Efficient mesh selection for collocation methods applied to singular BVPs

Winfried Auzinger
Othmar Koch
Ewa Weinmüller

Technische Universität Wien

Vienna University of Technology

Institute for Analysis and Scientific Computing

Efficient Mesh Selection for Collocation Methods Applied to Singular BVPs

W. Auzinger, O. Koch, E. Weinmüller

Institute for Analysis and Scientific Computing, Vienna University of Technology, A-1040 Vienna, Austria

Abstract

We describe a mesh selection strategy for the numerical solution of boundary value problems for singular ordinary differential equations. This mesh adaptation procedure is implemented in our MATLAB code `sbvp` which is based on polynomial collocation. We prove that under realistic assumptions our mesh selection strategy serves to approximately equidistribute the global error of the collocation solution, thus enabling to reach prescribed tolerances efficiently. Moreover, we demonstrate that this strategy yields a favorable performance of the code and compare its computational effort with other implementations of polynomial collocation.

Key words: singular BVPs, collocation methods, global error estimation, adaptive mesh selection

1 Introduction

We deal with the numerical solution of boundary value problems of the form

$$z'(t) = \frac{M(t)}{t}z(t) + f(t, z(t)), \quad t \in (0,1], \tag{1}$$

$$B_a z(0) + B_b z(1) = \beta, \tag{2}$$

$$z \in C[0,1], \tag{3}$$

where z is an n-dimensional real function, M is a smooth $n \times n$ matrix and f is an n-dimensional smooth function on a suitable domain. B_a and B_b are constant $r \times n$ matrices, with $r < n$. Condition (3) is equivalent to a set of $n-r$ linearly independent conditions $z(0)$ must satisfy. These boundary conditions are augmented by (2) to yield an isolated solution z of the problem, see [11].

To solve (1)–(3) numerically, most standard solvers for boundary value problems cannot be successfully applied. On the one hand, the methods implemented in some solvers require a special treatment of the singular point $t = 0$, since a straightforward evaluation of the right-hand side of (1) is not possible at this point, see [23]. Moreover, codes based on the shooting principle work only when the boundary value problem (1)–(3) can be formulated in the form of an equivalent, well-posed initial value problem, cf. [18]. For problems (1) this means that the coefficient matrix $M(0)$ must have no eigenvalues with positive real part. This restriction excludes some very important applications from the treatment, however. In addition, codes for regular problems are likely to use unnecessarily fine grids near the singularity to reach a prescribed tolerance. Particularly, residual control appears to be a bad choice in the presence of singularities, see [1]. Finally, many high-order methods for boundary value problems show order reductions and become inefficient when they are applied to singular problems, see for example [13].

Still, the search for an efficient numerical method to solve problems (1)–(3) is strongly motivated by numerous applications for example from physics, see [14,27], mechanics ([10,15]), or ecology, see [17,19].

Thus, we have designed the MATLAB solver `sbvp` intended especially for the solution of boundary value problems in ordinary differential equations with a singularity of the first kind (1), see [4]. This code is based on polynomial collocation, see Section 2, and equipped with an asymptotically correct a posteriori estimate of the global error described in Section 3, see also [8]. In this paper, we describe an improved version of the code, where we focus particularly on the implemented mesh selection algorithm aiming at the equidistribution of the global error. In Section 4, we describe the details of the mesh adaptation and analyze the asymptotic properties of our mesh strategy. It turns out that under realistic assumptions, the global error is indeed approximately equidistributed and tolerances are satisfied efficiently. Finally, in Section 5, we demonstrate the improvements made in our new code and compare the performance with the latest version of the MATLAB standard solver for boundary value problems `bvp4c`, which was also recently adapted to cope better with singularities, see [23].

2   Collocation methods

To compute the numerical solution of (1), `sbvp` uses collocation at an even number m of collocation points spaced equidistantly in the interior of every collocation interval. This means that on a grid

$$\{t_{i,j} = \tau_i + \rho_j(\tau_{i+1} - \tau_i), \ i = 0, \ldots, N-1, \ j = 0, \ldots, m+1\},$$

where

$$\Delta := \{\tau_i : i = 0, \ldots, N\}, \quad 0 = \tau_0 < \tau_1 \cdots < \tau_N = 1,$$
$$\rho_j = j/(m+1), \ j = 0, \ldots, m+1, \quad h_i := \tau_{i+1} - \tau_i, \ i = 0, \ldots, N-1,$$
$$h_{\max} := \max_i h_i, \quad h_{\min} = \min_i h_i,$$

we approximate the analytical solution by a continuous collocating function

$$p(t) := p_i(t), \quad t \in J_i := [\tau_i, \tau_{i+1}], \quad i = 0, \ldots, N-1.$$

Here p_i are polynomials of maximal degree m which satisfy

$$p_i'(t_{i,j}) = \frac{M(t_{i,j})}{t_{i,j}} p_i(t_{i,j}) + f(t_{i,j}, p_i(t_{i,j})), \tag{4}$$

together with the boundary conditions (2), (3).

