An Investigation Of Mesh Selection Algorithms In The Numerical Solution Of Boundary Value Problems By Piecewise Polynomial Collocation Method

Adeeb H. Seleman

Computing Laboratory
August 1984

University of Newcastle upon Tyne
Newcastle upon Tyne NE1 7RU

ACKNOWLEDGEMENTS

I should like to thank my supervisor, Dr. Kenneth Wright, whose help, advice and enthusiasm throughout was very much appreciated.

ABSTRACT

This thesis is concerned with the investigation and evaluation of adaptive mesh selection strategies for solving two points boundary value problems using a piecewise collocation method.

General definitions and descriptions for adaptive strategies and piecewise collocation methods are given at the beginning. A description of a data structure which is suitable for implementing adaptive algorithms is also given.

A preliminary investigation of four adaptive strategies is introduced and evaluated on a set of test problems. From this evaluation it is found that a strategy called the Q-matrix has done generally well, but its cost is high compared with the rest.

An improvement to one of the cheap strategies is introduced by improving the initial mesh from which the cheap strategy starts. An algorithm is designed to build a good initial mesh which is fairly dense in the layer regions. This algorithm is based on the behaviour of the asymptotic solution of the problem. A definition and a short description for such solution are also introduced.

A further improvement to the Q-matrix strategy, is then introduced. In this, we used an error estimate instead of the error bound used originally in the strategy. This estimate is obtained by multiplying two polynomials, one representing the residual and the other representing the kernel (using the elements of Q). The effect of having a singular point in the middle of an interval on these representations is also investigated.

A final evaluation of three strategies, the Q-matrix and the two new strategies is introduced. This evaluation shows the improvement in the new modified strategies in terms of cost and accuracy.

The thesis concludes with comment on the strategies and some suggestions for further research and improvement.

CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 GENERAL BACKGROUND

When Boundary Value Ordinary Differential Equation (B.V.O.D.E.) problems are solved by the **Piecewise polynomial Collocation** method, the range over which the solution is defined is divided into a number of intervals and a piecewise polynomial is fitted in each interval. The distribution of these intervals affects the accuracy of the approximation. In this work, we will investigate algorithms for choosing this distribution, what may be called mesh distribution algorithms; introduce some new algorithms, finally compare and evaluate these algorithms.

The work is concerned with single mth order Linear B.V.O.D.E problems. The general form for these problems is as follows :

$$\mathcal{L}y = y^{(m)} + \lambda \sum_{j=0}^{m-1} p_j(x)\, y^{(j)} = \eta(x) \qquad (1.1a);$$

Over(a,b), with the following boundary conditions :

$$(L_i y)(a) = A_i,\ i = 1, \ldots \tau;$$

$$(L_i y)(b) = B_i,\ i = \tau+1, \ldots\ldots m; \qquad (1.1b);$$

Here λ is a parameter which is convenient to introduce so that by changing its value the problem's stiffness could be changed, a, b are two finite boundary points, $p_j(x)$ and $\eta(x)$ are some continuous

functions.

This equation can be represented as an operator equation by defining the the differential operator Dm as follows

$$Dm = d^m/dx^m;$$

and the operator T as

$$T := -\lambda \sum_{j=0}^{m-1} p_j(x)\, d^j/dx^j;$$

Now, equation 1.1 may be written in the form

$$(Dm - T)\, y = \eta \; ; \qquad (1.2) \; ;$$

We assume that the operator (Dm - T) is invertible under the given boundary conditions.

Equation 1.1 can be put into another operator equation form by using the integral operator K defined by $K = T\, Dm^{-1}$. This can done by putting u = Dmy, then we get

$$(I - K)\, u = \eta \qquad (1.3);$$

where I is the identity operator.

The operator K may be written as follows:

$$(Ku)(x) = \int_{-1}^{1} k(x,t)\, u(t)\, dt$$

where k(x,t) is the kernel of K given by:

$$k(x,t) = -\{ \sum_{j=0}^{m-1} p(x)\, d^j/dx^j\, g(x,t) \}$$

where g(x,t) is the Green's function for the operator Dm^{-1}, with the

boundary conditions of 1.1(b). The transformation of the B.V.O.D.E. problem into the form of 1.3 is equivalent to transforming the differential problem into an integral equation. This transformation makes theory for operator equation, such as those of Kantorovich and Akilov[1964], Collatz [1966] and Anselone[1971], applicable.

All the mesh selection algorithms to be introduced in this work are of an adaptive nature. They are designed to be applied for linear B.V.O.D.E problems. But when non-linear problems are solved by linearizing them, these algorithms become applicable for non-linear problems. However, if a non-linear problem is solved directly, a more complicated algorithm may be required, as we may need a more complicated method for evaluating the criterion. An example for nonlinear algorithm can found in Humphrey and Carey[1978]. All the numerical results in this thesis are for second order B.V.O.D.E. problems, whose general form is obtained from 1.1a, 1.1b with m = 2.

1.2 PIECEWISE COLLOCATION METHOD

The collocation method is an old method, its history could return to 1936, when Kantorovich considered the use of polynomials in approximating the solution of different functions. Later, Kantorovich and Akilov[1964], Collatz[1964], Phillips[1972] and Coldrick[1972], all dealt with the method. Then, De Boor and Swartz[1973] considered collocation with Spline functions using

Legendre polynomials as collocation points. Cruickshank and Wright[1978] worked on error analysis for the method and introduced some error bounds. All these references, except the De Boor and Swartz one, have dealt with global collocation, which uses one polynomial to represent the solution over the whole range. The work in this thesis is for piecewise collocation which is discussed below. A comparison between these two forms can be found in Ahmed[1981].

The collocation method is a member of the family of the weighted residual methods, a study of these methods could be found in Finlayson and Scriven[1966], other members are the Galerkin method and the Least squares method. A comparison between these three methods is given in Russell and Varah[1974].

All the references we have mentioned so far have dealt with the theoretical part of the method, for the practical part, Lanczos[1938], considered the use of Chebyshev polynomials for approximation methods. Wright[1964], also worked with collocation with Chebyshev polynomials. Shampine[1969], consider the collocation with piecewise polynomial functions. Villadsen and Stewart[1967], considered different forms of collocation and compared them. Russell and Shampine[1972], studied the collocation with piecewise polynomials for linear and non-linear problems, this study is an extension to that of Shampine[1969]. Ascher et al.[1978], implemented the method in a code called the 'COLSYS' code. Finally Ahmed[1981], studied different algorithms for estimating and bounding

errors in collocation method.

In the piecewise collocation method, the range [a,b] over which the solution is defined is divided into a number of intervals, not necessarily of equal size. The intervals are refered to by their end points. Assume that the end points are $\{ t_i \}$ where we have

$$a = t_0 < t_1 \ldots\ldots < t_P = b \ ;$$

and P is the number of intervals.

In each interval (t_i , t_{i+1}), the solution is approximated by a linear combination of piecewise polynomial functions, from now on, we will refer to them as 'REPRESENTATION FUNCTIONS', which could be

1. Non zero in a single interval only.

2. Non zero in more than one interval.

If we assume that all the conditions required by the collocation method are homogeneous then, the set of piecewise polynomials that satisfy these conditions are, generally, refered to as 'BASIS FUNCTIONS', as they form a basis for the subspace in which the solution must lie. The REPRESENTATION FUNCTIONS do not necessarily form a basis for this space since the extra conditions may also be required to be satisfied. Clearly, however, with these conditions they also specify the subspace.

The coefficients for REPRESENTATION FUNCTIONS, when they are non zero in a single interval, may be obtained by solving a linear algebra problem arising from requiring the combination to:

1. Satisfy the differential equation at a certain set of points called 'COLLOCATION POINTS'. For simplicity, the number and the relative distribution of these points is taken to be the same for all intervals. Algebraically, if we put the estimated solution as :

$$y_{np}(t) = \sum_{j=0}^{n+m-1} a_j B_{ij}(t) \quad \text{on}(t_i, t_{i+1}) \qquad (1.4);$$

Then, the collocation condition will be :

$$\sum_{j=0}^{n+m-1} a_j \mathcal{L}(B_{ij}(x_k)) = \eta(x_k), \; k = 1....n; \qquad (1.5);$$

Where n is the number of collocation points, m is the order of the B.V.O.D.E. problem, $B_{ij}(t)$ is a set REPRESENTATION FUNCTIONS, y_n is the approximated solution, $\{x_i\}$ are the collocation points, $\{a_j\}$ are unknown coefficients and $\mathcal{L}$ is the differential operator.

2. Satisfy the boundary conditions of the problem, equation 1.1(b). This can be put as follows :

$$\sum_{j=0}^{n+m-1} a_j \; L_s B_{1j}(t_0) = A_s \;, \; s = 1.....\tau;$$

$$\sum_{j=0}^{n+m-1} a_j \; L_s B_{Pj}(t_p) = B_s \quad , \; s = \tau+1, \ldots, m \; ;$$

To ensure continuity of the estimated solution over the whole of [a,b], this solution should satisfy a set of join condition at the mesh points. Usually the degree of continuity is taken to be equal to m-1. Thus, REPRESENTATION FUNCTIONS used to approximate the solution should be at least of degree m-1. These conditions can be represented as follows :

$$\sum_{j=0}^{n+m-1} a_j \; B_{ij}^{(\nu)}(t_k - 0) = \sum_{j=0}^{n+m-1} a_j \; B_{ij}^{(\nu)}(t_k + 0) \quad \nu = 0, \ldots, m-1 \; k = 1, \ldots, P;$$

A variety of piecewise polynomial functions have been used in the literature. B-splines were used by De Boor[1972]. Hermite polynomials were used by De Boor and Swartz[1977]. Recently a new form of functions called 'MONOMIALS BASIS' were introduced by Ascher et al.[1983]. The B-splines are non zero in more than one interval, while, the other two are non zero in a single interval.

In our work we use a sub-set of shifted Chebyshev polynomials, which are of order n and do not automatically satisfy the join conditions, as our REPRESENTATION FUNCTIONS.

As there are a variety of piecewise polynomial functions, there are also a variety of collocation Points; Gauss points, Lobatto points, and Chebyshev zeros have all been suggested as sensible choices. In our work, we use the zeros of Chebyshev polynomials as

collocation points. These points are defined in the (-1,1) range as follows :

$$x_i = \cos((2*i-1)\pi/(2*n)); \; i = 1,\ldots\ldots,n.$$

The piecewise collocation method may be regarded as a projection method using an interpolation projection based on the collocation points. We assume that the interpolatory projection ϕ transforms a function to its piecewise interpolating polynomial. The operator equation of 1..2, will be transformed into

$$\phi\,(Dm - T)\,y_{np} = \phi\eta \qquad (1.5);$$

where y_{np} is the approximated solution, $y_{np} \in Xn \subset X = Dm^{-1}\,Y$ and $\phi\eta \in Ynp$. Where Y is a normed linear space and $Ynp = \phi Y$. Y may be taken as $R[a,b]$, the space of Riemann integrable functions to allow for discontinuities in the mth derivative of the solution at the break points, as in Wright[1984].

The norm used throughout our work is the infinity norm which is defined, for a a function V(x), as:

$$\|V\| = \max|V(x)| \; ;$$

Equation 1.5 is actually a projection equation, a study of the collocation method as a projection method can be found in Kantorovich and Akilov[1964].

1.3 AIM

To write a code for a piecewise collocation method, we require to specify the following :

1. The representation of the approximate solution (this depends on piecewise polynomial functions used in the approximation).

2. Choice of Collocation and mesh points.

3. Linear algebra solver.

4. Error estimation process (if any is required).

A number of papers have dealt with some of these requirements. For example the error estimation for the collocation methods has been dealt with by Gladwell[1972], Ito[1976], Cruickshank and Wright[1978], Ahmed[1981] and Kedem[1981]. Different basis functions have been also discussed in the literature, references have been given in the previous section.

Mesh selection algorithms have received less attention, although they form an important issue, references for works in this field are given in section 1.5. In our work, we will investigate a number ADAPTIVE mesh selection algorithms, where ADAPTIVE is used in the

sense defined in the next section. Every algorithm is based on a different strategy, we will also investigate these strategies, their motivations and theoretical backgrounds. Some new algorithms will be introduced, and a comparison between the algorithms will be carried out. At the end an evaluation of the algorithms will be made, depending on the accuracy of the solution and the time spent to achieve this accuracy.

1.4 STRUCTURE OF THE ADAPTIVE MESH SELECTION (AMS) ALGORITHMS

Historically, the adaptive mesh selection (AMS) algorithms were first used in quadrature integration methods. One of the early references is Davis and Rabinowitz[1967]. In this reference a definition for the adaptive algorithms were given, the definition is:

Definition 1.1 :-

When the points of subdivision of the integral are chosen according to some strategy depending on the behaviour of the integrand, the subdivision is said to be ADAPTIVE.

Definition 1.2 :-

A fixed choice of subdivision points is NON ADAPTIVE.

Definition 1.3 :-

An algorithm which incorporates definition 1 is an AMS algorithm.

An example for an AMS algorithm used in numerical integration can be found in Cranley and Patterson[1971].

For B.V.O.D.E. problems, definition 1.1 should be altered to be, 'When the points of subdivision needed to obtain the solution of B.V.O.D.E. problem are chosen according to some strategy dependant on the behaviour of the estimated solution, then, the subdivision is said to be ADAPTIVE.'. However, definitions 1.2 and 1.3 hold without any alterations. Recently, Rheinboldt[1981] studied adaptive algorithms, from engineering point of view, as control problems and gave a new definition for them.

In general, AMS algorithms have the following steps :

1. Start from an initial mesh (not necessarily uniform).

2. Evaluate the criterion, defined below, in each interval.

3. According to the criterion values found in 2, decide where to add the new mesh point(s), i.e which interval(s) are to be divided. Usually, the interval which gives largest criterion value is the selected one.

Step 2 and 3, will be repeated until the largest criterion value becomes less than a prescribed tolerance, or, sometimes, a limitation on the number of intervals allowed cause the algorithm to terminate early, this usually happens when the method fails to coverage. Steps 1 and 3 are common for all different AMS algorithms, step 2 is different for different algorithms. This step depends on the criterion used, a criterion is defined as a value that could be used

to indicate, in each interval, how accurate the approximation is, examples of this are the residual or an error estimate for each interval. The amount of work required in step 2 is also different for different algorithms, while the other two steps, always, require the same amount of work. We should mention here, though we do not consider them any further, that there are some AMS algorithms which produce a complete new mesh at each iteration, or others which join up intervals as well as dividing them.

The efficiency of the AMS algorithms depends on how well the criterion reflects the behaviour of the solution. A good criterion may be expensive to evaluate, so using such criteria for simple problems may be a waste of effort, and therefore, it should only be used for problems where it is really necessary.

A good AMS algorithm is a one which puts more points in the regions where the solution behaves badly, such as a boundary layer region, producing a more rapid convergance of the approximated solution.

For simplicity and to compare the criteria more clearly, all the algorithms to be introduced here divide one interval per iteration. However, they could be easily modified to divide more than one interval per iteration, and a variety of strategies are available for this. Examples of strategies for dividing more than one interval are:

1. Divide intervals with the largest and next to the largest criterion value.

2. Divide the interval with largest criterion more than once.

3. Divide all intervals with criterion values greater than C times the maximum criterion value, where C is a constant ($C < 1$).

A number of AMS algorithms produce meshes which are 'EQUIDISTRIBUTED' with respect to a certain function, such meshes are defined as follows:

Definition 1.4 :-

A mesh is equidistributed with respect to a function F, if

$$h_i \| F_i \| = \text{constant} \qquad i = 1 \ldots P \ .$$

where $\| F_i \|$ is the norm of F in the ith interval.

1.5 RELATED WORK

Until recent years, mainly uniform meshes have been used in obtaining solutions of B.V.O.D.E. problems by Collocation methods. Non-uniform meshes were used in finite difference methods before being used for Piecewise Collocation methods. An early work on non-uniform meshes for B.V.O.D.E. problems was done by Brown[1962]. Then, Pearson[1968], introduced an adaptive algorithm which uses the difference between the estimated solutions at two consecutive points

as criterion for the subdivision strategy. The algorithm inserts a mesh point between the two points (t_i, t_{i+1}), which give the largest difference. Derivas[1971], and Roberts[1971], introduced an algorithm depends on the idea of transforming the independant variable to another variable. The transformed problem should have no layers, and a mesh equidistributed ,with respect to the new variable, is sufficient. Pereyra and Sewell[1975], used an estimation of the truncation error as criterion for their AMS algorithm. Lentini and Pereyra[1975], have also designed an AMS algorithm for Variable Order Methods. All the previously mentioned algorithms were mainly concerned with with finite difference methods.

For Piecewise Collocation Methods, the following are some of the references. One of the earliest work was Dodson[1972]. De Boor[1973], introduced an AMS algorithm which changes the distributions of all the mesh points in each iteration. The criterion used in this algorithm was an estimation of the local error by a formula introduced in that paper. Burchar [1974], studied the use of a spline basis, with non-uniform knot distribution, in the collocation method. In the COLSYS code, designed by Ascher etal[1978], which implements the piecewise collocation method, an AMS algorithm was used. In this algorithm, intervals which give an error estimation larger than a certain tolerance are halved at each iteration. Russell and Christiansen[1978], compared different mesh selection algorithms and classified them according to the criterion used in each algorithm. Humphrey and Carey[1978], introduced an AMS

algorithm, which uses the RESIDUAL as a criterion. The algorithm could be used for both linear and non-linear problems.
Russell[1979], published a survey for mesh selection methods, for both finite difference and finite element methods. A comparison between different mesh selection algorithms used in the three codes, COLSYS, NONREF and PASVA3, was discussed by White[1979a]. In another work , White[1979b], studied the equidistribution of mesh points with respect to a transformed independant variable. A Monitor function , a function which reflects the behaviour of solution, is used in obtaining the transformed problem. A selection of such functions was given in the same reference. Kreiss and Kreiss[1981], considered the use of AMS algorithms to solve a system of singularly perturbed B.V.O.D.E. problems. They used the divided difference of the estimated solution to get an approximation of local errors, which are used as criterion for the algorithm. Ahmed[1981], compared some AMS algorithms which are based on some error estimates given in his thesis. One of the strategies given there was the Q-matrix strategy. This strategy will be one of the major strategies to be focused on throughout our work. One of the latest AMS algorithms is an algorithm introduced by Tewarson and Hulsak[1983]. It is suitable to use for the Variable Order methods of Lentini and Pereyra [1974], the criterion used is obtained by the interpolation technique.

1.6 SUMMARY

In the next chapter, we will tackle the problems which arise from implementing the Piecewise Collocation method and AMS algorithms in PASCAL. A suitable data structure will also be introduced and discussed. A special linear algebra solver which is designed to suit the special block structure of the Collocation matrix will be given.

In chapter 3, four AMS strategies will be discussed. Their theoretical background, motivations and algorithms will be given. The algorithms will be tested on a set of B.V.O.D.E problems, and a comparison is carried out between the four algorithms. Based on the comparison a decision on which strategy is the best among the four is made.

In chapter 4, an algorithm for locating the badly behaved regions of the solution depending on the characteristics of problem coefficients is introduced. An algorithm for estimating the width of these regions is designed depending on the WKB solution. By these two algorithms we would be able to get an initial mesh which reflects the behaviour of the solution. Then one of the AMS algorithms of chapter 3, is used to continue the mesh selection process.

In chapter 5, an approximation to the kernel of the operator $(D2 - T)^{-1}$ by a polynomial interpolating the values of the Q-matrix is obtained. A representation of the residual by another polynomial is

given. Then, an estimate of the error is obtained using these to form the basis of a new mesh selection algorithm. This algorithm is considered as a modification to the Q-matrix algorithm of Ahmed[1981].

In chapter 6, a final comparison between the best algorithm of chapter 3, the algorithm of chapter 4 and the algorithm of chapter 5, is introduced. An evaluation of these algorithms is also introduced based on the results of the comparison.

CHAPTER TWO

DATA STRUCTURE AND PROGRAM STRUCTURE

2.1 INTRODUCTION

In this chapter, we describe the implementation of piecewise collocation methods using the PASCAL language. A suitable data structure is used to represent the collocation matrix and the Q-matrix. The main feature required of this structure is that it should be DYNAMIC. This is because when the AMS algorithm is used, the number of intervals changes (usually increases) as the process of estimating the solution goes on. Such a feature makes the use of the ARRAY structure of PASCAL inconvenient, because it has a STATIC structure. A linked list seems a suitable structure. Such A list can be implemented in PASCAL by using RECORD and POINTER types, as POINTER types provide a facility for dynamic storage allocation. The use of pointers and records, however, while facilitating the adaptation aspect of the algorithm make the programming of the remainder a bit more difficult than for an array. A description of the structure of the collocation matrix and how to represent it are given in section 2.2 and 2.3 respectively. In section 2.4, we comment on the special requirements needed to implement the LU decompostion method in PASCAL by using records and pointers. In section 2.4, a description of the Q-matrix is given together with its representation by records and pointers. General comments on the

UBC/PASCAL program used in performing the work in this thesis are also given.

2.2 THE STRUCTURE OF THE COLLOCATION MATRIX

Fig. 2.1, shows the matrix structure which arises when the conditions required by the piecewise collocation method are satisfied. The REPRESENTATION FUNCTIONS used in this case is the set of shifted Chebyshev polynomials satisfying the conditions given in section 1.2. As we see from this figure, the matrix has a block structure. The shape of these blocks might be different if different REPRESENTATION FUNCTIONS were used, Russell and Varah[1975], showed the form of the blocks for the B-splines and the Hermite piecewise polynomials. The matrix of fig 2.1 is refered to, by Fourer[1984], as a REDUCED STAIRCASE MATRIX.

To make the matrix representation more convenient, we regard the matrix as consisting of a number of rectangular blocks, each block is related to an interval. As a result of this, the matrix will have the following properties:

1. The number of blocks is equal to the number of intervals.

2. Each internal block has three different regions, which are constructed independently, these regions are :

a. Two regions denoted by J.C., referring to the joint condition equations. These equations are required to ensure the continuity of the estimated solution at the break points. The region at the top ensures a continuity with the previous interval, while the one at the bottom ensures continuity with the next interval. The number of equations in each region depends on the degree of continuity required. Usually it is equal to m-1, where m is the order of the B.V.O.D.E. problem.

b. A region denoted by C, contains the equations which arise from satisfying the collocation conditions. The number of equation in region C is equal to the number of collocation points, which is related to the degree of approximation.

3. The first and last blocks have regions denoted by B.L. and B.R.. The B.L. region (in the 1st. block) contains the equations arising from satisfying the left boundary conditions. The B.R.(in the last block) contains the equations which arise from satisfying the right boundary conditions. It should be mentioned here that only separated boundary conditions are to be considered. These two blocks have also a J.C. region and a C region as in 2.

From these characteristics, we would have NQ1 rows in the 1st. block, where

NQ1 = NC+NJ+NL;

Where NC, is the number of rows in region C, NJ is the number of

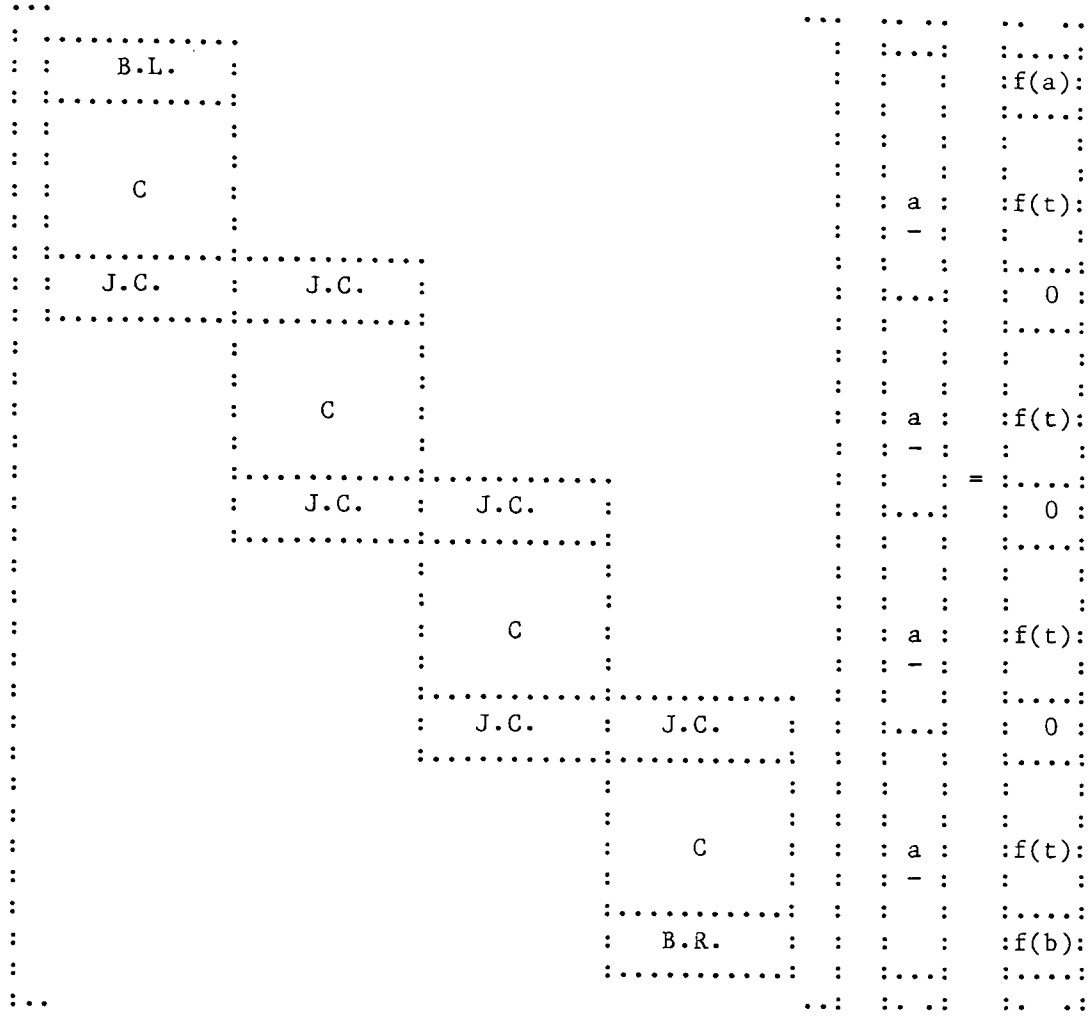


Figure 2.1

Collocation matrix using Chebyshev series

as Representation Functions, for P = 4

rows in region J.C. and NL is the number of rows in region B.L..

In the last block we have NQ2 rows, where

NQ2 = NC+NJ+NR;

Where NR is the number of rows in region B.R.. Finally, for any internal block, we have NQ3 rows, where

NQ3 = NC+(2*NJ);

Hence, for P intervals, the total number of rows in the matrix is TNQ, where

TNQ = P * (NC+NJ) ;

where it has been assumed that NJ = NR+NL. With this assumption, each interior block can be regarded as a square block plus NL rows at top and NR at the bottom of the square. The 1st and last block are also regarded as squares with NR rows added to bottom of the 1st and NL rows added to the top of the last. As a result of this, we will get a square collocation matrix.

4. The unknown vector $\underline{a}$ is also divided into blocks with TU unknowns, where

TU = NC + NJ ;

5. The right hand side is represented by a set of vectors, a vector for each block. The elements of the vector are stored into the corresponding record. The vectors have the following values.

a. At the top of the first vector, there are a set of NL values, these are the right hand side values of the left boundary conditions. At the bottom there are NR zeros corresponds to NR join condition rows of the 1st. block. A set of NC values lie in between, these values are obtained from evaluating the right hand side part of the B.V.O.D.E. problem at the collocation points.

b. At the top of the last vector, there is a set of NL zeros which corresponds to NL join condition rows of the last block. At the bottom, there is a set of NL values, they are right hand side values of the right boundary conditions. As in the lst. block, there is a set of NC values evaluated as in a.

c. At the top of any interior vector, there is a set of NL zeros corresponding to NL rows of the corresponding block. Similarly, at the bottom, there is a set of NR zeros. In between those two sets, there is a set of NC values as in the lst. and last blocks.

Thus for P intervals, we will have P vectors, with the total number of elements in the vectors is TRH, where

TRH = P * (NC + NJ) ;

It is clear that the number of equations in a block will increase as the number of collocation points is increased. If different REPRESENTATION FUNCTIONS are used, the shape of the matrix might change, for example, if B-spline basis are used, the J.C. region will disappear, as the continuity conditions are satisfied implicitly. Hence, fewer unknowns are needed, however, the collocation equations then involve more than one interval. An implementation for this case is given in the COLSYS code.

2.3 DATA STRUCTURE FOR REPRESENTING THE COLLOCATION MATRIX

To represent the collocation matrix, fig 2.1, it is convenient to use a 3-dimensional data structure. One dimension refers to a block, while the other two refer to rows and columns of a block. A PASCAL 3-D array, would be inconvenient, as we showed earlier, when adaptive algorithms are used. In addition to that, when a new block is added (an interval is divided), we must shift all array elements which correspond to blocks after the new block. These two difficulties do not arise if we use Pointer and Record types. As the number of records is not fixed at the compilation stage as in arrays, the addition of any number of new blocks could be done without any overflow in the allocated space or explicitly shifting of any record. Note that, fixed sized two dimensional arrays are used to store the blocks, such a fixed block size is a relatively minor problem.

With the use of Pointers and Records, the collocation matrix is represented as follows :

1. A Pointer is used to refer to a record containing information associated with a block.

2. These records have the following fields

   a. Two variables represent the end points of an interval.

b. A two dimensional array to store the matrix elements of a block. The size of this array is (NC + 2NJ) x (NC + NJ) to allow for the original equations. However, extra space is needed to allow for possible row interchange, and an extra NC+NL rows are allowed for that.

c. A one dimensional array to store the right hand side values corresponding to an interval (block). The size of this array is NC+NJ. This array is also used to store the solution coefficients in each interval.

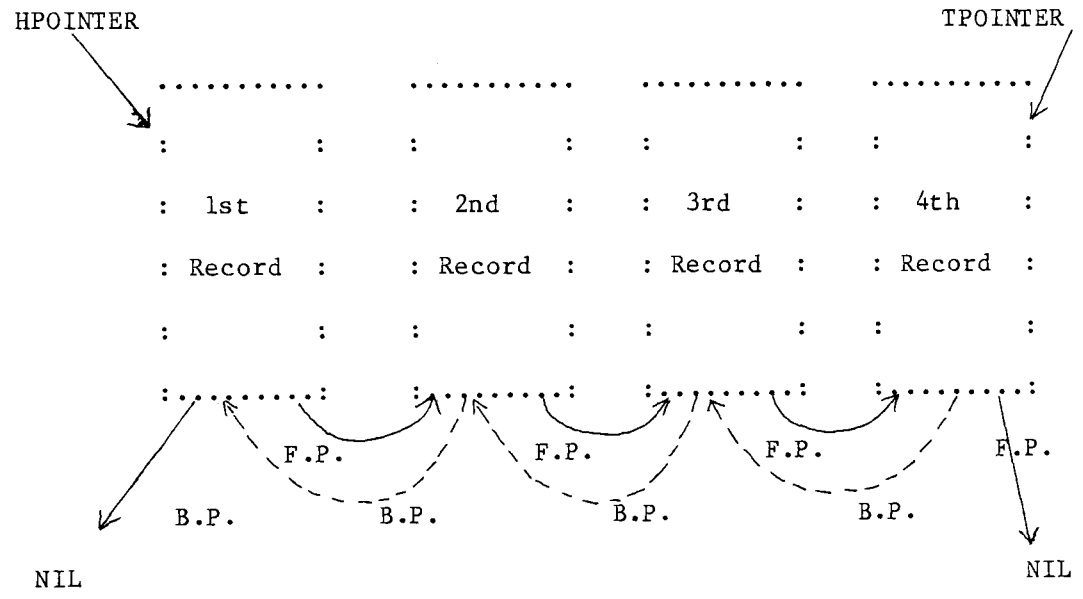
d. A one dimensional array of size NC+NJ needed to keep track of rows exchanged during the LU decompostion process, see next section.

e. An integer specifying the position of the first row in a block. This could be altered during the LU decompostion process, to allow for row interchanges.

f. Two fields of type Pointer. One points forward, while the other points backward. The forward pointer is needed to get access to records correspond to later blocks. The backward pointer is specially needed in the back substitution process of the solution, as in this process, we must start from the last block and move backwards to the first.

g. Other fields are introduced later in the work.

