THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Automation of Computational Mathematical Modeling

ANDERS LOGG

Department of Computational Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg, Sweden 2004

Automation of Computational Mathematical Modeling
ANDERS LOGG
ISBN 91–7291–436–X

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 2118
ISSN 0346–718X

Department of Computational Mathematics
Chalmers University of Technology
Göteborg University
SE–412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

Tryckeriet, Matematiskt centrum
Göteborg, Sweden 2004

Automation of Computational Mathematical Modeling

ANDERS LOGG

Department of Computational Mathematics
Chalmers University of Technology
Göteborg University

ABSTRACT

This thesis concerns the Automation of Computational Mathematical Modeling (CMM), involving the key steps of automation of (i) discretization, (ii) discrete solution, (iii) error control, (iv) modeling, and (v) optimization.

The automation of (i)–(ii) is realized through *multi-adaptive Galerkin methods* on general discretizations of space–time with time steps variable in space. The automation of (iii) is realized through a posteriori error estimation based on residuals of computed solutions and stability factors or stability weights obtained by the solution of associated dual problems. The automation of (iv) is studied in a model case, while that of (v) is reduced to an application of (i)–(iv).

An open-source software system for the Automation of Computational Mathematical Modeling is developed and implemented, based on general order multi-adaptive Galerkin methods, including automatic evaluation of variational forms, automatic discrete solution, automatic generation and solution of dual problems, and automatic model reduction.

Keywords: Computational mathematical modeling, automation, adaptivity, multi-adaptivity, individual time steps, dual problem, a priori, a posteriori, model reduction.

THESIS

This thesis consists of an introduction, a sequence of papers as listed below, and the source code included on the compact disc.

APPENDED PAPERS

The following papers are included in this thesis:

PAPER I: *Multi-Adaptive Galerkin Methods for ODEs I*, SIAM J. Sci. Comput., vol. 24 (2003), pp. 1879–1902.

PAPER II: *Multi-Adaptive Galerkin Methods for ODEs II: Implementation and Applications*, SIAM J. Sci. Comput., vol. 25 (2003), pp. 1119–1141.

PAPER III: *Multi-Adaptive Galerkin Methods for ODEs III: Existence and Stability*, submitted to SIAM J. Numer. Anal. (2004).

PAPER IV: *Multi-Adaptive Galerkin Methods for ODEs IV: A Priori Error Estimates*, submitted to SIAM J. Numer. Anal. (2004).

PAPER V: *Multi-Adaptive Galerkin Methods for ODEs V: Stiff Problems*, submitted to BIT (2004), with J. Jansson.

PAPER VI: *Explicit Time-Stepping for Stiff ODEs*, SIAM J. Sci. Comput., vol. 25 (2003), pp. 1142–1157, with K. Eriksson and C. Johnson.

PAPER VII: *Interpolation Estimates for Piecewise Smooth Functions in One Dimension*, Chalmers Finite Element Center Preprint Series, no. 2004–02 (2004).

PAPER VIII: *Estimates of Derivatives and Jumps Across Element Boundaries for Multi-Adaptive Galerkin Solutions of ODEs*, Chalmers Finite Element Center Preprint Series, no. 2004–03 (2004).

PAPER IX: *Algorithms for Multi-Adaptive Time-Stepping*, submitted to ACM Trans. Math. Softw. (2004), with J. Jansson.

PAPER X: *Simulation of Mechanical Systems with Individual Time Steps*, submitted to SIAM J. Appl. Dyn. Syst. (2004), with J. Jansson.

PAPER XI: *Computational Modeling of Dynamical Systems*, to appear in M^3AS (2004), with J. Jansson and C. Johnson.

The following papers are related, but are not included in the thesis:

- *Topics in Adaptive Computational Methods for Differential Equations*, CEDYA 2001: Congreso de Ecuaciones Diferenciales y Aplicaciones (2001), with J. Hoffman and C. Johnson.
- *Multi-Adaptive Time-Integration*, Applied Numerical Mathematics, vol. 48, pp. 339–354 (2004).
- *Adaptive Computational Methods for Parabolic Problems*, to appear in Encyclopedia of Computational Mechanics (2004), with K. Eriksson and C. Johnson.
- *DOLFIN: Dynamic Object oriented Library for FINite element computation*, Chalmers Finite Element Center Preprint Series, no. 2002–06 (2002), with J. Hoffman.
- *The FEniCS project*, Chalmers Finite Element Center Preprint Series, no. 2003–21 (2003), with T. Dupont, J. Hoffman, C. Johnson, R.C. Kirby, M.G. Larson, and L.R. Scott.

CONTENTS ON THE COMPACT DISC

The included compact disc contains the following material:

- The introduction;
- PAPER I–XI as listed above;
- Source code for **DOLFIN** version 0.4.11;
- Source code for Puffin version 0.1.6;
- Images and animations.

PREFACE

To Prof. Kenneth Eriksson and Prof. Claes Johnson, my two advisors, thanks for all your inspiration and support over the years. Special thanks to Claes for the piano lessons, the tennis lessons, and the endless discussions on music, philosophy, and life.

Thanks also to Prof. Endre Süli, Prof. Jan Verwer, and Prof. Ridgway Scott for giving me the opportunity to visit Oxford, Amsterdam, and Chicago during my time as a graduate student. These visits have all been invaluable experiences.

To all my friends and colleagues at the Department of Computational Mathematics at Chalmers, thanks for the many enlightening discussions on mathematics, programming, and football. In particular, I wish to acknowledge my co-authors Johan Hoffman and Johan Jansson. Thanks also to Göran Christiansson for valuable comments on the manuscript.

Special thanks to all the members of the **DOLFIN** team: Fredrik Bengzon, Niklas Ericsson, Georgios Foufas, David Heintz, Rasmus Hemph, Johan Hoffman, Johan Jansson, Karin Kraft, Aleksandra Krusper, Andreas Mark, Axel Målqvist, Andreas Nilsson, Erik Svensson, Jim Tilander, Thomas Svedberg, Harald Svensson, and Walter Villanueva.

But most of all, I would like to thank my wife Anna, my parents and grandparents, my little sister Katarina, and Lysekilarna, for all their love and support.

Göteborg, April 2004
Anders Logg

And now that we may give final praise to the machine we may say that it will be desirable to all who are engaged in computations which, it is well known, are the managers of financial affairs, the administrators of others' estates, merchants, surveyors, geographers, navigators, astronomers... For it is unworthy of excellent men to lose hours like slaves in the labor of calculations which could safely be relegated to anyone else if the machine were used.

Gottfried Wilhelm Leibniz (1646–1716)

INTRODUCTION

Computational Mathematical Modeling (CMM) can be viewed as the modern manifestation of the basic principle of the natural sciences: formulating equations (modeling) and solving equations (computation).

Models of nature are often expressed as differential equations of the canonical form

$$\dot{u} = f(u) \tag{1}$$

on a given domain in space–time, stating that the rate of change $\dot{u}$ of the *solution* u is given by $f(u)$ depending on u and space derivatives of u. Famous models of nature of the form (1) include the Navier–Stokes equations of fluid dynamics, Maxwell's equations of electromagnetics, and the Schrödinger equation of quantum mechanics.

Until recently, it has not been possible to solve differential equations in any generality. In particular, using classical analytic methods, solutions are only available in a few special cases. A prime example is the n-body problem; the general solution of the n-body problem was given by Newton for $n = 2$ [52], but no one has been able to find the general solution of the n-body problem for $n = 3$.

The situation has changed with the arrival of the modern computer. Using a computer, the solution of the three-body problem can be obtained in fractions of a second. Computational simulation of real world phenomena is becoming possible for the first time. Thus, the advancement of CMM opens new possibilities for our understanding of nature, by making possible the detailed study of complex phenomena, such as turbulence in fluid motion and the stability of our solar system.

The Principles of Automation

The modern industrial society is based on automated mass-production of material goods, such as food, clothes, cars, and computers. The emerging information society is based on automated mass-processing of information by computers. An important part of this process concerns the Automation of Computational Mathematical Modeling, which is the topic of this thesis.

An *automatic* system carries out a well-defined task without intervention from the person or system actuating the automatic process. The task of the automatic system may be formulated as follows: For given input satisfying a fixed set of conditions (the *input conditions*), produce output satisfying a given set of conditions (the *output conditions*).

An automatic process is defined by an *algorithm*, consisting of a sequential list of instructions (like a computer program). In automated manufacturing, each step of the algorithm operates on and transforms physical material. Correspondingly, an algorithm for the Automation of CMM operates on digits and consists of the automated transformation of digital information.

A key problem of automation is the design of a *feed-back control*, allowing the given output conditions to be satisfied under variable input and external conditions, ideally at a minimal cost. Feed-back control is realized through *measurement*, *evaluation*, and *action*; a quantity relating to the given set of conditions to be satisfied by the output is measured, the measured quantity is evaluated to determine if the output conditions are satisfied or if an adjustment is necessary, in which case some action is taken to make the necessary adjustments. In the context of an algorithm for feed-back control, we refer to the evaluation of the set of output conditions as the *stopping criterion*, and to the action as the *modification strategy*.

A key step in the automation of a complex process is *modularization*, i.e., the hierarchical organization of the complex process into components or subprocesses. Each subprocess may then itself be automated, including feed-back control. We may also express this as *abstraction*, i.e., the distinction between the properties of a component (its purpose) and the internal workings of the component (its realization).

Modularization (or abstraction) is central in all engineering and makes it possible to build complex systems by connecting together components or subsystems without concern for the internal workings of each subsystem.

We thus identify the following basic principles of automation: algorithms, feed-back control, and modularization.

The Automation of Computational Mathematical Modeling

In automated manufacturing, the task of the automatic system is to produce a certain product (the output) from a given piece of material (the input), with the product satisfying some measure of quality (the output conditions). In the Automation of CMM, the input is a given model of the form (1) and the output is a discrete solution $U \approx u$ satisfying some measure of quality. Typically, the measure of quality is given in the form of a *tolerance* $\mathrm{TOL} > 0$ for the size of the *error* $e = U - u$ in a suitable norm:

$$\|e\| \leq \mathrm{TOL}. \tag{2}$$

The key problem for the Automation of CMM is thus the design of a feed-back control for the automatic construction of a discrete solution U, satisfying the output condition (2) at minimal cost.

The design of this feed-back control is based on the solution of an associated *dual problem*, connecting the size of the *residual* $R(U) = \dot{U} - f(U)$ of the computed discrete solution to the size of the error e, and thus to the output condition (2). The solution of the dual problem is also the key to the realization of an automatic system for the construction of the discrete solution U at minimal cost. We return to this in detail below.

Following our previous discussion on modularization as a basic principle of automation, we identify the key steps in the Automation of CMM. These key steps concern the automation of

(i) discretization;

(ii) discrete solution;

(iii) error control;

(iv) modeling;

(v) optimization.

