{k_{1}}{2} \sum_{j=1}^{j_{n}}\left(\phi\left(\bar{t}_{j}\right)+\phi\left(\bar{t}_{j-1}\right)\right)+\frac{\bar{k}_{1}}{2}\left(\phi\left(\bar{t}_{j_{n}}\right)+\phi\left(t_{n-1}\right)\right)+k \phi\left(t_{n-1}\right) \equiv \sigma_{3}^{n}+\sigma_{2}^{n}+\sigma_{1}^{n} \tag{4.19}
\end{equation*}
$$

Following (4.16), an upper bound of the storage for this amended rule is given by

$$
S_{\max } \leq \mathfrak{t}^{0} /\left(\mathrm{m}_{1} k\right)+2
$$

since $2<\mathrm{m}_{1}$, we have

$$
S_{\max }=O(k-1 / 2)
$$

Since we apply the trapezoidal and rectangular rules only once on $\left[\bar{t}_{\mathrm{j},}, \mathrm{t}_{\mathrm{n}-1}\right]$ with $\mathrm{t}_{\mathrm{n}-1}$ as the node, ie. there are exactly 2 nodes on subinterval $\left[\mathrm{t}_{\mathrm{jn}}, \mathrm{t}_{\mathrm{n}}\right]$, we get the above formula for $S_{\text {max }}$.

Based on the ideas presented above, we shall discuss further methods for seeking more quadrature schemes based on the trapezoidal and Simpson's rules that have, as far as possible, the maximum time step or minimum number of quadrature points, without losing the order of accuracy of the whole scheme for problem (4.1).

Considering two special cases of (4.10) $\theta=1$ and $\theta=1 / 2$, we have following two schemes,

1) The Backward Euler scheme with the time step k for approximating (4.1) is

$$
\begin{gather*}
\frac{u_{n}-u_{n-1}}{k}+A u_{n}=\sum_{j=0}^{n-1} \omega_{n j} B\left(t_{n}, t_{j}\right) u_{j}+f_{n} \equiv \sigma_{n}\left(B\left(t_{n}\right) u\right)+f\left(t_{n}\right)  \tag{4.20}\\
0<t_{n} \leq t^{0} \\
u_{0}=u(0)
\end{gather*}
$$

where

$$
\begin{equation*}
\sigma_{n}(\phi)=\sum_{j=0}^{n-1} \omega_{n j} \phi_{j} \approx \int_{0}^{t_{n}} \phi(s) d s \tag{4.21}
\end{equation*}
$$

2) In the Crank-Nicolson scheme the time-discretized version of (4.1) is

$$
\begin{align*}
& \frac{u_{n}-u_{n-1}}{k}+A \frac{u_{n}+u_{n-1}}{2}=\sum_{j=0}^{n-1} \omega_{n} B\left(\frac{t_{n}+t_{n-1}}{2}, t_{j}\right) u_{j}+f\left(\frac{t_{n}+t_{n-1}}{2}\right) \\
& \equiv \sigma_{n}\left(B\left(\frac{t_{n}+t_{n-1}}{2}\right) u\right)+f\left(\frac{t_{n}+t_{n-1}}{2}\right) \quad 0<t_{n} \leq t^{0}  \tag{4.22}\\
& u_{0}=u(0)
\end{align*}
$$

where

$$
\begin{equation*}
\sigma_{n}(\phi)=\sum_{j=0}^{n-1} \omega_{n j} \phi_{j} \approx \int_{0}^{\left(s_{n}+t_{n-1}\right) / 2} \phi(s) d s \tag{4.23}
\end{equation*}
$$

We consider different quadrature formulas for approximating (4.21) and (4.23).

In the following we consider four different quadrature formulas for first-order type schemes:

## - The Rectangular Rule:

As mentioned in the previous section, the simplest quadrature rule that is consistent with the $O(k)$ accuracy of the Backward-Euler scheme is the rectangular rule. We show the rule which is due to Zhang [20] here. In this case the quadrature weights are $\omega_{\mathrm{nj}}=\mathrm{k}, 0 \leq$ $\mathrm{T}-\mathrm{j}<\mathrm{t}_{\mathrm{n}} \leq \mathrm{t}^{0}$. Here we define $\omega_{\mathrm{j}}=\mathrm{k}$.

$$
\begin{equation*}
\sum_{j=0}^{n-1} \omega_{j}=n k \leq t^{0} \tag{4.24}
\end{equation*}
$$

We may write the quadrature rule in the form

$$
\begin{equation*}
\sigma_{n}(\phi)=k \sum_{j=0}^{n-1} \phi_{j} \tag{4.25}
\end{equation*}
$$

For the rectangular rule, an upper bound on the storage is given by

$$
\mathrm{S}_{\max } \leq \mathrm{t}^{0} / \mathrm{k}=\mathrm{O}\left(\mathrm{k}^{-1}\right)
$$

## - The Trapezoidal Rule A:

As mentioned before, this quadrature rule which is due to Le Roux and Thomee [15]. It is based on the second-order trapezoidal rule but applied on intervals of length $k_{1}=m_{1} k$, where $\mathrm{m}_{1}=\mathrm{O}\left[\mathrm{k}^{-1 / 2}\right]$. ([.] denoting integral part), so that $\mathrm{k}_{1}=\mathrm{O}\left(\mathrm{k}^{1 / 2}\right)$. We set $\overline{\mathrm{t}}_{\mathrm{j}}=\mathrm{j} \mathrm{k}_{1}$ and define $j_{\mathrm{n}}$ to be the largest integer such that $\overline{\mathrm{t}}_{\mathrm{in}}<\mathrm{t}_{\mathrm{n}}$. We first write the interval of integration as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{j-1}, \bar{t}_{j}\right] \bigcup\left[\bar{t}_{j_{n^{\prime}}}, t_{n-1}\right] \bigcup\left[t_{n-1}, t_{n}\right] \quad \text { if } n>m_{1} \tag{4.26}
\end{equation*}
$$

More precisely, we shall apply the trapezoidal rule with stepsize $\mathrm{k}_{1}$ on $\left[0, \overline{\mathrm{t}}_{\mathrm{j} 1}\right]$. Then, on the remaining subinterval $\left[\bar{t}_{\mathrm{i}}, \mathrm{t}_{\mathrm{n}}\right]$, whose length is at most $\mathrm{k}_{1}$, let

$$
\begin{equation*}
\bar{m}_{1}=n-m_{1} j_{n}-1, \quad \bar{k}_{1}=\bar{m}_{1} k \tag{4.27}
\end{equation*}
$$

We apply once again the trapezoidal rule with stepsize $\overline{\mathrm{k}}_{1}$ on $\left[\overline{\mathrm{t}}_{\mathrm{j}}, \mathrm{t}_{\mathrm{n}-1}\right]$ and the rectangular rule with step size k on $\left[\mathrm{t}_{\mathrm{n}-1}, \mathrm{t}_{\mathrm{n}}\right]$. So we use for the approximation of the integral the trapezoidal rule on each of the subintervals except the last, where we use the rectangular rule, with $\mathrm{t}_{\mathrm{n}-1}$ as the node.

Using the above trapezoidal formula we then write an approximation to the integral term in (4.1) in the form

$$
\begin{equation*}
\sigma^{n}(\phi)=\frac{k_{1}}{2} \sum_{j=1}^{j_{n}}\left(\phi\left(\bar{t}_{j}\right)+\phi\left(\bar{t}_{j-1}\right)\right)+\frac{\bar{k}_{1}}{2}\left(\phi\left(\bar{t}_{j n}\right)+\phi\left(t_{n-1}\right)\right)+k \phi\left(t_{n-1}\right) \equiv \sigma_{3}^{n}+\sigma_{2}^{n}+\sigma_{1}^{n} \tag{4.28}
\end{equation*}
$$

We call this method the Trapezoidal Rule A. The corresponding quadrature weights are

$$
\omega_{n, j}= \begin{cases}\frac{k_{1}}{2} & j=0  \tag{4.29}\\ \frac{k_{1}}{k_{1}}+\frac{k_{1}}{\frac{2}{2}} & j=m_{1} i \quad 1 \leq i \leq j_{n}-1 \\ \frac{j=m_{1} j_{n}}{2}+k & j=m_{1} j_{n}+\bar{m}_{1}=(n-1) \\ 0 & \text { otherwise }\end{cases}
$$

The upper bound of the storage for this amended rule is given by

$$
\mathrm{S}_{\max } \leq \mathrm{t}^{0} /\left(\mathrm{m}_{1} \mathrm{k}\right)+2=\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)
$$

since we apply the trapezoidal and rectangular rules only once on $\left[\mathrm{t}_{\mathrm{jn}}, \mathrm{t}_{\mathrm{n}-1}\right]$ with $\mathrm{t}_{\mathrm{n}-1}$ as the node, ie., there are exactly 2 nodes on subinterval $\left[\mathrm{t}_{\mathrm{j} \mathrm{n}}, \mathrm{t}_{n}\right]$ so we get a fixed number 2 in the above formula.

