AN ADAPTIVE FINITE DIFFERENCE SOLVER FOR NONLINEAR TWO POINT BOUNDARY PROBLEMS WITH MILD BOUNDARY LAYERS

by

M. Lentini

V. Per eyra

STAN-CS-75-530
NOVEMBER 1975

COMPUTER SC IENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERS ITY

AN ADAPTIVE FINITE DIFFERENCE SOLVER FOR NONLINEAR TWO POINT BOUNDARY PROBLEMS WITH MILD BOUNDARY LAYERS**

by

M. Lentini* and V. Pereyra+

* Department of Applied Mathematics, Cal-tech, Pasadena, California.

\+ Department of Mathematics, Universfty of Southern California, Los Angeles.

** Also issued as LBL Report 4226 at the Lawrence Berkeley Laboratory, University of California.

"An Adaptive Finite Difference Solver for Nonlinear Two Point Boundary Problems with Mild Boundary Layers," M. Lentini and V. Pereyra

ABSTRACT. A variable order variable step finite difference algorithm for approximately solving m-dimensional systems of the form

$$y' = f(t,y), \qquad t \in [a,b]$$

subject to the nonlinear boundary conditions

$$g(y(a),y(b)) = 0$$

is presented.

A program, PASVAR, implementing these ideas have been written and the results on several test runs are presented together with comparisons with other methods. The main feautres of the new procedure are: a> Its ability to produce very precise global error estimates, which in turn allow a very fine control between desired tolerance and actual output precision;

b) Non-uniform meshes allow an economical and accurate treatment of boundary layers and other sharp changes in the solutions.

c) The combination of automatic variable order (via deferred corrections) and automatic (adaptive) mesh selection produces, as in the case of initial value problem solvers, a versatile, robust, and efficient algorithm.

AN ADAPTIVE FINITE DIFFERENCE SOLVER FOR NONLINEAR TWO POINT BOUNDARY PROBLEMS WITH MILD BOUNDARY LAYERS

M. Lentini* and V. Pereyra+

1. Introduction

We are interested in developing usable software for two-point boundary problems for m-dimensional systems of the form

$$\begin{aligned} Y' &= f(t,y), \quad t \in [a,b] \\ g(y(a),y(b)) &= 0 . \end{aligned} \tag{1.1}$$

In [8,9] we have already presented a finite difference algorithm (SYSSOL), based on deferred corrections, which has variable order capabilities. SYSSOL uses only uniform meshes, which can be refined automatically in order to reduce the maximum norm of the (estimated) global error on the current mesh below a requested tolerance.

SYSSOL behaves quite adequately for many problems (see [8,9]), but becomes inefficient or does not work at all as soon as the

---

(*) Department of Applied Math., Caltech, Pasadena, California.

(+) Department of Mathematics, Univ. of Southern California, Los Angeles.

The work of M. Lentini was partly supported by Conicit, and that of V. Pereyra by AEC, AT(04-3)-326 PA#30, while visiting Stanford University during the summer of 1974, and NSF Grant MPS74-13332 at USC. Both authors are on leave of absence from Universidad Central de Venezuela, Caracas.

solution to the problem or some of its derivatives have sharp gradients. Unfortunately, this type of phenomenon is frequently found in the applications.

In [9] we described the deferred correction algorithm for general nonuniform meshes, even allowing for multipoint boundary conditions and data with jump discontinuities.

In this paper we describe an implementation of an algorithm (PASVAR) for approximately solving (1.1) which is based on the results of [9]. The main new features in PASVAR consist of an automatic procedure for choosing nonuniform meshes, and various modifications in the general strategy of the method. Since in [9] and other earlier work we have described the necessary theoretical results and implementation details, we shall concentrate in this 'paper on the new features mentioned above, giving only the minimum general information necessary to make it readable. This basic groundwork will be found in Section 2, while Section 3 will be devoted to the mesh placement problem. Some theory justifying our mesh placement procedure has been published elsewhere [13]. In Section 4 we discuss the practical aspects of the mesh placement algorithm, which is based on the idea of equidistributing the norm of the local truncation error.

Section 5 is devoted to an operation count and storage requirements. In Section 6 we present numerical results on various problems with the type of difficulties mentioned earlier, i.e. boundary layers, steep spikes, and so on. We compare PASVAR with various other programs available : (a) SYSSOL, our uniform step deferred correction solver; (b) RICHAR, a Richardson extrapolation procedure developed by Hilda Lopez and Luis Ruiz [**10**], using some of the basic components of

SYSSOL; (c) a multiple shooting algorithm due to Bulirsch, Stoer, and Deuflhard [2]; (d) IDCBVP and PREV5, two deferred correction codes for scalar second order equations [12,25]; and (e) SUPORT, a linear systems solver based on superposition and orthogonalization [26].

The thickness of the boundary layers that can be resolved with PASVAR depends, as can be expected, on the maximum number of grid points that can be used. Thus the "mild" in the title stands for the fact that we have limited, for storage reasons, that maximum number of grid points in our program to 650/m, where m is the dimensionality of the system being solved.

We see that PASVAR performs efficiently and reliably in all the problems considered, within the limitations imposed by the maximum number of grid points allowed. That limitation is computer dependent.

We emphasize that all the finite difference codes presented here have provisions for estimating the global error of the computed solution, and that in all the problems run this estimate has given either the true error with at least one significant figure, or have been off for less than an order of magnitude. This is in sharp contrast with the techniques based on initial value problem solvers, since even the state of the art codes have no previsions to control the global error of the entire approximate trajectory. Of course, this additional, and we think, extremely valuable information, costs something in terms of computer time, but this cost is amply justified by the added reliability in the numerical results and the excellent correspondence between requested tolerance (TOL) and actual global error in the computer solution.

2. Basic results and notation

Given the m-vector functions $y(t)$, $f(t,y)$, $g(\alpha,\beta)$, we consider the problem of solving approximately

$$y'(t) = f(t,y(t)), \quad t\in[a,b] \quad ,$$
$$g(y(a),y(b)) = 0 \quad . \tag{2.1}$$

We assume that problem (2.1) has an isolated solution $y^*(t)$ (see [4 1). We assume also that f is smooth, so that all involved derivatives of $y^*(t)$ exist. Piecewise smooth data and multipoint boundary conditions can also be treated with slightly more work (see [4, 9]).

Let $l\text{-}r = \{ t_1,\ldots,t_{N+1} \}$ be a general partition of the interval [a,b 1 satisfying:

$$a = t_1 < t_2 < \ldots < t_{N+1} = b$$
$$h_i = t_{i+1}-t_i \; ; \; h = \max_i h_i \; ; \; \hat{h} = \min_i h_i \; ; \tag{2.2}$$
$$h / \hat{h} \leq K \quad ,$$

with K a given positive constant. Condition (2.2) implies:

$$\frac{b-a}{N} \leq h \leq \frac{K(b-a)}{N} \quad , \tag{2.3}$$

and we can use h and l/N interchangeably as equivalent asymptotic scales.

