University of Newcastle upon Tyne

Department of Computing Science

An Investigation of Collocation Algorithms for Solving Boundary Value Problems System of ODEs

A thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy in Computing Science

Edy Hermansyah

September 2001

Acknowledgements

I would like to express my special thanks to Dr Kenneth Wright, for having suggested this subject of research. I am very grateful to him for his valuable guidance, advice and encouragement during development of this thesis. He has demonstrated a great deal of patience during his time as my supervisor, and offered innumerable valuable insight into my research topic. I owe Dr Kenneth Wright a significant debt for this work, and I fear that it may be one I will never be able to repay sufficiently.

My deepest gratitude goes to Dr John Lloyd and Dr Chris Phillips, both member of my thesis committee, for their valuable inputs and their very helpful comments throughout this work.

Many staff members of the Department of Computing Science deserve my gratitude. In particular, I wish to acknowledge Ms Shirley Craig who helped me so much in obtaining bibliographical references for this research.

Financial support has been provided by The Higher Education Project – University of Bengkulu (*Proyek Pengembangan Sebelas Lembaga Pendidikan Tinggi, P2SLPT-Universitas Bengkulu*).

The last but not the least, I am very grateful to my family in Palembang and Bengkulu who gave me a lot encouragement in my most difficult time. My very especial thanks are due to Ika, Tity and her husband, Yanti and her husband, Donga Yassin and his wife, Pak Cik B and his wife, my brothers Co , Oji and Man, as well as to Risma and her husband.

This thesis is dedicated to Sri Wahyuni who gave me almost every things needed when I attempted to recognize myself; and also to Annisa si Kanda, Amelia si Intan, and Afif si Willy. *mang kimang kiming anak anak ayah...*

Abstract

This thesis is concerned with an investigation and evaluation of collocation algorithms for solving two-point boundary value problems for systems of ordinary differential equations. An emphasis is on developing reliable and efficient adaptive mesh selection algorithms in piecewise collocation methods.

General background materials including basic concepts and descriptions of the method as well as some functional analysis tools needed in developing some error estimates are given at the beginning. A brief review of some developments in the methods to be used is provided for later referencing.

By utilising the special structure of the collocation matrices, a more compact block matrix structure is introduced and an algorithm for generating and solving the matrix is proposed. Some practical aspects and computational considerations of matrices involved in the collocation process such as analysis of arithmetic operations and amount of memory spaces needed are considered. An examination of scaling process to reduce the condition number is also presented.

A numerical evaluation of some error estimates developed by considering the differential operator, the related matrices and the residual is carried out. These estimates are used to develop adaptive mesh selection algorithms, in particular as a cheap criterion for terminating the computation process.

Following a discussion on mesh selection strategies, a criterion function for use in adaptive algorithms is introduced and a numerical scheme to equidistributing values of the criterion function is proposed. An adaptive algorithm based on this criterion is developed and the results of numerical experiments are compared with those using some well known criterion functions. The various examples are chosen in such a way that they include problems with interior or boundary layers.

In addition, an algorithm has been developed to predict the necessary number of subintervals for a given tolerance, with the aim of improving the efficiency of the whole process.

Using a good initial mesh in adaptive algorithms would be expected to provide some further improvement in the algorithms. This leads to the idea of locating the layer regions and determining suitable break points in such regions before the numerical process. Based on examining the eigenvalues of the coefficient matrix in the differential equation in the specified interval, using their magnitudes and rates of change, the algorithms for predicting possible layer regions and estimating the number of break points needed in such regions are constructed. The effectiveness of these algorithms is evaluated by carrying out a number of numerical experiments.

The final chapter gives some concluding remarks of the work and comment on results of numerical experiments. Certain possible improvements and extensions for further research are also briefly given.

Contents

chapter 1

Introduction and Preliminaries

1.1 General Background

Two-point boundary value problems associated with systems of linear and nonlinear ordinary differential equations occur in many branches of mathematics, engineering and the various sciences. In these problems, conditions are specified at the endpoints of an interval and a solution of the differential equations over the interval is sought which satisfies the given endpoint conditions. Since it is usually impossible to obtain analytic solutions to the two-point boundary problems met in practice, numerical approaches must be considered to tackle these problems. Here we restrict our consideration to the collocation methods for solving first order system of ordinary differential equations.

A fairly general first order system of n differential equations may be written in the form :

$$\boldsymbol{x}'(t) = \boldsymbol{f}(t,\boldsymbol{x}(t)), \qquad a < t < b. \qquad \ldots(1.1)$$

with a given linear constraint as the two-point boundary condition

$$\boldsymbol{B}_1\,\boldsymbol{x}(a) + \boldsymbol{B}_2\,\boldsymbol{x}(b) = \boldsymbol{\beta} \qquad \ldots(1.2)$$

Here $\boldsymbol{x}(t)$, $\boldsymbol{f}(t,\boldsymbol{x}(t))$ and $\boldsymbol{\beta}$ are n-vectors; $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are $(n\text{x}n)$ matrices.

Throughout this thesis we confine our attention to first order linear systems of the form

$$\boldsymbol{x}'(t) = \boldsymbol{A}(t)\,\boldsymbol{x}(t) + \boldsymbol{y}(t), \qquad a < t < b \qquad \ldots(1.3)$$

which for convenience will be expressed in linear operator form

$$\boldsymbol{L}\boldsymbol{x}(t) \equiv \boldsymbol{x}'(t) - \boldsymbol{A}(t)\,\boldsymbol{x}(t) = \boldsymbol{y}(t) \qquad \ldots(1.3a)$$

subject to the linear separated two-point boundary conditions

$$\begin{aligned} \boldsymbol{B}_a\, \boldsymbol{x}(a) &= \boldsymbol{\beta}_1 \\ \boldsymbol{B}_b\, \boldsymbol{x}(b) &= \boldsymbol{\beta}_2 \end{aligned} \qquad \ldots(1.4)$$

in which $\boldsymbol{A}(t)$ is a given $(n\text{x}n)$ matrix valued function and $\boldsymbol{y}(t)$ is a given n-dimensional vector. $\boldsymbol{B}_a$ is an $(m\text{x}n)$ matrix where $m < n$, and $\boldsymbol{B}_b$ is an $((n\text{-}m)\text{x}n)$ matrix; these two matrices have constant elements. $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are fixed vectors of size m and $(n\text{-}m)$ respectively.

When nonlinear problem of the form (1.1) are encountered a choice of procedures is available. Either non-linear algebraic equations are set up and solved by an iterative procedure or the problem itself is linearised and solved successively. Even though we are not going to examine further detail of these procedures in this thesis, here we note that the Newton-Raphson-Kantorovich method as an example of the second of the alternatives is widely used in practice. The convergence of this scheme has been investigated by Kantorovich and Akilov [32]. Further discussion may also be found in Roberts and Shipman [43].

1.2 Collocation Methods

In this section the collocation methods with global and piecewise polynomials will be introduced, followed by initial consideration of the basic ideas in break points placement for piecewise polynomials collocation methods.

1.2.1 Global and Piecewise Polynomial Solutions

Although the basic idea of the collocation method is very general, it basically involves forming an approximate solution as a linear combination of a convenient set of functions, the coefficients of which are determined by requiring the linear combination to satisfy the boundary conditions and the differential equations at certain interior points in the specified interval.

Suppose the boundary value problem (1.3)-(1.4) will be solved. Let $\boldsymbol{x}_q$ denote the collocation solution which is required to satisfy the differential equation (1.3)

exactly at q distinct points $\{\xi_k\}$ and the boundary conditions (1.4). A set of points $\{\xi_k\}$, $1 \le k \le q$, known as the collocation points is chosen distributed throughout interval $[a,b]$. If $\boldsymbol{p}_i$ and $\psi_i(t)$ denote constant coefficients and the basis functions forming the approximate solution respectively, the collocation solution can then be written in the form of a finite sum

$$\boldsymbol{x}_q(t) = \sum_{i=1}^{z} \boldsymbol{p}_i \psi_i(t) \qquad \ldots(1.5)$$

where z denotes the number of unknown vectors. Since there are n equations formed by the boundary conditions, (qxn) equations generated at collocation points, we in total have ((q+1)xn) equations, hence the number of unknown vectors is (q+1).

The constant vectors $\boldsymbol{p}_i = [p_{i1} \quad p_{i2} \quad \ldots \quad p_{in}]^T$, $1 \le i \le z$, can be found by collocating the differential equations (1.3) on the selected points $\{\xi_k\}$, which lead to (qxn) equations satisfied by the constant vectors $\boldsymbol{p}_i$, i.e.

$$\sum_{i=1}^{z} \boldsymbol{p}_i \boldsymbol{L}\psi_i(\xi_k) = \boldsymbol{y}(\xi_k), \qquad 1 \le k \le q \qquad \ldots(1.6)$$

By constraining the approximate solution to satisfy the boundary conditions we then have n remaining equations needed to determine the unknowns, namely

$$\begin{aligned} \boldsymbol{B}_a \boldsymbol{x}_q(a) &= \boldsymbol{\beta}_1 \\ \boldsymbol{B}_b \boldsymbol{x}_q(b) &= \boldsymbol{\beta}_2 \end{aligned} \qquad \ldots(1.7)$$

Any polynomials such as simple powers, Chebyshev polynomials or Legendre polynomials are an obvious and natural choice for the basis functions $\psi_i(t)$. As described in [32], Karpilovskaya considered and use some orthogonal polynomials in particular the Chebyshev polynomials as the basis functions. This was followed by an extended work carried out by Shindler and Vainiko as described in [46]. By transforming the differential equation into an associated operator equation, they showed that using the zeros of some orthogonal polynomials the collocation methods

can be shown to converge and the rate of growth of the norms of such operators may be obtained.

Some practical considerations and the application of the methods to nonlinear problems using Chebyshev polynomials as the basis function have been investigated by a number of researchers, for example Clenshaw and Norton [15] and Wright [54]. Cruickshank and Wright [18] have intensively used Chebyshev polynomials in investigating some computable error bounds for collocation solutions.

So far, we have introduced the collocation methods which use one polynomial formed by linear combination of certain basis functions to represent the solution over the whole range $[a,b]$. These methods which are the so called global collocation methods have been shown to be considerably reliable algorithm for solving some kind of problems [12,15,17]. Further work in numerical analysis has generally suggested that using piecewise polynomial functions lead to better convergence results and simpler proofs than using global polynomials. Particularly the piecewise collocation methods have great flexibility in placing the collocation points to accommodate the cases where the problems behave badly in some regions.

To construct a piecewise collocation solution, firstly the interval $[a,b]$ is subdivided into a number of subintervals, not necessarily of equal size, to form a partition

$$\pi_w : \ a = t_1 < t_2 < t_3 < \ldots < t_w < t_{w+1} = b \qquad \ldots(1.8)$$

where w denotes number of subintervals. In each subinterval (t_j, t_{j+1}), $1 \le j \le w$, the solution is approximated by $x_{wq[j]}$, a linear combination of certain basis functions, by requiring it to satisfy the differential equation (1.3) at a set of q distinct points $\{\xi_{kj}\} \subset [t_j, t_{j+1}]$, $1 \le k \le q$, as in the case of global collocation methods. For simplicity, the collocation points are chosen to be distributed in the same way in each subinterval.

The piecewise approximate solutions $x_{wq[j]}$ form an approximate solution for whole range $[a,b]$ and let x_{wq} denote this approximate solution. Obviously, the piecewise polynomial function x_{wq} is required to satisfy the boundary conditions (1.4). Moreover x_{wq} is also required to be continuously matched (like the exact

solution) on the break points $\{t_j\}$, $2 \le j \le w$. Such restrictions will be called the continuity conditions.

By expressing the piecewise polynomial collocation solution as

$$\boldsymbol{x}_{wq}(t) = \sum_{i=1}^{z} \boldsymbol{p}_{i[j]}\psi_i(t), \qquad t_j \le t \le t_{j+1}, \quad j = 1, 2, 3, \ldots, w \qquad \ldots(1.9)$$

the unknown vectors $\boldsymbol{p}_{i[j]}$ can be found by solving the linear system generated by

- n boundary conditions

$$\boldsymbol{B}_a\, \boldsymbol{x}_{wq}(a) = \boldsymbol{\beta}_1$$
$$\boldsymbol{B}_b\, \boldsymbol{x}_{wq}(b) = \boldsymbol{\beta}_2$$

- $n(w\text{-}1)$ continuity conditions

$$\boldsymbol{x}_{wq[j-1]}(t_j) = \boldsymbol{x}_{wq[j]}(t_j), \quad j = 2, 3, \ldots, w$$

- nqw collocation conditions

$$\sum_{i=1}^{z} \boldsymbol{p}_{i[j]} \boldsymbol{L}\psi_i(\xi_{kj}) = \boldsymbol{y}(\xi_{kj}), \qquad 1 \le k \le q, \quad 1 \le j \le w$$

Piecewise collocation methods not only allow great flexibility in placement of the collocation points but also provide a wider choice of basis functions. Ahmed [1], Gerrard [26] and Seleman [48] in their theses have used local Chebyshev polynomials. The use of B-Spline can be found in Ascher et al. [8], de Boor [22] and Dodson [24]. Hermite polynomials were used by de Boor and Swartz [23]. Ascher et al. [8] also used monomial basis functions and further application of this kind of basis can be found in Ascher [6] and Ascher et al. [10].

As mentioned earlier, to guarantee the convergence of the global collocation methods the collocation points should be chosen from known orthogonal polynomials. In most references mentioned the collocation points are either the zeros of Chebyshev polynomial (Chebyshev zeros) or the zeros of Legendre polynomials (Gauss points). A discussion of piecewise collocation based on Lobatto and Radau points can be found in Ascher and Bader [7], while Wright [56] considered the use of zeros of the ultra-spherical polynomials which is a generalisation of the choice of collocation points which includes Chebyshev and Gauss points.

1.2.2 Adaptive Mesh Selection Algorithms

There are some problems of the form (1.3) in which a collocation method using only uniform meshes will be either inefficient or it will not work at all. In this category will be most problems whose solutions (or their derivatives) have very sharp gradients such as problems with boundary layers.

Choosing a good mesh is essential if a method is to be efficient, in the sense a sufficiently accurate solution should be obtained as inexpensively as possible, for problems with solution having a narrow region of rapid change such as occur even for the very simple problems of this type.

Since the collocation methods on a computer involve repetitions of some computation processes to achieve a desired approximate solution, two issues arise. Firstly a reliable strategy in choosing the break points in a mesh is needed; secondly a good initial mesh to start the collocation process is also important. The first issue corresponds to adaptive mesh selection, where the approximate solution and the mesh are repeatedly updated until prescribed error criteria are deemed satisfied. The second one relate to initial consideration of the problem itself, possibly by a preliminary mathematical analysis, for instance an initial solution profile can be used to construct an initial mesh, the stiffness may give some hint that in some regions we should place more points than other regions.

Despite the tremendous importance of mesh placement strategies in collocation methods, very little has been carried out about the problem of choosing those nonuniform meshes in the way most adequate for a given problem. Beside the most outstanding contribution of de Boor [22], the other references include Ahmed [1], Russell and Christiansen [45], Seleman [48] and Wright et al. [60].

1.3 Aim

This thesis is primarily concerned with developing some reliable, in terms of accuracy and efficiency, collocation algorithms for solving boundary value problems for system of ordinary differential equations. In particular we shall investigate a

variety of mesh selection strategies for piecewise collocation methods. An efficient criterion function to be used in adaptive mesh selection algorithms will be introduced and some results of numerical comparisons will be discussed, followed by drawing some conclusion.

When attempting to develop an adaptive mesh selection algorithm, at least we deal with the process of

- choosing an initial mesh
- setting up and solving system of linear equations
- constructing a new mesh for the next stage
- a criterion for terminating the process.

In this work all the above will be considered in detail, and some discussion and evaluation based on numerical experiment will be presented.

Some error estimate processes based on consideration of the differential operator involved and on the residual will be evaluated in numerical experiments. One of these estimates will then be used in constructing a new mesh and for terminating the computation process.

1.4 Structure of the Thesis

In chapter 2, the boundary value problem of the form (1.3) is transformed and defined in operator form. This enables us to relate it to the theory of the projection method needed in developing our work. Subsequently, a brief review of recent developments in collocation algorithms is presented.

Chapter 3 deals with developing algorithms for solving the system of linear equations generated in the collocation process. Some computational considerations such as column scaling schemes for reducing the condition number of the matrix involved in the equation system will be examined and some practical results will be displayed and discussed. Analysis of arithmetic operations and amount of memory needed for both full matrix representation and developed block matrix are considered and then followed by carrying out numerical comparisons.

Some numerical evaluations of the error estimates are presented in chapter 4. Particular attention is given to boundary value problems having severe layers, from which we observe the reliability of the error estimates when dealing with these problems.

Chapter 5 consists of investigation of some adaptive mesh selection algorithms and their numerical comparisons. Firstly we present some basic concepts used in adaptive mesh selection algorithms, followed by discussing some well established algorithms. We then introduce the $r_i h_i$ criterion function including the motivation for using it, and developing special numerical scheme to equidistribute the terms $r_i h_i$ in mesh placement algorithms. Finally, a comprehensive numerical comparison is carried out.

A possibility of using multiple interval increment/decrement in mesh selection algorithms is introduced in chapter 6. The developments discussed here are based on results of chapters 4 and 5.

In chapter 7, we introduce some possibilities in determining location of the layer regions based on consideration of behaviour of the eigenvalues of the matrix in the differential equations. This is followed by introducing an algorithm for estimating the width of layer regions and determining an initial mesh in the layer regions.

Finally, the last chapter gives some concluding remarks and final notes. These lead to some possibilities in further improvement and extension.

It is now convenient to state that a variety of algorithms developed in this work were implemented in g^{++} which is a free C^{++} compiler provided by GNU Project. All computations were performed in double precision arithmetic on a PC based on Intel Pentium® III 650MHz processor with 256 MHz RAM running Linux Mandrake™ 7.1. Some graphical illustrations were generated using MATLAB® 5.2. running on Microsoft Windows® 2000.

chapter 2

Review of Some Developments in the Collocation Methods

2.1 Introduction

As we have mentioned in the previous chapter we are principally concerned with the practical aspects in developing collocation algorithms. However much of the argument which we need to justify our developed algorithms utilise and make use some theoretical aspects of the collocation methods. These background materials involve a variety of areas in numerical analysis, as well as some functional analysis tools in addition to ordinary differential equation theory.

In this chapter, we first introduce the theoretical background for certain operator equations and their approximate solution. Most of the results are well known but are included for completeness. Based on these results the collocation method can be considered as projection method from which some properties may be deduced. Following discussion on the relationship between the projection and the collocation method we provide a brief survey of some development in the theory of these methods as a convenient review for later referencing.

A model problem we will consider is a linear system of n first order differential equations of the form

$$\boldsymbol{x}'(t) = \boldsymbol{A}(t)\,\boldsymbol{x}(t) + \boldsymbol{y}(t), \qquad a < t < b \qquad \ldots(2.1)$$

subject to the general form linear two-point boundary conditions for a first order system

$$\boldsymbol{B}_a\,\boldsymbol{x}(a) + \boldsymbol{B}_b\,\boldsymbol{x}(b) = \boldsymbol{\beta} \qquad \ldots(2.2)$$

here $\boldsymbol{B}_a$ and $\boldsymbol{B}_b \in \boldsymbol{R}^{n\times n}$ and $\boldsymbol{\beta} \in \boldsymbol{R}^n$. It is notable that, even though in all our work the boundary conditions considered are in the separated form, there is no difficulty to

convert a problem with condition of the form (2.2) to the separated one. A simple trick to do this can be found in Ascher et al. [10], though it does double the size of the system.

2.2 Collocation and Projection Method

To examine the relationship between the collocation and the projection method we need to introduce some concepts and notations usually used in functional analysis. A nice book of Moore [38] provides a practical approach to the theory of functional analysis and contains those basic concepts.

Let X and Y be normed linear spaces and suppose $\|.\|_X$ and $\|.\|_Y$ denote the norm in X and Y respectively. Let $[X,Y]$ denotes the space of bounded linear operator mapping $X \rightarrow Y$ with subordinate norm. Suppose we are given an equation of the form

$$F x = y \qquad \ldots(2.3)$$

where $F \in [X,Y]$, $y \in Y$. The equation (2.3) is to be solved for $x \in X$.

Since it is not always possible to solve (2.3) analytically, a numerical method should be considered to approximate (2.3). i.e.

$$\tilde{F} \tilde{x} = \tilde{y} \qquad \ldots(2.4)$$

should be solved for $\tilde{x}$, where $\tilde{x} \in \tilde{X} \subset X$ and $\tilde{y} \in \tilde{Y} \subset Y$, $\tilde{F} \in [\tilde{X}, \tilde{Y}]$.

Let φ be a projection from $Y \rightarrow \tilde{Y}$, i.e. $\varphi(Y) = \tilde{Y} = \varphi(\tilde{Y})$. With this background we now define a projection method as a method in which an approximate solution for equation (2.3) is sought to satisfy an equation in the form

$$\varphi(F \tilde{x} - y) = 0 \qquad \ldots(2.5)$$

In words, we can say that for any approximate solution $\tilde{x}$ to problem (2.3) the value of $(F \tilde{x} - y)$, known as the residual, should be made to be as close to zero as possible (since this is so for the true solution) and the projection methods require the approximate solution $\tilde{x}$ to satisfy the condition that the corresponding residual is mapped to zero under the influence of the projection method.

Having introduced the projection method in general setting, it is now demonstrated that collocation process can be viewed as a projection method.

As described in detail in chapter 1, the basic idea of collocation methods has great generality and simplicity. Given a system of ordinary differential equations which can be written as an operator equation and its associated boundary conditions, an approximate solution is then sought in the form of a linear combination of some basis functions. The coefficients in the linear combination are found by substitution into the equation, then by satisfying the boundary conditions and the differential equation at certain distinct points. The number of collocation points is chosen so that the number of generating equations is equal to the number of unknowns.

Suppose a collocation solution for problem (2.1) in the form

$$x_q(t) = \sum_{i=1}^{q+1} p_i \psi_i(t) \qquad \ldots(2.6)$$

is sought by requiring $x_q(t)$ to satisfy the boundary condition (2.2) and the equation (2.1) exactly at a set of distinct points $\{\xi_i\}$, $1 \le i \le q$. Let φ_q be the projection from $Y \rightarrow \tilde{Y}$ mapping each continuous function using interpolation at the collocation points $\{\xi_i\}$. This means that for a continuous function y, $\varphi_q y$ can be expressed as a combination of q basis functions for $\tilde{Y}$ and is such that

$$(\varphi_q y)(\xi_i) = y(\xi_i), \quad 1 \le i \le q$$

Since the method requires that the approximate solution $x_q(t)$ exactly satisfies the differential equation at the collocation points. i.e.

$$x_q'(\xi_i) - A(\xi_i)\, x_q(\xi_i) - y(\xi_i) = \mathbf{0}, \qquad 1 \le i \le q \qquad \ldots(2.7)$$

By rewriting (2.1) as

$$Lx(t) \equiv x'(t) - A(t)\, x(t) = y(t) \qquad \ldots(2.8)$$

then equation (2.7) can be written

$$L\, x_q(\xi_i) - y(\xi_i) = x_q'(\xi_i) - A(\xi_i)\, x_q(\xi_i) - y(\xi_i) = \mathbf{0}$$

This means that the polynomial of degree (q-1) interpolating the residual at these q points must be identically zero, i.e.

$$\varphi_q(x_q' - Ax_q - y) = \varphi_q(Lx_q - y) = 0 \qquad \ldots(2.9)$$

It is clear that the approximate solution x_q satisfies an equation of the form (2.5) showing that we have indeed a projection method. Even though we have shown this fact in this fairly specific case, but the collocation method is in fact a projection method under very general circumstances. For further details can be seen in [17] and [26].

As mentioned in the previous chapter, Karpilovskaya described in [32], Shindler and Vainiko described in [46] have considered and use some special polynomials as the basis functions in the global collocation methods. In their analysis the differential equation is transformed into an associated operator equation, the collocation condition then turns out to be equivalent to a projection of the operator equation into finite dimensional subspace. It is also described in [46] that despite of classical result of Natanson saying that the projection operator cannot be uniformly bounded, by using the special case of interpolation at the zeros of some orthogonal polynomials the collocation methods can be shown to converge and the rate of growth of the norms of such operators may be obtained.

2.3 Error Bounds for Collocation Solutions

The result in the previous section in which it is shown that the collocation methods may be viewed as projection methods is the starting point to theoretically obtain some error bounds. Concepts in functional analysis are other important tools, though this will not be discussed in detail here.

The following analysis is related to work of Anselone [5], Cruickshank [17], Kantorovich and Akilov [32], and Phillips [41].

To be more consistent in using notation, let X_q and Y_q be subspaces of the normed linear spaces X and Y respectively and let φ_q denotes linear projection

$Y \rightarrow Y_q$. So far, there is no restriction in dimensionality, here the subscript q indicates the dimension of subspace Y_q.

In the following discussion we are concerned with the operator F defined in equation (2.3) which may be split into two parts

$$F = D - M \qquad \ldots(2.10)$$

where the operator D denotes the differentiation operator which is assumed to be invertible, i.e. there exists $D^{-1} \in [Y,X]$. In certain circumstance F may be deduced to be invertible as well. Note that equation (2.3) may now be written as

$$(D - M)\, x = y \qquad \ldots(2.11)$$

and equation (2.9) becomes

$$\varphi_q ((D - M)\, x_q - y) = 0$$

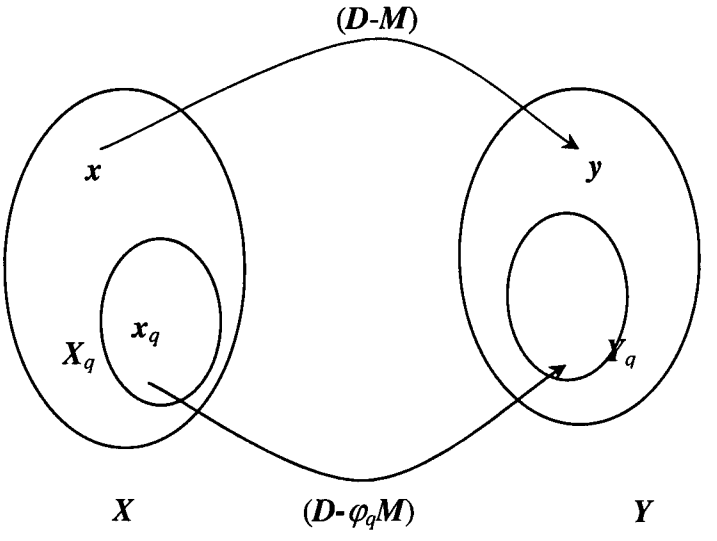
or

$$\varphi_q (Dx_q - Mx_q - y) = 0 \qquad \ldots(2.12)$$

It is assumed that $\varphi_q D x_q = D x_q$, i.e. D is defined in X_q establishes a bijection between X_q and $Y_q = \varphi_q Y$. Hence equation (2.12) can further be simplified

$$(D - \varphi_q M)\, x_q = \varphi_q y \qquad \ldots(2.13)$$

An illustration of the concept described can be drawn as follows



$(D - M)$: $X \rightarrow Y$. $\qquad (D - \varphi_q M)$: $X_q \rightarrow Y_q$

For the purposes of analysis it is convenient to work in terms of the functions $\boldsymbol{u} = \boldsymbol{Dx}$ and $\boldsymbol{u}_q = \boldsymbol{Dx}_q$, which satisfy the equations

$$(\boldsymbol{I} - \boldsymbol{K})\,\boldsymbol{u} = \boldsymbol{y} \qquad \ldots(2.14)$$

and

$$(\boldsymbol{I} - \varphi_q \boldsymbol{K})\,\boldsymbol{u}_q = \varphi_q \boldsymbol{y} \qquad \ldots(2.15)$$

where $\boldsymbol{I}$ denotes the identity operator and $\boldsymbol{K} = \boldsymbol{MD}^{-1}$.

Let the residual be defined as

$$\boldsymbol{r}_q = (\boldsymbol{I} - \boldsymbol{K})\,\boldsymbol{u}_q - \boldsymbol{y} \qquad \ldots(2.16)$$

then the error $\boldsymbol{e}_u = (\boldsymbol{u}_q - \boldsymbol{u})$ in $\boldsymbol{u}_q$ is related to $\boldsymbol{r}_q$ by

$$\boldsymbol{e}_u = (\boldsymbol{I} - \boldsymbol{K})^{-1}\,\boldsymbol{r}_q \qquad \ldots(2.17)$$

and the error in $\boldsymbol{x}_q$ is related to $\boldsymbol{e}_u$ by

$$\boldsymbol{x}_q - \boldsymbol{x} = \boldsymbol{D}^{-1}\,\boldsymbol{e}_u \qquad \ldots(2.18)$$

so that once a bound on $\| (\boldsymbol{I} - \boldsymbol{K})^{-1} \|$ has been obtained, (2.17) can be used to bound $\| \boldsymbol{e}_u \|$ and bound for the error can be obtained from (2.18).

By considering the collocation method as projection method and applying Anselone's proposition [5]

$$(\boldsymbol{I} - \boldsymbol{K})^{-1} = \boldsymbol{I} + (\boldsymbol{I} - \boldsymbol{K})^{-1}\boldsymbol{K}$$

a bound on $\| (\boldsymbol{I} - \boldsymbol{K})^{-1} \|$ can be related to bound on projection operator $\| (\boldsymbol{I} - \varphi_q \boldsymbol{K})^{-1} \|$ by

$$\| (\boldsymbol{I} - \boldsymbol{K})^{-1} \| = (\| (\boldsymbol{I} - \varphi_q \boldsymbol{K})^{-1} \|) / (1 - \delta_0)$$

if

$$\delta_0 = (\| (\boldsymbol{I} - \varphi_q \boldsymbol{K})^{-1} \|)(\| (\boldsymbol{I} - \varphi_q)\boldsymbol{K} \|) \qquad \ldots(2.19)$$

In [18] Cruickshank and Wright consider the m^{th}-order linear differential equation of the form

$$x^{(m)}(t) + \sum_{j=0}^{m-1} p_j(t) x^{(j)}(t) = y(t), \qquad a < t < b$$

with m associated boundary conditions. This may be written in operator form

$$(\boldsymbol{D}^m - \boldsymbol{M})x = y$$

where $\boldsymbol{D}^m$ denotes the differential operator $(\boldsymbol{D}^m x)(t) = \frac{d^m}{dt^m} x(t)$.

In the paper they discussed how to bound the norm of the operator $(\boldsymbol{I} - \varphi_q \boldsymbol{K})^{-1}$ by relating it to the matrix used in the numerical solution of the original problem. They also analysed some other quantities needed to establish a bound on $\| (\boldsymbol{I} - \boldsymbol{K})^{-1} \|$.

Further investigation by Wright [56] where he introduced certain matrix called matrix $\boldsymbol{W}_q$ which is related to q-point global collocation solution of m^{th}-order differential equation. The $\boldsymbol{W}_q$ is related to the associated collocation matrix using

$$\boldsymbol{W}_q = \boldsymbol{C}_0 \boldsymbol{C}^{-1}$$

where $\boldsymbol{C}_0$ is the collocation matrix corresponding to $\boldsymbol{D}^m$ and $\boldsymbol{C}$ is that of $(\boldsymbol{D}^m - \boldsymbol{M})$.

It was shown that if the zeros of certain orthogonal polynomials are taken as the collocation points, then the matrix $\boldsymbol{W}_q$ has maximum norm which tends, as $q \to \infty$, to the maximum norm of the operator $(\boldsymbol{I} - \boldsymbol{K})^{-1}$ related to the differential equation.

In similar spirit, for piecewise polynomial collocation with w subintervals Gerrard and Wright [27] shown that under suitable conditions if the maximum subinterval size tends to zero as $w \to \infty$, the norm of certain matrix $\boldsymbol{W}_{wq}$ related to q-point piecewise collocation solution of m^{th}-order differential equation, tends to the norm of $(\boldsymbol{I} - \boldsymbol{K})^{-1}$.

An extended analysis of Ahmed and Wright [2] in which they considered the related operator in the form $(\boldsymbol{D} - \boldsymbol{M})$ rather than $(\boldsymbol{I} - \boldsymbol{K})$ as in equation (2.16) resulted in introducing certain matrix $\boldsymbol{Q}_{wq}$. In the paper the matrix $\boldsymbol{Q}_{wq}$ is defined as the matrix that maps the right hand side values into solution values where the right hand side is evaluated at the collocation points while the solution is evaluated at a set of points which, for simplicity, could be the collocation points. The results suggest direct estimates for the error in the solution rather than the use of the m^{th} derivative as an intermediate stage. Under some assumptions it is shown that the norm of this matrix tends to the norm of $(\boldsymbol{D} - \boldsymbol{M})^{-1}$, provide either $q \to \infty$ or q fixed and $w \to \infty$.

2.4 Brief Review of Some Other Developments

In early 1970's, Russell and Shampine [46] analysed a class of collocation methods for the numerical solution of BVP for a higher order ordinary differential equations. They considered existence and uniqueness of a collocation solution for single higher order ordinary differential equations, and the convergence of such solution as h tends to zero. Here h denotes the maximum subinterval size of the partition π. Moreover, their results also indicate that the solution of an m^{th} order linear ordinary differential equation can be approximated to within $O(|h|^k)$ by collocation when using spline function of order $(m+k)$ on a partition π and the solution is in $C^{(m+k)}$. Their estimate is to be compared with error of $O(|h|^{k+m})$ often achievable with the same spline spaces using certain other projection methods, such as Galerkin's method, the least square method and certain of its variants.

Russell and Shampine's work was extended and supplemented by a number of ways, for example de Boor and Swartz [23] shown the same order of convergence $O(|h|^{k+m})$ can be achieved by collocation with spline function in $C^{(m-1)}$ using zeros of the Legendre polynomial (Gauss points) relative to each subinterval as the collocation points, provided the differential equation is sufficiently smooth. Furthermore, at the end of each subinterval additionally convergence order called superconvergence order, i.e. an order of convergence higher than the best possible global order, is also achieved. In [23] it was shown that the approximation is $O(|h|^{2k})$ accurate at these points.

While the concept of stability plays important rule for initial value problems as a description of asymptotic behaviour ($t \rightarrow \infty$), sensitivity of boundary value problems on a finite interval is more appropriately described in terms of conditioning which is closely connected with concept dichotomy. Details discussion of these can be referred to [10] and [11]. An interesting paper of Swartz [50] discusses, in particular, the conditioning of collocation matrices.

Ascher and Bader [7] considered stiff problem and carried out some comparisons using Gauss, Lobatto and Radau points. One of important results in the paper is that

using collocating at Gauss points give better results. Also when solving certain very stiff boundary value problems there is a reduction in the superconvergence order of Gaussian collocation points, and no such order reduction is present for collocation with Lobatto or Radau points.

We have mentioned stiffness without looking in further detail. Stiffness cannot be defined in precise mathematical terms in a satisfactory manner, even for restricted class of linear constant coefficient systems [36]. However, for our purposes, qualitatively a boundary value problem is said to be stiff if its solution rapidly change in some narrow regions. Stiffness has close connections with singular perturbation problems; indeed system exhibiting singular perturbation can be seen as a sub-class of stiff system [36]. Solving these problems with collocation have been widely discussed and references include Ascher and Weiss [9], Kreiss et al. [34,35], Russell and Shampine [47]. More general approach in solving such problems can be found in Aitken (ed.) [4], Ascher and Russell (eds.) [11], Hemker [29], Hairer and Wanner [31].

Another important issue is how to choose the basis function $\psi_i(t)$ so as to obtain an efficient and stable method. In series of Wright's papers and his PhD student's works the Chebyshev polynomials have been intensively used as basis function. The other references include Ascher at al. [8] and Clenshaw and Norton [15] . Notes on applied computing by the National Physical Lab. [39], Fox and Parker [25] summarise the properties of these famous polynomials and indicate their use in numerical analysis. The choice of B-Spline basis representation motivated primarily by the fundamental work of de Boor and Swartz [23] has been increasingly popular, for example Ascher et al. [10] provide a general purpose code for solving boundary value ordinary differential equations. Despite the popularity of B-Splines including their utility in approximation problems such as surface fitting and curve design, some doubt has been expressed as to their suitability for solving differential equations, especially when low continuity piecewise polynomials are used. With this motivation Ascher et al. [8] carried out some comparison of various representation of the solution. A notable point of their work is that using Chebyshev series representation

is recommended since experimentally it produces roundoff errors at most as large as those for B-splines, and it is much easier and shorter to implement. Moreover it is slightly cheaper than the others.

Most of the references mentioned deal with single higher order differential equations though some of them theoretically consider a general form of first order system of differential equations. Some theoretical and practical considerations for system of differential equations has been made by Russell [44].

A discussion of block matrix structures arising in the discretisation of a given boundary value problem can be found in Ascher et al [10], where they considered general banded matrices arise when one is solving a boundary value problem using multiple shooting or finite difference scheme. While in [58] a parallel treatment of some matrices in the solution of boundary value problems is discussed and some numerical comparisons are presented.

As mentioned in the previous chapter, very little has been added about the problem of choosing those nonuniform meshes in the way most adequate for a given problem. By utilising the matrix Q_{wq} mentioned in §2.3, Ahmed [1] introduced the use of such matrix in adaptive mesh selection algorithm. The algorithm works well for problems having smooth solutions, however the algorithm is very expensive since it involves forming for the inverse of the collocation matrix. The work of Seleman [48] tried to reduce the cost by developing some modified algorithms, but the cost is still fairly high. In [48] an algorithm based on an error estimate is also introduced. This error estimate is obtained by multiplying two polynomials, one representing the residual and the other approximating the Green's function at collocation points. Numerical results indicate that this algorithm performs better than those using the matrix Q_{wq}, however, again the computational cost is not cheap. For further references, two comprehensive reviews of some developments in error estimation and mesh selection for collocation methods illustrated with some results of numerical experiments can be found in [57] and [59].

The most recent work published by Wright et al. [60] introduce some subdivision criterion functions developed by taking into account the influence of the behaviour in

one subinterval on the error in others. They show that the algorithms developed do work well when the solution is sufficiently smooth. Unfortunately, their numerical experiments also indicate that the most sophisticated criterion function, SINFLB, gives very poor results when severe layers are present.

chapter 3

Developing Algorithms for Solving the Collocation Matrix

3.1 Introduction

We shall consider the linear system of n first order differential equations of the form

$$x'(t) = A(t)\,x(t) + y(t), \qquad a < t < b \qquad \ldots(3.1)$$

subject to the linear separated two-point boundary conditions

$$\begin{aligned} B_a\,x(a) &= \beta_1 \\ B_b\,x(b) &= \beta_2 \end{aligned} \qquad \ldots(3.2)$$

in which $x(t)$ and $y(t)$ are n-dimensional vectors. $A(t)$ is an $(n \times n)$ matrix valued function. B_a and B_b are $(m \times n)$ matrix and $((n-m) \times n)$ matrix respectively, where $m < n$. β_1 and β_2 are fixed vectors of size m and $(n-m)$.

Suppose the partition :

$$\pi_w : a = t_1 < t_2 < t_3 < \ldots < t_w < t_{w+1} = b,$$

where w denotes number of subintervals, is chosen and we wish to compute a piecewise approximate solution of the boundary value problem (3.1)-(3.2) using q collocation points in each subinterval. In each subinterval $[t_k, t_{k+1}]$, the q collocation points are determined by

$$\xi_{ik} = t_k + \frac{t_{k+1} - t_k}{2}(1 + \xi_i^*), \quad \text{where } i = 1, 2, \ldots, q;\ k = 1, 2, \ldots, w.$$

$\{\xi_i^*\}$, $i = 1, 2, \ldots, q$, denote the chosen reference points in interval [-1,1]. In principle, any point in [-1,1] can be taken as reference points, though they are particularly chosen as the zeros of either Legendre polynomials (Gauss points) or Chebyshev polynomials (Chebyshev zeros/points).

After carrying out the discretisation process to the boundary value problem (3.1)-(3.2) over the partition π_w we then encounter the need to solve a large, staircase form, system of linear equations

$$\boldsymbol{Cp} = \boldsymbol{g} \qquad \ldots(3.3)$$

here, matrix $\boldsymbol{C}$ and vector $\boldsymbol{p}$ will be referred as the collocation matrix and the parameter of collocation process respectively.

In this chapter, firstly we shall studysome numerical considerations in solving the collocation equations, specifically, some column scaling scheme will be examined and implemented to observe the numerical behaviour. Secondly we shall examine some well-known techniques in setting up the collocation matrix and, finally, followed by discussing some proposed algorithms to construct and deal with the special structure of the collocation matrix.

Gaussian elimination with partial pivoting works very well in practice, even though an accurate solution is not absolutely guaranteed, in the sense there exist ill-conditioned systems that simply can not be solved accurately in the presence of roundoff errors. A more accurate algorithm can be guaranteed, if the complete pivoting strategy is employed in the algorithm. The theoretical superiority of complete pivoting over partial pivoting is discussed in detail in [28] and [51].

In spite of the theoretical superiority of complete pivoting over partial pivoting, Gaussian elimination with partial pivoting is much more widely used for some reasons, firstly it works very well in practice, and secondly it is much less expensive. Hence, the Gaussian elimination with partial pivoting will be used as basic tool for solving the linear system (3.3).

3.2 Computational Consideration of Collocation Matrices

In this section, some relationship between the condition number of a matrix and column scaling operations will be highlighted. The column scaling operation particularly developed for collocation matrix will also be described in some details. Finally, a number of illustrative numerical results are presented.

3.2.1 Scaling Operation and Condition Number

The condition number $\kappa(C)$ of any non-singular matrix C is defined as $\|C\|\,\|C^{-1}\|$. The condition number $\kappa(C)$ provides a simple but useful measure of the sensitivity of the linear system $Cp = g$. If $\kappa(C)$ is large we say that C is ill conditioned.

It is well known that any matrix that has columns whose norms differ by several orders of magnitude is ill conditioned. The same can be said of the rows. Thus a necessary condition for a matrix to be well conditioned is that the norms of its rows and columns be of roughly the same magnitude.

Any equation in linear system (3.3) can be multiplied by any nonzero constant without changing the solution of the system. Such operation is called a row scaling operation. A similar operation can be applied to the columns of matrix C. By contrast this so called column scaling operations do change the solution, hence an appropriate descaling operations needs to be carried out afterward.

Although the rows and columns of any matrix can easily be rescaled so that all rows and columns have about the same magnitude, there is no unique way of doing it. This suggests that a particular matrix may need some special treatments such that its condition number reduces.

Gaussian elimination with partial pivoting, although not unconditionally stable, is stable in practice. Therefore, this could guarantee that if the collocation matrix is well conditioned then this method will solve the linear system (3.3) accurately.

In studying the methods for solving linear systems, some linear systems are ill conditioned simply because they are out of scale, this turns out that scaling operations are necessary since these operations may affect the numerical properties of a system.

This section contains the work on column scaling of collocation matrices. Firstly we consider the simple full matrix resulting from implementation of the global collocation method, then it is followed by considering the column scaling process for piecewise representation.

Let an approximate solution x_q of boundary value problem (3.1)-(3.2) for global collocation method be written as follows

$$x_q = \sum_{i=1}^{z} p_i \psi_i(t) \qquad \ldots(3.4)$$

where $p_i = [p_{i1}\ p_{i2}\ \ldots\ p_{in}]^T$ and $\{\psi_i\}$ are the unknown vectors and certain polynomials of degree $(i\text{-}1)$ respectively.

In the collocation process, the linear system generated will be in the form

$$[C]_{(nzXnz)} \cdot [p]_{(nzX1)} = [g]_{(nzX1)} \qquad \ldots(3.5)$$

where

C is a matrix associated with the collocation, continuity and boundary conditions

g is a column matrix , associated with the right hand side of equation (3.1)

p represents the unknown parameters of the collocation process which is a column matrix, the solution of the linear system $Cp = g$.

In the discretisation process, for convenience the elements of the collocation matrix C and parameters p_i are arranged such that they have the form

$$[C]_{(nzXnz)} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_z \end{pmatrix}_{(nzX1)} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_z \end{pmatrix}_{(nzX1)}, \text{ where } p_i = \begin{pmatrix} p_{i1} \\ p_{i2} \\ \vdots \\ p_{in} \end{pmatrix}_{(nX1)}, \quad 1 \le i \le z$$

Looking at the parameter $p = [p_{11}\ p_{12} \ldots p_{1n} \mid p_{21}\ p_{22} \ldots p_{2n} \mid \ldots \mid p_{z1}\ p_{z2} \ldots p_{zn}]^T$, we can see that the blocks $[p_{i1}\ p_{i2}\ \ldots\ p_{in}]$, $1 \le i \le z$, correspond to the $(i\text{-}1)$-degree of polynomial ψ_i, hence elements of the matrix C associated with these blocks will be treated in the same way.

The column scaling operation will be applied to the collocation matrix C by multiplying C with block diagonal matrix $D_{(nzXnz)} = diag(d_j)$, $j = 1, 2, \ldots, z$, to give

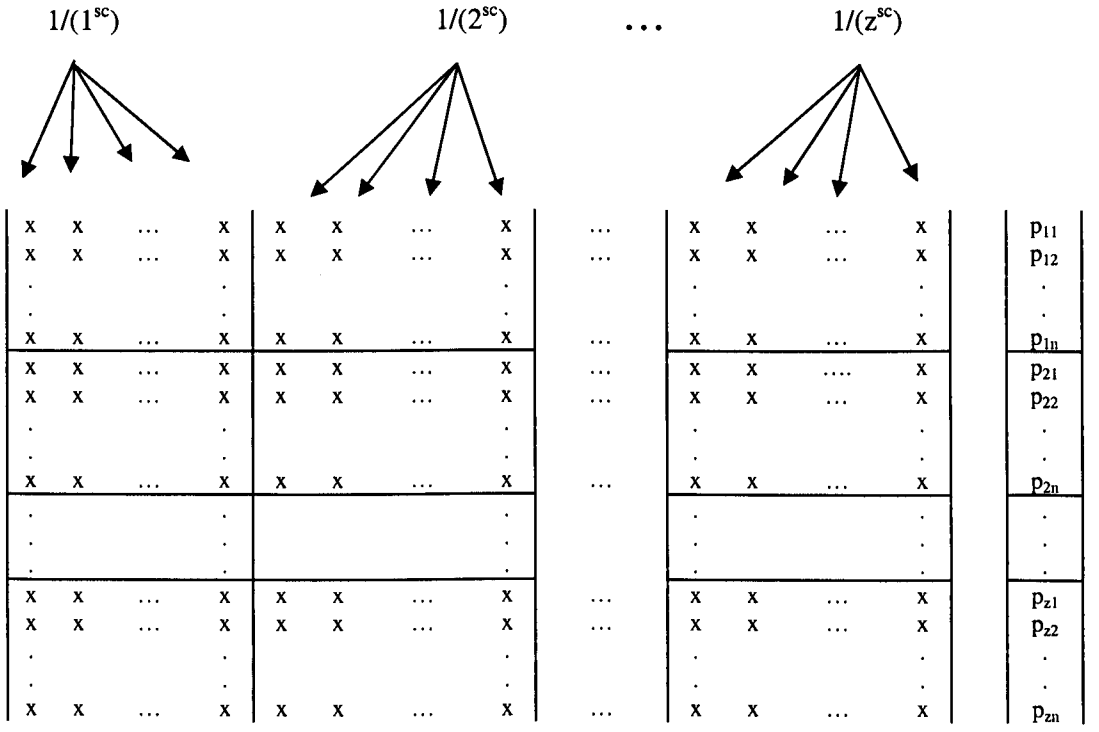
$$[C_s]_{(nzXnz)} = [C]_{(nzXnz)} \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_z \end{pmatrix}_{(nzXnz)}$$

where d_j are diagonal matrices of form

$$\boldsymbol{d}_j = \begin{pmatrix} \frac{1}{j^{sc}} & 0 & \cdots & 0 \\ 0 & \frac{1}{j^{sc}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{j^{sc}} \end{pmatrix}_{(nXn)}$$

sc is the scaling parameter and it is an integer. Here, **0** denotes square zeros matrix

Diagrammatically, the column scaling process will look like :



For the piecewise representation similar scaling is applied such that all elements of the subintervals corresponding to the same order of polynomial representation are treated in the same way.

Let $\boldsymbol{x}_{wq[j]}$ be the collocation solution for j^{th}-subinterval and it can be written as linear combination of basis functions ψ_i as follows

$$\boldsymbol{x}_{wq[j]} = \sum_{i=1}^{z} \boldsymbol{p}_{i[j]}\, \psi_i(t)\,, \qquad \ldots(3.6)$$

where $j = 1, 2, 3, \ldots w$; w is number of subintervals and t is the independent variable in the j^{th}-subinterval and $\boldsymbol{p}_{i[j]} = [p_{i1[j]}\ p_{i2[j]}\ \cdots\ p_{in[j]}]^T$. Note that the unusual notation for the additional index $[j]$ which appears in vector $\boldsymbol{p}_{i[j]}$ is for clarity since $\boldsymbol{p}_i$ itself is a vector.