The *automation of discretization* means the automatic translation of a continuous model of the form (1) to a discrete model, that can be solved on a computer to obtain a discrete solution of (1).

The *automation of discrete solution* means the automatic choice of an appropriate algorithm for the solution of the discrete system of equations obtained from the automatic discretization of (1).

The *automation of error control* means that the resolution of the discrete model is automatically chosen to produce a discrete solution satisfying a given accuracy requirement with minimal work. This includes an aspect of *reliability* (the error of the computed solution should be less than a given tolerance), and an aspect of *efficiency* (the solution should be computed with minimal work).

The *automation of modeling* concerns the process of automatically finding the optimal parameters describing the model (1); either to find the model from a given set of data or to construct from a given model a *reduced model* for the variation of the solution on resolvable scales.

The *automation of optimization* relies on the automation of (i)–(iv), with the solution of the primal problem (1) and an associated dual problem being the key steps in the minimization of a given cost functional. In particular, the automation of optimization relies on the automatic generation of the dual problem.

We return to (i)–(v) below. In all cases, feed-back control, or *adaptivity*, plays a key role.

The Realization of the Automation of CMM

The Automation of CMM includes its own realization, i.e., a software system implementing the automation of (i)–(v). This is the goal of the **FEniCS** project, recently initiated in cooperation between Chalmers University of Technology and the University of Chicago [12]. The necessary prerequisites for this venture are now available, including the mathematical methodology, the modern computer, and modern tools of computer science.

FEniCS is based on the open-source computational platform **DOLFIN** [31, 33], developed at the Dept. of Computational Mathematics at Chalmers, currently working as a prototype implementation of **FEniCS**. We discuss the current status of **DOLFIN** below in Section 6.

FEniCS also includes a minimal and educational implementation of the basic tools for the Automation of CMM, called *Puffin*. Puffin is used in a number of courses at Chalmers, ranging from introductory undergraduate courses to advanced undergraduate/beginning graduate courses, and is discussed in more detail below in Section 7.

DOLFIN and Puffin constitute an essential part of this thesis.

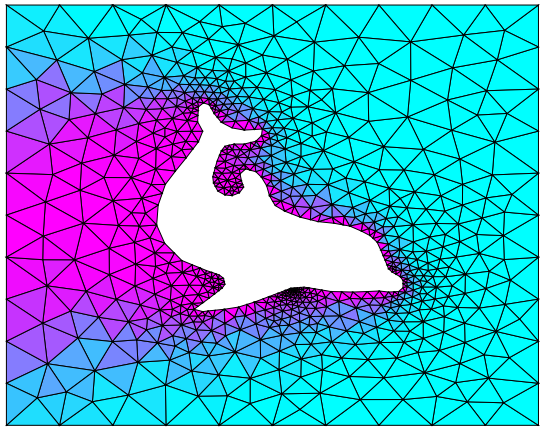

Figure 1: Solution of the equation $\dot{u} + b \cdot \nabla u - \nabla \cdot (a \nabla u) = 0$, modeling convection–diffusion around a hot dolphin.

1 The Automation of Discretization

The automation of discretization is based on *Galerkin's method*, providing a general framework for the discretization of differential equations.

In its basic form, Galerkin's method for (1) is given by specifying a pair of finite-dimensional subspaces $(V, \hat{V})$ and then formulating a discrete version of (1) for functions in the *trial space* V by testing (1) against all functions in the *test space* $\hat{V}$. The discrete version of (1) takes the form: Find $U \in V$, such that

$$(\dot{U}, v) = (f, v) \quad \forall v \in \hat{V}, \tag{3}$$

where $(\cdot, \cdot)$ is an inner product for functions on a domain $\Omega \times (0, T]$ in space–time.

Galerkin's method was originally formulated with global polynomial subspaces $(V, \hat{V})$ [4] and goes back to the variational principles of Leibniz, Euler, Lagrange, Dirichlet, Hamilton, Castigliano [6], Rayleigh, and Ritz [56]. Galerkin's method with piecewise polynomial subspaces $(V, \hat{V})$ is known as the *finite element method*. The finite element method was introduced by engineers for structural analysis in the 1950s and was independently proposed by Courant in 1943 [7]. The exploitation of the finite element method among engineers and mathematicians exploded in the 1960s. General references include [61, 54, 58, 13, 14, 19, 20, 18, 21, 22, 23, 2].

The Galerkin finite element method thus provides the basic tool for the automatic discretization of (1). The discrete model (3) represents a discrete system of (possibly nonlinear) equations. An important step in the Automation of CMM is thus the automated solution of these discrete equations, as discussed below in Section 2.

To automate this process, a *language* must be specified in which models of the general form (1) can be expressed and interpreted by a computer. With automatic discretization based on the Galerkin finite element method, it is convenient to specify models in the *language of variational forms* corresponding to (3). Both **DOLFIN** and Puffin implement this language, as discussed below. This allows a model to be specified in a notation which is close to the mathematical notation of (3). Furthermore, to complete the automation of discretization, an additional translation is necessary from a model given in the (strong) form (1) to a model given in the (weak) form (3).

With the automation of discretization made possible through the automation of the Galerkin finite element method, a necessary ingredient is the *automatic generation of finite elements*. Recent work [55] makes it possible to automatically generate a finite element, as defined in [54, 58], satisfying a given set of properties. New finite elements may thus be added

to a software system by just giving their definition. Another advantage is that by automating the generation of finite elements, it becomes easier to guarantee the *correctness* of the implementation, since the implementation is common for all types of finite elements. In addition, the finite elements may be chosen adaptively by feed-back from computation.

To give a concrete example of the automatic translation of (1) to a discrete model of the form (3), we consider the cG(1) finite element method in its basic form, and return below to the general multi-adaptive methods mcG(q) and mdG(q). We shall further assume that the model (1) takes the form of an initial value problem for a system of ordinary differential equations,

$$\begin{aligned} \dot{u}(t) &= f(u(t), t), \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{4}$$

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.

To obtain the discrete version of (4), we construct the trial space V and the test space $\hat{V}$ of (3) as follows. The time interval interval $(0, T]$ is partitioned into M subintervals, $\mathcal{T} = \{I_j\}_{j=1}^M$, with the length of each subinterval $I_j = (t_{j-1}, t_j]$ given by the *time step* $k_j = t_j - t_{j-1}$. As discussed further below in Section 3, the partition $\mathcal{T}$ is automatically determined by an adaptive algorithm with the goal of obtaining an optimal discrete representation U of the exact solution u.

Based on the partition $\mathcal{T}$, the trial and test spaces are now defined as follows:

$$\begin{aligned} V &= \{v \in [\mathcal{C}([0, T])]^N : v|_{I_j} \in [\mathcal{P}^1(I_j)]^N, \quad j = 1, \ldots, M\}, \\ \hat{V} &= \{v : v|_{I_j} \in [\mathcal{P}^0(I_j)]^N, \quad j = 1, \ldots, M\}, \end{aligned} \tag{5}$$

i.e., V represents the space of continuous and piecewise linear vector-valued functions on the partition $\mathcal{T}$ and $\hat{V}$ represents the space of (possibly discontinuous) piecewise constant vector-valued functions on the partition $\mathcal{T}$.

Following the general formulation (3), the cG(1) method for (4) now reads: Find $U \in V$ with $U(0) = u_0$, such that

$$\int_0^T (\dot{U}, v)\, dt = \int_0^T (f(U, \cdot), v)\, dt \quad \forall v \in \hat{V}, \tag{6}$$

where $(\cdot, \cdot)$ denotes the inner product on $\mathbb{R}^N$.

From (6), it follows that the cG(1) solution can alternatively be defined in terms of its nodal values, given by

$$U(t_j) = U(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(U(t), t)\, dt, \quad j = 1, \ldots, M, \tag{7}$$

with $U(0) = u_0$ and where $U(t) = U(t_{j-1}) + (t - t_{j-1})(U(t_j) - U(t_{j-1}))/k_j$ on each interval I_j. Note that (7) represents a (possibly nonlinear) system of equations for the degrees of freedom of the discrete solution U.

Multi-Adaptive Galerkin

> It is desirable, in short, that in things which do not primarily concern others, individuality should assert itself.
>
> John Stuart Mill (1806–1873), *On Liberty* (1859)

Standard methods for the discretization of (1) require that the temporal resolution is constant in space at each given time t. In particular, the cG(1) method for (4) requires that the same time step $k = k(t)$ is used for all components $U_i = U_i(t)$ of the solution U. This can be very costly if the system exhibits multiple time scales of different magnitudes. If the different time scales are localized in space, corresponding to a difference in time scales between the different components of the solution U of (4), efficient representation of the solution thus requires that this difference in time scales is reflected in the choice of discrete representation $(V, \hat{V})$. We refer to the resulting methods, recently introduced in a series of papers [44, 45, 48, 49, 36], as *multi-adaptive Galerkin methods*. See also [40, 50, 35, 37].

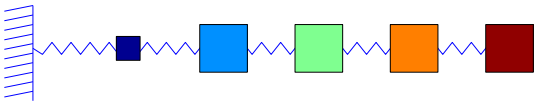

Figure 2: A simple mechanical system consisting of $n = 5$ particles exhibiting multiple time scales.

Surprisingly, individual time-stepping (multi-adaptivity) has previously received little attention in the large literature on numerical methods for ODEs, see e.g. [8, 29, 30, 5, 60, 1], but has been used to some extent for specific applications, including specialized integrators for the n-body problem [51, 9, 59], and low-order methods for conservation laws [53, 38, 10]. Early attempts at individual time-stepping include [62, 63]. Recently, a new class of related methods, known as asynchronous variational integrators (AVI) with individual time steps, has been introduced [42].

To extend the standard cG(q) and dG(q) methods for (4), we modify the trial space V and the test space $\hat{V}$ to allow individual time steps for each component. For $i = 1, \ldots, N$, the time interval $(0, T]$ is partitioned

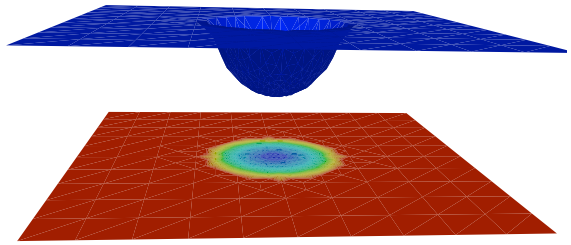

Figure 3: Local time steps in a multi-adaptive solution of the 2D heat equation with a variable heat source localized to the center of the domain.

into M_i subintervals, $\mathcal{T}_i = \{I_{ij}\}_{j=1}^{M_i}$, with the length of each subinterval $I_{ij} = (t_{i,j-1}, t_{ij}]$ given by the *local time step* $k_{ij} = t_{ij} - t_{i,j-1}$. The resulting individual elements $\{(I_{ij}, U_i|_{I_{ij}})\}$ are arranged in *time slabs*, collecting elements between common synchronized time levels $\{T_n\}_{n=0}^{M}$, as shown in Figure 4. As discussed in [35], the multi-adaptive algorithm consists of two phases: the generation of time slabs and the solution of the discrete equations on each time slab. We return to the iteration on time slabs below.