The basic finite difference approximation considered is the trapezoidal rule:

$$\Phi_\pi(u)_i \equiv h_i^{-1}(u_{i+1} - u_i) - \tfrac{1}{2}[f(t_{i+1},u_{i+1}) + f(t_i,u_i)] = 0, \; i=1,\ldots,N$$
$$g(u_1 , u_{N+1}) = 0 \quad . \tag{2.4}$$

Keller studies also the centered Euler scheme:

(2.5) $h_i^{-1}(u_{i+1}-u_i) = f(t_i + \frac{1}{2}h_i\ ,\ 3(u_{i+1}+u_i))\ ,$

which has properties similar to (2.4) and it is easier to use in the case of piecewise continuous data. However, it is considerably more difficult to perform deferred corrections with it, because of the presence of a discretization inside a nonlinear function, which forces partial derivatives of f with respect to u in the expansion for the local truncation error. That is the reason for our choice of the scheme (2.4).

As usual, the local truncation error is defined as what is left when one applies (2.4) to the discretization of the exact solution to the problem. By Taylor's expansion we get:

$$(2.6\quad \tau_\pi(y_i^*) = \sum_{\nu=1}^{L} \frac{-\nu}{2^{2\nu-1}(2\nu+1)}\ f_{i+\frac{1}{2}}^{(2\nu)}\ \frac{h_i^{2\nu}}{(2\nu)!} + O(h^{2L+2}), i=1,\ldots,N\ ,$$

where

$$f_{i+\frac{1}{2}}^{(2\nu)} = \frac{d^{2\nu}}{dt^{2\nu}}\ f(t,y^*(t))\ \Big|_{t=t_i+h_i/2}\ .$$

We shorten (2.6) for further reference to

$$(2.6')\ \tau_\pi(y_i^*) = \sum_{\nu=1}^{L} T_\nu(t_i)\ h_i^{2\nu} + O(h^{2L+2}).$$

Let $\tau_{\pi,k}$ be the mesh function obtained by adding up the first k terms in the asymptotic expansion (2.6), and let $S_\pi^{(k)}(y^*)$ be an $O(h^{2k+2})$ approximation to $\tau_{\pi,k}$. It is well known [9] that if $u^{(k-1)}$ is an $O(h^{2k})$ discrete approximation to $y^*(t)$ on π, and if $(u^{(k-1)}-y^*)$ has an asymptotic expansion in even powers of h, then $S_\pi^{(k)}(u^{(k-1)})$ is an $O(h^{2k+2})$ approximation to $\tau_{\pi,k}$. The operators

$S_\pi^{(k)}$ can be readily constructed via numerical differentiation, as explained in [9], and they are the basis for the deferred correction algorithm. They are also used in the dynamical monitoring of the global error $e^{(k)} = (u^{(k)} - y^*)$. Notice also the modification introduced in [23] which eliminates some earlier theoretical difficulties.

We hope it will be clear from the context that we are speaking of vector mesh functions on π , i.e. that an expression such as the one above means:

$$e_i^{(k)}(t_j) = u_i^{(k)}(t_j) - y_i^*(t_j) \ , \quad t_j \in \pi \ , \ i=1,\ldots,m \ .$$

Another important fact we shall need later is that the method is stable in the infinite norm $\| \ . \ \|$, i.e.

$$\| e^{(k)} \| \leq c \| \tau_\pi^{(k)} \| \ , \tag{2.7}$$

where the constant c is independent of the mesh π . The mesh function $\tau_\pi^{(k)}$ represents the local truncation error after the kth correction has been performed.

We recall now the deferred correction algorithm. Letting $S_\pi^{(o)}(u^{(-1)}) \equiv 0$, solve successively for k=0,1,2,...

$$\begin{aligned} \Phi_{l\text{-}f}(u) &= S_{l\text{-}r}^{(k)}(u^{(k-1)}) \ , \\ g(u_1 \ , \ u_{N+1}) &= 0 \ . \end{aligned} \tag{2.8}$$

We call $u^{(k)}$ to the solution of(2.8) (closest to $y^*(t)$). The main features of the deferred correction procedure are:

(a) Solutions of increased accuracy are obtained on the same mesh (compare with the Richardson extrapolation procedure);

(b) The same system of equations is solved all the time (with different right hand sides).

Under certain conditions, the successively corrected solutions will satisfy on the mesh :

$$\| e^{(k)} \| = \| u^{(k)} - y^* \| = O(h^{2k+2}) .$$

An asymptotic estimate for $e^{(k)}$ can be found by solving for Δ the variational (linear) equation

$$\Phi_\pi'(u^{(k)})\,\Delta = S_\pi^{(k)}(u^{(k-1)}) - S_\pi^{(k+1)}(u^{(k)}) , \tag{2.9}$$

where $\Phi_\pi'(u^0)$ is the Jacobian matrix of Φ_π evaluated at $u^{(k)}$. If $\Delta^{(k)}$ is the solution of this linear problem then:

$$\Delta^{(k)} = e^{(k)} + O(h^{2k+4}) . \tag{2.10}$$

Observe that if (2.8) is being solved by Newton's method, then $\Phi_\pi'(u^{(k)})$ will be available, and since $S_\pi^{(k)}$, $S_\pi^{(k+1)}$ are also available, the cost of the estimate (2.10) is just that of one Newton step, i.e. the solution of a sparse system of linear equations.

For the automatic mesh placement algorithm, we will be interested in having an $O(h^{2k+4})$ estimate of the leading term in $\tau_\pi^{(k)}$. For this purpose, it is necessary to use in $S_\pi^{(k)}$ formulas with a higher order interpolation error than is necessary for the rest of the process. In fact, we will insist that

$$S_\pi^{(k)}(y^*) = \tau_{\pi,k}(y^*) + O(h^{2k+4}) , \tag{2.11}$$

i.e., the numerical differentiation formula will be two orders more precise than before.

Assuming that at the (k-l)th correction we have an expansion for the global error of the form:

(2.12) $$u_i^{(k-1)} - y^*(t_i) = e_k(t_i)\, h^{2k} + e_{k+1}(t_i)\, h^{2k+2} + O(h^{2k+4}),$$

with $e_k(t_i)$ smooth and independent of π, we conclude that

(2.13) $$\begin{aligned} \tau_\pi^{(k)}(y_i^*) &\equiv \Phi_\pi(y_i^*) - S_\pi^{(k)}(u_i^{(k-1)}) \\ &= \tau_{\pi,k+1}(y_i^*) - \tau_{\pi,k}(y_i^*) - S_\pi^{(k)}(e_k(t_i))h^{2k} + O(h^{2k+4}) \\ &= T_{k+1}(t_i)\, h_i^{2k+2} - S_\pi^{(k)}(e_k(t_i))h^{2k} + O(h^{2k+4}) . \end{aligned}$$

Observing that $S_\pi^{(k)}(e_k)$ is itself $O(h^2)$, we see that we have in display the leading term of $\tau_\pi^{(k)}(y^*)$.

<u>Lemma 2.1</u> $\tilde{T}_{k+1} \equiv S_\pi^{(k+1)}(u^{(k)}) - S_\pi^{(k)}(u^{(k-1)})$ <u>is an</u> $O(h^{2k+4})$ <u>approximation to the leading term of</u> $\tau_\pi^{(k)}(y^*)$.

<u>Proof:</u> Because of **(2.11)** and (2.12)(with (k-1) replaced by k) we have

$$\begin{aligned} S_\pi^{(k+1)}(u_i^{(k)}) &= S_\pi^{(k+1)}(y_i^*) + S_\pi^{(k+1)}(e_{k+1}(t_i))\, h^{2k+2} + O(h^{2k+4}) \\ &= \tau_{\pi,k+1}(y_i^*) + O(h^{2k+4}) , \end{aligned}$$

and

$$\begin{aligned} S_\pi^{(k)}(u_i^{(k-1)}) &= S_\pi^{(k)}(y_i^*) + S_\pi^{(k)}(e_k(t_i))h^{2k} + O(h^{2k+4}) \\ &= \tau_{\pi,k}(y_i^*) + S_\pi^{(k)}(e_k(t_i))h^{2k} + O(h^{2k+4}) . \end{aligned}$$

In this last computation we have made use of two terms of the expansion (2.12) in order to obtain the $O(h^{2k+4})$ term.

