CICSR/COMPUTER SCIENCE
NOV 1 9 19
READING ROOM

V6T 1Z4
VANCOUVER, B.C. CANADA
262 - 2366 MAIN MALL
UNIVERSITY OF BRITISH COLUMBIA
CICSR/COMPUTER SCIENCE READING ROOM

A Collocation Solver for Mixed Order Systems of Boundary Value Problems

by

U. Ascher, J. Christiansen and R.D. Russell

Technical Report 77-13

November 1977

Department of Computer Science
The University of British Columbia
Vancouver, British Columbia V6T 1W5

Abstract

Implementation of a spline collocation method for solving boundary value problems for mixed order systems of ordinary differential equations is discussed.

The aspects of this method considered include error estimation, adaptive mesh selection, B-spline basis function evaluation, linear system solution and nonlinear problem solution.

The resulting general purpose code, COLSYS, is tested on a number of examples to demonstrate its stability, efficiency and flexibility.

CICSR/COMPUTER SCIENCE READING ROOM
UNIVERSITY OF BRITISH COLUMBIA
262 - 2366 MAIN MALL
VANCOUVER, B.C. CANADA
V6T 1Z4

A Collocation Solver for Mixed Order Systems of Boundary Value Problems

by

U. Ascher*, J. Christiansen**, and R.D. Russell**

1. Introduction

Recently there have been several efforts to develop high quality, general purpose software for the solution of boundary value problems for systems of ordinary differential equations. Most of the codes developed have been based on initial value methods, reflecting the current advanced state of such methods. In particular, multiple shooting codes have been developed by England, Nichols and Reid [19] and by Bulirsch, Stoer and Deuflhard [10]. Successful solution of some difficult nonlinear problems with the latter code is reported in [17]. Also, Scott and Watts have produced a super-position code with orthonormalization [36]. A comparison of some initial value type codes is given in [37].

A second approach has been implemented by Lentini and Pereyra [25,26], where a finite difference method with deferred corrections is used.

A thorough theoretical analysis of finite element methods has been available for some time [13], [34], [7], but, to our knowledge there has been no attempt, prior to this work, to write a general purpose code using these methods.

---

* Computer Science Department, University of British Columbia. Supported in part under NRC (Canada) Grant A4306.

**Mathematics Department, Simon Fraser University. Supported in part under NRC (Canada) Grant A7871.

In this paper we discuss an implementation of a spline collocation method for solving boundary value problems for mixed order systems of ordinary differential equations. While not in polished form, our code COLSYS (COLlocation for SYStems) is sufficiently stabilized that we are able to present a number of its theoretical and practical aspects and demonstrate the power of this preliminary version.

There are a number of reasons for our choice of the collocation method. It is the most suitable method among the finite element ones, for a general purpose code. See [1], [31] and [32], where complexity comparisons are made which support the above claim and also show collocation, when efficiently implemented, to be competitive with finite differences using extrapolation. The theoretical results on the convergence of the collocation method [11], [23], together with those on error estimation and mesh selection [33], [12] are more general than for the other methods mentioned. This, and the basic simplicity of the collocation procedure, also make programming of the method reasonably straightforward. COLSYS is designed to solve mixed order systems of nonlinear boundary value problems. This is in contrast to the other codes mentioned above which require conversion of a given problem to a first order system, thereby increasing the number of equations and changing the algebraic structure of the discretized problem. Numerous numerical experiments have demonstrated the stability of the collocation procedure, and recent attempts at adaptive mesh selection and error estimation have been quite successful [33]. For these reasons we feel that a robust, efficient collocation code can be developed to reliably solve a larger class of problems than has heretofore been possible.

Most of the points mentioned above are discussed and demonstrated in greater detail in the rest of the paper. In section 2 the collocation theory for mixed order boundary value systems [11], [23], is extended to obtain an error expression useful for adaptive mesh selection, generalizing a result in [33] for a scalar equation. Also, a theoretical justification of the error estimation strategy as well as practical aspects of these features are given.

Section 3 considers the method used for evaluating the piecewise polynomial collocation solution, expressed in terms of a B-spline basis. This involves appropriate modification of deBoor's B-spline evaluation procedures [4].

Section 4 describes some aspects of solving the collocation equations. Newton's method is currently used for solving nonlinear problems. For each Newton iteration, the resulting linear algebraic system of equations is solved using a package developed by deBoor and Weiss [8], after first bringing the equations into a banded block structure.

A number of representative test problems, demonstrating the stability and flexibility of COLSYS, are documented in section 5. These include linear and nonlinear problems of various degrees of difficulty. The linear examples are also tried with two other codes [36], [25] in order to put COLSYS in a perspective. From these results our code appears to be competitive in general and particularly suitable for mildly difficult and difficult problems. Some of the examples with a small parameter can only be solved by COLSYS. It is also the only one which can solve some problems with singularities without any modification. The relative efficiency of the code increases for problems of higher order and more than one component.

2. Error estimates and mesh selection

The class of problems treated by our code has the following general form: A system of d nonlinear differential equations of orders

$$m_1 \le m_2 \le \ldots \le m_d ,$$

$$(2.1) \qquad u_n^{(m_n)}(x) = F_n(x;u_1,u_1',\ldots,u_1^{(m_1-1)},u_2,\ldots,u_d^{(m_d-1)}) \equiv F_n(x;\underset{\sim}{z}(\underset{\sim}{u})),$$

$$a \le x \le b, \quad n = 1,\ldots,d ,$$

is subject to the nonlinear side conditions,

$$(2.2) \qquad g_j(\zeta_j;\underset{\sim}{z}(\underset{\sim}{u})) = 0 , \quad \zeta_1 \le \zeta_2 \le \ldots \le \zeta_{m^*} , \quad \zeta_j \in [a,b], \quad j = 1,\ldots,m^*,$$

where $m^* = \sum_{n=1}^{d} m_n$. To conveniently facilitate an efficient implementation we require that $m_d \le 5$ and that the side conditions (2.2) each involve only one point. Thus, for example, periodic boundary conditions are excluded. However, any problem with such nonseparated conditions (and even interface conditions) can be cast into form (2.1), (2.2) at the expense of increasing the size d of the problem, as we show by example in section 5.

To be able to apply the collocation theory we need to have an isolated solution u to (2.1)-(2.2). This occurs if the linearized problem at $\underset{\sim}{u}$ is uniquely solvable. Specifically, consider the curve $C \subset R^{m^*+1}$ defined by

$$C \equiv \{[x,u_1(x),\ldots,u_1^{(m_1-1)}(x),\ldots,u_d^{(m_d-1)}(x)] : x \in [a,b]\} ,$$

and the linear problem

$$(2.3) \qquad L_n \underset{\sim}{w} = 0 \qquad n = 1,\ldots,d ,$$

$$(2.4) \qquad B_j \underset{\sim}{w} = 0 \qquad j = 1,\ldots,m^* ,$$

where $\underset{\sim}{w} = (w_1,\ldots,w_d)$,

$$(2.3a)\qquad L_n\underset{\sim}{w} \equiv L_n(\underset{\sim}{u})\underset{\sim}{w} = w_n^{(m_n)} - \sum_{\ell=1}^{m^*} \frac{\partial F_n(\cdot;\underset{\sim}{z}(\underset{\sim}{u}))}{\partial z_\ell} \cdot z_\ell(\underset{\sim}{w}) ,$$

$$(2.4a)\qquad \beta_j\underset{\sim}{w} \equiv \beta_j(\underset{\sim}{u})\underset{\sim}{w} = \sum_{\ell=1}^{m^*} \frac{\partial g_j(\zeta_j;\underset{\sim}{z}(\underset{\sim}{u}))}{\partial z_\ell} \cdot z_\ell(\underset{\sim}{w}) .$$

If the Green's function $G(x,t)$ for (2.3)-(2.4) exists (implying unique existence for the linearized problem) and $F_1,\ldots,F_d,g_1,\ldots,g_{m^*}$ are sufficiently smooth in some δ-neighborhood of C, this is sufficient to guarantee that there exists a $\sigma > 0$ such that $u(x)$ is the unique solution of (2.1)-(2.2) in the sphere $B(D^m u,\sigma) = \{\underset{\sim}{w}(x) : \|w_n^{(m_n)} - u_n^{(m_n)}\| \le \sigma, n = 1,\ldots,d\}$ [11]. This also implies that Newton's method converges quadratically if the initial approximation is sufficiently close to $\underset{\sim}{u}(x)$.

To solve (2.1)-(2.2) numerically, we apply collocation at Gaussian points, using piecewise polynomial functions. If π is a partition of $[a,b]$

$$(2.5)\qquad \begin{cases} \pi : a = x_1 < x_2 < \ldots < x_N < x_{N+1} = b , \\ I_i = (x_i,x_{i+1}),\ h_i = x_{i+1} - x_i , \quad i = 1,\ldots,N , \\ h = \max\limits_{1\le i\le N} h_i \end{cases}$$

and $P_{k,\pi} = \{v \mid v$ is a polynomial of order k (degree $< k$) on I_i, $i=1,\ldots,N\}$, then we seek an approximate solution $\underset{\sim}{v} = (v_1,\ldots,v_d)$ such that $v_n \in P_{k+m_n,\pi} \cap C^{(m_n-1)}[a,b]$ $n = 1,\ldots,d$, or $\underset{\sim}{v} \in P_{k+\underset{\sim}{m},\pi} \cap C^{(\underset{\sim}{m}-1)}[a,b]$. We require $k \ge m_d$, where k is the number of collocation points per subinterval. If $\{\rho_j\}_{j=1}^k$ are the Gauss-Legendre points on $[-1,1]$, then $\{x_{ij}\}_{i=1,j=1}^{N,k}$ are the collocation points, where

$$(2.6)\qquad x_{ij} = \frac{x_i + x_{i+1}}{2} + \tfrac{1}{2}h_i\rho_j \equiv x_{i+\frac{1}{2}} + \tfrac{1}{2}h_i\rho_j .$$

The collocation equations which $\underset{\sim}{v}$ has to satisfy are thus

$$(2.7) \qquad v_n^{(m_n)}(x_{ij}) = F_n(x_{ij};\underset{\sim}{z}(\underset{\sim}{v})), \quad j = 1,\ldots,k, \quad i = 1,\ldots,N, \quad n = 1,\ldots,d ,$$

and (2.2).

