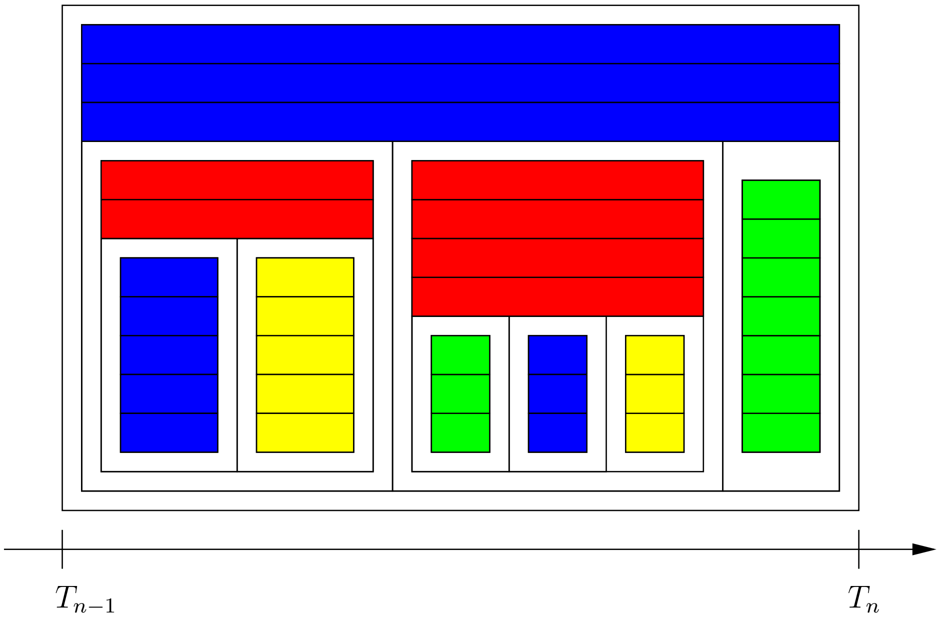

PREPRINT 2004–12

Multi-adaptive Galerkin methods for ODEs V: Stiff problems

Johan Jansson and Anders Logg

Chalmers Finite Element Center
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg Sweden 2004

CHALMERS FINITE ELEMENT CENTER

Preprint 2004–12

Multi-adaptive Galerkin methods for ODEs V: Stiff problems

Johan Jansson and Anders Logg

CHALMERS

Chalmers Finite Element Center
Chalmers University of Technology
SE–412 96 Göteborg Sweden
Göteborg, April 2004

Multi-adaptive Galerkin methods for ODEs V:
Stiff problems
Johan Jansson and Anders Logg
NO 2004–12
ISSN 1404–4382

Chalmers Finite Element Center
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone: +46 (0)31 772 1000
Fax: +46 (0)31 772 3595
`www.phi.chalmers.se`

Printed in Sweden
Chalmers University of Technology
Göteborg, Sweden 2004

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES V: STIFF PROBLEMS

JOHAN JANSSON AND ANDERS LOGG

Abstract. We develop the methodology of multi-adaptive time-stepping for stiff problems. The new algorithm is based on adaptively stabilized fixed point iteration on time slabs and a new method for the recursive construction of time slabs. Numerical examples are given for a series of well-known stiff and non-stiff test problems.

1. Introduction

This is part V in a sequence of papers [7, 8, 9, 10] on multi-adaptive Galerkin methods, mcG(q) and mdG(q), for approximate (numerical) solution of ODEs of the form

$$\begin{aligned} \dot{u}(t) &= f(u(t), t), \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{1.1}$$

where $u : [0, T] \to \mathbb{R}^N$ is the solution to be computed, $u_0 \in \mathbb{R}^N$ a given initial value, $T > 0$ a given final time, and $f : \mathbb{R}^N \times (0, T] \to \mathbb{R}^N$ a given function that is Lipschitz-continuous in u and bounded.

The mcG(q) and mdG(q) methods are based on piecewise polynomial approximation of degree q on partitions in time with time steps which may vary for different components $U_i(t)$ of the approximate solution $U(t)$ of (1.1). In part I and II of our series on multi-adaptive Galerkin methods, we prove a posteriori error estimates, through which the time steps are adaptively determined from residual feed-back and stability information, obtained by solving a dual linearized problem. In part III, we prove existence and stability of discrete solutions, which are used in part IV to prove a priori error estimates. In the current paper, we develop the methodology of multi-adaptive time-stepping for stiff problems.

1.1. **The stiffness problem.** As noted already by Dahlquist [1] in the 1950s, there is a certain class of problems, so-called *stiff problems*, for which standard explicit methods are not suitable. This is often referred to as the *stiffness problem*. As noted in [3], we run into the same difficulties when we try to solve the system of equations given by an implicit method using direct fixed point iteration. Within the setting of multi-adaptive Galerkin

Date: April 13, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, explicit, stiff problems.

Johan Jansson, *email*: johanjan@math.chalmers.se. Anders Logg, *email*: logg@math.chalmers.se. Department of Computational Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden.

methods, this becomes evident when the adaptively determined time steps become too large for the fixed point iteration to converge, which typically happens outside transients. We are thus forced to take (much) smaller time steps than required to meet the given error tolerance.

In [3], we present a new methodology for the stabilization of explicit methods for stiff problems, based on the inherent property of the stiff problem itself: rapid damping of high frequencies. Using sequences of stabilizing time steps, consisting of alternating small and large time steps, an efficient explicit method is obtained.

In the current paper, we extend the ideas presented in [3] for the mono-adaptive cG(1) method to general multi-adaptive time stepping. In particular, we show that the technique of stabilizing time step sequences can be extended to adaptively stabilized fixed point iteration on time slabs, where the damping factor α plays the role of the small stabilizing time steps.

1.2. **Implementation.** The presented methodology has been implemented in **DOLFIN** [5], the C++ implementation of the new open-source software project **FEniCS** [2] for the automation of Computational Mathematical Modeling (CMM). The multi-adaptive solver in **DOLFIN** is based on the original implementation *Tanganyika*, presented in [8], but has been completely rewritten for **DOLFIN**. The new implementation is discussed in detail in [6].

1.3. **Notation.** For a detailed description of the multi-adaptive Galerkin methods, we refer the reader to [7, 8, 9, 10]. In particular, we refer to [7] or [9] for the definition of the methods.

The following notation is used throughout this paper: Each component $U_i(t)$, $i = 1, \ldots, N$, of the approximate m(c/d)G(q) solution $U(t)$ of (1.1) is a piecewise polynomial on a partition of $(0, T]$ into M_i subintervals. Subinterval j for component i is denoted by $I_{ij} = (t_{i,j-1}, t_{ij}]$, and the length of the subinterval is given by the local *time step* $k_{ij} = t_{ij} - t_{i,j-1}$. This is illustrated in Figure 1. On each subinterval I_{ij}, $U_i|_{I_{ij}}$ is a polynomial of degree q_{ij} and we refer to $(I_{ij}, U_i|_{I_{ij}})$ as an *element*.

Furthermore, we shall assume that the interval $(0, T]$ is partitioned into blocks between certain synchronized time levels $0 = T_0 < T_1 < \ldots < T_M = T$. We refer to the set of intervals $\mathcal{T}_n$ between two synchronized time levels T_{n-1} and T_n as a *time slab*:

$$\mathcal{T}_n = \{I_{ij} : T_{n-1} \leq t_{i,j-1} < t_{ij} \leq T_n\}.$$

We denote the length of a time slab by $K_n = T_n - T_{n-1}$.

1.4. **Outline of the paper.** We first discuss a few basic and well-known properties of fixed point iteration in Section 2, and then present our new methodology of adaptively stabilized fixed point iteration in Section 3. In Section 4, we then discuss two important parts of the multi-adaptive algorithm: the construction of multi-adaptive time slabs, and the adaptively stabilized fixed point iteration on time slabs. Finally, in Section 5, we present numerical results for a sequence of stiff test problems taken from [3].

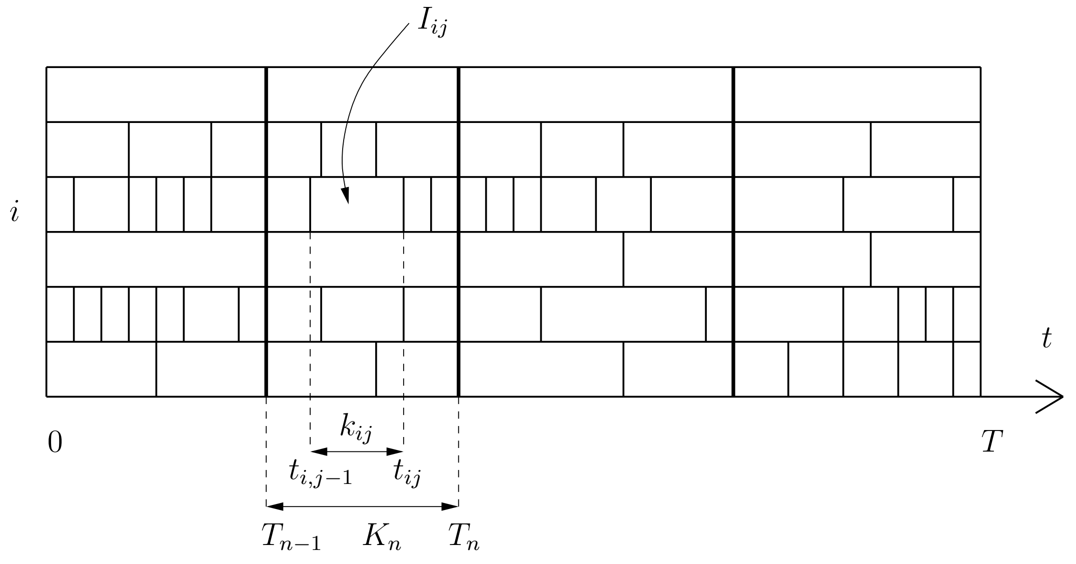

Figure 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

2. Fixed point iteration

Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be a given differentiable function of the form

$$F(x) \equiv x - g(x) \tag{2.1}$$

and consider the problem of solving the equation

$$F(x) = 0 \tag{2.2}$$

or, alternatively, $x = g(x)$ by fixed point iteration. Given an initial guess x^0, we iterate according to

$$x^n = g(x^{n-1}) = x^{n-1} - (x^{n-1} - g(x^{n-1})) = x^{n-1} - F(x^{n-1}), \tag{2.3}$$

for $n = 1, 2, \ldots$, to obtain the fixed point x satisfying $x = g(x)$. By the Banach fixed point theorem, this iteration converges to the unique solution x of (2.2), if the Lipschitz constant L_g of g satisfies

$$L_g < 1, \tag{2.4}$$

or, since the Lipschitz constant is bounded by the derivative of g, if $\|g'\| \leq C$ with $C < 1$ for a suitable norm $\|\cdot\|$. To see this, we note that $F(x^n) = x^n - g(x^n) = g(x^{n-1}) - g(x^n) = g'(\xi)(x^{n-1} - x^n)$, and thus, by (2.3),

$$F(x^n) = g'(\xi)F(x^{n-1}), \tag{2.5}$$

and so the *residual* $F(x^n)$ of (2.2) converges to zero if $\|g'\|$ is bounded by $C < 1$.