Subtracting these two expressions and comparing with (2.13) the result follows. II

3. The mesh placement algorithm

We have seen that at the kth step of the deferred correction algorithm, the local truncation error has the form:

$$\tau_\pi^{(k)}(y_i^*) = h_i^{2k+2}\,\hat{T}_{k+1}(t_i) + O(h^{2k+4}), \tag{3.1}$$

where the function $\hat{T}_{k+1}(t)$ does not depend upon the net π. Furthermore the leading term of (3.1) can be estimated to order h^2 by $S_\pi^{(k+1)}(u^{(k)}) - S_\pi^{(k)}(u^{(k-1)})$.

We are interested in choosing a mesh so that the first term of the local truncation error is nearly constant in norm on this mesh. Since we have a limitation on the ratio of the largest to the smallest mesh size (see (2.2)), we have to take into account the possibility that $T_{k+1}(t)$ be accidentally very small at some grid point. For this purpose, and assuming that $\sup_{t\in[a,b]} \| \hat{T}_{k+1}(t) \| = \hat{M}$ we define the function

$$G(t) = \max\left(\|\hat{T}_{k+1}(t)\|, \lambda\right)$$

where $\lambda = \hat{M}/K^{1/\sigma}$ (K defined in (2.2)), $\sigma = 1/(2k+2)$.

We shall call a mesh π (asymptotically) equidistributing iff

$$h_i^{2k+2}\,\|G(t)\|_{(i)} \equiv h_i^{2k+2} \sup_{t\in[t_i,t_{i+1})} \|G(t)\| \equiv E.(1+O(h)), \tag{3.2}$$

where E is a positive constant called the level of equidistribution. The norm $\|.\|$ is the ∞-vector norm. In [13] the properties of equidistributing meshes are studied in detail and more general L_p norms are also considered.

For an equidistributing mesh we then have the relationship

$$\int_a^b G(t)^\sigma \, dt \approx \sum_{i=1}^{N} h_i \, \| U(t) \|^\sigma_{(\tilde{I})} \quad N \cdot E^\sigma \,, \tag{3.3}$$

Thus the level E corresponding to an equidistributing net with N points is approximately equal to

$$E \approx N^{-1/\sigma} \| G \|_\sigma \,. \tag{3.4}$$

Observe that $\| G \|_\sigma \equiv \left(\int_a^b G(t)^\sigma \, dt \right)^{1/\sigma}$ is not a norm since $\sigma < 1$, and also that (3.4) is mesh independent.

We see then that for an equidistributing mesh, the level E itself is an asymptotic bound for the infinity norm of the local truncation error:

$$\max_{i=1,\ldots,N} \| \tau_\pi^{(k)}(y_i) \|_\infty = E(1 + O(h)). \tag{3.5}$$

By using (3.4) we can predict approximately how many points will be necessary to achieve a prescribed tolerance $\tilde{\varepsilon}$. In fact

$$N \geq \left(\| G \|_\sigma / \tilde{\varepsilon} \right)^\sigma . \tag{3.6}$$

Lemma 3.1 If the mesh π is such that

$$\int_{x_1}^{x_{i+1}} G(t)^\sigma \, dt = \tilde{E} \, (1 + O(h))$$

(i.e. $\int_6^b G(t)^\sigma \, dt$ is asymptotically equidistributed) then π-τ is a.e. with $E = \tilde{E}^{1/\sigma} = \| G \|_\sigma / N^{2k+2}$, and hence

$$\| \tau_\pi^{(k)}(y) \| \leq N^{-(2k+2)} \| G \|_\sigma (1+O(h)) \,. \tag{3.7}$$

Proof: The proof is entirely similar to that of Lemma 3.1 of [13] •

Since $\| \cdot \|_\sigma$ is not a norm it is convenient to have an estimate in $\| \cdot \|_\infty$. To obtain that estimate we need the following Lemma.

Lemma 3.2 **Let** $p \geq 1$, let $\varphi(x)$ be a scalar L_p function, and let $0 < \sigma < p$. Let $M_L = \{ x: |\varphi(x)| \geq L \}$ with $L > 0$ chosen so that

$$\int_{M_L} |\varphi(x)|^\sigma \, dx = \tfrac{1}{2} \int_a^b |\varphi(x)|^\sigma \, dx .$$

Then

$$\| \varphi \|_\sigma \leq 2^{1/\sigma} \, [\mu(M_L)]^{\frac{p-\sigma}{\sigma p}} \, \| \varphi \|_p \tag{3.8}$$

where $\mu(M_L)$ is the measure of the set M_L.

Proof: See [13] . ||

For $p = \infty$, (3.8) simply becomes

$$\| \varphi \|_\sigma \leq [2\mu(M_L)]^{1/\sigma} \, \| \varphi \|_\infty . \tag{3.8'}$$

If we combine **(2.7)** with (3.5) and (3.8'), we obtain the following

Theorem 3.3 Let the mesh π have $N+1$ points and be equidistributing for the k th step of the deferred correction algorithm. Then the global truncation error satisfies

$$\| e^{(k)} \| = c \left(\frac{2\mu(M_L)}{N} \right)^{2k+2} \| G_{k+1} \|_\infty \, (1 + O(h)) . \tag{3.9}$$

with ML defined as in Lemma 3.2 for the function

$$\varphi(t) = \| G(t) \|_\infty .$$

Proof: From (2.7) and (3.5) we obtain

$$\| e^{(k)} \| \leq c\, N^{-(2k+2)} \| G \|_{\sigma} (1 + O(h)) ,$$

and applying (3.8′) we get (3.9) . ||

Observe that $\mu(M_L)$ will be small if G(t) has sharp peaks. We give now a simple example to see how this bound compares with the standard one for uniform meshes in a boundary layer model problem. The essential difference between the two bounds is the appearance of the quantity $[\, 2\mu(M_L)\,]^{2k+2}$ when the mesh is equidistributing.

Example.

Consider the first order scalar equation

$$\delta y' = -y, \quad y(0) = 1, \quad t \in [0,1] ,$$

where δ is a small positive constant. This problem has been analyzed in detail in [5] . Its solution is simply $y(t) = e^{-t/\delta}$, and there is a boundary layer of width δ at t=0 . The successive derivatives of y(t) are

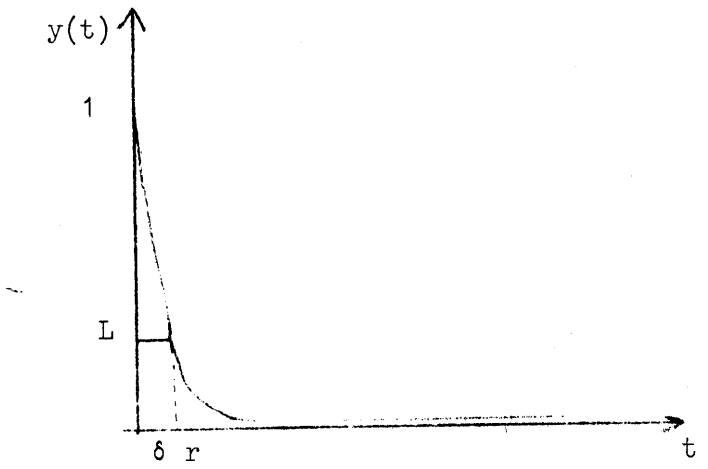
$$\frac{d^s y(t)}{dt^s} = (-1)^s \frac{e^{-t/\delta}}{\delta^s} .$$