## - The Trapezoidal Rule B:

We consider this alternative quadrature rule based above Rule A. It behaves better for non-smooth data ( see Thomee and Zhang [21]). This quadrature rule is also based on the second-order trapezoidal rule. Let $\overline{\mathrm{f}}_{\mathrm{j}}=\mathrm{j}^{2} \mathrm{k}$ and $\mathrm{j}_{\mathrm{n}}$ be the largest integer such that $\overline{\mathrm{f}}_{\mathrm{jn}}<$ $\mathrm{t}_{\mathrm{n}}$. We now write the interval of integration as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{j-1}, \bar{t}_{j}\right] \bigcup\left[\bar{t}_{j_{n}} t_{n-1}\right] \bigcup\left[t_{n-1}, t_{n}\right] \quad \text { if } j_{n} \geq 1 \tag{4.30}
\end{equation*}
$$

We shall apply the trapezoidal rule with variable stepsize $\left(\bar{t}_{\mathrm{j}+1}-\overline{\mathrm{t}}_{\mathrm{j}}\right)$ on intervals $\left[0, \overline{\mathrm{t}}_{\mathrm{j} n}\right]$. Next, on the remaining subinterval $\left[\mathrm{t}_{\mathrm{jn}}, \mathrm{t}_{\mathrm{n}}\right]$ let

$$
\begin{equation*}
\bar{m}_{1}=n-j_{n}^{2}-1, \quad \bar{k}_{1}=\bar{m}_{1} k \tag{4.31}
\end{equation*}
$$

In a similar manner to the discussion for rule A above, we apply once again the trapezoidal rule with stepsize $\mathrm{k}_{1}$ on $\left[\mathrm{t}_{\mathrm{j} \mathrm{n}}, \mathrm{t}_{\mathrm{n}-1}\right]$ and the rectangular rule with step size k on [ $\mathrm{t}_{\mathrm{n}-1}, \mathrm{t}_{\mathrm{n}}$ ]. So we use the trapezoidal rule for the approximation of the integral on each of the subintervals except the last, where we use the rectangular rule, with $\mathrm{t}_{\mathrm{n}-1}$ as the node. We call the method The Trapezoidal Rule B and the quadrature approximation is written by

$$
\begin{align*}
\sigma^{n}(\phi) & =\frac{1}{2} \sum_{j=1}^{j_{n}}\left(\bar{t}_{j}-\bar{t}_{j-1}\right)\left(\phi\left(\bar{t}_{j}\right)+\phi\left(\bar{t}_{j-1}\right)\right)+\frac{\bar{k}_{1}}{2}\left(\phi\left(\bar{t}_{j n}\right)+\phi\left(t_{n-1}\right)\right)+k \phi\left(t_{n-1}\right)  \tag{4.33}\\
& \equiv \sigma_{3}^{n}+\sigma_{2}^{n}+\sigma_{1}^{n}
\end{align*}
$$

The corresponding quadrature weights are

$$
\omega_{n, j}= \begin{cases}\frac{\bar{t}_{1}}{2} & j=0  \tag{4.34}\\ \frac{\left(\bar{t}_{j+1}-\bar{t}_{j-1}\right)}{2} & j=i^{2} \quad 1 \leq i \leq j_{n}-1 \\ \frac{\left(\bar{t}_{j}-\bar{t}_{j-1}\right)}{2}+\frac{\bar{k}_{1}}{2} & j=j_{n}^{2} \\ \frac{\bar{k}_{1}}{2}+k & j_{n}^{2}+\bar{m}_{1}=(n-1) \\ 0 & \text { otherwise }\end{cases}
$$

The upper bound of the storage for this rule is

$$
\mathrm{S}_{\max } \leq \mathrm{j}_{\mathrm{n}}+2=\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)
$$

## - The Simpson's Rule A:

This rule is based on the fourth-order Simpson's rule, which reduces the number of quadrature points even further so that the method requires only $\mathrm{O}\left(\mathrm{k}^{-1 / 4}\right)$ values of $\mathrm{u}^{\mathrm{n}}$ to be stored while the $O(k)$ convergence rate is retained. Let $m_{1}=\left[k^{-3 / 4}\right]$, now with $k_{1}=m_{1} k$ so that $\mathrm{k}_{1}=\mathrm{O}\left(\mathbf{k}^{1 / 4}\right)$. We denote $\overline{\mathrm{j}}_{\mathrm{j}}=\mathrm{j} \mathrm{k}_{1}, \mathrm{j}=0,1,2 \ldots$. For $\mathrm{n}=1,2$ we choose the rectangular rule $\sigma^{1}=\mathrm{k} \phi(0)$, and $\sigma^{2}=\mathrm{k}[\phi(0)+\phi(\mathrm{k})]$. When $\mathrm{n} \geq 3$, let $\mathrm{j}_{\mathrm{n}}$ be the largest integer such that $2 \mathrm{j}_{\mathrm{n}} \mathrm{k}_{1}<\mathrm{t}_{\mathrm{n}}$, and write

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{2 j-2,} \bar{t}_{2 j}\right] \bigcup\left[\bar{t}_{2 j_{\mu},} t_{n-1}\right] \bigcup\left[t_{n-1}, t_{n}\right] \quad \text { if } n>2 m \tag{4.35}
\end{equation*}
$$

Obviously, the length of the remaining interval $\left[\bar{t}_{2 j n}, \mathrm{t}_{\mathrm{n}}\right]$ is at most $2 \mathrm{k}_{1}$ and, like the earlier idea, we let

$$
\begin{equation*}
\bar{m}_{1}=\left[\frac{n-1-j_{n} m_{1}}{2}\right], \quad \bar{k}_{1}=\bar{m}_{1} k \quad([.] \text { denoting integral part }) \tag{4.40}
\end{equation*}
$$

Based on ideas given in Le Roux and Thomee [15] we propose minor changes as follows: We shall use Simpson's formula of order 4, on each of the first subintervals, on the second subintervals [ $\left[\mathrm{t}_{2 \mathrm{jn}}, \mathrm{t}_{\mathrm{n}-1}\right.$ ] we still use Simpson's formula except for the last, where we use the rectangular rule, with $\mathrm{t}_{\mathrm{n}-2}, \mathrm{t}_{\mathrm{n}-1}$ as the nodes if n is even, otherwise the rectangular rule with $\mathrm{t}_{\mathrm{n}-1}$ as the node.

We call the method The Simpson's Rule A. The quadrature approximation is written by

$$
\begin{align*}
\sigma^{n}(\phi)= & \frac{k_{1}}{3} \sum_{j=1}^{j_{n}}\left(\phi\left(\bar{t}_{2 j-2}\right)+4 \phi\left(\bar{t}_{2 j-1}+\phi\left(\bar{t}_{2 j}\right)\right)\right. \\
& +\frac{\bar{k}_{1}}{3}\left(\left(\phi\left(\bar{t}_{2 j_{n}}\right)+4 \phi\left(\bar{t}_{2 j_{n}+\bar{m} 1}\right)+\phi\left(t_{n-1}\right)\right)+k \phi\left(t_{n-1}\right)\right.  \tag{4.41}\\
& \equiv \sigma_{3}^{n}+\sigma_{2}^{n}+\sigma_{1}^{n}
\end{align*}
$$

