ON EXPLICIT TIME-STEPPING FOR STIFF ODES

KENNETH ERIKSSON, CLAES JOHNSON, AND ANDERS LOGG

ABSTRACT. We present a new strategy for solving stiff ODEs with explicit methods. By adaptively taking a small number of stabilizing small explicit time steps when necessary, a stiff ODE system can be stabilized enough to allow for time steps much larger than what is indicated by classical stability analysis. For many stiff problems the cost of the stabilizing small time steps is small and so the improvement is large. We illustrate the technique on a number of well-known stiff test problems.

1. INTRODUCTION

The classical wisdom developed in the 1950s regarding stiff ODEs is that efficient integration requires implicit (A-stable) methods, at least outside transients where the time steps may be chosen large from an accuracy point of view. Using an explicit method (with a bounded stability region) the time steps have to be small at all times for stability reasons, and thus the advantage of a low cost per time step may be counter-balanced by the necessity of taking a large number of steps. As a result, the overall efficiency of an explicit method for a stiff ODE may be small.

The question is now if it is possible to combine the low cost per time step of an explicit method with the possibility of choosing large time steps outside transients. To reach this favorable combination some kind of stabilization of the explicit method seems to be needed, and the basic question is then if the stabilization can be realized at a low cost.

The stabilization technique proposed in this note relies on the inherent property of the stiff problem itself: rapid damping of high frequencies. This is achieved by stabilizing the system with a couple of small stabilizing (explicit Euler) steps. We test this idea in adaptive form, where the size and number of the small time steps are adaptively chosen according to the size of different residuals. In particular, we compute rates of divergence to determine the current mode λ of largest amplification and determine a corresponding small time step $k \approx \frac{1}{\lambda}$ with high damping. We test the corresponding adaptive method in the setting of the cG(1) method with fixed point iteration, effectively corresponding to an explicit method if the number of iterations is kept small. We show in a sequence of test problems that the proposed adaptive method gives a significant reduction in work in comparison to a standard implementation of an explicit method, with a typical speedup factor of ~ 100. We conclude that for many stiff problems, we may efficiently use an explicit method,

Date: May 21, 2003.

Key words and phrases. Stiff ODE, explicit methods, explicit Euler, Galerkin methods.

Department of Computational Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden, *email*: kenneth@math.chalmers.se, claes@math.chalmers.se, logg@math.chalmers.se. This work was completed while the second and third authors were visiting the Newton Institute for Mathematical Sciences, Cambridge, UK..

if only the explicit method is adaptively stabilized with a relatively small number of small time steps, and so we reach the desired combination of a low cost per time step and the possibility of taking large time steps outside transients. If the proposed method can be efficient also in comparison to implicit methods remains to be seen.

We have been led to this question in our work on multi-adaptive cG(q) or dG(q) ODE-solvers based on Galerkin's method with continuous or discontinuous polynomials of degree q, where individual time steps are used for different components, see [7, 8]. These methods are implicit and to realize efficient implementations, we need to use fixed point iteration with simple preconditioners. With a limited (small) number of iterations, these iterative solvers correspond to explicit time-stepping and the same question of the cost of stabilization arises.

The motivation for this work comes also from situations where for some reason we are forced to using an explicit method, such as in simulations of very large systems of chemical reactions or molecular dynamics, where the forming of Jacobians and solution of the linear system become very expensive.

Possibly, similar techniques may be used also to stabilize multi-grid smoothers.

2. Basic strategy

We consider first the *test equation*: Find $u : [0, \infty) \to \mathbb{R}$ such that

$$\begin{aligned} \dot{u}(t) + \lambda u(t) &= 0 \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{2.1}$$

where $\lambda > 0$ and u^0 is a given initial condition. The solution is given by $u(t) = \exp(-\lambda t)u^0$. We define the *transient* as $\{t > 0 : \lambda t \leq C\}$ with C a moderate constant. Outside the transient $u(t) = \exp(-\lambda t)u^0$ will be small.

The *explicit Euler method* for the test equation reads

$$U^n = -k_n \lambda U^{n-1} + U^{n-1} = (1 - k_n \lambda)U^{n-1}.$$

This method is conditionally stable and requires that $k_n \lambda \leq 2$, which outside transients is too restrictive for λ large.

Now let K be a large time step satisfying $K\lambda > 2$ and let k be a small time step chosen so that $k\lambda < 2$. Consider the method

$$U^n = (1 - k\lambda)^m (1 - K\lambda)U^{n-1}, \tag{2.2}$$

corresponding to one explicit Euler step with large time step K and m explicit Euler steps with small time steps k, where m is a positive integer to be determined. Altogether this corresponds to a time step of size $k_n = K + mk$. Defining the polynomial function $P(x) = (1 - \theta x)^m (1 - x)$, where $\theta = \frac{k}{K}$, we can write the method (2.2) in the form

$$U^n = P(K\lambda)U^{n-1}.$$

We now seek to choose m so that $|P(K\lambda)| \leq 1$, which is needed for stability. We thus need to satisfy

$$|1 - k\lambda|^m (K\lambda - 1) \leq 1,$$

that is,

$$m \geq \frac{\log(K\lambda - 1)}{-\log|1 - k\lambda|} \approx \frac{\log(K\lambda)}{c}, \tag{2.3}$$

with $c = k\lambda$ a moderate constant, for definiteness $c = 1/2$.

We conclude that m will be of moderate size and consequently only a small fraction of the total time interval will be spent on time-stepping with the small time step k. To see this, define the *cost* as

$$\alpha = \frac{1+m}{K+km} \in (1/K, 1/k),$$

that is, the number of time steps per unit time interval. The classical stability analysis with $m = 0$ gives

$$\alpha = 1/k_n = \lambda/2, \tag{2.4}$$

with a maximum time step $k_n = K = 2/\lambda$. Using (2.3) we instead find

$$\alpha \approx \frac{1+\log(K\lambda)/c}{K+\log(K\lambda)/\lambda} \approx \frac{\lambda}{c}\log(K\lambda)/(K\lambda) \ll \lambda/c, \tag{2.5}$$

for $K\lambda \gg 1$. The cost is thus decreased by the cost reduction factor

$$\frac{2\log(K\lambda)}{cK\lambda} \sim \frac{\log(K\lambda)}{K\lambda},$$

which can be quite significant for large values of $K\lambda$.

