Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Applied Numerical Mathematics 48 (2004) 339–354
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

Multi-adaptive time integration

Anders Logg

Department of Computational Mathematics, Chalmers University of Technology, Göteborg, Sweden

Abstract

Time integration of ODEs or time-dependent PDEs with required resolution of the fastest time scales of the system, can be very costly if the system exhibits multiple time scales of different magnitudes. If the different time scales are localised to different components, corresponding to localisation in space for a PDE, efficient time integration thus requires that we use different time steps for different components.

We present an overview of the multi-adaptive Galerkin methods mcG(q) and mdG(q) recently introduced in a series of papers by the author. In these methods, the time step sequence is selected individually and adaptively for each component, based on an a posteriori error estimate of the global error.

The multi-adaptive methods require the solution of large systems of nonlinear algebraic equations which are solved using explicit-type iterative solvers (fixed point iteration). If the system is stiff, these iterations may fail to converge, corresponding to the well-known fact that standard explicit methods are inefficient for stiff systems. To resolve this problem, we present an adaptive strategy for explicit time integration of stiff ODEs, in which the explicit method is adaptively stabilised by a small number of small, stabilising time steps.

Keywords: Multi-adaptivity; Error control; Adaptivity; Explicit; Stiffness

1. Introduction

In earlier work [29,30], we have introduced the multi-adaptive Galerkin methods mcG(q) and mdG(q) for ODEs of the type

$$\begin{cases} \dot{u}(t) = f\big(u(t), t\big), \quad t \in (0, T], \\ u(0) = u_0, \end{cases} \tag{1}$$

where $u : [0, T] \to \mathbb{R}^N$ is the solution to be computed, $u_0 \in \mathbb{R}^N$ a given initial condition, $T > 0$ 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.

E-mail address: logg@math.chalmers.se (A. Logg).
URL: http://www.math.chalmers.se/~logg.

0168-9274/$30.00
doi:10.1016/j.apnum.2003.11.004

We use the term *multi-adaptivity* to describe methods with individual time-stepping for the different components $u_i(t)$ of the solution vector $u(t) = (u_i(t))$, including (i) time step length, (ii) order, and (iii) quadrature, all chosen adaptively in a computational feed-back process.

Surprisingly, individual time-stepping for ODEs has received little attention in the large literature on numerical methods for ODEs, see, e.g., [4,24,25,2,34]. For specific applications, such as the n-body problem, methods with individual time-stepping have been used, see, e.g., [31,1,5], but a general methodology has been lacking. For time-dependent PDEs, in particular for conservation laws of the type $\dot{u} + f(u)_x = 0$, attempts have been made to construct methods with individual (locally varying in space) time steps. Flaherty et al. [22] have constructed a method based on the discontinuous Galerkin method combined with local explicit Euler time-stepping. A similar approach is taken in [6] where a method based on the original work by Osher and Sanders [32] is presented for conservation laws in one and two space dimensions. Typically the time steps used are based on local CFL conditions rather than error estimates for the global error and the methods are low order in time (meaning $\leqslant 2$). We believe that our work on multi-adaptive Galerkin methods (including error estimation and arbitrary order methods) presents a general methodology to individual time-stepping, which will result in efficient integrators both for ODEs and time-dependent PDEs.

The multi-adaptive methods are developed within the general framework of adaptive Galerkin methods based on piecewise polynomial approximation (finite element methods) for differential equations, including the continuous and discontinuous Galerkin methods cG(q) and dG(q), which we extend to their multi-adaptive analogues mcG(q) and mdG(q). Earlier work on adaptive error control for the cG(q) and dG(q) methods include [7,17,27,19,18,21]. See also [10,11,9,12–14], and [8] or [20] in particular for an overview of adaptive error control based on duality techniques. The approach to error analysis and adaptivity presented in these references naturally carries over to the multi-adaptive methods.

1.1. The stiffness problem

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 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, in particular outside transients, and the advantage of a low cost per time step for the explicit method is counter-balanced by the necessity of taking a large number of small time steps. As a result, the overall efficiency of an explicit method for a stiff ODE is small.

We encounter the same problem when we try to use explicit fixed point iteration to solve the discrete equations given by the multi-adaptive Galerkin methods mcG(q) and mdG(q). However, it turns out that if a sequence of large (unstable) time steps are accompanied by a suitable (small) number of small time steps, a stiff system can be stabilised to allow integration with an effective time step much larger than the largest stable time step given by classical stability analysis. This idea of stabilising a stiff system using the inherent damping property of the stiff system itself was first developed in an automatic and adaptive setting in [16], and will be further explored in the full multi-adaptive setting. A similar approach is taken in recent independent work by Gear and Kevrekidis [3]. The relation to Runge–Kutta methods based on Chebyshev polynomials discussed by Verwer in [26] should also be noted.

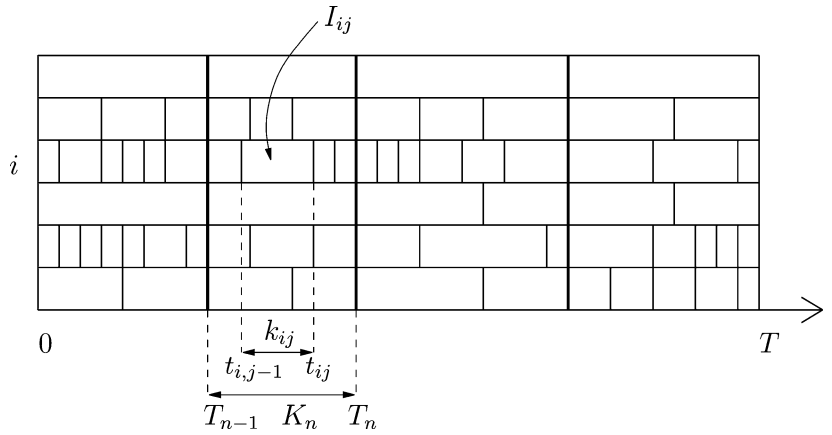

Fig. 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronised time levels are organised in time slabs. In this example, we have $N = 6$ and $M = 4$.