The collocation parameter $\boldsymbol{p}$ then has the form :

$$\boldsymbol{p} = [\boldsymbol{p}_{1[1]}\ \boldsymbol{p}_{2[1]}\ \cdots\ \boldsymbol{p}_{z[1]} \mid \boldsymbol{p}_{1[2]}\ \boldsymbol{p}_{2[2]}\ \cdots\ \boldsymbol{p}_{z[2]} \mid \ldots \mid \boldsymbol{p}_{1[w]}\ \boldsymbol{p}_{2[w]}\ \cdots\ \boldsymbol{p}_{z[w]}]^T$$

corresponding to : 1^{st}-subinterval 2^{nd}-subinterval w^{th}-subinterval

Elements of the collocation matrix $\boldsymbol{C}$ associated with the elements of $\boldsymbol{p}$ will be treated in the same ways as follows :

$$c_{ik[j]} := c_{ik[j]} * 1/(i^{sc})$$

where $i = 1, 2, 3, \ldots, z$; $j = 1, 2, 3, \ldots, n$; $k = 1, 2, 3, \ldots, w$.

3.2.2 Some Results of Numerical Experiments

For illustration we employ such column scaling operation in the collocation algorithm for solving the following boundary value problems.

Problem 1 :

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -t & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -2t^3 - 6 + te^t \end{pmatrix}, \quad -1 < t < 2$$

BCs : $x_1(-1) = e^{-1} + 1$; $x_2(-1) = e^{-1} - 3$; $x_1(2) = e^2 - 8$

Problem 2 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -(\pi^2 + 10^{-4}) & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 10^{-4}\sin(\pi) \end{pmatrix}, \quad 0 < t < 1$$

BCs : $x_1(0) = x_1(1) = 0$

These two problems have smooth solutions and their coefficient matrices vary slowly in the specified intervals. Nevertheless, since problem 2 is an almost singular BVP we may expect that they have some different behaviour when column scaling treatment is applied in numerical experiments.

Tables *3.1a*, *3.1b* and *3.1c* display some results of numerical experiments for problem 1. Here, *sc* stands for scaling parameter; *Ch, Ga, Cex, Eq* indicate the type of collocation points used, i.e. *Ch* and CEx are respectively indicating that Chebyshev and Chebyshev extrema points are used; *Ga* is Gauss Point and *Eq* stands for equi-spaced points.

Table 3.1a shows the condition number $\kappa(\boldsymbol{C})$ for global collocation matrix with various types and numbers of collocation points. The results demonstrate the superiority of the Gauss points over the others, in the sense that using Gauss points the collocation matrices without column scaling operation have the condition numbers $\kappa(\boldsymbol{C})$ smaller than to those using the others. However, the collocation method using Chebyshev points generate matrices in which their condition numbers are reasonably close to those using Gauss points. Nevertheless, it seems that if column scaling operation is applied to the collocation matrices then the collocation algorithm using Gauss points is a little bit better than those using Chebyshev points, in which using Gauss points with scaling parameter $sc = 1$ gives the similar results to those using Chebyshev points with scaling parameter $sc = 2$. Perhaps, the most notable observation from this table is that the collocation algorithm using equi-spaced points produces a collocation matrix which has a very large condition number $\kappa(\boldsymbol{C})$. For example, with 40 subintervals using equi-spaced points $\kappa(\boldsymbol{C})$ is in order 10^{13} while using Gauss and Chebyshev points they are in order 10^{4} . In this case, although the column scaling operation is able to reduce the condition numbers, they are still reasonably large.

Table 3.1b and *Table 3.1c* show some results of numerical experiments for piecewise collocation method using Chebyshev zeros and Gauss points respectively. These results indicate that in most cases we need to increase the scaling parameter *sc* if the number of subintervals *w* increases. Increasing the number of subinterval and number of collocation points will result in a need for larger scaling parameters. In comparing *Table 3.1b* and *Table 3.1c*, the results clearly show that significant reduction are made in both case. Further more, it is observed that collocation at the Chebyshev points need larger scaling parameters to reduce the condition number

$\kappa(C)$ compared to those using Gauss points. It is also observed that without column scaling operation the condition numbers for Gauss collocation points are smaller than those collocation at Chebyshev points.

For *problem* 1 it is observed that reducing the condition number is not necessarily reducing the global error. It is important to realise that truncation error usually dominates roundoff error when one is using collocation methods for solving BVPs. However, numerically, we often, but not always, encounter cases where reducing the condition number may result in improving the accuracy of the collocation solutions.

Problem 2 which is an almost singular BVP exhibits some interesting numerical results in *Table 3.2*. As shown in this table, the results indicate that employing column scaling may result in not only reducing the condition number $\kappa(C)$ but also improving the accuracy of the solution.

Though it is still not clear how to choose a sensible scaling parameter *sc* without calculating the condition number of the collocation matrix, the numerical results indicate that in all cases taking *sc* = 1 or *sc* = 2 will result in reducing the condition number. In addition, *Table 3.2* shows that in most cases if *sc* is taken to be one or two the accuracy improves by a small but significant amount. This suggests that it is reasonable to take either *sc* = 1 or *sc* = 2 if one simply wants to employ scaling on collocation matrices without doing massive computation task.

Table 3.1a

(Condition Number – Global Representation)

No of Collocation Points

	Sc	*q* → 3	5	7	10	12	15	20	30	40
Ch	0	6.183e+01	1.864e+02	4.092e+02	1.036e+03	1.703e+03	3.180e+03	7.239e+03	2.358e+04	5.503e+04
	1	5.984e+01	1.138e+02	1.775e+02	3.107e+02	4.232e+02	6.306e+02	1.077e+03	2.355e+03	4.150e+03
	2	7.700e+01	1.062e+02	1.252e+02	1.551e+02	1.748e+02	2.174e+02	3.476e+02	7.027e+02	1.193e+03
	3	1.610e+02	2.379e+02	3.583e+02	6.277e+02	8.585e+02	1.279e+03	2.275e+03	5.223e+03	9.569e+03
	4	5.542e+02	1.231e+03	2.394e+03	5.561e+03	8.816e+03	1.578e+04	3.434e+04	1.079e+05	2.571e+05
Ga	0	4.825e+01	1.183e+02	2.334e+02	5.354e+02	8.429e+02	1.512e+03	3.309e+03	1.039e+04	2.385e+04
	1	5.206e+01	8.540e+01	1.214e+02	1.904e+02	2.454e+02	3.449e+02	5.528e+02	1.133e+03	1.936e+03
	2	8.467e+01	1.250e+02	1.791e+02	3.026e+02	4.107e+02	6.189e+02	1.100e+03	2.544e+03	4.773e+03
	3	2.671e+02	5.462e+02	9.890e+02	2.056e+03	3.057e+03	5.044e+03	9.831e+03	2.729e+04	5.751e+04
	4	9.805e+02	2.905e+03	6.857e+03	1.913e+04	3.321e+04	6.644e+04	1.666e+05	6.301e+05	1.726e+06
CEx	0	9.666e+01	2.680e+02	5.693e+02	1.397e+03	2.278e+03	4.214e+03	9.518e+03	3.079e+04	7.165e+04
	1	8.131e+01	1.469e+02	2.261e+02	3.898e+02	5.305e+02	7.876e+02	1.342e+03	2.928e+03	5.152e+03
	2	9.259e+01	1.216e+02	1.418e+02	1.740e+02	1.965e+02	2.563e+02	4.168e+02	8.546e+02	1.457e+03
	3	1.707e+02	2.372e+02	3.122e+02	5.784e+02	8.106e+02	1.264e+03	2.352e+03	5.572e+03	1.028e+04
	4	5.087e+02	1.059e+03	1.965e+03	4.493e+03	7.124e+03	1.316e+04	3.090e+04	1.087e+05	2.634e+05
Eq	0	7.129e+01	2.853e+02	1.240e+03	8.726e+03	3.812e+04	4.498e+05	1.669e+07	2.485e+10	3.414e+13
	1	7.378e+01	2.045e+02	6.650e+02	3.584e+03	1.337e+04	1.216e+05	3.315e+06	3.333e+09	3.461e+12
	2	1.014e+02	2.307e+02	6.322e+02	3.577e+03	1.241e+04	8.928e+04	2.511e+06	2.196e+09	2.026e+12
	3	2.839e+02	7.871e+02	2.519e+03	1.835e+04	7.326e+04	5.553e+05	1.649e+07	1.634e+10	1.620e+13
	4	1.049e+03	4.279e+03	1.631e+04	1.455e+05	5.766e+05	4.974e+06	1.807e+08	2.154e+11	2.485e+14

Table 3.1b

(Condition number -- piecewise representation -- Chebyshev Points)

w/sc	3	5	7	10	12	15
5/0	4.098e+02	1.448e+03	3.580e+03	9.709e+03	1.634e+04	3.111e+04
1	1.492e+02	3.470e+02	6.444e+02	1.282e+03	1.837e+03	2.866e+03
2	1.092e+02	1.705e+02	2.375e+02	3.448e+02	4.196e+02	7.330e+02
3	1.376e+02	1.636e+02	1.839e+02	2.082e+02	2.630e+02	4.193e+02
4	2.186e+02	2.314e+02	4.682e+02	1.171e+03	1.907e+03	3.512e+03
10/0	1.308e+03	4.859e+03	1.221e+04	3.346e+04	5.648e+04	1.078e+05
1	4.394e+02	1.109e+03	2.131e+03	4.338e+03	5.955e+03	9.351e+03
2	1.792e+02	3.092e+02	4.549e+02	6.908e+02	8.135e+02	1.056e+03
3	1.513e+02	1.926e+02	2.264e+02	2.677e+02	2.822e+02	3.187e+02
4	2.144e+02	2.325e+02	2.660e+02	6.826e+02	1.123e+03	2.089e+03
20/0	4.548e+03	1.736e+04	4.403e+04	1.179e+05	1.994e+05	3.812e+05
2	5.357e+02	9.719e+02	1.482e+03	2.313e+03	2.896e+03	3.797e+03
3	2.480e+02	3.311e+02	4.061e+02	4.985e+02	5.504e+02	6.175e+02
4	2.367e+02	2.627e+02	2.812e+02	4.449e+02	7.397e+02	1.391e+03
5	3.645e+02	3.724e+02	9.113e+02	3.143e+03	6.045e+03	1.367e+04
32/0	1.096e+04	4.154e+04	1.057e+05	2.916e+05	4.934e+05	9.435e+05
3	5.128e+02	7.229e+02	9.075e+02	1.135e+03	1.263e+03	1.429e+03
4	2.817e+02	3.220e+02	3.505e+02	3.775e+02	5.977e+02	1.132e+03
5	3.879e+02	4.035e+02	6.402e+02	2.222e+03	4.285e+03	9.716e+03

Table 3.1c

(Condition number--piecewise representation--Gauss Points)

w/sc	3	5	7	10	12	15
5/0	2.461e+02	7.320e+02	1.660e+03	4.216e+03	6.915e+03	1.283e+04
1	1.004e+02	1.981e+02	3.344e+02	6.136e+02	8.502e+02	1.281e+03
2	8.588e+01	1.203e+02	1.543e+02	2.067e+02	2.418e+02	2.984e+02
3	1.189e+02	1.361e+02	2.276e+02	5.228e+02	8.127e+02	1.410e+03
4	1.981e+02	5.059e+02	1.321e+03	3.904e+03	6.926e+03	1.416e+04
10/0	7.898e+02	2.490e+03	5.796e+03	1.496e+04	2.469e+04	4.606e+04
1	2.925e+02	6.282e+02	1.108e+03	2.105e+03	2.955e+03	4.509e+03
2	1.350e+02	2.044e+02	2.774e+02	3.912e+02	4.689e+02	5.872e+02
3	1.300e+02	1.546e+02	1.738e+02	3.497e+02	5.550e+02	9.865e+02
4	1.975e+02	2.721e+02	7.281e+02	2.193e+03	3.920e+03	8.085e+03
20/0	2.743e+03	8.956e+03	2.114e+04	5.508e+04	9.113e+04	1.704e+05
2	3.918e+02	6.388e+02	9.044e+02	1.322e+03	1.608e+03	2.046e+03
3	2.040e+02	2.571e+02	2.996e+02	3.505e+02	4.287e+02	7.776e+02
4	2.153e+02	2.348e+02	4.424e+02	1.358e+03	2.449e+03	5.095e+03
5	3.475e+02	7.391e+02	2.603e+03	1.063e+04	2.231e+04	5.619e+04
32/0	6.611e+03	2.189e+04	5.195e+04	1.358e+05	2.250e+05	4.212e+05
3	4.126e+02	5.401e+02	6.437e+02	7.686e+02	8.375e+02	9.261e+02
4	2.498e+02	2.787e+02	3.372e+02	1.049e+03	1.903e+03	3.983e+03
5	3.675e+02	5.069e+02	1.798e+03	7.383e+03	1.552e+04	3.917e+04

Table 3.2

(Condition number & Actual Error -- Gauss Points)

w	q	sc →	0	1	2	3	4
8	3	κ(C)	2.011e+08	7.775e+07	3.830e+07	3.900e+07	6.222e+07
		error	2.2607276e-02	2.2607275e-02	2.2607276e-02	2.2607275e-02	2.2607275e-02
5	4	κ(C)	1.689e+08	5.602e+07	3.369e+07	4.291e+07	6.942e+07
		error	5.8773596e-04	5.8773594e-04	5.8773602e-04	5.8773592e-04	5.8773602e-04
5	5	κ(C)	2.773e+08	7.610e+07	3.887e+07	4.505e+07	7.039e+07
		error	7.5306217e-07	7.5309675e-07	7.5299208e-07	7.5291883e-07	7.5314650e-07
8	5	κ(C)	6.139e+08	1.593e+08	5.512e+07	4.471e+07	6.470e+07
		error	1.6222150e-08	1.6039261e-08	1.6195687e-08	1.6138884e-08	1.6073489e-08
8	7	κ(C)	1.414e+09	2.761e+08	7.294e+07	4.931e+07	6.639e+07
		error	1.5222268e-10	1.6232082e-10	1.3861712e-10	1.2780266e-10	9.2167607e-11
10	7	κ(C)	2.120e+09	4.077e+08	1.035e+08	5.254e+07	6.720e+07
		error	2.5969449e-11	3.3728575e-12	2.7202240e-11	3.2949199e-11	2.7739144e-11
5	8	κ(C)	8.802e+08	1.584e+08	5.472e+07	5.008e+07	7.224e+07
		error	1.5561996e-10	2.9401814e-11	4.6729731e-11	9.7475361e-11	8.1422868e-11
10	8	κ(C)	3.005e+09	5.155e+08	1.172e+08	5.487e+07	6.792e+07
		error	2.1915358e-11	1.8389290e-11	2.3828495e-11	3.2525094e-11	1.6675550e-12
5	10	κ(C)	1.583e+09	2.319e+08	6.545e+07	5.265e+07	7.296e+07
		error	2.7195135e-11	2.6083623e-11	2.2418511e-11	7.8483442e-11	8.2020613e-11
8	10	κ(C)	3.629e+09	5.172e+08	1.005e+08	5.470e+07	6.792e+07
		error	2.1026514e-10	2.7818636e-11	2.6207037e-11	1.1342660e-10	3.4949821e-12
5	12	κ(C)	2.594e+09	3.206e+08	7.631e+07	5.483e+07	7.347e+07
		error	1.6648372e-10	2.0892754e-10	2.4420910e-10	7.4883211e-11	1.5580515e-10
10	12	κ(C)	9.009e+09	1.081e+09	1.734e+08	6.250e+07	6.981e+07
		error	1.1209744e-10	7.0147887e-11	1.1179235e-10	9.8926645e-11	1.0296963e-10

3.3 Basic Structure of the Collocation Matrix

Let us reconsider the boundary value problem (3.1) and its separated boundary conditions (3.2). Note that, instead of considering the general boundary conditions of the form

$$\boldsymbol{B}_a\boldsymbol{x}(a) + \boldsymbol{B}_b\boldsymbol{x}(b) = \boldsymbol{\beta} \qquad \ldots(3.7)$$

where $\boldsymbol{B}_a$ and $\boldsymbol{B}_b \in \boldsymbol{R}^{n\times n}$ and $\boldsymbol{\beta} \in \boldsymbol{R}^n$, we particularly confine the boundary conditions to be the separated ones. This is particularly convenient, both theoretically and practically. In practice, many boundary value problems in application come with separated boundary conditions. Moreover, it is an advantageous to consider the boundary value problems with separated boundary conditions, since the structure of collocation matrix obtained will be in a simpler form than those from non-separated boundary conditions. Furthermore, as mentioned in chapter 2 a simple trick can be used to convert non-separated boundary conditions into a separated one.

In point of view of theoretical aspects, the separated boundary condition can be viewed as having allocated the increasing and decreasing modes such that to the left end a there correspond the decreasing modes while to the right boundary b there correspond to increasing ones. Thus the purely fast deceasing and purely fast increasing modes are well separated as far as the boundary conditions are concerned, and in turn this property can be utilised further when one is tempting to estimate the layer width (will be discussed in chapter 7).

We shall now examine some ways in setting up the collocation matrix. Let $\boldsymbol{x}_{wq}$ be the piecewise approximate solution as defined in equation (3.6). The collocation solution $\boldsymbol{x}_{wq}$ is then uniquely determined by $n(q+1)w$ coefficients $\boldsymbol{p}_{i[j]}$, where $\boldsymbol{p}_{i[j]}$ are vectors of the form $\boldsymbol{p}_{i[j]} = [p_{i1[j]} \quad p_{i2[j]} \quad \cdots \quad p_{in[j]}]^T$, $0 \le i \le (q+1)$. These coefficients are calculated from

- n boundary conditions,
- nqw collocation conditions and
- $n(w\text{-}1)$ continuity conditions.

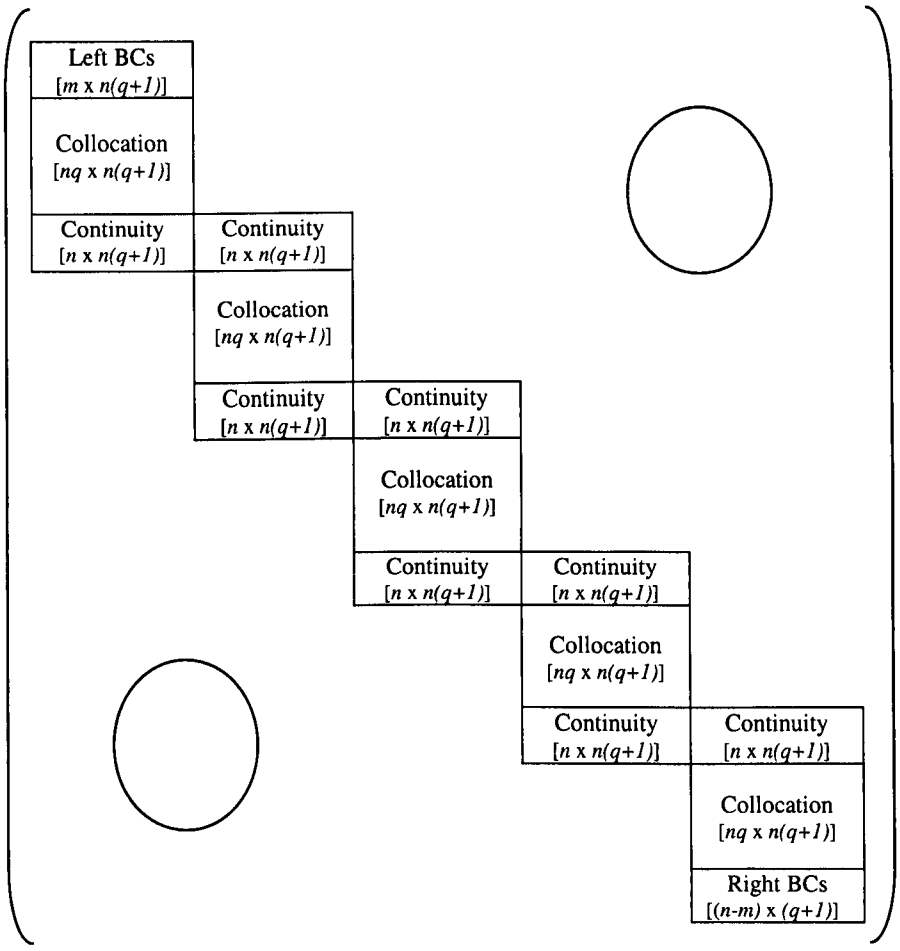
Arrange these equations in the order they arise when moving from left to right. That is,

1. the first equations are those from the left boundary conditions
2. then the collocation conditions at each point in the first subinterval
3. then the continuity condition at the first break point t_2
4. repeat steps (*2*) and (*3*) corresponding to appropriate subintervals, till the last break point t_w
5. for the last subinterval, set up the equations corresponding to the collocation points and followed by those from the right boundary conditions.

The resulting matrix consists of a number of rectangular blocks, each block is related to a subinterval. The structure of the matrix will look like

fig.1 Structure of the Collocation Matrix

(number of subintervals w = 5)



3.4 Block Matrix Representation

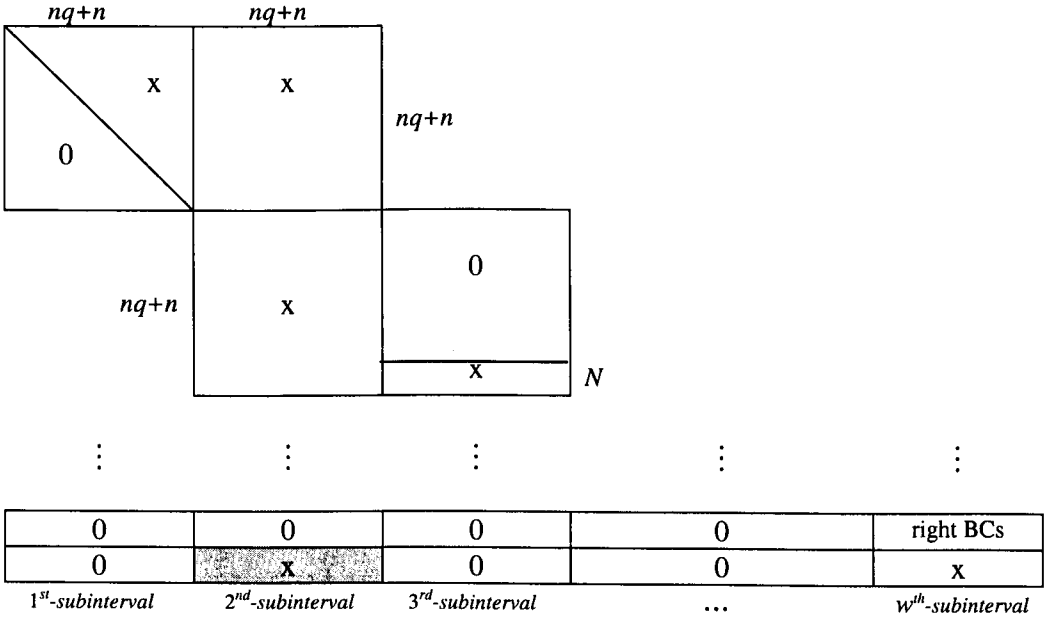
As described in previous section, discretisation of a boundary value problem lead to a system of linear equations $\boldsymbol{Cp} = \boldsymbol{g}$. Based on the fact that collocation matrix $\boldsymbol{C}$ is large and sparse with its nonzero elements concentrated near the main diagonal, we shall utilise the structure of the collocation matrix to reduce the amount of fill-in. Afterwards, an analysis of arithmetic cost and memory space needed will be discussed.

3.4.1 Reduction to Block Matrix Form

The full matrix can directly be solved using Gaussian elimination with partial pivoting to guarantee the stability. However, an improvement in terms of storage required and time consumed can be achieved by constructing an algorithm using the properties indicated by the following procedure

1. Set up the collocation matrix where the first equations are from collocation conditions in the first subinterval (instead of the left boundary conditions as in the previous section); the continuity conditions at the first break point t_2, then the collocation conditions in the second subinterval, etc. Finally the right boundary conditions followed by the left boundary conditions.
2. The Gaussian elimination with partial pivoting can be used to obtain the upper triangular matrix in the first block.
3. The non-zero elements in the last rows corresponding to the left boundary conditions can be eliminated using the elements of first block and as the result there will be non-zero elements in the next block associated with the second subinterval.

   The resulting matrix then looks like



Note that since we employ the Gaussian elimination with partial pivoting, certainly, the row interchanges include the rows corresponding to boundary conditions.

4. The procedure continues until the last block associated with w^{th} subinterval.
5. Finally we deploy backward substitution to obtain the solution of the system.

Note that, in implementation the above procedure is still in full matrix form. To make it more convenient and more efficient we will represent the collocation matrix using a 3-dimensional data structure, the first dimension refers to the block, while the others refer to the row and column of the a block. In the implementation we have w blocks, where each block is a $[n(q+1)+m \text{ x } 2n(q+1)]$ matrix as follows

1. The first block associated with the first and second subinterval consists of nq rows from collocation conditions, n rows from continuity conditions and m rows of the left boundary conditions. This block will look like

	$nq+n$	$nq+n$
nq	x	0
n	x	x
m	x	0

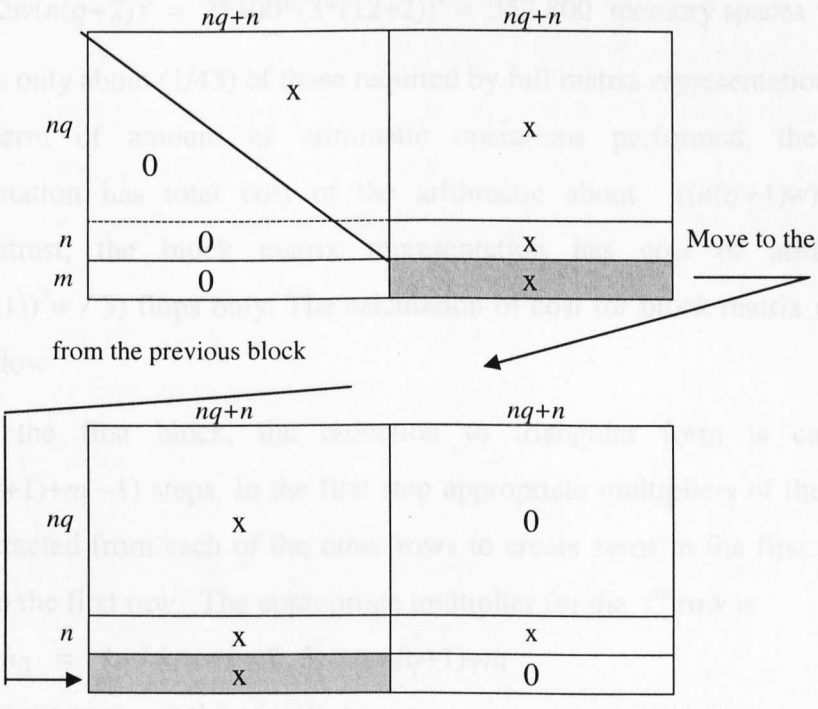
2. The second block associated with the second and third subinterval consists of nq rows from collocation conditions, n rows from continuity conditions.

	$nq+n$	$nq+n$
nq	x	0
n	x	x
m	0	0

3. The third block, fourth block, until the $(w\text{-}1)^{th}$ block is similar to the second block
4. The last block associated with the w^{th} subinterval consists of nq rows from collocation conditions, $(n\text{-}m)$ rows from the right boundary conditions and the rest is zeros. This last block will be in form of

	$nq+n$	$nq+n$
nq	x	0
$(n\text{-}m)$	x	0
m	0	0
m	0	0

The upper triangular form may be obtained by performing the Gaussian elimination with partial pivoting to the first block and then move the rows associated with the left boundary conditions to the second block. The result looks like



Finally, the solution can be found by employing backward substitution.

3.4.2 Analysis of Work and Amount of Memory Spaces

In implementation, the block matrix algorithm works quite well and its performance is very impressive, in the sense there is a massive saving in terms of memory spaces used and, in particular, reducing computation time compared to treatment as a full matrix. In some cases using the full matrix representation result in 'out of memory' while block matrix algorithm works smoothly. This is not surprising since the full matrix representation needs $(n(q+1)w)^2$ memory spaces, on the other hand for the block matrix it is roughly $2w(n(q+2))^2$. This means that if the number of collocation points q is large enough, the block matrix representation needs memory spaces about $(2/w)$ of those needed by full matrix representation. For example, a system with three differential equations, 12 collocation points in each subinterval and 100 subintervals, the full matrix representation needs

$$(n(q+1)w)^2 = (3*(12+1)*100)^2 = 15{,}210{,}000 \text{ memory spaces,}$$

while for the block matrix representation requires

$$2w(n(q+2))^2 = 2*100*(3*(12+2))^2 = 352{,}800 \text{ memory spaces}$$

which is only about (1/43) of those required by full matrix representation.

In term of amount of arithmetic operations performed, the full matrix representation has total cost of the arithmetic about $((n(q+1)w)^3 / 3)$ flops. By contrast, the block matrix representation has cost of arithmetic about $(2(n(q+1))^3 w / 3)$ flops only. The calculation of cost for block matrix representation is as follow

1. For the first block, the reduction to triangular form is carried out in $(n(q+1)+m-1)$ steps. In the first step appropriate multipliers of the first row are subtracted from each of the other rows to create zeros in the first column, apart from the first row. The appropriate multiplier for the i^{th} row is

$$m_{i1} = k_{i1} / k_{11}, \quad i = 2, 3, \ldots, n(q+1)+m \qquad \ldots(3.8)$$

Then we carry out the operations

$$k_{ij}^{(1)} = k_{ij} - m_{i1}k_{1j}, \quad j = 2, 3, \ldots, 2n(q+1). \quad i = 2, 3, \ldots, n(q+1)+m$$

$$g_i^{(1)} = g_i - m_{i1}g_1, \quad i = 2, 3, \ldots, n(q+1)+m \qquad \ldots(3.9)$$

2. The arithmetic cost in the first step are $(n(q+1)+m-1)$ divisions and
   $[(n(q+1)+m-1) + (2n(q+1)-1)(n(q+1)+m-1)] = [(n(q+1)+m-1)][(2n(q+1)]$ flops.
   Ignoring terms of order $(n(q+1)+m)$ or less the cost is about
   $$(n(q+1)+m-1)(2n(q+1)) \approx 2(n(q+1))^2 \text{ flops.}$$
3. For the second step, third step till the $(n(q+1)+m-1)^{th}$ step the costs are
   $$2(n(q+1)-1)^2,\ 2(n(q+1)-2)^2, \ldots, 2(2)^2$$
4. Thus the total cost for reducing to the triangular form is
   $$2(n(q+1))^2 + 2(n(q+1)-1)^2 + 2(n(q+1)-2)^2 + 2(2)^2 \approx 2(n(q+1))^3/3 \text{ flops}$$
5. Since there are w blocks, the total arithmetic cost of this algorithm is about
   $$\frac{2w}{3}(n(q+1))^3 \text{ flops}$$

Note that there are three other costs to consider. Firstly that of finding the largest pivot at each step, secondly that of physically interchanging the rows, and thirdly that of moving the last m-rows to the next block. These costs do not involve any arithmetic, but time is required to make comparison in order to get the largest pivot. In interchanging and moving the rows the time is needed to fetch and store numbers.

To illustrate how much the arithmetic cost could affect the performance of the algorithm, let us examine a system having 3 differential equations, to be solved using 4 collocation points in each 50 subintervals. The full matrix representation costs about

$$(3*(4+1)*50)^3 / 3 = 140{,}625{,}000 \text{ flops,}$$

while the block matrix representation costs about

$$2*(3*(4+1))^3*50 / 3 = 112{,}500 \text{ flops.}$$

This means, roughly, the block matrix algorithm may theoretically perform (140,625,000 / 112,500) = 1250 times faster than full matrix algorithm. In the next section, some results of numerical observations are presented.

Comparing the full and block matrices performance, theoretically we can say that for q sufficiently large the block matrix representation only requires $(2/w)$ than to

those required by full matrix representation. In the meantime, in term of arithmetic cost the block matrix representation will only cost $(\frac{2}{w^2})$ than to those the full one.

As mentioned before, the full matrix representation has arithmetic cost about $(n(q+1)w)^3/3$ flops. In this case, if the number of subintervals w is doubled then the arithmetic cost will increase by factor 2^3. Mean while, the block matrix representation having total cost about $\frac{2w}{3}(n(q+1))^3$ flops, doubling the number of subintervals w will result in increasing the arithmetic cost by factor two only. In other words, if we double w then the arithmetic cost will double as well. This result is very important since in developing the adaptive mesh selection algorithms, we shall carry out a lot numerical computations in which the number of subintervals w will vary substantially.

3.5 Computational Illustrations

We consider two illustrative examples as follows

Problem 3:

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} -1/(2t) & 2/(t^3) \\ -t/2 & -1/(2t) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad 0 < t < 1$$

The boundary conditions are

$x_1(0) = 0$ and $x_1(1) = 0$

Problem 4:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ (t^4+14t^3+49t^2+32t+2)e^t \end{pmatrix}, \qquad 0 < t < 1$$

with boundary conditions

$x_1(0) = x_2(0) = x_1(1) = x_2(1) = 0$

Problem 3 has been chosen since its coefficient matrix has some components which vary rapidly if t close to left hand boundary. For this problem, by increasing the number of subintervals w the first columns of collocation matrix will have large

values. On the other hand, the last columns of the matrix will have small values. Meanwhile, *problem* 4 of order 4 is intended to make comparison more straightforward, i.e. to observe the effect of doubling the size of problems, regardless the problems themselves are different.

In the following tables, T indicates computation time in millisecond (*ms*), while Qt stands for quotient of computation time of two consecutive computation. As can be seen in the tables, we always double the number of subintervals w so that we may observe the effect of doubling the number of subintervals.

The results for *problem* 3 are given in *Table* 3.3. They show that for full matrix representation, as can be seen on column under heading Qt_1, if the number of subintervals w doubles then the factor Qt_1 gradually tend to 8. In the meantime, for block matrix representation it is clear that values on column Qt_2 is about 2.

Similar observation can be seen in *Table* 3.4 displaying results for *problem* 4. By looking at columns Qt_1 and Qt_2, it is clear that Qt_1 and Qt_2 tend to 8 and 2 respectively. It is also notable that for large w the full matrix representation needs massive computation time and we can say that it is not sensible to use it in real applications. On the other hand, the block matrix performance is very impressive, for example from *Table* 3.4, let us take a look at $w = 160$, $q = 8$, for full matrix its computation time is 10019460 *ms* $\approx$ 2 *hrs* 46 *mins* 4 *secs*, while for the block representation it is only 8310 *ms* $\approx$ 8.3 *secs*

Comparing *Table* 3.3 and *Table* 3.4 and looking at the rows having the same w and q, it is observed that if w is large enough the arithmetic cost also increases by factor 8 and 2 for full matrix representation and block matrix representation respectively. For example, let us take a look at the rows of *Table* 3.3 and *Table* 3.4 in which number of collocation points $q = 8$ and number of subintervals $w = 160$; for full time representation the quotient of time is (10019460 / 1250490) $\approx$ 8, while for block matrix representation it is (8310/4580) = 1.8

Table 3.3

(boundary value problem 3)

		full matrix		block matrix	
w	q	T_1	Qt_1	T_2	Qt_2
5	3	100	-	90	-
10	3	190	1.900	170	1.889
20	3	520	2.737	340	2.000
40	3	2360	4.538	680	2.000
80	3	15120	6.407	1360	2.000
160	3	112410	7.435	2700	1.985
5	4	110	-	100	-
10	4	240	2.182	190	1.900
20	4	760	3.167	390	2.053
40	4	4110	5.408	760	1.949
80	4	28360	6.900	1520	2.000
160	4	216920	7.649	3040	2.000
5	5	120	-	110	-
10	5	290	2.417	210	1.909
20	5	1110	3.828	440	2.095
40	5	6660	6.000	850	1.932
80	5	47960	7.201	1710	2.012
160	5	373050	7.778	3430	2.006
5	8	180	-	150	-
10	8	540	3.000	290	1.933
20	8	2980	5.519	580	2.000
40	8	20680	6.940	1140	1.966
80	8	158280	7.654	2290	2.009
160	8	1250490	7.900	4580	2.000
5	10	230	-	190	-
10	10	820	3.565	350	1.842
20	10	5130	6.256	690	1.971
40	10	36950	7.203	1370	1.986
80	10	287420	7.779	2720	1.985
160	10	2277920	7.925	5420	1.993

Table 3.4
(boundary value problem 4)

		full matrix		block matrix	
w	q	T_1	Qt_1	T_2	Qt_2
5	2	130	-	120	-
10	2	310	2.385	230	1.917
20	2	1140	3.677	470	2.043
40	2	6700	5.877	920	1.957
80	2	48120	7.182	1840	2.000
160	2	373520	7.762	3680	2.000
5	3	150	-	140	-
10	3	440	2.933	270	1.929
20	3	2220	5.045	540	2.000
40	3	14810	6.671	1090	2.019
80	3	111740	7.545	2170	1.991
160	3	879640	7.872	4330	1.995
5	4	200	-	160	-
10	4	670	3.350	320	2.000
20	4	3940	5.881	630	1.969
40	4	28030	7.114	1250	1.984
80	4	216340	7.718	2510	2.008
160	4	1713210	7.919	5000	1.992
5	6	310	-	210	-
10	6	1490	4.806	410	1.952
20	6	9960	6.685	810	1.976
40	6	74870	7.517	1640	2.025
80	6	589480	7.873	3290	2.006
160	6	4694010	7.963	6580	2.000
5	8	490	-	270	-
10	8	2870	5.857	530	1.963
20	8	20470	7.132	1060	2.000
40	8	157700	7.704	2090	1.972
80	8	1249620	7.924	4160	1.990
160	8	10019460	8.018	8310	1.998

chapter 4

Numerical Evaluation of the Error Estimates

4.1 Introduction

This chapter is mainly concerned with numerical investigation of error estimates for the collocation solution of linear system of differential equations. Our main goal is to obtain a reasonable error estimate to be used directly in estimating the number of subintervals needed in an adaptive mesh selection algorithm. Hence, these error estimates should be inexpensive computationally and they can easily be implemented. Some error estimates based on consideration of the linear operator involved and on the residuals will be examined in some details.

Firstly, let us state precisely the form of problem considered and notations used. We shall consider the linear system of n first order differential equations of the form

$$x' - A(t)\,x(t) = y(t), \qquad a < t < b \qquad \ldots(4.1)$$

where $x(t)$ and $y(t)$ are n-dimensional vectors with components $x_i(t)$ and $y_i(t)$, $1 \le i \le n$, respectively. $A(t)$ is an $(n \times n)$ matrix valued function.

The system of ordinary differential equations (4.1) is furnished by m and $(n-m)$ associated homogeneous boundary conditions at the left and right boundaries respectively, for some positive constant $m < n$.

The equation (4.1) may be written in operator form

$$(D - M)\,x = y \qquad \ldots(4.2)$$

where $x \in X$ and $y \in Y$ in which X and Y are suitable Banach space. Here D denoting the differentiation operator $Dx(t) = dx/dt$ is the principal part of the operator, and we assume that both operators D and $(D - M)$ with the associated conditions are invertible, i.e. D^{-1} and $(D - M)^{-1}$ exist.

Suppose x_{wq} denote the collocation solution found by collocating at q points in each subinterval using the partition :

$$\pi_w : a = t_1 < t_2 < t_3 < \ldots < t_w < t_{w+1} = b.$$

where w indicates the number of subintervals. In each subinterval $[t_k, t_{k+1}]$, $1 \le k \le w$, the q collocation points are uniquely determined by

$$\xi_{ik} = t_k + \frac{t_{k+1} - t_k}{2}(1 + \xi_i^*), \quad \text{where } i = 1, 2, \ldots, q.$$

where $\{ \xi_i^* \}$, $1 \le i \le q$, are the chosen reference points in interval [-1,1].

Let X_{wq} and Y_{wq} be subspaces of X and Y respectively and φ_{wq} is a projection $Y \rightarrow Y_{wq}$. The approximate solution x_{wq} taken in a subspace X_{wq} is found by applying an interpolatory projection φ_{wq} to the equation (4.2) with x_{wq} substituted for x, that is

$$\varphi_{wq}(Dx_{wq} - Mx_{wq} - y) = 0 \qquad \ldots(4.3)$$

By assuming $\varphi_{wq} Dx_{wq} = Dx_{wq}$ that is that the operator D restricted to X_{wq} establishes a bijection between X_{wq} and Y_{wq} so that the equation (4.3) may be simplified as follows

$$\varphi_{wq} Dx_{wq} - \varphi_{wq} Mx_{wq} = \varphi_{wq} y$$

$$(\varphi_{wq} D - \varphi_{wq} M)x_{wq} = \varphi_{wq} y$$

or

$$(D - \varphi_{wq} M)x_{wq} = \varphi_{wq} y \qquad \ldots(4.4)$$

To be used in constructing the error estimate, it is convenient here to define the compact operator $\boldsymbol{K}$ by

$$\boldsymbol{K} = \boldsymbol{M}\boldsymbol{D}^{-1} \qquad \ldots(4.5)$$

As described by Anselone in [5], the inverse of $(\boldsymbol{I} - \boldsymbol{K})$ where $\boldsymbol{I}$ the identity operator on space $\boldsymbol{X}$ can be expressed in terms of the so called resolvent operator $(\boldsymbol{I} - \boldsymbol{K})^{-1}\boldsymbol{K}$ as follows

$$(\boldsymbol{I} - \boldsymbol{K})^{-1} = \boldsymbol{I} + (\boldsymbol{I} - \boldsymbol{K})^{-1}\,\boldsymbol{K} \qquad \ldots(4.6)$$

4.2 Behaviour of the Collocation Matrix Norms

For solving single higher order differential equations, Ahmed and Wright in [2] introduced certain matrices $\boldsymbol{Q}_{wq}$ developed from Ahmed's thesis [1] and the properties of those matrices were discussed in some detail. They also suggested some efficient ways in defining and constructing such matrix $\boldsymbol{Q}_{wq}$. Firstly they consider two vectors, i.e. the evaluation vector $\boldsymbol{\Phi}_q : \boldsymbol{Y} \rightarrow \boldsymbol{R}^q$ to give a vector consisting of the values of a function at the collocation points; and an additional evaluation vector $\boldsymbol{\Phi}_s : \boldsymbol{X} \rightarrow \boldsymbol{R}^s$ relating to a set of evaluation points $\{\zeta_i\}$, $1 \leq i \leq s$ for some s. Secondly, for convenience they define those evaluation vectors as $\boldsymbol{x}_0 = \boldsymbol{\Phi}_s \boldsymbol{x}_{wq}$ and $\boldsymbol{y}_0 = \boldsymbol{\Phi}_q \boldsymbol{y}$, both based on the collocation points. The matrix $\boldsymbol{Q}_{wq}$ can then be written as

$$\boldsymbol{x}_0 = \boldsymbol{Q}_{wq}\,\boldsymbol{y}_0 \qquad \ldots(4.7)$$

that is the matrix $\boldsymbol{Q}_{wq}$ relates the values of the right hand side and the approximate solution at the collocation points. The equation (4.7) can be regarded as the definition of $\boldsymbol{Q}_{wq}$. Under some assumptions and conditions it has been shown in [2] that by keeping q to be fixed the norm of matrix $\boldsymbol{Q}_{wq}$ converges to the norm of $(\boldsymbol{D} - \boldsymbol{M})^{-1}$ as w tends to infinity.

The results presented in [2] confirm the intuitively reasonable notion that a collocation matrix inverse can give an idea of the error magnification inherent in the collocation process, justifying the use of $\|Q_{wq}\|$ as estimates of $\|(D - M)^{-1}\|$. The details of matrices Q_{wq} will not be discussed further here since our motivation is to see the idea of using those matrices in constructing the error estimates.

For solving the first order systems of boundary value problems using global and piecewise representation, here we have carried out further work on observing the convergence of $\|Q_{wq}\|$ based on numerical investigations. The usefulness of $\|Q_{wq}\|$ as estimates of $\|(D - M)^{-1}\|$ is observed. Moreover, in some cases the $\|Q_{wq}\|$ may settle down early. Briefly, the numerical results which are not presented here indicate that

$$\|Q_{wq}\| \rightarrow \|(D-M)^{-1}\|, \quad \text{as } w \rightarrow \infty, \ q \text{ fixed,}$$

and

$$\|Q_{wq}\| \rightarrow \|(D-M)^{-1}\|, \quad \text{as } q \rightarrow \infty, \ w \text{ fixed}$$

However, since the evaluation of $\|Q_{wq}\|$ involves the inversion of a large matrix generated by the collocation process, certainly it is a massive computational task and the estimate is very expensive. This in turn suggests that one would expect considerable cancellation in using this estimate for adaptive mesh selection algorithms.

4.3 The Residual

Having carried out the collocation process, suppose an approximate solution x_{wq} satisfying the differential equation (4.1) and its associated boundary conditions has been found. The residual of the approximate solution x_{wq} denoted by r_{wq} is defined as

$$r_{wq} = (D - M)\, x_{wq} - y \qquad \ldots(4.8)$$

Using the following simple algebraic manipulation and applying the equation (4.4) we have relationship

$$\begin{aligned} \boldsymbol{r}_{wq} &= (\boldsymbol{D} - \boldsymbol{M} + \varphi_{wq}\boldsymbol{M} - \varphi_{wq}\boldsymbol{M})\boldsymbol{x}_{wq} - \boldsymbol{y} \\ &= (\boldsymbol{D} - \varphi_{wq}\boldsymbol{M})\boldsymbol{x}_{wq} - \boldsymbol{M}\boldsymbol{x}_{wq} + \varphi_{wq}\boldsymbol{M}\boldsymbol{x}_{wq} - \boldsymbol{y} \\ &= \varphi_{wq}\boldsymbol{y} - \boldsymbol{y} + \varphi_{wq}\boldsymbol{M}\boldsymbol{x}_{wq} - \boldsymbol{M}\boldsymbol{x}_{wq} \\ &= (\varphi_{wq} - \boldsymbol{I})\boldsymbol{y} + (\varphi_{wq} - \boldsymbol{I})\boldsymbol{M}\boldsymbol{x}_{wq} \\ &= (\varphi_{wq} - \boldsymbol{I})(\boldsymbol{M}\boldsymbol{x}_{wq} + \boldsymbol{y}) \qquad \ldots(4.9) \end{aligned}$$

From the equation (4.9), it is clear that the residual $\boldsymbol{r}_{wq}$ is the error in the interpolation of the function $(\boldsymbol{M}\boldsymbol{x}_{wq} + \boldsymbol{y})$, hence this enables us to examine its behaviour using some properties of the interpolation theory.

The Cauchy remainder theorem for polynomial interpolation described in detail in Davis [20] states that in interpolating a continuous function $f(t)$ over the closed interval $[a,b]$ based on $(n+1)$ interpolation points : $a \le t_0 < t_1 < \ldots < t_n \le b$, providing $f^{(n+1)}(t)$ exists at each point of (a,b), the remainder $R_n(t)$ then satisfies

$$R_n(t) = \frac{(t-t_0)(t-t_1)\ldots(t-t_n)}{(n+1)!} f^{(n+1)}(\xi)$$

where the point ξ depend upon $t, t_0, t_1, \ldots, t_n$ and function f.

As we can see from the Cauchy theorem, the remainder $R_n(t)$ splits into two parts. The first part, the factor $\frac{1}{(n+1)}\prod_{i=0}^{n}(t-t_i)$, is independent of function $f(t)$, but depends upon the interpolation points. The second part, $f^{(n+1)}(\xi)$, depends upon function interpolated, but is independent of the manner in which the interpolation is carried out. The second part tells us that the remainder $R_n(t)$ is affected by the smoothness of $f(t)$.

Looking at the function $(\boldsymbol{M}\boldsymbol{x}_{wq} + \boldsymbol{y})$, if we assume that the coefficient $\boldsymbol{A}(t)$ in differential equation (4.1) is sufficiently smooth and using the fact that $\boldsymbol{x}_{wq}$ itself is a piecewise polynomial function and if $\boldsymbol{y}(t)$ is assumed to be smooth enough as

well, then r_{wq} should be well approximated by a piecewise polynomial found by interpolation using additional points in each subinterval. Furthermore the residual r_{wq} will have a factor

$$\prod_{i=1}^{q}(t-\xi_{ik})$$

in the k^{th}-subintervals.