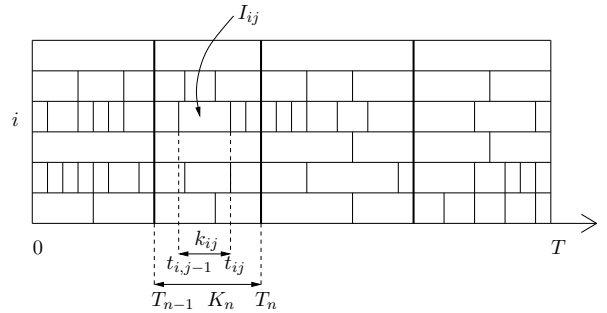

Figure 4: A sequence of $M = 4$ time slabs for a system of size $N = 6$.

For the multi-adaptive version of the standard cG(q) method, referred

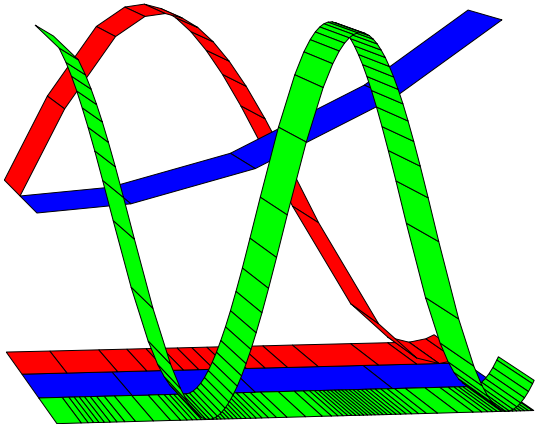

Figure 5: A multi-adaptive solution with individual time steps for the three different components.

to as the mcG(q) method, we define the trial and test spaces by

$$V = \{v \in [\mathcal{C}([0,T])]^N : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}), \quad j = 1,\ldots,M_i, \quad i = 1,\ldots,N\},$$
$$\hat{V} = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \quad j = 1,\ldots,M_i, \quad i = 1,\ldots,N\}. \tag{8}$$

In other words, V denotes the space of continuous piecewise polynomials of degree $q = q_i(t) = q_{ij}$, $t \in I_{ij}$, and $\hat{V}$ denotes the space of (possibly discontinuous) piecewise polynomials of degree $q-1$. This is illustrated in Figure 5, showing the mcG(1) solution of a system with three components with individual partitions for the different components. The trial and test spaces for the multi-adaptive version of the standard dG(q) method, referred to as the mdG(q) method, are defined similarly with discontinuous trial *and* test functions, see [44] or [48].

The mcG(q) method for (4) can now be defined as follows: Find $U \in V$ with $U(0) = u_0$, such that

$$\int_0^T (\dot{U}, v)\ dt = \int_0^T (f(U,\cdot), v)\ dt \quad \forall v \in \hat{V}. \tag{9}$$

Rewriting (9) as a sequence of successive local problems for each component,

we obtain

$$\int_{I_{ij}} \dot{U}_i v\, dt = \int_{I_{ij}} f_i(U,\cdot) v\, dt \quad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \tag{10}$$

for each local interval I_{ij}. For the solution of the discrete system given by (9), it is convenient to rewrite (10) in explicit form for the degrees of freedom $\{x_{ijm}\}_{m=0}^{q_{ij}}$ of the local polynomial $U_i|_{I_{ij}}$, corresponding to (7). This explicit form of (10) is given by

$$x_{ijm} = x_{ij0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\, f_i(U(t),t)\ dt, \quad m = 1,\ldots,q_{ij}, \tag{11}$$

where $\{w_m^{[q_{ij}]}\}_{m=1}^{q_{ij}} \subset \mathcal{P}^{q_{ij}-1}([0,1])$ is a set of polynomial weight functions and τ_{ij} maps I_{ij} to $(0,1]$: $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$.

An example of multi-adaptive time-stepping taken from [35] is presented in Figure 6, showing the solution and multi-adaptive time steps for the bistable equation (see [26, 57]),

$$\begin{aligned} \dot{u} - \epsilon\Delta u &= u(1-u^2) \quad \text{in } \Omega \times (0,T], \\ \partial_n u &= 0 \quad \text{on } \partial\Omega, \\ u(\cdot,0) &= u_0 \quad \text{in } \Omega, \end{aligned} \tag{12}$$

on the unit cube $\Omega = (0,1) \times (0,1) \times (0,1)$ with $T = 100$ and $\epsilon = 0.0001$.

2 The Automation of Discrete Solution

Depending on the model (1), the method used for the automatic discretization of (1), and the choice of discrete representation $(V, \hat{V})$, the *solution* of the resulting discrete system of equations may require more or less work.

Traditionally, the solution of the discrete system is obtained by some version of Newton's method, with the linearized equations being solved using direct methods based on Gaussian elimination, or iterative methods such as the conjugate gradient method (CG) or GMRES in combination with a suitable choice of problem-specific preconditioner.

Alternatively, direct fixed point iteration on the discrete equations (11) may be used. The fixed point iteration takes the form

$$x^n = g(x^{n-1}) = x^{n-1} - (x^{n-1} - g(x^{n-1})) = x^{n-1} - F(x^{n-1}), \tag{13}$$

for $n = 1, 2, \ldots$, converging to a unique solution of the equation $F(x) \equiv x - g(x) = 0$ if the Lipschitz constant L_g of g satisfies $L_g < 1$.

If the number of iterations remains small, fixed point iteration is a competitive alternative to standard Newton iteration. It is also an attractive

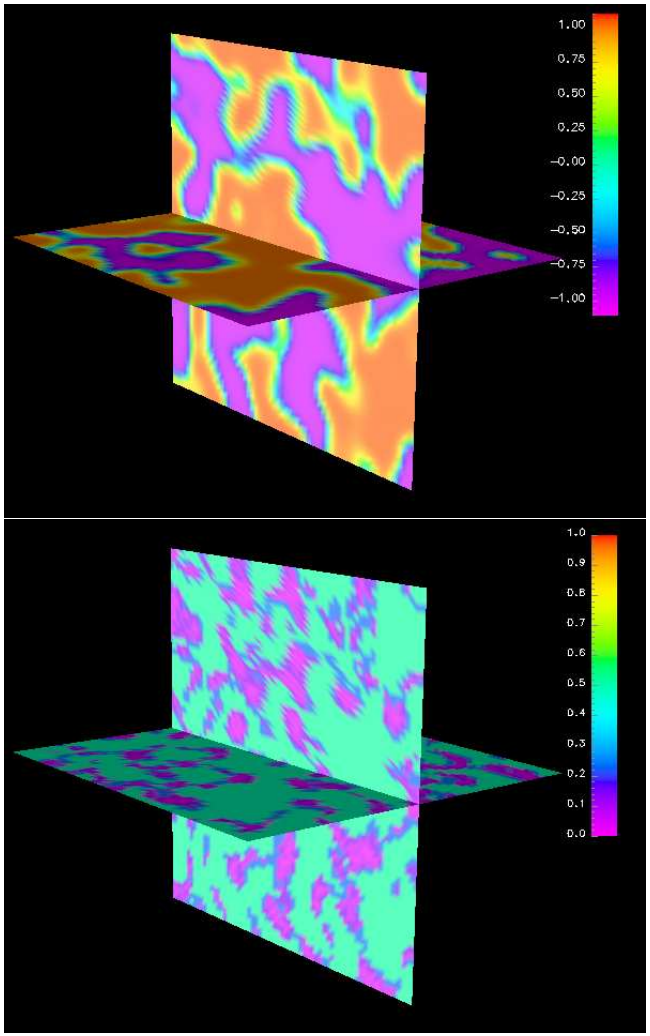

Figure 6: Cross-section of solution (above) and multi-adaptive time steps (below) at time $t = 10$ for the bistable equation (12).

alternative for the iterative solution of the discrete equations on time slabs in multi-adaptive methods, since it avoids the formation of large Jacobians on time slabs. However, if the problem being solved is stiff, which is characterized by a large Lipschitz constant $L_g > 1$, simple fixed point iteration may fail to converge.

In *adaptive fixed point iteration*, the simple fixed point iteration (13) is modified according to

$$x^n = (I - \alpha)x^{n-1} + \alpha g(x^{n-1}) = x^{n-1} - \alpha F(x^{n-1}), \tag{14}$$

with adaptively determined damping $\alpha : \mathbb{R}^N \to \mathbb{R}^N$. In [36], this idea is explored for the iterative solution of the discrete equations on time slabs, based on ideas for the adaptive stabilization of stiff systems presented in [24], relying on the inherent property of the stiff problem itself: rapid damping of high frequencies. The suitable damping α is automatically determined in each iteration, which in particular means that $\alpha = I$ for a non-stiff problem.

3 The Automation of Error Control

Ideally, an algorithm for the solution of (1) should have the following properties: Given a tolerance $\text{TOL} > 0$ and a functional L, the algorithm shall produce a discrete solution U approximating the exact solution u of (1), such that

(A) $|L(e)| \leq \text{TOL}$, where $e = U - u$;

(B) the computational cost of obtaining the approximation U is minimal.

The functional L is typically chosen to represent the error e in some output quantity of interest, for example the error in the drag coefficient of an object such as a car or an airplane in a numerical flow simulation. Other natural choices include $L(e) = \|e(T)\|$ for a suitable choice of norm $\|\cdot\|$ for functions on Ω. Conditions (A) and (B) are satisfied by an *adaptive algorithm*, with the construction of the discrete representation $(V, \hat{V})$ based on feed-back from the computed solution.

As discussed above, an adaptive algorithm typically involves a stopping criterion, indicating that the size of the error is less than the given tolerance, and a modification strategy to be applied if the stopping criterion is not satisfied. Often, the stopping criterion and the modification strategy are based on an *a posteriori* error estimate $E \geq |L(e)|$, estimating the error $e = U - u$ in terms of the residual $R(U) = \dot{U} - f(U)$ and the solution ϕ of a continuous dual problem connecting to the stability of (1).

Alternatively, the stopping criterion and the modification strategy can be based on an *a priori* error estimate involving the exact solution u of (1) and the solution Φ of a discrete dual problem.

