SU326P30-26

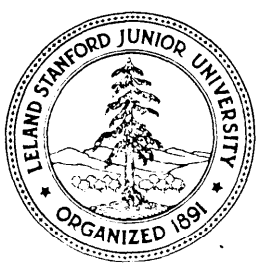
HIGH ORDER FINITE DIFFERENCE SOLUTION OF DIFFERENTIAL EQUATIONS

by

VICTOR PEREYRA

STAN-CS-73-348
April 1973

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY



HIGH ORDER FINITE DIFFERENCE SOLUTION OF DIFFERENTIAL EQUATIONS

V. Pereyra*

April 1973

*This report was written while the author was a Visiting Professor during the Winter quarter 1972-73. Computing and other support was provided by Contract NSF GJ-35135X. Permanent address: Departamento de Computación, Fac. Ciencias, Univ. Central de Venezuela, Apartado 59002, Caracas.

HIGH ORDER FINITE DIFFERENCE SOLUTION OF DIFFERENTIAL EQUATIONS

V. Pereyra

Abstract

These seminar notes give a detailed treatment of finite difference approximations to smooth nonlinear two-point boundary value problems for second order differential equations. Consistency, stability, convergence, and asymptotic expansions are discussed. Most results are stated in such a way as to indicate extensions to more general problems, Successive extrapolations and deferred corrections are described and their implementations are explored thoroughly. A very general deferred correction generator is developed and it is employed in the implementation of a variable order, variable (uniform) step method. Complete FORTRAN programs and extensive numerical experiments and comparisons are included together with a set of 48 references.

TABLE OF CONTENTS

I. INTRODUCTION. . . . . .

II. TWO POINT BOUNDARY VALUE PROBLEMS FOR NONLINEAR SECOND ORDER DIFFERENTIAL EQUATIONS 8

1. The problem and its discretization 8

2. Consistency, stability and convergence 12

3. An asymptotic expansion for the global discretization error 16

4. Successive extrapolations 19

5. Some comments on implementation 20

6. Deferred corrections 22

6.1. Introduction 22

6.2. Algorithms 25

6.3. An $O(h^8)$ method for the price of an $O(h^2)$ method 29

6.4. Some numerical results , ... 32

6.5. Discussion of' results and comparisons 38

6.6. A FORTRAN IV program for the $O(h^8)$ method of II.6.3 41

6.7. Operation count 50

Successive extrapolations (SE) 52

III. COMPUTER IMPLEMENTATION OF ITERATED DEFERRED CORRECTIONS FOR BOUNDARY VALUE PROBLEMS 61

1. An automatic weight generator for numerical differentiation and other applications 61

2. A Universal 2-point boundary value Deferred Correction Generator 67

3. Asymptotic error estimation by deferred correction 70

4. A variable order, variable (uniform) step, two point boundary value problem solver, based on deferred corrections 72

5. Numerical results 78

REFERENCES.....* 83

HIGH ORDER FINITE DIFFERENCE SOLUTION OF DIFFERENTIAL EQUATIONS

V. Pereyra

I. Introduction

These notes correspond to a six-week Seminar offered during the Winter quarter 1972-73. In them, we intend to give an overview on certain general techniques that permit the increase of the order of accuracy of simple discretizations to differential equations. Also, we will examine in detail one specific application. This will lead us naturally to consider some efficient tools which will permit the graceful implementation of the methods.

We shall consider the basic ideas in relation to a simple application: the two point boundary problem.

(1.1a) $-y''(x) + f(x,y) = 0$,

(1.1b) $y(a) = \alpha$, $y(b) = \beta$.

Most of the elements of the general theory are present here and we shall emphasize those points which are basic and can be transferred to other applications.

The problem and an $O(h^2)$ discretization are presented in Chapter II. The notions of consistency, stability and convergence are developed, and an asymptotic expansion for the global discretization error is obtained.

The method of successive extrapolations is introduced in Section II.4 together with some comments on implementation.

In Section 11.6, the method of deferred corrections is treated. An algorithm for obtaining an $O(h^8)$ discrete approximation with a cost

similar to the method of order $O(h^2)$ of II.1 is described in II.6.3. Numerical results for a set of four test problems frequently found in the literature are obtained with a FORTRAN computer implementation (Section II.6.6). An operation count and comparisons with the successive extrapolations method are offered at the end of Chapter II.

Finally, Chapter III is dedicated to the detailed discussion of a computer implementation for the iterated deferred corrections method. The automatic weight generator for numerical differentiation of III.1 is an indispensable tool in the "Universal Deferred Correction Generator" of 111.2. A theorem on asymptotic error estimation based on deferred corrections is proved in III.3 and it constitutes one of the important building blocks for the variable order, variable (uniform) step algorithm developed in 111.4. Numerical results and a computer program are also included.

It is in this final Chapter that we have collected some novelties not to be found in our former work on deferred corrections. In fact, the comparisons with Richardson extrapolations for these types of problems have not been performed before. It comes to no surprise that though the asymptotic behavior is very similar for both techniques, deferred corrections fare considerably better in terms of work for a given accuracy, giving the solution at more points as an additional bonus.

The aim of this Seminar was to evolve from the simple application we have described to more elaborate problems, such as: two-point boundary value problems for first order systems, elliptic boundary value problems on rectangular and general regions, parabolic mixed initial-boundary value problems, etc. Unfortunately, six lectures have not been quite enough to reach that goal and the second part of these notes will have to wait for

a better occasion. Nevertheless, we would like to refer the reader to the literature where some pointers are given on how to utilize the algorithms we develop here in more complex situations. A special mention should be made of the Deferred correction generator that can be used as presented here in many different problems. The same comment applies to the logical structure of the variable order, variable step method, whose flexibility and excellent results have no equal in the published literature on non-linear, second order, smooth, two-point boundary value problems.

Acknowledgements

My warmest thanks to Gene Golub who made the necessary arrangements for my spending a very fruitful and stimulating quarter at Stanford University, who dispensed his usual open-handed hospitality, and who was a key factor in my dedication to this work.

Thanks to all the people that made the writing of these notes possible: to the participants in the Seminar for their patience and interest; to Mike Malcolm, Paul Concus, my office mate, Les Jennings, and Sam Schecter for their encouragement and comments; and to Mary Bodley, for her tireless dedication to perfect typing and reproducing.

II. Two-point boundary value problems for nonlinear second order differential equations

II.1 The problem and its discretization

We consider in this chapter problem (1.1) under the additional conditions:

(2.1a) $f(x,y) \in C^{\infty}[[a,b] \times (-\infty,+\infty)]$,

(2.1b) $f_y(x,y) > -\pi^2/(b-a)^2$

It is well known (Lees (1964)) that in this case (1.1) has a unique solution $y^* \in C^{\infty}[a,b]$, which can be approximated by a three point finite difference method.

We call (1.1) the continuous problem. The finite difference approximation will constitute the discrete problem.

Let $h = \frac{b-a}{n}$ for a given natural number $n > 1$, and let $x_i = a + ih$, $i=0,1,\ldots,n$, define an uniform mesh on $[a,b]$. The discrete problem is obtained by replacing y'' in (1.1) by a second order symmetric difference at every interior mesh point:

(2.2a) $h^{-2}(-Y_{i-1} + 2Y_i - Y_{i+1}) + f(x_i,Y_i) = 0$, $i=1,\ldots,n-1$,

(2.2b) $Y_0 = \alpha$, $Y_n = \beta$.

For short, we can denote (1.1) by

(2.3) $F(y) = 0$,

and this is to be understood as a nonlinear equation in a certain function space. We won't make this any more precise here, since our emphasis is in quite a different direction, but nevertheless, we shall take advantage of the built-in power of synthesis that such a formulation has. In the

same spirit, (2.2) will be denoted by

(2.4) $F_h(Y) = 0$,

representing a nonlinear equation ("system of equations") in the Euclidean space E^{n-1} , the unknown being the vector $Y^T = (Y_1,\ldots,Y_{n-1})$. Naturally, the idea is that h will go to zero (or $n \to \infty$) and thus we really have an infinite family of these objects. Also we expect that, in some sense, the values $Y_j(h)$ will converge to the respective function values of the exact solution. In order to make these ideas more precise, we need to introduce some extra notation. For each function Z(x) defined in [a,b] and satisfying (1.1b) we define $\varphi_h[Z(x)] = [Z(x_1),\ldots,Z(x_{n-1})]^T$, The operator φ_h is sometimes referred to as an space discretization,

We shall say that the discrete solutions Y(h) converge discretely to the exact solution y*(x) if:

(2.5) $$\lim_{h\downarrow 0} \|Y(h) - \varphi_h y^*\|_{(h)} = 0 ,$$

where $\|\cdot\|_{(h)}$ is the maximum norm on $E^{\frac{(b-a)}{h}}$. In what follows we shall omit the subindex (h) from the norms.

As usual, this convergence depends on two properties of the discrete operator F_h : consistency and stability.

Definition 2.1. The operator F_h is consistent of order $p > 0$, if for the solution y(x) of (1.1) and $h \leq h_0$ it holds that:

(2.6) $\|F_h(\varphi_h y)\| = O(h^p)$.

Definition 2.2. The operator F_h is stable if for any pair of discrete functions U , V , and $h \leq h_0$ there exists a constant $c > 0$, independent of h , such that:

(2.7) $\|U - V\| \leq c \|F_h(U) - F_h(V)\|$.

Lemma 2.3. If F_h is stable then it is locally invertible around $\varphi_h y^*$, and the inverse mapping F_h^{-1} is uniformly Lipschitz continuous for all $h \leq h_0$.

Proof: Let us consider the open spheres $B_h \equiv B(\varphi_h y^*, p)$, where $p > 0$ is independent of h . For any $U, V \in B_h$ we have, because of the stability condition, that F_h is an one-to-one mapping (since otherwise the right hand side in the inequality (2.7) could be zero without the left hand side being zero!), and therefore is a bijection between B_h and its image $R_h \equiv F_h(B_h)$. Thus the inverse mapping F_h^{-1} exists in R_h . Let $X, Y \in R_h$, then we can write (2.7) as

$$\|F_h^{-1}(X) - F_h^{-1}(Y)\| \leq C \|X - Y\| .$$

□

With this result we can prove the discrete convergence of any consistent, stable discretization,

Theorem 2.4. Let us assume that the continuous problem $F(y) = 0$ has a unique solution y^* . Let F_h be a stable discretization on the spheres $B_h \equiv B(\varphi_h y^*, \rho)$, and be consistent of order p with F . Then there is an $\bar{h}_0 > 0$ such that:

(a) For any $h \leq \bar{h}_0$ there exists a unique solution $Y(h)$ for the discrete problem $F_h(Y) = 0$.

(b) The discrete solutions $Y(h)$ satisfy

(2.8) $\|Y(h) - \varphi_h y^*\| = O(h^p)$.

(i.e., they are convergent of order p).

Proof: Let R_h be, as in Lemma 2.3, the image of B_h by F_h , and let us call $Z_h^* \equiv F_h(\varphi_h y^*)$. Obviously $Z_h^* \in R_h$, and because of the

consistency $\|z_h^*\| = O(h^p)$. On the other hand, because of Lemma 2.3 we know that for $h \leq h_0$, the F_h are homeomorphisms between the spheres B_h and their images R_h. By Brouwer's Invariance of Domain Theorem (Aleksandrov [1956]) we know then that F_h maps the interior of B_h onto the interior of R_h, and the boundary onto the boundary. Let V be any vector on the boundary of B_h. Because of the stability condition we know that

$$(2.9) \qquad \frac{\rho}{c} \leq \|F_h(V) - z_h^*\| ,$$

and since $F_h(V)$ will run over the whole boundary of R_h while V runs over the boundary of B_h, we can conclude that the sphere $B(z_h^*, \rho/c)$ is fully contained in R_h. Since $\|z_h^*\| \to 0$ for $h \to 0$, we can now choose $\bar{h}_0 \leq h_0$ such that for $h \leq \bar{h}_0$, $\|z_h^*\| < \rho/c$, which in turn will imply that $\underset{\sim}{0} \in B(z_h^*, \rho/c) \subset R_h$. But R_h was the image of B_h by F_h, and therefore the last statement implies that for $h \leq \bar{h}_0$ there exists a unique $Y(h) \in B_h$ such that $F_h(Y(h)) = 0$. (All these statements are represented in Fig. I.) The discrete convergence of order p follows also from the stability. In fact

$$\|Y(h) - \varphi_h y^*\| \leq c \|z_h^*\| = O(h^p) . \qquad \square$$

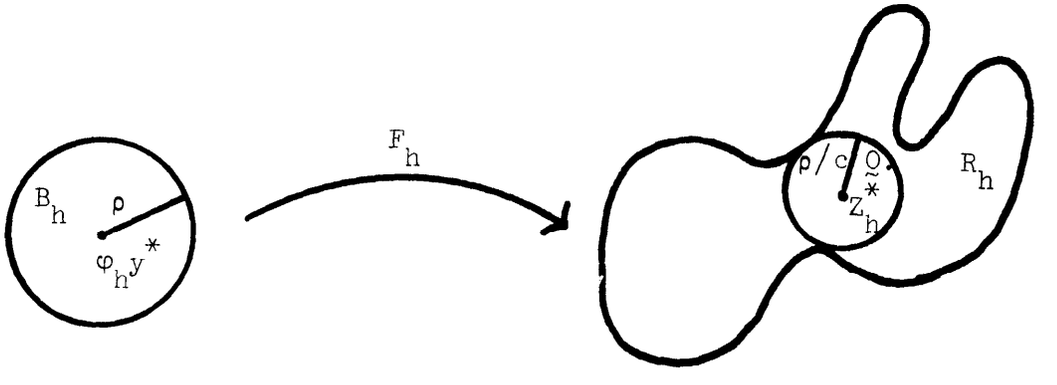


Fig. I

Remark. Observe that there is very little in the statements and results of this Section that is necessarily tied up to the two-point boundary value problem, and therefore they have more general applications.

II.2 Consistency, stability, and convergence

By using $\varphi_h y^*$ in (2.2) we obtain what is usually known as the local truncation error, This is an expression that shows how much our discrete operator fails to represent the continuous operator (for which we have $F(y^*) = 0$):

$$(2.10) \qquad \tau_h(x_i) \equiv [F_h(\varphi_h y^*)]_i \equiv h^{-2}(-y^*(x_i\ 1) + 2y^*(x_i) - y^*(x_{i+1})) + f(x_i, y^*(x_i)) .$$

We can obtain a more interesting expression for $\tau_h(x)$ by expanding in Taylor's formula around x , which we can do thanks to the smoothness assumptions. By using the fact that $f(x,y^*(x)) \equiv y^{*\prime\prime}(x)$ we get:

$$\tau_h(x) = -\sum_{k=1}^{K} \frac{2}{(2k+2)!} y^{*(2k+2)}(x) h^{2k} + O(h^{2K+2}) ,$$

This expansion then shows that the discrete method is consistent of order $p = 2$.

We shall now prove that the discrete method (2.1) is stable for h sufficiently small, which through Theorem 2.4 will give us the existence of unique discrete solutions of the nonlinear system of equations (2.1), and their discrete convergence of order h^2 to $y^*(x)$. The proof of the L_∞ stability is basically due to Lees [1964]. We need several definitions and Lemmas. The technique is a simple instance of the use of L_2 estimates often found in partial difference equations.

For every h we define the inner product of mesh functions by

$$(2.11) \qquad (V,U) \equiv h \sum_{i=1}^{n-1} V_i U_i \ .$$

This inner product induces a norm over the mesh functions that we denote by

$$(2.12) \qquad \|V\|_0 \equiv (V,V)^{\frac{1}{2}} \ .$$

By the usual relationships between the standard L_∞ and L_2 norms ($\|x\| \leq \|x\|_2 \leq \sqrt{n}\,\|x\|$) we have that

$$(2.13) \qquad h^{\frac{1}{2}}\|V\| \leq \|V\|_0 \leq (b-a)^{\frac{1}{2}}\|V\| \ ,$$

since $$\|V\|_0 = h^{\frac{1}{2}}\|V\|_2 = \left(\frac{b-a}{n}\right)^{\frac{1}{2}}\|V\|_2 \ .$$

Let us consider the difference operators Δ_+ and Δ_-:

$$(2.14) \qquad \begin{aligned} \Delta_+ u(x) &= h^{-1}(u(x+h) - u(x)) \ , \\ \Delta_- u(x) &= h^{-1}(u(x) - u(x-h)) \ . \end{aligned}$$

It is clear that $\delta^2 u(x) = h^{-2}[-u(x-h) + 2u(x) - u(x+h)]$ satisfies:

$$(2.15) \qquad -\delta^2 u(x) = \Delta_+\Delta_- u(x) \ .$$

We need still another norm in our space, that will involve the difference operator Δ_-:

$$(2.16) \qquad \|V\|_\Delta = (\Delta_- V, \Delta_- V)^{\frac{1}{2}} = \left(h \sum_{i=1}^{n} |\Delta_- V_i|^2 \right)^{\frac{1}{2}} \ .$$

We quote without proofs the following results of [29] .

a) $2\,\|V\| \leq \|V\|_\Delta$;

b) $(U, \delta^2 V) = (\Delta_- U, \Delta_- V)$.