The theory and a-priori error estimates for collocation have been presented in [11], (cf. also [23], [30], [7], [39]). Here, we merely quote the results that (assuming sufficient smoothness),

$$(2.8) \qquad \|u_n^{(\ell)} - v_n^{(\ell)}\|_\infty = O(h^{k+m_n-\ell}) \qquad \ell = 0,\ldots,m_n, \quad n = 1,\ldots,d$$

and, at the mesh points, superconvergence occurs

$$(2.9) \qquad |(u_n^{(\ell)} - v_n^{(\ell)})(x_i)| = O(h^{2k}) \quad i = 1,\ldots,N, \quad \ell = 0,\ldots,m_n-1, \quad n = 1,\ldots,d .$$

The phenomenon of higher order accuracy at the mesh points displayed in (2.9) may suggest (as has been noted in various places in the literature) using a-posteriori high order interpolation of an approximate solution at the mesh points to improve the overall accuracy, at least when $k > m_d$. However, it has been the experience of these authors and others that this is generally not a very useful idea, as the asymptotic range of h, $0 < h \le h_0$, where the superiority of the bound (2.9) over (2.8) is demonstrated, occurs very often for an h_0 which is effectively too small.

We feel that in practice it is more significant that the main term of the error expression is local if $k > m_d$. Below we briefly describe this analysis which is similar to that in [33] (cf. also [5]).

It is known [11] that a collocation solution of the linearized problem

$$(2.10a) \qquad L_n\underset{\sim}{w} = L_n\underset{\sim}{u} \qquad n = 1,\ldots,d$$

$$(2.10b) \qquad \beta_j\underset{\sim}{w} = \beta_j\underset{\sim}{u} \qquad j = 1,\ldots,m^* ,$$

where L_n and β_j are defined in (2.3a), (2.4a), lies within $0(h^{2k})$ of the collocation solution of the original problem (2.1),(2.2). Therefore, for terms of order less than $2k$ in h, one need only consider the form of the error for linear problems. The Green's function $G(x,t)$ exists if the linear problem has a unique solution. If (2.10a) is cast as a system of m^* first order equations, with one component assigned to each of $u_n^{(\ell)}$, $\ell = 0,\ldots,m_n-1$, $n = 1,\ldots,d$, then the Green's function $K(x,t)$ for this first order system can be constructed as in [30]. The Green's function $G(x,t)$ for the system (2.10a), (2.10b) then consists of a subset of the components of $K(x,t)$ — see [11]. Using a general form for $K(x,t)$ it can be shown that as a function of t, $G_{ni}(x,t)$ is in $C^{(m_n-1)}[a,b]$ if $i \neq n$ and $G_{nn}(x,t)$ is in $C^{(m_n-2)}[a,b]$ with $\dfrac{\partial^{m_n-1}}{\partial t^{m_n-1}} G_{nn}(t,t^+) - \dfrac{\partial^{m_n-1}}{\partial t^{m_n-1}} G_{nn}(t,t^-) = (-1)^{m_n}$. If

$\underset{\sim}{r}(x) \equiv L(\underset{\sim}{u}-\underset{\sim}{v})(x)$ then the error is $\underset{\sim}{e}(x) = \underset{\sim}{u}(x) - \underset{\sim}{v}(x) = \displaystyle\int_a^b G(x,t)\underset{\sim}{r}(t)dt$.

Using (2.7) and the convergence result (2.8) we obtain

$$r_n(t) = \frac{u_n^{(k+m_n)}(\sigma_i(t))}{k!} \cdot \prod_{j=1}^{k}(t-x_{ij}) \text{ for } t \in [x_i,x_{i+1}], \text{ for some } \sigma_i(t) \in [x_i,x_{i+1}].$$

So the error in the n-th component is

$$e_n(x) = \sum_{i=1}^{N}\sum_{\ell=1}^{d}\int_{x_i}^{x_{i+1}} G_{n\ell}(x,t)\, u_\ell^{(k+m_\ell)}(\sigma_i(t)) \prod_{j=1}^{k}(t-x_{ij})/k!\, dt, \quad n = 1,\ldots,d.$$

Continuity arguments as in [33] imply

$$(2.11)\qquad e_n^{(\ell)}(x) = \frac{u_n^{(k+m_n)}(x_i)}{2^{k+m_n-\ell}} P_n^{(\ell)}\left(\frac{2}{h_i}(x-x_{i+\frac{1}{2}})\right)h_i^{k+m_n-\ell} + 0(h^{k+m_n+1-\ell}) \quad n = 1,\ldots,d,$$

for $x \in I_i$, where

$$(2.12)\qquad P_n(\xi) = \int_{-1}^{\xi} \frac{(-1)^{m_n-1}(t-\xi)^{m_n-1}}{k!\,(m_n-1)!} \prod_{j=1}^{k}(t-\rho_j)dt = \frac{d^{k-m_n}}{d\xi^{k-m_n}} \frac{(\xi^2-1)^k}{2k!}$$

for $\xi \in (-1,1)$.

In arriving at error estimation and mesh selection schemes we assume that the local term in the error expression (2.11) is the dominant one. This, of course, can be guaranteed only when the mesh is quasiuniform, i.e. $\frac{h}{\min\limits_{1\le i\le N} h_i}$ is bounded, and h is small enough. If, for example, the solution behaves badly in one part of the domain and well in another, (2.11) indicates that h should still be taken small in the region of good behaviour in order to keep the $0(h^{k+m_n+1-\ell})$ term relatively small. However, our experience has been that the mesh selection and error estimation schemes usually work well, supporting our above assumption.

A-posteriori error estimate

Suppose we have approximations $\underset{\sim}{v}(\cdot)$ and $\underset{\sim}{v}^*(\cdot)$ on the meshes $\{x_i\}_{i=1}^{N+1}$ and $\{x_i^*\}_{i=1}^{2N+1}$ respectively, with $x^*_{2i-1} = x_i$ and $x^*_{2i} = x_{i+\frac{1}{2}} = \frac{1}{2}(x_i+x_{i+1})$. We want to estimate the maximum of the error $e_n^*(x) = u_n(x) - v_n^*(x)$ for $x \in [x_i, x_{i+1}]$. If $k \ge m_d$, v_n and v_n^* can be compared at several points to estimate [33]

$$\|v_n - v_n^*\| \frac{1}{2^{k+m_n+1}} + 0(h^{k+m_n+1}) \le \|e_n^*\| \le \|v_n - v_n^*\| \frac{1}{2^{k+m_n-1}} + 0(h^{k+m_n+1}) .$$

However, if $k > m_d$, we use the structure indicated by (2.11) as follows: Consider the points $x^*_{2i-2/3} = x_{i+1/6}$ and $x^*_{2i-1/3} = x_{i+1/3}$ (See fig. 2.1).

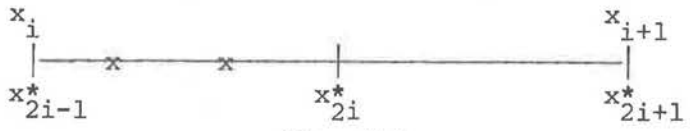

fig. 2.1

Let

$$\Delta_1 = |v_n(x_{i+1/6}) - v_n^*(x_{i+1/6})| = |e_n(x_{i+1/6}) - e_n^*(x_{i+1/6})|$$

$$= \frac{|u_n^{(k+m_n)}(x_i)|}{2^{k+m_n}} \left|P_n(-2/3) - \frac{1}{2^{k+m_n}} P_n(-1/3)\right| h_i^{k+m_n} + 0(h^{k+m_n+1}) \qquad (2.13)$$

and similarly

$$(2.14)\qquad \Delta_2 = \left|v_n(x_{i+1/3}) - v_n^*(x_{i+1/3})\right| = \frac{\left|u_n^{(k+m_n)}(x_i)\right|}{2^{k+m_n}}\left|P_n(-1/3) - \frac{1}{2^{k+m_n}}P_n(1/3)\right| h_i^{k+m_n} + 0(h^{k+m_n+1}),$$

where P_n is defined in (2.12). From (2.11),

$$(2.15)\qquad \max_{x\in[x^*_{2i-1},x^*_{2i}]}\left|e_n^*(x)\right| = \frac{\|P_n\|(\Delta_1+\Delta_2)}{\left|2^{k+m_n}P_n(-2/3)-P_n(-1/3)\right| + \left|2^{k+m_n}P_n(-1/3)-P_n(1/3)\right|} + 0(h^{k+m_n+1}).$$

When (2.15) is generalized to provide estimates of errors in all the components of $\underset{\sim}{z}(\underset{\sim}{v})$ then the weights multiplying $(\Delta_1+\Delta_2)$ are given by

$$(2.16)\qquad \omega_{k,\nu} = \frac{\|P^{(\nu)}\|}{\left|2^{2k-\nu}P^{(\nu)}(-2/3)-P^{(\nu)}(-1/3)\right| + \left|2^{2k-\nu}P^{(\nu)}(-1/3)-P^{(\nu)}(1/3)\right|} \qquad \nu = 0,\ldots,k-1$$

with $P(\xi) \equiv P(k,\xi) = \dfrac{\xi^2-1}{(2k)!}$.

These weights are precomputed and stored as constant data in the program, and the error is then estimated by ignoring the $0(h^{k+m_n-\ell+1})$ term in

$$(2.17)\qquad \max_{x\in[x_i^*,x_{i+1}^*]}\left|e_n^{*(\ell)}(x)\right| = \omega_{k,k-m_n+\ell}(\Delta_1+\Delta_2) + 0(h^{k+m_n-\ell+1}) \qquad \ell = 0,\ldots,m_n-1,$$

where Δ_1 and Δ_2 are taken for $v_n^{(\ell)}$, $n = 1,\ldots,d$.

Mesh Selection

The results below are a generalization of [5], [18], [12]. Given a set of tolerances TOL_j, $j = 1,\ldots,NTOL$, with a set of pointers $LTOL_j$, $j = 1,\ldots,NTOL$, COLSYS attempts to satisfy

$$(2.18)\qquad \|z_\ell(\underset{\sim}{u}) - z_\ell(\underset{\sim}{v})\| \le TOL_j \qquad \ell = LTOL_j,\ j = 1,\ldots,NTOL.$$

The aim of the mesh selection algorithm is to meet the above requirements with the least number of mesh points.

As before we neglect the global term in (2.11) and write

$$(2.19)\quad \max_{x\in[x_i,x_{i+1}]} |e_n^{(\ell)}(x)| \doteq C_{k,k-m_n+\ell} |u_n^{(k+m_n)}(x_i)| h_i^{k+m_n-\ell} \qquad \ell = 0,\ldots,m_n-1 ,$$

where

$$(2.20)\quad C_{k,\nu} = \|P^{(\nu)}(k;\cdot)\|/2^{2k-\nu} \qquad \nu = 0,1,\ldots,k-1 .$$

For each j $(1 \le j \le NTOL)$, let $\ell = LTOL_j$, let $n = JTOL_j$ indicate the component of $\underset{\sim}{u}$ that $z_\ell(\underset{\sim}{u})$ is a derivative of, let $WEIGHT_j$ be the appropriate $C_{k,\nu}$ divided by TOL_j, and let $ROOT_j$ be the inverse of the expected rate of convergence of $z_\ell(\underset{\sim}{v})$.

