Application of global methods in parallel shooting

Citation for published version (APA):

Kramer, M. E., & Mattheij, R. M. M. (1991). *Application of global methods in parallel shooting*. (RANA : reports on applied and numerical analysis; Vol. 9106). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1991

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

RANA 91-06

April 1991

**APPLICATION OF GLOBAL
METHODS IN PARALLEL
SHOOTING**

by

M.E. Kramer

R.M.M. Mattheij

ISSN: 0926-4507

Reports on Applied and Numerical Analysis

Department of Mathematics and Computing Science

Eindhoven University of Technology

P.O. Box 513

5600 MB Eindhoven

The Netherlands

Application of global methods in parallel shooting

by

M.E.Kramer
R.M.M.Mattheij

Eindhoven University of Technology
P.O.box 513
5600 MB Eindhoven
The Netherlands

Abstract

A new divide and conquer method is proposed. It employs a coarse grid discretization like multiple shooting does, but solves the local problems in a 'BVP-way', i.e. by some global method. A sophisticated error control is developed to combine local and global convergence of the Newton updating. An implementation based on using the collocation code COLNEW is briefly discussed and a number of examples is given to illustrate the success of the method, in particular for singular perturbation problems.

§1 Introduction

Consider the following well-conditioned two point boundary value problem (BVP)

$$(1.1a) \quad \dot{y} = h(x, y) \quad a < x < b$$

$$(1.1b) \quad g(y(a), y(b)) = 0$$

Roughly speaking, methods for solving (1.1) can be divided into two classes, viz. methods based on IVP (initial value problem) techniques, like invariant imbedding or multiple shooting and 'global' methods, like collocation or finite differences.

The main difference between these classes is that the first considers discretization at two levels, viz. a coarse and a fine level, whereas the second is using fine level discretization only. The advantage of the former is that it can deal with subintervals separately (the coarse level discretization) and so needs less memory space; in fact it may even lead to a parallel implementation. The main drawback, however, is that the local IVP may be very ill-conditioned, viz. if the underlying ODE allows for rapidly growing modes. This may cause severe problems for the (fine level) integration, which may only be overcome to some extent by 'refining' the coarse level mesh. This way one obtains a system of shooting vectors (initial values on the latter mesh) which, together with the boundary conditions (BC) (1.1b), lead to a, generally nonlinear, system. If this is solved by Newton's method, one encounters at each update a linear equation of the form

$$(1.2) \quad \begin{pmatrix} S_1 & R_1 & & & & \\ & S_2 & R_2 & & & \\ & & \ddots & \ddots & & \\ & & & S_N & R_N & \\ B_a & & & & & B_b \end{pmatrix} \cdot \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \vdots \\ \xi_{N+1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N+1} \end{pmatrix}$$

A better condition number of this system reflects the (partial) success of a finer coarse level discretization. Even more important is the fact that the convergence domain for Newton's method, which may be extremely small for a very crude coarse level grid, may be enlarged exponentially by refining.

Most of the unpleasant features mentioned for IVP methods are absent in global methods. Therefore it is an attractive idea to combine the virtues of both classes. To this end the interval $[a,b]$ is divided into subintervals, but now linear BC (rather than initial conditions) are defined for a local solution. In particular, one should try and solve these local BVP by a 'global' method rather than an IVP method. This idea has the following advantages. First it results in a more optimal memory usage and it renders a potential parallel feature as well. Incidentally, a slightly related parallelism idea is also mentioned in [4], though through a different motivation. Second it allows for the better convergence and stability properties of global methods and third, as a useful byproduct, it gives an opportunity to 'localize' unpleasant nonlinearities, while at the same time the coarse level nonlinear equation might become 'easier' to solve.

The outline of this paper is as follows. In §2 we first introduce a definition of conditioning of nonlinear BVP's and we estimate the bounds used in the Newton-Kantorovich theorem, when applied to the matching conditions of multiple shooting. In section 3 we address the choice of local boundary conditions and the influence of the use of subintervals on the computational costs of the Newton iteration on collocation schemes (which is the global method we shall consider here). We also investigate the choice of local boundary conditions (BC's). In §4 we prove local convergence of the method and derive a tolerance strategy. As we have two types of nonlinear problems, viz. a sequence of local BVP and a global equation, it is also investigated here how these two interfere. Finally, in section 5 we describe our implementation and give some numerical results.

§2 Conditioning of nonlinear BVP's and its influence on multiple shooting

For linear boundary value problems the concept of well-conditioning and topics related to it, like dichotomy and the influence of boundary conditions, has been studied extensively, see e.g. [9]. There is not much literature available about the conditioning of nonlinear BVP's. The main difference with the linear case is that the size of the perturbations plays an essential role; multiplying a perturbation by some factor, may cause a considerable larger change in the solution, than the original perturbation. In this paper we shall use the maximum norm for vectors and the supremum norm for functions, i.e.

$$\forall_{y \in C([a,b] \rightarrow \mathbb{R}^n)} \quad : \quad \|y\| := \max_{a \leq x \leq b} \|y(x)\|_\infty$$

Consider the nonlinear boundary value problem

$$(2.1a) \quad \dot{y} = h(x, y) \quad a < x < b$$

$$(2.1b) \quad g(y(a), y(b)) = 0$$

with solution $y^*(x)$ and also the slightly perturbed BVP

$$(2.2a) \quad \dot{y} = h(x, y) + \delta h(x, y) \quad a < x < b$$

$$(2.2b) \quad g(y(a), y(b)) + \delta g(y(a), y(b)) = 0$$

For nonlinear BVP we introduce the following conditioning concept.

2.3 Definition

Let $y^*(x)$ be an isolated solution of (2.1). For all $\varepsilon > 0$ the conditioning constant $\kappa(\varepsilon)$ is defined by

$$(2.3a) \quad \forall_{\delta h \in C([a,b] \times \mathbb{R}^n \rightarrow \mathbb{R}^n)} \quad \forall_{\delta g \in C(\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n)} \quad : \quad \text{If } \tilde{y} \text{ is a solution of (2.2) with} \\ \| \delta h(x, \tilde{y}(x)) \| < \varepsilon \quad \text{and} \quad \| \delta g(\tilde{y}(a), \tilde{y}(b)) \| < \varepsilon \\ \text{then} \quad \| y^*(x) - \tilde{y}(x) \| \leq \kappa(\varepsilon) (\| \delta h \| + \| \delta g \|)$$

The BVP is well conditioned at $y^*(x)$ if there is an ε such that $\kappa(\varepsilon)$ is of moderate size.

