ON COLLOCATION IMPLEMENTATION FOR SINGULARLY PERTURBED TWO-POINT PROBLEMS

By

Uri Ascher and Simon Jacobs

Technical Report 86-19

November 1986

Abstract

We consider the numerical solution of singularly perturbed two-point boundary value problems in ordinary differential equations. Implementation methods for general purpose solvers of first order linear systems are examined, with the basic difference scheme being *collocation* at Gaussian points. Adaptive mesh selection is based on localized error estimates at the *collocation points*. These methods are implemented as modifications to the successful collocation code COLSYS, which was originally designed for mildly stiff problems only. Efficient high order approximations to extremely stiff problems are obtained, and comparisons to COLSYS show that the modifications work relatively much better as the singular perturbation parameter gets small (i.e. the problem gets stiff), for both boundary layer and turning point problems.

1 Introduction

During the last decade a considerable amount of effort has been devoted to the numerical solution of singular perturbation problems in ordinary differential equations. The most prominent feature of such problems is one or more thin transition layers in which the profile of the analytic solution contains very steep gradients. Corner, shock, and boundary layer type features are commonly observed. These problems can pose significant numerical difficulty, yet they arise frequently in many practical applications, such as semiconductor theory, fluid dynamics, seismology, and nonlinear mechanics. Often in such problems we may expect some higher order derivatives to be multiplied by a small parameter, which we will call ε.

When discretizing a singularly perturbed problem, the large local variation of the derivatives usually suggests a highly nonuniform mesh that is fine where the solution varies rapidly, and coarse where the solution is smooth. If we call h the largest mesh stepsize used over the entire domain, then $\varepsilon \ll h$, i.e. h cannot, for efficiency reasons, be chosen small with respect to the problem coefficient ε. This raises theoretical difficulties because the usual asymptotic theory, which is valid when h can be chosen "sufficiently small", no longer holds in practice. Hence we are confronted with two major difficulties: The choice of a suitable discretization method and a theory to support its use; and the construction of an appropriate nonuniform mesh.

In this paper we restrict our attention to boundary value problems (BVPs) of the singularly perturbed type. We are interested in the solution of linear first order systems

and concentrate our efforts on implementation methods for general purpose codes.

One basic difficulty with designing suitable discretization methods for this class of problems is that the differential operators considered give rise to various modes with very different behaviour types: We may have rapidly increasing and rapidly decreasing fundamental solutions, as well as "harmless" slow ones. (Another problematic mode type, that of rapidly oscillatory, non-decaying solutions, is excluded from consideration in this work.)

There are two general approaches to solve numerically problems which contain markedly different mode types mixed together. One approach is to apply first a transformation to *decouple* the component types from one another. Following such a decoupling, separate discretization schemes, suitable for the different solution types, can be applied. For instance, decoupling "differential" and "algebraic" solution components in a system of differential/algebraic equations (DAEs) may allow the use of separate *nonstiff* discretizations for the numerical solution of such a problem (see, e.g. März [22]). Another instance is the decoupling of rapidly oscillatory solution components in a stiff BVP from other components, allowing recovery of a smooth solution (e.g. Kreiss [20]) or again treating different component types separately (e.g. Ascher & Spudich [7]).

A third instance is the decoupling of rapidly increasing and rapidly decreasing modes for a BVP like those under consideration here. This yields stiff initial value problems (IVPs) (in appropriate directions) only, and subsequently allows the use of (upwinded) one-sided difference schemes (like BDF, Gear [16]) which produce approx-

imate modes preserving the decay of the analytical ones. A number of methods for performing such a decoupling, followed by an appropriate one-sided integration, have been recently proposed (e.g. see Kreiss, Nichols & Brown [21], Brown & Lorenz [14], Dieci & Russell [15], Meyer [23]).

The second general approach to solve problems with mixed solution types is to attempt to find numerical schemes which are capable, at least to some acceptable degree, of simultaneously handling the various solution types, thus eliminating the need for an explicit decoupling transformation. Since an explicit decoupling transformation is generally expensive to calculate and at times tricky to find (especially for nonlinear problems; see the discussion in Ascher & Weiss [10]), a scheme which automatically deals with the problem is rather desirable, although not always attainable. For so-called transferrable initial value DAEs, the BDF schemes successfully fulfill the role (Gear & Petzold [17]); on the other hand, we are not aware of any corresponding general-purpose scheme for highly oscillatory BVPs.

For BVPs under consideration in this paper, piecewise polynomial collocation at (symmetric) Gaussian points is an instance of the second general approach, providing a family of schemes which have been extensively analyzed, implemented and tested (de Boor & Swartz [12], Ascher, Christiansen & Russell [5], Ascher & Weiss [8], [9], [10], Weiss [27], Ascher [1], Ascher & Bader [4]). The schemes do not need a preliminary decoupling transformation, so they can be applied efficiently, and they have proved to work very well in many practical applications. The general-purpose code COLSYS (Ascher, Christiansen & Russell [6]) implements these schemes. At the same time,

they are theoretically less satisfactory when applied to a stiff IVP than stiffly stable one-sided schemes (like BDF, or collocation at Radau points) are. This is typical for all symmetric difference schemes. It is not surprising therefore to observe that many theoretical papers for stiff BVPs in the literature deal with (upwinded) one-sided schemes, while most practical calculations use symmetric ones.

In this paper we propose an improved algorithm for adaptive mesh selection using collocation at Gaussian points. To help illustrate our point, we examine a simple example first.

Example 1 Consider the stable IVP for a single equation

$$\epsilon y' = -y + q(x) \qquad x > 0 \tag{1.1a}$$

$$y(0) = \alpha \tag{1.1b}$$

where $0 < \epsilon \ll 1$ and the inhomogeneity $q(x)$ is smooth and bounded by a constant of order 1. The *reduced solution* for this problem, obtained by setting $\epsilon = 0$ in (1.1a), is

$$\bar{y}(x) = q(x).$$

A well-scaled fundamental solution is $\exp(-x/\epsilon)$, and the solution y of (1.1) can be written as

$$y(x) = (\alpha - q(0)) \exp(-x/\epsilon) + \bar{y}(x) + O(\epsilon).$$

Thus, for $x \geq \epsilon \mid \ln \epsilon \mid$, $y(x) = \bar{y}(x) + O(\epsilon)$.

For this problem, two difference approximations are considered on a mesh

$$0 = x_1 < x_2 < \cdots, \qquad h_i := x_{i+1} - x_i.$$

(i) The *backward Euler scheme* is the 1st order BDF and also the collocation scheme at one Radau point. It gives

$$\frac{\varepsilon}{h_i}(y_{i+1} - y_i) = -y_{i+1} + q(x_{i+1}) \qquad i = 1, 2, \ldots$$

or

$$y_{i+1} = \frac{\varepsilon/h_i}{\varepsilon/h_i + 1} y_i + \frac{1}{\varepsilon/h_i + 1} q(x_{i+1}).$$

We therefore observe that for $\varepsilon \ll h_i$, the approximation for the fundamental solution satisfies

$$\frac{\varepsilon/h_i}{\varepsilon/h_i + 1} \approx 0 \approx \exp(-h_i/\varepsilon)$$

and

$$y_{i+1} = q(x_{i+1}) + O(\varepsilon/h_i).$$

The approximate solution improves as $\varepsilon \to 0$ and the reduced solution is retrieved in the limit.

