Automatic Grid Control in Adaptive BVP Solvers

Gernot Pulverer Gustaf Söderlind Ewa Weinmüller

8. April 2009

Abstract

Grid adaptation in two-point boundary value problems is usually based on mapping a uniform auxiliary grid to the desired nonuniform grid. Here we combine this approach with a new control system for constructing a grid density function $\phi(x)$. The local mesh width $\Delta x_{j+1/2} = x_{j+1} - x_j$ with $0 = x_0 < x_1 < \cdots < x_N = 1$ is computed as $\Delta x_{j+1/2} = \epsilon_N/\varphi_{j+1/2}$, where $\{\varphi_{j+1/2}\}_0^{N-1}$ is a discrete approximation to the continuous density function $\phi(x)$, representing mesh width variation. The parameter $\epsilon_N = 1/N$ controls accuracy via the choice of N. For any given grid, a solver provides an error estimate. Taking this as its input, the feedback control law then adjusts the grid, and the interaction continues until the error has been equidistributed. Digital filters may be employed to process the error estimate as well as the density to ensure the regularity of the grid. Once $\phi(x)$ is determined, another control law determines N based on the prescribed tolerance TOL. The paper focuses on the interaction between control system and solver, and the controller's ability to produce a near-optimal grid in a stable manner as well as correctly predict how many grid points are needed. Numerical tests demonstrate the advantages of the new control system within the `bvpsuite` solver, *ceteris paribus*, for a selection of problems and over a wide range of tolerances. The control system is modular and can be adapted to other solvers and error criteria.

1 Introduction

We shall consider automatic grid control for two-point boundary value problems (2p-BVPs) of the form

$$y'(x) = F(x, y), \quad x \in [0, 1] \tag{1}$$

and boundary conditions $B_0 y(0) + B_1 y(1) = \beta$. Most modern adaptive techniques use some variant of the following idea: we introduce a new auxiliary variable ξ, and a grid deformation map that maps a uniform grid in ξ to the desired nonuniform grid in the original independent variable x. In fact, every adaptive technique can be represented in this way, as the nonuniform grid trivially can be (invertibly) mapped to a uniform grid, by merely using the lexicographic ordering of the grid points.

In a rigorous mathematical setting one requires the map to be a diffeomorphism. It can be described in two equivalent ways. The first approach seeks a map $x = x(\xi)$, subject to the boundary conditions

$$x(0) = 0, \quad x(1) = 1, \tag{2}$$

with derivative $\mathrm{d}x/\mathrm{d}\xi$ denoted by x_ξ. Imposing the condition $0 < x_\xi < \infty$ implies that x is a monotone, invertible, differentiable function of ξ.

The second approach expresses ξ as a function of x and hence works with the inverse map $\xi(x)$. Its derivative $\mathrm{d}\xi/\mathrm{d}x$ is denoted by ξ_x, and it holds that $\xi_x = 1/x_\xi$. Further, the boundary conditions of the inverse transformation are

$$\xi(0) = 0, \quad \xi(1) = 1. \tag{3}$$

For an arbitrary $N > 0$, we introduce the grid $\Xi_N = \{\xi_j\}_{j=0}^N$, with equidistant grid points $\xi_j = j/N$ and mesh width $\epsilon_N = 1/N$. The deformed grid $X_N = \{x_j\}_{j=0}^N$ is the image of the *grid deformation map* $x : \Xi_N \mapsto X_N$. For the inverse map $\xi : X_N \mapsto \Xi_N$ it holds that

$$\Delta\xi_{j+1/2} = \xi(x_{j+1}) - \xi(x_j) \approx \xi_x(x_{j+1/2}) \cdot \Delta x_{j+1/2}, \tag{4}$$

which is a discrete analogue of the differential relation $\mathrm{d}\xi = \xi_x \,\mathrm{d}x$. We prefer a half-index notation for the mesh width, so that it is immediately clear what interval the step size refers to. Using $\Delta\xi_{j+1/2} = \epsilon_N$ and introducing the notation $\phi(x) \equiv \xi_x(x)$, [22], we have

$$\Delta x_{j+1/2} \approx \frac{\epsilon_N}{\phi(x_{j+1/2})}. \tag{5}$$

The greater the value of $\phi(x)$, the more dense are the grid points near x. The function $\phi(x)$ is therefore referred to as the *grid density function*. Its primitive function, $\xi(x)$, is the corresponding *grid distribution function*. Its boundary conditions imply that the density satisfies the normalization requirement

$$\xi(1) - \xi(0) = \int_0^1 \phi(x)\,\mathrm{d}x = 1. \tag{6}$$

In practice the nonuniform grid is generated by constructing a discrete approximation to ϕ, in terms of a positive sequence $\Phi_N = \{\varphi_{j+1/2}\}_{j=0}^{N-1}$, such that

$$\Delta x_{j+1/2} = \frac{\epsilon_N}{\varphi_{j+1/2}}. \tag{7}$$

Here (7) holds exactly, as opposed to (5). The discrete density function Φ_N then satisfies the normalization requirement

$$\sum_{j=0}^{N-1} \frac{\epsilon_N}{\varphi_{j+1/2}} = 1, \tag{8}$$

which imposes the condition that the step sizes exactly cover the entire interval $[0, 1]$. It is important to note that the discrete normalization (8) differs from the continuous normalization (6).

For two-point boundary value problems, an adaptive algorithm must determine the sequence Φ_N in terms of problem or solution properties. A simple technique is to construct the grid density so that the solution y of (1) has equal arclength over each subinterval. An arclength estimate can be obtained by first solving the problem on a coarse grid, after which the arclength of the computed solution is used to prepare a suitable nonuniform grid for efficient high precision computations. This approach is common e.g. in moving mesh algorithms for hyperbolic problems where a locally high grid density is required to resolve a moving shock wave.

Arclength equidistribution does not control errors, however. Instead, true adaptive approaches equidistribute some residual or error estimate over the interval. As Φ_N will depend on the error

estimates, which in turn depend on the distribution of the grid points, the process of finding the density becomes *iterative*. For some error control criteria, however, a local grid change typically has global effects. The techniques developed here avoid this difficulty by restricting the error estimators to those having the property that the estimated error on the interval $[x_j, x_{j+1}]$ only depends on the local mesh width, $\Delta x_{j+1/2} = \epsilon_N/\varphi_{j+1/2}$.
We refer to finding a suitable transformation $x(\xi)$ – or the corresponding density function ϕ – as *mesh generation*. Choosing a suitable number of grid points – or equivalently, choosing ϵ_N – is referred to as *mesh refinement*. As ϕ is found iteratively, we may regard this process as (pseudo) time-dependent. It is linked to *moving mesh* algorithms in time-dependent PDEs; this notion refers to the case when $x(\xi)$ is a time dependent grid deformation but where the number of grid points is kept constant.
For a *given* N (corresponding to a fixed computational effort), the density ϕ affects accuracy; the goal is to minimize the error by seeking the optimal distribution X_N of the N grid points. Conversely, for a *given* ϕ, it is—at least in principle—straightforward to solve the problem to a desired accuracy by taking a sufficiently large number of interior grid points, $N-1$. Thus the approach outlined above allows both *adaptivity*, in terms of finding ϕ, and *convergence* studies, by letting $N \to \infty$ (or $\epsilon_N \to 0$). In this setting, a rigorous convergence proof for an adaptive method is within reach.

A brief survey of previous work. In the last decades a large effort has been put into the efficient numerical solution of ordinary and partial differential equations. The key technique to saving work and speeding up computations is grid adaptation, which even becomes crucial when solving problems in 2D and 3D. Although there is a rich literature on adaptivity, the main principles have not changed over the years and constitute universal techniques which can be adapted to specific needs. One of these is the *equidistribution principle* that was studied in two different contexts in [8, 10] as a technique to solve differential equations, and to minimize the interpolation error of a known function, respectively, cf. [15, 20, 21]. The concept of *grading functions* and the convergence of the remeshing iteration have been discussed in [14] and [27], respectively. Here, restricting ourselves to boundary value problems in ordinary differential equations, we shall describe a few well established and interesting adaptive approaches in terms of the notions and notations introduced above.
In [17], Christara and Ng consider adaptive methods mapping a uniform grid to a nonuniform one using grading functions

$$\xi(x) = \frac{\int_0^x w \,\mathrm{d}x}{\int_0^1 w \,\mathrm{d}x}. \tag{9}$$

Here w is a *monitor function* (some measure of the error, involving higher derivatives, e.g. $|y^{(q)}|^{(1/p)}$, and approximated by a spline). This means that $\xi(x)$ represents the portion of the error coming from $[0, x]$ and that ξ_x (hence ϕ) is proportional to w. The nonuniform grid is computed by finding points $\{x_j\}$ such that for ϵ_N fixed, the equidistribution criterion

$$\xi(x_{j+1}) - \xi(x_j) \approx \epsilon_N \tag{10}$$

is satisfied. The points are redistributed until the 'drift'

$$\max_j \int_{x_j}^{x_{j+1}} w \,\mathrm{d}x - \epsilon_N \int_0^1 w \,\mathrm{d}x,$$

is smaller than a tolerance. Once the nonuniform grid has been generated, the map $x(\xi)$ is approximated using spline interpolation. This enables a change of the number of grid points through oversampling.

The approach taken by Auzinger et al. [6] is directly related. Its objective is to obtain a grid point allocation such that

$$\xi_x = \frac{w}{\int_0^1 w \, \mathrm{d}x},$$

where the monitor function is $w = (|\psi|/\mathrm{TOL})^{1/p}$ and ψ is an estimate of the principal error function, e.g. $\psi = y^{(p)}$. The quantity ξ_x equals the density function ϕ with unit integral. Once w has been calculated on an actual grid $\{x_k\}$, a piecewise linear function $\xi(x)$ is constructed by integration, so that

$$\xi(x_j) = \int_0^{x_j} \xi_x \, \mathrm{d}x.$$

This corresponds to the grading function (9). The necessary number of grid points to meet the tolerance is then computed, and a new grid $\{\tilde{x}_j\}$ constructed by inverse linear interpolation of $\xi(x)$ in the same manner as in (10).

In [35], Tang and Tang use calculus of variations and seek $\xi(x)$ such that

$$E(\xi) = \frac{1}{2} \int_0^1 \frac{\xi_x^2}{w} \, \mathrm{d}x$$

is minimized, subject to $\xi(0) = 0$ and $\xi(1) = 1$, for a monitor function w. This leads to 'Winslow's variable diffusion method,' $(w^{-1}\xi_x)_x = 0$, or

$$(w x_\xi)_\xi = 0.$$

For mesh redistribution, Tang and Tang use a Gauss–Seidel approach to solve this 'elliptic' equation. As an alternative, they introduce an additional time variable t is and instead solve the 'parabolic' problem

$$x_t = (w x_\xi)_\xi$$

by using the explicit Euler method. In both cases, the final density ϕ is proportional to the monitor function w.