Our decision to use collocation was motivated by its advantageous convergence properties for (1), while in the presence of a singularity other high order methods show order reductions and become inefficient (see §1). In [7,16] it is shown that the convergence order of collocation methods with polynomials of degree $\leq m$ is at least equal to the stage order m. The theory is an extension of results from [12] to nonlinear problems and to the case where $M(0)$ has eigenvalues with both negative and positive real parts. Moreover, refined error bounds necessary for the theoretical justification of our error estimation method are given in [7,16].

The restriction to an even number of equidistant collocation points is not necessary for the proofs of the convergence results described above. If we choose arbitrary collocation points $0 < \rho_1 < \cdots < \rho_m < 1$, the convergence order m is retained. However, we cannot expect superconvergence in the singular case in general even when Gaussian points are used, see [12]. Consequently, our choice of collocation nodes guarantees a reliable order m, where no superconvergence is to be expected even for regular problems. This fact also proves convenient in the discussion of the estimate for the global error of the collocation solution described in the next section.

3   A posteriori error estimation

We now construct an efficient asymptotically correct a posteriori estimate for the global error of the numerical solution obtained from (4). This estimate

was first introduced and analyzed for regular problems in [8], and is based on the defect correction principle, see for example [26]. In [7,16], the analysis of this error estimate for singular problems (1)–(3) is given. In our approach, we construct a 'neighboring problem' to (1) by adding a locally integrated defect term based on the collocation solution in the right-hand side. Subsequently, we solve both the original and the neighboring problems numerically, using the backward Euler rule at the points $t_{i,j}$, $i = 0, \ldots, N-1$, $j = 1, \ldots, m+1$ (for notational convenience, we write $t_{i,m+1} := \tau_{i+1}$). This yields the grid vectors $\xi_{i,j}$ and $\pi_{i,j}$ as the solutions of the following schemes, subject to boundary conditions (2) and (3),

$$\frac{\xi_{i,j} - \xi_{i,j-1}}{t_{i,j} - t_{i,j-1}} = \frac{M(t_{i,j})}{t_{i,j}}\xi_{i,j} + f(t_{i,j}, \xi_{i,j}), \quad \text{and} \tag{5}$$

$$\frac{\pi_{i,j} - \pi_{i,j-1}}{t_{i,j} - t_{i,j-1}} = \frac{M(t_{i,j})}{t_{i,j}}\pi_{i,j} + f(t_{i,j}, \pi_{i,j}) + \bar{d}_{i,j}, \tag{6}$$

where $\bar{d}_{i,j}$ is a defect term defined by

$$\bar{d}_{i,j} := \frac{p(t_{i,j}) - p(t_{i,j-1})}{t_{i,j} - t_{i,j-1}} - \sum_{k=1}^{m+1} \alpha_{j,k}\left(\frac{M(t_{i,k})}{t_{i,k}}p(t_{i,k}) + f(t_{i,k}, p(t_{i,k}))\right). \tag{7}$$

Here, the coefficients $\alpha_{j,k}$ are chosen in such a way that the quadrature rules given by

$$\frac{1}{t_{i,j} - t_{i,j-1}} \int\limits_{t_{i,j-1}}^{t_{i,j}} \varphi(\tau) \, \mathrm{d}\tau \approx \sum_{k=1}^{m+1} \alpha_{j,k}\varphi(t_{i,k})$$

have precision $m + 1$. The difference $\xi_{i,j} - \pi_{i,j}$ serves as the estimate of the global error of the collocation solution.

In [7,16] it is shown that the error of this error estimate satisfies

$$z(t_{i,j}) - p(t_{i,j}) - (\xi_{i,j} - \pi_{i,j}) = O(|\ln(h_{\max})|^{n_0-1}h_{\max}^{m+1})$$

for some positive integer n_0. Since $z(t_{i,j}) - p(t_{i,j}) = O(h_{\max}^m)$ in general, this means that our error estimate is asymptotically correct. In most cases relevant in applications we have $n_0 = 1$, so for reasons of simplicity we only consider the case where the error of the error estimate satisfies

$$z(t_{i,j}) - p(t_{i,j}) - (\xi_{i,j} - \pi_{i,j}) = O(h_{\max}^{m+1})$$

in the sequel.

4

4 The mesh adaptation algorithm

In this section, we discuss our mesh adaptation algorithm which aims at the equidistribution of the global error of the numerical solution. This procedure is similar to the ideas discussed in [20,22]. In contrast to these earlier approaches, the quantity we intend to equidistribute is not some mesh independent quantity related to the global error of the numerical solution computed by collocation, but the global error itself, or rather, its asymptotically correct estimate. The question of the mesh independence of this quantity is discussed below.

Consider a non-uniform mesh Δ. On this mesh, we compute the continuous approximate solution $p(t)$, $t \in [0,1]$, and assume that the asymptotically correct estimate for the global error is also available for $t \in [0,1]$. This means that we choose some suitable interpolant of $\xi_{i,j} - \pi_{i,j}$. Denote by $e(t) := |z(t) - p(t)|$ the absolute value of the global error of the numerical solution, and by $\varepsilon(t)$ the absolute value of the error estimate.