The corresponding quadrature weights are

$$
\omega_{n, j}= \begin{cases}\frac{k_{1}}{3} & j=0  \tag{4.42}\\ \frac{4}{3} k_{1} & j=i m_{1} \quad i=\text { odd, } \quad 1 \leq i \leq 2 j_{n}-1 \\ \frac{2}{3} k_{1} & j=i m_{1} \quad i=\text { even } \quad 1 \leq i \leq 2 j_{n}-1 \\ \frac{1}{3} k_{1}+\frac{1}{3} \bar{k}_{1} & j=2 j_{n} m_{1} \\ \frac{4}{3} \bar{k}_{1} & j=2 j_{n} m_{1}+\bar{m}_{1} \\ \frac{1}{3} \bar{k}_{1}+k & j=2 j_{n} m_{1}+\bar{m}_{1}(=n-1) \quad \text { may }(j=n-2) \\ 0 & \text { otherwis }\end{cases}
$$

The storage requirement for this rule is

$$
\mathrm{S}_{\max } \leq 2 \mathrm{j}_{\mathrm{n}}+4=\mathrm{O}\left(\mathrm{k}^{-1 / 4}\right)
$$

In the following we consider two different quadrature formulas for second-order type schemes:

## - The Trapezoidal Rule I :

For a second-order type scheme the standard trapezoidal rule is the simplest quadrature rule that is consistent with the $\mathrm{O}\left(\mathrm{k}^{2}\right)$ accuracy of the Crank-Nicolson scheme. We write an interval of integration as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{n-1}\left[t_{j-1}, t_{j}\right] \bigcup\left[t_{n-1,}, t_{n-1 / 2}\right] \tag{4.43}
\end{equation*}
$$

We shall apply the standard trapezoidal rule with mesh-size k on $\left[0, \mathrm{t}_{\mathrm{n}-1}\right]$ and rectangular rule with mesh-size $\mathrm{k} / 2$ on $\left[\mathrm{t}_{\mathrm{n}-1}, \mathrm{t}_{\mathrm{n}-1 / 2}\right.$. The Trapezoidal Rule I which is due to Zhang [20] is defined by

$$
\begin{equation*}
\sigma^{n}(\phi)=\frac{k}{2} \sum_{j=1}^{n-1}\left(\phi\left(t_{j}\right)+\phi\left(t_{j-1}\right)\right)+\frac{k}{2} \phi\left(t_{n-1}\right) \equiv \sigma_{2}^{n}+\sigma_{1}^{n} \tag{4.44}
\end{equation*}
$$

The corresponding quadrature weights are

$$
\omega_{n, j}= \begin{cases}\frac{k}{2} & j=0  \tag{4.45}\\
k & \begin{array}{l}
1 \leq j=n-1 \\
\text { otherwise }
\end{array}\end{cases}
$$

The storage requirement for this rule is

$$
\mathrm{S}_{\max } \leq \mathrm{t}^{0} / \mathrm{k}=\mathrm{O}\left(\mathrm{k}^{-1}\right)
$$

## - The Simpson's Rule B:

The storage requirement for the Trapezoidal Rule I is $\mathrm{O}\left(\mathrm{k}^{-1}\right)$. As considered previously for the backward Euler type scheme, we may use a quadrature rule with higher order truncation error on fewer quadrature points to reduce the memory requirement without sacrificing the accuracy. We consider such a quadrature formula based on Simpson's rule.

More precisely, let $m_{1}=\left[k^{-1 / 2}\right]$ and $k_{1}=m_{1} k$. Setting $\bar{t}_{j}=j k_{1}, j=0,1,2, \ldots$, we let $j_{n}$ to be the largest nonnegative integer such that $2 \overline{\mathrm{t}}_{\mathrm{jn}}=2 \mathrm{j}_{\mathrm{n}} \mathrm{k}_{1}<\mathrm{t}_{\mathrm{n}}$. We also write the interval of integration as

$$
\begin{equation*}
\left[0, t_{n}\right]=\bigcup_{j=1}^{j_{n}}\left[\bar{t}_{2 j-2,} \bar{t}_{2 j}\right] \bigcup\left[\bar{t}_{2_{j-}-2,}, t_{n-1}\right] \bigcup\left[t_{n-1,}, t_{n-\frac{1}{2}}\right] \quad \text { if } n>2 m_{1} \tag{4.46}
\end{equation*}
$$

Following our approach for the Simpson's rule A we let

$$
\begin{equation*}
\bar{m}_{1}=\left[\frac{n-1-j_{n} m_{1}}{2}\right], \quad \bar{k}_{1}=\bar{m}_{1} k \quad([.] \text { denoting integral part }) \tag{4.47}
\end{equation*}
$$

On each of the first intervals we shall use Simpson's formula, which is of order 4, on the second $\left[\bar{t}_{2 j n}, \mathrm{t}_{\mathrm{n}-1}\right]$ we use once again Simpson's formula except in the last, where we use a composite of the trapezoidal and rectangular rules, with $\mathrm{t}_{\mathrm{n}-2}, \mathrm{t}_{\mathrm{n}-1}$ as the nodes if n is even, otherwise the rectangular rule with $\mathrm{t}_{\mathrm{n}-1}$ as the node.

We call method The Simpson's Rule B and the quadrature approximation is written by

$$
\begin{align*}
\sigma^{n}(\phi) & =\frac{k_{1}}{3} \sum_{j=1}^{j_{n}}\left(\phi\left(\bar{t}_{2 j-2}\right)+4 \phi\left(\bar{t}_{2 j-1}+\phi\left(\bar{t}_{2 j}\right)\right)\right. \\
& +\frac{\bar{k}_{1}}{3}\left(\left(\phi\left(\bar{t}_{2 j_{n}}\right)+4 \phi\left(\bar{t}_{2 j_{n}+\bar{m} 1}\right)+\phi\left(t_{n-1}\right)\right)+\frac{k}{2} \phi\left(t_{n-1}\right)\right.  \tag{4.48}\\
& \equiv \sigma_{3}^{n}+\sigma_{2}^{n}+\sigma_{1}^{n}
\end{align*}
$$

The corresponding quadrature weights are

$$
\omega_{n, j}= \begin{cases}\frac{k_{1}}{3} & j=0  \tag{4.49}\\ \frac{4}{3} k_{1} & j=i m_{1} \quad i=o d d, \quad 1 \leq i \leq 2 j_{n}-1 \\ \frac{2}{3} k_{1} & j=i m_{1} \quad i=\text { even } \quad 1 \leq i \leq 2 j_{n}-1 \\ \frac{1}{3} k_{1}+\frac{1}{3} \bar{k}_{1} & j=2 j_{n} m_{1} \\ \frac{4}{3} \bar{k}_{1} & j=2 j_{n} m_{1}+\bar{m}_{1} \\ \frac{1}{3} \bar{k}_{1}+k & 2 j_{n} m_{1}+\bar{m}_{1}<j \leq n-1 \\ 0 & \text { otherwise }\end{cases}
$$

The storage requirement for this rule is

$$
\mathrm{S}_{\max } \leq \mathrm{j}_{\mathrm{n}}+4=\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)
$$

So far we have considered six kind of quadrature rules which suit problem (4.1). We obviously need to consider also an algorithmic implementation of the solution methods
and we examine them using some test problems in the next chapter.

## Chapter 5

## Parabolic Integro-Differential Equations II

### 5.1 Introduction

In this chapter we give some implementation details for those modified solution methods which were described previously. The numerical experiments together with some results analysis are also given.

We discuss the algorithmic implementation of Modified Method II (routine PIDECX), giving its essential details. For comparison purposes we also implement Modified Method I described in Zhang [20] (routine PIDETZ). We measure the efficiency of both solvers in terms of accuracy, storage requirement and CPU time.

### 5.2 Computer Implementation

We discuss some algorithmic implementation aspects for our solver, which was written in the FORTRAN 77 programming language, together with a main structure chart of the code and the structure charts of the principal improved quadrature rules.

As described before, we subdivide the x -t plane into sets of equal rectangles of side $\Delta \mathrm{x}$ $=\mathrm{h}, \Delta \mathrm{t}=\mathrm{k}$, and let the co-ordinates $\left(\mathrm{x}_{\mathrm{i}}, \mathrm{t}_{\mathrm{j}}\right)$ of the representative mesh point be

$$
x_{1}=\mathrm{ih}, \mathrm{t}_{\mathrm{j}}=\mathrm{jk}, \quad \mathrm{i}=0,1, \ldots, \mathrm{M} ; \quad \mathrm{n}=0,1, \ldots, \mathrm{~N},
$$

where $\mathrm{i}, \mathrm{n}, \mathrm{M}$ and N are integers, $\mathrm{h}=1 / \mathrm{M}$ and $\mathrm{k}=\mathrm{t}^{0} / \mathrm{N}$.

