AN ADAPTIVE MESH-GENERATION TECHNIQUE FOR BOUNDARY-LAYER TYPE DIFFERENTIAL EQUATIONS, USING THE BOX SCHEME

by

Obinna S. Anizor

Research Report CS-75-08

Department of Computer Science

University of Waterloo
Waterloo, Ontario, Canada

March 1975

TITLE

AN ADAPTIVE MESH-GENERATION TECHNIQUE FOR BOUNDARY-LAYER TYPE DIFFERENTIAL EQUATIONS, USING THE BOX SCHEME

by

Obinna S. Anizor

ABSTRACT

The nature of the solutions of differential equations of boundary layer type suggests the desirability of using non-uniform meshes in the numerical schemes used in solving them. There will be greater concentration of mesh points in the boundary layer regions than away from them.

In this study, we describe and implement a scheme that adaptively generates these mesh points. We explore the possibility of completely automating this procedure.

Tests are performed and the results are shown. Comparisons of the results on a non-uniform mesh with results on a comparable uniform mesh are made. Some conclusions are drawn.

Keywords: boundary layer, adaptive scheme, mid-point rule.

CONTENTS

CHAPTER 1

Introduction: The Problem

1.A Sources, Nature and Examples of Boundary Layer Problems

Two point boundary value problems of the boundary layer type arise in many areas of applied science. Examples are the boundary layer equations of fluid dynamics, the edge-effect-on-shells problems of elasticity and the W.K.B. (Wentzel, Kramers and Brillouin) problems of quantum mechanics [3,5,6].

These equations are characterized by very sharp changes in the solutions and/or their derivatives close to the boundaries.

Simple linear examples of such equations and the behaviour of their solutions are shown below:

Example 1

$$\varepsilon y'' - y = 0$$
$$y(-1) = 1,\ y(1) = 2$$

The behaviour of the solution is sketched in Fig.1.1 for $0 < \varepsilon << 1$. $y(x) = 0$ except near the end points, where $y(x)$ rises steeply in "boundary layer" regions - each of width $O(\sqrt{\varepsilon})$ - so as to satisfy the boundary conditions.

Example 2

$$\varepsilon y'' + xy' = 0$$
$$y(1) = 1;\ y(1) = 2$$

The behaviour of the solution is sketched in Fig.1.2 for $0 < \varepsilon << 1$. The solution consists (nearly) of two horizontal straight lines, joined together in a narrow transition region (interior boundary layer) near the origin. The width of the transition region is of $O(\sqrt{\varepsilon})$.

For more examples and discussion of the nature of equations of boundary layer type - linear and nonlinear - see C.E. Pearson [3,4].

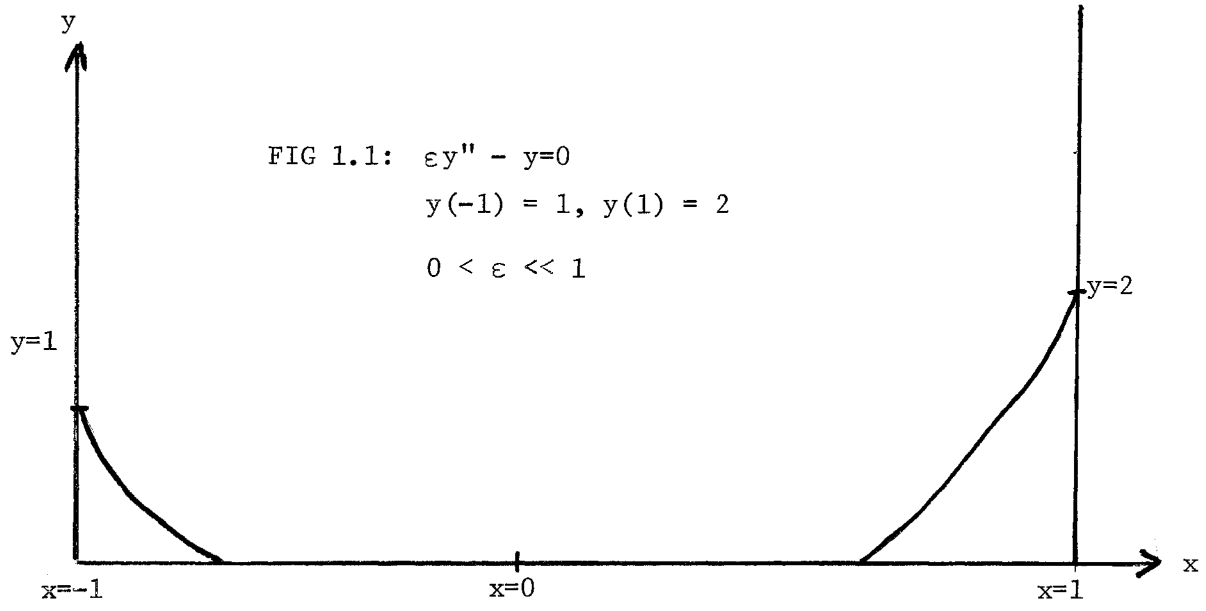


FIG 1.1: $\varepsilon y'' - y = 0$

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

$0 < \varepsilon \ll 1$

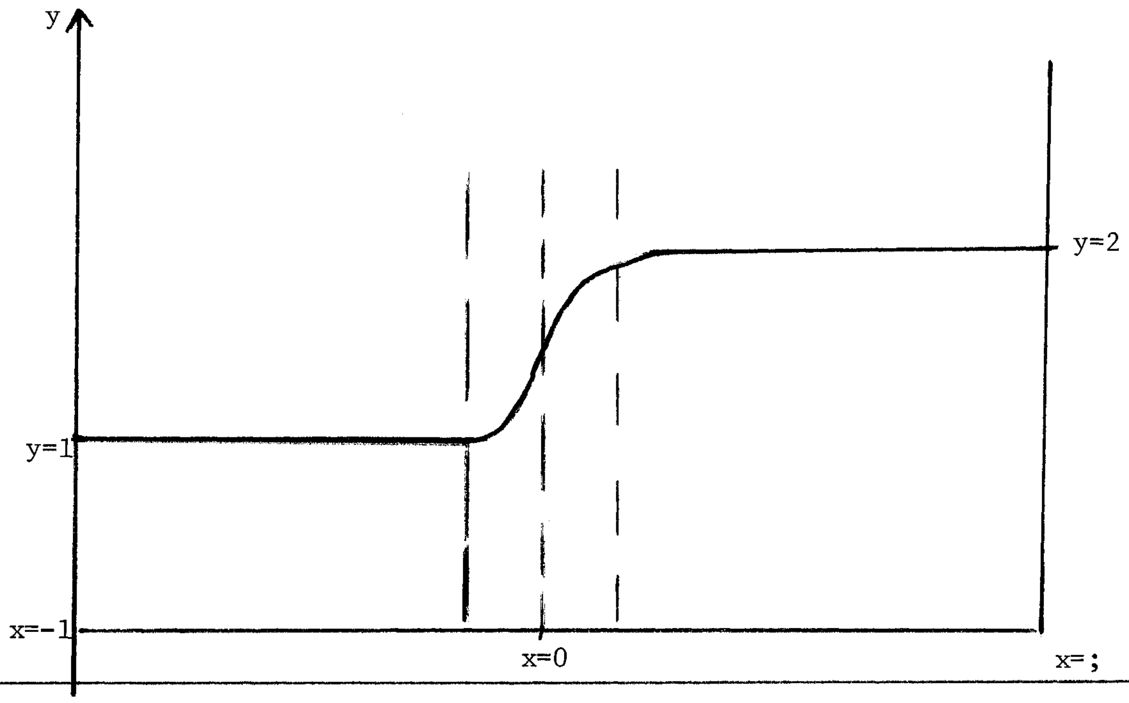


Fig 1.2: $\varepsilon y'' + xy' = 0$

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

$0 < \varepsilon \ll 1$

1.B Why use an adaptive scheme.

The nature of the solutions of boundary layer equations suggests the desirability of using variable mesh spacing for the difference schemes used in approximating their solutions.

H.B. Keller [1,2] remarked that when the 'Box Scheme' is combined with $h \to 0$ extrapolation on a fixed, pre-chosen mesh, uniform or variable, one can obtain sufficient accuracy for most practical purposes, even with crude meshes.

In many subtle cases of boundary layer type equations, one may not know beforehand where to use fewer or more mesh points. It is therefore desirable to have a scheme which picks the mesh points adaptively, starting with crude meshes.

1.C Aim of Report

We attempt in this report to describe a scheme which adaptively picks the mesh points. We also explore the possibility of completely automating the mesh selection process.

In chapter 2, the scheme is described and analysed. Its algorithmic details are available in the Appendix.

In Chapter 3, experiments with three examples are discussed with particular emphasis on comparing results of the non-uniform mesh adaptively generated with a comparable uniform mesh.