3. Two variables of type pointer (not record's fields), are needed to refer to the first and last Records of the list. These two variables help in accessing the list of records from either side. Hence, the collocation matrix is actually represented as a TWO WAY LINEAR LIST. Fig. 2.2 shows a picture of this list and how it is linked.



F.P. = Forward Pointer.

B.P. = Backward Pointer.

HPOINTER = HEADPOINTER.

TPOINTER = TAILPOINTER.

Figure 2.2

Two way linked list implementing the

Collocation matrix, for P = 4

2.4 NOTES ON THE LINEAR ALGEBRA USED TO SOLVE THE COLLOCATION EQUATIONS

A special LU decompostion procedure written in ALGOL W, has been designed by Dr. K. Wright (Computing lab., University of Newcastle upon Tyne), to take care of the block structured collocation matrix and the difficulties it implies. We translated these procedures to PASCAL and used them to obtain our solution. The main difference between the two versions is that in ALGOL W the matrix was represented as a 3-D Array, while in the PASCAL version, the data structure described above was used. The amount of work required by the two version is the same, although, in PASCAL case, the access to the elements of the matrix becomes more complicated. Algorithms for matrix of staircase type have been discussed in detail by Fourer[1984]. The algorithm used here is equivalent his row pivoting algorithm for a reduced staircase matrix of the special type described here.

The following are difficulties implied by the matrix structure:

1. Since half of the elements of the join condition equations lie in one block and the other half lie in the next block, the row exchange process requires special attention. If one of these rows is selected for exchange, both halves of the row must be exchanged. This would require movement of elements from two blocks.

2. If the row chosen for interchange has elements in the next block, and the row with which it is to be interchanged does not have elements in that block, then the integer which specify the first row in the next block is altered and the extra rows previously not set (in this block) are set to zeros.

3. For the right hand side, we may need also to move a value from one block to another, if this exchange takes place.

2.5 STRUCTURE OF THE Q-MATRIX AND DATA STRUCTURE REPRESENTING IT

The Q-matrix, of AHMED[1981], is defined as 'The matrix that maps the right hand side values into solution values, the right hand side is evaluated at the collocation points while the solution is evaluated at some set of points, which could be the collocation points'. It is a full block matrix and its blocks are also full. The theory and construction of this matrix will be discussed in the next chapter. The total number of blocks in the matrix is equal to P^2, where P is the number of intervals. Each block is of size n x d, where n is the number of collocation points and d is the number of evaluation points needed in constructing the matrix. Collocation points could be used as evaluation points, then, the size of a block will be n x n. The matrix size increases as the number of intervals increases. Thus, this size, goes up as the process of finding the estimated solution goes on. Again, this makes the use of a PASCAL Array to represent the matrix inconvenient. Pointers and Records are

considered , yet again, as an alternative way of representation.

Fig 2.3, shows the structure of the Q-matrix. Representation by Pointers and Records requires the following :

1. A Pointer, POINTER1, is needed to access the first block in a column of blocks. Here, we consider the matrix to consist of a set of columns of blocks.

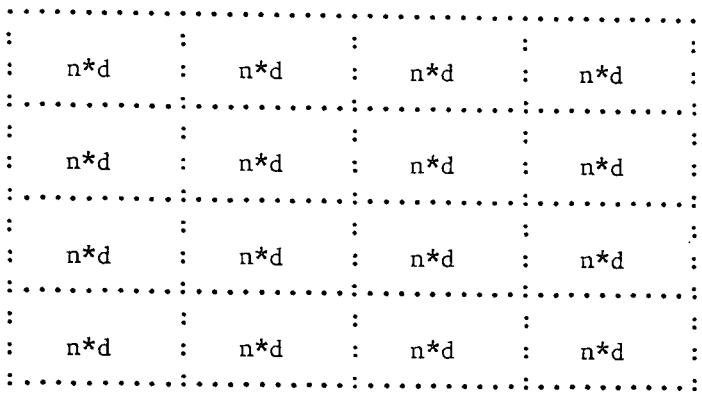
2. A Pointer, POINTER2, is needed to access the rest of the blocks in a column.

3. Each block is represented by a record, which has the following fields:

   a. A Pointer of of the same type as POINTER2. This is used to link records (blocks) of a column.

   b. A field to store values of the Q-matrix elements in a block, it is of type Array, its size is v x n, where v is the number of evaluation points.

In order not to use a large number of different Pointers, for space economy. The pointer used in representing the collocation matrix of the previous section is used as POINTER1. With records representing blocks of the collocation matrix have a new field of the same type as

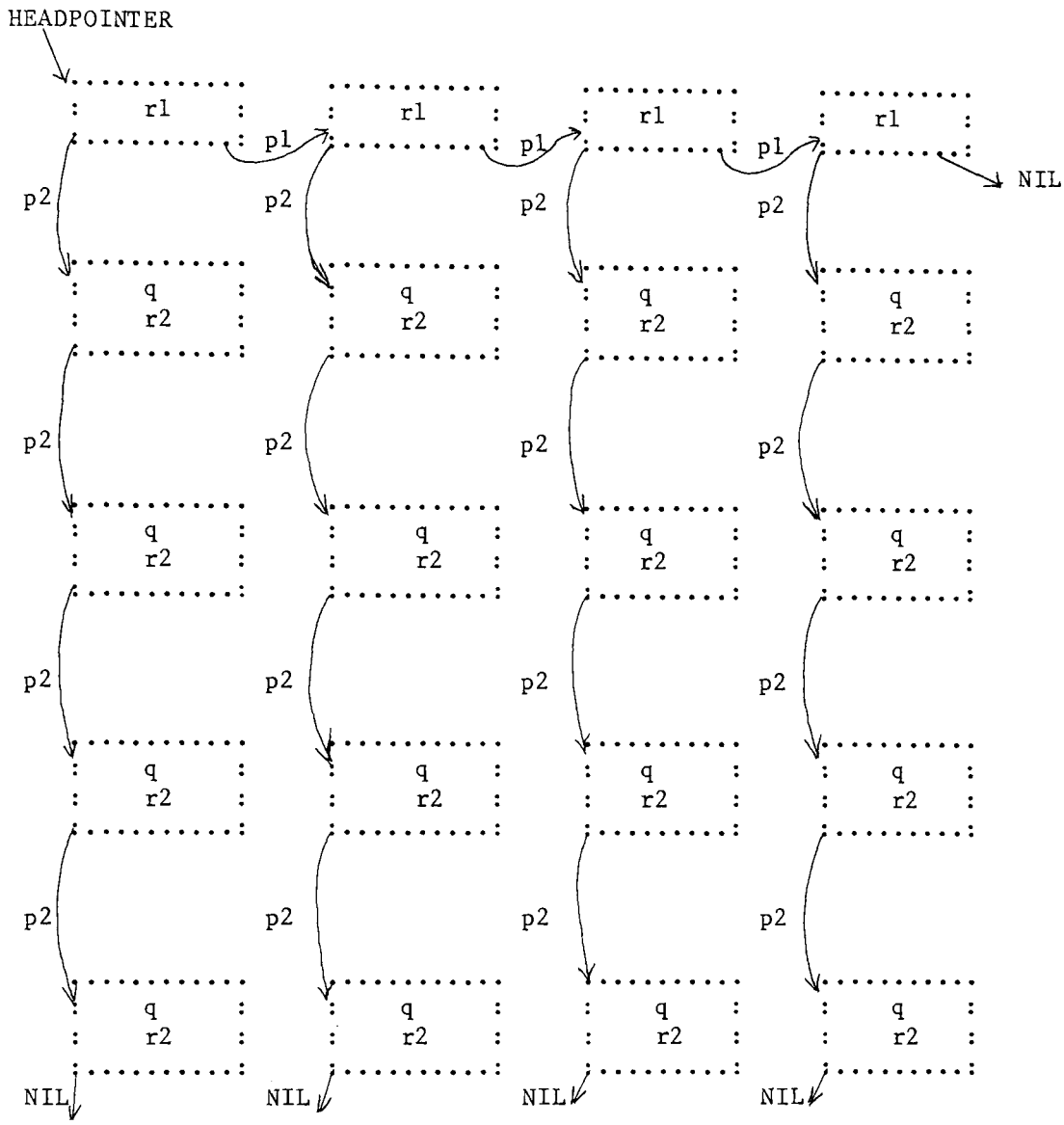


n is number of Collocation, d is number of Evaluation points

Figure 2.3

The Q-matrix for P = 4

POINTER2. This field is used to point to the first record of a column. By this representation, we can say that the Q-matrix has been represented by two One Way Linear Lists. One, is the list connecting records which point to the first record of each column, the other, is connecting records of a column. Fig 2.4, shows the representation of the Q-matrix by the two pointers. By this representation, accessing elements of the Q-matrix becomes a bit more difficult than when arrays are used. For example, if we want to access an element in block(3,2), we must first use POINTER1 to access to the first record of the 2nd. column. Then using POINTER2, we access the 3rd. record in that column. After that, we access the elements of block(3,2). Using an Array would make the access operation easier, but doing that would lose the dynamic properties of the data structure.



p1 = POINTER1, r1 = record1, p2 = pointer2, r2 = record2,

q = elements of Qmatrix in a block.

Figure 2.4

Representation of Q-matrix by Pointers and Records

2.6 COMMENTS ON THE PASCAL PROGRAM

We have designed the PASCAL program used in performing the work in this thesis to be flexible, i.e it can be changed and run easily. This is to allow the program to be used to test a variety of algorithm strategies and test problems. In addition, we tried to minimize the compilation cost. To achieve these two purposes, the program has been broken into a number of separately compiled parts. These parts are :

1. A part contains procedures needed to define the problem and its exact solution.

2. A part contains procedures that are used to build the collocation matrix of fig 2.1.

3. A part contains the LU decompostion procedures.

4. A part contains procedures corresponding to the different AMS algorithms introduced throughout this work.

5. A part contains a set of utility procedures including, a procedure to evaluate a Chebyshev series, a procedure to differentiate a Chebyshev series, another to reset the collocation matrix after each iteration, a procedure to insert a new interval, a set of procedures to evaluate quadrature weights

and a procedure to print out the solution or/and the collocation matrix (if required).

6. A main part connects all parts and links them to obtain the solution.

The only case where all these different parts are recompiled is when the number of collocation points is changed.

The following are cases introduced to reflect the flexibilty of the program :

1. If different polynomials (not Chebyshev polynomials) are used to represent the solution, a few procedures must be altered, these are, the evaluation procedure, procedure which is used to differentiate the polynomials and, some of the procedures used in building the matrix. The rest of the procedures need not be changed.

2. The program could start from different number of initial intervals, from one onward. Initial intervals could be of equal size or of different sizes.

3. For different problems, using the same block size, only one part needs to be changed and recompiled, that is the part which contains the corresponding procedures defining the different

equations.

Some special purpose procedures written by other people are needed in our work. These are NAG (Numerical Algorithms Group) library procedures and GHOST (GrapHic Output SysTem) package procedures. Both NAG and GHOST are written in Fortran. A NAG procedure which evaluates the Error Function is used in computing the true solutions for some test problems. A number of GHOST procedures are used to plot the mesh used in different iterations.

All the calculations were performed in double precision arithmetic on IBM 370/168 computer using UBC/PASCAL language.

CHAPTER THREE

PRELIMINARY COMPARISON FOR SOME MESH SELECTION ALGORITHMS

3.1 INTRODUCTION

In this chapter, we introduce four different mesh selection strategies with their corresponding AMS algorithms. These strategies are :

1. Largest Last Solution Coefficient (LLSC) strategy.

2. De Boor's derivative approximation strategy.

3. Largest Residual (LR) strategy.

4. Q-matrix strategy.

Each strategy will be discussed in detail focusing on the criterion used, motivation and theoretical background. The algorithm's structure will not be discussed as it is the same as that of section 1.4. However, we will comment on the algorithms and compare them with other algorithms which are based on the same criterion and implemented by other people. At the end of the chapter, we compare the four strategies, in terms of efficiency and cost, using a set of stiff test problem. Each of these problems has a different solution

characteristic, which could have one or more layer regions in different positions. A comparison similar to ours, in terms of problems used and factors on which the comparison is based, is given in Ahmed[1981]. Another comparison similar to ours in terms of some of the strategies used is given in Russell and Christiansen[1978].

In addition to the comparison, Russell and Christiansen classified mesh selection strategies according to the function with respect to which the mesh is equidistributed. These classes are :

1. The first class of strategies, are defined as strategies where the basic approach is to try to asymptotically equidistribute the mesh with respect to an approximation to a derivative of the estimated solution.

2. The second class of strategies, are defined as strategies where the basic approach is to try to equidistribute the mesh with respect to an approximation to the estimated error.

Formulae for the infinity norm of the local error are also given in Russell and Christiansen, the first one is

$$e_i = Cl\ h_i^s \left\| Z_i \right\| \qquad i=1....P; \qquad (3.1);$$

where, e_i is the norm of the error estimate in the ith interval, Cl is a constant, h_i is the size of the ith interval, s is the order of approximation and Z is a function representing the truncated Taylor's series of the local truncation error as given in Lentini and Pereyra

[1975].

The second formula is

$$e_i = C2\ h_i^s \left\| U_i^{(s)} \right\| \qquad i=1.....P; \qquad (3.2);$$

where U is the exact solution of the B.V.O.D.E. problem. Usually an estimation of U and Z are used in the formulae to obtain an estimation for the error norm. According to these two formulae, we can say that both the first and second class strategies will produce meshes which are equidistributed with respect to the infinity norm of the estimated local error. A definition for equidistributed meshes is given in section 1.4, definition 1.4.

Ahmed dealt with the following strategies:

1. A strategy that uses an approximation to the first derivative of solution as a criterion, this could be classified as of the first class.

2. A strategy that uses the residual as criterion.

3. A strategy that uses an error estimate, derived from the residual and the Q-matrix, as criterion. This strategy could be classified as of the second class.

Our four strategies are introduced in the next four sections. A comparison between them and problems used in this comparison will be

introduced in the last section.

3.2 LARGEST LAST SOLUTION COEFFICIENT (LLSC) STRATEGY

The solution is expressed, for the jth interval, as follows, where s = m+n:

$$y_{np}(x) = a_0T_0 + a_1T_1 + \ldots\ldots\ldots\ldots + a_{s-1}\,T_{s-1} \qquad (3.3)$$

where T_i, is the ith degree Chebyshev polynomial. For the LLSC strategy, the coefficient a_{s-1} is used as a criterion, and the interval which has largest a_{s-1} is selected for subdivision.

3.2.1 Motivations for the strategy

The reasons for using the last solution coefficient as a criterion are:

1. The rate of decrease of the Chebyshev coefficients, in an interval, could be used as an indicator of the smoothness of the solution's representation. For this strategy we use the a_{s-1} to measure this rate of decrease and hence the smoothness of the estimated solution. A large a_{s-1} implies a less smooth estimated solution, which is likely to give a large error. To improve the smoothness, and hence reduce the error, the interval with large a_{s-1} is divided.

2. The coefficient a_{s-1} is related to (s-1)th derivative of the

estimated solution as follows:

$$a_{s-1} = C\, h_i^s\, y_i^{(s-1)} \;;$$

where C is a constant and h_i is the size of the ith interval. This strategy, is actually producing a mesh which is equidistributed with respect to a_{s-1}, and consequently, according to above relation, is equidistributed with respect to $y^{(s-1)}$, i.e. we would have:

$$a_{s-1} = C\, h_i\, y_i^{(s-1)} = \text{constant} \qquad \text{for } i = 1...P;$$

This strategy is one of the cheapest. It does not require any extra calculations, all it requires is to search for the interval which has the largest last coefficient, and these coefficients have been already obtained in the process of evaluating the estimated solution.

3.3 DE BOOR STRATEGY

In this strategy, an estimate for the error norm of from (3.2) is used as criterion. For this criterion, we need to evaluate $U^{(s)}$, where U is the true solution. This is impossible as the U is not known explicitly, consequently its sth derivative could not be evaluated. Furthermore, evaluating the sth derivative of the approximated solution in a single subinterval is not useful as it is clearly zero. De Boor[1973] suggests using values from neighbouring intervals to obtain an estimation to $U^{(s)}$ in the form of a piecewise H(t)

which is defined as follow:

$$
H(t) =: \begin{cases} \ldots \\ : \; 2\left|\Delta V_{3/2}\right|/(t_1 - t_2) & \text{on } (t_1, t_2) \\ : \\ \left|\Delta V_{i-\frac{1}{2}}\right|/(t_{i+1} - t_{i-1}) + \left|\Delta V_{i+\frac{1}{2}}\right|/(t_{i+2} - t_i) & \text{on } (t_i, t_{i+1}) \; i=1..P-1 \\ : \\ : \; 2\left|\Delta V_{P-\frac{1}{2}}\right|/(t_P - t_{P-1}) & \text{on } (t_P, t_{P-1}) \\ :.. \end{cases} \quad (3.4);
$$

where Δ is the forward difference operator,

with $V_{i+\frac{1}{2}} = (U^{*(s-1)}(t_{i+\frac{1}{2}}))$ on (t_i, t_{i+1}) all i ;

where U^* is the approximated solution, t_i are the mesh points and P is number of intervals. From these formulae, H(t) is taken as the slope at point $t_{i+\frac{1}{2}}$, on interval (t_i, t_{i+1}), of the parabola interpolating the (s-1)th. derivative of U^*, taking $t_{i-\frac{1}{2}}$, $t_{i+\frac{1}{2}}$ and $t_{i+\frac{3}{2}}$ as interpolation points. Evaluating H(t) does not require a large amount of work as the (s-1)th derivative of V is related to last Chebyshev coefficient, a_{s-1}, as has been shown in the previous section.

This strategy can be regarded as a modified version of the LLSC strategy, because here we use the series coefficients to estimate $U^{(s)}$. Both strategies will produce a mesh which is equidistributed with respect to the derivative of a solution, though, this strategy may perform better due to the use of this estimation which is appropriate to the error estimate. The cost of this strategy is more than the LLSC

strategy as it requires the evaluation of H(t) and e_i then a search for the interval which gives the largest e_i, while we mentioned earlier that, the LLSC strategy only requires a search. This strategy is of the first class, as described in section 3.1.

3.3.1 Motivations

The reasons for using this strategy are:

1. To reduce the error in an interval, it seems sensible to subdivide it. This strategy is selecting intervals which give largest estimated error norm for subdivision.

2. This strategy produces a mesh which is equidistributed with respect to the derivative of the solution, i.e.

$$h_i^s H(t) = h_i^s \left\| U_i^{(s)} \right\| = \text{constant} \qquad \text{for } i=1....P.$$

3.3.2 De Boor algorithm

De Boor[1973], introduced an AMS algorithm which produces a complete new mesh in each iteration using H(t) given 3.4. The mesh points are defined by the following formula:

$$t_{i+1} = I^{-1}\left(\frac{i}{P} I(b)\right) ; \qquad i=1.....P ;$$

where $I(t) = \int_a^t H(w)^{1/s} \, dw$.

Our algorithm is different, we subdivide one interval in each

iteration keeping the rest of the intervals unchanged. Thus, his algorithm can be called mesh placement while ours can be called mesh subdivision. Both algorithms are based on the same criterion and produce meshes which are equidistributed (approximately) with respect to α derivative. The amount of work, per iteration, involved by our algorithm is less than his, but ours may require more iterations. However, this could be decreased by subdividing the interval more than once in each iteration. In his algorithm, the accuracy of evaluating the mesh points could vary, depending on the numerical integration method used in evaluating the two integrations. Of course, if a more accurate method is used, the cost will increase.

3.4 LARGEST RESIDUAL (LR) STRATEGY

For this strategy, we use the infinity norm of the residual as criterion. The residual function r(t) for problem 1.1a, 1.1b is expressed as follows:

$$r(t) = \eta(t) - (Dm - T)\, y_{np}(t) \qquad (3.5);$$

where (Dm - T), is the differential operator defined by equation 1.2. The residual function could be approximated by an interpolating polynomial which we call the principle part of the residual.

For the LR strategy, we subdivide the interval which gives the largest residual norm. To obtain this norm, we need a method of evaluating the maximum of r(t). Note that r(t) may be evaluated at any point straightforwardly, but still finding its maximum is not a

trivial or cheap process. An estimation of this is obtained by evaluating r(t) at a set of points in each interval. The accuracy of estimation increases as the number of points increases. However, we could get good accuracy, with relatively few points, by evaluating r(t) at the Chebyshev extrema. This is because r(t) has a factor of Tn(t) and one would expect the maximum of r(t) to occurs near the maxima of Tn(t). The largest r(t) evaluated at the extrema is then taken as an estimate of the maximum of r(t) and its norm. (Chebyshev extrema are defined as follows:

$$t_i = \cos(i\,\pi/n) \qquad i=0,\ldots\ldots\ldots,n\ ).$$

3.4.1 Motivations for the strategy

The reasons for using the residual norm as criterion are:

1. Using equation (1.2) and (3.5) the error $e = y_{nP} - y$ is related to the residual by:

$$e = (Dm - T)^{-1} r. \qquad (3.6)$$

If we assume that the values of the kernel, k(s,t), for the operator $(Dm - T)^{-1}$ are dominant near s = t, then, the behaviour of the error function will exactly follow the behaviour of the residual. This means that interval with large residual is actually giving large error, surely , such interval should be divided. However, there are some exceptional cases where the residual and the error behave differently. An example of such case will be given later in this chapter.

2. From equation 3.5, we see that the residual is actually taken as:

$$(\mathcal{L}\, y_{np})(t) - \eta(t) \ ;$$

where $\mathcal{L}$ is the differential operator defined in 1.1 and $\eta(t)$ is the R.H.S. of B.V.O.D.E. problem.

Thus, the residual should be zero for the true solution and it seems reasonable to reduce the interval with the largest residual.

3. Another relation between the global error norm and the global residual norm is introduced by Humphrey and Carey[1978]. They found the following empirical relation, for their specific problem:

$$\| e \| = 2.212E\text{-}3 \ \| r \|^{1.517} \ ;$$

where the norm is taken in Hilbert space H_0 and -r- is the residual whose norm is over the whole range of r.

It is clear from this relation that reducing the local residual will reduce the global residual and consequently, the global error.

Humphrey and Carey studied in detail the use of the residual as criterion for AMS algorithms. They introduced a strategy that uses a statistical approach involving mean value and standard deviation of the residual in deciding how many times the selected interval should be divided. The decision on which intervals should be divided and when to stop selecting intervals for subdivisions are made depending

on the residual. They also observed that, usually, the maximum value of the residual function occurs near an end point. We have observed this also, but, though not in all cases. The algorithm we used is a simple and cheap one compared with that of Humphrey and Carey. It requires only the evaluation of the residual norm in each interval, then a search for the interval which gives the largest residual norm.

3.5 Q-MATRIX STRATEGY

Before we talk about the strategy, we give a definition and background of the Q-matrix. In constructing this matrix, we require to evaluate the estimated solution at a set of points which we will refer to as 'evaluation points'. The position of these points is relatively the same for all intervals. The Q-matrix is introduced by Ahmed[1981]. He defined it as 'The left inverse of the approximation matrix when the parameters defining the solution are taken as the solution values at the interpolation points'.

To make this more clear, consider the vector $\underline{\eta}$ of right hand side values at the collocation points, the vector $\underline{y}$ of solution values at some (possibly different) set of evaluation points, and the vector $\underline{b}$ consisting of the inhomogenous terms in the boundary conditions. These are related by an equation of the form:

$$\underline{y} = Q\underline{\eta} + \bar{Q}\,\underline{b} \ .$$

where Q and $\bar{Q}$ are matrices of appropriate dimension. This can be considered as a defintion of Q.

Ahmed showed that under certain conidtions (in particular for certain choices of points); that:

$$||Q|| \Longrightarrow ||(Dm - T)^{-1}|| \quad \text{as } P \Longrightarrow \infty.$$

This suggests using $||Q||$ to estimate $||(Dm - T)^{-1}||$, and even using Q as a discrete appoximation to $(Dm - T)^{-1}$. Using (3.6) then gives:

$$||e|| <= e^* \simeq ||Q|| \cdot ||r|| \qquad (3.7);$$

Before Ahmed introduced the matrix Q for piecewise collocation method, a similar matrix called W , which involves values of Dm x rather than x, had been introduced, for global polynomial collocation by Cruickshank and Wright[1978], then, Wright[1984], studied the properties of this matrix. Gerrard and Wright[1984], studied the corresponding matrix for piecewise polynomial collocation. In these papers, it has been proved (again under suitable conditions) that

$$||W|| \Longrightarrow ||(I - T\,Dm^{-1})^{-1}|| \quad \text{as } P \longrightarrow \infty .$$

where I is the identity operator, Dm and T are defined in chapter 1. They also proved that Wn is related the inverse of the collocation matrix. This is not a straight forward relation, as the parts of the inverse matrix which correspond to boundary and join conditions are excluded from the relation which is as follows:

$$W = \text{sub}[S] ;$$

where sub means a sub-matrix, see below, and

$$S = A_0 \; A^{-1} ;$$

where A^{-1} is the inverse of the collocation matrix using some representation of the solution and A_0 is the matrix corresponding to

the to the operator Dm, using the same representation. Similarly, the matrix Q is related to the inverse of the collocation matrix as follows:

$$Q = \text{sub}[Z] \ ;$$

with

$$Z = A^{*}_{0} \ A^{-1} \ ;$$

here A^{*}_{0} is a matrix corresponds to the operator I.

The matrix Z is a full block matrix, each block is a full one with size $(n+2m) \times (n+m)$, where n is number of collocation points and m is the problem's order. The matrix Q is a sub matrix of Z, it contains rows corresponding to the collocation equations only. Q is obtained from Z by neglecting rows and columns resulting from multiplying join condition rows of A^{*}_{0} by columns of A^{-1}. Q is also a full block matrix, each block is full with size of n x n, assuming that the collocation points are used as evaluation points, the number of blocks in Q is P^2.

3.5.1 Constructing and Condensing the Q-matrix

To construct Q, we do not evaluate A_0 and A^{-1} neither do we multiply them explicitly. What we do is to obtain $A^{-1} \underline{e}_i$, where $\underline{e}_i$ is the unit vector (with i corresponding to a collocation equations) by repeatedly calling the forward and backward substitution procedures using the LU decompostion of A. Then, the multiplication is carried out implicitly as it is equivalent to evaluating a Chebyshev series with the solution coefficients. The

algorithm used in building Q constructs it column by column, there are n of them corresponding to each interval, one column generated for each collocation point. For example, for the 1st collocation point of the first interval, the algorithm for constructing these columns uses the following steps:

1. Set the right hand side elements in all blocks, except the first block, to zeros. For the first block, set the element corresponding to the 1st. collocation point to 1, and the rest of the elements to zero.

2. Use the back substitution to obtain the solution coefficients, this gives a column of A^{-1} .

3. Evaluate the Chebyshev series at the k evaluation points, in each interval, using the interval's Chebyshev solution coefficients. This will give us a column of k elements.

These steps will be repeated for each collocation point of the first interval, then for all other intervals.

In the computations of this algorithm, we do not need the actual elements of the Q-matrix. We actually use the norm of each block of Q . This also reduces the storage space required by the algorithm as we store only one element in each block instead of n x n elements.

To obtain the norm each block of Q , we first condense the blocks. The condensation operation is as follows, first assume that each Q element is refered to by four subscripts, these are:

i,j to refer to block(i,j) of Q .

p to refer to rows in a block.

k to refer to columns in a block.

Thus, an element of Q will refered to as q_{ijpk} .

Now, the condensation operation for block(i,j) can be put algebraically as follows:

$$q_{ijk} = \sum_{p=1}^{n} |q_{ijpk}| \quad ;$$

now Q^* is the condensed version of Q .

Each block of Q^* has a vector of n elements instead of n x n elements of Q .

After this condensation, the norm, in each block, is found by taking it to be equal to the largest among the n values of the vector, i.e.

$$\| Q^* \| = \max_k \left| q^*_{ijk} \right| \; ;$$

This norm is what we actually store and use in the algorithm. This saves a large amount of space as we store one element for each block instead of n x n elements. The error estimate (3.7) along with the block decompostion of Q suggest an alternative estimate $\bar{e}$ of error in the ith interval in the form of contibutions from the other intervals be used. That is:

$$\bar{e}_i <= \sum_{j=1}^{P} \| Q^*_{ij} \| \| r_j \| \quad i = 1,.....,P \qquad (3.8) .$$

Here the i and j subscripts are used to to refer to block(i,j).

3.5.2 Description and comments on the strategy

This strategy differs from the previous three strategies, where we selected the interval that gives largest criterion value. In this strategy we do the following:

1. Find the interval which gives largest $\bar{e}_i$ evaluated by using 3.8, and suppose it is the i^*th interval.

2. Look for the largest $\| Q_{ij} \| \| r_j \|$ term of $\bar{e}_{i^*}$, suppose it is the j^*th term.

3. The j^*th interval is selected as the subdivision interval.

From this, we see that this strategy does not select the interval with the largest error estimate, but it selects the interval which gives the largest contribution, to the largest error estimate.

This strategy is more complicated than the other three, as it uses a complicated process in finding the interval to be subdivided. It is also more expensive, it requires the evaluation of the residual in each interval, which could nearly cost as much as the LR strategy.

In addition, it requires the construction and condensation of $Q_{..}$, which are both costly. It also requires, as we mentioned earlier, a large storage space and a complicated data structure to represent $Q_{..}$. In spite of all these drawbacks, we still use this strategy because we expect that the error estimation to be good and this could produce meshes that might be better than the other three strategies.

The method of estimating the residual norm for this strategy is the same as the method used in the LR strategy. To save some computational effort, we used the collocation points as evaluation points used to construct $Q_{..}$, as we have already evaluated them.

The motivation for this strategy is that it uses an error estimation which is expected to be more reliable than that of equation 3.2. It also uses explicitly the relation between the residual and the error.

3.6 COMPARISON OF STRATEGIES

In this section we introduce a set of test problems and perform a comparison between the strategies. All test problems used in this chapter have some sort of severe layers, some simple problems have been used to check the program but their results are not included here as they are not important for comparison purpose. Before we do the comparison, we briefly describe the structure of the tables of results which is not only used for this chapter, but for some of the

next ones as well. All tables have the same form, they have a field which shows the number of intervals used followed by a field which contains the largest values of the actual error for different number of intervals. The corresponding interval numbers are shown in the next field. The 4th and 5th fields contain the largest criterion values and the corresponding interval numbers respectively. The last field contains letters indicating which strategy has been used. The following are the letters and what they mean:

M : LLSC strategy.

D : De Boor strategy.

R : LR strategy.

Q : Q-matrix strategy.

This field is not strictly necessary here, but it is used in chapter 5 where the strategy is permitted to change. At the bottom of the table, there are two lines of information, one to show the cpu time spent on this run and the other to explain which criterion has been used. We should mention here that, for all tests, we have used three collocation points and an initial mesh of five intervals of equal size.