A similar analysis applies to the system $\dot{u} + Au = 0$, if the eigenvalues $\{\lambda_i\}_{i=1}^N$ of A are separated with $0 \le \lambda_1 \le \ldots \le \lambda_{i-1} \le 2/K$ and $2/K \ll \lambda_i \le \ldots \le \lambda_N$. In this case, the cost is decreased by a factor

$$\frac{2\log(K\lambda_i)}{cK\lambda_i} \sim \frac{\log(K\lambda_i)}{K\lambda_i}. \tag{2.6}$$

In recent independent work by Gear and Kevredikis [2], a similar idea of combining small and large time steps for a class of stiff problems with a clear separation of slow and fast time scales, is developed. This work however, is not focused on adaptivity to the same extent as ours, which does not require any a priori information about the nature of the stiffness (for example distribution of eigenvalues).

3. Parabolic problems

For a parabolic system,

$$\begin{aligned} \dot{u}(t) + Au(t) &= 0 \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{3.1}$$

with A a symmetric positive semidefinite $N \times N$ matrix and $u^0 \in \mathbb{R}^N$ a given initial condition, the basic damping strategy outlined in the previous section may fail to be efficient. This can be seen directly from (2.6); for efficiency we need $K\lambda_i \gg 2$, but with the eigenvalues of A distributed over the interval $[0, \lambda_N]$, for example with $\lambda_i \sim i^2$ as for a typical (one-dimensional) parabolic problem, one cannot have both $\lambda_{i-1} \le 2/K$ and $K\lambda_i \gg 2$!

We conclude that for a parabolic problem, the sequence of time steps, or equivalently the polynomial $P(x)$, has to be chosen differently. We thus seek a more general sequence $k_1, \ldots, k_m$ of time steps such that $|P(x)| \le 1$ for $x \in [0, K\lambda_N]$, with

$$P(x) = \left(1 - \frac{k_1 x}{K}\right) \cdots \left(1 - \frac{k_m x}{K}\right),$$

and K a given maximum step size.

3.1. **Chebyshev damping.** A good candidate for P is given by the shifted Chebyshev polynomial of degree m,

$$P_c(x) = T_m\left(1 - \frac{2x}{K\lambda_N}\right).$$

This gives $P_c(0) = 1$ and $|P_c(x)| \leq 1$ on $[0, K\lambda_N]$, see Figure 1. A similar approach is taken in [10].

Analyzing the cost as before, we have $\alpha = m/(k_1 + \ldots + k_m)$, with

$$k_i = 2/(\lambda_N(1 - s_{m+1-i})),$$

and s_i the ith zero of the Chebyshev polynomial $T_m = T_m(s)$, given by $s_i = \cos((2i-1)\pi/(2m))$, $i = 1, \ldots, m$. It follows that

$$\alpha = \frac{m\lambda_N/2}{1/(1-s_1) + \ldots + 1/(1-s_m)} = \frac{m\lambda_N/2}{m^2} = \lambda_N/2m.$$

The reduction in cost compared to $\lambda_N/2$ is thus a factor $1/m$. A restriction on the maximum value of m is given by $k_m = 2/(\lambda_N(1-s_1)) \leq K$, that is,

$$K \geq \frac{2}{\lambda_N(1 - \cos(\pi/(2m)))} \approx \frac{2}{\lambda_N \pi^2/(8m^2)} = 16m^2/(\lambda_N \pi^2),$$

for $m \gg 1$. With $m = (\pi/4)\sqrt{K\lambda_N}$, the cost reduction factor for Chebyshev damping is thus given by

$$1/m = \frac{4}{\pi\sqrt{K\lambda_N}} \sim (K\lambda_N)^{-1/2}.$$

3.2. **Dyadic damping.** Another approach, more closely related to the original approach of alternating large and small time steps, is to take

$$P_d(x) = \prod_{j=q+1}^{p}\left(1 - \frac{2^j x}{K\lambda_N}\right)\prod_{i=0}^{q}\left(1 - \frac{2^i x}{K\lambda_N}\right)^{2^{q-i}},$$

where p and $q \leq p$ are two integers to be determined. This gives a sequence increasing time steps, starting with 2^q time steps of size $k_0 = 1/\lambda_N$, 2^{q-1} time steps of size $k_1 = 2k_0$ and so on, until we reach level q where we take one time step of size $k_q = 2^q k_0$. We then continue to increase the size of the time step, with only one time step at each level, until we reach $k_p = 2^p k_0 = K$.

Determining the minimal value of q for $|P_d(x)| \leq 1$ on $[0, K\lambda_N]$ for $p = 0, 1, \ldots$, we find $q(p) \approx \frac{2}{3}(p-1)$, see Table 1. The cost is now given by

$$\begin{aligned}\alpha &= \frac{(2^q + \ldots + 1) + (p-q)}{\frac{1}{\lambda_N}(2^q(q+1) + (2^{q+1} + \ldots + 2^p))} = \frac{\lambda_N((2^{q+1}-1) + (p-q))}{2^q(q+1) + (2^{p+1} - 2^{q+1})} \\ &\approx \frac{\lambda_N/2}{2^{p-q-1}} \approx \frac{\lambda_N}{2}2^{-p+2(p-1)/3+1} = \frac{\lambda_N}{2}2^{-(p-1)/3}.\end{aligned}$$

Since $p = \log(K\lambda_N)/\log(2)$, the reduction in cost for dyadic damping is then a factor

$$\frac{1}{(K\lambda_N/2)^{1/3}} \sim (K\lambda_N)^{-1/3},$$

which is competitive with the optimal cost reduction of Chebyshev damping.

p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
q	0	0	0	1	2	3	3	4	4	5	6	7	8	8	9	10	10

TABLE 1. Number of required levels with multiple zeros, q, as function of the total number of levels, p.

4. Comparison of methods

We summarize our findings so far as follows. The reduction in cost for the simple approach of large unstable time steps followed by small stabilizing time steps, outlined in Section 2, is a factor $\log(K\lambda_N)/(K\lambda_N)$, and thus this simple approach can be much more efficient than both the two approaches, Chebyshev damping and dyadic damping, discussed in the previous section. This however requires that the problem in question has a gap in its eigenvalue spectrum, that is, we have a clear separation into small (stable) eigenvalues and large (unstable) eigenvalues.

In the case of a parabolic problem, without a gap in the eigenvalue spectrum, the simple approach of Section 2 will not work. In this case, Chebyshev damping and dyadic damping give a reduction in cost which is a factor $(K\lambda_N)^{-1/2}$ or $(K\lambda_N)^{-1/3}$, respectively. The efficiency for a parabolic problem is thus comparable for the two methods. Since the method of dyadic damping is a slightly modified version of the simple approach of Section 2, consisting of gradually and carefully increasing the time step size after stabilization with the smallest time step, thus being simple and flexible, we believe this method to be advantageous over the Chebyshev method.