Chapter 4 contains our conclusions and a list of possibilities which exist in applying our scheme.

CHAPTER 2

The Adaptive Scheme

2.A Basis

The scheme outlined below is based on the decomposition of the given differential equation(s) into a system of first order equations, as done for the Box Scheme. (See [1],[5] and Appendix C for details.)

The system is now of the form

$$Dw'(x) = c \qquad (2.A.1)$$

$$a < x < b$$

with boundary conditions $P_a w(a) = g_a$

$$P_b w(b) = g_b$$

where x is the independent variable

w is the solution vector with components $w_i = w_i(x), i = 1,2,\dots,m$, where m is the number of first order equations in 2.A.1

D is an $m \times m$ matrix

c is an m-vector

D, c can be (nonlinear) functions of w and x

P_a is a $p_1 \times m$ matrix with constant entries

P_b is a $p_2 \times m$ matrix $(p_1+p_2 = m)$ with constant entries

g_a is a p_1-vector of boundary values at $x = a$

g_b is a p_2-vector of boundary values at $x = b$.

Let us denote the solution of the system 2.A.1 using a uniform mesh of size h by $w(x,h)$. We denote the ith-component of $w(x,h)$ by $w_i(x,h)$.

Consider the Box Scheme which is a second order accurate scheme. Choose a uniform mesh, size h, and denote the numerical values of w at $x = x_i$ by $w(x_i,h)$, and the actual values by $w(x_i)$. Then

$$w(x_i,h) = w(x_i) = \xi(x_i)h^2 + O(h^4) \qquad (2.A.2)$$

where $x_i = i*h$ and

$\xi(x_i)$ is a vector depending on x_i.

Similarly, for a uniform mesh, size h/2, we have

$$w(x_i,h/2) \cong w(x_i) + \xi(x_i)(h/2)^2 + O(h^4) \qquad (2.A.3)$$

From (2.A.2) and (2.A.3) we have

$$\xi(x_i) \cong (4/3h^2)(w(x_i,h) - w(x_i,h/2)) \qquad (2.A.4)$$

We make use of (2.A.4) in defining our relations and the associated subdivision technique. Our aim is to redistribute the points based on the magnitude of these error indicators.

We define $||\xi(x_i)||$ by

$$||\xi(x_i)|| = \sum_{j=1}^{m} \alpha_j |\xi_j(x_i)|$$

where m is the number of first order equations, $\xi_j(x_i)$ is the jth-component of $\xi(x_i)$ and $\Sigma\alpha_i = 1$.

2.B The Defining Relations and Subdivision Technique

Using (2.A.4) we define the weighted estimate of the total truncatin error at the point $x = x_i$ by

$$\begin{aligned} e_i &= 4/3h^2\{\alpha_1|w_1(x_i,h)-w_1(x_i,h/2)| + \alpha_2|w_2(x_i,h)-w_2(x_i,h/2)| +\ldots+ \\ &\qquad \alpha_m|w_m(x_i,h)-w_m(x_i,h/2)|\} \\ &= 4/3h^2||w(x_i,h)-w(x_i,h/2)||. \qquad (2.B.1) \end{aligned}$$

where $\alpha_1 + \alpha_2 + \ldots + \alpha_m = 1.$

e_i is thus a weighted norm of the error vector at the point $x = x_i$.

For the ith subinterval $[x_{i-1},x_i]$, with $x_i = a+(i-1)h$, we can define a measure of the relative size of the perceived discretization error as

$$s_i = f(e_{i-\frac{1}{2}}+d)^{-\frac{1}{2}}/T \tag{2.B.2}$$

where $e_{i-\frac{1}{2}} = (e_{i-1}+e_i)/2$

d is a 'safety margin' to avoid division by zero

$T = \sum_{k=1}^{N} (e_{k-\frac{1}{2}}+d)^{-\frac{1}{2}}$

f is a factor to be determined by our desired error tolerance.

The adaptive mesh subdivision then is to divide the original ith interval into

$$N_i = \lceil h/s_i \rceil \tag{2.B.3}$$

subintervals. (Here "$\lceil\ \rceil$" denotes the next largest integer function.) Hence, e.g., if $e_{i-\frac{1}{2}}$ were so small that s_i was larger than h then no subdivision of the original ith interval would occur. If we set $\theta_i = 1/N_i$, the new subinterval spacings are $\theta_i h$ in the old ith intervals.

We shall indicate how, based on the validity of the asymptotic estimate of discretization error, (2.A.2) we could expect the discretization errors for the adaptively generated mesh to be fairly uniform in size. The new approximate solution is $w(x,\theta_i h)$ for $a+(i-1)h \le x \le a+ih$ and (2.A.2) implies

$$w(x,\theta_i h) = w(x) + \xi(x)(\theta_i h)^2 + O(h^4) \tag{2.B.4}$$

But, using (2.B.2)

$$\xi(x)(\theta_i h)^2 = \xi(x)h^2/\lceil n/s_i \rceil^2$$
$$\tilde{=}\ \xi(x)s_i^2 \quad = (\xi(x)/(e_{i-\frac{1}{2}}+d)f^2T.$$

However, $e_{i-\frac{1}{2}} + d = (e_{i-1}+e_i)/2 + d$ is an estimate of $||\xi(x)||$ ((2.B.1) and (2.A.3)) so the vector $\ell = \xi(x)/(e_{i-\frac{1}{2}}+d)$ should be nearly a unit vector, and we have

$$w(x,\theta_i h) \ \tilde{=}\ w(x) + f^2T\ell \tag{2.B.5}$$

From (2.B.5) we can see how f should be chosen, for

$$T = \sum_{k=1}^{N} (e_{k-\frac{1}{2}}+d)^{-\frac{1}{2}}$$

is known prior to selecting the new mesh. If we wish to have an approximate solution with an absolute error not exceeding a tolerance E, we should get

$$f = c(E/T)^{\frac{1}{2}} \tag{2.B.6}$$

where c is a fraction $0 < c < 1$. In the tests reported on below, we took $c = 1/2$.

CHAPTER 3

Numerical Examples and Results

3.A The Numerical Scheme

The numerical scheme used to implement our technique is the "mid-point rule" of H.B. Keller [1], [2]. It is an $O(h^2)$ method which becomes an $O(h^4)$ method with $h \to 0$ extrapolation. Its implementation by Lam [5] is used.

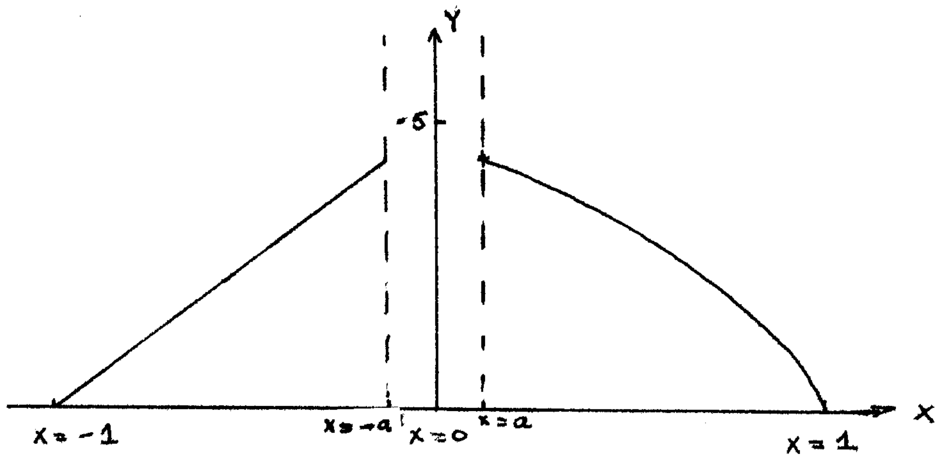
3.B Examples Solved

The examples solved are the two equations given as examples of equations of boundary layer type in Chapter 1. The third example is

$$-y''(x) = f_a(x) = \begin{cases} 0 & -1 \le x \le a \\ m(x+a)^2 & -a \le x \le 0 \\ 1-m(x-a)^2 & 0 \le x \le a \\ 1 & a \le x \le 1 \end{cases}$$

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

Its sketch is given below. In the interval $-a \le x \le a$, its solution is quartic.



Sketch of Example 3

As $a \to 0$

$$y(x) = cx + d \qquad x < 0$$
$$= -x^2 + cx + d \quad x > 0$$
$$c = d = 1/2$$

If we require $f_a(0) = 1/2$ we have $m = 1/2a^2$. Notice that this is not a boundary layer phenomenon equation in the usual sense, but it presents similar numerical difficulties. It is used to examine how our scheme reacts to a simple, non-smooth problem.