For the *increment* $d^n \equiv x^n - x^{n-1}$, we similarly obtain $d^n = x^n - x^{n-1} = g(x^{n-1}) - g(x^{n-2}) = g'(\xi)(x^{n-1} - x^{n-2})$, and thus

$$d^n = g'(\xi)d^{n-1}. \tag{2.6}$$

We finally note that for the *error* $e^n \equiv x^n - x$, with x the solution of (2.2), we obtain $e^n = x^n - x = g(x^{n-1}) - g(x) = g'(\xi)(x^{n-1} - x)$, and thus

$$e^n = g'(\xi)e^{n-1}. \tag{2.7}$$

By (2.5), (2.6), and (2.7), it now follows that we can measure either the residual $F(x^n)$ or the increment d^n to determine the convergence of the error e^n. If the solution does not converge, the fixed point iteration needs to be stabilized. In the next section, we present an algorithm for adaptively stabilized fixed point iteration, based on measuring the convergence of the residual $F(x^n)$ or the increment d^n.

3. Adaptive fixed point iteration

To stabilize the fixed point iteration, we modify the basic iteration (2.3) according to

$$x^n = (I - \alpha)x^{n-1} + \alpha g(x^{n-1}) = x^{n-1} - \alpha(x^{n-1} - g(x^{n-1}) = x^{n-1} - \alpha F(x^{n-1}), \tag{3.1}$$

where I is the $N \times N$ identity matrix and the *damping factor* α is an $N \times N$ matrix to be determined. We will mainly consider the case of a diagonal or scalar α. Note that (2.3) is recovered for $\alpha = I$.

To show the equivalent of (2.5), we write $F(x^n) = x^n - g(x^n)$ in the form $F(x^n) = (x^n - x^{n-1}) + (x^{n-1} - g(x^{n-1})) + (g(x^{n-1}) - g(x^n))$. It now follows by (3.1) that $F(x^n) = -\alpha F(x^{n-1}) + F(x^{n-1}) + g'(\xi)(x^{n-1} - x^n) = (I - \alpha)F(x^{n-1}) + g'(\xi)\alpha F(x^{n-1})$, and thus

$$F(x^n) = \left[I - (I - g'(\xi))\alpha\right] F(x^{n-1}). \tag{3.2}$$

Similarly, we obtain $d^n = x^n - x^{n-1} = (I - \alpha)x^{n-1} + \alpha g(x^{n-1}) - (I - \alpha)x^{n-2} - \alpha g(x^{n-2}) = (I - \alpha)d^{n-1} + \alpha g'(\xi)(x^{n-1} - x^{n-2})$, and thus

$$d^n = \left[I - \alpha(I - g'(\xi))\right] d^{n-1}. \tag{3.3}$$

We also note that $x = (I - \alpha)x + \alpha g(x)$ if $x = g(x)$, and thus the error e^n satisfies $e^n = x^n - x = (I - \alpha)x^{n-1} + \alpha g(x^{n-1}) - (I - \alpha)x - \alpha g(x) = (I - \alpha)e^{n-1} + \alpha g'(\xi)(x^{n-1} - x)$, i.e.,

$$e^n = \left[I - \alpha(I - g'(\xi))\right] e^{n-1}. \tag{3.4}$$

The question is now how to choose the damping factor α. One obvious choice is to take α such that $I - \alpha(I - g'(x^{n-1})) = 0$, where we have replaced the unknown intermediate value ξ with the latest known value x^{n-1}. This gives $\alpha = (I - g'(x^{n-1}))^{-1} = (F'(x^{n-1}))^{-1}$, and thus

$$x^n = x^{n-1} - (F'(x^{n-1}))^{-1}F(x^{n-1}), \tag{3.5}$$

which is *Newton's method* for the solution of (2.2).

We now present an algorithm for stabilized fixed point iteration which adaptively determines the damping factor α, and which avoids computing the Jacobian F' and solving a linear system in each iteration as in Newton's method. We focus on the case where α is either *diagonal* or *scalar*.

3.1. **Diagonal damping.** Let $\alpha = \mathrm{diag}(\alpha_1, \ldots, \alpha_N)$ be a diagonal matrix, and assume for simplicity that g' is constant and equal to $-B$. With this choice of α, the fixed point iteration is given by

$$x_i^n = (1-\alpha_i)x_i^{n-1} + \alpha_i g_i(x^{n-1}), \quad i = 1, \ldots, N, \tag{3.6}$$

or $x^n = G_\alpha x^{n-1}$, with $G_\alpha = I - \alpha(I+B)$. We assume that

(1) B is diagonally dominant, and
(2) $B_{ii} \geq 0, \quad i = 1, \ldots, N$.

By (3.4), it follows that the fixed point iteration converges for $\|G_\alpha\|_{l_\infty} < 1$, where $\|G_\alpha\|_{l_\infty}$ denotes the maximum absolute row sum, $\|G_\alpha\|_{l_\infty} = \max_i \sum_{j=1}^N |(G_\alpha)_{ij}|$. For $i = 1, \ldots, N$, we have

$$\sum_{j=1}^N |(G_\alpha)_{ij}| = |1 - \alpha_i - \alpha_i B_{ii}| + \alpha_i \sum_{j \neq i} |B_{ij}| \leq |1 - \alpha - \alpha B_{ii}| + \alpha_i B_{ii},$$

since B is diagonally dominant and $B_{ii} \geq 0$. We now take

$$\alpha_i = 1/(1 + B_{ii}), \tag{3.7}$$

which gives $\sum_{j=1}^N |(G_\alpha)_{ij}| = B_{ii}/(1+B_{ii}) < 1$. We thus conclude that if B is diagonally dominant and $B_{ii} \geq 0$ for each i, then we can choose α diagonal such that the stabilized fixed point iteration (3.1) converges. We also note that the convergence may be slow if some $B_{ii} \gg 1$, since then $B_{ii}/(1+B_{ii})$ is close to unity.

3.2. **Scalar damping.** With $\alpha \in (0,1]$ scalar, we assume as above that $g' = -B$ is constant. We further assume that

(1) B is non-defective, i.e., B is diagonalizable:
$$\exists\, V = \begin{bmatrix} v^1 \cdots v^N \end{bmatrix} \text{ non-singular} : V^{-1}BV = \mathrm{diag}(\lambda_1, \ldots, \lambda_N),$$
with $\lambda_1, \ldots, \lambda_N$ the eigenvalues of B;
(2) $\mathrm{Re}\,\lambda_i > -1$ for all λ_i;
(3) $|\mathrm{Im}\,\lambda_i|/(1 + \mathrm{Re}\,\lambda_i) \leq \tan\beta$, for some $\beta \in (0, \pi/2)$, i.e., $|\arg(1+\lambda_i)| \leq \beta$ for all λ_i, as illustrated in Figure 2.

Note that the first condition is not a major restriction. If B should be defective, B will be made non-defective by a small perturbation, which will always be introduced through round-off errors.

To determine the size of α, we write the error e^{n-1} in the form $e^{n-1} = \sum_{i=1}^N e_i^{n-1} v^i$. By (3.4), it follows that

$$e^n = (I - \alpha(I+B)) \sum_{i=1}^N e_i^{n-1} v^i = \sum_{i=1}^N e_i^{n-1}(1 - \alpha(1+\lambda_i))v^i = \sum_{i=1}^N \sigma_i e_i^{n-1} v^i,$$

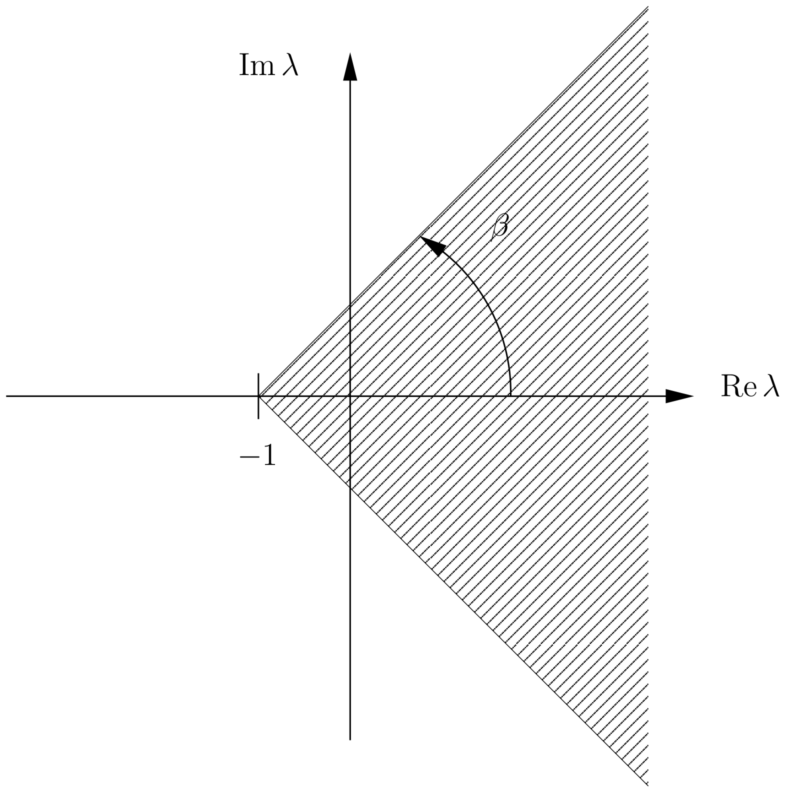

FIGURE 2. The eigenvalues λ of the matrix B are assumed to lie within the shaded sector.

where $\sigma_i = 1 - \alpha(1 + \lambda_i)$. It follows that the stabilized fixed point iteration converges if we take α such that $|\sigma_i| < 1$ for $i = 1, \ldots, N$. With

$$\alpha = \frac{\cos\beta}{1 + \max_i |\lambda_i|}, \tag{3.8}$$

we obtain

$$|\sigma_i|^2 = (\mathrm{Re}\,\sigma_i)^2 + (\mathrm{Im}\,\sigma_i)^2 = (1 - \alpha(1 + \mathrm{Re}\,\lambda_i))^2 + \alpha^2(\mathrm{Im}\,\lambda_i)^2 = 1 + \alpha^2\tilde{r}_i^2 - 2\alpha(1 + \mathrm{Re}\,\lambda_i),$$

where $\tilde{r}_i = |1 + \lambda_i|$. By assumption, $|\arg(1 + \lambda_i)| \leq \beta$, and thus $1 + \mathrm{Re}\,\lambda_i \geq \tilde{r}_i \cos\beta$. It follows that

$$\begin{aligned} |\sigma_i|^2 &\leq 1 + \alpha^2\tilde{r}_i^2 - 2\alpha\tilde{r}_i\cos\beta = 1 + \tfrac{\tilde{r}_i^2\cos^2\beta}{(1+\max_i|\lambda_i|)^2} - \tfrac{2\tilde{r}_i\cos^2\beta}{1+\max_i|\lambda_i|} \\ &\leq 1 + \tfrac{\tilde{r}_i\cos^2\beta}{1+\max_i|\lambda_i|} - \tfrac{2\tilde{r}_i\cos^2\beta}{1+\max_i|\lambda_i|} = 1 - \tfrac{\tilde{r}_i\cos^2\beta}{1+\max_i|\lambda_i|} < 1, \end{aligned}$$

and thus the fixed point iteration (3.1) converges.

We note that, since α is chosen based on the largest eigenvalue, the convergence for eigenmodes corresponding to smaller eigenvalues may be slow, if $\tilde{r}_i \ll 1 + \max_i |\lambda_i|$. To improve the convergence for these eigenmodes, the algorithm determines a suitable number m of stabilizing iterations with damping factor α given by (3.8), and then gradually increases α by a factor two, towards $\alpha = 1$,

$$\alpha \leftarrow 2\alpha/(1+\alpha). \tag{3.9}$$

This corresponds to the use of stabilizing time step sequences in [3] for the stabilization of explicit methods for stiff problems.