This suggests that the residual may split into two parts, the first part called the principal part of the residual consists of a polynomial which is interpolating the residual; the second part is the error in the interpolation. Let r_{wq}^{*} denotes the principal part of the residual r_{wq} and r_{wq}^{**} denotes the error term in the interpolation we then have

$$r_{wq} = r_{wq}^{*} + r_{wq}^{**}$$

From this point, there are several ways to construct a polynomial interpolation for the principal part of residual r_{wq}^{*}. At least there are two considerations which should be taken into account in constructing such a polynomial. Firstly, we wish that the integration process of the chosen polynomial interpolation can be carried out in a simple way, since we need to carry out the integrations to form $D^{-1}r_{wq}^{*}$ (discussed in the following section). Secondly, the polynomial interpolation should be sufficiently accurate even for the cases where the number of interpolation points is small. For the first consideration, it is convenient to represent r_{wq}^{*} as a Chebyshev series since the integration process can be carried out easily. The second consideration relates to the way of choosing the interpolation points. Here, since the residual is zero at the collocation points, one might consider to choose points between the collocation points, so as to get close to the extrema of the residual. If the end points of the subintervals are not the collocation points, one

might also consider to take them as additional interpolation points. Based on those considerations, it is convenient to use $(2q+1)$ interpolation points determined by

$t_i = \cos(\frac{i-1}{2q}\pi)$, $i = 1, 2, \ldots, (2q+1)$; q is the number of collocation points, and then express r^*_{wq} as

$$r^*_{wq}(t^*) = \sum_{j=0}^{2q} c_j T_j(t^*), \qquad \ldots(4.10)$$

where $T_j(t^*)$ are Chebyshev polynomials and t^* denotes a local variable in each subinterval.

To examine how well the polynomial interpolation based on equation (4.10) performs in computation, we observe some results of numerical experiments using a number of problems having different nature.

In tables (4.1), (4.2) and (4.3) the capital letters R, R^*, R^{**} denote values of the norms, i.e. $R = \|r\|$, $R^* = \|r^*_{wq}\|$, $R^{**} = \|r^{**}_{wq}\|$; the letters C and G standing for Chebyshev and Gauss indicate whether Chebyshev zeros or Gauss points have been used as the collocation points. As usual, q and w indicate the number of collocation points and the number of subintervals respectively.

For the first illustration, let us consider the following problem

Example 1 :

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -t & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -2t^3 - 6 + te^t \end{pmatrix}, \quad -1 < t < 2$$

constrained by the boundary conditions :

$x_1(-1) = e^{-1} + 1$. $x_2(-1) = e^{-1} - 3$;

$x_1(2) = e^2 - 8$

This boundary value problem is chosen since the coefficient matrix $A(t)$ is smooth and the non-homogeneous term can be well approximated by polynomial interpolation. Hence, for this example we would expect the residual $r(t)$ will be well approximated by polynomial interpolation r_{wq}^{*} reflected by small interpolation error norm $\| r_{wq}^{**} \|$.

As shown in *Table 4.1* below, using two Chebyshev zeros per subinterval the interpolation error r_{wq}^{**} reduces smoothly when the interval size reduces. The results show that in all cases the interpolation error r_{wq}^{**} is relatively much smaller than interpolation value r_{wq}^{*} indicating the residual is well approximated by r_{wq}^{*}.

Similar observation is shown for $q = 3$. It is notable from this table that for $q = 3$ (increasing by 1 point only), the interpolation error dramatically reduces and for $w = 40$ we can say that in practise $\| r_{wq}^{**} \|$ is equal to zero. An interpolation error converging quickly to zero is desirable, since we need it in developing a cheap error estimate.

Table 4.1

(*Chebyshev zeros*)

w -->	3	5	10	15	20	30	40
q = 2							
R	4.3841307e-01	1.5928095e-01	4.0367927e-02	1.8028252e-02	1.0164341e-02	4.5273507e-03	2.5492555e-03
R*	4.3793652e-01	1.5923643e-01	4.0366326e-02	1.8028031e-02	1.0164288e-02	4.5273434e-03	2.5492538e-03
R**	5.1845056e-04	4.8827495e-05	1.7652092e-06	2.4411925e-07	5.9370546e-08	8.0130628e-09	1.9251040e-09
q = 3							
R	2.7402297e-02	6.7546930e-03	9.2418064e-04	2.8481734e-04	1.2258035e-04	3.7041080e-05	1.5778951e-05
R*	2.7401488e-02	6.7546656e-03	9.2418039e-04	2.8481735e-04	1.2258036e-04	3.7041080e-05	1.5778951e-05
R**	9.9127848e-07	3.3633036e-08	3.0401111e-10	1.8684450e-11	2.5615681e-12	1.5484255e-13	2.5313859e-14

The following example is intended to observe the performance of the polynomial interpolation (4.10) when the right hand side $y(t)$ and matrix $A(t)$ of the

model problem (4.1) are not polynomials. Furthermore, the coefficient matrix $A(t)$ will vary faster if t is close to the left boundary. The problem is

Example 2 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -1/t \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ (8/(8-t^2))^2 \end{pmatrix}, \qquad 0 < t < 1$$

accompanied with boundary conditions :

$x_2(0) = 0$

$x_1(1) = 0$

Unlike *example 1*, in *example 2* the coefficient matrix $A(t)$ is not a simple polynomial matrix.

As shown in *Table 4.2* on the following page, by comparing the values of R^*s and $R^{**}s$, the interpolation error is reasonable. However, by carefully comparing the results in this table to those in *Table 4.1*, it is very clear that r^*_{wq}'s presented in *Table 4.2* are less accurate than those in *Table 4.1*, for example from *Table 4.2* we take a look at R (or R^*) having order of accuracy 10^{-s} for some integer $s > 0$, we can see that the order of R^{**} is greater or equal to 10^{-2s}. On the other hand, by looking at R with similar order in *Table 4.1* we observe that higher order is obtained in which the order of R^{**} is less than 10^{-2s}.

Regarding the type of collocation points, we observe that for small q, when Gauss points are used as the collocation points the interpolation error R^{**} produced by the interpolation is marginally smaller than those using Chebyshev zeros. By contrast, for larger q, i.e. $q = 4$, we found that using Gauss collocation points gives less accurate approximation than using Chebyshev zeros, though the difference in accuracy is marginal. This clearly indicates that using Chebyshev zeros may gives competitive results.

Table 4.2

(Chebyshev and Gauss points)

W	3	5	10	15	20	30	40
2C							
R	6.6668344e-03	2.5309352e-03	6.6156370e-04	2.9871119e-04	1.6938879e-04	7.5904320e-05	4.3027545e-05
R*	6.6649100e-03	2.5307770e-03	6.6155845e-04	2.9871048e-04	1.6938862e-04	7.5904297e-05	4.3027545e-05
R**	6.8745250e-06	2.1464742e-06	5.1021116e-07	2.2477360e-07	1.2605108e-07	5.5901756e-08	3.1421035e-08
2G							
R	8.9179692e-03	3.3867337e-03	8.8553475e-04	3.9986827e-04	2.2675765e-04	1.0161377e-04	5.7395882e-05
R*	8.9160449e-03	3.3865755e-03	8.8552950e-04	3.9986756e-04	2.2675748e-04	1.0161375e-04	5.7395877e-05
R**	6.7733724e-06	2.1038636e-06	4.9892835e-07	2.1970792e-07	1.2319160e-07	5.4627699e-08	3.0703752e-08
3C							
R	3.1874249e-04	7.6913089e-05	1.0630613e-05	3.2712877e-06	1.4067561e-06	4.2493404e-07	1.8101575e-07
R*	3.1874392e-04	7.6913108e-05	1.0630613e-05	3.2712877e-06	1.4067561e-06	4.2493404e-07	1.8101575e-07
R**	2.2628941e-08	2.6885739e-09	1.6120582e-10	3.1592284e-11	9.9684019e-12	1.9650607e-12	6.2160635e-13
3G							
R	5.1493314e-04	1.2445751e-04	1.6993276e-05	5.1897122e-06	2.2231427e-06	6.6893156e-07	2.8439878e-07
R*	5.1493462e-04	1.2445753e-04	1.6993276e-05	5.1897122e-06	2.2231427e-06	6.6893156e-07	2.8439878e-07
R**	2.3632020e-08	2.8086347e-09	1.6845818e-10	3.3016364e-11	1.0418039e-11	2.0535213e-12	6.4933853e-13
4C							
R	1.7495632e-05	2.6385067e-06	1.8616282e-07	3.8345555e-08	1.2392959e-08	2.5009590e-09	7.9989082e-10
R*	1.7495650e-05	2.6385069e-06	1.8616282e-07	3.8345555e-08	1.2392959e-08	2.5009591e-09	7.9989081e-10
R**	2.2943182e-10	2.2587793e-11	1.2498627e-12	2.4130636e-13	7.5996109e-14	1.4794067e-14	5.7388310e-15
4G							
R	3.1850319e-05	4.7404210e-06	3.3082205e-07	6.7881663e-08	2.1896069e-08	4.4100346e-09	1.4090773e-09
R*	3.1850302e-05	4.7404209e-06	3.3082205e-07	6.7881662e-08	2.1896069e-08	4.4100346e-09	1.4090772e-09
R**	2.3631290e-10	2.3270295e-11	1.2875823e-12	2.4822248e-13	7.7821640e-14	1.5649680e-14	7.2245647e-15

As the last illustration, we examine the following problem taken from Russell [44]. The problem is

Example 3 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 400 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 400\cos^2(\pi t) + 2\pi^2\cos(2\pi t) \end{pmatrix}, \qquad 0 < t < 1$$

The boundary conditions are

$x_1(0) = 0$ and

$x_1(1) = 0$

Unlike two previous problems, here the matrix $A(t)$ is constant, though one of its component is large. The right hand side of the problem is a trigonometric function, so it can be well approximated by a polynomial.

Table 4.3 displays results including those with large w. Looking at columns under heading $q = 2$, they show that if the number of subintervals w increases then the accuracy of r^*_{wq} improves. It is also observed that even though r^*_{wq} is relatively large for small number of subintervals w it provides an adequate approximation for r_{wq} indicated by small R^{**}. Similar indication is obtained for small number of points .

It is also notable that by doubling the number of subintervals w the accuracy of r^*_{wq} converges faster than the convergence of r^*_{wq} itself, for example for $q = 3$, R^* decreases by factor of order 10^{-1} while R^{**} reduces by factor 10^{-2} as the number of subintervals is doubled.

Comparing the results using either two Chebyshev zeros or two Gauss points, it is interesting to note that in all cases even though their interpolation norms are not the same their maximum interpolation errors are identical. The other results indicate that using both types of point the interpolations produce almost identical errors. This result shows that the polynomial interpolation based on equation (4.10) can be used to obtain a good approximation for the residual, regardless the type of collocation points.

Table 4.3

(*Chebyshev and Gauss points,* value problem 3)

q →	2		3		6		
	C	G	C	G	C	G	w
R*	9.911e+01	1.227e+02	2.758e+01	4.081e+01	1.828e-01	3.656e-01	5
R**	1.076e-02	1.076e-02	2.526e-05	2.526e-05	2.498e-13	3.626e-13	
R*	4.392e+01	5.605e+01	6.899e+00	1.057e+01	6.711e-03	1.359e-02	10
R**	3.491e-04	3.491e-04	2.052e-07	2.052e-07	4.047e-13	3.703e-13	
R*	1.587e+01	2.066e+01	1.303e+00	2.025e+00	1.666e-04	3.385e-04	20
R**	1.083e-05	1.083e-05	1.591e-09	1.591e-09	2.939e-13	3.165e-13	
R*	4.908e+00	6.440e+00	2.044e-01	3.188e-01	3.311e-06	6.734e-06	40
R**	3.412e-07	3.412e-07	1.269e-11	1.276e-11	3.698e-13	3.629e-13	
R*	1.376e+00	1.811e+00	2.878e-02	4.494e-02	5.849e-08	1.190e-07	80
R**	1.068e-08	1.068e-08	3.686e-13	3.829e-13	3.537e-13	3.543e-13	

4.4 The Error Estimates

In this section we shall explore some error estimates derived by considering the equation (4.2) and the equation (4.8). There has been considerable discussion concerning the error estimates based on examining the operator form and related matrices and on the residual in Ahmed's thesis [1], and further development can be found in Ahmed and Wright's paper [2] where they dealt with single higher order differential equations, even though they did not give much attention for problems with severe layers. The work developed here, essentially based on their idea, focuses on numerical investigation of the error estimates for system of differential equations rather than just single differential equations. Moreover we will observe numerically the performance of the error estimates for problems having severe layers, as well as smooth problems.

Recall the equation (4.8). Combining this and (4.2) the residual $\boldsymbol{r}_{wq}$ can be related to the error $\boldsymbol{e}_{wq}$ as follows

$$\begin{aligned} \boldsymbol{r}_{wq} &= (\boldsymbol{D}-\boldsymbol{M})\,\boldsymbol{x}_{wq} - \boldsymbol{y} \\ &= (\boldsymbol{D}-\boldsymbol{M})\,\boldsymbol{x}_{wq} - (\boldsymbol{D}-\boldsymbol{M})\,\boldsymbol{x} \\ &= (\boldsymbol{D}-\boldsymbol{M})(\boldsymbol{x}_{wq} - \boldsymbol{x}) \\ &= (\boldsymbol{D}-\boldsymbol{M})\,\boldsymbol{e}_{wq} \end{aligned}$$

or

$$\boldsymbol{e}_{wq} = (\boldsymbol{D}-\boldsymbol{M})^{-1}\boldsymbol{r}_{wq} \qquad \ldots(4.11)$$

Taking the norm of the last equation, this immediately gives

$$\|\boldsymbol{e}_{wq}\| \leq \|(\boldsymbol{D}-\boldsymbol{M})^{-1}\|\,\|\boldsymbol{r}_{wq}\| \qquad \ldots(4.12)$$

and then using $\|\boldsymbol{Q}_{wq}\|$ described in §4.2 as an approximation for $\|(\boldsymbol{D}-\boldsymbol{M})^{-1}\|$ we have an error estimate

$$E_1 = \|\boldsymbol{Q}_{wq}\|\,\|\boldsymbol{r}_{wq}\| \qquad \ldots(4.13)$$

This error estimate is likely to be larger than the error since it is an estimate of a bound on the error. Furthermore, the operator $(\boldsymbol{D}-\boldsymbol{M})^{-1}$ is essentially an integrating operator, hence one would expect considerable cancellation in the evaluation (4.11) of $\boldsymbol{e}_{wq}$ which again suggest that inequality (4.12) is likely to be very crude.

Note that $\boldsymbol{r}_{wq}$ may be evaluated at any point without difficulty since $\boldsymbol{x}_{wq}$ is a piecewise polynomial and it will have an oscillatory nature since it is constrained to be zero at the collocation points. These suggest that the residual $\boldsymbol{r}_{wq}$ might be taken into account in developing an error estimate for collocation methods. This can be done, firstly by writing the operator in a different form and the using the compact operator $\boldsymbol{K}$ defined in equation (4.5), we have

$$\begin{aligned}(\boldsymbol{D}-\boldsymbol{M})^{-1} &= ((\boldsymbol{I}-\boldsymbol{M}\boldsymbol{D}^{-1})\boldsymbol{D})^{-1} \\ &= ((\boldsymbol{I}-\boldsymbol{K})\boldsymbol{D})^{-1} \\ &= \boldsymbol{D}^{-1}(\boldsymbol{I}-\boldsymbol{K})^{-1} \qquad \ldots(4.14)\end{aligned}$$

this allows the identity (4.6) to be used giving

$$\begin{aligned}\boldsymbol{e}_{wq} &= \boldsymbol{D}^{-1}(\boldsymbol{I}+(\boldsymbol{I}-\boldsymbol{K})^{-1}\boldsymbol{K})\boldsymbol{r}_{wq} \\ &= \boldsymbol{D}^{-1}\boldsymbol{r}_{wq}+(\boldsymbol{D}-\boldsymbol{M})^{-1}\boldsymbol{K}\boldsymbol{r}_{wq} \qquad \ldots(4.15)\end{aligned}$$

Using results in §4.3, in which $\boldsymbol{r}^{*}_{wq}(t)$ can be used as approximation for $\boldsymbol{r}_{wq}(t)$, it is clearly straightforward to evaluate both $\boldsymbol{D}^{-1}\boldsymbol{r}^{*}_{wq}$ and $\boldsymbol{K}\boldsymbol{r}^{*}_{wq}$ as they involve integration of piecewise polynomials and then to estimate $\|\boldsymbol{D}^{-1}\boldsymbol{r}^{*}_{wq}\|$ and $\|\boldsymbol{K}\boldsymbol{r}^{*}_{wq}\|$ by evaluation at a suitable selection of points.

Using $\|\boldsymbol{Q}_{wq}\|$ for approximating $\|(\boldsymbol{D}-\boldsymbol{M})^{-1}\|$ the equation (4.15) then gives the following error estimate

$$E_2 = \|\boldsymbol{D}^{-1}\boldsymbol{r}^{*}_{wq}\| + \|\boldsymbol{Q}_{wq}\|\,(\|\boldsymbol{K}\boldsymbol{r}^{*}_{wq}\| + \|\boldsymbol{r}^{**}_{wq}\|) \qquad \ldots(4.16)$$

where $\|\boldsymbol{r}^{**}_{wq}\|$ is also estimated by evaluation at a suitable choice of points.

A simplified estimate can be obtained by ignoring $\| r_{wq}^{**} \|$ which should be valid if sufficient points are used in constructing r_{wq}^{*}. The simplified estimate will then be in the form

$$E^{*}_{2} = \| D^{-1} r_{wq}^{*} \| + \| Q_{wq} \| \| K r_{wq}^{*} \| \qquad \ldots(4.17)$$

Recall equation (4.4) which can be written as

$$x_{wq} = (D - \varphi_{wq} M)^{-1} \varphi_{wq} y,$$

This suggests that the original approximation $(D - \varphi_{wq} M)^{-1} \varphi_{wq}$ should be used to approximate the operator $(D - M)^{-1}$ in equation (4.15) which will give an approximate function rather than just its norm. Hence, using (4.4) and (4.15), we define

$$e_{wq}^{*} = D^{-1} r_{wq}^{*} + (D - \varphi_{wq} M)^{-1} \varphi_{wq} K r_{wq}^{*} \qquad \ldots(4.18)$$

The last approach is very convenient, since it makes use the same matrix as in the original approximate solution but with a new right hand side, i.e. $K r_{wq}^{*}$ instead of y.

The error estimate then can be defined as

$$E_3 = \| e_{wq}^{*} \| + \| Q_{wq} \| \| r_{wq}^{**} \| \qquad \ldots(4.19)$$

again if r^{*} is a good approximation for r then $\| r_{wq}^{**} \|$ can be ignored and the error estimate will be in the simpler form

$$E^{*} = \| e_{wq}^{*} \| \qquad \ldots(4.20)$$

The error estimate E^{*} is relatively much cheaper as we do not need to construct the matrix Q_{wq}.

As mentioned above, the residual $\boldsymbol{r}(t)$ may be evaluated at any point, hence it is local in nature and so $\|\boldsymbol{r}(t)\|_k$ the norm of the residual related to k^{th} subinterval, $1 \le k \le w$, which is giving some idea of error measure. Furthermore, another simple error measure is also provided by evaluating $\|\boldsymbol{r}(t)\|_k h_k$ (further discussion can be found in chapter 5). Simply by taking the maximum of these local error measures, we have two simple error estimates

$$\|\boldsymbol{r}\| = \max_k \|\boldsymbol{r}(t)\|_k$$

and

$$\|\boldsymbol{rh}\| = \max_k \|\boldsymbol{r}(t)h\|_k$$

In our developed algorithm, two error measures above do not add more computational cost, in particular the second one since we have constructed $\boldsymbol{r}^*_{wq}$ to approximate the residual and have computed the quantity $\|\boldsymbol{rh}\|$ to be used in our adaptive mesh selection algorithm.

4.5 Numerical Experiments

In the numerical experiments, both the Chebyshev zeros and Gauss points have been used as collocation points, in addition Chebyshev extrema was also used for boundary value problem 4. The basis functions constructing $\boldsymbol{x}_{wq}$ will be either Chebyshev or Legendre polynomials. In all computation we implemented the uniform mesh points and the norm values presented in the tables were estimated by evaluation at 400 equispaced points.

For the first illustrative example let us recall *example 2* discussed in §4.3. This boundary value problem taken from Ascher [6] has smooth solution but the coefficient matrix whose non-polynomial component is rapidly varying near the left boundary. Moreover, the coefficient matrix $\boldsymbol{A}(t)$ has singularity at $t = 0$, and

this causes big trouble for some procedures [10] while here it is shown that the collocation algorithm performs very well.

The problem considered is

Problem 1 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -1/t \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ (8/(8-t^2))^2 \end{pmatrix}, \qquad 0 < t < 1$$

The boundary conditions are

$$x_2(0) = 0 \text{ and } x_1(1) = 0.$$

And the analytical solution is given by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2\ln(7/(8-t^2)) \\ (4t)/(8-t^2) \end{pmatrix}$$

Table 4.4A and *Table 4.4B* display results using Chebyshev zeros and Gauss points respectively. Note that for simplicity the interpolation error r^{**}_{wq} is not presented in these tables but it can be referred to *Table 4.2*. Looking at the last two columns of the tables, we observe that the estimate E^* performs well, in particular for $q = 3$ where collocation algorithms implementing both type of points produce quite accurate estimates.

Using two Chebyshev points the results show that E^* underestimates the error slightly, but it is still very close. Using two Gauss points on the other hand slightly overestimate the error, even though it is again reasonably close. It is notable for this case that both types of points produce estimates which are almost identical, the difference is in accuracy of the solution where the error values using Gauss points are a bit smaller than those using Chebyshev zeros, as would be expected from the results of de Boor and Swartz [23].

Comparing the columns under headings $\|r\|$ and $\|rh\|$, the results clearly indicate that no matter what the points chosen the value of $\|rh\|$ is significantly smaller than $\|r\|$ and it much closer to the error with improvement as the interval size decreases.

Table 4.4A

(Chebyshev Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	6.665e-03	2.222e-03	7.995e-04	3.508e-03	8.632e-04	1.006e-03
	5	2.531e-03	5.062e-04	1.909e-04	1.248e-03	2.502e-04	3.354e-04
	10	6.616e-04	6.616e-05	2.595e-05	3.105e-04	5.534e-05	7.842e-05
	20	1.694e-04	8.469e-06	3.391e-06	7.753e-05	1.314e-05	1.906e-05
	40	4.303e-05	1.076e-06	4.337e-07	1.938e-05	3.209e-06	4.752e-06
3	3	3.187e-04	1.062e-04	2.212e-05	2.723e-05	1.805e-05	1.734e-05
	5	7.691e-05	1.538e-05	3.493e-06	3.885e-06	2.885e-06	2.729e-06
	10	1.063e-05	1.063e-06	2.545e-07	2.680e-07	2.118e-07	1.976e-07
	20	1.407e-06	7.034e-08	1.721e-08	1.765e-08	1.435e-08	1.330e-08
	40	1.810e-07	4.525e-09	1.119e-09	1.133e-09	9.343e-10	8.624e-10

Table 4.4B

(Gauss Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	8.916e-03	2.973e-03	2.831e-04	4.683e-03	7.586e-04	2.835e-04
	5	3.387e-03	6.773e-04	6.501e-05	1.667e-03	2.252e-04	6.515e-05
	10	8.855e-04	8.855e-05	8.562e-06	4.148e-04	5.160e-05	8.572e-06
	20	2.268e-04	1.134e-05	1.100e-06	1.036e-04	1.253e-05	1.101e-06
	40	5.740e-05	1.435e-06	1.395e-07	2.589e-05	3.096e-06	1.395e-07
3	3	5.149e-04	1.716e-04	9.408e-06	4.379e-05	9.394e-06	9.336e-06
	5	1.245e-04	2.489e-05	1.450e-06	5.602e-06	1.446e-06	1.446e-06
	10	1.699e-05	1.699e-06	1.037e-07	3.482e-07	1.037e-07	1.037e-07
	20	2.223e-06	1.112e-07	6.949e-09	2.173e-08	6.948e-09	6.948e-09
	40	2.844e-07	7.110e-09	4.498e-10	1.358e-09	4.498e-10	4.498e-10

As the second illustration, the followings boundary value problem describing the symmetrical bending of a laterally loaded circular plate [46] consists of three differential equations.

The problem is as follows

Problem 2 :

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1/t^2 & -1/t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1/t \end{pmatrix}, \qquad 1 < t < 2$$

The associated boundary conditions are

$$\begin{aligned} 0.3x_2(1) + x_3(1) &= 0 \\ 0.15x_2(2) + x_3(2) &= 0 \\ x_1(2) &= 0 \end{aligned}$$

and the exact solution is given by

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{t^2}{4}(-1+\ln\frac{t}{2}) - \frac{t^2}{8}(\frac{0.7}{1.3} + \frac{2}{3}\ln 2) - \frac{2.6}{2.1}(\ln 2)(\ln\frac{t}{2}) + \frac{3.3}{2.6} + \frac{\ln 2}{3} \\ \frac{t}{2}(-1+\ln\frac{t}{2}) + \frac{t}{4}(\frac{0.3}{1.3} - \frac{2}{3}\ln 2) - \frac{2.6}{2.1}(\ln 2)(\frac{1}{t}) \\ \frac{1}{2}(\ln\frac{t}{2}) + \frac{1}{4}(\frac{0.3}{1.3} - \frac{2}{3}\ln 2) + \frac{2.6}{2.1}(\ln 2)(\frac{1}{t^2}) \end{pmatrix}$$

Table 4.5 on the following page displays results of numerical experiments using Gauss collocation points, other results for Chebyshev zeros though not presented here indicate a similar observation.

This problem exhibits the fact that using Gauss points the estimate E^* may underestimate the error, as it occurred for Chebyshev zeros in the previous example. However, as we can see the estimate is reasonably close to the actual error. It is also notable for this problem that even though both the coefficient $A(t)$ and the right hand side $y(t)$ are not polynomials the estimate r_{wq}^* performs very well indicated by small $\| r_{wq}^{**} \|$ values.

It is observed that the estimate E^* is satisfactory and always very close to the actual error even for the high accuracy solutions as well as in cases where it underestimates the error. It is also worth to note here that $\|rh\|$ is a good estimate for the error, even though in fact theoretically it is a very crude error estimate. Meanwhile the norm of residual $\|r\|$ is significantly larger than the error, especially for high accuracy solutions. Comparing $\|rh\|$ and $\|r\|$, the results of numerical experiments presented in *Table 4.5* clearly indicate that $\|rh\|$ consistently produces better estimate for the error than $\|r\|$.

Table 4.5
(Gauss Points)

q	w	$\|r^{\cdot\cdot}\|$	$\|r\|$	$\|rh\|$	$\|D^{-1}r^{\cdot}\|$	$\|Kr^{*}\|$	$E^{\cdot}$	E
2	3	5.237e-05	1.030e-01	3.435e-02	3.254e-03	3.696e-03	7.481e-03	3.138e-03
	5	5.966e-06	4.547e-02	9.096e-03	8.712e-04	1.023e-03	1.240e-03	8.511e-04
	10	2.546e-07	1.344e-02	1.344e-03	1.298e-04	1.564e-04	1.344e-04	1.282e-04
	15	3.741e-08	6.338e-03	4.225e-04	4.092e-05	4.973e-05	4.074e-05	4.058e-05
	20	9.387e-09	3.674e-03	1.837e-04	1.782e-05	2.175e-05	1.763e-05	1.770e-05
	30	1.308e-09	1.684e-03	5.613e-05	5.451e-06	6.683e-06	5.397e-06	5.427e-06
	40	3.194e-10	9.620e-04	2.405e-05	2.337e-06	2.872e-06	2.318e-06	2.329e-06
3	3	4.613e-07	1.480e-02	4.934e-03	2.582e-04	2.847e-04	2.774e-04	2.600e-04
	5	2.113e-08	4.160e-03	8.320e-04	4.720e-05	5.135e-05	4.780e-05	4.732e-05
	10	2.462e-10	6.446e-04	6.446e-05	3.885e-06	4.425e-06	3.893e-06	3.887e-06
	15	1.658e-11	2.059e-04	1.373e-05	8.443e-07	9.771e-07	8.450e-07	8.445e-07
	20	2.377e-12	9.029e-05	4.514e-06	2.804e-07	3.272e-07	2.806e-07	2.805e-07
	30	1.494e-13	2.782e-05	9.272e-07	5.818e-08	6.844e-08	5.819e-08	5.818e-08
	40	2.125e-14	1.197e-05	2.992e-07	1.887e-08	2.229e-08	1.887e-08	1.887e-08
5	3	2.650e-11	1.978e-04	6.593e-05	1.856e-06	1.964e-06	1.841e-06	1.846e-06
	5	1.958e-13	2.244e-05	4.488e-06	1.335e-07	1.423e-07	1.338e-07	1.336e-07
	10	9.314e-16	9.533e-07	9.533e-08	2.993e-09	3.255e-09	2.995e-09	2.994e-09
	15	9.024e-16	1.398e-07	9.318e-09	2.980e-10	3.294e-10	2.982e-10	2.981e-10
	20	1.007e-15	3.504e-08	1.752e-09	5.655e-11	6.301e-11	5.658e-11	5.656e-11
	30	1.088e-15	4.877e-09	1.626e-10	5.297e-12	5.951e-12	5.333e-12	5.298e-12
	40	8.900e-16	1.190e-09	2.976e-11	9.742e-13	1.099e-12	1.038e-12	9.740e-13

As the third illustration, we consider a boundary value problem corresponding to the bending of a thin beam clamped at both ends [46]. This problem has been chosen since the coefficient is a simple constant matrix and the non-homogeneous term is not a polynomial but it can be well approximated by polynomial interpolation. Moreover its solution is smooth. The problem is

Problem 3 :

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ (t^4 + 14t^3 + 49t^2 + 32t - 12)e^t \end{pmatrix}, \quad 0 < t < 1$$

The boundary conditions at both end points are determined by

$$x_1(0) = x_2(0) = 0$$

$$x_1(1) = x_2(1) = 0$$

and the analytical solution is given by

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} t^2(1-t)^2 e^t \\ (2t - 5t^2 + 2t^3 + t^4)e^t \\ (2 - 8t - t^2 + 6t^3 + t^4)e^t \\ (-6 - 6t + 19t^2 + 10t^3 + t^4)e^t \end{pmatrix}$$

The results of numerical experiments are displayed in *Tables 4.6A* and *4.6B* on the following page. From these tables, we can see that in most cases the error estimate E^* improves steadily as the number of collocation points increases. Comparing the results displayed in both tables it is observed that using Gauss points gives more accurate solutions compared to those using Chebyshev points.

The results in *Tables 4.6A* and *4.6B* also show that both types of collocation points appear to occasionally underestimate the error slightly. The estimates produced, nevertheless, are still reasonably close to the actual errors. As we can see, the underestimation occurs for different q values, for Chebyshev zeros $q = 2$ and Gauss points $q = 3$.

Detailed inspection of $\|\boldsymbol{r}\|$, $\|\boldsymbol{rh}\|$ and E for different w reveal that the error estimate $\|\boldsymbol{rh}\|$ is more satisfactory than $\|\boldsymbol{r}\|$. Comparing *Table 4.6A* and *Table 4.6B*, it is observed that even though an $\|\boldsymbol{r}\|$ in *Table 4.6A* is significantly smaller than that corresponding $\|\boldsymbol{r}\|$ in *Table 4.6B*, this does not imply the actual error E in *Table 4.6A* will be smaller than the error E in *Table 4.6B*. It is important to realise that a smaller residual does not necessarily correspond to a smaller error.

Table 4.6A

(Chebyshev Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	9.340e+00	3.114e+00	1.149e+00	1.149e+00	9.308e-01	1.426e+00
	5	3.676e+00	7.352e-01	2.831e-01	2.831e-01	2.303e-01	5.002e-01
	10	9.834e-01	9.834e-02	3.904e-02	3.904e-02	3.234e-02	1.234e-01
	15	4.471e-01	2.981e-02	1.195e-02	1.195e-02	9.979e-03	5.468e-02
	20	2.544e-01	1.272e-02	5.126e-03	5.126e-03	4.298e-03	3.071e-02
	30	1.149e-01	3.830e-03	1.544e-03	1.544e-03	1.300e-03	1.363e-02
	40	6.519e-02	1.630e-03	6.566e-04	6.566e-04	5.543e-04	7.665e-03
3	3	5.238e-01	1.746e-01	4.049e-02	4.049e-02	4.140e-02	3.989e-02
	5	1.255e-01	2.510e-02	6.014e-03	6.014e-03	6.034e-03	5.780e-03
	10	1.700e-02	1.700e-03	4.162e-04	4.162e-04	4.141e-04	3.940e-04
	15	5.175e-03	3.450e-04	8.506e-05	8.506e-05	8.446e-05	8.013e-05
	20	2.213e-03	1.106e-04	2.737e-05	2.737e-05	2.716e-05	2.573e-05
	30	6.646e-04	2.215e-05	5.500e-06	5.500e-06	5.453e-06	5.158e-06
	40	2.823e-04	7.056e-06	1.755e-06	1.755e-06	1.739e-06	1.644e-06
5	3	5.939e-04	1.980e-04	2.035e-05	2.035e-05	2.608e-05	2.497e-05
	5	5.061e-05	1.012e-05	1.050e-06	1.050e-06	1.300e-06	1.248e-06
	10	1.694e-06	1.694e-07	1.769e-08	1.769e-08	2.148e-08	2.063e-08
	15	2.283e-07	1.522e-08	1.593e-09	1.593e-09	1.925e-09	1.847e-09
	20	5.481e-08	2.741e-09	2.871e-10	2.871e-10	3.524e-10	3.320e-10
	30	7.302e-09	2.434e-10	2.553e-11	2.553e-11	3.709e-11	2.937e-11
	40	1.743e-09	4.357e-11	4.572e-12	4.572e-12	1.363e-11	5.315e-12

Table 4.6B

(Gauss Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	1.249e+01	4.162e+00	3.942e-01	3.942e-01	4.084e-01	3.692e-01
	5	4.918e+00	9.837e-01	9.450e-02	9.450e-02	9.291e-02	8.980e-02
	10	1.316e+00	1.316e-01	1.275e-02	1.275e-02	1.253e-02	1.238e-02
	15	5.986e-01	3.990e-02	3.873e-03	3.873e-03	3.822e-03	3.799e-03
	20	3.406e-01	1.703e-02	1.654e-03	1.654e-03	1.637e-03	1.631e-03
	30	1.531e-01	5.104e-03	4.962e-04	4.962e-04	4.927e-04	4.916e-04
	40	8.662e-02	2.166e-03	2.106e-04	2.106e-04	2.095e-04	2.092e-04
3	3	8.478e-01	2.826e-01	1.668e-02	1.668e-02	1.619e-02	1.640e-02
	5	2.006e-01	4.012e-02	2.449e-03	2.449e-03	2.426e-03	2.436e-03
	10	2.686e-02	2.686e-03	1.680e-04	1.680e-04	1.676e-04	1.678e-04
	15	8.145e-03	5.430e-04	3.423e-05	3.423e-05	3.420e-05	3.421e-05
	20	3.476e-03	1.738e-04	1.100e-05	1.100e-05	1.099e-05	1.100e-05
	30	1.042e-03	3.473e-05	2.207e-06	2.207e-06	2.206e-06	2.206e-06
	40	4.421e-04	1.105e-05	7.036e-07	7.036e-07	7.035e-07	7.035e-07
5	3	1.145e-03	3.817e-04	1.211e-05	1.211e-05	1.211e-05	1.211e-05
	5	9.724e-05	1.945e-05	6.286e-07	6.286e-07	6.286e-07	6.286e-07
	10	3.247e-06	3.247e-07	1.064e-08	1.064e-08	1.064e-08	1.064e-08
	15	4.372e-07	2.915e-08	9.594e-10	9.594e-10	9.653e-10	9.594e-10
	20	1.049e-07	5.246e-09	1.731e-10	1.731e-10	1.804e-10	1.730e-10
	30	1.397e-08	4.657e-10	1.540e-11	1.540e-11	2.115e-11	1.541e-11
	40	3.334e-09	8.335e-11	2.759e-12	2.759e-12	1.044e-11	2.778e-12

As a further illustration we consider an almost singular boundary value problem for which one might expect a stability problem to occur. The specification of problem as follows

Problem 4 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -(\pi^2 + 10^{-4}) & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 10^{-4}\sin(\pi t) \end{pmatrix}, \quad 0 < t < 1$$

its boundary conditions are

$$x_1(0) = 0 \quad \text{and} \quad x_1(1) = 0$$

and the unique solution is

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sin(\pi t) \\ \pi\cos(\pi t) \end{pmatrix}$$

For this problem we also present the results using Chebyshev extrema as displayed in *Table 4.7C*, while *Table 4.7A* and *Table 4.7B* contain the results using Chebyshev zeros and Gauss points respectively.

In *Table 4.7A* it will be noticed that all $\|r\|$ are significantly smaller than the errors. This also occurs when Chebyshev extrema points are used as shown in *Table 4.7C*. By contrast, using Gauss points this only occurs occasionally.

Using three collocation points per subinterval, for $w = 3$, 5 and 10 in *Table 4.7A* it seems that the solutions are quite poor and they become worse as w increases. Results for $w = 10$ are exceptional, probably the collocation matrix is almost singular in this case. Looking at the results in *Table 4.7C*, we found a similar indication though there is no such a dramatic rise in the error for $w = 10$. In contrast, Gauss points again show their superiority over the others in which the errors are very small and they consistently improve as the interval size decreases, moreover the error estimate E^* perform quite satisfactorily.

Despite their superiority for small q, using Gauss points with larger q may produce solutions which become worse as w increases as displayed in the last two rows of *Table 4.7B*. In these cases the values of E^* are unsatisfactory, since they

in contrast decrease as w increases. This effect is probably due to round off error as the errors are very small in these cases. On the other hand the results for Chebyshev extrema points show that they consistently reduce as intervals size halved.

Table 4.7A

(Chebyshev Zeros)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
3	3	4.200e-04	1.400e-04	3.500e-05	1.269e-04	3.311e-02	3.168e+00
	5	8.099e-04	1.620e-04	4.050e-05	1.338e-04	2.818e-01	3.354e+00
	10	7.121e-01	7.121e-02	1.780e-02	5.593e-02	8.055e+05	1.422e+03
	20	2.117e-04	1.059e-05	2.646e-06	8.314e-06	2.793e-01	2.096e-01
	40	2.498e-05	6.244e-07	1.561e-07	4.904e-07	1.548e-02	1.234e-02
5	3	1.577e-04	5.258e-05	5.527e-06	2.005e-05	2.740e-01	2.638e-01
	5	1.480e-05	2.960e-06	3.111e-07	1.028e-06	1.636e-02	1.423e-02
	10	4.847e-07	4.847e-08	5.095e-09	1.601e-08	2.666e-04	2.291e-04
	20	1.531e-08	7.654e-10	8.045e-11	2.527e-10	4.200e-06	3.602e-06
	40	4.796e-10	1.199e-11	1.260e-12	3.959e-12	6.576e-08	5.650e-08
7	3	2.827e-07	9.422e-08	7.851e-09	2.848e-08	2.505e-04	2.260e-04
	5	8.753e-09	1.751e-09	1.459e-10	4.819e-10	4.431e-06	3.959e-06
	10	7.124e-11	7.124e-12	5.936e-13	1.865e-12	1.769e-08	1.578e-08
	20	5.627e-13	2.812e-14	2.347e-15	7.343e-15	6.870e-11	1.917e-10
	40	5.675e-15	1.533e-16	8.040e-17	1.955e-16	3.012e-13	1.866e-10

Table 4.7B

(Gauss Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
3	3	5.269e-02	1.756e-02	1.124e-03	4.079e-03	6.183e+00	5.273e+00
	5	2.128e-02	4.256e-03	2.725e-04	9.000e-04	8.361e-01	4.230e-01
	10	2.457e-03	2.457e-04	1.573e-05	4.943e-05	1.035e-02	5.913e-03
	20	3.100e-04	1.550e-05	9.925e-07	3.118e-06	1.623e-04	9.311e-05
	40	3.886e-05	9.715e-07	6.220e-08	1.954e-07	2.576e-06	1.493e-06
5	3	3.299e-04	1.100e-04	3.652e-06	1.325e-05	1.737e-04	9.635e-05
	5	2.845e-05	5.689e-06	1.889e-07	6.241e-07	1.232e-06	7.530e-07
	10	9.270e-07	9.270e-08	3.078e-09	9.671e-09	3.872e-09	3.420e-09
	20	2.927e-08	1.463e-09	4.860e-11	1.527e-10	4.727e-11	1.366e-10
	40	9.170e-10	2.292e-11	7.613e-13	2.392e-12	1.060e-12	1.965e-10
7	3	6.018e-07	2.006e-07	4.396e-09	1.595e-08	4.685e-09	4.475e-09
	5	1.863e-08	3.725e-09	8.163e-11	2.696e-10	8.704e-11	1.381e-10
	10	1.516e-10	1.515e-11	3.321e-13	1.043e-12	2.272e-12	4.325e-11
	20	1.196e-12	5.980e-14	1.312e-15	4.119e-15	1.532e-12	7.003e-11
	40	1.157e-14	2.851e-16	1.616e-16	2.828e-16	3.017e-13	1.870e-10

Table 4.7C

(Chebyshev Extrema Points)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
3	3	2.446e-04	8.152e-05	2.648e-05	9.588e-05	1.247e-02	3.152e+00
	5	4.509e-04	9.018e-05	2.929e-05	9.672e-05	9.785e-02	3.219e+00
	10	1.459e-03	1.459e-04	4.737e-05	1.488e-04	3.798e+00	5.040e+00
	20	3.666e-04	1.833e-05	5.952e-06	1.870e-05	9.419e-01	6.283e-01
	40	3.871e-05	9.677e-07	3.143e-07	9.873e-07	4.181e-02	3.311e-02
5	3	4.693e-04	1.564e-04	2.808e-05	1.019e-04	2.546e+00	1.485e+00
	5	2.800e-05	5.600e-06	1.005e-06	3.320e-06	5.675e-02	4.794e-02
	10	9.002e-07	9.002e-08	1.616e-08	5.076e-08	8.624e-04	7.391e-04
	20	2.843e-08	1.421e-09	2.551e-10	8.015e-10	1.348e-05	1.155e-05
	40	8.907e-10	2.227e-11	3.997e-12	1.256e-11	2.106e-07	1.804e-07
7	3	5.456e-07	1.819e-07	1.567e-08	5.686e-08	2.698e-04	2.463e-04
	5	1.692e-08	3.383e-09	2.916e-10	9.632e-10	4.965e-06	4.453e-06
	10	1.377e-10	1.377e-11	1.187e-12	3.729e-12	2.012e-08	1.791e-08
	20	1.087e-12	5.437e-14	4.687e-15	1.472e-14	7.909e-11	1.517e-10
	40	9.569e-15	2.694e-16	1.346e-16	2.783e-16	1.074e-12	3.211e-11

Finally let us examine a boundary value problem where its fundamental solution contains rapid growing and decaying terms. Stability difficulty arises when using shooting method to solve this problem [10]. The problem having a problem parameter μ is specified by

Problem 5 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & \mu \\ \mu & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \mu\cos^2(\pi t) + (2/\mu)\pi^2\cos(2\pi t) \end{pmatrix}$$

the boundary conditions and its unique exact solution are respectively given by

$$x_1(0) = x_1(1) = 0$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} (e^{-\mu(1-t)} + e^{-\mu t})/(1+e^{-\mu}) - \cos^2(\pi t) \\ (\mu e^{-\mu(1-t)} - \mu e^{-\mu t})/(1+e^{-\mu}) - (\frac{\pi}{\mu})\sin(2\pi t) \end{pmatrix}$$

Taking problem parameter $\mu = 10^2$ and collocating at Gaussian points, *Table 4.8* shows the results of computations.

The figures indicate that E^* is a reasonable error estimate even if the solution is very poor, which occurs when q and /or w are small. On the other hand, in all cases, the norm of residual $\|r\|$ clearly overestimates the error significantly . By contrast, the $\|rh\|$ is much better in the sense it is closer to the error and in some cases it is reasonably satisfactory.

Note that the matrix $A(t)$ in this example is large and it is clear that $\|Kr^*_{wq}\|$ is greater than $\|D^{-1}r^*_{wq}\|$ so that the second term in expression for E^* (see equation 4.18) will be dominant and in some cases resulting in a poor estimate E^*.

One of the most notable observations for this problem is that the estimate E^* smoothly reduces and it is closer to the error as q or w increases. This last result is very important since we shall develop some adaptive mesh selection algorithms utilising the error estimate E^*.

Table 4.8

(Gauss Points, problem parameter $\mu = 10^2$)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	1.017e+02	3.389e+01	3.274e+00	3.274e+02	1.556e+01	9.659e-01
	5	7.822e+01	1.564e+01	1.521e+00	1.521e+02	6.626e+00	6.087e-01
	10	5.762e+01	5.762e+00	5.612e-01	5.612e+01	1.131e+00	3.014e-01
	15	4.556e+01	3.037e+00	2.958e-01	2.958e+01	4.704e-01	1.821e-01
	20	3.688e+01	1.844e+00	1.796e-01	1.796e+01	2.311e-01	1.303e-01
	30	2.547e+01	8.489e-01	8.267e-02	8.267e+00	9.492e-02	7.125e-02
	40	1.857e+01	4.643e-01	4.522e-02	4.522e+00	5.254e-02	4.231e-02
	50	1.412e+01	2.823e-01	2.749e-02	2.749e+00	3.206e-02	2.690e-02
3	3	9.463e+01	3.154e+01	2.019e+00	2.019e+02	1.150e+01	7.838e-01
	5	5.577e+01	1.115e+01	7.141e-01	7.141e+01	1.775e+00	3.060e-01
	10	3.344e+01	3.344e+00	2.140e-01	2.140e+01	2.896e-01	1.229e-01
	15	2.143e+01	1.429e+00	9.147e-02	9.147e+00	1.195e-01	6.543e-02
	20	1.444e+01	7.220e-01	4.623e-02	4.623e+00	5.857e-02	3.667e-02
	30	7.373e+00	2.458e-01	1.573e-02	1.573e+00	1.875e-02	1.351e-02
	40	4.230e+00	1.057e-01	6.770e-03	6.770e-01	7.651e-03	6.087e-03
	50	2.638e+00	5.277e-02	3.378e-03	3.378e-01	3.657e-03	3.133e-03
5	3	4.082e+01	1.361e+01	4.518e-01	4.518e+01	1.261e+00	1.873e-01
	5	2.310e+01	4.620e+00	1.534e-01	1.534e+01	2.722e-01	8.154e-02
	10	7.035e+00	7.035e-01	2.336e-02	2.336e+00	3.426e-02	1.720e-02
	15	2.596e+00	1.731e-01	5.748e-03	5.748e-01	7.192e-03	4.855e-03
	20	1.113e+00	5.565e-02	1.848e-03	1.848e-01	2.114e-03	1.649e-03
	30	2.813e-01	9.377e-03	3.114e-04	3.114e-02	3.401e-04	2.935e-04
	40	9.486e-02	2.371e-03	7.875e-05	7.875e-03	8.329e-05	7.603e-05
	50	3.871e-02	7.741e-04	2.571e-05	2.571e-03	2.657e-05	2.512e-05

4.6 Numerical Evaluation of the estimate E^* for Stiff BVPs

Although here the concept of stiffness for system of differential equations will not be discussed in detail, in this section using numerical experiments we shall observe performance of the error estimate E^* when dealing with stiff problems having severe layers.

Firstly let us consider the following problem

Problem 6 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \mu\cos^2(\pi t) + 2\pi^2\cos(2\pi t) \end{pmatrix}$$

The boundary conditions and the analytical solutions are given by

$$x_1(0) = x_1(1) = 0$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} (e^{-\mu^{1/2}(1-t)} + e^{-\mu^{1/2}t})/(1+e^{-\mu^{1/2}}) - \cos^2(\pi t) \\ (\mu^{1/2}e^{-\sqrt{\mu}(1-t)} - \mu^{1/2}e^{-\mu^{1/2}t})/(1+e^{-\mu^{1/2}}) - \pi\sin(2\pi t) \end{pmatrix}$$

This is a 'real' problem and one reason for choosing this problem is that it has been used elsewhere to show that some procedures do not perform well [19]. In this problem, since the solution x_2 behaves badly, i.e. it undergoes a drop from $x_2 = \sqrt{\mu}$ to $x_2 = 0$ within small subinterval $[0\ ;\ 1/\sqrt{\mu}]$ and near the right end it rises from $x_2 = 0$ to $x_2 = \sqrt{\mu}$ within subinterval $[\frac{\sqrt{\mu}-1}{\sqrt{\mu}}\ ,\ 1]$, we may expect that the errors in calculating x_2 should be worse compared to those in calculating x_1. Moreover, even though the non-homogeneous term of this problem is not a polynomial it can be well approximated by a polynomial, hence the residual will be well approximated by r_{wq}^* in which the interpolation error r_{wq}^{**} will be very close to zero, hence we would expect that the cheaper estimates E^* can be used instead of the estimates E_2 and E_3.

Using the Gauss points, *Table 4.9A* and *Table 4.9B* contain results with problem parameter $\mu = 10^2$ and $\mu = 10^4$ respectively.

In *Table 4.9A* it is observed that the estimate E^* is reasonably satisfactory even if the approximate solution is poor. The estimate improves as the number of points q increases as well as the interval sizes decrease. Looking at $q = 2$ and $w = 3$, since $\|D^{-1}r^*_{wq}\|$ is much smaller than $\|Kr^*_{wq}\|$ it is clear that large $\|Kr^*_{wq}\|$ results in a poor error estimate. Comparing the norms $\|r\|$ and $\|rh\|$ it is again very clear that $\|rh\|$ provides a better approximation.