Moving mesh algorithms for hyperbolic problems are also considered by Stockie et al., [34]. They use a variant of the diffusion approach above. The ξ grid remains uniform and constant, but the deformed grid in x is time dependent. The moving mesh is governed by

$$(w\dot{x}_\xi)_\xi = -(w x_\xi)_\xi/\tau,$$

where τ is a 'time constant' controlling the approach to error equidistribution, and where the dot denotes differentiation with respect to time. Because the equation is stiff, the authors propose to solve it by the Crank-Nicolson method; as this method is implicit, the possible monitor functions are restricted. Spatial smoothing is considered, and temporal smoothing is mentioned as a possibility. (In a more general setting such operations would be handled by digital filters both in time and space.) Further work on moving meshes can be found in [9] and [19].

In Section 2 we characterize the optimal grid density function associated with a given monitor function, using a variation principle. Then in Section 3 we introduce a novel grid generation

algorithm. Based on a control theoretic approach, the procedure has been implemented in a test version of a working MATLAB code, `bvpsuite`, everything else equal. The controller has two stages. In the first the density function is constructed on a coarse grid with the objective of making the error equidistributed. In the second stage, using the so obtained density, the number of mesh points necessary to satisfy the tolerance is determined; the final grid has then been constructed. Finally in Section 4 numerical tests demonstrate the advantages of the new control system by comparing its performance to that of the conventional approach. Computational experiments are carried out for a selection of problems and over a wide range of tolerances.

2 Adaptivity as a variational problem

Here we shall be concerned with the problem of finding $\phi(x)$, from which the deformation map can readily be constructed. This approach is essentially the same as that in [6] except in its discrete implementation, where, instead of working with the independent variable x directly, the role of the auxiliary variable ξ is emphasized to facilitate a simple computation of the new grid, including oversampling, in order to determine the locations of grid points for an arbitrary number of interior points $N-1$, possibly determined by a global error criterion.
Assume that some positive monitor function w is given, and consider the problem of finding a positive function $\phi \in C[0,1]$ solving the variational problem

$$\min_{\phi} \int_0^1 \frac{w(y)\,\mathrm{d}x}{\phi^q} \quad \text{subject to} \quad \int_0^1 \phi(x)\,\mathrm{d}x = 1 \tag{11}$$

for some $q \neq -1$, to be specified later. The monitor function w must be independent of ϕ, and the constraint implies that the map $x(\xi)$ satisfies the proper boundary conditions, with the positivity of ϕ guaranteeing that $x(\xi)$ is a bijection.

Theorem 2.1 *Given $q \neq -1$ and a positive monitor function w in the variational problem* (11)*, the optimal grid density function ϕ^* is given by*

$$\phi^*(x) = \frac{w^{1/(q+1)}(y)}{\int_0^1 w^{1/(q+1)}(y)\,\mathrm{d}x}. \tag{12}$$

Proof We introduce a multiplier λ and the Lagrangian

$$L(\phi, \lambda) = \frac{w(y)}{\phi^q} + \lambda\phi.$$

At a stationary point the first variation with respect to the control variable ϕ must vanish. Hence

$$-\frac{qw(y)}{\phi^{q+1}} + \lambda = 0, \tag{13}$$

which implies

$$\phi = \left(\frac{qw(y)}{\lambda}\right)^{1/(q+1)}. \tag{14}$$

The constraint determines the multiplier by integrating (14) over $[0,1]$; eliminating λ we obtain the minimizer (12).

Remarks. The normalization of the minimizer can also be conveniently expressed in terms of Hölder means. For a positive function u the s-Hölder mean is defined by $\mathcal{M}_s(u) = \left(\int_0^1 u^s \,\mathrm{d}x\right)^{1/s}$, where one can take $-\infty \leq s \leq \infty$. Thus (12) is equivalent to

$$\phi^*(x) = \left(\frac{w(y)}{\mathcal{M}_{\frac{1}{q+1}}(w)}\right)^{1/(q+1)} = \frac{w^{1/(q+1)}(y)}{\mathcal{M}_1\left(w^{1/(q+1)}(\cdot)\right)}. \tag{15}$$

The choice of monitor function w, as well as the power q, leads to several important cases.

Case 1: Arc length adaptivity. As mentioned above, a simple adaptive technique is to choose grid point locations so that the solution's arc length on each subinterval is the same. A similar, but weaker, criterion is to require that the grid density ϕ is *covariant* with the solution's arc length. This corresponds to taking

$$w(y) = \sqrt{1 + |y'|^2} \quad \text{and} \quad q > -1.$$

The requirement $q > -1$ comes from the structure of the minimizer (12) and guarantees covariance. The choice $q = 0$ implies proportionality and *equidistribution of arc length between grid points.*

Case 2: Local error control in $L^s[0,1]$. Consider the local error model

$$|r(x)| = \frac{\epsilon_N^{p+1}|\psi(x)|}{\phi(x)^{p+1}}, \tag{16}$$

where p is the order of the method and $s \geq 1$ is arbitrary. Let our objective be to minimize $\|r\|^s_{L^s[0,1]}$; the minimization problem is then

$$\min_{\phi} \int_0^1 |r(x)|^s \,\mathrm{d}x \quad \text{subject to} \quad \int_0^1 \phi(x)\,\mathrm{d}x = 1 \tag{17}$$

and corresponds to taking $w(y) = \epsilon_N^{sp+s}|\psi|^s$ and $q = sp+s$. As the optimal solution is *independent of* ϵ_N, we obtain the grid distribution ϕ^*, which is to be used for any prescribed accuracy requirement TOL in the adaptive method.
The optimal solution (12) is

$$\phi^*(x) = \frac{|\psi(x)|^{s/(sp+s+1)}}{\int_0^1 |\psi(x)|^{s/(sp+s+1)}\mathrm{d}x} = \frac{|\psi(x)|^{s/(sp+s+1)}}{\mathcal{M}_1\left(|\psi(\cdot)|^{s/(sp+s+1)}\right)}, \tag{18}$$

which says that for local error control, the optimal density ϕ^* is covariant with (a fractional power of) the pointwise norm of the principal error function ψ. By letting $s \to \infty$, the optimal 'minimax' grid (corresponding to minimizing $\|r\|_\infty$) results in a pointwise equidistribution of the local error, see [22]. Thus, by combining (16) and (18), we see that for the minimax density ϕ^* it holds that

$$\forall x \quad |r(x)| = \|r\|_\infty = \epsilon_N^{p+1}\mathcal{M}_{\frac{1}{p+1}}(|\psi|) = \mathrm{TOL}, \tag{19}$$

where the last equality indicates how the necessary number of grid points is determined by TOL. Most implementations of adaptive methods seek such an equidistribution of the local error.

In [6] the case $q = 0$ is studied, but as the monitor function is taken to be $w \sim |\psi|^{1/(p+1)}$, one obtains $\phi^* \sim |\psi|^{1/(p+1)}$ directly – this grid is therefore equivalent to the minimax grid, although no optimality criterion is established. The approach achieves local error equidistribution when Gaussian points are used.
Because $|\psi|$ is not directly available it must be calculated from a local error estimate of the form (16), i.e.,

$$|\psi(x)| = \frac{|r(x)|\phi(x)^{p+1}}{\epsilon^{p+1}}. \tag{20}$$

Note that, because of the normalization in (18), it is not necessary to include the constant ϵ. Hence $|\psi|$ can be estimated from the function $|r|\phi^{p+1}$.

Finally, we note that the process of computing ϕ^* is iterative. In practice, one takes $\phi_0(x) \equiv 1$, and computes an error estimate $r_0(x)$ on this equidistant initial grid. Extracting $|\psi(x)|$ from the error estimate (20), $\phi_1(x)$ is then computed by applying (18), to obtain the first nonuniform grid. A few grid refinement iterations may be needed. This process may or may not converge to ϕ^*, depending on how strongly the estimate of $|\psi(x)|$ depends on ϕ. In view of (20), this dependence is weak only if the asymptotic local error model accurately reflects the local error, and a local change of ϕ only has a local effect on the error.

Case 3: Defect control. A variant of local error control is to control the *defect*

$$d(x) = P'(x) - F(x, P(x)),$$

which is assumed to have an asymptotic behavior

$$d(x) \approx c_p \frac{\epsilon^{p+1}\psi(x)}{\phi(x)^{p+1}}.$$

Case 4: Controlling global error on Gaussian points. For Gaussian collocation there is a strong relation between local and global errors. Assuming that we have a global error estimate

$$g(x) \approx C_p \frac{\epsilon^p \Psi(x)}{\phi(x)^p} \tag{21}$$

the control of the global error follows the same lines as above.

3 Grid generation and refinement: the control algorithm

Below we shall describe a modular control algorithm that uses the local error to find ϕ^* (grid generation) and determines the number of steps needed (grid refinement) to solve the boundary value problem to a prescribed accuracy TOL using the minimax grid.
Before the actual algorithm can be described, however, we need to give a detailed description of the construction, purpose and use of the component modules. Some of them are optional or could be employed in several different ways, and the modules are in principle independent. Thus, the final control algorithm offers possibilities for separate modifications of the modules without harming the overall algorithm.

Module 0: Initialization. We start with an equidistant grid X_N, i.e., initially $\phi(x) \equiv 1$. There are $N-1$ internal grid points, and N 'steps' or 'cells,' corresponding to the step sizes

$$\Delta x_{j+1/2} = x_{j+1} - x_j = \frac{\epsilon_N}{\varphi_{j+1/2}}, \quad j = 0, \ldots, N-1, \tag{22}$$

where $\epsilon_N = 1/N$, and x_0 and x_N correspond to the boundary points.
The sequence $\Phi_N = \{\varphi_{j+1/2}\}_{j=0}^{N-1}$ represents a discrete approximation to the continuous function $\phi(x)$. It can be interpreted as approximating N *equidistant samples of* $\phi\big(x(\xi)\big)$ *on the staggered grid* $\Xi_{N+1/2} = \{(j+1/2)\cdot\epsilon_N\}_{j=0}^{N-1}$; hence $\varphi_{j+1/2}$ approximates $\phi\big(x(\xi_{j+1/2})\big)$.
At the initialization step we take, by default, $\varphi_{j+1/2} = 1$ for all j. This initial density is denoted by $\Phi_N^{[0]}$, where the superscript is the mesh generation index. However, one can also initialize the control algorithm with a nonuniform grid, e.g. if a sequence of boundary value problems is to be solved for a set of different parameter values. Then much work might be saved by starting at an optimal grid taken from a previous run.