This implies in particular that

c) $(\delta^2 U, U) = \|U\|_\Delta^2$.

If δ^2 is considered as a linear operator over the mesh functions, then its matrix representation has the familiar tridiagonal form $h^{-2}(-1, 2, -1)$.

This matrix has eigenvalues $\lambda_j = \frac{4}{h^2} \sin^2 \frac{j\pi h}{2(b-a)}$, $j=1,\ldots,n-1$, and we have also:

d) $\lambda_1 \|u\|_0^2 \leq \|u\|_\Delta^2$

Theorem 2.5, Let η 3 inf f_Y . The discretization (2.2) is stable for $h \leq h_0$ satisfying:

$$\frac{-\pi^2}{(b-a)^2}\left[1 - \frac{\pi^2 h_0^2}{24(b-a)^2}\right]^2 < \eta \ .$$

Proof: Letus consider two mesh functions U, V, and let q_1, q2 be defined as:

$$F_h(U) \equiv \delta^2 U + f(x,U) = q_1 \ ,$$

$$F_h(V) \equiv \delta^2 V + f(x,V) = q_2 \ .$$

Putting $W = U - V$, and $q = q_1 - q_2$ we obtain by the mean value theorem in integral form:

$$\delta^2 W + \int_0^1 f_y(x, \xi U + (1-\xi)V)\, d\xi \cdot W = q \ .$$

Calling the integral term P we observe that $P \geq \eta$. By taking inner products (see (2.11)), we get

$$(W, \delta^2 W) = (W,q) - (W, PW) \ ,$$

and therefore by c) and Schwartz's inequality:

$$\|W\|_\Delta^2 \leq \|W\|_0 \|q\|_0 - \eta \|W\|_0^2 \ .$$

By d) we obtain $\|W\|_\Delta^2 \leq \lambda_1^{-\frac{1}{2}} \|W\|_\Delta \|q\|_0 - \frac{\eta}{\lambda_1} \|W\|_\Delta^2$ or,

$$(2.17) \qquad \left(1 + \frac{\eta}{\lambda_1}\right) \|W\|_\Delta \leq \lambda_1^{-\frac{1}{2}} \|q\|_0 \ .$$

It is easy to verify, by expanding sin x in Taylor series and truncating at the first and second term respectively, that

$$\frac{\mu^2}{(b-a)^2} \geq \lambda_1 \geq \frac{\pi^2}{(b-a)^2}\left[1 - \frac{(\pi h_0/(b-a))^2}{24}\right]^2$$

and since (2.17) can be written as:

$$\|W\|_\Delta \leq \frac{\lambda_1^{\frac{1}{2}}}{\lambda_1 + \eta} \|q\|_0$$

we have from the hypothesis that

$$(2.18) \qquad \|W\|_\Delta \leq \frac{\pi(b-a)^{-1}}{\frac{\pi^2}{(b-a)^2}\left[1 - \frac{(\pi h_0)^2}{(b-a)^2 24}\right]^2 + \eta} \|q\|_0 = K \|q\|_0$$

where the denominator is greater than zero. We still haven't got the inequalities in the infinity norm ($\|\cdot\|$). We recall (2.13) and a) in order to transform (2.18) into:

$$\|U-V\| \leq \frac{K}{2} (b-a)^{\frac{1}{2}} \|F_h(U) - F_h(V)\| \ .$$

Theorems 2.4, 2.5, and equation (2.10) prove that the discretization (2.2) is convergent of order 2, i.e.

$$(2.19) \qquad \|Y(h) - \varphi_h y^*\| = O(h^2) \ .$$

In the next section we shall develop a more detailed expression for the global discretization error (2.19).

II.3 An asymptotic expansion for the global discretization error

Under the assumptions of Sections 1 and 2 it is clear that the variational equation (linear!) associated with (1.1)

(2.20a) $-e'' + f_y(x,y)e = g(x)$

(2.20b) $e(a) = e(b) = 0$

has an unique solution $e(x) \in C^\infty[a,b]$ for each given C^∞ functions $y(x), g(x)$.

If we use for (2.20) the same discretization (2.2) as we used for (1.1), then an expression similar to (2.10) holds. In fact, it will be convenient to-use the notation $F'(y)e = g$ for (2.20), and $F_h'(\varphi_h y)E = \varphi_h g$, for its discrete analogue.

The prime here denotes derivative (in the sense of' Fréchet; Jacobian in the finite dimensional case). Therefore we have, at the solution of (1.1)

$$(2.21)\quad \tau_h'(x) = F_h'(\varphi_h y^*)\varphi_h e^*(x) = \varphi_h\{g - \sum_{k=1}^{K} a_k e^{*(2k+2)}(x)h^{2k}\} + O(h^{2K+2}) ,$$

where $e^*(x)$ is the corresponding solution ok' (2.20), and $a_k = \frac{2}{(2k+2)!}$. As it turns out, higher order derivatives of the mappings $\varphi_h F$, F_h coincide, having the form

$$(2.22)\quad \varphi_h F^{(j)} e^j \equiv F_h^{(j)} \varphi_h e^j \equiv \varphi_h \frac{\partial^j f}{\partial y^j} e^j \quad .$$

Theorem 2.6. Let F, F_h be as above. Then for $h \leq h_0$ the global discretization error has an asymptotic expansion in even powers of h :

$$(2.23)\quad Y(h) - \varphi_h y^* = e(h) = \varphi_h \sum_{k=1}^{K} e_k(x)h^{2k} + O(h^{2K+2}) .$$

The functions $e_k(x)$ are independent of h and satisfy the linear two point boundary value problems:

$$(2.24) \qquad F'(y^*)e_k \equiv -e_k'' + f_y(x,y^*(x))e_k = b_k\,,\ e_k(a) = e_k(b) = 0\ .$$

The functions b_k will be constructed in the proof.

Proof: We can rewrite (2.23) in the form,

$$S(h) = e(h) - \varphi_h \sum_{k=1}^{K} e_k(x)h^{2k} = O(h^{2K+2})\ .$$

Let us call for short $\mu(h) \equiv \sum_{k=1}^{K} e_k h^{2k}$, and $I = -F_h(\varphi_h(y^* + \mu(h)))$.

If we are able to prove that for appropriate choices of b_k $I = O(h^{2K+2})$, then by using the stability condition the expansion (2.23) will follow.

In fact we have, since $F_h(Y(h)) = 0$, that

$$(2.25) \qquad \|I\| = \|F_h(Y(h)) - F_h(\varphi_h(y^* + \mu(h)))\| \geq$$

$$\frac{1}{c}\,\|Y(h) - \varphi_h(y^*) - \mu(h)\| \,.\, \frac{1}{c}\,\|S(h)\|\ .$$

... that $\mu(h) = O(h^2)$, let us expand I in Taylor's series around $\varphi_h y^*$.

$$I \,.\, -\{F_h(\varphi_h y^*) \,.\, F_h'(\varphi_h y^*)\varphi_h\mu(h) + \sum_{j=2}^{K} \frac{F_h^{(j)}(\varphi_h y^*)}{j!}\,[\varphi_h\mu(h)]^j\} + O(h^{2K+2})\ .$$

Using the expansions (2.10), (2.21) and (2.22), we obtain:

$$I = \varphi_h\Big\{\sum_{k=1}^{K} a_k y^{*(2k+2)}h^{2k} + F'(y^*)\mu(h) + \sum_{j=1}^{K} a_j\mu^{(2k+2)}(h)\cdot h^{2j} +$$

$$+ \sum_{j=2}^{K} \frac{1}{j!}\,f_Y^{(j)}(x,y^*)\mu(h)^j\Big\} + O(h^{2K+2})\ .$$

Now we observe that:

a) $$F'(y^*)\mu(h) = \sum_{k=1}^{K} F'(y^*)e_k h^{2k} = \sum_{k=1}^{K} b_k h^{2k} ;$$

b) $$\sum_{j=1}^{K} a_j h^{(2k+2)} h^{2j} = \sum_{j=1}^{K} a_j \left(\sum_{\nu=1}^{K} e_\nu^{(2k+2)} h^{2\nu} \right) h^{2j}$$

$$= \sum_{k=1}^{K} \left(\sum_{\nu=1}^{k-1} a_{k-\nu} e_\nu^{(2(k-\nu)+2)} \right) h^{2k} + O(h^{2K+2}) ;$$

c) $$\mu(h)^j = \sum_{\xi=j}^{K} Q_{j,\xi}(e_1, \ldots, e_{\xi-j+1}) h^{2\xi} + O(h^{2K+2})$$

where the $Q_{j,\xi}$ are polynomials j-homogeneous on their variables,

Replacing these three expressions in I we get:

$$I = \varphi_h \left\{ \sum_{k=1}^{K} \left[a_k y^{*(2k+2)} + b_k + \sum_{\nu=1}^{k-1} a_{k-\nu} e_\nu^{(2(k-\nu)+2)} \right.\right.$$

$$\left.\left. + \sum_{j=1}^{k} \frac{1}{j!} f_y^{(j)}(x,y^*) Q_{j,k}(e_1, \ldots, e_{k-j+1}) \right] h^{2k} \right\} + O(h^{2K+2}) ,$$

where we assume $Q_{1,k} \equiv 0$.

Since we want I to be $O(h^{2K+2})$, that means we would like to choose b_k so that { } vanishes. Thus $b_1(x) = -\frac{1}{12} y^{*(4)}(x)$, with which we can determine $e_1(x)$ by solving (2.24);

$$b_2(x) = -\left[\frac{1}{360} y^{*(6)}(x) + \frac{1}{12} e_1^{(4)}(x) + \frac{1}{2} f_{yy}(x,y^*) e_1^2(x) \right] ,$$

which allows us to determine $e_2(x)$, and so on. We observe that in general, the determination of b_k involves derivatives of the solution y^* , and earlier error functions e_ν , $\nu=1,\ldots,k-1$. Therefore the b_k can be determined recursively. This proves the Theorem. □

II.4 Successive extrapolations

Expansion (2.23) is the basis for the well known method of successive extrapolations ("to h=0"), a fairly simple procedure used to increase the order of' the discretization. In other contexts, this procedure is associated with the names of Richardson [1910], Romberg [1955], Gragg[1963], Bulirsch and Stöer [1964], Stetter [1965], and Pereyra [1967a]. See Joyce [1971] for a detailed survey and a more complete set of references, and Widlund [1971] for a survey of recent developments.

We shall describe briefly the application of successive extrapolations to our present problem in order to emphasize certain aspects and establish a basis for comparison with other high order methods.

Let $Y(h_o)$, $Y\left(\frac{h_o}{2}\right)$, . . . , $Y\left(\frac{h_o}{2^i}\right)$, be the solutions of (2.2) for the indicated step sizes, that correspond to systems of dimensionality $n_o = \frac{b-a}{h_o} - 1$, . . . , $n_i = 2 \times n_{i-1} + 1$ (see Fig. 2 for an example).

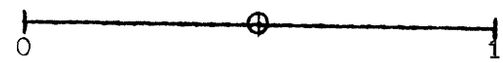
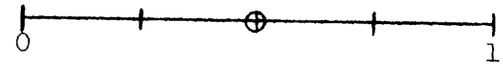
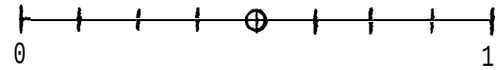
Function	n	Grid
Y(1/2)	1	
Y(1/4)	3	
Y(1/8)	7	

Fig. 2

It is clear that only the points corresponding to the coarsest mesh (ho) are common to all meshes. It is at those common points where we shall be able to improve the order of our solution. Let us then call Y_i^o the n_o vectors obtained from $Y(h_i)$ by extracting the components

$$Y_{t2^i}(h_i), \quad t=1,\ldots,n_0,$$

of $Y(h_i)$. With these initial values Y_i^0, we can form the (vector) Romberg triangle

$$(2.26) \qquad {}_iY_i^j = \frac{4^j Y_i^{j-1} - Y_{i-1}^{j-1}}{4^j - 1}, \quad i = 1,\ldots ; \ j=1,\ldots,i .$$

From (2.23) we can easily derive asymptotic expansions for $Y_i^j - cp_{h_o} y^*$:

$$(2.27) \qquad Y_i^j - \varphi_{h_o} y^* = \varphi_{h_o} \sum_{k=j+1}^{K} e_{kj}(x) h_i^{2k} + O(h^{2K+2}) .$$

Also, if we disregard terms of order greater than $h_i^{(2j+2)}$, then we can obtain an asymptotic error estimate for the global discretization error.

Lemma 2.7

$$(2.28) \qquad Y_i^j - \varphi_{h_o} y^* \approx (Y_i^j - Y_{i-1}^j)/(1-4^{j+1})$$

where $\approx$ stands for asymptotically equal.

Proof: Write (2.27) for Y_i^j and Y_{i-1}^j, and subtract, ignoring terms of order greater than $h^{(2j+2)}$:

$$Y_i^j - \varphi_{h_o} y^* \approx \varphi_{h_o} e_{j+1,j}(x) h_i^{(2j+2)} \approx \frac{Y_i^j - Y_{i-1}^j}{1 - 4^{j+1}}$$

□

II.5 Some comments on implementation

We have proved the existence of discrete solutions $Y(h)$ in a nonconstructive way. The most frequently used procedure for actually solving equations (2.2) is Newton's method. In cases in which f_Y is hard to compute some alternate procedure might be preferable. We won't go into the details of the implementation of Newton's method in this case since this

is fairly straightforward and it has been extensively discussed in the literature (cf. Henrici[1962], Keller [1968]). Let us only remark that system (2.2) is tridiagonal, which makes the solution of the linear systems that appear at each Newton step very simple. If there is no other information, a linear interpolation between the boundary values can provide a reasonable starting vector.

In constructing the successive extrapolates one can follow several paths. One of the most reasonable seems to be the following:

i) Compute $Y(h_o)$.

ii) Use $Y(h_o)$ and interpolation in order to have a good initial approximation for $Y(h_1)$.

iii) Use Y_o^o , Y_1^o in order to estimate the error in Y_1^o . If you are satisfied, quit. If not:

iv) Combine Y_0^o , Y_1^o in order to get Y_1^1 .

v) Obtain $Y(h_2)$ and construct a new row of the triangle, etc. . . .

Observe that for $Y(h_o)$ we shall use as a starting vector something probably pretty inaccurate, but the dimensionality of this problem will be the smallest. For any other $Y(h_i)$ we shall use in the Newton iteration the fairly accurate initial values provided by $Y(h_{i-1})$, using interpolation to fill into the new abscissas. This is a very important point, since it will tend to decrease considerably the number of Newton iterations necessary to carry the residuals below a level compatible with the discretization error.

We have always to remember that the dimensionaiity of the problem is multiplied by 2 every time we compute a new row. A source

of' criticism for this method has been the fact that one gets the most accurate results Y_j^j only on the coarsest mesh, wasting all the precious computation performed in the finer meshes. In a recent paper of Lindberg [1972] the author implements and justifies an idea of Dahlquist for producing accurate results on the finest mesh through a recursive interpolation procedure. This is done for initial value problems but it is clear that a similar principle will hold for our present problem (though it has not been done as far as I know; it would be worthwhile to investigate this matter further, clarifying Lindberg's statements).

II.6 Deferred corrections

II.6.1 Introduction

As early as 1947, Leslie Fox advocated a technique called "difference corrections". Through the years he and his collaborators have applied this technique to a variety of problems in differential and integral equations (see Pereyra [1967c] for a detailed bibliography and historical account). In Fox [1962], a wealth of information on the state of the art in the English School can be f'ound. It is there where we find the term "deferred corrections" used interchangeably with that of difference corrections. The reasons for this switch in nomenclature are not apparent, except perhaps for the feeling that the technique was in some way connected with the "deferred approach to the limit" that we were discussing in the earlier Sections, and also because the name reflected the fact that a posteriori corrections were performed.

We have preferred to adopt the latest name in our work on this technique since our approach is not tied up (at least in appearance) to

expansions in terms of differences, as it was in the earlier developments.