For Example 1, the boundary layer regions are near the points $x = -1$ and $x = 1$. For Example 2, we have an interior boundary layer region of width $O(\sqrt{\varepsilon})$ around the point $x = 0$. For Example 3, we expect a behaviour 'similar' to Example 2 in the interval $-a \le x \le a$.

We expect greater number of subdivisions in the boundary layer regions.

3.C Bases of Evaluation and Comparison

Let us review some notation which was introduced in Chapter 2. The boundary value problems for Examples 1, 2 and 3 are reduced to two first order equations of the form (2.A.1), i.e. $m = 2$, (see Appendix B). Denote by $w_k(x)$ the kth component of the exact solution, $k = 1,2$. We will use the presence of 'h' in the notation $w_k(x_i,h)$ to indicate that this is the value of the kth component of one of the discretized problems at a mesh point x_i. We will rely on the context to make it clear whether the original uniform mesh or the adaptively chosen mesh was used.

The error in $w(x_i,h)$ will be denoted by

$$Er_i = \alpha_1|w_1(x_i)-w_1(x_i,h)|+\alpha_2|w_2(x_i)-w_2(x_i,h)|$$

where the α_1, α_2 are the same as the weights used to compute the estimate of the discretization error in (2.B.1). The user specified desired absolute error tolerance will be denoted by E. We now introduce two parameters to try to quantify the performance of the adaptive scheme. One is the deficiency of the adaptive scheme defined as

$$\text{Deficiency} = (\max_i Er_i)|E$$

This parameter is a measure of the adaptive processes ability to meet the requested error tolerance. A deficiency of 1 would be highly desirable. The other parameter we call the uniformity of the adaptive scheme,

$$\text{Uniformity} = \log_{10}[\max_i Er_i|(\sum_{i=1}^{N} Er_i|N)]$$

It is a logarithmic comparison of the maximum error to the average error. A uniformity of zero would be highly desirable.

To compare the effectiveness of the non-uniform spacings chosen adaptively, we compare the results obtained on the non-uniform meshes with the results obtained by using a uniform mesh with the same number of subintervals. Hence the comparison uniform mesh is <u>not</u> the (relatively crude) uniform mesh used in the initial stages of the adaptive process. Nor is it the (relatively fine) mesh that the process would predict if the finest mesh spacing that was chosen adaptively were taken for a uniform mesh spacing. It is a uniform mesh which produces a problem of comparable magnitude to the non-uniform one.

For the tests made on the example problems, the problem parameters chosen were

Example 1	ε = .1, .01, .004
Example 2	ε = .01
Example 3	a = .05

and the method parameters used were $\alpha_1 = \alpha_2 = 1/2$, $c = 1/2$ (see (2.B.6)), and $d = E/10$.

3.D Results

A. Efficiency

To check on the accuracy and efficiency of our technique, in the sense of redistributing the mesh points in the appropriate regions, the three examples were solved using an initial number of mesh points of 21 (initial mesh size = .1). For desired tolerances $E = 10^{-2(.25)4}$, the number of subdivisions needed in each interval is shown in Tables 3.1, 3.2 and 3.3 for Examples 1, 2, 3 respectively. Figs.3.1, 3.2 and 3.3 are histograms of the distribution of points over the range of integration for $E = 10^{-3}$.

The scheme appears to be redistributing the points appropriately. The patterns shown conform to those of the solutions.

To further check on the adaptability of the scheme, Example 1 was solved with $\varepsilon = .1, .01, .004$. The mesh point distributions for $E = 10^{-2}$ are shown in Table 3.4 for each of these three cases. They show greater concentration of the points near the boundaries as ε decreases. They thus conform with the expected behaviour of the solutions for these values of ε.

B. Ability to meet Specified Tolerance

When the examples were solved using the newly generated mesh, we measured the deficiency (see section 3C for definition) of each solution for specified tolerances. The results are shown in Table 3.5.

It shows that for

TABLE 3.1

Example 1: Number of subdivisions of each interval for each tolerance

Initial N = 21;   E ≡ tolerance.

Interval \ $-\log_{10}E$	2	2.25	2.50	2.75	3.00	3.25	3.50	3.75
1	9	12	16	21	28	38	50	67
2	9	11	15	20	26	35	46	62
3	7	9	12	16	21	28	37	49
4	5	7	9	11	15	20	26	35
5	4	5	6	8	10	13	18	23
6	3	3	4	5	7	9	12	16
7	2	2	3	4	5	6	8	10
8	1	2	2	2	3	4	5	7
9	1	1	1	2	2	3	3	4
10	1	1	1	1	1	2	2	3
11	1	1	1	1	2	2	3	4
12	1	1	2	2	3	4	5	6
13	2	2	3	3	4	5	7	9
14	2	3	4	5	6	8	11	14
15	3	4	6	7	9	12	16	22
16	5	6	8	11	14	19	25	33
17	7	9	12	16	21	28	37	49
18	10	13	17	22	29	39	52	69
19	12	16	21	28	37	49	65	87
20	13	17	23	30	40	53	71	94
Final N = Total + 1	99	126	167	216	284	378	500	664

Note * Need about 3 times as many points when one goes from tolerance of 10^{-k} to tolerance of 10^{-k-1}.

TABLE 3.2

Example 2: Number of subdivisions of each interval for each tolerance

Initial N = 21; E ≡ tolerance

Interval \ $-\log_{10}E$	2	2.25	2.50	2.75	3.00	3.25	3.50	3.75	4.00
1	1	1	1	1	1	2	2	3	3
2	1	1	1	1	2	2	3	3	4
3	1	1	1	2	2	2	3	4	5
4	1	1	2	2	3	3	4	5	7
5	1	2	2	3	3	4	6	7	10
6	2	2	3	4	5	7	9	11	15
7	4	5	6	8	11	14	19	25	33
8	6	8	10	13	17	23	30	40	54
9	6	8	11	14	19	25	33	43	58
10	7	9	12	15	20	27	36	47	63
11	7	9	12	15	20	27	36	47	63
12	6	8	11	14	19	25	33	43	58
13	6	8	10	13	17	23	30	40	54
14	4	5	6	8	11	14	19	25	33
15	2	2	3	4	5	7	9	11	15
16	1	2	2	3	3	4	6	7	10
17	1	1	2	2	3	3	4	5	7
18	1	1	1	2	2	2	3	4	5
19	1	1	1	1	2	2	3	3	4
20	1	1	1	1	1	2	2	3	3
Final N = Total+1	61	77	99	127	167	219	291	377	505

* Same comments as below Table 3.1

TABLE 3.3

Example 3: Number of subdivisions of each interval for each tolerance

Initial N = 21; E ≡ tolerance

Interval \ $-\log_{10}E$	2	2.25	2.50	2.75	3.00	3.25	3.50	3.75	4.00	4.25
1	1	1	1	1	1	1	2	2	3	3
2	1	1	1	1	1	1	2	2	3	3
3	1	1	1	1	1	1	2	2	3	4
4	1	1	1	1	1	2	2	2	3	4
5	1	1	1	1	1	2	2	2	3	4
6	1	1	1	1	1	2	2	2	3	4
7	1	1	1	1	1	2	2	2	3	4
8	1	1	1	1	1	2	2	2	3	4
9	1	1	1	1	1	2	2	2	3	4
10	1	2	2	3	3	4	6	9	10	13
11	1	2	2	3	3	4	6	9	10	13
12	1	1	1	1	1	2	2	2	3	4
13	1	1	1	1	1	2	2	2	3	4
14	1	1	1	1	1	2	2	2	3	4
15	1	1	1	1	1	2	2	2	3	4
16	1	1	1	1	1	2	2	2	3	4
17	1	1	1	1	1	2	2	2	3	4
18	1	1	1	1	1	2	2	2	3	4
19	1	1	1	1	1	2	2	2	3	3
20	1	1	1	1	1	2	2	2	3	3
Final N = Total + 1	21	23	23	25	25	39	49	55	75	95

* Observation on Example 1 and Example 2 does not hold in this case.