♦

This definition of conditioning for nonlinear BVP's generalizes the normally used definition for linear BVP's (see e.g. [3]). However, this may not give a good estimate for large perturbations. Since the linearization of a BVP describes the first order effect of small perturbations on the nonlinear BVP, we expect the conditioning constant of the linearized BVP not to be considerably larger than that of the original nonlinear one.

Let

$$(2.4) \quad A(x) := \left. \frac{\partial h(x, y)}{\partial y} \right|_{y=y^*(x)}, \quad B_a := \left. \frac{\partial g(u, y^*(b))}{\partial u} \right|_{u=y^*(a)} \quad \text{and} \quad B_b := \left. \frac{\partial g(y^*(a), v)}{\partial v} \right|_{v=y^*(b)}$$

and consider the linearized BVP with inhomogeneities $q(x)$ and β

$$(2.5) \quad \begin{aligned} \dot{z}(x) &= A(x)z(x) + q(x) & a < x < b \\ B_a z(a) + B_b z(b) &= \beta \end{aligned}$$

2.6 Lemma

The conditioning constant κ_1 of the linear BVP (2.5) does not exceed $\kappa(0)$.

Proof

Let $\varepsilon > 0$, $q(x) \in C([a, b] \rightarrow \mathbb{R}^n)$ and $\beta \in \mathbb{R}^n$. And let $z(x)$ be the solution of the perturbed linearized BVP (2.5). Define $y(x) := y^*(x) - z(x)$, then $y(x)$ satisfies the perturbed BVP (2.2) with

$$\begin{aligned}\delta h &= h(x, y^*) - h(x, y) - A(x)(y^*(x) - y(x)) - q(x) \\ \delta g &= g(y^*(a), y^*(b)) - g(y(a), y(b)) - B_a z(a) - B_b z(b) - \beta\end{aligned}$$

Now the definition of the conditioning constant can be applied if both perturbations are smaller than ε . This is implied by

$$(*) \quad \begin{aligned}C_{gh} \|y - y^*\|^2 + \|q\| &< \varepsilon \\ 2C_{gh} \|y - y^*\|^2 + \|\beta\| &< \varepsilon\end{aligned}$$

with C_{gh} a bound on the second (partial) derivatives of the functions g and h . Since $z(x)$ is the solution of a linear problem these conditions can be met by scaling. Let α be a scalar and define

$\tilde{q} := \alpha q$, $\tilde{\beta} := \alpha \beta$ and $\tilde{z}(x) := \alpha z(x)$. Then $(*)$ is satisfied for sufficiently small α and from definition (2.3) we get

$$\begin{aligned}\|\tilde{z}\| &\leq \kappa(\varepsilon) (3C_{gh} \|\tilde{z}\|^2 + \|\tilde{q}\| + \|\tilde{\beta}\|) \\ \Rightarrow \alpha \|z\| &\leq \kappa(\varepsilon) (3C_{gh} \|z\|^2 \alpha^2 + \alpha \|q\| + \alpha \|\beta\|) \\ \Rightarrow \|z\| &\leq \frac{1 - \sqrt{1 - 12C_{gh}\kappa^2(\varepsilon)\alpha(\|q\| + \|\beta\|)}}{6C_{gh}\kappa(\varepsilon)\alpha} \\ &= (1 - \theta)^{-1/2} \kappa(\varepsilon) (\|q\| + \|\beta\|) \quad \text{with } 0 < \theta < 12\alpha C_{gh}\kappa^2(\varepsilon)(\|q\| + \|\beta\|)\end{aligned}$$

Finally let α , and thereafter ε , approach zero.

◆

From this lemma we see that if a nonlinear BVP is well conditioned, even on a small domain, then its linearization is well conditioned, too. This in turn implies that the linearized BVP is dichotomic (see [9]), i.e. the solution space can be split into a subspace of nondecreasing modes and a subspace of nonincreasing modes, where the angle between the two spaces is bounded away from zero. If the linearized BVP at the solution $y^*(x)$ is exponentially dichotomic, then this is also true for the linearization at neighbouring functions $y(x)$ (see [10]). For these kinds of BVP's the conditioning constant $\kappa(\varepsilon)$ may vary strongly with ε .

When solving this BVP by multiple shooting the interval $[a, b]$ is subdivided into N subintervals $[x_k, x_{k+1}]$ with $a = x_1 \leq x_2 \leq \dots \leq x_{N+1} = b$ and on each such subinterval the IVP

$$(2.7) \quad \begin{aligned}\dot{y} &= h(x, y) & x_k \leq x \leq x_{k+1} \\ y(x_k) &= s_k\end{aligned}$$

is solved. The solution of the IVP on the k^{th} interval is denoted by $y_k(x; s_k)$. The concatenation of these solutions is the solution to the original BVP (2.1) by requiring that they form a continuous function satisfying the boundary conditions. So the initial vectors s_k have to be the solution of the following set of nonlinear equations

$$(2.8) \quad f(s) = 0, \quad f \in C^1(\mathbb{R}^{n(N+1)} \rightarrow \mathbb{R}^{n(N+1)}) \quad ; \quad \text{with } s := (s_1^T, s_2^T, \dots, s_{N+1}^T)^T$$

and

$$(2.9) \quad f(s) := \begin{pmatrix} y_1(x_2; s_1) - y_2(x_2; s_2) \\ y_2(x_3; s_2) - y_3(x_3; s_3) \\ \vdots \\ y_{N-1}(x_N; s_{N-1}) - y_N(x_N; s_N) \\ g(y_1(x_1; s_1), y_N(x_N; s_N)) \end{pmatrix}$$

If the BVP is nonlinear, the function $f(s)$ is nonlinear too and (2.8) has to be solved by some iterative scheme.