We base our formulation of the method on the asymptotic expansion for the local truncation error:

$$(2.29)\qquad \tau_h(x_i) = -\sum_{k=1}^{K} a_k y^{*(2k+2)}(x_i)h^{2k} + O(h^{2K+2}) ,$$

which, as we have already observed in Section II.2, only needs smoothness of the exact solution $y^*(x)$ and the application of Taylor's formula for its derivation,

For any smooth function $y^*(x)$ we can approximate linear combinations of its derivatives with any order of accuracy in h at any grid point by using sufficient ordinates in a neighborhood. This is again a consequence of a wise application of Taylor's expansions and numerical differentiation techniques. Thus, there exist weights w_s such that

$$(2.30)\qquad T_\ell(x_i) \equiv -\sum_{k=1}^{\ell} a_k y^{*(2k+2)}(x_i)h^{2k+2} = \sum_{s=1}^{2\ell+2+q} w_s y^*(x_i+\alpha_s h)$$

$$+ O(h^q) \equiv S_\ell(y^*(x_i)) + O(h^q) , \ \alpha_s \text{ integers.}$$

We shall show later how to obtain w_s in an efficient and sufficiently accurate way. Observe that we have multiplied $\tau_h(x)$ by h^2. In this fashion S_ℓ becomes a bounded operator (for $h \downarrow 0$) and most of the dangers of numerical differentiation formulas are avoided. In fact, because of the linear relations between differences and function values, some choices of $S_\ell(y^*(x_i))$ will coincide with Fox's formulation, though we feel that this more general presentation, coupled with efficient weight generators is better adapted for use on a digital computer. In fact,

Fox's difference correction procedure was mostly advocated for desk calculator computation, where a table of' differences manipulated by an able person was a real asset, The main contributions of the author of these notes, starting with a Stanford Report (Pereyra [1965]), have been to put on a sound theoretical basis the asymptotic behavior of a very general procedure modeled on Fox's difference corrections, and what is even more relevant, he has produced tools and complete implementations of this technique in a variety of applications. However, so many years and developments later (with some minor changes) the words in Fox's[1963] very interesting expository paper are still very much actual: "This idea (difference correction) does not seem to have penetrated deeply into the literature of automatic computation . . . . Certainly we have to do some differencing, involving extra programming, extra space, and some difficulties in automatic inspection of differences, but machines are getting larger and programming easier (or so everybody tells me), and if we are concerned with accuracy, as we certainly should be, I should have thought that something like this was essential."

Probably one of the main reasons for this neglect in recent times has been the widespread interest in other high order methods (splines, finite elements). Unfortunately, the theoretical developments in these areas have very much surpassed (and overshadowed) the practical, efficient implementation of the methods. Thus, we find ourselves in the sad situation of having a highly promising, very general, theoretically well supported technique, that is begging for an at least equal treatment in its practical aspects, while on the other hand, for some applications at least, it is fairly clear that the results obtained with our more traditional finite

difference techniques will be hard to beat. (Compare the numerical results for similar problems in Ciarlet, et al [1967,1968], Perrin, et al [1969], and Herbold and Varga [1972], with those in Pereyra [1967c,1968, 1970] and this report.)

I wouldn't be surprised if it finally turns out that a successful implementation of high order spline methods comes about via a deferred correction type of approach, bypassing in some way the very expensive steps of high order quadrature formulae and complicated systems arising from the present approaches. See Fyfe [1969] for a first timid step in that direction.

II.6.2 Algorithms

There are many ways of producing deferred corrections. Fox's way consisted essentially of representing y" as a series of differences. In the first step, common to all procedures, one would compute using only the first term of the expansion (in this case the basic method (2.2)), and then use these $O(h^2)$ values in the difference expansion, and recompute in order to obtain a more accurate solution. The process was thought as iterative, providing in infinitely many steps the exact solution. This was never done in practice; in fact it is hard to find any published numerical example in which more than two corrections were performed, carrying perhaps three or four terms in the difference expansion. Naturally, the reason for this was that on a desk calculator any prolonged computation was a big undertaking.

Let $Y^{(o)}$ be the O(h2) solution to (2.2), and let S_1 be, as in (2.30), an $O(h^6)$ approximation to $T_1 \equiv -a_1 y^{*(4)} h^4$, the first term

in the local truncation error (multiplied by h^2). Observe that since there is already a factor h^4 in T_1, we only are requiring an $0(h^2)$ approximation to $y^{*(4)}$ at the grid points. If we have $y^*(x)$ available then, as we said before, there is no problem in obtaining the weights w_s for S_1. But all what we have is $Y^{(o)}$. In principle it cannot be expected that from an $O(h^2)$ discrete approximation to a function one can obtain an $O(h^2)$ approximation to a derivative. It is here where we make use of the expansion (2.23) for the global discretization error. In fact we have that because of linearity and (2.30):

$$S_1(Y^{(o)}) - S_1(\varphi_h y^*) = S_1(\varphi_h e_1)h^2 + S_1(\varphi_h e_2)h^4 + O(h^6) .$$

Observe that we have used the fact that $S_1 = O(1)$. But $S_1(\varphi_h y^*) = T_1 + O(h^6)$, $S_1(\varphi_h e_k) = -a_1 e_k^{(4)} h^4 + O(h^6)$, $k=1,2$. Therefore,

$$S_1(Y^{(o)}) = T_1 + O(h^6) ,$$

and we can use $Y^{(o)}$ instead of $\varphi_h y^*$ and still obtain the same asymptotic behavior. With $S_1(Y^{(o)})$ computed at every grid point we solve for a corrected value $Y^{(1)}$

$$F_h(Y) = h^{-2} S_1(Y^{(o)}) . \tag{2.31}$$

The local truncation error for this new discretization is $O(h^4)$ and therefore, since we are still talking about the same basic operator F_h, the stability condition proves that there exists a uniquesolution $Y^{(1)}$ to this problem and that

$$\| Y^{(1)} - \varphi_h y^* \| = O(h^4) . \tag{2.52}$$

Provided we can obtain an asymptotic expansion for $Y^{(1)} - \varphi_h y^*$

this procedure can be repeated, and each time two more orders in h will be gained. In general, the iterated deferred correction procedure can be described in the following way:

i) Let $Y^{(k)}$ be an $O(h^{2k+2})$ discrete solution.

ii) Compute $h^{-2}S_{k+1}(Y^{(k)})$, an h^{2k+2} approximation, to the first $(k+1)$ terms in the local truncation error expansion.

iii) Solve $F_h(Y) = h^{-2}S_{k+1}(Y^{(k)})$ for $Y^{(k+1)}$.

For boundary value problems there are some theoretical difficulties in obtaining the successive expansions needed to justify the method, The difficulty comes from the fact that different differentiation formulas must be used at different points of the mesh. In fact, in the first step we can use five point symmetric formulas in order to approximate $y^{*(4)}$ to order h^2 at the mesh points $x2,\ldots,x_{n-2}$, but we shall need six point unsymmetric formulas at the points x_1, x_n 1 . For the symmetric formulas we have asymptotic expansions in even powers of h :

$$(2.33)\quad S_1(y^*(x_i)) = T_1(x_i) + \sum_{\nu=5}^{K} t_\nu(x_i)h^{2\nu} + O(h^{2K+2}),\ i=2,\ldots,n-2,$$

while for the unsymmetric formulas we shall have (different) expansions with all powers of h . With a small manipulation it can be shown that

$$F_h(\varphi_h y^*) - h^{-2}S_1(Y^{(o)}) = \varphi_h \sum_{k=4}^{2K} r_k(x)h^k + O(h^{2K+1}),$$

but $r_k(x)$ will in general be discontinuous, because of the change of differentiation formulae. Therefore, Theorem 2.6 cannot be applied in order to guarantee the existence of expansions for $Y^{(1)} - \varphi_h y^*$, which

in turn would be necessary (in our approach) for proving the accuracy of the differentiation formulas in successive steps.

One way of deferring this until after the second correction is the following: Since $y^{*\prime\prime} = f(x,y^*)$ then we can replace all higher derivatives of y^* by total derivatives of' f two orders lower. Thus in our first correction we need to approximate $\frac{d^2}{dx^2} f(x,y^*(x))$ only to order h^2, and by using grid values of $f(x_i,Y_i)$ we can achieve this with a symmetric three point formula over the whole range. Naturally, the same problem we discussed above will appear after the second correction. We shall see later that by using a basic method of order h^4, we can rigourously obtain an h^8 order method applicable to the problem of this Chapter.

We can also rigourously perform deferred corrections (any number) for boundary value problems of the form (1.1a) with periodic f, i.e.: $f(x + b - a, y) = f(x)$, and periodic boundary conditions:

$$y(a) = y(b) , y'(a) = y'(b) .$$

In this case, we can use the same differentiation formula over the whole range since there are really no boundaries in this problem, and we can extend our solutions by periodicity.

Now the fact that we cannot obtain with the present methods the theoretical asymptotic behavior of the iterated deferred corrections for problems (1.1) does not mean that the technique is useless in this general case. Far from it, we shall show numerical results that should justify a more careful study in order to determine precisely what is that asymptotic behavior. We would like to stress the point that the asymptotic expansions for the successive global errors $Y^{(k)} - \varphi_h y^*$ are used only in the theoretical

justification of the method, but at no time are they needed in its practical implementation as in the case of successive Richardson extrapolations.

More general equations of the form

(2.54) $y'' = f(x,y,y')$

can and have been treated. We feel at the present time that those problems will be much more easily dealt with using a general procedure for systems of the form

$$\underset{\sim}{y}' = \underset{\sim}{f}(x,\underset{\sim}{y})$$

(2.35)

$$A\underset{\sim}{y}(a) + B\,\underset{\sim}{y}(b) = \underset{\sim}{\alpha}\ ,$$

which is now in development. We expect that our method will compete favourably with the multiple shooting techniques that have become fashionable in recent times. In Keller [1969, 1972] the relevant theory for an $O(h^2)$ discrete approximation to (2.35) is developed and asymptotic expansions are derived. Keller uses then this fact to justify a successive Richardson extrapolation procedure. See also Kreiss [1971].

II.6.3 An $O(h^8)$ method for the price of an $O(h^2)$ method

In this Section we consider problem (1.1) again, but we shall use the more accurate $O(h^4)$ discretization

(2.36) $$h^{-2}[-Y_{i-1} + 2Y_i - y_{i+1}] + \frac{1}{12}[f_{i-1} + 10\,f_i + f_{i+1}] = 0\ ,\quad i=1,\ldots,n-1$$

where $f_i \equiv f(x_i,Y_i)$. We symbolize (2.36) by $G_h(Y)$. By recalling that $f(x, y^*(x)) \equiv y^{*\prime\prime}(x)$ it is then easy to derive via Taylor expansions that the local truncation error is in this case:

$$G_h(\varphi_h y^*) = \varphi_h \sum_{k=2}^{K} a_k f^{(2k)}(x,\ y^*(x)) \frac{h^{2k}}{(2k)!} + O(h^{2K+2}) ,$$

where $a_k = \frac{1}{(k+1)(2k+1)} - \frac{1}{6}$.

This method can also be proven to be stable as was the case for the simpler method (2.2). (See [,].) Thus we can produce an algorithm similar to the one described in 11.6.2 but which now should gain 4 orders per correction. We shall make explicit that algorithm in the next Chapter, while presently we develop a correction method of order h^8 which is specially effective and economical. Paraphrasing terms which are very fashionable these days we could say that the method to be described is of a high computational "simplicity". The main idea is that one correction with the same asymptotic properties as in the usual procedure, can be obtained by solving the variational equation associated with the problem, with an appropriate right hand side. If Newton's method is being used to solve the nonlinear equations resulting from the basic discretization then the correction will look just like one extra Newton step. If we observe that the $O(h^4)$ method (2.36) is essentially not more complex than the $O(h^2)$ method (2.2) then the reason for the title of this Section becomes clear.

The linearized equations that obtains at each Newton step ν are the following:

$$(2.37)\quad \left[\frac{h^2}{12} f_y(x_{i-1},Y^\nu_{i-1}) - 1\right] E_{i-1} + \left[\frac{5h^2}{6} f_y(x_i,Y^\nu_i) + 2\right] E_i + \left[\frac{h^2}{12} f_y(x_{i+1},Y^\nu_{i+1}) - 1\right] E_{i+1} = r^\nu_i$$

where

$$(2.38)\quad r^\nu_i = -\{(-Y^\nu_{i-1} + 2Y^\nu_i - Y^\nu_{i+1}) + \frac{h^2}{12}(f^\nu_{i-1} + 10f^\nu_i + f^\nu_{i+1})\} .$$

For short, we can call the left hand side of (2.37): $h^2 G_h'(Y^\nu)E$. Once $\{E_i\}$ is obtained, then the new iterate results:

(2.39) $\quad Y_i^{\nu+1} = Y_i^\nu + E_i$.

Because of the stability, it is enough to reduce the residuals r_i to a level compatible with the global discretization error in the final corrected solution. In fact,

$$r^\nu = h^2 G_h(Y^\nu) \ , \ G_h(Y(h)) = 0 \ ,$$

and therefore we have that

$$\| Y^\nu - Y(h) \| \leq c \ \| G_h(Y^\nu) \| = ch^{-2} \ \| \underset{\sim}{r}^\nu \| \ .$$

Thus,

$$\| Y^\nu - \varphi_h y^* \| \leq \| Y^\nu - Y(h) \| + \| Y(h) - \varphi_h y^* \| \leq ch^{-2} \ \| \underset{\sim}{r}^\nu \| + Ch^4 \ ,$$

and a reasonable stopping criteria for Newton's method is then:

(2.40) $\quad \| \underset{\sim}{r}^\nu \| \leq c_1 h^{10}$,

where c_1 is usually chosen to be a small constant unless some more precise information about c and C is available. Let $Y^{(o)}$ be the computed $O(h^4)$ solution. If we now define

$$(2.41) \quad T(x_i) = - \frac{1}{10} \frac{d^4}{dx^4} f(x_i, y^*(x_i)) \frac{h^4}{4!} - \frac{11}{84} \frac{d^6}{dx^6} f(x_i, y^*(x_i)) \frac{h^6}{6!} \ ,$$

and

(2.42) $\quad S(f(x, Y^{(o)})) = T_1(x) + O(h^8)$.

Then by solving

(2.43) $\quad G_h'(Y^{(o)})E = S(f^{(o)})$

and putting

(2.44) $\quad Y^{(1)} = Y^{(o)} - E$,

we shall have an $O(h^8)$ approximation.

Proof: By an argument similar to Theorem 2.6, we know that the smooth function $e_1(x)$ satisfying

$$(2.45)\quad G'(y^*)e_1 = h^{-4}T$$

is such that $e \equiv Y^{(o)} - \varphi_h y^* = \varphi_h e_1 h^4 + O(h^8)$.

But also,

$$G_h'(\varphi_h y^*)\bar{e}_1 = \varphi_h T + O(h^8),$$

where $e_1 = \varphi_h e_1 h^4$

Therefore, using (2.42) we get

$$G_h'(\varphi_h y^*)\bar{e}_1 - G_h'(Y^{(o)})E = (G_h'(\varphi_h y^*)\bar{e}_1 - G_h'(Y^{(o)})\bar{e}_1) + G_h'(Y^{(o)})(\bar{e}_1 - E)$$

$$= -G_h''(\varphi_h y^*)e \cdot \bar{e}_1 + G_h'(Y^{(o)})(\bar{e}_1 - E) = O(h^8) .$$

But since the term $G_h'(\varphi_h y^*)e\ \bar{e}_1 = O(h^8)$ and $G_h'(Y^{(o)})$ is stable we obtain

$$\bar{e}_1 - E = O(h^8)$$

which in turn implies that

$$(Y^{(o)} - E) - \varphi_h y^* = O(h^8)$$

as we wanted to prove. □

II.6.4. Some numerical results

In this Section we present some test problems from the current literature in high order methods. Some limited comparisons are included. The limitations are generally due to the vagueness in which numerical results are often presented.,

Problem 1

$$-y'' + y^3 - \sin x \,(1 + \sin^2 x) = 0$$

$$y(0) = y(\pi) = 0$$

Exact solution:

$$y(x) = \sin x \,.$$

See Pereyra [1968].

Problem 2

$$-y'' + e^y = 0$$

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

Exact solution:

$$y(x) = -\ell n\, 2 + 2\ell n\, (c \cdot \sec(\tfrac{c}{2}(x - \tfrac{1}{2}))\}$$

The constant c satisfies: $c \sec \frac{c}{4} = \sqrt{2}$.

$$c = 1.33605569490610 8\ldots$$

See Perrin, Price and Varga [1969], H. B. Keller [1972].

Problem 3

$$-y'' + y + y^3 + e^{\sin 2\pi x}[4\pi^2(\cos^2 2\pi x - \sin 2\pi x) - e^{2\sin 2\pi x} - 1] = 0$$

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

Exact solution:

$$y(x) = e^{\sin 2\pi x} \,.$$

See Ciarlet, Schultz and Varga [1968].

Problem 4

$$-y'' + \tfrac{1}{2}(y + x + 1)^3 = 0$$

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

Exact solution:

$$y(x) = \frac{2}{2 - x} - x - 1 \ .$$

See Ciarlet, Schultz and Varga [1967] or Schultz [1973], p. 98.

The results of this Section were obtained with a FORTRAN IV implementation (WATFIV compiler) of the algorithm described in 11.6.3 running on the IBM 360/91 computer at the Stanford Linear Accelerator Center. Double precision (~ 16D) was used throughout. Newton's method was employed for solving the nonlinear equations, using as starting vectors in each case the linear interpolation of the boundary values.

The evaluation of the correction term was performed via the Universal Two-Point Boundary Value Problem Deferred Correction Generator which will be described in detail in the following Chapter.