With problem parameter $\mu = 10^4$ the problem is set to have more severe layers. As can be seen in *Table 4.9B*, this implies that $\|Kr^*_{wq}\|$ become even larger and in all cases $\|Kr^*_{wq}\|$ is much larger than $\|D^{-1}r^*_{wq}\|$. It is clear that $\|Kr^*_{wq}\|$ is the dominant term in the equation forming E^*. From this table, it is notable that if the solution is very poor then the estimate will also be very poor. With this important note, we realise that the estimate may perform very poorly at initial stages of a mesh selection algorithm.

Table 4.9A

(Gauss Points, problem parameter $\mu = 10^2$)

q	w	$\|r^{**}\|$	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
2	3	4.079e-02	2.971e+01	9.889e+00	9.262e-01	1.196e+01	5.084e+00	8.192e-01
2	5	3.415e-03	1.312e+01	2.624e+00	2.518e-01	3.258e+00	4.928e-01	2.388e-01
2	10	1.108e-04	4.661e+00	4.661e-01	4.526e-02	5.379e-01	5.290e-02	4.613e-02
2	20	3.438e-06	1.466e+00	7.329e-02	7.134e-03	8.082e-02	7.790e-03	7.362e-03
2	40	1.083e-07	4.156e-01	1.039e-02	1.012e-03	1.121e-02	1.067e-03	1.035e-03
2	50	3.556e-08	2.730e-01	5.460e-03	5.317e-04	5.864e-03	5.559e-04	5.424e-04
3	3	2.657e-04	6.779e+00	2.260e+00	1.245e-01	1.308e+00	1.796e-01	1.094e-01
3	5	8.018e-06	2.592e+00	5.184e-01	3.119e-02	2.840e-01	3.259e-02	2.880e-02
3	10	6.514e-08	5.038e-01	5.038e-02	3.157e-03	2.850e-02	3.074e-03	3.081e-03
3	20	5.048e-10	7.958e-02	3.979e-03	2.526e-04	2.300e-03	2.506e-04	2.509e-04
3	40	4.031e-12	1.123e-02	2.807e-04	1.790e-05	1.640e-04	1.786e-05	1.787e-05
3	50	9.018e-13	5.891e-03	1.178e-04	7.521e-06	6.899e-05	7.511e-06	7.513e-06
5	3	2.507e-09	3.088e-01	1.029e-01	3.361e-03	3.338e-02	3.695e-03	3.165e-03
5	5	9.861e-12	4.037e-02	8.074e-03	2.647e-04	2.737e-03	2.766e-04	2.589e-04
5	10	1.013e-13	1.942e-03	1.942e-04	6.416e-06	6.658e-05	6.387e-06	6.379e-06
5	20	9.490e-14	7.645e-05	3.822e-06	1.267e-07	1.311e-06	1.265e-07	1.265e-07
5	40	9.702e-14	2.691e-06	6.727e-08	2.232e-09	2.306e-08	2.234e-09	2.231e-09
5	50	1.081e-13	9.033e-07	1.807e-08	5.995e-10	6.190e-09	6.024e-10	5.994e-10

Table 4.9B

(Gauss Points, problem parameter $\mu = 10^4$)

q	w	$\|r^{**}\|$	$\|r\|$	$\|rh\|$	$\|D^{-1}r^{*}\|$	$\|Kr^{*}\|$	E^{*}	E
2	**3**	2.936e+00	1.017e+04	3.389e+03	3.274e+02	1.146e+04	1.556e+03	4.144e+01
2	**5**	2.458e-01	7.822e+03	1.564e+03	1.521e+02	1.290e+04	3.146e+02	4.475e+01
2	**10**	7.975e-03	5.746e+03	5.746e+02	5.595e+01	5.612e+03	1.131e+02	3.014e+01
2	**20**	2.475e-04	3.687e+03	1.843e+02	1.795e+01	1.796e+03	2.311e+01	1.303e+01
2	**40**	7.793e-06	1.857e+03	4.643e+01	4.522e+00	4.522e+02	5.254e+00	4.231e+00
2	**60**	1.028e-06	1.108e+03	1.846e+01	1.798e+00	1.798e+02	2.095e+00	1.805e+00
2	**80**	2.440e-07	7.330e+02	9.162e+00	8.923e-01	8.923e+01	1.033e+00	9.187e-01
3	**3**	1.912e-02	5.997e+03	1.999e+03	1.265e+02	2.019e+04	1.150e+03	7.838e+01
3	**5**	5.772e-04	5.505e+03	1.101e+03	7.035e+01	7.141e+03	1.775e+02	3.060e+01
3	**10**	4.689e-06	3.344e+03	3.344e+02	2.140e+01	2.140e+03	2.896e+01	1.229e+01
3	**20**	3.634e-08	1.444e+03	7.220e+01	4.623e+00	4.622e+02	5.856e+00	3.667e+00
3	**40**	2.903e-10	4.230e+02	1.057e+01	6.770e-01	6.770e+01	7.651e-01	6.087e-01
3	**60**	2.202e-11	1.752e+02	2.920e+00	1.869e-01	1.869e+01	1.954e-01	1.767e-01
3	**80**	8.473e-12	8.839e+01	1.105e+00	7.074e-02	7.074e+00	7.020e-02	6.832e-02
5	**3**	1.804e-07	3.859e+03	1.286e+03	4.271e+01	4.518e+03	1.261e+02	1.873e+01
5	**5**	7.088e-10	2.310e+03	4.619e+02	1.534e+01	1.534e+03	2.722e+01	8.154e+00
5	**10**	8.585e-12	7.035e+02	7.035e+01	2.336e+00	2.336e+02	3.426e+00	1.720e+00
5	**20**	8.821e-12	1.113e+02	5.565e+00	1.848e-01	1.848e+01	2.114e-01	1.649e-01
5	**40**	9.651e-12	9.486e+00	2.371e-01	7.875e-03	7.875e-01	8.329e-03	7.603e-03
5	**60**	9.670e-12	1.807e+00	3.012e-02	1.000e-03	1.000e-01	1.017e-03	9.839e-04
5	**80**	9.462e-12	5.194e-01	6.492e-03	2.156e-04	2.156e-02	2.142e-04	2.136e-04

For the second illustration let us recall *problem 5* and take the problem parameter $\mu = 10^4$.

The following table displays some results using Chebyshev and Gauss points. For the initial stage with small w, the results clearly indicate that the estimate is very unsatisfactory, even though there is an improvement as w increases, but it is quite slow. Looking back at the *Table 4.8* and take a look at the second row, we can see that the error and its estimate are 6.087E-01 and 6.626E+00 respectively, while in *Table 4.10* for q = 10 and w = 10 (where the error actual error is close to 6.087E-01) the value of E^* overestimates the actual error very significantly, i.e. the error and its estimate are 6.771E-01 and 1.727E+02 respectively.

Detailed inspection of $\|r\|$ and $\|rh\|$ reveals that $\|rh\|$ gives a better approximation than $\|r\|$, unfortunately this error measure is not very close to the actual error.

Table 4.10

(Chebyshev and Gauss Points, $\mu = 10^4$)

q	w	$\|r\|$	$\|rh\|$	$\|D^{-1}r^*\|$	$\|Kr^*\|$	E^*	E
7C	**3**	1.134e+05	3.781e+04	3.151e+03	3.151e+07	8.045e+04	1.134e+01
7G	**3**	1.769e+05	5.895e+04	1.292e+03	1.292e+07	2.193e+05	1.770e+01
7C	**5**	4.120e+04	8.240e+03	6.866e+02	6.866e+06	1.052e+04	4.120e+00
7G	**5**	6.385e+04	1.277e+04	2.798e+02	2.798e+06	2.850e+04	6.390e+00
7C	**10**	9.324e+03	9.324e+02	7.769e+01	7.769e+05	3.109e+02	9.322e-01
7G	**10**	8.673e+03	8.673e+02	1.900e+01	1.900e+05	4.131e+02	8.472e-01
7C	**20**	8.273e+03	4.136e+02	3.447e+01	3.447e+05	7.033e+01	8.269e-01
7G	**20**	7.992e+03	3.996e+02	8.757e+00	8.757e+04	1.164e+02	7.215e-01
7C	**40**	6.916e+03	1.729e+02	1.441e+01	1.441e+05	2.050e+01	6.904e-01
7G	**40**	7.130e+03	1.783e+02	3.906e+00	3.906e+04	2.310e+01	5.769e-01
10C	**3**	9.917e+03	3.306e+03	1.816e+02	1.816e+06	4.662e+02	9.917e-01
10G	**3**	7.899e+03	2.633e+03	3.887e+01	3.887e+05	3.560e+02	7.910e-01
10C	**5**	9.603e+03	1.921e+03	1.055e+02	1.055e+06	3.866e+02	9.602e-01
10G	**5**	7.815e+03	1.563e+03	2.308e+01	2.308e+05	3.338e+02	7.750e-01
10C	**10**	8.311e+03	8.311e+02	4.565e+01	4.565e+05	1.350e+02	8.310e-01
10G	**10**	7.270e+03	7.270e+02	1.073e+01	1.073e+05	1.727e+02	6.771e-01
10C	**20**	6.863e+03	3.432e+02	1.885e+01	1.885e+05	3.263e+01	6.857e-01
10G	**20**	6.359e+03	3.180e+02	4.694e+00	4.694e+04	3.090e+01	5.239e-01
10C	**40**	4.884e+03	1.221e+02	6.708e+00	6.708e+04	1.116e+01	4.868e-01
10G	**40**	5.111e+03	1.278e+02	1.886e+00	1.886e+04	7.885e+00	3.451e-01

chapter 5

Adaptive Mesh Selection Strategies for Collocation Algorithms

5.1 Introduction

At some stage of the piecewise polynomial collocation methods for solving boundary value problems, discretisation of the differential equations on a mesh is involved. The purpose of this chapter is to study the practical selection of such a mesh, with the objective of achieving a sufficiently accurate solution as cheaply as possible.

We will start by introducing some basic concepts. For comparison purposes, it will then be followed by reconsidering some well known mesh selection algorithms. Subsequently, we will introduce and discuss a proposed criterion function to be used in adaptive mesh selection algorithms. Finally some results of numerical comparisons are presented.

The basic problem considered is the two-point boundary value problem

$$\boldsymbol{x}'(t) = \boldsymbol{A}(t)\,\boldsymbol{x}(t) + \boldsymbol{y}(t), \qquad a < t < b \qquad \ldots(5.1a)$$

$$\begin{aligned} \boldsymbol{B}_a\,\boldsymbol{x}(a) &= \boldsymbol{\beta}_1 \\ \boldsymbol{B}_b\,\boldsymbol{x}(b) &= \boldsymbol{\beta}_2 \end{aligned} \qquad \ldots(5.1b)$$

Here we assume that there is an unique solution $\boldsymbol{x}(t)$ that we wish to compute and that matrix valued function $\boldsymbol{A}(t)$ is sufficiently smooth.

In the collocation process using q collocation points per subinterval, the discretisation of the differential equations will be carried out on the mesh

$$\pi : \; a = t_1 < t_2 < \ldots < t_w < t_{w+1} = b \qquad \ldots(5.2)$$

where w denotes number of initial subintervals.

The mesh sizes are defined as $h_i = t_{i+1} - t_i$, $1 \le i \le w$ and the maximum mesh size is $h = \max_i h_i$. For convenience, the resulting collocation solution on the mesh π will be denoted by $x_\pi(t)$ rather than $x_{wq}(t)$ as in previous chapters.

Since we shall be interested mainly in developing our proposed criterion function and examining its performance by carrying out numerical comparisons with those well known mesh selection strategies, here the problem of selecting a good mesh is considered independently from error estimates. To be precise, in the numerical experiments we shall directly examine and compare the performance of algorithms by looking at the actual error at certain number of collocation points w.

For simplicity and to make numerical comparisons more straightforward and more clear, in implementing the adaptive mesh selection algorithms to be presented here the number of subintervals will be incremented by one subinterval per iteration. This means that neither the possibility of adding a number of knots nor reducing the number of knots will be pursued here, however it will be considered in the next chapter.

5.2 Some Basic Concepts

In the three following subsections some basic concepts underlying our work will be introduced.

5.2.1 Structure of Adaptive Mesh Selection Algorithms

The idea of adaptive refinement of a mesh and a redistribution of meshes are now well established. A number of robust mesh selection algorithms are available and have been applied widely in developing many software packages.

There is not much theoretical justification for the different strategies which have been used widely. Even though some strategies based upon asymptotic formulas perform quite satisfactorily in many practical applications, despite this lack of rigorous theoretical justification. It is important to realise that in the development of

mesh selection algorithms that the choice of a good mesh is not very sensitive; i.e. often there is a wide range of acceptable meshes of a given size w for a given boundary value problem, even when a uniform mesh of the same size yields poor results.

In developing mesh selection strategy, the aim is to find an algorithm which determines a sequence of partitions defined by the points (knots) in an adaptive way, so that an accurate solution to the problem is obtained with as small number of subintervals w as possible.

To be more precise, let TOL be the desired tolerance, and suppose we wish to compute the approximate collocation solution $\boldsymbol{x}_\pi(t)$ of BVP (5.1) over partition (5.2) using q collocation point per subinterval. Having computed $\boldsymbol{x}_\pi(t)$ on mesh π, our main goal is to efficiently determine a new partition of $[a, b]$

$$\pi^* : \quad a = t_1^* < t_2^* < \ldots < t_{w^*}^* < t_{w^*+1}^* = b \qquad \ldots(5.3)$$

such that w^* is small but if the collocation solution is computed using π^* then the global error $\boldsymbol{e}(t)$ satisfies

$$\|\boldsymbol{e}\| \equiv \max_{a \le t \le b} \|\boldsymbol{e}(t)\| \le TOL \qquad \ldots(5.4)$$

Needless to say that the mesh (5.3) and collocation solution may need to be repeatedly updated until (5.4) is satisfied.

It is clear that some criterion function $\tau_i(t)$ is needed to construct a new mesh π^*. Basically, there are two approaches to do this, firstly by trying to redistribute $\tau_i(t)$ which should be some measure of the error in the i^{th}-subinterval, such that they have approximately the same norm in whole range $[a, b]$, i.e. by requiring

$$\|\tau_i\| = \text{constant} = \check{E}, \qquad 1 \le i \le w \qquad \ldots(5.5)$$

secondly, we directly attempt to minimise $\tau_i(t)$ simply by searching for the subinterval(s) having large $\|\tau_i\|$ and then subdivide these subintervals.

Having chosen a particular criterion function, an adaptive mesh selection algorithm can be constructed by means of an iterative procedure, adding or removing points as necessary to equilibrate to level $\check{E}$. An approximately equilibrating mesh is produced such that the equation (5.4) is fulfilled.

An outline of basic structure of the procedure is as follows

1. *Compute the first stage collocation solution on initial mesh* π
2. *Evaluate the global error and check whether either*
   - *(5.4) is satisfied or*
   - *number of iterations exceed some prescribed constant* I_{max} *or*
   - *number of subintervals greater than some constant* w_{max}

   → *and then break*
3. *Evaluate the criterion function*
4. *Construct a new partition* π^* *based on result of step-3*
5. *Compute the collocation solution on new mesh* π^*
6. *Repeat step-2*

5.2.2 Error Equidistribution and Criterion Function

A particular approach to adaptive collocation schemes was introduced by de Boor [22]. In the paper, de Boor proposed to equidistribute some certain measure of the error in each subinterval $[t_i, t_{i+1}]$. Furthermore, a paper of Pereyra and Sewell [40] discusses in some detail the concept of equidistribution for discrete solutions. In the paper they extend the idea of de Boor to discrete variable approximation for more general for boundary value problems.

Recall τ_i the local error measure mentioned in the previous section. The requirement using equation (5.5) may be regarded as the basic definition of the equidistribution concept. However, as described in Ascher et al. [10], since this error measure is not independent of its associated subinterval $[t_i, t_{i+1}]$ and in general τ_i increases as the mesh size increases, it will turn out to be convenient to

consider a corresponding error measure ϕ_i which only varies linearly with h_i , i.e. they are related by

$$\phi_i = T_i h_i \qquad \ldots(5.6)$$

where T_i is independent of mesh size h_i.

Having chosen a suitable criterion function τ_i (or perhaps, the converted one of the form (5.6)) the new mesh π^* may be found by requiring $\max_{1\le i\le w} \| \phi_i \|$ to be minimised. This brings us to the minimax problem with one constraint as follows

find the set of points $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \} \subset (a, b)$ such that

$$\max_{1\le i\le w^*} \| T_i \| (t_{i+1}^* - t_i^*) \quad \text{is minimum ,}$$

where $(t_{i+1}^* - t_i^*) > 0$ and the sum of $(t_{i+1}^* - t_i^*)$ must satisfy

$$\sum_{i=1}^{w^*} (t_{i+1}^* - t_i^*) = (b-a).$$

The above optimisation problem can be solved, simply by making all $\phi_i = T_i h_i$ equal to the same constant $\check{E}$. This result enables us to define formally the concept of equidistribution as follows

Definition : A mesh points is said to be equidistributing with respect to the function $T(t)$ if and only if

$$\| T(t) \|_i h_i = \text{constant}, \qquad i = 1, 2, \ldots, w \qquad \ldots(5.7)$$

For the sake of generality, especially to understand what has been done by de Boor, we can extend the definition (5.7) by considering a more general function $\rho(t)$, instead of just discrete values function $\| T(t) \|_i$ on partition π. Let us assume that a positive valued function $\rho(t)$ is continuous and sufficiently smooth. With these assumptions, we come to the following definition.

Definition : A mesh is said to be equidistributing with respect to a monitor function $\rho(t)$ on interval $[a, b]$ if for some constant E_0

$$\int_{t_i}^{t_{i+1}} \rho(t)dt = E_0, \quad i = 1, 2, \ldots, w \qquad \ldots(5.7a)$$

From which, it follows that

$$E_0 = \frac{1}{w} \int_a^b \rho(t)dt$$

5.2.3 Mesh Placement and Mesh Subdivision Strategy

The adaptive mesh selection algorithms can be distinguished into two types, firstly mesh subdivision algorithm where additional knot(s) are inserted into a given mesh. The second one is called equidistribution or mesh placement where a new mesh is chosen at each stage so that some criterion function should have the same value in each subinterval. The second type enables us to obtain a new mesh which is completely different with the previous mesh. Recall the basic structure of the adaptive mesh selection algorithms in §5.2.1, apparently these two types only differ in step-4, however the effectiveness and efficiency of the algorithms could be different significantly.

The procedure for mesh subdivision is straightforward and is much more simple than mesh placement. In this procedure it is expected that the subinterval with maximum $\| \tau_i \|$ determined using some criterion function gives maximum effect on the error $\| e \|$, consequently, it seems sensible to subdivide the subinterval having the largest $\| \tau_i \|$.

For mesh subdivision algorithm with one point increment, an outline of the basic procedure is as follows

1. Solve the BVP using a crude initial mesh points
2. Evaluate the criterion $\|\tau_i\|$, $i = 1, 2, \ldots, w$
3. Searching for the subinterval which has maximum $\|\tau_i\|$
4. Halve this subinterval
5. Repeat first step till either (5.4) is satisfied or $w > w_{max}$, for some constant w_{max}

Note that, the basic procedure above can be developed further to obtain an adaptive mesh algorithm with multiple subdivisions.

For mesh placement algorithms, a special procedure is needed which involves setting up and finding the inverse of a certain function. A detailed description on this can be found in §5.4.2.

5.3 Some Well Known Criterion Functions

In the following subsections, we shall examine some well known criterion functions widely used in applications. Our main attention is the maximum residual and de Boor criterion functions which will be employed for numerical comparisons. Some other criteria will be described briefly.

5.3.1 Maximum Residual

Residuals have been commonly used to estimate the local errors for mesh selection. Carey and Humphrey [14] studied in detail the use of residual as criterion function in developing adaptive mesh selection algorithms. In their work they also found some empirical relations for some specific problems for which they come to conclusion that reducing the residuals in some region will reduce the residual in whole interval and consequently, the global error. This result is also pointed out in Seleman's thesis [48].

To recall, for a given mesh π of (5.2) and q given collocation points in each subinterval, the residual $r_\pi(t)$ for the boundary value problem (5.1a-5.1b) is determined by

$$r_\pi(t) = (D - A)x_\pi(t) - y(t) \qquad \ldots(5.8)$$

where D denotes the differential operator defined in chapter 2.

Implementing this strategy which will be called the *MR* strategy is fairly simple. The main task is to search for the subinterval having the largest residual and then subdivide the subinterval into two equal subintervals. As we can see in equation (5.8), the residual can be evaluated at any point straightforwardly, nevertheless obtaining its maximum is not a cheap computation task, especially if the mesh is a non-uniform one. Obviously, the success of a maximum residual strategy also depends on the success of estimating the largest residual. There are various ways that an approximate residual can be found and used for this purpose, here we will make use the polynomial interpolation discussed in chapter 4 to obtain an estimate of maximum residual.

5.3.2 De Boor's Algorithm

De Boor's paper of 1973 [22] is recognised as one of the most outstanding contribution in developing adaptive mesh selection algorithms. De Boor introduced a criterion function based on the error analysis given in de Boor and Swartz's paper [23]. A comprehensive paper of Russell and Christiansen [45] discusses further de Boor's idea, and this is implemented in the COLSYS code by Ascher, Christiansen and Russell as described in Ascher et al. [10].

For comparison purposes, first of all we shall take a look at de Boor's idea in constructing an adaptive mesh placement algorithm.

Let us start by reconsidering some theoretical results about collocation approximation method and its error estimates which can be found in [22,45].

For $t \in (t_i, t_{i+1})$, under certain conditions it is known that for some integer $d > q$ the global error $e(t)$ satisfies the local inequality

$$\|e\|_i = \|x_\pi(t) - x(t)\|_i \leq Ch_i^{q+1} \|x^{(q+1)}(t)\|_i + O(h_i^{q+2}) + O(h^d) \quad \ldots(5.9)$$

and

$$\|x_\pi(t) - x(t)\|_i \leq O(h^d), \quad 1 \leq i \leq w+1$$

where h_i, h denote the interval sizes and C is a constant determined by

$$C = \frac{1}{2^{q+1} q!} \max_{-1 \leq t \leq 1} \{ \int_{-1}^{t} \prod_{j=1}^{q} (s - \xi_j) ds \} \quad \ldots(5.10)$$

where ξ_j are the collocation points in [-1,1]

It is also shown that closer examination of the error reveals that (5.9) can be replaced by the equality

$$\|x_\pi(t) - x(t)\|_i = Ch_i^{q+1} \|x^{(q+1)}(t)\|_i + O(h^{q+2}) \quad \ldots(5.11)$$

This implies that, for sufficiently small h

$$\|e\| = \|x_\pi(t) - x(t)\| \leq C \max_i \{ h_i^{q+1} \|x^{(q+1)}(t)\|_i \} \quad \ldots(5.12)$$

and therefore suggest that break points $t_2, t_3, \ldots, t_w$ be placed so as to minimise the maximum of local terms $h_i^{q+1} \|x^{(q+1)}(t)\|_i$. This can be achieved by requiring

$$h_i^{q+1} \|x^{(q+1)}(t)\|_i = \text{constant}, \quad i = 1, 2, \ldots, w \quad \ldots(5.13)$$

Based on (5.13) de Boor constructed an adaptive mesh selection procedure which produces a complete new mesh in each iteration, in other words it is a mesh placement algorithm. Due to unavailability of $x^{(q+1)}(t)$, and since $x_\pi^{(q+1)}(t)$ is zero within each subinterval, de Boor proposed a numerical scheme to obtain an approximate for the terms $\|x^{(q+1)}(t)\|_i$ using values in neighbouring subintervals. The piecewise constant function to approximate $x^{(q+1)}(t)$ is determined using

$$DB(t) = \begin{cases} \dfrac{2\|\Delta\, x_\pi^{(q)}(t_{3/2})\|}{t_3 - t_1}, & t \in (t_1, t_2) \\ \dfrac{\|\Delta\, x_\pi^{(q)}(t_{i-1/2})\|}{t_{i+1} - t_{i-1}} + \dfrac{\|\Delta\, x_\pi^{(q)}(t_{i+1/2})\|}{t_{i+2} - t_i}, & t \in (t_i, t_{i+1}),\ 1 < i < w. \\ \dfrac{2\|\Delta\, x_\pi^{(q)}(t_{w-1/2})\|}{t_{w+1} - t_{w-1}}, & t \in (t_w, t_{w+1}) \end{cases} \quad \ldots(5.14)$$

Here, Δ denotes the forward difference operator with $t_{i+1/2} = (t_i + t_{i+1})/2$. As we can see this amounts to taking for $DB(t)$ on the subinterval (t_i, t_{i+1}) as the slope at middle point $t_{i+1/2}$ of the parabola which interpolates the q^{th} derivative of the approximate solution $x_\pi(t)$ at $t_{i-1/2}$, $t_{i+1/2}$ and $t_{i+3/2}$.

In order to make a clearer comparison with our mesh subdivison algorithm, we slightly modify de Boor's algorithm by searching for the i_*^{th} subinterval $1 \le i_* \le w$ such that

$$h_{i_*}^{q+1} \| x^{(q+1)}(t) \|_{i_*} = \max_i \{ h_i^{q+1} \| x^{(q+1)}(t) \|_i \}, \quad 1 \le i \le w$$

where $x^{(q+1)}(t)$ is approximated by piecewise constant function DB. This procedure is called de Boor mesh subdivision algorithm.

5.3.3 Other Criterion Functions

In chapter 4 we have discussed some error estimates for the numerical solution of a linear first order system of ordinary differential equations by piecewise polynomial collocation which are based on consideration of the differential operator involved and related matrices and on the residual. It is also shown that a significant advantage may be obtained by considering the form of the residual rather than just its norm. This, in particular, gives us an error estimate E^* which provides an estimate of the error as a function of variable t. Unfortunately, some results of early numerical experiments clearly indicate that direct attempts to use those error estimates, in

particular E^*, as criterion function in developing an adaptive mesh selection algorithm for solving system BPVs give unsatisfactory results. Though, Wright, Ahmed and Seleman [60] have shown that if the influence of the behaviour in one subinterval on the error in others is taken into account then some criterion functions based on those error estimates for solving single higher order boundary value problems may give a good results in some cases. These modified criteria turn out, however, to be very expensive and their practical utility is doubtful.

For solving single higher order boundary value problems, there have been many suggestions for criteria. Some of these aim to reflect some measure of smoothness of the approximate solution, for example the magnitude of a particular derivative of the collocation solution in each subinterval. For this purpose, White [53] suggested the use of arc-length, while Dodson [24] proposed to approximate the particular derivative by differentiating the piecewise linear function that interpolates the derivative at the middle of subintervals. Other criterion functions relate to some measure of error. These criterion functions, however, will not be considered further here except to remark that the de Boor algorithm which involves approximating the particular derivative of $x_\pi(t)$ is widely used and performs quite well as pointed out by Russell and Christiansen [45]

5.4 Using $r_i h_i$ as the Criterion Function

In this section, a criterion function to be used in developing adaptive mesh selection algorithm will be introduced. Firstly we consider some motivation for choosing this criterion and then it is followed by developing a numerical scheme for mesh placement algorithm based on our criterion function.

5.4.1 Motivation for Using $r_i h_i$ as the Criterion Function

The standard analysis of collocation process for solving the boundary value problem (5.1a-5.1b) using q points per subinterval over mesh points (5.2), yields the error $e(t)$ which can be expressed as

$$e(t) = \sum_{i=1}^{w} \int_{t_i}^{t_{i+1}} G(t,s)r(s)ds \qquad \ldots(5.15)$$

where $r(s)$ is the residual defined in equation (5.8) and $G(t,s)$ denotes the Green's function. For $t \in (t_i, t_{i+1})$ let us consider the terms

$$e_i(t) = \int_{t_i}^{t_{i+1}} G(t,s)r(s)ds\,, \quad i = 1, 2, \ldots, w. \qquad \ldots(5.16)$$

It is notable that firstly the residual $r(s)$ is local in nature and has been used as criterion function in adaptive mesh selection algorithm [10,14,59], secondly regarding the relationship between the global error and local terms Russell and Christiansen [45] have pointed out that the global error is asymptotically dominated by local term when Gauss points are used. Lastly, in applications usually the Green matrix function is diagonally dominant [29,31]. These results suggest that $e_i(t)$ in equation (5.16) is dominated by the residual $r(s)$ and $G(t,s)$ may be taken to be constant for $t \in (t_i, t_{i+1})$, $s \in (t_i, t_{i+1})$ and zero elsewhere, we then have the local term

$$\| e_i(t) \| = \| C \int_{t_i}^{t_{i+1}} r(s)ds \| \le C \int_{t_i}^{t_{i+1}} \| r(s) \| ds = C \| r \|_i h_i \qquad \ldots(5.17)$$

For some constant C. h_i, as usual, denotes the mesh size of i^{th}-subinterval.

Since the equation (5.17) reflects some measurement of the error and is local in nature, it seems reasonable to take it as a criterion for an adaptive mesh selection algorithm which will be called *RH* mesh selection algorithm.

Furthermore, taking equation (5.17) as criterion is also suggested by the following results. Suppose the collocation points are the zeros of certain orthogonal polynomial and we take the number of collocation points q to be even, i.e. the ODE (5.1a) will not be collocated at the middle of subintervals. Using the fact that the residual $r(t)$ is zero at the collocation points ξ_{ij}, it has been shown in [45] that in each subinterval the residual satisfies

$$\boldsymbol{r}_{\pi}(t) = (\boldsymbol{x}^{(q+1)}(t_{i+1/2}) / (q!) \prod_{j=1}^{q} (t - \xi_{ij}) + O(h^{q+1}) \qquad \ldots(5.18)$$

Since the middle of subinterval is not the collocation point solving (5.18) for $\boldsymbol{x}^{(q+1)}$ in term of $\boldsymbol{r}(t_{i+1/2})$ gives

$$\boldsymbol{x}^{(q+1)}(t_{i+1/2}) = \frac{q!}{\prod_{j=1}^{q}(t_{i+\frac{1}{2}} - \xi_{ij})} r_{\pi}(t_{i+\frac{1}{2}})$$

By taking the norm of the last equation and substituting into (5.11) we then have an error estimate

$$\|\boldsymbol{e}\|_i = C_0 \|\boldsymbol{r}(t_{i+1/2})\|_i h_i + O(h^{q+2}) \qquad \ldots(5.19)$$

This suggests, corresponding to the discussion in §5.2.2, that trying to equidistribute the equation (5.19) introduce a mesh selection algorithm with respect to the function $T(t)$ (see equation (5.6))

$$\|T(t)\|_i \equiv \|\boldsymbol{r}(t_{i+1/2})\|,$$

hence equation (5.17) can be regarded as the general form of equation (5.19).

It is notable that in computing the error estimate $\boldsymbol{E}^*$, we have to evaluate residuals and construct an approximate residual $\boldsymbol{r}^*(t)$. This, in turn, makes the application of criterion $r_i h_i$ more convenient and less expensive if we develop an adaptive mesh selection algorithm while also using the error estimate $\boldsymbol{E}^*$, since we can make use the computed $\boldsymbol{r}^*(t)$ for approximating the local terms $\|\boldsymbol{r}(t)\|_i$. It is also notable that the *RH* algorithm is as cheap as the *MR* algorithm since the main cost is to evaluate the residual in each subinterval.

5.4.2 Developing the Scheme for Equidistributing the Terms $r_i h_i$

We will now employ the equation (5.17) in constructing two adaptive mesh selection algorithms, i.e. *RH* mesh subdivision and *RH* mesh placement algorithms.

The *RH* mesh subdivision algorithm is quite simple in implementation, where the main task consists of searching for the subinterval which has maximum $\|r(t)\|_i h_i$, and then subdividing this subinterval into two equal subintervals.

Unlike the *RH* mesh subdivision algorithm, the *RH* mesh placement algorithm needs a special scheme to equdistribute the local terms $\|r(t)\|_i h_i$ which will be developed in this section.

The residual $r(t)$ can be related to mesh size h_i by equation

$$\|r\|_i = k_i h_i^s \qquad \ldots(5.20)$$

for some positive constant k_i and some constant integer s.
Multiplying both sides by h_i we have

$$\|r\|_i h_i = k_i h_i^{s+1} \qquad \ldots(5.21)$$

Therefore, equidistributing the local terms in (5.17) is equivalent to equidistributing the local terms $k_i h_i^{s+1}$.

Having computed r_π on the initial mesh π, a new partition π^* of the form (5.3) producing a more accurate solution is desired.

Supposed that the width of i^{th}-subinterval in new mesh π^* is denoted by $h^*_i = (t^*_{i+1} - t^*_i)$. To equidistribute (5.21), it requires

$$k_1 h_1^{*s+1} = k_2 h_2^{*s+1} = \ldots = k_w h_w^{*s+1}$$

to give

$$\begin{aligned}
(h_1^* / h_2^*) &= (k_2 / k_1)^{1/(s+1)} \\
(h_1^* / h_3^*) &= (k_3 / k_1)^{1/(s+1)} \\
&\vdots \qquad\qquad \vdots \\
(h_1^* / h_w^*) &= (k_w / k_1)^{1/(s+1)}
\end{aligned}$$

By taking $(k_i / k_1)^{1/(s+1)}$, $1 \le i \le w$, as the slopes of a piecewise linear function with the slope in the first subinterval is set to be one, we can then construct a piecewise linear function $\theta(t)$ as follows

$$\theta(t) = \begin{cases} t - t_1 , & a < t < t_2 \\ h_1 + (k_2 / k_1)^{1/(s+1)} (t - t_2), & t_2 < t < t_3 \\ \vdots & \\ h_1 + (k_2 / k_1)^{1/(s+1)}\ h_2 + (k_3 / k_1)^{1/(s+1)} (t - t_3) , & t_3 < t < t_4 \\ \vdots & \\ h_1 + (k_2 / k_1)^{1/(s+1)}\ h_2 + \ldots + (k_w / k_1)^{1/(s+1)} (t - t_w), & t_w < t < b \end{cases} \qquad \ldots(5.22)$$

where $k_i = \|r\|_i / h_i^s$, $i = 1, 2, \ldots, w$.

Here we approximate $r(t)$ using the principal part of the residual (described in chapter 4) formed by evaluation at a suitable set of Chebyshev extrema.

Since $\theta(t)$ is a continuous and monotone increasing piecewise linear function, we can then easily compute $\theta^{-1}(t)$ and evaluate it at the (w^*+1) points

$$((i\text{-}1)\ \theta(b) / w^*), \quad 1 \le i \le (w^*+1)$$

giving us a new mesh π^*.

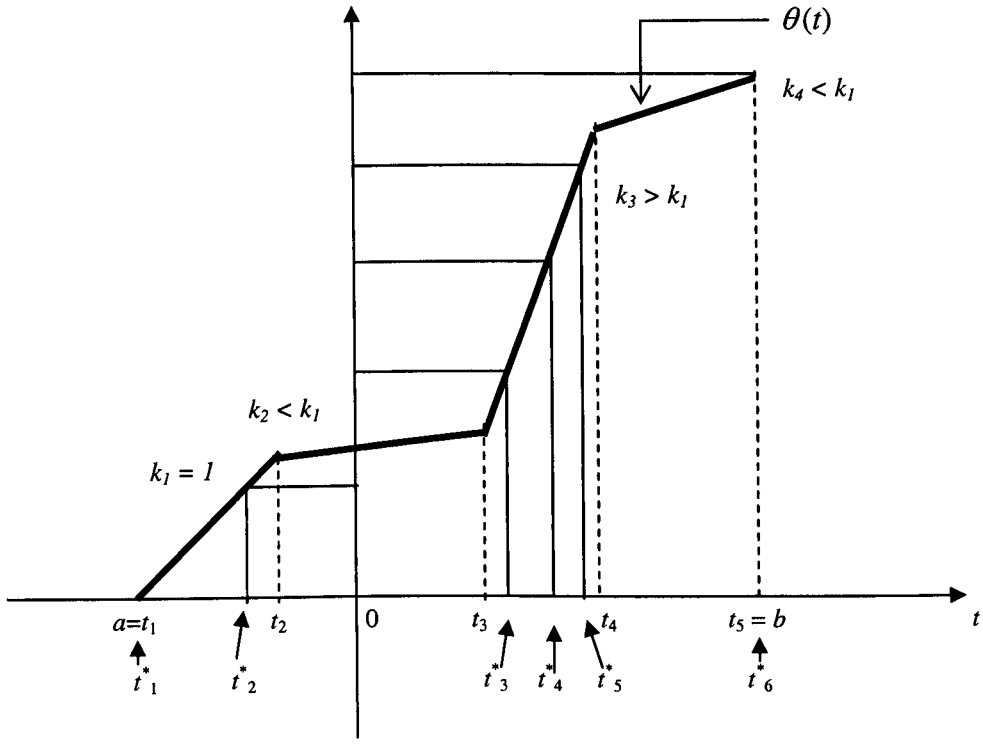
For illustration, suppose the current number of subintervals $w = 4$ and it is increased by one subinterval in the next stage, i.e. $w^* = 5$. Using the approximate residual $r^*(t)$ we then construct the piecewise linear function $\theta(t)$. The new mesh π^* having break points $\{t^*_i\}$, $1 \le i \le (w^*+1)$, is determined as follows

$$t^*_1 = a = \theta^{-1}(0) . \qquad t^*_2 = \theta^{-1}\left(\frac{\theta(b)}{w^*}\right)$$

$$t^*_3 = \theta^{-1}\left(\frac{2\ \theta(b)}{w^*}\right). \qquad t^*_4 = \theta^{-1}\left(\frac{3\ \theta(b)}{w^*}\right)$$

$$t^*_5 = \theta^{-1}\left(\frac{4\ \theta(b)}{w^*}\right). \qquad t^*_6 = b = \theta^{-1}(\theta(b))$$

A graphical illustration of the process is shown by the graph on the following page



5.5 Numerical Results

Several numerical experiments were performed with the adaptive mesh selection algorithms described in the previous sections. In order to give an impression of the performance of the algorithms we consider some examples having various features. These include some problems having either interior or boundary layers as well as problems with a singularity at an end point.

In the tables below displaying some results of numerical experiments, *MR* stands for *maximum residual* indicating the maximum residual is used as the criterion function. *RH* and *DB* indicate max *RH* and de Boor algorithms respectively, while *subd* and *plc* stand for *subdivision* and *placement* algorithm. The integer in the square bracket [...] shows the number of subintervals in the layer regions. The last abbreviations the capital letters *C* and *G* indicate that Chebyshev and Gauss points are used in the numerical experiments.

In all computations, we start with four-equal subintervals and then increase the number of subintervals w by one. At certain w, we retrieve the actual error to be displayed in the tables.

As the first example we consider the following problem

Problem 1 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 4 & -2/t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -2 \end{pmatrix}$$

furnished by boundary conditions

$x_2(0) = 0$ and $x_1(1) = 5.5$

The analytical solution is given by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \frac{5\sinh(2t)}{t\sinh(2)} \\ \frac{10t\cosh(2t) - 5\sinh(2t)}{t^2\sinh(2)} \end{pmatrix}$$

This boundary value problem is taken from Russell and Shampine [47] in which they discuss some collocation methods for dealing with singular boundary value problems. The problem has a singularity at $t = 0$, but only in the coefficient and its solution is smooth. In their paper, it is also shown that numerically the maximum error always occurred at the left boundary. In this example, it is expected that the adaptive mesh selection algorithms should concentrate break points near left boundary.

Tables *5.1A* and *5.1B* contain the results obtained by using two and five collocation points per subinterval respectively. By looking at the actual error, it is clear that all strategies perform quite well. In both tables we can see that the accuracy improves smoothly as the number of subinterval increases indicating the convergence of the collocation solution.

It is interesting to note that for the subdivision strategies, all criterion functions used produce identical results, this means if we use subdivision strategy for this problem the cheapest criteria *MR* may be adequate to obtain a sufficiently accurate

solution. However, the results indicate that the *RH* mesh placement algorithm, over all, gives the most sensible results.

Looking at number of break points in the subinterval (0, 0.01) placed by each algorithms, apart from de Boor placement algorithm, all algorithms put the same amount. This means that the algorithms react by putting more break points as required in the region where the worst error may occur.

Comparing the choice of collocation points, using Chebyshev points produce comparable results to those using Gauss points, though slightly poorer.

Table 5.1A

(2 collocation points per subinterval)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	2C	3.015e-03 [0]	3.015e-03 [0]	3.015e-03 [0]	2.670e-03 [0]	3.855e-03 [0]
	2G	1.846e-03 [0]	1.846e-03 [0]	1.846e-03 [0]	1.086e-03 [0]	2.516e-03 [0]
10	2C	5.674e-04 [0]	5.674e-04 [0]	5.674e-04 [0]	5.414e-04 [1]	1.849e-03 [0]
	2G	2.458e-04 [0]	2.458e-04 [0]	2.458e-04 [0]	1.213e-04 [0]	1.654e-04 [1]
20	2C	1.188e-04 [1]	1.188e-04 [1]	1.188e-04 [1]	1.225e-04 [1]	2.512e-04 [0]
	2G	3.193e-05 [1]	3.193e-05 [1]	3.193e-05 [1]	1.430e-05 [1]	9.114e-05 [2]
40	2C	2.733e-05 [3]	2.733e-05 [3]	2.733e-05 [3]	2.917e-05 [3]	1.962e-04 [0]
	2G	4.074e-06 [3]	4.074e-06 [3]	4.074e-06 [3]	1.730e-06 [3]	1.123e-05 [2]

Table 5.1B

(5 collocation points per subinterval)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	5C	2.445e-08 [0]	2.445e-08 [0]	2.445e-08 [0]	5.320e-09 [0]	1.730e-08 [0]
	5G	1.531e-08 [0]	1.531e-08 [0]	1.531e-08 [0]	3.457e-09 [0]	1.098e-08 [0]
10	5C	4.308e-10 [0]	4.308e-10 [0]	4.308e-10 [0]	7.831e-11 [0]	1.309e-10 [1]
	5G	2.723e-10 [0]	2.723e-10 [0]	2.723e-10 [0]	5.134e-11 [0]	8.483e-11 [1]
20	5C	7.153e-12 [1]	7.153e-12 [1]	7.153e-12 [1]	1.206e-12 [1]	5.821e-12 [2]
	5G	4.539e-12 [1]	4.539e-12 [1]	4.539e-12 [1]	7.940e-13 [1]	5.407e-12 [1]
40	5C	1.375e-12 [3]	1.375e-12 [3]	1.375e-12 [3]	1.288e-12 [3]	7.038e-13 [2]
	5G	1.375e-12 [3]	1.375e-12 [3]	1.375e-12 [3]	7.739e-13 [3]	1.353e-12 [4]

We take the following BVP as an illustrative example since its solution is highly oscillatory near left boundary. Hence we expect to have a chance to examine how the adaptive mesh selection algorithms handle this situation.

The problem considered is

Problem 2 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -1/t^4 & -2/t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

with boundary conditions at both end points

$$x_1(1/(3\pi)) = 0$$

and

$$x_1(2) = sin(1).$$

The analytical solution is given by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sin(1/t) \\ -(\cos(1/t))/t^2 \end{pmatrix}$$

Tables *5.2A* and *5.2B* display results of numerical experiments using two and five collocation points per subinterval respectively. As in the previous example, various mesh selection strategies are implemented.

From *Table 5.2A*, we can see using the *MR* algorithm with Chebyshev points, it is not easy to observe the convergence of the solution after doing almost 40 iterations, since after performing with $w = 40$ the accuracy does not improve, while the other criteria reduce the actual errors though not in a dramatic way.

The most notable result from this table is that in the subinterval (0, 0.1) the *MR* algorithm puts more points than others, but it does not produce reasonable solutions. Comparing the performance of all algorithms, it is again observed that the *RH* placement algorithm gives the most sensible results.

Looking at the last two rows of *Table 5.2B* and comparing the columns under heading *MR* and *DB*, it is clear that in the subinterval (0, 0.1) the *MR* algorithm puts many more break points than *DB* algorithm, but in fact it gives a worse solution. On the other hand, the *RH* algorithm places less points than *MR*, and produces significantly better accuracy. It is also observed that the *RH* algorithm is slightly better than de Boor algorithm.

From this numerical experiment we can point out two important results, firstly the *RH* algorithm places the break points in such a way that the accuracy improves significantly; secondly, the *MR* algorithm subdivides some intervals without reducing the actual error.

Though using Chebyshev points result in reasonable accuracy, for this problem we found that in most cases using Gauss collocation points give more accurate solutions.

Table 5.2A

(2 collocation points)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	2C	4.561e+01 [1]	4.561e+01 [1]	6.440e+01 [0]	4.256e+01 [2]	4.851e+01 [1]
	2G	1.684e+01 [1]	1.684e+01 [1]	4.636e+01 [0]	1.528e+01 [2]	2.003e+01 [1]
10	2C	2.065e+01 [6]	1.271e+01 [5]	2.499e+01 [5]	8.382e+00 [5]	1.797e+01 [4]
	2G	3.672e+00 [6]	1.618e+00 [5]	3.848e+00 [5]	3.506e-01 [6]	1.871e+00 [4]
20	2C	2.047e+00 [15]	3.541e+00 [12]	3.465e+00 [12]	2.341e+00 [11]	3.981e+00 [9]
	2G	3.032e-01 [15]	1.447e-01 [12]	1.923e-01 [12]	9.732e-02 [11]	1.831e-01 [9]
40	2C	2.673e+00 [32]	6.242e-01 [24]	6.812e-01 [24]	5.908e-01 [23]	1.032e+00 [18]
	2G	1.729e-01 [32]	1.677e-02 [24]	2.321e-02 [24]	8.408e-03 [23]	1.919e-02 [19]

Table 5.2B

(5 points per subintervals)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	5C	3.350e-01 [1]	3.350e-01 [1]	8.236e+00 [0]	9.019e-02 [2]	2.772e+00 [1]
	5G	2.295e-01 [1]	2.295e-01 [1]	6.539e+00 [0]	8.494e-03 [2]	1.993e+00 [1]
10	5C	1.021e-02 [5]	9.400e-03 [4]	1.719e-02 [3]	6.724e-04 [4]	1.962e-02 [2]
	5G	5.205e-03 [5]	1.922e-03 [4]	8.058e-03 [3]	3.049e-04 [4]	7.600e-03 [2]
20	5C	1.238e-04 [11]	3.653e-05 [10]	1.066e-03 [6]	6.622e-06 [9]	3.123e-04 [6]
	5G	3.324e-05 [11]	1.880e-05 [10]	5.307e-04 [6]	3.190e-06 [9]	1.233e-04 [6]
40	5C	3.972e-05 [23]	1.955e-06 [20]	7.238e-06 [15]	1.185e-07 [19]	4.843e-06 [12]
	5G	4.929e-06 [23]	3.582e-07 [20]	3.422e-06 [15]	4.659e-08 [19]	3.902e-06 [12]

Now we turn to boundary value problem having an interior layer centred at the middle of specified interval. The problem taken from Aziz [12] is as follows

Problem 3 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -(3\varepsilon)/\sqrt{\varepsilon + t^2} & 0 \end{pmatrix}$$

where $\varepsilon = 1/\mu$, μ is the problem parameter with $|\mu| >> 1$

The boundary conditions are

$$x_1(-0.1) = -\frac{0.1}{\sqrt{(\varepsilon+0.01)}} \quad \text{and} \quad x_1(0.1) = \frac{0.1}{\sqrt{(\varepsilon+0.01)}}.$$

The solution having an interior layer of thickness $\sqrt{\varepsilon}$ is given by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} t/\sqrt{\varepsilon+t^2} \\ (1-t^2(\varepsilon+t^2)^{-1}) \ / \ \sqrt{\varepsilon+t^2} \end{pmatrix}$$

For this problem we present some numerical results obtained by taking problem parameter $\mu = 10^4$ and $\mu = 10^8$. Tables *5.3A* and *5.3B* obtained by choosing $\mu = 10^4$ display the results using three and five collocation points respectively, while tables *5.3C* and *5.3D* resulted by taking $\mu = 10^8$ shows the results using eight and twelve collocation points respectively. The Chebyshev zeros and Gauss points are applied in all computations.

Table 5.3A shows that all cases indicate that the collocation solutions are converging as the number of subintervals increases. Occasionally, using Gauss points produces a bit more accurate solution, while using Chebyshev points steadily gives competitive results. Again, we observe that *MR* algorithm puts too many points in the layer region (-0.01, 0.01) producing poor accuracy. Similar results to those in *Table 5.3A* are observed in *Table 5.3B*.

With a severe interior layer in the middle interval, the results of numerical experiments displayed in tables *5.3C-5.3D* show very clearly that de Boor algorithm performs badly in both subdivision and mesh placement strategies, while the cheapest scheme, the *MR* algorithm works pretty well though not as good as the *RH* algorithm. Looking at the number of break points placed in the layer region (-0.0001, 0.0001), in all cases the de Boor algorithm fails to put more points in the region where they are required and consequently gives very poor results. This might be caused by the fact that the de Boor algorithm involves the interval size in constructing the linear piecewise constant function $DB(t)$ [see equation (5.14)] in such a way, if there is a drastic decrease in the interval size in some region then $DB(t)$ might increase dramatically. As a result the collocation process may continue by subdividing this subinterval until it reaches the stage where the other term of $DB(t)$ is small enough to compensate the interval size effect. On the other hand, in

the *RH* algorithm this will not occur since decreasing the interval size will automatically result in decreasing the value of $\theta(t)$.