We are concerned with a semidiscretization method for approximating the numerical solution $U^{\mathrm{n}}$ of problem (4.1) based on the $\theta$-scheme,

$$
\begin{equation*}
\left(I-\theta r \delta_{x}^{2}\right) u_{n}=\left(I+(1-\theta) r \delta_{x}^{2}\right) u_{n-1}+k\left[\sigma_{n}\left(B\left(\theta t_{n}+(1-\theta) t_{n-1}\right) u\right)+f\left(\theta t_{n}+(1-\theta) t_{n-1}\right)\right] \tag{5.1}
\end{equation*}
$$

where $I$ is the identity operator, and $\delta_{\mathrm{x}}{ }^{2}$ is the second order central difference operator with respect to the space variable x , and

$$
\begin{gather*}
\delta_{x}^{2} u_{i}=u_{i-1}-2 u_{i}+u_{i+1}, \\
r=\frac{k}{h^{2}},  \tag{5.2}\\
\sigma_{n}\left(B\left(\theta t_{n}+(1-\theta) t_{n-1}\right) u\right)=\sum_{j=0}^{n-1} \omega_{n j} B\left(\theta t_{n}+(1-\theta) t_{n-1}, t_{j}\right) u_{j} .
\end{gather*}
$$

The quadrature weights $\left\{\omega_{\mathrm{nj}}\right\}$ are chosen so that for $\varphi \in \mathrm{C}\left[0, \mathrm{t}^{3}\right]$ and $\varphi_{\mathrm{j}}=\varphi\left(\mathrm{t}_{\mathrm{j}}\right)$ we have

$$
\begin{equation*}
\sigma_{n}(\varphi)=\sum_{j=0}^{n-1} \omega_{n j} \varphi_{j} \approx \int_{0}^{\left.\theta r_{n}^{\prime}+(1-\theta)\right)_{n-1}} \varphi(s) d s . \tag{5.3}
\end{equation*}
$$

The approximation leads to

$$
\begin{equation*}
C_{\theta} U^{n}=D_{\theta} U^{n-1}+F_{\theta}^{n} \tag{5.4}
\end{equation*}
$$

where

$$
\begin{align*}
& C U^{n}=\left[\begin{array}{ccccccc}
1+2 \theta r & -\theta r & & & & \\
-\theta r & 1+2 \theta r & -\theta r & & & & \\
& -\theta r & 1+2 \theta r & -\theta r & & & \\
& & \cdot & \cdot & . & & \\
& & & & & & \\
& & & & & & \\
& & & & -\theta r & 1+2 \theta r & -\theta r \\
& & & & & -\theta r & 1+2 \theta r
\end{array}\right]\left[\begin{array}{c}
u_{1}^{n} \\
u_{2}^{n} \\
u_{3}^{n} \\
\cdot \\
\cdot \\
u_{M-2}^{n} \\
u_{M-1}^{n}
\end{array}\right]  \tag{5.5}\\
& D U^{n-1}=\left[\begin{array}{ccccc}
1-2(1-\theta) r & (1-\theta) r & & & \\
(1-\theta) r & 1-2(1-\theta) r & (1-\theta) r & & \\
& \cdot & \cdot & . & \\
& & 1-2(1-\theta) r & (1-\theta) r \\
& & & 1-2(1-\theta) r
\end{array}\right]\left[\begin{array}{c}
u_{1}^{n-1} \\
u_{2}^{n-1} \\
u_{3}^{n-1} \\
\cdot \\
\vdots \\
u_{M-1}^{n-1}
\end{array}\right] \tag{5.6}
\end{align*}
$$

and

$$
\begin{equation*}
F^{n}=k\left\{\sigma^{n}\left(B\left(\theta t_{n}+(1-\theta) t_{n-1}\right)+f\left(\theta t_{n}+(1-\theta) t_{n-1}\right)\right\}\right. \tag{5.7}
\end{equation*}
$$

It is easier to see that the following two well-known schemes can be obtained when we take the particular value $\theta$ in (5.1).
(a) the Crank-Nicolson scheme $(\theta=\mathbf{1 / 2})$
(b) the Backward-Euler scheme $(\theta=1)$

We therefore may write the main ingredients of implementation for the solution methods described in chapter 4 as follows:

- A set of initial parameters, such as a numerical method flag, initial value and
boundary value, a step size, an error estimate etc..
- A main program is used to set the initial parameters mentioned above and to call the subroutines needed to form the coefficient matrix $C$, and evaluate the right-hand side $D U^{n-1}$ and $f$ which are necessary to generate the finite difference approximations for problem (4.1) (based on the B.E. and C.N. schemes with a suitable quadrature rule).
- The set of subroutines which prepare the main strategies to accept the approximation are

1) the MAXA subroutine with DEC subroutine which generates the elements of the coefficient matrix C in (5.4) and forms its LU decomposition.
2) The driver subroutine, XINTEG, which makes calls to the main subroutines, TINT, FORMB and SOL. TINT implements the various quadrature formulas. FORMB defines and evaluates the right-hand side of (5.7). The SOL subroutine performs back-substitution on the right hand side (RHS) vector to obtain the numerical solution.

The overall structure of our solver is illustrated in the main structure chart of Figure 1. To give more detail for the code we present several structure charts of the principal modified quadrature rules in Figures 2, 3, 4 and 5, respectively.







Figure 4


### 5.3 Test Problems

In order to examine our solver we consider two initial-boundary value problems. We take the two examples below as test problems for the numerical experiments.

## Problem 1

We consider the initial boundary value problem

$$
\begin{array}{rc}
u_{t}=u_{x x}+\int_{0}^{t} B(x ; t, s) u(x, s) d s+f(x, t) & (x, t) \in[0,1] \quad\left(0, t^{0}\right], \\
u(0, t)=u(1, t)=0 & t \in\left(0, t^{0}\right], \\
u(x, 0)=u_{0}(x), & x \in[0,1] .
\end{array}
$$

where

$$
\begin{gather*}
B(x ; t, s)=e^{-\pi^{2}(t-s)},  \tag{5.8}\\
f(x, t)=-t e^{-\pi^{2} t} \sin \pi x \\
u_{0}(x)=\sin \pi x .
\end{gather*}
$$

The exact solution is

$$
u(x, t)=e^{-\pi^{2} t} \sin \pi x .
$$

Source: N.Y. Zhang [20]

## Problem 2

$$
\begin{align*}
& \qquad \begin{array}{c}
u_{t}=u_{x x}+\int_{0}^{t} B(x, t, s) u(x, s) d s+f(x, t) \\
u(0, t)=u(1, t)=0 \\
u(x, 0)=0 \\
\text { where } \quad B(x ; t, s)=e^{-\pi^{2}(t-s)} \\
f(x, t)=\sin (\pi x)\left(\pi^{2}-\frac{1}{\pi^{2}}+\frac{e^{-\pi^{2} t}}{\pi^{2}}+t e^{-\pi^{2} t}\right)
\end{array}
\end{align*}
$$

The exact solution is

$$
u(x, t)=\sin (\pi x)\left(1-e^{-\pi^{2} t}\right)
$$

Source: constructed example

### 5.4 Efficiency Considerations

In the previous chapter, we described some schemes for partial integro-differential equations of parabolic type, with emphasis on quadrature rules for the integral term. The computer implementation was introduced in the last section. Furthermore, the effectiveness of the improved algorithms Modified Method II and I will be examined and assessed in the routine PIDECX and PIDETZ.

In the following experiments, effectiveness is measured in terms of accuracy and efficiency.

The accuracy of each algorithm is illustrated in two ways. Firstly, the maximum absolute
component of error between the numerical and the exact solutions over the integration interval, ie., the error $U_{n}-u\left(t_{n}\right)$, is measured with respect to the $l_{\infty}$ norm. Secondly, the numerical estimation of the order of convergence based on the error bound of the form

$$
\begin{equation*}
\left|u-u^{h}\right|<C\left(h^{P}+k^{q}\right) \tag{5.10}
\end{equation*}
$$

is included in the tables. We use the double-mesh technique to calculate the order of convergence. To summarize the possibilities briefly, the calculation of the convergence rate is in time only, ie., to estimate q.