As a comparison, we plot in Figure 1 the shifted Chebyshev polynomial $P_c(x)$ and the polynomial $P_d(x)$ of dyadic damping for $K\lambda_N = 64$. We also plot in Figure 2 the stability regions for the two polynomials $P_c(z)$ and $P_d(z)$ for $z \in \mathbb{C}$. In this context, the two polynomials are given by

$$P_c(z) = \prod_{i=1}^{m} \left(1 + \frac{1 - s_1}{1 - s_{m+1-i}} z\right), \tag{4.1}$$

for $s_i = \cos((2i-1)\pi/(2m))$, and

$$P_d(z) = \prod_{j=q+1}^{p} \left(1 + \frac{2^j z}{2^p}\right) \prod_{i=0}^{q} \left(1 + \frac{2^i z}{2^p}\right)^{2^{q-i}}. \tag{4.2}$$

As can be seen from this figure, another advantage of the method of dyadic damping as compared to the method of Chebyshev damping, is the larger stability region in the complex plane. We thus need not assume that the eigenvalues of A in (3.1) lie in a narrow strip along the negative axis in the complex plane, as is needed with Chebyshev damping. (The stability region of the Chebyshev polynomial can be slightly modified to include a narrow strip along the negative real axis, see [10].) When comparing the size of the stability regions to the results presented in [10], it should be noted that here $z = -K\lambda$, with K the maximum time step in a sequence of time steps, rather than the sum of the time steps in the sequence. (The difference for Chebyshev damping is approximately a factor $2\pi^2/16 \approx 1.2$.)

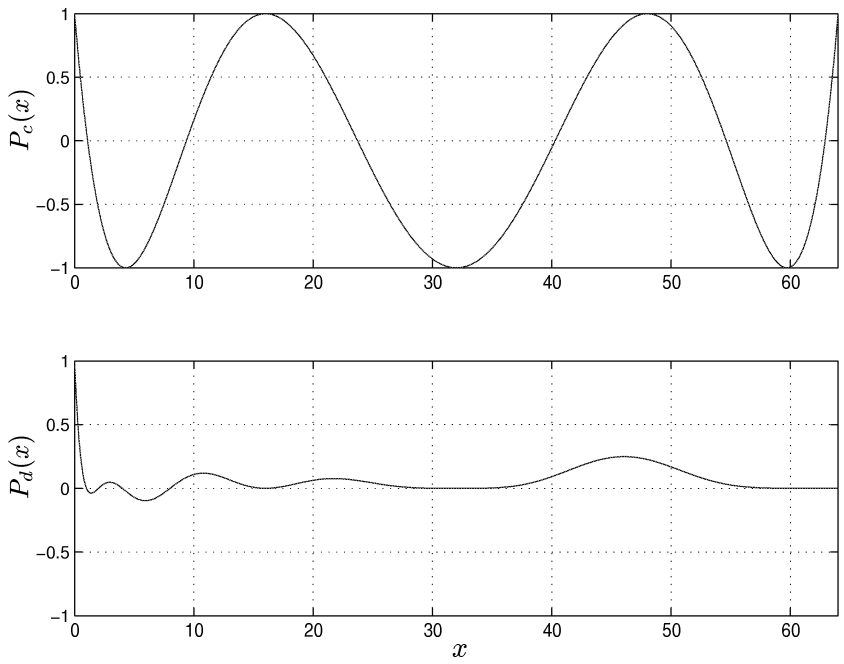

FIGURE 1. A comparison of the two polynomials for $K\lambda_N = 64$: we take $m = 6$ for the shifted Chebyshev polynomial $P_c(x)$ and $(p, q) = (6, 3)$ for the dyadic polynomial $P_d(x)$. With $K = 1$, the costs are 5.3 and 8, respectively.

5. The general nonlinear problem

We consider now the general nonlinear problem,

$$\begin{aligned} \dot{u}(t) &= f(u(t)) \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{5.1}$$

where $f : \mathbb{R}^N \times (0, T] \to \mathbb{R}^N$ is a given bounded and differentiable function. The explicit Euler method for (5.1) reads

$$U^n = U^{n-1} + k_n f(U^{n-1}), \tag{5.2}$$

where the function $U = U(t)$ is piecewise constant and right-continuous with $U^n = U(t_n^+)$. The exact solution u satisfies a similar relation,

$$u^n = u^{n-1} + \int_{t_{n-1}}^{t_n} f(u(t))\, dt, \tag{5.3}$$

with $u^n = u(t_n)$. Subtracting (5.2) and (5.3), we find that the error $e(t) = U(t) - u(t)$ satisfies

$$e(t_n^+) - e(t_{n-1}^+) = \int_{t_{n-1}}^{t_n} J(U(t) - u(t))\, dt = \int_{t_{n-1}}^{t_n} Je\, dt,$$

where J is the Jacobian $\frac{\partial f}{\partial u}$ of the right-hand side evaluated at a mean value of U and u. We conclude that the efficiency of the proposed method for the general