To estimate the error, either a priori or a posteriori, knowledge of the stability of the problem being solved is thus required. These stability properties are in general obtained by solving the dual problem of (1), which takes the form

$$-\dot{\phi} = (\partial f/\partial u)^* \phi, \tag{15}$$

on $\Omega \times [0,T)$, where $(\partial f/\partial u)^*$ denotes the adjoint of the derivative of f. By computing the solution of the dual problem (15), the required stability properties can be obtained e.g. in the form of a *stability factor* $S(T)$, typically involving derivatives of the dual solution,

$$S(T) = \int_0^T \|\phi^{(q)}\| \, dt. \tag{16}$$

Error control thus requires some extra effort from the adaptive algorithm, the solution of a dual problem in addition to the solution of the primal problem (1), which in most cases is comparable to the effort of solving the primal problem itself.

The Adaptive Algorithm

The basic adaptive algorithm for the solution of (1) can now be expressed as follows:

1. Choose an initial discrete representation $(V, \hat{V})$ for (1).
2. Solve the discrete problem (3) to obtain the discrete solution U approximating the exact solution u of (1).
3. Solve the dual problem (15) and compute the stability factor $S(T)$.
4. Compute an error estimate E based on the residual $R(U)$ and the stability factor $S(T)$.
5. If $E \leq$ TOL, then stop. If not, refine the discrete representation $(V, \hat{V})$ based on the residual $R(U)$ and the stability factor $S(T)$, and go back to 2.

As a concrete example, we consider below the cG(1) method for the basic model problem (4). We derive a priori and a posteriori error estimates and indicate how these estimates fit into the basic adaptive algorithm, following earlier work on adaptivity for the cG(q) and dG(q) methods [11, 39, 28, 27].

A Priori Error Estimates

To obtain an a priori error estimate for the discrete cG(1) solution U of (4), the error $e = U - u$ is represented in terms of a discrete dual solution Φ and the residual of an interpolant πu of the exact solution u of (4). The a priori error estimate then follows by subtracting the residual of the exact solution u in the representation of the error, together with an interpolation estimate.

Assuming that the functional L can be expressed as

$$L(e) = (e(T), \psi) + \int_0^T (e, g) \, dt, \tag{17}$$

for given *dual data* (ψ, g), the discrete dual problem of (6) takes the following form: Find $\Phi \in \hat{V}$ with $\Phi(T^+) = \psi$, such that

$$\int_0^T (\dot{v}, \Phi) \, dt = \int_0^T (J(\pi u, U, \cdot)v, \Phi) \, dt + L(v) \tag{18}$$

for all $v \in V$ with $v(0) = 0$, where $J(\pi u, U, \cdot) = \int_0^1 \partial f/\partial u(s\pi u + (1-s)U, \cdot) \, ds$. Integrating by parts, we note that (18) represents a Galerkin method for (15), see [48] for details.

Letting πu denote the piecewise linear nodal interpolant of u, i.e., $\pi u(t_j) = u(t_j)$ for $j = 0, 1, \ldots, M$, we note that $\bar{e} \equiv U - \pi u \in V$ and so if follows from (18), that $L(\bar{e}) = \int_0^T (\dot{\bar{e}} - J(\pi u, U, \cdot)\bar{e}, \Phi) \, dt = \int_0^T (R(U, \cdot) - R(\pi u, \cdot), \Phi) \, dt$ and thus that

$$L(\bar{e}) = -\int_0^T (R(\pi u, \cdot), \Phi) \, dt, \tag{19}$$

where we have used the *Galerkin orthogonality* (6), expressing that the residual $R(U, \cdot) = \dot{U} - f(U, \cdot)$ is orthogonal to the test space $\hat{V}$.

Subtracting the residual $R(u, \cdot) = \dot{u} - f(u, \cdot) = 0$ of the exact solution u of (4) in the *error representation* (19), it now follows that

$$\begin{aligned} L(\bar{e}) &= -\int_0^T (R(\pi u, \cdot) - R(u, \cdot), \Phi) \, dt \\ &= -\int_0^T (\dot{\pi} u - \dot{u}, \Phi) \, dt + \int_0^T (f(\pi u, \cdot) - f(u, \cdot), \Phi) \, dt \\ &= \int_0^T (f(\pi u, \cdot) - f(u, \cdot), \Phi) \, dt = \int_0^T (\pi u - u, J^*(\pi u, u, \cdot)\Phi) \, dt, \end{aligned} \tag{20}$$

since πu interpolates u at each t_j and Φ is constant on each interval I_j. The a priori error estimate now follows by an interpolation estimate,

$$|L(\bar{e})| \leq CS(T) \|k^2 \ddot{u}\|_{L_\infty([0,T], l_2)}, \tag{21}$$

where C is an interpolation constant and the stability factor $S(T)$ is given by

$$S(T) = \int_0^T \|J^*(\pi u, u, \cdot)\Phi\|_{l_2}\, dt. \tag{22}$$

Noting now that $e = \bar{e}$ at each node t_j, it follows from the a priori error estimate (21) that the cG(1) method is of second order in the time step k. In general, the cG(q) method is of order $2q$, see [49]. Similarly, an a priori error estimate can be obtained for the dG(q) method, showing that the mdG(q) method is of order $2q+1$.

Note that the a priori error estimate (21) does in fact take the form of a *quantitative* estimate, i.e., it involves computable quantities (if the exact solution u is approximated with the discrete solution U), including the stability factor $S(T)$ which may be obtained from a computed solution of the discrete dual problem (18).

A Posteriori Error Estimates

To obtain an a posteriori error estimate for the discrete cG(1) solution U of (4), the error is represented in terms of a continuous dual solution ϕ and the residual of the discrete solution U. The a posteriori error estimate then follows by subtracting an interpolant of the dual solution, together with an interpolation estimate.

The continuous dual problem of (4) takes the form

$$\begin{aligned} -\dot{\phi}(t) &= J^*(u, U, t)\phi(t) + g(t), \quad t \in [0, T), \\ \phi(T) &= \psi. \end{aligned} \tag{23}$$

Noting that the dual solution ϕ of (23) satisfies (18) for all v such that $v(0) = 0$ with $J(\pi u, U, \cdot)$ replaced by $J(u, U, \cdot)$, and thus in particular for $v = e$, it follows that $L(e) = \int_0^T (\dot{e} - J(u, U, \cdot)e, \phi)\, dt = \int_0^T (R(U, \cdot) - R(u, \cdot), \phi)\, dt$ and thus that

$$L(e) = \int_0^T (R(U, \cdot), \phi)\, dt. \tag{24}$$

Using the Galerkin orthogonality (6) to subtract a piecewise constant interpolant $\pi\phi$ of the dual solution ϕ in the *error representation* (24), we obtain the following a posteriori error estimate for the cG(1) method:

$$|L(e)| = \left| \int_0^T (R(U, \cdot), \phi - \pi\phi)\, dt \right| \le CS(T)\|kR(U, \cdot)\|_{L_\infty([0,T],l_2)}, \tag{25}$$

where C is an interpolation constant and the stability factor $S(T)$ is given by

$$S(T) = \int_0^T \|\dot{\phi}\|_{l_2}\, dt. \tag{26}$$

Note the similarities between the a priori error estimate (21) and the a posteriori error estimate (25), and also the two stability factors $S(T)$ defined in (22) and (26), showing two sides of the same coin. In particular, we note that by (6), the derivative $\dot{U}$ of the computed solution U is an interpolant (the projection onto piecewise constants) of the right-hand side $f(U, \cdot)$, and thus the residual $R(U, \cdot) = \dot{U} - f(U, \cdot)$ satisfies an estimate of the form $\|kR(U, \cdot)\|_{L_\infty([0,T],l_2)} \le C\|k^2\dot{f}(U, \cdot)\|_{L_\infty([0,T],l_2)} \sim \|k^2\ddot{u}\|_{L_\infty([0,T],l_2)}$.

We may now give concrete meaning to the adaptive algorithm discussed above, which may be based either on the a priori error estimate (21) or the a posteriori error estimate (25). Basing the adaptive algorithm on (25), the error estimate E is given by

$$E = CS(T)\|kR(U, \cdot)\|_{L_\infty([0,T],l_2)}. \tag{27}$$

This error estimate for the computed solution U is thus compared to the given tolerance TOL and if $E >$ TOL, a better approximation U needs to be computed, this time with smaller time steps. The new time steps are determined adaptively from the error estimate (27), i.e., we take $k = k(t)$ such that

$$\|kR(U, \cdot)\|_{L_\infty(I_j,l_2)} = \text{TOL}/(CS(T)), \tag{28}$$

for each interval I_j. This process is then repeated until $E <$ TOL. Note that several extensions to this simple strategy are possible. In particular, a sharper error estimate E may be obtained directly from (24), as discussed in [44], and stability *weights* may be used instead of stability factors.

The Stability Factor and the Dual Problem

The size of the stability factor $S(T)$ varies with the model (1), the method used for the automatic discretization of (1), and the chosen output quantity of interest represented by the functional L. The presence of a large stability factor indicates that the problem is sensitive to small perturbations, and pertinent questions concern the *predictability* (influence of perturbations in data and model) and the *computability* (influence of errors introduced by the discretization) of a given model.

A common classification of partial differential equations uses the terms *elliptic*, *parabolic*, and *hyperbolic*, with the prototype examples being Poisson's equation, the heat equation, and the wave equation, respectively. More generally, parabolic problems are often described in vague terms as being dominated by diffusion, while hyperbolic problems are dominated by convection in a setting of systems of convection-diffusion equations.

In the context of computational methods for a general class of models of the form (1), the notion of parabolicity may be given a precise quantitative

definition. We define a model of the form (1) to be *parabolic* if computational solution is possible over long time without error accumulation, i.e., if the stability factor $S(T)$ remains bounded and of moderate size as T increases. A more detailed discussion on this subject can be found in [25]. For a typical hyperbolic problem, the corresponding stability factor $S(T)$ grows linearly in time, while for more general models the growth may be polynomial or even exponential in time.

We consider two basic examples with very different stability properties; the first one is a parabolic system of chemical reactions and the second one is the Lorenz system which has an exponentially growing stability factor.

The first example is the following system of reaction–diffusion equations, discussed in more detail in [57, 45]:

$$\begin{cases} \dot{u}_1 - \epsilon\Delta u_1 = -u_1 u_2^2, \\ \dot{u}_2 - \epsilon\Delta u_2 = u_1 u_2^2, \end{cases} \tag{29}$$

on $\Omega \times (0, 100]$ with $\Omega = (0, 1) \times (0, 0.25)$, $\epsilon = 0.0001$, and homogeneous Neumann boundary conditions. With the initial conditions chosen to obtain a *reaction front* moving through the domain Ω, the stability factor for the computation of the solution at final time at a given point in Ω grows as shown in Figure 7. We note that the stability factor peaks at the time of active reaction, and that before and after the reaction front has swept the region of observation, the stability factor $S(T)$ is significantly smaller, indicating parabolicity for large values of T.