Module 1: Obtaining an error estimate. Given a density Φ_N and the corresponding grid X_N the two-point boundary value problem is solved on X_N. We assume that the solver also provides an error estimate, in the form of a staggered sequence $R = \{r_{j+1/2}\}_{j=0}^{N-1}$, where each (pointwise) error estimate is directly associated with its own cell $[x_j, x_{j+1}]$ and step size. It is further assumed that some (possibly user-defined) pointwise norm (possibly a mixed absolute–relative norm) $|\cdot|$ of the error estimate satisfies

$$r_{j+1/2} \approx \frac{|\psi_{j+1/2}|\epsilon_N^q}{\varphi_{j+1/2}^q} \tag{23}$$

for some known power q associated with the asymptotic order of the discretization method; $q = p+1$ for local error, and $q = p$ for global error control. Alternatively, if some other criterion is used, such as arclength control, one obtains values of the relevant monitor function on the staggered grid. As this information is going to be used to *recompute* the density Φ_N, which is a scalar sequence, it is necessary to take the norm of the error estimate at this stage.

Module 2: Regularizing the error estimate. Before using the error estimate it may be advantageous to apply a low-pass filter to it in order to remove possible noise. In a continuous setting, a filter is, in principle, a linear transformation

$$\hat{r}(x) := \int_0^1 K(x,y) r(y) \mathrm{d}y. \tag{24}$$

The kernel often has the form $K(x,y) = k(x-y)$, which corresponds to a convolution filter. After discretization the filter is 'digital' and corresponds to a matrix–vector transformation $\hat{R} = \mathcal{F}R$, where $\hat{R}$ is the filtered estimate. The filters can have dense as well as sparse kernels. A sparse, near diagonal, kernel appears to be preferable in our case, as this corresponds to a local filtering. This is usually sufficient in order to remove high frequency noise.
In the $K(x,y) = k(x-y)$ case, $\mathcal{F}$ is a Toeplitz matrix. As we want to suppress noise, and in particular $(-1)^n$ oscillations in space, we may use repeated averaging, e.g.,

$$\begin{pmatrix} \hat{r}_{1/2} \\ \hat{r}_{3/2} \\ \vdots \\ \hat{r}_{N-1/2} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 2 & 2 & \ldots & 0 \\ 1 & 2 & 1 & \vdots \\ \vdots & 1 & 2 & 1 \\ 0 & \ldots & 2 & 2 \end{pmatrix} \begin{pmatrix} r_{1/2} \\ r_{3/2} \\ \vdots \\ r_{N-1/2} \end{pmatrix}, \tag{25}$$

where the two deviant matrix elements $\mathcal{F}_{1,2}$ and $\mathcal{F}_{N,N-1}$ represent a possible choice of boundary corrections which are necessary in order to prevent the error from being reduced, especially if the filter matrix is applied repeatedly.
By repeated filtering a stronger noise suppression can be achieved, while the error estimate becomes successively smoother. This is of importance as the error estimate will be used to construct the next Φ_N, which in turn affects the regularity of the nonuniform grid X_N. A simple approach is to apply the filter repeatedly until the signal to noise ratio (S/N ratio) of the processed error is high enough. The 'noise' is the norm of the removed part, $\|R - \hat{R}\|$, while the 'signal' is $\|\hat{R}\|$. Thus no further filtering is necessary when $\|R - \hat{R}\|/\|\hat{R}\| \ll 1$. A suitable norm for checking the S/N ratio is the RMS norm (discrete L^2 norm).
There are many possible alternatives to the filter suggested above. For instance, the filter could be based on geometric rather than arithmetic averaging. Thus we could use a filter corresponding to the coefficients used above, but in the form

$$\hat{r}_{j+1/2} = \left(r_{j-1/2} \cdot r_{j+1/2}^2 \cdot r_{j+3/2} \right)^{1/4}, \tag{26}$$

where appropriate boundary corrections should again be applied.

Module 3: Updating the discrete density Φ_N (grid generation). Let $\Phi_N^{[k]}$ denote the kth grid density and consider the computation of $\Phi_N^{[k+1]}$, assuming that there is a suitably processed error estimate $\hat{R}^{[k]}$ available. In the updating process the sequences Φ_N and $\hat{R}$ are interpreted as representing the density and error *on the equidistant staggered grid* $\Xi_{N+1/2}$. As these functions are discrete, no further mapping of the sequences is required.
In view of the model (23) and the equidistribution principle, the density Φ should be pointwise proportional to $\hat{R}^{1/q}$. As the simplest possible alternative, we therefore employ the integrating controller

$$\log \Phi_N^{[k+1]} = \log \Phi_N^{[k]} + \frac{k_{\mathrm{I}}}{q} \log \lambda_k \hat{R}^{[k]} \tag{27}$$

$$\sum_{j=0}^{N-1} \frac{\epsilon_N}{\varphi_j^{[k+1]}} = 1, \tag{28}$$

where the logarithm is understood to be applied componentwise to the vectors $\hat{R}^{[k]}$ (the filtered error estimate) and $\Phi_N^{[k]}$. The algebraic constraint implies that each sequence $\Phi_N^{[k]}$ must be renormalized so that the grid X_N exactly matches the interval $[0, 1]$. The scalar parameter λ_k is selected in each update to maintain this normalization.
Because of the equidistribution principle (19) we see that consecutive updates will keep changing $\Phi_N^{[k]}$ until all components of the error estimate $\hat{R}^{[k]}$ are equal. Then $\lambda_k \hat{r}_{j+1/2}^{[k]} \equiv 1$, implying that the term added to $\log \Phi_n^{[k]}$ is zero. Thus the controller will compensate *variation in* $\hat{R}^{[k]}$, but the absolute magnitude of $\hat{R}^{[k]}$ does not matter until we are ready to choose the number of grid points, $\hat{N}$, that is required to meet the tolerance TOL.
The controller is a *deadbeat controller* for the *integral gain* $k_{\mathrm{I}} = 1$, but may also be used as a convolution filter (*exponential forgetting*) by using gains $0 < k_{\mathrm{I}} < 1$. The role of the gain is to adjust to what extent variations in $\hat{R}^{[k]}$ are allowed to affect $\Phi_N^{[k]}$. For $k_{\mathrm{I}} < 1$, less than the full variation of $\hat{R}^{[k]}$ will be used in the update; such a gain may reduce the risk of instability, but also makes it take longer for the controller to converge to the optimal density ϕ^*.

There are many alternative controllers. One may e.g. use a PI (proportional–integral) controller or a digital filter for the update, [32]. Note that the update of the density can be viewed as occurring in 'pseudo time,' which implies that a causal filter is needed in this step, as opposed to the filtering with respect variations (noise) along the x ('space') direction.
Finally, in practice the control algorithm does not require that one solves the difference–algebraic system above; it is sufficient to use componentwise multiplication of $\Phi_N^{[k]}$ and $(\hat{R}^{[k]})^{1/q}$, followed by a renormalization of the product, so that the new discrete density $\Phi_N^{[k]}$ satisfies the constraint. The algorithm is therefore easy to implement.

Module 4: Determination of N (grid refinement). Recalling that the boundary value problem has to be solved once on each grid, and that the optimal ϕ^* is independent of TOL, the cost of finding ϕ^* is reduced by *running the control algorithm with a fixed, low value of N*. The effort to solve the problem to high precision should be deferred until the optimal ϕ^* has been found; hopefully this can be achieved within a few iterations.
Let $N-1$ be the number of interior points used during the determination of ϕ^*, and let $\hat{N}-1$ be the number of interior points needed to solve the problem to the requested accuracy. Further, let $\|\hat{R}_N^{[k]}\|_\infty$ be the error observed when $N-1$ interior points were used. By (23),

$$\left(\frac{\|\hat{R}_N^{[k]}\|_\infty}{\mathrm{TOL}}\right)^{1/q} = \frac{\hat{N}}{N}, \tag{29}$$

from which $\hat{N}$ is directly obtained. However, as we are also going to change the density, we need to modify this formula. For example, if the error was computed for the initial, uniform grid, then (29) will be the number of steps needed to solve the problem to accuracy TOL on that uniform grid and not on the next, nonuniform, grid.
To compensate for the new density, we apply a weighted error norm. Thus we compute the weighted error vector,

$$E_N^{[k]} := \hat{R}_N^{[k]} \left(\frac{\Phi_N^{[k]}}{\Phi_N^{[k+1]}}\right)^q, \tag{30}$$

again using componentwise vector operations. The weighted error replaces $\hat{R}_N^{[k]}$ in (29). As a safety measure one would also have to put appropriate upper and lower bounds on $\hat{N}$, implying that $\hat{N}$ is computed according to the formula

$$\hat{N} := \min\left(N_{\max},\, \max\left(N_{\min},\, N\cdot\left(\frac{\|E_N^{[k]}\|_\infty}{\mathrm{TOL}}\right)^{1/q}\right)\right). \tag{31}$$

Note that the error criterion used to determine $\hat{N}$ need not be the same as criterion used for finding ϕ^*. Thus, when determining ϕ^* it is crucial to use a monitor function that only depends locally on ϕ^*, such as arclength control or local error control. To determine $\hat{N}$, however, we could use a global error model if desired, provided that a global error estimate is available. This approach is utilized in the new code `bvpsuite.nga` that we present in Section 4. Whether such combinations of different criteria for ϕ^* and $\hat{N}$ are advantageous depends on what the objectives for using the adaptive algorithm are.

Module 5: Oversampling the density Φ_N. Recalling that the final density Φ_N approximates the continuous density ϕ^* on the staggered grid $\Xi_{N+1/2}$, we need to *oversample* the sequence

Φ_N to $\hat{N}$ points on a new staggered grid $\Xi_{\hat{N}+1/2}$. This is done using spline interpolation; the operation is inexpensive as the cost is only $\mathrm{O}(N) + \mathrm{O}(\hat{N})$.
Let $S_N(\xi)$ be a cubic spline that interpolates $\Phi_N = \{\varphi_{j+1/2}\}_{j=0}^{N-1}$ on the the equidistant staggered grid $\Xi_{N+1/2} = \{(j+1/2)\cdot\epsilon_N\}_{j=0}^{N-1}$. We then generate a new, refined equidistant staggered grid with $\hat{N}$ interior points on $[0,1]$ according to

$$\Xi_{\hat{N}+1/2} = \{\hat{\xi}_{j+1/2}\}_{j=0}^{\hat{N}-1}, \tag{32}$$

with $\hat{\xi}_{j+1/2} = (j+1/2)\cdot\epsilon_{\hat{N}} = (j+1/2)/\hat{N}$. Evaluating $S_N(\xi)$ on $\Xi_{\hat{N}+1/2}$ we obtain the provisional, high-resolution density

$$\tilde{\Phi}_{\hat{N}} = \{S_N(\hat{\xi}_{j+1/2})\}_{j=0}^{\hat{N}-1}. \tag{33}$$

This is provisional for several reasons. First, the interpolating oversampling will typically not preserve the normalization condition (8). Second, interpolation errors could cause some density values to be negative. Third, even a too small density value is unacceptable, as it would lead to a very large step size somewhere in the grid. Therefore the oversampling must be equipped with several safety measures.
The provisional density is processed in the following way. The values generated in (33) are mapped via a *limiter* to avoid negative and excessively small values. First all negative values are replaced by zeros by applying the map $\tilde{\varphi}_{j+1/2} \mapsto \max(0, \tilde{\varphi}_{j+1/2})$. Further, it is reasonable to impose a maximum on the step size; the mesh width should not exceed (say) one tenth of the solution interval. This can e.g. be accomplished by the map

$$\tilde{\varphi}_{j+1/2} \mapsto \tilde{\varphi}_{j+1/2} + \frac{\epsilon_{\hat{N}}}{\tilde{\varphi}_{j+1/2} + L/10}, \tag{34}$$

where $[0, L]$ is the solution interval (in our case $L = 1$). Note that if the density is very high locally, then the density is unaffected there. However, if the density is low, the limiter makes sure that no step size exceeds one tenth of the interval.
Once the limiter has been applied, one could opt for applying a digital filter such as those mentioned in Module 2 above. Whether or not this is applied, a renormalization is still necessary, implying that the last step of processing the oversampled density is to apply (8) in order to obtain the final high-resolution density $\Phi_{\hat{N}}$.