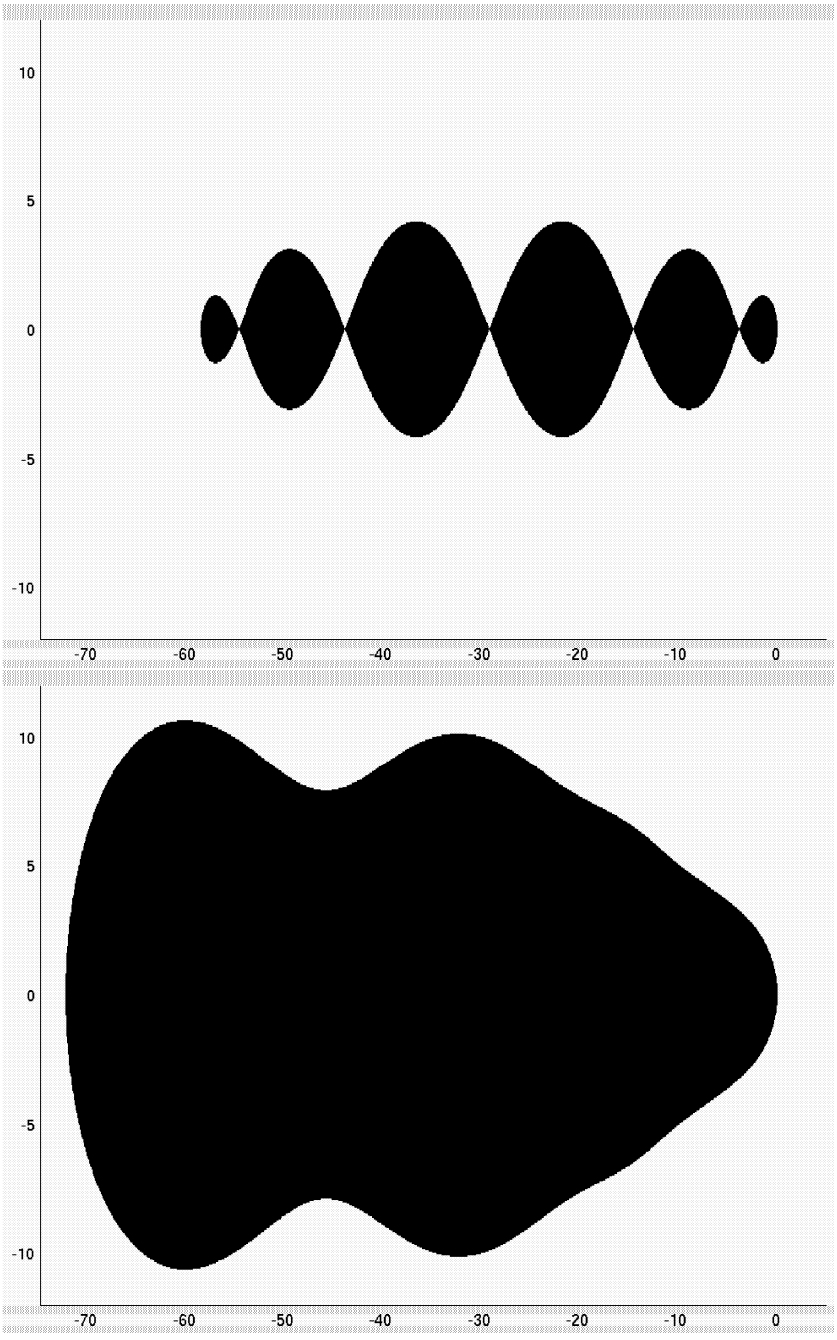

FIGURE 2. A comparison of stability regions for the two polynomials, with $m = 6$ for the shifted Chebyshev polynomial $P_c(z)$ and $(p, q) = (6, 3)$ for the dyadic polynomial $P_d(z)$.

nonlinear problem will be determined by the distribution of the eigenvalues of the Jacobian.

6. Iterative methods

From another viewpoint, we may consider using an explicit-type iterative method for solving the discrete equations arising from an implicit method. The implicit cG(1) method with midpoint quadrature for the general nonlinear problem (5.1) reads

$$U^n = U^{n-1} + k_n f\left(\frac{U^{n-1} + U^n}{2}\right). \tag{6.1}$$

We can solve this system of nonlinear equations for U^n using Newton's method, but the simplest and cheapest method is to apply fixed point iteration directly to (6.1), that is,

$$U^{nl} = U^{n-1} + k_n f\left(\frac{U^{n-1} + U^{n,l-1}}{2}\right),$$

for $l = 1, 2, \ldots$ until convergence with $U^{n,0} = U^{n-1}$. The fixed point iteration converges for small enough time step k_n, and so the stability condition for a standard explicit method appears also in explicit-type iterative methods as a condition for convergence of the iterative solution of the implicit equations.

To determine a stop criterion for the fixed point iteration, we measure the size of the *discrete residual*,

$$r^{nl} = \frac{1}{k_n}(U^{nl} - U^{n-1}) - f\left(\frac{U^{n-1} + U^{nl}}{2}\right),$$

which should be zero for the true cG(1) approximation. We continue the iterations until the discrete residual is smaller than some tolerance $\text{tol} > 0$. Usually only a couple of iterations are needed. Estimating the error $e(T) = U(T) - u(T)$ at final time T (see [7]), we have

$$\|e(T)\| \leq S(T) \max_{[0,T]} k\|R\| + S^0(T) \max_{[0,T]} \|r\|,$$

where $R(t) = \dot{U}(t) - f(U(t))$ is the *continuous residual* and $S(T)$ and $S^0(T)$ are stability factors. For the test equation we have $S(T) \leq 1$ and $S^0(T) \leq 1/\lambda$, which suggests that for a typical stiff problem we can take $\text{tol} = \text{TOL}$, where TOL is a tolerance for the error $e(T)$ at final time.

For the discrete residual, we have

$$\begin{aligned} r^{nl} &= \frac{1}{k_n}(U^{nl} - U^{n-1}) - f\left(\frac{U^{n-1} + U^{nl}}{2}\right) \\ &= f\left(\frac{U^{n-1} + U^{n,l-1}}{2}\right) - f\left(\frac{U^{n-1} + U^{nl}}{2}\right) = J\frac{U^{n,l-1} - U^{nl}}{2} \\ &= \frac{1}{2}J\left[U^{n,l-1} - U^{n-1} - k_n f\left(\frac{U^{n-1} + U^{n,l-1}}{2}\right)\right], \end{aligned}$$

which gives

$$r^{nl} = \frac{k_n}{2} J r^{n,l-1}. \tag{6.2}$$

Thus, by measuring the size of the discrete residual, we obtain information about the stiff nature of the problem, in particular the eigenvalue of the current unstable eigenmode, which can be used in an adaptive algorithm targeted precisely at stabilizing the current unstable eigenmode. We discuss this in more detail below in Section 8.

7. Multi-adaptive solvers

In a multi-adaptive solver we use individual time steps for different components. An important part of the algorithm described in [7, 8] is the iterative fixed point solution of the discrete equations on time slabs. The simple strategy to take small damping steps to stabilize the system applies also in the multi-adaptive setting, where we may also target individual eigenmodes (if these are represented by different components) using individual damping steps. This will be explored further in another note.

8. An adaptive algorithm

The question is now whether we can choose the time step sequence automatically in an adaptive algorithm. We approach this question in the setting of an implicit method combined with an explicit-type iterative solver as in Section 6. We give the algorithm in the case of the simple damping strategy outlined in Section 2, with extension to parabolic problems as described in Section 3.

A simple adaptive algorithm for the standard cG(1) method with iterative solution of the discrete equations reads as follows.

(1) Determine a suitable initial time step k_1 and solve the discrete equations for the solution $U(t)$ on (t_0, t_1).
(2) Repeat on (t_{n-1}, t_n) for $n = 2, 3, \ldots$ until $t_n \geq T$:
    (a) Evaluate the continuous residual R_{n-1} from the previous time interval.
    (b) Determine the new time step k_n based on R_{n-1}.
    (c) Solve the discrete equations on (t_{n-1}, t_n) using fixed point iteration.

In reality we want to control the global error, which means we have to solve also the dual problem, compute stability factors (or weights), evaluate an a posteriori error estimate and possibly repeat the process until the error is below a given tolerance $\mathrm{TOL} > 0$. The full algorithm is thus slightly more elaborate, but the basic algorithm presented here is the central part. See [1] for a discussion.