Our second example is the Lorenz system, given by

$$\begin{cases} \dot{x} &= \sigma(y - x), \\ \dot{y} &= rx - y - xz, \\ \dot{z} &= xy - bz, \end{cases} \tag{30}$$

with $\sigma = 10$, $b = 8/3$, $r = 28$, and initial value $(x(0), y(0), z(0)) = (1, 0, 0)$. The solution $u(t) = (x(t), y(t), z(t))$, shown in Figure 8 for $T = 40$, is very sensitive to perturbations and is often said to be *chaotic*. With our perspective, this is reflected by the rapid growth of the stability factor $S(T)$, as shown in Figure 9. The stability factor grows on the average as $10^{T/3}$, which limits the computability of the Lorenz system to time $T = 50$ with standard double precision arithmetic.

Multi-Adaptivity

The a priori and a posteriori error analysis presented above extends naturally to the general multi-adaptive methods mcG(q) and mdG(q), as shown in [44, 45, 48, 49]. In particular, we obtain a priori error estimates showing

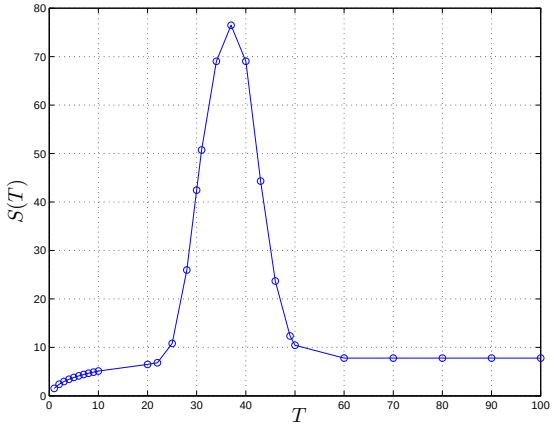

Figure 7: The growth of the stability factor $S(T)$ for the reaction front problem (29).

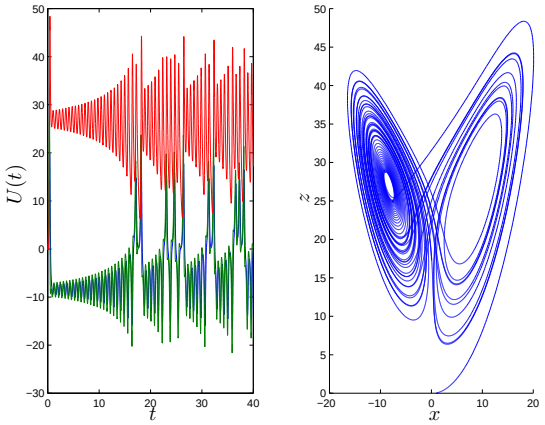

Figure 8: The solution of the Lorenz system (30) on the time interval $[0, 40]$.

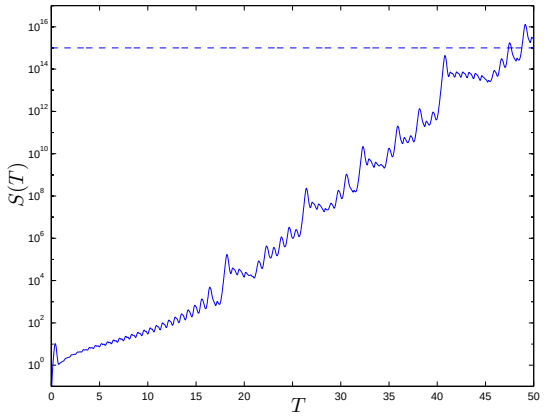

Figure 9: The exponential growth of the stability factor $S(T)$ connecting to modeling/round-off errors for the Lorenz system (30).

that the mcG(q) and mdG(q) methods are of order $2q$ and $2q+1$ in the local time step k_{ij}, respectively.

The main challenge in the error analysis of multi-adaptive methods lies in the proof of the a priori error estimates. As for the cG(1) method, we obtain an error representation of the form (20). For $q > 1$, the interpolant πu is chosen to satisfy a set of $q-1$ projection conditions in addition to interpolating u at each nodal point. It follows that we may subtract an interpolant $\pi J^*(\pi u, u, \cdot)\Phi$ of $J^*(\pi u, u, \cdot)\Phi$ in (20) to obtain an additional $q-1$ powers of k in the error estimate. Together with the $q+1$ powers of k obtained from the interpolation estimate for the difference $\pi u - u$, we thus obtain a total of $2q$ powers of k. However, since individual time steps are used for the different components of the multi-adaptive discrete dual solution Φ, the interpolation estimate for $\pi J^*(\pi u, u, \cdot)\Phi - J^*(\pi u, u, \cdot)\Phi$ is not straightforward. This interpolation estimate is proved in [46], based on basic interpolation estimates for piecewise smooth functions proved in [47].

The individual multi-adaptive time steps are determined based on an a posteriori error estimate of the form

$$|L(e)| \leq C \sum_{i=1}^{N} S_i(T) \|k_i^{q_i} R_i(U, \cdot)\|_{L_\infty([0,T])}, \tag{31}$$

corresponding to (25). Note that each component has an individual stability factor $S_i(T)$. The local time steps k_{ij} are determined automatically based on this posteriori error estimate, as discussed in detail in [45, 35].

4 The Automation of Modeling

The automation of modeling concerns both the problem of finding the parameters describing the model (1) from a given set of data (inverse modeling), and the automatic construction of a reduced model for the variation of the solution on resolvable scales (model reduction). We here discuss briefly the *automation of model reduction*.

In situations where the solution u of (1) varies on time scales of different magnitudes, and these time scales are not localized in space, computation of the solution may be very expensive, even with a multi-adaptive method.

To make computation feasible, we instead seek to compute an *average* $\bar{u}$ of the solution u of (1) on resolvable scales. Typical examples include meteorological models for weather prediction, with fast time scales on the range of seconds and slow time scales on the range of years, or protein folding represented by a molecular dynamics model, with fast time scales on the range of femtoseconds and slow time scales on the range of microseconds.

The automation of model reduction typically involves extrapolation from resolvable scales, or the construction of a large-scale model from local resolution of fine scales in time and space. In both cases, a large-scale model for the average $\bar{u}$ is constructed from the given model (1).

In the setting of the basic model problem (4), we thus seek to determine a model (equation) for the average $\bar{u}$ of u defined by

$$\bar{u}(t) = \frac{1}{\tau} \int_{-\tau/2}^{\tau/2} u(t+s)\, ds, \tag{32}$$

where τ is the size of the average (with a suitable construction near $t = 0$ and $t = T$). The model for the average $\bar{u}$ is obtained by taking the average of (4):

$$\dot{\bar{u}}(t) = \bar{\dot{u}}(t) = \overline{f(u,\cdot)}(t) = f(\bar{u}(t), t) + (\overline{f(u,\cdot)}(t) - f(\bar{u}(t), t)), \tag{33}$$

or

$$\dot{\bar{u}}(t) = f(\bar{u}(t), t) + \bar{g}(u, t), \tag{34}$$

where the variance $\bar{g}(u,t) = \overline{f(u,\cdot)}(t) - f(\bar{u}(t), t)$ accounts for the effect of small scales on resolvable scales.

The key step is now to model the variance $\bar{g}$ in terms of the average $\bar{u}$, i.e., to determine the *subgrid model* $\tilde{g}$ such that $\bar{g}(u,t) \approx \tilde{g}(\bar{u}(t), t)$ and

replace (4) with the reduced model

$$\begin{aligned} \dot{\bar{u}}(t) &= f(\bar{u}(t),t) + \tilde{g}(\bar{u}(t),t), \quad t \in (0,T], \\ \bar{u}(0) &= \bar{u}_0. \end{aligned} \tag{35}$$

The validity of the subgrid model may be estimated by computing the *modeling residual* $\tilde{g} - \bar{g}$ and stability factors obtained from the solution of the dual problem.

In [34], we explore this idea for a simple model problem by determining the solution $u(t)$ of (4), including fast time scales, accurately over a short time period and then matching a model of the form $\tilde{g}(\bar{u}(t),t)$ based on the computed solution.

5 The Automation of Optimization

With the automation of computation realized in the form of automated discretization, discrete solution, error control, and modeling, new possibilities open for the automation of optimization.

In optimization, one seeks to determine the value of a control parameter p which minimizes a given cost functional $\mathcal{J}(u,p)$, depending on the solution u of the model (1), with the model depending on the control parameter p. In the setting of the basic model problem (4), we thus seek to find the control parameter $p : [0,T] \to \mathbb{R}^M$ minimizing $\mathcal{J}(u,p)$, with u satisfying

$$\begin{aligned} \dot{u}(t) &= f(u(t),p(t),t), \quad t \in (0,T], \\ u(0) &= u_0. \end{aligned} \tag{36}$$

The optimization problem can be formulated as the problem of finding a stationary point of the associated Lagrangian,

$$L(u,p,\phi) = \mathcal{J}(u,p) + (\dot{u} - f(u,p,\cdot),\phi), \tag{37}$$

which takes the form of a system of differential equations, involving the primal and the dual problem, as well as an equation expressing stationarity with respect to variation of control variables:

$$\begin{aligned} \dot{u} &= f(u,p,\cdot), \\ -\dot{\phi}(t) &= (\partial f/\partial u)^*\phi - \partial\mathcal{J}/\partial u, \\ \partial\mathcal{J}/\partial p &= (\partial f/\partial p)^*\phi, \end{aligned} \tag{38}$$

together with initial values $u(0) = u_0$ and $\phi(T) = 0$ for the primal and dual problems, respectively. The optimization problem may thus be solved by computational solution of a system of differential equations.

The automation of optimization thus relies on the automated solution of both the primal problem (1) and the dual problem (15), including the automatic generation of the dual problem.

6 DOLFIN

DOLFIN is a platform for research and teaching in adaptive Galerkin finite element methods developed by Hoffman and Logg, with help from a team of graduate and undergraduate students at the Dept. of Computational Mathematics at Chalmers. **DOLFIN** functions as the current prototype implementation of **FEniCS**, aiming ultimately at the Automation of CMM.

Currently implemented features of **DOLFIN** include the automatic evaluation of variational forms, automatic assembly of the discrete system representing a given variational form (the automation of discretization), adaptive mesh refinement in two and three space dimensions, a number of algebraic solvers including preconditioners, multigrid, the ability to export a computed solution to a variety of visualization systems, a system for easy integration of new solvers/modules for specific problems, and a multi-adaptive ODE-solver with adaptive fixed point iteration for the solution of the discrete equations on time slabs (the automation of discrete solution), automatic validation of solutions through automatically generated dual problems (the automation of error control), and a basic implementation of the automatic generation of reduced models (the automation of model reduction).

