European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 5, no. 1-2, 2010, pp. 113-134
ISSN 1790–8140

The New MATLAB Code `bvpsuite` for the Solution of Singular Implicit BVPs[1]

G. Kitzhofer, O. Koch, G. Pulverer, Ch. Simon, and E.B. Weinmüller[2]

Institute for Analysis and Scientific Computing (E101),
Vienna University of Technology,
Wiedner Hauptstrasse 8–10, A-1040 Wien, Austria

Received 21 January, 2010; accepted in revised form 23 March, 2010

Abstract: Our aim is to provide the open domain MATLAB code `bvpsuite` for the efficient numerical solution of boundary value problems in ordinary differential equations. Motivated by applications, we are especially interested in designing a code whose scope is appropriately wide, including fully implicit problems of mixed orders, parameter dependent problems, problems with unknown parameters, problems posed on semi-infinite intervals, eigenvalue problems and differential algebraic equations of index 1. Our main focus is on singular boundary value problems in which singularities in the differential operator arise. We first briefly recapitulate the analytical properties of singular systems and the convergence behavior of polynomial collocation used as a basic solver in the code for both singular and regular ordinary differential equations and differential algebraic equations. We also discuss the a posteriori error estimate and the grid adaptation strategy implemented in our code. Finally, we describe the code structure and present the performance of the code which has been equipped with a graphical user interface for an easy use.

© 2010 European Society of Computational Methods in Sciences and Engineering

Keywords: Boundary value problems – singularity of the first kind – singularity of the second kind – analysis – collocation methods – a posteriori error estimation – mesh adaptation – pathfollowing – eigenvalue problems – index-1 differential algebraic equations

Mathematics Subject Classification: 65L10, 65L20, 65L50, 65L60

Introduction

The code `bvpsuite` is designed to solve general implicit mixed order systems for boundary value problems (BVPs) in ordinary differential equations (ODEs) specified in (20)–(21), subject to multi-point boundary conditions. The problem can be posed on a finite interval $[a, b]$ or on a semi-infinite interval $[a, \infty)$, where $a \geq 0$. In the latter case the code provides an automatic transformation of the semi-infinite interval to a finite domain. Parameter dependent problems are also in the scope of `bvpsuite`. We use the pseudo arclength parametrization to move around turning points in the solution/parameter path.

For years, special focus of our research has been on the analysis and numerical treatment of ODEs with time singularities. Therefore, in the introduction, we recapitulate the theoretical results available for model

[1]Published electronically October 15, 2010
[2]Corresponding author. E-mail: e.weinmueller@tuwien.ac.at

problems with such singularities. Let us consider the numerical solution of singular BVPs of the form

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

$$B_0 z(0) + B_1 z(1) = \beta, \tag{2}$$

where $\alpha \geq 1$, 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_0 and B_1 are constant matrices which are subject to certain restrictions for a well-posed problem. (1) is said to feature a *singularity of the first kind* for $\alpha = 1$, while for $\alpha > 1$ the problem has a *singularity of the second kind*, also commonly referred to as *essential singularity*. The analytical properties of problem (1)–(2) have been discussed in [16], [19] with a special focus on the most general boundary conditions which guarantee well-posedness of the problem. When analyzing singular problems, we first note that their direction field is very unsmooth, especially close to the singular point. Consequently, depending on the spectrum of the matrix $M(0)$, we can encounter unbounded contributions to the solution manifold, such that $z \in C(0, 1]$. However, irrespective of the eigenvalues of $M(0)$, by posing proper homogeneous initial conditions, we can extend the above solution to $z \in C[0, 1]$. It also turns out that in such a case the condition $M(0)z(0) = 0$ must hold. For singular problems the solution's smoothness depends not only on the smoothness of the inhomogeneity f but also on the size of the real parts of the eigenvalues of $M(0)$. To compute the numerical solution of (1)–(2) we use polynomial collocation. Our decision to use polynomial collocation was motivated by its advantageous convergence properties for (1)–(2), while in the presence of a singularity other high order methods show order reductions and become inefficient. In [8], [17], and [27] convergence results for collocation applied to problems with a singularity of the first kind, $\alpha = 1$, were shown. The usual high-order superconvergence at the mesh points does not hold in general for singular problems, however, the uniform superconvergence is preserved (up to logarithmic factors), see [27] for details.

Motivated by these observations, we have implemented the present MATLAB code. We stress that for `bvpsuite` one singular point at either endpoint or two singular points at both endpoints of the interval of integration[3] are admissible. The program can be applied directly to such singular BVPs and no prehandling is necessary. Otherwise a reformulation of the problem may be necessary, cf. (33) and (34). For higher efficiency, we provide an estimate of the global error and adaptive mesh selection. Transformation of problems posed on semi-infinite intervals to a finite domain makes the solution of such problems also accessible to our methods. All these algorithmic components have been integrated into the code. While the first version of the code, `sbvp`, solves explicit first order ODEs [5], the present version, `bvpsuite1.0`, can be applied to arbitrary order problems also in implicit formulation. Consequently, systems of differential algebraic equations (DAEs) [28] are also in the scope of the code. We have also implemented a module for eigenvalue value problems (EVPs) in which we recast the EVP as an appropriately defined BVP. As already mentioned, a pathfollowing strategy extends the scope of the code to parameter dependent problems. For special problem classes, such as singularly perturbed models, systems of DAEs, parameter dependent problems and EVPs very good and efficient software already exists. We have not performed comparisons with those codes. However, we assess the properties of `bvpsuite` when the code is applied to *singular BVPs* by comparing it with other, well-established software for BVPs in ODEs. Numerical simulation of relevant applications illustrates the scope and the performance of the implementation.

[3] The following equation is a typical model for such a situation

$$z'(t) = \frac{M(t)}{t^{\alpha_1}(t-1)^{\alpha_2}} z(t) + f(t, z(t)), \;\; t \in (0, 1), \;\; \alpha_1, \alpha_2 \geq 1.$$

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Notation

Throughout the paper, the following notation is used. For functions $y \in C[0,1]$, we define the maximum norm,

$$\|y\| := \max_{0\le t\le 1} |y(t)|.$$

For the numerical analysis, we define meshes

$$\Delta := (\tau_0, \tau_1, \ldots, \tau_N), \tag{3}$$

and

$$h_i := \tau_{i+1} - \tau_i, \quad J_i := [\tau_i, \tau_{i+1}], \quad i = 0, \ldots, N-1, \quad \tau_0 = 0,\ \tau_N = 1. \tag{4}$$

For a simpler presentation, we restrict the discussion to equidistant meshes,

$$h_i = h, \quad i = 0, \ldots, N-1,$$

see Figure 1. However, the results also hold for nonuniform meshes which have a limited variation in the

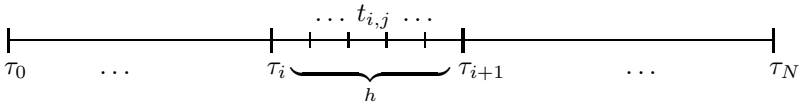

Figure 1: The computational grid

stepsizes [17]. On Δ, we define corresponding grid vectors

$$u_\Delta := (u_0, \ldots, u_N) \in \mathbb{R}^{(N+1)n}. \tag{5}$$

The norm on the space of grid vectors is given by

$$\|u_\Delta\|_\Delta := \max_{0\le k\le N} |u_k|. \tag{6}$$

For a continuous function $y \in C[0,1]$, we denote by R_Δ the pointwise projection onto the space of grid vectors,

$$R_\Delta(y) := (y(\tau_0), \ldots, y(\tau_N)). \tag{7}$$

For collocation, m points $t_{i,j},\ j = 1, \ldots, m$, are inserted in each subinterval J_i. We choose the same distribution of collocation points in every subinterval, thus yielding the (fine) grid[4]

$$\Delta^m = \Delta \cup \{t_{i,j} = \tau_i + \rho_j h, \quad i = 0, \ldots, N-1,\ j = 1, \ldots, m\}, \tag{8}$$

with

$$0 < \rho_1 < \rho_2 \cdots < \rho_m \le 1. \tag{9}$$

In the case where all collocation points are inner points of J_i, $\rho_1 < \ldots < \rho_m < 1$, we define $\rho_{m+1} := 1$. In the convergence analysis in Section 1, we restrict ourselves to grids where $\rho_1 > 0$ to avoid a special treatment of the singular point $t = 0$ [13]. For a grid Δ^m, $u_{\Delta^m},\ \|\cdot\|_{\Delta^m}$ and R_{Δ^m} are defined analogously to (5)–(7).

[4]For convenience, we denote τ_i by $t_{i,0} \equiv t_{i-1,m+1},\ i = 1, \ldots, N$.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

1 Collocation methods

In this section, we discuss collocation with continuous, piecewise polynomial functions of degree $\leq m$.