Inherent to a BVP is that the underlying ODE may contain exponentially growing modes. For the well conditioning of the problem it is vital that those modes are controlled at the endpoint. However, on the subintervals only initial conditions are imposed. Due to this a method based on shooting encounters several drawbacks. First the approximating solution may blow up and may not reach the proposed end point or the numerical solution to the IVP may become unstable. Such problems can be overcome to some extent by taking smaller subintervals. Second, after integration, one has to obtain the Newton update ξ from a linear system like $J(s)\xi = -f(s)$, where $J(s)$ is the Jacobian. If this equation is condensed in a standard way, the error in the solution becomes almost equal to that of single shooting, which might be considerable (see [3] Ch.4.3). Some stable schemes have been developed that implicitly distinguish between growing and decaying modes (see e.g. [3] Ch.7).

The problems just mentioned arise for linear and nonlinear BVP's alike. For nonlinear BVP one encounters an additional problem. Since $f(s)$ may be sensitive to changes of the vector s in certain directions, the convergence domain of Newton's method might be small. Before deriving our results we introduce some notation first.

2.10 Assumption

Let D_y be a convex neighbourhood of $y^*(x)$ such that

- the function $g(u, v)$ is twice continuously differentiable in both variables and $h(x, y)$ is twice differentiable with respect to y and all partial derivatives are bounded by a moderate constant, say C_{gh} .
- there is a moderate constant κ_{global} such that

$$\exists_{\epsilon > 0} \quad \forall_{y \in D_y} : \text{conditioning constant } \kappa(\epsilon) \text{ of (2.2) at } y(x) \text{ is bounded by } \kappa_{\text{global}}.$$

Let the vector $s^* \in \mathbb{R}^{n(N+1)}$ be such that $y^*(x) = y(x; s^*)$ and let D_s be the largest neighbourhood of s^* with $\{y(x; s) \mid s \in D_s\} \subseteq D_y$.

Finally assume that for all $1 \leq k \leq N$ there is a constant κ_k such that

$$\forall_{y \in D_y} : \text{the conditioning constant } \kappa(\epsilon) \text{ of (2.7) at } y(x) \text{ restricted to } [x_k, x_{k+1}] \text{ is bounded by } \kappa_k \text{ with } \epsilon = \max\{\|s_k - \sigma_k\| \mid s, \sigma \in D_s\}$$

◆

Now let the linearized ODE at $[x_k, x_{k+1}]$ be

$$(2.11) \quad \dot{z} = L_k(x; s) z \quad \text{i.e.} \quad L_k(x; s) = \left. \frac{\partial h(x, y)}{\partial y} \right|_{y=y_k(x; s)}$$

and denote the linearization of the boundary conditions by $B_a(s)$ and $B_b(s)$, i.e.

$$(2.12) \quad B_a(s) = \left. \frac{\partial g(u, y_N(x_{N+1}; s))}{\partial u} \right|_{u=y_1(x_1; s)} \quad \text{and} \quad B_b(s) = \left. \frac{\partial g(y_1(x_1; s), v)}{\partial v} \right|_{v=y_N(x_{N+1}; s)}$$

Furthermore $Y_k(x; s)$ and $G_k(x, t; s)$ denote the fundamental solution and the Green's function of (2.11) respectively, where the fundamental solution is scaled such that $Y_k(x; s) = I$.

According to the Newton Kantorovich theorem (see e.g. [12]), convergence of Newton's algorithm with initial guess s^0 is guaranteed, if the product $\eta \gamma \|J^{-1}(s^0) f(s^0)\| < 0.5$; here γ is the Lipschitz constant of $J(s)$ near s^* and η is an upper bound on $\|J^{-1}(s)\|$ near s^* . In practice the convergence domain will be larger, but often the bounds from this theorem are indicative.

2.13 Theorem

$$(2.13a) \quad \forall_{s, \sigma \in D_s} : \quad \frac{\|J(s) - J(\sigma)\|}{\|s - \sigma\|} = O(C_{gh} \max_k (\kappa_k^3))$$

and

$$(2.13b) \quad \|J^{-1}(s)\| \leq NC_{gh} \kappa_{global}$$

◆

Proof

Let $s, \sigma \in D_s$. Now for each of the first $(N-1)$ block rows of $J(s)$ we estimate the difference in the fundamental solutions.

$$Y_k(x; \sigma) = Y_k(x; s) + \int_{x_k}^{x_{k+1}} G_k(x, t; s) (L_k(t; \sigma) - L_k(t; s)) Y_k(t; \sigma) dt$$

Since $\|L_k(x; s) - L_k(x; \sigma)\| \leq C_{gh} \|y_k(x; s) - y_k(x; \sigma)\| \leq C_{gh} \kappa_k \|s - \sigma\|$, this yields

$$\|Y_k(x; s) - Y_k(x; \sigma)\| \leq \kappa_k^2 \max_t \|L_k(t; s) - L_k(t; \sigma)\| \leq C_{gh} \kappa_k^3 \|s - \sigma\|.$$

For the last block row we have the estimate

$$\begin{aligned} & \|B_a(s)Y_1(x_1; s) - B_a(\sigma)Y_1(x_1; \sigma)\| \\ & \leq \|B_a(s) - B_a(\sigma)\| \|Y_1(x_1; s)\| + \|B_a(\sigma)\| \|Y_1(x_1; s) - Y_1(x_1; \sigma)\| \\ & \leq C_{gh} \|y_N(x_{N+1}; s) - y_N(x_{N+1}; \sigma)\| \kappa_k + C_{gh} \|y_1(x_1; s) - y_1(x_1; \sigma)\| \kappa_k \\ & \quad + C_{gh} \|Y_1(x_1; s) - Y_1(x_1; \sigma)\| \\ & \leq C_{gh} \kappa_k^2 (2 + C_{gh} \kappa_k) \|s - \sigma\| \end{aligned}$$

For the block containing the endpoint conditions an analogous upper bound can be found. Combining these estimates we get (2.13a). The proof of (2.13b) has been given by many authors e.g. [2].