Fig.3.1 Example 1: Number of subdivisions of each interval
Epsilon = 10^{-2}, Tolerance = 10^{-3}

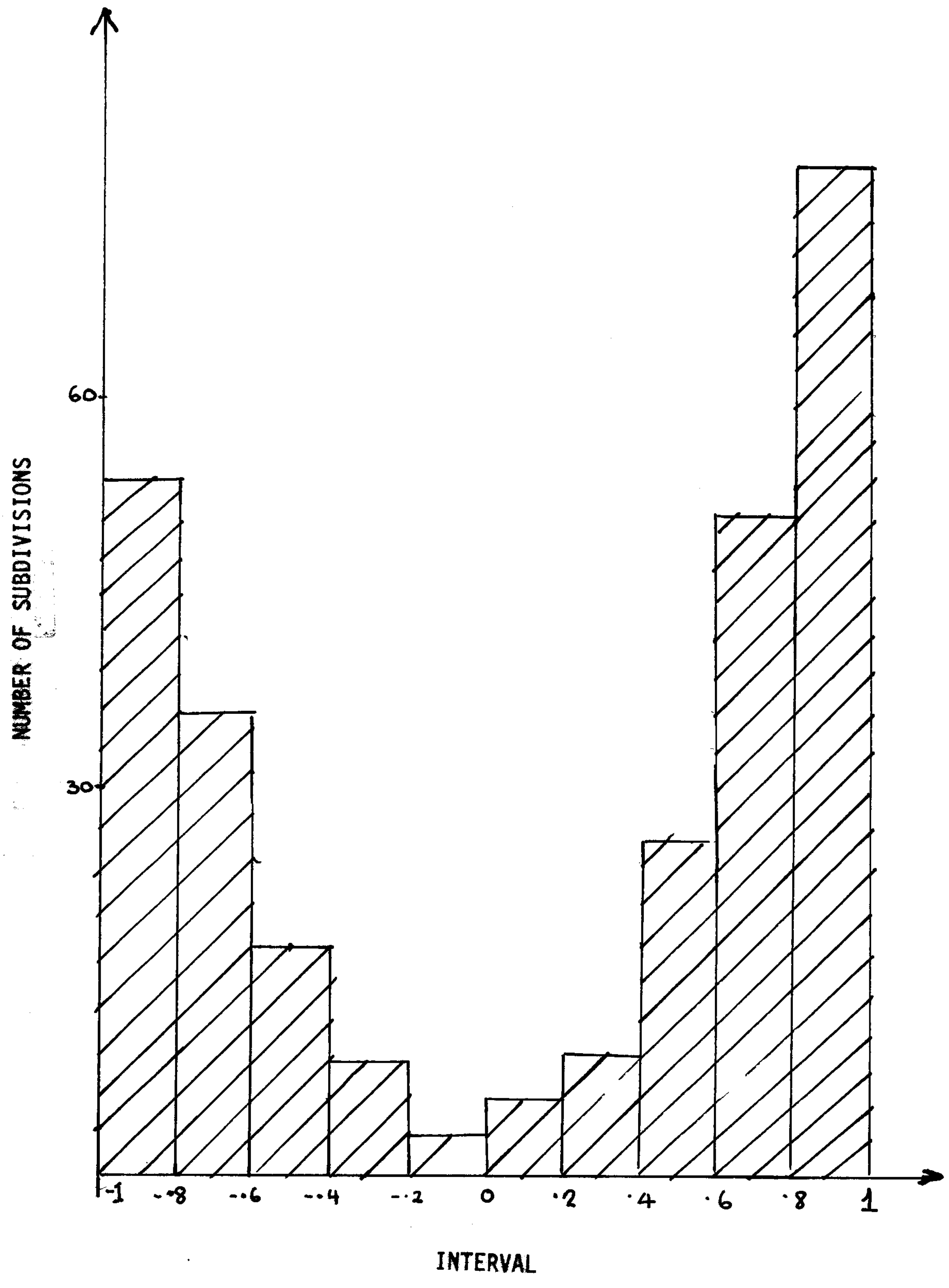


Fig.3.2 Example 2: Number of subdivisions of each interval
Epsilon = 10^{-2}, Tolerance = 10^{-3}

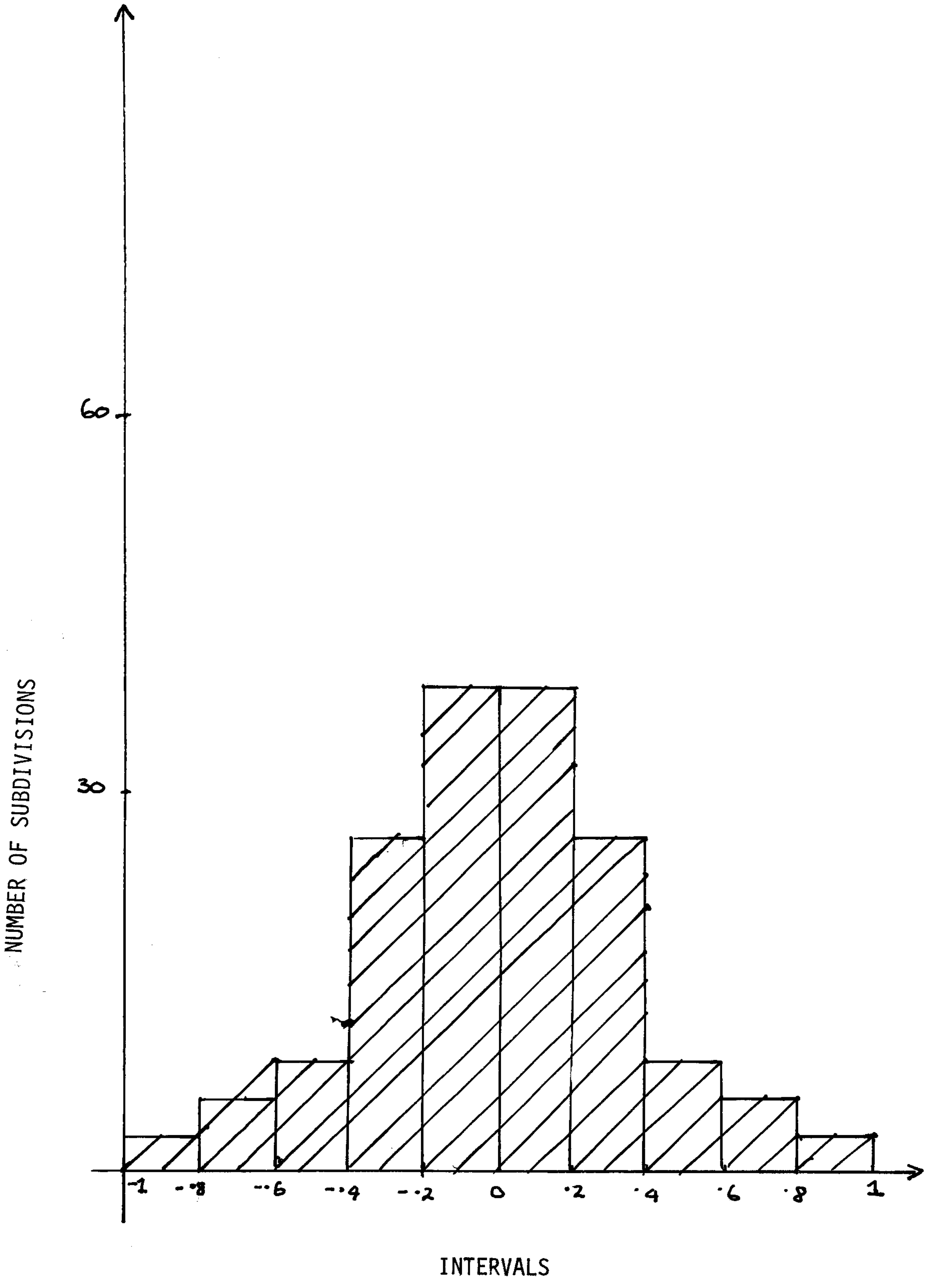


Fig.3.3 Example 3: Number of subdivisions of each interval
Epsilon = 10^{-2}, Tolerance = 10^{-3}