We comment also on step (2.b): For the cG(1) method we would like to take $k_n = \mathrm{TOL}/(S(T)\|R_{n-1}\|)$, but this introduces oscillations in the size of the time step. A small residual gives a large time step which results in a large residual, and so on. To avoid this, the simple step size selection has to be combined with a regulator of some kind, see [3, 9] or [8]. It turns out that a simple strategy that works well in many situations is to take k_n as the geometric mean value

$$k_n = \frac{2\tilde{k}_n k_{n-1}}{\tilde{k}_n + k_{n-1}}, \tag{8.1}$$

where $\tilde{k}_n = \mathrm{TOL}/(S(T)\|R_{n-1}\|)$.

Now, for a stiff problem, what may go wrong is step (2.c); if the time step k_n is too large, the fixed point iterations will not converge. To be able to handle stiff problems using the technique discussed above, we propose the following simple modification of the adaptive algorithm.

(1) Determine a suitable initial time step k_1 and solve the discrete equations for the solution $U(t)$ on (t_0, t_1).
(2) Repeat on (t_{n-1}, t_n) for $n = 2, 3, \ldots$ until $t_n \geq T$:
   (a) Evaluate the continuous residual R_{n-1} from the previous time interval.
   (b) Determine the new time step k_n based on R_{n-1}.
   (c) Solve the discrete equations on (t_{n-1}, t_n) using fixed point iteration.
   (d) If (2.c) didn't work, compute

$$L = \frac{2}{k_n} \frac{||r^l||}{||r^{l-1}||},$$

   and take $m = \log(k_n L)$ explicit Euler steps with time step $k = c/L$ and c close to 1.
   (e) Try again starting at (2.a) with $n \to n + m$.

In the analysis of Section 2 we had $c = 1/2$, but it is clear that the damping steps will be more efficient if we have c close to 1. An implementation of this algorithm in the form of a simple MATLAB code is available for inspection [6], including among others the test problems presented in the next section.

We also note that by (8.1), we have $k_n \leq 2k_{n-1}$, and so following the sequence of small stabilizing time steps, the size of the time step will be increased gradually, doubling the time step until we reach $k_n \sim K$ or the system becomes unstable again, whichever comes first. This automatically gives a sequence of time steps similar to that of dyadic damping described in Section 3, with the difference that most of the damping is made with the smallest time step.

9. Examples

To illustrate the technique, we take a simple standard implementation of the cG(1)-method (with explicit fixed point solution of the discrete equations) and add a couple of lines to handle the stabilization of stiff problems. We try this code on a number of well-known stiff problems taken from the ODE literature, and conclude that we are able to handle stiff problems with this explicit code.

When referring to the cost α below, this includes also the number of fixed point iterations needed to compute the cG(1) solution on intervals where the iterations converge. This is compared to the cost α_0 for the standard cG(1) method in which we are forced to take small time steps all the time. (These small time steps are marked by dashed lines.) For all example problems below we report both the cost α and the cost reduction factor α/α_0.

9.1. The test equation.

The first problem we try is the test equation,

$$\begin{aligned} \dot{u}(t) + \lambda u(t) &= 0 \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{9.1}$$

on $[0, 10]$, where we choose $u^0 = 1$ and $\lambda = 1000$. As is evident from Figure 3, the time step sequence is automatically chosen in agreement with the previous discussion. The cost is only $\alpha \approx 6$ and the cost reduction factor is $\alpha/\alpha_0 \approx 1/310$.

Note how the time steps are drastically decreased (step (2.d) in the adaptive algorithm) when the system needs to be stabilized, and then gradually increased until again the stabilization is needed.

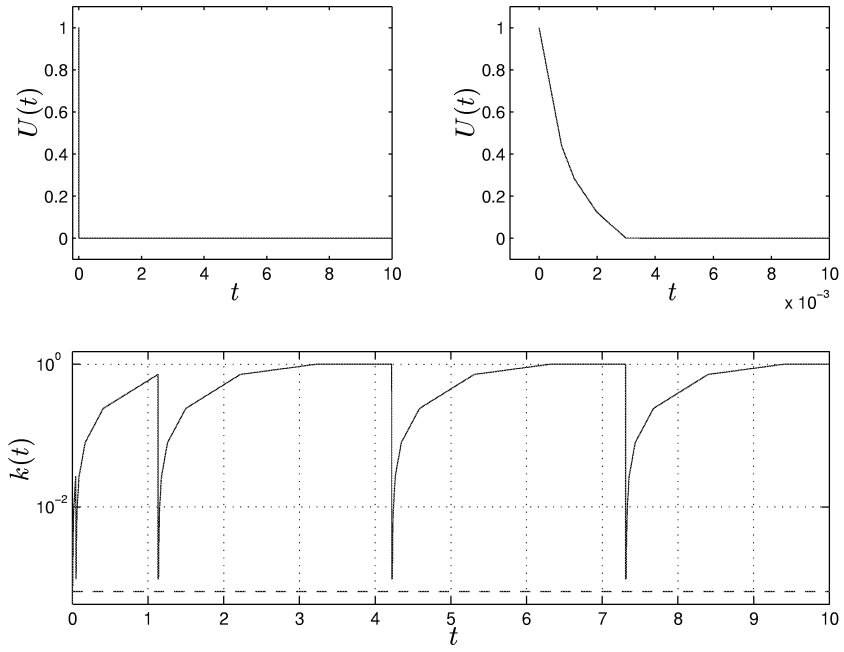

FIGURE 3. Solution and time step sequence for eq. (9.1), $\alpha/\alpha_0 \approx 1/310$.

9.2. **The test system.** For the test system,

$$\begin{aligned} \dot{u}(t) + Au(t) &= 0 \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{9.2}$$

on $[0, 10]$, we take $A = \text{diag}(100, 1000)$ and $u^0 = (1, 1)$. There are now two eigenmodes with large eigenvalues that need to be damped out. The dominant eigenvalue of A is $\lambda_2 = 1000$ and most of the damping steps are chosen to damp out this eigenmode, but some of the damping steps are chosen based on the second largest eigenvalue $\lambda_1 = 100$. When to damp out which eigenmode is automatically decided by the adaptive algorithm; the bad eigenmode that needs to be damped out becomes visible in the iterative solution process. Since there is an additional eigenvalue, the cost is somewhat larger than for the scalar test problem, $\alpha \approx 18$, giving a cost reduction factor of $\alpha/\alpha_0 \approx 1/104$.

9.3. **A linear non-normal problem.** The method behaves similarly even if we make the matrix A highly non-normal. We now solve

$$\begin{aligned} \dot{u}(t) + Au(t) &= 0 \quad \text{for } t > 0, \\ u(0) &= u^0, \end{aligned} \tag{9.3}$$

on $[0, 10]$, with

$$A = \begin{bmatrix} 1000 & -10000 \\ 0 & 100 \end{bmatrix},$$

and $u^0 = (1, 1)$. The cost is about the same as for the previous problem, $\alpha \approx 17$, but the cost reduction factor is better: $\alpha/\alpha_0 \approx 1/180$.