The efficiency of each algorithm may also be expressed in two ways. The first is the CPU time consumption which provides an excellent means of determining the relative efficiency of several algorithms carried out on the same platform, a VAX 11/6230. The CPU time consumption, measured in seconds, was presented for each implementation. The second is $S_{\max }$, which is the maximal number of time levels of the solution stored during the calculation. These are particularly useful since they are important features of the proposed methods that reduce greatly both the memory requirements and computational effort.

### 5.5 Numerical Experiments

In this section we present some numerical experiments to illustrate the application aspects of the proposed numerical methods to approximate the solutions of partial integrodifferential equations of parabolic type.

We classified the numerical experiments into two groups according to different strategies in choosing the stepsize.

- The first group of numerical experiments: For the B.E. scheme with error of order $\mathrm{O}\left(\mathrm{h}^{2}+\mathrm{k}\right)$, the stepsize of spatial discretizations and the time discretization will be chosen so as to maintain $k=h^{2}$ to display a convergence rate of $\mathrm{O}\left(\mathrm{k}^{2}\right)$. For the C.N. scheme with error of order $O\left(h^{2}+k^{2}\right)$, the stepsize will be $k=h$ for the same reason.
- The second group of numerical experiments: The spatial stepsize $h$ will be chosen such that $h<k$, in order to estimate the temporal rate of convergence by only changing the time stepsize $k$.

Following the analysis of Sloan and Thomee [17], we take two particular cases: B.E. scheme $(\theta=1)$ and C.N. scheme $(\theta=0.5)$ in (5.1). For each example in these two groups of experiments, we display the error $U_{n}-u\left(t_{n}\right)$ with respect to the $l_{\infty}$ norms, the storage requirement in terms of an upper bound $\mathrm{S}_{\text {max }}$, the maximal number of time levels of the solution stored during the calculation, the rate of convergence and the CPU time needed for the calculation. In a similar way, the results of numerical experiments presented for the comparison routine PIDETZ make use of Modified Method I. The two test problems were mentioned in section 5.3 where the exact solution is available.

The following is a summary of the "figures of merit" used in the tables.

METH the six quadrature rules for Modified Method II

| METH | DESCRIPTION |
| :---: | :---: |
| 1 | RECTANGULAR Rule |
| 2 | Trapezoidal Rule A |
| 3 | Trapezoidal Rule B |
| 4 | Simpson's Rule A |
| 5 | Trapezoidal Rule I |
| 6 | Simpson's Rule B |

$\theta \quad$ the value of $\theta$ for the $\theta$ scheme
NPTS the number of spatial mesh points used
$\mathrm{S}_{\mathrm{MAX}}$ the maximal number of time levels of the solution stored during the calculation
$\Delta \mathrm{t}, \mathrm{k} \quad$ the temporal step-size
$\operatorname{ERR}\left(1_{\infty}\right) \quad$ the error $U_{n}-u\left(t_{n}\right)$ with respect to the $l_{\infty}$ norm
RATE the rate of convergence
CPU the CPU time (in seconds) required for the solution. All experiments were carried out on a VAX 11/6230.

Table 1
Problem 1 The first group of numerical experiment $\mathbf{t}^{0}=0.1 \quad$ Modified Method II

| METH | $\theta$ | NPTS | $\Delta \mathrm{t}(\mathrm{k})$ | ERR( $\left.1_{m}\right)$ | RATE | Smax | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 21 | $.250 \mathrm{D}-02$ | $.525 \mathrm{D}-02$ | . | 40 | 2.04 |
| 2 | 1 | 21 | $.250 \mathrm{D}-02$ | $.525 \mathrm{D}-02$ | . | 3 | 0.73 |
| 3 | 1 | 21 | $.250 \mathrm{D}-02$ | $.525 \mathrm{D}-02$ |  | 8 | 1.3 |
| 4 | 1 | 21 | $.250 \mathrm{D}-02$ | $.525 \mathrm{D}-02$ |  | - | 4 |
| 5 | 0.5 | 21 | $.500 \mathrm{D}-01$ | $.689 \mathrm{D}-02$ | - | 2 | 0.81 |
| 6 | 0.5 | 21 | $.500 \mathrm{D}-01$ | $.689 \mathrm{D}-02$ | - | 2 | 0.02 |
| 1 | 1 | 41 | $.625 \mathrm{D}-03$ | $.132 \mathrm{D}-02$ | 2 | 160 | 0.03 |
| 2 | 1 | 41 | $.625 \mathrm{D}-03$ | $.132 \mathrm{D}-02$ | 2 | 5 | 55.87 |
| 3 | 1 | 41 | $.625 \mathrm{D}-03$ | $.132 \mathrm{D}-02$ | 2 | 14 | 15.61 |
| 4 | 1 | 41 | $.625 \mathrm{D}-03$ | $.132 \mathrm{D}-02$ | 2 | 4 | 6.8 |
| 5 | 0.5 | 41 | $.250 \mathrm{D}-01$ | $.169 \mathrm{D}-02$ | 2 | 4 | 0.17 |
| 6 | 0.5 | 41 | $.250 \mathrm{D}-01$ | $.169 \mathrm{D}-02$ | 2 | 4 | 0.15 |
| 1 | 1 | 81 | $.156 \mathrm{D}-03$ | $.331 \mathrm{D}-03$ | 2 | 640 | 1679.91 |
| 2 | 1 | 81 | $.156 \mathrm{D}-03$ | $.331 \mathrm{D}-03$ | 2 | 9 | 80.05 |
| 3 | 1 | 81 | $.156 \mathrm{D}-03$ | $.331 \mathrm{D}-03$ | 2 | 27 | 214.7 |
| 4 | 1 | 81 | $.156 \mathrm{D}-03$ | $.331 \mathrm{D}-03$ | 2 | 4 | 51.83 |
| 5 | 0.5 | 81 | $.125 \mathrm{D}-01$ | $.421 \mathrm{D}-03$ | 2 | 8 | 0.88 |
| 6 | 0.5 | 81 | $.125 \mathrm{D}-01$ | $.421 \mathrm{D}-03$ | 2 | 4 | 0.63 |

Problem 1 The second group of numerical experiment $\rho^{\circ}=1 \quad$ Modified Method II

| METH | $\theta$ | NPTS | $\Delta t(k)$ | ERR( $\mathrm{l}_{\mathrm{m}}$ ) | RATE | Smax | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 501 | .250D-01 | .412D-01 | - | 40 | 52.2 |
| 2 | 1 | 501 | .250D-01 | .413D-01 | - | 8 | 28.2 |
| 3 | 1 | 501 | .250D-01 | .413D-01 | - | 8 | 31.8 |
| 4 | 1 | 501 | .250D-01 | .413D-01 | - | 8 | 24.5 |
| 5 | 0.5 | 501 | .250D-01 | .188D-02 | - | 40 | 90.2 |
| 6 | 0.5 | 501 | .250D-01 | .188D-02 | - | 10 | 27.8 |
| 1 | 1 | 501 | .125D-01 | .216D-01 | 0.9 | 80 | 185.5 |
| 2 | 1 | 501 | .125D-01 | .216D-01 | 0.9 | 11 | 70.5 |
| 3 | 1 | 501 | .125D-01 | .216D-01 | 0.9 | 10 | 78.4 |
| 4 | 1 | 501 | .125D-01 | .216D-01 | 0.9 | 12 | 64.3 |
| 5 | 0.5 | 501 | .125D-01 | .467D-03 | 2 | 80 | 344.9 |
| 6 | 0.5 | 501 | .125D-01 | .467D-03 | 2 | 12 | 66.7 |
| 1 | 1 | 501 | .625D-02 | .111D-01 | 1 | 160 | 696 |
| 2 | 1 | 501 | .625D-02 | .111D-01 | 1 | 15 | 169.1 |
| 3 | 1 | 501 | .625D-02 | .111D-01 | 1 | 14 | 197.2 |
| 4 | 1 | 501 | .625D-02 | .111D-01 | 1 | 8 | 105.9 |
| 5 | 0.5 | 501 | .625D-02 | .116D-03 | 2 | 160 | 1353.3 |
| 6 | 0.5 | 501 | .625D-02 | .116D-03 | 2 | 16 | 153.9 |