To determine the number m of stabilizing iterations, we note that with $\alpha = 1$, the eigenmode corresponding to the largest eigenvalue will diverge by a factor $\max_i |\lambda_i|$. We further note that with damping factor α given by (3.8), this eigenmode will converge by a factor $\sim 1 - \cos\beta$. To compensate for one iteration with $\alpha = 1$, we thus need to choose m such that

$$(1-\cos\beta)^m \max_i |\lambda_i| < 1,$$

giving

$$m > \frac{\log(\max_i |\lambda_i|)}{\log 1/(1-\cos\beta)}. \tag{3.10}$$

We note that the number of stabilizing iterations becomes large for β close to $\pi/2$, and to limit the number of stabilizing iterations m, we assume in practice that $\beta = \pi/4$, which gives $\cos\beta = 1/\sqrt{2}$ and $m \approx \log(\max_i |\lambda_i|)$.

With α and m determined by (3.8) and (3.10), respectively, we need to determine the value of $\rho = \max_i |\lambda_i|$, which is obtained by *cumulative power iteration* as follows. The residual $F(x^n)$, or the increment d^n, is measured for a sequence of iterations with $\alpha = 1$. We let $\rho_1 = \|F(x^1)\|_{l_2}/\|F(x^0)\|_{l_2}$, and for $n = 2, 3, \ldots$ determine ρ_n according to

$$\rho_n = (\rho_{n-1})^{(n-1)/n}(\|F(x^n)\|_{l_2}/\|F(x^{n-1})\|_{l_2})^{1/n}, \tag{3.11}$$

until ρ_n has converged to within some tolerance, typically 10%. When ρ has been computed, the damping factor α and the number of stabilizing steps m are then determined for $\beta = \pi/4$ according to

$$\alpha = \frac{1/\sqrt{2}}{1+\rho}, \tag{3.12}$$

and

$$m = \log\rho. \tag{3.13}$$

As an example, we consider the solution of the linear system

$$x = g(x) = u_0 - KAx = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & -1 \\ \kappa & 200 \end{bmatrix} x, \tag{3.14}$$

by stabilized fixed point iteration, corresponding to one time step of size $K = 1$ with the dG(0) method for a mass-spring-dashpot system with damping $b = 200$ and spring constant κ. With $\kappa = 10^4$, the system is critically damped and $B = -KA$ is defective with two eigenvalues of size $\lambda = 100$. Although $\|B\|_{l_2} \approx 10^4$ and there is no $\alpha \in (0,1]$ such that $\|G_\alpha\|_{l_2} < 1$ with $G_\alpha = I - \alpha(I+B)$, the stabilized fixed point iteration converges, by targeting the stabilization at the largest eigenvalue $\lambda = 100$. This is illustrated in Figure 3, where we also plot the convergence for $\kappa = 2 \cdot 10^4$ and $\kappa = 10^3$, corresponding

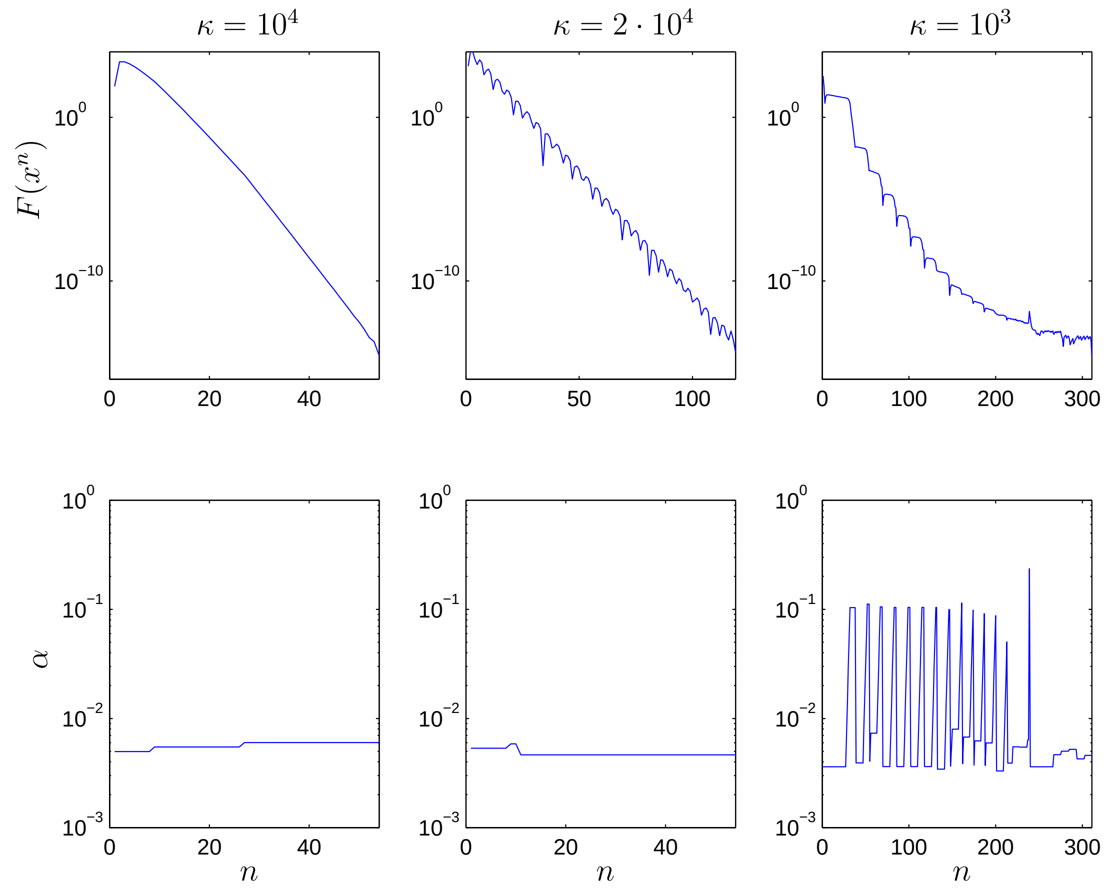

FIGURE 3. Convergence of the stabilized fixed point iteration for the solution of (3.14) with $\kappa = 10^4$, $\kappa = 2 \cdot 10^4$, and $\kappa = 10^3$.

to an under-damped and over-damped harmonic oscillator, respectively. All three results were obtained using same general iterative algorithm available in version 0.4.7 of **DOLFIN**, which automatically detects the appropriate size of α in each iteration.

Note the different behavior of the convergence for the three different systems. For $\kappa = 10^4$, there is only one eigenvalue and so α needs to be targeted only at this eigenvalue. For $\kappa = 2 \cdot 10^4$, the eigenvalues $\lambda = 100 \pm 100i$ of A have a large imaginary part, which results in oscillations in the convergence. For $\kappa = 10^3$, the matrix A has two eigenvalues $\lambda_1 \approx 5$ and $\lambda_2 \approx 195$ of different magnitudes. For the stabilized fixed point iteration to converge, it is then important that α is targeted not only at the large eigenvalue λ_2, but also at the smaller eigenvalue λ_1, which is accomplished by gradually increasing the damping factor α after each sequence of stabilizing iterations.

4. Algorithm

The algorithm we propose is a modified version of the algorithm presented earlier in [8]. Note that the method, mcG(q) or mdG(q), remains unchanged.

The original multi-adaptive iterative strategy of [8] is based on simple fixed point iteration. For certain problems (stiff problems), this iteration may fail to converge. We take this as the definition of stiffness: A problem is *stiff* when simple fixed point iteration does not converge. With this definition, the stiffness will depend on the size of the time steps (and thus on the tolerance) and the exact construction of the time slab, as well as properties of the differential equation (1.1) itself, such as the eigenvalues of the Jacobian of the right-hand side f.

The modified algorithm differs from the original algorithm both in how the time slabs are constructed, and in how the iteration is performed on each time slab. The new algorithm is intelligent, in the sense that the iterative strategy is automatically adapted to the detected level of stiffness. This means in particular that for a non-stiff problem, simple fixed point iteration is used, essentially corresponding to the original algorithm of [8].

4.1. **Recursive construction of time slabs.**

In [8], time slabs are constructed in a way that allows each component to have its individual time step sequence, independent of the time step sequences for other components. No restrictions are made on the time step sequences, other than that the first time step is the same for each component and also that the last time step for each component may be adjusted to match the given end time T for the simulation.

The algorithm presented in [8] gives the proper time step sequence for each component, but has the disadvantage that there is little structure in the organization of the time slabs. In particular, a time slab does not have a well-defined left end-point T_{n-1} and right end-point T_n.

The new algorithm recursively constructs a time slab between two synchronized time levels T_{n-1} and T_n, consisting of at least one element for every component. Each element $(I_{ij}, U|_{I_{ij}})$ within the time slab satisfies the relation $T_{n-1} \leq t_{i,j-1} < t_{ij} \leq T_n$.

The time slab is organized recursively as follows. The root time slab covering the interval $(T_{n-1}, T_n]$ contains a non-empty list of elements, which we refer to as an *element group*, and a possibly empty list of time slabs, which in turn may contain nested groups of elements and time slabs. This is illustrated in Figure 4.

For the construction of the time slab, we first examine each component for its desired time step. This time step is adaptively determined by a controller from the current component residual with the goal of satisfying a given error tolerance, as discussed in [8]. In the current implementation, a simple controller, based on the harmonic mean value with the previous time step, has been used. For each component, an individual controller is used. We let $k = (k_i)$ denote the vector of desired time steps for the different components, as determined by the controllers.

To construct the time slab, we let K be the largest time step contained in the vector k,

$$K = \max_i k_i, \quad i \in I_0 = \{1, 2, \ldots, N\}. \tag{4.1}$$

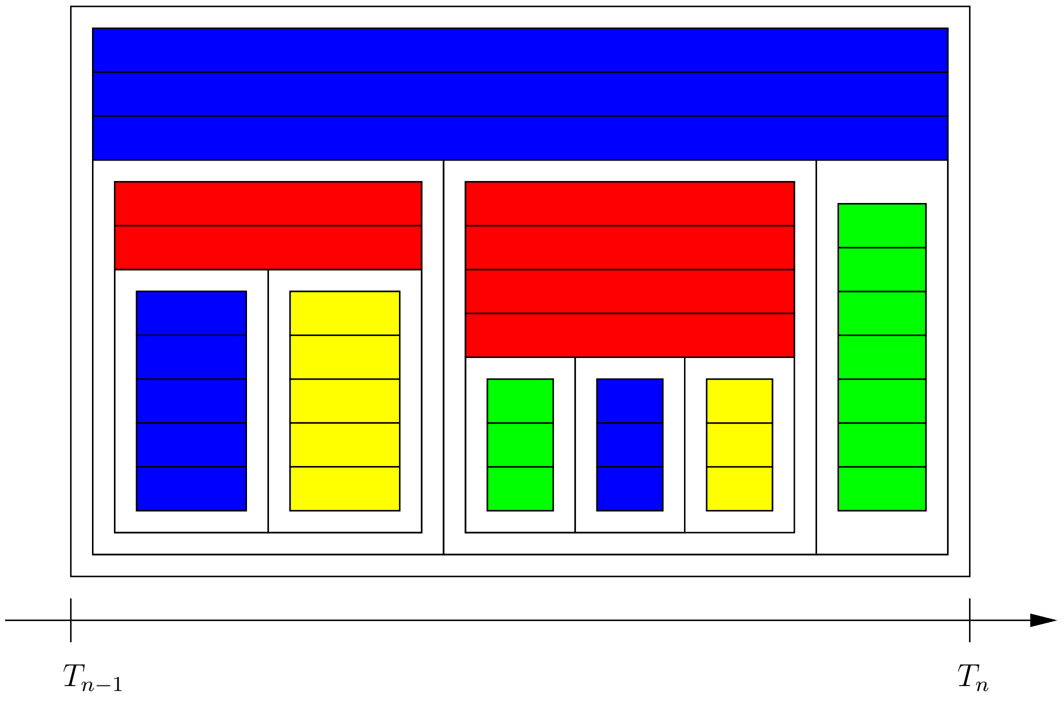

FIGURE 4. The recursive organization of the time slab. Each time slab contains an element group (shaded) and a list of recursively nested time slabs. The root time slab in the figure contains one element group of three elements and three time slabs. The first of these sub slabs contains an element group of two elements and two nested time slabs, and so on. The root time slab recursively contains a total of nine element groups and 35 elements.