In Table i we present results for Problem i , i=1,...,4 , "Error" stands for the maximum absolute error at the grid points between the exact and discrete solutions. Runs with maximum relative error gave similar results. $\underline{\text{Error}}_4$ corresponds to the basic h^4 approximation and $\underline{\text{Error}}_8$ to the corrected solution. (n+1) is the number of grid points, while (n-1) is the dimensionality of the systems solved. $\underline{m}$ is the computed order obtained by comparing the errors for two solutions for different step sizes. Thus,

$$(2.46) \quad m = \frac{\log[\text{error}(h)/\text{error}(h/2)]}{\log 2} \ .$$

$\underline{\text{Oper.}}$ stands for (number of operations)/1000 . A detailed operation count study will be performed in the next Chapter, and it is

from there where we obtain the results for this column. Function evaluations are not included in the operation count, but their number is connected in an obvious fashion with the column iter., which gives the number of Newton iterations necessary to reduce the maximum norm of the residuals in the solution of the basic problem below the level EPS . We adopted EPS = $10^{-4} \times h^8$ which gives the following stopping criterion for the Newton iteration:

$$(2.47) \quad \|G_h(Y^\nu)\|_\infty \leq \max\,(10^{-4} \times h^8,\ 5 \times 10^{-16}) ,$$

where the constant 5×10^{-16} is related to the IBM System 360 double precision.

We list in res. the norm of' the last residual. The notation a , b means $a \times 10^b$.

n	$error_4$	m	$error_8$	m	iter.	res.	oper.
8	2.90, -5	----	1.05, -7	----	7	1.22, -13	1.3
16	1.81, -6	4.00	1.12, -10	9.87	7	3.13, -13	2.8
32	1.13, -7	4.00	5.37, -13	7.70	7	7.92, -15	5.9
64	7.04, -9	4.00	1.97, -13	1.45	7	2.07, -15	12.0
128	4.40, -10	4.00	4.60, -14	2.10	8	1.97, -16	24.1
256	2.79, -11	3.98	3.92, -13	----	7	3.76, -16	48.5

Table 1

n	$error_4$	m	$error_8$	m	iter.	res.	oper.
8	3.86, -7	----	7.36, -10	----	4	1.47, -17	.8
16	2.42, -8	4.00	1.64, -12	8.81	4	2.15, -17	1.8
32	1.52, -9	3.99	4.08, -15	8.65	4	1.55, -17	3.8
64	9.48, -11	4.00	3.93, -16	3.38	4	2.27, -17	7.6
128	5.92, -12	4.00	7.91, -16	----	4	2.50, -17	15.4
256	3.74, -13	3.98	4.02, -15	----	4	2.51, -17	30.9

Table 2

n	$error_4$	m	$error_8$	m	iter.	res.	oper.
8	1.97, -2	----	9.02, -2	----	6	2.22, -16	1.2
16	1.06, -3	4.22	1.37, -4	9.36	6	2.78, -16	2.5
32	6.40, -5	4.05	7.06, -7	7.60	6	3.19, -16	5.2
64	3.97, -6	4.01	7.97, -10	9.79	6	3.76, -16	10.5
128	2.47, -7	4.01	2.49, -12	8.32	6	3.81, -16	21.2
256	1.55, -8	3.99	1.33, -13	4.23	6	3.87, -16	42.6

Table 3

n	$error_4$	m	$error_8$	m	iter.	res.	oper.
8	1.64, -5	----	4.65, -7	----	4	2.09, -13	.8
16	1.05, -6	3.97	2.20, -9	7.72	5	1.91, -17	2.2
32	6.60, -8	3.99	5.63, -12	8.61	5	1.80, -17	4.5
64	4.13, -9	4.00	1.39, -14	8.66	5	1.62, -17	9.1
128	2.58, -10	4.00	5.72, -16	4.60	5	2.12, -17	18.3
256	1.54, -11	4.07	9.27, -13	----	4	2.32, -16	30.9

Table 4

n	$error_4$	m	$error_8$	m	iter.	res.	oper.
10	1.19, -5	----	9.39, -9	----	7	7.86, -14	1.7
20	7.39, -7	4.01	1.74, -11	9.08	7	2.00, -14	3.6
40	4.61, -8	4.00	2.42, -13	6.17	7	5.19, -15	7.4
80	2.88, -9	4.00	2.06, -13	----	7	1.35, -15	15.0
160	1.81, -10	3.99	2.74, -13	----	7	4.98, -16	30.2

Table 5 (Problem 1)

n	time in seconds (all problems)
8	0.21
16	0.34
32	0.58
64	1.09
128	2.26
256	5.88

Table 6

II .6.5 Discussion of results and comparisons

The first thing we must observe is that the residual in the solution of the basic problem by Newton's method must be reduced to a level compatible with the accuracy expected in the corrected solution. That is the rationale behind our stopping criterion (2.47). For this type of problem, Newton.9 method is known to be quadratically convergent and this theoretical fact is supported by the numerical behavior of our iteration. Therefore, we see that as soon as the residual is reduced below $.1h^2$ (and this has occurred in all our experiments after four iterations at the most), then in the following two steps we should have residuals approximately $< .01\ h^4$, $10^{-4}h^8$ and stop. Experiments using a less stringent stopping criterion show that on the average one might save one Newton iteration, at the risk of losing several figures accuracy.

Unfortunately, the "double precision" in IBM System 360 does not provide a sufficiently long word to test the asymptotic behavior of this very precise technique, and therefore the computed exponents for the corrected, supposedly $O(h^8)$ solution, are somewhat erratic. However, in the regions where there is not too much round off contamination, the computed exponents lie around 8 as they should.

We can compare the results of Table 5 with those in Pereyra [1968] There, an iterated deferred correction procedure was implemented, based on the $O(h^4)$ formula (2.36). Details of this implementation will be given later on. It is interesting to compare the results of the first correction, as performed in [38] with the results of Table 5, the difference in the two procedures being that if we plan to correct more than once then a full

nonlinear problem has to be solved at each correction, as opposed to the procedure described here. The other important difference is that in [38] advantage was taken of the periodicity of the solution, thus using symmetric formulas throughout. Naturally the basic solutions coincide, so we only compare the errors for the $O(h^8)$ corrected solutions.

$error_8/n$	10	20	40	80
	(8)	(8)	(8)	
[38]	4.2, -9	1.6, -11	6.2, -14	2.4, -16
This	--	(9)	(6)	(-)
method	9.4, -0	1.7, -11	2.4, -13	[illegible]

The numbers in parentheses are the computed exponents. We see that the method in [38] gives results that have a more clear asymptotic behavior. This can partly be explained by the fact that the results of that paper were obtained using double precision on a CDC 3600 computer, i.e. with numbers with 84 binary digits mantissas. We would like to point out however that the actual errors are comparable for n = 10, 20, 40 where the point of diminishing returns (on this computer) is reached for the present, more economical algorithm.

Problem 2 is used as a test problem in various papers that deal with high order spline approximations via a Raleigh-Ritz approach [5, 41, 46], in Keller [1972] where a successive Richardson extrapolation procedure is employed, and also in Wasserstrom [1973] using a continuation technique. Since the only meaningful comparison is that of programs written in the same language, running on the same machine (ideally under the same conditions or environment), we feel that it is useless to compare our results with those

presented in [5, 41, 46] since very little information is provided in those papers about the actual implementation of the methods.

The only comment we shall make is that the highest accuracy reported in [5, 41, 46] for this problem is max.abs.error $\leq 5 \times 10^{-8}$ (cf. Table 2!). Keller reports max.abs.error $\leq 4.01 \times 10^{-12}$, obtained with an $O(h^2)$ discrete method for systems of first order equations, plus three extrapolations. The mesh sizes used by Keller were $\frac{1}{3}$, $\frac{1}{6}$, $\frac{1}{12}$ and $\frac{1}{24}$. Unfortunately, as we pointed out in Section II.5, the accurate solution is obtained only on the coarsest mesh, i.e. at the two points $x = \frac{1}{3}, \frac{2}{3}$. A glance to Table 2 shows that results slightly more accurate than those of Keller can be obtained by the method of this Section with a 15 point mesh, and that these results are valid over the whole grid. In the next Section, we shall make some general comments comparing the amount of arithmetic and function evaluations that are necessary for successive Richardson extrapolations and for our procedure. Wasserstrom reports results accurate to six figures with 16 seconds of computing time on a GE-635 machine (cf. Table 6!).

Similar comments apply to the results of Tables 3 and 4. For instance, the best results (in terms of accuracy) of Ciarlet, Schultz and Varga [1968] for Problem 3 are improved by our results of Table 3 with $n = 32$:

max.error [5] = 5.49×10^{-6},

max.error [this method: n=32] = 7.06×10^{-7}.

As we said above, these comparisons are not too meaningful. For instance, it can be argued that the spline approach produces solutions defined over the whole interval, as opposed to the discrete solutions furnished by finite difference techniques. On the other hand, there is nothing to prevent us from obtaining a posteriori an accurate spline interpolation

of our discrete data . Once a definite goal is stated, for instance: "find an algorithm capable of approximating the solution of the differential equation at any point of the interval [a,b] with absolute precision ε", then two different algorithms can be analyzed in terms of their costs to achieve the desired goal. In order to obtain this, fair implementations must be tested on the same installation and the true costs compared. It is in this light that we have tried to produce careful, usable implementations of the techniques described in these notes, and that we include here the actual computer programs with, and conditions under, which the numerical results were obtained, with the hope that our experiments will be reproducible and therefore future, better methods, can make accurate claims. Also we expect that by making available these well-documented, easy to use, subroutines, the public: will be served in an area which is begging for such material.

In the next Section we present a computer printout of the program used to obtain the numerical results of this Chapter.