◆

For an exponentially dichotomic BVP with the strongest growing mode growing like $e^{\lambda x}$, one can easily prove that $\kappa_k = O(\exp(\lambda(x_{k+1} - x_k)))$. Hence taking smaller subintervals does enlarge the convergence domain indicated by the Newton-Kantorovich theorem. However, one has to solve a larger system.

§3 Unbiased multiple shooting

In the previous section we have seen that the convergence domain and behaviour of the Newton's method applied in the multiple shooting process is influenced by the conditioning constant of the local problems. This renders the idea that the convergence behaviour can be improved by defining well conditioned problems on the subintervals instead of IVP's, i.e. that (2.3) should be replaced by

$$(3.1a) \quad \dot{y} = h(x, y) \quad x_k < x < x_{k+1}$$

$$(3.1b) \quad A_k \lim_{x \rightarrow x_k} y_k(x) + B_k \lim_{x \rightarrow x_{k+1}} y_k(x) = s_k$$

One can easily prove that Theorem 2.13 also holds for this more general formulation of the shooting method. In both [8] and [2] this formulation was used to analyze the convergence and stability of finite difference methods applied to linear BVP with internal layers and/or turning points. Here we investigate the actual implementation of this generalisation of multiple shooting. First of all we see that if $A_k \neq 0$ and $B_k \neq 0$, a BVP with linear boundary conditions is defined on every subinterval. Now it may seem unwise to replace one problem by N problems of the same type with additional unknowns s_k , $1 \leq k \leq N$. However, there could be some merit in this splitting.

As we just pointed out the use of ordinary multiple shooting may be disadvantageous. Hence for solving the 'local' problems we only consider the use of global methods, i.e. finite differences or collocation. A divide and conquer method, is the following *Unbiased Multiple Shooting* (UMS) algorithm

- given an initial estimate y^0 for y^* , compute the vector s^0 from (3.1b).
- while $\|f(s)\|$ is not sufficiently small
do begin
 - (A1) on every subinterval compute a new approximation y^j to the solution of (3.1) for the new value s^j , with y^{j-1} as initial guess by collocation or finite differences
 - (A2) compute the residual vector $f(s^j)$ and perform a Newton iteration rendering s^{j+1} .
- end

The two steps (A1) and (A2) do not have an equal status. An important difference is that every update of s requires a new approximation to the solution of (3.1), i.e. at every iteration only one update on s is made. On the other hand in step (A1) the vector s is kept fixed and obtaining a new approximation y^j may require several Newton iterations or even choosing a new local collocation grid. In fact (A1) can stand for a call to a collocation algorithm and will generally contain what we call an 'inner' iteration loop (as opposed to the 'outer' iteration on s).

Notice that at step (A1) every subinterval can be treated completely independently. Thus a major part of the memory needed for the collocation or finite difference process at one step can be used

again at the next one, as we only store information about $y(x)$ and not about the linearized system.

This way it may be possible to handle more difficult problems, that would otherwise require more memory for storing information on the global grid.

On the other hand step (A1) lends itself to implementation on a parallel computer in a more or less straightforward way. For every vector s^j the local BVP's on the subintervals can be distributed over the available processors. The Newton update of s^j requires the fundamental solution of the linearized local BVP (these form $\frac{\partial y(x;s)}{\partial s}$). Again this can be done on different processors for the different subintervals. This could be combined with a stable parallel algorithm to solve $J\xi = -f$, for instance the one described in [4] or [13].

One may also encounter a situation where the problem at hand has a few regions where the problem is essentially more difficult than elsewhere. This may be due to a locally poor initial guess or to local sensitivity of the BVP. When a collocation code is applied to the BVP on the entire interval, the internal Newton solver may require a considerable amount of iterations. If the interval is split into smooth regions and more difficult ones, application of the same code will generally require only a few iterations on the smooth regions; at the same time it is to be expected that solving the BVP on the difficult subintervals does not take more iterations than solving the BVP on the entire interval. However, these iterations for the former require less function calls and the solution of smaller linear systems. So the unbiased multiple shooting algorithm can reduce the computational costs of solving a BVP, provided that determining the 'shooting vectors' s_k is not too expensive. A nice class of such BVP's are formed by singularly perturbed problems, where a *reduced solution* (i.e. the -outer- solution of the reduced problem) is easy to find.

3.2 Example

Consider the singularly perturbed BVP, cf. [11]

$$(3.2a) \quad \epsilon \ddot{y} = \dot{y} - \dot{y}^3$$

$$(3.2b) \quad \begin{aligned} y(0) &= 0 \\ y(1) &= 0.5 \end{aligned}$$

The limiting solution for $\epsilon \downarrow 0$,

$$(3.2c) \quad y(x) = \begin{cases} 0 & 0 \leq x \leq 0.5 \\ x - 0.5 & 0.5 \leq x \leq 1 \end{cases}$$

has a discontinuity in its first derivative at $x = 0.5$. We tried to solve this BVP with the collocation code COLNEW cf. [6] on the entire interval and with the UMS algorithm (for details on the implementation and further comments see §5) up to a tolerance 10^{-6} . The reduced solution (3.2c) was used as initial guess and the local boundary conditions were the analogue of (3.2b), i.e.

$$A_k = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad B_k = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

In the UMS algorithm with $\epsilon \in [10^{-2}, 10^{-4}]$ no iterations on s were needed as the norm of the first update was already less than 10^{-8} . The results are listed in the table 3.1 and 3.2. The column 'memory use' states the number of double precision places (in standard IBM Fortran) required for

the collocation algorithm; the additional memory used for integers is negligible. The UMS algorithm saves about 35% to 75% on both function evaluations and memory use.