It is easy to see that if one applies method (2.4) to this problem, then the stability constant c is O(1) for $\delta \to 0$. Also

$$\hat{T}_{k+1}(t) = c_k \frac{d^{2k+2} y(t)}{dt^{2k+2}} .$$

Thus $\| G \|_{\infty} = c' \delta^{-(2k+2)}$, and from a uniform mesh estimate we deduce that $N = \delta^{-1}$ points will be necessary to get an O(1) accuracy. A simple calculation shows that in this case $\mu(M_L) = c_1 \delta$, with c_1 a small constant, and we see from (3.9) that for an equidistributing mesh an $O(N^{-(2k+2)})$ error bound holds for any N, and the effect of

the boundary layer is completely neutralized by the equidistribution of the ∞ - norm of the local truncation error. Observe also that equidistributing any other smaller derivative of the solution (as in Pearson [**11**]) will not have this effect.



For k=0, and a local truncation error level of 0.01, the mesh-step function must satisfy

$$h^2(t)\,\delta^{-2}\,e^{-t/\delta} = \mathbf{0.01}\ ,$$

or

$$h(t) = 0.1\,\delta\ \mathbf{e}^{t/2\delta}\ .$$

The total number of points is approximately given by

$$N = \sum_{i=1}^{N} 1 = \sum_{\mathbf{i=1}}^{N} h_i\ \frac{1}{h_i} \approx \int_0^1 \frac{1}{h(t)}\ dt \text{"} 20,$$

and the number of points in $[\,0,\delta\,]$ will be

$$N_{[\,0,\delta\,]} \approx \int_0^{\delta} \frac{1}{h(t)}\ dt \approx 13\ .$$

We see then that for any δ more than half of the grid points will be concentrated on the boundary layer, as one expects. Of course, $N = 20$ is the optimal number of points for that level of error, but we have also to enforce the condition $h/\hat{h} \leq K$, with a moderate K which may mean increasing somewhat the density of the mesh outside the boundary layer. Still this will require far less points than the $(10\delta)^{-1}$ points required by a uniform mesh algorithm to give the same order of accuracy.

More general results of the type described in this Section, detailed proofs, and references to related work can be found in [13].

4. Practical mesh placement algorithm

In introducing the concept of asymptotically equidistributing meshes we have taken a step towards the practical implementation of the ideas in Section 3.

As a matter of fact, we won't strive to make our computed meshes even approximately equidistributing, but rather we shall use a somewhat more lax criterion. The first ground rule in our iterative procedure to obtain a grid π is that we will only add points, and once a point is in the mesh it will never be touched again.

In other algorithms proposed [11, 17, 21], either a fixed number of points is moved around, or points are added and removed in order to satisfy some equidistribution condition. In our experience those procedures have a tendency to be more unstable and to produce rougher meshes than can be tolerated. There are, of course, ways of improving that situation, like smoothing, but that only complicates the algorithm unduly. On the other hand, the closer the mesh is to an equidistributing one, the fewer number of points it will have for a given tolerance; so that fact and the cost of producing such a mesh must be carefully balanced. Also, from our example above, it is seen that, for a given level of truncation error, there is an order of the method which minimizes the number of points required for a given problem. Of course, one should take into consideration the amount of work for each order when drawing true optimality results. For the time being, these considerations are far too complicated to be taken strictly into account in our algorithm, but they provide guidelines for useful heuristics.

Our procedure starts with a given mesh $\pi^{(0)}$ with N_0+1 points. If no *a priori* information is available about the problem difficulties then $\pi^{(0)}$ will usually be a uniform mesh with step size $h^{(0)}$.

Obviously, because of computer storage restrictions, one will also have a maximum number of mesh points that can be considered in any given mesh, say NMAX . In our program we have chosen NMAX = 650/m .

On $\pi^{(o)}$ we obtain an approximate $O(h^2)$ solution $u^{(o)}_{\pi^{(o)}}$ by solving $\Phi_{\pi^{(o)}}(u) = 0$, $g(u_1, u_{N_o+1}) = o$ (see (2.8)) .

Then we compute $S^{(1)}_{\pi^{(o)}}(u^{(o)}_{\pi^{(o)}})$, which is **an estimate** for the leading term in the local truncation error $\tau^{(o)}_{\pi^{(o)}}$. The infinity norm is used throughout. If we want $\| \tau^{(k)}_{\pi}(y) \|_\infty \approx \tilde{\epsilon}$, then from Lemma 3.1 and (3.4) we obtain that

(4.1) $\tilde{E} = \tilde{\epsilon}^{\sigma}$.

The initial tolerance requested, $\tilde{\epsilon}^{(o)}$, is up to a certain extent arbitrary, but nevertheless it should be chosen judiciously. As $\tilde{\epsilon}^{(o)}$ becomes smaller, more points will be added to the mesh at the beginning, which may be unwise. Let TEM= $\| u^{(o)} \|$. We put

(4.2) $\tilde{\epsilon}^{(o)} = \max$ (BMA * TEM , TOL) ,

where BMA is a parameter used to control the size of $\tilde{\epsilon}^{(o)}$. The maximum norm of the approximate solution, TEM, is what connects the level $\tilde{\epsilon}^{(o)}$ with the particular problem being solved. Essentially what we are saying is that we would like to have an equidistributing mesh with sufficiently many points as to achieve, at the start, a relative precision BMA with the $O(h^2)$ method. BMA should not be too small, since at the early stages of the game the information available $(u^{(o)})$ will tend to be more unreliable, especially for problems with difficulties.

We call (see Section 3 for the definition of λ):

$$EJ(I) = h_i \max\left(\| \hat{T}_{k+1}(t_i) \|, \lambda \right)^{\sigma} \cong$$

$$h_i \max\left(\| \tilde{T}_{k+1}(t_i) \|, \lambda \right)^{\sigma},$$

and

$$UUN = \sum_{I=1}^{N} EJ(I) \approx \int_a^b G(t)^{\sigma} dt \ .$$

The equidistributing procedure adds points, according to the following rule. "In the present interval (x_I, x_{I+1}) add $IQJ(I) - 1$ uniformly distributed points, where

(4.3) $$IQJ(I) \triangleq \lfloor EJ(I) / \tilde{E} \rfloor ,$$

and $\lfloor \ \rfloor$ stands for "integer part of ".

Thus the total number of points added in each sweep is

$$IQ = \sum_{J=1}^{N} IQJ(I) \ ,$$ where $(N+1)$ is the number of points in the mesh being modified.

These new points are actually added into the mesh if the following conditions are satisfied

(4.4) $$0.04\ N \leq IQ \leq \min(NMAX-N, 70) \ .$$

If $IQ < .04N$, and the mesh has been modified during the present process then the equidistribution terminates.

The condition on the right of (4.4) prevents too many points being added in any given sweep.