We next partition the components into two groups, one group $I_{00} \subset I_0$ containing components with small time steps, and another group $I_{01} \subseteq I_0$ containing components with large time steps. For the partition of the components, we introduce a parameter $\theta \in [0, 1]$, referred to as the *partitioning threshold*, which determines the granularity of the time slab. A large partitioning threshold means that each component will have its own time step, and a small partitioning threshold means that all components will use the same time step. By varying θ, we may thus vary the multi-adaptive nature of the algorithm. The value of θ determines (roughly speaking), the maximum quotient between two different time steps within the time slab.

All components for which $k_i < \theta K$ are assigned to the group I_{00} of components with small time steps, and the remaining components for which $k_i \geq \theta K$ are assigned to the group I_{01} of components with large time steps. Among the components with large time steps, we determine the minimum time step

$$\bar{K} = \min_i k_i, \quad i \in I_{01}. \tag{4.2}$$

The size of the time slab is then adjusted according to

$$T_n = \min(T_{n-1} + \bar{K}, T), \tag{4.3}$$

i.e., we let $\bar{K}$ be the size of the time slab and adjust the size if we should reach the given final time T. We illustrate the partition of components in Figure 5.

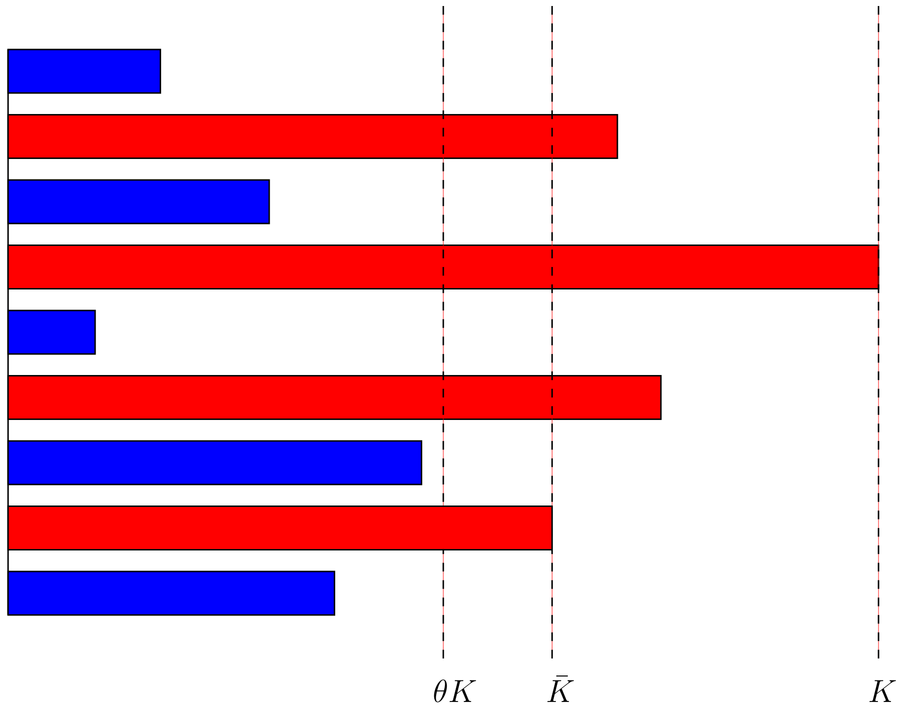

FIGURE 5. The partition of components into groups of small and large time steps for $\theta = 1/2$.

The time slab is then recursively created by first constructing a list of sub slabs for the components contained in the group I_{00}, and then constructing an element group containing one element for each of the components within the group I_{01}, as illustrated in Figure 4. Each of these elements covers the full length of the time slab, $t_{i,j-1} = T_{n-1}$ and $t_{ij} = T_n$.

Note that we construct the list of sub slabs before we create the element group. The tree of time slabs is thus constructed recursively depth first. This means in particular that the first element that is constructed is for the component with the smallest time step. The original multi-adaptive algorithm presented in [8] is based on the principle

(4.4) *The last component steps first.*

This property is automatically obtained through the depth first nature of the new recursive algorithm. In Figure 6, we illustrate the recursive construction of elements by numbering the elements in the order in which they are created.

The list of sub slabs is constructed sequentially until the list of sub slabs covers the interval $[T_{n-1}, T_n]$ of the parent (root) time slab. For the construction of each of these sub

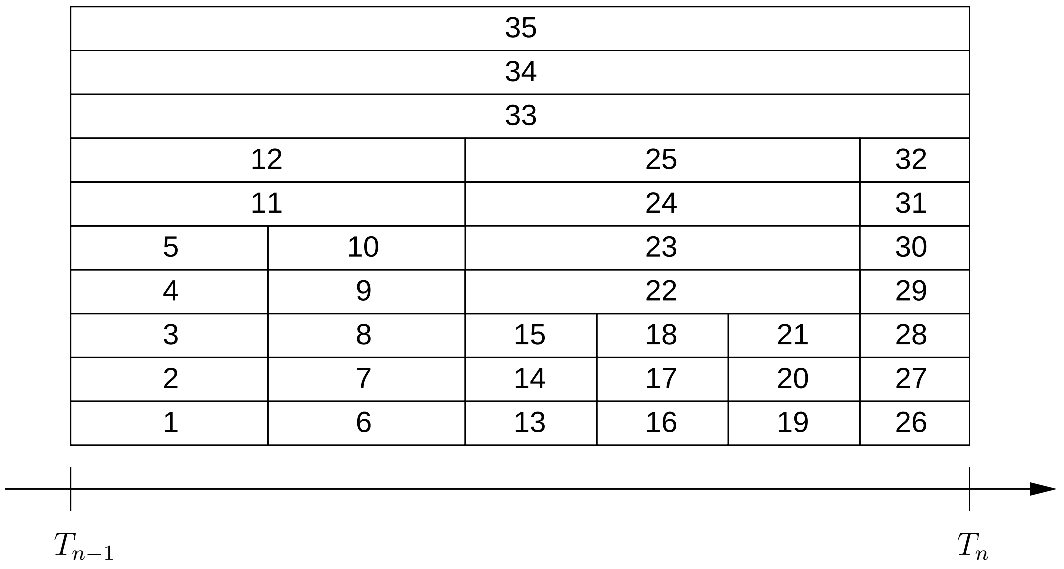

FIGURE 6. Numbering of the elements in the order in which they are created.

slabs, we again determine the maximum time step,

$$K = \max_i k_i, \quad i \in I_{00}. \tag{4.5}$$

All components within I_{00} for which $k_i < \theta K$ are assigned to the group I_{000} of components with small time steps within I_{00}, and the remaining components for which $k_i \geq \theta K$ are assigned to the group I_{001} of components with large time steps within I_{00}. As before, we let

$$\bar{K} = \min_i k_i, \quad i \in I_{001}, \tag{4.6}$$

and let $\bar{K}$ be the length of the sub slab. For the components in the group I_{000}, we continue recursively, until the group of components with small time steps is empty. At that point, the second time slab in the list of time slabs within the current parent time slab is constructed, until finally all time slabs and element groups within the root time slab have been created.

4.2. **Adaptive fixed point iteration on time slabs.** On each time slab, the system of discrete equations given by the mcG(q) or mdG(q) method is solved using adaptive fixed point iteration, as discussed in Section 3. Different iterative strategies are used, depending on the stiffness of the problem.

As described in Section 4.1, a time slab contains a number of element groups (counting also the element groups of the recursively contained time slabs). Each element group contains a list of elements, and each element represents a set of degrees of freedom for a component $U_i(t)$ on a local interval I_{ij}.

We may thus view the time slab as a large system of discrete equations of the form

$$F(x) = 0 \tag{4.7}$$

for the degrees of freedom x of all elements of all element groups contained in the time slab.

Alternatively, we may view the time slab as a set of coupled sub systems of the form (4.7), one for each element group, where each sub system consists of the degrees freedom of each element within the element group.

Since each element can also be viewed as a sub system for the element degrees of freedom, the time slab can be viewed as a set of sub systems, one for each element group within the time slab, which each in turn consists of a set of sub systems, one for each element within the element group.

We present below an iterative strategy that takes advantage of this nested structure of sub systems, in combination with adaptive stabilization at the different levels of iteration. We refer to iterations at the element level as *level 1* iterations, and to the iterations at the element group and time slab level as *level 2* and *level 3* iterations, respectively.

4.2.1. *Nested fixed point iteration.* The basic principle of the nested fixed point iteration is that each iteration on a given system consists of fixed point iteration on each sub system. For fixed point iteration on a time slab, the nested structure of sub systems consist of elements (level 1) contained in element groups (level 2), which in turn are contained in the time slab (level 3).

The general algorithm for nested fixed point iteration is given in Table 1. On each level, fixed point iteration is performed as long as a certain condition (*1*, *2*, or *3*) holds. This condition is typically of the form

$$r > \text{tol}, \tag{4.8}$$

where r is the size of the residual for the current sub system and $\text{tol} > 0$ is a tolerance for the residual. In each iteration, the current system is updated and each update consists of successive fixed point iteration on all sub systems. By different choices of condition for the fixed point iteration and for the type of adaptive damping on each level, different overall iterative methods are obtained. We present below four different versions of the iterative algorithm, which we refer to as *non-stiff iteration*, *adaptive level 1 iteration*, *adaptive level 2 iteration*, and *adaptive level 3 iteration*. The solver automatically detects which version of the algorithm to use, depending on the stiffness of the problem.

4.2.2. *Non-stiff iteration.* Unless otherwise specified by the user, the problem (1.1) is assumed to be non-stiff. The non-stiff version of the iterative algorithm is specified as follows. As discussed in [8], the system of equations to be solved for the degrees of freedom $\{\xi_m\}$ on each element is of the form

$$\xi_m = \xi_0 + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) f_i(U(t), t)\, dt, \quad m = 1, \ldots, q_{ij}, \tag{4.9}$$

for the mcG(q) method, where $\tau_{ij}(t) = (t - t_{ij})/(t_{ij} - t_{i,j-1})$ and $\{w_m^{[q_{ij}]}\}_{m=1}^{q_{ij}} \subset \mathcal{P}^{[q_{ij}-1]}([0,1])$ are polynomial weight functions. For the mdG(q) method, the system of equations on each element has a similar form.

```
Iterate(time slab)                     Update(time slab)
                                ──►
while condition 3 do                   for each element group do
    Update(time slab)           ◄─        Iterate(element group)
end while                              end for

Iterate(element group)                 Update(element group)
                                ──►
while condition 2 do                   for each element do
    Update(element group)       ◄─        Iterate(element)
end while                              end for

Iterate(element)                       Update(element)
while condition 1 do                   for each degree of freedom do
    Update(element)             ──►       Update(degree of freedom)
end while                              end for
```

Table 1. Nested fixed point iteration on the time slab.