ϵ	x_k	memory use	#f-calls		
			1st	2nd	3rd interval
10^{-2}	0.380 , 0.620	9728	18	2968	30
10^{-3}	0.485 , 0.515	14592	18	3366	22
10^{-4}	0.499 , 0.501	14592	18	3414	22

UMS on (3.2a,b) with tol = 10^{-6}

Table 3.1

ϵ	memory use	#f-calls
10^{-2}	14592	4578
10^{-3}	58368	13140
10^{-4}	58368	13500

COLNEW on (3.2a,b) with tol= 10^{-6}

Table 3.2

◆

An important issue in our algorithm is the choice of the local boundary conditions (BC's). Occasionally local BC's are provided naturally by the problem (e.g. (3.2a,b)). Otherwise the choice

$$(3.3) \quad A_k = B_k = I$$

is generally an acceptable one. However, the complexity of most algorithms for collocation and finite differences reduces considerably if the boundary conditions are separated. This can of course be done by adding as many auxiliary variables as there are coupled BC. Fortunately, there is an alternative to enlarging the system.

Suppose we have an initial guess y^0 of the solution of (3.1),(3.3) and we are able to compute the fundamental solution $Y_k(x)$ of the linearized problem at y^0 (i.e. the BVP is not too ill-posed). Now let $U_k \Sigma_k V_k^T$ be the singular value decomposition of $Y_k(x_{k+1})Y_k^{-1}(x_k)$. Note that this operation is not excessively expensive, since $Y_k(x_{k+1})$ and $Y_k(x_k)$ have to be computed anyway for the Newton iteration on s and a SVD requires $O(n^3)$ operations. This is essentially smaller than $O(kn^3)$, with k the number of collocation grid points, required for solving the linear equations to update y in the collocation process.

Now let p , $1 \leq p \leq n$, be such that the singular values $\sigma_i \geq 1$ for $1 \leq i \leq p$ and $\sigma_i \leq 1$ for $p+1 \leq i \leq n$. According to [9] the boundary conditions

$$(3.4) \quad \tilde{A}_k = \begin{pmatrix} 0 & 0 \\ 0 & I_{n-p} \end{pmatrix} V_k^T \quad \text{and} \quad \tilde{B}_k = \begin{pmatrix} I_p & 0 \\ 0 & 0 \end{pmatrix} U_k^T$$

induce a BVP (3.1) on $[x_k, x_{k+1}]$ with conditioning constant $\alpha + 4\alpha^2$, where α is the conditioning constant of (3.1),(3.3). This can be unsatisfactory if α is large. However, the conditioning constant of (3.1),(3.4) can be related to that of the global BVP (2.1) at y^0 as is shown by:

3.5 Lemma

Let $y^0 \in D_y$. The conditioning constant of (3.1),(3.4) does not exceed $\kappa_{global} + 4\kappa_{global}^2$.

Proof

Since $y^0 \in D_y$, the conditioning constant of the global linearized BVP (2.11),(2.12) at y^0 is bounded by κ_{global} . According to [8] there exists BC on every subinterval of $[a,b]$ such that the conditioning constant of (2.7) does not exceed κ_{global} . The proof now readily follows from the fact that the product $Y(x_{k+1})Y^{-1}(x_k)$ is independent of the BC.

♦

In practice one has to be careful when implementing boundary conditions such as (3.4). From a computational point of view it is preferable to compute the SVD from one of the first approximations y^0 or y^1 . However, if they differ greatly from the solution $y^*(x)$, i.e. lie outside D_y , both the directions and the singular values may be so inaccurate that the conditioning of (3.1),(3.4) is not very good. This is demonstrated by the next example.

3.6 Example

Consider the boundary value problem, proposed e.g. in [7],

$$\begin{aligned}
 \dot{y}_1 &= y_2 \\
 \dot{y}_2 &= y_3 \\
 \dot{y}_3 &= -\frac{(3-n)}{2}y_1y_3 - ny_2^2 + 1 - y_4^2 + sy_2 \\
 \dot{y}_4 &= y_5 \\
 \dot{y}_5 &= -\frac{(3-n)}{2}y_1y_5 - (n-1)y_2y_4 + s(y_4 - 1)
 \end{aligned} \quad 0 < x < \infty$$

(3.6a)

with boundary conditions

$$(3.6b) \quad y_1(0) = y_2(0) = y_4(0) = 0, \quad y_2(\infty) = 0, \quad y_4(\infty) = 1$$

We applied the UMS algorithm to this BVP for the parameters $n = s = 0.2$ and $L = 60$, with subintervals $[0,5]$, $[5,10]$, $[10,30]$ and $[30,60]$. First we choose the boundary conditions

$$A_k = B_k = I$$

This required 6 'outer' iterations on s to obtain a precision of 10^{-6} . In the first and fifth iteration the singular value decomposition of $Y_k(x_{k+1})Y_k^{-1}(x_k)$ was computed. In both cases we found that 3 singular values were larger than 1, implying that the problem has three nondecreasing modes. This seems to be contradicted by the global BC's, that have 2 end point conditions, indicating 2 non-decreasing modes. However, only 2 singular values were considerably larger than 1, ranging from $\sim 10^2$ at $[0,5]$ to $\sim 10^6$ at the last subinterval, and the third singular value was only just larger than 1. We applied the BC's resulting from both singular value decompositions using both 2 and 3 initial conditions.

Although convergence was obtained with the local BC resulting from the first iteration, the computational costs are, as expected, higher than for the local BC resulting from the fifth iteration. The failure on the last interval $[30,60]$ with local BC resulting from the first iteration with 2 initial conditions can be viewed as a standard example of ill-conditioning due to boundary conditions. Let $B_k(i,j)$ denote the endpoint conditions on the k^{th} subinterval resulting from the fundamental solutions obtained in the i^{th} iteration with j initial conditions. The BC resulting from the 5th iteration seem to give rather well-conditioned local BVP's. However, the angle between the 3-dimensional

subspaces of $\text{range}(B_4(1,2))$ and $\text{range}(B_4(5,2))$ is almost 90 , so $B_4(1,2)$ does not control one of the directions controlled by $B_4(5,2)$.

In a parallel implementation the computing time for the UMS algorithm is less than for the globally used collocation algorithm, when using the coupled BC $A_k = B_k = I$ or the BC resulting from the fifth iteration. However, the reduction of computational time would have been considerably larger if less iterations on the shooting vectors had been necessary, as we saw for the singularly perturbed BVP in example 3.1.