II.6.6 A FORTRAN IV program for the $O(h^8)$ method of II.6.3

In this Section we present the FORTRAN IV subroutine DCBVP8 with which we obtained the results of Section II.6.4. This subroutine calls the unsymmetric tridiagonal linear systems solver subroutine TRISOL and the subroutine U2DCG that generates the necessary correction terms. These two subroutines, the driver program and the subroutines defining the equations, are also provided. U2DCG will be described in detail in Chapter 3.

```
      SUBROUTINE DCBVP8(N, F, DFY, X, Y)
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL DEFCOR
      DIMENSION X(257),Y(257),A(257),B(257),C(257),R(257),AA(50)
     *  ,FU(257),DFU(257)
C     ***************************************************************
C     8TH ORDER FINITE DIFFERENCE TWO POINTBOUNDARY VALUE PROBLEM
C     SOLVER FOR
C                        -Y''+F(X,Y) = 0 , Y(X(1))=Y(1) , Y(X(N+1))=Y(N+1)
C     THE H**4 ORDER METHOD
C      H**-2 *(-Y(I-1)+2*Y(I)-Y(I+1))+(F(I-1)+10*F(I)+F(I+1))/12 = 0
C     IS COUPLED WITH ONE LINEAR DEFERRED CORRECTION IN ORDER TO PRODUCE
C     AN H**8 ORDER METHOD.
C  ****LIMITED TO N =(X(N+1)-X(1))/H .LE. 256 *****
C     TO PROCESS FINER MESHES CHANGE THE DIMENSION STATEMENTS
C     IN ALL SUBROUTINES ACCORDINGLY.
C*****USER PROVIDED DATA****+
C     X(1) = LEFT END ABSCISA
C     X(N+1) = RIGHT END ABSCISSA
C     Y(1) AND Y(N+1) : CORRESPONDING BOUNDARY VALUES.
C     N+1 IS THE NUMBER OF MESH POINTS (COUNTING THE END POINTS.
C     THEY ARE ASSUMED TO BE EVENLY SPACED BY H =(X(N+1)-X(1))/N
C     F , DFY ARE EXTERNAL USER PROVIDED SUBROUTINES THAT SHOULD PRODUCE
C     THE MESH FUNCTIONS F(X(I),Y(I)) , DF/DY(X(I),Y(I)), I=2,...,N, RESP.
C     THEIR CALLING SEQUENCES MUST BE:
C                          F(N, X, Y, FU)
C                        DFY (N, X, Y, DFU)
C     WHERE FU(257),DFU(257) ARE THE ONE-DIMENSIONAL ARRAYS TO BE
C     FILLED WITH THE REQUIRED MESH FUNCTIONS.
c     ON OUTPUT THE ARRAY Y WILL CONTAIN THE COMPUTED DISCRETE SOLUTION.
      NP1=N+1
      H=(X(NP1)-X(1))/N
      HSQ=H**2
C*****  NEXT STATEMENT IS INSTALLATION DEPENDENT *********
C     IF THIS PROGRAM IS NOT USED ON AN IBM/360 COMPUTER IN REAL*8 PREC.
C       THE CONSTANT 5.0D-16 SHOULD BE REPLACED BY (APPROXIMATELY)
C       10*MACHINE PRECISION IN ORDER TO AVOID UNDUE CYCLING IN THE
C       NEWTON ITERATION.
      EPS=DMAX1(5.0D-16,.0001*HSQ**4)
      DEFCOR=.FALSE.
      C1=(Y(NP1)-Y(1))/N
      DO 5 I=1,50
    5 AA(I)=0.D0
      DO 10 I=2,N
      X(I)=X(1)+(I-1)*H
   10 Y(I)=C1*(I-1)+Y(1)
      HSQ12=HSQ/12
      A1=5.*HSQ/6
      ITNEW=0
   15 CALL F(N, X, Y, FU)
      RESN=0.
```

```
      DO 20 I=2,N
      R(I)=Y(I-1)-2*Y(I)+Y(I+1)-HSQO12*(FU(I-1)+10.*FU(I)+FU(I+1))
      TE=DABS(R(I))
20    IF(TE .GT. RESN) RESN=TE
      IF(RESN .LE. EPS) GO TO 500
25    CALL DFY(N,X,Y,DFU)
      DO 30 I=2,N
      A(I-1)=A1*DFU(I)+2.
      B(I)=HSQO12*DFU(I)-1.
30    C(I-1)=HSQO12*DFU(I+1)-1.
      NM1=N-1
      CALL TRISOL(A,B,C,R,NM1)
      ITNEW=ITNEW+1
      DO 40 I=2,N
40    Y(I)=Y(I)+R(I)
      IF(DEFCOR) RETURN
      IF(ITNEW .LE. 10) GO TO 15
500      AA(5)=-.1
      AA(7)=-11.D0/84.D0
      DEFCOR=.TRUE.
      CALL U2DCG(1,4,4,N, AA,FU,R, IERROR,.TRUE.)
      DO 100 I=2,N
100   R(I)=-HSQ*R(I)
      GO TO 25
      END
```

```
      SUBROUTINE TRISOL(A,B,C,F,N)
      IMPLICIT REAL*8(A-H,O-Z)
C     ***************************************************************
C     UNSYMMETRIC TRIDIAGONAL SYSTEM SOLVER
C     A : MAIN DIAGONAL; B : LOWER SUBD.; C : UPPER SUBD.
C     F : RIGHT HAND SIDE. DESTROY EDANDREPLACEDBYSOLUT I ON.
C     THE ITH EQUATION IS:
C              B(I)*X(I-1)+A(I)*X(I)+C(I)*X(I+1)=F(I+1) ,
C     I=1,...,N ;  N CORRESPONDS TO (N-1) IN THE MAIN PROGRAM.
C     ***************************************************************
      DIMENSION A(257),B(257),C(257),F(257)
C     FACTORIZATION
      no 10 I=2,N
      IM1=I-1
        C(IM1)=C(IM1)/A(IM1)
 10   A(I)=A(I)-B(I)*C(IM1)
      F(2)=F(2)/A(1)
      DO 20 I=2,N
 20   F(I+1)=(F(I+1)-B(I)*F(I))/A(I)
C     BACK SOLUTION
      NM1=N-1
      NP1=N+1
      DO 30 I=1,NM1
      IN=NP1-I
 30   F(IN)=F(IN)-C(IN-1)*F(IN+1)
      RETURN
      END
```

```
      SUBROUTINE U2DCG(K,P,Q,N,A,Y,S,IERROR,EVEN)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER P,Q
      LOGICAL EVEN
      DIMENSION A( 50),Y(257),S(257),C(50)
C     ********************************************************************
C     THIS IS AN UNIVERSAL TWO POINT BOUNDARY VALUE DEFERRED CORRECTION
C     GENERATOR.
C     GIVEN THE ASYMPTOTIC EXPANSION
C
C              T(K) = SUM(A(J)*(D**(J-1))Y/(J-1)! * H**(J-1))
C                          J = Q+1,...,Q+P*K
C
C     AND FUNCTION VALUES Y(1),...,Y(N+1), CORRESPONDING TO AN
C     UNIFORMLY H-SPACED MESH : X(I) = X(1) + (I-1)*H , I = 1,...,N+1,
C     U2DCG WILL PRODUCE S(2),...,S(N) : AN H**(Q+P*K) ORDER
C     APPROXIMATION TO T(K) AT THE INTERIOR GRID POINTS.
C     FOR FIXED INTEGERS N,P,Q, A RESTRICTION ON K IS
C     ********                    K .LE. (N+1-Q)/P                ********
C     ALSO P .GE. 1 , Q .GE. 1 , K .GE. 1
C     IERROR = 1 MEANS THAT ONE OF THESE CONDITIONS HAVE BEEN VIOLATED
C     AND NO CORRECTION HAS SEEN COMPUTED.
C     A(1), . . .,A(Q) APE SET TO ZERO BY U2DCG.
C     IF THE EXPANSION T(K) HASONLY EVEN DERIVATIVES THEN EVEN SHOULD
C     BE SET TO .TRUE. OTHERWISE IT SHOULD SF SET TO .FALSE.
C
C     FEBRUARY 1973   ****************    VICTOR PERFYRA
C     ********************************************************************
      IF (K .GT. (N+1-Q)/P .OR. P .LT. 1 .OR. Q .LT. 1 .OR. K .LT. 1)
     *  GO TO 100
      DO 20 I=1,Q
   20 A(I)=0.
      KK1=Q+P*K
      KK=KK1-1
      KMID=KK1/2
      IERROR=0
      KMID1=KMID-1
      KINT=KK1
C     UNSYMMETRIC APPROXIMATION. LEFT BOUNDARY.
    1 IF(KMID1.LT.2 ) GO TO 1 0
      DO 5 I=2,KMID1
      CALL COEGEN(KK1, I,C,A)
      ACUM= 0.
      DO 4 J=1,KK1
    4 ACUM=ACUM+C(J)*Y(J)
    5 S(I)=ACUM
C     CENTER, RANGE
   10 IF(.NOT. EVEN) GO TO 25
   30 KINT=KK
   25 CALL COEGEN(KINT,KMID,C,A)
      NF=N+1-KINT+KMID
      DO 40 I=KMID,NF
      ACUM=0.
      II=I-KMID
      DO 38 J=1,KINT
   38 ACUM=ACUM+C(J)*Y(II+J)
   40 S(I)=ACUM
```

```
C       RIGHT BOUNDARY
        IF(KMID1 .LT. 2) RETURN
        KMIDP1=KMID+1+KK1-KINT
        DO 50 I=KMIDP1, KK
        CALL COEGEN(KK1, I, C, A)
        ACUM= 0.
        II=N-KK
        I1=N+1-KK1
        DO 48 J=I, KK1
   48   ACUM=ACUM+C(J)*Y(II+J)
   50   S(I+I1)=ACUM
        RETURN
  100   IERROR=1
        RETURN
        END
        SUBROUTINECOEGEN(N, NP, C, BR)
        IMPLICIT REAL*8(A-H, O-Z)
        DIMENSION C(50), BB(50), ALF(50)
C       ***************************************************************
C       THIS IS A SLIGHTLY MODIFIED FORTRAN 4 VERSION OF THEALGOL
C       PROCEDURE PVAND, P. 901 OF
C       "SOLUTION OF VANDERMONDE SYSTEMS OF EQUATIONS" BY
C       A. BJORCK AND V. PEREYRA. MATH. COMP. VOL. 24, PP. 893-903 (1970),
C       WHERE A COMPLETE DESCRIPTION OF THE METHOD USED CAN BE FOUND.
C       THIS IMPLEMENTATION ASSUMES THAT THEALF(I) ARE INTEGERS.
C       ***************************************************************
        DO 1 I=1, N
    1   C(I)=BB(I)
        DO 11 I=1, N
   11   ALF(I)=I-NP
    2   NN=N-1
        DO 6 I=1, NN
        LL=N-I
        DO 6 J=1, LL
        K=N-J+1
    6   C(K)=C(K)-ALF(I)*C(K-1)
        DO 8 I=1, NN
        K=N-I
        XKIN=1.DO/K
        KM1=K+1
        DO 8 J=KM1, N
        C(J)=C(J)*XKIN
        JM1=J-1
    8   C(JM1)=C(JM1)-C(J)
        RETURN
        END
```

```
C       MA IN PROGRAM FOR TESTING 8TH ORDER METHOD FOR 2 PVBPR.
        IMPLICIT REAL*8(A-H.O-7)
        EXTERNAL F1, F2, F3, F4, DFY1, DFY3, DFY4
        DIMENSION X(257),Y(257), IPROB(10),YEX(257)
        PI=3.14159265358979300
        READ,(IPROB(I), I=1,4),JJ,N
        DO 100 J=1,JJ
        N=2*N
        DO 100 I=1,4
        IF(IPROB(I) .GT. 0) GO TO (1,2,3,4),I
        GO TO 100
 1       X(1)=0.
        X(N+1)=PI
        Y(1)=0.
        Y(N+1)=0.
        PRINT,' PROBLEM. J , N=',N
        CALL DCBVP8(N, F1, DFY1, X, Y)
        CALL EXACT1(YEX, X, N)
        GO TO 10
 2       X(1)=0.
        X(N+1)=1.0
        Y(1)=0.
        Y(N+1)=0.
        PRINT, ' PROBLEM 2 .N=',N
        CALL DCBVP8 (N, F2, Fi, X, Y)
        CALL EXACT2(YEX, X, N)
        GO TO 10
 3       X(1)=0.
        X(N+1)=1.
        Y(1)=1.
        Y(N+1)=1.
        PRINT,' PROBLEM 3 , N=',N
        CALL DCBVP8(N, F3, DFY3, X, Y)
        CALL EXACT3(YEX, X, N)
        GO TO 10
 4       X(1)=0.
        X(N+1)=1.
        Y(1)=0.
        Y (N+1)=0.
        PRINT,' PROBLEM 4 , N=',N
        GALL DCBVP8(N, F4, DFY4, X, Y)
        CALL EXACT4(YEX, X, N)
 10       ERRNOR=0.
        DO 35 L=2,N
        ERR=DABS(YEX(L)-Y(L))
C         IF(YEX(L) .EQ. 0.) GO TO 35
c         ERR=ERR/DABS(YEX(L))
        IF(ERR .GT. ERRNOR) ERRNOR=ERR
c         PRINT,X(L),Y(L),YEX(L),ERR

 35     CONT I NUE
        PRINT, ERRNOR
 100     CONTINUE
        STOP
        END
```

```
      SUBROUTINE  F1(N,X,Y,FU)
       IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION  X(257),Y(257),FU(257)
      N1=N+1
      DO  10 I=1,N1
      SI=DSIN(X(I))
10    FU(I)=Y(I)**3-SI*(1.+SI**2)
      RETU RN
      END
      SURROUT t NE DFY1(N, X, Y, DFU)
      IMPL I C IT REAL*8 (A-H, O-Z)
      DIMENSION X(257), Y(257), DFU(257)
      N1=N+ 1
      DO  10 I=1,N1
10    DFU( I )=3. *Y( I )**2
      RETURN
      END
      SUBROUTINE  EXACT1(YEX,X,N)
      IMPL I C IT REAL*8 (A-H, 0- Z)
      DIMENSION  YEX(257),X(257)
      no  10 I=2, N
10    YEX(I)=DSIN(X(I))
      RETURN
      END
      SUBROUT INE EXACT2 (YEX, X, N)
      IMPL IC IT REAL+8 (A-H, O-Z)
      DIMENSION YEX(257),X(257)
      C=1.336055694906 10800
      CO2=.5*C
      DLN2=-DLOG(2. DO)
      DO  10 I=2,N
10    YEX(I)=DLN2+2.*DLOG(C/DCOS(CO2*(X(I)-.5)))
      RETURN
      END
      SUBROUT I NE  F2 (N, X, Y, FU)
      IMPL IC IT REAL*8 (A-H, O-Z)
      DIMENSION  X(257),Y(257), FU(257)
      N1=N+1
      DO  10 I=1,N1
10    FU( I )=DEXP(Y(I))
      RETURN
      END
```

```
      SUBROUTINE F3(N, X, Y, FU)
      IMPLICIT REAL*8(A-H, O-t)
      DIMENSION X(257), Y(257), FU(257)
      N1=N+1
      TWOPI=6.2831853071795860D0
      TPSQ=TWOPI*TWOPI
      DO 10 I=1,N1
      TPX=TWOPI*X(I)
      SI=DSIN(TPX)
      EXPSI=DEXP(SI)
   10 FU(I)=Y(I)*(1.+Y(I)*Y(I))+EXPSI*(TPSQ*(DCOS(TPX)**2-SI)-
     *  EXPSI*EXPSI-1.)
      RETURN
      END
      SUBROUTINE DFY3(N, X, Y, DFU)
      IMPLICIT REAL+8 (A-H, O-Z)
      DIMENSION X(257), Y(257), DFU(257)
      N1=N+I
      DO 10 I=1,N1
   10 DFU(I)=1.+3.*Y(I)*Y(I)
      RETURN
      END
      SUBROUTINE EXACT3(YEX, X, N)
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION YEX(257), X(257)
      TWOPI=6.2831853071795860D0
      DO 10 I=2,N
   10 YEX(I)=DEXP(DSIN(TWOPI*X(I)))
      RETURN
      END
      SUBROUTINE F4 (N, X, Y, FU)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(257),Y(257), FU(257)
      N1=N+1
      DO 10 I=1,N1
   10 FU(I)=.5*(Y(I)+X(I)+1.)**3
      RETURN
      END
      SUBROUTINE DFY4 (N, X, Y, DFU)
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION X(257),Y(257),DFU(257)
      N1=N+1
      DO 10 I=1,N1
   10 DFU(I)=1.5*(Y(I)+X(I)+1.)**2
      RETURN
      END
      SUBROUTINE EXACT4 (YEX, X, N)
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION YEX(257),X(257)
      DO 10 I=2,N
   10 YEX(I)=2./(2.-X(I))-X(I)-1.
      RETURN
      END
```

II.6.7 Operation count

In this Section we shall make an operation count for the $O(h^8)$ algorithm just described. First of all, each Newton iteration requires (n-l) evaluations of the function f and its partial derivative f_Y. All the other operations required are arithmetic or logical and our count refers to the former. "M" will stand for multiplications or divisions, and "A" will stand for additions or subtractions. Integer operations are not counted. We call nl = n-l , and we shall essentially keep only the higher order terms in the total count.

The main steps in a Newton iteration are:

(a) Computation of residual: $5n_1A + 3n_1M$

(b) Setting tridiagonal system: $3n_1A + 3n_1M$

(c) Solution of tridiagonal system: $3n_1A + 5n_1M$

(d) Updating of Y n_1A

(2.48) $12n_1A + 11n_1M$/Newton iteration

We won't count the operations involved in the computation of the initial value Y^o by linear interpolation since that can be considered as a step common to all techniques of this type.

Finally we have to account for the work involved in computing the correction. We have:

(a) 5 calls to the Vandermonde solver: $96A + 64M$ (independent of n!)

(b) Calculation of S: $7n_1A + 7n_1M$

(c) Parts b), c), d) of Newton: $7n_1A + 8n_1M$

$14n_1A + 15n_1M$.

Therefore, if' i Newton iterations are performed, then the total work will be

(2.49) $TW = (12i + 14)n_1A + (11i + 15)n_1M$.

Let us consider, for instance, Problem 2. For $h = 1/32$, four Newton iterations were required in order to decrease the residual below 1.55, -17, and the corrected $O(h^8)$ result had a max. abs. error of 4.08, -15 at the grid points. Formula (2.49) tells us that the total number of operations is then:

(2.50) $TW(\text{prob.2;def.corr.;n=32}) = 1922\ A + 1829\ M$.

Since the basic method is clearly $O(h^4)$ we can estimate what kind of a mesh would give us equivalent accuracy (n=256 is almost there, but not quite). In fact we would need 790 points in order to achieve that accuracy (provided that roundoff' does not ruin it first). From the number of Newton iterations required for the various mesh sizes shown in Table 2 we can expect that again 4 iterations will be needed for n=790 and the operation count will be in this case:

(2.51) $TW(\text{prob.2;}O(h^4)\text{method;n=780}) = 37440\ A + 34320\ M$.

We see comparing (2.50) and (2.51) that the $O(h^4)$ method will need approximately 20 times more arithmetic operations than the corrected one. Also we must count the number of function evaluations. In this problem $f_y \equiv f \equiv e^y$, but in order to make a general statement we shall count $2n_1$ function evaluations per Newton step. Thus the $O(h^8)$ method, with n=32 , requires $FE_8 = 330$, while the $O(h^4)$ method with n=780 will require $FE_4 = 6248$, i.e. again about 20 times more work. We should also mention that 25 times more storage will be needed for the $O(h^4)$ method to achieve the desired accuracy. However, all this comparison is unfair. After all we expect a bona fide high order method to

perform better than a lower order one; therefore, except to emphasize this fact in a specific case, we should look for stronger competitors.

Successive Extrapolations (SE)

With minor modifications our program for deferred corrections can be employed for performing an algorithm similar to the one described in 11.4. In fact, what we have done is to introduce the necessary changes in the Main Program of p. 47 and "short-circuit" the correction step in DCBVP8 by replacing the 5th statement of p. 43 by

IF(RESN.LC.EPS) RETURN

Thus, Subroutines U2DCG and COEGEN are unnecessary. Since our basic method has order 4 then, given h_o , we call Y_i^1 , i=1,..., to the (n_o-1) vectors obtained from the solutions $Y\left(\frac{h_o}{2^{i-1}}\right)$. These are to be, of course, approximations to $y^*(a + kh_o)$. We then form the successive columns of the extrapolation triangle by:

$$Y_i^{j+1} = \frac{4^{(j+1)}Y_i^j - Y_{i-1}^j}{4^{(j+1)} - 1} .$$

Observe that

$$Y_i^{j+1} - \varphi_h y^* = O(h_o^{2j+4})$$

and only two orders of h are gained per extrapolation.

We report now the max. absolute error in Y_3^j for the various problems and different initial meshes.

	Problem 1		Problem 2	
n_o	$error_8$	m	$error_8$	m
4	1.69, -9	----	4.01, -12	----
8	1.60, -11	6.72	1.64, -14	7.93
16	1.68, -13	6.57	2.87, -16	5.84
32	3.31, -14	2.34	8.01, -16	----

Table 7

	Problem 3		Problem 4	
n_0	$error_8$	m	$error_8$	m
4	6.06, -5	----	2.97, -9	----
8	8.14, -7	6.22	1.45, -11	7.68
16	2.06, -9	8.63	6.07, -14	7.90
32	7.51, -12	8.10	6.24, -16	6.60
64	1.38, -13	5.77	---	----

Table 8

n_o	Computer Times in sec. (all problems)
4	0.38
8	0.75
16	1.45
32	2.96

Table 9

It is somewhat hard to choose a reasonable criterion of comparison between these two methods. One that seems adequate is to choose two results of similar accuracies and compute the work necessary to obtain them. From Tables 3 and 8 we find that for Problem 3, DC with n-256 has an accuracy of 1.33, -13 , while SE with n_o=64 attains an accuracy of 1.38, -13 . From Table 3 also we learn that 6 Newton iterations are necessary to reduce the residuals to the necessary level for n=64, 128, 256 . Thus the total number of operations for DC is, according to (2.49):

(2.52) TW(prob.3;def.corr.;n=256) = 21930 A + 20655 M ,

For the SE procedure we recall that the basic problems for meshes n=64, 128, 256 must be solved and their results combined linearly. This last part requires $3n_oA + 6n_oM$, and combining this figure with the work required by the various Newton iterations we get

(2.53) TW(prob.3;Rich.ext.;n_o=64) = 32448A + 29952M .

The number of function evaluations is in each case:

(2.54) FE(DC) = 3598 ; FE(SE) = 5412 .

Finally, we must remark that DC gives its $O(h^8)$ solution at 256 points while SE only gives it at 64 points. The computer time required by SE for this problem was 1.79 sec., while DC took only 1.18 sec. Thus we see that in this problem, for the same accuracy and 4 times more detail, deferred corrections is 1.5 times faster than successive extrapolations, both methods being of the same asymptotic order, Also observe that Problem 3 is the most "difficult" of our set of test examples.

In the following Chapter, an Iterated Deferred Correction procedure will be developed, and we will be able to carry out additional comparisons with higher order successive extrapolations.

Finally we would like to point that in higher dimensional problems the effect of' mesh refining on the amount of work and storage is much more dramatical, as it has been indicated in Pereyra [1967].