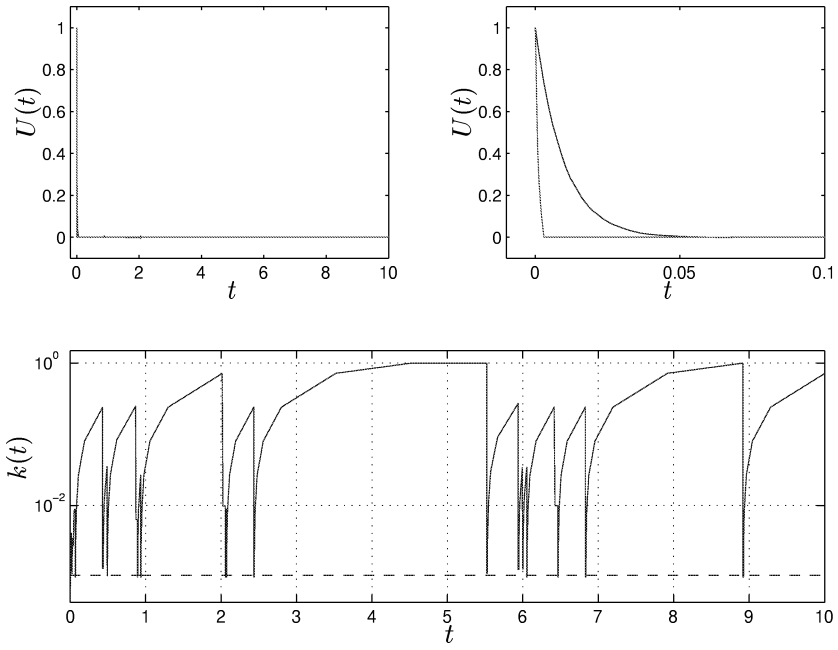

FIGURE 4. Solution and time step sequence for eq. (9.2), $\alpha/\alpha_0 \approx 1/104$.

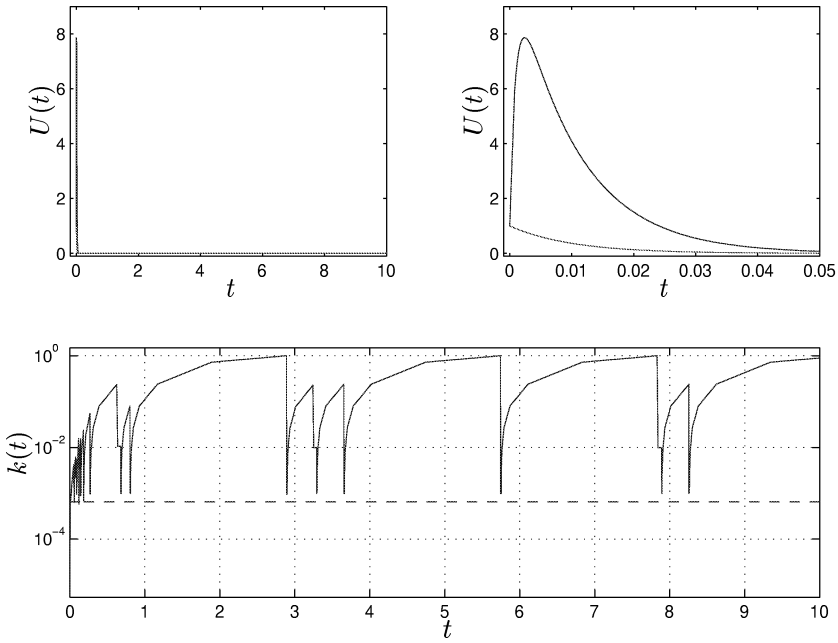

FIGURE 5. Solution and time step sequence for eq. (9.3), $\alpha/\alpha_0 \approx 1/180$.

9.4. **The HIRES problem.** The so-called HIRES problem ("High Irradiance RESponse") originates from plant physiology and is taken from the test set of ODE problems compiled by Lioen and de Swart [5]. The problem consists of the following eight equations:

$$
\left\{
\begin{array}{rcl}
\dot{u}_1 &=& -1.71u_1 + 0.43u_2 + 8.32u_3 + 0.0007,\\
\dot{u}_2 &=& 1.71u_1 - 8.75u_2,\\
\dot{u}_3 &=& -10.03u_3 + 0.43u_4 + 0.035u_5,\\
\dot{u}_4 &=& 8.32u_2 + 1.71u_3 - 1.12u_4,\\
\dot{u}_5 &=& -1.745u_5 + 0.43u_6 + 0.43u_7,\\
\dot{u}_6 &=& -280.0u_6u_8 + 0.69u_4 + 1.71u_5 - 0.43u_6 + 0.69u_7,\\
\dot{u}_7 &=& 280.0u_6u_8 - 1.81u_7,\\
\dot{u}_8 &=& -280.0u_6u_8 + 1.81u_7,
\end{array}
\right. \tag{9.4}
$$

together with the initial condition $u^0 = (1.0, 0, 0, 0, 0, 0, 0, 0.0057)$. We integrate over $[0, 321.8122]$ (as specified in [5]) and present the solution and the time step sequence in Figure 6. The cost is now $\alpha \approx 8$ and the cost reduction factor is $\alpha/\alpha_0 \approx 1/33$.

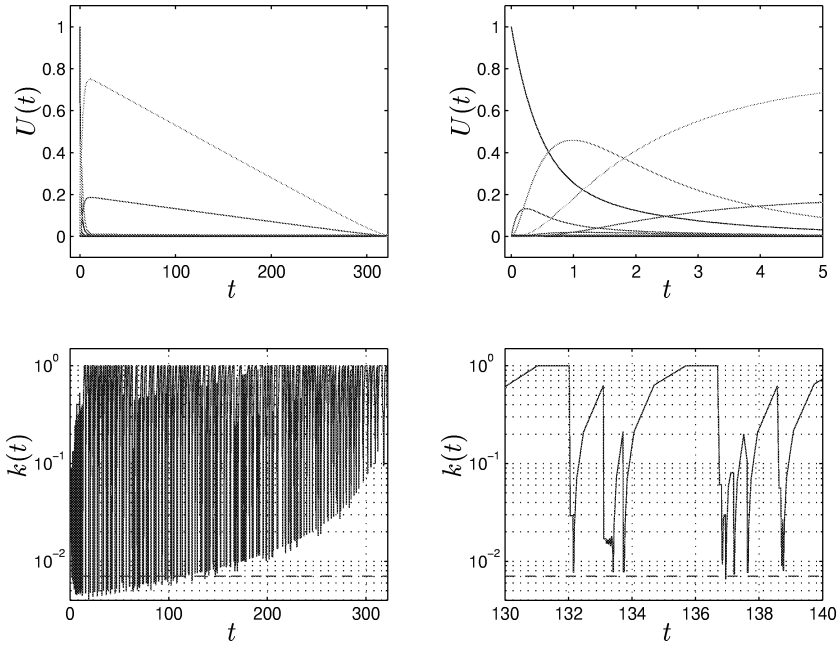

FIGURE 6. Solution and time step sequence for eq. (9.4), $\alpha/\alpha_0 \approx 1/33$.