For each element $(I_{ij}, U|_{I_{ij}})$, we define the element residual R_{ij}^e as

$$R_{ij}^e = \xi_{q_{ij}} - \xi_0 - \int_{I_{ij}} f_i(U(t), t)\, dt, \tag{4.10}$$

noting that $w_{q_{ij}}^{[q_{ij}]} \equiv 1$. For a given tolerance tol, we choose *condition 1* for the element iteration as

$$|R_{ij}^e| > \text{tol}. \tag{4.11}$$

For the iteration on element group level, *condition 2* is given by $\|R_{ij}^e\|_{l_2} > \text{tol}$, with the l_2 norm taken over all elements in the element group. Similarly, *condition 3* for the iteration on time slab level is given by $\|R_{ij}^e\|_{l_2} > \text{tol}$, with the l_2 norm taken over all elements in the time slab. In each iteration and for each element, the degrees of freedom are updated according to the fixed point iteration (4.9). On the element level, the update is of Gauss–Jacobi type, meaning that the degrees of freedom $\{\xi_m\}$ are computed using previously computed values, i.e., the new values $\xi_1, \ldots, \xi_{m-1}$ are not used when computing the new value of ξ_m. On the element group level and time slab level, the update is of Gauss–Seidel type, meaning that when an element is updated, the latest known values are used for all previously updated elements.

The nested fixed point iteration continues as long as *condition 3* is fulfilled. In each iteration at all levels, the convergence rate is measured, and if the convergence rate is not acceptable (or if the residual diverges), the system is stiff and a different iterative strategy is needed. The new choice of strategy is given by the iteration level at which stabilization is needed: If the iteration at element level needs stabilization, we change strategy to adaptive level 1 iteration. If the iteration at element group level needs stabilization, we change strategy to adaptive level 2 iteration, and if the iteration at time slab level needs stabilization, we change strategy to adaptive level 3 iteration.

4.2.3. *Adaptive level 1 iteration.* If the fixed point iteration at element level does not converge, the strategy is changed to adaptive level 1 iteration, which is similar to non-stiff iteration except that the iterations at element level are stabilized. For the mdG(0) method, we modify the fixed point iteration (4.9) according to

$$\xi_m \leftarrow (1-\alpha)\xi_m + \alpha \left[\xi_0 + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) f_i(U(t),t)\, dt \right], \tag{4.12}$$

with damping factor α determined by

$$\alpha = 1/(1 - k_{ij}\partial f_i/\partial u_i(U(t_{ij}), t_{ij})), \tag{4.13}$$

and appropriate modifications for higher-order methods, corresponding to the diagonal damping discussed in Section 3.1. As noted in [8], this type of iteration may be expected to perform well if the stiffness of the problem is of sufficient diagonal nature, i.e., if the Jacobian of the right-hand side is diagonally dominant, which is the case for many problems modeling chemical reactions.

As before, we measure the rate of convergence. If necessary, the strategy is changed to adaptive level 2 iteration, if the iterations at element group level do not converge, and to adaptive level 3 iteration, if the iterations at time slab level do not converge.

4.2.4. *Adaptive level 2 iteration.* If the fixed point iteration at element group level does not converge, the strategy is changed to adaptive level 2 iteration. The algorithm is similar to adaptive level 1 iteration, except that *condition 1* is modified so that exactly one iteration is performed on each element, and that the damping factor α is determined at the element group level. We also perform the iteration at element group level using Gauss–Jacobi type iteration, with the iteration at time slab level of Gauss–Seidel type. In each iteration, the convergence rate is measured. Whenever stabilization is necessary, the damping factor α is determined by cumulative power iteration according to (3.12), the number of stabilizing iterations is determined according to (3.13), and adaptive stabilization performed as discussed in Section 3.2.

4.2.5. *Adaptive level 3 iteration.* If the fixed point iteration at time slab level does not converge, the strategy is changed to adaptive level 3 iteration. We now modify *condition 1* and *condition 2*, so that exactly one iteration is performed on each element and on each element group. The iteration is now of Gauss–Jacobi type on the entire time slab, except that values are propagated forward in time between elements representing the same component. In each iteration, the convergence rate is measured and adaptive stabilization is performed as discussed in Section 3.2.

4.2.6. *Adaptive time step stabilization.* If the adaptively stabilized fixed point iteration fails to converge, we adjust the size of the time slab according to

$$K_n \leftarrow \alpha K_n, \tag{4.14}$$

and let K_n be the maximum allowed size of the time slab for a sequence of m successive time slabs, with m determined by (3.13), corresponding to the algorithm presented in [3] for the stabilization of explicit methods for stiff problems. Note that the limitation on the time step size might force different components to use the same time steps during the stabilization, in particular if α is small. After the sequence of m stabilizing time slabs, the maximum allowed size of the time slab is gradually increased by a factor two, with the hope that the stiffness can be handled by the adaptively stabilized fixed point iteration, as discussed above.

5. Examples

We illustrate the behavior of the multi-adaptive solver for a sequence of well-known stiff test problems that appear in the ODE literature. To a large extent, the problems are identical with those presented earlier in [3], where a stabilized mono-adaptive method was used to solve the problems.

This rich set of test problems presents a challenge for the multi-adaptive solver, since each of the test problems requires a slightly different strategy as discussed in Section 4. As shown below, the solver automatically and adaptively detects for each of the test problems which strategy to use and when to change strategy.

The results were obtained using the standard implementation of the stabilized multi-adaptive solver available in **DOLFIN** [5] version 0.4.7, which includes the full set of test problems. In all examples, the multi-adaptive dG(0) method was used, unless otherwise stated. For comparison, we present the (wall clock) time of simulation for each of the test problems, obtained on a standard desktop computer (Intel Pentium 4, 1.6 GHz) running Debian GNU/Linux.

5.1. **The test equation.** The first test problem is the scalar problem

$$\begin{aligned} \dot{u}(t) + \lambda u(t) &= 0, \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{5.1}$$

for $T = 10$, $\lambda = 1000$, and $u_0 = 1$. The solution, which is shown in Figure 7, was computed in less than 0.01 seconds, using (diagonally) damped fixed point iteration (adaptive level 1 iteration).

5.2. **The test system.** The second test problem is the diagonal problem

$$\begin{aligned} \dot{u}(t) + Au(t) &= 0, \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{5.2}$$

for $T = 10$, $A = \mathrm{diag}(100, 1000)$, and $u_0 = (1, 1)$. The solution, which is shown in Figure 8, was computed in 0.01 seconds, using diagonally damped (individual for each component) fixed point iteration (adaptive level 1 iteration).

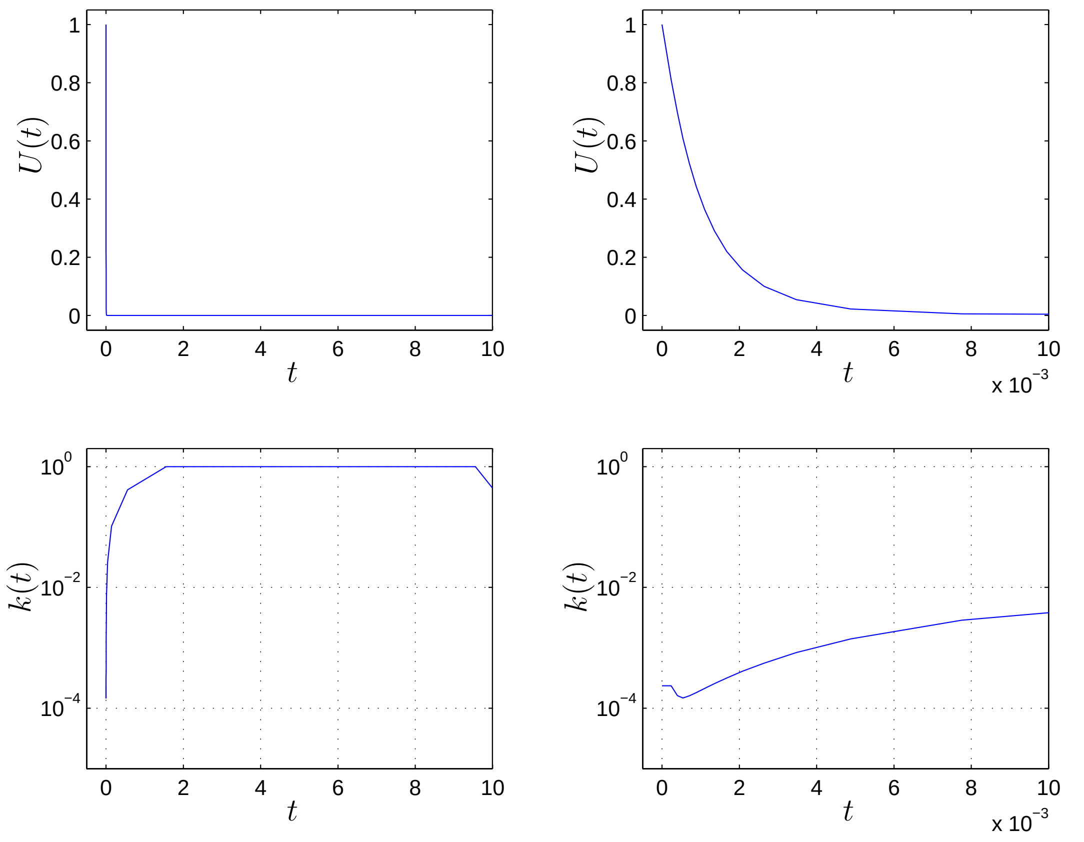

FIGURE 7. Solution and adaptive time step sequence for (5.1).

Note that the second component, which decays faster, initially uses smaller time steps than the first component. Later, when the second component is out of the transient with the first component still in its transient, the situation is the opposite with smaller time steps for the first component.

5.3. **A non-normal test problem.** The next problem is the mass-spring-dashpot system (3.14) with $\kappa = 10^4$, corresponding to critical damping, i.e.,

$$\begin{aligned} \dot{u}(t) + Au(t) &= 0, \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{5.3}$$

for $T = 1$, with

$$A = \begin{bmatrix} 0 & -1 \\ 10^4 & 200 \end{bmatrix}$$

and $u_0 = (1, 1)$. The solution, which is shown in Figure 9, was computed in 0.38 seconds, using a combination of adaptively stabilized fixed point iteration on the time slab level (adaptive level 3 iteration), and stabilizing time steps.