(ii) The *midpoint scheme* is obtained by collocation at one Gauss point. It gives

$$\frac{\varepsilon}{h_i}(y_{i+1} - y_i) = -\frac{1}{2}(y_i + y_{i+1}) + q(x_{i+1/2}) \qquad x_{i+1/2} := x_i + \frac{1}{2}h_i, \quad i = 1, 2, \ldots \quad (1.2)$$

or

$$y_{i+1} = \frac{\varepsilon/h_i - 1/2}{\varepsilon/h_i + 1/2} y_i + \frac{1}{\varepsilon/h_i + 1/2} q(x_{i+1/2}).$$

We therefore observe that the approximation for the fundamental solution, while satisfying

$$\left|\frac{\varepsilon/h_i - 1/2}{\varepsilon/h_i + 1/2}\right| \leq 1$$

(and more generally A-stability; see e.g. [16]), also satisfies for $\varepsilon \ll h_i$

$$\frac{\varepsilon/h_i - 1/2}{\varepsilon/h_i + 1/2} \approx -1 \neq 0,$$

and for $1 \leq l \leq i$,

$$\mathrm{y}_{i+1} \approx -\mathrm{y}_i + 2q(x_{i+1/2}) \approx (-1)^{i-l}\mathrm{y}_l + 2\sum_{j=l}^{i}(-1)^{i-j}q(x_{j+1/2}).$$

The reduced solution is not retrieved in the limit $\varepsilon \to 0$ at mesh points. Moreover, when the local truncation error

$$\tau_j = \frac{1}{8}h_j^2\mathrm{y}''(x_{j+1/2}) + O(h_j^3)$$

is substituted for $q(x_{j+1/2})$ and when the error δ emanating from the boundary layer is substituted for y_l (this can be made small using a fine mesh in the narrow layer $0 < x < \varepsilon \mid \ln\varepsilon \mid$), the above equation gives an error estimate which is *nonlocal* and is of a possibly lower order than in the nonstiff case.

The situation is better, however, at the collocation point $x_{i+1/2}$. For $\mathrm{y}_{i+1/2} = \frac{1}{2}(\mathrm{y}_i + \mathrm{y}_{i+1})$, which is the collocation approximation at $x_{i+1/2}$, we obtain from (1.2) (using the stability, albeit marginal, of the scheme),

$$\mathrm{y}_{i+1/2} = q(x_{i+1/2}) + O(\varepsilon/h_i).$$

This gives a localized error estimate, and the reduced solution is obtained in the limit $\varepsilon \to 0$, as for the backward Euler scheme.

In particular, it is important to note the effect of the layer error δ *away* from the layer, because this error component may not be small before we obtain a good fine mesh at the layer: While at mesh points away from the layer the error has an $O(\delta)$ component, at midpoints the corresponding error component is $O(\delta\varepsilon h_i^{-1})$ only.

□

This simple example displays the basic observation underlying this paper: *When using collocation at Gaussian points for a stiff BVP, the error when $\varepsilon \ll h_i$ is better localized at collocation points than elsewhere (including mesh points).* The evaluation of the approximate solution at collocation points cannot, of course, be used to improve the stability of the scheme which generates it, but it may possibly be used to help to generate a better mesh in an adaptive mesh selection algorithm like the one used in COLSYS.

In § 2 we discuss error estimates for collocation at Gaussian points. As it turns out, the mesh selection algorithm in COLSYS, which was designed on the basis of nonstiff theory, essentially attempts to do the right thing even in the very stiff case. However the evaluation of the solution derivative required in this mesh selection algorithm is better achieved using solution values at collocation points, as discussed in § 3. An experimental implementation in the context of linear first order BVPs has been carried out, and in § 4 we demonstrate its potentially remarkable improvement over the performance of COLSYS.

2 Collocation at Gaussian points

We consider a collocation method for the BVP of order n

$$\mathbf{u}' = A(x)\mathbf{u} + \mathbf{q}(x), \qquad a < x < b, \tag{2.1a}$$

$$B_a\mathbf{u}(a) + B_b\mathbf{u}(b) = \beta. \tag{2.1b}$$

First select a set of k *canonical collocation points*, ρ,

$$0 \le \rho_1 < \rho_2 < \cdots < \rho_k \le 1$$

which are later chosen to be Gauss points (i.e. zeroes of the Legendre polynomial) on $[0,1]$. Now, given the mesh

$$\pi : a = x_1 < x_2 \cdots < x_N < x_{N+1} = b \tag{2.2}$$

$$h_i := x_{i+1} - x_i, \qquad h := \max_{1\le i\le N} h_i$$

and the points ρ, a piecewise polynomial *collocation solution* $\mathbf{u}_\pi(x)$ of (2.1) is determined such that

$$\mathbf{u}_\pi(x) \in C[a,b] \cap P_{k+1,\pi} \tag{2.3a}$$

$$\mathbf{u}'_\pi(x_{ij}) = A(x_{ij})\mathbf{u}_\pi(x_{ij}) + \mathbf{q}(x_{ij}) \qquad 1 \le i \le N \quad 1 \le j \le k \tag{2.3b}$$

$$\mathrm{B}_a(a)\mathbf{u}_\pi(a) + \mathrm{B}_b(b)\mathbf{u}_\pi(b) = \beta \tag{2.3c}$$

where

$$x_{ij} := x_i + h_i\rho_j \qquad 1 \le i \le N \quad 1 \le j \le k$$

i.e. the solution is continuous on $[a,b]$, satisfies the boundary conditions, and is on each interval of π a polynomial of degree $< k+1$ that satisfies the differential equation at the k *collocation points*, x_{ij}. It can be shown that the requirements (2.3) uniquely determine the approximate solution $\mathbf{u}_\pi(x)$, and are equivalent to a Runge-Kutta formulation (see, e.g. [9]).

Now, consider collocating a smooth, *nonstiff* BVP (2.1). The following convergence results are known to hold [2]:

Theorem 2.4 Assume that the BVP (2.1) is well posed in the sense that the conditioning constant κ (i.e. a bound on the Green's function) is of moderate size, and denote its unique solution by $\mathbf{u}$. Assume further that $A, \mathbf{q} \in C^{2k}[a,b]$, and that collocation at k Gaussian points is used.

Then, *for h sufficiently small*

(a) the collocation method has a unique solution $\mathbf{u}_\pi(x)$

(b) there exists an implementation that is stable, with condition number (for the resulting algebraic system) $\sim \kappa N$

(c) at mesh points there is *superconvergence*, i.e.

$$\|\mathbf{u}(x_i) - \mathbf{u}_\pi(x_i)\| = O(h^{2k}) \qquad 1 \le i \le N$$

(*d*) while at other points

$$\left\|\mathbf{u}^{(j)}(x) - \mathbf{u}_\pi^{(j)}(x)\right\| = \left\|\mathrm{p}^{(j)}(x)\mathbf{u}^{(k+1)}(x_i)\right\| h_i^{k+1-j} + O(h_i^{k+2-j}) + O(h^{2k})$$

$$x_i < x < x_{i+1} \qquad 1 \le i \le N \qquad 0 \le j \le k$$

$p(x)$ a known smooth function □

For adaptive mesh selection in COLSYS the expression (2.4d) is used because of the localization of its leading term. Further, at subinterval midpoints,

$$\mathbf{u}^{(k)}(x_{i+1/2}) - \mathbf{u}_\pi^{(k)}(x_{i+1/2}) = O(h_i^2) \tag{2.5}$$

so an $O(h)$ estimate of $\mathbf{u}^{(k+1)}$ is obtained by differentiating a linear interpolant of the values $\mathbf{u}_\pi^{(k)}(x_{i+1/2})$, $1 \le i \le N$. Thus, having solved on a current mesh, the code attempts to refine it by equidistributing an approximation of the leading error term in (2.4d).