From (2.18)-(2.20), the goal is to pick a mesh $\{x_i^*\}_{i=1}^{N^*+1}$ for which

$$(2.21)\quad \max_{1\le j\le NTOL} WEIGHT_j \cdot |u_n^{(k+m_n)}(x_i^*)| h_i^{*\,1/ROOT_j} \le 1 \quad (n=JTOL_j);\ i=1,\ldots,N^*,$$

for the smallest N^* possible. Actually finding this mesh is impossible since the $u_n^{(k+m_n)}(x_i^*)$ are unknown. Moreover, for COLSYS the final mesh is a halving of the one before last (so that an error estimate is at hand).

If

$$(2.22)\quad S_j(x) = WEIGHT_j |u_n^{(k+m_n)}(x)| ,$$

and

$$(2.23)\quad S(x) = \max_{1\le j\le NTOL} S_j^{ROOT_j}(x) ,$$

then (2.21) is equivalent to

$$(2.24)\quad S(x_i^*) h_i^* \le 1 \qquad i = 1,\ldots,N^* .$$

A collocation solution $\underset{\sim}{v}$ on a mesh satisfying (2.24) would satisfy

$$(2.25)\qquad \|z_\ell(\underset{\sim}{u}) - z_\ell(\underset{\sim}{v})\| \le \mathrm{TOL}(1 + O(h)) \ , \ \ell = \mathrm{LTOL}_j, \ j = 1,\ldots,\mathrm{NTOL},$$

the $O(h)$ term arising from neglecting higher order terms in (2.19). By requiring that

$$(2.26)\qquad \int_{x_i^*}^{x_{i+1}^*} S(x)\,dx = 1$$

instead of (2.24), (2.25) still holds [5]. To approximately satisfy (2.26), we still need to approximate $u_n^{(k+m_n)}(x)$, $n = 1,\ldots,d$. Given a mesh $\{x_i\}_{i=1}^{N+1}$ and an approximate collocation solution $\underset{\sim}{v}$, an accurate approximation for the higher order derivatives can be constructed as follows (cf.[12]): The polynomial in the error expression for the $(k+m_n-1)$-st derivative of the n-th component is $\frac{1}{(2k)!}\frac{d^{2k-1}}{d\xi^{2k-1}}(\xi^2-1)^k = \xi$. Therefore, $e_n^{(k+m_n-1)}(x_{i+\frac{1}{2}}) = O(h^2)$ and

$$(2.27)\qquad \hat{u}_n(x_{i+1}) := \frac{2\left|v_n^{(k+m_n-1)}(x_{i+1}) - v_n^{(k+m_n-1)}(x_i)\right|}{x_{i+2} - x_i}$$

$$= \left|u_n^{(k+m_n)}(x_{i+1})\right| + O(h) = \left|u_n^{(k+m_n)}(x)\right| + O(h)$$

$$\text{for } x \in [x_i, x_{i+2}] \quad i = 1,\ldots,N-1 \ .$$

Define $\hat{u}_n(x)$ over the whole interval $[a,b]$ by

$$(2.28)\qquad \hat{u}_n(x) = \begin{cases} \hat{u}_n(x_i) & x \in [x_i, x_{i+1}] \quad i = 2,\ldots,N \\ \hat{u}_n(x_2) & x \in [x_1, x_2] \end{cases}$$

so that $\left|u_n^{(k+m_n)}(x)\right| = \left|\hat{u}_n(x)\right| + O(h)$. Then

$$(2.29)\qquad \hat{s}(x) = \max_{1\le j\le \mathrm{NTOL}} [\mathrm{WEIGHT}_j \cdot \hat{u}_n(x)]^{\mathrm{ROOT}_j}, \quad n = \mathrm{JTOL}_j \ ,$$

is a piecewise constant computable function, and (2.25) is satisfied for $\{x_i^*\}_{i=1}^{N^*+1}$ by requiring

$$(2.30) \qquad \int_{x_i^*}^{x_{i+1}^*} \hat{s}(x)dx = 1 \qquad i = 1,\ldots,N^* \ .$$

In practice (2.30) may lead to a very large N^*, compared to N, which could mean that N^* has been determined by premature data. Also, an error estimate is needed at the end to check whether the tolerances have been satisfied. So, we modify the criterion (2.30) to allow for these considerations by picking a new mesh (for some N^*), according to

$$(2.31) \qquad \int_{x_i^*}^{x_{i+1}^*} \hat{s}(x)dx = \gamma \equiv \frac{1}{N^*}\int_a^b \hat{s}(x)dx = \frac{1}{N^*}\sum_{i=1}^{N} \hat{s}(x_i)h_i \ .$$

There are still two questions to be answered: When to redistribute the points, as opposed to just halving the current mesh, and how to choose N^*. When an approximate solution on the current mesh $\{x_i\}_{i=1}^{N+1}$ has been obtained, the diagnostics $r_1 = \max_i \hat{s}(x_i)h_i$, $r_2 = \sum_{i=1}^{N} \hat{s}(x_i)h_i$, and $r_3 = r_2/N$ are computed. The ratio r_1/r_3 , gives some idea of the gain to be achieved by redistribution. Specifically, the code feels it can reduce the error by as much in redistributing with $N^* = N$ as by taking $N^* = \frac{r_1}{r_3}\cdot N$ with the current distribution. Our present policy is to redistribute only when $r_1 \geq 2r_3$. This includes an amount of skepticism about the data derived from the old mesh.

When redistributing, $r_2 = \gamma N^*$ predicts the number of points needed to satisfy the tolerances. If r_2 is much larger or much smaller than N, then we do not put much faith in this prediction. The current policy is to take

$$(2.32) \qquad N^* = \min\{\tfrac{1}{2}\bar{N}, N, \tfrac{1}{2}\max[N,r_2]\} \ ,$$

where $\bar{N}$ is the maximum number of subintervals allowed by the storage specifications. This allows for changes up to a factor of 2 in N and for later

halving of the mesh in order to obtain an error estimate. Also, restrictions are placed on the number of times a mesh can be redistributed before halving.

3. B-spline Evaluation

For reasons of efficiency, stability, and flexibility in order and continuity, B-splines are chosen as the basis functions. Efficient algorithms for calculating with B-splines are given by deBoor [ ], who implements these algorithms in a Fortran package [6]. Evaluation of the basis functions is a major cost for finite element methods, and careful implementation of the selected algorithms is necessary for the code to be competitive. Our use of B-splines is somewhat special because (i) we are solving a system of differential equations, so many repetitive calculations can be avoided, (ii) the continuity in the solution at the mesh points is more restricted here than in [6], allowing us to trade unneeded generality for an increase in speed, and (iii) in many occasions we evaluate the B-splines at points which are placed in a regular fashion in each subinterval. We take advantage of these special features in implementing restricted versions of deBoor's algorithms.

As we only outline the modifications to these algorithms, the interested reader is referred to [2] for the complete details.

A. Evaluation of the B-splines and the solution

Recall that $v_n(x) \in P_{k+m_n,\pi} \cap C^{(m_n-1)}[a,b]$ $(1 \le n \le d)$ for a given mesh $\pi : a = x_1 < x_2 < \ldots < x_{N+1} = b$. If $N_{j,k}$ is the j-th B-spline of

order k [4], then

$$(3.1) \qquad v_n(x) = \sum_{j=-k-m_n+2}^{Nk} \alpha_{j,n} N_{j,k+m_n}(x) .$$

Defining the knot sequence

$$(3.2) \qquad t_j = \begin{cases} x_1 & j \le k+m_d \\ x_{i+1} & ik+m_d < j \le (i+1)k+m_d , \quad (1 \le i \le N-1) \\ x_{N+1} & Nk+m_d < j \le (N+1)k+2m_d , \end{cases}$$

then only $k+m_n$ B-splines may be nonzero at $x \in [t_i, t_{i+1})$, viz.

$$(3.3) \qquad v_n(x) = \sum_{j=-k-m_n+1}^{0} \alpha_{i+j,n} N_{i+j,k+m_n}(x) .$$

The algorithm in [4] for the evaluation of these B-splines is

Algorithm I:

Let $N_{i,1}(x) \equiv 1$

Do for $\ell = 1,\ldots,k+m_d-1$:

  $N_{i-\ell,\ell+1}(x) = 0$

  Do for $j = 1,\ldots,\ell$:

    $M_{i+j-\ell,\ell}(x) = N_{i+j-\ell,\ell}(x)/(t_{i+j} - t_{i+j-\ell})$

    $N_{i+j-\ell-1,\ell+1}(x) = N_{i+j-\ell-1,\ell+1}(x) + (t_{i+j}-x)M_{i+j-\ell,\ell}(x)$

    $N_{i+j-\ell,\ell+1}(x) = (x-t_{i+j-\ell})M_{i+j-\ell,\ell}(x)$.

From the recursive manner in which the B-splines are defined it is clear that algorithm I need only be performed once for a given x to produce the B-splines needed to evaluate all components of $\underset{\sim}{v}(x)$ by (3.3). Also, since the structure of the knot sequence is known in terms of the mesh π, there is no need to generate the t_j's. If $x \in [x_I, x_{I+1})$ we can make the changes in algorithm I according to

$$(3.4)\qquad t_{i+j} - t_{i+j-\ell} = \begin{cases} h_{I-1} + h_I & \text{for} \quad j \le k,\ j+k \le \ell \\ h_I & \text{for} \quad j \le k,\ \ell \le j+k-1 \\ h_I + h_{I+1} & \text{for} \quad k+1 \le j \end{cases}$$

and

$$(3.5)\qquad t_{i+j} - x = \begin{cases} \rho h_I & \text{for} \quad 1 \le j \le k \\ \rho h_I + h_{I+1} & \text{for} \quad k+1 \le j \le k+m_d-1 \quad (\le 2k) \end{cases}$$

where ρ is chosen appropriately. The substitutions (3.4) and (3.5) have lead to an algorithm about 50% faster than the general one [6] (when running on an IBM 370/168).