These two tables also indicate the superiority of the Gauss points, where in most cases they produce better approximate solutions.

Table 5.3A

(problem parameter $\mu = 10^4$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	3C	4.387e+01 [1]	4.387e+01 [1]	3.646e+01 [1]	4.818e+01 [2]	8.215e+01 [0]
	3G	1.121e+02 [1]	9.199e+00 [1]	1.049e+02 [1]	3.484e+01 [2]	7.638e+01 [0]
10	3C	1.055e+00 [3]	1.055e+00 [3]	7.062e+00 [1]	7.764e-01 [3]	3.843e+00 [3]
	3G	8.846e-02 [3]	8.846e-02 [3]	1.929e+00 [1]	1.052e-01 [3]	7.110e-01 [2]
20	3C	1.443e-01 [9]	5.814e-02 [7]	5.814e-02 [7]	4.101e-02 [7]	5.207e-02 [5]
	3G	1.644e-02 [9]	1.523e-02 [7]	1.523e-02 [7]	6.155e-03 [7]	1.664e-02 [6]
40	3C	1.443e-02 [21]	1.830e-03 [15]	5.101e-03 [13]	2.928e-03 [15]	4.056e-03 [12]
	3G	1.997e-03 [21]	7.294e-04 [15]	1.266e-03 [13]	2.801e-04 [15]	1.165e-03 [12]

Table 5.3B

(problem parameter $\mu = 10^4$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	5C	1.109e+01 [1]	1.109e+01 [1]	1.998e+01 [1]	6.571e+00 [2]	2.532e+01 [0]
	5G	1.811e+01 [1]	1.811e+01 [1]	2.690e+01 [1]	6.873e+00 [0]	2.739e+01 [0]
10	5C	5.586e-03 [3]	5.586e-03 [3]	4.894e-01 [1]	5.342e-03 [3]	7.808e-01 [1]
	5G	3.592e-03 [3]	3.592e-03 [3]	1.928e-01 [1]	1.657e-03 [3]	5.466e-02 [2]
20	5C	1.992e-03 [9]	2.481e-04 [7]	4.926e-01 [1]	1.214e-04 [7]	5.314e-04 [5]
	5G	5.548e-04 [9]	9.341e-05 [7]	6.528e-04 [5]	3.595e-05 [7]	8.244e-04 [5]
40	5C	1.485e-05 [17]	5.627e-06 [13]	1.476e-05 [11]	1.437e-06 [15]	1.362e-05 [9]
	5G	7.941e-06 [17]	1.921e-06 [13]	7.900e-06 [11]	3.857e-07 [15]	5.510e-06 [10]

Table 5.3C

(problem parameter $\mu = 10^8$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	8C	6.112e+03 [1]	2.442e+03 [1]	5.038e+02 [1]	8.833e+03 [0]	3.124e+03 [0]
	8G	6.113e+03 [1]	6.113e+03 [1]	5.082e+02 [1]	8.690e+03 [0]	3.124e+03 [0]
10	8C	9.982e+03 [1]	8.724e+03 [1]	6.162e+03 [1]	8.015e+01 [1]	9.429e+03 [1]
	8G	9.984e+03 [1]	9.894e+03 [1]	6.109e+03 [1]	7.336e+00 [1]	9.375e+03 [1]
20	8C	3.706e+02 [1]	3.706e+02 [1]	9.744e+03 [1]	1.781e-01 [3]	6.714e+03 [1]
	8G	2.410e+01 [1]	2.410e+01 [1]	9.756e+03 [1]	1.979e-03 [3]	4.357e+03 [1]
40	8C	1.078e-02 [9]	1.189e-03 [9]	3.704e+02 [1]	5.582e-04 [7]	1.382e+01 [1]
	8G	2.104e-05 [9]	3.371e-06 [9]	2.410e+01 [1]	6.367e-07 [7]	8.758e-01 [1]

Table 5.3D

(problem parameter $\mu = 10^8$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
5	12C	6.106e+03 [1]	2.475e+03 [1]	5.469e+02 [1]	8.716e+03 [0]	3.259e+03 [0]
	12G	6.112e+03 [1]	2.446e+03 [1]	5.058e+02 [1]	8.666e+03 [0]	3.259e+03 [0]
10	12C	9.973e+03 [1]	8.655e+03 [1]	6.131e+03 [1]	7.341e-01 [1]	9.091e+03 [1]
	12G	9.983e+03 [1]	8.802e+03 [1]	6.047e+03 [1]	1.571e-02 [1]	7.727e+03 [1]
20	12C	1.733e+00 [1]	1.733e+00 [1]	1.150e+04 [1]	7.317e-05 [3]	1.693e+03 [1]
	12G	2.690e-02 [1]	2.690e-02 [1]	1.183e+04 [1]	3.066e-07 [3]	1.966e+03 [1]
40	12C	1.454e-06 [9]	1.883e-07 [9]	1.733e+00 [1]	1.330e-08 [7]	6.455e-01 [1]
	12G	7.241e-09 [9]	8.216e-10 [9]	2.690e-02 [1]	1.794e-09 [7]	2.270e-01 [1]

As the fourth example consider the following problem which can be found in Hemker [29] in which the phenomenon of stiffness in boundary value problems is discussed. The problem is chosen since it has a layer at the left boundary. It is a simple problem with constant coefficient but could raise difficulties for some numerical algorithms.

Problem 4 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu+1 & \mu \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu \end{pmatrix}$$

where μ, is the problem parameter and $|\mu| >> 1$.

The differential equation is accompanied by boundary conditions

$$x_1(0) = 1 + exp(-\mu-1)$$

$$x_1(1) = 1 + exp(-1).$$

The exponential vector valued function

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \exp((\mu+1)(t-1) + \exp(-t) \\ (\mu+1)\ \exp((\mu+1)(t-1) - \exp(-t) \end{pmatrix}$$

is the unique solution of the BVP.

Tables *5.4A* and *5.4B* show the results using four and seven collocation points in each subinterval respectively with problem parameter $\mu = 10^4$.

From *Table 5.4A*, we observe again that in most cases using the mesh placement algorithm with the *RH* criterion function gives the best approximate solutions.

Regarding the collocation points, though in most cases Gauss points produce a better solution, it is notable that for w = 30, *RH* and *DB* mesh placement algorithms with Gauss points dramatically fail to put required break points in the layer region, while using Chebyshev they react as expected by putting more points.

Meanwhile, in the last column of *Table 5.4B* we can see for w = 50, *DB* mesh placement algorithm behaves badly by producing very poor solution though it has previously produced a better one. We suspect that there might be a dramatic change in the interval size in some region raising this trouble.

Table 5.4A

(*problem parameter* $\mu = 10^4$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
10	4C	2.495e+03 [3]	1.159e+04 [0]	1.159e+04 [0]	1.197e+04 [0]	1.187e+04 [0]
	4G	1.427e+03 [3]	1.138e+04 [0]	1.138e+04 [0]	1.190e+04 [0]	1.177e+04 [0]
20	4C	8.839e-02 [13]	2.495e+03 [3]	4.766e+03 [2]	8.772e+03 [0]	8.919e+03 [0]
	4G	5.194e-02 [13]	7.415e+03 [0]	7.415e+03 [0]	8.528e+03 [0]	8.466e+03 [0]
30	4C	7.558e-03 [23]	6.312e-02 [13]	9.149e-02 [12]	3.549e-04 [25]	1.704e+00 [23]
	4G	4.505e-03 [23]	5.970e-01 [8]	5.970e-01 [8]	1.940e+03 [3]	5.841e+03 [1]
40	4C	2.735e-03 [33]	1.661e-03 [23]	9.346e-03 [20]	8.383e-04 [21]	3.041e-01 [11]
	4G	3.933e-04 [33]	4.580e-03 [18]	8.433e-03 [18]	3.602e-04 [24]	1.468e+01 [6]
50	4C	2.735e-03 [43]	3.864e-04 [33]	1.217e-03 [30]	9.741e-05 [34]	2.290e-02 [17]
	4G	7.257e-05 [43]	3.986e-04 [28]	5.609e-04 [28]	6.779e-05 [28]	3.906e-03 [19]
60	4C	2.735e-03 [53]	7.800e-05 [43]	3.878e-04 [39]	3.729e-05 [43]	4.559e-03 [21]
	4G	3.423e-05 [53]	6.624e-05 [38]	2.245e-04 [38]	7.873e-06 [45]	2.417e-03 [26]

Table 5.4B

(*problem parameter* $\mu = 10^4$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
10	7C	2.852e+02 [3]	7.564e+03 [0]	8.947e+03 [0]	9.333e+03 [0]	9.388e+03 [0]
	7G	1.710e+02 [3]	7.946e+03 [0]	7.946e+03 [0]	8.366e+03 [0]	8.362e+03 [0]
20	7C	3.408e-05 [13]	6.203e-04 [9]	5.622e+03 [0]	1.929e-02 [15]	6.712e+03 [0]
	7G	2.149e-05 [13]	4.383e-03 [6]	5.525e+03 [0]	3.417e+03 [0]	5.647e+03 [0]
30	7C	9.388e-07 [23]	3.884e-07 [19]	7.142e-03 [8]	2.838e-08 [21]	2.963e+03 [1]
	7G	5.620e-07 [23]	7.846e-07 [16]	1.710e+02 [3]	1.961e-08 [21]	2.423e+03 [1]
40	7C	9.275e-09 [33]	2.341e-08 [29]	5.390e-05 [13]	4.484e-09 [30]	8.300e-04 [10]
	7G	1.053e-08 [33]	2.699e-08 [26]	3.319e-05 [12]	1.911e-08 [30]	1.989e-03 [24]
50	7C	1.401e-08 [43]	1.597e-08 [39]	7.017e-05 [15]	2.643e-08 [42]	1.602e-01 [5]
	7G	4.684e-09 [43]	1.053e-08 [36]	3.057e-05 [14]	3.694e-09 [36]	4.658e+01 [3]
60	7C	1.402e-08 [53]	1.597e-08 [49]	1.418e-05 [17]	9.382e-09 [54]	3.290e-05 [13]
	7G	4.688e-09 [53]	4.691e-09 [46]	9.020e-06 [17]	5.494e-08 [51]	3.898e-06 [19]

Lastly, we consider a problem having non constant coefficient matrix. The problem which is again taken from Hemker [29] has a layer at around the right boundary $t = 1$.

The problem is

Problem 5 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu & \mu cos((5/12)\pi - t) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Where μ, $|\mu| >> 1$, is the problem parameter.

The associated boundary conditions are

$$x_1(0) = sin(5\pi/12) + exp(-\mu)$$

and

$$x_1(1) = 1 + sin((5\pi/12) - 1).$$

Here we present some results of numerical experiments for problem parameter $\mu = 10^4$ in *Table 5.5*. As we can see the cheapest strategy using the *MR* criterion function performs much better than the de Boor algorithm, though using the *RH* algorithm which is also cheap, is preferred in particular for larger w, since it clearly gives more reasonable results.

From this table we observe that in most cases the de Boor algorithm gives the worst results. For example, looking at the rows where $w = 40$, we can see that using Chebyshev zeros and Gauss points the de Boor placement algorithm puts only one and two break points respectively in the layer region while the others put many more break points. Consequently, in this case the de Boor algorithm gives the worst approximate solution.

It is observed that at the beginning of computation process the *MR* algorithm is better since it puts more break points in the layer region than the other algorithms, though for higher accuracy the *MR* algorithm places more break points than the others without producing better solution.

The numerical results also indicate clearly that the *RH* placement algorithm with either Chebyshev or Gauss points gives very sensible results.

Table 5.5

(problem parameter $\mu = 10^4$)

w	q	MR	RH-subd	DB-subd	RH-plc	DB-plc
10	5C	1.286e+03 [3]	9.071e+03 [0]	1.006e+04 [0]	1.029e+04 [0]	1.018e+04 [0]
	5G	7.592e+02 [3]	1.054e+04 [0]	9.614e+03 [0]	1.087e+04 [0]	1.058e+04 [0]
20	5C	2.462e-03 [13]	1.834e-01 [7]	7.030e+03 [0]	4.716e+02 [6]	7.695e+03 [0]
	5G	9.100e-03 [13]	1.586e+02 [4]	6.808e+03 [0]	4.984e+03 [0]	6.785e+03 [0]
30	5C	6.310e-05 [23]	6.213e-04 [17]	1.834e-01 [8]	5.505e-06 [24]	5.361e-03 [11]
	5G	4.199e-05 [23]	8.576e-04 [14]	1.824e+03 [2]	5.369e-05 [16]	4.382e+03 [1]
40	5C	2.129e-05 [33]	1.575e-05 [27]	7.240e-04 [18]	1.677e-06 [31]	2.299e+03 [1]
	5G	6.497e-06 [33]	3.436e-05 [24]	5.543e-03 [12]	1.552e-05 [22]	1.749e+03 [2]
50	5C	2.119e-05 [43]	4.540e-06 [37]	3.377e-04 [23]	2.686e-07 [40]	1.817e-01 [11]
	5G	3.385e-06 [43]	3.655e-06 [34]	3.947e-04 [22]	1.867e-07 [40]	2.642e-04 [16]
60	5C	2.121e-05 [53]	7.434e-07 [47]	1.830e-04 [26]	1.690e-07 [50]	1.371e-03 [18]
	5G	2.502e-06 [53]	8.503e-07 [44]	2.143e-04 [23]	7.124e-08 [49]	3.083e-04 [18]

Chapter 6

Predicting the Number of Subintervals Needed in the Collocation Processes

6.1 Introduction

All algorithms discussed in the previous chapter allow us to use multiple subdivisions, though in our numerical experiments there the mesh and the approximate solution were repeatedly updated using one subinterval increment until either the required precision is satisfied or the number of subinterval is greater than some constant number. In this chapter we shall make use of the results in the previous chapter by predicting the necessary number of subintervals w^* and then by using this w^* in the algorithms.

The automatic mesh placement algorithms developed in chapter 5, in particular the *RH* algorithm, perform very well and the numerical evidences show that the *RH* mesh placement algorithms we are proposing are more reliable than the others considered. Here we shall restrict consideration to the *RH* algorithm and develop a technique to estimate the number of subintervals w^* needed to reduce the error to a tolerance *TOL*. Since predicting a reasonable value for the number of subintervals needed in the collocation process is quite important to improve the efficiency of algorithms where a small value may result in time wasting by requiring further stages, while using a larger value than necessary the algorithm may spend too much time at earlier stages. Having obtained a basic algorithm for estimating w^* we shall develop it further in order to obtain more reliable algorithms.

In the spirit of previous chapters we shall consider the first order linear system of n differential equations

$$x'(t) = A(t)\,x(t) + y(t), \qquad a < t < b \qquad \ldots(6.1)$$

furnished by associated homogeneous boundary conditions at the end points.

Suppose that the interval $[a,b]$ is subdivided into w subintervals by the break points $\{t_2, t_3, \ldots, t_w\}$ to form a partition

$$\pi: \quad a = t_1 < t_2 < t_3 < \ldots < t_w < t_{w+1} = b$$

with

$$h = \max_{1 \le i \le w} h_i, \qquad h_i = t_{i+1} - t_i$$

Having obtained the collocation solution $x_\pi(t)$ based on collocation over the mesh π, a more accurate solution $x_{\pi^*}(t)$ over new partition

$$\pi^*: \quad a = t_1^* < t_2^* < t_3^* < \ldots < t_{w^*}^* < t_{w^*+1}^* = b$$

is desired. Here $w^* > 0$ denotes the number of subinterval in the new mesh. Introducing the notation for mesh sizes $h_i^* = t_{i+1}^* - t_i^*$, $1 \le i \le w^*$, we then have the maximum mesh

$$h^* = \max_{1 \le i \le w^*} h_i^*.$$

The error of some collocation methods generally yields that the error can be expressed as

$$e(t) = \int_a^b G(t,s)\, r(s)\, ds = \sum_{i=1}^{w} \int_{t_i}^{t_{i+1}} G(t,s) r(s) ds \qquad \ldots(6.2)$$

where $G(t,s)$ and $r(s)$ are the Green's function and the residual respectively, while w denotes the number of subintervals.

Looking at $t \in (t_i, t_{i+1})$, $1 \le i \le w$, and corresponding local term of (6.2)

$$e_i(t) = \int_{t_i}^{t_{i+1}} G(t,s)\, r(s)\, ds, \qquad \ldots(6.3)$$

By assuming that the Green's function is constant in rectangle $t \in (t_i, t_{i+1})$, $s \in (t_i, t_{i+1})$, and taking the norm of the above expression we then have

$$\|e\|_i = C \|r(s)\|_i h_i, \qquad 1 \le i \le w \qquad \ldots(6.4)$$

The equation (6.4) is the starting point for the work in developing our proposed mesh selection algorithms.

It is convenient here to introduce some notations and basic definitions used in this chapter. Since in our algorithms we found not only the estimate for number of subintervals needed in the next iteration but also the estimate for number of subintervals needed in each subinterval, let us denote such estimates as w_i^*.

The maximum and average of subintervals needed are respectively denoted and calculated by

$$w_{max}^* = \max_i \{ w_i^* \}$$

and

$$\overline{w}^* = (\sum_{i=1}^{w} w_i^*) \; / \; w$$

The standard deviation *STD* is defined as

$$STD = \sqrt{\sum_{i=1}^{w} \frac{(w_i^* - \overline{w}^*)^2}{(w-1)}}$$

6.2 Mesh Placement Algorithms

As described in the previous chapter the residual norm in each subinterval can be written in the form $\|r(t)\|_i = k_i h_i^s$, $1 \le i \le w$, for some constants k_i, s > 0, hence equation (6.4) may be written as

$$\|e\|_i = C \, \|r\|_i \, h_i = C \, k_i h_i^{s+1}, \qquad \ldots(6.5)$$

for some constant C.

By assuming that $e_i(t)$ is large in i^{th} subinterval but small elsewhere, taking the maximum of these norms, give us

$$\max_i \|e\|_i = \max_i C \, \|r\|_i \, h_i = \max_i C \, k_i h_i^{s+1} \qquad \ldots(6.6)$$

and therefore suggests that the break points $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \}$ in new partition π^* be placed so as to minimise

$$\max_i \{ C\, k_i (h_i^*)^{s+1} \}$$

by requiring

$$C\, k_i (h_i^*)^{s+1} = E \approx TOL \qquad \ldots(6.7)$$

for some constant E, where TOL is a desired tolerance and $1 \le i \le w^*$.

The exact determination of such points $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \}$ from equation (6.7) is very difficult. But, this task is equivalent to determining $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \}$ so that

$$(C\, k_i)^{1/(s+1)}\, h_i^* = (E)^{1/(s+1)} \approx (TOL)^{1/(s+1)} \qquad \ldots(6.8)$$

and produces therefore asymptotically the same distribution of t_i^*'s as the problem of determining $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \}$ so that

$$(C\, k_i)^{1/(s+1)}\, h_i^* = \frac{1}{w^*} \sum_{i=1}^{w} (C\, k_i)^{1/(s+1)} h_i^*$$

since k_i are constant values, the above can be written as

$$\int_{t_i}^{t_{i+1}} (Ck_i)^{1/(s+1)}\, dt = \frac{1}{w^*} C^{1/(s+1)} \int_a^b (k(t))^{1/(s+1)}\, dt \;, \quad 1 \le i \le w^* \qquad \ldots(6.9)$$

where $k(t)$ is a piecewise constant function, i.e. $k(t) = k_i$, $t \in (t_i, t_{i+1})$

Using (6.8) and (6.9) we have

$$\frac{1}{w^*} C^{1/(s+1)} \int_a^b k^{1/(s+1)}\, dt = (TOL)^{1/(s+1)}$$

or

$$w^* = \frac{\int_a^b k^{1/(s+1)}\, dt}{(\frac{TOL}{C})^{1/(s+1)}} \qquad \ldots(6.10)$$

Our developed numerical scheme to determine the break points $\{ t_2^*, t_3^*, \ldots, t_{w^*}^* \}$ using criterion function $r_i h_i$ does not have the form $\int_a^b k^{1/(s+1)} dt$, hence we slightly modify the equation (6.9) by multiplying both sides with factor $\frac{1}{(k_1)^{1/(s+1)}}$ to give

$$\frac{1}{(k_1)^{1/(s+1)}} \frac{1}{w^*} C^{1/(s+1)} \int_a^b k^{1/(s+1)} dt = \frac{1}{(k_1)^{1/(s+1)}} (TOL)^{\frac{1}{s+1}}$$

$$\frac{1}{w^*} C^{1/(s+1)} \int_a^b (k/k_1)^{1/(s+1)} dt = (\frac{TOL}{(k_1)})^{1/(s+1)}$$

or

$$w^* = \frac{\int_a^b (k/k_1)^{1/(s+1)} dt}{(TOL/Ck_1)^{1/(s+1)}}$$

The estimate w^*, therefore, can be written

$$w^* = \left\lceil \frac{\theta_b}{(TOL/Ck_1)^{1/(s+1)}} \right\rceil \qquad \ldots(6.11)$$

here the notation $\lceil \ldots \rceil$ indicates the smallest integer greater than expression.

With a given TOL, w^* can be calculated easily since θ_b is the value of the piecewise constant function at the right boundary described in §5.4, $k_i = \|r\|_i / h_i^s$, while C is a constant estimated using the algorithm on the following section.

Let w_i^* denotes the estimate of number of subintervals needed in i^{th}-subinterval for the next iteration. This estimate will be proportional to $\int_{t_i}^{t_{i+1}} (k/k_1)^{1/(s+1)} dt$, and can be determined

$$w_i^* = \frac{C^{1/(s+1)} \int_{t_i}^{t_{i+1}} (k/k_1)^{1/(s+1)} dt}{C^{1/(s+1)} \int_a^b (k/k_1)^{1/(s+1)} dt} w^*$$

to give

$$w_i^* = \left\lceil (\frac{C}{TOL})^{1/(s+1)} \int_{t_i}^{t_{i+1}} (k)^{1/(s+1)} dt \right\rceil, \quad 1 \le i \le w \qquad \ldots(6.12)$$

6.3 Estimating the Constant C

Let the error estimate for the current iteration E^* be found using the formulae described in chapter 4. It has been shown that for problems having sufficiently smooth solution and the matrix coefficient is not very large the error estimate E^* is a very reliable estimate even if the approximate solution is very poor. The basic idea here is to employ the estimate E^* in predicting the constant C in equation (6.7).

Since the residual $r(s)$ is zero at the collocation points, it can be written in form

$$r(s) = \frac{r^{(q)}(s^*)}{q!}\prod_{j=1}^{q}(s-\xi_{ij}), \quad \text{for some } s^* \in [t_i, t_{i+1}],\ s \in [t_i, t_{i+1}] \qquad \ldots(6.13)$$

By considering the local terms e_i, $1 \le i \le w$, as in equation (6.3) and then using (6.13) and the properties of the Green's function, it has been shown in [23] that the global error $e(t) = x(t) - x_{wq}(t)$ satisfies the local terms :

$$\|e\|_i = C\,\|x^{(q+1)}(t)\|_i h_i^{q+1} + O(h^{q+2}) \qquad \ldots(6.14)$$

where $x(t)$ and $x_{wq}(t)$ are the exact and the approximate solution respectively, and the constant C shown in the appendix part of [45] is dependent only on the number of collocation points q, and satisfies

$$C = \frac{1}{2^{q+1}q!}\max_{-1\le t\le 1}\{\int_{-1}^{t}\prod_{j=1}^{q}(s-\xi_j)ds\},\ \xi_j \text{ the collocation points in [-1,1]}$$

Furthermore, Russell and Christiansen [45] have also shown that the residual $r(t)$ and exact solution $x(t)$ are related by

$$r^{(q)}(t) = x^{(q+1)}(t) + O(h)$$

Using equations (6.7) and (6.14), the *RH* algorithm may be regarded as an adaptive mesh algorithm trying to equidistribute the local terms in equation (6.5).
Hence, using equation (6.6) we then have

$$\max_i\{C\,\|r\|_i h_i\} = \max_i\{C\,k_i h_i^{s+1}\} = \max_i\|e\|_i \approx E^*, \quad 1 \le i \le w$$

this immediately gives an estimation for C

$$C^* = E^* / (\max_i \{ \|r\|_i h_i \}) \qquad \ldots(6.15)$$

It is also worthwhile to note here that for some boundary value problems having severe layers the estimate E^* is unsatisfactory if the approximate solution is very poor, in particular it is larger than the actual error. As the result, the estimate C^* might be very large. This clearly indicates that the estimate w^* should be used carefully, especially in the initial stages of the collocation process, where the approximation may be very poor.

6.4 Practical Implementation

As mentioned in the previous section the approximation w_i^* might be very poor in the first iterations of the collocation process, especially when the initial mesh points are very crude and the problems have severe layers. From this point of view, it is unreasonable to apply the estimate w^* into algorithm without any additional restrictions in the first few iterations. Hence, to implement a practical mesh selection strategy, additional modifications are needed to ensure that the strategy does not go awry.

Firstly we note that it seems sensible to restrict the size of w^* particularly in the first few iterations. Some simple techniques widely used in mesh adaptive algorithms, for example doubling the number of subintervals, immediately give us an obvious choice of such restriction, i.e. w^* should be taken no more than doubling the current number of subintervals w. Furthermore, we may utilise some values arising in the computation process, for example the average and the maximum of w_i^*. Involving the average and standard deviation might be relevant since these two quantities are widely used as a simple tool to examine the distribution of data, hence in our case they would be useful to predicting the homogeneity of w_i^*.

Basically the outline of proposed strategies which could be applied before directly attempting to use w^* is as follows

A. If the estimate w^* is greater than $2w$, there are two obvious choices
   1. The number of subintervals in new mesh π^* is set by doubling the current one, i.e. $w^* = 2w$.

2. Increasing the number of subintervals by one, i.e. $w^* = (w+1)$.

B. Before applying the first restriction mentioned in A, one may include some additional test in the algorithm by searching for the maximum of $\{ w_i^* \}$ where w_i^*'s are the estimations of number of intervals needed for each subintervals , $1 \leq i \leq w$, and then calculate the average $\overline{w}^*$ and, if necessary, the standard deviation *STD*.

The strategies then are

1. If $w_{\max}^*$ is less than $2\overline{w}^*$ we may expect that the local terms are sufficiently equidistributed. Hence, we keep using the current mesh points and halve all subintervals where the estimate w_i^* is greater than $\overline{w}^*$, i.e. we use subdivision rather than mesh placement.

2. Like the above step $B1$, however here, we restrict subdivision all subintervals having $w_i^* > (\overline{w}^* + STD)$, and keeping the rest.

C. This is a variant of algorithm B, where firstly we check some constraints mentioned in B, if they are not satisfied, then we use the second strategy of A, i.e. increase w by one.

We summarise the above by writing them in C^{++} like pseudo code form for each strategy as follows

Algorithm A1:

```
if (w* > 2w)    w* = 2w;
else w* is used in the next iteration;
```

Algorithm A2:

```
if (w* > 2w)    w* = (w+1);
else w* is used in the next iteration;
```

Algorithm B1:

```
if ( w*max < 2 w̄* )      // the mesh sufficiently equidist
{
   int  j=0;
   for(i=1; i≤w; i++)
   {
      if( w*i > w̄* )     // halving this subinterval
      {
         halve i^th subinterval;
         j++;
      }
   }
   w* = w*+j;
}
else if (w* > 2w)    w* = 2w;
else  w* is used  in the next iteration;
```

Algorithm B2:

```
if ( w*max < 2 w̄* )      // the mesh sufficiently equidist
{
   int  j=0;
   for(i=1; i≤w; i++)
   {
      if( w*i >( w̄* +STD))     // halving this subinterval
      {
         halve i^th subinterval;
         j++;
      }
   }
   w* = w*+j;
}
else if (w* > 2w)    w* = 2w;
else  w* is used  in the next iteration;
```

Algorithm C1:

```
if ( w*max < 2 w̄* )      // the mesh sufficiently equidist
{
   int  j=0;
   for(i=1; i≤w; i++)     // halving this subinterval
   {
      if( w*i > w̄* )
      {
         halve i^th subinterval;
         j++;
      }
   }
   w* = w*+j;
}
else if (w* > 2w)    w* = (w+1);
else  w* is used  in the next iteration;
```

Algorithm C2:

```
if ( w*_max < 2 w̄* )     // the mesh sufficiently equidist
{
   int  j=0;
   for(i=1; i≤w; i++)     // halving this subinterval
   {
      if( w*_i > ( w̄* +STD) )
      {
         halve i^th subinterval;
         j++;
      }
   }
   w* = w*+j;
}
else if (w* > 2w)   w* = (w+1);
else  w* is used in the next iteration;
```

To illustrate how the above strategies work in practice and to show how important the estimate w^* can be in improving efficiency of the collocation algorithms, we consider two problems having severe boundary layers.

Problem 1 :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \mu\cos^2(\pi t) + 2\pi^2\cos(2\pi t) \end{pmatrix}, \qquad 0 < t < 1$$

subject to boundary conditions :

$x_1(0) = x_1(1) = 0$

μ is the problem parameter, $|\mu| >> 1$

Taking the problem parameter $\mu = 10^8$, the problem has severe boundary layers with thickness 10^{-4} near both ends. Tables *6.1A - 6.1B - 6.2* and *6.3* display selected results of numerical experiments using different strategies. In the tables, E and E^* indicate the actual error and its estimate, while the sequence of w value displays the number of subintervals for each iteration. The rows indicated by *time* show the total time needed to satisfy a required accuracy. The error estimate E^* is used to terminate the computation process.

The following *Table 6.1A* is intended to show what may happen if the estimate w^* is directly taken as the number of subintervals in next stage of the collocation process without any additional restriction.

Table 6.1A

(w^* used without any additional restriction)

TOL -->	1e-01	1e-02		1e-03	
q = 3					
E*	7.400e-02	E*	6.620e-03	E*	4.153e-04
E	3.988e-02	E	2.699e-03	E	3.084e-04
time	0:6:10	time	0:6:36	time	0:11:1
w	161	w	255	w	404
	454		417		407
	49		80		133
	50		85		155
	129		163		217
	100		126		187
	61		104		165
	67		130		152
	76				160
	93				220
	105				
	100				
q = 4					
E*	8.666e-02	E*	4.147e-03	E*	9.301e-04
E	9.145e-03	E	2.623e-03	E	1.182e-05
time	0:5:5	time	0:4:40	time	0:7:46
w	83	w	123	w	180
	384		375		355
	36		45		72
	33		48		75
	46		89		108
	47				93
	53				112
	52				87
	55				109
	57				109
	57				121
	59				113
	89				186
q = 5					
E*	1.619e-02	E*	3.719e-03	E*	5.970e-04
E	1.202e-02	E	1.711e-03	E	3.627e-04
time	0:2:27	time	0:3:36	time	0:4:7
w	52	w	73	w	102
	269		317		320
	38		37		40
	19		27		71
	36		50		45
					44
					50

As we can see in *Table 6.1A*, the estimate w^* is quite large in particular for the first two iterations so that most computation time is spent at these stages. We suspect these poor estimates are caused by poor estimates at early stages of computation process which results in a poor estimate of C^*. It is clear that if, by some means of numerical scheme, we are able to avoid using a poor estimate at early iterations, a considerable improvement may be obtained.

Using single subinterval increment, *Table 6.1B* presents results using just three collocation points in each subinterval. Comparing *Table 6.1B* with the first part of *Table 6.1A*, the results given in *Table 6.1A* indicate that firstly the number of subintervals in the final iteration is reasonably close to the corresponding one in *Table 6.1B*, secondly the estimates E^* is quite satisfactory later in the process. These results are telling us that the estimate w^* works satisfactorily and the error estimate E^* performs well as the criterion to terminate the process.

Table 6.1B
(single interval increment)

TOL -->	1e-01	1e-02		1e-03	
q = 3		q = 3		q = 3	
E*	4.240e-02	E*	7.240e-03	E*	9.533e-04
E	2.792e-02	E	5.825e-03	E	5.657e-04
time	0:5:48	time	0:10:33	time	0:43:53
w	90	w	112	w	186

In *Table 6.2* on the following page the notation ⋮ indicates that the values increase by one. From this table the results show that both *algorithm A1* and *algorithm A2* perform satisfactorily, even though at this point it is not clear which algorithm is more reliable. However, we can say that for this problem the *algorithm A1* and *algorithm A2* are competitive to each other. Comparing the results displayed in *Table 6.1A* and the results given in *Table 6.2*, they illustrate quite clearly that using *algorithm A1* and *algorithm A2* significantly reduce the computation time.

Table 6.2

		A1			A2		
q	TOL -->	1e-01	1e-02	1e-03	1e-01	1e-02	1e-03
3	time	0:3:52	0:3:11	0:8:28	0:1:21	0:2:12	0:5:28
	w	8	8	8	5	5	5
		16	16	16	:	:	:
		32	32	32	14	14	14
		64	64	64	18	18	18
		128	128	128	24	24	24
		256	256	256	40	40	40
		47	74	118	64	64	64
		94	148	216	65	65	65
		125	136	188	66	66	66
		85	94	158	67	67	67
		78	144	152	70	112	68
		58		290	106	84	106
		62		352	156	94	140
		100				132	212
		131				200	148
		66					238
		94					320
		132					
4	time	0:1:35	0:2:6	0:5:46	0:2:59	0:3:22	0:3:48
	w	8	8	8	5	5	5
		16	16	16	:	:	:
		32	32	32	14	14	14
		64	64	64	20	20	20
		128	128	128	28	28	28
		117	173	254	40	40	40
		36	51	71	41	41	41
		37	58	89	42	42	42
		74	89	113	43	43	43
		58	59	114	57	57	57
		47	75	109	73	73	73
		50	73	118	90	90	90
		84		119	130	130	130
				115	180	180	180
				120	29	42	62
				114	52	71	92
				154	53	75	138
					52	112	194
					56	164	
					55		
					76		
5	time	0:1:17	0:0:58	0:1:33	0:1:20	0:1:42	0:1:45
	w	8	8	8	5	5	5
		16	16	16	:	:	:
		32	32	32	14	14	14
		64	64	64	18	18	18
		128	128	128	24	24	24
		47	66	92	32	32	32
		21	27	38	42	42	42
		35	48	36	43	43	43
		28		48	44	44	44
		28		56	56	56	56
		24		44	70	70	70
		32		37	92	92	92
		34		62	116	116	116
		33				24	34
		39				33	58
		65				50	80
						74	

Detailed inspection of the sequence w displayed in *Table 6.2* reveals some interesting observations. For columns under heading *A1*, we can see that w is doubled until a certain number and then it starts to reduce (or increase) precisely at the same stage for all tolerances; for instance look at the number of points $q = 3$ and tolerance $TOL = 10^{-1}, 10^{-2}, 10^{-3}$, the number of subinterval w is doubled at the first 7 iterations and then it reduces to 47, 74 and 118 for each desired accuracy. For the *algorithm A2* we observe that w increases by the same value (not always one) and then start to differ until the desired accuracy is satisfied. These results confirm that the accuracy in estimating w^* may be very poor at the early iterations and improves after several iterations.

Still with *problem 1* we observe now in more detail the performance of each algorithm by examining their computation time as displayed in *Table 6.3* on the following page.

For more straightforward comparison, for all algorithms we tabulate selected results in *Table 6.3*. We realise that it is not easy to make a comprehensive comparison using a limited numerical results. Perhaps one of the best ways to assess the performance of each strategy is to observe which algorithms perform badly in the sense their computation time is considerably larger than the others.

From *Table 6.3*, it is observed that in all cases the algorithms *A1*, *A2*, *B1* give somewhat worse results, whilst algorithms *B2*, *C1* and *C2* perform better and indicate roughly even performance. As we can see the *algorithm B1* works very well for $q = 4$ and $TOL = 10^{-1}$, but for $q = 3$ and $TOL = 10^{-3}$ its computation time is the worst. Similar observations can be found in columns under heading *A1* and *A2*. In this table it will be noticed that the *algorithm C2* consistently works well and is never the worst in term of computation time.

The estimate E^* is again reasonably close to the actual error indicating that it is a reliable error estimate and suitable as a criterion for terminating iteration.

Table 6.3
(problem 1)

	A1	A2	B1	B2	C1	C2
q = 3						
TOL=1e-1						
E* -->	3.501e-02	3.412e-02	2.501e-02	8.743e-02	4.771e-02	2.602e-02
E -->	8.206e-03	3.414e-02	6.679e-03	1.729e-02	2.660e-02	1.838e-02
time -->	0:3:52	0:1:21	0:4:4	0:1:33	0:1:12	0:1:2
q = 3						
TOL=1e-2						
E* -->	6.757e-03	5.226e-03	6.798e-03	8.472e-03	6.472e-03	9.287e-03
E -->	3.389e-03	2.376e-03	6.804e-03	3.598e-03	1.891e-03	1.241e-03
time -->	0:3:11	0:2:12	0:3:4	0:2:9	0:2:11	0:2:29
q = 3						
TOL=1e-3						
E* -->	1.181e-04	1.237e-04	2.426e-04	7.670e-04	6.137e-04	6.606e-04
E -->	6.179e-05	1.036e-04	5.338e-05	1.072e-04	4.059e-04	5.883e-04
time -->	0:8:28	0:5:28	0:9:2	0:7:2	0:3:13	0:2:56
q = 4						
TOL=1e-1						
E* -->	6.967e-03	7.902e-02	7.815e-02	3.547e-02	9.720e-02	2.285e-02
E -->	4.727e-03	1.585e-02	5.573e-02	1.972e-02	1.842e-02	7.390e-03
time -->	0:1:35	0:2:59	0:1:0	0:1:2	0:1:0	0:1:7
q = 4						
TOL=1e-2						
E* -->	9.809e-03	2.009e-04	2.377e-03	1.050e-03	4.242e-03	1.748e-03
E -->	2.144e-03	1.224e-04	1.619e-03	1.035e-03	5.401e-04	2.536e-04
time -->	0:2:6	0:3:22	0:1:23	0:1:44	0:1:12	0:1:54
q = 4						
TOL=1e-3						
E* -->	9.342e-04	6.844e-05	3.381e-04	4.083e-04	3.264e-04	1.308e-04
E -->	6.968e-05	2.084e-05	7.763e-05	3.995e-05	8.704e-05	8.115e-05
time -->	0:5:46	0:3:48	0:2:15	0:2:46	0:1:54	0:3:11
q = 5						
TOL=1e-1						
E* -->	1.245e-02	6.674e-02	1.179e-02	7.901e-02	3.392e-03	7.901e-02
E -->	3.131e-03	6.519e-02	1.175e-02	4.388e-02	1.563e-03	4.388e-02
time -->	0:1:17	0:1:20	0:0:44	0:0:58	0:0:50	0:0:59
q = 5						
TOL=1e-2						
E* -->	6.889e-03	2.802e-03	3.363e-04	7.738e-03	6.428e-03	4.256e-04
E -->	3.515e-03	9.538e-04	3.309e-04	1.873e-03	2.255e-03	4.227e-04
time -->	0:0:58	0:1:42	0:1:6	0:1:17	0:0:50	0:0:49
q = 5						
TOL=1e-3						
E* -->	1.322e-04	1.785e-04	2.247e-04	8.534e-04	1.227e-04	9.885e-05
E -->	1.024e-04	1.606e-04	8.341e-05	7.637e-04	3.412e-05	9.568e-05
time -->	0:1:33	0:1:45	0:1:3	0:1:5	0:1:3	0:1:4

We now consider as the second illustrative example a problem having a layer near the left boundary as follows

Problem 2 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\mu \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu \end{pmatrix}$$

BCs : $x_1(0) = x_1(2) = 0.$

μ is the problem parameter, $|\mu| >> 1$

Though a variety of values of μ have been used in numerical experiments, they indicate similar results so *Table 6.4* only presents the results with problem parameter $\mu = 10^4$.

From *Table 6.4* we observe that algorithm *A1* does not perform very satisfactorily while *A2* and *B1* give roughly similar more satisfactory results, however they occasionally produce the longest computation time. Though *C1* produces reasonable computation time in *Table 6.3*, here it performs very badly in some cases, for example for $q = 4$ and $TOL = 10^{-5}$. A final note made here is that the algorithms *B2* and *C2* give comparable results, and overall *C2* consistently performs well.

Looking at the estimate E^* in most cases it is very satisfactory, though occasionally it overestimates the error slightly but it is still close to the error.

Table 6.4

(problem 2, problem parameter $\mu = 10^4$)

	A1	A2	B1	B2	C1	C2
q = 4						
TOL = 1e-03						
E* -->	6.502e-04	5.138e-04	8.954e-05	4.171e-05	7.665e-04	4.171e-05
E -->	4.435e-05	5.078e-04	2.576e-05	3.768e-05	3.092e-05	3.768e-05
time -->	0:3:26	0:1:48	0:1:57	0:1:54	0:2:58	0:1:54
q = 4						
TOL = 1e-04						
E* -->	3.156e-05	1.995e-05	1.242e-05	4.171e-05	2.549e-05	4.171e-05
E -->	2.632e-05	1.688e-05	8.914e-06	3.768e-05	1.803e-05	3.768e-05
time -->	0:2:58	0:1:58	0:2:41	0:1:54	0:3:29	0:1:54

... cont'd

...(cont'd) **Table 6.4**

	A1	A2	B1	B2	C1	C2
q = 4						
TOL = 1e-05						
E* -->	9.561e-06	3.045e-06	6.681e-06	3.850e-06	9.637e-06	3.850e-06
E -->	4.343e-06	5.442e-07	4.096e-07	1.221e-06	4.558e-06	1.221e-06
time -->	0:3:27	0:2:56	0:3:29	0:2:58	0:6:27	0:2:59
q = 5						
TOL = 1e-03						
E* -->	3.110e-04	3.673e-04	1.676e-05	7.900e-04	4.611e-05	7.900e-04
E -->	1.219e-04	3.660e-04	1.657e-05	7.895e-04	4.294e-05	7.895e-04
time -->	0:1:34	0:0:51	0:1:12	0:0:46	0:1:24	0:0:46
q = 5						
TOL = 1e-04						
E* -->	3.135e-05	6.052e-05	4.832e-06	1.601e-05	6.343e-06	1.601e-05
E -->	3.589e-06	5.935e-05	4.156e-06	1.434e-05	2.862e-06	1.434e-05
time -->	0:2:13	0:0:58	0:2:21	0:1:4	0:1:29	0:1:4
q = 5						
TOL = 1e-05						
E* -->	3.923e-06	3.332e-06	2.066e-06	3.581e-06	4.576e-06	3.581e-06
E -->	2.497e-06	1.362e-07	8.696e-07	2.417e-07	3.644e-06	2.417e-07
time -->	0:2:26	0:1:29	0:4:58	0:1:36	0:2:2	0:1:36
q = 8						
TOL = 1e-03						
E* -->	9.546e-04	3.460e-04	9.311e-05	7.388e-04	7.715e-04	7.388e-04
E -->	2.341e-04	3.361e-04	5.167e-05	6.628e-04	4.396e-04	6.628e-04
time -->	0:1:12	0:1:34	0:1:1	0:0:21	0:0:51	0:0:21
q = 8						
TOL = 1e-04						
E* -->	3.236e-05	4.602e-06	7.082e-06	7.044e-05	7.476e-05	3.695e-05
E -->	2.824e-05	1.682e-06	6.531e-06	1.950e-06	1.892e-06	3.452e-05
time -->	0:1:36	0:0:49	0:1:0	0:0:29	0:0:41	0:0:48
q = 8						
TOL = 1e-05						
E* -->	2.461e-06	3.257e-06	2.878e-06	2.924e-06	4.381e-06	3.442e-06
E -->	1.809e-06	9.585e-07	1.574e-07	1.262e-07	4.252e-06	6.464e-07
time -->	0:1:36	0:0:52	0:1:12	0:0:33	0:0:23	0:0:46

6.5 Mesh Subdivision Algorithms

Apart from replacing the integral process used in section 6.3 with appropriate summation notation, in principle there is no difference in obtaining the estimate w^* for mesh subdivision algorithms. Nevertheless, for completeness, we briefly derive the estimate w^* and then take a look another simple way to obtain it.

We have to solve the problem of determining $\{t_2, t_3, \ldots, t_{w*}\} \subset (a,b)$ so as to minimise the problem :

$$\max_i \{ C k_i (h_i^*)^{s+1} \} = \max_i \{ \|e\|_i \}, \quad 1 \le i \le w*$$

which is equivalent to determining $\{t_2^*, t_3^*, \ldots, t_{w*}^*\} \subset (a,b)$ so that

$$(Ck_i)^{1/(s+1)} h_i^* = (E)^{1/(s+1)} \approx (TOL)^{1/(s+1)} \qquad \ldots(6.16)$$

$$(Ck_i)^{1/(s+1)} h_i^* = \frac{1}{w^*}((Ck_1)^{1/(s+1)} h_1^* + (Ck_1)^{1/(s+1)} h_2^* + \ldots + (Ck_1)^{1/(s+1)} h_{W*}^*$$

$$= \frac{1}{w^*} \sum_{i=1}^{w*} (Ck_i)^{1/(s+1)} h_I^* = (TOL)^{1/(s+1)}$$

$$w^* = \left\lceil \frac{\sum_{i=1}^{w} k_i^{1/(s+1)} h_i}{(TOL/C)^{1/(s+1)}} \right\rceil \qquad \ldots(6.17)$$

The number of subintervals in i^{th}-subintervals w_i^* needed for next iteration, will be proportional to $(Ck_i)^{1/(s+1)} h_i$, to give

$$w_i^* = \frac{(Ck_i)^{1/(s+1)} h_i}{C^{1/(s+1)} \sum_{i=1}^{w} k_i^{1/(s+1)} h_i} w^*$$

or

$$w_i^* = \left\lceil \left(\frac{Ck_i}{TOL}\right)^{1/(s+1)} h_i \right\rceil \qquad 1 \le i \le w \qquad \ldots(6.18)$$

For completeness, on the following page we will describe how to establish the equation (6.18) using a different way.

Consider h_i^*, the width of i^{th}-subinterval for next iterations, since we attempt to equidistribute the terms $k_i\ h_i^{s+1}$, this requires

$$\max_i \{ Ck_i h_i^{*s+1} \} = E \approx TOL$$

or

$$h_i^{*s+1} = \frac{TOL}{Ck_i}$$

$$h_i^* = (\frac{TOL}{Ck_i})^{1/(s+1)}, \quad 1 \le i \le w$$

The estimate of number of subintervals for i^{th}-subintervals in the next iteration, then can be found using

$$w_i^* = \frac{h_i}{h_i^*} = \frac{h_i}{(\frac{TOL}{Ck_i})^{1/(s+1)}},$$

or, in simpler form

$$w_i^* = \lceil (\frac{Ck_i}{TOL})^{1/(s+1)}\ h_i \rceil, \quad 1 \le i \le w.$$

6.6 Numerical Illustrations

For mesh subdivision algorithms, unlike mesh placement algorithms, the number of subintervals will always increase for each stage of collocation process. Strictly speaking we can not reduce the number of subintervals to be smaller than the current one, though the optimal w^* might be smaller than the current w. Consequently, for this type of strategy we have to be more careful using the estimation of w. As for the mesh placement algorithms, the approximation w_i^* might be very poor in the first iterations of the collocation process, especially if the initial mesh points are very crude.

This poor estimate w_i^* may be caused by $O(h^{q+2})$ term in (6.14), where it dominates the local terms $Cr_i h_i$ and it can not be ignored. Another source of trouble is the fact that

the error estimate E^* might be very unsatisfactory at the first stages of the collocation process, as a result the estimate C^* might be very poor, more precisely the value of C^* may be larger than it should be.

Basically, the strategies which will be applied here are the same as described in the previous section. By carrying out some numerical experiments we shall observe which strategies are more suitable for mesh subdivision algorithms.

For convenience and to make comparison more clear we consider the problems examined in section 6.4.

The following tables *6.5* and *6.6* display some results for problem 1 with problem parameter $\mu = 10^4$.

As can be seen in *Table 6.5*, after the initial stages in which the mesh is doubled the figures in the table clearly show that the *algorithm A1* place too many break points in the mesh. Unlike in mesh placement algorithms in which *A1* and *A2* is comparable, here it is quite clear that *A2* performs much better than *A1*.

Examining performance of the estimate E^* as the criterion function for terminating the computation process in *Table 6.5*, we observe its reliability indicated by the closeness of E^* and the actual error E.

To make comparison more straightforward, in *Table 6.6* we display some results using all algorithms. From this table we may conclude that *A1* has to be discarded when using mesh subdivision, despite the fact that the estimate E^* performs quite satisfactorily.

Looking more deeply at the column under heading *A2*, though in most cases algorithm *A2* works very well, we observe that it occasionally performs unsatisfactorily as we can see for $q = 3$ and $TOL = 10^{-2}, 10^{-3}$. However it is still much better than *A1*.

For algorithms *B1* and *B2*, the numerical results show that these algorithms in some cases take unreasonably long computation time, even though again E^* approximate E closely.