We include some sample results and the modified programs for SE.

```
 PROBLEM              1
 RICHARDSON EXTRAPOLATION MAX. ERROR ON GRID NO=          4
      0.4705196568757231D-03
      0.2901055348636383D-04        0.6041761562297632D-06
      0.1805482215955223D-05        0.8189202094222026D-08        0.1687163991848095D-08
 ******************************
 PROBLEM              2
 RICHARDSON EXTRAPOLATION MAX. ERROR ON GRID NO=          4
      0.6084865959515851D-05
      0.3861318372266887D-06        0.6216229059274880D-08
      0.2422800153556448D-07        0.1010791311761849D-09        0.4013248067202824D-11
 ******************************
 PROBLEM              3
 RICHARDSON EXTRAPOLATION MAX. ERROR ON GRID NO=          4
      0.5058044622527962D 00
      0.1966159307805748D-01        0.1274793153359188D-01
      0.1058792328471281D-02        0.1813943816811102D-03        0.6056091188910706D-04
 ******************************
 PROBLEM              4
 RICHARDSON EXTRAPOLATION MAX. ERROR ON GRID NO=          4
      0.24777174801152 16D-0
      0.16424180240753 87D-0        0.1001009056020297D-05
      0.10481304870413 32D-0        0.1856154331625337D-07        0.2967138340585599D-08
 ********************** *****
```

```
C     MAIN PROGRAM FOR TESTING SUCCESSIVE RICH. EXTRAPOL. FOR 2PRVP.
C     FOR A GIVEN BASIC MESH N, PROBLEMS FOR WHICH IPROB(I)=1 ARE RUN.
C     (JJ-1) RICHARDSON EXTRAPOLATIONS ARE PERFORMED.
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL F1,F2,F3,F4,DFY1,DFY3,DFY4
      DIMENSION X(257),Y(257),IPROB(10),YEX(257),R(257,6),RER(6,6)
      PI=3.14159265358979300
      READ,(IPROB(I),I=1,4),JJ,N
      N1=N
      N0=N*2
      DO 100 I=1,4
      N=N1
      IF(IPROB(I) .EQ. 0) GO TO 100
      DO 1000 J=1,JJ
      N=2*N
      GO TO(1,2,3,4),I
    1  X(1)=0.
      X(N+1)=PI
      Y(1)=0.
      Y(N+1)=0.
      CALL DCBVP8(N,F1,DFY1,X,Y)
      CALL EXACT1(YEX,X,N)
      GO TO 10
    2  X(1)=0.
      X(N+1)=1.0
      Y(1)=0.
      Y(N+1)=0.
      CALL DCBVP8(N,F2,F2,X,Y)
      CALL EXACT2(YEX,X,N)
      GO TO 10
    3  X(1)=0.
      X(N+1)=1.
      Y(1)=1.
      Y(N+1)=1.
      CALL DCBVP8(N,F3,DFY3,X,Y)
      CALL EXACT3(YEX,X,N)
      GO TO 10
    4  X(1)=0.
      X(N+1)=1.
      Y(1)=0.
      Y(N+1)=0.
      CALL DCBVP8(N,F4,DFY4,X,Y)
      CALL EXACT4(YEX,X,N)
```

```
 10      ERRNOR=0.
         PO 35 L=2,N
         ERR=DABS(YEX(L)-Y(L))
         IF(ERR .GT. ERRNOR) ERRNOR=ERR
 35      CONTINUE
         RER(J,1)=ERRNOR
         LST=2**(J-1)
         LL=LST+1
         no 90 L=2,N0
         P(L,J)=Y(LL)
 90      LL=LL+LST
1000      CONTINCJE
         H0=(X(N+1)-X(1))/N0
         P O 180 L=2,N0
 180     X(L)=X(1)+(L-1)*H0
         PRINT,' PROBLEM',I
         GO TO (11,12,13,14),I
 11      CALL EXACT1(YEX,X,N0)
         GO TO 15
 12      CALL EXACT2(YEX,X,N0)
         GO TO 15
 13      CALL EXACT3(YEX,X,N0)
         CO TO 15
 14      CALL EXACT4(YEX,X,N0)
 15        PO 200 J=2,JJ
         CO=4**J
         DIV=1.D0/(CO-1.D0)
         PO 200 I1=J,JJ
         IR=JJ-I1+J
         ERRNOR=0.
         PO 190 L=2,N0
         R(L,IR)=DIV*(CO*R(L,IR)-R(L,IR-1))
         ERR=DABS(R(L,IR)-YEX(L))
         IF(ERR .GT. ERRNOR) ERRNOR=ERR
190      CONTINUE
200      RER(IR,J)=ERRNOR
         PRINT,' RICHARDSON EXTRAPOLATIONMAX. ERROR ON GRID N0=',N0
         PO 300 I1=1,JJ
300      PRINT,(RER(I1,J),J=1,I1)
         PRINT,'*****************************'
100      CONTINUE
         STOP
         END
```

```
      SUBROUTINE DCBVP8(N,F,DFY,X,Y)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(257),Y(257),A(257),B(257),C(257),R(257),AA(50)
     *  ,FU(257),DFU(257)
C     ************************************************************************
C     TWO POINT BOUNDARY VALUE PROBLEM  SOLVER  FOR
C                        -Y''+F(X,Y) = 0 ,  Y(X(1))=Y(1) ,  Y(X(N+1))=Y(N+1)
C     TYE  H**4 ORDER  METHOD
C      H**-2 *(-Y(I-1)+2*Y(I)-Y(I+1))+(F(I-1)+10*F(I)+F(I+1))/12 = 0
C      IS USED
C ****LIMITED T  O   N = (X(N+1)-X(1))/H  .LE. 256   *****
C     TOPROCESS  FINER  MESHES CHANGETHEDIMENSION STATEMENTS
C     IN ALL SUBROUTI NES ACCORD I NGLY.
C*****USER  PROVIDED  DATA*++++
C     X(1) = LEFT END ABSCISA
c     X(N+1) = RIGHT E N D ABSCISSA
C     Y(1) A N D Y(N+1): CORRESPOND I NGBOUNDARYVALUES.
C     N+1   IS THE NUMRER OF MESH POINTS (COUNTING THE END  POINTS),
C     N MUST BE GREATER OR EQUAL THAN  TWO.
C     THEY ARE ASSUMED  TO BE EVENLY SPACED BY H=(X(N+1)-X(1))/N
C     F . DFY APE EXTERNAL USER PROVIDED SUBROUTINES  THAT  SHOULD  PRODUCE
C     THE MESH FUNCTIONS F(X(I),Y(I)) , DF/DY(X(I),Y(I)), I=2,...,N, RESP.
C     THEIR CALLING  SEQUENCES  MUST  RF:
C                          F(N,X,Y,FU)
C                        DFY(N,X,Y,DFU)
C     WHERE FU(257),DFU(257) ARE THE ONE-DIMENSIONAL ARRAYS TO BE
C     F I LLED W I T H THE REQUI R E D MESH FUNCTIONS.
C     ON  OUTPUT  THE  APRAY Y WILL  CONTAIN  THE  COMPUTED  DI SCRETE SOLUTION.
      NP1=N+1
      H=(X(NP1)-X(1))/N
      HSQ=H**2
C*****  NEXT STATEMENT  IS I NSTALLATI ON DEPENDENT *********
C      IF  THIS PROGRAM IS NOT  USED  ON  AN  I BM/360 COMPUTER  IN  REAL*8 PREC.
C        THE  CONSTANT  5.0D-16 SHOULD RF REPLACED BY (APPROX IMATELY)
C        10*MACHINE PRECISION   IN ORDER TO AVO I D  UNDUE  CYCLING  IN THE
C        NEWTON  ITERATION,
      EPS=DMAX1(5.0D-16,.0001*HSQ**4)
      C1=(Y(NP1)-Y(1))/N
      PO  10 I=2,N
      X(I)=X(1)+( I-1)*H
 10   Y(I)=C1*(I-1)+Y(1)
      HSQ012=HSQ/12
      A1=5.*HSQ/6
      I TNEW=0
```

```
15    CALL F(N,X,Y,FU)
      RESN=0.
      PO 20 I=2,N
      R(I)=Y(I-1)-2*Y(I)+Y(I+1)-HSQ012*(FU(I-1)+10.*FU(I)+FU(I+1))
      TE=DABS(R(I))
20    IF(TE .GT. RESN) RESN=TE
      IF(RESN .LE. EPS) RETURN
25    CALL DFY(N,X,Y,DFU)
      DO 3 0 I=2,N
      A(I-1)=A1*DFU(I)+2.
      B(I)=HSQ012*DFU(I)-1.
30    C(I-1)=HSQ012*DFU(I+1)-1.
      IF(N-2)35,35,36
35    R(2)=R(2)/A(1)
      GO TO 3 7
36    NM1=N-1
      CALL TRISOL(A,B,C,R,NM1)
37    ITNEW=ITNEW+1
      DO 4 0 I-2,N
40    Y(I)=Y(I)+R(I)
      IF(ITNEW.LE.10)GO T O 1 5
      END
```

III. Computer implementation of iterated deferred corrections for boundary value problems

The iterated deferred correction (IDC) algorithm described in Chapter II, p. 27, requires the computation of the various correction operators S_k that approximate the sections of the local truncation error. Given a known basic discretization F_h of order p, and the kth segment of the asymptotic expansion for the local truncation error:

$$(3.1) \qquad T_k(x) = \sum_{j=q}^{q+pk-1} a_{j+1} \frac{y^{*(j)}(x)}{j!} h^j ,$$

where the coefficients a_j are independent of h, we would like to have a flexible, fast, and accurate algorithm capable of producing the weights w_s that define S_k (see (2.30)). In the next Section, we develop such an algorithm, which can also be used for other applications. The fact that the sum (3.1) starts from $q \neq p$ has been added for even further flexibility. There are situations in which the order p and the first derivative appearing in the expansion do not coincide, in which case this added flexibility will come in handy. Subroutine U2DCG of p. 45 is a FORTRAN IV implementation of our algorithm.

III.1 An automatic weight generator f'or numerical differentiation and other applications

Given a smooth function $y(x)$, an uniform mesh of: size h with points $\{x_i\}$, and an abscissa $\bar{x}$, we are interested in approximating the number

$$(3.2) \qquad L(y)(?) = \sum_{j=0}^{m} a_{j+1} \frac{y^{(j)}(\bar{x})}{j!} hj$$

by means of a linear combination of values of the function y(x) at some of the mesh points:

$$(3.3) \qquad M(y) = \sum_{s=1}^{m+1} w_s y(x_{s-r}) ,$$

where r is a given integer. The algorithm that we are about to describe can be easily adapted to the case of nonuniform meshes, but here we prefer to present it in its simplest form.

We assume that the accuracy required is $O(h^{m+1})$. It is well known (see Collatz [1960], Ballester and Pereyra [1967]) that if y(x) has (m+1) continuous derivatives then the approximation (3.3) exists if one takes $t = m + 1$ different abscissas.

Proof: Let $\alpha_s = (x_{s-r} - \bar{x})/h$, and let us expand M(y) in a Taylor series around $\bar{x}$:

$$M(y) = \sum_{s=1}^{t} w_s \sum_{j=0}^{t-1} \frac{y^{(j)}(\bar{x})}{j!} h^j \alpha_s^j + \sum_{s=1}^{t} w_s \alpha_s^t \frac{y^{(t)}(\xi)}{t!} h^t$$

or

$$(3.4) \qquad M(y) = \sum_{j=0}^{t-1} \left(\sum_{s=1}^{t} w_s \alpha_s^j \right) \frac{y^{(j)}(\bar{x})}{j!} h^j + \left(\sum_{s=1}^{t} w_s \alpha_s^t \right) \frac{y^{(t)}(\xi)}{t!} h^t .$$

Our aim is to make the difference $M(y) - L(y)(\bar{x})$ as large an order of h as possible. Matching terms with the same powers of h generates the following conditions for the weights w_s

$$(3.5) \qquad \sum_{s=1}^{t} w_s \alpha_s^j = a_{j+1} .$$

In order to make the linear system (3.5) square we can impose as many as t of these conditions, i.e. $j=0,\ldots,t-1$. If this system of linear equations can be solved then the resulting M(y) will have the property

$$(3.6) \qquad M(y) = L(y)(\bar{x}) + \left(\sum_{s=1}^{t} w_s \alpha_s^t\right) \frac{y^{(t)}(\xi)}{t!} h^t .$$

But system (3.5) is a Vandermonde system of equations and since the α_s are distinct it is non-singular. □

Therefore, our problem of evaluating the appropriate weights w_s has been reduced to solving systems of linear equations of the form:

$$(3.7) \qquad V(\underset{\sim}{\alpha})\underset{\sim}{w} = \underset{\sim}{a}$$

where $\alpha^T = (\alpha_1,\ldots,\alpha_t)$, $\underset{\sim}{a}^T = (a_1,\ldots,a_t)$, $\underset{\sim}{w}^T = (w_1,\ldots,w_t)$

and $V(\alpha)$ is the Vandermonde matrix:

$$(3.8) \qquad V(\underset{\sim}{\alpha}) = \begin{bmatrix} 11 & . & . & . & 1 \\ \alpha_1 & \alpha_2 & & & \alpha_t \\ \alpha_1^2 & \alpha_2^2 & & & \alpha_t^2 \\ . & . & & & . \\ \alpha_1^{t-1} & & & & \alpha_t^{t-1} \end{bmatrix} .$$

It is well known that Vandermonde matrices are ill-conditioned (cf. Gautschi [1962, 1965]), and if one attempts to use a standard Gaussian elimination code on this type of problem, failure occurs even for very modest sizes. Great loss of accuracy is also common, even for $t=5$. Fortunately, there are techniques for solving this and similar kinds of problems that not only are more accurate and stable, but that also

take much less arithmetic operations to produce the desired result. Naturally, they take advantage of the special structure of the matrix $V(\underset{\sim}{\alpha})$ (see Ballester and Pereyra [1967], Björck and Pereyra [1970], Galimberti and Pereyra [1970, 1971] and Pereyra and Scherer [1973] for more details, generalizations and other applications).

In Björck and Pereyra [1970], a method for solving the transpose (dual) problem

$$V^T(\underset{\sim}{\alpha})\,\underset{\sim}{a} = f \tag{3.9}$$

is developed. A matrix interpretation of the method permits then the consideration of the direct problem (3.7) with little difficulty. For problem (3.9), advantage is taken of the fact that if we think of the elements of the vector f as values of a certain function, then the equations (3.7) are just the conditions of interpolation by a polynomial of degree (t-1), and therefore, the solution a will have as components the coefficients of the unique interpolation polynomial:

$$P(x) = \sum_{s=0}^{t-1} a_{s+1} x^s \ .$$

The Newton form of the interpolation polynomial P(x) is obtained if one considers the new basis given by the polynomials

$$Q_0(x) = 1 \ , \ Q_k(x) = \prod_{i=1}^{k} (x - \alpha_i) \ , \ k=1,\ldots,t-1 \tag{3.10}$$

With this basis, we have

$$P(x) \equiv \sum_{k=0}^{t-1} c_k Q_k(x) \ , \tag{3.11}$$

where the coefficients c_k are the divided differences constructed with the function values f and the abscissas α :

$$c_k = f[\alpha_1,\ldots,\alpha_{k+1}] \quad , \quad k=0,\ldots,t-1 \ .$$

It is well known that these divided differences can be obtained recursively by

$$(3.12) \quad f[\alpha_{j-k},\ldots,\alpha_{j+1}] = \frac{f[\alpha_{j-k+1},\ldots,\alpha_{j+1}] - f[\alpha_{j-k},\ldots,\alpha_j]}{\alpha_{j+1} - \alpha_{j-k}} \ ,$$

$$k=0,\ldots,t-2 \ ; \ j=k+1,\ldots,t-1 \ .$$

Once we have computed the vector c of divided differences then a Horner-like scheme permits to evaluate (3.11). In fact, we can compute

$$(3.13) \quad q_{t-1}(x) = c_{t-1} \ , \quad q_k(x) = (x - \alpha_{k+1})q_{k+1}(x) + c_k \ , \ k=t-2,\ldots,0 \ ,$$

and then clearly,

$$q_0(x) \equiv P(x) \ .$$

Let

$$q_k(x) = a_{k+1}^{(k)} + a_{k+2}^{(k)}x + \ldots + a_t^{(k)}x^{t-1-k} \ ,$$

$$a_{j+1}^{(t-1)} = c_j \quad , \quad j=0,\ldots,t-1 \ .$$

If we replace these expressions in (3.13) we obtain a simple recursion for the coefficients $a_j^{(k)}$, $k=t-2,t-3,\ldots,0$

$$a_j^{(k)} = a_j^{(k+1)} \quad , \quad j=1,\ldots,k \ ; \ t$$

$$(3.14) \quad a_{k+1}^{(k)} = c_k - \alpha_{k+1}a_{k+2}^{(k+1)} \ ,$$

$$a_j^{(k)} = a_j^{(k+1)} - \alpha_{k+1}a_{j+1}^{(k+1)}, \quad j=k+2,\ldots,t-1 \ .$$

Recursions (3.12) and (3.14) furnish the solution to problem (3.9). Let us consider the lower bidiagonal matrices of order t , $L_k(a)$, defined by

$$L_k(\alpha)_{ii} = 1 \; ; \quad L_k(\alpha)_{i+1,i} = 0 \; , \; i=1,\ldots,k \; ;$$

(3.15)

$$L_k(\alpha)_{i+1,i} = -\alpha \; , \quad i=k+1,\ldots,t-1 \; .$$

We shall call

$$L_k \equiv L_k(\alpha_k) \; , \quad M_k = L_k(1) \; ,$$

and D_k to the diagonal matrices;

(3.16) $$D_k \equiv \text{diag}\,(1,\ldots,1,(\alpha_{k+2} - \alpha_1),\ldots,(\alpha_t - \alpha_{t-k-1}))\} \; .$$

With this notation it is easy to see that recursions (3.12) and (3.14) can be represented in matrix form as:

$$\underset{\sim}{c}^{(0)} = \underset{\sim}{f} \quad \underset{\sim}{c}^{(k+1)} = D_k^{-1} M_k \underset{\sim}{c}^{(k)} \; , \quad k=0,\ldots,t-2$$

(3.17)

$$\underset{\sim}{a}^{(t-1)} = \underset{\sim}{c}^{(t-1)} \; , \quad \underset{\sim}{a}^{(k)} = L_k^t \underset{\sim}{a}^{(k+1)} \; , \quad k=t-2,\ldots,0 \; .$$

Calling U^T , L^T to the lower and upper unit triangular matrices

(3.18) $$UT = D_{t-2}^{-1} M_{t-2} \ldots D_0^{-1} M_0 \; , \quad L^T = {}_0L_1^T L_2^T \ldots L_{t-2}^T$$

then we see that (3.17) can be expressed as

(3.19) $$c = U^T \underset{\sim}{f} \; , \quad \underset{\sim}{a} = L^T c \; .$$

Since $\underset{\sim}{a} = V^{-T} f$, we have then $V^{-T} = L^T U^T$, or

(3.20) $$V^{-1} = UL \; , \; V = L^{-1} U^{-1} \; ,$$

and we have found a factorization in bidiagonal factors of the unique triangular matrices furnishing the UL decomposition of the inverse Vandermonde matrix V^{-1} . With this factorization we can easily write a recursive algorithm for solving the problem of our more immediate interest, i.e. the

primal or direct Vandermonde system (3.8). In fact, we have

$$(3.21)\qquad \underset{\sim}{w} = V^{-1}(\underset{\sim}{\alpha})\underset{\sim}{a} = UL\underset{\sim}{a} = (M_0^T D_0^{-1} \ldots M_{t-2}^T D_{t-2}^{-1})(L_{t-2} \ldots l_0)\underset{\sim}{a} ,$$

from where we can easily derive a recurrence to compute w . Subroutine COEGEN (see p. 46) is a FORTRAN IV implementation of this recurrence, while in [3] Algol 60 implementations of both Vandermonde solvers and some variations can be found.