We observe that with the notation above

$$UUN / \tilde{E} \approx N + IQ + 1 \ .$$

If IQ violates one of conditions (4.4) and this is not terminal, we can attempt to find the "right level" E* which will bring in a

preset number of points IQ* , by putting

$$(4.5) \qquad E^* = E \times \frac{N + IQ + 1}{N + IQ^* + 1} ,$$

and going again through the mesh in order to obtain a vector IQJ(I) for this new level. We use in our program

$$N + IQ^* + 1 = \min (ALG^*N, NMAX)$$

with ALG = 1.1(0.1) **1.4** . If the case IQ < .04N is not terminal, then we redefine the level as in (4.5) with ALG = 1.4 . If that level still does not bring enough new points into the mesh, then we decrease the correction index k by 1 , until either the mesh is modified or k = 0 . In this latter case a complete bisection of the mesh is requested (if possible). If the right hand condition (4.4) is violated then we define a new level, also with ALG = 1.4 , but now we allow AU to decrease down to 1.1 in steps of 0.1 . This process is of course stopped whenever an allowable number of new points is produced.

This series of tests and modifications are intermixed in a somewhat complex logical structure which is better understood by looking at a flowchart or the actual computer program. Here we have only tried to list some of the main features of the algorithm.

In particular, indefinite cycling is precluded by various controls so that the mesh refinement process always terminates, though not necessarily with an equidistributed mesh.

We have insisted in not removing points from the mesh since this provides an easy way of insuring that the condition $h/\hat{h} \leq K$ is fulfilled with a reasonable K , and also produces smoother meshes.

As we shall see in the numerical examples, in problems with transition regions as thin as 10^{-3} (on an interval of size 0.2), the algorithm has produced solutions accurate to 10^{-8} with a mesh in which $K \leq 50$.

5. Operation count and storage requirements

In this Section we shall make an operation count for algorithm PASVAR. There are essentially three large modules in PASVAR, and two main loops: the deferred correction iteration and the Newton solver.

(a) The linear equation solver (SYSLIN).

SYSLIN is an implementation of the algorithm of Section 3 of [9]. SYSLIN is called at each Newton step, and at the end of each correction, in order to estimate the global error.

The relevant parameters for SYSLIN are: m the number of differential equations in the system, and N the number of mesh points. The systems solved by SYSLIN are then of size $m(N+1) \times m(N+1)$. They are also sparse and highly structured. In fact the coefficient matrices involved have the form

$$(5.1)\quad \mathcal{A} = \begin{bmatrix} G_1 & 0 & & \cdot & G_{N+1} \\ S_1 & R_1 & 0 & \cdot & \cdot \\ 0 & S_2 & R_2 & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ 0 & \cdot & \cdot & S_N & R_N \end{bmatrix} = \begin{matrix} m \\ mN \end{matrix} \overbrace{\begin{bmatrix} A & B \\ C & D \end{bmatrix}}^{\begin{matrix} m & mN \end{matrix}}$$

That is, they are block bidiagonal, with the exception of the first block row. The blocks are of size $m \times m$. The system of equations have the form

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_0 \\ \hat{x} \end{bmatrix} = \begin{bmatrix} b_0 \\ \hat{b} \end{bmatrix}$$

We solve the superblock 2 x 2 system by Gaussian elimination, and that implies solving subsystems with the lower block bidiagonal matrix D . Putting $\tilde{C} = [\, C \mid \hat{b} \,]$, $V = [\, V \mid w \,]$ we have

(5.2a) $x_0 = (A - BV)^{-1} (b_o - Bw)$,

(5.2b) $\hat{x} = D^{-1} (\hat{b} - Cx_0)$,

where $V = D^{-1} C$, $w = D^{-1} \hat{b}$ are obtained by solving the system

(5.3) $D \tilde{V} = \tilde{C}$.

This is done by means of the recursion

(5.4) $\tilde{V}_j = R_j^{-1} (\tilde{C}_j - S_j \tilde{V}_{j-1})$, $j = 1,\ldots,N$.

Forming the expression in each parenthesis of (5.4) takes $m^3P + m^2S$ operations , where P stands for multiplications or divisions, and S stands for additions or subtractions. Solving one matrix system (5.4) takes $\frac{4}{3} m^3$ (P + S) operations, and thus we have a total of

$N (\frac{7}{3} m^3P + (\frac{4}{3} m^3 + m^2) S)$ operations for the recursion (5.4).

The calculation of (5.2.a) and (5.2.b) takes $(\frac{4}{3} m^3 + Nm^2) P + (\frac{4}{3} m^3 + Nm^2 + Nm) S$, and the total number of operations for SYSLIN is (most significant terms only)

(5.5) $\frac{Nm^3}{3} (7P + 4 S)$

(cf. [7] also).

This is about twice the number of operations obtained by Varah [16] for the case of uncoupled boundary conditions. In that case, by arranging the equations properly one obtains a band matrix, or a block tridiagonal one, depending how he looks at it.

(b) Calculation of the correction vectors $S_{\pi}^{(k)}(u^{(k-1)})$.

This calculation is performed in Subroutine U2DCGS . The relevant parameters here are : k the correction number, and m, N as before.

For each grid point we have to generate weights for a differentiation formula approximating $\tau_{\pi,k}(y_i^*)$ to order h^{2k+4} . Since the abscissas are not uniformly distributed, and since $\tau_{\pi,k}$ is $O(h^2)$, then $(2k+2)$ ordinates are necessary to produce the required approximation.

The weights are obtained in Subroutine COEGEN; for each grid point the weight generation takes $k^2(4P + 6S)$ operations (see [1]), and thus, forming $S_{\pi}^{(k)}$ costs

(5.6) $Nk((4k + 2m)\ P + (6k + 2m)\ S)$.

(c) The mesh selection procedure is a process taking a small multiple of mN operations. Under certain circumstances it may also require a call to U2DCGS .

The Newton loop. For each correction, a sparse system of mN non-linear equations must be solved. We use a descent Newton iteration with step and angle control to solve those equations. In cases where there are convergence difficulties, an optional automatic continuation procedure is also available (see [9,22]).

Each Newton iteration takes one evaluation of the right hand side $f(t,u)$ (vector mesh function), and one evaluation of its Jacobian matrix. Then, a computation of the residual $\Phi_\pi(u)$ is required (see (2.4)); this is a $(4mN)$ operations process. Finally we have a call to SYSLIN.

If the process is going to converge at all, it usually takes no more than three iterations to achieve $\| \Phi_\pi(u) \| \leq$ EPS . The tolerance EPS varies with the correction order, and with the actual estimated global error, in such a way that the equations are solved to a level compatible with the truncation error. After the first system is solved, and some accuracy has been obtained, the following systems take usually fewer iterations since better initial values are used,

Thus we can reasonably assess the work for a complete Newton process, including one extra iteration for the error estimate, as:

$$(5.7) \qquad \boxed{N \left[\frac{4}{3} m^3 (7P + 4S) + 16m (P+S)_I + 4 (FE + JE) \right]} ,$$

where FE, JE stand for evaluation of f and its Jacobian over the whole mesh.

If the problem is linear, and the system of linear equations is not too ill conditioned, this work estimate should be halved. If the system is ill conditioned, and after passing through SYSLIN the residual has not been diminished sufficiently (it should be zero!), then more "New-ton iterations"will be performed. This process is actually equivalent to iterative refinement, a procedure to improve the precision of numerical solutions to linear systems, and it is automatically built into the program.