Another notable observation from these tables is the superiority of algorithms *C1* and *C2*, in particular *C2* where we observe its performance is very impressive.

Table 6.5
(problem 1, mesh subdivision algorithm)

		A1			A2		
q	TOL :	1e-01	1e-02	1e-03	1e-01	1e-02	1e-03
3	E*	6.466e-02	9.286e-03	9.707e-04	9.030e-02	5.147e-03	4.207e-04
	E	6.471e-02	9.292e-03	9.693e-04	7.751e-02	1.421e-03	4.192e-04
	time	0:25:1 9	0:38:12	1:21:32	0:1:57	0:10:1	0:16:1
	w	8	8	8	5	5	5
		16	16	16	:	:	:
		32	32	32	44	89	116
		64	64	64	45	90	117
		128	128	128	46	91	118
		256	256	256	47	92	119
		458	512	512	48	93	120
		470	648	734	49	94	121
		478	668	768	98	95	122
		479	684	798	110	184	228
		480		799	112	192	252
				800		201	
						211	
4	E*	5.698e-02	8.867e-03	9.513e-04	6.056e-02	5.893e-03	7.273e-04
	E	5.195e-02	8.509e-03	9.244e-04	5.626e-02	7.640e-04	2.650e-05
	time	0:11:22	0:14:31	0:34:54	0:1:23	0:1:49	0:6:52
	w	8	8	8	5	5	5
		16	16	16	:	:	:
		32	32	32	36	38	53
		64	64	64	37	39	54
		128	128	128	38	40	55
		256	256	256	39	41	56
		334	378	450	40	74	57
		342	390	466	76	80	58
		346	396	478	80	95	116
				479	81	110	122
				480			126
							134
							142
							148
							156
							206
5	E*	8.835e-02	5.600e-03	5.555e-04	6.431e-02	5.552e-03	8.442e-04
	E	7.907e-02	5.130e-03	5.532e-04	5.894e-02	5.541e-03	8.429e-04
	time	0:5:29	0:14:6	0:19:8	0:0:49	0:0:52	0:1:16
	w	8	8	8	5	6	5
		16	16	16	:	:	:
		32	32	32	29	:	:
		64	64	64	30	30	:
		128	128	128	31	31	32
		220	256	256	32	32	33
		228	316	342	33	33	34
		230	322	354	34	34	35
			326	358	42	46	68
					44	50	73
							74

Table 6.6

(problem 1, mesh subdivision algorithm)

	A1	A2	B1	B2	C1	C2
q = **3**						
TOL=1e-1						
E^* -->	6.466e-02	9.030e-02	8.037e-02	8.526e-02	2.514e-02	2.514e-02
E -->	6.471e-02	7.751e-02	8.050e-02	8.537e-02	2.518e-02	2.518e-02
time -->	0:25:19	0:1:57	0:8:18	0:2:27	0:2:59	0:2:8
q = **3**						
TOL=1e-2						
E^* -->	9.286e-03	5.147e-03	9.286e-03	8.315e-03	4.466e-04	5.141e-03
E -->	9.292e-03	1.421e-03	9.292e-03	8.318e-03	3.910e-04	1.256e-03
time -->	0:38:12	0:10:1	0:7:2	0:6:31	0:6:59	0:3:47
q = **3**						
TOL=1e-3						
E^* -->	9.707e-04	4.207e-04	9.175e-04	6.378e-04	1.897e-04	5.706e-04
E -->	9.693e-04	4.192e-04	9.162e-04	2.580e-04	9.403e-05	5.687e-04
time -->	1:21:32	0:16:1	0:35:29	0:7:30	0:9:59	0:6:37
q = **4**						
TOL=1e-1						
E^* -->	5.698e-02	6.056e-02	4.807e-02	3.767e-02	5.698e-02	3.767e-02
E -->	5.195e-02	5.626e-02	4.328e-02	1.707e-02	5.195e-02	1.707e-02
time -->	0:11:22	0:1:23	0:3:33	0:1:14	0:2:6	0:1:14
q = **4**						
TOL=1e-2						
E^* -->	8.867e-03	5.893e-03	8.867e-03	8.866e-03	1.833e-03	5.112e-03
E -->	8.509e-03	7.640e-04	8.509e-03	8.508e-03	1.783e-03	8.877e-04
time -->	0:14:31	0:1:49	0:6:9	0:2:27	0:2:56	0:2:8
q = **4**						
TOL=1e-3						
E^* -->	9.513e-04	7.273e-04	9.130e-04	7.281e-04	5.281e-05	1.347e-04
E -->	9.244e-04	2.650e-05	8.877e-04	2.926e-05	3.635e-05	2.154e-05
time -->	0:34:54	0:6:52	0:12:20	0:11:9	0:3:20	0:5:26
q = **5**						
TOL=1e-1						
E^* -->	8.835e-02	6.431e-02	9.691e-02	6.431e-02	3.330e-02	6.431e-02
E -->	7.907e-02	5.894e-02	6.644e-02	5.894e-02	3.158e-02	5.894e-02
time -->	0:5:29	0:0:49	0:2:47	0:0:52	0:1:35	0:0:35
q = **5**						
TOL=1e-2						
E^* -->	5.600e-03	5.552e-03	5.600e-03	5.551e-03	9.910e-03	9.910e-03
E -->	5.130e-03	5.541e-03	5.130e-03	5.541e-03	9.892e-03	9.892e-03
time -->	0:14:6	0:0:52	0:3:33	0:0:59	0:1:24	0:0:45
q = **5**						
TOL=1e-3						
E^* -->	5.555e-04	8.442e-04	5.555e-04	8.472e-04	8.472e-04	8.472e-04
E -->	5.532e-04	8.429e-04	5.532e-04	8.429e-04	5.635e-04	8.429e-04
time -->	0:19:8	0:1:16	0:5:58	0:1:5	0:2:33	0:0:44

The following two tables present numerical results for *problem 2* with problem parameter $\mu = 10^4$.

In *Table 6.7*, *A0* indicates that the estimate w^* is used for next iteration without any of the modifications we have discussed before. As indicated by the numerical results, here we emphasise that it is not sensible at all using w^* without any additional constraints for mesh subdivision, even though it might be all right for mesh placement algorithms. Looking at the other algorithms, the best performance is shown by algorithms *A2* and *C2* while algorithm *C1* is also reasonable though it puts too many break points when w increases from 72 to 112. On the other hand algorithm *B1* gives somewhat poorer results in terms of both time and final w. Again we observe that the algorithms *A1* lead to completely unsatisfactory results, even though the order of accuracy obtained is the same with other algorithms.

Table 6.7

	A0	A1	A2	B1	B2	C1	C2
q = 4							
TOL=1e-3							
E*	9.418e-04	9.552e-04	9.050e-04	9.552e-04	8.668e-04	9.284e-04	8.365e-04
E	9.246e-04	9.377e-04	6.090e-04	9.377e-04	8.458e-04	9.003e-04	8.262e-04
time	1:40:4	0:48:16	0:1:11	0:9:7	0:1:49	0:1:44	0:0:47
w	806	8	5	6	5	6	5
	867	16	6	8	6	8	6
	877	32	7	12	8	12	8
	882	64	:	16	10	16	10
		128	:	24	11	24	11
		256	38	37	14	37	14
		512	39	51	16	51	16
		620	40	102	19	52	19
		631	41	204	23	70	23
		636	42	312	26	71	26
			43	323	30	72	30
			54	328	36	112	36
			58		41	119	41
					82	121	42
					127		43
					137		44
					141		59
							68
							69

As the last illustration, the following table presents results when the proposed algorithms are used to solve *problem 2*.

The most notable observation about these results is that the algorithm *B1* performs very badly where in most cases its computation time is the worst. As indicated in *Table 6.6*, here again we found the fact that though in most cases algorithm *B2* performs in reasonable computation time, in some cases it needs longer computation time than the others. Of other algorithms *A2*, *C1* and *C2* give reasonable results, with algorithms *A1* and *C1* producing marginally worse results than those using algorithm *C2*.

Looking at the estimate E^* and the actual error E, it is clear that this error estimate is quite reliable as a criterion to terminate the computation process. Moreover in all cases it is very close to the actual error.

Before we end this chapter it might be useful to make some final concluding remarks regarding our proposed algorithm in determining the number of subinterval needed in the next stage of collocation process.

Since there are many minor variants in the algorithms it is not easy in making assessment, however at least we observe some interesting and useful facts. Firstly using the estimate w^* without any restriction may lead to completely unsatisfactory results, in particular for mesh subdivision strategies. Secondly, if the estimate w^* is greater then $2w$ then the simple techniques to determine the number of subinterval in next iteration such as doubling the current number of subintervals w or setting $w^* = (w+1)$ can be utilised further to obtain more reliable algorithms.

The results of numerical experiments indicate that the algorithm *A1* may perform comparably with the others in mesh placement algorithms but not for mesh subdivision algorithms. Using the algorithm *B1* the results indicate similar observation. Conversely algorithms *A2* may be efficient in mesh subdivision strategy, but it can be very inefficient for mesh placement algorithms. For the algorithm *B2* though it usually gives satisfactory results, unfortunately in all tables we found that it occasionally produces a poorer result.

The most notable result is that the algorithm *C2* performs efficiently in both strategies placement and subdivision while algorithm *C1* is also comparable.

Table 6.8

	A2	B1	B2	C1	C2
q = 4					
TOL = 1e-03					
E* -->	9.050e-04	9.552e-04	8.668e-04	9.284e-04	8.365e-04
E -->	6.090e-04	9.377e-04	8.458e-04	9.003e-04	8.262e-04
time -->	0:1:11	0:9:7	0:1:49	0:1:44	0:0:47
q = 4					
TOL = 1e-04					
E* -->	9.140e-06	9.735e-05	9.176e-05	9.590e-05	8.540e-05
E -->	8.979e-06	9.254e-05	9.048e-05	9.504e-05	8.417e-05
time -->	0:1:30	0:13:16	0:2:28	0:2:16	0:1:9
q = 4					
TOL = 1e-05					
E* -->	2.721e-06	9.126e-06	9.121e-06	9.789e-06	8.650e-06
E -->	5.173e-07	8.979e-06	8.237e-06	8.858e-06	8.192e-06
time -->	0:3:38	0:31:59	0:6:15	0:2:11	0:2:8
q = 5					
TOL = 1e-03					
E* -->	5.943e-04	7.898e-04	7.898e-04	8.399e-04	8.399e-04
E -->	5.920e-04	7.889e-04	7.889e-04	8.392e-04	8.392e-04
time -->	0:0:38	0:6:53	0:1:43	0:1:4	0:0:39
q = 5					
TOL = 1e-04					
E* -->	8.412e-05	8.481e-05	9.423e-05	8.406e-05	8.406e-05
E -->	8.396e-05	8.396e-05	9.339e-05	8.396e-05	8.396e-05
time -->	0:0:40	0:8:37	0:3:38	0:1:15	0:0:46
q = 5					
TOL = 1e-05					
E* -->	4.504e-06	8.434e-06	9.193e-06	7.539e-06	7.540e-06
E -->	4.440e-06	8.147e-06	9.097e-06	7.361e-06	7.361e-06
time -->	0:1:2	0:14:13	0:1:52	0:1:38	0:0:52
q = 8					
TOL = 1e-03					
E* -->	6.057e-04	2.183e-04	6.057e-04	7.443e-05	6.057e-04
E -->	5.868e-04	2.070e-04	5.868e-04	7.193e-05	5.868e-04
time -->	0:0:18	0:1:32	0:0:32	0:0:50	0:0:19
q = 8					
TOL = 1e-04					
E* -->	5.270e-05	5.269e-05	1.446e-05	1.446e-05	1.446e-05
E -->	5.107e-05	5.107e-05	1.405e-05	1.405e-05	1.405e-05
time -->	0:0:20	0:2:7	0:0:20	0:0:52	0:0:20
q = 8					
TOL = 1e-05					
E* -->	6.458e-06	8.220e-06	6.458e-06	4.624e-06	4.624e-06
E -->	5.724e-06	7.454e-06	5.724e-06	4.402e-06	4.402e-06
time -->	0:0:20	0:2:4	0:0:59	0:0:47	0:0:30

chapter 7

Locating the Layer Regions and Estimating Their Initial Mesh Points

7.1 Nature of Stiffness

Throughout this chapter we are concerned with the standard first order linear system ODE of the form

$$x'(t) = A(t)\,x(t) + y(t), \qquad a < t < b \qquad \ldots(7.1)$$

with associated boundary conditions at the end of range $[a,b]$.

Here, we assumed that there exists one small parameter ε occurring in the boundary value problem.

In studying IVPs, the problems whose solutions exhibit both quickly and slowly change modes in such a manner that a numerical computation process for its solution must be stable for all step sizes to facilitate efficiency in the computation process, are referred to as stiff differential equations. They are important in numerical analysis since they frequently arise in practical problems and they are difficult to solve by some numerical methods, even though the methods perform quite well in solving non-stiff problems.

Stiffness arises if there is a conflict between stability and accuracy requirements that appear in certain problems. Focusing on initial value problems, Lambert [36] discuss in some detail various aspects of the phenomenon of stiffness, and propose to use word 'phenomenon' instead of 'property', since the latter rather implies that stiffness can be defined in precise mathematical terms. In [36] it is also illustrated that two different systems of ordinary differential equations with the same conditions could have identically the same exact solution, but they behave very differently when tackled numerically. This implies that the phenomenon cannot be a function of the

particular solution, it must be a property of the differential equation itself. In turn this suggests that we consider, not the particular solution of problem satisfying the given boundary condition but the general solution of the systems, which in turn requires us to look at the eigenvalues of the coefficient matrix of the systems. In general, if the eigenvalues of matrix A vary over several orders of magnitude, there are difficulties.

There are essentially two features of stiff boundary value problems that make their solution by numerical methods difficult. One is that the matrix A has large eigenvalues. The second is that there may be turning points in the problem. The concept of a turning points is not particularly well defined in the literatures [34], however, for our purposes we take it to mean a subinterval of $[a,b]$ to which an eigenvalue of A changes its order of magnitude and its sign from positive to negative.

When studying singular perturbation theory, it can be found that there seem some connection between stiffness and that phenomenon; As described in [36] systems exhibiting singular perturbation can be seen as a sub-class of stiff system. Though we are not going to pursue this connection further, let us quote a simple example as follows

Let A be a constant (2x2) matrix and $y(t) = 0$ and consider a system in form

$$\begin{pmatrix} x_1'(t) \\ x_2'(t) \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

Suppose that matrix A has real eigenvalues λ_1 , λ_2 such that $\lambda_1 << \lambda_2 < 0$. By eliminating x_1 and x_2 we obtain the equivalent second order scalar equation

$$x_1'' - (a_{11} + a_{22})x_1' + (a_{11}a_{22} - a_{12}a_{21})x_1 = 0$$

Since λ_1 , λ_2 are the zeros of the quadratic

$$\lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21})$$

which can be written as

$$(1/\lambda_1)x_1'' - (1 + \lambda_2/\lambda_1)x_1' + \lambda_2 x_1 = 0$$

As $\lambda_1 \rightarrow -\infty$ we have the singular perturbation situation.

7.2 Eigenvalues for Predicting the Layer Location

In solving some simple boundary value problems a crude equal-spaced initial mesh might suffice to obtain the required approximate solution successfully in terms of effort and time consumed. However for boundary value problem having severe layers, it might be not sensible to expect the collocation iteration schemes to perform efficiently when one does not have some reasonable estimate of the location of layer regions.

In this section, our aim is to involve computing eigenvalues of matrix $A(t)$ within interval $[a,b]$ and using their magnitude and rate of change to predict possible transition regions.

Firstly let us consider the linear constant coefficient system

$$x'(t) = Ax(t) + y(t), \quad a < t < b \qquad \ldots (7.2)$$

where A is a constant (nxn) matrix with eigenvalues $\lambda_i \in C$, and corresponding eigenvectors v_i, $i = 1, 2, \ldots, n$.

The general solution of (7.2) takes the form

$$x(t) = \sum_{i=1}^{m} c_i \exp(\lambda_i t)\, v_i + \gamma(t) \qquad \ldots (7.3)$$

where c_i are arbitrary constants and $\gamma(t)$ is a particular integral.

In general we will deal with situation where λ_i is very large in magnitude for some $i = 1, 2, 3, \ldots, n$. If the imaginary part of the eigenvalues λ_i , $i = 1, 2, \ldots, n$ of matrix A is dominant then we would expect that the solution of the system will have oscillatory behaviour.

Let $Re(|\lambda_p|)$ be the maximum of $Re(|\lambda_i|)$, $i = 1, 2, \ldots, n$ and let us consider two cases for the matrix A in equation (7.2)

1. Suppose that $Re(\lambda_p) < 0$. If λ_p is very large in magnitude, the term $\exp(Re(\lambda_p)t)$ will decay rapidly if t is close to the left boundary a and it will be slower when t is away from a. We could then expect that there is a boundary layer around a.

2. Suppose that $Re(\lambda_p) > 0$. The term $exp(Re(\lambda_p)t)$ will increase rapidly if t is closer to the right boundary b. Again if λ_p is very large in magnitude, we then could expect that there is a boundary layer around b.

Having considered a linear system having a constant coefficient matrix, we now examine a first order system in the form of equation (7.1). Let the coefficient matrix $A(t)$ assumed to be constant (or 'frozen') in a subregion $[t^*-\delta, t^*+\delta] \subset [a, b]$ for some $t^* \in [a, b]$ and $\delta > 0$. By taking $A_c = A(t^*)$, the differential equation in this small region may be written as $x'(t) = A_c x(t) + y(t)$. With this constant matrix A_c, behaviour of the exact solution $x(t)$ will not be correctly represented . However, if the matrix A is taken to be piecewise frozen, then we could expect that the solution of this frozen system to behave like the exact solution. Moreover, in cases of boundary layers, since the layers are located in small regions around end points, it is reasonable to expect the solution corresponding to the frozen matrix A_c to behave like corresponding to those of $A(t)$ in these small regions. In cases of transition layers, the similar situation is expected to occur around small region in which the eigenvalue changes sign from positive to negative, since it is well known that if an eigenvalue changes sign from positive to negative around t_0 there might be a transition layer around t_0 [4,29]. For this case we should examine behaviour of the solution in two small subregions, i.e. by taking $t^* \in [t_0-\delta, t_0]$ and then $t^* \in [t_0, t_0+\delta]$.

By assuming that the matrix $A(t)$ can be locally frozen, then the equation (7.1) takes the form $x' = Ax + y(t)$. By ignoring $y(t)$ in our analysis we can conclude that the behaviour of the solution $x' = A_c x$ where A_c is a piecewise frozen of the matrix A, in some way locally represents the behaviour of the solution of (7.1), thus justifying the use of the linear test equation $x' = A_c x$ in predicting the location of layer region.

Let t^* be some fixed value of t; then the piecewise frozen of A would assert that in some neighbourhood of t^*, the solution of (7.1) behaves like those of $x' = A(t^*)x$. Since $A(t^*)$ is constant matrix the general solution of $x' = A(t^*)x$ has the form of (7.3) and we can carry out similar analysis.

Let $\lambda_i(t)$, $i = 1, 2, \ldots, n$ be eigenvalues of matrix $A(t)$ and suppose that $Re(|\lambda_p(t)|)$ is the maximum of $Re(|\lambda_i(t)|)$, for some $p \in \{1, 2, \ldots, n\}$ and $t \in [a, b]$. We now observe a number of possibilities as follows

1. Suppose that for some $t_L \in [a, b]$, and $t^* \in [a, t_L]$ the term $|exp(Re(\lambda_p(t^*)\, t_1)) - exp(Re(\lambda_p((t^*))\, t_2))|$ is much larger than $|t_2 - t_1|$ for all $t_1, t_2 \in [a, t_L]$, then for $t \in [a, t_L]$ the term $exp((\lambda_p((t^*))t)$ will vary rapidly if t is close to the left boundary a and it will be slower when t is away from a, we then could expect that there is a layer around left boundary a.
2. Suppose that for some $t_R \in [a, b]$, and $t^* \in [t_R, b]$ the term $|exp(Re(\lambda_p(t^*)\, t_1)) - exp(Re(\lambda_p(t^*)\, t_2))| >> |t_2 - t_1|$ for all $t_1, t_2 \in [b - t_R, b]$, then for $t \in [b - t_R, b]$ the term $exp((\lambda_p(t^*)t)$ will vary rapidly if t is close to b and will be slower when t is away from b, we then could expect that there is a boundary layer around b.
3. From 1 and 2 it is possible to have boundary layers at both sides
4. Suppose that the eigenvalue $\lambda_p(t)$ changes sign from positive to negative at t_0 for some t_0 within small subinterval $(t_{T1}, t_{T2}) \subset (a, b)$. Let $t^* \in (t_{T1}, t_0)$ and suppose that the term $|exp(Re(\lambda_p(t^*)\, t_1)) - exp(Re(\lambda_p(t^*)\, t_2))|$ is much larger than $|t_2 - t_1|$ for all $t_1, t_2 \in (t_{T1}, t_0)$, then for $t \in (t_{T1}, t_0)$ the term $exp((\lambda_p(t^*)t)$ will grow rapidly. It is also assumed that that the term $exp(Re(\lambda_p(t^*)\, t)$ varies faster if t tend to t_0^-. Similarly, let $t^* \in (t_0, t_{T2})$ and suppose that the term $|exp(Re(\lambda_p(t^*)\, t_1)) - exp(Re(\lambda_p(t^*)\, t_2))|$ is much larger than $|t_2 - t_1|$ for all $t_1, t_2 \in (t_0, t_{T2})$, then for $t \in (t_0, t_{T2})$ the term $exp((\lambda_p(t^*)t)$ will decrease rapidly. We could expect there is a transition layer inside subinterval $[t_{T1}, t_{T2}]$.

It is worth noting that all observations also take into account the rate of change in the fundamental solution component $x(t) = exp(Re(\lambda_p)t)$ which is assumed to be large. In particular, for determining the location of transition layer, we need to check whether it varies over several order of magnitude as well as checking whether the eigenvalue (equivalently the derivative term) changes sign in such a neighbourhood.

To illustrate how we work out to locate possible layer regions, we consider a number of examples. Firstly consider the following problem

Example 1 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\mu \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu \end{pmatrix}$$

μ is the problem parameter, $|\mu| >> 1$

The boundary conditions are

$x_1(0) = 0$

$x_1(2) = 0.$

The eigenvalues are $\lambda_1 = 0$, $\lambda_2 = -\mu$; and $\lambda_p = -\mu$. The term $exp(\lambda_p t) = exp(-\mu t)$ will vary very rapidly within subinterval $[0; -1/\mu]$. It will vary more rapidly if t is closer to the left boundary and will be slower if t is away from the left boundary. We may expect that there will be a boundary layer around the left hand boundary.

Example 2 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & \mu t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The boundary conditions are

$x_1(-1) = 0$

$x_1(1) = 0.$

While μ is a parameter, $|\mu| >> 1.$

The eigenvalues are $\lambda_1 = 0$ and $\lambda_2(t) = \mu t$. λ_p is given by $\lambda_p(t) = \mu t$. By taking a fixed point t^* on small subinterval $(-1, -1+\delta)$ for some $\delta > 0$ sufficiently small, the term $exp(\lambda_p(t^*)t) = exp(\mu t^* t)$ will vary rapidly if t is close to the left boundary. Similarly, if we take $t^* \in (1-\delta, 1)$ for some $\delta > 0$ we also found that the term $exp(\mu t^* t)$ will vary rapidly if t is close to the right boundary. As we can see there is a change of sign of the eigenvalue but there is no interior layer as the change of sign is from negative to positive. In this case we could then expect that there are two boundary layers, i.e. one boundary layer for each side.

Example 3 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\mu t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \pi^2 \cos(\pi t) - \mu t \sin(\pi t) \end{pmatrix}$$

BCs : $x_1(-1) = x_1(1) = 0$ μ is a parameter , $|\mu| \gg 1$.

The eigenvalues are $\lambda_1 = 0$ and $\lambda_2(t) = -\mu t$. The second eigenvalue $\lambda_2(t)$ changes sign from positive to negative at point $t = 0$. Let t^* be a fix point within subinterval $(-\delta, 0)$ for some $\delta > 0$ sufficiently small, then the term $exp((-\mu t^*)t)$ varies rapidly if $t \to 0^-$. Similar observation can be obtained if t^* is taken from $(0, \delta)$ and $t \to 0^+$. Looking at the following graphs where $\mu = 10^2$ we can see that the term $exp((-\mu t)t)$ will vary rapidly if t is around the origin points $t = 0$. Moreover the derivative term $(-2\mu t)exp((-\mu t)t)$ changes sign and it varies rapidly around $t = 0$. We then could expect that there is a transition layer at the middle of interval.

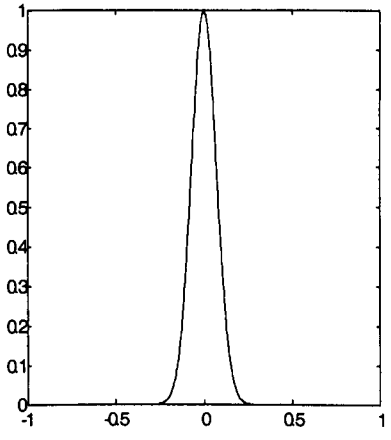


Fig.7.1 Graphs $exp(-\mu t^2)$

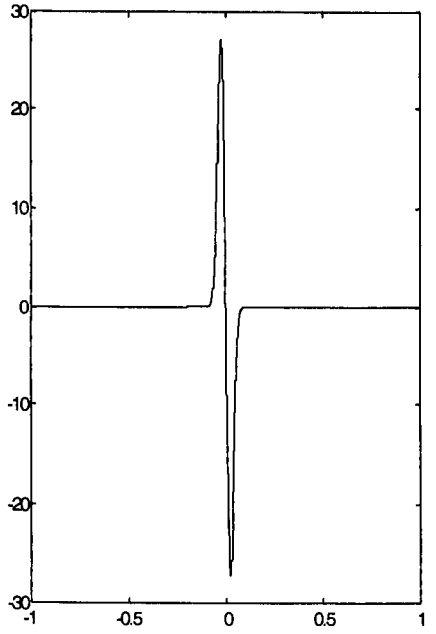


Fig.7.2 Graphs $(-2\mu t)exp(-\mu t^2)$

The next example illustrates situation where the interior layer is not at the origin.

Example 4 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\mu(t-0.5) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

BCs : $x_1(0) = x_1(1) = 0$, μ is a parameter , $|\mu| \gg 1$.

The eigenvalues are $\lambda_1 = 0$ and $\lambda_2(t) = -\mu(t\text{-}0.5)$. The second eigenvalues $\lambda_2(t)$ changes value from positive to negative at $t = 0.5$. Taking a fix point $t^* \in (0.5\text{-}\delta, 0.5)$ for some $\delta > 0$ sufficiently small and then checking the rate of change of the term $exp(-\mu(t^*\text{-}0.5)t)$ for $t \in (0.5\text{-}\delta, 0.5)$, we can see that the term varies rapidly and it will be faster if $t \to 0.5^-$. Similar observation found if t^* is taken from $(0.5, 0.5+\delta)$ and t tend to 0.5 from right side. For illustration, the graph of the functions $exp(-\mu(t\text{-}0.5)t)$ and $(-\mu(2t\text{-}0.5))exp(-\mu(t\text{-}0.5)t)$ around the middle of interval are depicted below. As shown in the graphs the term $exp(-\mu(t\text{-}0.5)t)$ will vary very fast if t is around the point $t = 0.5$. Moreover the derivative term $(-\mu\ (2t\text{-}0.5))exp(-\mu(t\text{-}0.5)t)$ changes sign and varies rapidly in small region around $t = 0.5$. We then could expect that there is a transition layer.

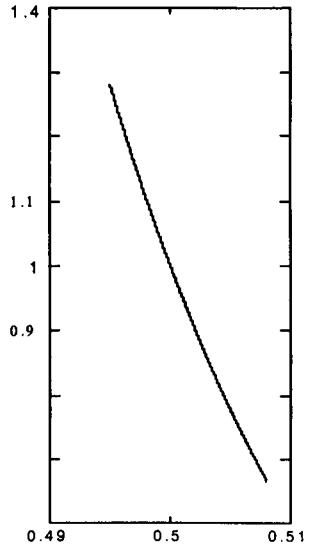


Fig.7.3 Graphs $exp(-\mu(t\text{-}0.5)t)$

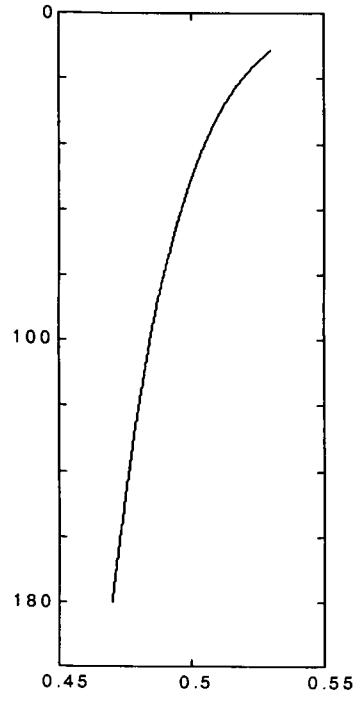


Fig.7.4 Graphs $(-\mu(2t\text{-}0.5))exp(-\mu(t\text{-}0.5)t)$

The following example shows the case where the eigenvalue changes sign but there is no a transition layer.

Example 5 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & \mu(t-0.5) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The boundary conditions are

$$x_1(0) = 0$$

and

$$x_1(1) = 0.$$

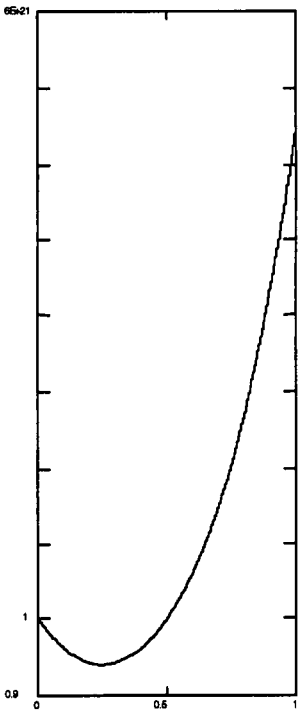
While μ denotes the problem parameter, $|\mu| \gg 1$.

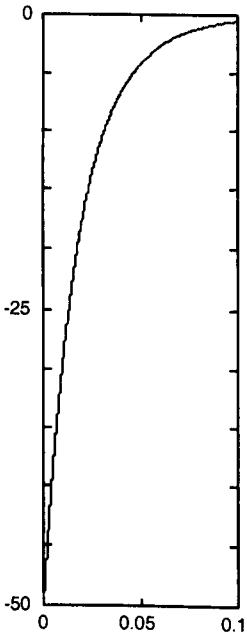
Two eigenvalues are $\lambda_1 = 0$ and $\lambda_2(t) = \mu(t\text{-}0.5)$. Here the second eigenvalue $\lambda_2(t)$ changes value from negative to positive at $t = 0.5$.

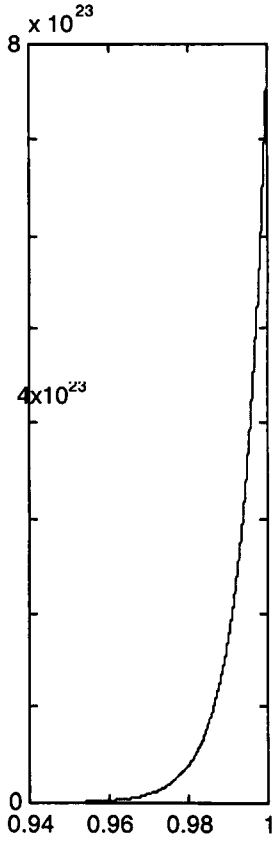
Even though the second eigenvalue changes sign at $t = 0.5$, we observe that for a fixed $t^* \in (0.5\text{-}\delta, 0.5)$ the term $exp(\mu(t^*\text{-}0.5)t)$ will vary slower if $t \to 0.5^-$. Similar case if t^* is taken within subinterval $(0.5;0.5+\delta)$, the term will vary slower if $t \to 0.5^+$. The following graphical illustrations on the following page, *figure 7.5* and *figure 7.6,* show that the term $exp(\mu\,(t\text{-}0.5)t)$ does not vary rapidly if t is around the point $t = 0.5$. Comparing with its behaviour around the left and right boundaries, this function is a smooth in subinterval $(½-\delta, ½+\delta)$, for some $\delta > 0$. Similar behaviour is shown by the term $(\mu(2t\text{-}0.5))exp(\mu(t\text{-}0.5)t)$, even though it changes sign at $t = 0.5$.

In this example, the term $exp(\mu(t\text{-}05)t)$ will vary rapidly if t tends to the end points, i.e. it varies rapidly in a small region around boundary points.

For this problem we may expect that there are boundary layers at end points. This example confirms the well known fact that a boundary value problem having changing eigenvalue at a point is not necessarily to have an interior boundary layer at that point.







left boundary right boundary

Fig.7.5 Graphs $exp(\mu(t-0.5)t)$ Fig.7.6 Graphs $(\mu(2t-0.5))exp(\mu(t-0.5)t)$

Example 6 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu(2-t^2) & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \mu \end{pmatrix}$$

BCs : $x_1(-1) = x_1(1) = 0$. μ is the problem parameter , $|\mu| \gg 1$.

The eigenvalues are determined by functions

$$\lambda_1(t) = -\sqrt{\mu(2-t^2)}$$

$$\lambda_2(t) = \sqrt{\mu(2-t^2)}$$

There is no changing values in both eigenvalues. The first eigenvalue $\lambda_1(t)$ has negative values in the whole interval [-1,1], while the second one has positive values.

Taking a fixed point t^* close to the left boundary, it is clear that the solution component set up by this fix point varies rapidly around the left boundary. For t^* close to the right boundary, the similar situation is observed. In this problem, we may expect there is a boundary layer at each end point.

7.3 Determining the Width of Layers

Having predicted possible layer regions, we shall now attempt to determine the width of the layer region and initial mesh points in such region. Here, the term width of layer is taken to mean a suitable width of layer for collocation process such that using it in the collocation algorithms should improve the performance of the algorithms.

Recall equation (7.1), and let us assume, for simplicity, that there are n_- rapidly decreasing modes and n_+ rapidly increasing modes throughout interval $[a, b]$. In case of separated boundary conditions, then the initial n_- boundary conditions control the decreasing modes and the n_+ terminal boundary conditions control the increasing modes.

Let us examine a simple test problem

$$x'(t) = \lambda x(t), \quad 0 \le t \le 1 \qquad \ldots(7.4)$$
$$x(0) = 1$$

The coefficient λ represents an eigenvalue, so it is in general complex.

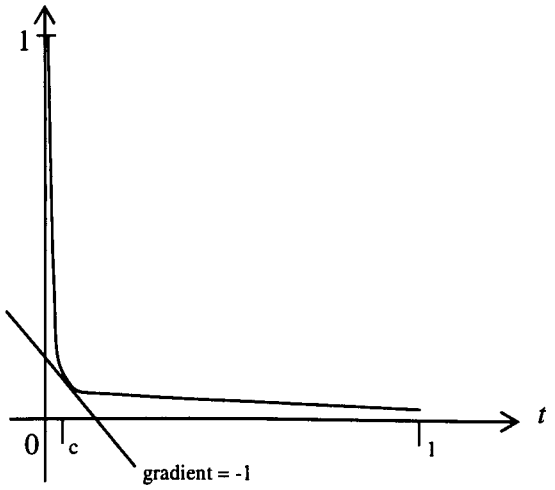
Let λ_R denotes the real part of the eigenvalue λ

$$\lambda_R = Re(\lambda).$$

Assuming that $\lambda_R < 0$ and its magnitude is very large, i.e. $|\lambda_R| >> 1$. It is also assumed that the highly oscillatory case is excluded by assuming that

$$\rho|\lambda_R| \ge |Im(\lambda)|, \quad \text{for some constant } \rho \text{ of moderate size.}$$

Let $x(t)$ be the solution of (7.4). Since $|\lambda_R| >> 1$, $x(t)$ is a fast decreasing mode in small subinterval $[0,c] \subset [0,1]$ for some $c > 0$ and is sufficiently smooth in subinterval $[c,1]$. This situation is illustrated graphically as follows



As t moves away from the left boundary the magnitude of gradient of the solution $x(t) = exp(\lambda_R t)$ will decrease. When the magnitude of gradient is equal to one the solution $x(t)$ could be said to be no longer in fast decreasing mode. This enable us to determine c, a suitable subdivision point, in a simple way by requiring

$$x'(t) = \lambda_R e^{\lambda_R c} = -1$$

or

$$e^{\lambda_R c} = -1/\lambda_R$$

We then take the natural logarithm of both sides

$$\lambda_R c = ln\,(-1/\lambda_R)$$

to give

$$h_{max1} = c = \frac{-ln(-\lambda_R)}{\lambda_R} \qquad \ldots(7.5)$$

We may extend the idea to more general problem of form (7.1). Let λ_R^i, $i = 1, 2, \ldots, n$ denote the real part of the negative eigenvalues of matrix A. Obtaining expression like (7.5) for each negative eigenvalue, we have a set of points $\{c_i\}$, $i = 1, 2, \ldots, n$ and the width of the left layer region h_{max1} may be taken as their minimum.

$$h_{max1} \approx \min\,\{-ln(\lambda_R^i) \;/\; \lambda_R^i\}, \quad i = 1, 2, \ldots, n.$$

Similarly, the point d close to the right end point b can be determined by considering the positive eigenvalues.

Note that the points c found using the equation (7.5) depend solely on the parameter λ_R, more precisely there is no direct relationship with the desired tolerance *TOL*. The following alternative estimation of the layer region attempts to relate not only the parameter λ_R^i but also the required tolerance *TOL*.

Since the fundamental solution component $x(t) = exp(\lambda_R t)$ decays rapidly in a small region $[0,c]$ close to the left boundary, and outside this region it is approximately zero. This gives us another simple way to determine such point c, by requiring

$$e^{\lambda_R c} < TOL$$

we then take the natural logarithm of both side to give

$$\lambda_R c < ln\,(TOL)$$

which finally gives us the desired point

$$h_{max2} = c = \frac{ln(TOL)}{\lambda_R} \qquad \ldots(7.6)$$

Note that h_{max2} will increase if the desired tolerance *TOL* decreases, in other words a more accurate desired approximate solution will produce a larger layer region.

7.4 Initial Mesh Points in the Layer Regions

Padé approximations provide both the optimal rational approximations and the error in the approximation for exponential function $exp(t)$. Here we shall relate these optimal approximations to the fundamental solution component $x(t) = exp(\lambda_R t)$ and use them in estimating the number of initial mesh points in the layer regions.

The (k,j) Padé approximation is given by the rational function :

$$R_{kj}(t) = P_{kj}(t) \,/\, Q_{kj}(t)$$

where $P_{kj}(t)$ and $Q_{kj}(t)$ are

$$P_{kj}(t) = 1+\frac{k}{j+k}t+\frac{k(k-1)}{(j+k)(j+k-1)}\frac{t^2}{2!}+\ldots+\frac{k(k-1)\ldots 1}{(j+k)\ldots(j+1)}\frac{t^k}{k!}$$

$$Q_{kj}(t) = 1-\frac{j}{k+j}t+\frac{j(j-1)}{(k+j)(k+j-1)}\frac{t^2}{2!}-\ldots+(-1)^j\frac{j(j-1)\ldots 1}{(k+j)\ldots(k+1)}\frac{t^j}{j!}$$

The error for Padé approximation is given by

$$e_p = (-1)^j\frac{j!k!}{(j+k)!(j+k+1)!}t^{j+k+1}+O(t^{j+k+2}) \qquad \ldots(7.7)$$

It is the unique rational approximation to $exp(t)$ of order $(j+k)$ such that the degree of numerator and denominator are k and j respectively. The diagonal Padé approximations are those with $k=j$.

As discussed in chapter 2 the solution of an m^{th}-order ordinary non-linear differential equation can be approximated to within order $O(h^q)$ by collocation when using spline function of order $(m+q)$ and the solution is in $C^{(m+q)}$. Here, h is the maximum mesh length of the partition π. Furthermore the results of investigation of the direct effect of using certain collocation points in improving the accuracy of the collocation solutions indicate that the $(m+q)^{th}$ order of approximation can be achieved by collocating the Gauss points, provided the solution is in $C^{(2q+m)}$ and the differential operator is sufficiently smooth. Moreover, at the ends of each subinterval, the approximation is $O(h^{2q})$ accurate.

In point of view of the stability concept, a paper of Wright [55] studied the stability function of collocation methods. In the paper it is shown that the stability function of the collocation points based on the points $\xi_1, \xi_2, \ldots, \xi_q$ is given by the rational function

$$R(t) = \frac{M^{(q)}(1)+M^{(q-1)}(1)t+\ldots+M(1)t^q}{M^{(q)}(0)+M^{(q-1)}(0)t+\ldots+M(0)t^q} = \frac{P(t)}{Q(t)}$$

where

$$M(t) = \frac{1}{t!}\sum_{i=1}^{q}(t-\xi_i)$$

It is also shown that for any polynomial $M(t)$ of exact degree q, $R(t)$ is an approximation to $exp(t)$ of order greater or equal to q and its error is given by

$$e^t - R(t) = (t^{q+1} \int_0^t e^{t(1-\xi)} M(\xi) d\xi) \; / \; Q(t)$$

Moreover, Hairer and Wanner [31] show that the stability function of some numerical schemes for solving test function $x' = \lambda x$, $x(0) = 1$, $Re(\lambda) < 0$ satisfy diagonal Padé approximation.

Let us now consider the fundamental solution component $v(t) = exp(\lambda_R t)$ which is assumed to dominate the behaviour of the solution in the layer region. We can relate this solution component with the Padé approximation by firstly noting that for a given degree of the numerator and the denominator the Padé approximation is a rational function having highest order of approximation. Secondly, from the above results the collocation solution using q Gauss collocation points has an approximation of order $(2q)$ at the end of subinterval, and thirdly the diagonal Padé approximation should be used since it is associated with the stability function.

We are now attempting to relate the error of the Padé approximation determined in (7.7) with the required tolerance TOL by choosing h appropriately. By taking $k = j$ and the fact that the Padé approximation is of order $O(j+k)$ which have to be equal to the order of q stage collocation solution, these give

$$k + j = 2q, \qquad \ldots(7.8)$$

hence

$$j = k = q \qquad \ldots(7.9)$$

If h_{p^*} denotes the estimate of the first subinterval used in the collocation algorithm and TOL is a required tolerance, we then apply the equation (7.9) into equation (7.7) and obtain

$$TOL = \frac{q!\;\; q!}{(2q)!\;\; (2q+1)!} (\lambda_R h_{p^*})^{2q+1}$$

to give

$$h_{p^*} = \sqrt[2q+1]{\frac{(2q)!\,(2q+1)!}{q!\;\; q!} TOL} \;\; / \;\; |\lambda_R| \qquad \ldots(7.10)$$

The next step, using the results from previous section, there are now three computable values h_{p*}, h_{max1}, and h_{max2} which immediately give two estimates for the number of initial subintervals in the layer region, they are

$$w_1^o = \lceil h_{max1} / h_{p*} \rceil \qquad \ldots(7.11)$$

and

$$w_2^o = \lceil h_{max2} / h_{p*} \rceil \qquad \ldots(7.12)$$

Since h_{p*}, h_{max1}, and h_{max2} are real numbers, in implementation we have to take the integer part of the estimates w_1^o and w_2^o. Here notation $\lceil \ldots \rceil$ indicates the round up of a real number.

In preliminary numerical experiments using the estimates (7.11) and (7.12) it was observed that these estimates are not satisfactory when dealing with problems having severe layers. This can be explained since for problems having severe layers the additional accuracy at break points in the Gauss collocation scheme due to higher order $O(h^{2q})$ is lost, and the order of approximation is then $O(h^{q+1})$. Such reduction in superconvergence order has been pointed out by Ascher and Bader [7].

Now let h_p be the estimate of the first subinterval. Using similar steps to obtain equation (7.10), we then have

$$k + j = q + 1$$

or

$$j = k = (q+1) / 2 \qquad \ldots(7.13)$$

and substitute equation (7.13) to equation (7.7) giving us

$$h_p = \sqrt[q+2]{\frac{(q+1)!\,(q+2)!}{((q+1)/2)!\;\;((q+1)/2)!}\,TOL} \;\; / \;\; |\lambda_R| \qquad \ldots(7.14)$$

Using equation (7.14) we now have the estimates for the number of initial subintervals in the layer region

$$w_1^o = \lceil h_{max1} / h_p \rceil \qquad \ldots(7.15)$$

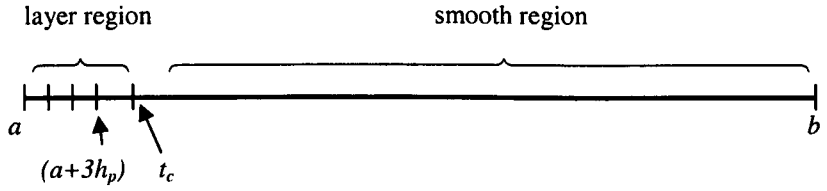
and

$$w_2^o = \lceil h_{max2} / h_p \rceil \qquad \ldots(7.16)$$

Using either w_1^o or w_2^o as the estimate of the number of initial mesh points in the layer region, and let w_L^o denotes the actual estimate, the initial mesh points will be

$$\pi_0 : \underbrace{a = t_1 < a+h_p < a+2h_p < \ldots < a+(w_L^o-1)\,h_p}_{\text{layer region}} < t_c < \underbrace{\ldots < \ldots < b = t_w}_{\text{smooth region}}$$

For illustration, let us take $w_L^o = 4$ to give a typical example as follows



7.5 Numerical Implementation

In this part, our aim is to observe how well the estimates w_1^o and w_2^o perform in the numerical computation. In practice it is reasonable that the estimate for layer region width h_{max} is taken from $\max\{h_{max1}, h_{max2}\}$, this gives the number of initial subintervals in the layer region to be $w^o = max\ \{ w_1^o, w_2^o \}$. In case both ends have boundary layers, w_L^o and w_R^o will denote the estimate for initial subintervals in the left and right boundary respectively. For the smooth region, at the first sight the simplest choice for initial number of subintervals is to take the maximum of w_L^o and w_R^o, even though as we will see later in the numerical experiments a small modification may be helpful.

In the numerical experiments, firstly we employ w^o the estimates of number of initial subintervals in the layer regions taken from $max\{ w_1^o, w_2^o \}$, in cases either $w_2^o > 2\,w_1^o$ or $w_1^o > 2\,w_2^o$ we make a slight modification. i.e. $w^o = 2\,w_1^o$ or $w^o = 2\,w_2^o$. In the smooth region we place (w^o-1) break points. Through experiments, we shall intensively compare the performance of the *RH* algorithms and de Boor algorithms using multiple and single point increment as well as the effect of using Gauss and Chebyshev collocation points. For predicted initial mesh we use algorithm *C2*

described in chapter 6. In addition, for comparison purposes we also implement the adaptive mesh algorithms using a crude initial mesh, here we use 4-equally spaced initial mesh.

7.5.1 Mesh Placement Algorithm

As the first illustration, we consider a boundary value problem having a severe layer at the left boundary, where the layer thickness depends upon the problem parameter μ.

The problem is

Problem 1 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\mu \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu \end{pmatrix}$$

BCs : $x_1(0) = x_1(2) = 0.$ μ is the problem parameter, $|\mu| \gg 1$

Taking $\mu = 10^2$, tables *7.1a, 7.1b* and *7.2* display the number of initial subintervals in the layer region obtained using h_p, h_{max1}, h_{max2}, q Gauss collocation points per subintervals, the desired tolerance *TOL*, and eigenvalue $|\lambda_R| = 10^2$. In these tables the single subinterval increment is used. The tables also contain values w and w^c which denote the total number of subintervals and the number of subintervals in the layer region used in the final stage of the collocation process. The heading h_{first} and h_{last} indicate the width of the first and the last subinterval in the final computation process respectively. The columns under heading i and T show the number of iterations and the time needed in the numerical computation. The results using 4-equal initial subintervals are indicated by B in the tables.

From tables *7.1a* and *7.1b* where the problem parameter $\mu = 10^2$, it clearly indicates that using predicted initial subintervals improve the performance of the collocation algorithm, especially in terms of decreasing number of iterations and time needed in computation process. From these tables it is also observed that the width of the first subinterval in the actual computation is reasonably close to h_p,

moreover the number of mesh points in the layer region is reasonable. It is notable that for $q = 2$ and $TOL = 10^{-4}$, A and B give almost identical results for w and w^c, however in term of number of iterations i (and computation time T), there is a significant improvement, i.e. $i = 11$ in A compared to $i = 80$ in B. In the meantime the estimate w^o is reasonable not only for lower accuracy but also for higher accuracy, though in some cases the final stage puts too many points in the layer region, for example from *Table 7.1a* for $q = 3$ and $TOL = 10^{-4}$ the actual number of subintervals in the layer region w^c is 40 subintervals (computation time $T = 6$ secs) which is larger than those needed in the computation B. Similar indication is also observed for number of collocation points $q = 5$ with tolerances 10^{-5}, 10^{-8}; and $q = 8$ with tolerances 10^{-8}, 10^{-10}, 10^{-12}.