9.5. **The Akzo-Nobel problem.** The next problem is a version of the "Chemical Akzo-Nobel" problem taken from the ODE test set [5], consisting of the following

six equations:

$$
\left\{
\begin{array}{rcl}
\dot{u}_1 &=& -2r_1 + r_2 - r_3 - r_4, \\
\dot{u}_2 &=& -0.5r_1 - r_4 - 0.5r_5 + F, \\
\dot{u}_3 &=& r_1 - r_2 + r_3, \\
\dot{u}_4 &=& -r_2 + r_3 - 2r_4, \\
\dot{u}_5 &=& r_2 - r_3 + r_5, \\
\dot{u}_6 &=& -r_5,
\end{array}
\right. \tag{9.5}
$$

where $F = 3.3 \cdot (0.9/737 - u_2)$ and the reaction rates are given by $r_1 = 18.7 \cdot u_1^4\sqrt{u_2}$, $r_2 = 0.58 \cdot u_3 u_4$, $r_3 = 0.58/34.4 \cdot u_1 u_5$, $r_4 = 0.09 \cdot u_1 u_4^2$ and $r_5 = 0.42 \cdot u_6^2\sqrt{u_2}$. We integrate over the interval $[0, 180]$ with initial condition $u^0 = (0.437, 0.00123, 0, 0, 0, 0.367)$. Allowing a maximum time step of $k_{\max} = 1$ (chosen arbitrarily), the cost is only $\alpha \approx 2$ and the cost reduction factor is $\alpha/\alpha_0 \approx 1/9$. The actual gain in a specific situation is determined by the quotient between the large time steps and the small damping time steps, as well as the number of small damping steps that are needed. In this case, the number of small damping steps is small, but the large time steps are not very large compared to the small damping steps. The gain is thus determined both by the stiff nature of the problem and the tolerance (or the size of the maximum allowed time step).

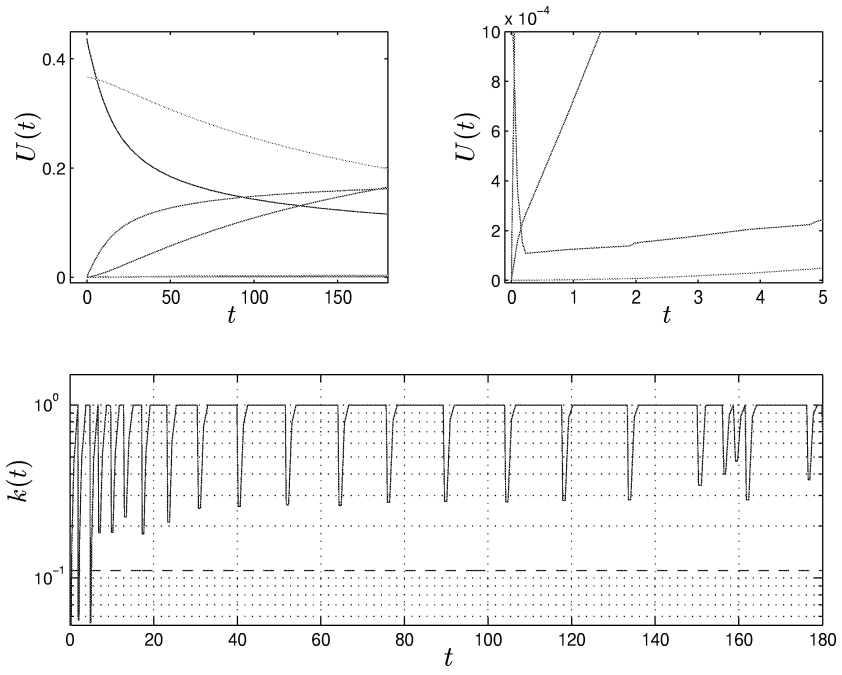

FIGURE 7. Solution and time step sequence for eq. (9.5), $\alpha/\alpha_0 \approx 1/9$.

9.6. **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 as

$$\begin{cases} \dot{u}_1 &= u_2, \\ \dot{u}_2 &= -\mu(u_1^2 - 1)u_2 - u_1. \end{cases} \tag{9.6}$$

We take $\mu = 1000$ and compute the solution on the interval $[0, 10]$ with initial condition $u^0 = (2, 0)$. The time step sequence behaves as desired with only a small portion of the time interval spent on taking small damping steps. The cost is now $\alpha \approx 140$ and the cost reduction factor is $\alpha/\alpha_0 \approx 1/75$.

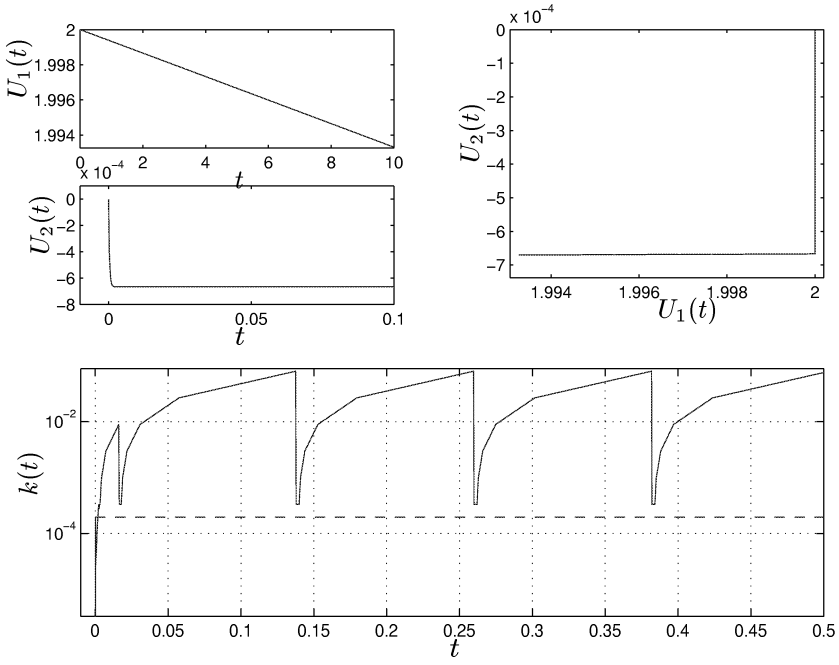

FIGURE 8. Solution and time step sequence for eq. (9.6), $\alpha/\alpha_0 \approx 1/75$.