The basic difficulty with mesh selection for (very) *stiff* BVPs is that neither Theorem 2.4 nor (2.5) hold (cf. [8], [9], [10], [1]). Firstly, (2.4b) is not guaranteed, i.e. the stability of the collocation method does not automatically follow from well-posedness of the BVP (2.1). This has to be assumed in addition, although the assumption rarely poses a practical restriction. Secondly, and more importantly in practice, the error is no longer localized, as indicated in Example 1. Thirdly, the superconvergence order (2.4c) drops [9].

To be more specific, consider a segment $[x_{\underline{i}}, x_{\bar{i}}]$ of $[a, b]$ where the solution is known to be smooth (i.e. it and some of its derivatives are bounded). We make the dependence

on a small parameter $0 < \varepsilon \ll 1$ explicit and assume the ODE (2.1a) to be in the form

$$\begin{aligned} \varepsilon \mathbf{y}' &= A_{11}(x;\varepsilon)\mathbf{y} + A_{12}(x;\varepsilon)\mathbf{z} + \mathbf{q}_1(x;\varepsilon) \\ \mathbf{z}' &= A_{21}(x;\varepsilon)\mathbf{y} + A_{22}(x;\varepsilon)\mathbf{z} + \mathbf{q}_2(x;\varepsilon) \end{aligned} \tag{2.6}$$

i.e.

$$A(x;\varepsilon) = \begin{bmatrix} A_{11}(x;\varepsilon) & A_{12}(x;\varepsilon) \\ \varepsilon A_{21}(x;\varepsilon) & \varepsilon A_{22}(x;\varepsilon) \end{bmatrix}$$

$$\mathbf{u}(x;\varepsilon) = \begin{pmatrix} \mathbf{y}(x;\varepsilon) \\ \mathbf{z}(x;\varepsilon) \end{pmatrix} \qquad \mathbf{q}(x;\varepsilon) = \begin{pmatrix} \mathbf{q}_1(x;\varepsilon) \\ \varepsilon \mathbf{q}_2(x;\varepsilon) \end{pmatrix}$$

where $A_{lm}, \mathbf{q}_l$ are smooth and bounded, $1 \le l,\ m \le 2$, and $A_{11}(x;0)$ has a hyperbolic splitting. Thus, there is an appropriate transformation $T(x)$ satisfying

$$T^{-1}(x)A_{11}(x;0)T(x) = \begin{pmatrix} \Lambda^-(x) & \\ & \Lambda^+(x) \end{pmatrix} \qquad x_{\underline{i}} \le x \le x_{\bar{i}} \tag{2.7}$$

with $\Lambda^-(x)$ an upper triangular matrix with diagonal elements satisfying

$$Re(\lambda_j(x)) \le \lambda_- < 0 \qquad 1 \le j \le n_-,$$

$\Lambda^+(x)$ is an upper triangular matrix satisfying

$$Re(\lambda_j(x)) \ge \lambda_+ > 0 \qquad n_- + 1 \le j \le n,$$

and

$$|Im(\lambda_j(x))| \le const\, |Re(\lambda_j(x))| \qquad 1 \le j \le n,$$

for all $x_{\underline{i}} \le x \le x_{\bar{i}}$.

Defining

$$\mathbf{w}(x) := T^{-1}(x)\mathbf{y}(x) \tag{2.8a}$$

$$\mathbf{w}(x) \equiv \begin{pmatrix} \mathbf{w}^-(x) \\ \mathbf{w}^+(x) \end{pmatrix} \qquad \mathbf{w}^- \in \mathrm{R}^{n-} \tag{2.8b}$$

we assume that the BVP consisting of (2.6) subject to

$$\mathbf{w}^-(x_{\underline{i}}), \quad \mathbf{z}(x_{\underline{i}}), \quad \mathbf{w}^+(x_{\bar{i}}) \quad \text{specified} \tag{2.9}$$

is well-conditioned. The analysis in [9] then yields (using single bars to denote maximum vector norms)

Theorem 2.10 Let the assumptions above hold for (2.6). Define $\mathbf{w}_\pi(x)$ as a transformation of the collocation solution $\mathbf{y}_\pi(x)$ using (2.8). If $\varepsilon \ll \min_{\underline{i} \le j < \bar{i}} h_j^2$ and

$$\left|\mathbf{w}^-(x_{\underline{i}}) - \mathbf{w}_\pi^-(x_{\underline{i}})\right|, \quad \left|\mathbf{w}^+(x_{\bar{i}}) - \mathbf{w}_\pi^+(x_{\bar{i}})\right| \le \delta \tag{2.10a}$$

then the error satisfies

$$|\mathbf{u}(x_i) - \mathbf{u}_\pi(x_i)| \le c_1 \left|\Sigma_{j=\underline{i}}^{\bar{i}-1} \gamma_j \|\mathbf{u}^{(k+1)}\|_j h_j^{k+1}\right| + c_2\delta + c_3 h^{2k} \qquad \underline{i} \le i \le \bar{i} \tag{2.10b}$$

where c_l are constants of moderate size, $\|\phi\|_j := \max_{x_j \le x \le x_{j+1}} |\phi(x)|$, and

$$\gamma_j = (-1)^{kj} + O(\varepsilon h_j^{-1}) \tag{2.10c}$$

□

The estimate (2.10b) for the error at mesh points is generally not better than the error at other points x, $x_{\underline{i}} \le x \le x_{\bar{i}}$.

From (2.10b) we see that if the layer regions for a given BVP are adequately resolved by a fine mesh, so that δ in (2.10a) is small, then the leading term of the error in regions

where the mesh is sparse is

$$c\Sigma_{j=\underline{i}}^{\bar{i}-1}\gamma_j \left|\mathbf{u}^{(k+1)}(x_j)\right| h_j^{k+1}.$$

This indicates that the superconvergence order (2.4c) drops and the error is no longer localized. However, COLSYS still attempts to equidistribute a relevant quantity!