In *Table 7.1b*, the numerical results using the de Boor criterion function are presented for comparison. For this criterion function, it can be seen that using the predicted initial subintervals the algorithms clearly perform better in terms of iterations and time needed compared to those using 4-equal initial subintervals.

Comparing *Table 7.1a* and *Table 7.1b*, the most notable result we have is that in all cases the *RH* criterion function gives much better results than those using de Boor criterion function. A dramatic improvement shown when $q = 2$ and the tolerance $TOL = 10^{-4}$ where de Boor algorithm needs 61 iterations (computation time 9 minutes and 21 seconds), in contrast the *RH* algorithm just needs 11 iterations (1 minutes and 2 seconds), further more it is also observed that the de Boor algorithm puts too many points in the layer region before reaching the desired tolerance, even when the width of the first subintervals in both algorithm is reasonable close.

As mentioned above, in some cases, the results in A are poorer than B. Since these appear in cases where w_2^o is greater than twice w_1^o, hence we might suspect that these might be caused by putting too many points in the layer and smooth region at the first stage of the computation process. This suggests making a slight modification to the predicted initial mesh points by taking

$$w^o = 2w_1^o \quad \text{if } w_2^o \text{ is greater than } 2w_1^o$$

or

$$w^o = 2w_2^o \quad \text{if } w_1^o \text{ is greater than } 2w_2^o.$$

Table 7.1a

(problem 1, RH Criterion, 1-interval increment, $\mu = 10^2$)

	q	TOL	h_p	h_{max1}	h_{max2}	w_1^o	w_2^o	w^o	h_{first}	h_{last}	w^c	w	i	T
A	2	1e-01	1.377e-02	4.605e-02	2.303e-02	3	1	3	3.676e-03	7.747e-01	7	13	8	0:0:1
	2	1e-02	7.746e-03	4.605e-02	4.605e-02	5	5	5	1.771e-03	6.366e-01	13	22	13	0:0:6
	2	1e-03	4.356e-03	4.605e-02	6.908e-02	10	15	15	8.332e-04	6.239e-01	32	41	12	0:0:18
	2	1e-04	2.449e-03	4.605e-02	9.210e-02	18	37	37	3.784e-04	6.367e-01	76	84	11	0:1:2
	3	1e-01	1.516e-02	4.605e-02	2.303e-02	3	1	3	8.234e-03	8.046e-01	3	10	5	0:0:1
	3	1e-02	9.564e-03	4.605e-02	4.605e-02	4	4	4	4.355e-03	1.593e+00	6	12	5	0:0:1
	3	1e-03	6.034e-03	4.605e-02	6.908e-02	7	11	11	1.894e-03	1.769e+00	18	24	3	0:0:2
	3	1e-04	3.807e-03	4.605e-02	9.210e-02	12	24	24	1.082e-03	1.607e+00	40	49	2	0:0:6
	4	1e-01	2.493e-02	4.605e-02	2.303e-02	1	0	1	1.382e-02	1.030e+00	2	8	7	0:0:1
	4	1e-02	1.698e-02	4.605e-02	4.605e-02	2	2	2	1.038e-02	1.004e+00	3	9	6	0:0:1
	4	1e-03	1.157e-02	4.605e-02	6.908e-02	3	5	5	5.329e-03	1.429e+00	7	13	4	0:0:1
	4	1e-04	7.883e-03	4.605e-02	9.210e-02	5	11	11	2.272e-03	1.793e+00	19	24	3	0:0:3
	5	1e-01	2.511e-02	4.605e-02	2.303e-02	1	0	1	3.000e-02	7.799e-01	1	6	5	0:0:0
	5	1e-02	1.807e-02	4.605e-02	4.605e-02	2	2	2	1.574e-02	1.062e+00	2	8	5	0:0:1
	5	1e-03	1.301e-02	4.605e-02	6.908e-02	3	5	5	6.438e-03	1.355e+00	6	12	3	0:0:1
	5	1e-04	9.361e-03	4.605e-02	9.210e-02	4	9	9	3.190e-03	1.799e+00	15	20	3	0:0:3
	5	1e-05	6.737e-03	4.605e-02	1.151e-01	6	17	17	2.171e-03	1.530e+00	27	35	2	0:0:5
	5	1e-08	2.511e-03	4.605e-02	1.842e-01	18	73	73	4.963e-04	2.513e-01	133	147	2	0:1:11
	8	1e-01	4.967e-02	4.605e-02	2.303e-02	0	0	1	8.920e-02	8.826e-01	0	4	3	0:0:0
	8	1e-02	3.946e-02	4.605e-02	4.605e-02	1	1	1	4.701e-02	1.007e+00	0	5	4	0:0:0
	8	1e-03	3.134e-02	4.605e-02	6.908e-02	1	2	2	2.365e-02	1.263e+00	2	7	4	0:0:1
	8	1e-04	2.490e-02	4.605e-02	9.210e-02	1	3	3	2.165e-02	8.944e-01	2	8	3	0:0:1
	8	1e-05	1.977e-02	4.605e-02	1.151e-01	2	5	5	1.492e-02	6.429e-01	4	11	2	0:0:1
	8	1e-08	9.911e-03	4.605e-02	1.842e-01	4	18	18	9.911e-03	1.009e-01	17	36	1	0:0:4
	8	1e-10	6.253e-03	4.605e-02	2.303e-01	7	36	36	6.253e-03	4.916e-02	35	72	1	0:0:13
	8	1e-12	3.946e-03	4.605e-02	2.763e-01	11	70	70	1.114e-03	4.358e-02	100	141	2	0:1:39
B	2	1e-01	-	-	-	-	-	-	3.980e-03	1.378e+00	7	12	9	0:0:1
	2	1e-02	-	-	-	-	-	-	1.758e-03	6.429e-01	13	22	19	0:0:7
	2	1e-03	-	-	-	-	-	-	8.321e-04	6.413e-01	32	41	38	0:0:31
	2	1e-04	-	-	-	-	-	-	3.841e-04	6.329e-01	74	83	80	0:3:12
	3	1e-01	-	-	-	-	-	-	6.389e-03	1.853e+00	5	10	7	0:0:1
	3	1e-02	-	-	-	-	-	-	4.537e-03	1.767e+00	6	11	8	0:0:1
	3	1e-03	-	-	-	-	-	-	2.930e-03	9.021e-01	11	17	14	0:0:4
	3	1e-04	-	-	-	-	-	-	1.575e-03	8.947e-01	23	29	26	0:0:16
	4	1e-01	-	-	-	-	-	-	9.676e-03	1.762e+00	4	9	6	0:0:1
	4	1e-02	-	-	-	-	-	-	9.676e-03	1.762e+00	4	9	6	0:0:1
	4	1e-03	-	-	-	-	-	-	6.562e-03	1.653e+00	6	10	7	0:0:1
	4	1e-04	-	-	-	-	-	-	4.390e-03	1.174e+00	10	14	11	0:0:3
	5	1e-01	-	-	-	-	-	-	1.463e-02	1.781e+00	3	8	5	0:0:1
	5	1e-02	-	-	-	-	-	-	1.463e-02	1.781e+00	3	8	5	0:0:1
	5	1e-03	-	-	-	-	-	-	1.463e-02	1.781e+00	4	8	5	0:0:1
	5	1e-04	-	-	-	-	-	-	8.812e-03	1.782e+00	6	9	6	0:0:1
	5	1e-05	-	-	-	-	-	-	5.877e-03	1.491e+00	9	12	9	0:0:2
	5	1e-08	-	-	-	-	-	-	1.796e-03	1.530e+00	32	35	32	0:0:39
	8	1e-01	-	-	-	-	-	-	2.738e-02	1.683e+00	1	7	4	0:0:1
	8	1e-02	-	-	-	-	-	-	2.738e-02	1.683e+00	1	7	4	0:0:1
	8	1e-03	-	-	-	-	-	-	2.738e-02	1.683e+00	2	7	4	0:0:1
	8	1e-04	-	-	-	-	-	-	2.738e-02	1.683e+00	3	7	4	0:0:1
	8	1e-05	-	-	-	-	-	-	1.460e-02	1.732e+00	5	8	5	0:0:1
	8	1e-08	-	-	-	-	-	-	1.011e-02	1.735e+00	8	10	7	0:0:2
	8	1e-010	-	-	-	-	-	-	6.219e-03	1.529e+00	13	16	13	0:0:8
	8	1e-012	-	-	-	-	-	-	3.452e-03	1.651e+00	25	27	24	0:0:30

A : *using predicted initial subinterval*
B : *using 4-equal initial subinterval*

Table 7.1b

(problem 1, de Boor Criterion, 1-interval increment, $\mu = 10^2$)

	q	TOL	h_p	h_{max1}	h_{max2}	w^o_{L1}	w^o_{L2}	w^o	h_{first}	h_{last}	w^c	w	i	T
A	2	1e-01	1.377e-02	4.605e-02	2.303e-02	3	1	3	2.548e-03	1.142e+00	8	14	9	0:0:2
	2	1e-02	7.746e-03	4.605e-02	4.605e-02	5	5	5	9.017e-04	1.769e+00	20	30	21	0:0:13
	2	1e-03	4.356e-03	4.605e-02	6.908e-02	10	15	15	4.376e-04	1.650e+00	56	66	37	0:1:31
	2	1e-04	2.449e-03	4.605e-02	9.210e-02	18	37	37	1.529e-04	1.503e+00	124	134	61	0:9:21
	3	1e-01	1.516e-02	4.605e-02	2.303e-02	3	1	3	9.988e-03	1.722e+00	4	10	5	0:0:1
	3	1e-02	9.564e-03	4.605e-02	4.605e-02	4	4	4	4.735e-03	1.607e+00	6	14	7	0:0:2
	3	1e-03	6.034e-03	4.605e-02	6.908e-02	7	11	11	2.090e-03	3.752e-01	15	25	4	0:0:3
	3	1e-04	3.807e-03	4.605e-02	9.210e-02	12	24	24	1.274e-03	1.540e+00	27	49	2	0:0:6
	4	1e-01	2.493e-02	4.605e-02	2.303e-02	1	0	1	1.350e-02	1.792e+00	3	9	8	0:0:1
	4	1e-02	1.698e-02	4.605e-02	4.605e-02	2	2	2	1.092e-02	4.440e-01	3	10	7	0:0:1
	4	1e-03	1.157e-02	4.605e-02	6.908e-02	3	5	5	5.969e-03	1.799e+00	7	13	4	0:0:1
	4	1e-04	7.883e-03	4.605e-02	9.210e-02	5	11	11	2.195e-03	3.807e-01	16	25	4	0:0:4
	5	1e-01	2.511e-02	4.605e-02	2.303e-02	1	0	1	1.388e-02	1.665e+00	3	9	8	0:0:1
	5	1e-02	1.807e-02	4.605e-02	4.605e-02	2	2	2	1.476e-02	1.763e+00	3	9	6	0:0:1
	5	1e-03	1.301e-02	4.605e-02	6.908e-02	3	5	5	6.902e-03	1.756e+00	7	13	4	0:0:1
	5	1e-04	9.361e-03	4.605e-02	9.210e-02	4	9	9	6.076e-03	1.765e+00	12	20	3	0:0:3
	5	1e-05	6.737e-03	4.605e-02	1.151e-01	6	17	17	4.202e-03	1.438e+00	12	35	2	0:0:5
	5	1e-08	2.511e-03	4.605e-02	1.842e-01	18	73	73	4.547e-04	1.609e+00	118	147	2	0:1:11
	8	1e-01	4.967e-02	4.605e-02	2.303e-02	0	0	1	6.589e-02	1.272e+00	0	7	6	0:0:1
	8	1e-02	3.946e-02	4.605e-02	4.605e-02	1	1	1	6.589e-02	1.272e+00	0	7	6	0:0:1
	8	1e-03	3.134e-02	4.605e-02	6.908e-02	1	2	2	2.877e-02	1.688e+00	2	8	5	0:0:1
	8	1e-04	2.490e-02	4.605e-02	9.210e-02	1	3	3	2.547e-02	1.680e+00	3	9	4	0:0:1
	8	1e-05	1.977e-02	4.605e-02	1.151e-01	2	5	5	2.164e-02	1.610e+00	5	12	3	0:0:2
	8	1e-08	9.911e-03	4.605e-02	1.842e-01	4	18	18	9.911e-03	1.009e-01	17	36	1	0:0:4
	8	1e-10	6.253e-03	4.605e-02	2.303e-01	7	36	36	6.253e-03	4.916e-02	35	72	1	0:0:13
	8	1e-12	3.946e-03	4.605e-02	2.763e-01	11	70	70	7.220e-04	1.416e+00	116	141	2	0:1:42
B	2	1e-01	-	-	-	-	-	-	3.276e-03	1.306e+00	7	13	10	0:0:2
	2	1e-02	-	-	-	-	-	-	6.919e-04	1.129e+00	21	30	27	0:0:14
	2	1e-03	-	-	-	-	-	-	1.818e-04	1.358e+00	60	73	70	0:2:15
	2	1e-04	-	-	-	-	-	-	1.152e-04	1.502e+00	145	563	151	0:17:30
	3	1e-01	-	-	-	-	-	-	7.813e-03	1.828e+00	5	10	7	0:0:1
	3	1e-02	-	-	-	-	-	-	4.897e-03	1.828e+00	7	13	10	0:0:2
	3	1e-03	-	-	-	-	-	-	2.576e-03	1.495e+00	14	22	19	0:0:8
	3	1e-04	-	-	-	-	-	-	1.715e-03	6.322e-01	30	41	38	0:0:39
	4	1e-01	-	-	-	-	-	-	1.401e-02	1.789e+00	3	9	6	0:0:1
	4	1e-02	-	-	-	-	-	-	1.144e-02	4.147e-01	3	10	7	0:0:1
	4	1e-03	-	-	-	-	-	-	6.678e-03	8.337e-01	6	13	10	0:0:3
	4	1e-04	-	-	-	-	-	-	4.393e-03	1.142e+00	10	17	14	0:0:5
	5	1e-01	-	-	-	-	-	-	1.385e-02	1.705e+00	3	9	6	0:0:1
	5	1e-02	-	-	-	-	-	-	1.385e-02	1.705e+00	3	9	6	0:0:1
	5	1e-03	-	-	-	-	-	-	1.385e-02	1.705e+00	4	9	6	0:0:1
	5	1e-04	-	-	-	-	-	-	6.482e-03	9.645e-01	8	15	12	0:0:4
	5	1e-05	-	-	-	-	-	-	5.547e-03	1.176e+00	10	17	14	0:0:6
	5	1e-08	-	-	-	-	-	-	1.278e-03	8.340e-01	45	56	53	0:2:6
	8	1e-01	-	-	-	-	-	-	6.974e-02	1.235e+00	0	7	4	0:0:1
	8	1e-02	-	-	-	-	-	-	2.629e-02	1.577e+00	1	8	5	0:0:1
	8	1e-03	-	-	-	-	-	-	2.629e-02	1.577e+00	2	8	5	0:0:1
	8	1e-04	-	-	-	-	-	-	2.629e-02	1.577e+00	3	8	5	0:0:1
	8	1e-05	-	-	-	-	-	-	1.659e-02	1.235e+00	5	11	8	0:0:3
	8	1e-08	-	-	-	-	-	-	1.028e-02	5.085e-01	10	19	16	0:0:12
	8	1e-10	-	-	-	-	-	-	5.766e-03	1.936e-01	18	31	28	0:0:40
	8	1e-12	-	-	-	-	-	-	2.620e-03	8.203e-02	35	57	54	0:3:20

A : *using predicted initial subinterval*
B : *using 4-equal initial subinterval*

The last four columns of *Table 7.1c* present some numerical results using the modified algorithm, in which '*mod*' stands for modified algorithm, '*4-eq*' and '*without mod*' indicate that 4-equal initial subintervals and predicted subinterval without modification are used respectively. Comparing the modified algorithm and

without modification, it is clear that the modified algorithm gives better estimate w^0 indicated by impressive reduction in both the actual number of subintervals w^0 in the layer region and actual number of subintervals. It is observed that in some cases the modified algorithm still produces larger w^c than those using 4-equally spaced initial mesh. However, if we take a look at the computation time, using 4-equally spaced initial mesh needs longer computation time. It is also notable that since in these cases the estimate w^0 is close to the associated w^c in the column under the heading 4-*eq* indicating that the estimate w^0 is reasonable, we may suspect that the large w might be caused by putting too many points in the smooth region.

Table 7.1c

				4-eq				without mod				mod			
q	TOL	w_1^0	w_2^0	w^0	w^c	w	Time	w^0	w^c	w	Time	w^0	w^c	w	Time
RH :															
5	1e-05	6	17	-	9	12	0:0:2	17	27	35	0:0:5	13	17	27	0:0:2
5	1e-08	18	73	-	32	35	0:0:39	73	133	147	0:1:11	36	57	73	0:0:19
8	1e-05	2	5	-	5	8	0:0:1	5	4	11	0:0:1	4	4	10	0:0:1
8	1e-08	4	18	-	8	10	0:0:2	18	17	36	0:0:4	9	12	20	0:0:2
8	1e-10	7	36	-	13	16	0:0:8	36	35	72	0:0:13	14	16	29	0:0:5
8	1e-12	11	70	-	25	27	0:0:30	70	100	141	0:1:39	23	27	47	0:0:23
de Boor :															
5	1e-05	6	17	-	10	17	0:0:11	17	12	35	0:0:5	13	19	28	0:0:28
5	1e-08	18	73	-	45	56	0:2:6	73	118	147	0:1:11	36	48	74	0:1:08
8	1e-05	2	5	-	5	11	0:0:3	5	5	12	0:0:2	4	5	11	0:0:2
8	1e-08	4	18	-	10	19	0:0:12	18	17	36	0:0:4	9	11	20	0:0:8
8	1e-10	7	36	-	18	31	0:0:40	36	35	72	0:0:23	14	18	30	0:0:19
8	1e-12	11	70	-	35	57	0:3:20	70	116	141	0:1:42	23	19	53	0:1:12

As the last illustration for problem 1, *Table 7.2* contains results of numerical experiments using the estimate for number of subintervals needed w^* (the details is in chapter 6) to be used in the next stage of the collocation process. Here, the problem parameter is taken to be 10^5 and Chebyshev zeros are chosen as collocation points. For comparison purposes, the results for single point increment algorithm are also presented. As before, here we also compare the experimental results of using predicted initial mesh points and 4-equal subintervals initial mesh points.

From *Table* 7.2, in comparing part *A* and part *B*, by looking carefully at columns under heading *i*, it is clear that using predicted initial subintervals could improve the

RH algorithm performance even though for some i in A it is occasionally worse than those in B. Comparing the results using multi and single subinterval increment, it is observed that using multi points performs quite better than using single subinterval algorithm indicated by smaller w^c and i. Perhaps, the most notable observation from this table is the fact that using single point increment may lead to putting too many points in some regions before the collocation solution dramatically improves, these are shown in most cases of using single subinterval increment. This phenomenon can be understood since in single interval increment algorithms, there is no direct relationship between obtained collocation solution and the desired tolerance *TOL*, for which in any stage of collocation process the algorithm just tries to improve the accuracy of collocation solution by adding one more subinterval in the next iteration. From these results it is clear that using multi points algorithm is not just giving better performance in term of number of iteration needed but also providing a more reliable algorithm.

Table 7.2

(problem 1, RH Criterion, $\mu = 10^5$)

							multi-points			1-point increment		
	q	TOL	h_p	w_1^c	w_2^c	w^c	h_{first}	w^c	i	h_{first}	w^c	i
A	3	1e-02	9.564e-06	12	4	12	7.659e-07	49	10	3.764e-07	126	117
	3	1e-03	6.034e-06	19	11	19	4.300e-07	88	9	3.764e-07	126	103
	3	1e-04	3.807e-06	30	24	30	1.303e-07	190	14	2.736e-07	144	98
	4	1e-02	1.698e-05	6	2	6	2.349e-06	19	11	9.737e-07	84	101
	4	1e-03	1.157e-05	9	5	9	7.816e-07	58	10	9.737e-07	84	95
	4	1e-04	7.883e-06	14	11	14	9.077e-07	50	10	9.737e-07	84	85
	5	1e-02	1.807e-05	6	2	6	5.220e-06	10	11	1.567e-06	62	84
	5	1e-03	1.301e-05	8	5	8	2.026e-06	25	10	1.568e-06	62	80
	5	1e-04	9.361e-06	12	9	12	1.229e-06	32	17	1.570e-06	62	72
	8	1e-02	3.946e-05	2	1	2	1.455e-05	4	14	1.303e-05	8	66
	8	1e-03	3.134e-05	3	2	3	9.301e-06	7	12	1.319e-05	8	64
	8	1e-04	2.490e-05	4	3	4	7.281e-06	9	19	1.329e-05	8	62
B	3	1e-02	-	-	-	-	9.059e-07	42	22	3.764e-07	126	137
	3	1e-03	-	-	-	-	2.581e-07	147	19	3.764e-07	126	137
	3	1e-04	-	-	-	-	1.613e-07	234	19	2.426e-07	158	166
	4	1e-02	-	-	-	-	2.332e-06	19	25	9.737e-07	84	109
	4	1e-03	-	-	-	-	8.587e-07	53	24	9.737e-07	84	109
	4	1e-04	-	-	-	-	1.009e-06	47	16	9.737e-07	84	109
	5	1e-02	-	-	-	-	5.235e-06	10	33	1.566e-06	62	92
	5	1e-03	-	-	-	-	2.007e-06	26	23	1.566e-06	62	92
	5	1e-04	-	-	-	-	1.977e-06	25	16	1.566e-06	62	92
	8	1e-02	-	-	-	-	2.281e-05	3	21	1.357e-05	8	66
	8	1e-03	-	-	-	-	1.397e-05	6	22	1.357e-05	8	66
	8	1e-04	-	-	-	-	1.018e-05	7	16	1.357e-05	8	66

A : *using predicted initial subintervals*
B : *using 4-equal initial subintervals*

As a further illustration, an example is given to show how well the estimates w_1^o and w_2^o perform in case the problem has two layers one at each end. The problem is also used to observe the effect of using Gauss and Chebyshev points as the collocation points.

The problem is

Problem 2 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \mu\cos^2(\pi t) + 2\pi^2\cos(2\pi t) \end{pmatrix}$$

The boundary conditions are

$$x_1(0) = 0$$

and

$$x_1(1) = 0$$

This 'real' problem has been considered in chapter 4. Here we shall carry out some numerical experiments by setting the problem parameter to be large.

Table 7.3a-7.3b and *Table 7.4* present results of numerical experiment with problem parameter $\mu = 10^6$ and $\mu = 10^{10}$ respectively. The results in the tables indicate that using estimates w_L^o and w_R^o give satisfactory results, even though in some case they occasionally overestimate the w^c slightly, for example in the first part of *Table 7.3a* for $q = 8$ and $TOL = 10^{-4}$ and $TOL = 10^{-5}$. Furthermore the results show that the width of the first subinterval and the last subinterval in the final iteration are reasonably close to h_p, this means the estimate h_p is quite satisfactory.

Comparing results in *Table 7.3a* and *Table 7.3b,* we observe that the *RH* algorithm with predicted number of break points performs better than those using single subinterval increment. Looking at the computation time and in particular the number of iterations, in most cases using multi points algorithm needs shorter computation time and smaller numbers of iteration.

Table 7.3a

(problem 2 , single subinterval increment, $\mu = 10^6$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	E	w_L^c	w_R^c	w	i	T
							I. Criterion : RH (Gauss Points)							
4G	1e-02	1.698e-03	4	2	4	4	8.118e-04	8.118e-04	9.532e-03	4	4	27	16	0:0:15
4G	1e-03	1.157e-03	5	5	5	5	5.108e-04	5.108e-04	9.384e-04	7	7	38	24	0:0:37
4G	1e-04	7.883e-04	8	11	11	11	3.143e-04	3.143e-04	9.835e-05	13	13	56	24	0:1:25
4G	1e-05	5.371e-04	12	21	21	21	1.950e-04	1.950e-04	8.934e-06	23	23	84	22	0:3:5
5G	1e-02	1.807e-03	3	2	3	3	1.275e-03	1.275e-03	4.321e-03	3	3	19	11	0:0:7
5G	1e-03	1.301e-03	5	5	5	5	9.411e-04	9.411e-04	6.548e-04	4	4	23	9	0:0:9
5G	1e-04	9.361e-04	7	9	9	9	6.103e-04	6.103e-04	6.430e-05	8	8	33	7	0:0:15
5G	1e-05	6.737e-04	10	17	17	17	3.499e-04	3.499e-04	2.277e-06	15	15	54	4	0:0:22
8G	1e-02	3.946e-03	1	1	1	1	3.672e-03	3.690e-03	6.571e-04	1	1	13	11	0:0:5
8G	1e-03	3.134e-03	2	2	2	2	3.457e-03	3.483e-03	4.203e-04	1	1	13	8	0:0:4
8G	1e-04	2.490e-03	2	3	3	3	2.580e-03	2.632e-03	5.343e-05	2	2	14	6	0:0:4
8G	1e-05	1.977e-03	3	5	5	5	1.606e-03	1.606e-03	2.027e-06	4	4	19	5	0:0:6
							II. Criterion : de Boor (Gauss Points)							
4G	1e-02	1.698e-03	4	2	4	4	8.703e-04	8.703e-04	8.096e-03	5	5	47	36	0:1:6
4G	1e-03	1.157e-03	5	5	5	5	5.465e-04	5.465e-04	9.207e-04	8	8	67	53	0:2:52
4G	1e-04	7.883e-04	8	11	11	11	2.305e-04	2.258e-04	7.531e-05	16	15	120	88	0:13:46
4G	1e-05	5.371e-04	12	21	21	21	1.724e-04	1.525e-04	9.103e-06	29	27	195	133	0:55:20
5G	1e-02	1.807e-03	3	2	3	3	1.627e-03	1.627e-03	8.667e-03	3	3	33	25	0:0:32
5G	1e-03	1.301e-03	5	5	5	5	1.027e-03	1.027e-03	7.334e-04	5	5	47	33	0:1:16
5G	1e-04	9.361e-04	7	9	9	9	6.916e-04	6.916e-04	8.030e-05	8	8	73	47	0:4:0
5G	1e-05	6.737e-04	10	17	17	17	3.549e-04	3.984e-04	5.793e-06	14	15	107	57	0:11:6
8G	1e-02	3.946e-03	1	1	1	1	4.853e-03	4.853e-03	4.779e-03	1	1	20	18	0:0:14
8G	1e-03	3.134e-03	2	2	2	2	3.666e-03	3.666e-03	6.487e-04	1	1	24	19	0:0:21
8G	1e-04	2.490e-03	2	3	3	3	2.701e-03	2.701e-03	6.453e-05	2	2	30	22	0:0:36
8G	1e-05	1.977e-03	3	5	5	5	2.071e-03	2.071e-03	7.941e-06	4	4	37	23	0:0:58
							III.Criterion: RH (Chebyshev Points)							
4C	1e-02	1.698e-03	4	2	4	4	7.152e-04	7.152e-04	9.576e-03	5	5	28	17	0:0:17
4C	1e-03	1.157e-03	5	5	5	5	4.234e-04	4.234e-04	8.695e-04	9	9	42	28	0:0:50
4C	1e-04	7.883e-04	8	11	11	11	2.530e-04	2.530e-04	9.349e-05	17	17	65	33	0:2:18
4C	1e-05	5.371e-04	12	21	21	21	1.481e-04	1.481e-04	9.498e-06	30	30	105	43	0:7:51
5C	1e-02	1.807e-03	3	2	3	3	1.327e-03	1.327e-03	6.554e-03	3	3	18	10	0:0:6
5C	1e-03	1.301e-03	5	5	5	5	8.951e-04	8.951e-04	7.668e-04	4	4	24	10	0:0:11
5C	1e-04	9.361e-04	7	9	9	9	6.273e-04	6.273e-04	8.587e-05	7	7	32	6	0:0:12
5C	1e-05	6.737e-04	10	17	17	17	3.755e-04	3.755e-04	9.254e-06	14	14	53	3	0:0:16
8C	1e-02	3.946e-03	1	1	1	1	3.250e-03	3.333e-03	4.101e-04	1	1	12	10	0:0:4
8C	1e-03	3.134e-03	2	2	2	2	2.777e-03	2.785e-03	1.250e-04	2	2	13	8	0:0:4
8C	1e-04	2.490e-03	2	3	3	3	2.176e-03	2.183e-03	5.378e-05	2	2	15	7	0:0:5
8C	1e-05	1.977e-03	3	5	5	5	1.619e-03	1.619e-03	3.388e-06	4	4	19	5	0:0:6
							IV.Criterion: de Boor(Chebyshev)							
4C	1e-02	1.698e-03	4	2	4	4	7.426e-04	7.426e-04	8.808e-03	6	6	53	42	0:1:31
4C	1e-03	1.157e-03	5	5	5	5	3.895e-04	3.898e-04	6.986e-04	10	10	81	67	0:4:49
4C	1e-04	7.883e-04	8	11	11	11	1.772e-04	2.012e-04	6.409e-05	20	18	147	115	0:24:50
4C	1e-05	5.371e-04	12	21	21	21	9.478e-05	9.605e-05	6.954e-06	42	40	272	210	2:29:16
5C	1e-02	1.807e-03	3	2	3	3	1.485e-03	1.485e-03	8.106e-03	3	3	37	29	0:0:43
5C	1e-03	1.301e-03	5	5	5	5	9.715e-04	9.715e-04	8.271e-04	5	5	55	41	0:1:59
5C	1e-04	9.361e-04	7	9	9	9	6.521e-04	6.521e-04	9.037e-05	8	8	71	45	0:3:43
5C	1e-05	6.737e-04	10	17	17	17	3.825e-04	7.520e-04	9.387e-06	13	12	102	52	0:9:13
8C	1e-02	3.946e-03	1	1	1	1	4.287e-03	4.287e-03	3.071e-03	1	1	21	19	0:0:15
8C	1e-03	3.134e-03	2	2	2	2	3.645e-03	3.645e-03	9.567e-04	1	1	24	19	0:0:21
8C	1e-04	2.490e-03	2	3	3	3	2.389e-03	2.389e-03	3.893e-05	3	3	33	25	0:0:46
8C	1e-05	1.977e-03	3	5	5	5	1.970e-03	1.970e-03	8.452e-06	4	4	40	26	0:1:12

Table 7.3b

(problem 2 , multiple subintervals , $\mu = 10^6$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	E	w_L^c	w_R^c	w	i	T
							RH, Gauss Points							
4G	1e-02	1.698e-03	4	2	4	4	8.492e-04	8.492e-04	7.230e-03	6	6	36	5	0:0:10
4G	1e-03	1.157e-03	5	5	5	5	5.486e-04	5.486e-04	9.851e-04	7	8	38	8	0:0:14
4G	1e-04	7.883e-04	8	11	11	11	1.601e-04	1.601e-04	7.388e-06	27	28	111	6	0:1:16
4G	1e-05	5.371e-04	12	21	21	21	2.056e-04	2.056e-04	8.161e-06	23	23	84	9	0:0:44
5G	1e-02	1.807e-03	3	2	3	3	1.542e-03	1.542e-03	6.732e-03	3	3	20	6	0:0:5
5G	1e-03	1.301e-03	5	5	5	5	1.054e-03	1.054e-03	8.568e-04	5	5	26	5	0:0:8
5G	1e-04	9.361e-04	7	9	9	9	6.565e-04	6.565e-04	5.975e-05	7	7	37	9	0:0:13
5G	1e-05	6.737e-04	10	17	17	17	4.510e-04	4.510e-04	7.973e-06	12	12	47	7	0:0:20
8G	1e-02	3.946e-03	1	1	1	1	2.692e-03	3.080e-03	1.275e-03	2	2	16	5	0:0:4
8G	1e-03	3.134e-03	2	2	2	2	3.134e-03	3.134e-03	2.017e-04	2	2	18	4	0:0:4
8G	1e-04	2.490e-03	2	3	3	3	2.681e-03	2.681e-03	6.946e-05	2	2	17	7	0:0:6
8G	1e-05	1.977e-03	3	5	5	5	1.925e-03	1.925e-03	4.807e-06	3	3	18	5	0:0:5
							RH, Chebyshev Points							
4C	1e-02	1.698e-03	4	2	4	4	4.246e-04	4.246e-04	5.975e-03	8	8	40	7	0:0:12
4C	1e-03	1.157e-03	5	5	5	5	2.470e-04	2.470e-04	2.867e-04	15	15	69	11	0:0:38
4C	1e-04	7.883e-04	8	11	11	11	1.545e-04	1.545e-04	1.757e-05	28	28	111	5	0:1:14
4C	1e-05	5.371e-04	12	21	21	21	9.702e-05	9.702e-05	3.032e-06	48	48	164	10	0:2:1
5C	1e-02	1.807e-03	3	2	3	3	1.472e-03	1.472e-03	8.521e-03	3	2	21	6	0:0:5
5C	1e-03	1.301e-03	5	5	5	5	4.953e-04	4.953e-04	9.453e-05	8	8	41	4	0:0:9
5C	1e-04	9.361e-04	7	9	9	9	6.474e-04	6.474e-04	8.676e-05	7	7	36	9	0:0:12
5C	1e-05	6.737e-04	10	17	17	17	2.179e-04	2.179e-04	1.200e-06	24	24	86	5	0:0:18
8C	1e-02	3.946e-03	1	1	1	1	2.713e-03	3.885e-03	5.190e-04	2	1	18	4	0:0:4
8C	1e-03	3.134e-03	2	2	2	2	3.618e-03	3.596e-03	9.065e-04	1	1	14	4	0:0:3
8C	1e-04	2.490e-03	2	3	3	3	1.394e-03	1.396e-03	2.525e-06	4	4	21	9	0:0:5
8C	1e-05	1.977e-03	3	5	5	5	1.020e-03	1.020e-03	4.187e-07	6	6	28	8	0:0:6

When the problem 2 is set up having very severe layer ($\mu = 10^{10}$) , the results presented in the last two rows of *Table 7.4* demonstrate a more dramatic difference, for example collocating at 4 Chebyshev points to obtain an accuracy of order 10^{-2} using single point increment algorithm needs 329 iterations (computation time 7 hrs, 11 mins and 19 secs), while using multi points algorithm just needs 9 iterations (computation time 6 mins and 3 secs. Detailed inspection of width of the first and last subintervals produced by both algorithms we can see that they are reasonably close to each other. However, by looking at the values of w_L^o and w_R^o, the single point increment algorithm clearly places more points at the layer regions than those using multi points. It is clear from *Table 7.4* that it is not sensible to use single subinterval increment algorithm in 'real' computation.

Looking again at *Table 7.3a* it is observed that de Boor algorithm using Chebyshev points gives rather poorer results compared to those using Gauss points. Perhaps, these results can be understood by noting that in the de Boor algorithm the

process involves the collocation solution, where a more accurate collocation solution might result in better mesh point distribution. Since using Gauss points could produce better collocation solutions, one may expect that this may produce a better mesh point distribution.

Comparing the results for *RH* criterion using Gauss and Chebyshev points by looking carefully at the number of iteration i, number of subintervals w and computation time T in *Table 7.3a-7.3b* and *Table 7.4* clearly shows that Chebyshev points give satisfactory results, and it is very competitive to those using Gauss points.

Table 7.4

(problem parameter $\mu = 10^{10}$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	Err	w_L^c	w_R^c	i	T
4G	1e-02	1.698e-05	6	2	6	6	1.646e-06	1.646e-06	9.057e-03	28	27	9	0:9:32
4C	1e-02	1.698e-05	6	2	6	6	1.557e-06	1.557e-06	5.631e-03	28	28	9	0:6:3
5G	1e-02	1.807e-05	6	2	6	6	3.513e-06	3.513e-06	4.434e-03	14	14	12	0:6:35
5C	1e-02	1.807e-05	6	2	6	6	3.202e-06	3.202e-06	1.100e-03	16	16	9	0:4:37
5G	1e-03	1.301e-05	8	5	8	8	4.365e-06	4.365e-06	6.050e-04	12	12	9	0:8:40
5C	1e-03	1.301e-05	8	5	8	8	2.205e-06	2.205e-06	5.321e-05	23	24	11	0:6:17
8G	1e-02	3.946e-05	2	1	2	2	2.796e-05	2.792e-05	9.035e-03	3	3	15	0:4:40
8C	1e-02	3.946e-05	2	1	2	2	1.335e-05	1.340e-05	3.479e-04	5	6	11	0:2:36
8G	1e-03	3.134e-05	3	2	3	3	2.093e-05	2.094e-05	9.602e-04	3	3	9	0:3:44
8C	1e-03	3.134e-05	3	2	3	3	1.025e-05	1.025e-05	2.772e-05	6	6	12	0:2:51
8G	1e-04	2.490e-05	4	3	4	4	7.593e-06	7.593e-06	1.610e-06	8	8	10	0:4:47
8C	1e-04	2.490e-05	4	3	4	4	7.532e-06	7.532e-06	7.965e-06	8	8	9	0:4:37
8G	1e-05	1.977e-05	5	5	5	5	1.111e-05	1.111e-05	8.624e-06	6	6	8	0:7:48
8C	1e-05	1.977e-05	5	5	5	5	5.761e-06	5.761e-06	2.743e-07	11	11	9	0:5:4
8G	1e-08	9.911e-06	11	18	18	18	2.721e-06	2.721e-06	2.033e-10	29	29	6	0:24:11
8C	1e-08	9.911e-06	11	18	18	18	3.722e-06	3.722e-06	7.851e-10	21	21	7	0:23:17
10G	1e-02	5.175e-05	2	0	2	2	2.167e-05	2.167e-05	2.317e-03	3	3	11	0:3:11
10C	1e-02	5.175e-05	2	0	2	2	2.492e-05	2.499e-05	8.913e-04	3	3	13	0:2:3
10G	1e-03	4.271e-05	2	1	2	2	1.787e-05	1.788e-05	1.616e-05	4	4	13	0:3:45
10C	1e-03	4.271e-05	2	1	2	2	1.829e-05	1.831e-05	2.621e-05	4	4	12	0:2:27
10G	1e-04	3.526e-05	3	2	3	3	1.469e-05	1.469e-05	1.555e-06	5	5	10	0:3:17
10C	1e-04	3.526e-05	3	2	3	3	1.508e-05	1.508e-05	1.190e-06	5	5	11	0:2:47
10G	1e-05	2.910e-05	3	3	3	3	1.112e-05	1.112e-05	8.352e-08	6	6	10	0:4:42
10C	1e-05	2.910e-05	3	3	3	3	1.113e-05	1.113e-05	1.116e-07	6	6	10	0:4:10
10G	1e-08	1.636e-05	7	11	11	11	7.164e-06	7.164e-06	9.479e-10	12	12	7	0:12:50
10C	1e-08	1.636e-05	7	11	11	11	7.666e-06	7.666e-06	1.794e-10	12	12	7	0:25:8
single increment:													
4G	1e-02	1.698e-05	6	2	6	6	5.721e-06	4.819e-06	9.389e-03	43	41	372	8:35:42
4C	1e-02	1.698e-05	6	2	6	6	9.228e-06	8.472e-06	8.972e-03	41	41	329	7:11:19

To illustrate the behaviour of the above aspects when the matrix $A(t)$ has eigenvalues which vary in the specified interval, we consider the following example

Problem 3 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu(2-t^2) & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu \end{pmatrix}$$

The boundary conditions are determined by

$$x_1(-1) = 0$$

$$x_1(1) = 0$$

and the eigenvalues are

$$\lambda_1(t) = -\sqrt{\mu(2-t^2)}$$
$$\lambda_2(t) = \sqrt{\mu(2-t^2)}$$

μ, $|\mu| >> 1$, is the problem parameter. This problem has boundary layers at both end points.

The first eigenvalue $\lambda_1(t)$ has negative values in whole interval [-1,1], while the second one has positive values. The eigenvalues at both ends are $\lambda_1 = -\sqrt{\mu}$ and $\lambda_2 = \sqrt{\mu}$ which will be used for predicting the width of layer regions and determining the initial mesh in the layer regions. Results of numerical experiments with problem parameter $\mu = 10^4$ are presented in *Table 7.5* on the following page.

For this problem, as in the previous problems it is observed that the estimates w_L^o and w_R^o are reasonably close to w_L^c and w_R^c, and the break points are well distributed in the layer regions indicated by the fact that the width of the first and the last subinterval are very close to h_p. It is again shown that in terms of number of iterations and time needed in most cases the de Boor algorithm using Gauss points produces better results than using Chebyshev points, on the other hand the *RH* algorithm using Chebyshev points give reasonable results.

Comparing the *RH* algorithm and de Boor algorithm it is clear that the *RH* algorithm performs significantly better than the de Boor algorithm.

Table 7.5

(problem parameter $\mu = 10^4$)

					RH							de Boor					
q	TOL	h_p	w_L^o	w_R^o	h_{first}	h_{last}	w_L^c	w_R^c	w	i	T	h_{first}	h_{last}	w_L^c	w_R^c	w	i	T
3G	1e-02	6.762e-03	5	5	6.062e-03	6.062e-03	4	4	23	9	0:0:6	6.848e-03	6.868e-03	3	3	29	15	0:0:13
3C	1e-02	6.762e-03	5	5	5.426e-03	5.426e-03	4	4	25	11	0:0:8	5.548e-03	5.551e-03	4	4	33	19	0:0:19
3G	1e-03	4.267e-03	11	11	3.481e-03	3.481e-03	8	8	37	5	0:0:9	2.700e-03	2.661e-03	10	11	57	25	0:1:9
3C	1e-03	4.267e-03	11	11	3.148e-03	3.148e-03	9	9	40	8	0:0:16	2.160e-03	1.960e-03	12	10	65	33	0:1:46
3G	1e-04	2.692e-03	24	24	1.648e-03	1.648e-03	20	20	74	3	0:0:19	1.029e-03	1.170e-03	23	21	105	34	0:5:2
3C	1e-04	2.692e-03	24	24	1.574e-03	1.574e-03	21	21	75	4	0:0:27	8.847e-04	1.012e-03	28	26	127	56	0:10:38
3G	1e-05	1.699e-03	47	47	8.117e-04	8.117e-04	45	45	143	3	0:1:5	5.903e-04	6.138e-04	41	47	196	56	0:28:10
3C	1e-05	1.699e-03	47	47	8.012e-04	8.012e-04	45	45	143	3	0:1:6	5.436e-04	9.253e-04	63	57	259	119	1:23:31
4G	1e-02	1.201e-02	2	2	1.233e-02	1.232e-02	2	2	15	10	0:0:3	1.427e-02	1.427e-02	2	2	21	16	0:0:8
4C	1e-02	1.201e-02	2	2	1.059e-02	1.059e-02	2	2	16	11	0:0:4	1.283e-02	1.283e-02	2	2	23	18	0:0:11
4G	1e-03	8.182e-03	5	5	7.549e-03	7.549e-03	4	4	22	8	0:0:6	8.745e-03	8.745e-03	4	4	31	17	0:0:20
4C	1e-03	8.182e-03	5	5	7.312e-03	7.312e-03	4	4	22	8	0:0:6	7.715e-03	7.715e-03	4	4	36	22	0:0:31
4G	1e-04	5.574e-03	11	11	4.437e-03	4.437e-03	8	8	35	3	0:0:6	5.473e-03	4.479e-03	8	8	47	15	0:0:42
4C	1e-04	5.574e-03	11	11	4.299e-03	4.299e-03	9	8	35	3	0:0:6	3.915e-03	3.021e-03	9	11	57	25	0:1:27
4G	1e-05	3.798e-03	21	21	2.238e-03	2.238e-03	19	18	65	3	0:0:18	2.480e-03	2.125e-03	16	16	80	18	0:2:15
4C	1e-05	3.798e-03	21	21	2.202e-03	2.202e-03	19	19	65	3	0:0:18	1.711e-03	1.990e-03	19	19	95	33	0:4:59
5G	1e-02	1.278e-02	2	2	2.175e-02	2.136e-02	1	1	11	6	0:0:2	2.362e-02	2.362e-02	1	1	16	11	0:0:5
5C	1e-02	1.278e-02	2	2	2.103e-02	2.079e-02	1	1	11	6	0:0:2	2.213e-02	2.213e-02	1	1	17	12	0:0:6
5G	1e-03	9.198e-03	5	5	1.089e-02	1.089e-02	3	3	18	4	0:0:3	1.692e-02	1.692e-02	2	2	21	7	0:0:6
5C	1e-03	9.198e-03	5	5	1.084e-02	1.084e-02	3	3	18	4	0:0:3	1.536e-02	1.536e-02	2	2	23	9	0:0:9
5G	1e-04	6.619e-03	9	9	6.350e-03	6.350e-03	6	6	29	3	0:0:5	1.013e-02	1.015e-02	4	4	33	7	0:0:14
5C	1e-04	6.619e-03	9	9	6.192e-03	6.192e-03	6	6	29	3	0:0:5	1.022e-02	1.023e-02	4	4	33	7	0:0:14
5G	1e-05	4.764e-03	17	17	3.551e-03	3.551e-03	14	14	52	2	0:0:10	6.181e-03	6.181e-03	11	11	53	3	0:0:15
5C	1e-05	4.764e-03	17	17	3.558e-03	3.558e-03	14	14	52	2	0:0:10	6.348e-03	6.348e-03	11	11	53	3	0:0:15
8G	1e-02	2.790e-02	1	1	4.724e-02	4.822e-02	0	0	8	6	0:0:1	6.682e-02	6.682e-02	0	0	10	8	0:0:2
8C	1e-02	2.790e-02	1	1	5.866e-02	5.975e-02	0	0	8	6	0:0:1	5.786e-02	5.786e-02	0	0	11	9	0:0:3
8G	1e-03	2.216e-02	2	2	4.846e-02	4.915e-02	1	0	9	4	0:0:1	5.302e-02	5.302e-02	0	0	12	7	0:0:3
8C	1e-03	2.216e-02	2	2	4.386e-02	4.434e-02	1	1	9	4	0:0:1	5.116e-02	5.116e-02	0	0	12	7	0:0:3
8G	1e-04	1.760e-02	3	3	2.344e-02	2.344e-02	2	2	12	4	0:0:2	3.876e-02	3.877e-02	1	1	15	7	0:0:5
8C	1e-04	1.760e-02	3	3	2.394e-02	2.394e-02	2	2	12	4	0:0:2	3.555e-02	3.554e-02	1	1	16	8	0:0:6
8G	1e-05	1.398e-02	5	5	1.654e-02	1.654e-02	3	3	17	3	0:0:3	2.864e-02	2.857e-02	2	2	20	6	0:0:7
8C	1e-05	1.398e-02	5	5	1.688e-02	1.688e-02	3	3	17	3	0:0:3	2.509e-02	2.513e-02	2	2	21	7	0:0:9

7.5.2 Mesh Subdivision Algorithm

In order to make numerical comparisons clearer and more straightforward, the examples in the preceding section have been used in carrying out further numerical experiments comparing subdivision algorithms. For convenience, the notations are also retained and we present similar tables to those in § 7.5.1, though in some tables we simplify the tables by reducing some columns and rows.

Unlike the mesh placement algorithms, in implementing the mesh subdivision algorithms the number of subintervals will always increase. In the other words, once we start the computation process with w initial subintervals then in the next iteration the number subintervals increases and we will never get a smaller number of subintervals than the current one, even though the actual number of subintervals

needed is less than w. This means that if the number of initial subintervals in the layer region w^0 is too large compared to the actual requirement, we will never have chance to reduce it, moreover this will force the algorithms to carry out unnecessary computations.

Tables *7.6a - 7.6b - 7.6c* and *Table 7.7* corresponding to problem 1 present the results of numerical experiments with problem parameter $\mu = 10^2$ and $\mu = 10^5$ respectively. In the experiments it is shown that similar observations to those from tables *7.1a - 7.1b - 7.1c* and *Table 7.2* are observed, in which the estimate w^o is satisfactory.