3.6.1 Problem 1

This problem is :

$$y'' - \lambda(2-x^2)\, y = -\lambda \ ;$$

over (-1,1); with boundary conditions

$$y(-1) = 0 \ ; \ y(1) = 0;$$

its true solution is

$$y(x) \triangleq (1/(2-x^2))-\exp(\sqrt{\lambda}(1+x)) - \exp(\sqrt{\lambda}(1-x)) ;$$

The results are obtained for λ = 1E8 .

This problem has two boundary layers, one at each of the end points. Tables 3.1, 3.2, 3.3 and 3.4 contains the results obtained by using the LLSC, DeBoor, LR and Q-matrix strategies respectively. Looking at the actual error and the tested values, in these tables, we see that they are going down smoothly as the number of intervals increases. This means that the solution is converging as the number of interval increases and all the strategies are producing sensible meshes. However, for the De Boor strategy (table 3.2), we see that the value of the actual error, with 30 intervals, is larger than the corresponding values in the other tables. This is because, for this problem, we have two boundary layers, the De Boor strategy picks one of them at the start, and keep subdividing the corresponding interval, then, after it has finished with it, it picks the other layer. While the other three algorithms alternate the subdivision between the two layers regions to keep an equal number of intervals in both layers. This behaviour of the De Boor strategy could put points more than required in one region while we need them in the other. This is because, in this strategy, the interval size is involved in evaluating H(t), and as the interval size decrease, H(t) increases, till we reach a state where the other terms used in evaluating H(t) becomes small enough to compensate for the interval size effect.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.06743E-01	1	2.424419E-01	5	M
10	9.06743E-01	1	2.424418E-01	1	M
15	7.60006E-01	15	2.410074E-01	15	M
20	6.44130E-01	1	2.094882E-01	1	M
25	2.18946E-02	1	1.378514E-02	2	M
30	2.69628E-04	27	2.765299E-04	2	M

The total time is = 10.460 seconds.

The tested value corresponds to the last solution coefficient.

Table 3.1

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.06743E-01	1	6.222965E-01	4	D
10	9.06743E-01	1	1.977096E+01	9	D
15	9.06743E-01	1	3.437208E+01	14	D
20	9.06743E-01	1	6.222851E-01	1	D
25	7.60006E-01	1	1.977093E+01	1	D
30	2.18946E-02	1	4.372082E+01	1	D

The total time = 9.062 seconds.

The tested value corresponds to the estimated error norm of De boor.

Table 3.2

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.06743E-01	1	9.999919E+07	5	R
10	9.00031E-01	1	9.998690E+07	1	R
15	7.60006E-01	15	9.916829E+07	15	R
20	6.44130E-01	1	8.817500E+07	1	R
25	2.18946E-02	1	9.432900E+06	1	R
30	4.71533E-04	5	3.432793E+05	30	R

The total time = 10.975 seconds.

The test value corresponds to estimated residual norm.

Table 3.3

no. of intervals:	largest error (actual)	interval: number	largest tested value	interval number	strategy
5	9.06743E-01	1	9.497704E-01	5	Q
10	9.06743E-01	1	9.851880E-01	10	Q
15	8.24714E-01	1	9.925259E-01	1	Q
20	6.44130E-01	20	8.917390E-01	20	Q
25	2.18946E-02	1	7.487558E-02	1	Q
30	5.12772E-04	5	1.770392E-03	28	Q

The total time is = 37.063 seconds.

The tested value corresponds to the largest share of error.

Table 3.4

1.6.2 Problem 2

This problem is :

$$y'' + \lambda x\, y' + \lambda y = 0 \ ;$$

over (0,1); with the following boundary conditions:

$$y(0) = 1 \ ; \ y(1) = \exp(-\lambda/2) \ ;$$

its true solution is: $y(x) = \exp(-(\lambda/2)\, x^2)$;

The results are for $\lambda = 300$.

This problem has a single boundary region near the left end point x=0. Tables 3.5, 3.6, 3.7 and 3.8 contains the results for this problem when the LLSC, De Boor, LR and Q-matrix strategies are used to select the mesh points. From these results, we see that all of them have performed well, with the actual errors and the criterion values going down smoothly as the number of intervals increase. However, if we put them in order of final accuracy, the Q-matrix comes first, De Boor comes second, LR comes third and LLSC comes fourth. But if we take the time factor into consideration,as well as accuracy, the De Boor method will be first and Q-matrix will be last. This is because the Q-matrix strategy requires more cpu time than the others while it gives nearly the same accuracy as any of the others (nearly the same value of largest actual error is obtained by all strategies).

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.987479E-01	1	5.2309377E-02	1	M
10	2.727010E-03	5	1.8687266E-04	5	M
15	4.791559E-05	4	1.9821173E-05	5	M
20	1.036622E-05	5	3.8969423E-06	12	M
25	6.560835E-06	13	1.3139307E-06	9	M
30	7.517590E-06	17	7.9153450E-07	8	M

The total time = 8.171

The tested value corresponds to the last solution coefficient.

Table 3.5

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.987479E-01	1	2.6367407E-01	1	D
10	3.552821E-04	6	3.7276213E-03	3	D
15	9.423310E-05	5	5.7470494E-04	9	D
20	1.011366E-05	7	9.6636735E-05	6	D
25	7.672437E-06	6	5.7863475E-05	15	D
30	2.758760E-06	20	4.3896845E-05	10	D

Total time is = 8.213 seconds.

The tested value corresponds to the estimated error norm of De boor.

Table 3.6

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.987479E-01	1	2.5521401E+02	1	R
10	3.552821E-04	6	2.2960213E+00	4	R
15	9.423310E-05	5	4.0234170E-01	5	R
20	1.036622E-05	5	1.7282074E-01	11	R
25	7.252552E-06	10	5.2969762E-02	18	R
30	5.197336E-06	10	3.1459781E-02	1	R

The total time is = 9.328 seconds.

The tested value corresponds to the estimated residual norm.

Table 3.7

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.987479E-01	1	2.9527467E+00	1	Q
10	9.508838E-04	1	5.0812075E-03	2	Q
15	4.846671E-05	2	5.0013713E-04	3	Q
20	3.192901E-05	1	1.0127092E-04	4	Q
25	4.853424E-06	6	3.1807136E-05	6	Q
30	1.726986E-06	2	1.8566527E-05	9	Q

The total time is = 37.421 seconds.

The tested value corresponds to the largest share of error.

Table 3.8

3.6.3 Problem 3

This problem is :

$$y'' + \lambda x\, y' = -\pi \cos(\pi x) - \lambda \pi x \sin(\pi x) \;;$$

over (-1,1); with the following boundary conditions:

$$y(-1) = -2 \;;\; y(1) = 0 \;;$$

its true solution is given as:

$$y(x) = \cos(\pi x) + (\mathrm{erf}(x\sqrt{\lambda}/2)/\mathrm{erf}(\sqrt{\lambda}/2));$$

THe results are for λ = 1E6.

This problem has a layer region in the middle of the range near x =0 with a turning point , a definition of such a point is given in chapter 4. The results for this problem using the four strategies are shown in tables 3.9, 3.10, 3.11 and 3.12. Looking at the values of the actual error in tables 3.9 and 3.11 (for LLSC and LR strategies), we see that these values have not decreased as new mesh points are added, in fact they stayed nearly unchanged. This indicates that the solution is not converging and both strategies have failed to put mesh points in the regions where they are needed, near x = 0 in this case. For table 3.10 (De Boor strategy), we see that the actual error and the tested values have increased at the beginning, then they start to decrease, but, their values, with 30 intervals, are still large. At this stage, we are not sure whether convergence is occurring or not, what we are sure of is that, this strategy has put some points near x = 0, which causes the jump in the actual error and the tested values, but these points are not enough

to make the actual error small. The results for the Q-matrix strategy are shown in table 3.12. Looking at the actual error and the tested values, we see that these values have increased at the beginning, then, after the 10 intervals stage they start to decrease steadily. With 30 intervals, both values become small indicating that the estimated solution is converging, in fact, if we carry on with this strategy up to 40 intervals, the actual error becomes $\simeq$ 1E-4. This indicates that the Q-matrix strategy has put enough mesh points near the layer region (near x = 0). The distribution of the mesh points obtained by LLSC, De Boor, LR and Q-matrix strategy are shown in fig. 3.1, fig. 3.2, fig. 3.3 and fig. 3.4 respectively. Graphs of the actual errors are given in figure 3.4-a.

The LLSC strategy failed due to the existence of the turning point which causes the estimated solution, in an interval that contains it, to be very inaccurate making the last solution coefficient, consequently, inacccurate. The reason for the failure of the LR strategy is that for this problem, the residual is large in a different region to the error, hence, the LR strategy puts mesh points in regions where the error is small, and leaves the regions where the error is large with few points. De Boor strategy failed because the error estimation used depends on an estimation of the derivative of the solution, which for this problem is estimated inaccurately. This is because this estimation also depends on the estimated solution which is as we mentioned earlier inaccurate in the interval containing a singular points.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.645755E-01	3	7.5734602E-02	5	M
10	9.639900E-01	5	3.7480679E-02	2	M
15	9.639650E-01	9	3.7384198E-02	15	M
20	9.639592E-01	11	1.8621781E-02	14	M
25	9.639561E-01	13	1.8449242E-02	4	M
30	9.639523E-01	16	1.8362504E-02	29	M

The total time is = 7.622 seconds.

The tested value corresponds to the last solution coefficients.

Table 3.9

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.645755E-01	3	3.7760646E-01	4	D
10	2.844489E+02	5	7.1007006E+02	9	D
15	1.921933E+02	5	5.5031699E+02	4	D
20	7.118522E+01	10	3.5282835E+02	1	D
25	7.098966E+01	15	4.1090447E+02	10	D
30	4.884422E+01	21	2.6506005E+02	20	D

The total time is = 7.643 seconds.

The tested value corresponds to the estimated error norm of De boor.

Table 3.10

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.645755E-01	3	3.0010465E+06	5	R
10	9.644693E-01	3	2.9947375E+06	1	R
15	9.643899E-01	7	2.9788838E+06	1	R
20	9.643517E-01	9	2.9465760E+06	1	R
25	9.643487E-01	12	2.9249745E+06	17	R
30	9.643515E-01	15	2.8978264E+06	9	R

The total time is = 8.023

The tested value corresponds to the estimated residual norm.

Table 3.11

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.645755E-01	3	4.0010138E+03	3	Q
10	2.814752E+02	1	2.0239675E+05	10	Q
15	6.770273E+01	4	1.2738437E+04	4	Q
20	2.833283E+01	2	1.9133383E+03	1	Q
25	1.242431E+01	9	5.3501659E+02	9	Q
30	5.000664E-02	16	2.7011817E-01	16	Q

The total time is = 34.612 seconds.

The tested values corresponds to the estimated share of error.

Table 3.12

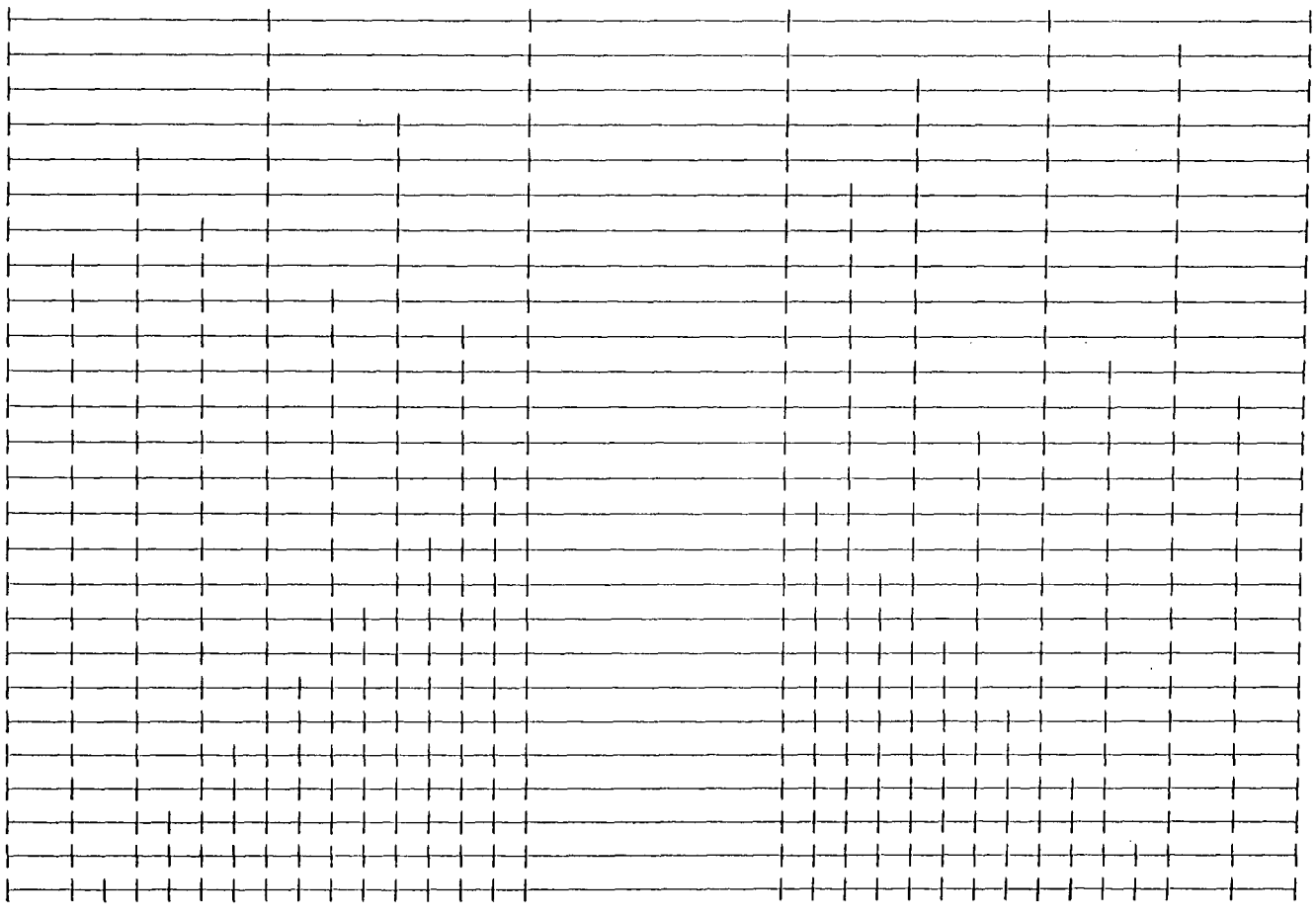


Figure 3.I - Mesh distribution for problem 3, Using the LLSC algorithm, $n = 3$, $P = 5$.

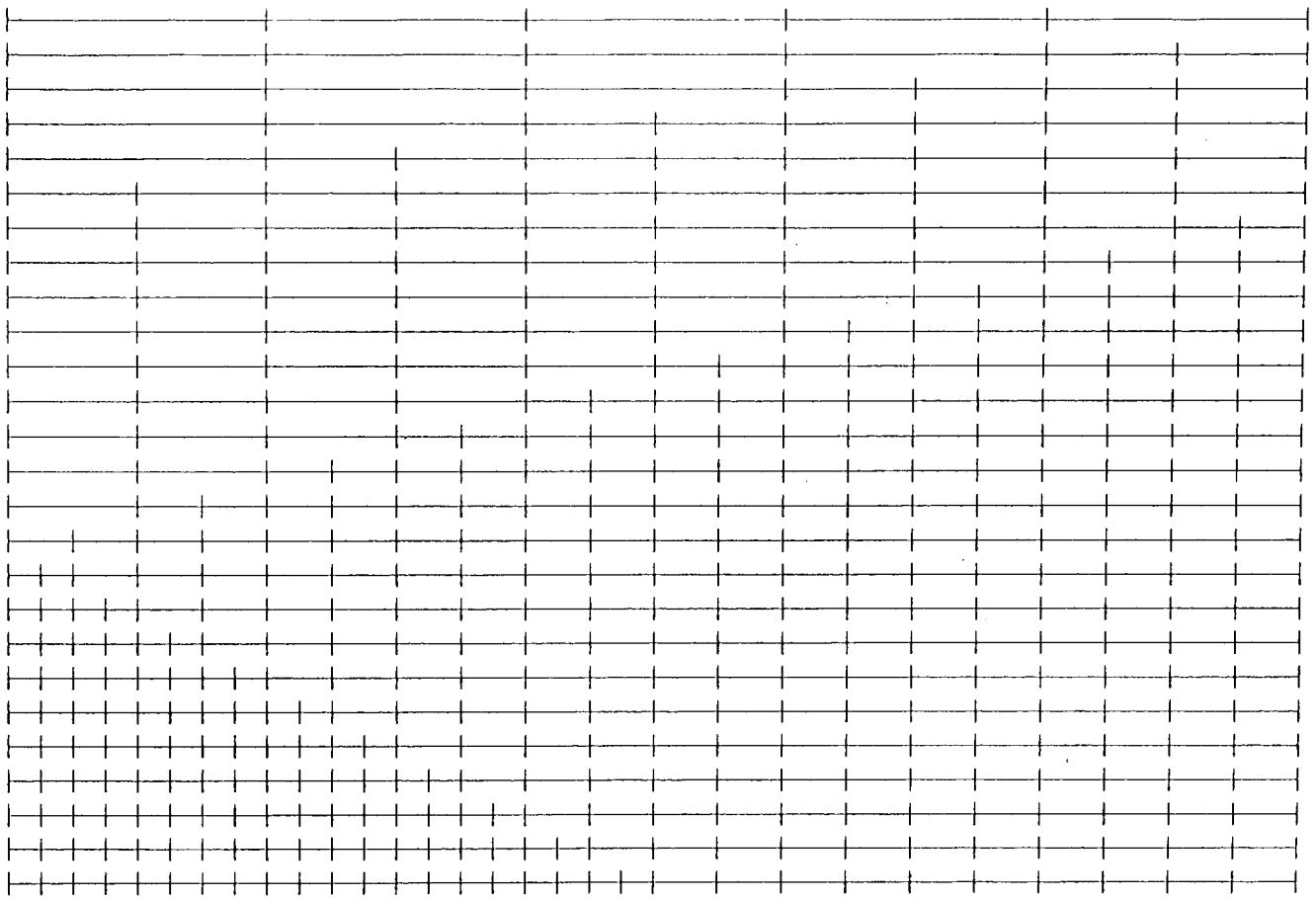


Figure 3.2- Mesh distribution for problem 3, Using De Boor algorithm, n=3, P = 5.

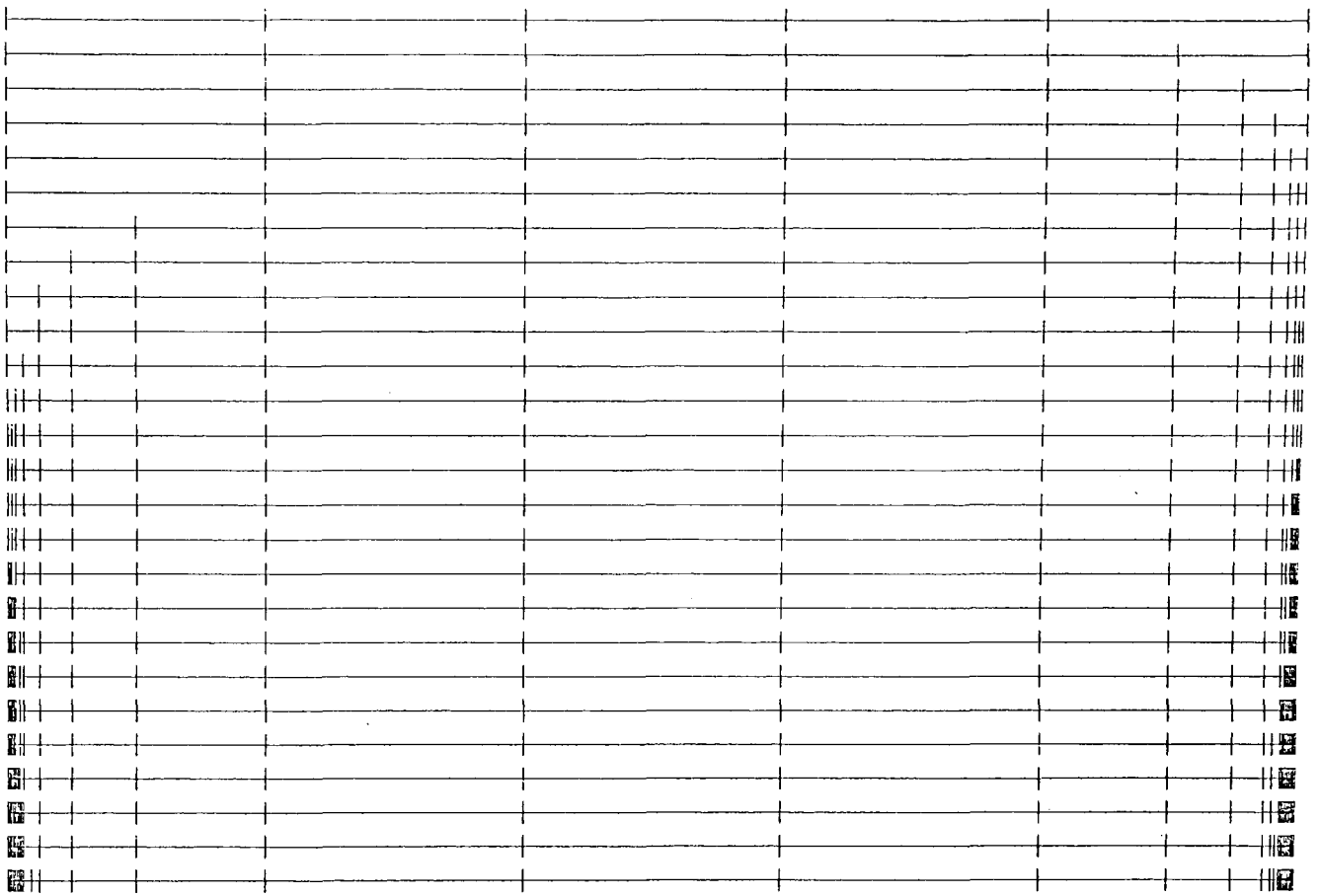


Figure 3.3- Mesh distribution for problem 3, Using the LR algorithm, n = 3, P = 5.

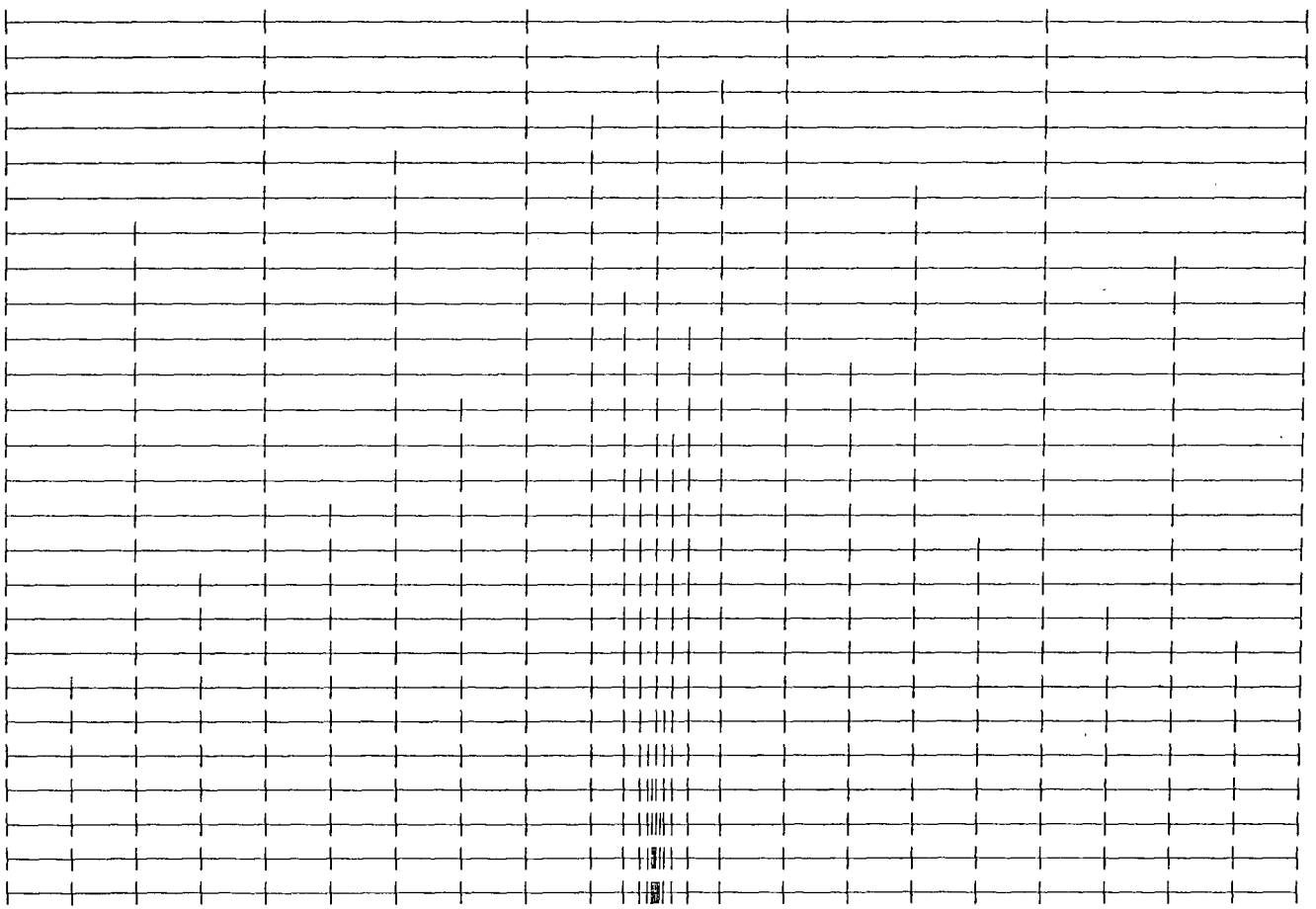


Figure 3.4- Mesh distribution for problem 3, Using the Q matrix algorithm, $n = 3$, $P = 5$.

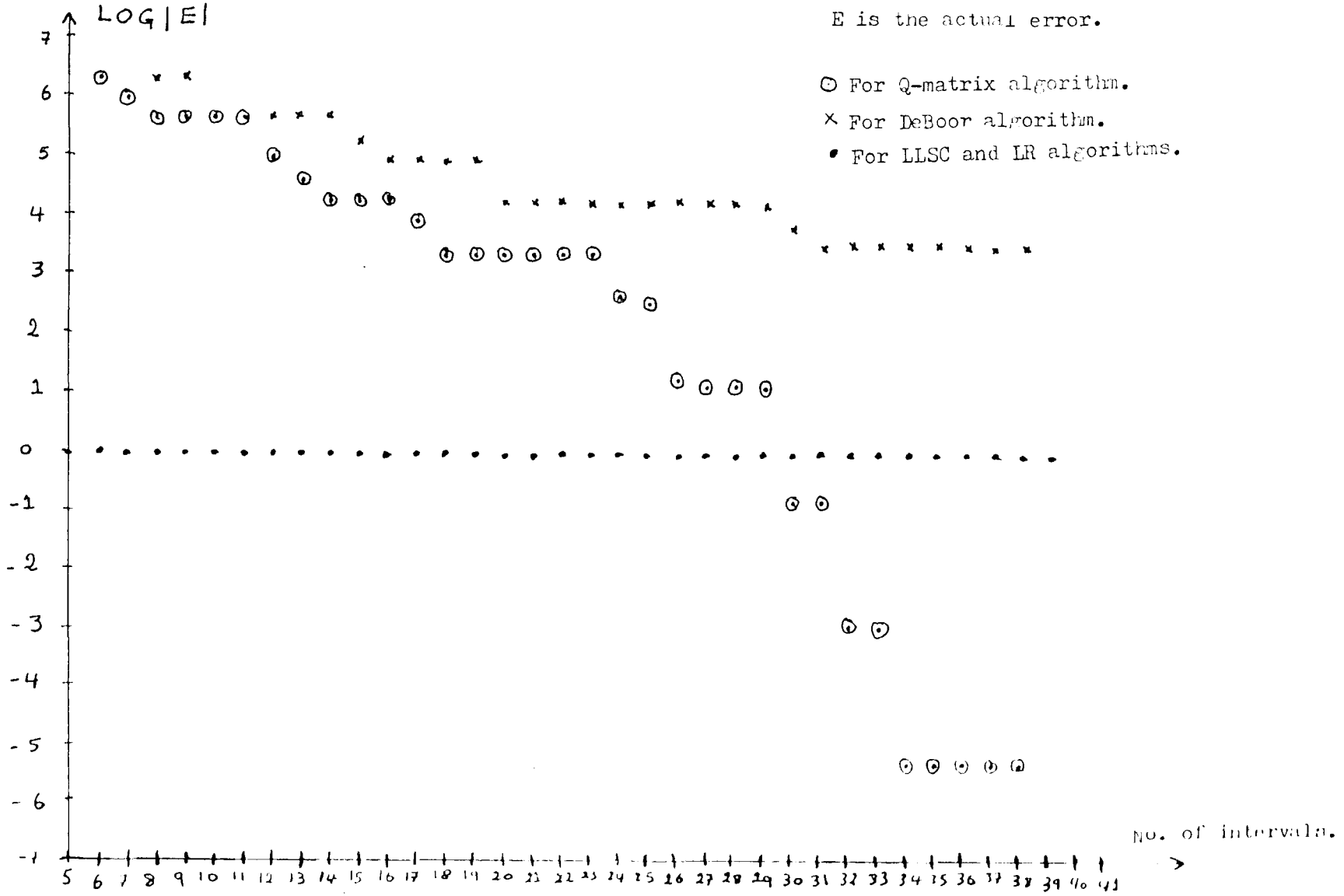

LOG|E|
E is the actual error.
For Q-matrix algorithm.
For DeBoor algorithm.
For LLSC and LR algorithms.
No. of intervals.

Figure 3.4-a.

3.6.4 Problem 4

This problem is:

$y'' + (2/x)\, y' + (1/x^4)\, y = 0$; over $(1/3\pi, 1)$; with the boundary condintions $y(1/3\pi) = 0$; $y(1) = \sin(1)$;

its true solution is

$$y(x) = \sin(1/x) .$$

The solution of this problem has no layer regions, but it has a highly oscillatory part in a region near the left end point. Thus, a good strategy is a one that puts enough points in the oscillatory region. Tables 3.13, 3.14, 3.15 and 3.16 shows the results for this problem using the four strategies. Looking at the errors and the tested values, we see that they are decreasing as the number of intervals increases, indicating that the solution is converging and the four strategies are producing sensible meshes. From tables 3.14, 3.15 and 3.16 we see that there are fluctuations in the values of the actual error, this is due to the fact that the corresponding strategies have placed more mesh points than required in the oscillatory region, while these points are needed somewhere else. This causes a little loss of accuracy in the estimated solution. For this problem, we could put the four strategies in the following order, from the accuracy point of view (taking into account the value of the actual error after 30 intervals), LLSC, Q-matrix, De Boor and LR.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.133246E+00	2	2.8444626E-01	1	M
10	3.687982E-02	7	2.9917671E-03	4	M
15	1.061317E-03	11	3.1779134E-04	5	M
20	4.804098E-04	16	7.5692626E-05	19	M
25	1.592066E-04	5	2.4956959E-05	8	M
30	9.287837E-05	6	1.4967811E-05	26	M

The total time is =9.034 seconds.

The tested value corresponds to the last solution coefficients.

Table 3.13

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.133246E+00	2	1.5702279E+00	1	D
10	3.687982E-02	7	4.8683274E-02	6	D
15	3.581989E-03	12	9.8683274E-03	10	D
20	5.276099E-04	12	1.8585200E-03	14	D
25	4.553759E-04	23	9.5828906E-04	23	D
30	6.232857E-04	27	4.1764170E-04	11	D

The total time is =9.453 seconds.