In the present implementation the user has to provide the size of the system N , the integer location of $\bar{x}$, NP , with respect to the nodes used in (-3.3). We assume here that $\bar{x}$ is actually a grid point. With this information COEGEN generates the vector α , whose components are integers: $\alpha_i = i - NP$. Therefore the elements different from 1 in the diagonal matrices D_k (see (3.16)) are simply: $\alpha_{s - *s-k-l} = k + l$, which amounts to the small modification we mentioned above. The right hand side of the system must be supplied in the array BB , while the solution to the system will be found upon output in the array C . In our application COEGEN -is called by the subroutine U2DCG that we pass to describe .

III.2 A Universal 2-point boundary value Deferred Correction Generator

As we said before the gist to an effective implementation of iterated deferred corrections lies in being able to obtain the correction operators S_k approximating the expansions (2.30) or II.6.3. The correction operators we are going to develop are of the general form (3.3), and therefore the subroutine COEGEN will be an important component in our algorithm. Other types of corrections are possible as we have pointed out in [34]. See also Denny

and Landis [1972]. The word Universal in the title of this Section refers to the fact that we hope that the generality of the subroutine U2DCG will be able to cope with a variety of different boundary value problems and various discretizations. For instance, we have seen already for the simple problem (1.1) two different discretizations which in turn produce different asymptotic expansions (cf. (2.29) and 11.6.3, p. 30). The theory developed in Chapter II, which carries over to many other situations, and our comments above are the reasons for the choice of the form (3.1) as the type of general expansions we would like to approximate.

In a two point boundary value problem, where approximations to an expansion of the form (3.1) are necessary at all the interior grid points of an uniform grid, we are faced with various standard problems:

(a) The order of the approximation must be $O(h^{q+pk})$ at each point.

(b) We like to use as centered formulas as possible since they have the smallest truncation constants and smaller weights. In the "center" of the interval we can do this without difficulties, but as soon as we get closer to the boundaries we need to use unsymmetric formulas.

These tasks are fulfilled by the subroutine U2DCG which is listed on p. 45 of these notes.

The user needs to know what kind of an expansion he wants to approximate, i.e. he has to provide the coefficients a_j in (3.1), for $j = q, \ldots, q+pk-1$ (setting to zero those for which the corresponding derivative does not appear; observe that due to programming language limitations, the coefficient, a_j corresponds to the (j-1) derivative).

The other two parameters required are the order p of the basic method from which the expansion came, and the correction step k which is desired. The number of interior nodal points plus one, N , is also required, and finally the grid function (array) Y , which will be used in formulas like (3.3) must also be provided. It is assumed that Y is an $O(h^{q+p\cdot(k-1)})$ order approximation to $y^*(x)$. On output, the correction mesh function $S_k(Y)$ is produced in the array S . The integer variable IERROR will be equal to 1 and no correction will be computed in case some of the following assumptions are violated,

(3.22) $K \leq (N+1-Q)/P$; P , Q , $K > 1$.

The condition P , Q , $K > 1$ is pretty obvious; the first condition is a constraint motivated by the fact that a minimum number of grid points are necessary to achieve a given accuracy for a certain derivative. Thus in order to obtain the required accuracy we need q + pk points. Since we count with N+1 grid points and p and q are given then that imposes a condition on k , Of course, if our expansion consists only of even derivatives then we can have sufficiently accurate symmetric formulas with only q+ pk - 1 points, at least in the center range. This case is indicated to the subroutine by setting the logical parameter EVEN to .TRUE., otherwise this parameter should be set to .FALSE. .

III.3 Asymptotic error estimation by deferred corrections

The procedure of' Section II.6.3 was actually a way of obtaining an asymptotic error estimate for the basic $O(h^4)$ solution. It turns out that a similar technique can be employed in general to estimate the error in an iterated deferred correction algorithm. We shall give now an explanation associated with the $O(h^4)$ method of 11.6.3, but this result, as most of the others, carries over to much more general situations (cf. [39]).

For k=1,... let

$$(3.23)\qquad T_k(x) = \sum_{j=4}^{4*(k+1)-1} a_j \frac{f^{(j)}(x,y^*)}{j!} h^j$$

with

$$(3.24)\qquad a_j = \begin{cases} 0 , & j \text{ odd} , \\ \frac{1}{6} - \frac{1}{(j+1)(2j+1)} , & j \text{ even} . \end{cases}$$

$$\text{Let } S_k(\varphi_h y^*)(x_i) = \sum_{s=1}^{4*(k+1)} w_{si} f(x_{i-r_i+s}, y^*_{i-r_i+s}) ,$$

where the displacement r_i will be dependent upon the position of x_i in the interval [a,b] (cf. III.2). The weights w_{si} are chosen so that

$$(3.25)\qquad S_k(Y^{(k-1)}) = T_k(x) + O(h^{4*(k+1)}) ,$$

where the discrete function $Y^{(k-1)}$ satisfies

$$(3.26)\qquad G_h(Y^{(k-1)}) = S_{k-1}(Y^{(k-2)}) ,$$

and

$$(3.27)\qquad e_{k-1} = Y^{(k-1)} - \varphi_h y^* = O(h^{4k}) ,$$

Theorem 3.1. Let Δ_{k-1} be the solution to the linear problem

(3.28) $G_h'(Y^{(k-1)})\Delta = S_{k-1}(Y^{(k-2)}) - S_k(Y^{(k-1)})$.

Then

(3.29) $\Delta_{k-1} = e_{k-1} + O(h^{4*(k+1)})$

(i.e.: Δ_{k-1} is an asymptotic error estimator for $Y^{(k-1)}$) .

Proof: Since $G_h(\varphi_h y^*) = T_k(x) + O(h^{4*(k+1)})$, we obtain, combining this relationship with (3.26):

$$G_h(Y^{(k-1)}) - G_h(\varphi_h y^*) = S_{k-1}(Y^{(k-2)}) - T_k(x)) + O(h^{4*(k+1)}) .$$

But from (3.25) and the Mean Value Theorem we can deduce that

$$O(h^{8k}) + G_h'(Y^{(k-1)})(Y^{(k-1)} - \varphi_h y^*) = S_{k-1}(Y^{(k-2)}) -$$

$$- S_k(Y^{(k-1)}) + O(h^{4*(k+1)}) ,$$

or, since $8k \geq 4*(k+1)$ for $k > 1$,

$$G_h'(Y^{(k-1)})(Y^{(k-1)} - \varphi_h y^*) = S_{k-1}(Y^{(k-2)}) - S_k(Y^{(k-1)}) + O(h^{4*(k+1)}) .$$

Subtracting this last expression from (3.28) we obtain finally

$$G_h'(Y^{(k-1)})\,[\Delta_{k-1} - e_{k-1}] = O(h^{4*(k+1)}) ,$$

and since G_h is stable, so is G_h' and the result follows. □

III.4 A variable order, variable (uniform) step, two point boundary value problem solver, based on deferred corrections

In this Section, we address ourselves to the following task:

"Given problem (1.1), (2.1), $n \geq 5$, and 10* machine precision $< \varepsilon$, find a discrete solution on an uniform mesh with at least (n+1) points and maximum absolute error less than equal to ε ."

We won't claim that our algorithm is optimal with respect to the solution of this problem, but we shall try to show that it has some good points as compared with other available techniques. In fact, the algorithm will be designed in the style of an adaptive scheme, except that the mesh will be automatically refined over the whole interval. A more complicated algorithm could be designed, such that local refinements are performed in order to follow better the local variations of the exact solution. In fact, the vector Δ_k of Theorem 3.1 provides an excellent tool for that more complicated task since it measures the error at each individual mesh point. We prefer to reserve this type of approach for situations in which the use of non-uniform meshes is unavoidable, like in the case of multipoint boundary value problems, or problems with isolated, interior discontinuities (interfaces) (see Keller (1969, 1972)).

Our strategy will be based on the Iterated Deferred Corrections (IDC) algorithm of II. 6.2, for the $O(h^4)$ discretization (2.36). We know from III.2 that for a given n there is a natural limitation on the number of corrections that can be performed. Also, from past numerical experience (and common sense), we know that for a given problem and mesh size there are also limitations on the number of corrections that will do us good.

Unfortunately, while the first limitation can be exactly predicted (i.e. $k \leq (n+1 - 4)/4)$, the second one is problem and machine dependent.

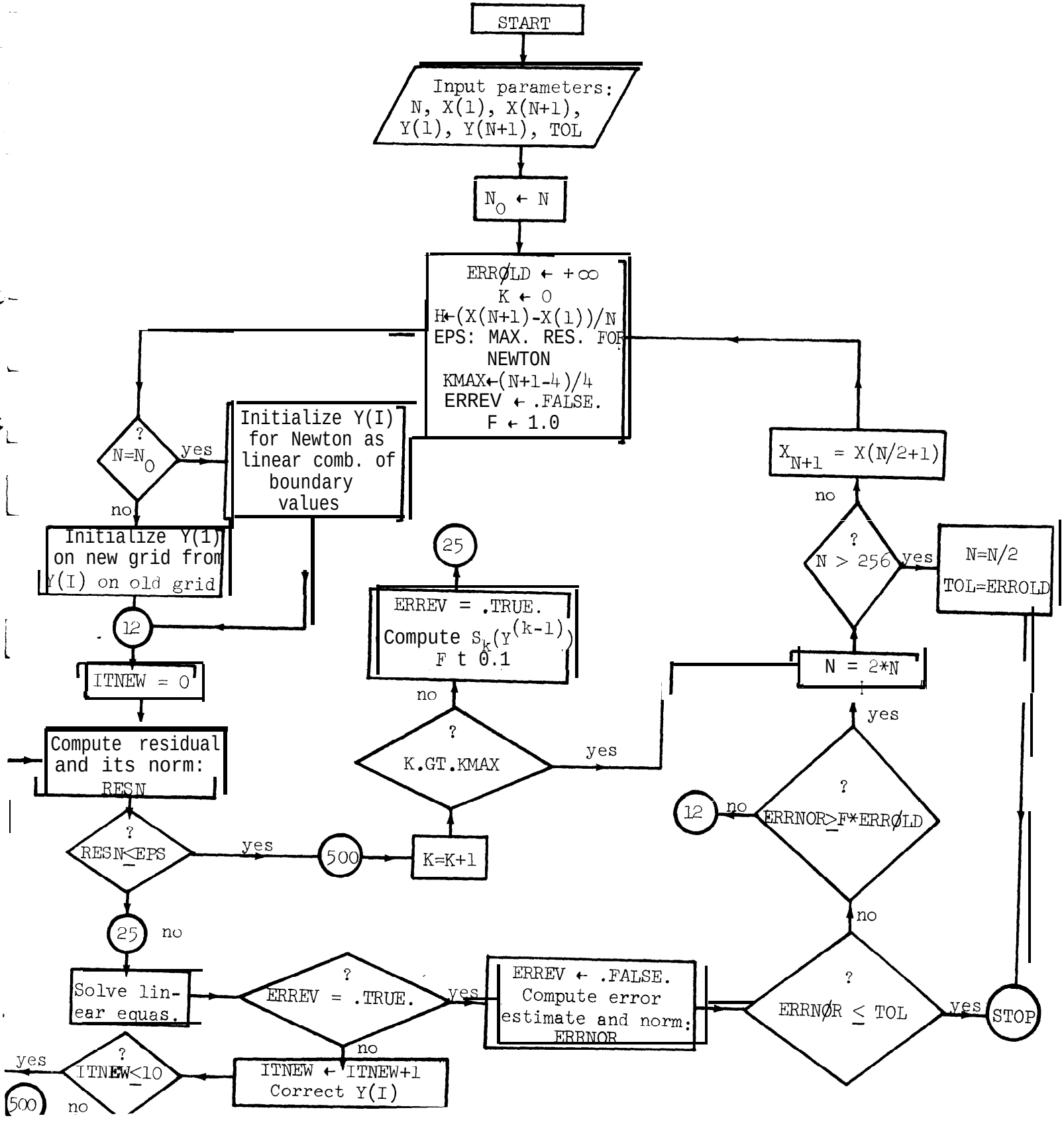
On the other hand, it turns out that the asymptotic error estimation procedure of' III.3 provides a fine, reliable tool for detecting on line the behavior of the corrections. Thus , $\| \Delta_k \| \geq \| \Delta_{k-1} \|$ is a clear indicator that the $(k+1)$th correction will not improve our solution (and also that Δ_k is not a reliable estimator for e_k) . This phenomenon is obviously connected with what in the past has been known as "the growth of high order differences", which served as a signal-to the pencil and paper numerical analyst to cut his series of differences (see also [26,27]). We see then that without having to construct, store and computer inspect a table of differences we can still extract the useful information inherently contained in the procedure. As a matter of fact, we use in our program the more strict test

$$\| \Delta_k \| \leq .1 * \| \Delta_{k-1} \| .$$

If this condition is violated we halve the mesh since we are not obtaining a sufficient reward for our pains.

A flow chart and a FORTRAN IV program for the algorithm follow. We emphasize here that by changing appropriately some boxes in the program, one can solve other problems with this same logical arrangement. Subroutine IDCBVP calls Subroutines TRISOL and U2DCG, which have been listed earlier in these notes.