1.2. Notation

The following notation is used in the discussion of the multi-adaptive Galerkin methods below: Each component $U_i(t)$, $i = 1, \ldots, N$, of the approximate m(c/d)G(q) solution $U(t)$ of (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 Fig. 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 synchronised time levels $0 = T_0 < T_1 < \cdots < T_M = T$. We refer to the set of intervals $\mathcal{T}_n$ between two synchronised time levels T_{n-1} and T_n as a *time slab*: $\mathcal{T}_n = \{I_{ij}\colon\ T_{n-1} \leqslant t_{i,j-1} < t_{ij} \leqslant T_n\}$, and we denote the length of a time slab by $K_n = T_n - T_{n-1}$. The partition consisting of the entire collection of intervals is denoted by $\mathcal{T} = \bigcup \mathcal{T}_n$.

1.3. Outline

The outline of the paper is as follows: In Section 2, we formulate the multi-adaptive Galerkin methods mcG(q) and mdG(q). In Section 3, we discuss error control and adaptivity. In particular, we show how to choose the individual time steps based on an a posteriori error estimate for the global error. In Section 4, we give a quick overview of an iterative method (based on fixed point iteration) for the system of nonlinear discrete equations that needs to be solved on each time slab, and in Section 5 we describe a technique that can be used to stabilise the explicit fixed point iterations for stiff problems. Finally, in Section 6, we present a number of numerical examples chosen to illustrate both the potential of multi-adaptivity and the use of explicit fixed point iteration (or explicit time-stepping) for stiff problems.

2. Multi-adaptive Galerkin

2.1. *Multi-adaptive continuous Galerkin,* mcG(q)

To give the definition of the mcG(q) method, we define the *trial space* V and the *test space* W as

$$\begin{aligned} V &= \left\{v \in \left[\mathcal{C}\left([0,T]\right)\right]^N\colon\ v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}),\ j=1,\ldots,M_i,\ i=1,\ldots,N\right\},\\ W &= \left\{v\colon\ v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}-1}(I_{ij}),\ j=1,\ldots,M_i,\ i=1,\ldots,N\right\}, \end{aligned} \tag{2}$$

where $\mathcal{P}^q(I)$ denotes the linear space of polynomials of degree $q \geqslant 0$ on the interval I. In other words, V is the space of continuous piecewise polynomials of degree $q = q_i(t) = q_{ij}$, $t \in I_{ij}$ on the partition $\mathcal{T}$, and W is the space of (in general discontinuous) piecewise polynomials of degree $q-1$ on the same partition.

We define the mcG(q) method for (1) as follows: Find $U \in V$ with $U(0) = u_0$, such that

$$\int_0^T \left(\dot{U}, v\right) \mathrm{d}t = \int_0^T \left(f(U,\cdot), v\right) \mathrm{d}t \quad \forall v \in W, \tag{3}$$

where $(\cdot,\cdot)$ denotes the standard inner product in $\mathbb{R}^N$. If now for each local interval I_{ij} we take $v_n = 0$ when $n \neq i$ and $v_i(t) = 0$ when $t \notin I_{ij}$, we can rewrite the global problem (3) as a number of successive local problems for each component: For $i = 1,\ldots,N$, $j = 1,\ldots,M_i$, find $U_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$ with $U_i(t_{i,j-1})$ given from the previous time interval, such that

$$\int_{I_{ij}} \dot{U}_i v \,\mathrm{d}t = \int_{I_{ij}} f_i(U,\cdot) v \,\mathrm{d}t \quad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}). \tag{4}$$

We define the *residual* R of the approximate solution U to be $R(U,t) = \dot{U}(t) - f(U(t),t)$. In terms of the residual, we can rewrite (4) as $\int_{I_{ij}} R_i(U,\cdot) v \,\mathrm{d}t = 0$ for all $v \in \mathcal{P}^{q_{ij}-1}(I_{ij})$, i.e., the residual is orthogonal to the test space on every local interval. We refer to this as the *Galerkin orthogonality* of the mcG(q) method.

2.2. *Multi-adaptive discontinuous Galerkin,* mdG(q)

For the mdG(q) method, we define the trial and test spaces by

$$V = W = \left\{v\colon\ v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}),\ j=1,\ldots,M_i,\ i=1,\ldots,N\right\}, \tag{5}$$

i.e., both trial and test functions are (in general discontinuous) piecewise polynomials of degree $q = q_i(t) = q_{ij},\ t \in I_{ij}$ on the partition $\mathcal{T}$.

We define the mdG(q) method for (1) as follows, similar to the definition of the continuous method: Find $U \in V$ with $U(0^-) = u_0$, such that

$$\sum_{i=1}^{N}\sum_{j=1}^{M_i}\left[[U_i]_{i,j-1} v\left(t_{i,j-1}^+\right) + \int_{I_{ij}} \dot{U}_i v_i \,\mathrm{d}t\right] = \int_0^T \left(f(U,\cdot), v\right) \mathrm{d}t \quad \forall v \in W, \tag{6}$$

where $[\cdot]$ denotes the jump, i.e., $[v]_{ij} = v(t_{ij}^+) - v(t_{ij}^-)$.

The mdG(q) method in local form, corresponding to (4), reads: For $i = 1, \ldots, N$, $j = 1, \ldots, M_i$, find $U_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$, such that

$$[U_i]_{i,j-1} v(t_{i,j-1}) + \int_{I_{ij}} \dot{U}_i v \, \mathrm{d}t = \int_{I_{ij}} f_i(U, \cdot) v \, \mathrm{d}t \quad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}), \tag{7}$$

where the initial condition is specified for $i = 1, \ldots, N$, by $U_i(0^-) = u_i(0)$. In the same way as for the continuous method, we define the residual R of the approximate solution U to be $R(U, t) = \dot{U}(t) - f(U(t), t)$, defined on the inner of every local interval I_{ij}, and rewrite (7) in the form $[U_i]_{i,j-1} v(t_{i,j-1}) + \int_{I_{ij}} R_i(U, \cdot) v \, \mathrm{d}t = 0$ for all $v \in \mathcal{P}^{q_{ij}}(I_{ij})$. We refer to this as the Galerkin orthogonality of the mdG(q) method.