Let us denote by $\mathbb{P}_m$ the Banach space of continuous, piecewise polynomial functions $q \in \mathbb{P}_m$ of degree $\leq m,\ m \in \mathbb{N}$ (m is called the *stage order* of the method), equipped with the norm $\|\cdot\|$. As an approximation for the exact solution z of (1)–(2), we define an element of $\mathbb{P}_m$ which satisfies the differential equation (1) at a finite number of points and which is subject to the same boundary conditions. Thus, we are seeking a function $p(t) = P_i(t),\ t \in J_i,\ i = 0, \ldots, N-1$, in $\mathbb{P}_m$ which satisfies

$$p'(t_{i,j}) = \frac{M(t_{i,j})}{t_{i,j}^{\alpha}} p(t_{i,j}) + f(t_{i,j}, p(t_{i,j})), \tag{10a}$$
$$i = 0, \ldots, N-1,\ j = 1, \ldots, m,$$
$$B_0 p(0) + B_1 p(1) = \beta. \tag{10b}$$

We consider collocation on grids Δ^m (see (8)), subject to the restriction $\rho_1 > 0$.

In [27] the following convergence result was shown for problems with a singularity of the first kind, $\alpha = 1$ in (10a):

Theorem 1.1 *Assume that $M \in C^{m+2}[0,1]$, f is $m+1$ times continuously differentiable in $[0,1] \times \mathbb{R}^n$ with $\frac{\partial f}{\partial z}$ bounded on that domain and $\sigma_+ > m+2$, where σ_+ is the smallest positive real part of the eigenvalues of the matrix $M(0)$. Then the collocation scheme* (10) *has a unique solution $p \in \mathbb{P}_m$ in a neighborhood of an isolated solution $z \in C^{m+2}[0,1]$ of* (1)–(2)*. This solution can be computed using Newton's method, which converges quadratically. Moreover,*

$$\|p - z\| = O(h^m), \tag{11}$$
$$\left\| \frac{M(0)}{t}(p(t) - z(t)) \right\| = O(h^m), \quad t \in [0,1], \tag{12}$$
$$\left\| p^{(k+1)} - z^{(k+1)} \right\| = O(h^{m-k}), \quad k = 0, \ldots, m-1, \tag{13}$$
$$\left\| p'(t) - \frac{M(t)}{t} p(t) - f(t, p(t)) \right\| = O(h^m), \quad t \in [0,1]. \tag{14}$$

Note that the condition $\sigma_+ > m+2$ does not impose a restriction of generality. For $\sigma_+ \leq m+2$ we cannot in general guarantee that $z \in C^{m+2}$ [16], and therefore we cannot expect to observe the desired convergence orders in this case. Consequently, the restriction $\sigma_+ > m+2$ is natural in this context. If this assumption is not satisfied, we can always transform the equation (1) by letting $t \to t^{\lambda},\ 1 > \lambda > 0$, whence $\sigma_+ \to \sigma_+/\lambda$. Thus, as already mentioned, the assumption $\sigma_+ > m+2$ imposes no restriction of generality.

The usual high-order superconvergence at the mesh points does not hold in general for singular problems, however, the uniform superconvergence is preserved (up to logarithmic factors):

$$\|R_\Delta(p) - R_\Delta(z)\|_\Delta = O(h^{m+1} |\ln(h)|^{n_0 - 1}), \tag{15}$$
$$\|p - z\| = O(h^{m+1} |\ln(h)|^{n_0 - 1}), \tag{16}$$

if

$$\int_0^1 s^k \prod_{l=1}^{m} (s - \rho_l)\ ds = 0, \quad k = 0, \ldots, \nu - 1 \tag{17}$$

holds with $\nu \geq 1$, see [8], [27] for details.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

For problems with an essential singularity, no theoretical results are known for general high-order collocation methods. However, we observed experimentally that the *stage order* $O(h^m)$ is retained for any choice of symmetric collocation points. The superconvergence orders for $\nu \geq 1$ in (17) are

$$\|R_\Delta(p) - R_\Delta(z)\|_\Delta = O(h^{m+\gamma}), \tag{18}$$

$$\|p - z\| = O(h^{m+\gamma}), \tag{19}$$

where $0 < \gamma = \gamma(\alpha) < 1$, and γ decreases with increasing α in (1). For non-symmetric collocation points, we observed rapid divergence of the numerical solution. Experimental evidence for these propositions is given in [7].

The analysis of the box scheme given in [18] implies that its order of convergence is $1 + \gamma$, where $0 < \gamma < 1$. Since the box scheme is equivalent to collocation at Gaussian points with $m = 1$, this is consistent with the above conjecture.

2 Basic Solver in the MATLAB Code `bvpsuite`

The code is designed to solve systems of differential equations of arbitrary mixed order including zero[5], subject to initial or boundary conditions,

$$F\big(t, p_1, \ldots, p_s, z_1(t), z_1'(t), \ldots, z_1^{(l_1)}(t), \ldots, z_n(t), z_n'(t), \ldots, z_n^{(l_n)}(t)\big) = 0, \tag{20}$$

$$B\big(p_1, \ldots, p_s, z_1(c_1), \ldots, z_1^{(l_1-1)}(c_1), \ldots, z_n(c_1), \ldots, z_n^{(l_n-1)}(c_1), \ldots, z_1(c_q), \ldots, z_1^{(l_1-1)}(c_q), \ldots, z_n(c_q), \ldots, z_n^{(l_n-1)}(c_q)\big) = 0, \tag{21}$$

where the solution $z(t) = (z_1(t), z_2(t), \ldots, z_n(t))^T$, and the parameters p_i, $i = 1, \ldots, s$, are unknown. In general, $t \in [a, b]$ or $t \in [a, \infty)$[6], $a \geq 0$. Moreover,

$$F : [a, b] \times \mathbb{R}^s \times \mathbb{R}^{l_1} \times \cdots \times \mathbb{R}^{l_n} \to \mathbb{R}^n,$$

and

$$B : \mathbb{R}^s \times \mathbb{R}^{ql_1} \times \cdots \times \mathbb{R}^{ql_n} \to \mathbb{R}^{l+s},$$

where $l := \sum_{i=1}^n l_i$. Note that boundary conditions can be posed on any subset of distinct points $c_i \in [a, b]$, $a \leq c_1 < c_2 < \cdots < c_q \leq b$.

For the numerical treatment, we assume that the boundary value problem (20)–(21) is well-posed and has a locally unique solution z.

In order to find a numerical solution of (20)–(21) we introduce a mesh Δ, partitioning the interval $[a, b]$ as shown in Figure 1. Every subinterval J_i contains m collocation points $t_{i,j} = \tau_i + \rho_j h$, $j = 1, \ldots, m$, with

$$0 < \rho_1 < \rho_2 \cdots < \rho_m < 1. \tag{22}$$

To avoid a special treatment of the possible singular points $t = a$ and $t = b$, [13], grids with $\rho_1 > 0$ and $\rho_m < 1$, e.g. Gaussian points or inner equidistant points, are used. Let $\mathbb{P}_r$ be the space of piecewise polynomial functions of degree $\leq r$, see Section 1, which are globally continuous in $[a, b]$. In every subinterval J_i we make an ansatz $P_{i,k} \in \mathbb{P}_{m+l_k-1}$ for the k-th solution component z_k, $k = 1, \ldots, n$, of the problem (20)–(21). In order to compute the coefficients in the ansatz functions we require that (20)–(21) is satisfied exactly at the collocation points. Moreover, we require that the collocation polynomial $p(t) := P_i(t)$, $t \in J_i$, is

[5] This means that differential algebraic equations are also in the scope of the code.

[6] For the extension to unbounded domains, see Section 5.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

a globally continuous function on $[a,b]$ with components in $C^{l_i-1}[a,b],\ i=1,\dots,n$, and satisfies the boundary conditions. All these conditions imply a nonlinear system of equations for the unknown coefficients in the ansatz function. For more details see [3] and [23]. It can be easily seen that the number of equations in this nonlinear system amounts to $Nmn+Nl+s$. Every component of the polynomial $P_{i,k}$, $i=0,\dots,N-1$, $k=1,\dots,n$ is characterized by $m+l_k$ unknown coefficients, and therefore for every i the polynomial $P_{i,k}$ has $nm+l$ coefficients to be determined. This together with s unknown parameters adds to a total number of unknowns which is also equal to $N(nm+l)+s$.

For the representation of the collocation polynomial p we use the Runge-Kutta basis, see [3], and solve the resulting nonlinear system for the coefficients in this representation by a Newton iteration implemented in the subroutine 'solve_nonlinear_sys.m' from the MATLAB code `sbvp` [5], which is based on the 'fast frozen' Newton method.