Since the estimate is asymptotically correct, we may write

$$e(t) = h_i^m \frac{\varepsilon(t)}{h_i^m} + O(h_{\max}^{m+1}) = h_i^m \left(\frac{\Theta(t)}{h_i} \right)^m + O(h_{\max}^{m+1}), \quad t \in J_i, \tag{8}$$

where

$$\Theta(t) := \sqrt[m]{\varepsilon(t)}, \quad t \in [0,1], \tag{9}$$

and m is the order of the underlying collocation method.

Let $g(t)$ be defined as the continuous piecewise linear function satisfying $g(\tau_i) = i/N$, $i = 0, \ldots, N$. The function g is monotonously increasing with function values in $[0,1]$, and its piecewise constant derivative

$$\rho(t) := g'(t) = \frac{1}{N h_i}, \quad t \in J_i, \ i = 0, \ldots, N-1, \tag{10}$$

represents the local grid density. Note that

$$\int_0^1 \rho(t) \, \mathrm{d}t = 1$$

holds. Conversely, for a given step function $\rho(t)$ the corresponding mesh Δ

5

can be reconstructed for given N by integrating ρ,

$$g(t) := \int_0^t \rho(\tau)\,d\tau$$

and letting

$$\tau_i := g^{-1}\left(\frac{i}{N}\right).$$

This will be used in the mesh selection algorithm described below.

Furthermore, we define

$$\tilde{\Theta}(t) := \frac{\Theta(t)}{\sqrt[m]{\min_{\tau\in[0,1]}(aTOL + rTOL|p(\tau)|)}}, \tag{11}$$

$$\widehat{\rho\Theta}(t) := \max\left\{\rho(t)\tilde{\Theta}(t), \frac{\max_{\tau\in[0,1]}\rho(\tau)\tilde{\Theta}(\tau)}{K}\right\}, \tag{12}$$

$$I := \int_0^1 \widehat{\rho\Theta}(t)\,dt. \tag{13}$$

In (11), $aTOL$ and $rTOL$ denote absolute and relative tolerance parameters used for the mixed tolerances. The definition (12) implies that $h_{\max}/h_{\min} \leq K$ (see Remark 2).

Now, the mesh redistribution is organized as follows:

$$\rho(t) \mapsto \frac{1}{I}\widehat{\rho\Theta}(t) =: \tilde{\rho}(t), \tag{14}$$

$$N \mapsto \max\{1.5N, (1+\delta)IN\} =: \tilde{N}, \tag{15}$$

where δ is a safety margin which we set to $\delta := 0.1$. Note that (14) implies

$$\int_0^1 \tilde{\rho}(t)\,dt = 1. \tag{16}$$

The new mesh $\tilde{\Delta} := (\tilde{\tau}_0, \dots, \tilde{\tau}_{\tilde{N}})$ is computed according to

$$\tilde{g}'(t) := \tilde{\rho}(t), \tag{17}$$

$$\tilde{\tau}_j := \tilde{g}^{-1}\left(\frac{j}{\tilde{N}}\right), \quad j = 0, \dots, \tilde{N}. \tag{18}$$

6

Note that

$$\tilde{g}(0) = 0, \quad \tilde{g}(1) = \int\limits_0^1 \tilde{\rho}(t) \, \mathrm{d}t = 1, \tag{19}$$

$$\tilde{g}(\tilde{\tau}_j) = \frac{j}{\tilde{N}} \Rightarrow \tilde{g}(\tilde{\tau}_j) - \tilde{g}(\tilde{\tau}_{j-1}) = \int\limits_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \tilde{\rho}(t) \, \mathrm{d}t = \frac{1}{I} \int\limits_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \widehat{\rho\Theta}(t) \, \mathrm{d}t = \frac{1}{\tilde{N}}. \tag{20}$$

Thus, our mesh adaptation amounts to the equidistribution of

$$\int \widehat{\rho\Theta}(t) \, \mathrm{d}t$$

on the mesh $\tilde{\Theta}$.

Remark 1 *We use the trapezoidal rule to approximate $\tilde{g}$,*

$$\tilde{g}(\tau_i) = \int\limits_0^{\tau_i} \tilde{\rho}(t) \, \mathrm{d}t \approx \sum_{l=0}^{i-1} \frac{\tilde{\rho}(\tau_{l+1}) + \tilde{\rho}(\tau_l)}{2}(\tau_{l+1} - \tau_l). \tag{21}$$

To compute $\tilde{\tau}_j = \tilde{g}^{-1}\left(\frac{j}{\tilde{N}}\right)$, we use cubic splines interpolating $(\tilde{g}(\tau_i), \tau_i)$.