DOLFIN is implemented as a C++ library and can be used either as a stand-alone solver, or as a tool for the development and implementation of new methods. To simplify usage and emphasize structure, **DOLFIN** is organized into three levels of abstraction, referred to as *kernel level*, *module level*, and *user level*, as shown in Figure 10. Core features, such as the automatic evaluation of variational forms and adaptive mesh refinement, are implemented as basic tools at kernel level. At module level, new solvers/modules can be assembled from these basic tools and integrated into the system. At user level, a model of the form (1) is specified and solved, either using one of the built-in solvers/modules or by direct usage of the basic tools.

Automatic Evaluation of Variational Forms

Automatic evaluation of variational forms is implemented by operator overloading in C++, allowing simple specification of variational forms in a language that is close to the mathematical notation. Performance is ensured by automatic precomputation and tabulation of integrals, in combination with special techniques to avoid object construction.

Consider as an example Poisson's equation

$$-\Delta u(x) = f(x), \tag{39}$$

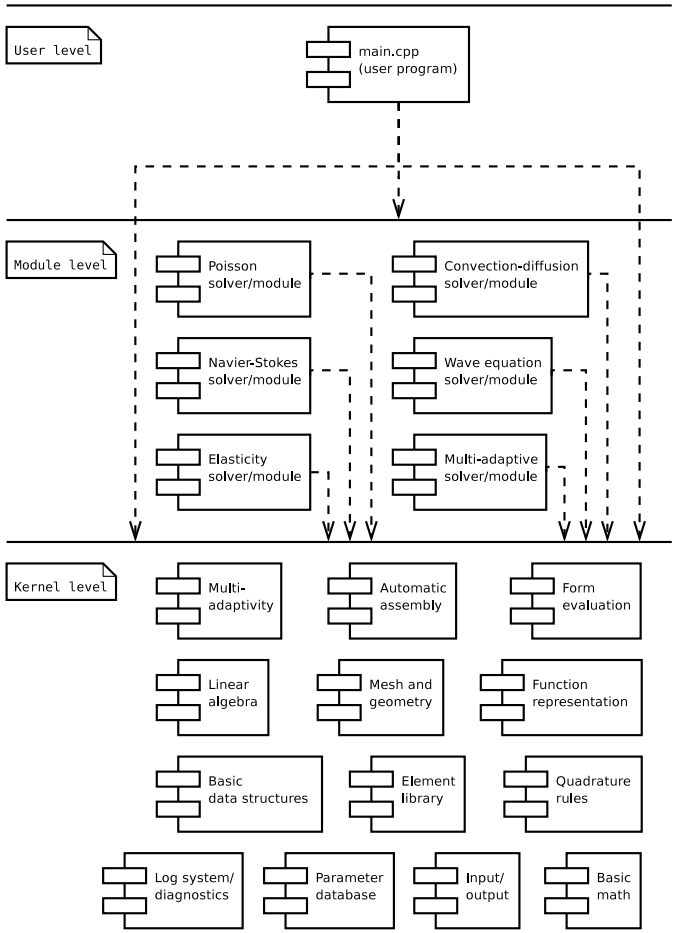

Figure 10: Simplified component diagram showing the modularized structure of **DOLFIN**.


```
class Poisson : PDE
{
  real lhs(ShapeFunction u, ShapeFunction v)
  {
    return (grad(u),grad(v)) * dx;
  }
  real rhs(ShapeFunction v)
  {
    return f*v * dx;
  }
}
```

Table 1: Sketch of the specification of the variational formulation of Poisson's equation in **DOLFIN**.

on some domain Ω. Assuming homogeneous Dirichlet boundary conditions, the variational (weak) formulation of (39) takes the form: Find $u \in H_0^1(\Omega)$, such that

$$\int_\Omega (\nabla u, \nabla v)\, dx = \int_\Omega f v\, dx \quad \forall v \in H_0^1(\Omega). \tag{40}$$

The specification of (40) in **DOLFIN** is given in Table 1, and consist of the specification of the two variational forms `lhs` and `rhs`, representing the left- and right-hand sides of (40), respectively.

Automatic Assembly

DOLFIN automatically assembles the discrete system representing a given variational formulation, based on the automatic evaluation of variational forms. This automates a large part of the implementation of a solver. In the case of Poisson's equation, the algorithm becomes particularly simple: assemble the discrete system consisting of a matrix (the *stiffness matrix*) and a vector (the *load vector*), and solve the linear system.

Automation of all parts of **DOLFIN** makes the implementation short, simple and clear at all levels. In fact, the algorithm for the automatic assembly is just a couple of lines, as shown in Table 2.

Adaptive Mesh Refinement

The concept of a *mesh* is central in the implementation of adaptive Galerkin finite element methods for partial differential equations. Other important concepts include *nodes*, *cells*, *edges*, *faces*, *boundaries*, and *mesh hierarchies*.

```
for (CellIterator cell(mesh); !cell.end(); ++cell)
  for (TestFunctionIterator v(element); !v.end(); ++v)
    for (TrialFunctionIterator u(element); !u.end(); ++u)
      A(v.dof(), u.dof()) += pde.lhs(u,v);

for (CellIterator cell(mesh); !cell.end(); ++cell)
  for (TestFunctionIterator v(element); !v.end(); ++v)
    b(v.dof) += pde.rhs(v);
```

Table 2: Sketch of the implementation of automatic assembly in **DOLFIN**.

```
for (CellIterator c(m); !c.end(); ++c)
  for (NodeIterator n1(c); !n1.end(); ++n1)
    for (NodeIterator n2(n1); !n2.end(); ++n2)
      cout << *n2 << endl;
```

Table 3: Iteration over all node neighbors `n2` of the nodes `n1` within all cells `c` of the mesh `m`.

These concepts are all implemented as C++ classes in **DOLFIN**, as shown in Figure 11.

Algorithms operating on a mesh, including adaptive mesh refinement, can often be expressed in terms of *iterators*, i.e., objects used for the traversal of aggregate structures, such as the list of nodes contained in a mesh. Iterators implemented in **DOLFIN** include a `NodeIterator`, `CellIterator`, `EdgeIterator`, `FaceIterator`, and a `MeshIterator`. Table 3 provides an example of a code that displays all node neighbors of all nodes of all cells within a given mesh.

Adaptive mesh refinement is implemented in **DOLFIN** for triangular meshes (in 2D) and tetrahedral meshes (in 3D), see Figure 12, based on the algorithm given in [3]. To refine a mesh, the cells (triangles or tetrahedrons) are first marked according to some criterion for refinement, before the mesh is refined. A hierarchy of meshes, that can be used for example in a multigrid computation, is automatically created.

Linear Algebra

DOLFIN includes an efficient implementation of the basic concepts of linear algebra: matrices and vectors. Both sparse and dense matrices are implemented, as well as generic matrices only defined through their action (multiplication with a given vector).

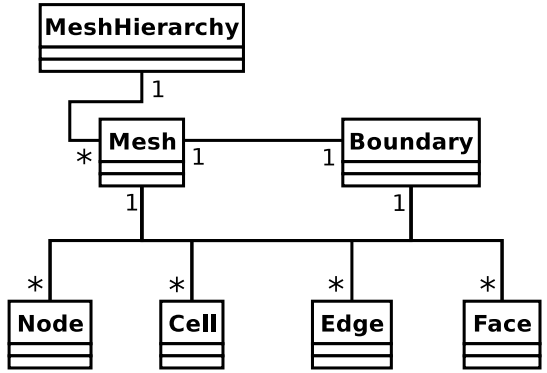

Figure 11: Class diagram of the basic mesh classes in **DOLFIN**.

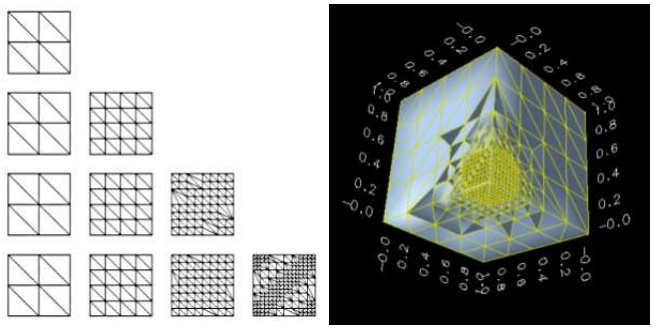

Figure 12: Adaptive mesh refinement of triangular and tetrahedral meshes in **DOLFIN**.

Several algebraic solvers are implemented in **DOLFIN**. These include preconditioned iterative methods such as CG, GMRES, and BiCGSTAB, and direct methods such as LU factorization.

Multigrid

A multigrid solver has recently been implemented in **DOLFIN** as part of a student project, showing the benefits of the open-source nature of **DOLFIN**; it functions as a tool in both research and education, and at the same time benefits from this usage by incorporating new methods and solvers developed as a consequence of its usage.

Visualization

DOLFIN relies on interaction with external tools for visualization, such as the open-source program *OpenDX*, based on IBM's Visualization Data Explorer. Post-processing (as well as pre-processing) is thus accomplished by implementation of the file formats needed for exchange of data. Using a modular approach, **DOLFIN** has been designed to allow easy extension by addition of new file formats. In addition to OpenDX, **DOLFIN** currently supports GNU Octave, MATLAB, and GiD.

Easy Integration of New Modules

New solvers/modules for specific models may easily be added to **DOLFIN**. Current modules include solvers for Poisson's equation, the heat equation, convection–diffusion, the wave equation, linear elasticity, incompressible Navier–Stokes (in preparation), and the general multi-adaptive ODE-solver.

Multi-Adaptive ODE-Solver

A general multi-adaptive ODE-solver, including automatic generation and solution of dual problems, automatic error estimation, and automatic model reduction, is currently being implemented in **DOLFIN**, as the result of a merge with the existing multi-adaptive solver *Tanganyika* [43]. The new solver is developed with specific focus on the ability to handle stiff problems and large-scale applications, as discussed in [36, 35, 37].

Limitations and Future Directions

To reach a point where **DOLFIN** becomes attractive and useful to a wider audience, and ultimately to realize the Automation of CMM, a number of new features and improvements of existing features are necessary, including all features listed above. Important areas of active development which will

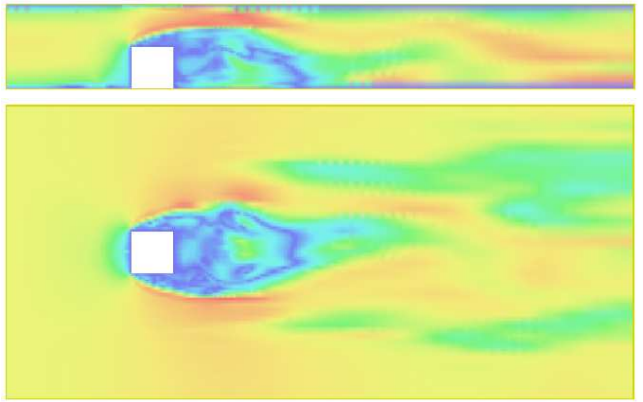

Figure 13: Cross-sections of 3D turbulent flow around a surface mounted cube computed with **DOLFIN**. (Courtesy of Johan Hoffman)

see improvements in the near future include the automatic evaluation of variational forms, support for general finite elements, support for a range of file formats, linear algebra and iterative solvers, specification of general boundary conditions, and the automatic integration of time-dependent partial differential equations using the multi-adaptive solver.