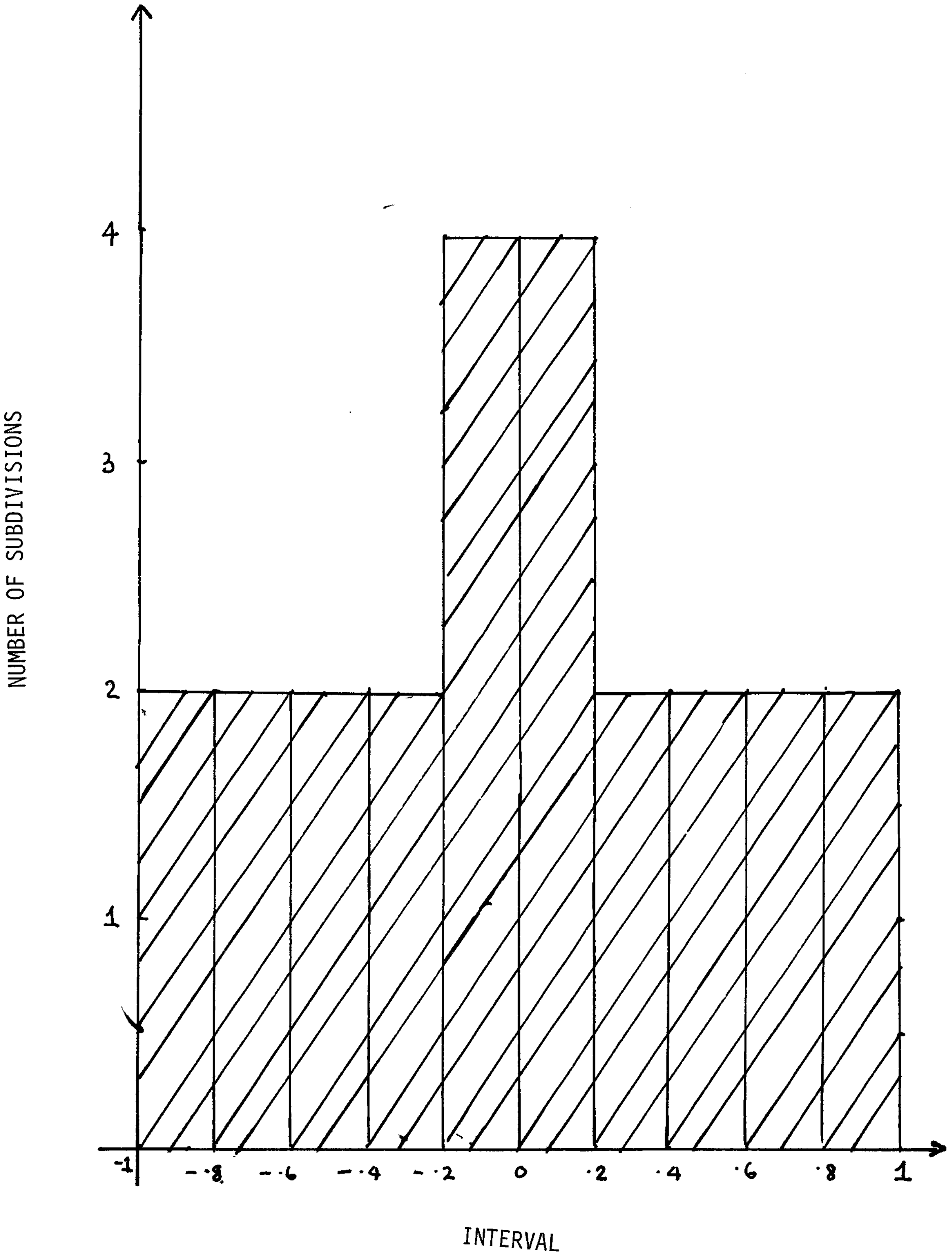


TABLE 3.4

Example 1: Dependence of subdivisions on size of ε (equation parameter)

Tolerance = 10^{-3}; Initial N = 21

Interval \ ε	.1	.01	.004
1	1	9	18
2	2	9	15
3	2	7	8
4	2	5	4
5	2	4	2
6	2	3	1
7	2	2	1
8	1	1	1
9	1	1	1
10	2	1	1
11	2	1	1
12	2	1	1
13	2	2	1
14	2	2	1
15	3	3	2
16	3	5	3
17	3	7	6
18	3	10	11
19	2	12	21
20	2	13	26
Final N = Total + 1	42	99	126

Example 1: maximum error $\cong$ .4 $\times$ E

Example 2: maximum error $\cong$ .6 $\times$ E

Example 3: maximum error $\cong$ 3.0 $\times$ E

For Examples 1 and 2, the results more than met the desired tolerance and the results for Example 3 are not bad.

C. <u>How Uniform is the Error?</u>

After the redistribution of the mesh points, we expect the errors to be fairly uniformly distributed over the interval of integration. For our examples, the uniformity (see section 3.C for definition) of each solution for each specified tolerance is shown in Table 3.6. For the three cases, the uniformity decreased with N - the final number of points generated.

For Example 1, maximum error $\cong$ 1.7 $\times$ average error

For Example 2, maximum error $\cong$ 2.2 $\times$ average error

For Example 3, 1.7 $\times$ average error $<$ maximum error $<$ 25.12 $\times$ average error

These mean that for Example 1, the maximum error is less than twice the average error (actually $\approx$ 1.66 $\times$ average error). For Example 2, the maximum error is about twice the average error (2.24 $\times$ average error). For Example 3, the maximum error ranges from less than twice average error to about twenty-five times the average error.

The errors, therefore, appear to be fairly uniformly distributed over the range of integration.

D. <u>Non-uniform Mesh Solution versus Uniform Mesh Solution</u>

After solving the Examples with a non-uniform mesh, they were also solved with a uniform mesh using the same number of points generated by the adaptive technique.

The deficiencies and uniformities of the solutions for specified tolerances are given in Tables 3.5 and 3.6, respectively.

Fig.3.4 shows graphs of the errors in solving Example 2 (with $\varepsilon = 10^{-3}$, number of points generated was 167) with a non-uniform mesh and with a uniform mesh respectively. Maximum error is expected at the centre of the interval.

In Fig.3.4, the distribution of errors as they occur for Example 2 is compared for the adaptively chosen mesh and for the comparison uniform mesh. The 'X''s are marked on the error curve at the positions of the mesh nodes used. It can be seen that the adaptive mesh has bunched the nodes near the origin and left them spread out near the ends of the interval. The basic distribution of error is similar in both cases; with the adaptive mesh having **reduced the peak error by a factor of about 10.**

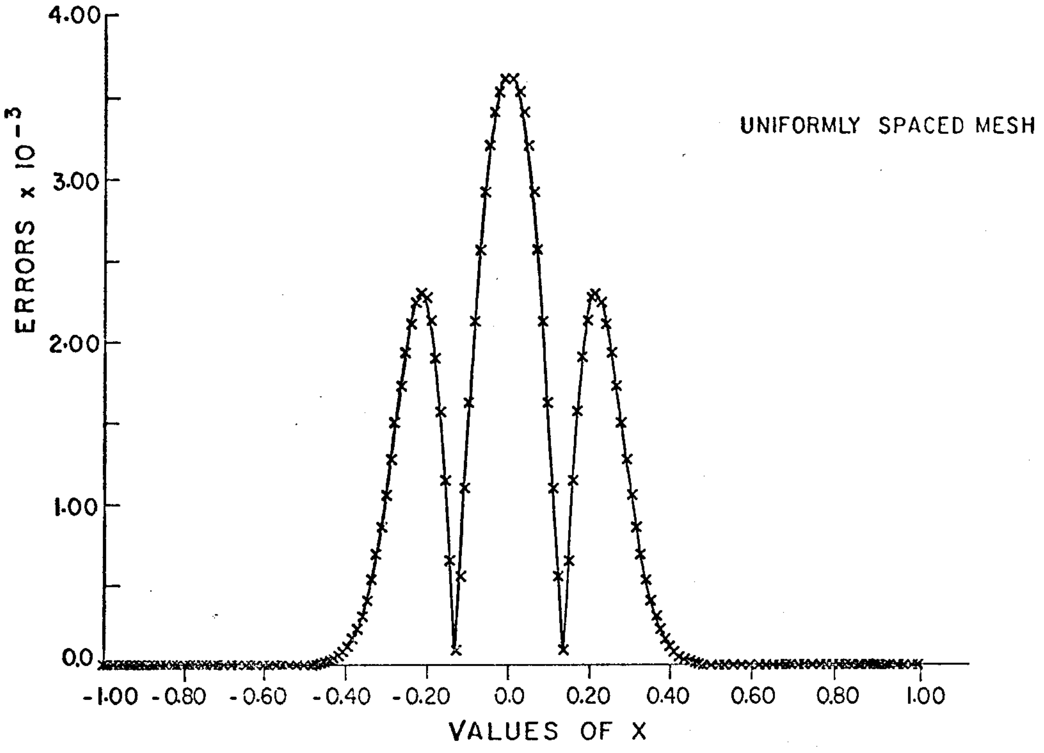
E. Dependence of Final Number of Mesh Points on Tolerance

It has been observed recently by Malcolm and Simpson [7] and Rice [8] for adaptive quadrature that the order of the method used is preserved, in a sense. In the case of a 2nd order method, the sense is that the selected number of subintervals (N_{final}), is expected to be proportional to $E^{-1/2}$. This appears to be the case in our scheme.

Starting with N = 21 points the examples were solved and the final number of mesh points generated, (N_{final}), were recorded for each tolerance. See Table 3.7. Fig.3. contains graphs of $\log_{10}(N_{final})$ vs $-\log_{10}E$ for the three examples. For examples 1 and 2, the results give straight lines **with slopes nearly equal to 1/2. Therefore, $\log_{10}N \cong -\frac{1}{2}\log_{10}E + K$, i.e. $N_{final} \propto E^{-1/2}$.**

The erratic behaviour of Example 3 is probably due to its lack of smoothness. In Fig.3.5, its behaviour for bigger tolerances ($-\log_{10}$(tolerance) in the range 2.0 to 3.25) appears to be fourth order; however, it seems to climb rather unsteadily like a second order method for smaller tolerances.