3 Error Estimate for the Global Error of the Collocation

Our estimate for the global error of the collocation solution is a classical error estimate based on mesh halving. In this approach, we compute the collocation solution at N points on a grid Δ with step sizes h_i and denote this approximation by $p_\Delta(t)$. Subsequently, we choose a second mesh Δ_2 where in every interval J_i of Δ we insert two subintervals of length $h_i/2$. On this mesh, we compute the numerical solution based on the same collocation scheme to obtain the collocating function $p_{\Delta_2}(t)$. Using these two quantities, we define

$$\mathcal{E}(t) := \frac{2^m}{1-2^m}(p_{\Delta_2}(t)-p_\Delta(t)) \tag{23}$$

as an error estimate for the approximation $p_\Delta(t)$. Assume that the global error $\delta(t) := p_\Delta(t)-z(t)$ of the collocation solution can be expressed in terms of the principal error function $e(t)$,

$$\delta(t) = e(t)h_i^m + O(h_i^{m+1}), \quad t\in J_i, \tag{24}$$

where $e(t)$ is independent of Δ. This property is known to hold for many standard discretization methods, [33], [40], in case that the analytical solution of the problem is appropriately smooth. Note that a similar representation holds for the residual of the collocation solution [33]. Then obviously the quantity $\mathcal{E}(t)$ satisfies $\mathcal{E}(t)-\delta(t)=O(h^{m+1})$. This holds for problems with a singularity of the first kind and for regular problems. However, numerical results reported in [6] indicate that in case of an essential singularity (24) reads

$$\delta(t) = e(t)h_i^m + O(h_i^{m+\gamma}), \quad t\in J_i, \tag{25}$$

with $\gamma<1$. Generally, estimates of the global error based on mesh halving work well for both problems with a singularity of the first kind and for essentially singular problems [6]. Since they are also applicable to higher-order problems and problems in implicit form (as for example DAEs) without the need for modifications, we have implemented this strategy in our code `bvpsuite`.

4 Adaptive Mesh Selection

The mesh selection strategy implemented in `bvpsuite` was proposed and investigated in [31]. Most modern mesh generation techniques in two-point boundary value problems construct a smooth function mapping a uniform auxiliary grid to the desired nonuniform grid. In [31] a new system of control algorithms for constructing a grid density function $\phi(t)$ is described. The local mesh width $h_i=\tau_{i+1}-\tau_i$ is computed as $h_i=\epsilon_N/\varphi_{i+1/2}$, where $\epsilon_N=1/N$ is the accuracy control parameter corresponding to $N-1$ interior points, and the positive sequence $\Phi=\{\varphi_{i+1/2}\}_{i=0}^{N-1}$ is a discrete approximation to the continuous density function $\phi(t)$, representing the mesh width variation. Using an error estimate, a feedback control

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

law generates a new density from the previous one. Digital filters may be employed to process the error estimate as well as the density.

For boundary value problems, an adaptive algorithm must determine the sequence $\Phi^{[\nu]}$ in terms of problem or solution properties. True adaptive approaches equidistribute some *monitor function*, a measure of the residual or error estimate, over the interval. As $\Phi^{[\nu]}$ 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 a local grid change typically has global effects. The techniques developed here avoid this difficulty by restricting the error estimates to those having the property that the estimated error on the interval J_i only depends on the local mesh width, $h_i = \epsilon_N/\varphi_{i+1/2}$.

In order to be able to generate the mesh density function, we decided to use the residual $r(t)$ to define the monitor function. The values of $r(t)$ are available from the substitution of the collocation solution $p(t)$ into the analytical problem (20)–(21). We first compute

$$R_k(\tau_{i+1/2}) = \int_{\tau_i}^{\tau_{i+1}} r_k(t)\,dt \approx \frac{r_k(\tau_i) + r_k(\tau_{i+1})}{2}(\tau_{i+1} - \tau_i)$$

for $i = 0, \ldots, N-1$ and for each component $r_k,\ k = 1, \ldots, n$, of the residual r. Now, for each subinterval J_i, we calculate

$$\hat{R}(\tau_{i+1/2}) := \left(\sum_{k=1}^{n} R_k^2(\tau_{i+1/2})\right)^{\frac{1}{2}}, \quad i = 0, \ldots, N-1,$$

to obtain the values of the monitor function related to subintervals J_i, necessary for the update of $\Phi^{[\nu]}$.

While the residual based monitor function $R(t) := \hat{R}(\tau_{i+1/2}),\ t \in J_i$, is used to update the mesh density, the number of necessary mesh points in the final mesh is determined from the requirement that the absolute global error satisfies the tolerance. The mesh halving routine provides the values of the error estimate (23) in the entire interval $[a, b]$, so we can compute

$$G_{\Delta^m} := \max_t(\max_{1\le k\le n} |\mathcal{E}_k(t)|), \quad t \in \Delta^m.$$

The number of points for the next iteration step is predicted from

$$N_{\nu+1} = N_0 \left(\frac{G_{\Delta^m}}{0.9\mathrm{TOL}}\right)^{1/(m+1)}, \tag{26}$$

where $N_0 = 50$ is the fixed number of points in the control grid. Below, we specify in more detail the grid adaptation routine implemented in the code.

1. Grid generation, finding the optimal density function, is separated from mesh refinement, finding the proper number of mesh points. We first try to provide a good density function Φ on a rather coarse mesh with a fixed number of points $N_0 = 50$. The mesh density function is chosen to equidistribute the monitor function $R(t)$.

2. For each density profile in the above iteration, we estimate the number of mesh points necessary to reach the tolerance, according to (26).

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

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

4. Since the computation of a residual is reasonably cheap we always update the density profile to make use of the information provided by the most recent available numerical solution associated with the function $\Phi^{[\nu]}$.
5. We finally solve the problem on the mesh based on $\Phi^{[\nu+1]}$ with $N^{[\nu+1]}$ mesh points, and estimate the global error of this approximation. If the accuracy requirement is satisfied, we stop the calculations, otherwise we refine again.

In Figure 2 we illustrate how the mesh adaptation is performing when it is applied to solve problem (29)–(30) below for $k = 5$.

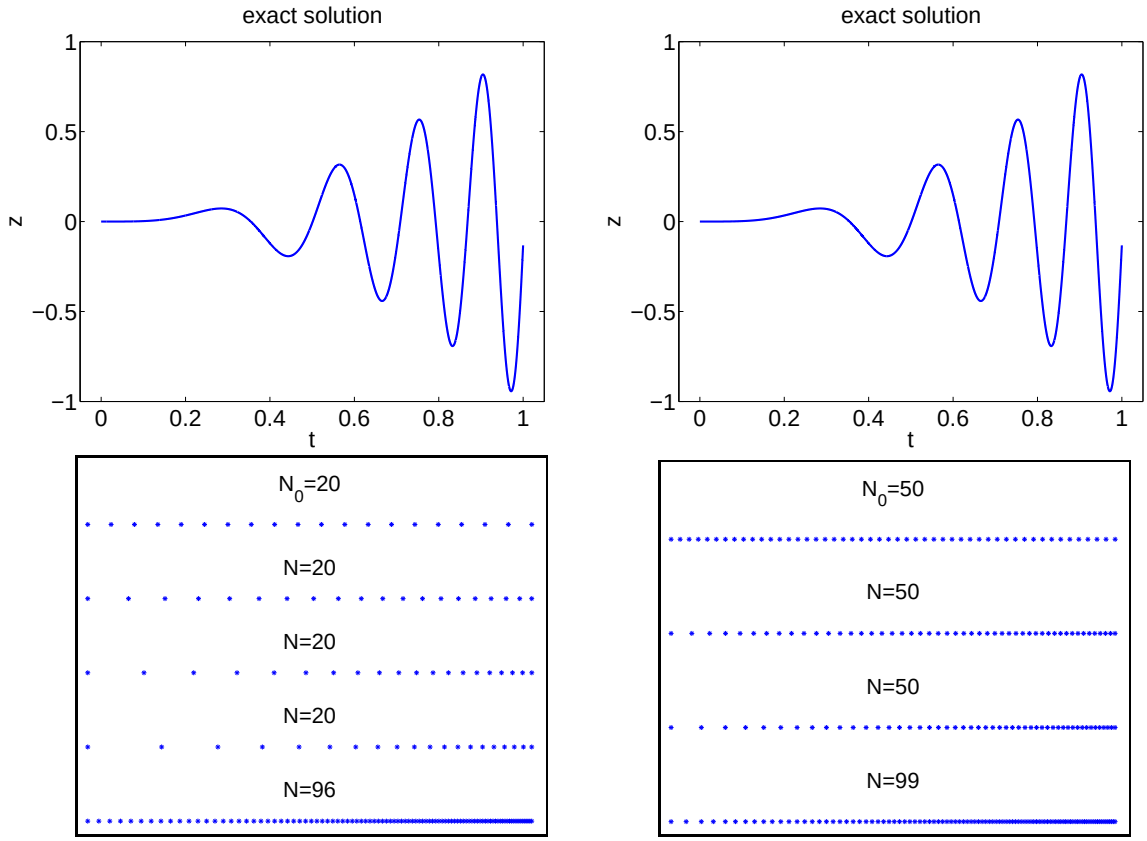

Figure 2: Problem (29)–(30): Steps of the grid adaptation procedure carried out for collocation at $m = 4$ Gaussian points and $\mathrm{TOL}_a = 10^{-6}$

For more details and the results of numerical tests, we refer the reader to [31].