The tested values corresponds to the estimated error norm of De boor.

Table 3.14

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.133246E+00	2	2.6828378E+00	1	R
10	1.789371E-02	7	3.2300533E+01	3	R
15	5.592431E-03	11	1.0437382E+01	10	R
20	2.789053E-03	17	3.4817068E+00	1	R
25	3.951210E-04	23	1.3650187E+00	12	R
30	4.114992E-04	28	6.4701285E-01	25	R

The total time is =10.012

The tested value corresponds to the estimated residual norm.

Table 3.15

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.133246E+00	2	7.3358019E+00	1	Q
10	1.715930E-02	4	3.8330402E-02	4	Q
15	7.790217E-03	3	7.7621710E-03	3	Q
20	9.184755E-04	9	3.9239140E-03	19	Q
25	5.561207E-04	11	1.3987310E-03	5	Q
30	1.019629E-04	4	4.4176378E-04	22	Q

The total time is = 35.962 seconds.

The tested value corresponds to the estimated share of error.

Table 3.16

3.6.5 Evaluation of strategies

From the previous tests, we saw that the only strategy that produced reasonable meshes, which means converging solutions, for all four problems was the Q-matrix strategy. The other three have failed to produce reasonable meshes for problem 3. Thus, Q-matrix strategy is regarded as the superior one among the four. However, this strategy requires a large amount of cpu time, looking at the tables, we see that it requires a time nearly 4 times that required by any of the other strategies. The reason for this good accuracy of the Q-matrix strategy is that the matrix Q used in evaluating the error estimation may be regarded as a discrete approximation to the Green's function of the problem, which is a function that represent the inverse of the operator in the B.V.O.D.E. This helps in locating the layer regions for the problem, it also allows for the correct the relationship between the residual and the error. This makes the strategy work for problems where the residual and the error behave differently.

The four strategies are put in the following order, according to their performance on the previous four test problems:

1- Q-matrix strategy.

2- De Boor strategy.

3- LR strategy.

4- LLSC strategy.

Note that, in spite of the cost of the Q-matrix strategy, we put it as the best one, this is because, we believe that there is no point of using a strategy that does not produce a converging solution no matter how cheap it is. However, we could either try to improve the bad (cheap) strategies or try to find a way of reducing the cost of the good (expensive) ones. These two possibilities are investigated in the next two chapters.

One simple modification of the LLSC method was tried, this was to use the sum of the moduli of the last two elements. Clearly in some cases symmetry properties might cause the last series coefficients to be zero, and this modification would avoid breakdown in these cases. Problem 3 while not having a precisely symmetric or anti-symmetric solutions, is dominated by an anti-symmetric term near the turning point. This gives a small last coefficient in the series corresponding to the central interval and explain why the central interval is never reduced as indicated in figure 3.1. The modified algorithm did avoid this problem so that the central interval was halved first, but the later mesh reduction were evenly spread over the whole of (-1,1) and so the mesh produced was still not satisfactory.

CHAPTER FOUR

USING THE ANALYTICAL PROPERTIES OF THE B.V.O.D.E. PROBLEM FOR MESH SELECTION

4.1 INTRODUCTION

In this chapter, we study the effect of the coefficients in the differential equations on the behaviour of their solutions. Then, depending on the values and signs of these coefficients, we try to detect and locate the layer regions of the solution, hence, obtain an initial mesh which is dense in those regions. A method of estimating the width of the layer region, which is needed in constructing the initial mesh, is then introduced. At the end, algorithms for detecting layers, locating and estimating their width and constructing the initial mesh are introduced.

The overall strategy is to use the initial mesh obtained by the algorithm described in this chapter as a starting mesh for one of the AMS strategies given in chapter 3. This can be regarded as a modification to the strategies of chapter 3. Many of the previous algorithms did not perform well because the initial mesh was so poor. The performance of any algorithm could be improved if it starts from a well structured non-uniform mesh which reflects to a certain extent the behaviour of the solution. With this we might get a converging estimated solution with a cheap strategy. This is the main

motivation for the analysis of this chapter.

In our work, we decided to use the De Boor strategy to continue the mesh selection process after an initial mesh has been found. The reasons for this are:

1. It is cheaper than the Q-matrix strategy, which makes it preferable from the cost point of view.

2. From the comparison of chapter 3, we found that De Boor strategy has performed, generally, better than the LR and LLSC strategies, and it cost is nearly the same. This makes it preferable over the LR and LLSC strategies.

The form of the problem dealt with in this chapter is:

$$L\epsilon = \epsilon y'' + f(x)\, y' + g(x) y = \eta(x) \text{ over } (a,b)$$

with

$$y(a) = A \; ; \quad y(b) = B \qquad (4.1) \; ;$$

where ϵ is the problem's parameter which is positive and may be small.

When $\epsilon \Longrightarrow 0$, formally neglecting y'' in (4.1) we get the following equation, which is refered to as 'the reduced equation':

$$f(x)\, y' + g(x) y = \eta(x) \qquad (4.2)$$

clearly a solution is obtained using either of the boundary conditions of (4.1).

This sort of problems could be either 'Regular' or 'Singular', the following definition given by Eckhaus[1973] distinguishes the two types:

Definition 4.1:-

Suppose that a general differential operator (not necssarily of form 4.1) can be written as:

$$\mathcal{L} = \mathcal{L}_0 + \epsilon\mathcal{L}_1 ;$$

where $\mathcal{L}_0$ and $\mathcal{L}_0$ are independant of ϵ then problem $\mathcal{L}y = \eta$ is said to be regular if there is a solution y_0 satisfying $\mathcal{L}_0 y_0 = \eta$ such that

$$\| y - y_0 \| \Longrightarrow 0 \text{ as } \epsilon \Longrightarrow 0.$$

If no such y_0 exist, then the problem is singular.
In our work, we dealt with singular problems only. This is because regular problems do not have any layer regions, which means that any simple uniform mesh could be used to solve them. In general, singular problems could have layers near the following points:

1. either of end points or both,in this case we have boundary layers.

2. A turning point, in this case we have an interior layer or turning point layer. A turning point is defined by Hemker[1977] as 'The point which is a zero of the function f(x) of 4.1'.

The proofs of theorems introduced in this chapter require the finding of an asymptotic solution of (4.1). Such a solution is defined as a solution of (4.1) which is obtained when $\epsilon \Longrightarrow 0$. Two methods for obtaining an asymptotic solution of (4.1), are described in the next two sub-sections. The detailed derivation of these methods will not be given here as they are very long and out of the scope of our work, however, references will be given for this.

4.1.1 The Matched Asymptotic (MA) Method

In this method, the solution is represented by an asymptotic series. Such series is defined by O'Malley[1974] as follows:

Definition 4.2

A sequence of functions$\{F_n(x,\epsilon)\}$is an asymptotic sequence if for $n > 0$, we have

$F_{n+1}(x,\epsilon) = o(F_n(x,\epsilon))$ as $\epsilon \Longrightarrow 0$; i.e.

$F_{n+1}(x,\epsilon)/F_n(x,\epsilon) \Longrightarrow 0$ as $\epsilon \Longrightarrow 0$. The asymptotic solution is then defined as follows:

Definition 4.3

The series

$$\sum_n A_n F_n(x,\epsilon)$$

is an asymptotic approximation to function $F(x,\epsilon)$ as $\epsilon \Longrightarrow 0$, with respect to the asymptotic sequence $\{F_n(x,\epsilon)\}$. If

$$F(x,\epsilon) = \sum_{n=0}^{N} A_n F_n(x,\epsilon) + o(F_N(x,\epsilon)) \text{ as } \epsilon \Longrightarrow 0;$$

F^* is written as

$$F^{*}(x,\epsilon) = \sum_{n=0}^{N} A_n F_n(x,\epsilon).$$

which is an approximation to $F(x,\epsilon)$.

For regular problems, an asymptotic approximation using one sequence is used over the whole range of the solution. For singular problems, we may need two or more sequences depending on the number of layers. For example, if we have a layer near the left end point, then, we need two sequences, one for the smooth part and the other for the layer part. The first is a sequence using the independant variable x, as in the definition above, the second is in a new variable t, which is usually taken as $t = x/\epsilon$ and defined in the layer region only somewhere a change of origin to the layer point has been assumed. The second sequence will give the following solution series:

$$F(t,\epsilon) = \sum_{n=0}^{N} B_n F_n(t,\epsilon) ;$$

where F_n here is different from that of definitions 4.2 and 4.3. The two solution are matched together to obtain the series coefficients, this is why it is called the Matched Asymptotic method. A full account of this method, how and at which points the matching occurs can be found in Eckhaus[1973].

4.1.2 The Wentzel-Kramer-Brillonin (WKB) Method

In this method, (e.g. Sirovich[1971]), the solution is represented in an exponential form, as follows:

$$y(t) = \exp[\ 1/\epsilon \int_0^t u(x)\ dx\ ]$$

with

$$u(x) = \sum_{n=0}^{\infty} Pn(x)\ \epsilon^n .$$

The analysis which follows concerns the homogeneous form of (4.1) where $\eta(x)= 0$. In the inhomogeneous case smoothness conditions are required on $\eta(x)$ are required in some cases to ensure the existance of an asymptotic solution, but generally the position and form of the layers is the same as for $\eta(x)=0$. Details of the precise conditions on $\eta(x)$ are given in Abrahamsson 1977. The illustrative examples considerd later confirm this. For singular problems, the solution is represented as a summation of local solutions, as

$$y = c1\ y1 + c2\ y2 \qquad (4.3)\ ;$$

where, c1, c2, are constants, y1 is a smooth solution and y2 is a layer solution . Sometimes however these solutions are not valid over the whole range and a third solution y3 must be introduced for part of the range and the solutions in the different parts must be matched to give a solution over the whole range.

Pearson[1968], derived y1 and y2, for problems with no turning point, by expanding u(x) we get:

$$y(x) = \{1/\epsilon \int_0^t [p0(x)+\epsilon p1(x)+ \epsilon^2 p2(x)+.....+\epsilon^n pn(x)]\ dx\} \quad (4.4);$$

He found y1 and y2 and substituted them in 4.1, then collected the terms of the same order of ϵ to get:

$$p0(p0 + f) = 0 ;$$

$$p'0 + 2\ p0\ p1 + f\ p1 + g = 0 ;$$

$$p'1 + p\ 1 + 2\ p0\ p2 + f\ p2 = 0 ;$$

etc.

By terminating this expansion with the p1 term, we get

$$p0(p0 + f) = 0 ;$$

$$-f\ p1 + g = 0;$$

which gives either $p0 = 0$ and $p1 = -g/f$, or $p0 = -f$ and $p1 = g/f$;

Substituting the values of p0 and p1 into 4.6 we get

$$y1 = \exp\left[-\int_0^t g(x)/f(x)\ dx\ \right] \text{ and}$$

$$y2 = 1/f(x)\ \exp\left[-1/\epsilon \int_0^t f(x)\ dx + \int_0^t g(x)/f(x)\ dx\right] \qquad (4.7)\ ;$$

For problems with a single turning point, he found y4 and y5, where y4 and y5 corresponds to y1 and y2 of (4.5):

$$y4 = x^{L} \exp\left[-\int_0^t (g(x)/f(x) + L/x\ )\ dx\ \right] \text{ and}$$

$$y5 = x^{-L-1}/f(x)\ \exp\left[-1/\epsilon \int_0^t f(x)\ dx + \int_0^t (g(x)/f(x)+L/x)dx\right] \qquad (4.8);$$

where $L = -g(0)/f'(0)$; assuming that the turning point is at $x = 0$. The derivation of WKB solution for the turning point problems is a complicated one, it requires the use of functions called the Parabolic Cylinder Functions which are related to Hermite polynomials. Another derivation for the WKB solutions is given in Sirovich[1971].

4.2 PROBLEM ANALYSIS

In this section, we introduce theorems used in locating the layer region. These theorems show how the coeffficients of the problems are used to predict the form of the solution. Problems are introduced to illustrate these theorems. The theory again assumes $\eta(x) = 0$.

The behaviour of the asymptotic solution of problem 4.1 depends on $f(x)$. This is because as $\epsilon \Longrightarrow 0$, the effect of the term $\epsilon y''$ becomes small, hence the term $f(x)\, y'$ would mainly affect the solution. This makes $f(x)$ very important in determining the form of the asymptotic solution. The function $f(x)$ could:

1. Have an isolated zero in the solution range (or more than one), the problem is then called a turning point problem.

2. Have no zero over the solution range.

3. Be zero throughout the range.

For these three cases different behaviour of the asymptotic solution and hence the solution occurs. Consequently the positions of layer regions would be affected by f(x). The case when f(x) has a zero is more difficult than the others as in this case, we may have an interior layer near the zero in addition to the boundary layers. We will study each of the three cases separately starting from the easy one, leaving the turning point case to the end.

4.2.1 Problems With no Turning Points and f(x) is Not Zero

This sort of problems have been studied by Pearson[1968]. Clearly, they have no interior layers as they have no turning points. The following theorem helps in locating these layers.

Theorem 4.1:-

For problem 4.1, assume that f(x) is not zero, and has no zeros in (a,b), then

1. If $f(x) > 0$, then the boundary layer would be at $x = a$;

2. If $f(x) < 0$, then the boundary layer would be at $x = b$;

Proof:-

For the case when f(x) has no zeros, the WKB solution in the layer region is given in Hemker[1977] as follows

$$y2 = C/f(x)\ \exp\{-\ 1/\epsilon \int_0^t f(x)\ dx\ \}$$

Thus

1. For case 1, f(x) is +ve, the exponential term will be -ve, which means that y2 decreases as we move from (a) toward (b). Thus, if there is any layer, it would occur near x = a as y2 is exponentially large there.

2. For case 2, f(x) is -ve, the exponential term will be +ve. This means that y2 increases as we move from (a) toward (b) and y2 is exponentially large near x = b, thus, if there is any layer it will be near x = b.

Pearson[1968], has also proved the results given in theorem 4.1. Grassman[1971] obtained the same results by using the MA solution, he worked on the range (0,1) and gave the following lemma:

Lemma 4.1

For the function $F(x,\epsilon)$ to satisfy the differential equation 4.1, a number M independant of ϵ exists such that:

1. $\left| F(x,\epsilon) - A \right| < M\,x$ if $f(x) < 0$;

2. $\left| F(x,\epsilon) - B \right| < M(1-x)$ if $f(x) > 0$;

where A and B are given in the boundary conditions.

Proof

Given in Grassman[1971].

From this lemma, we can prove that:

1. For $f(x) < 0$, putting $x = 0$ into relation 1 of lemma 4.1, we get

   $$|F(0,\epsilon) - A| = 0 ;$$

   this means that F satisfies the boundary conditions at $x = 0$ and no layer is expected near $x = 0$. Hence, if there is any layer it would be at $x = 1$.

2. For $f(x) > 0$, putting $x = 1$ in the second relation of lemma 4.1, we get

   $$|F(1,\epsilon) - B| = 0 ;$$

   for the same reason given in 1, a boundary layer could be at $x = 0$.

To illustrate the case with $f(x) < 0$ we use

Example 4.1:-

$\epsilon y'' - y' = 0$ over $(-1,1)$;

with the boundary conditions

$$y(-1) = 1,\ y(1) = 2;$$

with the exact solution

$$y = A + B \exp(x/\epsilon);$$ where

$A = 1 - \exp(-1/\epsilon)/(\exp(1/\epsilon) - \exp(-1/\epsilon))$;

$B = 1/(\exp(1/\epsilon) - \exp(-1/\epsilon))$;

with a boundary layer at x = 1, which agrees with theorem 4.1 as f(x) = -1 which is < 0.

If we change the sign of the y' term to +ve, we will get a problem which is a mirror image to the above problem, with boundary layer at x =-1. This agrees with theorem 4.1 for the case f(x)> 0. These two example are practical illustrations for theorem 4.1.

4.2.2 Problems where f(x) is Zero Over The range and g(x) <> 0

Sirovich[1971], derived the WKB solution for this sort of problem, Hemker[1977], gave the WKB solution as follows:

$$y(t) = (1/\ g(x))\ \exp(\pm \int_a^b \sqrt{-g(x)/\epsilon}\ \ dx) \qquad (4.9);$$

As we see from this equation, the form of the solution depends on g(x). The following theorem state where layers would be for these problems.

Theorem 4.2

For problem 4.1, assume that f(x) is zero over (a,b), then,

1. If g(x)< 0, then, we could have two boundary layers, one at each end point.

2. If g(x) > 0, then, the solution will be of oscillatory nature, with no boundary layers.

Proof:

1. From equation 4.9, we see that we have two solutions, one for the +ve case and the other for the -ve case. Thus

   a. The first solution has the following +ve dominant term, as ϵ ====> 0:

   $$\exp\left(+\int_a^t \sqrt{-g(x)/\epsilon}\ dx\right) ;$$

   The value of this term increases as we move from (a) toward (b), and becomes exponentially large near (b). Thus, a boundary layer occurs near x = b.

   b. The second solution has the following -ve dominant term, as ϵ ====> 0:

   $$\exp\left(-\int_a^t \sqrt{-g(x)/\epsilon}\ dx\right) ;$$

   The value of this term decreases as we move from (a) toward (b), it becomes exponentially large near x = a, which could give a layer near (a).

2. For g(x) > 0, the exponent value becomes imaginary (in the complex numbers sense). In this case, the exponential term can be transformed to SIN and COS terms, which means that the solution will be an oscillatory one. The frequency of oscillation depends on ϵ , for small ϵ the frequency becomes large and the problem becomes difficult to solve numerically.

An example of problems of this kind is

Example 4.2:-

$$\epsilon y'' - (2 - x^2)\, y = -1 \ ; \text{ over } (-1,1)$$

with the following boundary conditions

$$y(-1) = y(1) = 0;$$

with the exact solution

$$y(x) \simeq (1/(2 - x^2)) - \exp((1 + x)/\sqrt{\epsilon}) - \exp((1-x)/\sqrt{\epsilon}).$$

This problem has boundary layers at the two end points. It illustrates part 1 of theorem 4.2 as it has $f(x) = 0$ and $g(x) < 0$. The following problem illustrates part 2 of theorem 4.2, as it has an oscillatory solution.

Example 4.3:-

$$\epsilon y'' + y = 0; \text{ over } (-1,1) \ ;$$

with the boundary conditions

$$y(-1) = 1; \ y(1) = 2 \ ;$$

with the true solution

$$y(x) = A \sin(x/\epsilon) + B \cos(x/\epsilon) \ .$$

For this problem $g(x) > 0$, which gives an oscillatory solution.

4.2.3 Problems With One Turning Point

The solution of these problems could have an interior layer or/and boundary layers. Pearson[1968], was one of the earliest people to study these problems, he stated the following preliminary results, assuming that the turning point is at $x = 0$:

1. If $f'(0) > 0$, then, there is a turning point layer near $x = 0$, with no boundary layers at the end points.

2. If $f'(0) < 0$, then, there are no turning point layers, but, two boundary layers exist, one at each of the end points.

He proved these two results by studying the behaviour of the WKB solution for each case.

Here, we study these two cases in more details

1 - Case 1, $f'(0) < 0$:-

Ackerberg and O'Malley[1970], studied these problems. They introduced two new factors that could affect the location of layers, these are:

$$L = - g(0)/f'(0) \text{ ; and}$$

$$I = \int_a^b f(x)\, dx \text{ ;}$$

Then, they introduced the following results, with $L = 0,1,2,3,\ldots$:

1. If $I > 0$, then a boundary layer would be at $x = a$.

2. If $I < 0$, then a boundary layer would be at $x = b$.

3. If $I = 0$, then we have two boundary layers, one at each end point. For other L values, the results quoted from Pearson above hold.

The proof of these results is given in Ackerberg and O'Malley, it is done by showing that y4 of 4.7 satisfies, the right boundary condition when $I > 0$, the left boundary condition when $I < 0$ and neither of the boundary conditions when $I = 0$. Another proof for these results has also been given in Kreiss and Parter[1974]. Abrahamsson[1975] and [1977], studied the turning point problems in more detail. He studied, in addition to the case where $f'(0) < 0$, the cases when $f'(0) > 0$ and when a turning point coincides with a boundary point.

2- Case 2, $f'(0) > 0$:-

Hemker[1977], collected the results for this case, he stated the following :

When $f'(0) > 0$, we have no boundary layers, but, we have a turning point layer. The shape of this layer depends on L, as follows:

1. If $L = -1,-2,-3,.....$, then, the solutions may explode exponentially over the whole range.

2. If $L <> -1,-2,-3,....$, then

   a. If $L = 0$, then, according to Hemker's terminology, we have a Shock layer which occurs due to a sudden jump in the solution at the turning point.

   b. If $L > 0$, we have a Cusp or Corner layer. In this case, the

solution at the turning point is zero. This layer occurs as the two side solutions meet at the turning point.

c. If $L < 0$, the values of the solution in the vicinity of the turning point becomes unbounded, a complicated behaviour of the solution is expected here.

The following are examples for some of the cases we mentioned in this section.

1. Problems with $f'(0) > 0$;

a. Example 4.4:-

$$\epsilon y'' + x y' + y = 0; \text{ over } (-1,1) ;$$

with boundary conditions

$$y(-1) = y(1) = 0.5 ;$$

The exact solution is

$$y(t) = \exp(-x^2/2\epsilon)[A + B \int_0^t \exp(x^2/2\epsilon)\, dx] .$$

For this problem, $L = -1$, it has a turning point layer due to the meeting of the exponential solutions at each side of the turning point.

b. Example 4.5:-

$$\epsilon y'' + x y' = -\epsilon \pi^2 \cos(\pi x) - (\pi x) \sin(\pi x); \text{ over } (-1,1);$$

with boundary conditions

$y(-1) = -2,\ y(1) = 0$;

The exact solution for this problem is

$y(x) = \cos(\pi x) + \mathrm{erf}(x/\sqrt{2\epsilon})/\mathrm{erf}(1/\sqrt{2\epsilon})$;

For this problem L = 0, its solution has a shock layer.

c. Example 4.6:-

$\epsilon y'' + x y' - y = -(1 + \epsilon\pi^2)\cos(\pi x) - (\pi x)\sin(\pi x)$; over $(-1,1)$;

with boundary conditions

$y(-1) = 1$; $y(1) = 1$;

its exact solution is

$$y(x) = \cos(\pi x) + x + \frac{(x\ \mathrm{erf}(x/\sqrt{\epsilon}) + \sqrt{2\epsilon/\pi}\exp(-x^2/2\epsilon))}{\mathrm{erf}(1/\sqrt{2\epsilon}) + \sqrt{2\epsilon/\pi}\exp(-1/2\epsilon)}.$$

This problem has L = 1, it has a Cusp layer near the turning point .

2. Problems with $f'(0) < 0$,

a. Example 4.7:-

$\epsilon y'' - x y' = 0$; over $(-1,1)$

with boundary conditions

$y(-1) = 1$; $y(1) = 2$;

its exact solution is $y(t) = A + B\int_0^t \exp(x^2/2\epsilon)\,dx$.

For this problem. we have $I = \int_0^0 f(x)\,dx = \int_{-1}^{1} x\,dx = 0$. From the exact solution we see that it has two layers.

b. Example 4.7-a:-

The same problem taken over(0,1), we get $I = \int_0^1 x\,dx = 1/2$. From the exact solution, integrating it over (0,1) we get a boundary layer at the right end point.

c. Example 4.7-b:-

Also the same problem, but this time is taken over (-1,0), we get I = -1/2. The exact solution in this case has a layer near the left end point.

4.3 LAYER(S) LOCATING (LL) ALGORITHM

In this section we introduce an algorithm which is used in locating layer regions. This algorithm uses Hemker's results given in the previous section. A flow chart for the LL algorithm is given in figure 4.1. This algorithm has been designed to be used for problems with a single turning point only. It first looks at f(x) to find if it has a zero, which is done by looking for the sign change of f(x) evaluated on a number of points in the range (a,b). The points of evaluation are

k = a + i h; h =(a - b)/M; i = 0,1,2,...,M

If a turning point exist, a NAG Fortran procedure is called to obtain the zero of f(x).

For turning point problems,the LL algorithm requires an algorithm to evaluate or estimate $f'(x_o)$, whose x_o is the turning point.

Another algorithm is needed to evaluate $I = \int_a^b f(x)\,dx$. For the evaluation of $f'(x)$, accuracy is not very important, what is important is the signs of $f'(x)$, thus, any simple numerical differentiation method could be used. For the evaluation of I, accuracy becomes important when the result of evaluation is small, but not zero. In this case, we do not know whether to regard it a zero or a small value. To treat this, we use two integration methods, a simple one to be used first, then, if better accuracy is needed, a more complicated integration method, such as Adaptive Quadrature, is used.

After locating the layers, the LL algorithm returns the information about them to another algorithm that uses this information in obtaining the initial mesh. The information is stored in variables refered to them by the word 'indicator' in the flow chart.

4.4 LAYERS WIDTH ESTIMATION

From the WKB solutions given in equation 4.7 and 4.8, y2 and y5 are the layer solutions. Making a change of variable so that the origin is at the layer point and reversing the sign of the independent variable if necessary, the layer solution have the following dominant term:

$$\exp\{(-1/\epsilon) \int_0^t f(x)\,dx\} \qquad (4.9);$$

The cases of interest now are $f(x)>0$, or $f(0)=0$ and $f'(0)>0$. These terms approches zero outside the layer region.

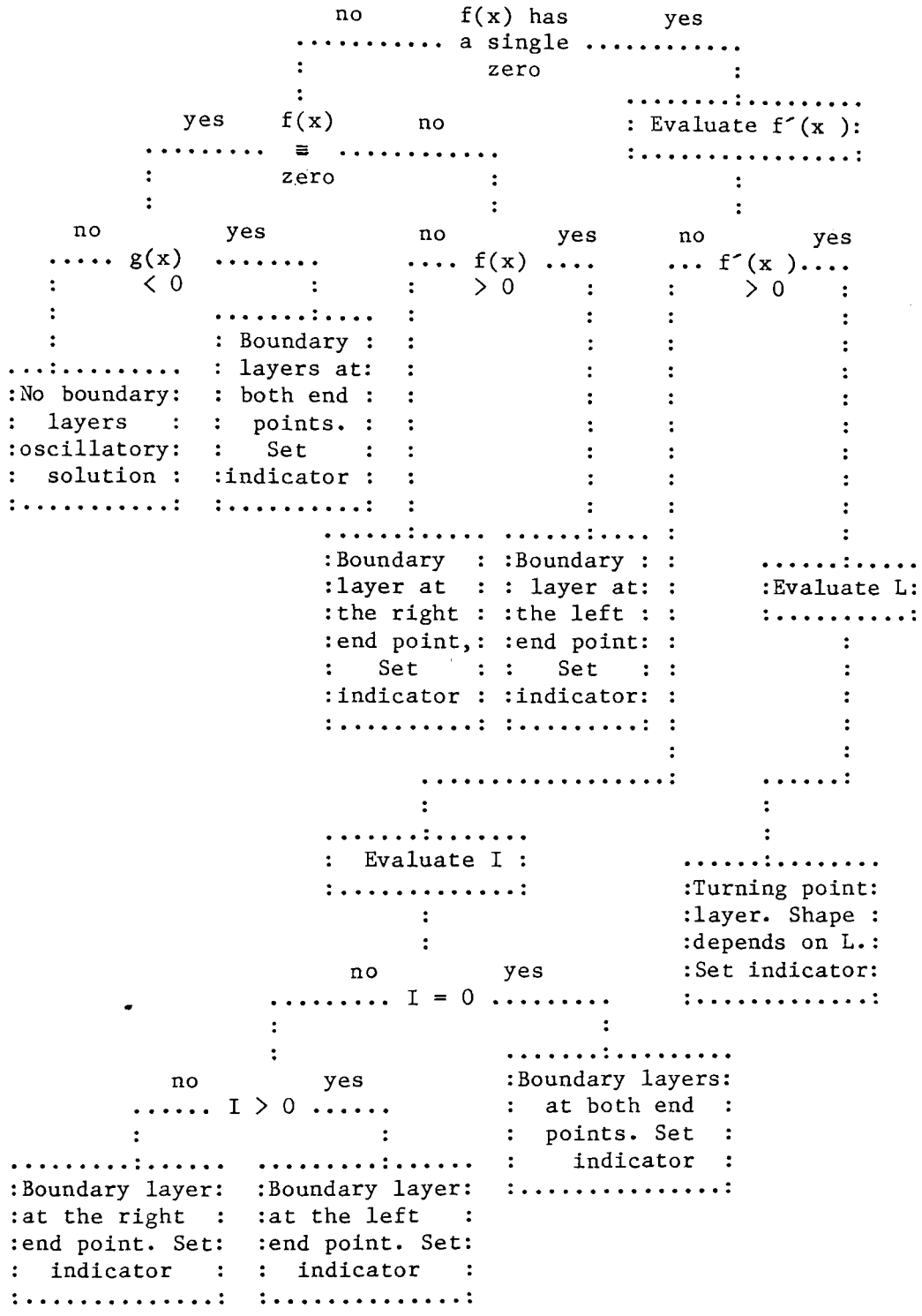


Figure 4.1

Flow Chart For LL Algorithm

Depending on this fact, the width of a layer is estimated by finding the point at which this term becomes smaller than a given tolerance. This condition is put as follows:

$$\exp\{(-1/\epsilon) \int_0^t f(x)\,dx \} < tol \qquad (4.10)$$

where tol is a prescribed tolerance.

Assume that, the point at which this condition is satisfied is xl and the boundary layer is at point a, then, the width W is

$$W = a - xl \ ;$$

To make condition 4.10 simple, we assume that

$$\int_0^t f(x)\,dx = -F(t)$$

then 4.10 becomes

$$\exp[(-1/\epsilon)\ F(t)\ ] < tol$$

take the log of both sides, we get

$$(-1/\epsilon)\ F(t) < \log(tol) \ ;$$

which is put as

$$|F(t)| > \ |\log(tol)|\,|\epsilon| \qquad (4.11);$$

Condition 4.11, is used in the following algorithm to find xl and hence W:

Algorithm W:

1. Start from xl = (a-b)/2 ; evaluate F(xl).

2. While condition 4.11 is not satisfied, do

   a. Find a new xl from the following relation

      xl = a - (a - xl)/2 ;

b. evaluate F(xl);

c. Go to 2

3. Find W, W = a - xl, stop.

This algorithm deals with the case when the boundary layer occurs near x = a. If it occurs near x = b, then -a- in steps 2 and 3 of algorithm W is replaced by b. For a turning point layer, -a- in all the steps of algorithm W is replaced by x0, the turning point.

There are two exceptional cases where condition 4.11 is not appropriate and another condition is used, these are:

1. When f(x) is zero over (a,b), the dominant term in the WKB solution which is responsible for layers will be as follows:

$$\exp(\pm \int_0^t \sqrt{-g(x)/\epsilon}\ dx\ )$$

This gives the following condition

$$|\ G(t)| < |\sqrt{\epsilon}\ \log(tol)|\ ; \qquad (4.12)$$

where

$$G(t) = \int_0^t \sqrt{-g(x)}\ dx\ .$$

For this case 4.12 is used instead of 4.11.

2. When the WKB solution at the turning point is zero, the case when $f'(0) > 0$ and $L > 0$. Condition 4.11 is reversed to become:

$$|F(t)| < |\epsilon \log(tol)|\ ; \qquad (4.13).$$

In the turning point layer, the estimated width obtained by this algorithm is actually half the layer width, it is the width at one side of the turning point.

In general, the width of boundary layers is taken to be of $O(\epsilon)$ for $f(x) \neq 0$ and of $O(\sqrt{\epsilon})$ for the case $f(x)$ is zero. A turning point layer is of $O(\sqrt{\epsilon})$. The width obtained by algorithm W also depends on either ϵ or $\sqrt{\epsilon}$, as conditions 4.11, 4.12 and 4.13 are all related either to ϵ or to $\sqrt{\epsilon}$. Actually, it can be said that algorithm W tries to find the constant which is to be multiplied by the order function to obtain the width.