Fig.3.4 Comparison of Error Distribution from Uniform and Adaptively chosen Meshes



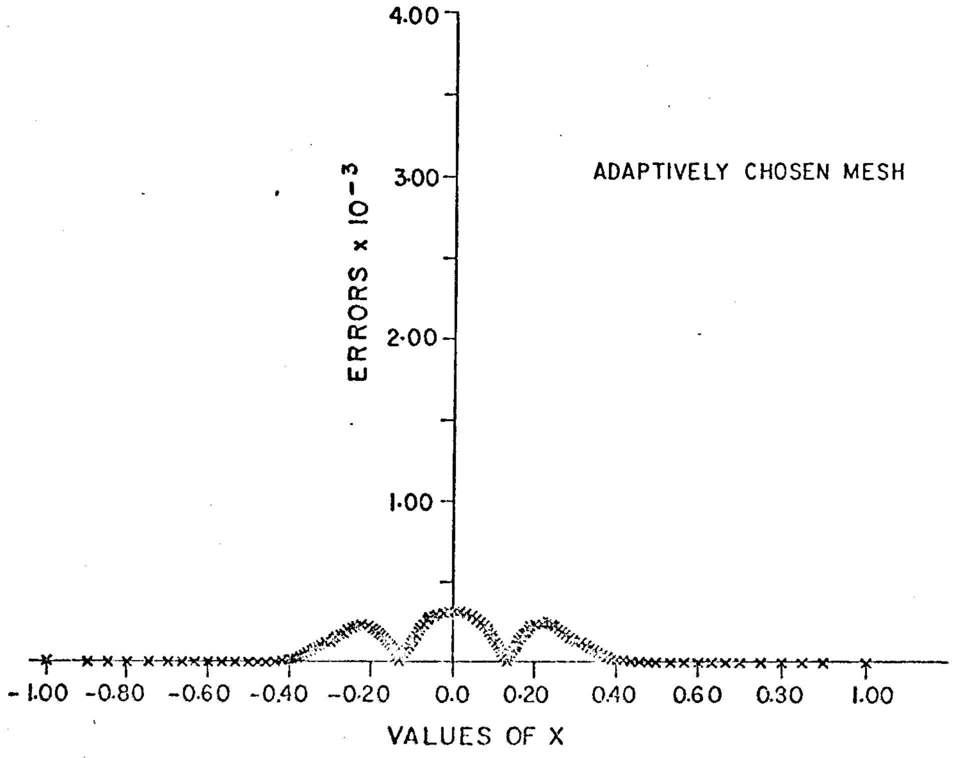


TABLE 3.5

Deficiency of Solutions for each tolerance.

$\log_{10}E$	Deficiency					
	Example 1		Example 2		Example 3	
	Non-Uniform Mesh	Uniform mesh	Non-Uniform mesh	Uniform mesh	Non-Uniform mesh	Uniform mesh
2	.416	2.82	.548	2.78	.844	.844
2.25	.422	3.08	.580	3.08	.416	.910
2.50	.420	3.10	.574	3.28	.742	1.618
2.75	.432	3.28	.646	3.54	.648	2.728
3.00	.434	3.36	.642	3.62	1.15	4.846
3.25	.432	3.38	.636	3.74	1.32	7.092
3.50	.434	3.42	.640	3.76	1.408	11.830
3.75	*	*	.670	3.98	2.15	20.476
4.00	*	*	*	*	2.938	34.982

Note: * indicates those deficiencies which could not be obtained because the number of points generated by the scheme was more than we wanted to allow in our programs.

TABLE 3.6

Uniformity of Solutions for each tolerance.

$-\log_{10}E$	UNIFORMITY					
	Example 1		Example 2		Example 3	
	Non-uniform mesh	Uniform mesh	Non-Uniform mesh	Uniform mesh	Non Uniform mesh	Uniform mesh
2	.252	.697	.363	.755	1.322	1.322
2.25	.232	.695	.361	.753	.857	1.333
2.50	.231	.694	.347	.748	.857	1.333
2.75	.223	.693	.362	.749	.659	1.308
3.00	.219	.693	.352	.749	.659	1.308
3.25	.216	.692	.342	.748	.544	.964
3.50	.213	.692	.340	.748	.380	.784
3.75	*	*	.337	.748	.319	.737
4.00	*	*	*	*	.230	.659

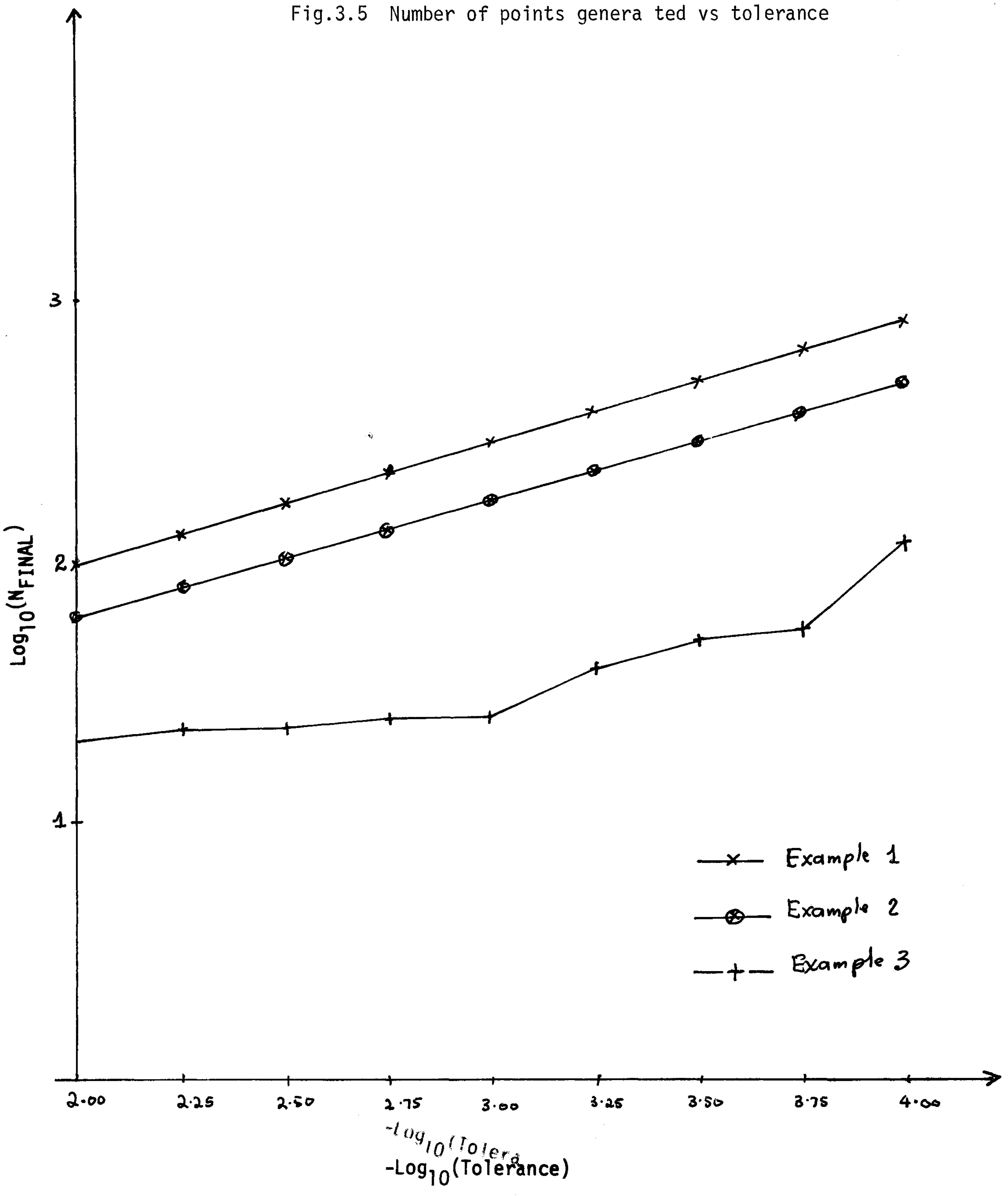
Note: * (see note below Table 3.5)

TABLE 3.7

DEPENDENCE OF FINAL N ON TOLERANCE DESIRED (E)

$-\log_{10}E$	EXAMPLE 1		EXAMPLE 2		EXAMPLE 3	
	N	$\text{Log}_{10}N$	N	$\text{Log}_{10}N$	N	$\text{Log}_{10}N$
2	99	1.996	61	1.785	21	1.322
2.25	126	2.100	77	1.886	23	1.362
2.50	167	2.233	99	1.996	23	1.362
2.75	216	2.334	127	2.104	25	1.398
3.00	284	2.453	167	2.223	25	1.398
3.25	378	2.577	219	2.340	39	1.591
3.50	500	2.699	291	2.464	49	1.690
3.75	664	2.822	377	2.576	55	1.740
4.00	864	2.94	505	2.703	75	1.875

Fig.3.5 Number of points genera ted vs tolerance



F. Choice of Initial N

Because the first two solutions, $\bar{w}(x,h)$ and $\bar{w}(x,h/2)$ are used only to generate new mesh spacings, they are in a sense wasted thereafter. It is therefore desirable and important that one does not start with too many points and yet should be able to generate enough points to satisfy the desired tolerance.