5 Pathfollowing and Problems on Semi-Infinite Intervals

In this and in the following sections, we discuss the scope of `bvpsuite` and its special features which allow to cover a very wide range of applications. First of all, our code realizes a pathfollowing strategy to follow solution branches in dependence of a known parameter. To describe the strategy in general terms, we consider (1)–(2) as a parameter-dependent operator equation

$$F(y; \lambda) = 0, \tag{27}$$

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

where $F : Y \times \mathbb{R} \to Z$, and Y, Z are Banach spaces (of possibly infinite dimension).

Pathfollowing in this general setting has been discussed in detail in [41].

We are particularly interested in computing solution branches Γ with *turning points*. By definition, in a turning point the solution of (27) constitutes a local maximum (or minimum) of λ, and consequently is not locally unique as a function of the parameter λ. The situation is illustrated in Figure 3. There, we plot some functional of the solution against the parameter λ. The arrows indicate the turning points. Thus, in a turning point we cannot parametrize Γ as a function of λ. However, it is sufficient for our procedure that a tangent is uniquely determined at all points of Γ. This is guaranteed by realistic assumptions formulated for our problem in [25].

Now, we proceed by describing our pathfollowing strategy. As explained in [25], our assumptions on the problem ensure that at a point $(y_0, \lambda_0) \in \Gamma$, a tangent can be uniquely determined up to the sign. Additional criteria determine how to choose the direction. On the tangent just computed, a predictor (y_P, λ_P) is chosen for the computation of the next point on Γ, and finally a corrector equation is solved yielding (y_C, λ_C). One step of our procedure starting at (y_0, λ_0) is illustrated in Figure 3.

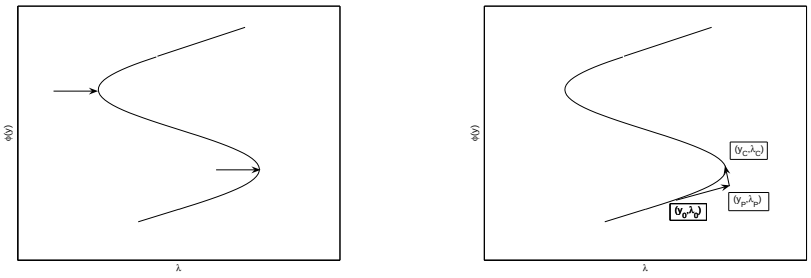

Figure 3: A solution branch with two turning points (left), one step of the pathfollowing procedure (right).

As one example to demonstrate that our pathfollowing strategy indeed works for singular boundary value problems and generates meshes adapted to the solution profile, in [25] we considered an example from [14], describing the buckling of a spherical shell.

We followed the solution path Γ shown in Figure 4, starting at $\lambda = 0$. Figure 4 shows the maximum norm of the first solution component, $\|\beta\|_\infty$ along the path Γ. The crosses indicate points of Γ where the solution profiles of β and the second solution component Ψ are plotted in Figure 5, together with the meshes generated by our adaptive mesh selection procedure. A comparison with [14, Figure 10] shows that the solution is computed reliably and obviously the meshes are denser where the solution varies more rapidly.

Our code can also treat problems which are posed on semi-infinite intervals $t \in [a, \infty),\ a > 0$ (and by a splitting of the interval, also for $a = 0$). In order to exploit our efficient and robust mesh selection strategy also in this case, we use the transformation $t = \frac{a}{\tau},\ z(t) = x\left(\frac{a}{\tau}\right)$ to restate

$$x'(\tau) = \tau^\beta f(\tau, x(\tau)), \quad \tau \in [a, \infty), \quad \beta > -1,$$

as

$$z'(t) = -\frac{1}{t^{\beta+2}} f(1/t, z(t)), \quad t \in (0, 1].$$

This is in general a problem with an essential singularity, which however is in the scope for our collocation methods, error estimation procedure and adaptive mesh refinement. In this approach, the mesh is adapted only according to the unsmoothness of the solution without the need for mesh grading on long intervals, and moreover no truncation of the unbounded interval is necessary. This strategy was employed successfully for example in [11], [12] and [24].

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

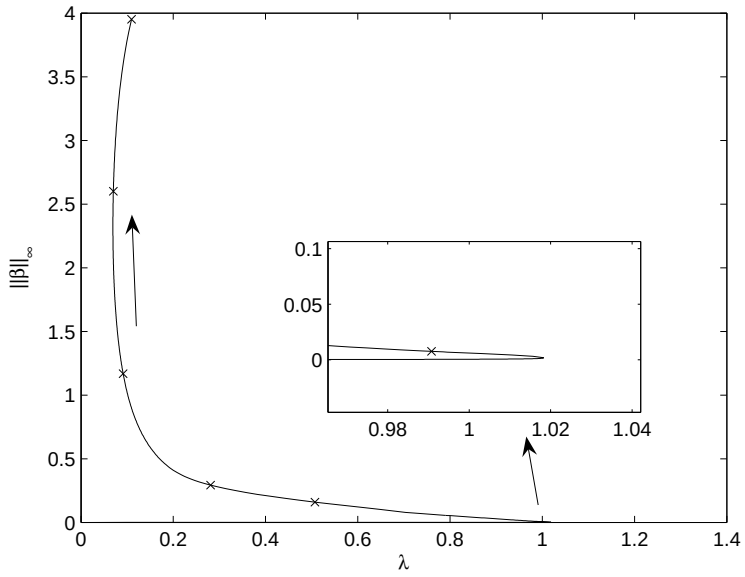

Figure 4: Values of $\|\beta\|_\infty$ along a solution branch.

6 Code Structure

To install the code, create an empty folder and copy the files of `bvpsuite1.0.zip` into it. This archive is available from `http://www.math.tuwien.ac.at/~ewa`. The code has been extensively tested for the MATLAB versions 7.1–7.2 (R2006a) and needs the Symbolic Math Toolbox (optimized for Version 3.1) based on the Maple Engine. This toolbox is only necessary for the *automatic* transformation of problems posed on the semi-infinite interval to the finite one. Newer versions of MATLAB either do not contain the Symbolic Math Toolbox or it is based on the MuPAD Engine. Unfortunately the syntax differs, so these versions of the Symbolic Math Toolbox are not compatible with the above modules of `bvpsuite1.0`. For computations involving problems posed on finite domains newer versions of MATLAB, e.g. MATLAB version 7.8 (R2009a), can also be used. For future releases of `bvpsuite` it is planned to make the code compatible with the new engine syntax.

6.1 Files in the Package

The package `bvpsuite` contains the following m-files

- bvpsuite.m – main routine to start the graphical user interface (GUI).
- equations.m – contains the most important parts of the code, e.g. setting up the nonlinear system of equations for the Newton solver.
- solve_nonlinear_sys.m – contains the Newton solver.
- run.m – manages routine calls.
- errorestimate.m – provides error estimates.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

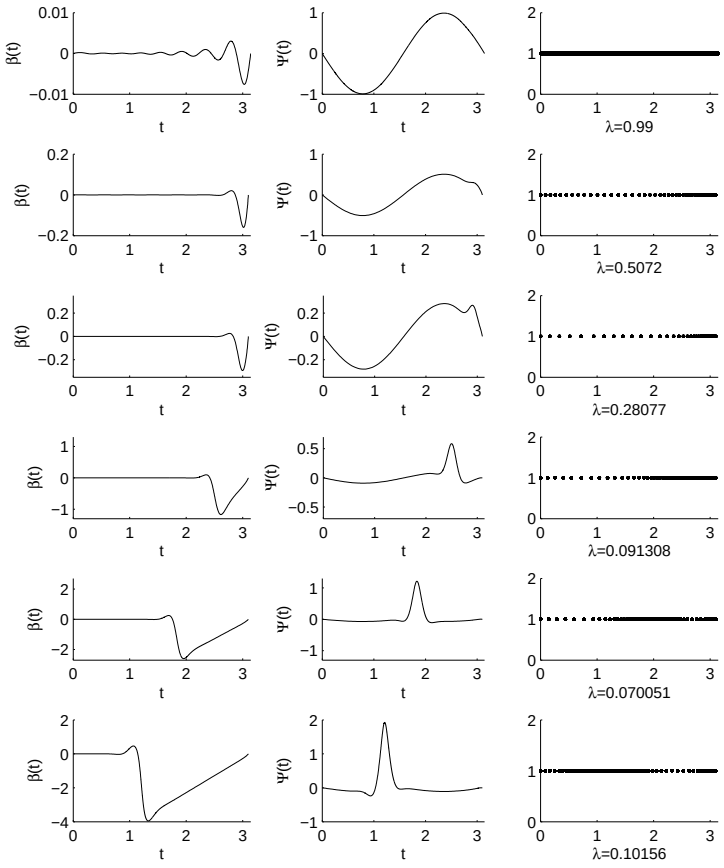

Figure 5: Solution profiles and automatically selected meshes at the points marked in Figure 4 along the solution branch.