Module 6: Construction of the nonuniform grid. The previous step generated the mesh widths. Using the notation $\Phi_{\hat{N}} = \{\hat{\varphi}_{j+1/2}\}_0^{\hat{N}-1}$, the new grid points $X_{\hat{N}} = \{\hat{x}_j\}_{j=0}^{\hat{N}}$ are generated by summation of $\hat{x}_{j+1} - \hat{x}_j = \epsilon_{\hat{N}}/\hat{\varphi}_{j+1/2}$, i.e., through partial sums

$$\hat{x}_j = \sum_{k=0}^{j-1} \frac{\epsilon_{\hat{N}}}{\hat{\varphi}_{k+1/2}}\,; \qquad j = 1, \ldots, \hat{N}-1,$$

noting that $x_0 = 0$ and $x_{\hat{N}} = 1$.

The control algorithm. Using the modules above, the control algorithm takes the following form.

1. Choose a suitable number of points M for the control grid. This remains unchanged until the final density has been computed. Construct an initial (usually uniform) density $\Phi_M^{[0]}$ and initialize the grid generation number to $k = 0$. A nonuniform initial grid can sometimes be preferable, e.g. if one solves a sequence of parameterized problems, where previous runs provide the density $\Phi_M^{[0]}$. (Module 0)

2. Given the density $\Phi_M^{[k]}$, generate the computational grid $X_M^{[k]}$. (Module 6)

3. Solve the boundary value problem on $X_M^{[k]}$ and compute an error estimate $R_M^{[k]}$. Compute the pointwise norm of the error estimate. Make sure that all error components are nonzero. (Module 1)

4. (Optional) Regularize the error estimate by applying a suitable spatial digital filter to compute $\hat{R}_M^{[k]}$. (Module 2)

5. If $k = 0$, compute $\hat{N}_0$ from (29) to find the number of steps that would have been required to solve the boundary value problem using the initial (uniform) density $\Phi_M^{[0]}$.

6. Update the density by pointwise multiplication

$$\Phi_M^{[k+1]} \leftarrow \Phi_M^{[k]} .* \, (\hat{R}_M^{[k]})^{\frac{1}{q}} \tag{35}$$

   followed by a renormalization of $\Phi_M^{[k+1]}$. (Module 3)

7. Compute weighted error $E_M^{[k]}$ and predict the number of points $\hat{N}_{k+1}$ required to solve the problem to accuracy TOL using the density $\Phi_M^{[k+1]}$ according to (31). (Module 4)

8. Repeat from Step 2 unless $\hat{N}_{k+1} > (1 - \vartheta) \cdot \hat{N}_k$. This termination criterion implies that it is not considered worthwhile to solve the problem once more on the control grid and recompute $\Phi_M^{[k+1]}$ if this would lead to removing less than a prescribed fraction ϑ of the grid points in the final grid. The fraction could be selected as, say, $\vartheta = 0.05$ or $\vartheta = 0.1$, depending on whether the accuracy requirement is strict or loose.

9. When the final density profile $\Phi_M^{[k+1]}$ has been found, oversample the density to obtain the high-resolution density $\Phi_{\hat{N}_{k+1}}^{[k+1]}$, taking care to apply a limiter such as (34) to the oversampled vector. As an option, the new density could be further processed by filtering. Finally apply the renormalization (8). (Modules 5 and 2)

10. Construct the corresponding high-resolution grid $X_{\hat{N}_{k+1}}$ and solve the boundary value problem on that grid. (Module 6)

11. Use the error estimate to verify that the desired accuracy has been achieved. Evaluate the efficiency of the adapted grid by computing the ratio $\hat{N}_0/\hat{N}_{k+1}$. As the boundary value solver typically has an $\mathrm{O}(\hat{N}_{k+1})$ complexity, the computed ratio represents an estimate of how much more efficient an adaptive method is compared to a nonadaptive one.

4 Computational experiments

We implemented the new mesh control algorithm within our collocation code `bvpsuite`, which is a new version of the general purpose MATLAB code `sbvp`, cf. [5, 4, 23]. Both programs have already been successfully applied to a variety of regular and singular problems exhibiting singularities of the first and second kind, see for example [11, 12, 13, 24, 28]. The performance of the routine `sbvp` has previously been compared to that of MATLAB code `bvp4c` and the FORTRAN code `colnew`, see [7].

The code `bvpsuite` is designed to solve systems of differential equations of arbitrary order. For simplicity of notation we formulate below a problem whose order varies between four and zero, which means that algebraic constraints that do not involve derivatives are also admitted. Moreover, the problem can be given in a fully implicit form,

$$F(x, y^{(4)}(x), y^{(3)}(x), y''(x), y'(x), y(x)) = 0, \quad x \in [0,1], \tag{36a}$$

$$b(y^{(3)}(0), y''(0), y'(0), y(0), y^{(3)}(1), y''(1), y'(1), y(1)) = 0. \tag{36b}$$

The numerical approximation defined by collocation is computed as follows. On a mesh

$$X_N = \{x_j\}_{j=0}^{N}, \quad 0 = x_0 < x_1 \cdots < x_N = 1$$

we approximate the analytical solution by a collocation polynomial,

$$P_{X_N}(x) := P_j(x), \quad x \in [x_j, x_{j+1}], \quad j = 0, \ldots, N-1,$$

where we require $P_{X_N} \in C^{r-1}[0,1]$ for a differential equation of order r. Here, the functions P_j are polynomials of maximal degree $m - 1 + r$ which satisfy (36a) at the *collocation points*

$$\{t_{j,l} = x_j + \rho_l(x_{j+1} - x_j),\ j = 0, \ldots, N-1;\ l = 1, \ldots, m\}, \quad 0 < \rho_1 < \cdots < \rho_m < 1,$$

and the associated boundary conditions (36b).

We use a classical error estimate based on mesh halving. In this approach, we compute the collocation solution P_{X_N} on a mesh X_N. We then choose a second mesh X_{2N} where in every interval $[x_j, x_{j+1}]$ of X_N we insert two subintervals of equal length. On this new mesh, we compute the numerical solution based on the same collocation scheme to obtain the collocating function $P_{X_{2N}}(x)$. Using these two quantities, we define

$$g(x) := \frac{2^m}{1 - 2^m}(P_{X_{2N}}(x) - P_{X_N}(x)) \tag{37}$$

as an error estimate for the approximation $P_{X_N}(x)$. Expressing the global error of the collocation solution, $\delta(x) := P_{X_N}(x) - y(x)$, in terms of the principal error function $e(x)$, we obtain

$$\delta(x) = e(x)|x_{j+1} - x_j|^m + \mathrm{O}(|x_{j+1} - x_j|^{m+1}), \quad x \in [x_j, x_{j+1}], \tag{38}$$

where $e(x)$ is independent of X_N. Then the error of the error estimate shows the following asymptotic behavior, $g(x) - \delta(x) = \mathrm{O}(|x_{j+1} - x_j|^{m+1})$, and the error estimate is asymptotically correct.

Numerical experiments have been carried out for five test problems, cf. Appendix 5. We have chosen to use mainly singularly perturbed boundary value problems to be able to increase the

difficulty of the problem arbitrarily by the adjustment of a single parameter. One of the models is a singular boundary value problem with a singularity of the first kind. We implemented the new grid adaptation algorithm and the standard strategy within `bvpsuite` and refer to the two variants as `bvpsuite.nga` and `bvpsuite.sga`, respectively. In all other respects the two codes are identical. We stress that our present aim is not to compare `bvpsuite.nga` with other existing codes, but to show its advantages compared to standard adaptation techniques. We will return to comparisons with other codes in near future.

4.1 Description of `bvpsuite.nga` adaptivity

In order to illustrate how `bvpsuite.nga` behaves in the solution process we apply it to the test problem T5, see (44) in the appendix. We give a comprehensive record of this test, in order to demonstrate how to interpret subsequent comparisons. Here we use collocation at four Gaussian points and set the absolute tolerance requirement to $\text{TOL} = 10^{-8}$.
In this first demonstration, we run `bvpsuite.nga` without restrictions on the number of generated grids. First, `bvpsuite.nga` iteratively computes the density on a coarse mesh with $M = 50$ points. Then, for each density function the number of mesh points necessary to solve the problem to the desired accuracy is predicted. This process is continued until the density function stabilizes in the sense that the number of mesh points does not change. Theoretically, the final density function is associated with a mesh containing the minimal number of points necessary to satisfy the tolerance requirement. Next, we let the program solve the problem on this final mesh to see if the tolerance requirement has indeed been satisfied. Figure 1 shows the quick convergence of the mesh density function.

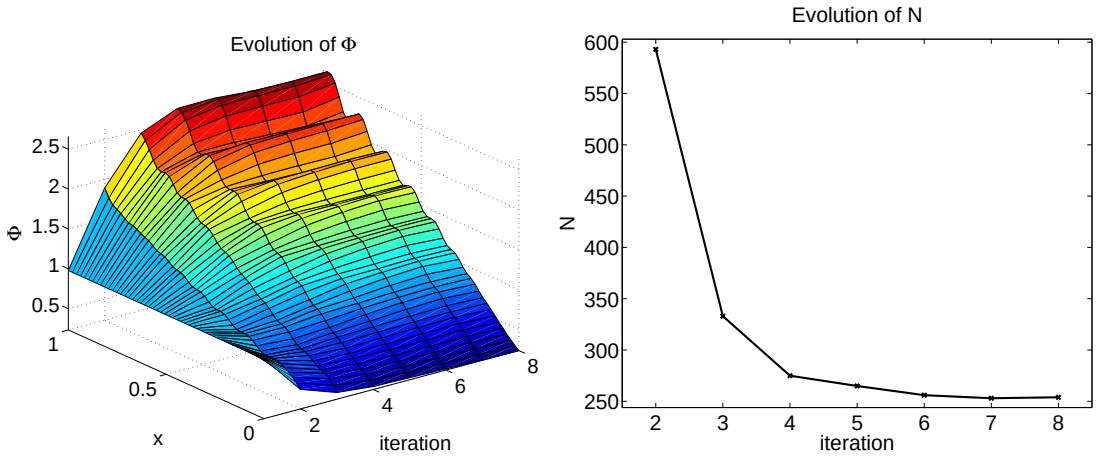

Figure 1: Problem T5 (44). Evolution of the density function Φ (left) and estimated number of mesh points N (right) needed to satisfy the tolerance requirement. Note that the density function stabilizes after a few steps. Solving the problem on a control grid with $M = 50$ points is inexpensive, implying very short run-times.