The total work for the kth correction is essentially

$$(5.8)\quad N\left\{\left(\frac{28}{3}m^3 + 4k^2 + (2k+16)m\right)P + \left(\frac{16}{3}m^3 + 6k^2 + (2k+16)m\right)S\right\} + 4(FE+JE)\ .$$

There are indications that more sophisticated equation solvers (both linear and nonlinear) can be valuable in difficult problems [4,24], and we are presently working in this direction.

In order to analize the cost of any given actual run, we have to consider the following quantities. $N_0, N_1, \ldots, N_r$: the different number of grid points used; $c_0, c_1, \ldots c_r$ the number of corrections performed with each fixed mesh. Since the amount of work in a correction depends upon its order, we also have to consider as parameters the starting orders $k_0, k_1, \ldots, k_r$. Clearly $k_0 = 0$. From (5.8), and after some simplifications, we obtain the following estimate

$$\begin{aligned}(5.9)\quad &\sum_{j=0}^{r} N_j c_j \left\{\frac{28}{3}m^3 + 16m + 4(k_j + c_j)^2 + 2(k_j + c_j)m\right) P \\ &\quad + \left(\frac{16}{3}m^3 + 16m + 6(k_j + c_j)^2 + 2(k_j + c_j)m\right) S \Big\} \\ &\quad + 4c_j(FE + JE)\ .\end{aligned}$$

Except for small systems $(m \leq 5)$, this estimate can be further simplified to

$$(5.9')\quad \sum_{j=0}^{r} N_j c_j m^3 (9P + 5S) + 4c_j(FE + JE)$$

For a given problem it is impossible to predict the program path, i.e. to determine a priori the 'parameters N_j, c_j, k_j, unless some very strong and unrealistic hypotheses are made.

It is plausible that with the information we have provided here, a more elaborate complexity analysis could be performed. Also, comparisons

of the type carried out by Keller [18] can be performed by making appropriate hypotheses. For instance, assuming that instead of SYSLIN the same linear equations solver as in [18] is used, that the same number of Newton iterations is required, and that the basic mesh need not be changed, then iterated deferred corrections require always less operations and function evaluations (for a given order) than successive Richardson Extrapolations. We feel, however, that these work estimates give only pointers and general indications. A computer test on several actual implementations and on a large, representative set of problems is what is required in order to make more final assessments. One step in this direction--.is furnished by the results of the following Section. See also [10] .

Storage requirements. The storage requirements (most significant terms)for our implementation, depending upon the two problem parameters m,N, are given below. In the case that no dynamical array space allocation is available, those parameters should be replaced by maximal values. We have considered a maximum of 20 deferred corrections, which should be more than sufficient for most problems, but in any case that is not a storage consuming part of the algorithm. The expressions below correspond to number of real words required. The actual storage in bytes will depend upon the kind of computer and precision being used.

PASVAR : Data : $2\,m^2 + (m+1)\,N$

Working area : $4\,m\,N + 2\,N + 170$.

SYSLIN :

Working area : $m^2\,(N + 8)$.

Thus the total storage required is

(5.10) $$\boxed{\text{Storage} = m^2\,(N+10) + (5m+3)\,N + 170}$$

real numbers.

6. Numerical results and comparisons

In this Section we give results for program PASVAR, and compare them with results obtained with other FORTRAN programs:

SYSSOL: the uniform mesh version..of PASVAR [9];

RICHAR: a Richardson extrapolation, finite differences code [10] ;

MUISHO: a multiple shooting code [2] ;

IDCBVP: A deferred correction code for scalar second order equations with no y' present [12];

PREV5 : an improved version of IDCBVP by Daniel and Martin [25];

SUPORT: A linear systems solver based on the Godunov method [26].

In [8] we have anticipated similar results, but the ones here correspond to different versions of the various programs (with the exception of SYSSOL). For instance, RICHAR can now perform extrapolations with any sequence of steps h_0/k_i , i = 0,1,.... We call RICHAR1 to the one using the sequence $k_i = 2^i$, and RICHAR2 to the one using k_i = 1,2,3,4,6,8,12,16...

The results for MULSHO were obtained by MM. Deuflhard, Rentrop and Pesch, under the direction of R. Bulirsch, and we are very grateful to them for their cooperation. Appropriately chosen parameters and shooting points now produce convergence from zero initial values in all cases tested. Also, much improved results in terms of total number of function evaluations are obtained with MULSH02 , in which the integration routine has been replaced by VOAS , an initial value code provided by T. Hull.

The results for SUPORT were obtained by M. Scott and H. Watts, using a Runge-Kutta-Fehlberg integrator for achieving the absolute

error tolerance of 10^{-3} and a variable order Adams integrator for absolute tolerances of 10^{-8} and below. Since SUPORT, as opposite to all our codes, has no way of requesting (and obtaining) a desired accuracy in the computed solution (see [26], Section 12), the results given in Tables 1 and 2 were obtained by running each problem with a large spectrum of input tolerances and selecting those results which satisfied the output tolerances more closely (and with the least work, of course).

The test problems are all small systems, but they show in one way or another troublesome behavior. One exception is Problem 6, which is used as an indicator of how the programs behave when confronted with a smooth problem. All problems and programs were started with 17 points, uniform meshes, and initial values for Y identically zero with the exception of the shooting programs for which we indicate the shooting points in each instance, and of SUPORT which does not require a starting mesh. We have collected all the numerical results in Table 1. In the case of convergence to the desired tolerance we record: EFE = equivalent function evaluations $\equiv F + wJ$, where F is the number of times the right-hand side $f(t,y)$ has been evaluated for one value of t, and J is the number of Jacobian evaluations. The weight w varies from problem to problem and it is indicated in Table 1; in all cases $w \leq 1$, and it reflects the relative cost of evaluating the Jacobian matrix as compared with that of evaluating the vector function f. Otherwise we print the precision reached (if it is close to the one requested), or:

NC = no convergence;

— = results not available.

In our programs we request that the estimated maximum absolute error on the whole grid, and for all components of the solution vector, be less than TOL for successful termination. MUISHO has a relative tolerance parameter available to the user (EPS), and we give its value in the various cases run.

We give computer times (when available) as a matter of reference. The times for SUPORT were obtained at a different installation (same computer but a different compiler). The computer times (in seconds) can be found in Table 2. The high order scalar equations have been treated as first order systems in the standard way. The exact solutions (when available) are given in [8].

Problem 1 [15]

$$y'' = 400(y + \cos^2 \pi t) + 2\pi^2 \cos 2\pi t$$

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

This is a problem which is troublesome for methods based on standard initial value problems techniques. It can also be interpreted as a problem with boundary layers of thickness 1/20 at t = 0, 1. MULSHO used here three equally spaced shooting points, and MULSH02 used five.

Problem 2 Falkner-Skan equation [3].

$$y''' + yy'' + \beta[1 - (y')^2] = 0$$

$$y(0) = y'(0) = 0, \quad y'(\infty) = 1 .$$

As β approaches the value 2, the solutions of the initial value problem associated with this equation become very sensitive with respect to the value of the missing initial condition $y''(0)$. This problem has required continuation in order to provide adequate starting values for the Newton iteration in all the programs with the exception of PASVAR. We have used β in SYSSOL and RICHAR as a natural continuation parameter, performing just one Newton iteration for each of the values $\beta = 0\ (0.2)\ 1.8$, and then completing the process for $\beta = 2$. This is done only once, at the very beginning, on the coarsest mesh and with the basic second order method. Afterwards, the initial values provided are sufficiently accurate to produce convergence without difficulties. All this process is performed automatically, using a continuation option. The results reported below correspond to the full computation for $\beta = 2$ and $\infty \approx 10$.