Remark 2 *Note that definition* (12) *implies a bound on the variation of step sizes: Let $M := \max_{\tau \in [0,1]} \rho(\tau)\tilde{\Theta}(\tau)$. Then*

$$\frac{1}{I}(\tilde{\tau}_j - \tilde{\tau}_{j-1})\frac{M}{K} \le \int\limits_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \tilde{\rho}(t) \, \mathrm{d}t = \frac{1}{\tilde{N}} \int\limits_0^1 \tilde{\rho}(t) \, \mathrm{d}t \le \frac{1}{I}(\tilde{\tau}_k - \tilde{\tau}_{k-1})M, \quad \forall j, k.$$

Consequently,

$$\frac{h_{\max}}{h_{\min}} \le K.$$

As an example to demonstrate that this strategy to bound the variation in the step sizes indeed works and is necessary for a stable numerical solution in the context of boundary value problems, we consider the linear test problem

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 1 + \alpha^2 t^2 & 0 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ ct^{k-1} \, \mathrm{e}^{-\alpha t}(k^2 - 1 - \alpha t(1 + 2k)) \end{pmatrix}, \tag{22}$$
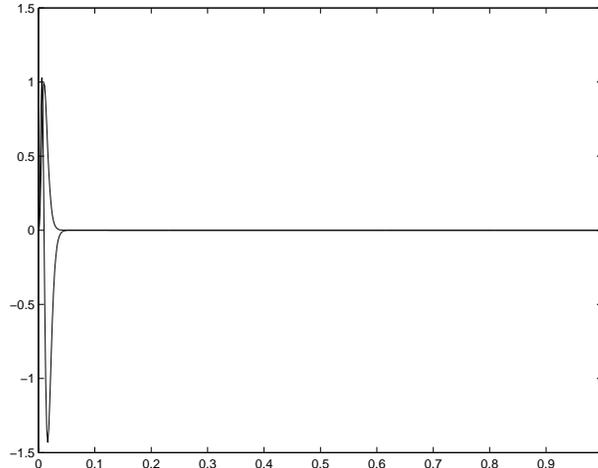
Fig. 1. Solution of (22), (23).

$$
\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ c \, \mathrm{e}^{-\alpha} \end{pmatrix}, \tag{23}
$$

where $\alpha = 400$, $k = 4$ and $c = \left(\frac{\alpha}{k}\right)^k \mathrm{e}^k$. The exact solution of this test problem is

$$
z(t) = (ct^k \, \mathrm{e}^{-\alpha t}, ct^k \, \mathrm{e}^{-\alpha t}(k - \alpha t))^T.
$$

A plot of the solution profile is given in Figure 1.

All computations reported in this paper were performed with MATLAB 6.5 R13 on a personal computer in IEEE double precision arithmetic with relative machine accuracy $\approx 1.11 \cdot 10^{-16}$.

Table 1 gives the value 'ratio' $= h_{\max}/h_{\min}$ and the number of mesh points N in the final mesh to reach tolerances $aTOL = rTOL = 10^{-8}$ for (22) and (23), and additionally the runtime in seconds required for the whole computation. We notice that the variation in the step sizes all but assumes its theoretical maximum, and the value $K = 100$ yields the best results. While a conservative choice $K = 10$ implies that too many mesh points are used in the region where the solution is smooth, a large variation in the step sizes with $K = 1000$ makes the solution process unstable and the efficiency deteriorates inacceptably. We found that $K = 100$ provided good results throughout all test runs and therefore this value is the default for our code, see [6].

Next, we show that under realistic assumptions, the global error of the numerical solution is indeed approximately equidistributed. The following definition is motivated by similar considerations in [20,22].

8

Table 1
Computations for (22), (23), variation of K, tolerances 10^{-8}.

K	ratio	N	runtime
10	9.127	140	0.681
100	94.668	96	0.431
1000	959.218	576	4.596

Definition 3 *A sequence of meshes Δ is called* asymptotically equidistributing, *if*

$$\max_{t \in J_i} |e(t)| = \mathcal{C}(1 + O(h_{\max})), \quad i = 0, \ldots, N - 1, \tag{24}$$

where we use $\mathcal{C}$ as a generic constant.

If we assume that the step size restriction resulting from the introduction of the parameter K in (12) does not actually occur in a practical situation, that is,

$$\widehat{\rho\tilde{\Theta}}(t) = \rho(t)\tilde{\Theta}(t), \tag{25}$$

our approach guarantees that

$$\int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \tilde{\rho}(t) \, \mathrm{d}t = \frac{1}{I\tilde{N}} \int_0^1 \rho(t)\tilde{\Theta}(t) \, \mathrm{d}t = \frac{1}{\tilde{N}}. \tag{26}$$

If we define the step function $h(t) := h_i, \ t \in J_i, \ i = 0, \ldots, N - 1$, we thus obtain

$$\int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \frac{\Theta(t)}{h(t)} \, \mathrm{d}t = \frac{IN}{\tilde{N}} \sqrt[m]{\min_{\tau \in [0,1]} (aTOL + rTOL|p(\tau)|)}$$

$$\leq \frac{1}{1 + \delta} \sqrt[m]{\min_{\tau \in [0,1]} (aTOL + rTOL|p(\tau)|)}. \tag{27}$$

Finally, we may conclude that for a grid computed from (14) and (15), where $\Delta = \tilde{\Delta}$ (that is, the algorithm recognizes the tolerances to be satisfied), we have

$$\int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \frac{\Theta(t)}{h(t)} \, \mathrm{d}t = \int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \sqrt[m]{\frac{\varepsilon(t)}{h^m(t)}} \, \mathrm{d}t = \int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \sqrt[m]{\frac{e(t)}{h^m(t)} (1 + O(h_{\max}^{m+1}))} \, \mathrm{d}t$$