Better results than (2.10b) are obtained at collocation points, as in Example 1. Consider the reduced equations at x_{ij}, $1 \le j \le k$, when $\varepsilon \ll h_i$. This gives

$$\mathbf{y}_\pi(x_{ij}) \approx -A_{11}^{-1}(x_{ij})[A_{12}(x_{ij})\mathbf{z}_\pi(x_{ij}) + \mathbf{q}_1(x_{ij})], \tag{2.11}$$

which may be substituted to eliminate $\mathbf{y}_\pi(x_{ij})$ from the nonstiff equations for $\mathbf{z}_\pi(x_{ij})$. The localized estimate (like (2.4d)) which holds for the error in the slow variables $\mathbf{z}_\pi(x_{ij})$ also gives a localized bound for the error in the fast variables $\mathbf{y}_\pi(x_{ij})$ through (2.11). This heuristic argument can be made rigorous following the analysis in [9], obtaining

Theorem 2.12 Under the assumptions of Theorem 2.10

$$|\mathbf{u}(x_{ij}) - \mathbf{u}_\pi(x_{ij})| \le c\|\mathbf{u}^{(k+1)}\|_i h_i^{k+1} + O(\delta\varepsilon h_i^{-1}) + O(h_i^{k+2}) + O(h^{2k}) \tag{2.12a}$$

$$\underline{i} \le i \le \bar{i}, \quad 1 \le j \le k.$$

□

3 Mesh selection implementation

The results in the previous section indicate that even when $\varepsilon \ll h_i$, an attempt to select a mesh which equidistributes the leading error term in (2.4d) is reasonable. A good mesh on the entire interval must still be fine in layer regions, i.e. $h_i \lesssim \varepsilon$ there, so (2.4d) does hold in some regions; however, the practical transition between this asymptotic form and (2.10b) is hard to identify. For these reasons we have kept the basic mesh selection strategy in COLSYS, as described in [5]. It remains to discuss the choice of constant in the error expression and the approximation of $\mathbf{u}^{(k+1)}(x_i)$ appearing there.

The equidistribution of a mesh for a given mesh size is not affected by the particular choice of the error constant, whose primary use is to help to intelligently estimate the mesh size needed for a given error tolerance. Since little is known about such a choice in the stiff case we use the constant for the nonstiff case, as implemented in COLSYS [5].

The estimation of $\mathbf{u}^{(k+1)}$ is much more critical, and care must be taken to find even a "good", relatively $O(1)$ approximation. If the values of $\mathbf{u}_\pi$ used for this purpose are heavily polluted by nonlocal effects (such as may emanate from a layer region with a poorly chosen mesh) then the approximation to the $(k+1)$st derivative may be extremely poor. We take advantage of the localized error estimate *at collocation points* (2.12a) when $\varepsilon \ll h_i$.

We first estimate $\mathbf{u}^{(k-1)}$ on each interval. Let $L_1(t), \ldots, L_k(t)$ be the Lagrange

interpolating polynomials of the k canonical points ρ on $[0,1]$. Then

$$\mathbf{u}_L(x) := \sum_{j=1}^{k} L_j\left(\frac{x-x_i}{h_i}\right)\mathbf{u}_\pi(x_{ij}) \qquad x_i \le x \le x_{i+1} \quad \underline{i} \le i \le \bar{\imath}-1$$

is a polynomial of order k which interpolates $\mathbf{u}_\pi(x)$ at the k collocation points on $[x_i, x_{i+1}]$. Using (2.12a),

$$\mathbf{u}(x) = \mathbf{u}_L(x) + \sum_{j=1}^{k} L_j\left(\frac{x-x_i}{h_i}\right)\left\{O(\delta\epsilon h_i^{-1}) + O(h_i^{k+1}) + O(h^{2k})\right\} + \mathrm{R}_i(x) \qquad (3.1)$$

$$x_i \le x \le x_{i+1} \quad \underline{i} \le i \le \bar{\imath}-1$$

with the remainder written using divided differences as

$$\mathrm{R}_i(x) = \mathbf{u}[x_{i1}, \ldots, x_{ik}, x]\prod_{j=1}^{k}(x - x_{ij}).$$

In general, of course, $\mathrm{R}_i = O(h_i^k)$, so $\mathrm{R}_i^{(k-1)} = O(h_i)$, but for symmetric schemes it is easy to see that

$$\mathrm{R}_i^{(k-1)}(x_{i+\frac{1}{2}}) = O(h_i^2)$$

Differentiation of (3.1) then gives

$$\mathbf{u}^{(k-1)}(x_{i+\frac{1}{2}}) = \mathbf{u}_L^{(k-1)}(x_{i+\frac{1}{2}}) + O(h_i^2) + O(\delta\epsilon h_i^{-k}) + O(h_i^{1-k}h^{2k}) \qquad (3.2)$$
$$\underline{i} \le i \le \bar{\imath}-1$$

Now, for each interval i, $\underline{i}+1 \le i \le \bar{\imath}-2$, interpolating quadratically from $\mathbf{u}_L^{(k-1)}(x_{i+\frac{1}{2}})$ on adjacent intervals and then differentiating gives the piecewise constant approxima-

tion

$$\begin{aligned}\mathbf{u}^{(k+1)}(x) =& \frac{2}{(x_{i-\frac{1}{2}} - x_{i+\frac{1}{2}})(x_{i-\frac{1}{2}} - x_{i+\frac{3}{2}})}\mathbf{u}_L^{(k-1)}(x_{i-\frac{1}{2}}) \\ &+ \frac{2}{(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}})(x_{i+\frac{1}{2}} - x_{i+\frac{3}{2}})}\mathbf{u}_L^{(k-1)}(x_{i+\frac{1}{2}}) \\ &+ \frac{2}{(x_{i+\frac{3}{2}} - x_{i-\frac{1}{2}})(x_{i+\frac{3}{2}} - x_{i+\frac{1}{2}})}\mathbf{u}_L^{(k-1)}(x_{i+\frac{3}{2}}) + \hat{\mathrm{R}}_i\end{aligned} \tag{3.3}$$

$$x_{i-1} \le x \le x_{i+2} \qquad \underline{i}+1 \le i \le \bar{\imath}-2$$

where, assuming that the rightmost two terms in (3.2) can be neglected,

$$\left\|\hat{R}_i\right\| \le K \left\|\mathbf{u}^{(k+1)}\right\|_i \tag{3.4}$$

for K a constant of order 1.

The bound (3.4) is less theoretically satisfactory than what we have for the non-stiff case, however, the computable expression obtained when ignoring $\hat{R}_i$ in (3.3) is (excluding pathological examples) of the order of magnitude of $\left|\mathbf{u}^{(k+1)}(x_i)\right|$. Recall that from the point of view of error equidistribution, we are really interested in some indication of the relative magnitudes of the error only.

The approximation to $\mathbf{u}^{(k+1)}$ and the error coefficient are both piecewise constant, hence are easily used to generate new equidistributed meshes.

In practice, (3.3) usually works well. However, difficulty is encountered when the quadratic interpolation is applied for a mesh interval where h_{i-1}, h_i, h_{i+1} differ by many orders of magnitude. The error estimate obtained becomes large in magnitude relative to that from other intervals and often dominates over the error estimates for

all intervals. Usually, when this problem occurs we simply apply the interpolation in a one-sided way (as at the endpoints, normally). Difficulty arises, however, when no three consecutive intervals of similar magnitude are available to make even the one-sided estimation. If there are only two intervals of comparable magnitude we can, similarly to (3.3), estimate $\mathbf{u}^{(k)}$ on two intervals and then linearly interpolate. The necessary $(k+1)^{st}$ collocation point is taken from the adjacent interval, and an error term like $O(\frac{h_{i+1}^k}{h_i^k})$ results from the differentiation to $\mathbf{u}^{(k+1)}$ (ignoring the contribution from global terms). Since we assume $h_{i+1} \sim h_i$ in this case, this causes no difficulty.