4.5 MESH SELECTION ALGORITHM

In the previous two sections, we described algorithms that locate layers and estimate their widths. In this section, we give an algorithm that obtains an initial mesh and carries out the mesh selection process.

The initial mesh is obtained by using information from algorithm LL and algorithm W. From algorithm LL, we get the points near which there are layers, while algorithm W gives us the width of the layer. This information is used by an algorithm that obtains an initial mesh, this algorithm will put more mesh points in the layer regions obtained by Algorithm LL within a width found by Algorithm W. The distribution of mesh points within layer regions is non-uniform, we

actually distribute them in a logarithmic manner, which reflects the exponential behaviour of the WKB solution in layer regions. The mesh points within a layer are defined as follows:

Assume that we want to put p points in the layer, define

$$t_i = (p - i) \qquad i = 0,1,.....,p-1$$

The mesh points are then defined as

$$x_i = C \log(t_i) + C1 \ ;$$

where C and C1 are constants.

To determine C and C1, we assume that we have a layer at x = a, then, we have

$$x_{p-1} = a \qquad \text{and } x0 = X1 \ ;$$

where X1 is the point at which condition 4.11 is satisfied.

Thus for x_{p-1}, we have

$$a = C \log(1) + C1 \qquad \text{that is } C1 = a \ .$$

For x0, we have

$$X1 = C \log(p) + C1$$

this gives

$$C = (X1 - C1) \ / \ \log(p) = W \ / \ \log(p).$$

So the points may now be written as follows:

$$x_i = (W/\log(p)) \log(t_i) + a \ ; \qquad i = 0,1,.....,p-1 \quad (4.14).$$

With this distribution we get a mesh which has intervals whose size increases as we move away from the layer point. Practically, we found that using a large value of p does not improve the accuracy, however, p should not be taken very small. A reasonable value of p is found to be between 5 to 10, when no points outside the layers are used.

A few points will be put in the smooth parts of the solution, the number of these points does not seem very important. As the De Boor algorithm, used after the initial mesh has been constructed, would put more points in those parts if it is required.

Finally, the AMS algorithm is put as follows:

1. Use the LL algorithm to locate layer regions.

2. Use algorithm W to estimate the width.

3. Construct the non-uniform initial mesh as described above.

4. Use the De Boor algorithm given in chapter 3 to continue the mesh selection process.

We will refer to this algorithm as the 'LLWD' algorithm, as it combines the LL algorithm, algorithm W and De Boor algorithm. In the next section, we test the LL algorithm , algorithm W and illustrate the location of layers and estimation of their width. In chapter 6 algorithm LLWD will be tested and compared with other algorithms.

4.6 TESTING THE LL ALGORITHM AND ALGORITHM W

In this section, we use some of the illustrative examples, given throughout this chapter, to test the LL algorithm and algorithm W. The results are put in form of table, see tables 4.1 and 4.2, the tables contain the following fields:

1. A field that contains the problem number, here we use the example number.

2. A field contains values of the problem's parameter.

3. A field contains information about the layer positions.

4. A field contains the values of the end points (within layers) obtained by the algorithm.

5. A field contains values of the estimated width.

6. The last field contains the figure numbers for the corresponding meshes.

Note that for all problems, the algorithms puts 5 mesh points in each layer, so if we have two boundary layers or a turning point layer, we would expect to have a total number of 10 mesh points.

Tolerance is = 1E-8.

Problem number	Problem's parameter	Layers at	End points	Estimated width	Figure: number:
			1.0000000		
		Right	9.783E-01		
4.1	1E-3	end	9.950E-01	1.562E-02	4.1
		point	9.911E-01		
			9.843E-01		
			-1.0000000		
		Two	-9.994E-01		
4.2	1E-7	end	-9.987E-01	3.906E-03	4.2
		points	-9.977E-01		
			-9.960E-01		
			8.239E-17		
			5.415E-04		
4.5	1E-6	Turning	1.239E-03	3.906E-03	4.3
		point	2.223E-03		
			3.906E-03		

Table 4.1

Tolerance is = 1E-10.

Problem number	Problem's parameter	Layer at	End points	Estimated width	Figure number
			-1.0000000		
		Two	-9.997E-01		
4.7	1E-4	end	-9.993E-01	1.953E-03	4.4
		points	-9.988E-01		
			-9.980E-01		
			1.0000000		
		Right	9.997E-01		
4.7-a	1E-4	end	9.993E-01	1.953E-03	4.5
		point	9.988E-01		
			9.980E-01		
			-1.0000000		
		Left	-9.997E-01		
4.7-b	1E-4	end	-9.993E-01	1.953E-03	4.6
		point	-9.988E-01		
			-9.980E-01		

Table 4.2

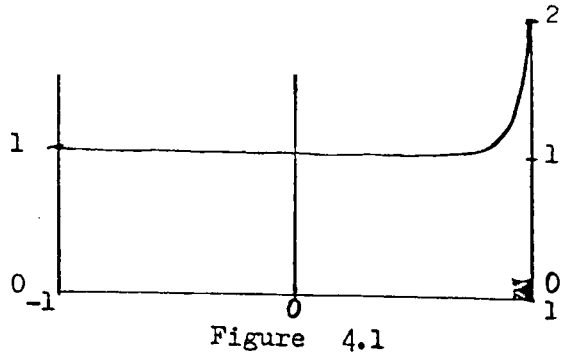


Figure 4.1

$\epsilon y'' - y' = 0$, over$(-1,1)$

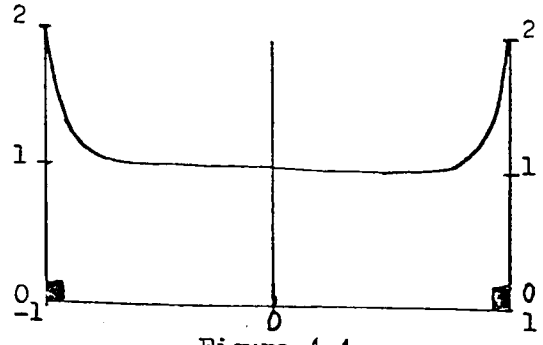


Figure 4.4

$\epsilon y'' - xy' = 0$, over$(-1,1)$

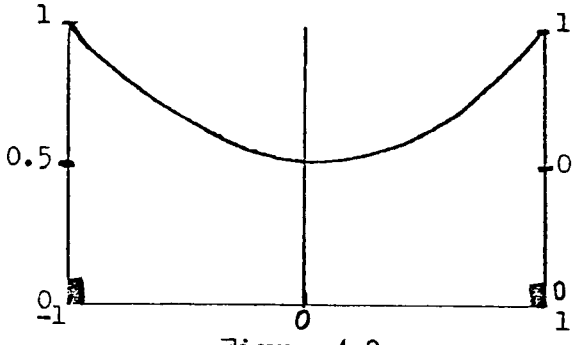


Figure 4.2

$\epsilon y'' - (2-x^2)y = -1$, over$(-1,1)$

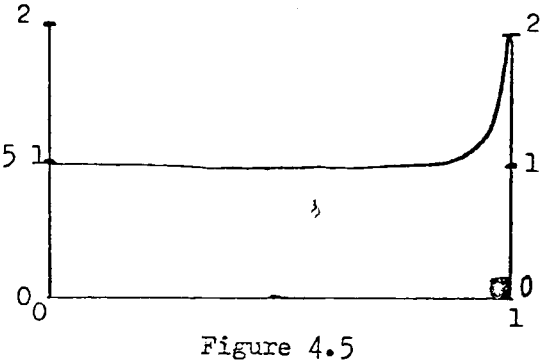


Figure 4.5

$\epsilon y'' - xy' = 0$, over$(0,1)$

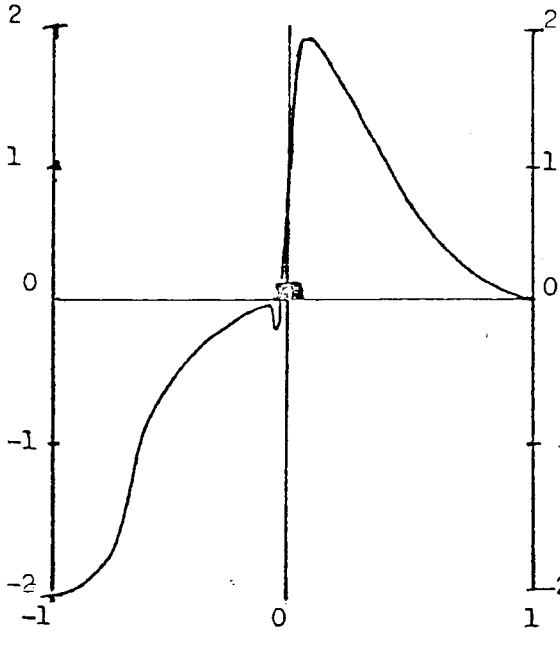


Figure 4.3

$\epsilon y'' - xy' = -\epsilon\pi^2\cos(\pi x) - \pi x\sin(\pi x)$

over$(-1,1)$

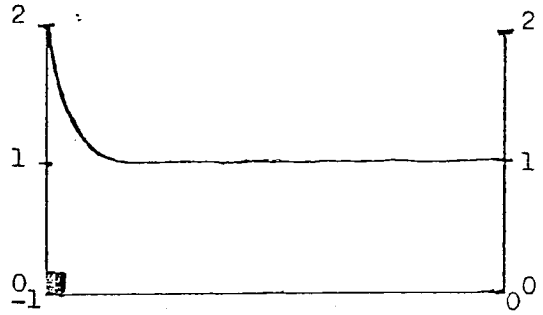


Figure 4.6

$\epsilon y'' - xy' = 0$, over$(-1,0)$

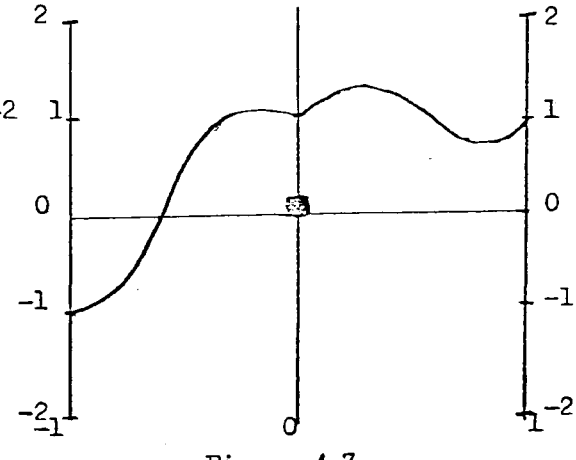


Figure 4.7

$\epsilon y'' - xy' - y = -(1+\epsilon\pi^2)\cos(\pi x) - (\pi x)\sin(\pi x)$ over$(-1,1)$

Note that problems 4.2 and 4.7 have two boundary layers, in the tables we gave the values of the mesh points inside the left layer only as the mesh points values in the right layer are the same but with an opposite sign. For problems with a turning point, such as problem 4.3, the values of the end points shown in the table are the points inside the right half of the layer only. The values inside the left half are obtained by only reversing the signs of the values given in the table. From tables 4.1, 4.2 and the figures we conclude

1. In all figures, layer regions appear as a thick band, as their width is very small relative to the (-1,1) range. This makes it difficult to see how the mesh points are distributed within the layers. However, the values of the end points given in the tables indicate that the distribution is done according to the algorithm of section 4.5.

2. From the graphs, the layer position is found easily, as they appear as bands. In all the cases, the positions shown in the graphs agree with the information given in the tables under the field 'Layer at'. This clearly indicates that algorithm LL is performing well.

3. The values of the estimated width given in the tables are of $O(\epsilon)$ for problems 4.1, 4.7, 4.7-a and 4.7-b, and of $O(\sqrt{\epsilon})$ for problems 4.2 and 4.3. These estimated values agree with what we stated before.

CHAPTER FIVE

THE MODIFIED Q-MATRIX STRATEGY

5.1 INTRODUCTION

In this chapter, we describe how to approximate the kernel of $(Dm - T)^{-1}$ and the residual function by Chebyshev series. Then, combining the Chebyshev coefficients, we obtain an error estimate which is used as a criterion for the modified strategy.

The error is related to the residual by

$$e(s) = (Dm - T)^{-1} r(s) \qquad (5.1) ;$$

where Dm and T are operators defined in 1.2, 1.3 and r(s) is the residual function. Assuming that k(s,t) is a kernel for the integral operator $(Dm - T)^{-1}$, the error of 5.1 becomes

$$e(s) = \int_{-1}^{1} k(s,t)\, r(t)\, dt \qquad (5.2) .$$

If k(s,t) and r(t) are approximated by Chebyshev series then a corresponding estimate of e(s) would be relatively easy to evaluate.

The motivation for this modification of the Q-matrix strategy is that using a proper estimate of error as criterion is more appropriate than using an estimation of an error bound as in the original Q-matrix strategy.

In the next section, we show how to relate the kernel to the Q-matrix, and how to represent it by a Chebyshev series; we also show how to represent the residual. In section (5.3), we show how to carry out the numerical integration (5.2). In section (5.4), an algorithm based on the modified strategy is introduced and tested to show that it produces suitable meshes; a case where this algorithm fails is also investigated. In the last section, some algorithms that deal with the failure case are proposed and tested, then, the best one is chosen to be used with the original algorithm to form a new algorithm.

5.2 THEORETICAL FOUNDATION

Ahmed[1981], showed under suitable conditions that:

$$\| Q \| \Longrightarrow \| (Dm - T)^{-1} \|$$

and

$$\| Q - Z \| \Longrightarrow 0 \qquad \text{as } P \Longrightarrow \infty ;$$

with

$$Z = \Phi (Dm - T)^{-1} \Psi$$

where Φ is an evaluation operator ($\Phi : \chi \Longrightarrow R^{np}$) which gives a vector consisting of the values of the function at the collocation points and Ψ is an extension operator ($\Psi : R^{np} \Longrightarrow \gamma$) which is a piecewise constant function whose values at the collocation points agree with component of the vector. Both are defined in detail by Gerrard and Wright[1984].

The results show that the elements of the Q matix are related to the operator $(Dm - T)^{-1}$, and suggest that an approximation to the kernel k(s,t) of $(Dm - T)^{-1}$ might be constructed fom the Q matrix This could then be used with (5.2) to estimate the error.

The elements of Z are related directly to k(s,t) by

$$Z_{ij} = \int_{-1}^{1} k(t_i,t)\ (\bar{\Psi} \underline{e}_j)(t)\ dt \qquad (5.3)$$

where $\underline{e}_j$ is the jth unit vector and $\{t_i\}$ are the collocation points. The extention $\bar{\Psi}$ is chosen so that $\bar{\Psi}\,\underline{e}_j$ is piecewise 1 or 0 and such that

$$Z_{ij} = \int_{\xi_j}^{\xi_{j+1}} k(t_i,t)\ dt$$

where $\{\xi_j\}$ are some points defined in Gerrard and Wright [1984] satisfying

$$t_{j-1} <= \xi_j <= t_j <= \xi_{j+1} <= t_{j+1} .$$

and $\xi_{j+1} - \xi_j = W_j$ the usual piecewise polynomial interpolatory quadature weight based on the collocation points. Hence for k(s,t) sufficiently smooth we have

$$Z_{ij} \simeq W_j\, k(t_i,t_j).$$

The relationship between Q and Z suggest using the further approximation that $Q_{ij} \simeq Z_{ij}$, so that we could estimate $k(t_i,t_j)$ by

$$k(t_i,t_j) = Q_{ij}/W_j \qquad (5.4)$$

Now some form of interpolation may be used to construct a function $\hat{q}(s,t)$ which agrees with these estimated values at the points (t_i,t_j).

The details of how this interpolation are carried out are described

in detail below. Since only one estimate is required for each interval and only the relative size of the estimates in different intervals are required the approximation is simplified further. For smooth problems at least it is observed that the variation with s is slow and this suggests using an averging process to produce a further approximation $\bar{q}(s,t)$ where the function of s is constant on each interval. This also reduces the amount of work involved.

By substituting (5.4) into (5.2), we get

$$\bar{e}(s) = \int_{-1}^{1} \bar{q}\,(s,t)\; r(t)\; dt \qquad (5.5)\;;$$

Practically this transformation is done by averaging the rows of the Q matrix, in each of its blocks, this averaging is equivalent to the condensing operation described in section 3.5.1. Now the values of $\bar{q}(s,t_j)$ in block (k,1) of the Q matrix are

$$\bar{q}(s,t_j) = \sum_{i}^{n} Q_{klij} \,/\, W_j = Q^*_{klj} \qquad (5.6)\;;$$

where Q^* is the condensed version of Q.
The values obtained from 5.6 are regarded as estimated values of the kernel evaluated at t_j. These values are used in obtaining the Chebyshev representation, $\bar{q}(s,t)$.

The Chebyshev representation for the residual r(t), is obtained by evaluating the residual at the Chebyshev extrema, then using these as values of $r(t_j)$ at the interpolation points, t_j. The method for evaluating $r(t_j)$ is described in section 3.4.

With these two representations, the error estimation for interval -i- is given as

$$\bar{e}_i \simeq \sum_{j=1}^{p} [\ \bar{q}(s,t)\ ]_j\ r_j(t) \qquad (5.7)\ ;$$

Here, the error estimation is represented as a multiplication of a kernel representation in block(i,j) by the residual of interval j.

5.3 NUMERICAL EVALUATION OF THE ERROR ESTIMATE

Before we show how to evaluate 5.7 numerically, we will show how to get $\bar{q}(s,t)$ and r(t).

5.3.1 Chebyshev Representation For The Residual

The residual is represented by Chebyshev series as follows:

$$r(t) = Tn(t) \sum_{d=0}^{n\ '} a_d T_d(t) \qquad (5.8)\ ;$$

where n is number of collocation points and $T_d(t)$ is the Chebyshev polynomial of order d.

From the orthogonality properties of Chebyshev polynomials, the coefficients a_d are given by :

$$a_d = (2/n)\sum_{v=o}^{n}[r(s_v)/T_n(s_v)]\cos(\pi d\, v/n) \qquad (5.9);$$

Where the s_v points are the extrema of T_n .

Since

$T_n(s_v) = (-1)^v$ then

$$a_d = (2/n)\sum_{v=o}^{n}(-1)^v\, r(s_v)\cos(\pi v\, d\, /n\,) \qquad (5.10)\ ;$$

With this equation, we obtain the Chebyshev coefficients that are used in evaluating (5.8).

5.3.2 Chebyshev Representation For The Kernel estimation

For this representation, the Q matrix elements are used to approximate values of the kernel function evaluated at the interpolation points, Chebyshev zeros. Since this matrix is a full block one, we will get, in each block, a different Chebyshev series. Thus, for block(i,j), the Chebyshev representation for $\bar{q}(s,t)$ is found as follows:

$$\bar{q}(s,t) = \sum_{z=0}^{n-1} b_z T_z(t) \qquad (5.11);$$

Using the orthogonality properties of Chebyshev polynomials, the coefficients b_z are given by,

$$b_z = (2/(n+1))\sum_{p=0}^{n-1} Q^*_{ij\,p}\ Tp(t_z) \qquad (5.12);$$

where $Q^*_{ij\,p}$ are given in 5.6, and t_z are zeros of Chebyshev polynomial of order n.

For simplicity, we will drop the subscripts that refer to intervals or blocks for which the coefficients belong, when -a- and -b- are mentioned.

5.3.3 Evaluation of The Integral in (5.7)

By substituting r(t) of (5.8) and q(s,t) of (5.11) into (5.7) we get

$$\tilde{e}_i = \sum_{j=1}^{P} \int_{-1}^{1} \Big(\sum_{r=0}^{n-1} \sum_{s=0}^{n}{}' b_r a_s T_n(t) T_r(t) T_s t) \Big) dt \quad i=1...p \qquad (5.15)$$

which is put as

$$\tilde{e}_i = \sum_{j=1}^{P} \Big(\sum_{r=0}^{n-1} \sum_{s=0}^{n}{}' b_r a_s \int_{-1}^{1} T_n(t) T_r(t) T_s(t) dt \Big) \qquad (5.16)$$

Before we simplify 5.16 any further, we introduce the following properties of Chebyshev series, given e.g. in Fox and Parker[1966], which help in simplifying 5.16, these are, assuming that $n > r$:

Property 1 :-

$$\int_{-1}^{1} T_k(t)\, dt = 0 \qquad \text{for odd } k.$$

Property 2 :-

$$\int_{-1}^{1} T_r(t)\, dt = -2/(r^2 - 1) \quad \text{for } r \text{ even.}$$

Property 3 :-

$$T_r(t)\, T_n(t) = 1/2[T_{n+r}(t) + T_{n-r}(t)] \qquad (5.17);$$

By putting $T_r(t)\, T_n(t)\, T_s(t)$ of (5.16) as

$$[T_n(t) T_r(t)]\, T_s(t)$$

and from property 3 we get

$$T_n(t)\,T_r(t)\,T_s(t) = 1/2[T_{n+r}(t) + T_{n-r}(t)]\,T_s(t)$$

By applying property 3 again to this equation, we get

$$T_n(t)\,T_r(t)\,T_s(t) = 1/4[T_{n+r+s}(t)+T_{n+r-s}(t)+T_{n-r+s}(t)+T_{n-r-s}(t)]$$

now the integral part of 5.16 is

$$\int_{-1}^{1} T_n(t)\,T_r(t)\,T_s(t) = 1/4\Big[\int_{-1}^{1} T_{n+r+s}(t) + \int_{-1}^{1} T_{n-r+s}(t) + \int_{-1}^{1} T_{n+r-s}(t) + \int_{-1}^{1} T_{n-r-s}(t)\Big]dt$$

According to property 1, the value of this integration is zero when r+s+n is odd. By using property 2, this integration gives the following:

$$\int_{-1}^{1} T_n(t)\,T_r(t)\,T_s(t) = 1/4\,[-2/((n+r+s)^2-1)+-2/((n+r-s)^2-1) +-2/((n-r+s)^2-1)+-2/((n-r-s)^2-1)] \quad (5.18)$$

(for even n+r+s).

This equation shows how easy the numerical evaluation of the integration is. If we substitute 5.18 into 5.16 we get, for n+r+s even:

$$\bar{e}_i = \sum_{j=1}^{P}\Big[-1/2\sum_{r=0}^{n-1}\sum_{s=0}^{n'} b_r\,a_s\,(1/((n+r+s)^2-1) + 1/((n+r-s)^2-1) + 1/((n-r+s)^2-1) + 1/((n-r-s)^2-1))\Big] \quad (5.19)$$

This equation is used to estimate the error in each interval to obtain the criterion of the modified strategy. As we see, this evaluation is not, as one may have expected, complicated. It requires only a multiplication of the coefficients then a summation.

5.4 THE AMS ALGORITHM AND ITS TESTS

The algorithm for the modified strategy, call it Algorithm I, is similar to the Q-matrix algorithm given in section 3.5 in the way the subdivided interval is selected. It looks for the interval that gives the largest contribution of error to the interval of largest error, however, a different method is used to obtain the criterion. A summary of the algorithm for this method is given below.

1. Construct and condense the Q matrix as described in section 3.5.

2. Evaluate the Chebyshev coefficients for the polynomials that approximate the kernel and the residual, as given in equation 5.10 and 5.12.

3. In each interval, evaluate the error estimate $\bar{e}_i$ as given in (5.19) and store the value of the largest contribution with the corresponding interval.

From the comparison of chapter 3, we found that all AMS strategies given there, apart from the Q-matrix strategy, have failed to produce a suitable mesh for problem 3. Thus, we decided to test algorithm I with this problem. Algorithm I has been tested for some simple examples and performed well. The details of these tests are not included here, as we concentrate on the difficult cases only.

The first test is done by using four collocation points, n = 4, starting with an initial mesh of 5 equal intervals. The mesh obtained by this test is shown in figure 5.1, looking at this figure, we see that the mesh points, which are introduced, are concentrated in the middle of the solution region, which is sensible as problem 3 has a singular point at 0.

The second test is done with n = 3, starting with an initial mesh of 5 equal intervals. Figure 5.2, shows the mesh obtained by this test. Looking at this figure, we see that the mesh points are concentrated around one of the end points, while they should be concentrated in the middle. This means that this mesh is not a suitable one, hence indicates that algorithm I has failed and we will refer to this as a ´failure case´.

When we did more tests on algorithm I, we found that it performed well when n is even and fails when n is odd. The reason for failure is that problem 3 has a singular point at 0, starting from an initial mesh with 5 equal intervals would cause the singular point to lie

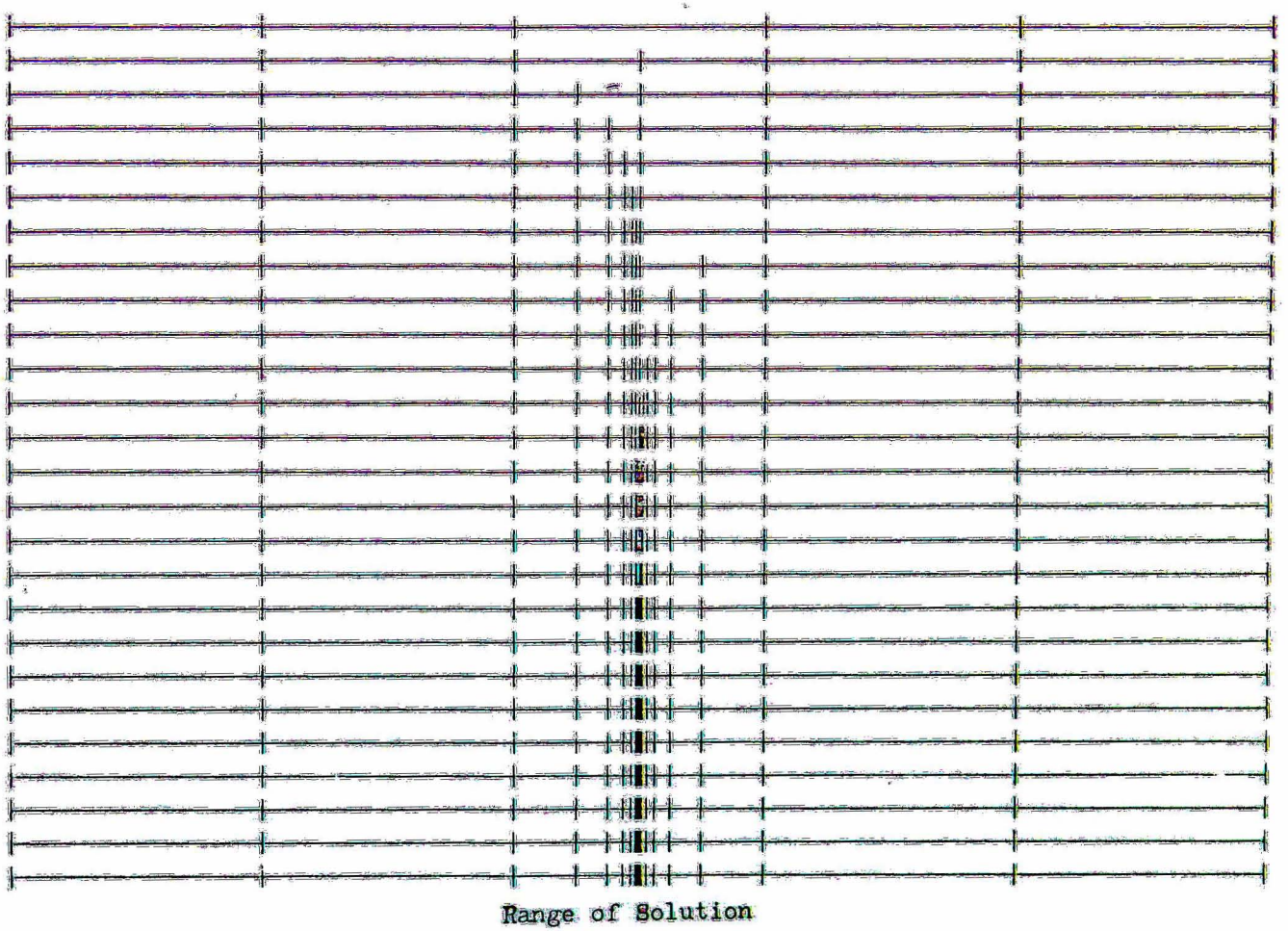


Figure 5.I, Mesh distribution for problem 3, Using Algorithm I with n = 4, P = 5.

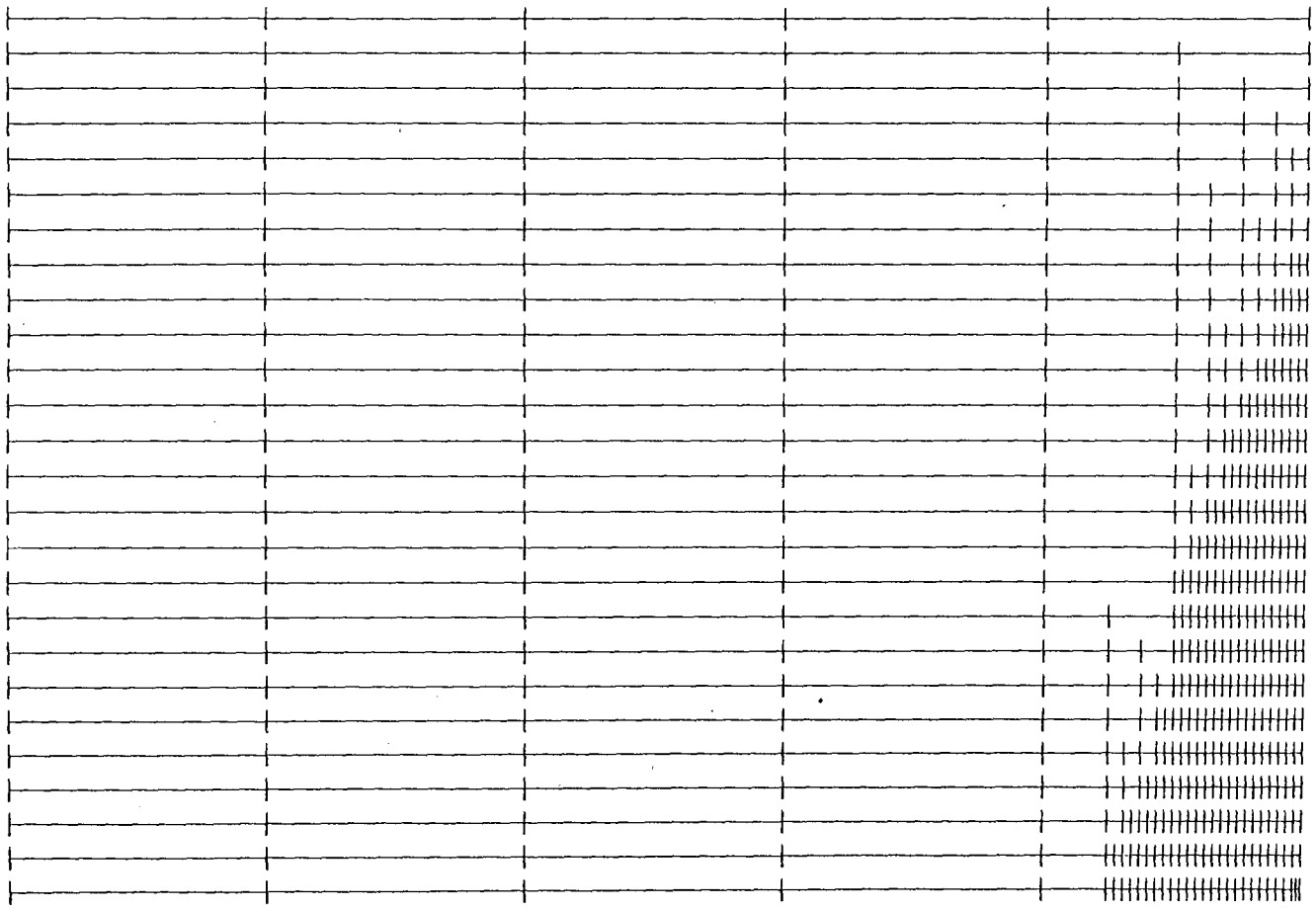


Figure 5.2, Mesh distribution for problem 3, Using Algorithm I, n = 3, P = 5.