9

$$= \int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \frac{1}{h(t)} \sqrt[m]{e(t)} \, dt \, (1 + O(h_{\max})) = \int_{\tilde{\tau}_{j-1}}^{\tilde{\tau}_j} \frac{1}{h(t)} \sqrt[m]{e(\xi_j)} \, dt \, (1 + O(h_{\max}))$$

$$= \sqrt[m]{\max_{t \in \tilde{J}_j} e(t)} \, (1 + O(h_{\max})) = \frac{1}{1 + \delta} \sqrt[m]{\min_{\tau \in [0,1]} (aTOL + rTOL|p(\tau)|)}, \quad (28)$$

where ξ_j is the point such that

$$\max_{t \in \tilde{J}_j} e(t) = e(\xi_j),$$

which implies

$$e(t) = e(\xi_j)(1 + O(h_{\max})).$$

This means that

$$\max_{t \in \tilde{J}_j} e(t) = \left(\frac{1}{1 + \delta}\right)^m \min_{\tau \in [0,1]} (aTOL + rTOL|p(\tau)|)(1 + O(h_{\max})), \quad (29)$$

and the mesh $\Delta = \tilde{\Delta}$ which results from our mesh selection procedure is asymptotically equidistributing according to Definition 3. The assumptions used in this derivation are the following:

(i) $e \in C^1[0, 1]$, and

$$\lim_{h_{\max} \to 0} \max_{t \in [0,1]} \frac{|e'(t)|}{|e(t)|} \le \text{const.}$$

This condition is satisfied for instance if a smooth principal error function exists for the collocation solution $p(t)$, cf. [25]. While this has not yet been proven theoretically for singular problems, experimental evidence suggests this assumption to be justified.

(ii) For (29) to hold, we require in (28)

$$\int_0^1 \frac{\Theta(t)}{h(t)} \, dt = \mathcal{C}, \quad (30)$$

or at least

$$\int_0^1 \frac{\Theta(t)}{h(t)} \, dt = \mathcal{C}(1 + O(h_{\max})),$$

which is equivalent to the fact that $\frac{e(t)}{h^m(t)}$ does not depend on the step size h_i or the mesh Δ in the limit for $h_{\max} \to 0$.

10

We observed that for coherent mesh refinement, where the mesh density ρ is fixed, $\lim\limits_{h_{\max}\to 0} \frac{e(t)}{h^m(t)}$ exists, so $\frac{e(t)}{h^m(t)}$ is indeed independent of $h_{\max}$ for sufficiently small $h_{\max}$. Unfortunately, for non-coherent mesh refinement $\lim\limits_{h_{\max}\to 0} \frac{e(t)}{h^m(t)}$ in fact depends on $\rho(t)$. However, it is realistic to assume that

$$\mathcal{C}(1-\epsilon) \le \int\limits_0^1 \frac{\Theta(t)}{h(t)}\,\mathrm{d}t \le \mathcal{C}(1+\epsilon) \tag{31}$$

with small ϵ. In this case, (29) can be replaced by

$$(1-\epsilon)^m \le \frac{(1+\delta)^m \max_{t\in\tilde{J}_j} e(t)}{\min_{\tau\in[0,1]}(aTOL + rTOL|p(\tau)|)}(1+O(h_{\max})) \le (1+\epsilon)^m.$$

The favorable performance of our mesh selection algorithm demonstrated in Section 5 shows that in practical situations, the assumptions above appear to be justified. To illustrate the validity of assumption (31), in Table 2 we give the values of the integral from (30) in the following situation: The solution of the test problem (22) and (23) is computed with four equidistant collocation points. In our code, the decision whether the mesh is redistributed according to (14), or coherent mesh refinement is chosen (i.e., $\tilde{\rho} = \rho$), depends on a parameter, see §5. We prescribe an unrealistically strict tolerance in order to ensure that the mesh selection process consists of a number of nontrivial steps. At the starting mesh with $N = 100$, mesh redistribution takes place (we denote this by 'refinement' taking the value 'redis'). The parameters are chosen such that in the further steps, our mesh selection algorithm decides to use coherent refinement ('refinement' takes the value 'coh'). Table 2 gives the values of the integrals (30) for each mesh of size N in the mesh selection procedure, where J denotes this quantity resulting from coherent mesh refinement, while I is the value of this integral which would result from a redistribution of the mesh. In the first step, naturally $I = J$ because the mesh is actually redistributed, while in the subsequent adaptation steps the values J that actually occur in the solution process converge to a constant value as required in (30) (which we have normalized to 1, see (16)), while redistribution in the respective adaptation steps would result in a small variation in the values of I. This variation is not very large, however, so we may expect our mesh adaptation procedure to work dependably even if the mesh is redistributed at some point during mesh adaptation. We conclude that for coherent mesh refinement, the assumption (30) is justified, while mesh redistribution results in (31) with a small value of ϵ.

Remark 4 *Note that if (25) does not hold, that is, the step size limitation using the parameter K in (12) takes effect, then the step sizes may be smaller*

Table 2

The integral (30) for the mesh selection process for (22), (23).