The memory requirement for the UMS-algorithm for separated local BC is almost 45% less than for collocation on the entire interval. When using the nonseparated BC $A_k = B_k = I$, the memory use is considerably larger than for the other, separated BC. This is due to the fact that the collocation solver used needs separated BC, hence we have to add 5 trivial differential equations to create the separation artificially (see [5]). Application of COLNEW to each of the subintervals does not require more grid points for the coupled BC than for the other local BC.

UMS applied to (3.6a+b), $\text{eps} = 10^{-6}$

	iter	#f-calls				memory use
		[0,5]	[5,10]	[10,30]	[30,60]	
$A_k = B_k = I$	7	2832	1192	2688	3072	88101
1 st iter., 3 init. cond.	8	2960	1408	3264	4256	25961
5 th iter., 3 init. cond.	6	1952	928	2224	2560	25961
1 st iter., 2 init. cond.	4	fail of COLNEW on [30,60]				
5 th iter., 2 init. cond.	7	2378	1408	2336	2816	25641

COLNEW applied to (3.6a+b), $\text{tol} = 10^{-6}$

4080	46336
------	-------

Table 3.3

◆

§4 Convergence

In the *Unbiased Multiple Shooting* algorithm as sketched in the previous section two iterative processes are interacting, viz. a process on s^j and another on y^j , however, the algorithm is not symmetrical with respect to s^j and y^j .

One can implement the algorithm of section 3 in various ways. For the 'outer' iteration on s^j we can use well known adaptations of Newton's method such as damping and keeping the Jacobian fixed. Additionally there are several ways to perform the 'inner' step (A1) on y^j . One can call a collocation or finite difference routine to obtain y^j and estimates $\hat{Y}_k^j(x)$ for the fundamental solutions $Y_k^j(x; y(x; s^j))$ (that are required for the Jacobian $J(s)$) with prescribed tolerances. These tolerances need not be kept constant during the entire process. At the first few steps, when s^j is far away from the solution s^* , it is not necessary to approximate $y_k(x; s^j)$ very well. But, as we show, eventual quadratic convergence requires the tolerance for y^j to decrease like $\|s^j - s^*\|^2$ and the tolerance for $\hat{Y}_k^j(x)$ like $\|s^j - s^*\|$ at the last few steps of the algorithm.

Another way to implement step (A1) hinges even stronger on the thought that it does not pay to compute $y_k(x; s^j)$ very accurately if s^j is still far from s^* . We can choose a fixed collocation grid on every subinterval and at the j^{th} 'outer' iteration step we perform only a few Newton iterations on the collocation scheme to obtain y^j (i.e. without an accuracy requirement). This way the iteration on the vector s^j plays a more dominant role than in the implementation suggested before. Once convergence of s^j and y^j on a grid has been established, the discretization error is estimated and the grid adjusted and refined accordingly.

An important issue is convergence of the UMS algorithm. In both implementations the Newton update on the vector s will not be computed using the real Jacobian $J(s)$, but only through an approximation J^j . Suppose (without loss of generality) that both the approximation $y_k^j(x)$ to $y_k(x; s^j)$ and $\hat{Y}_k^j(x)$ to $Y_k(x; y(x; s^j))$ are continuous functions. Let $Y_k(x; y_k^j)$ denote the fundamental solution of the linearized BVP at $y_k^j(x)$ and define

$$(4.1a) \quad \epsilon_{y,k}^j := \max_{x \in [x_k, x_{k+1}]} \|y_k^j(x) - y_k(x; s^j)\| \quad , \quad \epsilon_y^j := \max_k \epsilon_{y,k}^j$$

$$(4.1b) \quad \epsilon_{Y,k}^j := \max_{x \in [x_k, x_{k+1}]} \|\hat{Y}_k^j(x) - Y_k^j(x; y_k^j)\| \quad , \quad \epsilon_Y^j := \max_k \epsilon_{Y,k}^j$$

To investigate the difference between the Jacobian $J(s)$ and its approximation J^j , we estimate the difference between $Y_k(x; y(s^j))$ and its computed approximation $\hat{Y}_k^j(x)$.

4.2 Lemma

Let $s^j \in D_s$ and $y^j \in D_y$. Assume that the machine precision is essentially smaller than $\epsilon_{y,k}^j$ and $\epsilon_{Y,k}^j$. Then

$$(4.2a) \quad \forall_{1 \leq k \leq N} \quad \forall_{x \in [x_1, x_{k+1}]} : \|\hat{Y}_k^j(x) - Y_k(x; y(s^j))\| \leq C_{gh} \kappa_k^2 \epsilon_{y,k}^j + \epsilon_{Y,k}^j$$

If the local BVP's are well conditioned there is a constant C of moderate size such that

$$(4.2b) \quad \|J(s^j) - J^j\|_{\infty} \leq 2\epsilon_Y^j + 2C\epsilon_{Y,k}^j \quad .$$

Proof

The matrix $Y_k^j(x)$ contains errors due to

- (i) the error in y^j
- (ii) discretization errors in integrating the linearized problem
- (iii) rounding errors

The influence of the first error can be estimated by

$$Y_k(x; y_k^j) - Y_k(x; y(s^j)) = \int_{x_0}^{x_1} G(x, t; y(s^j)) (L(t; y_k^j) - L(t; y(s^j))) Y_k(t; y^j) dt$$

Since $\|L(t; y^j) - L(t; y(s^j))\| \leq C_{gh} \|y_k^j(x) - y_k(x; s^j)\|$ this yields

$$\max_x \|Y_k(x; y^j) - Y_k(x; y(s^j))\| \leq C_{gh} \kappa_k^2 \max_x \|y_k^j(x) - y_k(x; s^j)\|.$$

Now (4.2a) follows from the fact that the discretization error mentioned under (ii) is controlled by a parameter $\varepsilon_{y,k}^j$ and the assumption that the computational errors are negligible compared to the other errors.