- meshadaptation.m – runs the automatic grid control.
- initialmesh.m – provides the initial data for the Newton solver.
- pathfollowing.m – realizes the pathfollowing routine.
- settings.m – opens a window to set parameters.
- sbvpset.m – sets the options for the Newton solver.
- EVPmodule.m – carries out the reformulation of an EVP to a BVP.
- trafomodule.m – automatically transforms a problem posed on a semi-infinite interval $[a, \infty)$, $a \geq 0$ to a finite domain $[0, 1]$.
- backtransf.m – back-transforms the solution to the interval $[a, L] \subset [a, \infty)$, L large.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

- plot_results.m – provides graphical solution output.
- plotrange.m – defines settings for a solution plot on a subinterval $[a, L] \subset [a, \infty)$, L large.
- err.m – contains error messages.

More information on the code, input/output parameters, and GUI can be found in the manual [26].

7 Code Performance

In this section, we comment on the performance of our code `bvpsuite` when compared to other available software for the numerical solution of boundary value problems in ordinary differential equations. Since our focus is on singular boundary value problems, we have chosen those codes which explicitly claim that singular problems are in their scope. Therefore, we take into consideration the standard MATLAB code `bvp4c` [35] and the related solvers `bvp5c` [22], `bvp6c` [15], and two FORTRAN codes, BVP_SOLVER specified in [36] and COLNEW described in [1] and based on one of the best established BVP solvers COLSYS [2].

Although the objective of this paper has been to introduce our code and describe its scope and features, we shall have a brief look at how `bvpsuite` performs compared to existing codes. A full code comparison is a major investigation and is beyond the scope of this paper; we shall return to this issue in future work. Thus the test that follows only demonstrates feasibility on a single problem, and is not claimed to represent code comparison issues in full. Indeed, we were rather interested to see where there is a potential to improve the performance and efficiency of the `bvpsuite` package.

Our main intention while designing `bvpsuite` was to provide a MATLAB code which can cope with a wide range of applications and works dependably and efficiently for a large range of tolerances with emphasis on high-precision solution. Therefore, we have chosen the *fully implicit formulation* of the nonlinear system of equations and nonlinear boundary conditions, see Section 2. The order of the differential equations in the components of the system can be arbitrary and different for different components. Thus, there is no need to transform a higher order system to its first order form. The code can cope with free unknown parameters for which the appropriate number of additional boundary conditions are specified at the borders or within the interval of integration. In its scope are nonlinear singular boundary value problems with a singularity of the first or of the second kind[7]. Over the years, we have been able to give a good theoretical justification for all components of the code, also in the context of singular problems, see for instance the list of publications at `http://www.math.tuwien.ac.at/`$\sim$`ewa`.
The code can solve index-1 differential algebraic equations, a coupled system of differential equations and algebraic constraints. Also, it is equipped with a pathfollowing strategy in case of known parameter values such that the turning points in the solution/parameter path do not constitute a difficulty. Recently, we have equipped the code with modules for the solution of eigenvalue problems of first and second order, see the references below. Moreover, for a problem posed on a semi-infinite interval $[a, \infty)$, $a \geq 0$ the code automatically reduces the problem to the interval $[0, 1]$ and after numerical computations it provides the approximate solution transformed back to a suitable interval $[0, L]$, $L < \infty$, with L specified by the user.
The order of the collocation solver is chosen automatically in dependence of the tolerance specified by the user and varies between two and eight. We stress that since in our code Gaussian points (or equidistant interior collocation points) are used, we avoid the evaluation at the singular point and therefore also in the case of singular problems only *one numerical method on the whole interval* is used and no pre-handling is necessary. In other words, there is no distinction between the solution of singular or regular problems with `bvpsuite`. The error estimate and the grid adaptation routine have been described in Sections 3 and 4,

[7] and clearly, problems with no singularity

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

respectively

The code `bvp4c`[8] [34] can solve *explicit nonlinear systems of order one* with nonlinear boundary conditions and unknown parameters. However, the singular problems have to show a special structure,

$$z'(t) = \frac{S}{t} z(t) + f(t, z(t)), \quad t \in (0, T], \tag{28}$$

with a constant matrix S. This means that only a singularity of the first kind in this particular form is in the scope of the code. The basic solution method is based on polynomial collocation with four, five or six Lobatto points, respectively. Within one routine the order of the method is fixed to four, five or six. The quantity to be estimated and controlled is the residual, and residual and error in case of `bvp5c` [22].

The BVP_SOLVER [36] covers the same class of problems, regular and singular, as `bvp4c`. The methods used here are implicit Runge-Kutta schemes (MIRKDC) of orders two, four, and six. The code controls the defect in the differential equations and boundary conditions and also provides an estimate for the global error using the extrapolation technique.

Finally, COLNEW [1] can solve *explicit* nonlinear systems of ordinary differential equations of mixed order up to four. The basic solver is collocation based on Gaussian points whose number ranges from one to seven. The code controls the global error estimated from the mesh halving principle which in the case of Gaussian points is strongly related to the residual. In this code a pathfollowing strategy is also available and the code can cope with free parameters.

We compare the performance of the codes by solving the following boundary value problem:

$$z'(t) = \frac{1}{t} \begin{pmatrix} 0 & 1 \\ 2 & 6 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ 4k^2 t^5 \sin(k^2 t^2) + 10t \sin(k^2 t^2) \end{pmatrix}, \tag{29}$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ \sin(k^2) \end{pmatrix}, \tag{30}$$

where the analytical solution is known,

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

We have used all the codes with fixed orders four and six and a variable order version of `bvpsuite`, that allows the order to be selected automatically depending on TOL, with order varying from 2 to 8, see curves 'bvpsuite vo'. All codes have been run for the same tolerance settings and the same number of points in the initial mesh. The results show that our approach provides the most efficient solution method. Therefore the flexibility of our code also constitutes a significant improvement of the performance. The model problem (29)–(30) discussed here, gives a typical picture observed in many tests.

We investigate the following parameters. First, we check the total number of function calls (Figure 6), then the number of grid points on the final computational grid (Figure 7), and finally the CPU time (Figure 8). When the order is fixed to four for comparison, the number of grid points required by `bvpsuite` and COLNEW is comparable and smaller throughout than for `bvp4c` and BVP_SOLVER. Especially, for strict tolerances, the gap between `bvpsuite` and COLNEW and the other two codes is significant. The strictest tolerance successfully reached by the BVP_SOLVER was 10^{-10}, and by `bvp4c` 10^{-11}, while `bvpsuite` and COLNEW reached an accuracy of 10^{-13}. The test also shows that for a wide range of tolerances, `bvpsuite` with the variable order option produces grids with the fewest points. The number of function calls in `bvpsuite` can still be reduced, as the global error estimate currently implemented is a simple approach using an extra grid with twice as many points as the actual grid; this will be modified to a strategy

[8] In the following, we refer to `bvp4c` only, even when all three variants of the code are addressed including `bvp5c` [22] and `bvp6c` [15].

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

where the global error is estimated on the actual grid. We expect this to save up to another factor of 2 in CPU time.

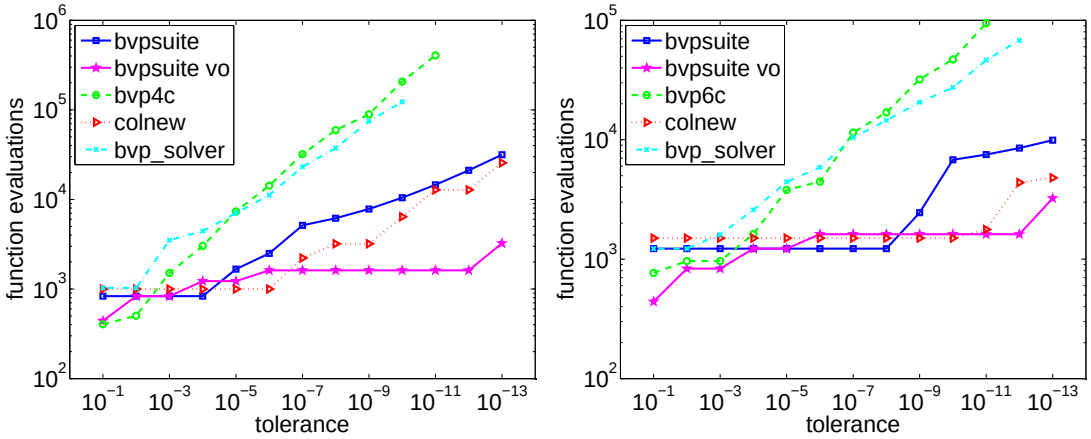

Figure 6: Problem (29)–(30), $k = 5$: Total number of function calls for the method of order four (left) and order six (right) plotted as a function of TOL.