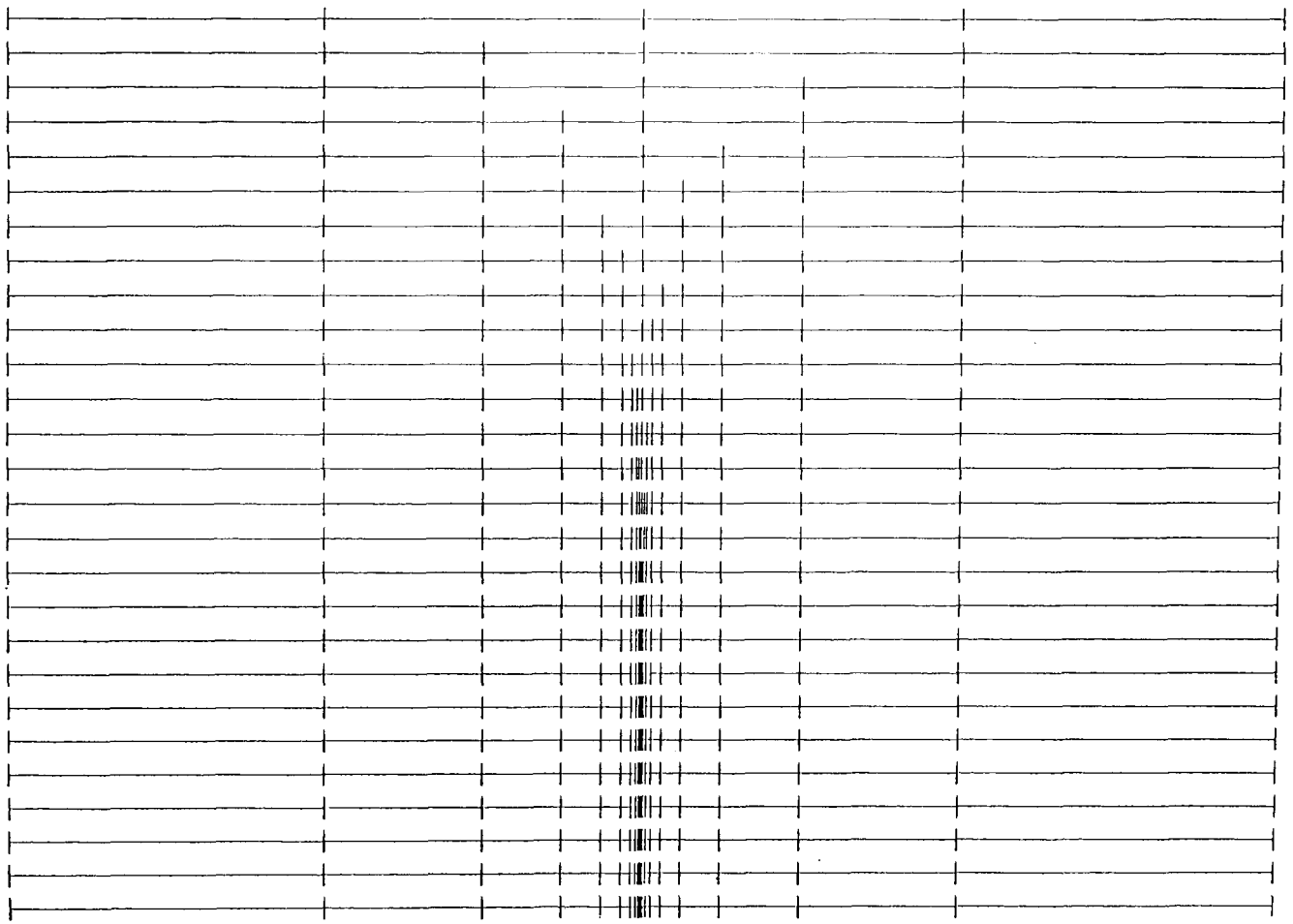


Figure 5.3, Mesh distribution for problem 3, Using Algorithm I, n=3, P = 4

exactly in the middle of the third interval, which leads to a some cancellation in evaluating the error contribution for that interval as it is shown in the next section.

To illustrate this, we did a 3rd test in which we used also an odd number of collocation points, 3, but started with an initial mesh of 4 equal intervals which means that the singular point is at an end point. Figure 5.3, shows the mesh for this test, looking at this figure, we see that the mesh produced is a suitable one, as it is dense near the singular point. This indicates that algorithm I performs well for this test.

5.4.1 Characteristics of The Chebyshev Coefficients When The Singular Point is in the Middle of an Interval

Tables 5.1, 5.2, 5.3, 5.4 and 5.5 shows the Chebyshev coefficients for polynomials approximating the residual and the kernel for problem 3, using a mesh of 5 equal intervals with n = 3, 4, 5, 6 and 7 respectively. All these tables have been put in the form of a block matrix. For example, a set of values in a block of 5.1 are coefficients of a polynomial that approximate the kernel using the Q-matrix values of the corresponding block. The coefficients in each block have been put in order, i.e, the top one is the first coefficient the next one is the second coefficient and so on. In the last column of table 5.1 we have a vector which is divided into a set of values, each set represents coefficients of a polynomial that

					7.182E+06
-3.209E-07	-5.386E-07	-2.248E-03	-9.729E-08	-4.354E-08	-9.012E+05
5.827E-07	-4.671E-07	-1.349E-06	5.188E-08	1.105E-08	1.326E+02
2.416E-07	-7.916E-07	4.498E-03	-1.297E-08	-1.381E-09	9.966E+01
					-3.591E+06
-1.205E-06	-6.880E-07	2.248E-03	9.730E-08	4.354E-08	8.954E+05
-3.061E-07	1.072E-06	1.349E-06	-5.189E-08	-1.105E-08	7.669E+02
-3.825E-08	6.056E-07	-4.498E-03	-1.730E-07	1.381E-08	-1.027E+01
					1.799E+06
-5.809E-07	-1.298E-06	-2.251E-03	-1.298E-06	-5.809E-07	-1.419E+02
-1.475E-07	-6.922E-07	-3.429E-18	6.922E-07	1.475E-07	6.402E-10
-1.843E-08	-1.730E-07	4.499E-03	-1.730E-07	-1.843E-08	-9.434E+01
					-3.604E+06
4.354E-08	9.730E-08	2.248E-03	-6.880E-07	-1.205E-06	-9.036E+05
1.105E-08	5.189E-08	-1.349E-06	-1.072E-06	3.061E-07	-7.669E+02
1.381E-09	1.297E-08	-4.498E-03	-7.916E-07	-3.825E-08	-1.027E+01
					7.208E+06
-4.354E-08	-9.729E-08	-2.248E-03	-5.386E-07	-3.209E-07	8.976E+05
-1.105E-08	-5.188E-08	1.349E-06	4.671E-07	-5.827E-07	-1.326E+02
-1.381E-09	-1.297E-08	4.498E-03	-7.916E-07	2.416E-07	9.966E+01

The Chebyshev Coefficients of q

n = 3 , P =5

Table 5.1

approximate the residual in the corresponding interval.

Looking at values of coefficients in these tables, we see that the set of coefficients in the 3rd column of blocks, this corresponds to the 3rd interval which has the singular point in its middle, has some properties that differ from those of other columns. The properties

are:

Property 1 :-

For an odd n, tables 5.1, 5.3 and 5.5, in all blocks of the 3rd column, the even labelled coefficients, b0, b2, b4 etc. (note that the value of the first coefficients printed here is half the actual value), behave as follows:

-b2 is slightly greater than bo, b4 is slightly greater than b2 etc. and

bo, b2, b4, etc. $\gg$ bl, b3, b5 etc. For an even n, tables 5.2 and 5.4, only the coefficients in block(3,3) have this property, the rest of the blocks have coefficients with nearly the same magnitude.

Property 2 :-

For odd and even values of n, we have in block(3,3), the block that corresponds to block(3,3) of the Q-matrix:

b0, b2, b4 etc. $\gg$ bl, b3, b5 etc.

These two properties could indicate that the approximation of the Kernel in the 3rd interval is not accurate which could lead to inaccurate error evaluation in that interval. They also show that for the odd cases, in all blocks of the 3rd column the kernel has been represented by a symmetric function. While for the even cases it is symmetric in block(3,3) only and not symmetric in all other blocks.

The residual coefficients in the 3rd interval also differ from those in other intervals. In this interval, a0 is dominant when n is odd while al is dominant when n is even. This means that for n odd,

for example n =3, the residual is approximately $a_0 T_3$ while in the even case, n = 4, the polynomial is $a_1 T_1 T_4$, both of them are anti-symmetric.

The error contribution of interval -j- to interval -i- is obtained by multiplying the kernel polynomial of block(i,j) by the residual polynomial of interval j. Thus, the contribution of the 3rd interval to any interval, in the odd cases, is obtained by multiplying an anti-symmetric polynomial (the residual) by a symmetric one (the kernel). The result will be an anti-symmetric function which when integrated could cause a cancellation that makes its value very small. In the even case, the kernel polynomial is symmetric in block(3,3) only and anti symmetric every where else, the residual polynomial is not symmetric. Thus, a cancellation would occur only in evaluating the contribution of the 3rd interval to itself. The evaluation of the contributions of the 3rd interval to the other intervals will result in a not anti-symmetric function with no cancellation.

To show how this happens, we give the following detailed examination of the error estimate evaluation for n odd and even. For the odd case, we recall equation (5.16), and substitute n by 3, we get in each interval:

$$\int_{-1}^{1} (1/2\ b_0 T_0 + b_1 T_1 + b_2 T_2)\ T_3\ (1/2\ a_0 T_0 + a_1 T_1 + a_2 T_2 + a_3 T_3)\ dx$$

by expanding this and neglecting the terms with r+s+n is odd, we get:

					-4.800E+07
-8.791E-07	2.783E-07	4.569E-07	-6.457E-07	-2.444E-07	6.000E+06
2.807E-07	-7.158E-07	-1.828E-06	7.540E-07	1.266E-08	6.644E+01
-1.675E-07	-2.211E-07	1.370E-06	-3.534E-07	-6.680E-08	-9.765E-01
-1.580E-07	-5.351E-07	1.463E-06	2.940E-07	4.265E-08	-2.953E-01
					-2.401E+07
-1.710E-06	-1.194E-06	1.370E-06	-1.937E-06	-7.333E-07	6.003E+06
-8.860E-07	-1.121E-06	-5.485E-06	2.262E-06	3.798E-07	-6.848E+00
-4.680E-07	-7.861E-07	4.111E-06	-1.060E-06	-2.006E-07	-3.319E+00
-2.984E-07	-1.293E-06	4.391E-06	8.822E-07	1.279E-07	4.730E-02
					-1.261E+02
-1.222E-06	-3.227E-06	-1.172E-06	-3.227E-06	-1.222E-06	6.007E+06
-6.329E-07	-3.769E-06	-9.952E-21	3.769E-06	6.329E-07	-6.289E+01
-3.353E-07	-1.776E-06	5.481E-06	-1.766E-06	-3.343E-07	4.191E-09
-2.132E-07	-1.469E-06	1.736E-20	1.469E-06	2.132E-07	2.971E-01
					2.400E+07
-7.333E-07	-1.937E-06	1.370E-06	-1.194E-06	-1.710E-06	6.002E+06
-3.798E-07	-2.262E-06	5.485E-06	1.121E-06	8.860E-07	-6.848E+00
-2.006E-07	-1.060E-06	4.111E-06	-7.861E-07	-4.680E-07	3.319E+00
-1.279E-07	-8.822E-07	-4.391E-06	1.293E-06	2.984E-07	4.730E-02
					4.800E+07
-2.444E-07	-6.457E-07	4.569E-07	2.783E-07	-8.791E-07	6.001E+06
-1.266E-07	-7.540E-07	1.828E-06	7.158E-07	-2.807E-07	6.644E+01
-6.688E-08	-3.534E-07	1.370E-06	-2.211E-07	-1.675E-07	9.765E-01
-4.265E-08	-2.940E-07	1.463E-06	5.351E-07	1.580E-07	-2.953E-01

Chebyshev Coefficients of q

n = 4 , P = 5

Table 5.2

					-1.301E+07
-1.253E-06	-1.062E-06	1.357E-03	9.787E-08	4.377E-08	1.627E+06
1.3293-06	-9.536E-06	-1.138E-06	-5.245E-08	-1.111E-08	-1.223E+00
5.449E-07	-1.691E-06	-2.711E-03	1.405E-08	1.411E-09	-1.846E-01
4.441E-07	-6.435E-07	-1.444E-06	-3.746E-09	-1.791E-10	2.636E-03
2.363E-07	-8.837E-07	2.713E-03	9.374E-10	2.178E-11	4.886E-04
					6.511E+06
-3.183E-06	-2.038E-06	-1.357E-03	-9.789E-08	-4.378E-08	-1.628E+06
-8.087E-07	2.528E-06	1.138E-06	5.246E-08	1.112E-08	-4.156E+00
-1.026E-07	1.593E-06	2.711E-03	-1.405E-08	-1.411E-09	2.956E-02
-1.302E-08	1.260E-06	1.444E-06	3.746E-09	1.791E-10	7.254E-03
-1.584E-09	6.764E-07	-2.714E-03	-9.375E-10	-2.178E-11	-9.665E-05
					-3.259E+06
-1.613E-06	-3.608E-06	1.353E-03	-3.608E-06	-1.613E-06	5.584E-01
-4.099E-07	-1.933E-06	8.362E-19	1.933E-06	4.099E-07	-1.906E-09
-5.202E-08	-5.179E-07	-2.717E-03	-5.179E-07	-5.202E-08	1.857E-01
-6.602E-09	-1.381E-07	-6.837E-19	1.381E-07	6.602E-09	-2.764E-10
-8.030E-10	-3.455E-08	2.717E-03	-3.455E-08	-8.030E-10	-5.106E-04
					6.511E+06
-4.378E-08	-9.789E-08	-1.357E-03	-2.038E-06	-3.183E-06	1.628E+06
-1.112E-08	-5.246E-08	-1.138E-06	-2.528E-06	8.087E-07	4.156E+00
-1.411E-09	-1.405E-08	2.711E-03	1.593E-06	-1.026E-07	2.956E-02
-1.791E-10	-3.746E-09	-1.444E-06	-1.260E-06	1.302E-08	-7.254E-03
-2.178E-11	-9.395E-10	-2.714E-03	6.764E-07	-1.584E-09	-9.665E-05
					-1.301E+07
4.377E-08	9.787E-08	1.357E-03	-1.062E-06	-1.253E-06	-1.627E+06
1.111E-08	5.245E-08	1.138E-06	9.536E-07	-1.329E-06	1.223E+00
1.411E-09	1.450E-08	-2.711E-03	-1.691E-06	5.449E-07	-1.846E-01
1.791E-10	3.746E-09	1.444E-06	6.435E-07	-4.441E-07	-2.636E-03
2.178E-11	9.374E-10	2.713E-03	-8.837E-07	2.363E-07	4.886E-04

Chebyshev Coefficients of q

n = 5, P = 5

Table 5.3

					-2.793E+07
-1.542E-06	6.347E-07	8.643E-07	-1.092E-06	-4.327E-07	6.983E+06
7.858E-07	-7.341E-07	-3.274E-06	1.099E-06	1.900E-07	-7.384E-02
-1.775E-07	-2.933E-07	1.061E-06	-5.400E-07	-9.697E-08	1.055E-03
-1.654E-07	-9.495E-07	2.761E-06	5.528E-07	7.698E-08	9.770E-05
-2.198E-07	-2.746E-07	1.241E-06	-2.913E-07	-5.869E-08	-1.289E-06
-1.724E-07	-5.041E-07	-3.119E-06	2.589E-07	3.836E-08	-1.837E-07
					-1.397E+07
-3.038E-06	-1.738E-06	2.593E-06	-3.278E-06	-1.298E-06	6.987E+06
-1.334E-06	-3.769E-07	-9.825E-06	3.297E-06	5.700E-07	1.182E-02
-6.809E-07	-8.715E-07	3.184E-06	-1.620E-06	-2.909E-07	2.905E-03
-5.405E-07	-2.089E-06	8.285E-06	1.658E-06	2.309E-07	-1.934E-05
-4.121E-07	-1.023E-06	3.725E-06	-8.741E-07	-1.760E-07	-3.162E-06
-2.693E-07	-1.292E-06	-9.359E-06	7.767E-07	1.151E-07	2.638E-08
					7.438E-02
-2.168E-06	-5.475E-06	-1.893E-06	-5.475E-06	-2.168E-06	6.999E+06
-9.521E-07	-5.508E-06	-1.084E-19	5.508E-06	9.521E-07	7.428E-02
-4.859E-07	-2.706E-06	3.161E-06	-2.706E-06	-4.859E-07	8.304E-09
-3.857E-07	-2.770E-06	1.360E-19	2.770E-06	3.857E-07	-1.021E-04
-2.941E-07	-1.460E-06	5.131E-06	-1.460E-06	-2.941E-07	1.397E-09
-1.922E-07	-1.297E-06	-1.413E-19	1.297E-06	1.922E-07	1.806E-07
					1.397E+07
-1.298E-06	-3.278E-06	2.593E-06	-1.738E-06	-3.038E-06	6.987E+06
-5.700E-07	-3.297E-06	9.825E-06	3.769E-07	1.334E-06	1.182E-02
-2.909E-07	-1.620E-06	3.184E-06	-8.715E-07	-6.809E-07	-2.905E-03
-2.309E-07	-1.658E-06	-8.285E-06	2.089E-06	5.405E-07	-1.931E-05
-1.760E-07	-8.741E-07	3.735E-06	-1.023E-06	-4.121E-07	3.170E-06
-1.151E-07	-7.765E-07	9.359E-06	1.292E-06	2.693E-07	4.074E-08
					2.793E+07
-4.327E-07	-1.092E-06	8.643E-07	6.347E-07	-1.542E-06	6.983E+06
-1.900E-07	-1.099E-06	3.274E-06	7.341E-07	-7.858E-07	-7.384E-02
-9.697E-08	-5.400E-07	1.061E-06	-2.933E-07	-1.775E-07	-1.055E-03
-7.698E-08	-5.528E-07	2.761E-06	9.495E-07	1.654E-07	9.776E-05
-5.869E-08	-2.913E-07	1.241E-06	-2.746E-07	-2.198E-07	1.297E-06
-3.836E-08	-2.589E-07	3.119E-06	5.041E-07	1.724E-07	1.626E-07

Chebyshev Coefficients of q

n = 6 , P = 5

Table 5.4

					1.753E+07
-2.470E-06	-1.223E-06	-9.126E-04	-9.231E-08	-4.128E-08	-2.192E+06
2.416E-06	-1.221E-06	-2.313E-06	4.947E-08	1.048E-08	3.695E-03
8.720E-07	-2.444E-06	1.827E-03	1.325E-08	-1.330E-09	3.420E-04
8.252E-07	-1.094E-06	-1.413E-06	3.552E-09	1.688E-10	-4.518E-06
5.271E-07	-1.936E-06	-1.826E-03	-9.534E-10	-2.022E-11	-2.977E-07
4.241E-07	-7.225E-07	-1.547E-06	2.541E-10	2.474E-12	1.885E-08
2.469E-07	-9.557E-07	1.829E-03	-6.455E-11	3.079E-13	-1.350E-08
					-8.776E+06
-6.367E-06	-4.667E-06	9.129E-04	9.234E-08	-4.129E-08	2.194E+06
-1.617E-06	4.155E-06	2.314E-06	-4.948E-08	-1.049E-08	1.016E-02
-2.052E-07	2.558E-06	-1.829E-03	1.326E-08	1.330E-09	-6.765E-05
-2.603E-08	2.349E-06	1.414E-06	-3.553E-09	-1.688E-10	-1.109E-05
-3.119E-09	1.637E-06	1.826E-03	9.537E-10	2.023E-10	6.645E-08
-3.817E-10	1.238E-06	1.548E-06	-2.542E-10	-2.475E-12	1.789E-09
4.749E-11	6.969E-07	-1.829E-03	6.458E-11	-3.080E-13	-1.615E-09
					4.398E+06
-3.162E-06	-7.072E-06	-9.254E-04	-7.072E-06	-3.162E-06	-1.077E-03
-8.034E-07	-3.790E-06	-6.270E-19	3.790E-06	8.034E-07	3.594E-09
-1.019E-07	-1.015E-06	1.831E-03	-1.015E-06	-1.019E-07	-3.574E-04
-1.293E-08	-2.721E-07	5.899E-19	2.721E-07	1.293E-08	4.947E-10
-1.549E-09	-7.304E-08	-1.833E-03	-7.304E-08	-1.549E-09	3.219E-07
-1.849E-10	-1.946E-08	-7.642E-19	1.946E-08	1.896E-10	4.161E-09
2.359E-11	-4.946E-09	1.833E-03	-4.946E-09	2.359E-11	-1.164E-09
					-8.776E+06
4.129E-08	9.234E-08	9.129E-04	-4.667E-06	-6.367E-06	-2.194E+06
1.049E-08	4.948E-08	-2.314E-06	-4.155E-06	1.617E-06	-1.016E-02
1.330E-09	1.326E-08	-1.829E-03	2.558E-06	-2.052E-06	-6.765E-05
1.688E-10	3.553E-09	-1.414E-06	-2.349E-06	2.603E-07	1.184E-05
2.023E-11	9.537E-10	1.826E-03	1.637E-06	-3.119E-09	6.413E-08
2.475E-12	2.542E-10	-1.548E-06	-1.238E-06	3.817E-10	-1.357E-08
-3.080E-13	6.458E-11	-1.829E-03	6.969E-07	4.749E-11	-5.296E-09
					1.753E+07
-4.128E-08	-9.231E-08	-9.126E-04	-1.223E-06	-2.470E-06	2.192E+06
-1.048E-08	-4.947E-08	2.313E-06	1.221E-06	-2.416E-06	-3.695E-03
-1.330E-09	-1.325E-08	1.829E-03	-2.444E-06	8.720E-07	3.420E-04
-1.688E-10	-3.552E-09	1.413E-06	1.094E-06	-8.252E-07	4.532E-06
-2.022E-11	-9.534E-10	-1.826E-03	-1.936E-06	5.271E-07	-2.954E-07
-2.474E-12	-2.541E-10	1.547E-06	7.225E-07	-4.241E-07	9.720E-09
3.079E-13	-6.455E-11	1.829E-03	-9.557E-07	2.469E-07	-9.080E-09

Chebyshev Coefficients of q

n = 7 , P = 5

Table 5.5

$$
\begin{array}{l}
\cdots\cdots \\
\quad : \\
1/2\ b_0\, a_1 \int_{-1}^{1} T_0\, T_1\, T_3\, dx + 1/2\ b_0\, a_3 \int_{-1}^{1} T_0\, T_3\, T_3\, dx \ : \\
\quad : \\
+\ 1/2\ b_1\, a_0 \int_{-1}^{1} T_0\, T_1\, T_3\, dx + \quad b_1\, a_2 \int_{-1}^{1} T_1\, T_2\, T_3\, dx \ : \\
\quad : \\
+ \quad b_2\, a_1 \int_{-1}^{1} T_1\, T_2\, T_3\, dx + \quad b_2\, a_3 \int_{-1}^{1} T_2\, T_3\, T_3\, dx \ : \\
\cdots\cdots :
\end{array}
\qquad (5.20)
$$

Similarly, for the even case, n = 4, we get:

$$
\begin{array}{l}
\cdots\cdots\cdots \\
\quad : \\
1/4\ a_0\ b_0 \int_{-1}^{1} T_0\ T_0\ T_4 + 1/2\ a_2\ b_0 \int_{-1}^{1} T_0\ T_2\ T_4 \ : \\
+\ 1/2\ a_4\ b_0 \int_{-1}^{1} T_0\ T_4\ T_4 + \quad a_1\ b_1 \int_{-1}^{1} T_1\ T_1\ T_4 \ : \\
+ \quad a_3\ b_1 \int_{-1}^{1} T_1\ T_3\ T_4 + 1/2\ a_0\ b_2 \int_{-1}^{1} T_0\ T_2\ T_4 \ : \\
+ \quad a_2\ b_2 \int_{-1}^{1} T_2\ T_2\ T_4 + \quad a_4\ b_2 \int_{-1}^{1} T_2\ T_4\ T_4 \ : \\
+ \quad a_1\ b_3 \int_{-1}^{1} T_1\ T_3\ T_4 + \quad a_3\ b_3 \int_{-1}^{1} T_3\ T_3\ T_4 \ : \\
\cdots\cdots :
\end{array}
\qquad (5.21)
$$

Where T is actually T(x), $\{ a \}$ are the coefficients of the polynomial that represent the residual and $\{ b \}$ are the coefficients for the kernel polynomial.

To find the contributions of the 3rd interval, we evaluate equation (5.20) for n = 3 and (5.20) for n = 4, using the kernel coefficients in blocks (i,3) and the residual coefficients in the 3rd block. For the n = 3 case, all terms of (5.20) will gives small values. This is because we have a0 >> ai and b0, b2 > b1 (property 1), and terms with the large a0 are multiplied by the small b1 while terms with the large b0, b2 are multiplied by the small a1, a2 and a3. This indicates that a cancellation has occurred in evaluating the terms of (5.20), because we originally used a0 and b1, b2 with large values and ended up with a small result. Similarly, for the n = 4 case, this happens only when we evaluate the contribution of the 3rd interval to itself as block(3,3) is involved which is the only block where its coefficients possess property 1. In this case the residual coefficients have

$$a1 >> ai \qquad i = 0,2,3,.....,n.$$

Finally, to illustrate that the above behaviour of the coeffficents and that the failure would not occur if the singular point is not in the middle of an interval, we carried out this test, in which we used an odd number of collocation points, n = 3, but we start from 4 equal intervals, which means that the singular point lay at an end point and not in a middle of interval. Table 5.6 shows the coefficient values for this test. Studying these values we see that they do not have properties 1 and 2.

-3.621E-04	-3.072E-03	3.068E-03	3.617E-04	-3.274E+10
-1.232E-04	-4.097E-03	-4.090E-03	-1.240E-04	5.456E+09
-2.031E-05	-2.049E-03	2.045E-03	2.066E-05	8.286E+02
				3.514E+02
				1.091E+10
3.603E-04	3.070E-03	-3.069E-03	-3.618E-04	-5.458E+09
1.235E-04	4.096E-03	4.092E-03	1.240E-04	1.835E+03
2.058E-05	2.048E-03	-2.045E-03	-2.067E-05	-2.185E+02
				1.091E+10
-3.618E-04	-3.069E-03	3.070E-03	3.603E-04	5.458E+09
-1.240E-04	-4.092E-03	-4.096E-03	-1.235E-04	-1.835E+03
-2.067E-05	-2.045E-03	2.048E-03	2.058E-05	-2.185E+02
				-3.274E+10
3.617E-04	3.068E-03	-3.072E-03	-3.621E-04	-5.456E+09
1.240E-04	4.090E-03	4.091E-03	1.232E-04	-8.286E+02
2.066E-05	2.045E-03	-2.049E-03	-2.031E-05	3.514E+02

Chebyshev Coefficients of q

n = 3 , P = 4

Table 5.6

All the previous discussion was about the cancellation that occurs in evaluating the error contributions of an interval that a singular point is in its middle to the error of the others. Now we turn to the error estimation in that interval. Tables 5.7, 5.8, 5.9, 5.10 and 5.11 shows the error estimation of each of the 5 equal size intervals and the error contributions from the others. For n = 3, 4, 5, 6 and 7. Looking at the error estimate of the 3rd interval, we see that it has suffered a big cancellation and the amount of cancellation increases as n increases. Furthermore, the contributions of the 3rd interval, in the odd cases, is always small, while in the even cases, the contribution from the 3rd interval to itself is the only small one. This confirms what we have said above.

Contributions: to 1st intv.:	Contributions: to 2nd intv.:	Contributions: to 3rd intv.:	Contributions: to 4th intv.:	Contributions to 5th intv.
-1.12786E+00:	1.66183E-02:	8.00898E-03:	-6.00356E-04:	6.00309E-04
-3.83674E-01:	1.30272E+00:	-8.53145E-02:	6.39520E-03:	-6.39470E-03
6.00684E-01:	-6.00730E-01:	6.10772E-02:	4.78801E-01:	-4.78764E-01
6.19682E-03:	-6.19730E-03:	8.26745E-02:	-1.30921E+00:	3.84845E-01
-7.40672E-04:	7.40729E-00:	-9.88162E-03:	-2.05040E-02:	1.13033E+00
-9.05392E-01:	7.13148E-01:	5.65646E-02:	-8.45114E-01:	1.03061E+00

Error Contributions and Error Estimates

For problem 3, with n = 3 , P = 5

Table 5.7

Contributions: to 1st intv.:	Contributions: to 2nd intv.:	Contributions: to 3rd intv.:	Contributions: to 4th intv.:	Contributions to 5th intv.
-3.11321E+00:	-5.48808E+00:	-3.92052E+00:	-2.35296E+00:	-7.84293E-01
-2.81699E-01:	-3.79716E+00:	-6.24050E+00:	-3.74532E+00:	-1.24840E+00
3.40964E+00:	1.02293E+01:	-3.69006E-05:	-1.02292E+01:	-3.40963E+00
1.24833E+00:	3.74512E+00:	6.24016E+00:	3.79692E+00:	2.81685E+00
7.84265E-01:	2.35287E+00:	3.92038E+00:	5.48788E+00:	3.11319E+00
2.04734E+00:	7.04201E+00:	-5.14579E-04:	-7.04270E+00:	-2.04745E+00

Error Contributions and Error Estimates

For problem 3, with n =3 , P = 5

Table 5.8

Contributions: to 1st intv.:	Contributions: to 2nd intv.:	Contributions: to 3rd intv.:	Contributions: to 4th intv.:	Contributions to 5th intv.
7.81735E-01:	-5.03465E-04:	-2.55194E-04:	-6.92341E-06:	6.92223E-06
2.84653E-01:	-9.23791E-01:	1.11101E-02:	3.01416E-04:	-3.01365E-04
-3.99477E-01:	3.99545E-01:	1.30737E-04:	-3.99806E-01:	3.99737E-01
3.01266E-04:	-3.01318E-04:	-1.11065E-02:	9.23802E-01:	-2.84654E-01
-6.98469E-06:	6.98589E-06:	2.57497E-04:	5.08008E-04:	-7.81743E-01
6.67206E-01:	-5.25044E-01:	1.36667E-04:	5.24799E-01:	-6.66954E-01

Error Contributions and Error Estimates

For Problem 3, with n = 5, P = 5

Table 5.9

Contributions: to 1st intv.:	Contributions: to 2nd intv.:	Contributions: to 3rd intv.:	Contributions: to 4th intv.:	Contributions to 5th intv.
-5.50850E+00:	-9.53022E+00:	-6.80115E+00:	-4.07207E+00:	-1.35725E+00
-5.22305E-01:	-6.94301E+00:	-1.06347E+01:	-6.36734E+00:	-2.12228E+00
-9.60840E+00:	-2.88274E+01:	-4.47321E-09:	2.88274E+01:	9.60840E+00
2.12228E+00:	6.36734E+00:	1.06347E+01:	6.94301E+00:	5.22305E-01
1.35725E+00:	4.07207E+00:	6.80115E+00:	9.53022E+00:	5.50850E+00
-1.21597E+01:	-3.48613E+01:	8.31876E-07:	3.48613E+01:	1.21597E+01

Error Contributions and Error Estimates

For Problem 3, with n = 6, P = 5

Table 5.10

Contributions to 1st intv.	Contributions to 2nd intv.	Contributions to 3rd intv.	Contributions to 4th intv.	Contributions to 5th intv.
-5.76351E-01:	-1.47526E-05:	-7.32844E-06:	9.56858E-08:	-9.56550E-08
-2.12573E-01:	6.76865E-01:	-1.07610E-03:	1.40504E-05:	-1.40459E-05
3.31281E-01:	-3.31388E-01:	-8.87501E-08:	3.31388E-01:	-3.31281E-01
1.40459E-05:	-1.40504E-05:	1.07610E-03:	-6.76865E-01:	2.12573E-01
9.56201E-08:	-9.56510E-08:	7.32577E-06:	1.47472E-05:	5.76351E-01
-4.57629E-01:	3.45448E-01:	-9.57206E-08:	-3.45448E-01:	4.47629E-01

Error Contributions and Error Estimates

For Problem 3, with n = 7, P = 5

Table 5.11

5.5 THE MODIFIED ALGORITHM I (ALGORITHM MI)

In this section, we introduce some modifications to Algorithm I to try to make it work for the case when a singular point is in middle of an interval using an odd number of collocation points. The modification should be done in a way that would not adversely effect the case where Algorithm I behaves well. Obviously, we want Algorithm MI to detect failure cases of Algorithm I and then deal with them. In the next sub-section, we will suggest some detecting algorithms, then after some tests, we select the best one and use it in Algorithm MI.