Table 2
Problem 2 The first group of numerical experiment $t^{0}=0.1 \quad$ Modified Method II

| METH | $\theta$ | NPTS | $\Delta t(k)$ | ERR( $\left.1^{\circ}\right)$ | RATE | $S_{\text {max }}$ | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 21 | $.250 \mathrm{D}-02$ | $.403 \mathrm{D}-02$ |  | 40 | 2 |
| 2 | 1 | 21 | $.250 \mathrm{D}-02$ | $.398 \mathrm{D}-02$ |  | 3 | 0.76 |
| 3 | 1 | 21 | $.250 \mathrm{D}-02$ | $.400 \mathrm{D}-02$ | - | 8 | 1.21 |
| 4 | 1 | 21 | $.250 \mathrm{D}-02$ | $.401 \mathrm{D}-02$ | - | 4 | 0.79 |
| 5 | 0.5 | 21 | $.500 \mathrm{D}-01$ | $.807 \mathrm{D}-02$ | - | 2 | 0.03 |
| 6 | 0.5 | 21 | $.500 \mathrm{D}-01$ | $.807 \mathrm{D}-02$ | - | 2 | 0.03 |
| 1 | 1 | 41 | $.625 \mathrm{D}-03$ | $.102 \mathrm{D}-02$ | 2 | 160 | 54.01 |
| 2 | 1 | 41 | $.625 \mathrm{D}-03$ | $.100 \mathrm{D}-02$ | 2 | 5 | 7.23 |
| 3 | 1 | 41 | $.625 \mathrm{D}-03$ | $.101 \mathrm{D}-02$ | 2 | 14 | 15.45 |
| 4 | 1 | 41 | $.625 \mathrm{D}-03$ | $.101 \mathrm{D}-02$ | 2 | 4 | 6.24 |
| 5 | 0.5 | 41 | $.250 \mathrm{D}-01$ | $.198 \mathrm{D}-02$ | 2 | 4 | 0.15 |
| 6 | 0.5 | 41 | $.250 \mathrm{D}-01$ | $.197 \mathrm{D}-02$ | 2 | 4 | 0.15 |
| 1 | 1 | 81 | $.156 \mathrm{D}-03$ | $.255 \mathrm{D}-03$ | 2 | 640 | 1671.52 |
| 2 | 1 | 81 | $.156 \mathrm{D}-03$ | $.251 \mathrm{D}-03$ | 2 | 9 | 78.91 |
| 3 | 1 | 81 | $.156 \mathrm{D}-03$ | $.253 \mathrm{D}-03$ | 2 | 27 | 211.46 |
| 4 | 1 | 81 | $.156 \mathrm{D}-03$ | $.253 \mathrm{D}-03$ | 2 | 4 | 50.95 |
| 5 | 0.5 | 81 | $.125 \mathrm{D}-01$ | $.493 \mathrm{D}-03$ | 2 | 8 | 0.83 |
| 6 | 0.5 | 81 | $.125 \mathrm{D}-01$ | $.490 \mathrm{D}-03$ | 2 | 4 | 0.62 |

Problem 2 The second group of numerical experiment $\mathrm{t}^{\circ}=1 \quad$ Modified Method II

| METH | $\theta$ | NPTS | $\Delta \mathrm{t}$ | ERR $^{\infty}$ | RATE | $\mathrm{S}_{\text {ma }}$ | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 501 | $.250 \mathrm{D}-01$ | $.416 \mathrm{D}-01$ |  | 40 | 51.9 |
| 2 | 1 | 501 | $.250 \mathrm{D}-01$ | $.414 \mathrm{D}-01$ |  | 8 | 28.2 |
| 3 | 1 | 501 | $.250 \mathrm{D}-01$ | $.414 \mathrm{D}-01$ |  | - | 8 |
| 4 | 1 | 501 | $.250 \mathrm{D}-01$ | $.415 \mathrm{D}-01$ |  | 81.8 |  |
| 5 | 0.5 | 501 | $.250 \mathrm{D}-01$ | $.185 \mathrm{D}-02$ |  | - | 40 |
| 6 | 0.5 | 501 | $.250 \mathrm{D}-01$ | $.184 \mathrm{D}-02$ |  | 10 | 90.7 |
| 1 | 1 | 501 | $.125 \mathrm{D}-01$ | $.218 \mathrm{D}-01$ | 0.9 | 80 | 183.8 |
| 2 | 1 | 501 | $.125 \mathrm{D}-01$ | $.216 \mathrm{D}-01$ | 0.9 | 11 | 70.2 |
| 3 | 1 | 501 | $.125 \mathrm{D}-01$ | $.216 \mathrm{D}-01$ | 0.9 | 10 | 78.1 |
| 4 | 1 | 501 | $.125 \mathrm{D}-01$ | $.217 \mathrm{D}-01$ | 0.9 | 12 | 64.1 |
| 5 | 0.5 | 501 | $.125 \mathrm{D}-01$ | $.460 \mathrm{D}-03$ | 2 | 80 | 343.3 |
| 6 | 0.5 | 501 | $.125 \mathrm{D}-01$ | $.458 \mathrm{D}-03$ | 2 | 12 | 67.2 |
| 1 | 1 | 501 | $.625 \mathrm{D}-02$ | $.112 \mathrm{D}-01$ | 1 | 160 | 690.4 |
| 2 | 1 | 501 | $.625 \mathrm{D}-02$ | $.110 \mathrm{D}-01$ | 1 | 15 | 168.2 |
| 3 | 1 | 501 | $.625 \mathrm{D}-02$ | $.111 \mathrm{D}-01$ | 1 | 14 | 197.6 |
| 4 | 1 | 501 | $.625 \mathrm{D}-02$ | $.111 \mathrm{D}-01$ | 1 | 8 | 105.3 |
| 5 | 0.5 | 501 | $.625 \mathrm{D}-02$ | $.116 \mathrm{D}-03$ | 2 | 160 | 1341.8 |
| 6 | 0.5 | 501 | $.625 \mathrm{D}-02$ | $.115 \mathrm{D}-03$ | 2 | 16 | 154.8 |

Table 3
The second group of numerical experiment for Problem $1 \quad \mathbf{t}^{\boldsymbol{0}}=1$
Modified Method II

| METH | $\theta$ | NPTs | $\Delta \mathbf{\Delta t}(\mathbf{k})$ | ERR( $\left.\mathbf{l}_{\infty}\right)$ | RATE | $\mathbf{S}_{\text {max }}$ | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 501 | $.25 \mathrm{D}-1$ | $.412 \mathrm{D}-1$ | - | 40 | 52.2 |
| 2 | 1 | 501 | $.25 \mathrm{D}-1$ | $.413 \mathrm{D}-1$ | - | 8 | 28.2 |
| 3 | 1 | 501 | $.25 \mathrm{D}-1$ | $.413 \mathrm{D}-1$ | - | 8 | 31.8 |
| 4 | 1 | 501 | $.25 \mathrm{D}-1$ | $.413 \mathrm{D}-1$ | - | 8 | 24.5 |
| 1 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-1$ | 0.9 | 80 | 185.5 |
| 2 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-1$ | 0.9 | 11 | 70.5 |
| 3 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-1$ | 0.9 | 10 | 78.4 |
| 4 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-1$ | 0.9 | 12 | 64.3 |
| 1 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-1$ | 1 | 160 | 696 |
| 2 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-1$ | 1 | 15 | 169.1 |
| 3 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-1$ | 1 | 14 | 197.2 |
| 4 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-1$ | 1 | 8 | 105.9 |

The second group of numerical experiment for Problem $1 \quad t^{0}=1$
Modified Method I

| METH | $\theta$ | NPTs | $\Delta \mathbf{t}(\mathbf{k})$ | ERR(l $\left.{ }_{\infty}\right)$ | RATE | $\mathbf{S}_{\text {max }}$ | CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 1 | 501 | $.25 \mathrm{D}-1$ | $.412 \mathrm{D}-01$ | - | 40 | 52.3 |
| 2 | 1 | 501 | $.25 \mathrm{D}-1$ | $.412 \mathrm{D}-01$ | - | 10 | 29.4 |
| 3 | 1 | 501 | $.25 \mathrm{D}-1$ | $.413 \mathrm{D}-01$ | - | 10 | 34.6 |
| 4 | 1 | 501 | $.25 \mathrm{D}-1$ | $.413 \mathrm{D}-01$ |  | 8 | 30.7 |
| 1 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-01$ | 0.9 | 80 | 182 |
| 2 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-01$ | 0.9 | 17 | 76 |
| 3 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-01$ | 0.9 | 24 | 91.8 |
| 4 | 1 | 501 | $.125 \mathrm{D}-1$ | $.216 \mathrm{D}-01$ | 0.9 | 14 | 79 |
| 1 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-01$ | 1 | 160 | 688.6 |
| 2 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-01$ | 1 | 17 | 197.9 |
| 3 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-01$ | 1 | 28 | 245.7 |
| 4 | 1 | 501 | $.625 \mathrm{D}-2$ | $.111 \mathrm{D}-01$ | 1 | 12 | 160.2 |