To check if our scheme has this capability, our examples were solved with different initial values of N. The final value of N generated for each tolerance were recorded and are shown in Tables 3.8, 3.9 and 3.10. The deficiencies are also shown (Tables 3.11, 3.12 and 3.13).

For examples 1 and 2, the desired tolerances were satisfied in all cases. For example 3, they were satisfied in some cases. As expected, all the desired tolerances tend to be satisfied as final N gets larger. In this case, more points fall into the boundary layer region and a better distribution of points is obtained.

For example 1, final N varies little with initial N for each tolerance. This does not appear to be the case in the other two examples.

From these results, there does not appear to be a clearcut guideline for choosing initial N so as to achieve desired tolerance. It will have to depend on the problem being solved.

TABLE 3.8

EXAMPLE 1: VARIATION OF FINAL N WITH INITIAL N

$-\log_{10}E$ \ Initial N	9	11	13	15	17	19	21
2	110	99	99	97	97	97	99
2.25	146	127	129	127	127	126	126
2.50	193	168	169	168	166	164	167
2.75	254	224	222	220	217	219	216
3.00	335	297	293	291	289	286	284
3.25	447	392	388	383	381	380	378
3.50	594	521	516	512	506	501	500
3.75	792	693	687	680	669	665	664
4.00	1054	922	912	902	894	884	881

TABLE 3.11

Example 1: Variation of Deficiency with Initial N.

$-\log_{10}E$ \ Initial N	9	11	13	15	17	19	21
2	.218	.200	.194	.191	.206	.208	.208
2.25	.213	.196	.198	.196	.194	.205	.211
2.50	.213	.202	.195	.200	.198	.204	.210
1.75	.217	.205	.203	.201	.205	.204	.216
3.00	.215	.206	.203	.198	.203	.206	.217
3.25	.219	.206	.211	.203	.203	.210	.216
3.50	*	*	*	*	*	*	*
3.75	*	*	*	*	*	*	*
4.00	*	*	*	*	*	*	*

* - the number of points generated was larger than the number allowed in the program (500)

TABLE 3.9

Example 2: Variation of Final N with Initial N

$-\log_{10}E$ \ Initial N	9	11	13	15	17	19	21
2	125	51	89	85	71	59	61
2.25	165	63	113	111	89	73	77
2.50	217	83	147	145	113	95	99
2.75	289	111	193	187	153	123	127
3.00	381	143	255	249	201	163	167
3.25	507	189	343	331	267	215	219
3.50	679	251	453	439	351	283	291
3.75	903	333	599	583	463	373	377
4.00	1199	441	797	773	613	495	505

TABLE 3.12

Example 2: Variation of Deficiency with Initial N

$-\log_{10}E$ \ Initial N	9	11	13	15	17	19	21
2	.241	.338	.356	.398	.316	.316	.274
2.25	.241	.378	.376	.380	.347	.340	.290
2.50	.241	.394	.392	.416	.365	.340	.287
2.75	.249	.369	.416	.405	.344	.341	.323
3.00	.250	.399	.425	.415	.370	.327	.321
3.25	*	.392	.419	.425	.367	.334	.318
3.50	*	.396	.437	.428	.374	.347	.320
3.75	*	.393	*	*	.387	.348	.335
4.00	*	.399	*	*	*	.356	*

* - number of points generated was larger than allowed in the program (500).

TABLE 3.10

Example 3: Variation of Final N with Initial N

Initial N / $-\log_{10}E$	13	15	17	19	21	31	41
2	13	15	17	19	21	31	41
2.25	13	15	19	21	23	48	41
2.50	13	17	19	21	23	49	41
2.75	13	17	19	23	25	69	43
3.00	15	19	21	23	25	91	43
3.25	15	19	29	35	39	115	43
3.50	17	31	39	43	49	141	45
3.75	23	35	41	49	55	192	57
4.00	29	45	57	67	75	247	87

TABLE 3.13

Example 3: Variation of Deficiency with Initial N

Initial N / $-\log_{10}E$	13	15	17	19	21	31	41
2	.422	.422	.422	.422	.422	.242	.117
2.25	.750	.750	.371	.276	.208	.105	.208
2.50	**	**	.661	.491	.371	.182	.371
2.75	**	**	**	.346	.324	.168	.209
3.00	**	**	.748	.616	.577	.223	.371
3.25	**	**	.986	.821	.660	.326	.660
3.50	**	**	**	.944	.709	.496	.704
3.75	**	**	**	**	**	.752	.961
400	**	**	**	**	**	**	**

** - deficiency greater than 1

CHAPTER 4

Conclusions and Possibilities

Our scheme appears to possess some desirable properties of an adaptive mesh-generation scheme.

New mesh points are inserted in the appropriate places. It appears that so long as the numerical scheme used to implement it is stable on a uniform mesh, it is also stable on the non-uniform mesh generated.

It appears that one need not start with too many point initially, but it is not clear how one should choose this number. It will have to depend on the problem being solved.

The solutions obtained from the non-uniform mesh are more accurate than those obtained from the uniform mesh (using the same number of points generated by the scheme). The errors are smaller and more evenly spread over the interval of integration. However, these effects are not strongly pronounced.

How does one know that he has generated enough mesh points to give a solution whose deficiency is less than 1? There does not appear to be a clear dependence of the final number of points generated on the initial number of points, as shown in Table 3.9. Moreover, there may be no way of knowing the deficiency of a solution before actually obtaining the approximate solution.

So the question is whether the advantages obtained from our scheme are worth the overhead - extra number of computations and code - involved? The answer, at this stage, may be 'no'. But non-uniform meshes have been effectively used in solving several problems and have shown remarkable improvement over uniform mesh solutions.

We have, here, only explored a possible method of automating the mesh-generation and we believe that the effectiveness of our scheme can be improved. It has to be more fully tested.

Possibilities

Several possibilities exist in the application of our scheme.

1. If one knew beforehand where the boundary layer region(s) are, it may be advantageous to start with a non-uniform mesh, smaller mesh size in the boundary layer region(s), larger mesh size in the other places.

2. The choice $\alpha_i = 1/m$ $1 \le i \le m$ for the error weights, may not be ideal. A different choice of α_i may give better distribution of points and better results. For a given problem there may be an optimal choice of α_i's. To determine this may be a more difficult subproblem than the origianl problem. It will be interesting to know what happens when $\alpha_1=1$, $\alpha_i=0$, $i \ne 1$ or when $\alpha_k=1$, $\alpha_i=0$, $i \ne k$.

3. Different choices of the scale factor, c, are possible. If the deficiency of a solution is greater than 1, it may be necessary to choose a new scale factor (smaller than the original one) so as to increase the number of points generated. If the deficiency is very much less than 1, than one may have used more points than necessary and a larger scale factor could be used subsequently to reduce the number of points and computational effort.

4. For time-dependent problems which exhibit boundary layer phenomenon, it may be advantageous to redistribute the points after a certain number of time steps. This will result in doing fewer computations and getting good results or doing more computations and getting better results.

5. It is possible to choose a certain fixed number $k>0$, such that if the number of subdivisions of any interval is greater than k, then one applies the adaptive scheme to only that interval to obtain a better distribution.

It is not clear how to choose k and it is doubtful if the accuracy obtained is worth the overhead. A choice of k may result in very many of the intervals being further subdivided. This is time consuming.