Table 7.6a

(problem 1, RH Criterion, 1-interval increment, $\mu = 10^2$)

q	TOL	h_p	w_1^o	w_2^o	w^o	h_{first}	h_{last}	w^c	w	i	T
A 2	1e-01	1.377e-02	3	1	3	3.444e-03	6.513e-01	7	17	12	0:0:3
2	1e-02	7.746e-03	5	5	5	9.682e-04	3.908e-01	18	30	21	0:0:13
2	1e-03	4.356e-03	10	15	15	5.445e-04	1.287e-01	46	66	37	0:1:32
2	1e-04	2.449e-03	18	37	37	3.062e-04	5.156e-02	107	148	75	0:13:4
5	1e-03	1.301e-02	3	5	5	1.301e-02	3.862e-01	4	13	4	0:0:2
5	1e-04	9.361e-03	4	9	9	9.361e-03	2.120e-01	8	20	3	0:0:3
5	1e-05	6.737e-03	6	17	17	3.369e-03	1.109e-01	17	36	3	0:0:8
5	1e-08	2.511e-03	18	73	73	1.256e-03	2.487e-02	79	153	8	0:4:51
8	1e-04	2.490e-02	1	3	3	2.490e-02	6.360e-01	2	8	3	0:0:1
8	1e-05	1.977e-02	2	5	5	1.977e-02	3.770e-01	4	12	3	0:0:2
8	1e-08	9.911e-03	4	18	18	9.911e-03	1.009e-01	17	36	1	0:0:4
8	1e-10	6.253e-03	7	36	36	6.253e-03	4.916e-02	35	72	1	0:0:14
8	1e-12	3.946e-03	11	70	70	1.973e-03	2.462e-02	71	142	3	0:2:34

q	TOL	h_p	w_1^o	w_2^o	w^o	h_{first}	h_{last}	w^c	w	i	T
B 2	1e-01	-	-	-	-	3.906e-03	5.000e-01	8	16	13	0:0:3
2	1e-02	-	-	-	-	9.766e-04	5.000e-01	18	29	26	0:0:13
2	1e-03	-	-	-	-	4.883e-04	5.000e-01	46	58	55	0:1:16
2	1e-04	-	-	-	-	2.441e-04	5.000e-01	107	119	116	0:8:31
5	1e-03	-	-	-	-	7.812e-03	5.000e-01	5	12	9	0:0:2
5	1e-04	-	-	-	-	7.812e-03	5.000e-01	6	13	10	0:0:3
5	1e-05	-	-	-	-	3.906e-03	5.000e-01	12	19	16	0:0:8
5	1e-08	-	-	-	-	9.766e-04	5.000e-01	41	47	44	0:1:22
8	1e-04	-	-	-	-	3.125e-02	5.000e-01	2	8	5	0:0:1
8	1e-05	-	-	-	-	1.562e-02	5.000e-01	3	9	6	0:0:2
8	1e-08	-	-	-	-	7.812e-03	5.000e-01	10	16	13	0:0:8
8	1e-10	-	-	-	-	3.906e-03	5.000e-01	18	23	20	0:0:19
8	1e-12	-	-	-	-	1.953e-03	5.000e-01	31	37	34	0:1:5

Table 7.6a and *Table 7.6b* present some results of numerical experiments using the *RH* and the de Boor criterion functions respectively to the problem-1 with problem parameter $\mu = 10^2$. Comparing *Table 7.6a* and *Table 7.6b*, unlike in the mesh placement algorithm where it is very clear that the *RH* algorithm is very competitive, in the mesh subdivisions the superiority of the *RH* algorithm over de Boor algorithm is less impressive, even though in most cases the *RH* algorithm is still a bit better than the de Boor algorithm. In addition, *Table 7.6b* also illustrates the cases where an inefficiency in computation process could occur when using very crude initial mesh points, as shown for $q = 2$ and $TOL = 10^{-4}$ in which using 4-equal initial mesh points the collocation process requires 400 subintervals (computation time 4 hrs 14 mins and 36 secs) while it only needs 188 subintervals (computation time 27 mins and 8 secs) if the estimate w^0 is employed in the initial mesh.

Table 7.6b

(problem 1, de Boor Criterion, 1-interval increment, $\mu = 10^2$)

	q	TOL	h_p	w_1^0	w_2^0	w^0	h_{first}	h_{last}	w^e	w	i	T
A	2	1e-01	1.377e-02	3	1	3	1.722e-03	6.513e-01	10	19	14	0:0:4
	2	1e-02	7.746e-03	5	5	5	9.682e-04	3.908e-01	29	41	32	0:0:23
	2	1e-03	4.356e-03	10	15	15	5.445e-04	1.287e-01	67	87	58	0:3:26
	2	1e-04	2.449e-03	18	37	37	3.062e-04	5.156e-02	147	188	115	0:27:8
	5	1e-03	1.301e-02	3	5	5	1.301e-02	3.862e-01	4	13	4	0:0:2
	5	1e-04	9.361e-03	4	9	9	9.361e-03	2.120e-01	8	20	3	0:0:3
	5	1e-05	6.737e-03	6	17	17	3.369e-03	1.109e-01	17	37	4	0:0:11
	5	1e-08	2.511e-03	18	73	73	1.256e-03	2.487e-02	79	154	9	0:1:31
	8	1e-04	2.490e-02	1	3	3	2.490e-02	6.360e-01	2	8	3	0:0:1
	8	1e-05	1.977e-02	2	5	5	1.977e-02	3.770e-01	4	12	3	0:0:2
	8	1e-08	9.911e-03	4	18	18	9.911e-03	1.009e-01	17	36	1	0:0:4
	8	1e-10	6.253e-03	7	36	36	6.253e-03	4.916e-02	35	72	1	0:0:13
	8	1e-12	3.946e-03	11	70	70	1.973e-03	2.462e-02	71	142	3	0:2:42
B	2	1e-01	-	-	-	-	1.953e-03	5.000e-01	11	19	16	0:0:5
	2	1e-02	-	-	-	-	9.766e-04	5.000e-01	25	37	34	0:0:24
	2	1e-03	-	-	-	-	4.883e-04	5.000e-01	86	98	95	0:5:2
	2	1e-04	-	-	-	-	1.221e-04	5.000e-01	388	400	397	4:14:36
	5	1e-03	-	-	-	-	7.812e-03	5.000e-01	5	13	10	0:0:3
	5	1e-04	-	-	-	-	7.812e-03	5.000e-01	7	15	12	0:0:4
	5	1e-05	-	-	-	-	3.906e-03	5.000e-01	13	22	19	0:0:11
	5	1e-08	-	-	-	-	9.766e-04	5.000e-01	56	67	64	0:3:22
	8	1e-04	-	-	-	-	3.125e-02	5.000e-01	2	8	5	0:0:1
	8	1e-05	-	-	-	-	1.562e-02	5.000e-01	4	12	9	0:0:4
	8	1e-08	-	-	-	-	7.812e-03	5.000e-01	11	20	17	0:0:13
	8	1e-10	-	-	-	-	3.906e-03	2.500e-01	23	36	33	0:1:1
	8	1e-12	-	-	-	-	1.953e-03	1.250e-01	44	63	60	0:4:26

A : using predicted initial mesh
B : using 4 equal spaced initial mesh

Looking more closely at *Table 7.6a* and *Table 7.6b*, it is notable that in cases w_2^o is greater than $2\,w_1^o$ the numerical results in part-*A* indicate that the algorithm inserts too many subintervals in the layer region compared to those in part-*B*, for example in part-*A* of *Table 7.6a* for $q = 5$ and $TOL = 10^{-8}$ the number of subintervals in the layer region w^c is 79 while it is 41 in part-*B*. This case also occurred in the mesh placement algorithm when we attempted to obtain the higher accuracy solutions using for $q = 5$ and $q = 8$ (see *Table 7.1a*). Similar to the numerical results obtained using the mesh placement algorithms, it seems that these are due to using too many points initially in the boundary layer as well as in the smooth region. As can be seen these cases appear when w_2^o is larger than double w_1^o, hence give us a hint that this might be caused by putting too many points in both the layer and smooth region at the initial stage of computation process. In *Table 7.6c* we use the modification proposed in §7.5.1, i.e. by taking either $w^o = 2\,w_1^o$ if $w_2^o > 2\,w_1^o$ or $w^o = 2\,w_2^o$ in case $w_1^o > 2\,w_2^o$. The last three columns of *Table 7.6c* present some numerical results using the modified algorithm, in which '*mod*' stands for modified algorithm. The notation '*4-eq*' and '*predicted*' indicate using 4-equally spaced initial mesh and predicted mesh (without modification) respectively. It is clear that in both adaptive algorithms the modification gives better estimate w^0 indicated by an impressive reduction in both the final number of subintervals w^0 in the layer region and final total number of subintervals w^c.

Table 7.6c

				4-eq			predicted			mod		
q	TOL	w_1^o	w_2^o	w^o	w^c	w	w^o	w^c	w	w^o	w^c	w
RH												
5	1e-05	6	17	-	12	19	17	17	36	13	14	29
5	1e-08	18	73	-	41	47	73	79	153	36	43	86
8	1e-05	2	5	-	3	9	5	4	12	4	4	11
8	1e-08	4	18	-	10	16	18	17	36	9	9	21
8	1e-10	7	36	-	18	23	36	35	72	14	14	33
8	1e-12	11	70	-	31	37	70	71	142	23	24	55
de Boor												
5	1e-05	6	17	-	13	22	17	17	37	13	14	32
5	1e-08	18	73	-	56	67	73	79	154	36	43	91
8	1e-05	2	5	-	4	12	5	4	12	4	4	11
8	1e-08	4	18	-	11	20	18	17	36	9	9	22
8	1e-10	7	36	-	23	36	36	35	72	14	14	34
8	1e-12	11	70	-	44	63	70	71	142	23	24	69

In the following *Table 7.7* the *RH* algorithm is used. By comparing the numerical results which are obtained by implementing single subinterval increment and multi points algorithm, it is observed that using multi points algorithm is able to improve the performance significantly. Another important observation from *Table 7.7* is that in some cases, even though the predictive initial mesh is employed, the adaptive mesh selection algorithm with single subinterval increment may performs very inefficiently, for example using three collocation points and $TOL = 10^{-3}$, using single subinterval increment needs more than five and half hours while using multi points algorithm the computation time is just about 36 minutes.

The results in *Table 7.7* highlight the cases for which the estimates w_1^o's are greater than w_2^o's, hence in practice the estimate w_1^o will be the actual estimate w^0. By looking at columns under heading h_p and h_{first}, it can be seen that the associated values of these columns are reasonably close, in addition the results in column w^c indicate that the estimate w^0 is also reasonable. Comparing the results in part *A* and part *B*, by looking carefully at the number of iterations i and computation time T it is clear that using predicted initial mesh points dramatically improve the *RH* algorithm performance, especially when using multiple subdivision algorithms in which the estimate w^* is employed.

Table 7.7

(problem 1, RH Criterion, 1-interval increment, $\mu = 10^5$)

						multi-points				single subinterval			
	q	TOL	h_p	w_1^o	w_2^o	w^o	h_{first}	w^c	i	T	h_{first}	w^c	i	T
A	3	1e-02	9.564e-06	12	4	12	7.970e-07	56	28	0:13:41	7.856e-07	56	29	0:30:52
	3	1e-03	6.034e-06	19	11	19	5.028e-07	109	76	0:36:31	3.301e-07	105	85	5:31:23
	3	1e-04	3.807e-06	30	24	30	2.380e-07	180	97	0:55:48	1.877e-07	184	62	1:39:21
	4	1e-02	1.698e-05	6	2	6	2.123e-06	23	38	0:8:6	1.691e-06	22	22	0:13:28
	4	1e-03	1.157e-05	9	5	9	1.286e-06	37	17	0:16:26	1.776e-06	34	39	0:28:25
	4	1e-04	7.883e-06	14	11	14	5.631e-07	56	46	0:37:21	6.414e-07	61	34	0:35:59
	5	1e-02	1.807e-05	6	2	6	4.518e-06	13	41	0:6:31	3.543e-06	12	19	0:16:29
	5	1e-03	1.301e-05	8	5	8	3.252e-06	19	10	0:6:40	2.325e-06	17	18	0:16:11
	5	1e-04	9.361e-06	12	9	12	2.340e-06	26	19	0:13:47	1.786e-06	28	19	0:19:0
	8	1e-02	3.946e-05	2	1	2	1.973e-05	4	55	0:3:48	1.356e-05	4	53	0:3:49
	8	1e-03	3.134e-05	3	2	3	1.567e-05	6	48	0:3:55	1.017e-05	5	51	0:3:49
	8	1e-04	2.490e-05	4	3	4	1.245e-05	8	36	0:3:46	1.085e-05	6	50	0:3:53

...cont'd

...cont'd **Table 7.7**

q	TOL	h_p	w_1^o	w_2^o	w^o	h_{first}	w^c	i	T	h_{first}	w^c	i	T
B 3	1e-02	-	-	-	-	5.977e-07	54	166	0:40:57	9.537e-07	53	180	0:37:34
3	1e-03	-	-	-	-	3.771e-07	98	197	1:13:16	4.768e-07	98	226	1:9:25
3	1e-04	-	-	-	-	2.380e-07	175	253	2:51:17	2.384e-07	176	309	2:51:5
4	1e-02	-	-	-	-	2.123e-06	22	113	0:15:23	1.907e-06	22	124	0:16:23
4	1e-03	-	-	-	-	1.446e-06	35	126	0:22:52	9.537e-07	34	137	0:21:27
4	1e-04	-	-	-	-	9.854e-07	56	136	0:33:15	9.537e-07	56	159	0:32:23
5	1e-02	-	-	-	-	4.518e-06	12	87	0:9:33	3.815e-06	12	94	0:9:16
5	1e-03	-	-	-	-	3.252e-06	17	89	0:11:14	3.815e-06	17	100	0:11:2
5	1e-04	-	-	-	-	2.340e-06	26	90	0:14:15	1.907e-06	27	111	0:15:0
8	1e-02	-	-	-	-	1.973e-05	3	58	0:4:2	1.526e-05	3	57	0:3:52
8	1e-03	-	-	-	-	1.567e-05	5	58	0:4:24	1.526e-05	5	60	0:4:25
8	1e-04	-	-	-	-	1.245e-05	6	58	0:4:48	7.629e-06	6	61	0:4:38

Table 7.8 corresponding to problem-2 with problem parameter $\mu = 10^6$ contains some numerical results obtained by implementing the *RH* and de Boor algorithms. Using the results shown in this table we shall focus on comparing the performance of the *RH* algorithm and de Boor algorithm. On the other hand, *Table 7.9a* and *Table 7.9b* are intended to illustrate the effect of using Gauss and Chebyshev collocation points as well as comparing single and multi points increment when we deal with the problems which have more severe layers indicated by a large problem parameter, i.e. $\mu = 10^{10}$. The results in the tables show that the estimates w_L^o and w_R^o are reasonably close to the actual number of mesh points in the layer region required in the computation.

From *Table 7.8*, the numerical results confirm the indication obtained in the previous section that the *RH* adaptive mesh selection algorithm performs very well and it is much better than the de Boor mesh selection algorithm. An observation can be taken from this table is that in some cases the performance of the de Boor mesh subdivision algorithm is very poor, for example for $q = 8$ and $TOL = 10^{-8}$ the de Boor algorithm required 61 iterations (number of subintervals $w = 114$, computation time $T = 20$ mins and 13 secs) while the *RH* algorithm just needs 8 iterations (number of subintervals $w = 61$, computation time $T = 1$ min and 17 secs). Looking at the columns w_L^c, w_R^c and w, a further important observation which can also be made is that the de Boor algorithm may put too many points in the smooth region, even though the distribution of mesh points in the layer regions are as good as those using

the *RH* algorithm. As can be seen for case q = 8, TOL = 10^{-8} the number of subintervals in the smooth region is 77 (= 114–19–18), while for the *RH* algorithm it is 24 (= 61–19–18), even though the number of mesh points in the layer regions produced by both algorithms is same. From this observation it seems that the de Boor criterion function puts unnecessary break points in the smooth region before placing required points in the layer region.

Table 7.8

(problem parameter $\mu = 10^6$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	Err	w_L^c	w_R^c	w	i	T
I. Criterion : RH (Gauss Points)														
4G	1e-02	1.698e-03	4	2	4	4	8.492e-04	8.492e-04	7.230e-03	6	6	36	25	0:0:34
4G	1e-03	1.157e-03	5	5	5	5	2.893e-04	2.893e-04	9.141e-04	10	9	49	35	0:1:15
4G	1e-04	7.883e-04	8	11	11	11	1.971e-04	1.971e-04	9.960e-05	18	18	68	36	0:2:41
4G	1e-05	5.371e-04	12	21	21	21	1.343e-04	1.343e-04	7.859e-06	35	34	118	56	0:11:34
5G	1e-02	1.807e-03	3	2	3	3	9.037e-04	1.807e-03	7.478e-03	4	3	23	15	0:0:12
5G	1e-03	1.301e-03	5	5	5	5	6.504e-04	6.504e-04	7.207e-04	6	6	32	18	0:0:26
5G	1e-04	9.361e-04	7	9	9	9	4.681e-04	4.681e-04	6.743e-05	11	10	45	19	0:0:55
5G	1e-05	6.737e-04	10	17	17	17	3.369e-04	3.369e-04	9.173e-06	20	20	65	15	0:1:38
8G	1e-02	3.946e-03	1	1	1	1	3.454e-03	3.946e-03	3.010e-03	2	1	16	14	0:0:8
8G	1e-03	3.134e-03	2	2	2	2	3.134e-03	3.134e-03	2.017e-04	2	2	18	13	0:0:10
8G	1e-04	2.490e-03	2	3	3	3	2.490e-03	2.490e-03	4.445e-05	3	3	19	11	0:0:11
8G	1e-05	1.977e-03	3	5	5	5	1.977e-03	1.977e-03	5.473e-06	5	5	25	11	0:0:18
8G	1e-08	9.911e-04	6	18	18	18	4.955e-04	9.911e-04	6.501e-09	19	18	61	8	0:1:17
II. Criterion : de Boor (Gauss Points)														
4G	1e-02	1.698e-03	4	2	4	4	8.492e-04	8.492e-04	7.230e-03	6	5	35	24	0:0:31
4G	1e-03	1.157e-03	5	5	5	5	2.893e-04	5.785e-04	6.759e-04	11	9	71	57	0:3:27
4G	1e-04	7.883e-04	8	11	11	11	1.971e-04	1.971e-04	8.779e-05	19	19	94	62	0:7:3
4G	1e-05	5.371e-04	12	21	21	21	1.343e-04	1.343e-04	7.859e-06	36	36	167	105	0:34:41
5G	1e-02	1.807e-03	3	2	3	3	9.037e-04	9.037e-04	7.478e-03	4	4	40	32	0:0:53
5G	1e-03	1.301e-03	5	5	5	5	6.504e-04	6.504e-04	7.207e-04	6	6	40	26	0:0:49
5G	1e-04	9.361e-04	7	9	9	9	4.681e-04	4.681e-04	6.743e-05	11	10	68	42	0:3:16
5G	1e-05	6.737e-04	10	17	17	17	3.369e-04	3.369e-04	9.173e-06	20	19	114	64	0:13:31
8G	1e-02	3.946e-03	1	1	1	1	3.454e-03	3.946e-03	4.177e-04	2	1	21	19	0:0:16
8G	1e-03	3.134e-03	2	2	2	2	3.134e-03	3.134e-03	2.017e-04	2	2	22	17	0:0:17
8G	1e-04	2.490e-03	2	3	3	3	2.490e-03	2.490e-03	4.445e-05	3	3	24	16	0:0:20
8G	1e-05	1.977e-03	3	5	5	5	1.977e-03	1.977e-03	5.473e-06	5	5	34	20	0:0:46
8G	1e-08	9.911e-04	6	18	18	18	4.955e-04	9.911e-04	6.501e-09	19	18	114	61	0:20:13

By employing the estimate of the number subintervals needed in the collocation process w^* in the *RH* algorithm, some numerical results are presented in *Table 7.9a,* while *Table 7.9b* shows the results using single subinterval increment.

Detailed inspection on columns under heading i and T in *Table 7.9a* and *Table 7.9b* reveal two obvious indications that firstly using multi points algorithm is significantly better than those using single point increment, secondly it is again observed that single subinterval increment algorithm may perform very inefficiently as we can see for q = 4 and TOL = 10^{-4}.

Results in *Table 7.9a* and *Table 7.9b* clearly indicate that in terms of computation time T and number of iterations i, the Chebyshev collocation points may be able to produce a better numerical results compared to those using Gauss collocation points. Further observation on the columns under heading h_p, h_{first}, and h_{last}, it is clear that the h_p is reasonably close to the width of the first and the last subinterval obtained from actual computation, this enables us to come to conclusions that, firstly h_p could be used to obtain a good estimate for the number of initial mesh points in the layer region; secondly the number of break points in both layer regions is reasonable.

Table 7.9a

(multiple subintervals, $\mu = 10^{10}$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	Err	w_L^c	w_R^c	i	T
4G	1e-02	1.698e-05	6	2	6	6	2.831e-06	2.831e-06	5.052e-03	19	19	40	0:9:39
4C	1e-02	1.698e-05	6	2	6	6	1.415e-06	1.415e-06	8.241e-04	36	36	46	0:18:14
4G	1e-03	1.157e-05	9	5	9	9	7.232e-07	7.232e-07	9.377e-05	52	52	70	0:27:43
4C	1e-03	1.157e-05	9	5	9	9	9.642e-07	9.642e-07	1.880e-04	51	51	65	0:23:8
4G	1e-04	7.883e-06	14	11	14	14	4.927e-07	4.927e-07	1.096e-05	84	84	93	0:43:54
4C	1e-04	7.883e-06	14	11	14	14	4.927e-07	4.927e-07	1.586e-05	89	90	76	0:30:9
4G	1e-05	5.371e-06	21	21	21	21	3.357e-07	3.357e-07	4.662e-07	129	129	102	1:39:22
4C	1e-05	5.371e-06	21	21	21	21	2.984e-07	2.984e-07	2.827e-06	138	136	72	0:46:1
8G	1e-02	3.946e-05	2	1	2	2	1.973e-05	1.973e-05	1.581e-03	4	4	51	0:6:23
8C	1e-02	3.946e-05	2	1	2	2	1.973e-05	1.973e-05	6.055e-03	4	3	35	0:3:7
8G	1e-03	3.134e-05	3	2	3	3	1.567e-05	1.567e-05	8.201e-05	6	5	39	0:6:54
8C	1e-03	3.134e-05	3	2	3	3	1.567e-05	1.567e-05	1.320e-04	6	5	31	0:3:40
8G	1e-04	2.490e-05	4	3	4	4	1.245e-05	1.245e-05	2.353e-05	7	6	26	0:7:13
8C	1e-04	2.490e-05	4	3	4	4	1.245e-05	1.245e-05	9.883e-05	7	6	15	0:3:13
8G	1e-05	1.977e-05	5	5	5	5	4.944e-06	4.944e-06	2.191e-06	11	10	29	0:7:5
8C	1e-05	1.977e-05	5	5	5	5	4.944e-06	4.944e-06	6.707e-07	12	11	27	0:7:9
10G	1e-02	5.175e-05	2	0	2	2	2.587e-05	2.587e-05	3.014e-03	3	3	47	0:6:14
10C	1e-02	5.175e-05	2	0	2	2	2.587e-05	2.587e-05	6.760e-04	3	3	36	0:3:16
10G	1e-03	4.271e-05	2	1	2	2	2.136e-05	2.136e-05	7.671e-05	4	3	43	0:5:59
10C	1e-03	4.271e-05	2	1	2	2	2.136e-05	2.136e-05	6.760e-04	4	3	29	0:2:45
10G	1e-04	3.526e-05	3	2	3	3	1.763e-05	1.763e-05	3.527e-05	4	4	34	0:4:57
10C	1e-04	3.526e-05	3	2	3	3	1.763e-05	1.763e-05	5.461e-05	4	4	25	0:3:15
10G	1e-05	2.910e-05	3	3	3	3	1.455e-05	1.455e-05	1.765e-06	6	5	25	0:5:43
10C	1e-05	2.910e-05	3	3	3	3	1.455e-05	1.455e-05	2.700e-06	6	5	16	0:5:41

Table 7.9b

(single subinterval, $\mu = 10^{10}$)

q	TOL	h_p	w_1^o	w_2^o	w_L^o	w_R^o	h_{first}	h_{last}	Err	w_L^c	w_R^c	i	T
4G	1e-02	1.698e-05	6	2	6	6	2.123e-06	2.123e-06	7.724e-03	21	20	154	0:42:41
4C	1e-02	1.698e-05	6	2	6	6	1.101e-06	1.019e-06	8.372e-04	39	41	176	0:49:3
4G	1e-03	1.157e-05	9	5	9	9	1.446e-06	1.446e-06	8.931e-04	48	49	205	1:39:37
4C	1e-03	1.157e-05	9	5	9	9	9.642e-07	9.642e-07	8.561e-04	49	49	198	1:23:8
4G	1e-04	7.883e-06	14	11	14	14	2.885e-07	2.885e-07	6.806e-05	96	96	314	6:33:54
4C	1e-04	7.883e-06	14	11	14	14	2.889e-07	2.889e-07	6.214e-05	109	109	338	6:57:42
4G	1e-05	5.371e-06	21	21	21	21	1.173e-07	1.176e-07	8.365e-07	157	157	361	7:46:12
4C	1e-05	5.371e-06	21	21	21	21	5.844e-07	5.852e-07	3.892e-06	148	149	346	7:8:1
8G	1e-02	3.946e-05	2	1	2	2	1.973e-05	1.973e-05	1.581e-03	4	3	66	0:8:41
8C	1e-02	3.946e-05	2	1	2	2	1.973e-05	1.973e-05	2.435e-03	4	3	54	0:6:3
8G	1e-03	3.134e-05	3	2	3	3	1.567e-05	1.567e-05	8.781e-04	5	4	68	0:9:26
8C	1e-03	3.134e-05	3	2	3	3	1.567e-05	1.567e-05	9.428e-04	5	5	62	0:8:48
8G	1e-04	2.490e-05	4	3	4	4	1.245e-05	1.245e-05	7.565e-05	6	5	67	0:9:1
8C	1e-04	2.490e-05	4	3	4	4	1.245e-05	1.245e-05	3.707e-05	7	6	59	0:6:52
8G	1e-05	1.977e-05	5	5	5	5	9.887e-06	9.887e-06	4.979e-06	9	8	71	0:12:50
8C	1e-05	1.977e-05	5	5	5	5	9.887e-06	9.887e-06	7.642e-06	9	7	58	0:6:37
10G	1e-02	5.175e-05	2	0	2	2	2.587e-05	2.587e-05	3.014e-03	3	3	56	0:7:26
10C	1e-02	5.175e-05	2	0	2	2	2.587e-05	5.175e-05	4.948e-03	3	2	44	0:4:2
10G	1e-03	4.271e-05	2	1	2	2	2.136e-05	2.136e-05	7.671e-05	4	3	58	0:7:56
10C	1e-03	4.271e-05	2	1	2	2	2.136e-05	2.136e-05	1.196e-04	4	3	45	0:4:11
10G	1e-04	3.526e-05	3	2	3	3	1.763e-05	1.763e-05	9.889e-06	4	4	56	0:7:19
10C	1e-04	3.526e-05	3	2	3	3	1.763e-05	3.526e-05	1.513e-05	4	3	47	0:4:46
10G	1e-05	2.910e-05	3	3	3	3	1.455e-05	2.910e-05	4.418e-06	5	3	56	0:7:10
10C	1e-05	2.910e-05	3	3	3	3	1.455e-05	1.455e-05	6.770e-06	5	4	50	0:5:22

As the last illustration, the following *Table 7.10* corresponding to problem-3 displays some results of numerical experiments comparing the *RH* and the de Boor algorithms as well as Chebyshev and Gauss points. From this table, it is again observed that w_L^o and w_R^o are reasonable estimations for number of subintervals in the layer region. This is also indicated by the fact that the width of the first and the last subinterval are reasonably close to h_p. It can again be seen that the *RH* algorithm with Chebyshev points may perform well, moreover in some cases it can be better than those using Gauss points. In contrast, the de Boor algorithm with Chebyshev collocation points never produces better results than using Gauss points.

In comparing performance of the *RH* and de Boor algorithms, the numerical results in this table demonstrate the superiority of the *RH* algorithm over the de Boor algorithm.

Table 7.10

(problem parameter $\mu = 10^4$)

					RH							de Boor					
q	TOL	h_p	w_L^o	w_R^o	h_{first}	h_{last}	w_L^c	w_R^c	w	i	T	h_{first}	h_{last}	w_L^c	w_R^c	w	i	T
3G	1e-02	6.762e-03	5	5	6.762e-03	6.762e-03	5	5	30	16	0:0:15	6.762e-03	6.762e-03	5	5	30	16	0:0:15
3C	1e-02	6.762e-03	5	5	3.381e-03	3.381e-03	6	6	32	18	0:0:18	3.381e-03	3.381e-03	6	6	32	18	0:0:18
3G	1e-03	4.267e-03	11	11	2.133e-03	2.133e-03	13	13	52	20	0:0:52	2.133e-03	2.133e-03	13	13	54	22	0:0:59
3C	1e-03	4.267e-03	11	11	2.133e-03	2.133e-03	14	14	56	24	0:1:8	2.133e-03	2.133e-03	14	14	58	26	0:1:13
3G	1e-04	2.692e-03	24	24	1.346e-03	1.346e-03	27	28	92	21	0:2:48	1.346e-03	1.346e-03	27	28	94	23	0:3:9
3C	1e-04	2.692e-03	24	24	1.346e-03	1.346e-03	30	31	105	34	0:5:14	1.346e-03	1.346e-03	30	31	108	37	0:5:52
3G	1e-05	1.699e-03	47	47	8.493e-04	8.493e-04	56	57	175	35	0:15:38	8.493e-04	8.493e-04	55	56	175	35	0:15:31
3C	1e-05	1.699e-03	47	47	8.493e-04	8.493e-04	60	61	187	47	0:22:25	8.493e-04	8.493e-04	59	60	187	47	0:22:23
4G	1e-02	1.201e-02	2	2	1.201e-02	1.201e-02	3	2	20	15	0:0:7	1.201e-02	1.201e-02	3	2	21	16	0:0:8
4C	1e-02	1.201e-02	2	2	1.201e-02	1.201e-02	3	2	20	15	0:0:7	1.201e-02	1.201e-02	3	2	21	16	0:0:8
4G	1e-03	8.182e-03	5	5	4.091e-03	4.091e-03	6	6	29	15	0:0:16	8.182e-03	8.182e-03	5	5	29	15	0:0:16
4C	1e-03	8.182e-03	5	5	4.091e-03	4.091e-03	6	6	28	14	0:0:14	8.182e-03	4.091e-03	6	6	39	25	0:0:38
4G	1e-04	5.574e-03	11	11	2.787e-03	2.787e-03	11	12	46	14	0:0:38	5.574e-03	5.574e-03	10	11	47	15	0:0:42
4C	1e-04	5.574e-03	11	11	2.787e-03	2.787e-03	12	13	49	17	0:0:49	2.787e-03	2.787e-03	11	13	58	26	0:1:31
4G	1e-05	3.798e-03	21	21	1.899e-03	1.899e-03	22	23	75	13	0:1:35	1.899e-03	1.899e-03	22	23	87	25	0:3:32
4C	1e-05	3.798e-03	21	21	1.899e-03	1.899e-03	23	24	78	16	0:1:57	1.899e-03	1.899e-03	23	24	93	31	0:4:36
5G	1e-02	1.278e-02	2	2	1.278e-02	1.278e-02	2	2	16	11	0:0:5	1.278e-02	1.278e-02	2	2	18	13	0:0:7
5C	1e-02	1.278e-02	2	2	1.278e-02	1.278e-02	2	2	16	11	0:0:5	1.278e-02	1.278e-02	2	2	18	13	0:0:7
5G	1e-03	9.198e-03	5	5	9.198e-03	9.198e-03	5	5	24	10	0:0:10	9.198e-03	9.198e-03	5	5	26	12	0:0:13
5C	1e-03	9.198e-03	5	5	9.198e-03	9.198e-03	5	5	23	9	0:0:9	9.198e-03	9.198e-03	5	5	26	12	0:0:13
5G	1e-04	6.619e-03	9	9	6.619e-03	6.619e-03	8	9	33	7	0:0:14	6.619e-03	6.619e-03	8	9	35	9	0:0:19
5C	1e-04	6.619e-03	9	9	6.619e-03	6.619e-03	8	9	33	7	0:0:14	6.619e-03	6.619e-03	8	9	35	9	0:0:19
5G	1e-05	4.764e-03	17	17	4.764e-03	4.764e-03	16	17	55	5	0:0:27	4.764e-03	4.764e-03	16	17	55	5	0:0:27
5C	1e-05	4.764e-03	17	17	4.764e-03	4.764e-03	16	17	55	5	0:0:26	4.764e-03	4.764e-03	16	17	55	5	0:0:26
8G	1e-02	2.790e-02	1	1	3.501e-02	2.790e-02	1	1	11	9	0:0:3	3.501e-02	2.790e-02	1	1	11	9	0:0:3
8C	1e-02	2.790e-02	1	1	3.501e-02	2.790e-02	1	1	11	9	0:0:3	3.501e-02	2.790e-02	1	1	11	9	0:0:3
8G	1e-03	2.216e-02	2	2	2.216e-02	2.216e-02	2	2	14	9	0:0:5	2.216e-02	2.216e-02	2	2	14	9	0:0:5
8C	1e-03	2.216e-02	2	2	2.216e-02	2.216e-02	2	2	14	9	0:0:5	2.216e-02	2.216e-02	2	2	14	9	0:0:5
8G	1e-04	1.760e-02	3	3	1.760e-02	1.760e-02	2	3	15	7	0:0:5	1.760e-02	1.760e-02	2	3	15	7	0:0:5
8C	1e-04	1.760e-02	3	3	1.760e-02	1.760e-02	2	3	16	8	0:0:6	1.760e-02	1.760e-02	2	3	17	9	0:0:7
8G	1e-05	1.398e-02	5	5	1.398e-02	1.398e-02	4	5	21	7	0:0:9	1.398e-02	1.398e-02	4	5	23	9	0:0:13
8C	1e-05	1.398e-02	5	5	1.398e-02	1.398e-02	4	5	21	7	0:0:9	1.398e-02	1.398e-02	4	5	23	9	0:0:13

chapter 8

Concluding Remarks and Future Improvements

As stated at the beginning of this thesis, we primarily intended to investigate some collocation algorithms, in particular our aim was to develop practical mesh selection algorithms by comparing their performance with those using some well known algorithms. This task certainly needed a lot experimental works which in turn required substantial programming. Nevertheless, more importantly we also needed background theoretical aspects of the methods to be used in developing and implementing such algorithms which this was covered in chapter 2.

In Chapter 3, by utilising the special structure of the collocation matrices we have developed a block matrix with more compact structure. A very significant reduction in the amount of memory needed and number of arithmetic operation performed has been shown, and computational examples demonstrated that a tremendous time saving can be made. An improvement in condition number made by the use of column scaling operations is also presented in chapter 3 . It was observed that without any column scaling operation the condition number for Gauss collocation points are smaller than those using Chebyshev points. Note also that the results show that significant reductions are made in both cases. Moreover, the results indicate that employing column scaling operation may result in not only reducing the condition number but also improving the accuracy of the solution, even though this may occur only in a few problems.

For future extension, it might be interesting to do further development in order to obtain a parallel version of this block matrix structure, such that it can be solved using multi processor machines.

From chapter 4, firstly we note that the interpolation polynomial for the residual is fairly good and its form is also convenient for carrying out the integration needed in developing the error estimates. Secondly, some error estimates have been described and for our purposes it is more convenient to use the cheapest one E^*. Numerical evidence indicates that the error estimate E^* described is effective and does appear to be satisfactory at least later in the process, especially for problems with sufficiently smooth solution. Nevertheless, the results of numerical experiments clearly indicate that the estimate is pretty poor when dealing with problems having severe boundary or interior layers, and it is worse when the approximate solution itself is very poor. This result implies that adaptive mesh selection algorithms utilising this error estimate may lead to inappropriate results in whole process since a poor approximate solution at the initial stage of collocation process is likely. It is hoped to consider further investigation for the estimate E^* in future, possibly by developing some additional corrections when dealing with difficult problems.

In chapter 5, we have discussed in some detail various aspects of the mesh selection strategies including their theories and motivations. For the *RH* algorithm, a special scheme to equidistribute local terms $r_i h_i$ has been developed and it seems that this scheme is fairly simple with low cost since we used the approximate residual developed in chapter 3. It is notable that although the *MR* algorithm often performs very well, unfortunately in some other cases it gives very unsatisfactory results, by putting too many break points in some regions without improving the accuracy of the approximate solution. On the other hand, the widely used de Boor algorithm though better than the *MR* algorithm occasionally gives unsatisfactory results. Unsatisfactory results using de Boor algorithms were also observed by Seleman [48] where he considered $\boldsymbol{Q}$ matrix mesh selection algorithm for solving boundary value single higher order differential equations.

Perhaps the most notable observation about results in chapter 5 is that in our selected examples the *RH* algorithm is more reliable than the de Boor algorithms, indicated by the fact that in some cases the de Boor algorithm performed very unsatisfactorily while the *RH* algorithm worked very well. In most cases using the

RH algorithm gave better results than those using de Boor algorithm. In particular, the results of numerical experiments clearly indicate that in most cases using the mesh placement algorithm with the *RH* criterion function gave the best approximate solution.

In chapter 6, we have derived and evaluated w^*'s the estimates for number of subintervals needed in the next stage of collocation process which enable multiple subdivisions to be applied in the adaptive algorithms. The results of numerical experiments clearly show that using the estimate w^* without any additional restrictions may lead to completely unsatisfactory results, in particular for mesh subdivision strategies. This is not surprising since these estimates make use the error estimate E^* which may perform very poorly in the early stages of collocation process. To cope with this problem some modified algorithms have been introduced, and together with the supporting numerical results of section 6.4 and section 6.6, it is clear that they are sound and valuable. It is also notable that one strategy may be very efficient in, for example, mesh subdivision strategy but it performs unsatisfactorily in mesh placement strategy, and vice versa. The results in chapter 6 provide practical indications about which modified algorithms more suitable for each strategy. Further study and experiments would be useful to obtain more efficient algorithms, possibly by using a more refined statistical approaches.

In the beginning of chapter 7 we have described phenomenon of stiffness arising in some boundary value problems. This phenomenon has connection with the eigenvalues of associated coefficient matrix in the differential equations. We then utilised these eigenvalues to predict the layer locations. This was followed by developing some algorithms to estimate the width of such regions and suitable number of breakpoints in such regions. Finally, a number of numerical experiments were carried out and some improvements, especially in term of computation time, were observed. Moreover the estimates for number of subintervals needed in the layer regions perform satisfactorily in practice. This investigation needs further work, for example on how to determine suitable number of break points in an interior layer, as well as more experiments using a wider selection of problems. Also the possibility

of unequal distribution within the layer, and multiple layers in larger systems should be considered.

Comparing the results using Gauss and Chebyshev points, in most cases the superiority of Gauss points over Chebyshev points in producing higher accuracy solutions was observed. However, the results also indicate that the *RH* algorithms using Chebyshev points gives very satisfactory results and they are comparable with those using de Boor algorithms with Gauss points.

It is important to realise that since the numerical experiments we have carried out here are based on a limited set of test problems, even though they have been chosen carefully to accommodate problems with various natures, some conclusions which have been drawn should not be generalised too far. What we can say is that the results of numerical experiments presented here indicate the relative merits of algorithms. Clearly, a more extensive comparison both on a wider selection of problems and with alternative algorithms would be valuable.

Bibliography

[1] **Ahmed, A.H.** (1981), Collocation Algorithm and Error Analysis for Approximation Solutions of ODEs, Ph.D. Thesis, University of Newcastle upon Tyne

[2] **Ahmed, A.H.** and **Wright, K.** (1985), Further Asymptotic Properties of Collocation Matrix Norm. *IMA Journal of Numerical Analysis* 5, pp 235-246

[3] **Ahmed, A.H.** and **Wright, K.** (1986), Error Estimation for Collocation Solution of Linear Ordinary Differential Equation. *Comp. and Maths with Applications 12B,* pp. 1053-1059

[4] **Aitken, Richard C.**(ed) (1985), *Stiff Computation*, Oxford University Press. Inc., NewYork – Oxford

[5] **Anselone, P.M.** (1971), *Collectively Compact Approximation Theory*, Prentice-Hall. Englewood Cliffs, New Jersey.

[6] **Ascher, U.** (1986), Collocation for Two point Boundary Value Problems Revisited, *SIAM Journal of Numerical Analysis 23,* pp. 596-609

[7] **Ascher, U.** and **Bader, G.** (1986), Stability of Collocation at Gaussian Points, *SIAM Journal of Numerical Analysis* 23, pp. 412-422

[8] **Ascher, U., Pruess, S.** and **Russell, R.D.** (1983), On Spline Basis Selection for Solving Differential Equations, *SIAM Journal of Numerical Analysis* 20, pp.121-142

[9] **Ascher, U.** and **Weiss, R.** (1983), Collocation for Singular Perturbation Problem I : First Order System with Constant Coefficients, *SIAM Journal of Numerical Analysis* 20, pp. 537-557

[10] **Ascher, U.** and **Mattheij, R.M.M** and **Russell, R.D.** (1988), *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs, New Jersey

[11] **Ascher, U.** and **Russell, R.D.**(ed) (1985), *Numerical Boundary Value ODEs*, Birkhauser, Boston - Basel – Stuttgart

[12] **Aziz, A.K.** (ed) (1975), *Numerical Solutions of Boundary Value problems for Ordinary Differential Equations*, Academic Press, Inc. New York – San Francisco – London

[13] **Butcher, J.C.** (1987), *The Numerical Analysis of Ordinary Differential Equations*, John Wiley and Sons, Chichester - New York - Brisbane - Toronto – Singapore

[14] **Carey, G.E.** and **Humphrey, D.L.** (1979), Finite element Mesh Refinement Algorithm using Element Residuals., in *Code For Boundary Value Problems in Ordinary Differential Equations, ed. B. Childs et al* , Springer-Verlag, New York

[15] **Clenshaw, C.W.** and **Norton, J.J.** (1963), The Solution of Linear Differential Equations Chebyshev Series, *Computer Journal* 6, pp. 88-92

[16] **Collatz, L.** (1966), *The numerical of Differential Equations*, Springer-Verlag, Berlin-Heidelberg – New York

[17] **Cruickshank, D.M.** (1974), Error Analysis of Collocation Methods for the Numerical Solution of ODEs. Ph.D. Thesis, University of Newcastle upon Tyne

[18] **Cruickshank, D.M.** and **Wright, K.** (1978), Computable error bounds for polynomial collocation methods, *SIAM Journal of Numerical Analysis* 15, pp.134-151

[19] **Daniel, J.W.** and **Martin, A.J.** (1977), Numerov's Method with Differed Corrections for two-point Boundary Value Problems, *SIAM Journal of Numerical Analysis* 14, pp. 1046-1062

[20] **Davis, P.J.** (1963) *Interpolation and Approximation*, Blaisdell

[21] **Davis, P.** and **Rabinowitz, P.** (1967) *Numerical Integration*, Blaisdell

[22] **de Boor, C.** (1973), Good Approximation by Splines with Variable Knots II, in *Lecture Notes in mathematics*, vol 363, Springer-verlag, Berlin. pp. 12-20

[23] **de Boor, C.** and **Swartz, B.** (1973), Collocation at Gaussian Points, *SIAM Journal of Numerical Analysis* 10, pp. 582-606

[24] **Dodson, D.S.** (1972), Optimal Order Approximation by Polynomial Spline Functions, PhD Thesis, Purdue University. Lafayette.

[25] **Fox, L.** and **Parker,** (1966), *Chebyshev Polynomials in Numerical Analysis*. Oxford Mathematical Handbook, London

[26] **Gerrard, C.** (1979), Computable Error Bounds for Approximate Solution of Ordinary Differential Equations. PhD Thesis, University of Newcastle upon Tyne

[27] **Gerrard, D.** and **Wright, K.** (1984), Asymptotic Properties of Collocation Matrix Norms 2 : Piecewise Polynomial Approximation, *IMA Journal of Numerical Analysis* 4. pp 185-202.

[28] **Golub, G** and **Van Loan, C.** (1983), *Matrix Computation*, John Hopkins Press, Baltimore

[29] **Hemker, P.W.** and **Miller, J.J.H.** (ed) (1979), *Numerical Analysis of Singular Perturbation Problems*, Academic Press, London

[30] **Hairer, E.** and **Norsett, S.P.** and **Wanner, G.** (1991), *Solving Ordinary Differential Equations I*, Springer-Verlag, Berlin – Heidenberg - New York – London – Tokyo

[31] **Hairer, E.** and **Wanner, G.** (1991), *Solving Ordinary Differential Equations II,* Springer-Verlag, Berlin-Heidenberg-New York- Tokyo-Budapest

[32] **Kantorovich, L.V. and Akilov, G.P.** (1964), *Functional Analysis in Normed Space*, Pergamon, New York

[33] **Keller, H.B.** (1968), *Numerical Methods For Two-Point Boundary Value Problems*, Blaisdell Publishing Company, Waltham-Massachusetts-Toronto-London

[34] **Kreiss, B.** and **Kreiss, H.O.** (1981), Numerical Methods For Singular Perturbation Problems, *SIAM Journal of Numerical Analysis* 18, pp. 262-276

[35] **Kreiss, H.O.** and **Nichols, N.K,** and **Brown, D.L.** (1986), Numerical Methods For Stiff Two point Boundary Value Problems, *SIAM Journal of Numerical Analysis* 23, pp. 325-368

[36] **Lambert, J.D.** (1991), *Numerical Methods For Ordinary Differential Systems*, John Wiley and Sons, Chichester, New York, Brisbane, Singapore

[37] **Lentini, M.** and **Pereyra, V.** (1977), An Adaptive Finite Difference Solver for nonlinear two-point Boundary Value Problems with Mild Boundary Layers, *Numerische Mathematik* 14, pp. 91-111

[38] **Moore, R.E.** (1985), *Computational Functional Analysis*, Ellis Horwood LTD., John Wiley and Sons, Chichester, New York, Ontario, Brisbane

[39] **National Physical Lab.** (1962), *Modern Computing Methods*, Notes on Applied Science 16, Her Majesty's Stationery Office, London

[40] **Pereyra, V.** and **Sewell, G.E.** (1975), Mesh Selection for Discrete Solution of Boundary Value Problems in DEs, *Numerische Mathematik* 23, pp. 261-268

[41] **Philips, J.L.** (1972), Collocation as a Projection Method for Solving Integral and other Operator Equation, *SIAM Journal of Numerical Analysis.* 9, pp. 14

[42] **Rheinboldt, W.C.** (1980), On a Theory of Mesh-refinement Processes, *SIAM Journal of Numerical Analysis* 17, pp. 766-778

[43] **Roberts, S.M.** and **Shipman, J.S.** (1972), *Two-Point Boundary Value Problems: Shooting Methods*, American Elseivier Publishing Company, New York

[44] **Russell, R.D.** (1974), Collocation for Systems of Boundary Value Problems, *Numerische Mathematik* 23, pp. 119-133

[45] **Russell, R.D.** and **Christiansen, J.** (1978), Adaptive Mesh Selection Strategies for Solving BVPs, *SIAM Journal of Numerical Analysis* 15, pp. 59-80

[46] **Russell, R.D.** and **Shampine, L.F.** (1972), A Collocation Method for Boundary Value Problems, *Numerische Mathematik* 19, pp. 1-28

[47] **Russell, R.D.** and **Shampine, L.F.** (1975), Numerical Methods for Singular Boundary Value Problems, *SIAM Journal of Numerical Analysis 12,* pp. 13-36

[48] **Seleman, A.H.** (1984), An Investigation of Mesh Selection Algorithms in the Numerical Solution of BVPs by Piecewise Polynomial Collocation. PhD Thesis, University of Newcastle upon Tyne

[49] **Stetter, H. J.** (1979), The Defect Correction Principle and Discretization Methods, *Numeriche Mathematik,* 29, pp. 425-433

[50] **Swartz, B.** (1988), Conditioning Collocation, *SIAM Journal of Numerical Analysis* 25, pp. 124-147

[51] **Watkins, D.S.** (1991), *Fundamentals of Matrix Computations*, John Wiley and Sons, Chichester- New York-Brisbane-Toronto-Singapore

[52] **Weinmuller, E.** (1986), Collocation for Singular Boundary Value Problems of Second Order, *SIAM Journal of Numerical Analysis* 23, pp. 1062-1095

[53] **White, A.B.** (1979), On the Selection of Equidistributing Meshes for Two-Point Boundary Value Problem, *SIAM Journal of Numerical Analysis* 16, pp. 472-502

[54] **Wright, K.** (1964), Chebyshev Collocation Methods for Ordinary Differential Equations, *Computer Journal* 6, pp. 358-365

[55] **Wright, K.** (1970), Some Relationships between Implicit Runge Kutta, Collocation, Lanczos *tau* Methods and Their Stability Properties, *BIT* 20. pp. 217-227

[56] **Wright, K.** (1984), Asymptotic Properties of Collocation Matrix Norms 1 : Global Polynomial Approximation, *IMA Journal of Numerical Analysis* 4, pp 185-202.

[57] **Wright, K.** (1992), *A Review of some Developments in Collocation Algorithms*, in Computational ODEs, Clarendon Press, Oxford, pp. 215-223

[58] **Wright, K.** (1993), Parallel Treatment of Block –bidiagonal Matrices in the Solution of OD BVPs, *Journal of Comp and Applied Maths* 45, pp 191-200

[59] **Wright, K.** (1995), *Recent Developments in Collocation Methods for Ordinary Differential Equations*, proceeding Fifth International Colloquium on Differential Equations, The Netherlands. pp. 353-362

[60] **Wright, K., Ahmed, A.H.** and **Seleman, A.H.** (1991), Mesh Selection in Collocation for Boundary Value Problems, *IMA Journal of Numerical Analysis* 11, pp. 7-20.