Some of the B-spline values at x depend only on their relative position in $[x_I, x_{I+1})$ and not on the subinterval itself. For example, the collocation points are located at the same relative positions in all subintervals, so it is only necessary to evaluate these mesh independent splines once for each relative position. The points at which the approximate solution is evaluated for the error estimate (2.17) are another instance where this saving may be made. Since $\frac{1}{2}(k+m_d)(k+m_d-1)$ B-splines are needed for any x and only $m_d(m_d-1)$ are subinterval dependent, a saving of at least 50% is obtained for $k \ge m_d$.

We have used two routines in the implementation of the modified version of algorithm I. The first evaluates those B-splines which are mesh independent, while the second is for the splines whose values depend on I(where $x \in [x_I, x_{I+1})$).

We do not exploit the symmetry of the collocation points or the error estimation points; the saving is too small given the additional complexity. Also, we do not incorporate the nonconvex modification suggested in [31].

While it can save a multiplication in the last line of algorithm I and our experiments have not yielded a case where accuracy was significantly affected, the improvement in efficiency proved small enough that we have decided to be conservative.

B. Evaluation of spline derivatives

Given an approximate solution component $v_n(x)$, as in (3.3), its derivatives are given by

$$(3.6) \qquad v_n^{(r)}(x) = (k+m_n-1) \ldots (k+m_n-r) \sum_{j=-k-m_n+r+1}^{0} \alpha_{i+j,n}^{(r)} N_{i+j,k+m_n-r}(x)$$

where

$$(3.7) \qquad \alpha_{i+j,n}^{(r)} = \begin{cases} \alpha_{i+j,n} & \text{for } r = 0 \\ \dfrac{\alpha_{i+j,n}^{(r-1)} - \alpha_{i+j-1,n}^{(r-1)}}{t_{i+j+k+m_n-r} - t_{i+j}} & \text{for } r > 0 . \end{cases}$$

The B-spline package in [6] contains a subroutine which prepares the divided difference table (3.7) (with $\alpha_{i+j,n}^{(r)}$ multiplied by $(k+m_n-1) \ldots (k+m_n-r)$). We have written a similar routine which uses the particular form of $(t_{i+j+k+m_n-r} - t_{i+j})$.

To compute $\underset{\sim}{z}(\underset{\sim}{v}) = (v_1, v_1', \ldots, v_1^{(m_1-1)}, v_2, \ldots, v_d, \ldots, v_d^{(m_d-1)})$ we only need $v_n^{(r)}(x)$, $r = 0,\ldots,m_n-1$, $n = 1,\ldots,d$. There are several occasions where evaluation of $\underset{\sim}{z}(\underset{\sim}{v})$ is necessary. Values of $\underset{\sim}{z}(\underset{\sim}{v})$ are needed for setting up the equations during the iterations on nonlinear problems and for the error estimation procedure. Also, when COLSYS has terminated successfully the user can evaluate $\underset{\sim}{z}(\underset{\sim}{v})$ for the final approximation. There are two efficient ways to evaluate $\underset{\sim}{z}(\underset{\sim}{v})$:

Algorithm II:

(a) Generate $\alpha_{i,n}^{(r)}$ $i = 1,\ldots,Nk+m_n$; $r = 1,\ldots,m_n-1$; $n = 1,\ldots,d$,

(b) for $x \in (x_I, x_{I+1})$, form the $\frac{1}{2}(k+m_d)(k+m_d+1)$ nonzero B-splines up to order $k + m_d$,

(c) form $v_n^{(r)}(x)$ $(r = 0,\ldots,m_n-1;\ n = 1,\ldots,d)$ by (3.6).

Algorithm II':

(a) Generate $v_n^{(r)}(x_i)$ $r = 1,\ldots,k+m_n-1$; $n = 1,\ldots,d$; $i = 1,\ldots,N$ by algorithm II,

(b) for $x \in [x_I, x_{I+1})$

$$(3.8) \qquad v_n^{(r)}(x) = \sum_{j=r}^{k+m_n-1} \frac{v_n^{(j)}(x_I)}{(j-r)!}(x-x_I)^{j-r} .$$

While algorithm II' requires more than twice the storage and more initialization than algorithm II, it is many times more efficient when $\underset{\sim}{z}(\underset{\sim}{v})$ is required for a large number of points. For the collocation example in [6], algorithm II' was used.

In [2] these algorithms are examined in our setting for two cases - when $\underset{\sim}{z}(\underset{\sim}{v})$ is to be evaluated at

(i) M_1 points irregularly distributed in [a,b] and (ii) $M_1 = M_2 N$ points, consisting of M_2 regularly distributed points in each subinterval. The numbers of multiplications plus divisions required for Algorithm II are approximately

(i) $(m^*-d)(k+2)N + [(k+m_d)(k+m_d-1) + \bar{M}]M_1$

(ii) $\{(m^*-d)(k+2) + [2(m_d^2-m_d) + \bar{M}]M_2\}N$

and for Algorithm II',

(i) and (ii) $[(m^*-d)(k+2) + kd(k+2) + 2(m_d^2-m_d) + \bar{M} + \frac{d}{2}k(k+1)]N +$

$$[\bar{M} + 2(k+m_d-2)]M_1$$

where $\bar{M} = (k+\frac{1}{2})m^* + \frac{1}{2}\sum_{n=1}^{d} m_n^2$.

For case (i) algorithm II is more efficient when $M_1 \leq \lambda N$ where, e.g., $\lambda \doteq 2.5$ if $d = 1$, $m = 2$, $k = 3$ and $\lambda \doteq 4.8$ if $d = 3$, $m_1 = 1$, $m_2 = m_3 = 3$, $k = 4$. In general, λ grows with d. In case (ii) algorithm II is more efficient for most practical situations. Consequently, we at present use only algorithm II.

C. Derivatives of the B-splines

In order to generate the collocation equations an algorithm is needed to evaluate the B-splines derivatives. Formulas (3.6) and (3.7) could be used with $\alpha^{(0)}_{i+j,n} = \delta_{j\ell}$ for the function $N_{i+\ell,k+m_n}(x)$, but a number of savings can be made. First, if $m_n = m_{n+1}$ there is no need to repeat the computations, so COLSYS initially isolates the set of strictly increasing orders and deals only with them. Second, the algorithm avoids performing (3.7) on the many zero coefficients (as is also done in [6]). Third, the special form of $(t_{i+j+k+m_n-r} - t_{i+j})$ is used. The fourth improvement arises from the fact that we have a system of differential equations. If the B-spline derivatives are evaluated for $n = d$ then a number of the $\alpha^{(r)}_{i+j,n}$ may be determined directly from

$$\alpha^{(r)}_{i+j,n} = \begin{cases} \alpha^{(r)}_{i+j+(m_d-m_n),d} & 1 \leq j \leq k - (m_d-m_n) - r + 1 \\ \alpha^{(r)}_{i+j,d} & m_d \leq j \leq k + m_n - r \,. \end{cases} \tag{3.9}$$

D. Highest order derivatives

Selecting a new mesh requires the values of the piecewise constant functions $v_n^{(k+m_n-1)}(x)$, $1 \leq n \leq d$. These are obtained by starting with the values

$\alpha_{i+j,n}^{(m_n-1)}$ $(-k \le j \le 0)$, which have been obtained in algorithm II, and repeatedly applying (3.7) with $t_{i+j+k+m_n-r} - t_{i+j} = h_I$ to get $\alpha_{i,n}^{(k+m_n-1)} = v_n^{(k+m_n-1)}(x)$ for $x \in [t_i,t_{i+1}) = [x_I,x_{I+1})$.

4. The nonlinear iteration and the linear system solver

In this section we briefly discuss the handling of nonlinear problems and the implementation of the linear system solver.

A. Newton iteration

To solve (2.1), (2.2) we apply the Newton process of linearization and iteration. Specifically, choose an initial approximation $\underset{\sim}{v}^{\prime 0} \in P_{k+m} \cap C^{(m-1)}[a,b]$. Then, for $s = 0,1,2,\ldots$ until a convergence criterion is satisfied, solve by collocation the problem

$$(4.1) \qquad L_n(\underset{\sim}{v}^{\prime s})\underset{\sim}{w} = f_n \qquad n = 1,\ldots,d$$

$$(4.2) \qquad \beta_j(\underset{\sim}{v}^{\prime s})\underset{\sim}{w} = \gamma_j \qquad j = 1,\ldots,m^*,$$

for the solution $\underset{\sim}{v}^{\prime s+1}$. Here L_n, β_j are defined in (2.3a), (2.4a) and

$$(4.3) \qquad f_n \equiv f_n(\cdot;\underset{\sim}{v}^{\prime s}) = F_n(\cdot;\underset{\sim}{v}^{\prime s}) - \sum_{\ell=1}^{m^*} \frac{\partial F_n(\cdot;\underset{\sim}{v}^{\prime s})}{\partial z_\ell} \cdot z_\ell(\underset{\sim}{v}^{\prime s}) \qquad n = 1,\ldots,d$$

$$(4.4) \qquad \gamma_j \equiv \gamma_j(\underset{\sim}{v}^{\prime s}) = \sum_{\ell=1}^{m^*} \frac{\partial g_j(\zeta_j;\underset{\sim}{v}^{\prime s})}{\partial z_\ell} \cdot z_\ell(\underset{\sim}{v}^{\prime s}(\zeta_j)) - g_j(\zeta_j;\underset{\sim}{v}^{\prime s}) \qquad j = 1,\ldots,m^*.$$

Most advantages and disadvantages of the Newton method are well-known. Generally, if the initial approximation $\underset{\sim}{v}^{\prime 0}$ is close enough to $\underset{\sim}{v}$, the method performs very satisfactorily. However, when $\underset{\sim}{v}^{\prime 0}$ is far from $\underset{\sim}{v}$, the behaviour of the algorithm is unpredictable (cf. [9], [16]). We are

currently conducting an investigation to find more reliable fast algorithms to handle nonlinearities, and intend to report the results elsewhere.