7 Puffin

With new tools available to science and engineering based on the Automation of CMM, a reform of education from basic to graduate level becomes necessary. The Body&Soul-project, involving books [32, 15, 16, 17, 41] and software (**FEniCS**), represents the first coordinated effort to meet these demands. The potential impact of **FEniCS** as the computational tool in a reformed education is thus very strong.

Puffin is a simple and minimal 2D implementation of **FEniCS** in GNU Octave (or MATLAB) designed for education, including automatic assembly and evaluation of variational forms. Puffin is currently used by students in a number of courses at Chalmers. The students follow a sequence of computer sessions and in each session implement a solver for a given model, including Poisson's equation, the heat equation, convection–diffusion, the

bistable equation, the Navier–Stokes equations, and general systems of convection–diffusion–reaction equations (see Figure 14 and 15).

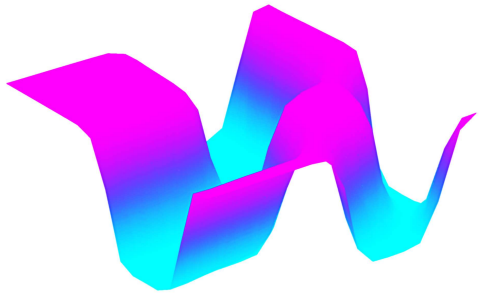

Figure 14: Solution of the bistable equation (12) in Puffin.

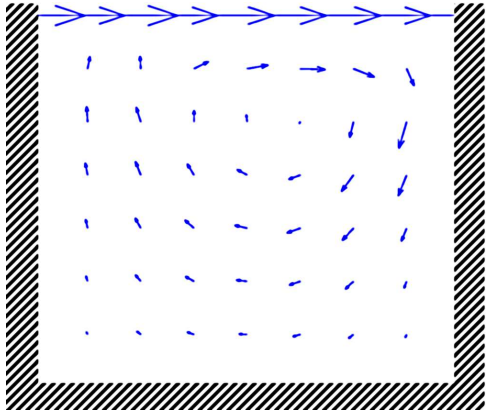

Figure 15: Solution of the driven cavity problem for the Navier–Stokes equations in Puffin.

8 Summary of Appended Papers

PAPER I: ***Multi-Adaptive Galerkin Methods for ODEs I***

This paper introduces the multi-adaptive Galerkin methods mcG(q) and mdG(q) for initial value problems for ordinary differential equations and contains the a posteriori error analysis of the methods.

PAPER II: ***Multi-Adaptive Galerkin Methods for ODEs II: Implementation and Applications***

Continuing from PAPER I, this paper discusses the implementation of the mcG(q) and mdG(q) methods, in particular the adaptive algorithm based on the a posteriori error estimates from PAPER I. This paper also discusses a number of applications of the multi-adaptive methods, and includes an evaluation of the performance of the multi-adaptive methods.

PAPER III: ***Multi-Adaptive Galerkin Methods for ODEs III: Existence and Stability***

This paper contains proofs of existence and stability for the discrete multi-adaptive solutions, which is an important step in the proof of a priori error estimates for the multi-adaptive methods.

PAPER IV: ***Multi-Adaptive Galerkin Methods for ODEs IV: A Priori Error Estimates***

Based on the stability estimates of PAPER III and a pair of special interpolation estimates proved in PAPER VII and PAPER VIII, this paper contains the proofs of the a priori error estimates for the multi-adaptive methods mcG(q) and mdG(q).

PAPER V: ***Multi-Adaptive Galerkin Methods for ODEs V: Stiff Problems***

This paper extends the applicability of the multi-adaptive methods to stiff problems. In particular, a new strategy for adaptive fixed point iteration on time slabs is presented, based on ideas of adaptive stabilization of stiff problems presented in PAPER VI.

PAPER VI: ***Explicit Time-Stepping for Stiff ODEs***

This paper presents new ideas for adaptive stabilization of stiff problems, making it possible to solve stiff problems with explicit methods.

PAPER VII: ***Interpolation Estimates for Piecewise Smooth Functions in One Dimension***

This paper contains basic interpolation estimates for piecewise smooth functions. These estimates are used in the a priori error analysis of the multi-adaptive methods, where interpolation estimates are needed for functions with discontinuities.

PAPER VIII: ***Estimates of Derivatives and Jumps Across Element Boundaries for Multi-Adaptive Galerkin Solutions of ODEs***

The interpolation estimates proved in PAPER VII are expressed in terms of the jumps in function values and derivatives at the points of discontinuity for the interpolated function. This paper estimates the size of these jumps for the discrete multi-adaptive solutions.

PAPER IX: ***Algorithms for Multi-Adaptive Time-Stepping***

This paper presents the key algorithms for multi-adaptive time-stepping, including the recursive construction of time slabs and adaptive fixed point iteration on time slabs.

PAPER X: ***Simulation of Mechanical Systems with Individual Time Steps***

This paper discusses the application of the multi-adaptive methods to mechanical systems, in particular to large mass-spring systems modeling deformable solids. The performance of the multi-adaptive methods is examined, both for stiff and non-stiff problems.

PAPER XI: ***Computational Modeling of Dynamical Systems***

This paper is a short note on the basic approach to automated computational modeling of dynamical systems.

References

[1] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, 1998.

[2] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica, 10 (2001).

[3] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 355–378.

[4] B.G. Galerkin, *Series solution of some problems in elastic equilibrium of rods and plates*, Vestnik inzhenerov i tekhnikov, 19 (1915), pp. 897–908.

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

[6] C.A.P. Castigliano, *Théorie de l'équilibre des systèmes élastiques et ses applications*, A.F. Negro ed., Torino, 1879.

[7] R. Courant, *Variational methods for the solution of problems of equilibrium and vibrations*, Bull. Amer. Math. Soc., 49 (1943), pp. 1–23.

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

[9] R. Davé, J. Dubinski, and L. Hernquist, *Parallel treeSPH*, New Astronomy, 2 (1997), pp. 277–297.

[10] C. Dawson and R.C. Kirby, *High resolution schemes for conservation laws with locally varying time steps*, SIAM J. Sci. Comput., 22, No. 6 (2001), pp. 2256–2281.

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

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

[13] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, Acta Numerica, (1995), pp. 105–158.

[14] ——, *Computational Differential Equations*, Cambridge University Press, 1996.

[15] K. Eriksson, D. Estep, and C. Johnson, *Applied Mathematics: Body and Soul*, vol. I, Springer-Verlag, 2003.

[16] ——, *Applied Mathematics: Body and Soul*, vol. II, Springer-Verlag, 2003.

[17] ——, *Applied Mathematics: Body and Soul*, vol. III, Springer-Verlag, 2003.

[18] K. Eriksson and C. Johnson, *Adaptive finite element methods for parabolic problems III: Time steps variable in space*, in preparation.

[19] ——, *Adaptive finite element methods for parabolic problems I: A linear model problem*, SIAM J. Numer. Anal., 28, No. 1 (1991), pp. 43–77.

[20] ——, *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), pp. 706–740.

[21] ——, *Adaptive finite element methods for parabolic problems IV: Nonlinear problems*, SIAM J. Numer. Anal., 32 (1995), pp. 1729–1749.

[22] ——, *Adaptive finite element methods for parabolic problems V: Long-time integration*, SIAM J. Numer. Anal., 32 (1995), pp. 1750–1763.

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

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

[25] ——, *Adaptive computational methods for parabolic problems*, to appear in Encyclopedia of Computational Mechanics, (2004).

[26] D. Estep, *An analysis of numerical approximations of metastable solutions of the bistable equation*, Nonlinearity, 7 (1994), pp. 1445–1462.

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

[28] D. Estep and D. French, *Global error control for the continuous Galerkin finite element method for ordinary differential equations*, M^2AN, 28 (1994), pp. 815–852.

[29] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations I — Nonstiff Problems*, Springer Series in Computational Mathematics, vol. 8, 1991.

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

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

[32] J. Hoffman, C. Johnson, and A. Logg, *Dreams of Calculus — Perspectives on Mathematics Education*, Springer-Verlag, 2004.

[33] J. Hoffman and A. Logg, *DOLFIN: Dynamic Object oriented Library for FINite element computation*, Tech. Rep. 2002–06, Chalmers Finite Element Center Preprint Series, 2002.

[34] J. Jansson, C. Johnson, and A. Logg, *Computational modeling of dynamical systems*, to appear in M^3AS, (2004).

[35] J. Jansson and A. Logg, *Algorithms for multi-adaptive time-stepping*, submitted to ACM Trans. Math. Softw., (2004).

[36] ——, *Multi-adaptive Galerkin methods for ODEs V: Stiff problems*, submitted to BIT, (2004).

[37] ——, *Simulation of mechanical systems with individual time steps*, submitted to SIAM J. Appl. Dyn. Syst., (2004).

[38] J.E. Flaherty, R.M. Loy, M.S. Shephard, B.K. Szymanski, J.D. Teresco, and L.H. Ziantz, *Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 139–152.

[39] 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 (1988), pp. 908–926.

[40] C. Johnson, J. Hoffman, and A. Logg, *Topics in adaptive computational methods for differential equations*, CEDYA 2001: Congreso de Ecuaciones Diferenciales y Aplicaciones, (2001).

[41] C. Johnson and A. Logg, *Dynamical Systems, Applied Mathematics: Body and Soul vol. IV*, Springer-Verlag, 2004.

[42] A. Lew, J.E. Marsden, M. Ortiz, and M. West, *Asynchronous variational integrators*, Arch. Rational. Mech. Anal., 167 (2003), pp. 85–146.

[43] A. Logg, *Tanganyika*, http://www.phi.chalmers.se/tanganyika/.

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

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

[46] ———, *Estimates of derivatives and jumps across element boundaries for multi-adaptive Galerkin solutions of ODEs*, Tech. Rep. 2004–03, Chalmers Finite Element Center Preprint Series, 2004.

[47] ———, *Interpolation estimates for piecewise smooth functions in one dimension*, Tech. Rep. 2004–02, Chalmers Finite Element Center Preprint Series, 2004.

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

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

[50] ———, *Multi-adaptive time-integration*, Applied Numerical Mathematics, 48 (2004), pp. 339–354.