The last block row of $J(s)$ contains, besides the fundamental solutions $Y_k(x; y(s^j))$, derivatives of the boundary conditions $g(u, v)$, so by writing

$$B_a(y^j) - B_a(y(s^j)) = \left. \frac{\partial g(u, y_N^j(x_{N+1}))}{\partial u} \right|_{u=y_1^j(x_1)} - \left. \frac{\partial g(u, y_N(x_{N+1}; s^j))}{\partial u} \right|_{u=y_1(x_1; s^j)}$$

we obtain $\|B_a(y_1^j(x_1)) \hat{Y}_1^j(x_1) - B_a(y_1(x_1; s^j)) Y(x_1; s^j)\| \leq C_{gh}^2 \kappa_1^2 \varepsilon_{y,1}^j + C_{gh} (\varepsilon_{y,1}^j + \varepsilon_{y,N}^j) \kappa_1$

Now (4.2b) follows immediately from this relation and (4.2a).

♦

From estimate (4.2a) we see that the choice of $\varepsilon_{f,k}$ per subinterval can be used to equidistribute the errors in the Jacobian. This can be useful if the bound C_{gh} is known to vary over the subintervals.

The iterates s^j are found by Newton's method, hence we have local quadratic convergence, if the Jacobian matrices are determined with sufficient accuracy. In our algorithm the accuracy depends on $\varepsilon_{y,k}^j$ and $\varepsilon_{y,k}^j$.

4.3 Lemma

Let $s^j \in D_s$ and $y^j \in D_y$. And let C_J and $\hat{C}_J$ be bounds on $\|J(s)\|$, $\|J^{-1}(s)\|$ respectively, $s \in D_s$ and let L_J be the Lipschitz constant of $J(s)$ on D_s . Then there is a moderate constant C (cf. Lemma 4.2) such that

$$(4.3a) \quad \|s^{j+1} - s^*\| \leq L_J \hat{C}_J \|s^j - s^*\|^2 + 2 \hat{C}_J^2 C_J (\varepsilon_y^j + C \varepsilon_y^j) \|s^j - s^*\| + 2 \hat{C}_J \varepsilon_y^j$$

Proof

Although f is a function of s , it can be evaluated at every piece wise continuous function $y(x)$ on $[a, b]$; this function will be denoted by $F(y)$. Application of Newton's algorithm with the approximate Jacobian $\hat{J}$ gives

$$\begin{aligned}
s^{j+1} - s^* &= s^j - s^* - \hat{f}^{-1}F(y^j) \\
&= s^j - s^* - J^{-1}(s^j)f(s^j) + J^{-1}(s^j)f(s^j) - \hat{f}^{-1}F(y^j)
\end{aligned}$$

$$\begin{aligned}
\|s^{j+1} - s^*\| &\leq \hat{C}_j L_j \|s^j - s^*\|^2 + \|J^{-1}(s^j) - \hat{f}^{-1}\| \cdot \|f(s^j)\| + \|\hat{f}^{-1}\| \cdot \|f(s^j) - F(y^j)\| \\
\text{Hence} \quad &\leq \hat{C}_j L_j \|s^j - s^*\|^2 + \|\hat{f}^{-1}(\hat{f} - J(s^j))J^{-1}(s^j)\| \cdot \|f(s^j) - f(s^*)\| + \hat{C}_j 2\varepsilon_y^j \\
&\leq \hat{C}_j L_j \|s^j - s^*\|^2 + 2\hat{C}_j^2 C_j (C\varepsilon_y^j + \varepsilon_y^j) \|s^j - s^*\| + 2\hat{C}_j \varepsilon_y^j
\end{aligned}$$

◆

The previous lemmas prove that an implementation of the UMS algorithm where y^j and $Y_k^j(x)$ are computed within given accuracy at every step, is locally convergent. Moreover, this convergence is quadratic if eventually the tolerances are decreased such that

$$\varepsilon_y^j \approx \|s^j - s^*\| \quad \text{and} \quad \varepsilon_y^j \approx \|s^j - s^*\|^2$$

The other implementation where the collocation grid is kept fixed and only a few Newton updates for y^j are computed before computing a new s^j , can give at most a linear convergence rate. Since $Y_k^j(x)$ is the solution of a linear BVP the error $\varepsilon_{y,k}^j$ is fully determined by the discretization error, i.e. by the grid choice. Hence the factor $\varepsilon_y^j \|s^j - s^*\|$ is a linear term in the error estimate (4.3a).

§5 Numerical results

In the previous sections we looked at theoretical aspects of the UMS algorithm. Next we want to investigate its performance in practice. We wrote a code for first order ODE (a higher order can be reformulated into first order, see [5]) and using the existing collocation code COLNEW, cf.[1,6], to solve the local BVP's on the subintervals. Because COLNEW can only deal with separated BC the use of coupled BC increased the memory use substantially, as dummy variables have to be added to artificially separate the BC (see example 3.7). Although our numerical results show some effects to be attributed to peculiarities of COLNEW, rather than UMS, the overall results indicate a satisfactory agreement with the analysis. Yet, to understand the actual numbers more in detail we shall describe our implementation below.

Our UMS implementation has two precision parameters EPSS and $TOLF^j$. Convergence of the algorithm is declared if the norm of the update δs on the shooting vector s is less than EPSS. The parameter $TOLF^j$ is the error tolerance for the solution of the local BVP's obtained by COLNEW, hence $TOLF^j$ is equivalent to ϵ_j^j in §4. Accordingly, $TOLF^j$ is rather large at first ($10^{-2}, 10^{-3}$) and is decreased thereafter. We found that the requirement that $TOLF^j \approx \|\delta s^j\|^2$ is not always sufficient. Sometimes it occurred that $\|f(s^j)\| \leq TOLF^j$. Since most components of $f(s^j)$ are the difference between two values of y^j , these components may have no significant number at all, due to cancelation. Hence the computed direction of δs may be inaccurate. In order to prevent this, we want $TOLF^j$ to be less or equal than a tenth of the expected value of $\|f(s^{j+1})\|$, i.e. $\|f(s^j)\|^2$. This finally leads to the following algorithm to determine TOLF

$$(5.1) \quad TOLF^{j+1} = \max\left(\frac{EPSS}{10}, \min\left(TOLF^j, \frac{\|f(s^j)\|^2}{10}, \frac{\|\delta s^j\|^2}{10}\right)\right)$$

As an example singularly perturbed BVP are very well suited for application of the UMS algorithm if the position and (approximate) width of the layers can be obtained analytically. The reduced solution (i.e. for $\epsilon = 0$) is a very good initial guess for the solution at small ϵ values. It is advisable to choose the subintervals such that each one of them contains either an entire layer or a smooth region. Then the shooting vector s , obtained from the reduced solution, is quite accurate and none or very few 'outer' iterations are necessary.