When only one interval is available (i.e. when the ratios of h_{i-1} to h_i and h_i to h_{i+1} vary considerably or a boundary interferes), no reasonable approximation is at hand, and we cannot resolve the problem. To prevent this situation, an additional mesh point is placed in such a case at the midpoint of $[x_i, x_{i+1}]$ when the mesh is created in order that a sufficient number of intervals will always exist for further redistribution.

While the procedure for estimating $\mathbf{u}^{(k+1)}$ is certainly less elegant than might be desired, it is the best that we are currently aware of. Other more general approaches have been examined. One idea is to equidistribute a lower derivative, say $\mathbf{u}^{(k)}$, instead of $\mathbf{u}^{(k+1)}$ (§3 in Russell and Christiansen [26]), for which an $O(h)$ approximation exists. However, experience shows that this is inferior in practice because the code does not detect and respond to trouble areas as well as with the above method. Another idea is to estimate $\mathbf{u}^{(k+1)}$ directly, by interpolating from k points on one interval and 2 points from adjacent intervals, and then differentiating. The problem with this idea

is that the $O(1)$ term of the error in estimating $\mathbf{u}^{(k+1)}$ behaves instead like $O\left(\frac{h_{i+1}^{k+1}}{h_i^{k+1}}\right)$ (ignoring the contribution from global terms), and creates a much more severe problem for adjacent intervals of differing orders of magnitude than with the method used.

Note that the above considerations hold only for $\varepsilon \ll h$. The main advantage of the proposed algorithm is in the improved error estimate and mesh selection that it offers away from layers, where we expect the assumption $\varepsilon \ll h$ to hold. But if we now assume $h \lesssim \varepsilon$, we get the nonstiff error estimates at collocation points. Therefore, as the algorithm succeeds in "regularizing" the problem in a transition layer, it still equidistributes the mesh in an appropriate way. Observe, however, that the equidistribution when $h \ll \varepsilon$ is inferior to that of COLSYS, which equidistributes on the basis of a nonstiff problem.

4 Numerical examples

The a posteriori mesh selection considerations of the previous sections have been incorporated into an experimental adaptive collocation code which we refer to as RKNEW. The program integrates first order linear systems with separated two-point boundary conditions by implementing a symmetric Runge-Kutta solution representation (see §4 of Bader & Ascher [11] and Jacobs [19]). RKNEW has been tested on a large number of very stiff problems, with both boundary layers and turning points, and we now examine three representative examples, all of the singularly perturbed type. In some instances, comparisons are made with the most recent version of COLSYS, called COLNEW [11].

Differential systems are converted to first order before integration (even for COLNEW, which can integrate higher order equations directly), a limit of 500 mesh intervals is placed on any mesh generated by either code, and the error tolerance is imposed on all components of the first order system. Both codes were run in double precision ($\approx$ 15 decimal digits) on a VAX 750, using the Fortran 77 compiler of BSD 4.2 UNIX. Run time comparisons are not meaningful given the preliminary state of RKNEW, however, the mesh selection procedure of RKNEW is slightly more costly because, unlike COLNEW, the solution must be evaluated at the k collocation points on each interval of the mesh every time a mesh redistribution is possible. Since the cost to find an approximation on a mesh of size N is $O(N)$, the best indication of the relative work associated with finding a solution for a given ϵ is to compare the total number of mesh

intervals required in all meshes generated by each code. The largest N encountered serves as a relative measure for storage.

The following notation will be used:

u_l - l^{th} component of the exact solution

$E(u_l)$ - maximum of error in u_l calculated at

x_i, $x_{i+\frac{1}{3}}$, $x_{i+\frac{2}{3}}$ for each mesh interval i, $1 \le i \le N$

δ - mixed absolute and relative error tolerance

k - number of collocation points per mesh interval

N - number of mesh intervals

Mesh Sequence - sizes of successive meshes generated to meet tolerances

Ntot - total number of mesh intervals in the Mesh Sequence

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

$*$ - an extra point was added to make the mesh sufficiently locally uniform.

Example 2 A Turning Point, ex. 3.7.4 in Hemker [18], ex. 1 in [5]. The BVP

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

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

is converted to first order by $u_1 := y$, $u_2 := y'$, with solution

$$u_1(x) = \cos(\pi x) + \frac{\operatorname{erf}\left(\dfrac{x}{(2\varepsilon)^{\frac{1}{2}}}\right)}{\operatorname{erf}\left(\dfrac{1}{(2\varepsilon)^{\frac{1}{2}}}\right)}$$

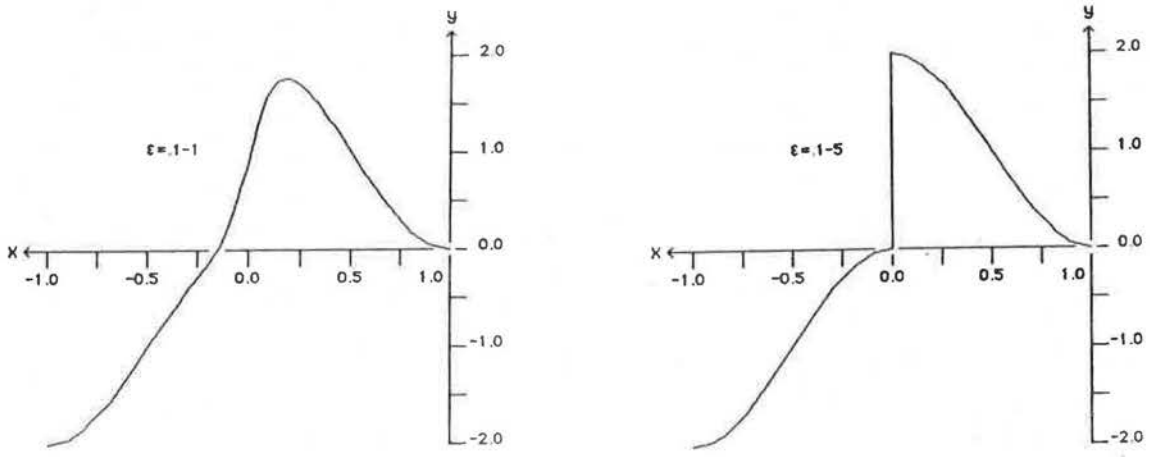

Figure 1: The solution for Example 2

As $\varepsilon \to 0$ the solution develops a shock layer in the turning point region near $x = 0$ and is smooth elsewhere (Figure 1). To find a cheap, accurate approximation to this problem, each code must recognize the shock and place a sufficiently dense mesh around it, using as few mesh intervals as possible (i.e. N small). Results for COLNEW and RKNEW are given in Table 1 for an initially uniform mesh.