N	refinement	J	I
100	redis	$1.2553 \cdot 10^1$	$1.2553 \cdot 10^1$
1394	coh	$9.9970 \cdot 10^{-1}$	$9.7803 \cdot 10^{-1}$
2893	coh	$9.9985 \cdot 10^{-1}$	$8.3784 \cdot 10^{-1}$
4964	coh	$9.9991 \cdot 10^{-1}$	$9.1316 \cdot 10^{-1}$
9643	coh	$9.9995 \cdot 10^{-1}$	$8.8796 \cdot 10^{-1}$
18275	coh	$9.9998 \cdot 10^{-1}$	1.2412

in some regions and (29) does not hold. Rather, the error is smaller than predicted by (29) in these areas.

5 Performance comparisons

A description of the implementation of our algorithms in the original code is given in [4]. Moreover, this reference attempts a comparison with other standard collocation solvers. More technical details of the code, a description of its usage and a broader survey of experimental results are given in [2,3]. In [6], a more extensive study is conducted. Here, we give a few examples illustrating the improvements in the performance of the new version of our code.

We demonstrated in §4 that a redistribution of the mesh – while potentially enabling to reach a prescribed tolerance efficiently, that is, with as few mesh points as possible – introduces a certain amount of uncertainty in the solution process, because the integral (30) is no longer constant. In our original code [4], we allowed for only one mesh redistribution and used coherent mesh refinement later on. In the new version, we attempt to find a balance between the possible inefficiency of coherent mesh refinement and the uncertainty introduced by mesh redistribution. To this end, we introduced a parameter r, which determines whether the mesh is redistributed or refined coherently: The number of points to reach the tolerances predicted according to the formula (15) is computed for both the integral I occurring when the mesh is redistributed, and alternatively using J instead of I for the scenario of coherent mesh refinement. If $\tilde{N}$ denotes the number of mesh points for redistribution and $\hat{N}$ for coherent refinement, then the mesh is redistributed if
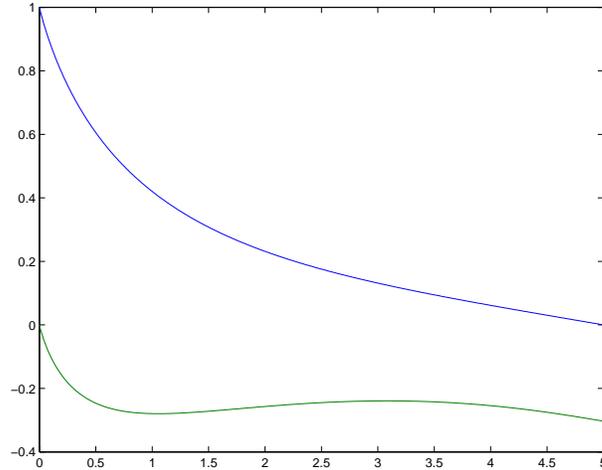
$$\tilde{N} \leq (1 - r)\hat{N}, \tag{32}$$

Fig. 2. Solution of (33), (34).

Table 3

Variation of mesh redistribution parameter r for (33), (34).

r	N	runtime
0.1	149	1.642
0.25	203	2.193
0.33	288	3.405
0.5	288	3.275

and refined coherently otherwise. We found that on average for a large set of singular test problems, the best choice was $r = 0.1$, see [6]. To illustrate this observation, we report the results for the following test example taken from [21]:

$$z'(t) = \frac{1}{t} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ t^{1/2} z_1(t) \end{pmatrix}, \tag{33}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(b) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{34}$$

with $b = 5$. The solution is plotted in Figure 2.

Table 3 shows the number of mesh points N in the final mesh and the runtimes for (33), (34), where tolerances $aTOL = rTOL = 10^{-6}$ were prescribed and in (32) $r = 0.1,\ 0.25,\ 0.33,\ 0.5$ was chosen, respectively. For more examples refer to [6].

The monitor function $\tilde{\Theta}$ from (11) could also be demonstrated to yield better

13

results than the one implemented in our original code [4]. There, we used

$$\tilde{\Theta}_2(t) := \frac{\Theta(t)}{\sqrt[m]{aTOL + rTOL|p(t_{i,j})|}}, \tag{35}$$

where $t_{i,j}$ is the grid point such that

$$\max_{k,l} \frac{\Theta(t_{k,l})}{\sqrt[m]{aTOL + rTOL|p(t_{k,l})|}} = \frac{\Theta(t_{i,j})}{\sqrt[m]{aTOL + rTOL|p(t_{i,j})|}}. \tag{36}$$

A third variant for the monitor function was to choose the denominator dependent of t,

$$\tilde{\Theta}_3(t) := \frac{\Theta(t)}{\sqrt[m]{aTOL + rTOL|p(t)|}}. \tag{37}$$