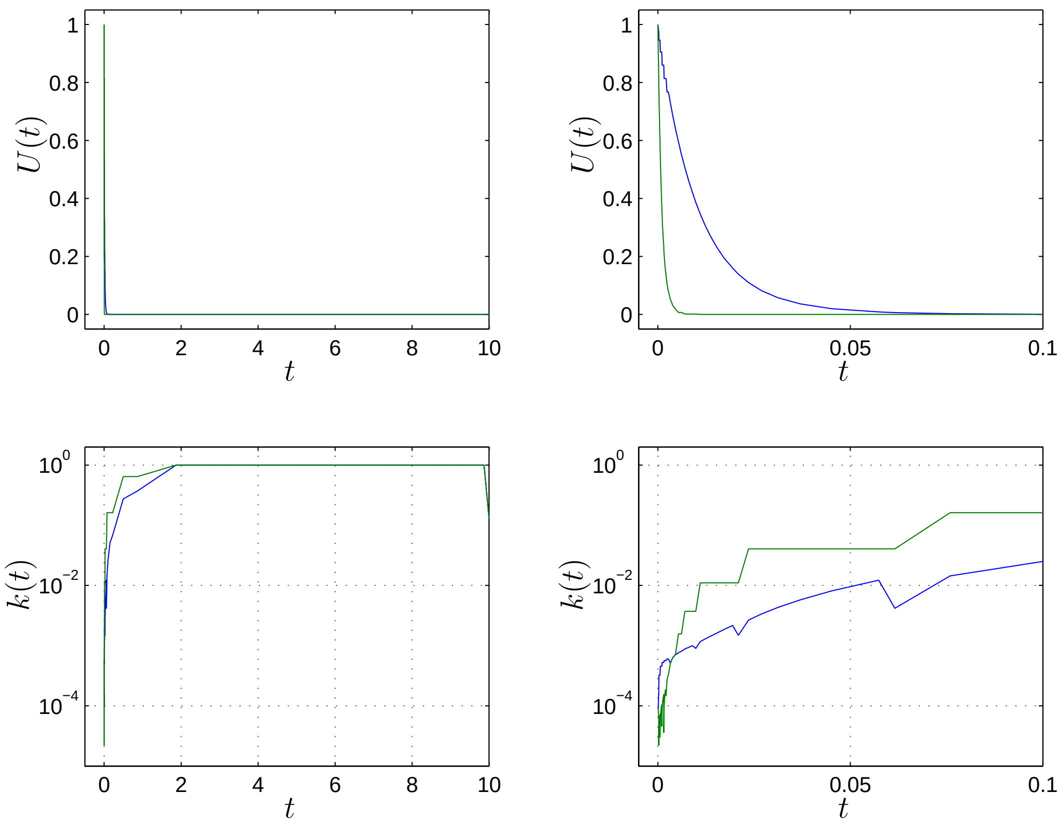

FIGURE 8. Solution and adaptive time step sequence for (5.2).

5.4. **The HIRES problem.** The HIRES problem ("High Irradiance RESponse") originates from plant physiology and is taken from a test set of initial value problems [11] for ODE solvers. The problem consists of the following eight equations:

$$
\left\{
\begin{array}{lcl}
\dot{u}_1 & = & -1.71u_1 + 0.43u_2 + 8.32u_3 + 0.0007, \\
\dot{u}_2 & = & 1.71u_1 - 8.75u_2, \\
\dot{u}_3 & = & -10.03u_3 + 0.43u_4 + 0.035u_5, \\
\dot{u}_4 & = & 8.32u_2 + 1.71u_3 - 1.12u_4, \\
\dot{u}_5 & = & -1.745u_5 + 0.43u_6 + 0.43u_7, \\
\dot{u}_6 & = & -280.0u_6u_8 + 0.69u_4 + 1.71u_5 - 0.43u_6 + 0.69u_7, \\
\dot{u}_7 & = & 280.0u_6u_8 - 1.81u_7, \\
\dot{u}_8 & = & -280.0u_6u_8 + 1.81u_7,
\end{array}
\right.
\tag{5.4}
$$

on $[0, 321.8122]$ with initial condition $u_0 = (1.0, 0, 0, 0, 0, 0, 0, 0.0057)$. The solution, which is shown in Figure 10, was computed in 0.32 seconds, using diagonally damped fixed point iteration (adaptive level 1 iteration).

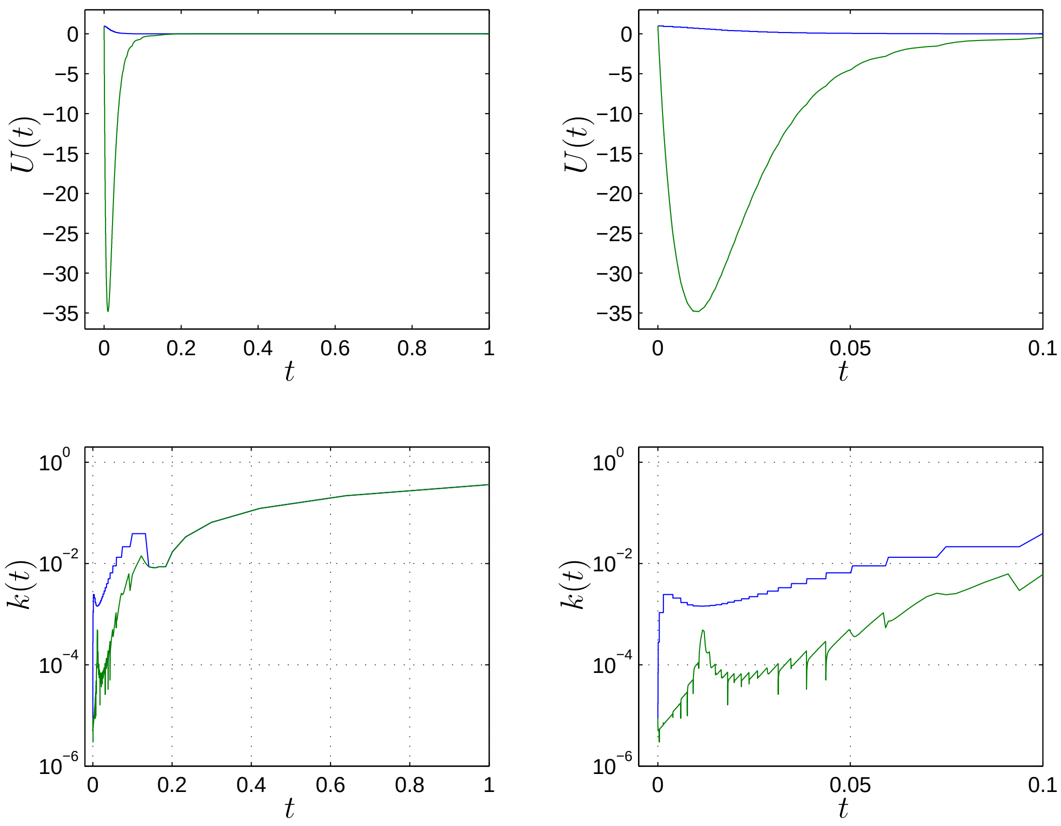

FIGURE 9. Solution and adaptive time step sequence for (5.3).

5.5. **The Akzo-Nobel problem.** We next solve the “Chemical Akzo-Nobel” problem taken from the test set [11], consisting of the following six equations:

$$(5.5)\qquad \left\{\begin{array}{lcl} \dot{u}_1 & = & -2r_1 + r_2 - r_3 - r_4, \\ \dot{u}_2 & = & -0.5r_1 - r_4 - 0.5r_5 + F, \\ \dot{u}_3 & = & r_1 - r_2 + r_3, \\ \dot{u}_4 & = & -r_2 + r_3 - 2r_4, \\ \dot{u}_5 & = & r_2 - r_3 + r_5, \\ \dot{u}_6 & = & K_s u_1 u_4 - u_6, \end{array}\right.$$

on $[0, 180]$, where $F = 3.3 \cdot (0.9/737 - u_2)$ and the reaction rates are given by $r_1 = 18.7 \cdot u_1^4\sqrt{u_2}$, $r_2 = 0.58 \cdot u_3u_4$, $r_3 = 0.58/34.4 \cdot u_1u_5$, $r_4 = 0.09 \cdot u_1u_4^2$, and $r_5 = 0.42 \cdot u_6^2\sqrt{u_2}$. The initial condition is given by $u_0 = (0.444, 0.00123, 0, 0.007, 0, 0.36)$. The solution, which is shown in Figure 11, was computed in 0.14 seconds, using a combination of adaptively stabilized fixed point iteration on the time slab level (adaptive level 3 iteration), and stabilizing time steps. Note that for this particular problem, the partitioning threshold with a default value of $\theta = 0.5$ forces all components to use the same time steps.

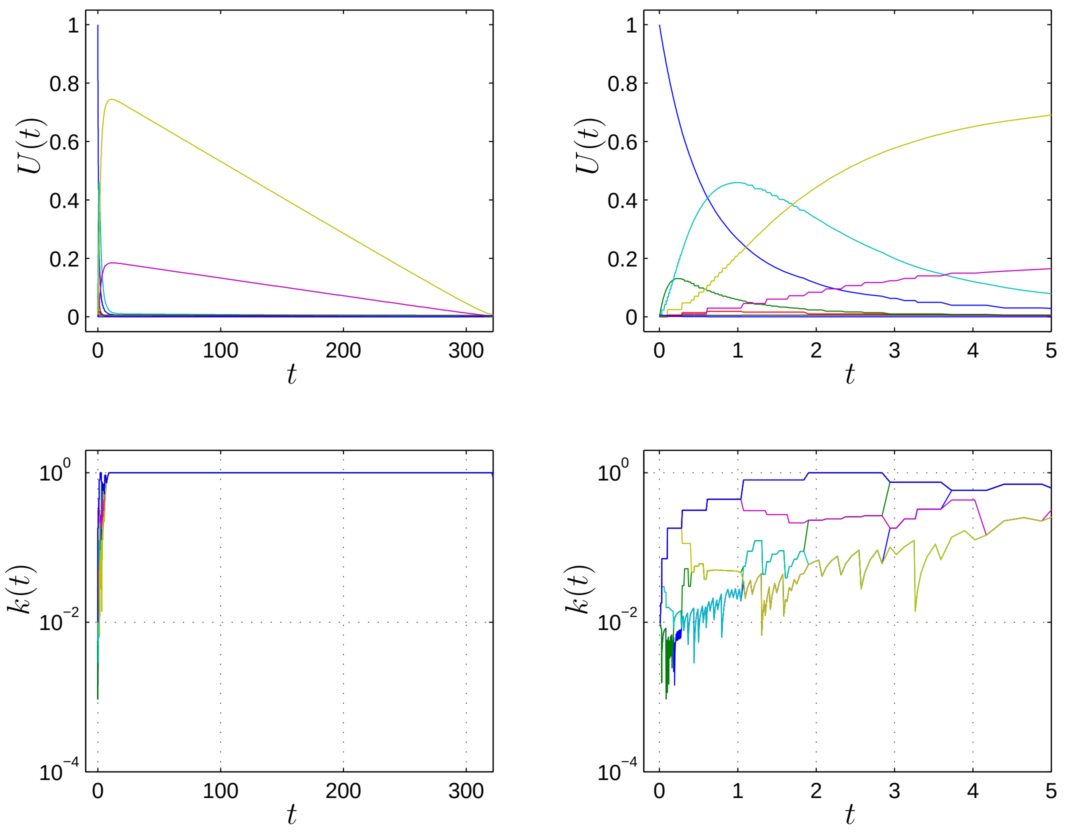

FIGURE 10. Solution and adaptive time step sequence for (5.4).

5.6. **Van der Pol's equation.** A stiff problem discussed in the book by Hairer and Wanner [4] is Van der Pol's equation,

$$\ddot{u} + \mu(u^2 - 1)\dot{u} + u = 0,$$

which we write in the form

$$\left\{ \begin{array}{rcl} \dot{u}_1 & = & u_2, \\ \dot{u}_2 & = & -\mu(u_1^2 - 1)u_2 - u_1. \end{array} \right. \tag{5.6}$$

We take $\mu = 10$ and compute the solution on the interval $[0, 100]$ with initial condition $u_0 = (2, 0)$. The solution, which is shown in Figure 12, was computed in 1.89 seconds, using a combination of adaptively stabilized fixed point iteration on the time slab level (adaptive level 3 iteration) and stabilizing time steps.