When transforming a higher order singularly perturbed BVP into a first order BVP, we have to pay special attention to scaling. In some cases (e.g. example 5.3) the first derivative reaches values of approximately ϵ^{-1} . Hence we scale down the derivative term as in

$$(5.2) \quad \begin{cases} u(x) = y(x) \\ v(x) = \epsilon \dot{y}(x) \end{cases}$$

The results of applying COLNEW to singularly perturbed problems, as shown in the examples, indicate that the memory use at for $\epsilon=10^{-k}$, $k \in \{2,3,4,5\}$, is generally a multiple of that for the previous ϵ . This is due to the fact that we do not allow COLNEW to use its grid generator, but just halving the grid successively for obtaining the required tolerances. In [1] this strategy is suggested for this type of problems, because the grid generator fails to 'see' the layers at first and produces a grid on which no convergence can be obtained, leading to failure of the code. We tested several

initial grids for the forementioned ϵ -values and tabulated some results. An unintentional advantage of the UMS algorithm is that the grid generator of COLNEW worked properly on subintervals that contained a layer and a small part of a smooth region only.

5.3 Example

Consider the singularly perturbed BVP

(5.3a)

$\epsilon \ddot{y} = y(1 - \dot{y})$

(5.3b)

$y(0) = 0.5$

(5.3c)

$y(1) = 2$

$0 < x < 1$

The solution to the reduced problem is $y(x) = x+1$. Since it satisfies the end point condition, there will be a boundary layer at $x = 0$. The problem has a rather simple form and like in example 3.1 no iteration on the shooting vectors is necessary, when they are obtained from the reduced solution. Transformation (5.2) was used to convert the problem into a first order system. As we anticipated the correctness of s , we set $TOLF_u$, the required tolerance for the first variable u , to 10^{-6} . The tolerance for the second variable needed special attention. Because $v \approx \epsilon$ on a mayor part of the interval and COLNEW uses the mixed convergence criterion

$$\| \text{ absolute error in } v \| \leq TOLF_v (1 + \| v \|) \; ,$$

the variable v has only $-\log(TOLF_v / \epsilon)$ correct digits. Indeed experiments with $\epsilon = 10^{-5}$ and $TOLF_u = TOLF_v = 10^{-6}$ yielded a highly oscillatory 'solution'. To ensure that v has at least 3 correct digits we imposed the tolerance $TOLF_v = 10^{-3} \epsilon$.

The results tabulated in Table 5.1 show that the UMS algorithm saves both memory and function calls compared to COLNEW. Note, however, that the typical doubling of grid points is a COLNEW feature and is open to improvement. The subinterval choice is clearly not optimal in balancing the work load for different processors. However, splitting the layer region into several subintervals does not pay. Because then the correct value of the shooting vector is not known in advance and several iterations on s are needed.

ϵ	x_2	UMS, epss = 10^{-6}			COLNEW, tol = 10^{-6}	
		memory use	#f-calls		memory use	#f-calls
			1st	2nd interval		
1e-2	15e-2	5624	1538	18	7296	1932
1e-3	15e-3	6080	1610	18	7296	2484
1e-4	15e-4	12160	2890	18	29184	6204
1e-5	15e-5	24320	5510	18	58368	12540

Table 5.1

Acknowledgement

The authors wish to thank P.J. den Haan for programming the UMS-code and for his assistance with the numerical examples.

References

- [1] U.Ascher, J.Christiansen, R.D.Russell, A collocation solver for mixed order systems of boundary value problems. *Math. Comput.* 33 (1979) pp 659-679.
- [2] U.M.Ascher, R.M.M.Mattheij, General framework, stability and error analysis for numerical stiff boundary value methods. *Numer. Math.* 54 (1988) pp 355-372.
- [3] U.M.Ascher, R.R.M.Mattheij, R.D.Russell, Numerical solution of boundary value problems for ordinary differential equations. Englewood Cliffs : Prentice-Hall, 1988.
- [4] U.Ascher, S.Y.Pat Chan, On parallel methods for boundary value ODE's. Technical report 89-19, University of British Columbia.
- [5] U.M.Ascher, R.D.Russell, Reformulation of boundary value problems into "standard" form. *Siam Review* 23 (1981) pp 238-254.
- [6] G.Bader, U.Ascher, A new basis implementation for a mixed order boundary value ODE solver. *SIAM J. Scient. Stat. Comput* 8 (1987) pp 483-500.
- [7] J.F.Holt, Numerical solution of nonlinear two-point boundary problems by finite difference methods. *Comm. of ACM* 7 (1964), pp 366-373.
- [8] F. de Hoog, R.M.M.Mattheij, The role of conditioning in shooting techniques in : Numerical Boundary value ODE's, U.Ascher, R.Russell eds.. Birkhäuser Boston Inc. 1985.
- [9] F. de Hoog, R.M.M.Mattheij, On dichotomy and well conditioning in BVP. *SIAM J.Numer. Anal* 24 (1987), pp 89-105.
- [10] J.L.Massera, J.J.Schäffer, Linear differential equations and function spaces. New York : Academic Press, 1966.
- [11] R.E. O'Malley Jr., Introduction to singular perturbations. New York: Academic Press, 1974.
- [12] J.M.Ortega, W.C.Rheinboldt, Iterative solution of nonlinear equations in several variables. New York : Academic Press, 1970.
- [13] S.J.Wright, Stable parallel algorithms for two-point boundary value problems. To appear in SISSC.