3. Error control and adaptivity

Our goal is to compute an approximation $U(T)$ of the exact solution $u(T)$ of (1) at final time T within a given tolerance TOL > 0, using a minimal amount of computational work. This goal includes an aspect of *reliability* (the error should be less than the tolerance) and an aspect of *efficiency* (minimal computational work). To measure the error we choose a norm, such as the Euclidean norm $\|\cdot\|$ on $\mathbb{R}^N$, or more generally some other quantity of interest.

We discuss below both a priori and a posteriori error estimates for the multi-adaptive Galerkin methods, and the application of the a posteriori error estimates in multi-adaptive time-stepping.

3.1. A priori error estimates

Standard (duality-based) a priori error estimates show that the order for the ordinary Galerkin methods cG(q) and dG(q) is $2q$ and $2q + 1$, respectively. A generalisation of these estimates to the multi-adaptive methods gives the same result. The multi-adaptive continuous Galerkin method mcG(q) is thus of order $2q$, and the multi-adaptive discontinuous Galerkin method mdG(q) is of order $2q + 1$.

3.2. A posteriori error estimates

A posteriori error analysis in the general framework of [8] relies on the concept of the *dual problem*. The dual problem of the initial value problem (1) is the linearised backward problem given by

$$\begin{cases} -\dot{\phi} = J^*(u, U, \cdot)\phi & \text{on } [0, T), \\ \phi(T) = e(T)/\|e(T)\|, \end{cases} \tag{8}$$

where the Jacobian J is given by $J(u, U, \cdot) = \int_0^1 \frac{\partial f}{\partial u}(su + (1 - s)U, \cdot) \, \mathrm{d}s$ and * denotes the transpose. We use the dual problem to represent the error in terms of the dual solution ϕ and the residual R. For the mcG(q) method the representation formula is given by

$$\|e(T)\| = \int_0^T (R, \phi) \, \mathrm{d}t, \tag{9}$$

and for the mdG(q) method, we obtain

$$\left\|e(T)\right\| = \sum_{i=1}^{N}\sum_{j=1}^{M_i}[U_i]_{i,j-1}\phi_i(t_{i,j-1}) + \int_{I_{ij}} R_i(U,\cdot)\phi_i \,\mathrm{d}t. \tag{10}$$

Using the Galerkin orthogonalities together with special interpolation estimates (see [29]), we obtain a posteriori error estimates of the form

$$\left\|e(T)\right\| \leqslant \sum_{i=1}^{N} S_i^{[q_i]} \max_{[0,T]}\left\{Ck_i^{q_i} r_i\right\}, \tag{11}$$

for the mcG(q) method, and

$$\left\|e(T)\right\| \leqslant \sum_{i=1}^{N} S_i^{[q_i+1]} \max_{[0,T]}\left\{Ck_i^{q_i+1} r_i\right\}, \tag{12}$$

for the mdG(q) method, where C is an interpolation constant, r_i is a local measure of the residual, and the individual *stability factors* S_i are given by $S_i^{[q_i]} = \int_0^T |\phi_i^{(q_i)}| \,\mathrm{d}t$. Typically, the stability factors are of moderate size for a stiff problem (and of unit size for a parabolic problem), which means that accurate computation is possible over long time intervals. Note that the Lipschitz constant, which is large for a stiff problem, is not present in these estimates.

The analysis can be extended to include also *computational errors*, arising from solving the discrete equations using an iterative method, and *quadrature errors*, arising from evaluating the integrals in (4) and (7) using quadrature.

3.3. Adaptivity

To achieve the goals stated at the beginning of this section, the adaptive algorithm chooses individual time steps for the different components based on the a posteriori error estimates. Using for example a standard *PID* regulator from control theory, we choose the individual time steps for each component to satisfy

$$S_i C k_{ij}^{p_{ij}} r_{ij} = \mathrm{TOL}/N, \tag{13}$$

or, taking the logarithm with $C_i = \log(\mathrm{TOL}/(NS_iC))$,

$$p_{ij}\log k_{ij} + \log r_{ij} = C_i, \tag{14}$$

with maximal time steps $\{k_{ij}\}$, following work by Söderlind and coworkers [23,35]. Here, $p_{ij} = q_{ij}$ for the mcG(q) method and $p_{ij} = q_{ij} + 1$ for the mdG(q) method.

To solve the dual problem (8), which is needed to compute the stability factors, it would seem that we need to know the error $e(T)$, since this is used as an initial value for the dual problem. However, we know from experience that the stability factors are quite insensitive to the choice of initial data for the dual problem. (A motivation of this for parabolic problems is given in [15].) Thus in practice, a random (and normalised) value is chosen as initial data for the dual problem. Another approach is to take the initial value for the dual problem to be $\phi(T) = (0,\ldots,0,1,0,\ldots,0)$, i.e., a vector of zeros except for a single component which is of size one. This gives an estimate for a chosen component of the error. By

other choices of data for the dual problem, other functionals of the error can be controlled. In either case, the stability factors are computed using quadrature from the computed dual solution.

The adaptive algorithm can be expressed as follows: Given a tolerance TOL > 0, make a preliminary estimate for the stability factors and then

(i) Solve the primal problem with time steps based on (13);
(ii) Solve the dual problem and compute the stability factors;
(iii) Compute an error bound E based on (9) or (10);
(iv) If $E \leqslant \text{TOL}$ then stop; if not go back to (i).

Note that we use the error representations (9) and (10) to obtain sharp error estimates. On the other hand, the error estimates (11) and (12) are used to determine the adaptive time step sequences.