Note how the time steps are drastically reduced at the points where the derivatives, and thus also the residuals, of the mdG(0) solution are large.

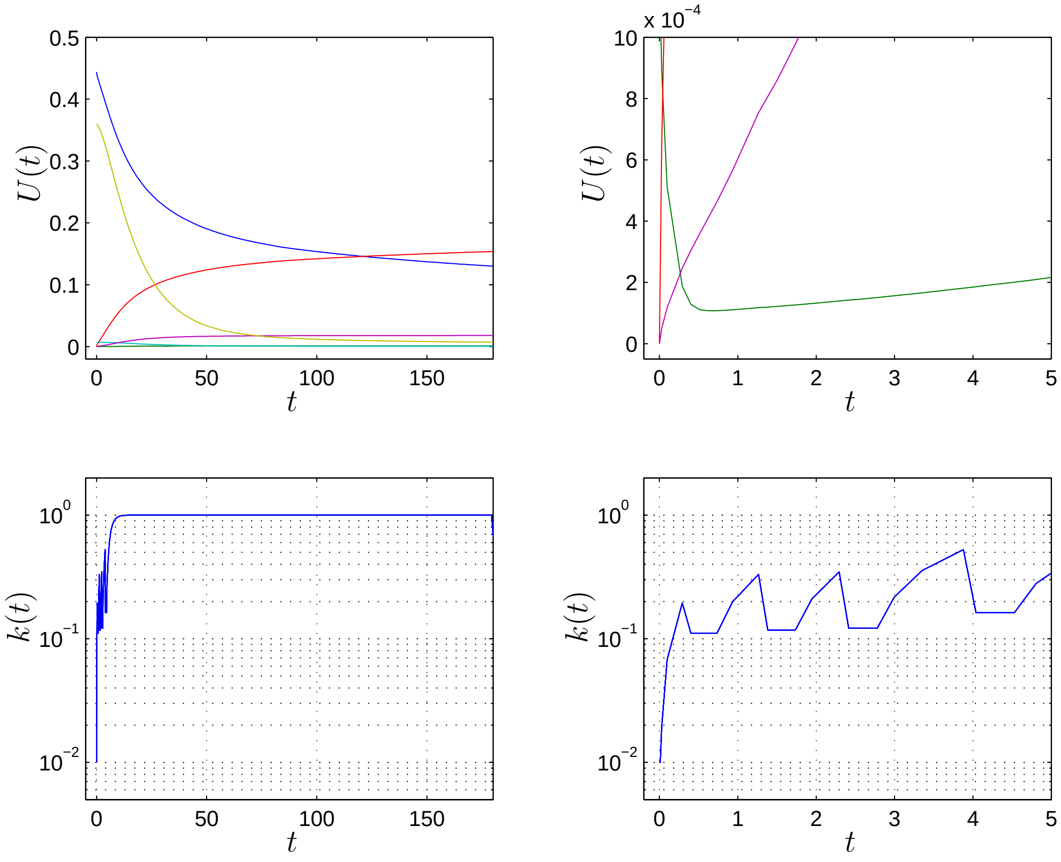

FIGURE 11. Solution and adaptive time step sequence for (5.5).

5.7. **The heat equation.** A special stiff problem is the one-dimensional heat equation,

$$\begin{aligned} \dot{u}(x,t) - u''(x,t) &= f(x,t), \quad x \in (0,1), \quad t > 0, \\ u(0,t) = u(1,t) &= 0, \quad t > 0 \\ u(x,0) &= 0, \quad x \in [0,1], \end{aligned}$$

where we choose $f(x,t) = f(x)$ as an approximation of the Dirac delta function at $x = 0.5$. Discretizing in space, we obtain the ODE

$$\begin{aligned} \dot{u}(t) + Au(t) &= f, \\ u(0) &= 0, \end{aligned} \tag{5.7}$$

where A is the *stiffness matrix* (including lumping of the mass matrix). With a spatial resolution of $h = 0.01$, the eigenvalues of A are distributed in the interval $[0, 4 \cdot 10^4]$. The solution, which is shown in Figure 13, was computed with a partitioning threshold $\theta = 0.1$ in 2.99 seconds, using a combination of adaptively stabilized fixed point iteration on the time slab level (adaptive level 3 iteration) and stabilizing time steps.

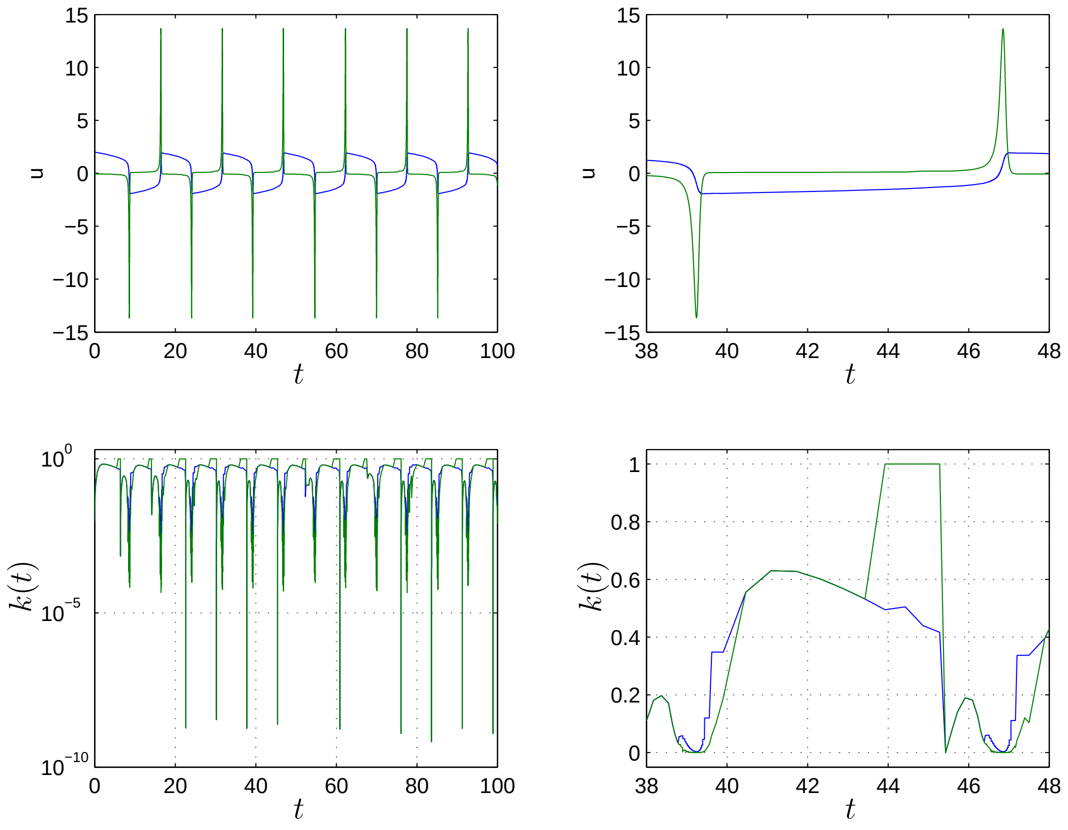

FIGURE 12. Solution and adaptive time step sequence for (5.6).

5.8. **A chemical reaction test problem.** The next problem originating from 1966 (Robertson) is taken from Hairer and Wanner [4]. This problem models the following system of chemical reactions:

$$\begin{array}{cccl} A & \xrightarrow{0.04} & B & \text{(slow)} \\ B+B & \xrightarrow{3\cdot 10^7} & B+C & \text{(very fast)} \\ B+C & \xrightarrow{10^4} & A+C & \text{(fast)} \end{array}$$

which gives an initial value problem of the form

$$\left\{ \begin{array}{lcl} \dot{u}_1 & = & -0.04u_1 + 10^4 u_2 u_3, \\ \dot{u}_2 & = & 0.04u_1 - 10^4 u_2 u_3 - 3\cdot 10^7 u_2^2, \\ \dot{u}_3 & = & 3\cdot 10^7 u_2^2. \end{array} \right. \tag{5.8}$$

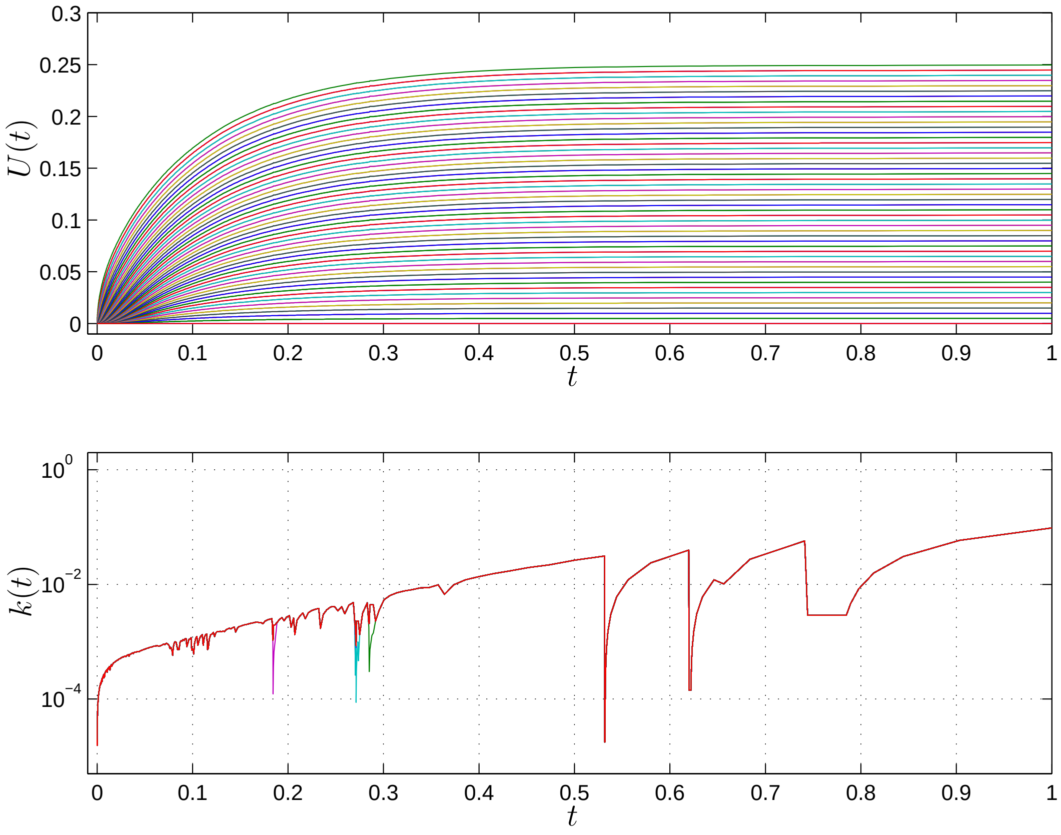

FIGURE 13. Solution and adaptive time step sequence for (5.7).

We compute the solution on the interval $[0, 0.3]$ with $u_0 = (1, 0, 0)$. The solution, which is shown in Figure 14, was computed in 0.01 seconds, using diagonally damped fixed point iteration (adaptive level 1 iteration).