Implementing the Newton iteration requires determining when the desired error tolerances (2.18) are satisfied. For a nonlinear problem, the error has two components, $\underset{\sim}{v}'^{s+1} - \underset{\sim}{v}$ and $\underset{\sim}{v} - \underset{\sim}{u}$, where $\underset{\sim}{v}'^{s+1}$ is the Newton iterate which satisfies the convergence criterion to be specified and is thus taken as the approximation to $\underset{\sim}{v} = \lim_{s\to\infty} \underset{\sim}{v}'^{s}$. For any superlinearly convergent method, $\lim \dfrac{\|\underset{\sim}{v}'^{s+1} - \underset{\sim}{v}'^{s}\|_\infty}{\|\underset{\sim}{v}'^{s} - \underset{\sim}{v}\|_\infty} = 1$ (see[15]), so that in the limit $\underset{\sim}{v}'^{s+1} - \underset{\sim}{v}'^{s}$ is a good measure for $\underset{\sim}{v}'^{s} - \underset{\sim}{v}$. Thus, the convergence criterion for the nonlinear iteration in (4.1),(4.2) is

$$\|z_\ell(v'^{s+1}) - z_\ell(v'^{s})\|_\infty \le TOL_j \qquad \ell = LTOL_j\ , \quad j = 1,\ldots,NTOL\ . \tag{4.5}$$

To check (4.5) efficiently, recall that

$$v_n^{(r)}(x) = \sum_i \alpha_{i,n}^{(r)} N_{i,k+m_n}(x)\ , \qquad r = 0,\ldots,m_n-1,\ n = 1,\ldots,d\ , \tag{4.6}$$

where the $\alpha_{i,n}^{(r)}$ satisfy (3.7). The $N_{i,j}$ are normalized B-splines, so

$$\|v_n^{(r)}\|_\infty \le \max_i |\alpha_{i,n}^{(r)}|\ . \tag{4.7}$$

The $\alpha_{i,n} = \alpha'^{s}_{i,n}$ are precisely the coefficients computed when solving the linear system in each Newton iteration. Thus the nonlinear iteration convergence criterion is as follows:

1. Having obtained $\alpha'^{s+1}_{i,n}$, compute $\alpha_{i,n}^{(r),s+1}$ for all i, $r = 0,\ldots,m_n-1$, $n = 1,\ldots,d$.

2. For $j = 1,\ldots,NTOL$, let $\ell = LTOL_j$ and let (n,r) correspond to the coordinate ℓ of $\underset{\sim}{z}(\cdot)$.

If $\|\alpha^{(r),s+1}_{(\cdot),n} - \alpha^{(r),s}_{(\cdot),n}\|_\infty > TOL_j$ then go to step 4.

3. Dump $\alpha^{(\cdot),s+1}$ onto $\alpha^{(\cdot),s}$, signal success, and exit the Newton iteration.

4. Dump $\alpha^{(\cdot),s+1}$ onto $\alpha^{(\cdot),s}$ set $s = s+1$, and reiterate.

In fact, steps 1 and 2 above are combined so that only the array $\alpha^{(\cdot),s}_{(\cdot),(\cdot)}$ is stored. Since the computation in step 1 is always needed to evaluate the approximate solution (see section 3 and [2]), it is not wasteful. Finally note that the criterion is somewhat pessimistic and is scaling-resistant [14].

B. The linear system solver

Here we consider the method for the solution of the set of algebraic equations resulting from collocation applied to (4.1), (4.2). With $x_{i1},\ldots,x_{ik}$ the k Gaussian points in the i-th subinterval $I_i = (x_i, x_{i+1})$, $1 \le i \le N$ (cf. (2.5), (2.6)), write these equations as

$$L_n \underset{\sim}{v}(x_{ij}) = f_n(x_{ij}) \qquad j = 1,\ldots,k,\ i = 1,\ldots,N,\ n = 1,\ldots,d, \tag{4.8}$$

$$\beta_j \underset{\sim}{v} = \gamma_j \qquad j = 1,\ldots,m^*. \tag{4.9}$$

The total number of equations in (4.8), (4.9) is $Nkd + m^*$, the dimension of the approximation space (or the number of parameters $\alpha_{i,n}$ to be determined).

Consider next the structure of the matrix obtained from (4.8), (4.9). Fixing i and n, $1 \le i \le N$, $1 \le n \le d$, there are m_n nonzero B-splines on $I_{i-1} \cup I_i$, $k-m_n$ B-splines which vanish outside I_i and m_n which vanish outside $I_i \cup I_{i+1}$. Giving $\alpha = (\alpha_{in})$ the natural ordering $\alpha_{11},\alpha_{21},\ldots,\alpha_{Nk+m_1,1}$, $\alpha_{12},\ldots\alpha_{Nk+m_d,d}$ causes an inconvenient zero structure, since it is desirable to have all nonzero elements concentrated around the main diagonal. Thus, we reorder the coefficient vector $\underset{\sim}{\alpha}$ is such a way that, for each i, all the columns in the matrix which contain nonzero entries corresponding to the i-th subinterval are adjacent. This produces a block-structured matrix whose i-th

block, $1 \le i \le N$, is characterized as follows:

Rows: With ℓ_i side conditions given at points ζ_ℓ, $x_i \le \zeta_\ell < x_{i+1}$ (when $i = N$, $x_N \le \zeta_\ell \le x_{N+1}$), there are $kd + \ell_i$ corresponding rows. (For each n, $1 \le n \le d$, k rows correspond to $x_{i1},\ldots,x_{ik}$). The numbering of the rows increases with the argument x.

Columns: For each $v_n(\cdot)$, $1 \le n \le d$, there are m_n B-splines which do not vanish on $I_i \cup I_{i-1}$. The corresponding columns, m_n for each n, will appear first in the order $m_1,m_2,\ldots,m_d$, totalling m^* columns. Then come $k-m_1,k-m_2,\ldots,k-m_d$ columns corresponding to the $kd-m^*$ B-splines which vanish outside I_i. The m^* columns of those B-splines which vanish outside $I_i \cup I_{i+1}$ appear last, ordered the same way as the first m^* columns. The total number of columns is therefore $kd + m^*$ (Note that $m^* = \sum_{i=1}^{N} \ell_i$ and $kd + m^* \ge kd + \ell_i$).

initial coordinates: The upper left element of the i-th block is the (i_1,i_2)-th element of the matrix, where

$$i_2 = (i-1)kd + \sum_{j=1}^{i-1} \ell_j + 1, \quad i_2 = (i-1)kd + 1 .$$

As an example, take $d = 2$, $m_1 = 1$, $m_2 = 2$, $k = 3$, $z_1 = z_2 = a$, $z_3 = b$, and $N = 3$. Then the reordered collocation matrix has the form

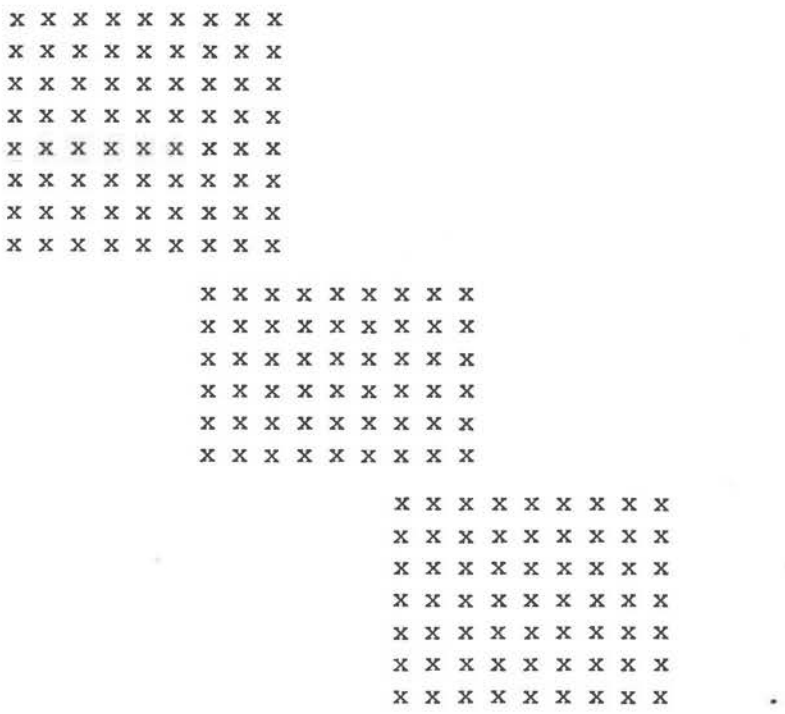

fig. (4.1).

This is precisely the zero structure for the collocation method with $k \cdot d = 6$ points per subinterval applied to a problem of one differential equation of order $m^* = 3$. The matrix can be considered as banded (asymmetric), but this would almost double the amount of nonzero entries. It is better considered as almost block diagonal [8].

For the solution of the linear systems we have adopted the code developed in [8] which performs Gauss elimination with scaled row pivoting. This proceeds as follows:

For $i = 1,2,\ldots,N$ do the following:

1. If $i > 1$, append the $\sum_{j=1}^{i-1} \ell_j$ rows of the $(i-1)$st block, not used as pivotal rows, to the beginning of the i-th block, to form a block of $kd + \sum_{j=1}^{i} \ell_j$ rows.
2. Apply kd steps of Gauss elimination with scaled row pivoting, storing the resulting factorization in place of the original data.

3. If $i = N$ (the last block is square of size $kd + m^*$), apply $m^* - 1$ more elimination steps.

This produces an LU factorization of the original matrix. For a given right hand side, the solution $\underset{\sim}{\alpha}$ is then obtained by a forward-backward substitution.

The permuted ordering in the solution vector $\underset{\sim}{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{Ndk+m^*})^T$ is as follows: For $x \in [x_i, x_{i+1})$ and $1 \le n \le d$, let $\mu_n = \sum_{\ell=1}^{n-1} m_\ell$, and $\eta_n = (n-1)k - \mu_n$. Then

$$
\begin{aligned}
(4.14) \qquad v_n(x) = & \sum_{\ell=1}^{m_n} \alpha_{(i-1)kd+\mu_n+\ell} \, N_{(i-1)kd+\mu_n+\ell,\,k+m_n}(x) \\
& + \sum_{\ell=1}^{k-m_n} \alpha_{(i-1)kd+\eta_n+m^*+\ell} \, N_{(i-1)kd+\eta_n+m^*+\ell,\,k+m_n}(x) \\
& + \sum_{\ell=1}^{m_n} \alpha_{ikd+\mu_n+\ell} \, N_{ikd+\mu_n+\ell,\,k+m_n}(x) \, .
\end{aligned}
$$

The package [8] implementing the above method is used in COLSYS because of its availability, stability, and its advantages over treating the system as merely banded. There is, however, some amount of fill-in generated. The i-th block has $\sum_{j=1}^{i-1} \ell_i$ rows appended to it, and so the ratio of storage from fill-in to total storage is

$$
\frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{j=1}^{i-1} \ell_j}{kd + \ell_i} \, .
$$

For the example in fig. 4.1 this ratio is about 1/5. If N is large and we consider a two-point boundary value system with half of the boundary conditions at each end, the ratio tends to $\frac{1/2\ m^*}{kd}$. This is always less than $1/2$ since $k \ge m_d \ge m_{d-1} \ge \ldots \ge m_1$, but the value $1/2$ is obtained when $k = m_d = m_1$.