To limit the computational work, it is desirable that only a few iterations in the adaptive algorithm are needed. In the simplest case, the error estimate will stay below the given tolerance on the first attempt. Otherwise, the algorithm will try to get below the tolerance the second time. It is also possible to limit the number of times the dual problem is solved. It should also be noted that to obtain an error estimate at a time $t = \bar{t}$, different from the final time T, the dual problem has to be solved backwards also from time $\bar{t}$. This may be necessary in some cases if the stability factors do not grow monotonically as functions of the final time T, but for many problems the stability factors grow with T, indicating accumulation of errors.

Our experience is that automatic computation based on this adaptive strategy is both reliable (the error estimates are quite close to the actual error) and efficient (the additional cost for solving the dual problem is quite small). See [28] for a discussion on this topic.

4. Iterative methods for the nonlinear system

The nonlinear discrete algebraic equations given by the mcG(q) and mdG(q) methods presented in Section 2 (including numerical quadrature) to be solved on every local interval I_{ij} take the form

$$\xi_{ijm} = \xi_{ij0} + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} f_i\big(U\big(\tau_{ij}^{-1}\big(s_n^{[q_{ij}]}\big)\big), \tau_{ij}^{-1}\big(s_n^{[q_{ij}]}\big)\big), \tag{15}$$

for $m = 0, \ldots, q_{ij}$, where $\{\xi_{ijm}\}_{m=0}^{q_{ij}}$ are the degrees of freedom to be determined for component $U_i(t)$ on the interval I_{ij}, $\{w_{mn}^{[q_{ij}]}\}_{m=0,n=0}^{q_{ij}}$ are weights, τ_{ij} maps I_{ij} to $(0, 1]$: $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$, and $\{s_n^{[q_{ij}]}\}_{n=0}^{q_{ij}}$ are quadrature points defined on $[0, 1]$.

The strategy we use to solve the discrete equations (15) is by direct fixed point iteration, possibly in combination with a simplified Newton's method. To evolve the system, we need to collect the degrees of freedom for different components between two time levels and solve the discrete equations for these degrees of freedom. We refer to such a collection of elements between two time levels as a time slab (see Fig. 1). New time slabs are formed as we evolve the system starting at time $t = 0$, in the same way as new time intervals are formed in a standard solver which uses the same time steps for all components. On each time slab, we thus compute the degrees of freedom $\{\xi_{ijm}\}_{m=0}^{q_{ij}}$ for each element within the time slab using (15), and repeat the iterations until the computational error is below a given tolerance for the

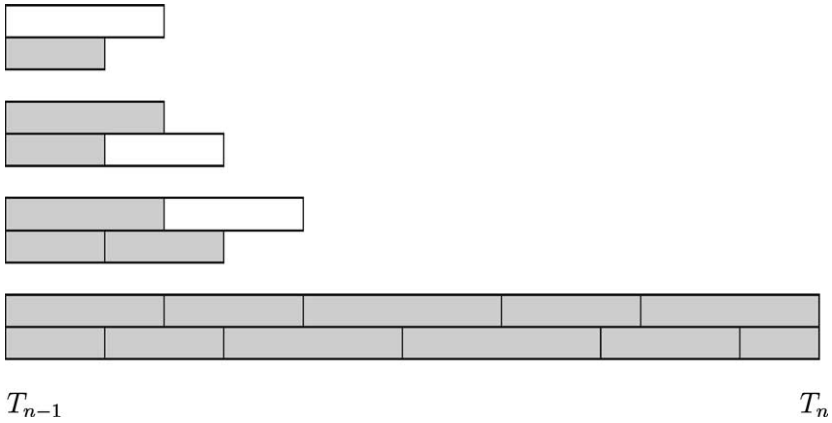

Fig. 2. Multi-adaptive time-stepping within a time slab for a system with two components.

computational error. The iterations are carried out in order, starting at the element closest to time $t = 0$ and continuing until we reach the last element within the time slab. This is illustrated in Fig. 2.

The motivation for using direct fixed point iteration, rather than using a full Newton's method, is that we want to avoid forming the Jacobian (which may be very large, since the nonlinear system to be solved is for the entire time slab) and also avoid solving the linearised system. Instead, using the strategy for adaptive damping of the fixed point iterations as described in the next section, the linear algebra is built into the adaptive solver. Since often only a small number of fixed point iterations are needed, typically only two or three iterations, we believe this to be an efficient approach.

5. Stiff problems

As discussed in the previous section, the nonlinear discrete equations given by the (implicit) multi-adaptive Galerkin methods are solved using fixed point iteration on each time slab. For stiff problems these iterations may fail to converge. We now discuss a simple way to stabilise a stiff system, in order to make the explicit fixed point iterations convergent.

For simplicity, we assume that the time step sequence, $k_1, k_2, \ldots, k_M$, is the same for all components.

5.1. The test equation

To demonstrate the main idea, we consider the stabilisation of the explicit Euler method applied to the simple *test equation*:

$$\begin{cases} \dot{u}(t) + \lambda u(t) = 0 \quad \text{for } t > 0, \\ u(0) = u_0, \end{cases} \tag{16}$$

where $\lambda > 0$ and u_0 is a given initial condition. The solution is given by $u(t) = \exp(-\lambda t)u_0$.

The explicit Euler method for the test equation reads

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

This method is conditionally stable, with stability guaranteed if $k_n\lambda \leqslant 2$. If λ is large, this is too restrictive outside transients.

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

$$U^n = (1 - k\lambda)^m(1 - K\lambda)U^{n-1}, \tag{17}$$

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 $k_n = K + mk$. For the overall method to be stable, we require that $|1 - k\lambda|^m(K\lambda - 1) \leqslant 1$, that is

$$m \geqslant \frac{\log(K\lambda - 1)}{-\log|1 - k\lambda|} \approx \frac{\log(K\lambda)}{c}, \tag{18}$$

if $K\lambda \gg 1$ and $c = k\lambda$ is of moderate size, say $c = 1/2$.