It turned out that our original choice did not perform as well as (11). The monitor function (37), although comparable in performance for most problems, did not guarantee a stable solution process, see [6]. To illustrate this observation, we give the results for the following linear test problem:

$$z'(t) = \frac{1}{t} \begin{pmatrix} 0 & 1 \\ 2 & 6 \end{pmatrix} z(t) - \begin{pmatrix} 0 \\ 4k^4 t^5 \sin(k^2 t^2) + 10t \sin(k^2 t^2) \end{pmatrix}, \tag{38}$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ \sin(k^2) \end{pmatrix}, \tag{39}$$

where $k = 8$. A plot of the exact solution

$$z(t) = (t^2 \sin(k^2 t^2), 2k^2 t^4 \cos(k^2 t^2) + 2t^2 \sin(k^2 t^2))^T$$

is given in Figure 3. In Table 4, we give for the three choices of the monitor function (the value $r = 0.1$ is chosen in each case) the number N of points in the final mesh and the runtimes for the solution of (38), (39) with tolerances $aTOL = rTOL = 10^{-3}$. It turns out that for this example $\tilde{\Theta}$ is indeed the best choice. In general, the results are not as clearly in favor of $\tilde{\Theta}$, but on average it yields the most reliable and efficient results, see [6].

Finally, we give a comparison of the efficiency of our new version of sbvp as compared with the published code [4]. We also include the MATLAB standard solver for boundary value problems bvp4c available with MATLAB 6.5
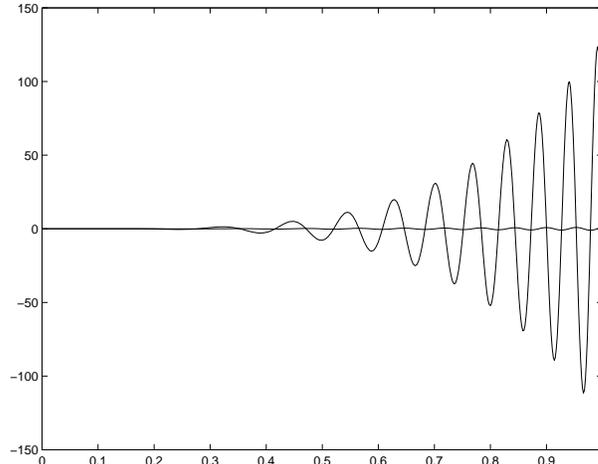
Fig. 3. Exact solution of (38), (39).

Table 4
Variation of monitor function $\tilde{\Theta}$ for (38), (39).

monitor function	N	runtime
$\tilde{\Theta}$	90	0.170
$\tilde{\Theta}_2$	114	0.260
$\tilde{\Theta}_3$	135	0.351

R13 in this discussion, see [23,24]. This code is not intended for high efficiency, and uses collocation with fixed order four, while our code uses higher order methods for stricter tolerances, see [5,6]. Consequently, `bvp4c` cannot be expected to outperform our code for strict tolerances, but is included as a standard reference. The complete results of the performance comparisons are given in [6]. For a sample of 26 test problems, we measured the runtime of the codes for three different tolerances, $aTOL = rTOL = 10^{-3}$, 10^{-6}, 10^{-8}, respectively. Table 5 gives the following indicators for the performance of the codes: For each code, we give the number of test runs (out of a total of 78) where the code showed the shortest runtime ('fastest'), the longest runtime ('slowest') and the examples where the respective code failed altogether ('failure'). The last two lines give the number of examples where the fastest code in the respective test configuration was no more than 10 percent faster than the code considered ('<10% slower'), and the number of test situations where the fastest code was more than 40 percent faster ('>40% slower'). The complete results given in [6] show that for the mild tolerances $aTOL = rTOL = 10^{-3}$, the results are rather mixed, while the new version of `sbvp` demonstrates its advantages particularly for the strict tolerances $aTOL = rTOL = 10^{-8}$ and for difficult problems, where the solution process requires an efficient mesh selection strategy.

15

Table 5
Performance comparisons.

	sbvp new	sbvp old	bvp4c
fastest	43	11	24
slowest	3	30	45
failure	0	1	11
<10% slower	56	17	14
>40% slower	9	28	44

6   Conclusions and Outlook

In this paper, we have discussed a mesh selection strategy aiming at the equidistribution of the global error of the numerical solution of singular boundary value problems computed by collocation. We have shown that under realistic assumptions, this leads to a mesh which indeed approximately equidistributes the global error and allows for an efficient numerical solution where the tolerances are satisfied on a mesh with a small number of mesh points. Moreover, the favorable performance of our MATLAB code sbvp based on this mesh selection procedure was demonstrated. We have excluded superconvergent collocation methods from our discussion here, because for singular problems superconvergence cannot be expected to hold in general, see [12]. However, for regular problems with steep layers or singularly perturbed problems, the more distinct localization of the global error in the case of, for example, Gaussian points could be advantageous for our mesh selection procedure, see [22]. This will be the subject of future investigations.