Generally, Algorithm MI should have the following structure:

1. Use one of the detecting algorithms (given in the next sub-section) to detect the failure case.

2. If there is a failure case, find the interval in which it occurs, select that interval as the interval to be subdivided then go to 4.

3. If there is no failure case, use Algorithm I to find the interval that requires subdivision.

4. Stop.

5.5.1 Possible Algorithms For Detecting Failure Cases

All algorithms introduced in this sub-section are based on either of the two properties of the kernel coefficients given in section 5.3.

1- Algorithm A:-

This algorithm is based on property 1 that the last kernel coefficient (bn) is slightly > than the lst one (b0), in an interval that a singular point is in its middle. This could indicate that approximation of the kernel in that interval is not smooth, and may give an inaccurate error estimation, actually, this is the main motivation for Algorithm A. Algorithm A has following structure.

1. For each column of coefficients, look through the blocks of that column, if a block is found with its last coefficient greater than the lst, add the column number to a column list. Store the largest last coefficient of that column in a coefficient list.

2. If there is more than one column in the column list then, look through the coefficient list to find which column has the largest 'largest last coefficient', the corresponding interval is then selected for subdivision, go to 4.

3. If the column list contains only one column number, the corresponding interval is then selected for subdivision.

4. Stop

Here the column list is a list that contains column numbers and the coefficients list is a list that contains column numbers and their largest last coefficient.

A version of Algorithm MI is obtained using Algorithm A as detecting algorithm, call it Algorithm MI1. Algorithm MI1 is used in testing Algorithm A. The following are the tests:

1. The first test is done on problem 3, using n = 3, starting from 5 equal intervals. The purpose of this test is to see if Algorithm MI1 produces a suitable mesh for the failure case. Figure 5.4

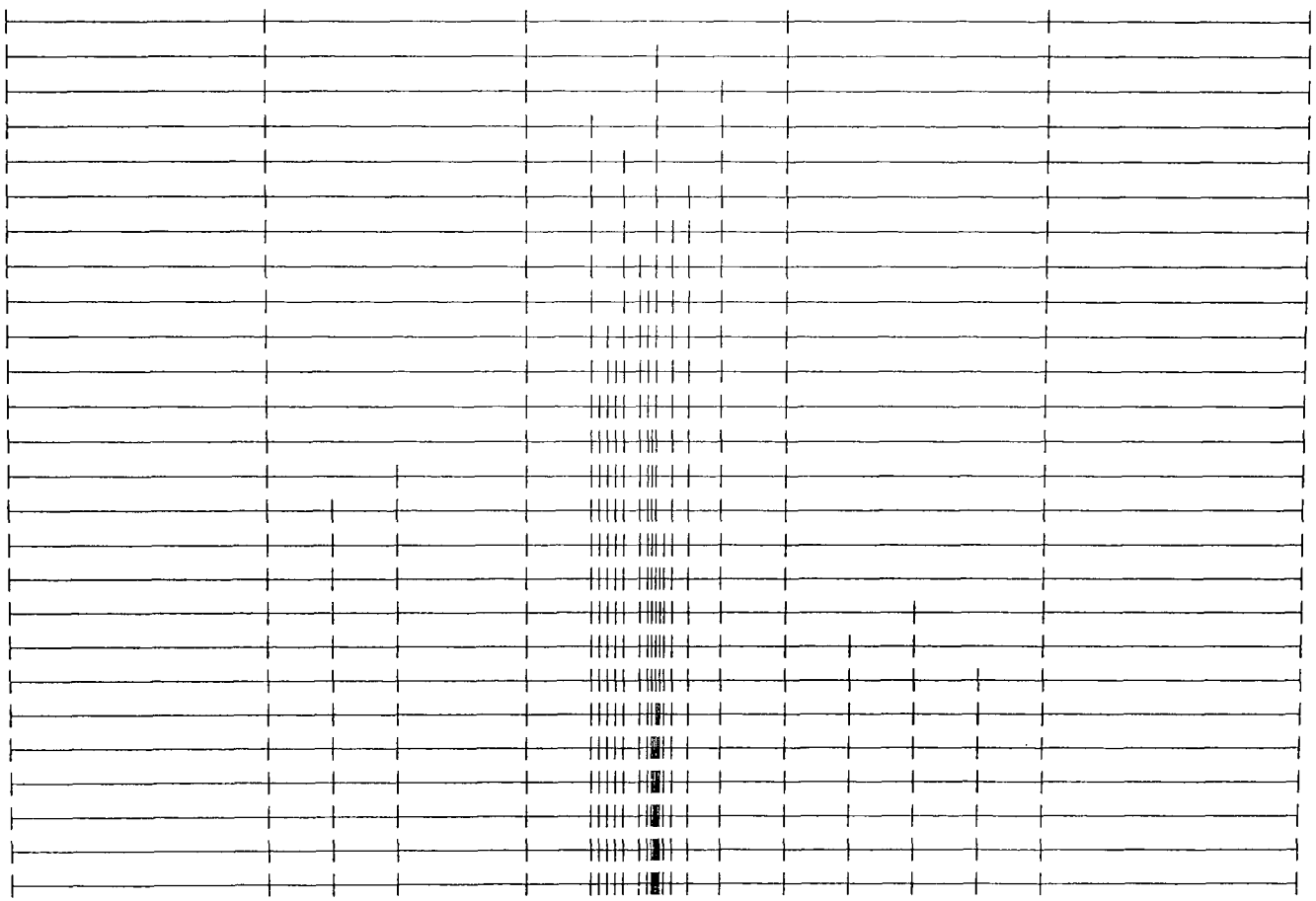


Figure 5.4- Mesh distribution for problem 3, Using Algorithm MI1, n=3, P=5.

shows that the mesh obtained is a sensible one.

2. The second test is also done on problem 3, but, using n = 4 starting from 5 equal intervals, which is not a failure case. The purpose of this test is to see whether the use of Algorithm A may affect the cases where, originally, Algorithm I has performed well. Fig 5.5 shows the choice is not a sensible one. This indicates that Algorithm MI1 has failed, where, originally, Algorithm I has performed well as it is seen in figure 5.2.

2- Algorithm B:-

This algorithm is also based on property 1 as Algorithm A, but instead of comparing the lst and last coefficients, we add , for each column, the lst coefficients together and the last ones. Then subdivide the interval corresponds to the column if the ratio between the two summations is >= 2.

Algorithm B is regarded as a modification to Algorithm A. Here,instead of using coefficients in a block of a column to indicate that the kernel approximation in an interval that corresponds to a column is not good, we use a combination of coefficients in all blocks of a column. This gives a better indication of how well the kernel is represented. The motivation for this algorithm is the same as that for Algorithm A.

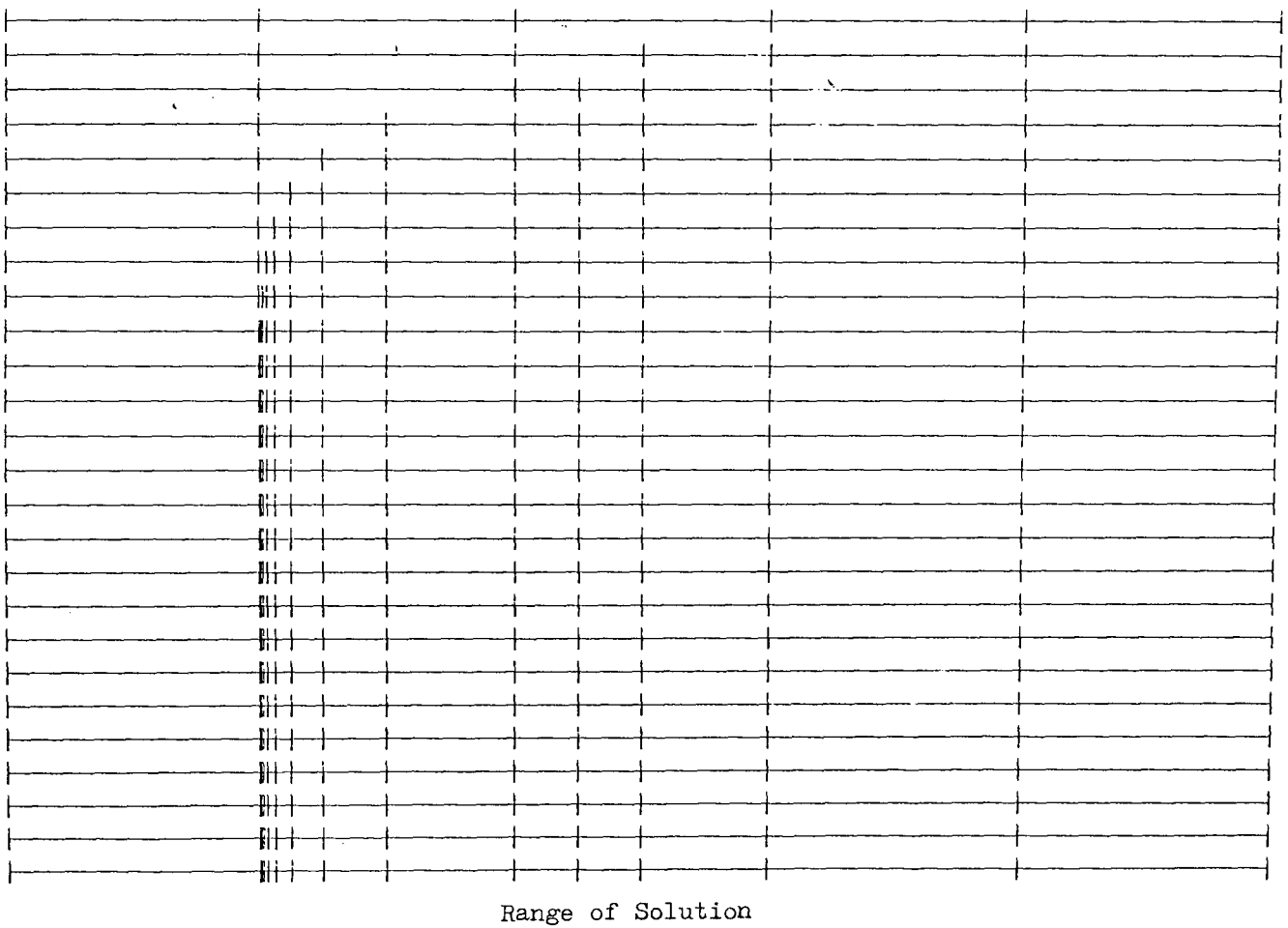


Figure 5.5- Mesh distribution for problem 3, Using Algorithm MI1, n=4, P=5.

Another version of Algorithm MI is obtained by using Algorithm B as the detecting algorithm, call this Algorithm MI2. The previous two tests are also used to test this version. These tests showed that this algorithm behaves in a similar manner to the previous one. Figures 5.6 and 5.7 show the meshes that corresponds to the two tests.

3- Algorithm C:-

This algorithm is based on property 2 of the kernel coefficients and has the following structure:

1. For each column, look through its blocks comparing the 1st and 3rd coefficient with the 2nd one. If the 2nd coefficient is smaller than C times the first and C times the 3rd, where C is a constant to be specified, then, add the column number to a column list. Find the largest last coefficient of that column and add it to a coefficient list.

2. Look through the column list, if there is one column number in it then, select the corresponding interval for subdivision. If there is more than one interval then, find the column having the largest last coefficient, using the coefficient list. The interval corresponding to this column is then selected for subdivision.

Here, the column and coefficient lists are as defined for Algorithm A.

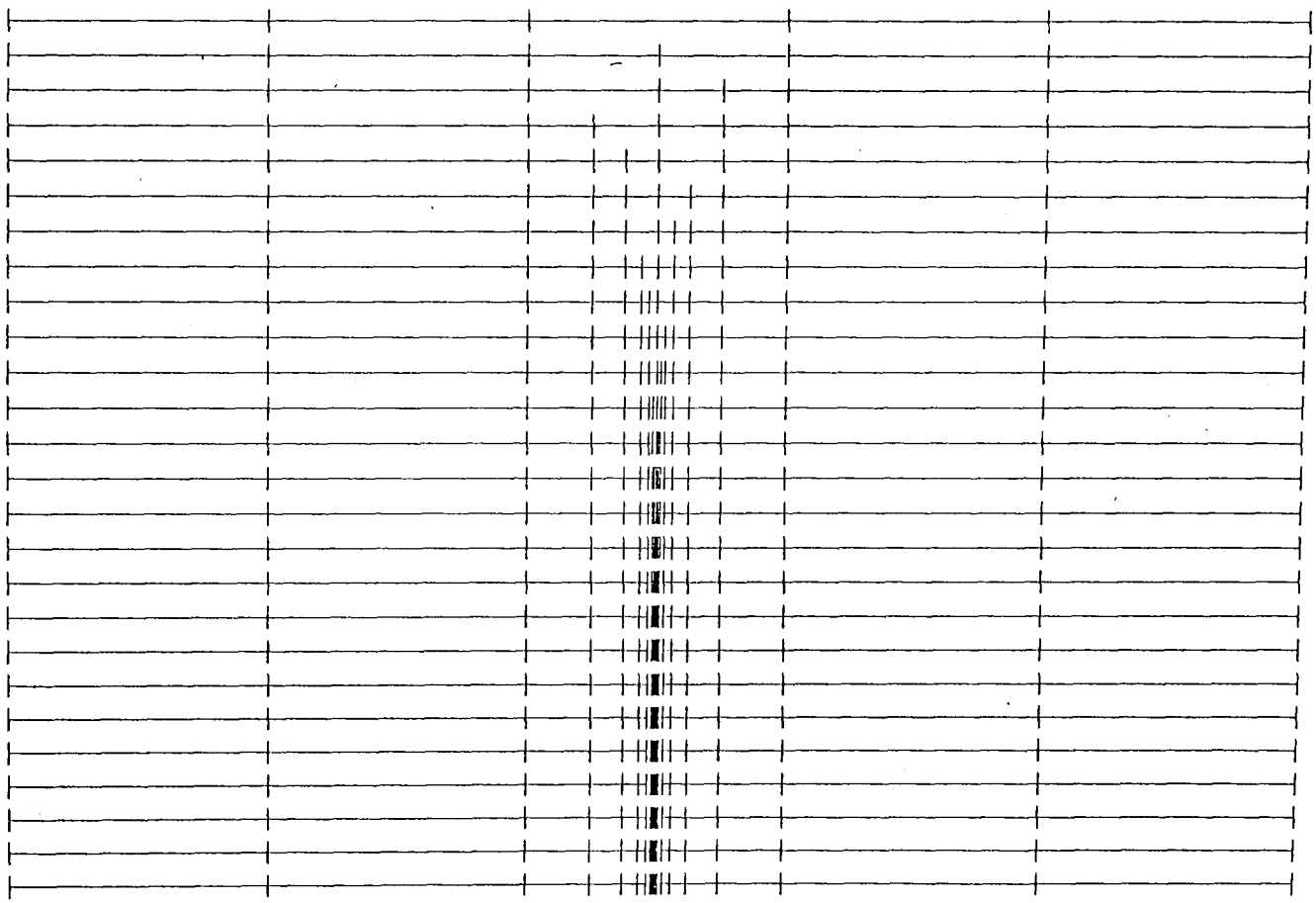


Figure 5.6, Mesh distribution for problem 3, Using Algorithm MI2, n = 3, P = 5.

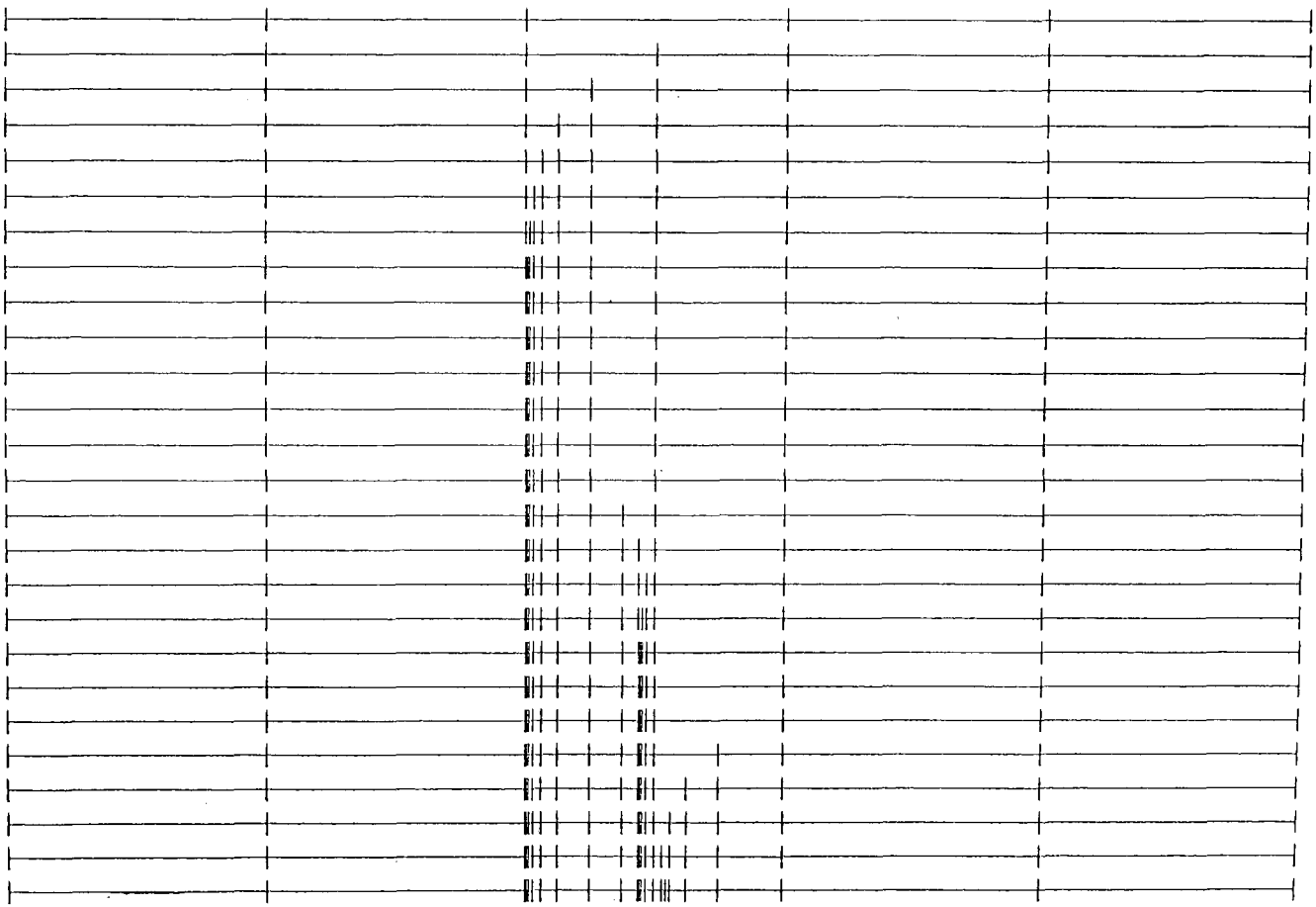


Figure 5.7, Mesh distribution for problem 3, Using Algorithm MI2, n = 4, P = 5.

Using Algorithm C as detecting algorithm, we get another version of Algorithm MI, call it Algorithm MI3. The two tests used in testing Algorithm A and Algorithm B are also used to test Algorithm C, with a value of the constant C of 1E-4.

1. For the first test, we obtained the mesh shown in figure 5.8, this mesh is a sensible one as it is dense around the singular point. It is also the same as the mesh obtained by Algorithm I, figure 5.1.

2. For the second test, the mesh shown in figure 5.9 has been obtained. This mesh is also a sensible one. It is also the same as the mesh obtained by Algorithm I, figure 5.2.

By Comparing the three detecting algorithms introduced here, depending on the results of the two tests, we find that only Algorithm C is satisfactory. Thus, the first two versions of Algorithm MI are neglected and we concentrate our attention on Algorithm MI3 only.

Algorithm MI3 is a compound algorithm, it uses either Algorithm C or Algorithm I to find the interval that requires a subdivision. The form of this algorithm is a special form of Algorithm MI which is as follows:

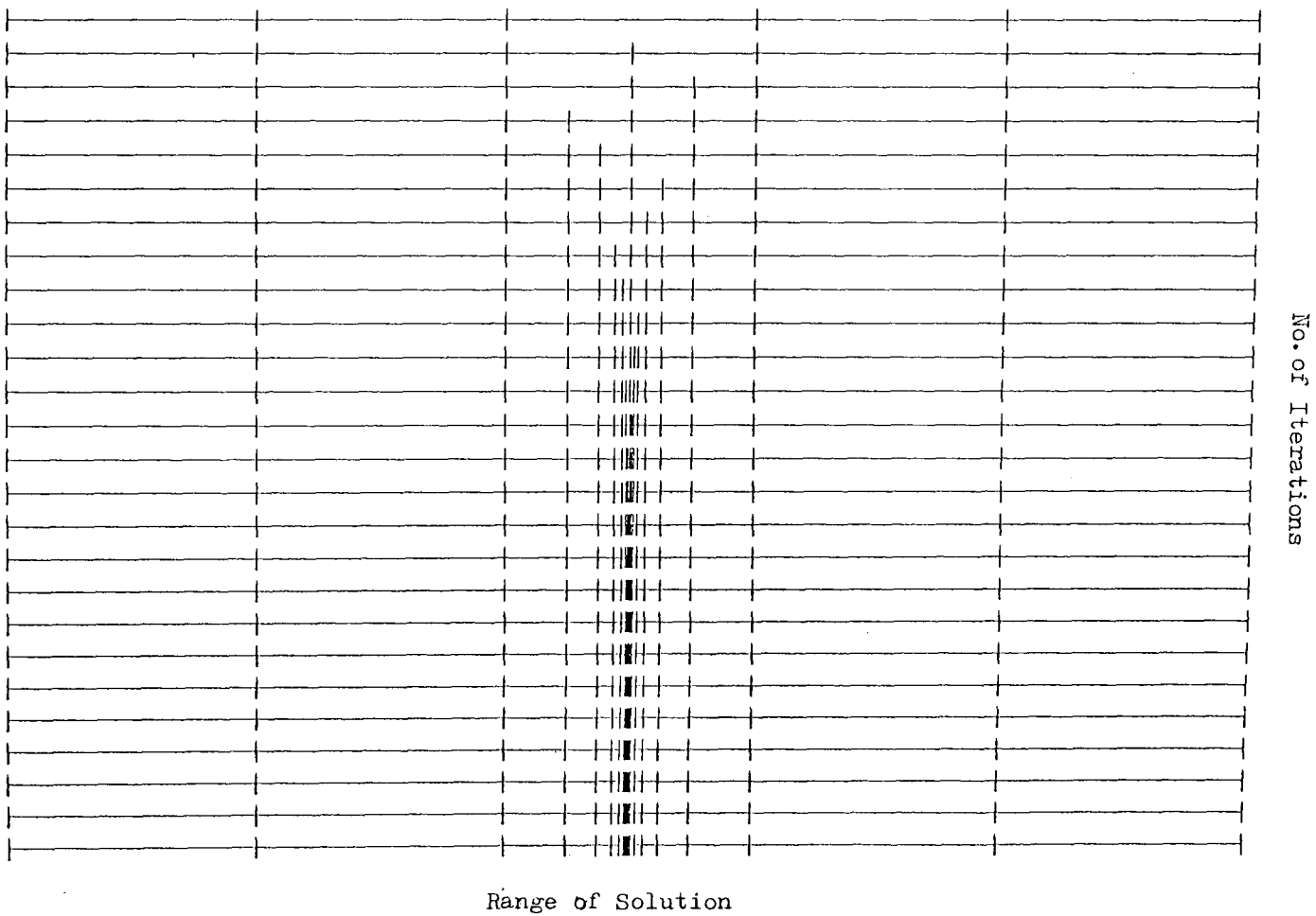


Figure 5.8, Mesh distribution for problem 3, Using Algorithm MI3, $n = 3$, $P = 5$.

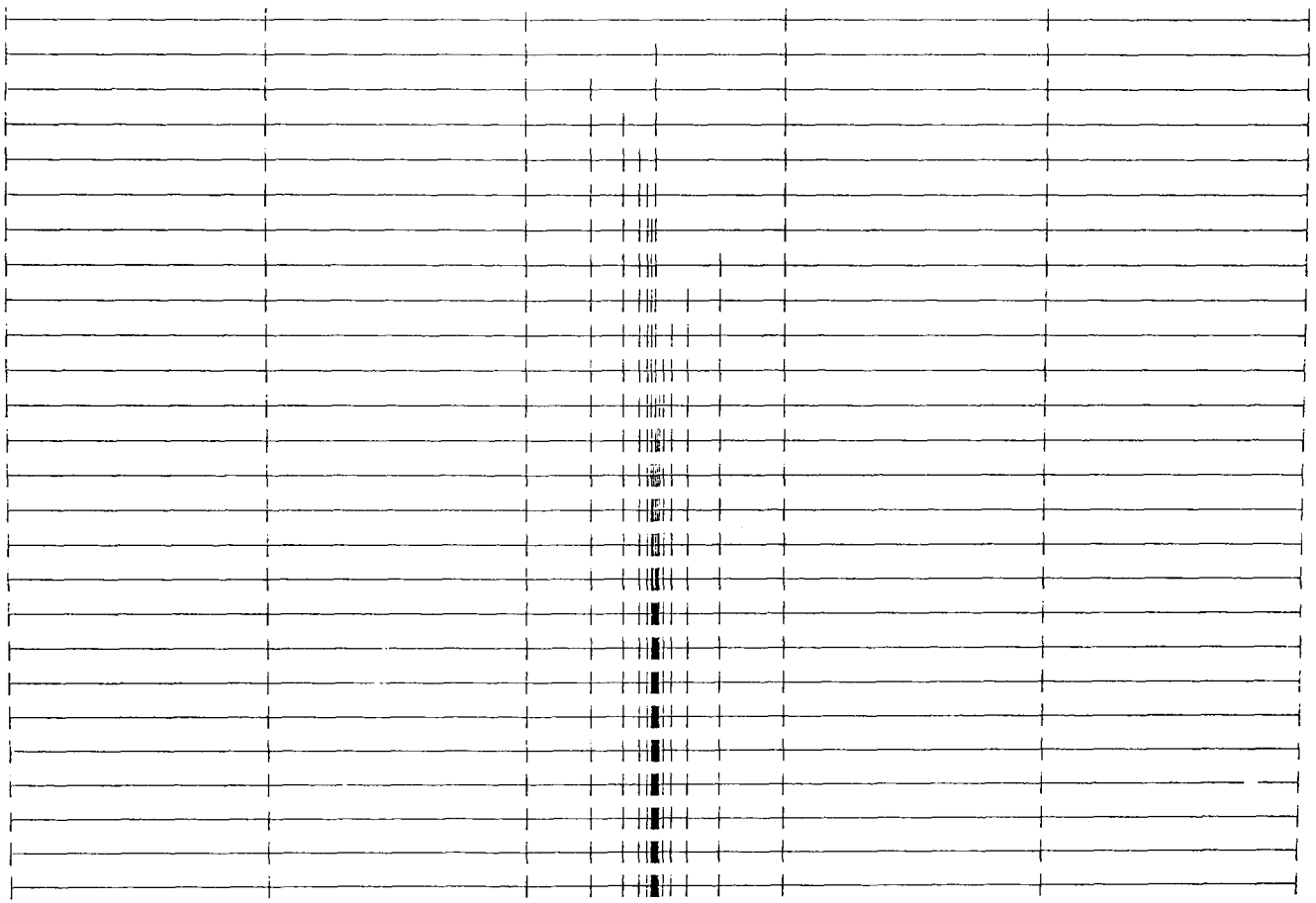


Figure 5.9 - Mesh distribution for problem 3, Using Algorithm MI3, n=4, P =5.

1. Use Algorithm C to detect any failure case, if such case is found then, find the corresponding interval, select it for subdivision and go to 3.

2. If no failure case is detected then, use Algorithm I to find the interval that should be divided.

3. Stop.

To see how Algorithm MI3 would perform if a singular point lies in an interval, but not in the middle, we need to do more tests. These tests are introduced in the next sub-section. In the next chapter, the efficiency of Algorithm MI3 will be tested by comparing it with other algorithms using different test problems.

5.5.2 Testing Algorithm MI3 With Different Locations of The Singular Point

A modified form of problem 3, of chapter 3, is used to perform these tests. This form is as follows:

$$y'' + \lambda(x+A)\, y' = -\pi \cos(\pi(x+A)) - \pi\lambda(x+A) \sin(\pi(x+A))$$

over $(-1,1)$ with

$$y(-1) = B \quad ; \quad y(1) = B1 \; ; \qquad (5.15)$$

where λ is taken to be 1E06 for all the tests, A is a parameter whose value determines the position of the singular point in the interval. B and B1 are the values of the two boundary conditions, they vary with A to agree with the known solution.

In these tests, different values of A are used so that the position of the singular point relative to the interval could be altered. Another purpose of these tests, in addition to the one mentioned earlier, is to see whether Algorithm MI3 detects the case when a singular point may appear in a middle of an interval at later stages of the solution and not in the first stage as in the previous tests. The following are the tests, in all these tests we have started from an initial mesh of 5 equal intervals.

Test 1:-

For this test, we use A = 0.1, that gives

B = -1.95106 and B1 = 4.8943E-2 .

These values are substituted in (5.15) to get the test problem. This problem has a singular point at x = -0.1. As we start from 5 equal intervals, this point lies in the middle of the left half of interval 3. Table 5.12 shows the results obtained by Test 1, the form of this table and the following tables is the same as the tables of section 3.5. For table 5.12 we have n =3, from this table we see that the actual error values and the tested values are decreasing smoothly as the number of intervals increases which indicates that Algorithm MI3 has performed well. When we look at the strategy fields, we see that at the second stage (with 6 intervals),the strategy used was C, while

at the first stage (with 5 intervals), the strategy used was I. This is expected to happen as after the first subdivision, in which interval 3 is subdivided as it is shown in the table, the singular point will lie in the middle of 3rd interval of the new mesh. At this stage, Algorithm I fails and Algorithm C is invoked to deal with the situation. After this stage, Algorithm I is used again as shown in the table.