We conclude that m will be quite small and hence the small time steps will be used only in a small fraction of the total time interval, giving a large effective time step. To see this, define the *cost* as $\alpha = \frac{1+m}{K+km} \in (1/K, 1/k)$, i.e., the number of time steps per unit interval. Classical stability analysis gives $\alpha = 1/k = \lambda/2$ with a maximum time step $k = 2/\lambda$. Using (18) we instead find

$$\alpha \approx \frac{1 + \log(K\lambda)/c}{K + \log(K\lambda)/\lambda} \approx \frac{\lambda}{c}\log(K\lambda)/(K\lambda) \ll \lambda/c, \tag{19}$$

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 $K\lambda$.

5.2. The general nonlinear problem

For the general nonlinear problem (1), the gain is determined by the distribution of the eigenvalues of the Jacobian, see [16]. The method of stabilising the system using a couple of small stabilising time steps is best suited for systems with a clear separation of the eigenvalues into small and large eigenvalues, but even for the semi-discretised heat equation (for which we have a whole range of eigenvalues) the gain can be substantial, as we shall see below.

5.3. An adaptive algorithm

In [16] we present an adaptive algorithm in which both the size of the small stabilising time steps and the number of such small time steps are automatically determined. Using adaptive stabilisation, the damping is targeted precisely at the current unstable eigenmode, which as a consequence allows efficient integration also of problems with no clear separation of its eigenvalues.

6. Numerical examples

The numerical examples presented in this section are divided into two categories: examples illustrating the concept of multi-adaptivity and examples illustrating explicit time-stepping (or explicit fixed point iteration) for stiff problems.

6.1. Multi-adaptivity

The two examples presented below are taken from [30], in which further examples are presented and discussed in more detail.

6.1.1. A mechanical multi-scale system

To demonstrate the potential of the multi-adaptive methods, we consider a dynamical system in which a small part of the system oscillates rapidly. The problem is to compute accurately the positions (and velocities) of the N point-masses attached together with springs of equal stiffness as in Fig. 3.

We choose a small time step for the smallest mass and large time steps for the larger masses, and measure the work for the mcG(1) method as we increase the number of larger masses. The work is then compared to the work required for the standard cG(1) method using the same (small) time step for all

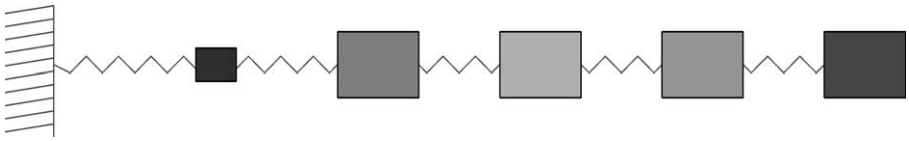

Fig. 3. A mechanical system consisting of $N = 5$ masses attached together with springs.

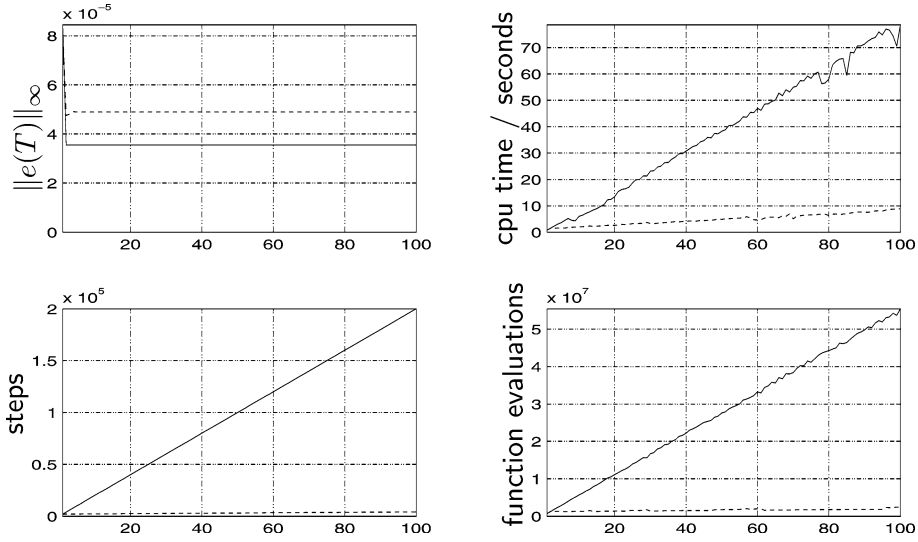

Fig. 4. Error, cpu time, total number of steps, and number of function evaluations as function of the number of masses, for the multi-adaptive cG(1) method (dashed) and the standard cG(1) method (solid).

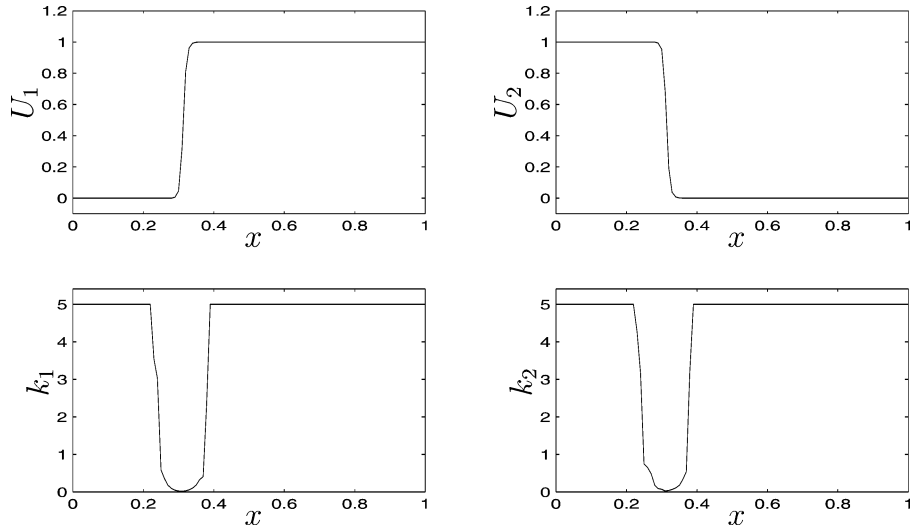

Fig. 5. The concentrations of the two species, U_1 and U_2, at time $t = 50$ as function of space (above), and the corresponding time steps (below).

masses. As is evident in Fig. 4, the work (in terms of function evaluations) increases linearly for the standard method, whereas for the multi-adaptive method it remains practically constant.