References

[1]   W. Auzinger, E. Karner, O. Koch, D. Praetorius and E. Weinmüller, Globale Fehlerschätzer für Randwertprobleme mit einer Singularität zweiter Art, Techn. Rep. ANUM Preprint Nr. 6/03, Inst. for Appl. Math. and Numer. Anal., Vienna University of Technology, Austria, 2003.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[2] W. Auzinger, G. Kneisl, O. Koch and E. Weinmüller, SBVP 1.0 — A MATLAB solver for singular boundary value problems, Techn. Rep. ANUM Preprint Nr. 2/02, Inst. for Appl. Math. and Numer. Anal., Vienna University. of Technology, Austria, 2002.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[3] W. Auzinger, G. Kneisl, O. Koch and E. Weinmüller, A solution routine for singular boundary value problems, Techn. Rep. ANUM Preprint Nr. 1/02, Inst. for Appl. Math. and Numer. Anal., Vienna University of Technology, Austria, 2002.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[4] W. Auzinger, G. Kneisl, O. Koch and E. Weinmüller, A collocation code for boundary value problems in ordinary differential equations, *Numer. Algorithms* **33** (2003) 27–39.

[5] W. Auzinger, O. Koch, W. Polster and E. Weinmüller, Ein Algorithmus zur Gittersteuerung bei Kollokationsverfahren für singuläre Randwertprobleme, Techn. Rep. ANUM Preprint Nr. 21/01, Inst. for Appl. Math. and Numer. Anal., Vienna University of Technology, Austria, 2001.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[6] W. Auzinger, O. Koch, D. Praetorius, G. Pulverer and E. Weinmüller, Performance of collocation software for singular BVPs, Techn. Rep. ANUM Preprint Nr. 4/04, Inst. for Anal. and Sci. Comput., Vienna University of Technology, Austria, 2004.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[7] W. Auzinger, O. Koch and E. Weinmüller, Analysis of a new error estimate for collocation methods applied to singular boundary value problems, *SIAM J. Numer. Anal.*, to appear.

(Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.)

[8] W. Auzinger, O. Koch and E. Weinmüller, Efficient collocation schemes for singular boundary value problems, *Numer. Algorithms* **31** (2002) 5–25.

[9] W. Auzinger, P. Kofler and E. Weinmüller, Steuerungsmaßnahmen bei der numerischen Lösung singulärer Anfangswertaufgaben, Techn. Rep. Nr. 124/98, Inst. for Appl. Math. and Numer. Anal., Vienna University of Technology, 1998.

(Available at `http://www.math.tuwien.ac.at/~ewa/`.)

[10] M. Drmota, R. Scheidl, H. Troger and E. Weinmüller, On the imperfection sensitivity of complete spherical shells, *Comp. Mech.* **2** (1987) 63–74.

[11] F. d. Hoog and R. Weiss, Difference methods for boundary value problems with a singularity of the first kind, *SIAM J. Numer. Anal.* **13** (1976) 775–813.

[12] F. d. Hoog and R. Weiss, Collocation methods for singular boundary value problems, *SIAM J. Numer. Anal.* **15** (1978) 198–217.

[13] F. d. Hoog and R. Weiss, The application of Runge-Kutta schemes to singular initial value problems, *Math. Comp.* **44** (1985) 93–103.

[14] T. Kapitula, Existence and stability of singular heteroclinic orbits for the Ginzburg-Landau equation, *Nonlinearity* **9** (1996) 669–685.

[15] H. Keller and A. Wolfe, On the nonunique equilibrium states and buckling mechanism of spherical shells, *SIAM J.* 13 (1965) 674–705.

[16] O. Koch, Asymptotically correct error estimation for collocation methods applied to singular boundary value problems, submitted to Numer. Math.

[17] O. Koch and E. Weinmüller, Analytical and numerical treatment of a singular initial value problem in avalanche modeling, *Appl. Math. Comput.* **148** (2003) 561–570.

[18] O. Koch and E. Weinmüller, The convergence of shooting methods for singular boundary value problems, *Math. Comp.* **72** (2003) 289–305.

[19] D. M. McClung and A. I. Mears, Dry-flowing avalanche run-up and run-out, *J. Glaciol.* **41** (1995) 359–369.

[20] V. Pereyra and E. Sewell, Mesh selection for discrete solution of boundary problems in ordinary differential equations, *Numer. Math.* **23** (1975) 261–268.

[21] P. Rentrop, Eine Taylorreihenmethode zur numerischen Lösung von Zwei-Punkt Randwertproblemen mit Anwendung auf singuläre Probleme der nichtlinearen Schalentheorie, TUM-MATH-7733, Technische Universität München, 1977.

[22] R. D. Russell and J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, *SIAM J. Numer. Anal.* **15** (1978) 59–80.

[23] L. Shampine, Singular boundary value problems in odes, *Appl. Math. Comput.* **138** (2003) 99–112.

[24] L. Shampine, J. Kierzenka and M. Reichelt, Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with `bvp4c`,

(Available at `ftp://ftp.mathworks.com/pub/doc/papers/bvp/`.)

[25] H. J. Stetter, *Analysis of Discretization Methods for Ordinary Differential Equations*, Springer-Verlag, Berlin-Heidelberg-New York, 1973.

[26] H. J. Stetter, The defect correction principle and discretization methods, *Numer. Math.* **29** (1978), 425–443.

[27] C.-Y. Yeh, A.-B. Chen, D. Nicholson and W. Butler, Full-potential Korringa-Kohn-Rostoker band theory applied to the Mathieu potential, *Phys. Rev. B* **42** (1990) 10976–10982.