As ε gets smaller, the problem gets more difficult to solve and more mesh points are required. It is clear that for large ε there is little difference in the performance, but as the problem gets harder, RKNEW is far superior, finding adequate meshes of relatively small size for extremely small values of ε. The size of the final mesh is important because it determines the amount of storage used, and it also reflects how well the code has detected and responded to difficulties early in the mesh selection process; once a particular size, $\hat{N}$ say, has been reached, further meshes are always

			COLNEW				RKNEW			
ε	k	δ	$E(u_1)$	$E(u_2)$	Ntot	Mesh Sequence	$E(u_1)$	$E(u_2)$	Ntot	Mesh Sequence
.1-1	4	.1-1	.52-4	.79-3	24	8,16	.52-4	.79-3	24	8,16
.1-1	4	.1-5	.89-7	.59-6	119	8,16,32,21,42	.15-6	.37-6	132	8,16,16,32,20,40
.1-3	4	.1-5	.12-7	.49-6	426	8,16,32,64,51,102, 51,102	.58-7	.37-6	312	8,16,16,16,32,32,64, 128
.1-5	4	.1-5	.30-9	.27-7	1352	8,16,16,32,32,32,64, 64,64,128,128,128, 256,128,256	.83-7	.37-6	474	8,16,16,16,32,32,32, 64,43,86,43,86
.1-6	4	.1-5				8,16,16,32,32,32,64, 64,64,128,128,128, 256,256,256,>500	.15-6	.24-5	406	8,16,16,16,32,32,32, 64,64,42,84
.1-8	4	.1-5					.60-7	.77-7	942	8,16,16,16,32,32,32, 64,60,120,111,222, 111,222
.1-11	4	.1-5					.59-7	.24-6	1263	8,16,16,16,32,32,32, 64,39,78,53,106,85, 170,86,172,86,172

Table 1: Results for Example 2 with a uniform initial mesh

of size $\geq \frac{\hat{N}}{2}$. Note how both codes redistribute the mesh several times early in each mesh sequence, indicating how quickly the difficulty has been detected. For the final mesh of RKNEW at $\varepsilon = .1-11$, 144 of 172 mesh intervals are in $(-.1-4, .1-4)$, $\max_{1\leq i\leq 172} h_i = .69-1$, $\min_{1\leq i\leq 172} h_i = .15-6$, and $\frac{h_{159}}{h_{158}} = .85+5$. RKNEW does not have difficulty dealing with the locally nonuniform mesh interval sizes. □

Example 3 A Boundary Layer, ex. 1.3.2 in [18]. The BVP

$$\varepsilon\, y'' + y' = 0 \qquad 0 < x < \frac{1}{4}$$

$$y(0) = 1 \qquad y\left(\frac{1}{4}\right) = \exp\left(-\frac{1}{4\varepsilon}\right)$$

is converted to first order by $u_1 := y$, $u_2 := y'$, with solution

$$u_1(x) = \exp\left(-\frac{x}{\varepsilon}\right)$$

As $\varepsilon \to 0$ the solution forms a steep boundary layer of width $O(\varepsilon)$ at $x = 0$ and is smooth elsewhere; there are no turning points. Results for an initially uniform mesh

			COLNEW				RKNEW			
ε	k	δ	$E(u_1)$	$E(u_2)$	Ntot	Mesh Sequence	$E(u_1)$	$E(u_2)$	Ntot	Mesh Sequence
.1-1	5	.1-1	.10-3	.88-3	14	5,3,6	.12-3	.11-2	14	5,3,6
.1-1	5	.1-3	.80-6	.27-5	33	5,4,8,16	.25-5	.96-5	26	5,3,6,12
.1-1	5	.1-5	.12-6	.34-6	44	5,5,10,8,16	.52-8	.13-7	69	5,5,10,7,14,28
.1-3	5	.1-5	.97-9	.27-7	279	5,10,20,40,34,68,34,68	.41-8	.25-4	295	5,10,20,40,40,20,40, 20,40,20,40
.1-4	5	.1-5	.26-10	.20-9	955	5,10,20,40,80,160,160, 80,160,80,160	.11-11	.17-9	1206	5,10,20,40,80,160,99, 198,99,198,99,198
.5-5	5	.1-5	.53-9	.12-8	1949	5,10,20,40,80,160,320, 219,438,219,438	.61-7	.17-6	1595	5,10,20,40,80,160,320, 160,320,160,320
.1-5	5	.1-5				5,10,20,40,80,160,320, >500				5,10,20,40,80,160,320, >500

Table 2: Results for Example 3 with a uniform initial mesh

are given in Table 2. There is no evidence to suggest that one code is superior over the other. Note that for $\varepsilon = .1 - 5$, both codes fail to locate the problem and keep doubling past the arbitrary 500 mesh point limit. In fact, as ε gets smaller, more consecutive doublings of the initial mesh are required before equidistribution is even attempted. The difficulty is that as $\varepsilon \to 0$ the layer gets thinner, and while coarse meshes "see" the effect as a large global error, they are unable to pinpoint the location of the disturbance; the layer is simply "skipped over".

This is easy to understand for the algorithm proposed here. As in Example 1, when $\varepsilon \ll h_i$ the approximate solution values at collocation points are essentially equal to the reduced solution values, and the latter do not vary faster in the layer region than away from it. Hence the code has "no reason" to concentrate mesh points in the layer region. (The situation is similar to that for one sided schemes, cf. [21].) We emphasize that the advantage offered by our algorithm as compared to that in COLSYS is in exploiting the accurate solution values at collocation points *away* from layers. Note also that the error estimates in the code are calculated at points other than collocation

				COLNEW				RKNEW			
ε	α	k	δ	E(u_1)	E(u_2)	Ntot	Mesh Sequence	E(u_1)	E(u_2)	Ntot	Mesh Sequence
.1-3	.1-1	5	.1-1	.29-8	.93-6	258	5,5,10,20,40,40, 23,46,23,46	.11-6	.37-6	156	12,24,20,40,20,40
.1-4	.1-2	5	.1-3	.17-9	.39-9	1220	5,5,10,20,40,80, 160,150,300,150, 300				
.1-5	.1-1 .1-2 .1-3	5	.1-5				5,10,20,40,80, 160,320,>500	.26-10	.40-7	654	6*,12,24,48,96,52, 104,52,104,52,104
.1-7	.1-4	5	.1-5					.26-10	.41-7	762	6*,12,24,48,96,64, 128,64,128,64,128
.1-9	.1-6	5	.1-5					.27-10	.42-7	870	6*,12,24,48,96,76, 152,76,152,76,152
.1-11	.1-8	5	.1-5					.27-10	.43-7	978	6*,12,24,48,96,88, 176,88,176,88,176

Table 3: Results for Example 3 with a simple nonuniform initial mesh

points.

An alternative is to use the knowledge that a boundary layer at $x = 0$ actually exists (from a solution for a larger ε, or by some analytic means), and to choose the initial mesh so that the codes can detect the layer. A surprisingly simple way is to choose a new initial mesh, $\pi_\alpha := \{0.0, \alpha, 2\alpha, 3\alpha, 4\alpha, 0.25\}$, where α is some small parameter that controls the initial concentration of mesh points around the layer. Results for smaller ε with this nonuniform initial mesh are given in Table 3. Various values of α were tried for the initial COLNEW mesh at $\varepsilon = .1 - 5$, but none was found to generate a solution in under 500 mesh intervals. In all instances the initial mesh was adjusted a little for small N and then simply doubled past 500 intervals. With RKNEW, on the final mesh at $\varepsilon = .1 - 11$, 174 of 176 intervals were in $(0.0, .1 - 9)$, and $\frac{h_{175}}{h_{174}} = .62 + 11$. It is interesting to note that after the first mesh of 176 intervals, the maximum error is still $\sim .1 + 9$, and that after only two mesh redistributions, it is reduced to $\sim .1 - 6$. This quick improvement is a common feature of both codes, but is more pronounced

for RKNEW.