APPENDIX A

IMPLEMENTATION

a. Computational Steps

Subject to the constraints outlined above, the basic computational steps are:

1. Choose initial number of mesh points and tolerance.
2. Solve the system (uniform mesh). Store results.
3. Double the number of mesh points.
4. Solve the system again (uniform mesh). Store results.
5. Use results of steps (2) and (4) to generate error estimates.
6. Use estimates in (5) to obtain the number of subdivisions of each original interval.
7. Generate new (non-uniform) mesh.
8. Solve the system.

b. Routines

The routines for carrying out steps 2, 4 and 8 depend on the numerical scheme being used. Possible routines for carrying out steps 5 and 6 are given below:

SUBROUTINE ERROR - For step 5 ($m = 2$)

SUBROUTINE NUMESH - For step 6

The parameters passed through these routines are explained within them.

```
      SUBROUTINE NUMESH( JE , EROR, C, EPSLON, DX,THETA, JENEW)
C  THIS SUBROUTINE GENERATES THE NEW SUBDIVISION.
C
C  JE ----- NUMBER OF MESH POINTS --- OLD ONE.
C  EROR ----- VECTOR OF EROR ESTIMATES OBTAINED FROM "SUBROUTINE EROR"
C  C ----- FACTOR FOR DETERMINING SCALE FACTOR 0 <= C <= 1.
C  EPSLON ----- DESIRED TOLERANCE.
C  DX ----- VECTOR CONTAINING INTERVAL LENGTHS.
C  THETA ----- VECTOR CONTAINING THE NUMBER OF SUBDIVISIONS OF EACH
C              INTERVAL.
C  JENEW ----- NEW TOTAL NUMBER OF MESH POINTS.
C
       DIMENSION EROR(JE), DX(JE), THETA(JE), ER(JE)
C
C  ER ----- VECTOR CONTAINING THE AVERAGE OF THE ERRORS AT TWO
C           SUCCESSIVE POINTS.
C
       SUM = 0.
       JJ = JE/2
C
C  SET 'SAFETY MARGIN' (D) EQUAL TO EPSLON / 10.
C
       D = EPSLON * (.1)
C
C  OBTAIN THE AVERAGES AND THEIR SUM.
C
       DO 1 I = 1 , JJ
          ERR = (EROR(I) + EROR(I + 1)) * (.5)
          ER(I) = (ERR + D) ** (-.5)
          SUM = SUM + ER(I)
     1 CONTINUE
C
C  TAKE THE SQUARE OF THE SUM
C
       SUMSQ = SUM * SUM
C
C  OBTAIN STEP FUNCTION VALUES AND NUMBER OF SUBDIVS. OF EACH INTERVAL
C
       SCALEF = C * SQRT(SUMSQ * EPSLON)
       DO 2 I = 1 , JJ
          S = (SCALEF * ER(I)) / SUM
          THETA(I) = DX(I) / S + 1
     2 CONTINUE
C
C  COUNT THE NUMBER OF MESH POINTS GENERATED
C
       ISUM = 0
       DO 3 K = 1 , JJ
            KK = THETA(K)
            DO 3 I = 1 , KK
                 ISUM = ISUM + 1
     3 CONTINUE
       JENEW = ISUM + 1
       RETURN
       END
?
```

```
    SUBRØUTINE ERRØR(JE, U1, U2, DX, ALPHA, ERØR)
C  SUBRØUTINE CØMPUTES  THE "TØTAL" ERRØR ESTIMATES (SEE 2.B.1)
C
C  JE ----- THE NUMBER ØF MESH PØINTS, INTERNAL + BØUNDARY.
C  U1 ----- THE SØLUTIØN ØF THE SYSTEM WITH UNIFØRM MESH SIZE H.
C  U2 ----- THE SØLUTIØN ØF THE SYSTEM WITH UNIFØRM MESH SIZE H/2.
C  DX ----- INTERVAL LENGTHS. DX(I) = X(I + 1) - X(I).
C  ALPHA ----- ARRAY ØF WEIGHTS. (SEE 2.B.1)
C  ERØR ----- VALUES ØF ERRØRS E(I) (SEE 2.B.1). THIS IS THE
C             ØUTPUT ØF THIS SUBRØUTINE.
C
     DIMENSIØN U1(JE,2), U2(JE,2), DX(JE), ALPHA(2), ERØR(JE)
C  THE DIMENSIØN ØF ALPHA SHØULD BE EQUAL TØ THE NUMBER ØF
C  FIRST ØRDER EQUATIØNS IN THE SYSTEM.
C
   JJ = JE/2 + 1
   DØ   1 I = 1,JJ
     KK = 2* I - 1
     A = ABS(U1(I,1) - U2(KK,1))
     B = ABS(U1(I,2) - U2(KK,2))
     BØT = 3.0 * DX(I) * DX(I)
     TØP = 4. *(ALPHA(1) * A + ALPHA(2) * B)
     ERØR(I) = TØP/BØT
 1 CØNTINUE
   RETURN
   END
?
```

APPENDIX B

BREAKING UP THE GIVEN EQUATION(S) INTO A SYSTEM OF FIRST ORDER EQUATIONS

We demonstrate with a few examples.

Consider Examples 1 and 2 (Chapter 3)

1. $\varepsilon y'' - y = 0$

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

   $0 < \varepsilon << 1$

2. $\varepsilon y'' + xy' = 0$

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

   $0 < \varepsilon << 1$

We can break them up as follows:

1. Let $y \equiv w_1$

   $y' \equiv w_2$

Then $\varepsilon w_2' = w_1$

$w_1' = w_2$

$w_1(-1) = 1, \quad w_1(1) = 2$

Thus

$$D = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix}$$

$$C = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

$$w' = \begin{pmatrix} w_1' \\ w_2' \end{pmatrix}$$

$$p_a = (1, 0)$$

$$p_b = (1, 0)$$

$$g_a = 1$$

$$g_b = 2$$

Alternatively, we can break it up as follows:

$$\text{Let } y \equiv w_1$$

$$y' \equiv w_1' = w_2/\sqrt{\varepsilon}$$

$$\therefore \sqrt{\varepsilon}\, w_2' = w_1$$

$$w_1(-1) = 1, \quad w_1(+1) = 2.$$

Thus

$$D = \begin{pmatrix} \sqrt{\varepsilon} & 0 \\ 0 & \sqrt{\varepsilon} \end{pmatrix}$$

$$C = \begin{pmatrix} w_2 \\ w_1 \end{pmatrix}$$

w', p_a and p_b, g_a and g_b remain the same.

2. Let $y \equiv w_1$

$$y' \equiv w_2 = w_1'$$

Then $w_2' = -xw_2$

$$w_1' = w_2$$

$$w_1(-1) = 1, \quad w_1(+1) = 2$$

Thus

$$D = \begin{pmatrix} \sqrt{\varepsilon} & 0 \\ 0 & \sqrt{\varepsilon} \end{pmatrix}$$

$$C = \begin{pmatrix} w_2 \\ w_1 \end{pmatrix}$$

w', p_a and p_b, g_a and g_b remain the same.

2. Let $y \equiv w_1$

$$y' \equiv w_2 = w_1'$$

Then $\varepsilon w_2' = -xw_2$

$$w_1' = w_2$$

$$w_1(-1) = 1, \quad w_1(+1) = 2$$

Thus

$$D = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix}$$

$$C = \begin{pmatrix} w_2 \\ -xw_2 \end{pmatrix}$$

$$p_a = (1,0), \quad p_b = (1,0)$$

$$g_a = 1, \quad g_b = 2.$$

Alternatively let $y \quad w_1$

$$y' = w_2/\sqrt{\varepsilon} = w_1'$$

Then $\varepsilon y'' = w_2'/\sqrt{\varepsilon} = \sqrt{\varepsilon}\, w_2' = -xw_2/\sqrt{\varepsilon}$

$$w_1(-1) = 1, \quad w_1(+1) = 2.$$

Thus

$$D = \begin{pmatrix} \sqrt{\varepsilon} & 0 \\ 0 & \sqrt{\varepsilon} \end{pmatrix}$$

$$C = \begin{pmatrix} w_2 \\ -xw_2/\sqrt{\varepsilon} \end{pmatrix}$$

p_a, p_b, g_a, g_b remain the same.

Remarks

If one is interested in the values of y' or higher other derivatives, then the first method of breaking up is preferable to the second. This is because of the division by $\sqrt{\varepsilon}$ in the second method. If ε is very small, then the estimates of the derivates will be very poor.

If one is interested in only y, then any of the methods is fine.

Our computations in the Chapter 3 were done using the first method.

References

[1] Keller, H.B., "Accurate Difference Methods for Linear Ordinary Differential Systems subject to Linear Constraints", SIAM Journal of Numerical Analysis 6(1969), pp.8-30.

[2] Keller, H.B., "A New Difference Scheme for Parabolic Problems", SYNSPADE, B. Hubbard, ed. Academic Press, New York, 1971.

[3] Pearson, C.E., "A Numerical Method for Ordinary Differential Equation of Boundary Layer Type", J. Math. Physics 47 (1968), pp.134-154.

[4] Pearson, C.E., "On non-linear Ordinary Differential Equations of Boundary Layer Type", Journal of Math. Physics 48 (1969), pp.351-358.

[5] Lam, D.C.L., "Implementation of the Box Scheme and Modal Analysis of Diffusion-Convertion Equations", Ph.D. Thesis, U. of Waterloo (1974).

[6] Carrier, G.F., Krock, M. and Pearson, C.E., "Functions of a Complex Variable: Theory and Technique", McGraw Hill (1966), pp.294.

[7] Malcolm, M.A., Simpson, R.B., "Local vs Global Strategies for Adaptive Quadrature", U. of Waterloo, Dept. of Comp. Sci., Technical Report CS-74-07, May 1974, pp.33-34.

[8] Rice, J., "An Educational Adaptive Quadrature Algorithm", SIGNUM Newsletter vol.8, No.2, April 1973 (27-41).