We see that by changing the density function, it is possible to significantly decrease the number of points in the final mesh.

A working code would not keep refining the density once it is stabilizing. As a termination criterion for `bvpsuite.nga` we have chosen to stop the iteration if the next step would save less than 10 % of the mesh points when compared to the previous step. Above, we see that after step four, `bvpsuite.nga` suggests to use $N = 275$ and in the fifth step $N = 265$. The gain is only 4% and the procedure stops. The final numerical solution is calculated using the density $\Phi^{[5]}$ and $N = 265$. Naturally, a quickly converging procedure for the adaptation Φ is vital for this approach.

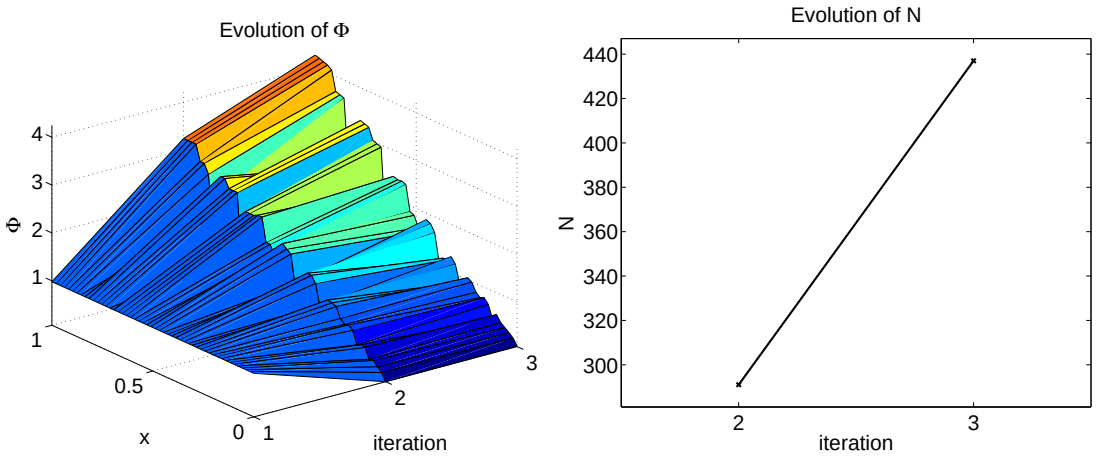

Figure 2: Problem T5 (44). Evolution of the density function Φ (left) and estimated number of mesh points N (right) needed to satisfy the tolerance requirement. In `bvpsuite.sga` the number of points is adjusted in every step. Although the number of iterations is smaller than for `bvpsuite.nga`, total computational cost is higher as the density is far from optimal, forcing the number of grid points to increase.

In contrast to `bvpsuite.nga`, the `bvpsuite.sga` program adjusts both grid density and the number of mesh points simultaneously. Consequently, one has to face the risk of solving the problem using a density that is not yet optimal, thus resulting in using too many points. This is seen in Figure 2, where in the third iteration $N = 437$ points are required to satisfy the tolerance. Thus `bvpsuite.sga` solved the problem three times, on grids with $N = 50$, $N = 291$, and $N = 437$ points, while `bvpsuite.nga` solved the problem four times on the initial control grid with $M = 50$ points, and once on the final grid with $N = 265$ points.
The difference in the number of points necessary to satisfy the tolerance on the final grid is explained by the quality of the density function associated with the final mesh, cf. Figure 3.

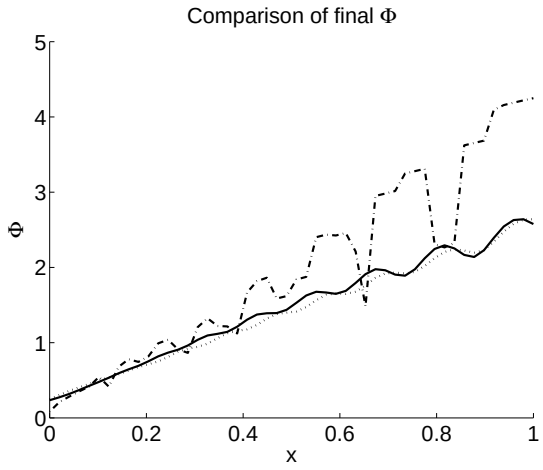

Figure 3: Problem T5 (44). Final density functions Φ for `bvpsuite.nga` run without restrictions on the number of generated grids (solid line), `bvpsuite.nga` with proper stopping criterion (dotted line), and `bvpsuite.sga` (dash-dotted line).

In order to generate the mesh, the residual $r(x) := d(x)$ is used as monitor function. The values of $r(x)$ are available from the substitution of the collocation solution $P_{X_N}(x)$ into the system of ODEs (36a). We first compute

$$\bar{R}(x_{j+1/2}) = \int_{x_j}^{x_{j+1}} r(x)\,\mathrm{d}x \approx \frac{r(x_j) + r(x_{j+1})}{2}(x_{j+1} - x_j),$$

for $j = 0, \ldots, N-1$. As a monitor function we use the pointwise Euclidean norm $R(x) := |\bar{R}(x)|_2$. Figure 4 shows the evolution of $R(x)$.

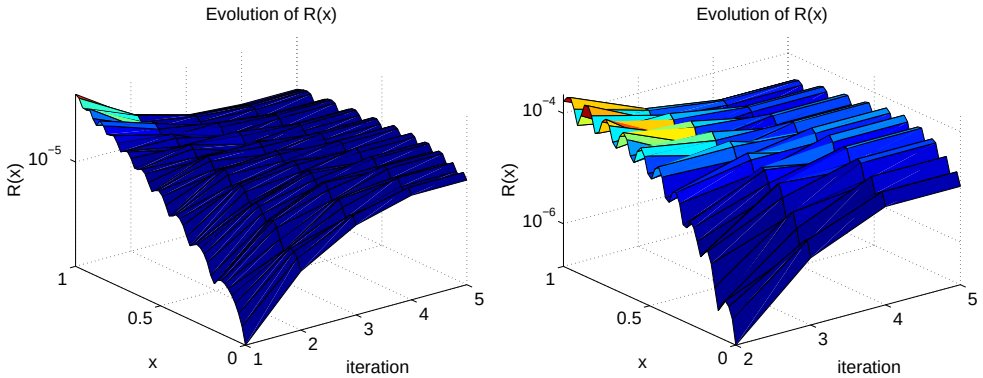

Figure 4: Problem T5 (44). Plot of $R(x)$ for all five iterations (left), and for iteration 2 through 5 (right). Along with the changing grid density, $R(x)$ eventually becomes equidistributed.

The number of grid points is determined by requiring that the absolute global error satisfies the tolerance. Let X_{coll} denote a grid consisting of the mesh points x_j from the coarser grid X_N and

its collocation points $t_{j,l}$. Then we compute $G_{X_{coll}} := \max_x |g(x)|_\infty$ for $x \in X_{coll}$, see (30). The number of points for the next iteration is predicted according to

$$\hat{N}_k = M \left(\frac{G_{X_{coll}}}{\text{TOL}} \right)^{1/(m+1)} . \tag{39}$$

Figure 5 shows the evolution of the global error estimate $|g(x)|_\infty$.

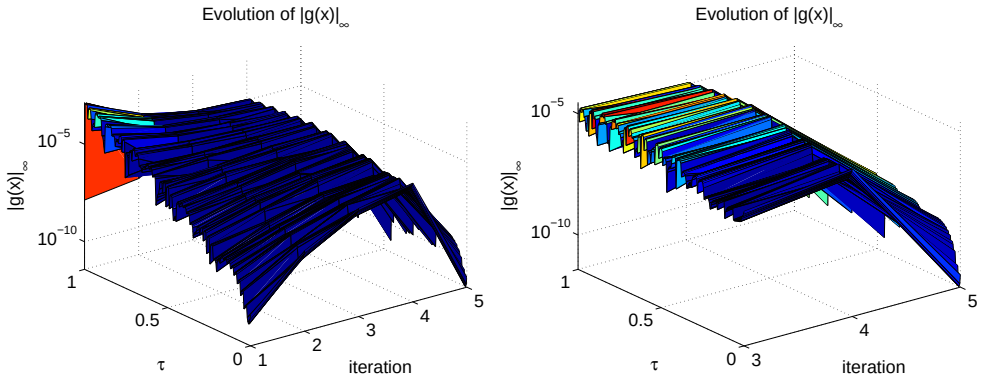

Figure 5: Problem T5 (44). Global error $|g(x)|_\infty$ for all five iteration (left), and for iteration steps 3 through 5 (right). When, in iteration 5, the final grid is employed instead of the 50-point control grid, the error drops to the tolerance level.

4.2 Comparison of the code variants `bvpsuite.nga` and `bvpsuite.sga`

The same Gaussian collocation is used both in `bvpsuite.nga` and `bvpsuite.sga`. In the latter, the global error estimate is used both for grid generation and refinement, while `bvpsuite.nga` uses the residual to update the mesh density and the global error to refine the mesh. In both codes the tolerance controls the absolute global error.

4.2.1 Accuracy versus tolerance and convergence orders

In order to show how numerical accuracy is related to the tolerance, we solve all test problems for 21 logarithmically distributed values of TOL. We also record the final number of mesh points, N, necessary to satisfy TOL.

The accuracy versus tolerance graphs in Figures 6 to 9 show the reference line where the error equals the tolerance. We see that `bvpsuite.sga` usually computes solutions that are far more accurate than requested. In the case of `bvpsuite.nga` there is a much better agreement between the error and the tolerance, although a discrepancy remains, due to a safety factor.

In the graphs of the number of grid points versus tolerance in Figures 6 to 9, we also indicate the convergence order, represented by the slope of a triangle. Here it is worthwhile to note that in all four cases we observe a faster convergence than the stage order m, viz. the uniform superconvergence order $m + 1$, due to the use of Gaussian points in the collocation. In spite of the grids being nonuniform, this superconvergence is observed over a wide range of tolerances, showing that the adaptivity and implementation reliably produces results in full agreement with the well-known theoretical convergence order on uniform grids.