6.1.2. Reaction–diffusion

Next consider the following system of PDEs:

$$\begin{cases} \dot{u}_1 - \varepsilon u_1'' = -u_1 u_2^2, \\ \dot{u}_2 - \varepsilon u_2'' = u_1 u_2^2, \end{cases} \tag{20}$$

on $(0,1) \times (0,T]$ with $\varepsilon = 0.001$, $T = 100$ and homogeneous Neumann boundary conditions at $x = 0$ and $x = 1$, which models isothermal auto-catalytic reactions (see [33]): $A_1 + 2A_2 \to A_2 + 2A_2$. As initial conditions, we take $u_1(x,0) = 0$ for $0 < x < x_0$, $u_1(x,0) = 1$ for $x_0 \leqslant x < 1$, and $u_2(x,0) = 1 - u_1(x,0)$ with $x_0 = 0.2$. An initial reaction where substance A_1 is consumed and substance A_2 is formed will then take place at $x = x_0$, resulting in a decrease in the concentration u_1 and an increase in the concentration u_2. The reaction then propagates to the right until all of substance A_1 is consumed and we have $u_1 = 0$ and $u_2 = 1$ in the entire domain.

Computing the solution using the mcG(2) method, we find that the time steps are automatically chosen to be small only in the vicinity of the reaction front, see Fig. 5, and during the computation the region of small time steps will propagate to the right at the same speed as the reaction front.

6.2. Explicit time-stepping for stiff problems

To illustrate the technique of stabilisation for stiff problems, we present below some examples taken from [16]. In these examples, the cost α is compared to the cost α_0 of a standard implementation of the cG(1) method in which we are forced to take a small time step all the time. (These small time steps are marked by dashed lines in the figures.) Comparison has not been made with an implicit method, since

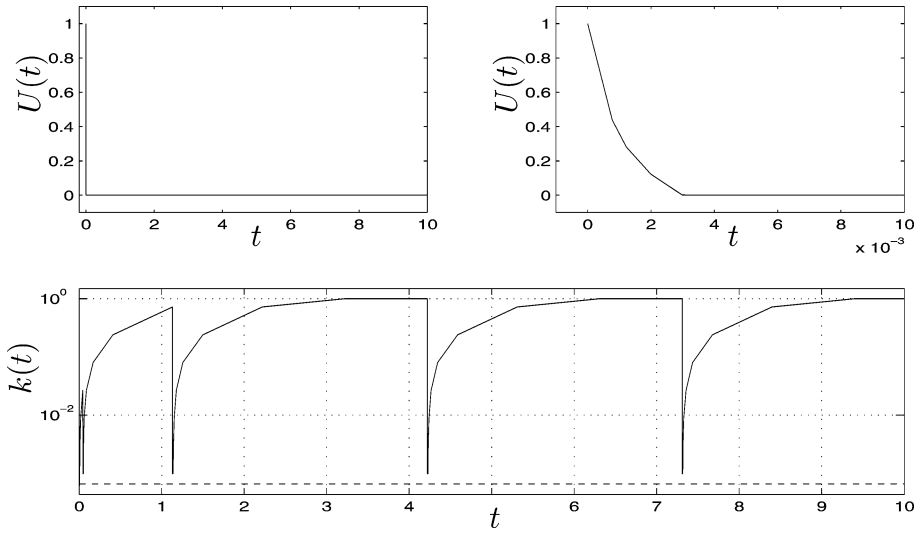

Fig. 6. Solution and time step sequence for Eq. (21), $\alpha/\alpha_0 \approx 1/310$.

it would be difficult to make such a comparison fair; one could always argue about the choice of linear solver and preconditioner. However, judging by the modest restriction of the average time step size and the low cost of the explicit method, we believe our approach to be competitive also with implicit methods, although this remains to be seen.

6.2.1. The test equation

The first problem we try is the test equation:

$$\begin{cases} \dot{u}(t) + \lambda u(t) = 0 \quad \text{for } t > 0, \\ u(0) = u_0, \end{cases} \tag{21}$$

on $[0, 10]$, where we choose $u_0 = 1$ and $\lambda = 1000$. As is shown in Fig. 6, the time step is repeatedly decreased to stabilise the stiff system, but overall the effective time step is large and the cost reduction factor is $\alpha/\alpha_0 \approx 1/310$.

6.2.2. The test system

For the test system,

$$\begin{cases} \dot{u}(t) + Au(t) = 0 \quad \text{for } t > 0, \\ u(0) = u_0, \end{cases} \tag{22}$$

on $[0, 10]$, we take $A = \text{diag}(100, 1000)$ and $u_0 = (1, 1)$. As seen in Fig. 7, most of the stabilising steps are chosen to damp out the eigenmode corresponding to the largest eigenvalue, $\lambda_2 = 1000$, but some of the damping steps are targeted at the second eigenvalue, $\lambda_1 = 100$. The selective damping is handled automatically by the adaptive algorithm and the cost reduction factor is again significant: $\alpha/\alpha_0 \approx 1/104$.