Flow Chart f'or Variable Order, Variable (Uniform) Step,
Deferred Correction Solver



```
      SUBROUTINE IDCBVP(N,F,DFY,X,Y,TOL)
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL ERREV
      DIMENSION X(257),Y(257),A(257),B(257),C(257),R(257),AA(50)
     *  ,FU(257),DFU(257),S(257)
C     ***********************************************************************
C     VARIABLE ORDER, VARIABLE (UNIFORM) STEP FINITE DIFFERENCE
C     TWOPOINT BOUNDARY VALUE PROBLEM SOLVER FOR
C                     -Y''+F(X,Y) = 0 , Y(X(1))=Y(1) , Y(X(N+1))=Y(N+1)
C     THE  H**4  ORDER METHOD
C      H**-2 *(-Y(I-1)+2*Y(I)-Y(I+1))+(F(I-1)+10*F(I)+F(I+1))/12 = 0
C     IS COUPLED WITH ITERATED DEFERRED CORRECTIONS IN ORDER TO PRODUCE
C     A  DISCRETE  SOLUTION WITH  MAX,  ABS. ERROR < TOL ON A GRID NOT
C     COARSER THAN THE GIVEN N.
C ****LIMITED TO    N = (X(N+1)-X(1) )/H .LE. 256    *****
C     TO PROCESS FINER MESHES CHANGE THE DIMENSION STATEMENTS
C     IN ALL SUBROUTINES ACCORDINGLY.
C*****USER PROVIDED DATA*****
C     X(1) = LEFT END ABSCISA
C     X(N+1) = RIGHT END ABSCISSA
C     Y(1) AND Y(N+1) : CORRESPONDING BOUNDARY VALUES.
C     N+1  IS THE NUMBER OF MESH POINTS AT WHICH THE SOLUTION IS DESIRED
C     (COUNTING THE END POINTS). ON OUTPUT  N  WILL CONTAIN THE SIZE OF
C     THE MESH ON WHICH THE FINAL Y WAS ACTUALLY COMPUTED.
C     THEY ARE ASSUMED TO BE EVENLY SPACED BY  H = (X(N+1)-X(1))/N
C     TOL : USER'S DESIRED MAXIMUM ABSOLUTE ERROR NORM ON A MESH NOT
C           COARSER THAN N. ON OUTPUT  TOL  WILL CONTAIN THE ERROR
C           ESTIMATED BY  IDCBVP.
C     F , DFY ARE EXTERNAL USER PROVIDED SUBROUTINES THAT SHOULD PRODUCE
C     THE MESH FUNCTIONS  F(X(I),Y(I)) , DF/DY(X(I),Y(I)), I=2,...,N, RESP.
C     THEIR CALLING SEQUENCES MUST BE:
C                       F(N,X,Y,FU)
C                     DFY(N,X,Y,DFU)
C     WHERE FU(257),DFU(257) ARE THE ONE-DIMENSIONAL ARRAYS TO BE
C     FILLED WITH THE REQUIRED MESH FUNCTIONS.
C     ON OUTPUT THE ARRAY Y WILL CONTAIN THE COMPUTED DISCRETE SOLUTION.
      EPS=1.
      N0=N
    4 K=0
      FACT=1.D0
      ERROLD=1.D10
      NP1=N+1
      KMAX=(NP1-4)/4
      H=(X(NP1)-X(1))/N
      HSQ=H**2
      ERREV=.FALSE.
      DO 5 I=1,50
    5 AA(I)=0.D0
```

```
C       INITIALIZATION FOR NEWTON
        IF(N .EQ. N0)GO TO 9
        NHALF=N/2
        N1=NHALF+1
        DO 6 I=1,N1
    6   A(I)=Y(I)
        DO 7 J=1,NHALF
        J2=2*J
        Y(J2+1)=A(J+1)
    7   Y(J2)=.5D0*(A(J+1)+A(J))
        DO 8 I=2,N
        S(I)=0.
    8   X(I)=X(1)+(I-1)*H
        GO TO 11
    9   C1=(Y(NP1)-Y(1))/N
        DO 10 I=2,N
        S(I)=0.
        X(I)=X(1)+(I-1)*H
   10   Y(I)=C1*(I-1)+Y(1)
   11   HS0012=HS0/12
        A1=5.*HS0/6
   12   ITNEW=0
        IF(EPS .EQ. 5.0D-16) GO TO 15
        EPS=DMAX1(5.0D-16,.1*H**(4*K+10))
C*****  NEXT STATEMENT IS INSTALLATION DEPENDENT *********
C       IF THIS PROGRAM IS NOT USED ON AN IBM/360 COMPUTER IN REAL*8 PREC.
C         THE CONSTANT 5.0D-16 SHOULD BE REPLACED BY (APPROXIMATELY)
C         10*MACHINE PRECISION  IN ORDER TO AVOID UNDUE CYCLING IN THE
C         NEWTON ITERATION.
   15   CALL F(N,X,Y,FU)
        RESN=0.
        DO 20 I=2,N
        R(I)=Y(I-1)-2*Y(I)+Y(I+1)-HS0012*(FU(I-1)+10.*FU(I)+FU(I+1))-S(I)
        TE=DABS(R(I))
   20   IF(TE .GT. RESN) RESN=TE
        IF(RESN .LE. EPS) GO TO 500
   25   CALL DFY(N,X,Y,DFU)
        DO 30 I=2,N
        A(I-1)=A1*DFU(I)+2.
        B(I)=HS0012*DFU(I)-1.
   30   C(I-1)=HS0012*DFU(I+1)-1.
        NM1=N-1
        CALL TRISOL(A,B,C,R,NM1)
        IF(ERREV) GO TO 150
        ITNEW=ITNEW+1
        DO 40 I=2,N
   40   Y(I)=Y(I)+R(I)
        IF(ITNEW .LE. 10) GO TO 15
```

```
C       COMPUTATION OF SK
 500    K=K+1
        FACT=.100
        IF(K .GT. KMAX) GO TO 300
        K2=2*K+1
        DO 50 I=2,K2
        I2=2*I+1
 50     AA(I2)=1.00/((I+1)*I2)-1.00/6.00
        ERREV=.TRUE.
        CALL U2DCG(K,4,4,N,AA,EU,R,IERROR,.TRUE.)
        DO 100 I=2,N
        STE=USO*R(I)
        R(I)=S(I)-STE
 100    S(I)=STE
        GO TO 25
        ERROR CONTROL AND DECISION CENTER
 150    ERREV=.FALSE.
        ERRNOR=0.00
        DO 60 I=2,N
        TE=DABS(R(I))
  60    IF(TE .GT. ERRNOR)ERRNOR=TE
        K1=K-1

        IF(ERRNOR .LT. TOL) GO TO 200
        IF(ERRNOR .GE. ERROLD*FACT .OR. K+1 .GT. KMAX) GO TO 300
        ERROLD=ERRNOR
        GO TO 12
 200    TOL=ERRNOR
        RETURN
 300    N=2*N
        IF(N .GT. 256) GO TO 400
        X(N+1)=X(N/2+1)
        GO TO 4
 400    N=N/2
        TOL=ERROLD
        RETURN
        END
```

III.5 Numerical results

We give in this Section numerical results corresponding to the four test problems of II.6.4, and for a boundary layer type equation suggested by Sam Schecter (Stanford Research Institute). The new test problem is linear:

Problem 5:

$$-y'' - \frac{3\varepsilon y}{(\varepsilon + x)^4} = 0$$

$$y(-a) = \frac{-a}{(\varepsilon + a^2)^{\frac{1}{2}}}, \quad y(a) = -y(-a), \quad \varepsilon, a > 0 .$$

Solution:

$$y(x) = \frac{x}{(\varepsilon + x^2)^{\frac{1}{2}}} .$$

For $\varepsilon \to 0$, $y(x) \to \text{sign } x$ which has a jump discontinuity at $x = 0$. For small ε, this is a fairly hard problem to solve with finite differences.

In Table 10 we have collected various statistics about IDCBVP for Problems i, i=1,...,5. For all problems we have started with N = 8, and requested a final max.abs.error tolerance of EPS = 10^{-13}. Problem 5 parameters were $\varepsilon = 10^{-4}$, a = 0.1.

Problem	Final Estimated Error	Final True Error	Final Number of Points
1	7.0, -17	2.8, -15	32
2	5.5, -15	5.2, -15	16
3	9.0, -16	3.2, -14	128
4	2.2, -14	2.3, -14	32
5	3.8, -13	6.1, -13	256

Table 10

We see from these results that the automatic step adjustment follows closely the difficulties of the problem (recall earlier results for Problem 3). In order to have a better feeling for the actual flow of the computation for each problem we give in Table 11 some additional information. For each problem we list under the mesh size the number of nonlinear systems of that size that have been solved and the total number of Newton corrections employed (in parentheses).

Prob. No.	Points 8	16	32	64	128	256
1	1 (5)	3 (4)	3 (4)	---	---	---
2	1 (3)	3 (4)	---	---	---	---
3	1 (5)	2 (5)	3 (6)	4 (7)	3 (4)	---
4	1 (3)	3 (5)	3 (4)	---	---	---
5	1 (1)	2 (2)	3 (3)	3 (3)	4 (4)	4 (4)

Table 11

To give some idea of the behavior of the asymptotic error estimator we show the detailed evolution for a bad case: Problem 3, in Table 12.

N	k	Estimated Error	Exact Error
8	0	8.3, -2	2.0, -2
16	0	9.6, -4	1.1, -3
	1	1.4, -3	1.4, -3
32	0	6.4, -5	6.4, -5
	1	2.1, -7	7.1, -7
64	2	1.2, -6	8.6, -7
	0	4.0, -6	4.0, -6
	1	7.8, -10	8.0, -10
	2	3.86, -11	4.3, -11
	3	3.87, -12	4.4, -12
128	0	2.5, -7	2.5, -7
	1	2.5, -12	2.5, -12
	2	9.0, -16	3.2, -14

Table 12

In the following Table, we give some information about the performance of successive extrapolations on the same problems. The basic grid size is $N = 8$. We indicate the final grid size and number of extrapolations needed to reach accuracies similar to those in Table 10 for IDC, or if that was not possible for $N < 256$, then we show the best accuracy attained on the diagonal of the Richardson triangle. The number of Newton iterations is taken from Tables 1-4, pp. 35-36.

Problem	Accuracy Attained	Finer Mesh	Number of Extrapolations	Number of Newton Iterations
1	3.2, -14	128	4	7,7,7,7,8
2	2.5, -16	64	3	4,4,4,4
3	1.4, -13	256	5	6,6,6,6,6,6
4	3.6, -15	64	3	4,5,5,5
5	1.6, -7	256	5	linear problems

Table 13

Using Lemma 2.7 of p. 20 we could have actually implemented an asymptotic error estimator for the successive extrapolations method and developed an automatic error monitoring and stopping procedure. Though we cannot vouch for its success (since we didn't have the time to do it), past experience and the similarities with the asymptotic behavior of IDC indicate that it is probably a good idea. Making believe that we have done such a thing (and that the asymptotic predictions were accurate), we now indicate the best results in the whole Richardson triangles (not only in the diagonal) for each problem. Rows and columns are numbered from 1. The column (i,j) of Table 14 indicates the position of the best result in the Richardson triangle.

Problem	Best Result	(i,j)
1	6.6, -15	(5,2)
2	1.9, -16	(4,3)
3	1.3, -13	(6,3)
4	5.8, -16	(5,5)
5	3.2, -9	(6,3)

Table 14

We see that, with the exception of Problem 4, the best results are not located on the diagonal of' the Richardson triangle. These results indicate that the error monitoring should be carried out on all the elements of the Richardson triangle.

Moral: FINITE DIFFERENCES: ARE YOU REALLY DEAD?

REFERENCES

1. Aleksandrov, P. S., Combinatorial Topology, I. Graylock Press, Rochester, New York (1956).

2. Ballester, C. and V. Pereyra, "On the construction of discrete approximations to linear differential expressions". Math. Comp. 21, pp. 297-302 (1967).

3. Björck, Å. and V. Pereyra, "Solution of Vandermonde systems of equations." Math. Comp. 24, pp. 893-903 (1970).

4. Bulirsch, R. and J. Stöer, "Fehlerabschatzungen und Extrapolation mit rationalen Funktionen bei Verfharen vom Richardson-Typus." Numer. Math. 6, pp. 413-427 (1964).

5. Ciarlet, P. G., M. H. Schultz and R. S. Varga, "Numerical methods of high-order accuracy for nonlinear boundary value problems, I." Numer. Math. 9, pp. 394-430 (1967).

6. Collatz, L., The Numerical Treatment of Differential Equations. 3rd Ed., Springer-Verlag, Berlin (1960).

7. Denny, V. E. and R. B. Landis, "A new method for solving two-point boundary value problems using optimal node distribution." J. Comp. Physics 9, pp. X20-137 (1772).

8. Fox, L., "Some improvements in the use of relaxation methods for the solution of' ordinary and partial differential equations." Proc. Royal Soc. London A170, pp. 31-59 (1947).

9. Fox, L. (editor), Numerical Solution of Ordinary and Partial Differential Equations. Pergamon Press, Oxford (1962).

10. Fox, L., "Partial differential equations." Comp. J. 6, pp. 69-74 (1963).

11. Fyfe, D. J., "The use of cubic splines in the solution of two-point boundary value problems." The Comp. J. 12, pp. 188-192 (1969).

12. Galimberti, G. and V. Pereyra, "Numerical differentiation and the solution of multidimensional Vandermonde systems." Math. Comp. 24, pp. 357-364 (1970).

13. _____, "Solving confluent Vandermonde systems of Hermite type." Numer. Math. 18, pp. 44-60 (1971).

14. Gautschi, W., "On the inverses of Vandermonde and confluent Vandermonde matrices I." Numer. Math. 4, pp. 117-123 (1962). II, ibid. 5, pp. 425-430 (1963).

15. Gragg, W., "Repeated extrapolation to the limit in the numerical solution of ordinary differential equations." Doctoral dissertation, UCLA (1963).

16. _____, "On extrapolation algorithms for ordinary initial value problems." J. SIAM Numer. Anal. 2, pp. 384-403 (1965).

17. Hanson, James N., "Experiments with equation solutions by functional analysis algorithms and formula manipulation." J. Comp. Physics 9, pp. 26-52 (1972).

18. Henrici, P., Discrete Variable Methods in Ordinary Differential Equations. Wiley, New York (1962).

19. Herbold, R. J. and R. S. Varga, "The effect of quadrature errors in the numerical solution of two-dimensional boundary value problems by variational techniques." Aequationes Mathematicae, 7, pp. 36-58 (1972).

20. Joyce, C. C., "Survey of extrapolation processes in numerical analysis." SIAM Rev. 13, pp. 435-490(1971).

21. Kalaba, R. and E. Ruspini, "Identification of parameters in nonlinear boundary value problems." J. Opt. Th. App. 4, pp. 371-377 (1969).

22. Keller, H. B., "Accurate difference methods for linear ordinary differential systems subject to linear constraints." SIAM J. Numer. Anal. 6, pp. 8-30 (1969).

23. _____ Numerical Methods for Two-Point Boundary-Value Problems: Blaisdell, Waltham, Mass. (1968).

24. _____ "Accurate difference methods for nonlinear two-point boundary'value problems." To appear (1972).

25. Kreiss, H.-O., "Difference approximations for ordinary differential equations." Uppsala University Computer Science Department Report NR 35 (1971).

26. Krogh, F. T., "On testing a subroutine for the numerical integration of ordinary differential equations." Techn. Memo. No. 217, Jet Propulsion Laboratory, Pasadena, California (1970). To appear in ACM J.

27. _____, "Changing stepsize in the integration of differential equations using modified divided differences." Techn. Memo. No. 312, Jet Propulsion Laboratory, Pasadena, California (1912).

28. Kronsjö, L. and G. Dahlquist, "On the design of nested iterations for elliptic difference equations." Report nr NA 71.30, Royal Inst. of Techn., Stockholm, Sweden (1971).

29. Lees, M., "Discrete methods for nonlinear two-point boundary value problems." In Numerical Solution of Partial Differential Equations (editor, J. H. Bramble, Academic Press, New York), pp. 59-72 (1966).

30. Lentini, Marianela, "Correcciones diferidas para problemas de contorno en sistemas de ecuaciones diferenciales ordinarias de primer orden." Trabajo especial de Grado, Depto. de Comp. Fac. Ciencias , Univ. Central de Venezuela (1972).

31. Lindberg, B., "A simple interpolation algorithm for improvement of the numerical solution of a differential equation." SIAM J. Nurner. Anal. 9, pp. 662-668(1972).

32. Osborne, M. R., "On shooting methods for boundary value problems." J. Math. Anal. App. 27, pp. 417-433 (1969).

33. Parter, S. and V. Pereyra, "Nonlinear two-point boundary value problems with multiple solutions." BIT 11, pp. 53-83 (1971).

34. Pereyra, V., "The difference correction method for nonlinear two-point boundary value problems." Stanford University, CS18, Stanford, California (1965).

35. _____, "Accelerating the convergence of discretization algorithms ," SIAM J. Numer. Anal. 4, pp. 508-533 (1967a).

36. _____, "Iterated deferred corrections for nonlinear operator equations." Numer. Math. 10, pp. 316-323 (1967b).

37. _____, "Highly accurate discrete methods for nonlinear problems." MRC Techn. Rep. 728, Univ. of Wisconsin, Madison (1967c).

38. _____, "Iterated deferred corrections for nonlinear boundary value problems." Numer. Math. 11, pp. 111-125(1968).

39. _____, "Highly accurate numerical solution of casilinear elliptic boundary value problems in n dimensions." Math. Comp. 24, pp. 771-783 (1970).

40. Pereyra, V. and G. Scherer, 'Efficient computer manipulation of tensor products with applications to multidimensional approximation." Accepted for publication in Math. Comp. (1973).

41. Richardson, L. F., "The approximate arithmetical solution by finite differences of physical problems involving differential equations." Phil. Trans. Roy. Soc. London 210, pp. 307-357 (1910).

42. Romberg, W., "Vereinfachte numerische integration," Det. Kong. Norske Vidensk. Selskab Forh. 28, 7 Trondheim (1955).

43. Schultz, M. H., Spline Analysis. Prentice Hall, New York (1973).

44. Stetter, H., "Asymptotic expansions for the error of' discretization algorithms for nonlinear functional equations." Numer. Math. 7, pp. 18-31 (1965).

45. Starius, G., "An investigation of' the solution to a compact difference scheme for certain elliptic boundary-value problems with respect to smoothness." Report No. 43, Dep. of Comp. Sci., Uppsala University, Sweden (1972).

46. Varga, R., *'Accurate numerical methods for nonlinear boundary value problems." Manuscript (circa 1968).

47. Wasserstrom, E., "Numerical solutions by the continuation method." SIAM Rev. 15, pp. 89-119(1973).

48. Widlund, O., "Some recent applications of asymptotic error expansions to finite-difference schemes." Proc. Roy. Soc. London A323, PP. 167-177 (1971).