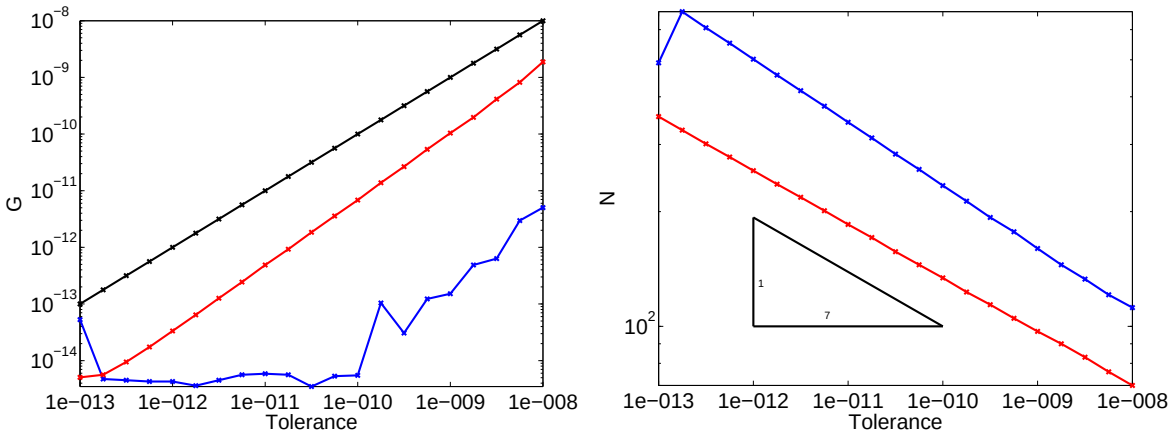

Figure 6: Problem T1 with collocation degree 6. Maximal error on X_{coll} vs. TOL (left), and mesh points N vs. TOL (right). For `bvpsuite.nga` accuracy is more closely related to TOL than for `bvpsuite.sga`. The number of mesh points required for `bvpsuite.nga` is significantly smaller, especially for strict tolerances. Further, for `bvpsuite.nga` the Gaussian superconvergence order of $m + 1 = 7$ is clearly observed throughout the entire range of tolerances, in spite of the nonuniform grids.

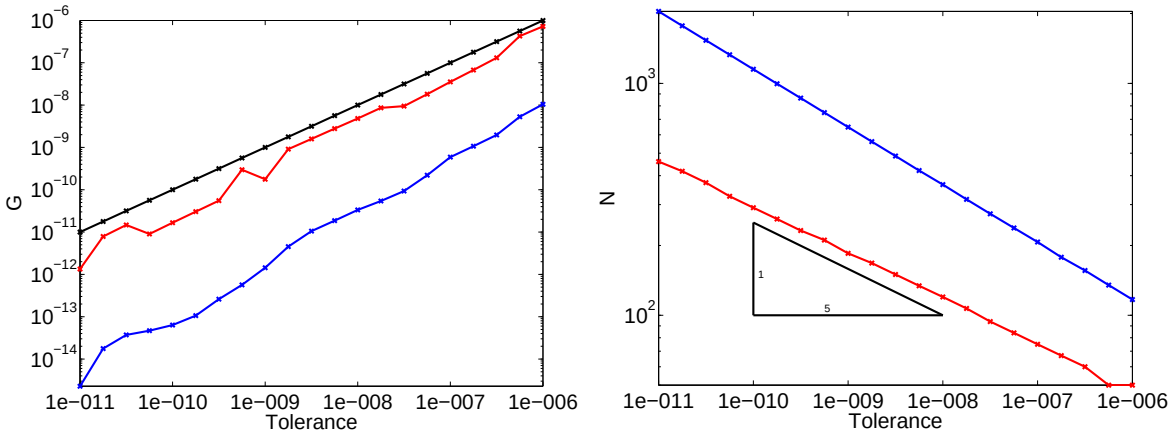

Figure 7: Problem T2 with collocation degree 4. Maximal error on X_{coll} vs. TOL (left), and mesh points N vs. TOL (right). Again `bvpsuite.nga` shows better results. The effort of computing the proper density results in a comparably small number of mesh points in the final grid. Again superconvergence is observed.

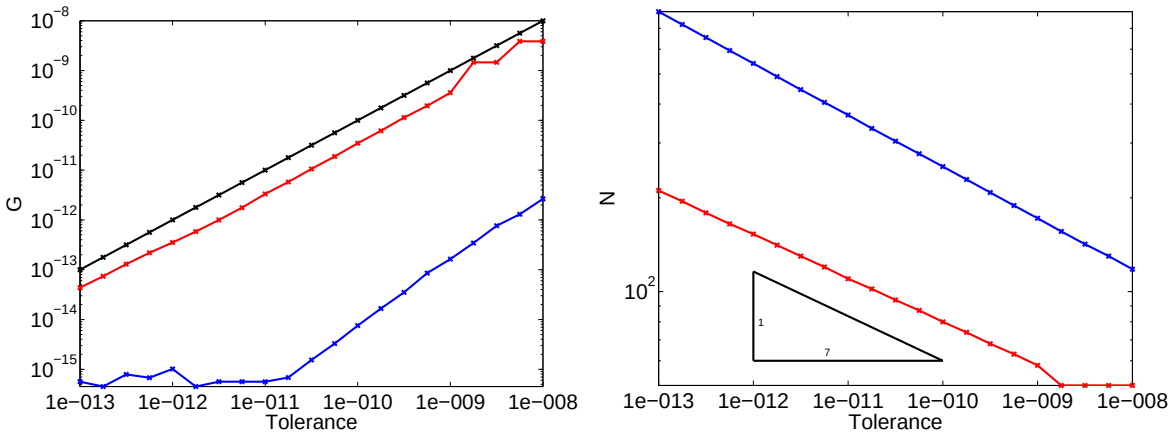

Figure 8: Problem T2 with collocation degree 6. Maximal error on X_{coll} vs. TOL (left), and mesh points N vs. TOL (right). There is a close agreement between error and TOL for `bvpsuite.nga`. For `bvpsuite.sga` the accuracy reaches roundoff level at TOL $\leq 10^{-11}$; the code overachieves by using a too fine mesh for the task.

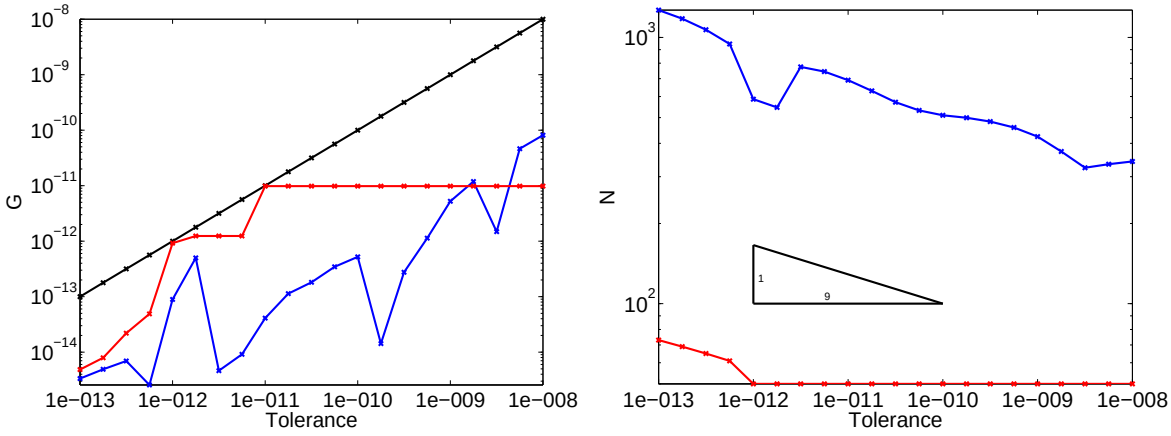

Figure 9: Problem T4 with collocation degree 8. Maximal error on X_{coll} vs. TOL (left), and mesh points N vs. TOL (right). Down to TOL $= 10^{-11}$ `bvpsuite.nga` satisfies the tolerance already on the 50-point control grid.

4.2.2 Comparing the performance of the two codes

We outline the basic steps, safety measures and termination criteria of `bvpsuite.nga`.

1. Grid generation is separated from grid refinement. We adjust Φ on a coarse mesh with a fixed number of points $M = 50$, in order to equidistribute the residual.

2. For each density profile in the above iteration, we estimate the number of mesh points necessary to reach the tolerance, according to (39).

3. The calculation of the density function is terminated when $\hat{N}_{k+1} > 0.9\hat{N}_k$. Clearly, it can be expected that in course of the optimization of the density function the number of associated mesh points will decrease monotonically. This process is stopped when the next density profile $\Phi^{[k+1]}$ would result in saving less than 10% of the mesh points compared to the current density profile $\Phi^{[k]}$.

4. Since the calculation of a residual is not too expensive we nevertheless update the density profile to make use of the most recent available information on the numerical solution associated with $\Phi^{[k]}$.

5. We finally solve the problem on the mesh based on $\Phi^{[k+1]}$ and $\hat{N}_{k+1}$ points, and estimate the global error of the approximation. If the tolerance requirement is satisfied, we stop, otherwise we refine again.

Tables 1 to 4 collect statistics on the number of points in the final grids and the run times for different values of TOL and orders m of the collocation method. Problems T1 and T2 are linear and singularly perturbed, whereas Problems T3 and T4 are nonlinear and singularly perturbed. We have chosen this class because it offers problems of varying difficulty, which can easily be controlled via a single parameter.

m	TOL	ε	$\hat{N}$ `bvpsuite.sga`	time `bvpsuite.sga`	$\hat{N}$ `bvpsuite.nga`	time `bvpsuite.nga`
4	10^{-8}	10^{-3}	518	4.59	181	2.69
4	10^{-9}	10^{-3}	921	8.11	386	3.41
4	10^{-10}	10^{-3}	1636	19.48	452	4.66
4	10^{-11}	10^{-3}	2908	52.31	715	7.16
6	10^{-10}	10^{-3}	234	4.88	134	3.72
6	10^{-11}	10^{-3}	343	8.16	185	4.25
6	10^{-12}	10^{-3}	502	14.56	256	4.98
6	10^{-13}	10^{-3}	491	5.98	355	6.20
8	10^{-10}	$3 \cdot 10^{-4}$	241	7.20	102	5.70
8	10^{-11}	$3 \cdot 10^{-4}$	322	10.56	131	6.13
8	10^{-12}	$3 \cdot 10^{-4}$	429	16.52	168	6.66
8	10^{-13}	$3 \cdot 10^{-4}$	571	26.06	216	7.39

Table 1: Problem T1. Performance data for different orders m, tolerances and values of ε. One can see that `bvpsuite.nga` is superior to `bvpsuite.sga`. Run times are short and in most cases `bvpsuite.sga` is slower than `bvpsuite.nga`.