5.9. **A mixed stiff/non-stiff test problem.**

As a final test problem, we solve the simple system

$$\begin{cases} \dot{u}_1 &= u_2, \\ \dot{u}_2 &= -(1-u_3)u_1, \\ \dot{u}_3 &= -\lambda(u_1^2+u_2^2)u_3, \end{cases} \tag{5.9}$$

which, since $u_3 \approx 0$ and $u_1^2 + u_2^2 \approx 1$, we recognize as the combination of a harmonic oscillator (the first and second components) and the simple scalar test problem (5.1). We take $\lambda = 1000$ and compute the solution on $[0, 30]$ with initial condition $u_0 = (0, 1, 1)$. As an illustration, we use the mcG(1) method for the first two non-stiff components and the mdG(0) method for the third stiff component.

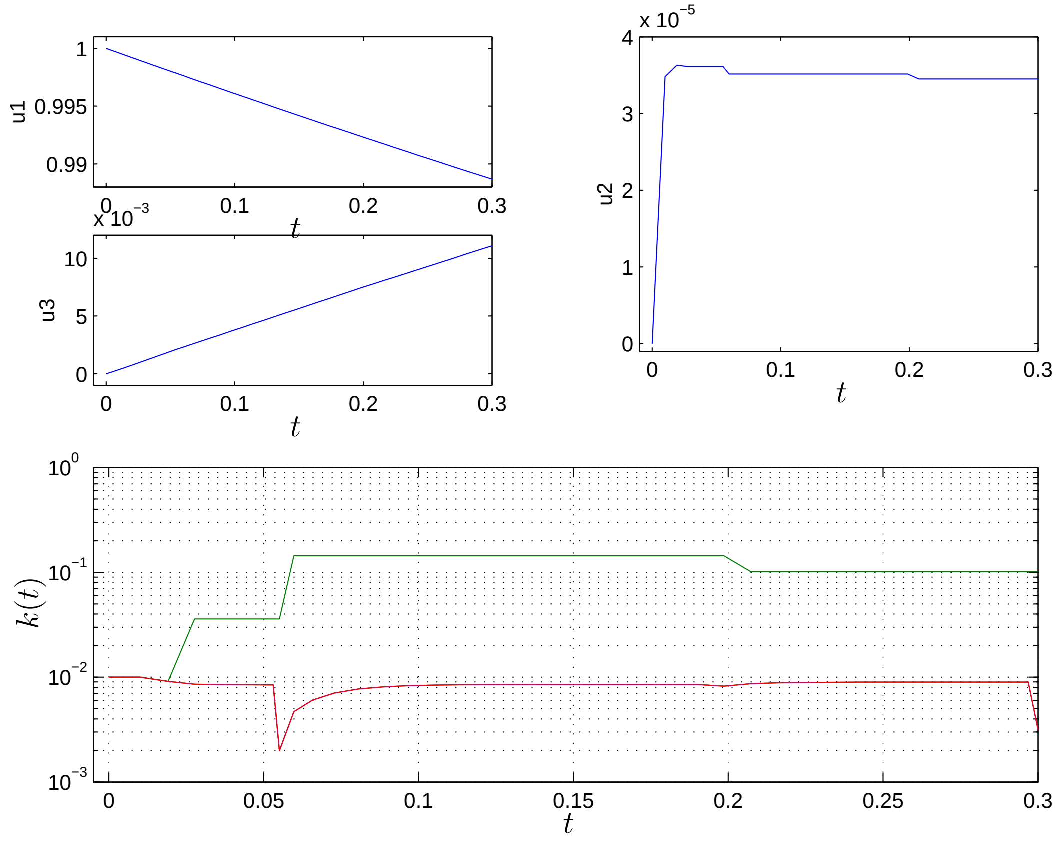

FIGURE 14. Solution and adaptive time step sequence for (5.8).

The solution, which is shown in Figure 16, was computed in 0.06 seconds, using diagonally damped fixed point iteration (adaptive level 1 iteration). With a partitioning threshold of default size $\theta = 0.5$, we notice that the time steps for the stiff third component are sometimes decreased, whenever $k_i > \theta k_3$ for $i = 1$ or $i = 2$. This is also evident in Figure 15, where we plot the sequence of time slabs on the interval $[10, 30]$.

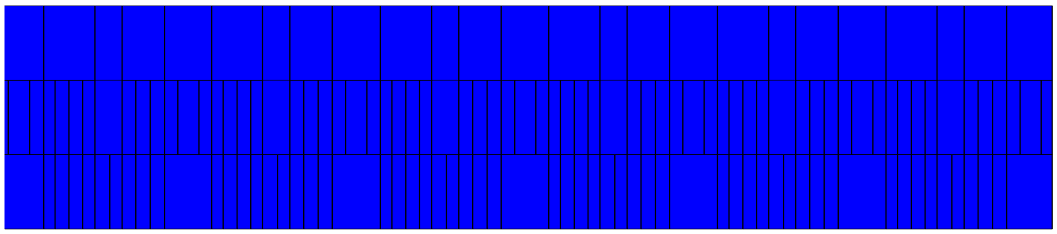

FIGURE 15. The structure of the time slabs on the interval $[10, 30]$ for the solution of (5.9).

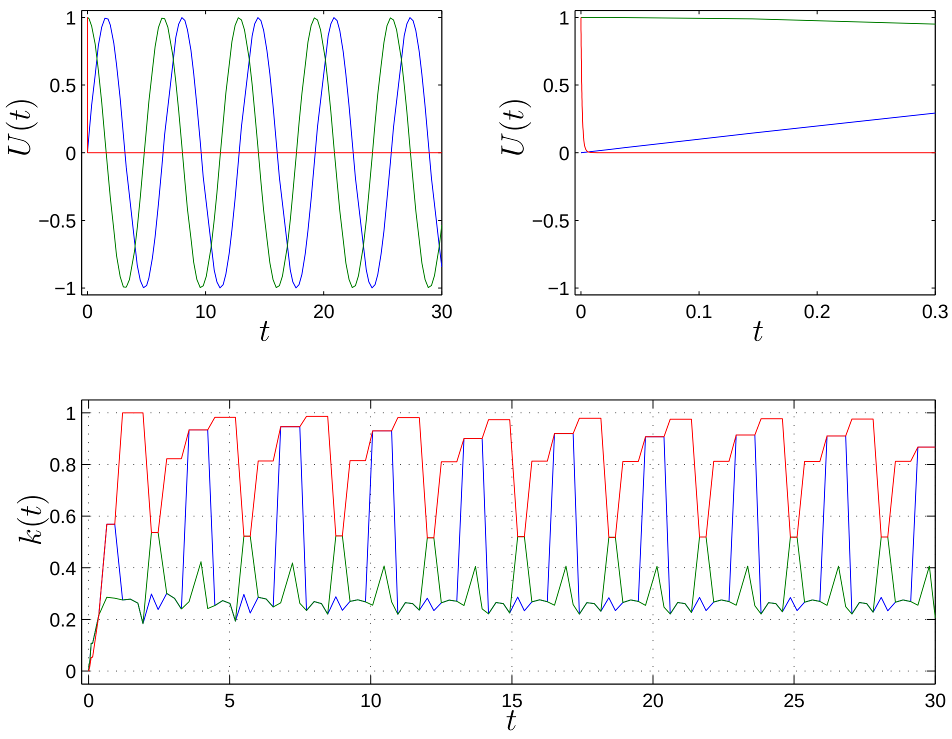

Figure 16. Solution and adaptive time step sequence for (5.9).

References

[1] G. Dahlquist, *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*, PhD thesis, Stockholm University, 1958.

[2] T. Dupont, J. Hoffman, C. Johnson, R.C. Kirby, M.G. Larson, A. Logg, and R. Scott, *The FEniCS project*, Tech. Rep. 2003–21, Chalmers Finite Element Center Preprint Series, 2003.

[3] K. Eriksson, C. Johnson, and A. Logg, *Explicit time-stepping for stiff ODEs*, SIAM J. Sci. Comput., 25 (2003), pp. 1142–1157.

[4] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II — Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, vol 14, 1991.

[5] J. Hoffman and A. Logg et al., *DOLFIN*, http://www.phi.chalmers.se/dolfin/.

[6] J. Jansson and A. Logg, *Algorithms for multi-adaptive time-stepping*, Tech. Rep. 2004–13, Chalmers Finite Element Center Preprint Series, 2004.

[7] A. Logg, *Multi-adaptive Galerkin methods for ODEs I*, SIAM J. Sci. Comput., 24 (2003), pp. 1879–1902.

[8] ———, *Multi-adaptive Galerkin methods for ODEs II: Implementation and applications*, SIAM J. Sci. Comput., 25 (2003), pp. 1119–1141.

[9] ———, *Multi-adaptive Galerkin methods for ODEs III: Existence and stability*, Submitted to SIAM J. Numer. Anal., (2004).

[10] ———, *Multi-adaptive Galerkin methods for ODEs IV: A priori error estimates*, Submitted to SIAM J. Numer. Anal., (2004).

[11] F. Mazzia and F. Iavernaro, *Test set for initial value problem solvers, release 2.2*, Department of Mathematics, University of Bari, Report 40/2003, (2003).

2003–19 *Multi-adaptive time integration*
Anders Logg

2003–20 *Adaptive computational methods for parabolic problems*
Kenneth Eriksson, Claes Johnson, and Anders Logg

2003–21 *The FEniCS project*
T. Dupont, J. Hoffman, C. Johnson, R. Kirby, M. Larson, A. Logg, and R. Scott

2003–22 *Adaptive finite element methods for LES: Computation of the mean drag coefficient in a turbulent flow around a surface mounted cube using adaptive mesh refinement*
Johan Hoffman

2003–23 *Adaptive DNS/LES: a new agenda in CFD*
Johan Hoffman and Claes Johnson

2003–24 *Multiscale convergence and reiterated homogenization of parabolic problem*
Anders Holmbom, Nils Svanstedt, and Niklas Wellander

2003–25 *On the relationship between some weak compactnesses with different numbers of scales*
Anders Holmbom, Jeanette Silfver, Nils Svanstedt, and Niklas Wellander

2003–26 *A posteriori error estimation in computational inverse scattering*
Larisa Beilina and Claes Johnson

2004–01 *Computability and adaptivity in CFD*
Johan Hoffman och Claes Johnson

2004–02 *Interpolation estimates for piecewise smooth functions in one dimension*
Anders Logg

2004–03 *Estimates of derivatives and jumps across element boundaries for multi-adaptive Galerkin solutions of ODEs*
Anders Logg

2004–04 *Multi-adaptive Galerkin methods for ODEs III: Existence and stability*
Anders Logg

2004–05 *Multi-adaptive Galerkin methods for ODEs IV: A priori error estimates*
Anders Logg

2004–06 *A stabilized non-conforming finite element method for incompressible flow*
Erik Burman and Peter Hansbo

2004–07 *On the uniqueness of weak solutions of Navier-Stokes equations: Remarks on a Clay Institute prize problem*
Johan Hoffman and Claes Johnson

2004–08 *A new approach to computational turbulence modeling*
Johan Hoffman and Claes Johnson

2004–09 *A posteriori error analysis of the boundary penalty method*
Kenneth Eriksson, Mats G. Larson, and Axel Målqvist

2004–10 *A posteriori error analysis of stabilized finite element approximations of the helmholtz equation on unstructured grids*
Mats G. Larson and Axel Målqvist

2004–11 *Adaptive variational multiscale methods based on a posteriori error estimation*
Mats G. Larson and Axel Målqvist

2004–12 *Multi-adaptive Galerkin methods for ODEs V: Stiff problems*
Johan Jansson and Anders Logg