9.7. **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

$$\begin{aligned} \dot{u}(t) + Au(t) &= f, \\ u(0) &= 0, \end{aligned} \tag{9.7}$$

where A is the *stiffness matrix*. With a spatial resolution of $h = 0.01$, the eigenvalues of A are distributed in the interval $[0, 4 \cdot 10^4]$ (see Figure 9).

Using the technique of dyadic damping described in Section 3 for this parabolic problem, the cost is $\alpha \approx 2000$, with a cost reduction factor of $\alpha/\alpha_0 \approx 1/31$.

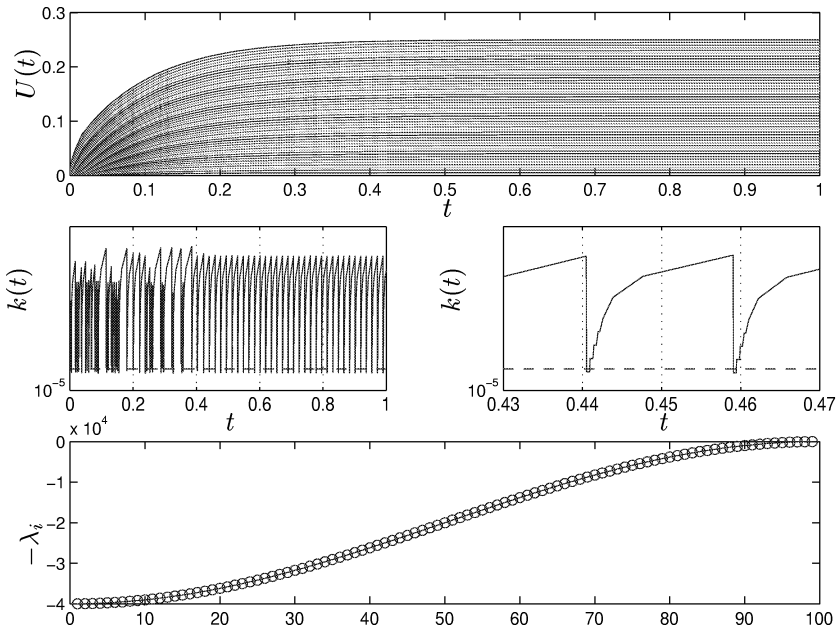

FIGURE 9. Solution and time step sequence for eq. (9.7), $\alpha/\alpha_0 \approx 1/31$. Note that the time step sequence in this example is chosen slightly differently than in the previous examples, using the technique of dyadic damping discussed in Section 3. This is evident upon close inspection of the time steps sequence.

9.8. **A non-stiff problem.** To show that the method works equally well for non-stiff problems, we finally consider the following simple system:

$$\begin{cases} \dot{u}_1 &= 5u_2, \\ \dot{u}_2 &= -u_1. \end{cases} \tag{9.8}$$

With the initial condition $u^0 = (0, 1)$, the solution is $u(t) = (\sqrt{5}\sin(\sqrt{5}t), \cos(\sqrt{5}))$. Since this problem is non-stiff (for reasonable tolerances), no stabilization is needed and so the solver works as a standard non-stiff cG(1) solver with no overhead. This is also evident from the time step sequence (see Figure 9) which is chosen only to match the size of the (continuous) residual.

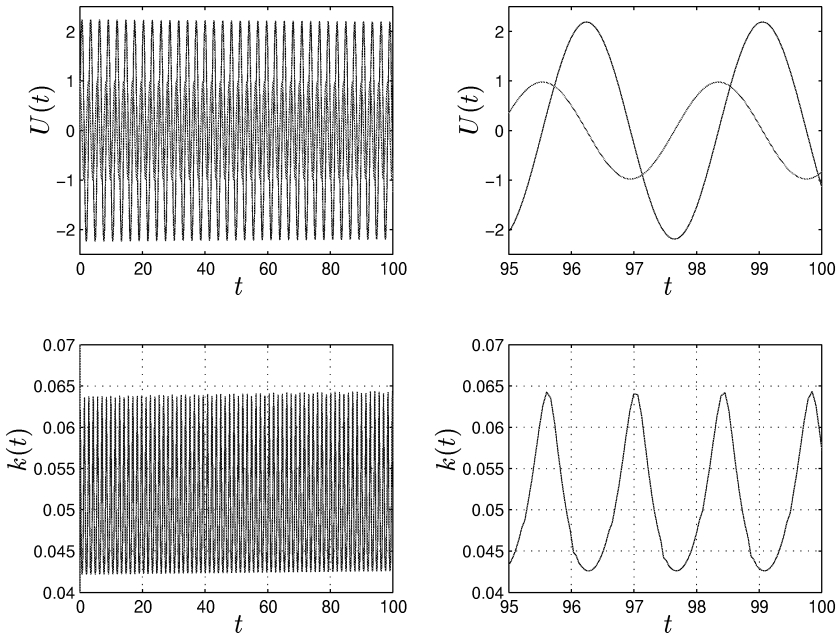

FIGURE 10. Solution and time step sequence for eq. (9.8), $\alpha/\alpha_0 = 1$.

REFERENCES

[1] K. ERIKSSON, D. ESTEP, P. HANSBO, AND C. JOHNSON, *Introduction to adaptive methods for differential equations*, Acta Numerica, (1995), pp. 105–158.

[2] C. W. GEAR AND I. G. KEVREKIDIS, *Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum*, Submitted to SIAM J. Sci. Comp., (2001).

[3] K. GUSTAFSSON, M. LUNDH, AND G. SÖDERLIND, *A pi stepsize control for the numerical solution of ordinary differential equations*, BIT, 28 (1988), pp. 270–287.

[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] W. LIOEN AND H. DE SWART, *Test set for initial value problem solvers*, CWI Report MAS–R9832, ISSN 1386–3703, Amsterdam, (1998).

[6] A. LOGG, *Chalmers finite element center software*, `www.phi.chalmers.se/software/`, (2002).

[7] ———, *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., (2003). To appear.

[9] G. SÖDERLIND, *The automatic control of numerical integration*, CWI Quarterly, 11 (1998), pp. 55–74.

[10] J. VERWER, *Explicit runge-kutta methods for parabolic partial differential equations*, Appl. Numer. Math., 22 (1996), pp. 359–379.