The code `bvpsuite.nga` solves on courser meshes than `bvpsuite.sga`. Also the run times of `bvpsuite.nga` are more favorable. Only for one test, collocation order 6 and TOL $= 10^{-13}$, is `bvpsuite.sga` faster. This is due to the fact that the number of points was predicted very precisely and no further mesh refinement was necessary. Concerning the final number of mesh points `bvpsuite.nga` is very competitive due to a better density function.

m	TOL	ε	N `bvpsuite.sga`	time `bvpsuite.sga`	N `bvpsuite.nga`	time `bvpsuite.nga`
4	10^{-10}	10^{-4}	359	2.67	92	2.53
4	10^{-11}	10^{-4}	637	5.02	144	2.86
4	10^{-12}	10^{-4}	1131	10.88	227	3.36
4	10^{-13}	10^{-4}	2010	26.78	358	4.28
6	10^{-10}	10^{-5}	591	7.42	50	2.48
6	10^{-11}	10^{-5}	866	12.33	60	4.23
6	10^{-12}	10^{-5}	1271	22.25	83	4.42
6	10^{-13}	10^{-5}	1865	40.14	114	4.64
8	10^{-10}	10^{-6}	969	21.81	50	4.33
8	10^{-11}	10^{-6}	1292	33.78	50	4.34
8	10^{-12}	10^{-6}	1722	55.16	50	5.19
8	10^{-13}	10^{-6}	2296	90.27	73	6.17

Table 2: Problem T2. Performance data for different orders m, tolerances and values of ε. In every test setting `bvpsuite.nga` requires fewer mesh points to find the numerical solution. Its run times are considerably shorter for stricter tolerances.

The second linear example shows the robustness of the approach implemented in `bvpsuite.nga`. For coarse meshes `bvpsuite.sga` is performing well. However, in more difficult settings, the better density function of `bvpsuite.nga` becomes a great advantage. For example, for the method of order 8, `bvpsuite.nga` satisfies the tolerance on the starting mesh, while `bvpsuite.sga` requires around 30 times more points, cf. TOL $= 10^{-12}$. This is due to an inaccurate prediction of the necessary number of points, causing the code to overachieve. The run time is larger by a factor of 11. For TOL $= 10^{-13}$ this factor becomes 15. This test shows how important it is to obtain a good mesh density function.

m	TOL	ε	N bvpsuite.sga	time bvpsuite.sga	N bvpsuite.nga	time bvpsuite.nga
4	10^{-8}	10^{-3}	366	12.67	120	7.69
4	10^{-9}	10^{-3}	648	24.72	185	9.31
4	10^{-10}	10^{-3}	1150	67.89	291	10.81
4	10^{-11}	10^{-3}	2044	258.84	460	13.39
6	10^{-10}	10^{-3}	252	12.09	80	10.17
6	10^{-11}	10^{-3}	369	17.68	110	10.67
6	10^{-12}	10^{-3}	540	27.66	153	11.44
6	10^{-13}	10^{-3}	792	52.34	211	12.45
8	10^{-10}	$5 \cdot 10^{-4}$	157	11.94	50	12.47
8	10^{-11}	$5 \cdot 10^{-4}$	208	14.70	105	16.81
8	10^{-12}	$5 \cdot 10^{-4}$	277	18.56	135	18.14
8	10^{-13}	$5 \cdot 10^{-4}$	370	25.52	180	20.05

Table 3: Problem T3. Performance data for different orders m, tolerances and values of ε. The new code is more efficient, as run times can sometimes be substantially decreased.

m	TOL	ε	N bvpsuite.sga	time bvpsuite.sga	N bvpsuite.nga	time bvpsuite.nga
4	10^{-10}	100	524	16.14	77	8.84
4	10^{-11}	100	931	27.17	120	9.64
4	10^{-12}	100	1655	51.95	189	10.98
4	10^{-13}	100	2941	112.45	298	13.23
6	10^{-10}	500	502	27.98	50	12.84
6	10^{-11}	500	736	39.55	50	14.86
6	10^{-12}	500	1080	51.17	75	19.25
6	10^{-13}	500	1584	82.66	104	20.05
8	10^{-10}	1300	510	106.34	50	15.75
8	10^{-11}	1300	691	140.63	50	15.69
8	10^{-12}	1300	586	97.55	50	22.22
8	10^{-13}	1300	1267	329.80	73	26.14

Table 4: Problem T4. Performance data for different orders m, tolerances and values of ε. For this test problem, the advantage of `bvpsuite.nga` is even more apparent. Both run time and points are occasionally reduced by up to one order of magnitude.

The two nonlinear examples support the previous observations. In the final example for high convergence order 8 and the tolerances TOL $= 10^{-10}$, TOL $= 10^{-11}$ and TOL $= 10^{-12}$, the grid with 50 points is fine enough for `bvpsuite.nga` to successfully solve the problem. In comparison, the meshes provided by `bvpsuite.sga` contained 510, 691 and 586 points, respectively. The tests show that the new algorithm is more efficient than the one implemented in `bvpsuite.sga`.
Finally in Table 5 we show the benefits of the new control algorithms by comparing the two codes to using a uniform mesh. But there are also pitfalls. While it is trivial that more accuracy

can be delivered at a higher cost, the real challenge is to obtain high accuracy at low cost. Thus the main objective of adaptivity is to save work. As the tests demonstrate, a control algorithm cannot afford being overly conservative or ambitious as the price for overachieving (whether by mistake or by design) can entirely eliminate the benefits of adaptivity. The cost of adaptation must also be accommodated; this is the main reason for constructing the `bvpsuite.nga` control algorithms to find the grid density only on a control grid, before the final number of grid points can be calculated and employed.

BVP	m	TOL	ε	$\hat{N}$ `bvpsuite.sga`	$\hat{N}$ `bvpsuite.nga`	$\hat{N}$ equidistant
T1	6	10^{-10}	10^{-3}	234	134	2383
T2	6	10^{-10}	10^{-5}	591	50	242
T3	4	10^{-8}	10^{-3}	366	120	2927
T4	8	10^{-10}	1300	510	50	579
T5	4	10^{-8}	-	439	265	526

Table 5: Number of points for both mesh adaptation strategies of `bvpsuite`. The importance of a good grid distribution is revealed by comparing with the number of points required had the mesh been uniform. In T2, `bvpsuite.sga` inserts far too many points too early in the solution process. As a consequence, it overachieves and produces a solution of much higher accuracy than requested; the tolerance requirement could have been met on a uniform grid with fewer points. The grid control of `bvpsuite.nga` overcomes such pitfalls.

4.3 Conclusions

Results shown in Tables 1 to 4 suggest that the procedure implemented in `bvpsuite.nga` generates excellent density profiles. Moreover, it turns out that the final, almost optimal Φ is obtained in a relatively small number of iterations. The grid adaptation strategy proposed in this paper is especially favorable for strict tolerance requirements. The performance of the two programs is comparable for less strict tolerances. The new control strategies will be available in the next version of `bvpsuite`, following further devolvement and verification.

5 Appendix: Model problems

Here, we present the test problems used for the numerical tests. We have chosen Problems T1 to T4 from the test set which can be found on `http://www.ma.ic.ac.uk/~jcash/`.

Problem T1 is a linear singularly perturbed boundary value problem,

$$\varepsilon y''(x) + y'(x) - (1+\varepsilon)y(x) = 0, \tag{40a}$$

$$y(-1) = 1 + e^{-2}, \;\; y(1) = 1 + e^{\frac{-2(1+\varepsilon)}{\varepsilon}}, \tag{40b}$$

whose solutions for different values of ε are depicted in Figure 10.

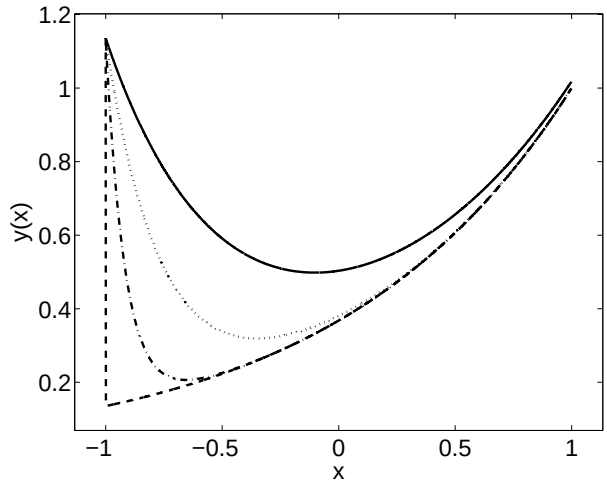

Figure 10: Solutions of Problem T1 for $\varepsilon = 1$ (solid line), $\varepsilon = 0.3$ (dotted line), $\varepsilon = 0.09$ (dash-dotted line) and $\varepsilon = 10^{-4}$ (dashed line).

For the numerical experiments we used $\varepsilon = 10^{-3}$ and $\varepsilon = 3 \cdot 10^{-4}$.

Problem T2 reads

$$y''(x) = \frac{-3\varepsilon y(x)}{(\varepsilon + x^2)^2}, \tag{41a}$$

$$y(-0.1) = -\frac{0.1}{\sqrt{\varepsilon + 0.01}}, \;\; y(0.1) = \frac{0.1}{\sqrt{\varepsilon + 0.01}}. \tag{41b}$$

Its solutions can be found in Figure 11. For this problem, we used $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-5}$ and $\varepsilon = 10^{-6}$.

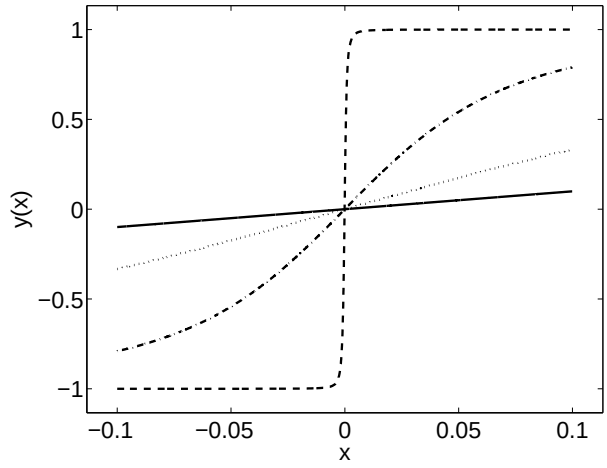

Figure 11: Solutions of Problem T2 for $\varepsilon = 1$ (solid line), $\varepsilon = 0.08$ (dotted line), $\varepsilon = 0.006$ (dash-dotted line) and $\varepsilon = 10^{-6}$ (dashed line).

Problem T3 is nonlinear and given by

$$\varepsilon y''(x) + y(x)y'(x) - y(x) = 0, \tag{42a}$$

$$y(0) = -\frac{1}{3}, \;\; y(1) = \frac{1}{3}. \tag{42b}$$

For the solutions of T3 see Figure 12. The problem was solved for $\varepsilon = 10^{-3}$ and $\varepsilon = 5 \cdot 10^{-4}$.