Table 5.13 shows the results for the same test but with n = 4. It shows that Algorithm MI3 is performing well as the values of the actual error and tested values are decreasing as the number of intervals increased.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	3.936013E+03	3	-2.822754E+07	3	I
6	9.439272E-01	3	3.739722E-03	3	C
7	2.847103E+02	2	3.881024E+04	4	I
8	1.945029E+02	2	-1.860889E+04	3	I
9	1.354676E+02	2	5.155135E+03	5	I
10	1.004014E+02	2	-3.035484E+03	4	I
15	1.678982E+00	2	-1.899186E+01	8	I
20	1.776059E-02	20	4.279484E-01	13	I
25	8.832456E-04	2	3.490587E-01	13	I

n = 3

Table 5.12

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	1.753992E+00	3	-1.039979E+01	3	I
6	1.408150E+00	3	-1.589577E+01	3	I
7	1.356054E+00	6	-1.769894E+01	4	I
8	1.382158E+00	7	2.295931E+02	4	I
9	1.910918E+00	8	-1.938085E+02	5	I
10	2.758900E+00	9	-1.162873E+02	7	I
11	3.125386E+00	10	-2.851075E+02	7	I
12	6.656764E-01	11	3.667983E+02	7	I
13	7.536793E-01	2	-3.749375E+02	6	I
14	4.665479E-01	13	-1.203597E+02	8	I
15	5.481727E-01	2	1.270815E+02	7	I
20	1.259774E-02	11	-6.207576E-01	11	I
25	1.233093E-02	15	5.306743E-02	12	I

n = 4

Table 5.13

It also shows that only algorithm I has been used with no need for Algorithm C. This is what we expected for the even case, where the singular point would not coincide with the middle collocation point as there is no such middle point.

Test 2:-

For this test, we use A = 0.05, that gives

B = -1.98769 B1 = 2.3117E-02 ;

With this problem, we have a singular point at x = -0.05 ; it lies in the middle of the first quarter of 3rd interval. Table 5.14, shows the results for this test, using n = 3. As for the two previous cases, this table shows the Algorithm MI3 is performing well. Looking at the strategy field of this table, we see that there is a C in the 3rd stage, which indicates that Algorithm C has been used st this stage. This is expected because the singular point at this stage is in the middle of the 3rd interval. At the start, the singular point lay in the 3rd interval but not in its middle, after two of subdivisions of the 3rd interval, as it is can be seen from the table, the point becomes the middle of the 3rd interval of the new mesh. After this stage, Algorithm I was used till the end of the process. Table 5.15 shows the results for n = 4, these results show that the algorithm is doing well and only Algorithm I has been used as there is no need for Algorithm C in this case.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.751384E+03	3	-2.007549E+07	3	I
6	1.989527E+03	2	3.614714E+06	3	I
7	1.191198E+00	7	-3.706960E-03	4	ϵ
8	1.377087E+02	7	-5.221549E+03	4	I
9	1.014436E+01	8	3.040041E+03	6	I
10	5.736339E+01	9	-7.697947E+02	5	I
15	1.766912E+00	15	-2.011518E+01	9	I
20	1.663697E-03	11	-3.334979E+00	11	I
25	9.073681E-04	2	1.829242E+00	12	I

n = 3

Table 5.14

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	1.786668E+00	3	1.443055E+01	3	I
6	1.520186E+00	3	9.741043E+00	3	I
7	1.399543E+00	6	-1.394441E+02	4	I
8	1.486245E+00	2	2.020441E+02	4	I
9	2.103390E+00	2	1.987764E+02	4	I
10	3.125253E+00	2	-6.087130E+01	4	I
11	3.073554E+00	2	8.525600E+01	3	I
12	3.144202E+00	2	1.214867E+02	4	I
13	3.568457E+00	2	2.850974E+02	5	I
14	7.181808E-01	2	-3.632785E+02	6	I
15	4.974123E-01	14	-1.238309E+02	8	I
20	5.221672E-02	9	-9.131361E-01	9	I
25	4.891165E-02	25	-5.306719E-02	13	I

n = 4

Table 5.15

Test 3:-

For this test, we used A = 0.66, that gives

B = -1.97858 B1 = 2.14141E-02 .

This problem has a singular point at x = 0.66. This point would never lie in a middle of interval at any stage of the solution. Table 5.16, shows the results for this problem using n = 3. Obviously, Algorithm MI3 is performing well as it can be seen from the table. The table also shows that Algorithm C has never been used. This is expected because as we mentioned earlier, the singular point could never be in a middle of interval.

All the previous tests demonstrate that property 2 of the kernel coefficients occurs whenever an odd number of collocation points is used and a singular point lies in a middle of an interval.

In addition to these tests, we have tested Algorithm MI3 using different values of the parameter λ of problem 3, the values we used are 10, 100, 1000, 1E+04 and 1E+05. The main purpose of these tests is to show that Algorithm MI3 performs well for the smooth problems, when λ is small. A secondary purpose is to see after what value of λ the problems becomes sufficiently stiff to cause property 1 and 2 to appear and Algorithm C to be used. For the first purpose, we found that the algorithm performed well for all different values of λ , for the secondary purpose, we find that the value λ is $\lambda > 1000$.

no. of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	3.384201E+03	3	-2.466264E+07	3	I
6	1.738772E+03	2	3.200944E+06	3	I
7	8.156791E+02	6	-3.703435E+05	4	I
8	4.291033E+02	2	4.937937E+04	4	I
9	1.666222E+02	8	-5.264928E+03	5	I
10	7.973846E+01	2	8.476737E+02	5	I
11	1.159350E+00	10	-4.978663E+01	6	I
12	3.116782E-01	1	-1.068999E+01	7	I
13	3.134306E-01	1	2.678243E+01	6	I
14	5.005233E-01	1	-9.553920E-01	9	I
15	5.005916E-01	1	5.550474E-01	11	I
20	1.701161E-01	2	-2.906241E+00	8	I
25	3.995868E-02	25	1.922318E-01	22	I

n = 3

Table 5.16

CHAPTER SIX

TESTING THE NEW ALGORITHMS AND FINAL COMPARISON

6.1 INTRODUCTION

In this chapter, we use six test problems to test the following algorithms:

1. Q-matrix algorithm (c.f. chapter 3).

2. Algorithm LLWD (c.f. chapter 4).

3. Algorithm MI3 (c.f. chapter 5).

The aims of the tests are:

1. To prove that the two new algorithms perform well.

2. To evaluate the above three algorithms.

Four of the test problems are already given in chapter 3, these are, Problem 1, Problem 2, Problem 3 and Problem 4. The other two problems are:

Problem 5:-

$\epsilon y'' + x y' - y = -(1 + \epsilon\pi^2) \cos(\pi x) - (\pi x) \sin(\pi x)$;

over (-1,1), with the following boundary conditions

$y(-1) = -1$, $y(1) = 1$ and the exact solution

$y(x) = \cos(\pi x) + x + (F/R)$ with

$F = x\ \mathrm{erf}(x/\sqrt{2\epsilon}) + \sqrt{2\epsilon/\pi}\ \exp(-x^2/2\epsilon)$ and $R = \mathrm{erf}(1/\sqrt{2\epsilon}) + \sqrt{2\epsilon/\pi}\ \exp(-1/2\epsilon)$.

This problem has a corner layer at the turning point x = 0;

Problem 6:-

$\epsilon y'' + y' - (1+\epsilon)\, y = 0$;

over (-1,1), with the boundary conditions

$y(-1) = 1 + \exp(-2)$; $y(1) = 1 + \exp(-2(1+\epsilon)/\epsilon)$

and the exact solution

$y(x) = \exp(x-1) + \exp(-(1+\epsilon)\,(1+x)/\epsilon)$.

This problem has a boundary layer near the left end point.

To test algorithm LLWD the first four problems must be divided by λ to get a new problem parameter $\epsilon = 1/\lambda$ which is small.

6.2 EVALUATION OF ALGORITHMS

In this section we evaluate the three algorithms mentioned above using the above four test problems. The form of the result tables used in this section is the same as those described in chapter3. In these tables, we use the following letters to refer to the different algorithms (strategies):

Q - For the Q-matrix Algorithm.

W - For Algorithm LLWD.

C or I - For Algorithm MI3.

To evaluate these algorithms, we study the performance of each algorithm on each test problem individually, then we study the general performance of each one. In both studies, we concentrate on accuracy only, the time required by each strategy is discussed later.

Problem 1 :-

The result tables for this problems are, table 3.4 for the Q-matrix algorithm, table 6.1 for Algorithm LLWD and table 6.2 for Algorithm MI3. Looking at the values of the actual error in these tables, we see that after 30 subdivisions, the smallest of these comes from table 6.1 which contains the results of Algorithm LLWD, while the largest comes from table 6.2. Thus, for this problem, the algorithms could be put in the following order:

1- Algorithm LLWD. 2- Q-matrix algorithm. 3- Algorithm MI3

Problem 2 :-

The corresponding tables are 3.8, 6.3 and 6.4, from these tables we see that the performances of the four algorithms are nearly the same for all cases.

Problem 3 :-

The corresponding tables are 3.12, 6.5 and 6.6, the smallest value of actual error comes from table table 6.6 which corresponds to Algorithm MI3, while the worst comes from table 3.12. From the results of these tables, we can put the algorithms in the following order:

1- Algorithm MI3. 2- Algorithm LLWD. 3- Q-matrix algorithm.

Problem 4 :-

Since this problem has no layer regions, Algorithm LLWD would not construct any initial mesh and it starts with a mesh of 5 equal size intervals. Thus, Algorithm MI3 would behave in the same way as De Boor algorithm and the result table for De Boor algorithm is used in this comparison instead. Hence, the corresponding tables are 3.16, 1.14 and 6.7, studying the values of actual error in these table, we can say that the performances of the three algorithms are the same.

Problem 5 :-

The result tables for this problem are 6.8, 6.9 and 6.10, from these tables we see that the values of the actual error, after 30 subdivisions, in tables 6.8 and 6.10 are nearly equal and smaller than the value in table 6.9. This means that the performances of the Q-matrix algorithm and Algorithm MI3 are the same and both have performed better than Algorithm LLW.

Problem 6 :-

The tables for this problem are 6.11, 6.12 and 6.13, form the values of the actual error in this table, we can say that Algorithm MI3 and Algorithm LLWD have both performed slightly better than the Q-matrix algorithm.

If we take into consideration the performances of each algorithm on all test problems, then, the Q-matrix algorithm comes last while the performance of the other two algorithms is the same. This is because, Algorithm MI3 was the best, among these algorithms, for problem 3 while Algorithm LLWD was the best for problem 1 and the

Q-matrix algorithm has never been the best.

Tables 6.1, 6.3, 3.14, 6.5, 6.9 and 6.12 all shows that Algorithm LLWD has done well for these test problems, while tables 6.2, 6.4, 6.6, 6.7, 6.10 and 6.13 shows the same thing happens for Algorithm MI3.

When comparing the performances of the Q-matrix algorithm with those of Algorithm MI3, we find that, generally, the later has done better. This algorithm has done exceptionally well for problem 3, which is the most difficult one, where it obtained a good accuracy after only 25 subdivision. While, for this problem, the Q-matrix algorithm may require another 10 subdivisions to achieve the same accuracy which, of course, would require a considerable amount of extra execution time. This performance of Algorithm MI3 illustrate the point that using an error estimation as criterion is better than using an error bound, as in the Q-matrix algorithm.

Table 6.14, shows the cpu time required by each algorithm to solve problem 3 using 30 subdivisions. This table shows the that Algorithm LLWD need nearly 1/6 of the time needed by Algorithm MI3 or Q-matrix algorithm. It also shows that the cost of Algorithm MI3 is nearly the same as that of the Q-matrix algorithm. So, if we take both the cost and accuracy into consideration, then, the algorithms can be put in the following order:

1- Algorithm LLWD. 2- Algorithm MI3. 3- Q-matrix algorithm.

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
10	9.130567E-03	5	8.859784E+05	5	W
15	3.333880E-04	15	5.539189E+04	5	W
20	3.333884E-04	1	8.537175E+02	16	W
25	3.333911E-04	25	5.101853E+01	5	W
30	3.333923E-04	30	3.517622E+00	29	W

Table 6.1

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.067432E-01	1	-2.278549E-24	3	C
10	9.067432E-01	10	-9.197026E+00	1	I
15	9.067432E-01	15	-2.738198E+01	1	I
20	9.067432E-01	20	3.595890E+00	1	I
25	7.666072E-01	25	-2.455316E+00	25	I
30	1.457231E-01	30	-5.789063E+01	30	I
35	9.262194E-05	3	2.220788E+00	35	I

Table 6.2

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	4.819249E-02	3	7.615645E+00	5	W
10	8.788784E-04	3	6.999429E-03	2	W
15	5.263121E-05	9	1.227023E-03	6	W
20	2.886393E-05	7	2.935552E-04	2	W
25	4.616496E-06	8	1.273832E-04	10	W
30	3.291270E-06	9	4.323466E-05	5	W

Table 6.3

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.987479E-01	1	4.087821E+00	1	I
10	3.552821E-04	6	-6.088339E-03	6	I
15	3.458062E-05	4	-1.782675E-03	1	I
20	1.036622E-05	5	4.562979E-04	12	I
25	6.106185E-06	14	4.321468E-04	14	I

Table 6.4

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
10	2.626783E-01	1	1.8520528E+05	10	W
15	1.022079E-01	1	5.7579799E+03	3	W
20	8.937548E-02	20	8.5922711E+01	15	W
25	3.175149E-02	1	4.6544711E+00	8	W
30	1.685233E-03	12	1.6260392E-01	13	W

Table 6.5

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	9.645755E-01	3	-3.752025E-03	3	C
10	1.353951E+02	9	5.170936E+03	6	I
15	1.324446E+01	2	-1.009189E+02	7	I
20	1.931260E-02	2	4.276020E-01	13	I
25	9.097463E-04	24	4.733074E-01	16	I

Table 6.6

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	2.133246E+00	2	5.302643E+00	1	I
10	1.789671E-02	7	-3.808973E-01	7	I
15	6.253089E-03	12	-1.763005E-02	8	I
20	5.276099E-04	12	1.272107E-02	18	I
25	5.726620E-04	14	-7.040934E-03	14	I

Table 6.7

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	4.410799E-02	4	2.3340510E-01	2	Q
10	1.566499E-02	5	2.2126766E-01	5	Q
15	5.346351E-03	11	1.0301266E-01	11	Q
20	4.851569E-04	4	2.2113011E-03	4	Q
25	3.006846E-05	23	1.2485983E-04	23	Q
30	1.518748E-05	15	4.3181174E-05	15	Q

Table 6.8

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
10	2.216783E-02	1	1.2144495E-01	1	W
15	1.290430E-02	14	2.8964155E-02	13	W
20	7.765113E-03	14	4.7619362E-02	11	W
25	4.557616E-03	4	7.6237408E-02	12	W
30	2.986518E-03	4	2.5320473E-02	9	W

Table 6.9

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	4.410799E-02	4	-9.734506E-01	3	C
10	1.102389E-02	2	-5.206472E-02	7	I
15	6.597443E-03	14	-4.678072E-02	9	I
20	6.965275E-04	2	-1.534622E-03	9	I
25	3.007616E-05	22	-1.006616E-04	3	I
30	1.719414E-05	25	-1.217503E-04	29	I

Table 6.10

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	8.911372E+00	1	2.9340427E+02	1	Q
10	2.678916E+01	9	5.0328869E+02	1	Q
15	6.945406E+01	14	1.9302748E+03	1	Q
20	5.7723410+01	19	9.9143638E+02	3	Q
25	3.198513E-01	1	8.4206152E-01	1	Q
30	2.274362E-04	1	7.4925299E-03	2	Q

Table 6.11

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	4.388686E-01	5	1.0988915E+07	5	W
10	1.545131E-01	10	4.1967571E+03	5	W
15	8.718280E-04	1	3.5995104E+00	2	W
20	1.008037E-04	20	3.5931059E-01	1	W
25	1.006198E-04	25	7.4345030E-02	11	W
30	1.006076E-04	30	1.8761258E-02	4	W

Table 6.12

no of intervals:	largest error (actual)	interval number	largest tested value	interval number	strategy
5	8.911372E+00	1	-4.1049902E+01	2	I
10	6.508212E+00	1	6.7089756E+00	1	I
15	2.004482E+01	13	2.0347156E+02	1	I
20	4.591416E+00	1	-1.7900454E+03	1	I
25	9.462318E-04	4	-3.8245870E+00	3	I
30	1.313257E-04	10	-4.6523060E-01	5	I

Table 6.13

Algorithm	LLWD	Q-matrix	MI3
Time in seconds	6.937	39.435	39.720

Table 6.14

CHAPTER SEVEN

FURTHER REMARKS AND CONCLUSIONS

At the beginning of this thesis, we stated that our aim was to investigate different AMS algorithms then compare them. To achieve this task, we required a lot of experimental work which needed a lot of programming. In chapter 2, we discussed the structure of the Pascal program and the possible data structure which is suitable for adaptive algorithms. From this discussion we found that a structure using Pointers and Records is suitable and the Array structure is not.

In chapter 3, we discussed four AMS algorithms with their theories and motivations. Some of these algorithms were introduced for the first time and others, the Largest residual and Q-matrix algorithms, have been dealt with before. Then, we compared the four algorithms by using four test problems. The result of the comparison showed the superiority of the Q-matrix algorithm with respect to the others. It also showed that the cost of the Q-matrix algorithm is very high, a large amount of this cost comes from constructing the Q-matrix. Further research to reduce the cost of the Q-matrix algorithm by reducing the construction cost could be useful.

We have tried to reduce this cost by, instead of reconstructing all the blocks of the matrix, reconstructing the new blocks that

correspond to the two sub-intervals which resulted from dividing an interval. The values in the new blocks are obtained from the values of the block that corresponds to the divided interval, from the previous iteration. We have noticed that for smooth problems, the norms of the blocks do not change substantially from one iteration to the next, except for those new blocks. For the difficult problems, with few intervals, this does not happen, and the matrix needs a number of iterations, probably large, to settle down. Thus, the above technique of constructing the matrix does not work for the difficult cases unless we know the stage at which the matrix settles, which need more research.

During the work of chapter 3, we observed that the choice of the initial mesh could change the performance of any AMS algorithm. So, in chapter 4, we introduced an algorithm which first, constructs a good initial mesh that reflects to a certain extent the behaviour of the solution of the B.V.O.D.E. problems, then invoke the De Boor algorithm to continue the mesh selection process. The algorithm is based on some theorems which are introduced in that chapter with their proofs which depend on the asymptotic solution of the problems. A brief introduction to such solutions and methods was also given in that chapter.

From chapter 4, we proved that, for second order singular problems, the coefficient of the first order term plays an important role in determining the behaviour of the solution. This, can be

generalized for higher order problems where the coefficient of the 2nd highest term could play an important role in determining the solution behaviour. This has not been proven practically and needs more investigations both theoretically and practically.

Another point about this chapter is the applicability of the algorithm to non-linear problems. For problems without a turning point, the algorithm is applicable, but, it is difficult to apply it to problems with turning points due to the difficulty in determining and locating the turning point. As when non-linear problems are solved by the iteration of linearized problems, where each problem has a different location for the turning point. This is another point for further research.

In chapter 4, we also found that the accuracy of the approximation could change by changing the number of initial mesh points to be placed within a layer. From the work of this chapter, we found that using too many or too few points give poor accuracy, while using an appropriate number of points gives good accuracy.

The accuracy of the estimated width obtained by the algorithm given in chapter 4 depends on the accuracy of the integration method used in its evaluation. For this estimation we used a simple integration method which gives a reasonable accuracy. Note that, to build the initial mesh, we do not have to know the exact width of the layer so we do not require a very accurate method of estimation.

Though, a better accuracy could improve the performance of Algorithm LLWD but with a bit more cost. From the results of chapter 6, it is obvious that Algorithm LLWD gives a reasonable accuracy with the current integration method. In the evaluation of I and $f'(0)$, we did not pay a lot of attention to the accuracy and we used cheap methods, except when we get an I value which is close to zero, where a more accurate method is used to find if it is actually zero. Another way to deal with this is to take the I value as zero and regard the problem as having two boundary layers. This could result in wasting some mesh points, in case I is not actually zero. At the end of chapter 4, we introduced some numerical results that illustrated all theorems given in that chapter and showed that the algorithms that locate layers and estimate their widths are doing well for the given test problems.

In chapter 5, a modification to the criterion used by the Q-matrix strategy is introduced. This criterion is an error estimate obtained by multiplying two polynomials, one representing the kernel and the other representing the residual. We showed that the evaluation of this is not as complicated as it may look and it involves a multiplication of the coefficients of the two polynomials only. The main motivation for using this new criterion is that an error estimate is more appropriate than an error bound for mesh selection purpose. We also introduced Algorithm I which is based on the new criterion. Before we tested Algorithm I, we were expecting it to perform much better than the Q-matrix algorithm. But when we tested

it on problem 3, we found that it did well when an even number of collocation points are used and failed for the odd cases. We investigated the failure cases by studying the coefficients of the two polynomials involved in evaluating the error estimate and we found that the coefficients of the polynomial that represent the kernel behave strangely when a singular point is exactly in a middle of an interval. This behaviour is described by property 1 and 2. We found that because of this behaviour a cancellation occurs when evaluating the share of error of that interval which makes it small, and causes the algorithm to fail. A number of tests was done in that chapter that illustrate this. It was also found that a cancellation occurs in evaluating the error estimate of an interval with a singular point in its middle, this has been illustrated with a table.

In the same chapter, we introduced three algorithms that could be used to recover the cases where Algorithm I has failed. All three algorithms were based on the special behaviour of the kernel coefficients mentioned above. After testing these algorithm, we found that Algorithm C, which is based on property 2, is an appropriate recovery algorithm to be used before algorithm I is used. When these two algorithms are combined they form Algorithm MI3.

In chapter 6, we introduced a final comparison between the Q-matrix algorithm, Algorithm LLWD and Algorithm MI3. From this comparison, we found that Algorithm LLWD was the cheapest and it has a reasonable accuracy that makes it favourite among the others. This

shows that this algorithm has reasonable accuracy although it uses cheap evaluation methods and proves that the performance of any algorithm could be highly improved if it starts from a good initial mesh.

The comparison also showed that Algorithm MI3 has, generally, better accuracy than the Q-matrix algorithm with both having virtually the same cost. Thus, we could say that Algorithm MI3 could achieve an accuracy which is the same as that of the Q-matrix with fewer number of subdivisions, which consequently means less cost.

In addition to the algorithms described in this thesis, we have found a simple ad-hoc algorithm that performed well for all test problems of chapter3. This was to choose the interval with largest penultimate solution coefficients for subdivision. This algorithm has no theoretical foundation or motivations except that during the work of chapter 3, we noticed that, for problems of this chapter, the penultimate solution coefficient for the interval which is near to a boundary or turning point layer is large. However, when we tested this algorithm on problem 5 of chapter 6, we found that it failed.

It should be noticed that the conclusions we stated here are all based on a limited set of test problems and it is a bit dangerous to generalize them to far. This is because, as we saw above, a strategy may perform well for, possibly, a considerable number of problems then fails for another. However, the tests given here do give some

indications of the relative merits of the algorithms and show the useful ones. We should note also that there are always possibilities for further testing and improving to any of the strategies we introduced.

Finally, it may be true that no AMS algorithm will work for all problems unless a quite good initial mesh is used, as successively more difficult problems are posed.

References

(1) Abrahamsson L.R. (1975), "A priori estimate for solutions of singular perturbations with a turning point". Studies in Applied Math., 56, pp 51-69.

(2) Abrahamsson L. R. (1977), "Difference approximations for singular perturbations with a turning point". Report No. 58, Department of Computer Sciences, Uppsala University.

(3) Ackerberg R. C. and O´Malley R. E. JR. (1970), "Boundary layer exhibiting resonance". Studies in Applied Math. , pp 277-295.

(4) Ahmed A. H. (1981), "Collocation algorithms and error analysis for approximate solutions of ordinary differential equation". Dectoral thesis, University of Newcastle upon Tyne.

(5) Anselone P. M. (1971), "Collectively compact operator approximation theory". Prentice Hall.

(6) Ascher U., Christiansen J. and Russell R. (1978), "COLSYS - A collocation code for boundary problems". Proc. Conf. for B.V.P´S, Housten.

(7) Ascher U., Preuss S. and Russell R. D. (1983), "On spline basis selection for solving differential equations". S.I.A.M. J. Numer. Anal., vol. 20, No. 1, pp 121-141.

(8) Brown R. R. (1962), "Numerical solution of boundary value problems using non-uniform grids". J. S.I.A.M., 10, pp 475-495.

(9) Burchard H. G. (1974), "Splines (with optimal knots) are

better". Applicable Anal., vol. 3, pp 309-319.

(10) Collatz L. (1966), "Functional analysis and numerical mathematics". Academic Press.

(11) Cranley R. and Patterson T. N. L. (1971), "On the automatic numerical evaluation of definite integral". Comp. J., 14 (2), pp 189-198.

(12) Cruickshank D. M. and Wright K. (1978), "Computable error bound for polynomial collocation methods". S.I.A.M. J. Numer. Anal., 15, pp134-151.

(13) Davis P. and Rabinowitz P. (1967), "Numerical integration". Blaisdell.

(14) De Boor C. (1972), "On calculation with B-splines". J. Approx. Th., 6, pp 50-56.

(15) De Boor C. (1973), "Good approximation by splines with variable knots". II conference on the numerical solution of differential equations; lecture notes in math., vol. 363, Springer-Verlag, Berlin and New-York.

(16) De Boor C. and Swartz B. (1973), "Collocation at Gaussian points". S.I.A.M. J. Numer. Anal., No. 10, p 582.

(17) De Boor C. and Swartz B. (1977), "Comment on the comparison of global methods for linear two-point boundary value problems". Math. of Computation, vol. 31, No. 140, pp 916-921.

(18) Derivas E. K. (1971), "On the use of non-uniform grids in finite difference equations". J. of Computational Phys., 10, pp 202-210.

(19) Dodson D. J. (1972), "Optimal order approximation by spline

functions". Dectoral thesis, Purdue University.

(20) Eckhaus W. (1973), "Matched asymptotic expansions and singular perturbations". Mathematic studies, North Holland Publications, Amsterdam.

(21) Finlayson B. A. and Scriven L. E. (1966), "The method of weighted residuals - A review". Applied mechanics reviews, vol. 19, no. 9, pp 735-748.

(22) Fourer R. (1984), "Staircase matrices and system". S.I.A.M. review, vol. 26, No. 1, pp 1-70.

(23) Fox L. and Parker (1966), "Chebyshev polynomials in numerical analysis". Oxford mathematical handbook, London.

(24) Gerrard C. and K. Wright (1984), "Asymptotic properties of collocation matrix norms 2: piecewise polynomial approximation". IMA J. of Numer. Anal., 4, pp 363-373.

(25) Gladwell I. (1972), "A posteriori error bounds for approximate solutions of linear second order differential equations". J. Inst. Maths. Applics., 9, pp 323-349.

(26) Grassman J. (1971), "On the birth of boundary layers". Mathematical Centre Tracts 36, Amsterdam.

(27) Hemker P. W. (1978), "A numerical study of stiff two point boundary problems". Mathematical Centre Tracts 80, Amsterdam.

(28) Humphrey D. and Garey G. F (1978),"Adaptive mesh refinement using element residuals". TICOM report 78-1, Texas Institute for Computational Mechanics, Jan 1978.

(29) Ito K. (1976), "An aposteriori error estimation of approximate solutions of two point boundary value problems for piecewise smooth

systems". Memoirs of the faculty of science, Kyushu University, ser. A, vol. 30, No. 1, pp 75-94.

(30) Kantorovich L. V. and Akilov G. P. (1964), "Functional analysis in normd spaces". Pergamon.

(31) Kedem G. (1981), "A posteriori bounds for two point boundary problems". S.I.A.M. J. Numer. Anal., vol. 18, No. 3, pp 431-448.

(32) Kreiss B. and Kreiss H. O. (1981), "Numerical methods for singular perturbation problems, S.I.A.M. J. Numer. Anal., vol. 18, pp 262-276.

(33) Kreiss H. T. and Parter S. V. (1974), "Remarks on singular perturbation with turning points". S.I.A.M. J. Math. Anal., vol. 5, No. 2, pp 230-251.

(34) Lanczos C. (1938), "Trignometric interpolation of empirical and analytical functions". J. math. phys., pp 123-129.

(35) Lentini M. and Pereyra V. (1974), "A variable order finite difference method for non-linear multipoint boundary value problems". Math. Comp., 28, pp 134-154.

(36) Lentini M. and Pereyra V. (1975), "An adaptive finite difference solver for non-linear two point boundary problems with mild boundary layers". STAN-CS-75-530, Computer science department, Stanford University.

(37) O'Malley R. E. JR. (1974), "Introduction to singular perturbations". Academic Press, New-York.

(38) Pearson C. E. (1968), "On a differential equation of boundary layer type". J. Math. Phys., 47, pp 134-154.

(39) Pereyra V. and Sewell G.E. (1975), "Mesh selection for discrete

solution of boundary problems in ordinary differential equations". Numer. Math., 23, pp 261-268.

(40) Phillips J. L. (1972), "Collocation as a projection method for solving integral and other operator equations". S.I.A.M. 9, No. 1, p 14.

(41) Rheinboldt W. C. (1981), "Adaptive mesh refinement process for finite element solution". International J. for numerical methods in engineering". No. 17, pp 649-662.

(42) Roberts G. O. (1971), "Computational meshes for layer problems". Proceeding of the second international conference on numerical methods in fluied dynamic, Lecture notes in physics, 8, pp 171-177.

(43) Russell R. D. (1979), "Mesh selection methods". Lecture notes on computer science: codes for boundary value problems in ordinary differential equation, Edited by G. Goos and J. Hartmanis, pp 228-243.

(44) Russell R. D. and Christiansen J. (1978), "Adaptive mesh selection strategies for solving boundary value problems". S.I.A.M. J. Numer Anal., vol. 15, No. 1, p 59.

(45) Russell R. D. and Shampine L. F. (1972), "A collocation method for boundary value problems". Numer Math., 9, p 1.

(46) Russell R. D. and Varah J. M. (1975),"A comparison of global methods for linear two-point boundary value problems". Math. of Comp., vol 29, No. 132, pp 1007-1019.

(47) Shampine L. F. (1969), "Boundary value problems for ordinary differential equations II: Patch bases and monotone methods".

S.I.A.M. J. Numer. Anal., Vol. 6, No. 3, pp 414-431.

(48) Sirovich L. (1971), "Techniques of asymptotic analysis". Springer-Verlag; New-York.

(49) Tewarson R. P. and N. S. Hulsak N. S. (1983), "An adaptive implementation of interpolation methods for boundary differential equations". BIT, 23 (3), pp 382-387.

(50) Villadsen J. V. and Stewart W. E. (1967), "Solution of boundary value problems by orthogonal collocation". Chemical engineering science, vol. 22, pp 1483-1501.

(51) White A. B. (1979 a), "Mesh selection for boundary value codes". Same as reference (43), pp 266-274.

(52) White A. B. (1979 b), "On selection of equidistributing meshes for two points boundary value problems. S.I.A.M. J. Numer. Anal., vol. 16, pp 472-502.

(53) Wright K. (1964), "Chebyshev collocation methods for ordinary differential equations". Computer J., 6, p358.

(54) Wright K. (1979), "Asymptotic properties of the norms of certain collocation matrices". Technical report No. 135, Computing Lab., University of Newcastle Upon Tyne.

(55) Wright K. (1984), "Asymptotic properties of collocation matrix norms I : Global polynomial approximation". IMA J. Numer. Anal., 4, pp 185-202.