A method which generates no fill-in has been proposed in [38]. Here row and column pivoting are performed alternately. A comparison between the above two methods, regarding their efficiency and stability, is planned for the future.

5. Numerical examples

COLSYS has been tested on a large variety of problems. We examine here a representative selection of them to demonstrate the performance of the code. Some comparisons with other codes are made as well, in order to gain a relative perspective. However, these comparisons are not exhaustive and should not be treated as final. In particular, COLSYS is not yet a production code, and substantial modifications are expected in the future.

The examples quoted here were run in double precision (14 hexadecimal digits) on the IBM 370/155 at Simon Fraser University, using the Fortran G1 compiler. Because of fluctuations in the computing environment, variations of 5 - 10% in the run times are meaningless.

In the examples below, the following notation is used:

$u_i(x)$ - i^{th} component of the exact solution

$E(u_i^{(j)})$ - uniform error in $u_i^{(j)}(x)$ (available when the exact solution is known).

est $E(u_i^{(j)})$ - estimated uniform error in $u_i^{(j)}(x)$.

$TOL(u_i^{(j)})$ - absolute error tolerance for the component $u_i^{(j)}(x)$. (COLSYS allows the user to specify different tolerances for different components, and the mesh selection algorithm considers only those components for which tolerances are specified).

time - the actual solution time in seconds (not including error checking time).

$a \pm b$ - $a \cdot 10^{\pm b}$

k - number of collocation points per subinterval.

mesh sequence (iterations) - successive mesh sizes, i.e. numbers N of subintervals required, followed in parenthesis by the number of Newton iterations performed on each mesh for nonlinear problems.

The errors $E(u_i^{(j)})$ are approximated by measuring the error at 4 equally spaced points in each subinterval. Unless otherwise stated, the initial mesh for COLSYS is uniform. Incidentally, if NMAX is the maximum number of subintervals used and n_r boundary conditions are given at the right end point, then approximately $[(kd+m^*-n_r)(kd+m^*+2) + m^*(4+k) + 3kd + k + 8]NMAX$ words of storage are required by COLSYS.

Example 1 [22]

$$\varepsilon y'' + xy' = -\varepsilon\pi^2\cos(\pi x) - (\pi x)\sin(\pi x) \qquad -1 \le x \le 1$$

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

$$u(x) = \cos(\pi x) + \operatorname{erf}(x/\sqrt{2\varepsilon})/\operatorname{erf}(1/\sqrt{2\varepsilon})$$

The solution has a spike at $x = 0$ (see fig. 5.1). Results for various values of ε are given in table 1 below.

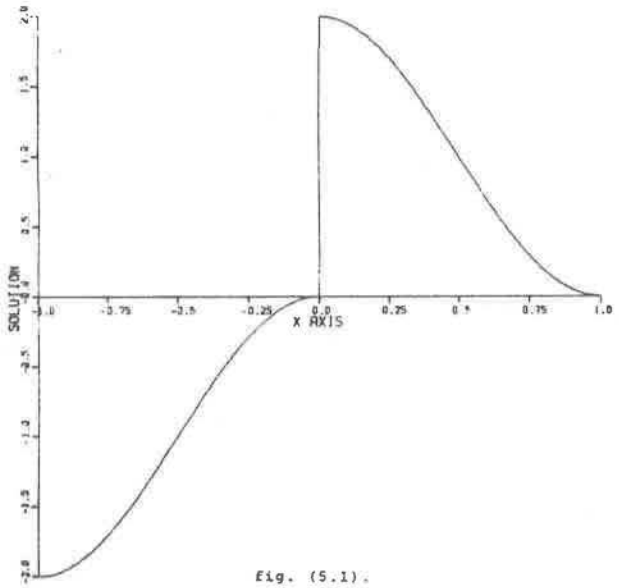

Fig. (5.1).

Table 1 - example 1

ε	k	TOL(u)	E(u)	estE(u)	TOL(u')	E(u')	estE(u')	time	mesh sequence
.1-1	4	.1-1	.45-4	.19-4	.1-1	.22-2	.12-2	1.1	8,16
.1-1	4	.1-5	.22-8	.18-8	.1-5	.24-6	.15-6	8.16	8,16,15,30,17,34,68
.1-3	4	.1-5	.16-7	.19-8	.1-5	.67-5	.25-6	25.9	8,16,32,64,35,70,35,70,35,70,140
.1-5	4	.1-5	.48-9	.43-9	.1-5	.23-6	.45-7	68.23	8,16,32,64,128,128,128,128,256,128,256,128,256.

As ε gets smaller, the problem gets stiffer and the mesh selection algorithm has to have more mesh points in order to resolve the difficulty. In fact, if we choose a good initial mesh (concentrated around $x = 0$) which contains only a few points, the mesh selection algorithm often produces worse meshes at first, because the error is very much different from the assumed asymptotic form. In order to allow use of knowledge about where the region of fast variation is located (as many special purpose methods do) COLSYS has an option for choosing an initial mesh and repeatedly halving it until the tolerances are satisfied. Doing so for $\varepsilon = 10^{-10}$, $TOL(u) = 10^{-7}$, $TOL(u') = 10^{-2}$, and the initial mesh -1, -.1, -.01, -.001, -.0001, -.00001, 0, .00001, .0001, .001, .01, .1, 1 has resulted in a final mesh of 384 subintervals, with E(u) = .30-8, est E(u) = .30-8, E(u') = .61-2, est E(u') = .98-2. A large amount of storage, however, is necessary, and special methods such as expansion techniques [21], [22] for singular perturbation problems with very small ε will obviously be superior in many situations.

Example 2 [3]

$$.1176y_1'' - 52.59072y_1' - 4y_1 + 4cy_2 = 0$$
$$1.8225y_2'' + 32.67135y_2' - 4cy_2 + 4y_1 = 0 \qquad 0 \le x \le 8$$

$$52.59072y_1(0) - .1176y_1'(0) = 10,\ y_2'(0) = 0$$

$$2.629536y_1(8) - 32.67135y_2(8) - 1.8225y_2'(8) = 0$$

$$y_1'(8) = 0$$

$$c = 7.243657224749792$$

Solution: $u_1(x) = a_1 + a_2e^{b_1x} + a_3e^{b_2x} + a_4e^{b_3x-b_4}$

$$u_2(x) = a_5u_1''(x) + a_6u_1'(x) + a_7u(x)$$

$a_1 = .19016221835237111$, $a_2 = -1.4675293400578486 \times 10^{-5}$

$a_3 = 2.6341295691033977 \times 10^{-8}$, $a_4 = 1.2408745047670897 \times 10^{-5}$

$a_5 = -4.0587236193350687 \times 10^{-3}$, $a_6 = 1.8150612521116087$,

$a_7 = 1.380518134032868 \times 10^{-1}$

$b_1 = .77395646951204557$, $b_2 = -18.776674645942983$, $b_3 = 447.27606412369653$

$b_4 = 3578.2085129895722$

This solution is only accurate to the 7-th, 5-th, 8-th and 8-th digits after the decimal point in u_1, u_1', u_2 and u_2', respectively. The problem arises in the study of coal gasification, and the solutions are oscillatory through [0,8] with large derivatives near $x = 8$ (and to a lesser extent near $x = 0$). In table 2 we list results for $k = 4$ with (i) all tolerances $= 10^{-4}$, (ii) $TOL(u_1) = TOL(u_2) = 10^{-6}$ and no tolerances on the derivatives.

Table 2 - example 2

	$E(u_1)$	$estE(u_1)$	$E(u_1')$	$estE(u_1')$	$E(u_2)$	$estE(u_2)$	$E(u_2')$	$estE(u_2')$	time	mesh sequence
(i)	.67-5	.10-6	.12-2	.75-4	.39-7	.15-8	.56-6	.85-7	15.88	2,4,8,16,32, 16,32
(ii)	-	.19-12	-	.12-8	-	.68-11	-	.58-9	30.06	2,4,8,16,32 16,32,16,32, 16,32

For the second case, no significant digits of the exact error are available. For this case, the final mesh (N = 32) has 12 of its points in [7.5, 8.0] and 10 in [7.96, 8.0].

Note that the error estimates are off by about a factor of 10, and the desired tolerance for $u_1'(x)$ is not quite achieved. This is probably because few significant digits have been computed (the solutions u_1, u_1', u_2, and u_2' are of magnitude .2, .005, .03, and .009 respectively). In such cases the assumptions about the asymptotic form of the error, upon which the error estimate is based, can fail to hold.

For our rough comparison, some of the examples were run with two other general purpose codes, SUPORT and PASUNM. SUPORT, an initial value type code based on superposition with orthonormalization, was developed by Scott and Watts to solve linear two-point boundary value problems [36]. Unless stated otherwise, SUPORT uses the initial value solver RKF. PASUNI, a finite difference code using deferred corrections and a uniform mesh to solve nonlinear multi-point problems, was developed by Lentini and Pereyra [25], who later modified it to make the linear system solver more reliable. Christiansen and Russell have subsequently made further modifications to the code which can typically halve the run times; this last version we call PASUNM.

Both SUPORT and PASUNM require converting problems to first order systems and use a mixed absolute and relative error tolerance which is the same for all solution components. (In the experiments below we use only the absolute tolerance.) SUPORT has no a posteriori error estimation procedure (in fact the local error control tolerance often does not indicate what the global error is), while PASUNM produces very reliable error estimates at the equally spaced mesh points. Moreover, if SUPORT fails it does not produce any intermediate results. On the other hand, usually SUPORT requires much less storage than PASUNM and COLSYS. The fact that PASUNM works only with uniform meshes both limits its scope of applicability and makes the code perform

very efficiently when it is successful. In many cases, when PASUNM falls short of solving a problem, it is because the allotted storage is exceeded. Lentini and Pereyra have produced a variable mesh solver [26] which we intend to compare in the future.