Table 4

| The second group of numerical experiment for Problem 2 <br> Modified Method II |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |
| METH | $\theta$ | NPTs | $\Delta t$ | ERR( $\mathbf{l}_{\infty}$ ) | RATE | $\mathbf{S}_{\text {max }}$ | CPU |
| 1 | 1 | 501 | .25D-1 | .416D-1 | - | 40 | 51.9 |
| 2 | 1 | 501 | .25D-1 | .414D-1 | - | 8 | 28.2 |
| 3 | 1 | 501 | .25D-1 | .414D-1 | - | 8 | 31.8 |
| 4 | 1 | 501 | .25D-1 | .415D-1 | - | 8 | 24.5 |
| 1 | 1 | 501 | .125D-1 | .218D-1 | 0.9 | 80 | 183.8 |
| 2 | 1 | 501 | .125D-1 | .216D-1 | 0.9 | 11 | 70.2 |
| 3 | 1 | 501 | .125D-1 | .216D-1 | 0.9 | 10 | 78.1 |
| 4 | 1 | 501 | .125D-1 | .217D-1 | 0.9 | 12 | 64.1 |
| 1 | 1 | 501 | .625D-2 | .112D-1 | 1 | 160 | 690.4 |
| 2 | 1 | 501 | .625D-2 | .110D-1 | 1 | 15 | 168.2 |
| 3 | 1 | 501 | .625D-2 | .111D-1 | 1 | 14 | 197.6 |
| 4 | 1 | 501 | .625D-2 | .111D-1 | 1 | 8 | 105.3 |
| The second group of numerical experiment for Problem 2 $\quad \mathbf{t}^{\mathbf{0}=1}$ |  |  |  |  |  |  |  |
| METH | $\theta$ | NPTs | $\Delta t$ | ERR(1) ${ }_{\text {c }}$ ) | RATE | $S_{\text {max }}$ | CPU |
| 1 | 1 | 501 | .25D-1 | .416D-01 | - | 40 | 51.7 |
| 2 | 1 | 501 | .25D-1 | .416D-01 | - | 10 | 29.3 |
| 3 | 1 | 501 | .25D-1 | .415D-01 | - | 10 | 35.3 |
| 4 | 1 | 501 | .25D-1 | .415D-01 | - | 8 | 30.6 |
| 1 | 1 | 501 | .125D-1 | .218D-01 | 0.9 | 80 | 181.6 |
| 2 | 1 | 501 | .125D-1 | .218D-01 | 0.9 | 17 | 77.1 |
| 3 | 1 | 501 | .125D-1 | .217D-01 | 0.9 | 24 | 92.3 |
| 4 | 1 | 501 | .125D-1 | .217D-01 | 0.9 | 14 | 79.3 |
| 1 | 1 | 501 | .625D-2 | .112D-01 | 1 | 160 | 688.7 |
| 2 | 1 | 501 | .625D-2 | .111D-01 | 1 | 17 | 198.3 |
| 3 | 1 | 501 | .625D-2 | .111D-01 | 1 | 28 | 245.8 |
| 4 | 1 | 501 | .625D-2 | .111D-01 | 1 | 12 | 159.9 |

### 5.6 Analysis of Results

Our work has involved writing two codes, PIDECX and PIDETZ, designed as easy-to use and robust methods for solving the partial integro-differential equations of parabolic type. The routine PIDECX makes use of Modified Method II, which includes four different quadrature rules that are consistent with the $\mathrm{O}(\mathrm{k})$ accuracy of the BackwardEuler scheme and two quadrature rules that are consistent with $\mathrm{O}\left(\mathrm{k}^{2}\right)$ accuracy of the Crank-Nicolson scheme. The PIDETZ makes use of Modified Method I which includes four different quadrature rules that are consistent with the $\mathrm{O}(\mathrm{k})$ accuracy of the Backward-Euler scheme. The results of some experiments presented in the section 5.5 , allow us to conclude that the proposed algorithms are competitive, and produce very significant reduction in both the storage and computation requirements.

Table 1 show the results obtained using Modified Method II for test Problem 1. The upper part of Table 1 gives the statistics for the first group of numerical experiments choosing $k=h^{2}, t^{0}=0.1$. The lower part of the table presents solutions obtained for the second group of numerical experiments, making the spatial stepsize $\mathrm{h}=1 / 500$ and only changing the time stepsize k and set $\mathrm{t}^{0}=1$.

For both cases in Table 1, it is clear that those improved rules show the expected savings in both storage and execution time with no loss in accuracy. The methods of first order accurate, METH 1 to 4, display a convergence rate $O(k)$. METH (5-6), of second-order accuracy, show a convergence rate $\mathrm{O}\left(\mathrm{k}^{2}\right)$. The standard method, Rectangular Rule, METH (1), has a storage requirement $\mathrm{O}\left(\mathrm{k}^{-1}\right)$. Trapezoidal Rules A and B METH (2 and 3) and Simpson's Rule A, METH (4) have storage requirements which are $\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)$ and $\mathrm{O}\left(\mathrm{k}^{-1 / 4}\right)$ respectively. Comparing Trapezoidal Rule I, METH (5), with Simpson's Rule B, METH (6), we have found that the storage requirement for METH (5) is $\mathrm{O}\left(\mathrm{k}^{-1}\right)$ but the Simpson's Rule B METH (6) is only $\mathrm{O}\left(\mathrm{k}^{-1 / 2}\right)$, without sacrificing the overall order of convergence. Obviously, this corresponds to significant savings in CPU time consumption. In a similar way, Table 2 shows the solutions obtained using Modified Method II for test Problem 2. All results have the same characteristics as those in Table 1.

Table 3 shows the statistics for results obtained using both Modified Method II and I
for test Problem 1. The upper part of the table gives the solution obtained with Modified Method II and the lower part of the table presents solutions with Modified Method I. To summarize the possibilities briefly, we present only some solutions obtained for METH (1-4) of first-order accuracy in Tables 3 and 4.

As expected, from the lower part of Table 3, we have found that the comparison solver, Modified Method I, has been shown to be very robust and efficient in terms of computational expense. However Modified Method II in the upper table is more efficient than the previous solver particularly in storage requirements when solving the same test problem. In a similar way, Table 4 shows the solution obtained using Modified Method II for test Problem 2. All results have the same characteristics as those in Table 3.

The results obtained using Modified Method II and Modified Method I to solve Problems 1 and 2 are summarised in terms of storage requirements and CPU time consumption in Tables 5 and 6 respectively, which are displayed below.

Table 5

| Problem $1 \quad \mathbf{t}^{\mathbf{0}}=\mathbf{1}$ <br> Modified Method II |  |  |  |
| :---: | :---: | :---: | :---: |
| $h=1 / 500, \quad k=1 / 40,1 / 80$ | $k=1 / 40,1 / 80$ and 1/160 | $\mathbf{t}^{0}=1$ |  |
| METH | $\Delta t(k)$ | $S_{\text {max }}$ | CPU |
| Rectangular Rule | .25D-1 | 40 | 52.2 |
| Trapezoidal Rule A |  | 8 | 28.2 |
| Trapezoidal Rule B |  | 8 | 31.8 |
| Simpson's Rule A |  | 8 | 24.5 |
| Rectangular Rule | .125D-1 | 80 | 185.5 |
| Trapezoidal Rule A |  | 11 | 70.5 |
| Trapezoidal Rule B |  | 10 | 78.4 |
| Simpson's Rule A |  | 12 | 64.3 |
| Rectangular Rule | .625D-2 | 160 | 696 |
| Trapezoidal Rule A |  | 15 | 169.1 |
| Trapezoidal Rule B |  | 14 | 197.2 |
| Simpson's Rule A |  | 8 | 105.9 |
| Modified Method I |  |  |  |
| METH | $\Delta t(k)$ | $S_{\text {max }}$ | CPU |
| Rectangular Rule | .25D-1 | 40 | 52.3 |
| Modified Trapezoidal I |  | 10 | 29.4 |
| Modified Trapezoidal II |  | 10 | 34.6 |
| Modified Simpson's I |  | 8 | 30.7 |
| Rectangular Rule | .125D-1 | 80 | 182 |
| Modified Trapezoidal I |  | 17 | 76 |
| Modified Trapezoidal II |  | 24 | 91.8 |
| Modified Simpson's I |  | 14 | 79 |
| Rectangular Rule | . $625 \mathrm{D}-2$ | 160 | 688.6 |
| Modified Trapezoidal I |  | 17 | 197.9 |
| Modified Trapezoidal II |  | 28 | 245.7 |
| Modified Simpson's I |  | 12 | 160.2 |