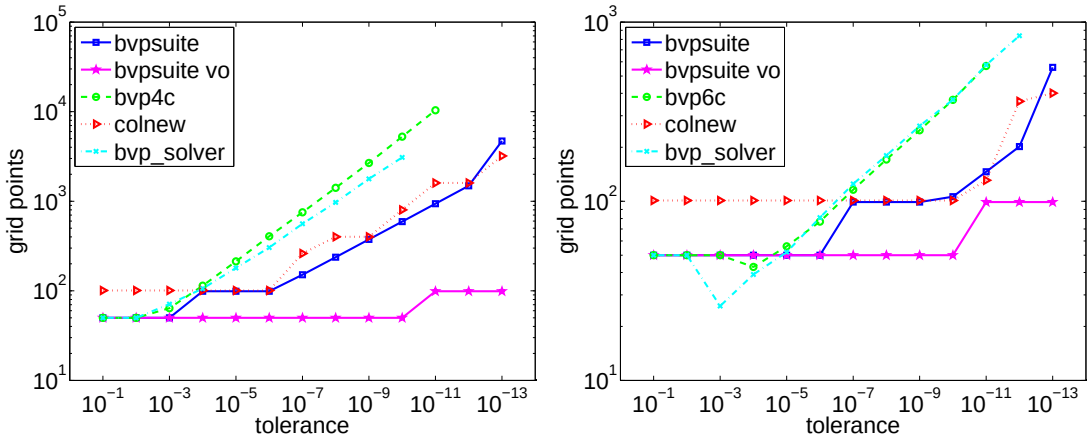

Figure 7: Problem (29)–(30), $k = 5$. Number of grid points used by methods of order four (left) and six (right) plotted as functions of TOL. Note that the variable order `bvpsuite` solves the problem on the $N_0 = 50$ adaptive control grid for all except the strictest tolerances. The control algorithms implemented in the code require the least number of grid points over a wide range of tolerances.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

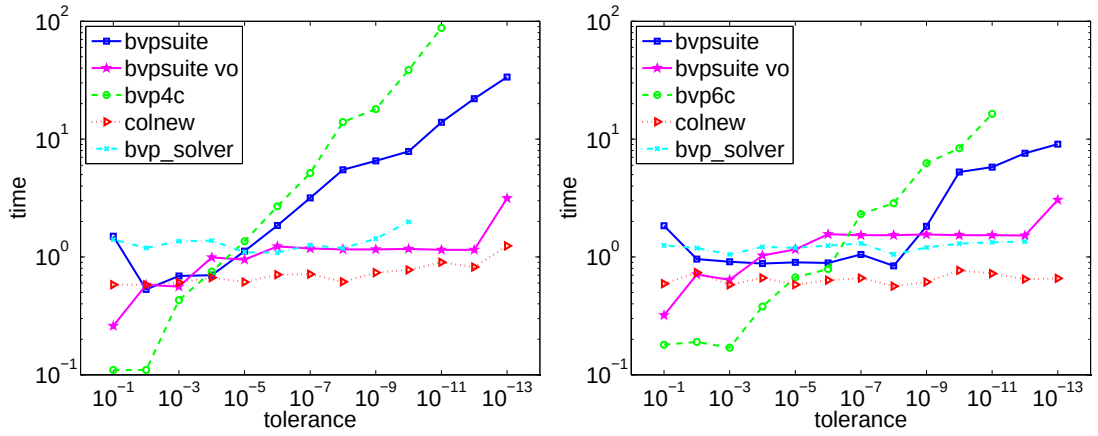

Figure 8: Problem (29)–(30), $k = 5$. CPU time used by methods of order four (left) and six (right) plotted as functions of TOL.

Finally, we also investigate the actual error produced by each code (Figure 9). The latter test suggests that `bvpsuite`'s automatic order selection can be further improved to enhance the code's performance. It is worth mentioning that the BVP_SOLVER and `bvp4c` work for this example very well at target, which means that the required tolerance and the achieved accuracy are closely related.

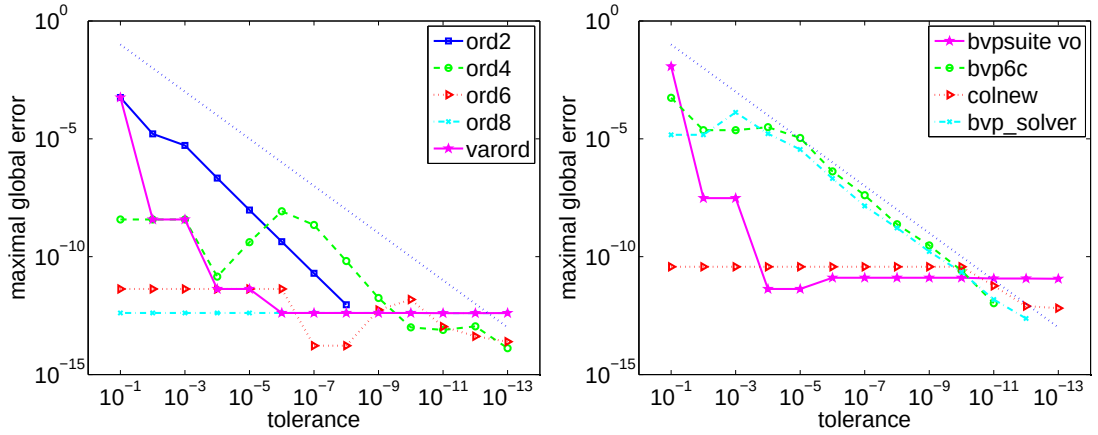

Figure 9: Problem (29)–(30), $k = 5$. Achieved global error for orders two, four, six, eight and variable order in `bvpsuite` (left) and comparison between methods of order six and the variable order `bvpsuite` (right) plotted as functions of TOL. The graphs indicate that with the new grid generation algorithm, `bvpsuite`'s order selection strategy – not studied yet in detail – may be too conservative, and that further work on order selection has a potential for enhancing code efficiency.

8 Applications

As already mentioned, eigenvalue problems [4], [37] and differential algebraic equations [9], [28], are within the scope of our code, but it can also be applied in case of non-standard singularities. In [32], we

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

investigated the following singular equation which originates from the theory of shallow membrane caps,

$$(t^3u'(t))' + t^3\Big(\frac{1}{8u^2(t)} - \frac{a_0}{u(t)} - b_0t^{2\gamma-4}\Big) = 0, \quad t \in (0,1], \tag{31}$$

subject to asymptotic boundary conditions

$$\lim_{t\to 0+} t^3u'(t) = 0, \quad u(1) = 0,$$

where a_0, b_0, and γ are given constants. Note that this problem has a more challenging structure than (1)–(2). After rewriting (31), we obtain the explicit version of the equation,

$$u''(t) + \frac{3}{t}\,u'(t) + \Big(\frac{1}{8u^2(t)} - \frac{a_0}{u(t)} - b_0t^{2\gamma-4}\Big) = 0, \quad u(1) = 0. \tag{32}$$

Here, a singularity of the first kind occurs at $t = 0$, but at the same time due to the boundary condition at $t = 1$ the problem has a so-called *phase singularity* at the other end of the interval. For such more involved problems existence and uniqueness of solutions is shown by means of generalized lower and upper functions, involving limiting processes, cf. [32] and references therein. Our code `bvpsuite` could be used to approximate solutions[9] of the membrane problem. However, a theoretical justification for the collocation method in view of the problem structure is still an open question.

Another source of challenging problems with an interesting solution structure are reaction-diffusion equations, see [38], [39]. In [38], the simple looking, parameter dependent problem of the form

$$u''(t) = \frac{\lambda}{\sqrt{u(t)}}, \quad t \in (0,1], \quad u(0) = u(1) = 1, \tag{33}$$

where λ is a given parameter, turns out to have a very challenging structure. Depending on the value of λ there exist the so-called positive solutions, $u(t) > 0$ for all $t \in [0,1]$, pseudo dead core, and dead core solutions, such that $u(t) = 0$ for a certain point $t \in (0,1)$, or $u(t) = 0$ on a certain subinterval $t \in [\alpha,\beta], 0 < \alpha < \beta < 1$, respectively. In order to find the latter two solutions, we simulated the problem numerically using `bvpsuite`. Here, we utilized the fact that the above equation can be treated in its fully implicit form,

$$u''(t)\sqrt{u(t)}u(t) = \lambda u(t), \quad t \in (0,1], \quad u(0) = u(1) = 1. \tag{34}$$

Clearly, in cases where the analytical problem is especially involved, the numerical approach may sometimes constitute the only source of information about the solution structure. We faced this type of difficulty in [39]. Since the problem is again parameter dependent, we applied the pathfollowing strategy implemented in `bvpsuite` to solve

$$((u'(t))^3)' + \frac{u'(t)}{t^2} = \lambda\left(\frac{1}{\sqrt{u(t)}} + (u'(t))^2\right), \quad t \in (0,1), \tag{35}$$

$$u'(0) = 0, \quad 0.1u(1) + u'(1) = 1. \tag{36}$$

The results of this simulation are shown in Figure 12. We can see that for a certain range of λ the positive solution is unique, and for the other part of the path, we could find two different positive solutions, see Figures 10 and 11. According to Figure 12, we have moved around a turning point at $\lambda \approx 1.8442$.