Another option is to use the exponential meshes of [8], [9], [1] as initial meshes. We have experimented with these, but found that while their performance is far superior to the initially uniform meshes, it is also rather inferior to the performance of the very crude nonuniform meshes, π_α. To understand this, note first that at the initial stages of our adaptive mesh selection algorithm the recognition of a fast solution variation in a layer region is more important than making the error size very small everywhere. Secondly, COLSYS and its derivatives tested here measure a global error, while the exponential mesh of [8] relies on superconvergence at mesh points. □

Example 4 A Turning Point and Boundary Layer, ex. 2 in Brown & Lorenz [14], reads

$$-\varepsilon\, y'' - \frac{x}{2}\, y' + \frac{x}{2}\, z' + z = \varepsilon\pi^2 \cos(\pi x) + \frac{\pi}{2}\, x \sin(\pi x) =: g(x) \qquad -1 < x < 1$$

$$-\varepsilon\, z'' + z = 0$$

$$y(-1) = -1 \qquad z(-1) = 1 \qquad y(1) = z(1) = \exp\left(-\frac{2}{\varepsilon^{\frac{1}{2}}}\right)$$

This is converted to first order by introducing the variables w and v such that

$$\varepsilon\, z' = v$$

$$w' = \frac{1}{2}\, y + \frac{1}{2}\, x\, v + z - g$$

and $u_1 := y$, $u_2 := v$, $u_3 := w$, $u_4 := z$. We get, upon integrating,

$$\begin{aligned}
\varepsilon\, u_1' &= -\frac{x}{2}\, u_1 + \frac{1}{2}(\varepsilon - 1)u_2 + u_3 + (1-\varepsilon)\frac{x}{2}\, u_4 \\
u_2' &= u_4 \\
u_3' &= \frac{1}{2}\, u_1 + \frac{x}{2}\, u_2 + u_4 - g \\
\varepsilon\, u_4' &= u_2
\end{aligned}$$

under appropriate boundary conditions. The exact solution is

$$u_1(x) = \frac{\operatorname{erf}\left(\dfrac{x}{2\varepsilon^{\frac{1}{2}}}\right)}{\operatorname{erf}\left(\dfrac{1}{2\varepsilon^{\frac{1}{2}}}\right)} + u_4 + \cos(\pi x) \qquad u_4(x) = \exp\left(-\frac{(x+1)}{\varepsilon^{\frac{1}{2}}}\right)$$

Under the given boundary conditions, the solution for z has a boundary layer at $x = -1$. The differential equation for y is formed essentially by adding terms involving z and its derivative to the problem of Example 2. Hence, the solution for y contains a boundary layer similar to z, plus a shock layer at the turning point $x = 0$ similar to the solution of Example 2 (Figure 2). The turning point is the more difficult feature to resolve. Results for an initially uniform mesh are given in Table 4.

Again, RKNEW performs better. The turning point is noticed first, and for instance with $\varepsilon = .1 - 7$, it is not until the second mesh of size 126 that any attempt is made to resolve the boundary layer. For the final mesh at $\varepsilon = .1 - 7$, 73 points are in $(-1, -.99)$ and 79 points are in $(-.1 - 2, .1 - 2)$; the rest of the mesh is a smooth transition between these regions and the boundaries (unlike the previous two examples).

The results in Table 4 can be extended using a simple continuation of meshes: The

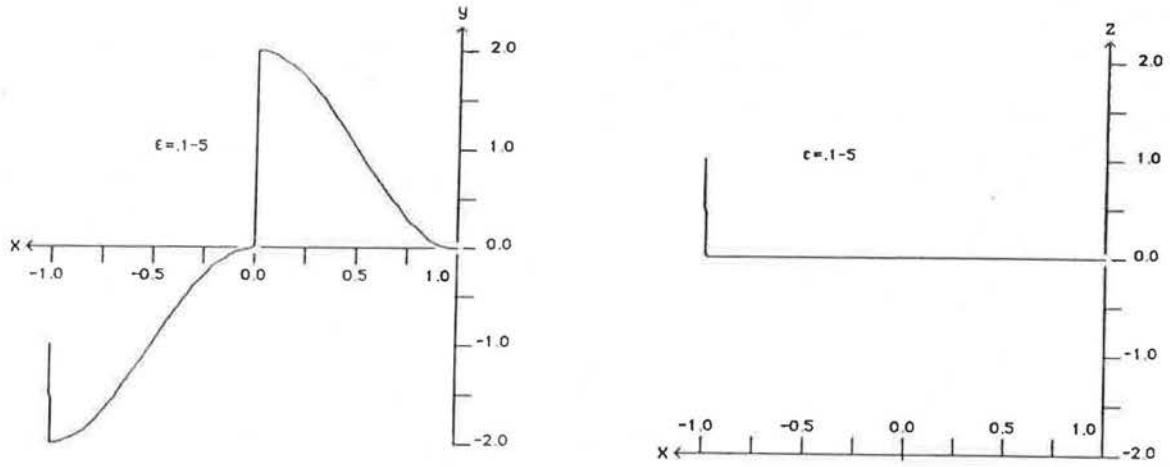

Figure 2: The solution for Example 4

			COLNEW				RKNEW			
ϵ	k	δ	E(u_1)	E(u_4)	Ntot	Mesh Sequence	E(u_1)	E(u_4)	Ntot	Mesh Sequence
.1-1	4	.1-1	.11-3	.24-3	15	5,10	.11-3	.24-3	15	5,10
.1-1	4	.1-5	.27-6	.35-6	75	5,10,20,40	.27-6	.35-6	75	5,10,20,40
.1-2	4	.1-5	.79-7	.96-7	175	5,10,20,20,40,80	.59-7	.13-7	233	5,10,20,17,34,21,42,84
.1-3	4	.1-5	.81-6	.29-7	390	5,10,20,40,35,70,35, 70,35,70	.14-6	.31-7	290	5,10,20,20,20,40,25,50, 100
.1-5	4	.1-5	.44-6	.11-7	817	5,10,10,10,20,20,40, 40,40,80,80,77,154, 77,154	.14-6	.78-8	635	5,10,20,40,40,40,80,40, 80,40,80,160
.1-6	4	.1-5				5,10,10,10,20,20,20, 40,40,40,80,80,80,160, 160,160,320,>500				
.1-7	4	.1-5					.27-5	.24-6	1343	5,10,20,20,40,40,40,80, 80,63,126,63,126,63, 126,63,126,252
.1-8	4	.1-5								5,10,20,20,40,40,40,80, 80,80,160,160,320,>500