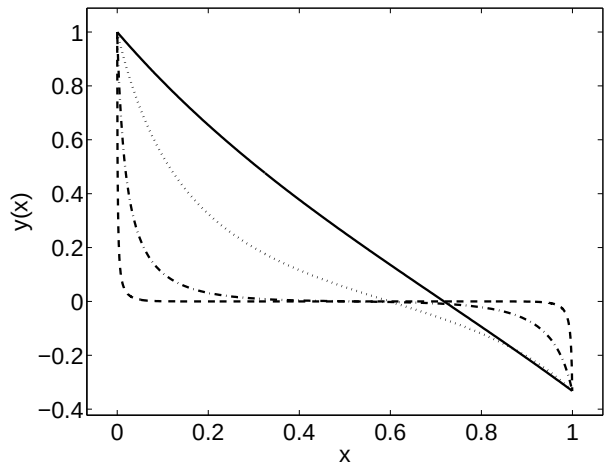

Figure 12: Solutions of T3 for $\varepsilon = 1$ (solid line), $\varepsilon = 0.1$ (dotted line), $\varepsilon = 10^{-2}$ (dash-dotted line) and $\varepsilon = 5\ 10^{-4}$ (dashed line).

Problem T4 is also nonlinear,

$$y^{(4)}(x) - \varepsilon(y'(x)y''(x) - y(x)y'''(x)) = 0, \tag{43a}$$

$$y(0) = 0, \;\; y'(0) = 0, \;\; y(1) = 1, \;\; y'(1) = 0, \tag{43b}$$

with solutions shown in Figure 13. Here, ε is a 'large' parameter, and we selected the value $\varepsilon = 100$ for collocation orders 4 and 6, and $\varepsilon = 1000$ for collocation order 8.

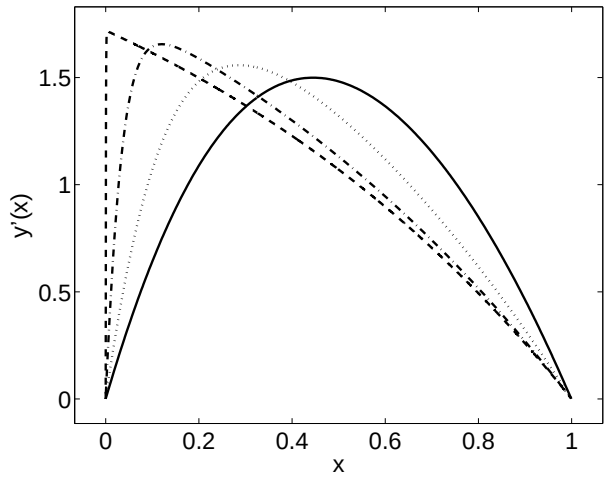

Figure 13: First solution derivative of Problem T4 for $\varepsilon = 1$ (solid line), $\varepsilon = 5$ (dotted line), $\varepsilon = 20$ (dash-dotted line) and $\varepsilon = 1300$ (dashed line).

Problem T5 is singular and has been discussed in [26]. It has the form

$$y'(x) = \frac{1}{x}\begin{pmatrix} 0 & 1 \\ 2 & 6 \end{pmatrix} y(x) - \begin{pmatrix} 0 \\ 4k^4x^5\sin(k^2x^2) + 10x\sin(k^2x^2) \end{pmatrix}, \tag{44a}$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} y(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} y(1) = \begin{pmatrix} 0 \\ \sin(k^2) \end{pmatrix}, \tag{44b}$$

with $k = 5$.

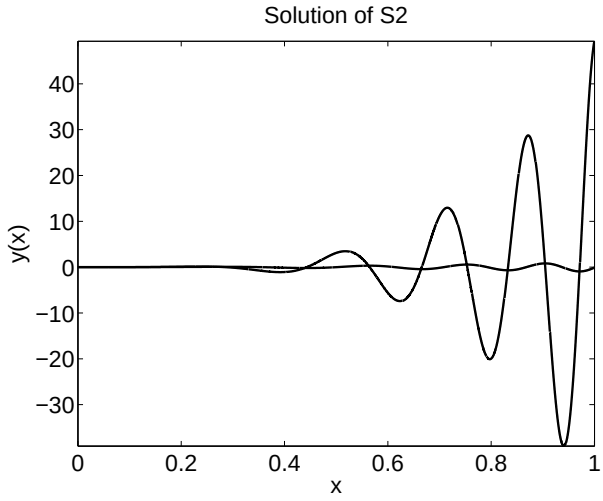

Figure 14: Solution of Problem T5.

References

[1] U. Ascher, J. Christiansen and R.D. Russell, *A Collocation Solver for Mixed Order Systems of Boundary Value Problems*, Math. Comp. **33**, 659–679 (1979).

[2] U. Ascher, J. Christiansen and R.D. Russell, *Collocation Software for Boundary-Value ODEs*, ACM Trans. Math. Software **7**, 209–222 (1981).

[3] U. Ascher, R.M.M. Mattheij, and R.D. Russell, *Numerical solution of boundary value problems for ordinary differential equations*, Prentice-Hall, Englewood Cliffs, New York (1988).

[4] W, Auzinger, G. Kneisl, O. Koch and E.B. Weinmüller, *A Collocation Code for Boundary Value Problems in Ordinary Differential Equations*, Numer. Algorithms **33**, 27–39 (2003).

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

[6] W. Auzinger, O. Koch and E.B. Weinmüller, *Efficient mesh selection for collocation methods applied to singular BVPs*, J. Comp. Appl. Math. **180**, 213–227 (2005).

[7] W. Auzinger, O. Koch, G. Pulverer and E.B. Weinmüller, *Performance of collocation software for singular BVPs*, Technical Report, ANUM Preprint No. 4/04, Vienna University of Technology, Austria (2004).

[8] I. Babuska, and W.C. Rheinboldt, *Analysis of optimal finite-elemt meshes in R1*, Math. Comp. **33** 435-463 (1979).

[9] G. Beckett, J.A. Mackenzie, A. Ramage, and D.M. Slone, *On the Numerical Solution of One-Dimensional PDEs Using Adaptive Methods Based on Equidistribution*, J. Comput. Phys. **167** 372–392 (2001).

[10] C. de Boor, *Good approximations by splines with variable knots*, In A. Meir and A. Sharma, editors, *Spline Functions and Approximation Theory*, 57–73, Balel and Stuttgard, 1973, Birkhüser Verlag.

[11] C.J. Budd, O. Koch, and E.B. Weinmüller, *Self-similar blow-up in nonlinear PDEs*, AURORA Technical Report, TR-2004-15, Institute for Analysis and Scientific Computing, Vienna University of Technology, Austria, 2004. Available at `http://www.vcpc.univie.ac.at/aurora/publications/`.

[12] C.J. Budd, O. Koch, and E.B. Weinmüller, *Computation of Self-Similar Solution Profiles for the Nonlinear Schrödinger Equation*, Computing **77**, 335–346 (2006).

[13] C.J. Budd, O. Koch, and E.B. Weinmüller, *From Nonlinear PDEs to Singular ODE*s, Appl. Numer. Math. **56**, 413–422 (2006).

[14] G.F. Carey and H.T. Dinh, *Grading Functions and Mesh Redistribution*, Siam J. Numer. Anal. **22**, 1028–1040 (1985).

[15] K. Chen, *Error Equidistrubution and Mesh Adaptation*, Siam J. Sci. Comput. **15**, 798–818 (1994).

[16] C.C. Christara and K.S. Ng, *Optimal Quadratic and Cubic Spline Collocation on Nonuniform Partitions*, Computing **76**, 227 - 257 (2006).

[17] C.C. Christara and K.S. Ng, *Adaptive Techniques for Spline Collocation*, Computing **76**, 259 - 277 (2006).

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

[19] W. Huang, Y. Ren, and R.D. Russell, *Moving Mesh Partial Differentail Equations (MMPDEs) Based on the Equidistribution Principle*, SIAM J. Numer. Anal. **31**, 709 (1994).

[20] W. Huang, *Variational Mesh Adaptation: Isotropy and Equidistribution*, J. Comput. Phys. **174**, 903–924 (2001).

[21] W. Huang, and W. Sun, *Variational Mesh Adaptation II: Error Estimates and Monitor Functions*, SIAM J. Numer. Anal. **13**, 775–813 (1976).

[22] S. Ilie, G. Söderlind and R. Corless, *Adaptivity and Computational Complexity in the Numerical Solution of ODEs*. J. Complexity **24**, 341-361 (2008).

[23] G. Kitzhofer, *Numerical treatment of implicit singular BVPs*, Ph.D. Thesis, Institute for Analysis and Scientific Computing, Vienna University of Technology, Austria. In preparation.

[24] G. Kitzhofer, O. Koch, and E.B. Weinmüller, *Collocation methods for the computation of bubble-type solutions of a singular boundary value problem in hydrodynamics*, J. Sci. Comp.. To appear. Available at `http://www.math.tuwien.ac.at/~ewa`.

[25] V. Pereyra and E.G Sewell, *Mesh Selection for Discrete Solution of Boundary Problems in Ordinary Differential Equations*, Numer. Math. **23**, 261–268 (1975).

[26] W. Polster, *Ein Algorithmus zur Gittersteuerung bei Kollokationsverfahren für singuläre Randwertprobleme*, Master Thesis, Institute for Applied Mathematics and Numerical Analysis, Vienna University of Technology, Austria (2001).

[27] J.D. Pryce, *On the Convergence of Iterated Remeshhing*, IMA J. Numer. Anal. **9**, 315–335 (1989).

[28] I. Rachunková, O. Koch, G. Pulverer and E.B. Weinmüller, *On a singular boundary value problem arising in the theory of shallow membrane caps*, J. Math. Anal. Appl. **332**, 523–541 (2007).

[29] 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).

[30] R.D. Russell and J. Christiansen, *Adaptive Mesh Selection Strategies for Solving Boundary Value Problems*, SIAM J. Numer. Anal. **15**, 59–80 (1978).

[31] L. Shampine, J. Kierzenka, and M. Reichelt, *Solving Boundary Value Problems for Ordinary Differential Equations in* Matlab *with* `bvp4c` (2000). Available at `ftp://ftp.mathworks.com/pub/doc/papers/bvp/`.

[32] G. Söderlind, *Digital Filters in Adaptive Time-stepping*, ACM Trans. Math. Software **29**, 1–26 (2003).

[33] G. Söderlind, *Time-step Selection Algorithms: Adaptivity, Control and Signal Processing*, Appl. Num. Math. **56**, 488–502 (2006).

[34] J.M. Stockie, J.A. Mackenzie, and R.D. Russell, *A Moving Mesh Method for One-Dimensional Hyeperbolic Conservation Laws*, SISC **22**, 1791–1813 (2001).

[35] H. Tang, T. Tang, *Adaptive Methods for One- and Two-Dimensional Hyperbolic Conservation Laws*, SINUM **41**, 487–515 (2003).