[9] even though $u'(0)$ may become unbounded

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

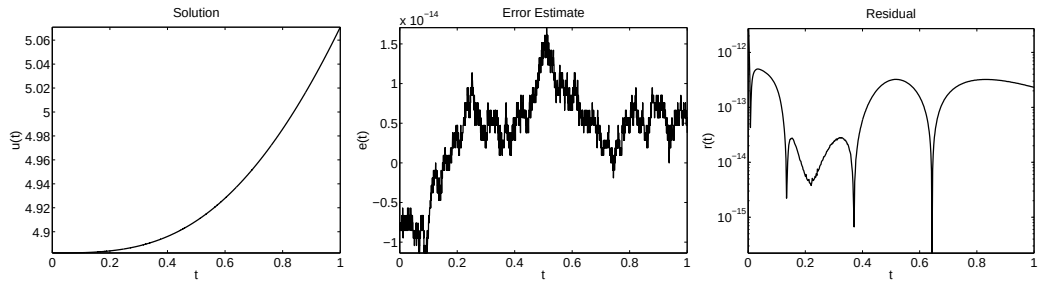

Figure 10: Problem (35)–(36): The numerical solution, the error estimate and the residual for $\lambda = 1.42604644036221$.

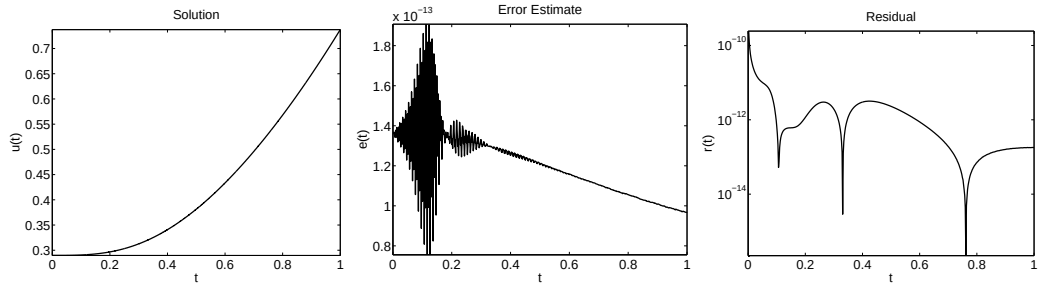

Figure 11: Problem (35)–(36): The numerical solution, the error estimate and the residual for $\lambda = 1.42139222684689$.

Finally, in the last step of the procedure, we obtained a solution which nearly reaches a pseudo dead core solution with the collocation solution $p(0) \approx u(0) \approx 0$.

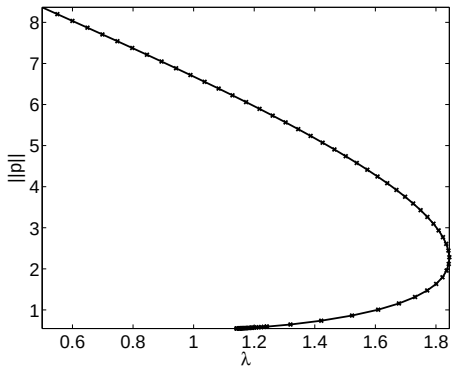

Figure 12: Graph of the $\|p\|/\lambda$ path obtained in 76 steps of the pathfollowing procedure, where $\|p\| = \max_{t\in[0,1]} |p(t)|$. The turning point has been determined as $\lambda \approx 1.8442$.

Finally, we present a boundary value problem which originates from a theory for the explosive crystallization of thin amorphous layers on a substrate [10], [29], [30]. The speed, form and temperature distri-

© *2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)*

bution of a crystallization front propagating through a thin layer of amorphous material on a substrate is calculated. While the simplified formulation treated here is useful as a demonstration of a problem behavior, the ultimate aim is to solve a more complicated problem including heat loss into the substrate, and the influence of said heat loss on the crystallization process. The original problem posed on a semi-infinite interval $\tau \in [0, \infty)$ has been transformed to a finite interval $t \in [0, 1]$ by means of the following transformation [10]:

$$t = 1 - \frac{1}{\sqrt{1+\tau}}.$$

The resulting boundary value problem for a system of two equations, for the temperature distribution $\Theta(t)$ and the the crystallization $\xi(t)$, $t \in [0, 1)$, reads,

$$\Theta'(t) = 2\frac{\Theta(t) - \xi(t)}{(1-t)^3}, \tag{37}$$

$$\xi'(t) = 2\frac{\lambda^2 e^{-3/\Theta(t)}(1-\xi(t))\left(\frac{-3}{\sqrt{2}}\ln(1-\xi(t))\right)^{2/3}}{(1-t)^3}, \tag{38}$$

$$\Theta(0) = 0.1284, \quad \Theta(1) = 1. \tag{39}$$

Due to the above transformation an essential singularity occurs at $t = 1$. In this case λ is an unknown parameter related to the speed of the crystallization front. The third condition necessary to close the problem reflects the fact that at the beginning of the process tiny crystals may already exist in the material, and therefore $\xi(0)$ is very small. For the calculations, we used $\xi(0) = 10^{-10}$. The results of the numerical experiment are shown in Figures 13 and 14. It can be seen from Figure 14 that the rather strict tolerances have been satisfied on a final mesh with 198 subintervals (199 mesh points). Note that due to strict tolerance requirement, meshes are rather dense on the whole domain, but points accumulate in the region where the solution varies strongly. The accumulation of points near $t = 1$ can be attributed to our deforming transformation of the independent variable. In the original variable, grid points thin out for large τ.

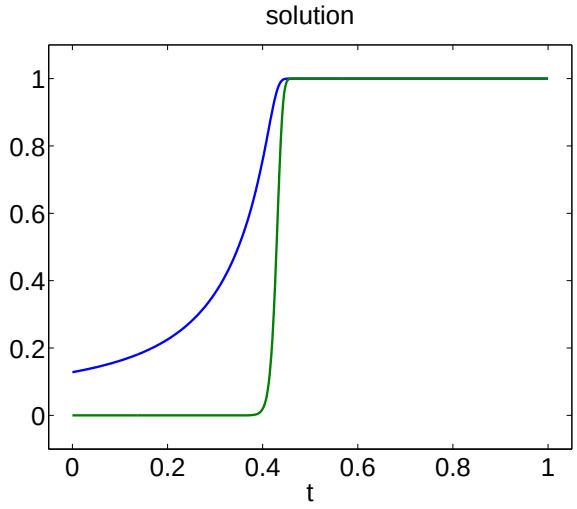

Figure 13: Problem (37)–(38): Graph of the solution components $\Theta(t)$ (blue) and $\xi(t)$ (green) obtained from `bvpsuite` using collocation with $m = 8$ Gaussian points and $\mathrm{TOL}_a = \mathrm{TOL}_r = 10^{-12}$. Here, $\lambda = 11.03605$.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

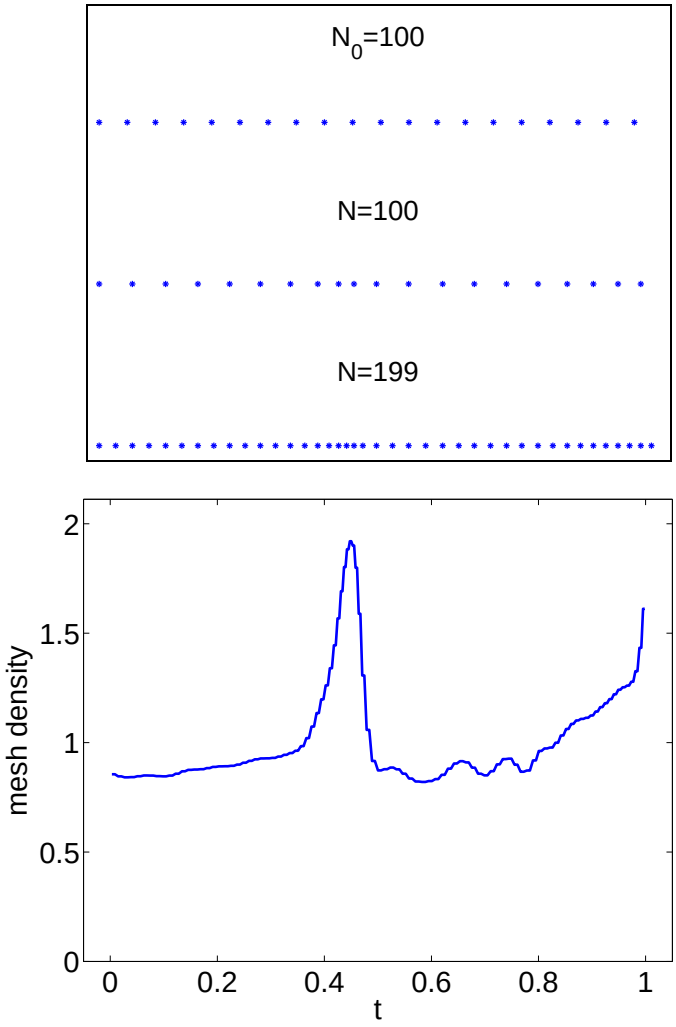

Figure 14: Problem (37)–(38): Steps of the grid adaption procedure carried out for collocation with $m = 8$ Gaussian points and $\mathrm{TOL}_a = \mathrm{TOL}_r = 10^{-12}$ (top) and the mesh density function in the final mesh (bottom). In the meshes shown left only every fifth mesh point is depicted in order to better visualize its location.