Both examples 1 and 2 above are extremely difficult for SUPORT. For $\varepsilon \leq 10^{-2}$ in example 1, SUPORT overflows using RKF. Using GERK with tolerances of 10^{-6}, SUPORT requires 23.05 seconds with $\varepsilon = 10^{-2}$ and 284.47 seconds with $\varepsilon = 10^{-3}$. In example 2 SUPORT faces a stiff initial value problem, resulting with extremely small integration steps (e.g., with TOL = 10, a very accurate solution was computed using 183 orthonormalizations and 173.04 seconds of execution time; with TOL = 10^{-2}, nothing was produced after 8½ minutes). PASUNM looks better here:

Table 3

Example 1

ε	E(u)	estE(u)	E(u')	estE(u')	TOL (all components	time	mesh sequence
.1-1	.63-7	.55-7	.89-6	.76-6	.10-5	2.59	5,10,20,40,80
.1-3	.97-7	.73-7	.16-4	.12-4	.10-5	19.49	5,10,20,40,80,160, 320,640*
.1-5	.36-1	.14-1	.27+3	.32+1	.10-5	14.13	5,10,20,40,80,160, 320,640*

Example 2

$E(u_1)$	$estE(u_1)$	$E(u_1')$	$estE(u_1')$	$E(u_2)$	$estE(u_2)$	$E(u_2')$	$estE(u_2')$	time	mesh sequence
.47-5	.17-4	.21-2	.73-2	-	.67-6	.21-7	.61-6	28.46	5,10,20,40,80, 160,320*

*Maximum N with 600 K bytes of total storage for PASUNM

The errors $E(u_1^{(j)})$ for PASUNM are measured at the evenly spaced mesh points

and thus may be overly optimistic about a global error; nonetheless, these are the only values available. Note that if the error for COLSYS were measured that way, only superconvergence points would be encountered.

In the results for SUPORT which follow, the errors $E(u_i^{(j)})$ are approximately obtained by measuring the error at 51 predetermined points (some of which are chosen near any regions of bad behaviour). NO denotes the number of orthonormalizations for SUPORT, if the number is > 0.

Example 3 [7]

$$y'' - 4y = 4\cosh(1)$$

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

$$u(x) = \cosh(2x-1) - \cosh(1)$$

Here we run the three codes on an easy problem. Results are listed in table 4. Considering the fact that COLSYS is designed to solve general problems on arbitrary meshes and produces an error estimate, its run time is quite satisfactory.

Example 4 [25,36]

$$y''(x) + \frac{3\varepsilon}{(\varepsilon+x^2)^2}\, y = 0 \qquad -.1 \le x \le .1$$

$$y(-.1) = -\frac{.1}{(\varepsilon+.01)^{\frac{1}{2}}} = -y(.1)$$

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

The solution to this problem has a turning point region of width $0(\sqrt{\varepsilon})$ at 0 (but does not vary exponentially there). Results are listed in table 4. For small ε, SUPORT is considerably faster than COLSYS, while PASUNM is unable to solve the problem with $\varepsilon \le 10^{-4}$ and 600 K. bytes of storage. The success of SUPORT is based on the fact that the associated

initial value problem is well conditioned. (By contrast, for the problem

$$\varepsilon y'' + xy' = 0, \quad y(-1) = -1, \quad y(1) = 1 ,$$

whose solution's graph looks similar to that of example 4 and has $y'(-1) \doteq 0$, the slightly perturbed problem with the initial conditions $y(-1) = -1$, and $y'(-1) = 0$ has the solution $y \equiv -1$. Thus, the associated initial value problem is badly conditioned and indeed SUPORT cannot solve this problem for $\varepsilon < 10^{-3}$. COLSYS on the other hand does as well as for example 1).

Table 4 - examples 3,4

Example #	code	ε	k	TOL(u)	E(u)	estE(u)	TOL(u')	E(u')	estE(u')	time	mesh sequences
3	COLSYS		4	.1-7	.17-10	.11-10	.1-7	.16-8	.21-8	1.28	2,4,8,16
3	SUPORT			.1-7	.11-10		.1-7	.24-9		.64	
3	PASUNM			.1-7	.25-8	.25-8	.1-7	.46-8	.46-8	.44	5,10,20
4	COLSYS	.1-3	3	.1-5	.16-7	.11-7	.1-3	.18-4	.16-4	7.89	8,8,16,16, 32,64,128
4	SUPORT	.1-3		.1-5	.12-6		.1-5	.10-4		.95	
4	PASUNM	.1-3		.1-3	.58	-	.1-3	.77+2	.23+0	13.20	4,8,16,32, 64,128,256, 512,1024
4	COLSYS	.1-5	5	.1-5	.22-7	.14-7	.1-3	.95-4	.39-4	13.27	8,6,12,7,14, 14,28,56,28, 56
4	SUPORT	.1-5		.1-5	.12-6		.1-5	.69-4		1.75	
4	COLSYS*	.1-7	5	.1-5	.29-9	.36-9	.1-2	.29-5	.10-4	40.40	8,4,8,4,8,16, 8,16,8,16, 9,18,36,18, 36,72,36,72, 36,72,144,72, 144
4	SUPORT	.1-7		.1-5	.36-6		.1-5	.36-2		2.65	

*initial mesh points ±.1, ±.01, ±.004, ±.001, 0.0

Table 5 - example 5

code	TOL	E(u)	estE(u)	$E(u^{(3)})$	$estE(u^{(3)})$	$E(u^{(7)})$	$estE(u^{(7)})$	time	mesh sequences
COLSYS	.1-1	.31-9	.29-9	.11-4	.79-5	.82-3	.61-3	7.8	4,8,16
SUPORT	.1-1	.12-10		.47-9		.58-4		71.56	
PASUNM	.1-1	.41-16	.42-10	.13-8	.13-8	.99-3	.11-2	36.41	4,8,16,32,64, 128,256
COLSYS	.1-3	.63-10	.40-10	.12-6	.10-6	.49-4	.74-5	14.12	4,8,16,8,16
SUPORT	.1-3	.91-12		.11-10		.43-5		179.72	
PASUNM	.1-3	.57-12	.58-12	.86-10	.92-10	.70-4	.75-4	39.13	4,8,16,32,64, 128,256

Example 5 [36]

$$y^{(8)} - 914y^{(6)} + 12649y^{(4)} - 44136y'' + 32400y = 0 \qquad 0 \le x \le 5$$

$$y^{(j)}(0) = u^{(j)}(0), \quad y^{(j)}(5) = u^{(j)}(5) \qquad j = 0,1,2,3$$

$$u(x) = e^{-x} - 2e^{-2x} + e^{-3x} .$$

This is a problem favorable to COLSYS, since it can be converted to two 4-th order problems rather than eight first order ones, (recall the restriction $m_d \le 5$ in COLSYS), and the exponential homogeneous solutions cause initial value codes difficulty. Table 5 contains the numerical results for this problem. We get NO = 10 for SUPORT in both cases. As expected, COLSYS is best for this problem. Notice also how well PASUNM does, in contrast to the results in [36] for PASUNI, the major difference being the change to Lentini and Pereyra's new system solver.

We now turn to the solution of some nonlinear problems by COLSYS.

Example 6 [20]

Perhaps the greatest relative advantage of COLSYS is in solving problems in which the coefficients in the differential equations may contain singularities.

These problems commonly arise when reducing partial to ordinary differential equations by physical symmetry [35], [20], [29]. Unlike other general purpose codes, no matching of the numerical solution to an analytic expansion in the neighborhood of a singularity is necessary.

As a simple example consider the equation

$$y'' = \frac{1}{x} y' - \left(\frac{8}{7} \right)^2 e^y , \quad y'(0) = y(1) = 0$$

which has the solution $u(x) = 2\log\left(\frac{7}{8-x^2} \right)$ [20].

Results with the initial guess $u \equiv 0$ are tabulated below

Table 6 - example 6

k	TOL(u)	E(u)	estE(u)	TOL(u')	E(u')	estE(u')	time	mesh sequence
4	.1-5	.33-8	.24-8	.1-5	.77-7	.90-7	.56	2(3),4(1)

We have also solved more complicated singular problems such as the Ginsburg-Landau equations [29] with little difficulty. This will be reported in more detail elsewhere.

Example 7 [24]

$$y_i'' = - \frac{1 + y_i'^2}{20 + y_i} , \quad \frac{100}{3} (i-1) < x < \frac{100}{3} i, \quad i = 1,2,3 ,$$

$$y_1(0) = 10, \; y_1\left(\frac{100}{3} \right) = y_2\left(\frac{100}{3} \right), \quad y_2\left(\frac{200}{3} \right) = y_3\left(\frac{200}{3} \right), \quad y_3(100) = 0 ,$$

$$\frac{y_i'\left(\frac{100i}{3} \right)}{\left[4+.2y_i\left(\frac{100i}{3} \right)\right]\left[1+y_i'\left(\frac{100i}{3} \right)^2\right]^{\frac{1}{2}}} = \frac{y_{i+1}'\left(\frac{100}{3} i \right)}{\left[4+.2y_{i+1}\left(\frac{100}{3} i \right)\right]\left[1+y_{i+1}'\left(\frac{100}{3} i \right)^2\right]^{\frac{1}{2}}} , \; i=1,2,$$

$$u_i(x) = \left[3156.25 - (x-47.5)^2\right]^{\frac{1}{2}} - 20 \quad \text{on} \quad \left[\frac{100}{3} (i-1), \frac{100}{3} i \right], \quad i = 1,2,3.$$

This problem arises in studying seismic ray tracing problems, where the ray travels through three different homogeneous materials. The nonlinear boundary conditions represent Snell's law at the interfaces. Such conditions can be treated by COLSYS when the problem is converted to a system if we map

$$[0, \tfrac{100}{3}] \to [0,1], \quad [\tfrac{200}{3}, \tfrac{100}{3}] \to [0,1], \quad [\tfrac{200}{3}, 100] \to [0,1],$$

because the resulting boundary conditions each involve a single point. This trick also works in general for nonseparated boundary conditions, with the obvious disadvantage that the size of the system of equations increases. Two sets of results are given in table 7 below, where the initial guess was $u_i \equiv 0$, $i = 1,2,3$.

Table 7 - example 7

	k	i	TOL(u_i)	E(u_i)	estE(u_i)	TOL(u_i')	E(u_i')	estE(u_i')	time	mesh sequences
(i)	3	1	.1-5	.72-7	.67-7	-	.18-4	.16-4	16.85	8(5),16(2), 8(1),16(1), 32(1)
		2	.1-5	.13-8	.18-8	-	.78-7	.72-7		
		3	.1-5	.17-6	.14-6	-	.77-4	.63-4		
(ii)	4	1	.1-5	.21-9	.16-9	.1-5	.56-7	.66-7	26.37	8(5),16(1), 11(1),22(1), 44(1).
		2	.1-5	.12-9	.44-11	.1-5	.32-9	.33-9		
		3	.1-5	.33-9	.26-9	.1-5	.16-6	.19-6		

Example 8 [27]

The last example arises when considering the flow between two counter-rotating infinite plane disks. The equations which describe the motion can be cast into the form

$$\varepsilon G'' + HG' - H'G = 0$$
$$\varepsilon H^{iv} + HH''' + GG' = 0 \qquad -1 \le x \le 1$$

and

$$G(-1) = \mu, \quad G(1) = 1, \quad H(-1) = H'(-1) = H(1) = H'(1) = 0$$

with $\mu = -1$ for the case where the disks are counter-rotating at the same speed. In this latter case, there exists an odd solution [27]. This solution has boundary layers near both ends and varies smoothly in between.