MULSH0 and MULSH02 used the four shooting points $x_j = 0,1,3,$ and 6.

Problem 3 An artificial boundary layer problem [12]

$$y'' = \frac{-3\epsilon y}{(\epsilon + x^2)^2}$$

$$y(-0.1) = \frac{-0.1}{(\epsilon + 0.01)^{1/2}}, \qquad y(0.1) = -y(-0.1) .$$

For $\epsilon \to 0$, $y(t) \to \text{sign } t$. The problem has a turning point at $t = 0$ of thickness $\epsilon^{1/2}$. The values of ϵ are indicated in parentheses on the heading of the respective columns. In this problem, all programs with the exception of SUPORT used the final values for an ϵ to start the computation for the following smaller ϵ.

MULSHO used 5 equally spaced shooting points (including the origin), and it was successful up to $\epsilon = 10^{-9}$, using 26139 F.E. for that case.

Problem 4 [14]

$$y'' + (3 \text{ cotan } t + \tan t)y' + 0.7y = 0$$

$$y(30^o) = 0 , \quad y(60^o) = 5 .$$

This problem has a sharp spike at approximately $t = 30.65^o$, where $y(30.65^o) \approx 285$, and the high order derivatives are even larger.

The MULSHO codes used the four shooting points $x_j = 30^o$, 31^o, 35^o, 60^o.

Problem 5 [11] Another artificial boundary layer problem.

$$y'' + \epsilon^{-1}y' = 0$$

$$y(-1) = 1, \quad y(1) = 2, \quad \epsilon > 0 .$$

This problem has a boundary layer of thickness ϵ at $t = -1$, where the solution passes from the value one to the value two. The results reported correspond to $\epsilon = .01$.

MULSHO shooting points were x_j = -1, -0.8, -0.5, 1; while MULSHO2 used the sequence x_j = -1, -0.8, -0.5, 0, 0.5, 1.

This problem was also solved successfully with PASVAR for ϵ = 0.001, 0.0001, TOL = 10^{-3}, and for ϵ = 0.001, TOL = $10^{-8,5} \times 10^{-10}$. In this last case PASVAR required 2753 equivalent function evaluations and used 3.75 seconds of computer time on a CDC 6600/6400 machine. The meshes and solutions for large ϵ were used to start the computation for smaller ϵ.

Problem 6[12] An easy problem.

$$y'' = y^3 - \sin t \cdot (1 + \sin^2 t)$$

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

MULSHO and MULSHO2 used the three shooting points x_j = 0, $\pi/2$, π.

problem / code	1	2	$3(10^{-3})$	$3(10^{-6})$	$3(10^{-7})$	4	5	6
weight for Jacobians	0·1	0.75	0.75	0.75	0.75	0.75	0.75	0.5
TOL = 10 -3								
SYSSOL	419	829	--	NC	'NC	NC	NC	229
RICHAR1	531	815	671	NC	NC	NC	1378	227
RICHAR2	--	743	451	NC	NC	NC	1248	--
PASVAR	327	543	1088	7891	9997	1892	1140	195
MULSHO	2061	7657	1232	13076	5892	15815	16363	1866
MULSHO2	1224	1188	912	2631	3508	2700	3960	559
IDCBVP	75	--	398	NC	NC	--	--	115
PREV5	75	--	306	NC	NC	--	--	115
SUPORT	312	--	334	1246	1880	403	802	--
TOL = 10 -8								
SYSSOL	1203	3063	2990	NC	NC	NC	NC	331
RICHAR1	1733	NC	1378	NC	NC	NC	NC	1052
RICHAR2	1008	3135	1248	NC	NC	NC	NC	732
PASVAR	806	1425	2325	12982	14621	7264	2753	297
IDCBVP	385	--	2424	NC	NC	--	--	148
PREV5	354	--	1460	NC	NC	--	--	148
SUPORT	572	--	626	2580	3460	688	3832	--

Table I. Equivalent Function Evaluations: F + w*J

Weights for IDCBVP, PREV5 were w = 0, 1, 0.1 in Problems 1, 3, 6 respectively.

Problem / Code	1	2	3 (10^{-3})	3(10^{-6})	3 (10^{-7})	4	5	6
Limiting precision	2472 10^{-13}	4418 2.4×10^{-12}	--	--	NC	NC	NC	616 10^{-13}
SYSSOL								
PASVAR	2026 1.6×10^{-12}	2439 10^{-14}	3559 7.8×10^{-11}	--	--	9827 1.6×10^{-11}	3212 8.2×10^{-11}	747 10^{-14}
IDCBVP	2006 10^{-13}	--	2866 10^{-13}	NC	NC	--	--	573 10^{-13}
PREV5	1222 10^{-13}	--	1838 10^{-13}	NC	NC	--	--	371 10^{-13}
SUPORT	3.2×10^{-10}	--	3.6×10^{-12}	1.7×10^{-9}	3832 4.2×10^{-8}	10^{-8}	4.8×10^{-10}	--

Table I Cont. Equivalent Function Evaluations: F + w*J

Weights for IDCVBP, PREV5 were w = 0, 1, 0.1 in Problems 1, 3, 6 respectively.

Problem / Code	1	2	$3(10^{-3})$	$3(10^{-6})$	$3(10^{-7})$	4	5	6
TOL = 10-3								
PASVAR	0.57	0.87	1.16	9.90	11.33	2.11	1.13	0.19
IDCBVP	0.02	--	0.03	--	--	--	--	0.01
PREV5	0.02	--	0.04	--	--	--	--	0.02
SUPORT*	0.08	--	0.07	0.23	0.35	0.12	0.15	--
TOL = 10^{-8}								
PASVAR	1.77	2.67	1	19.60	20.42	12.77	3.78	0.34
IDCBVP	0.15	--	0.12	--	--	--	--	0.03
PREV5	0.15	--	0.09	--	--	--	--	0.03
SUPORT*	0.42	--	0.50	2.06	2.81	0.74	2.51	--
Limiting precision								
PASVAR	7.34	8.34	5.23	--	--	19.19	4.62	1.28
IDCBVP	0.83	--	0.43	--	--	--	--	0.12
PREV5	0.72	--	0.41	--	--	--	--	0.12
SUPORT*	1.65	--	0.95	2.70	3.17	50.90		--

Table 2.

CPU times in seconds on CDC 6600/6400 at LBL, University of California Berkeley; RUN76 compiler.

*On CDC 6600 at Sandia Labs., Albuquerque; FUN compiler.

Conclusions

From this limited set of tests we can draw some preliminary conclusions.

Overall, PASVAR is far superior to RICHAR1 and SYSSOL for all accuracies, and this is more marked for higher accuracy. RICHAR2 is competitive for low accuracies in the problems in which it works (c.f. [10] for comparisons on smooth problems). In all fairness, we should use a Richardson extrapolation program with nonuniform mesh capabilities, but this code is still to be developed. It is clear, that whenever applicable, the scalar equations codes are by far the fastest and most efficient.