Table 4: Results for Example 4 with a uniform initial mesh

			COLNEW				RKNEW			
ε	k	δ	$E(u_1)$	$E(u_4)$	Ntot	Mesh Sequence	$E(u_1)$	$E(u_4)$	Ntot	Mesh Sequence
.1-7	4	.1-5	.28-5	.27-8	740	77,51,102,51,102,51, 102,204				
.1-9	4	.1-5				102,61,122,61,122,65, 130,65,130,122,244, 244,244,488,309,>500	.24-6	.14-6	567	126,63,126,252
.1-11	4	.1-5					.56-4	36-7	1701	126,63,126,63,126,63, 126,252,126,252,126, 252

Table 5: Results for Example 4 using continuation of meshes

final mesh at one value of ε is used as the initial mesh for a smaller value of ε. This is most natural to do in the context of nonlinear problems (see e.g. [6], [10]). In practice we use the second last mesh (of which the last is a doubling), in order to convey the character of the final mesh with as few points as possible. Results are given in Table 5, where the first initial mesh is taken from the smallest value of ε in Table 4, and subsequent initial meshes are taken from the previous value of ε in the table. It is interesting to note that if we decrease ε by a factor of .1 instead of $.1 - 1$, then the step from $.1 - 8$ to $.1 - 9$ for RKNEW requires > 500 mesh points. This indicates a sensitivity of the process. □

5 Conclusions

In this paper we have discussed an a posteriori mesh selection procedure for collocation at Gaussian points based on localized error estimates at *collocation points*. This procedure can be used to give efficient high order approximations to extremely stiff problems. These features have been incorporated within the framework of COLSYS to give an experimental, general purpose code, RKNEW, for linear first order problems.

RKNEW was compared to COLNEW (the most recent version of COLSYS) for numerous linear examples and the results are promising. As the perturbation parameter, ϵ, gets small, RKNEW usually outperforms COLNEW, with respect to both the total work required (based on the total number of mesh points in the mesh sequence), and the size of largest mesh required. Furthermore, RKNEW recognizes and responds to stiff problems better and is more methodical in the way it chooses meshes. The results are quite dramatic for turning points, and to a lesser degree, for boundary layers. RKNEW does especially well when meshes with very large local nonuniformities of interval sizes are present, a situation that causes COLNEW difficulty. COLNEW does work better for some nonstiff and mildly stiff problems; however, RKNEW usually still performs adequately in such cases. The creation of a new mesh is slightly more expensive in RKNEW.

The performance of RKNEW may be critically affected in extreme cases by the choice of initial mesh. If ϵ is extremely small, a coarse initial mesh may actually "miss" the disturbance, and RKNEW will require very large meshes to recover. Simple

nonuniform meshes, continuation of meshes, and the exponential meshes of [8] are all reasonable alternatives, if used intelligently.

Extension of the ideas presented here to nonlinear problems and their full general-purpose implementation are under investigation and will be reported elsewhere.

References

[1] U. Ascher, "On some difference schemes for singular singulary-perturbed boundary value problems," *Numer. Math.*, v. 46, 1985, pp. 1-30.

[2] U. Ascher, "Collocation for two-point boundary value problems revisited," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 596-609.

[3] U. Ascher, "Two families of symmetric difference schemes for singular perturbation problems," in Proc. Workshop on Numerical Boundary Value ODEs, U. Ascher and R.D. Russell (Eds.) Birkhauser, 1985.

[4] U. Ascher & G. Bader, "Stability of collocation at gaussian points," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 412-422.

[5] U. Ascher, J. Christiansen & R.D. Russell, "A collocation solver for mixed order systems of boundary value problems," *Math. Comp.*, v. 33, 1979, pp. 659-679.

[6] U. Ascher, J. Christiansen & R.D. Russell, "Collocation software for boundary value ODEs," *ACM Trans. Math. Software*, v. 7, 1981, pp. 209-222.

[7] U. Ascher & P. Spudich, "A hybrid collocation method for calculating complete theoretical seismograms in vertically varying media", *Geophys. J. R. Astr. Soc.*, v. 86, 1986, pp. 19-40.

[8] U. Ascher & R. Weiss, "Collocation for singular perturbation problems I: First order systems with constant coefficients," *SIAM J. Numer. Anal.*, v. 15, 1983, pp. 537-557.

[9] U. Ascher & R. Weiss, "Collocation for singular perturbation problems II: Linear first order systems without turning points," *Math. Comp.*, v. 43, 1984, pp. 157-187.

[10] U. Ascher & R. Weiss, "Collocation for singular perturbation problems III: Nonlinear problems without turning points," *SIAM J. Sci. Stat. Comput.*, v. 5, 1984, pp. 811-829.

[11] G. Bader & U. Ascher, "A new basis implementation for a mixed order boundary value ODE solver," Tech. Rep. 85-11, University of British Columbia, Vancouver, Canada, 1986, to appear SIAM J. Sci. Stat. Comput.

[12] C. de Boor & B. Swartz, "Collocation at Gaussian points," *SIAM J. Numer. Anal.*, v. 10, 1973, pp. 582-606.

[13] C. de Boor & R. Weiss, "SOLVEBLOK: A package for solving almost block diagonal linear systems," *ACM Trans. Math. Software*, v. 6, 1980, pp. 80-87.

[14] D.L. Brown & J. Lorenz, "A high-order method for stiff boundary-value problems with turning points," Los Alamos Report LA-UR-85-4406, 1985.

[15] L. Dieci & R.D. Russell, "Riccati and other methods for singularly perturbed BVPS," in Proc. BAILI Conference, J. Miller (Ed.), Boole Press, 1986.

[16] C.W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, 1971.

[17] C.W. Gear & L. Petzold, "ODE methods for the solution of differential/algebraic systems", SIAM J. Numer. Anal., v. 21, 1984, pp. 716-728.

[18] P.W. Hemker, "A numerical study of stiff two-point boundary value problems," Tech. Rep. 80, Math. Centrum, Amsterdam, 1977.

[19] S. Jacobs, "Implementation methods for singularly perturbed two-point boundary value problems", M.Sc. Thesis, Dept. Computer Sc., Univ. of British Columbia, Vancouver, Canada, 1986.

[20] H.-O. Kreiss, "Problems with different time scales for ordinary differential equations", *SIAM J. Numer. Anal.*, v. 16, 1979, pp. 980-998.

[21] H.-O. Kreiss, N. Nichols & D. Brown, "Numerical methods for stiff two-point boundary value problems," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 325-368.

[22] R. März, "On boundary value problems in differential-algebraic equations", manuscript, 1986.

[23] G. Meyer, "Continuous orthonormalization for boundary value problems", *J. Comp. Phys.*, v. 62, 1986, pp. 248-262.

[24] V. Pereyra & E.G. Sewell, "Mesh selection for discrete solution of boundary value problems in ordinary differential equations," *Numer. Math.*, v. 23, 1975, pp. 261-268.

[25] R.D. Russell, "Mesh selection methods," in Springer Lecture Notes in Computer Science 76, B. Childs et al (Eds.), New York, 1979.

[26] R.D. Russell & J. Christiansen, "Adaptive mesh selection strategies for solving boundary value problems," *SIAM J. Numer. Anal.*, v. 15, 1978, pp. 59-80.

[27] R. Weiss, "An analysis of the box and trapezoidal schemes for linear singularly perturbed boundary value problems," *Math. Comp.*, v. 42, 1984, pp. 41-67.