Table 6

| Problem 2 <br> ified Method II |  |  |  |
| :---: | :---: | :---: | :---: |
| $h=1 / 500, \quad k=1 / 40,1$ | 1/160 |  |  |
| METH | $\Delta t$ | $\mathbf{S}_{\text {max }}$ | CPU |
| Rectangular Rule | .25D-1 | 40 | 51.9 |
| Trapezoidal Rule A |  | 8 | 28.2 |
| Trapezoidal Rule B |  | 8 | 31.8 |
| Simpson's Rule A |  | 8 | 24.5 |
| Rectangular Rule | .125D-1 | 80 | 183.8 |
| Trapezoidal Rule A |  | 11 | 70.2 |
| Trapezoidal Rule B |  | 10 | 78.1 |
| Simpson's Rule A |  | 12 | 64.1 |
| Rectangular Rule | .625D-2 | 160 | 690.4 |
| Trapezoidal Rule A |  | 15 | 168.2 |
| Trapezoidal Rule B |  | 14 | 197.6 |
| Simpson's Rule A |  | 8 | 105.3 |
| Modified Method I |  |  |  |
| METH | $\Delta t$ | $\mathbf{S}_{\text {max }}$ | CPU |
| Rectangular Rule | .25D-1 | 40 | 51.7 |
| Modified Trapezoidal I |  | 10 | 29.3 |
| Modified Trapezoidal II |  | 10 | 35.3 |
| Modified Simpson's I |  | 8 | 30.6 |
| Rectangular Rule | .125D-1 | 80 | 181.6 |
| Modified Trapezoidal I |  | 17 | 77.1 |
| Modified Trapezoidal II |  | 24 | 92.3 |
| Modified Simpson's I |  | 14 | 79.3 |
| Rectangular Rule | .625D-2 | 160 | 688.7 |
| Modified Trapezoidal I |  | 17 | 198.3 |
| Modified Trapezoidal II |  | 28 | 245.8 |
| Modified Simpson's I |  | 12 | 159.9 |

A comparison can be obtained by considering the performances of Modified Method II and Modified Method I applied to solve test problems 1 and 2. By examining Tables 5 and 6 we see, for the same choice of $h$ and $k$, there are differences in both storage requirements and overall execution times. All rules of Modified Method II have lower upper bounds of the storage $S_{\max }$ and CPU time than those rules of Modified Method I, with the exception of the Rectangular Rule (standard method).

## Chapter 6

## Conclusions

The goal of our project has been to implement and test a number of methods, developed in Roux and Thomée [15], Sloan and Thomee [17] and Zhang [20], to approximate the solution of integro-differential equations of parabolic type in a more efficient fashion. We used the Modified Method II whose code is called PIDECX. We applied the methods to solve two test problems of integro-differential equations of parabolic type to measure the improvements in accuracy, storage requirements and execution times of Modified Method II. In addition we implemented Modified Method I of Zhang [20], whose solver is called PIDETZ. We compared it with the Modified Method II with respect to accuracy, storage and execution time.

Sloan, Thomee [17] and Zhang [20] proved that it was possible to reduce greatly both the memory and computational requirements of the method if the integral term of (4.1) is approximated by a quadrature formula with relatively high-order truncation error so that a relatively large time step can be used for the quadrature formula. The analysis of results in section 5.6 demonstrate this conclusion.

As expected, from the results in Tables 3 and 4, (or 5, 6), Modified Method I has been shown to be very robust and efficient when solving certain test problems. However Modified Method II is more efficient than the previous solver when solving the same test problems in terms of both the storage requirements and execution times of CPU.

Our investigation of the proposed methods suggest several other possibilities for future work. One project would be to adapt the quadrature scheme to the case where nonuniform step-sizes of integration would be employed. This is based on some variable step-size schemes for time integration, such as the second order composite integration scheme ( $\theta$ BDF2) as proposed in Carroll [13]. Future implementations should be more effective in terms of computational expense and the composite integration scheme is
competitive with alternative approaches, particularly at moderate tolerances. Two other possible projects would be to conduct some numerical experiments for the problem with nonsmooth initial data, as have recently been done in Thomee and Zhang [21], and to consider implementations for the problem on a 2 -dimensional domain.

## Bibliography

[1] J. R. Rice, Numerical methods, software, and Analysis: IMSL ${ }^{\mathrm{R}}$ Reference Edition, McGraw-Hill Inc. 1983.
[2] T. Hopkins, C. Phillips, Numerical Methods in Practice: Using the NAG Library, Addison-Wesley Publishers Ltd. 1988.

## Finite Element Methods

[3] E. B. Becker, G. F. Carey, J. T. Oden, Finite Elements - An Introduction, Prentice-Hall Inc. 1981.

## Initial-Value Problems

[4A] W. H. Enright, T. E. Hull and B. Lindberg, Comparing numerical methods for stiff systems of ODE's, BIT 15 (1975) 10-48.
[4] C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Englewood Cliffs, NJ 1971.
[5] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II - Stiff and Differential Algebraic Equations, Springer-Verlan 1991.
[6] J. D. Lambert, Computational Methods in Ordinary Differential Equations, John Wiley \& Sons 1973.
[7] L. F. Shampine, M. K. Gordon, Computer Solution of Ordinary Differential Equations - The Initial Value Problem, W. H. Freeman \& Co., San Francisco 1975.
[8] J. Carroll, A composite integration scheme for the numerical solution of systems of ordinary differential equations, J. Comp. Appl. Math. Vol. 25, 1-13, 1989.

## Parabolic Differential Equations

[9] N. K. Madsen, R. F. sincovec, Algorithm 540: PDECOL, General collocation software for partial differential equations, ACM Trans. Math. Software, 5 (1979), pp. 326-351.
[10] R. F. Sincovec, N. K. Madsen, Software for nonlinear partial differential equations, ACM Trans. Math. software, 1 (1975), pp. 232-260.
[11] R. F. Sincovec, N. K. Madsen, Algorithm 494: PDEONE, solution of systems of partial differential equations, ACM Trans. Math. Software, 1 (1975), pp. 261-263.
[12] G. D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3rd Edition, Oxford University Press 1985.
[13] J. Carroll, A composite integration scheme for the numerical solution of systems of parablic PDEs in one space dimension, to appear in J. Comp. Appl. Math.

## Parabolic Integro-Differential Equations

[14] A. J. Jerri, Introduction to Integral Equations with Applications, Marcel Dekker Inc.,
[15] M. N. Le Roux, V. Thomée, Numerical solution of semilinear integro-differential equations of parabolic type, SIAM J. Numer. anal. 26 (1989), pp.1291-1301.
[16] A. K. Pani, V. Thomée, L. R. Wahlbin, Numerical methods for hyperbolic and parabolic integro-differential equations, Research Report CMA-R39-90, Centre for Mathematical Analysis, the Australian National University 1990.
[17] I. H. Sloan, V. Thomée, Time discretization of an integro-differential equation of parabolic type, SIAM J. Numer. Anal., 23 (1986), pp. 1052-1061.
[18] V. Thomèe, On the numerical solution of integro-differential equations of parabolic type, International Series on Numerical Mathematics, Vol. 86, Birkhauser-Verlag Basel, 1988.
[19] V. Thomèe, Numerical solution of integro-differential equations of parabolic type, Research Report, Chalmers University of Technology, June 1990.
[20] N. Y. Zhang, On the Discretization in Time and Space of Parabolic IntegroDifferential Equations, Ph.D.Thesis, Department of Mathematics, Goteborg 1990.
[21] V. Thomėe, N. Y. Zhang, Backward Euler Type Methods for Parabolic IntegroDifferential Equations with Nonsmooth Data, to be published.