The multiple shooting code MULSH02 compares well with PASVAR in terms of total number of function evaluations and reliability for most of the problems tested. The main exception is the turning point Problem 3 where MULSH02 obtains the solution with considerably fewer function evaluations than PASVAR . Furthermore, MULSH02 obtains good results for $\epsilon = 10^{-8}, 10^{-9}$, while PASVAR cannot resolve the boundary layer with the allotted maximum number of grid points. However, it is worth mentioning that in Problem 4 MULSH02 takes 40% more computer time than MULSHO , despite the fact that this last program requires almost 6 times more function evaluations to achieve convergence. We should point out also that the multiple shooting codes do not choose the shooting points and various other parameters automatically, and only give final results on the shooting points. Thus, PASVAR requires much less user interaction and foreknowledge, and outputs a much more detailed mesh solution. This detail is automatically more dense in

the regions of rapid variation of any component of the solution vector. It would be also useful to compare the performance of MULSHO2 for higher accuracy. Professor Bulirsch has indicated that a more user oriented version of his program, correcting some of these drawbacks, will be available in the future.

The comparisons with SUPORT show that PASVAR work too hard in solving the turning point problem 3 for all tolerances and the spike problem 4 for TOL = 10^{-3}, 10^{-8}. This indicates that our net selection procedure is too slow for handling this type of quasi-singularities.

The performance of SUPORT is consistently good for low and moderate accuracies, though we have to keep in mind that the user has no way of assuring that he will get that accuracy by specifying an input parameter. We should also keep in mind that, so far, SUPORT only solves linear problems, and that it can take advantage of certain special situations, like homogeneous equations (probs. 3, 4, 5) and zero initial values (Probs. 1, 4). The somewhat disappointing results for high or limiting tolerance seem to stem from the inability of the initial value codes to produce such accuracies. Apparently the boundary value techniques can reach tolerances close to full machine accuracy without excessive degradation.

We are presently working on a new version of PASVAR which among other features has a new system of equations solver (both linear and nonlinear). Preliminary results indicate that this new code will solve problems for which PASVAR fails, and also that it will cut the number of function evaluations and time by half in most cases.

In Table 3 we report some information about the mesh placement and deferred correction procedures on the various problems. We give N_1, the number of times that a mesh refinement was requested. Each one of these refinements requires several mesh modifications. The quantity N_2 is the average number of these modifications. The row $\hat{k}$ gives the higher correction reached, and K is the total number of corrections performed.

We see from these results that the mesh placement routine 'does not wander" since the average number of inner sweeps is never large than 3, which is reached in only one case (Prob. 5, $Tol = 10^{-13}$). On the other hand we see that high order methods really came into play, and although we do not claim that a correction of index $k = 10$ will produce an $O(h^{22})$ accurate solution, it is quite remarkable that such high order corrections do actually produce visible improvements in the computed solution.

Problem	1	2	3(-3)	3(-6)	3(-7)	4	5	6
Tol = 10 -3								
N_1	1	1	2	4	2	3	1	0
N_2	1	1	1.5	1.25	2	2	2	0
$\hat{k}$	3	3	2	5	4	3	2	1
K	4	6	5	11	8	8	3	2
Tol = 10 -8								
N_1	2	2	3	3		6	2	0
N_2	1	2	1.33	1.33	0.5	1.83	1.5	0
$\hat{k}$	6	5	4	8	10	7	6	3
K	9	10	9	13	13	17	8	4
Tol = 10^{-13}								
N_1	2	4	-	-	-	-	3	1
N_2	1	1.75	-	-	-	-	3	1
$\hat{k}$	10	8	-	-	-	-	7	6
K	13	17	-	-	-	-	11	8

TABLE 3

REFERENCES

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

2. R. Bulirsch, J. Stoer, and P. Deuflhard, "Numerical solution of nonlinear two-point boundary value problems I," To be published in Numer. Math., Handbook Series Approximation.

3. T. Cebeci and H. B. Keller, "Shooting and parallel shooting methods for solving the Falkner-Skan boundary-layer equation," J. Comp. Phys. 7, 289-300 (1971).

4. H. B. Keller, "Accurate difference methods for nonlinear two-point boundary value problems," SIAM J. Numer. Anal. 11, 305-320 (1974).

5. H.-O Kreiss, "Difference approximation for singular perturbation problems," Proc. NSF-Symp. on Numerical Solution of Boundary Problems for Ordinary Differential Equations, Academic Press, New York (1975).

6. _____ and N. Nichols, Personal communication (1974).

7. M. Lentini, "Correcciones diferidas para problemas de contorno en sistemas de ecuaciones diferenciales ordinarias de primer orden," Pub. 73-04, Depto. de Comp., Fac. Ciencias, Univ. Central de Venezuela, Caracas (1973).

8. _____ and V. Pereyra, "Boundary problem solvers for first order systems based on deferred corrections," Proc. NSF Symp. on Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, 293-316, Academic Press, New York (1975).

9. _____, "A variable order finite difference method for nonlinear multipoint boundary value problems," Math. Comp. 28, 981-1004 (1974).

10. H. López and L. Ruiz, "Extrapolaciones sucesivas para problemas de contorno en sistemas no lineales de ecuaciones diferenciales ordinarias," Tesis de Grado, Univ. Central de Venezuela (1974).

11. Carl E. Pearson, "On a differential equation of boundary layer type," J. Math. Phys. 47, 134-154 (1968).

12. V. Pereyra, "High order finite difference solution of differential equations," Stanford University Computer Science Dept. STAN-CS-73-348 (1973).

13. _____ and G. Sewell, "Mesh selection for discrete solution of boundary problems in ordinary differential equations," Numer. Math. 23, 261-268 (1975).

14. R. D. Russell and L. F. Shampine, "A collocation method for boundary value problems," Numer. Math. 19, I-28 (1972).

15. J. Stoer and R. Bulirsch, "Einführung in die Numerische Mathematik II Heidelberg Taschenbücher No. 114, Springer-Verlag, Berlin (1973).

16. J. M. Varah, "On the solution of block tridiagonal systems arising from certain finite difference equations," Math. Comp. 26, 859-868 (1972).

17. C. de Boor, "Good approximation by splines with variable knots II," Lecture Notes in Math. No. 363, 12-20, Springer-Verlag, Berlin (1973).

18. H. B. Keller, "Numerical solution of boundary value problems for ordinary differential equations: survey and some recent results on difference methods," Proc. NSF Symp. on Numerical Solution of Boundary Problems for Ordinary Differential Equations, 27-88, Academic Press, New York (1975).

19. H. G. Burchard "Splines (with optimal knots) are better," App. Analysis 3, 309-319 (1974).

20. J. F. Holt, "Numerical solution of nonlinear two-point boundary problems by finite differences using Newton's method," Aerospace Rep. No. TR-0158(3307-02)-10 (1968).

21. D. 0. Gough, E. A. Spiegel, and J. Toomre, "Highly stretched meshes as functionals of solutions," Proc. 4th Int. Conf. Num. Methods Fluid Dyn., Boulder, Colorado (1974).

22. J. H. Avila, "The feasibility of continuation method for nonlinear equations," SIAM J. Numer. Anal. 11, 102422 (1974).

23. V. Pereyra and 0. Widlund," A family. of elliptic difference schemes suggested by H.-O. Kreiss," To appear.

24. P. Deuflhard, "A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with applications to multiple shooting," Numer. Math. 22, 289-315 (1974).

25. J. W. Daniel and A. J. Martin, "Deferred corrections for the fourth-order difference method for second-order two-point boundary-value problems," Center for Numer. Anal. Report, Univ. of Texas, Austin. To appear (1975).

26. M. R. Scott and H. A. Watts, "SUPORT--A computer code for two-point boundary-value problems via orthonormalization," SAND 75-0198, Sandia Labs., Albuquerque, N.M. (1975).