Various investigators have computed solutions to this problem (see references in [27], [28]) with conflicting results. The problem becomes very ill-conditioned for small values of $\varepsilon > 0$. Except for [28], solutions have been computed for $\varepsilon \ge 10^{-3}$. In [28], the antisymmetry for the particular case $\mu = -1$ is used to solve the problem on the half interval [0,1] with $G(0) = 0$ and $H(0) = H''(0) = 0$ replacing the boundary conditions at -1. This improves the condition of the problem significantly, enabling the authors to obtain solutions for $\varepsilon = 10^{-4}$.

We solve the problem with $\mu = -1$ and $\varepsilon = 10^{-3}$ on $[-1,1]$, obtaining the odd solution without using antisymmetry. This enables us to demonstrate the stability and reliability of COLSYS. By comparing values of the obtained solution at points x and $-x$ (with the final mesh being nonuniform) we get an idea of how well the error estimates do.

For $k = 5$, TOL(G) = TOL(H) = TOL(H') = .1-5, and the initial guess $G = x^3$, $H = -x(x-1)^2(x+1)^2$, the results are listed in table 7. Tests based on the symmetries of the solution support the error estimates.

Table 7 - example 8

estE(G)	estE(G')	estE(H)	estE(H')	estE(H")	estE(H''')	time	mesh sequences
.78-6	.11-3	.79-8	.22-6	.15-4	.33-2	56.45	10(10),5(4), 10(5),20(2)

The computed curves of G and H are plotted in figs. 5.2 and 5.3, respectively.

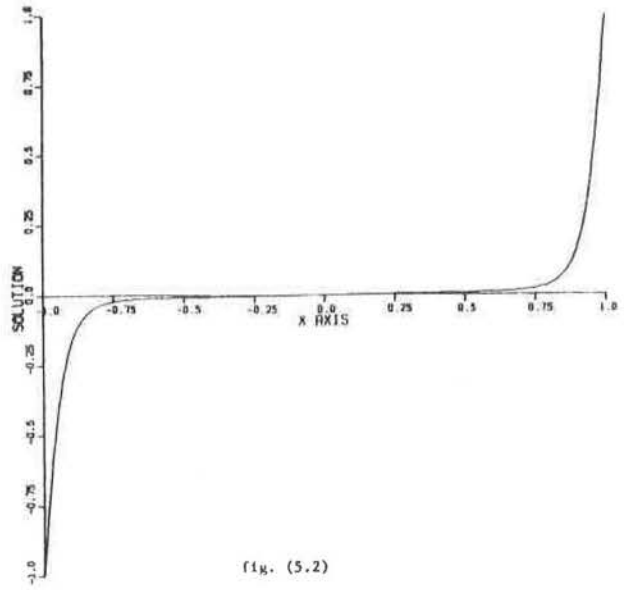

fig. (5.2)

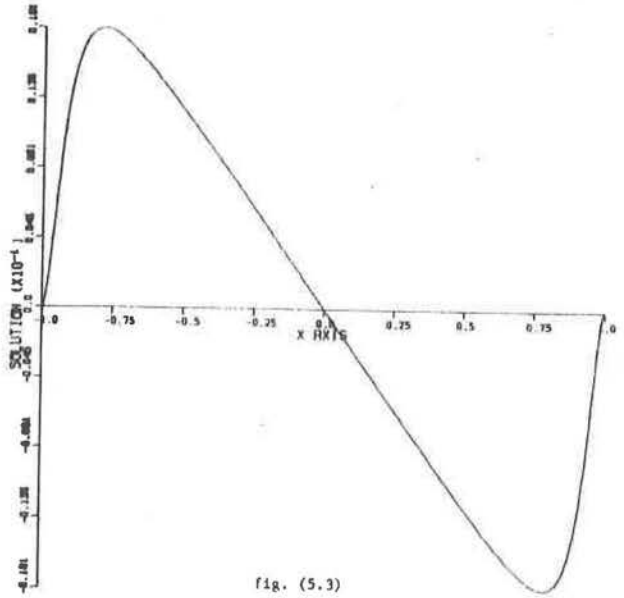

fig. (5.3)

References

1. U. Ascher, Discrete least squares approximations for ordinary differential equations, to appear in SIAM J. Numer. Anal.

2. U. Ascher and R.D. Russell, Evaluation of B-splines for solving systems of boundary value problems, Comp. Sci. Tech. Rep. 77-14, Univ. of British Columbia, 1977.

3. D. Billingsly, Experience and problems concerning numerical solution for linear o.d.e. two-point B.V. problem, IBM internal memorandum, Palo Alto, Jan. 1970.

4. C. de Boor, On calculating with B-splines, J. Approx. Th. 6 (1972), 50-62.

5. C. de Boor, Good approximation by splines with variable knots II, Springer Lecture Notes in Math., 363 (1973).

6. C. de Boor, Package for calculating with B-splines, SIAM J. Numer. Anal. 14 (1977), 441-472.

7. C. de Boor and B. Swartz, Collocation at Gaussian points, SIAM J. Numer. Anal. 10 (1973), 582-606.

8. C. de Boor and R. Weiss, Solveblok: a package for solving almost block diagonal linear systems, with applications to spline approximation and the numerical solution of ordinary differential equations, MRC TSR #1625 (1976), Madison, Wisconsin.

9. C. Broyden, Recent developments in solving nonlinear, algebraic systems, in Numerical Methods for Nonlinear Algebraic Equations, ed. by P. Rabinowitz, Gordon Beach, 1977.

10. R. Bulirsch, J. Stoer and P. Deuflhard, Numerical solution of nonlinear two-point boundary value problems I, Numer. Math. Handbook Series Approximation (1976).

11. J. Cerutti, Collocation for systems of ordinary differential equations, Comp. Sci. Tech. Rep. 230, Univ. Wisconsin-Madison, 1974.

12. J. Christiansen and R.D. Russell, Error analysis for spline collocation methods with application to knot selection, to appear in Math. Comp.

13. P. Ciarlet, M. Schultz, and R. Varga, Numerical methods of high-order accuracy for nonlinear boundary value problems I. One dimensional problem, Numer. Math. 9 (1967), 394-430.

14. J. Daniel and A. Martin, Implementing deferred corrections for Numerov's difference method for second-order two-point boundary-value problems, to appear in SIAM J. Numer. Anal.

15. J. Dennis and J. Moré, Quasi-Newton methods, motivation and theory, SIAM Review 19 (1977), 46-89.

16. P. Deuflhard, A relaxation strategy for the modified Newton method. (1975), 59-73. In Conference Proceedings on Optimization and Optimal Control, Bulirsch, Oettli and Stoer (eds.) Lecture Notes 477.

17. H.J. Diekoff, P. Lory, H.J. Oberle, H.J. Pesch, P. Rentrop and R. Seydel, Comparing routines for the numerical soltuion of initial value problems of ordinary differential equations in multiple shooting, Numer. Math. 27 (1977), 449-469.

18. D. Dodson, Optimal order approximation by polynomial spline functions, Ph.D. thesis, Purdue Univ., 1972.

19. R. England, N. Nichols and J. Reid, Subroutine DOO3AD, 1973, Harwell subroutine library. Harwell, England.

20. S.C. Eisenstadt, R.S. Schreiber and M.H. Schultz, Finite element methods for spherically symmetric elliptic equations, Res. Rept. #109, Comp. Sc., Yale Univ., 1977.

21. J.E. Flagherty and R.E. O'Malley, Jr., The numerical solution of boundary value problems for stiff differential equations, Math. Comp. 31 (1977), 66-93.

22. P. Hemker, A Numerical Study of Stiff Two-point Boundary problems, Math. Centrum, Amsterdam, 1977.

23. E. Houstis, A collocation method for systems of nonlinear ordinary differential equations, to appear in J. Math. Anal. & Appl.

24. W.H.K. Lee and V. Pereyra, Solving two-point seismic ray-tracing problems in a heterogeneous medium: part 2, preprint 1977.

25. M. Lentini and V. Pereyra, A variable order finite difference method for nonlinear multipoint boundary value problems, Math. Comp. 28 (1974), 981-1004.

26. M. Lentini and V. Pereyra, An adaptive finite difference solver for nonlinear two point boundary problems with mild boundary layers, SIAM J. Numer. Anal. 14 (1977), 91-111.

27. J.B. McLeod and S.V. Parter, On the flow between two counter-rotating infinite plane disks, Arch. Rat. Mech. Anal. 54 (1974), 301-327.

28. H.J. Pesch and P. Rentrop, Numerical solution of the flow between two counter-rotating infinite plane disks by multiple shooting, Rep. #7621, Technische Universität München, 1976.

29. P. Rentrop, Numerical solution of the singular Ginzburg-Landau equations by multiple shooting, Computing 16 (1976), 61-67.

30. R.D. Russell, Collocation for systems of boundary value problems, Numer. Math. 23 (1974), 119-133.

31. R.D. Russell, Efficiencies of B-splines methods for solving differential equations, Proc. Fifth Conference on Numerical Mathematics, Manitoba (1975), 599-617.

32. R.D. Russell, A comparison of collocation and finite differences for two-point boundary value problems, SIAM J. Numer. Anal. 14 (1977), 19-39.

33. R.D. Russell and J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, to appear in SIAM J. Numer. Anal.

34. R.D. Russell and L.F. Shampine, A collocation method for boundary value problems, Numer. Math. 19 (1972), 1-28.

35. R.D. Russell and L.F. Shampine, Numerical methods for singular boundary value problems, SIAM J. Numer. Anal. 12 (1975), 13-36.

36. M.L. Scott and H.A. Watts, Computational solutions of linear two-point boundary value problems via orthonormalization, SIAM J. Numer. Anal. 14 (1977), 40-70.

37. M.L. Scott and H.A. Watts, A systematized collection of codes for solving two-point boundary-value problems, in Numerical Methods for Differential Systems (1976), Academic Press, pp. 197-227.

38. J.M. Varah, Alternate row and column elimination for solving certain linear systems, SIAM J. Numer. Anal. 13 (1976), 71-75.

39. K. Wittenbrink, High order projection methods of moment and collocation type for nonlinear boundary value problems, Computing 11 (1973), 255-274.