6.2.3. 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 [36]. The problem consists of

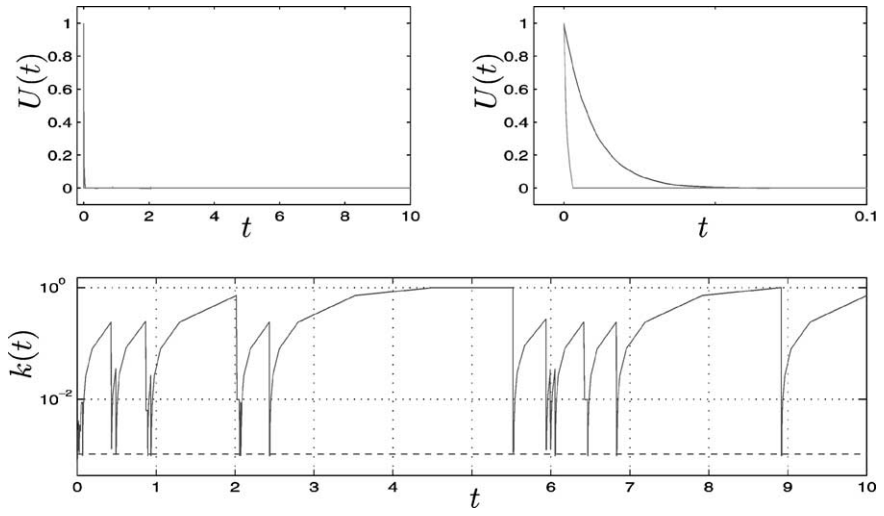

Fig. 7. Solution and time step sequence for Eq. (22), $\alpha/\alpha_0 \approx 1/104$.

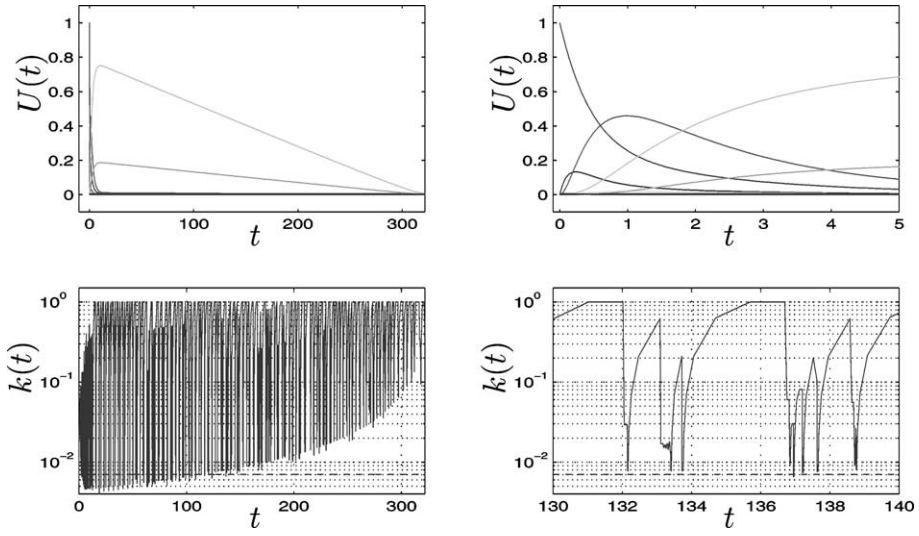

Fig. 8. Solution and time step sequence for Eq. (23), $\alpha/\alpha_0 \approx 1/33$.

the following eight equations:

$$
\begin{cases}
\dot{u}_1 = -1.71u_1 + 0.43u_2 + 8.32u_3 + 0.0007, \\
\dot{u}_2 = 1.71u_1 - 8.75u_2, \\
\dot{u}_3 = -10.03u_3 + 0.43u_4 + 0.035u_5, \\
\dot{u}_4 = 8.32u_2 + 1.71u_3 - 1.12u_4, \\
\dot{u}_5 = -1.745u_5 + 0.43u_6 + 0.43u_7, \\
\dot{u}_6 = -280.0u_6u_8 + 0.69u_4 + 1.71u_5 - 0.43u_6 + 0.69u_7, \\
\dot{u}_7 = 280.0u_6u_8 - 1.81u_7, \\
\dot{u}_8 = -280.0u_6u_8 + 1.81u_7,
\end{cases}
\tag{23}
$$

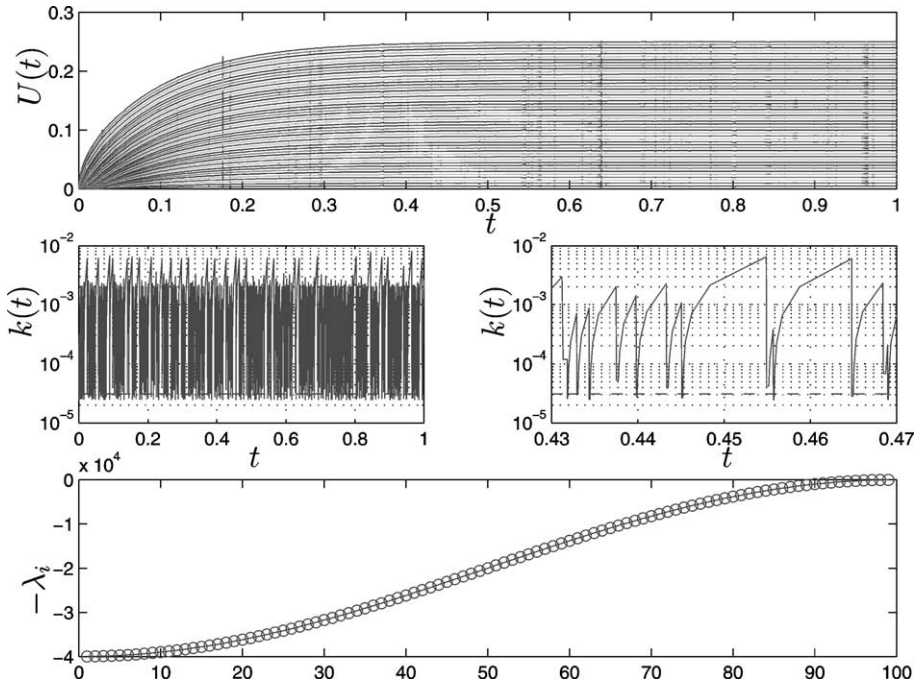

Fig. 9. Solution and time step sequence for Eq. (25), $\alpha/\alpha_0 \approx 1/17$.

on $[0, 321.8122]$ (as specified in [36]). The initial condition is given by $u_0 = (1.0, 0, 0, 0, 0, 0, 0, 0.0057)$. The cost reduction factor is now $\alpha/\alpha_0 \approx 1/33$, see Fig. 8.