9 Conclusions

In this paper we gave an overview of the very intense activities carried out for many years at Vienna University of Technology and focused on the analysis, numerical solution and code development for singular boundary value problems in ordinary differential equations, differential algebraic equations, and problems posed on semi-infinite intervals.

When analyzing singular problems, we first note that their direction field is very unsmooth, especially close to the singular point. Consequently, we can encounter unbounded contributions to the solution manifold, such that $z \in C(0,1]$. However, irrespective of the spectrum of the matrix $M(0)$, by posing proper homogeneous initial conditions, we can extend the above solution to $z \in C[0,1]$. It also turns out that in

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

such a case the condition $M(0)z(0) = 0$ must hold. For singular problems the solution's smoothness depends not only on the smoothness of the inhomogeneity f but also on the size of real parts of the eigenvalues of $M(0)$.

Concerning the numerical treatment of singular problems one usually assumes that the underlying analytical problem is well-posed and has a smooth solution. On the basis of such an assumption, one would like to design a high order method, and error estimate and grid adaptation strategies, which remain unaffected by the steep direction field. This means that the grids should become dense only in the regions where the solution is unsmooth. Especially they should stay coarse close to the singularity when the solution is smooth there. It turns out that collocation at Gaussian (or inner equidistant) points remains robust for singular problems and can serve as a dependable solver in the code design, while other high order methods suffer from order reductions. Also, defect correction and mesh halving principles constitute a reliable basis for the a posteriori error estimation. We have put a lot of effort in the grid adaptation strategy. Here, the main idea is to split the adaptation of the grid density and the number of grid points necessary to satisfy the tolerance requirements. This idea has proven to be very fruitful and results in grids which in a very satisfactory way reflect the solution behavior.

Finally, we introduced and described in detail our new MATLAB solver `bvpsuite` and demonstrated that concerning the scope and efficiency it is a very competitive candidate among the available software for singular boundary value problems.

References

[1] U. Ascher and U. Bader. A new basis implementation for a mixed order boundary value ODE solver. *SIAM J. Scient. Stat. Comput.*, 8:483–500, 1987.

[2] U. Ascher, J. Christiansen, and R.D. Russell. A collocation solver for mixed order systems of boundary values problems. *Math. Comp.*, 33:659–679, 1978.

[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, NJ, 1988.

[4] W. Auzinger, E. Karner, O. Koch, and E.B. Weinmüller. Collocation methods for the solution of eigenvalue problems for singular ordinary differential equations. *Opuscula Math.*, 26(2):229–241, 2006.

[5] 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.

[6] W. Auzinger, O. Koch, D. Praetorius, and E.B. Weinmüller. New a posteriori error estimates for singular boundary value problems. *Numer. Algorithms*, 40:79–100, 2005.

[7] W. Auzinger, O. Koch, and E.B. Weinmüller. Collocation methods for boundary value problems with an essential singularity. In I. Lirkov, S. Margenov, J. Wasniewski, and P. Yalamov, editors, *Large-Scale Scientific Computing*, volume 2907 of *Lecture Notes in Computer Science*, pages 347–354. Springer Verlag, 2004.

[8] W. Auzinger, O. Koch, and E.B. Weinmüller. Analysis of a new error estimate for collocation methods applied to singular boundary value problems. *SIAM J. Numer. Anal.*, 42(6):2366–2386, 2005.

[9] W. Auzinger, H. Lehner, and E.B. Weinmüller. Defect-based a posteriori error estimation for index-1 DAEs. Submitted to *BIT*.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)

[10] Ch. Buchner and W. Schneider. Explosive crystallization in thin amorphous layers on heat conducting substrates. In *Proceedings of the International Heat Transfer Conference*, 2010, ISBN: 978-0-7918-3879-2.

[11] 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.

[12] C. J. Budd, O. Koch, and E.B. Weinmüller. From nonlinear PDEs to singular ODEs. *Appl. Numer. Math.*, 56:413–422, 2006.

[13] J. Cash, G. Kitzhofer, O. Koch, G. Moore, and E.B. Weinmüller. Numerical solution of singular two-point BVPs. *JNAIAM J. Numer. Anal. Indust. Appl. Math.*, 4:129–149, 2009.

[14] M. Gräff, R. Scheidl, H. Troger, and E.B. Weinmüller. An investigation of the complete post-buckling behavior of axisymmetric spherical shells. *ZAMP*, 36:803–821, 1985.

[15] N. Hale and D. Moore. A Sixth-Order Extension to the MATLAB `bvp4c` of J. Kierzenka and L. Shampine. Techn. Rept. No. NA-08/04, Oxford University Computing Laboratory, Oxford, United Kingdom, 2008. Available at `http://web.comlab.ox.ac.uk//files/720/NA-08-04.pdf`.

[16] F.R. 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.

[17] F.R. de Hoog and R. Weiss. Collocation methods for singular boundary value problems. *SIAM J. Numer. Anal.*, 15:198–217, 1978.

[18] F.R. de Hoog and R. Weiss. The numerical solution of boundary value problems with an essential singularity. *SIAM J. Numer. Anal.*, 16:637–669, 1979.

[19] F.R. de Hoog and R. Weiss. On the boundary value problem for systems of ordinary differential equations with a singularity of the second kind. *SIAM J. Math. Anal.*, 11:41–60, 1980.

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

[21] H. Keller. Approximation methods for nonlinear problems with application to two-point boundary value problems. *Math. Comp.*, 29:464–474, 1975.

[22] J. Kierzenka and L. Shampine. A BVP solver that controls residual and error. *JNAIAM J. Numer. Anal. Indust. Appl. Math.*, 3:27–41, 2008.

[23] G. Kitzhofer. *Numerical Treatment of Implicit Singular BVPs*. Ph.D. Thesis, Inst. for Anal. and Sci. Comput., Vienna Univ. of Technology, Austria. In preparation.

[24] G. Kitzhofer, O. Koch, P. Lima, and E.B. Weinmüller. Efficient numerical solution of the density profile equation in hydrodynamics. *J. Sci. Comput.*, 32:411–424, 2007.

[25] G. Kitzhofer, O. Koch, and E.B. Weinmüller. Pathfollowing for essentially singular boundary value problems with application to the complex Ginzburg–Landau equation. *BIT Numerical Mathematics*, 49:141, 2009.

[26] G. Kitzhofer, G. Pulverer, O. Koch, Ch. Simon, and E.B. Weinmüller. *BVPSUITE – A New MATLAB Code for Singular Implicit Boundary Value Problems*, 2009. Available at `http://www.math.tuwien.ac.at/~ewa`.

© *2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)*

[27] O. Koch. Asymptotically correct error estimation for collocation methods applied to singular boundary value problems. *Numer. Math.*, 101:143–164, 2005.

[28] O. Koch, R. März, D. Praetorius, and E.B. Weinmüller. Collocation methods for index-1 DAEs with a singularity of the first kind. *Math. Comp.*, 79:129–149, 2009.

[29] A. Köppl. *Anwendung von Ratengleichungen auf anisotherme Kristallisation von Kunststoffen*. Ph. D. Thesis, Vienna Univ. of Technology, Austria, 1990.

[30] A. Köppl, J. Berger, and W. Schneider. Ausbreitungsgeschwindigkeit und Struktur von Kristallisationswellen. In *Proceedings of GAMM*, Stuttgard, Germany, 1987.

[31] G. Pulverer, G. Söderlind, and E.B. Weinmüller. Automatic grid control in adaptive BVP solvers. Accepted for Numer. Algorithms.

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

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

[34] L. Shampine and J. Kierzenka. A BVP solver based on residual control and the MATLAB PSE. *ACM Trans. Math. Software*, 27:299–315, 2001.

[35] 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/`.

[36] L. Shampine, P. Muir, and H. Xu. A User-Friendly Fortran BVP Solver, *JNAIAM J. Numer. Anal. Indust. Appl. Math.*, 1:201–217, 2006.

[37] Ch. Simon. *Numerical Solution of Singular Eigenvalue Value Problems for Systems of ODEs with a Focus on Problems Posed on Semi-Infinite Intervals*. Master's thesis, Vienna Univ. of Technology, Vienna, Austria, 2009.

[38] S. Staněk, G. Pulverer, and E.B. Weinmüller. Analysis and numerical solution of positive and dead core solutions of singular two-point boundary value problems. *Comp. Math. Appl.*, 56:1820-1837, 2008.

[39] S. Staněk, G. Pulverer, and E.B. Weinmüller. Analysis and numerical solution of positive and dead core solution of singular Sturm-Liouville problems. *Adv. Difference Equ.*, Volume 2010 (2010), Article ID 969536, 37 pages, doi:10.1155/2010/969536.

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

[41] R. Winkler. Path-following for two-point boundary value problems. Tech. Rept. 78, Department of Mathematics, Humboldt-University Berlin, Germany, 1985.

© 2010 European Society of Computational Methods in Sciences and Engineering (ESCMSE)