6.3. *The heat equation*

Finally, we consider the heat equation in one dimension:

$$\begin{cases} \dot{u}(x,t) - u''(x,t) = f(x,t), \quad x \in (0,1), t > 0, \\ u(0) = u(1) = 0, \\ u(\cdot,t) = 0, \end{cases} \tag{24}$$

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

$$\begin{cases} \dot{u}(t) + Au(t) = f, \quad t > 0, \\ u(0) = 0, \end{cases} \tag{25}$$

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 Fig. 9). The selective damping produced by the adaptive algorithm performs well and the cost reduction factor is $\alpha/\alpha_0 \approx 1/17$.

References

[1] S. Alexander, C. Agnor, N-body simulations of late stage planetary formation with a simple fragmentation model, ICARUS 132 (1998) 113–124.

[2] J. Butcher, The Numerical Analysis of Ordinary Differential Equations—Runge–Kutta and General Linear Methods, Wiley, New York, 1987.

[3] C.W. Gear, I.G. Kevrekidis, Projective methods for stiff differential equations: Problems with gaps in their eigenvalue spectrum, SIAM J. Sci. Comput. 24 (2002) 1091–1106.

[4] G. Dahlquist, Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations, Ph.D. Thesis, Trans. of the Royal Inst. of Techn., Stockholm, Sweden, Number 130, Uppsala, 1958.

[5] R. Dave, J. Dubinski, L. Hernquist, Parallel TreeSPH, New Astron. 2 (1997) 277–297.

[6] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (6) (2001) 2256–2281.

[7] M. Delfour, W. Hager, F. Trochu, Discontinuous Galerkin methods for ordinary differential equations, Math. Comp. 36 (1981) 455–473.

[8] K. Eriksson, D. Estep, P. Hansbo, C. Johnson, Introduction to adaptive methods for differential equations, Acta Numer. 1995 (1995) 105–158.

[9] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems III: Time steps variable in space, in preparation/personal communication.

[10] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems I: A linear model problem, SIAM J. Numer. Anal. 28 (1) (1991) 43–77.

[11] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems II: Optimal order error estimates in $L_\infty L_2$ and $L_\infty L_\infty$, SIAM J. Numer. Anal. 32 (1995) 706–740.

[12] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems IV: Nonlinear problems, SIAM J. Numer. Anal. 32 (1995) 1729–1749.

[13] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems V: Long-time integration, SIAM J. Numer. Anal. 32 (1995) 1750–1763.

[14] K. Eriksson, C. Johnson, S. Larsson, Adaptive finite element methods for parabolic problems VI: Analytic semigroups, SIAM J. Numer. Anal. 35 (1998) 1315–1325.

[15] K. Eriksson, C. Johnson, A. Logg, Adaptive computational methods for parabolic problems, Encyclopedia Comput. Mech., March 2003, submitted for publication.

[16] K. Eriksson, C. Johnson, A. Logg, Explicit time-stepping for stiff ODEs, SIAM J. Sci. Comput. (2003), submitted for publication.

[17] K. Eriksson, C. Johnson, V. Thome, Time discretization of parabolic problems by the discontinuous Galerkin method, RAIRO Modél. Math. Anal. Numér. 19 (1985) 611–643.

[18] D. Estep, A posteriori error bounds and global error control for approximations of ordinary differential equations, SIAM J. Numer. Anal. 32 (1995) 1–48.

[19] D. Estep, D. French, Global error control for the continuous Galerkin finite element method for ordinary differential equations, Modél. Math. Anal. Numér. 28 (1994) 815–852.

[20] D. Estep, M.G. Larson, R. Williams, Estimating the error of numerical solutions of systems of reaction–diffusion equations, Mem. Amer. Math. Soc. 146 (2000) 1–109.

[21] D. Estep, R. Williams, Accurate parallel integration of large sparse systems of differential equations, Math. Models Methods Appl. Sci. 6 (1996) 535–568.

[22] J. Flaherty, R. Loy, M. Shephard, B. Szymanski, J. Teresco, L. Ziantz, Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws, J. Parallel Distributed Comput. 47 (1997) 139–152.

[23] K. Gustafsson, M. Lundh, G. Söderlind, A PI stepsize control for the numerical solution of ordinary differential equations, BIT 28 (1988) 270–287.

[24] E. Hairer, G. Wanner, Solving Ordinary Differential Equations I—Nonstiff Problems, in: Springer Ser. Comput. Math., vol. 8, Springer, Berlin, 1991.

[25] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II—Stiff and Differential-Algebraic Problems, in: Springer Ser. Comput. Math., vol. 14, Springer, Berlin, 1991.

[26] J.G. Verwer, Explicit Runge–Kutta methods for parabolic partial differential equations, Appl. Numer. Math. 22 (1996) 359–379.
[27] C. Johnson, Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations, SIAM J. Numer. Anal. 25 (4) (1988) 908–926.
[28] A. Logg, Multi-adaptive Galerkin methods for ODEs II: Applications, Preprint 2001-10, Chalmers Finite Element Center, Göteborg, 2001.
[29] A. Logg, Multi-adaptive Galerkin methods for ODEs I, SIAM J. Sci. Comput. 24 (2003) 1879–1902.
[30] A. Logg, Multi-adaptive Galerkin methods for ODEs II: Implementation and applications, SIAM J. Sci. Comput. (2003), submitted for publication.
[31] J. Makino, S. Aarseth, On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems, Publ. Astron. Soc. Japan 44 (1992) 141–151.
[32] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comp. 41 (1983) 321–336.
[33] R. Sandboge, Adaptive Finite Element Methods for Reactive Flow Problems, Ph.D. Thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, 1996.
[34] L. Shampine, Numerical Solution of Ordinary Differential Equations, Chapman & Hall, London, 1994.
[35] G. Söderlind, The automatic control of numerical integration, CWI Quarterly 11 (1998) 55–74.
[36] W.M. Lioen, J.J.B. de Swart, Test set for initial value problem solvers, Report MAS-R9832, CWI, Amsterdam, 1998, ISSN 1386-3703.