[51] J. Makino and S. Aarseth, *On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems*, Publ. Astron. Soc. Japan, 44 (1992), pp. 141–151.

[52] I. Newton, *Philosophiae Naturalis Principia Mathematica*, vol. I–III, 1687.

[53] S. Osher and R. Sanders, *Numerical approximations to nonlinear conservation laws with locally varying time and space grids*, Math. Comp., 41 (1983), pp. 321–336.

[54] P.G. Ciarlet, *Numerical Analysis of the Finite Element Method*, Les Presses de l'Universite de Montreal, 1976.

[55] R.C. Kirby, *A linear algebraic approach to representing and computing finite elements*, submitted to Math. Comp., (2003).

[56] W. Ritz, *Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik*, J. reine angew. Math., 135 (1908), pp. 1–61.

[57] R. Sandboge, *Adaptive Finite Element Methods for Reactive Flow Problems*, PhD thesis, Department of mathematics, Chalmers University of Technology, Göteborg, 1996.

[58] S.C. Brenner and L.R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, 1994.

[59] S.G. Alexander and C.B. Agnor, *n-body simulations of late stage planetary formation with a simple fragmentation model*, ICARUS, 132 (1998), pp. 113–124.

[60] L. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, 1994.

[61] G. Strang and G.J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, 1973.

[62] T.J.R Hughes, I. Levit, and J. Winget, *Element-by-element implicit algorithms for heat-conduction*, J. Eng. Mech.-ASCE, 109 (1983), pp. 576–585.

[63] ———, *An element-by-element solution algorithm for problems of structural and solid mechanics*, Computer Methods in Applied Mechanics and Engineering, 36 (1983), pp. 241–254.

SIAM J. SCI. COMPUT.
Vol. 24, No. 6, pp. 1879–1902

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES I*

ANDERS LOGG†

Abstract. We present *multi-adaptive* versions of the standard continuous and discontinuous Galerkin methods for ODEs. Taking adaptivity one step further, we allow for individual time-steps, order and quadrature, so that in particular each individual component has its own time-step sequence. This paper contains a description of the methods, an analysis of their basic properties, and a posteriori error analysis. In the accompanying paper [A. Logg, *SIAM J. Sci. Comput.*, submitted], we present adaptive algorithms for time-stepping and global error control based on the results of the current paper.

Key words. multi-adaptivity, individual time-steps, local time-steps, ODE, continuous Galerkin, discontinuous Galerkin, global error control, adaptivity, mcG(q), mdG(q)

AMS subject classifications. 65L05, 65L07, 65L20, 65L50, 65L60, 65L70

PII. S1064827501389722

1. Introduction.

In this paper, we present multi-adaptive Galerkin methods for initial value problems for systems of ODEs of the form

$$\left\{ \begin{array}{rcll} \dot{u}(t) & = & f(u(t),t), & t \in (0,T], \\ u(0) & = & u_0, & \end{array} \right. \tag{1.1}$$

where $u : [0,T] \to \mathbb{R}^N$, $f : \mathbb{R}^N \times (0,T] \to \mathbb{R}^N$ is a given bounded function that is Lipschitz-continuous in u, $u_0 \in \mathbb{R}^N$ is a given initial condition, and $T > 0$ is a given final time. 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 feedback process. In the companion paper [29], we apply the multi-adaptive methods to a variety of problems to illustrate the potential of multi-adaptivity.

The ODE (1.1) models a very large class of problems, covering many areas of applications. Often different solution components have different time-scales and thus ask for individual time-steps. A prime example to be studied in detail below is our own solar system, where the moon orbits around Earth once every month, whereas the period of Pluto is 250 years. In numerical simulations of the solar system, the time-steps needed to track the orbit of the moon accurately are thus much less than those required for Pluto, the difference in time-scales being roughly a factor 3,000.

Surprisingly, individual time-stepping for ODEs has received little attention in the large literature on numerical methods for ODEs; see, e.g., [4, 21, 22, 3, 34]. For specific applications, such as the n-body problem, methods with individual time-stepping have been used—see, e.g., [31, 1, 5] or [25]—but a general methodology has been lacking. Our aim is to fill this gap. 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. [20] have constructed a method based on the discontinuous Galerkin method combined with local forward

*Received by the editors May 23, 2001; accepted for publication (in revised form) November 13, 2002; published electronically May 2, 2003.
http://www.siam.org/journals/sisc/24-6/38972.html
†Department of Computational Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden (logg@math.chalmers.se).

Euler time-stepping. A similar approach is taken in [6], where a method based on the original work by Osher and Sanders [33] 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 ≤ 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 also for time-dependent PDEs.

The methods presented in this paper fall within the general framework of adaptive Galerkin methods based on piecewise polynomial approximation (finite element methods) for differential equations, including the continuous Galerkin method cG(q) of order $2q$, and the discontinuous Galerkin method dG(q) of order $2q+1$; more precisely, we extend the cG(q) and dG(q) methods 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, 16, 24, 18, 17, 19]. The techniques for error analysis used in these references, developed by Johnson and coworkers (see, e.g., [11, 12, 10, 13, 14, 15], and [8] in particular) naturally carries over to the multi-adaptive methods.

The outline of the paper is as follows: In section 2 we summarize the key features of the multi-adaptive methods, and in section 3 we discuss the benefits of the new methods in comparison to standard ODE codes. We then motivate and present the formulation of the multi-adaptive methods mcG(q) and mdG(q) in section 4. Basic properties of these methods, such as order, energy conservation, and monotonicity, are discussed in section 5. In the major part of this paper, section 6, we derive a posteriori error estimates for the two methods based on duality arguments, including Galerkin errors, numerical errors, and quadrature errors. We also prove an a posteriori error estimate for stability factors computed from approximate dual solutions.

2. Key features.

We summarize the key features of our work on the mcG(q) and mdG(q) methods as follows.

2.1. Individual time-steps and order.

To discretize (1.1), we introduce for each component, $i = 1, \ldots, N$, a partition of the time-interval $(0,T]$ into M_i subintervals, $I_{ij} = (t_{i,j-1}, t_{ij}]$, $j = 1, \ldots, M_i$, and we seek an approximate solution $U(t) = (U_i(t))$ such that $U_i(t)$ is a polynomial of degree q_{ij} on every local interval I_{ij}. Each individual component $U_i(t)$ thus has its own sequence of time-steps, $\{k_{ij}\}_{j=1}^{M_i}$. The entire collection of individual time-intervals $\{I_{ij}\}$ may be organized into a sequence of *time-slabs*, collecting the time-intervals between certain synchronised time-levels common to all components, as illustrated in Figure 2.1.

2.2. Global error control.

Our goal is to compute an approximation $U(T)$ of the exact solution $u(T)$ 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 (see [32]).

The mathematical basis of global error control in $\|\cdot\|$ for mcG(q) is an error representation of the form

$$\|U(T) - u(T)\| = \int_0^T (R, \varphi)\, dt, \tag{2.1}$$

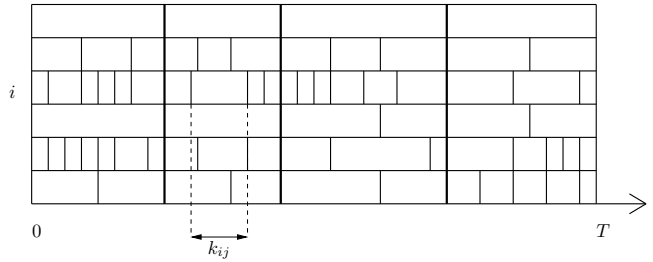

FIG. 2.1. *Individual time-discretizations for different components.*

where $R = (R_i) = R(U,t) = \dot{U}(t) - f(U(t),t)$ is the *residual* vector of the approximate solution $U(t)$, $\varphi(t)$ is the solution of an associated linearized dual problem, and $(\cdot,\cdot)$ is the $\mathbb{R}^N$ scalar product.

Using the Galerkin orthogonality, the error representation can be converted into an error bound of the form

$$\|U(T) - u(T)\| \leq \sum_{i=1}^{N} S_i(T) \max_{0\leq t\leq T} k_i(t)^{q_i(t)} |R_i(U,t)|, \tag{2.2}$$

where $\{S_i(T)\}_{i=1}^N$ are *stability factors* for the different components, depending on the dual solution $\varphi(t)$, and where $k_i(t) = k_{ij}$, $q_i(t) = q_{ij}$ for $t \in I_{ij}$. The error bound may take different forms depending on how $\int_0^T (R,\varphi)\, dt$ is bounded in terms of R and φ.

By solving the dual problem numerically, the individual stability factors $S_i(T)$ may be determined approximately, and thus the right-hand side of (2.2) may be evaluated. The adaptive algorithm seeks to satisfy the *stopping criterion*

$$\sum_{i=1}^{N} S_i(T) \max_{0\leq t\leq T} k_i(t)^{q_i(t)} |R_i(U,t)| \leq \mathrm{TOL}, \tag{2.3}$$

with maximal time-steps $k = (k_i(t))$.

2.3. Iterative methods. Both mcG(q) and mdG(q) give rise to systems of nonlinear algebraic equations, coupling the values of $U(t)$ over each time-slab. Solving these systems with full Newton may be quite heavy, and we have instead successfully used diagonal Newton methods of more explicit nature.

2.4. Implementation of higher-order methods. We have implemented mcG(q) and mdG(q) in C++ for arbitrary q, which in practice means $2q \leq 50$. The implementation, *Tanganyika*, is described in more detail in [29] and is publicly (GNU GPL) available for Linux/Unix [30].

2.5. Applications. We have applied mcG(q) and mdG(q) to a variety of problems to illustrate their potential; see [29]. (See also [27] and [26].) In these applications, including the Lorenz system, the solar system, and a number of time-dependent PDE problems, we demonstrate the use of individual time-steps, and for each system we solve the dual problem to collect extensive information about the problems stability features, which can be used for global error control.

3. Comparison with standard ODE codes. Standard ODE codes use time-steps which are variable in time but the same for all components, and the time-steps are adaptively chosen by keeping the "local error" below a given local error tolerance set by the user. The global error connects to the local error through an estimate, corresponding to (2.2), of the form

$$\{\text{global error}\} \leq S \ \max\{\text{local error}\}, \tag{3.1}$$

where S is a stability factor. Standard codes do not compute S, which means that the connection between the global error and the local error is left to be determined by the clever user, typically by computing with a couple of different tolerances.

Comparing the adaptive error control of standard ODE codes with the error control presented in this paper and the accompanying paper [29], an essential difference is thus the technique to estimate the global error: either by clever trial-and-error or, as we prefer, by solving the dual problem and computing the stability factors. Both approaches carry extra costs and what is best may be debated; see, e.g., [32] for a comparison.

However, expanding the scope to multi-adaptivity with individual stability factors for the different components, trial-and-error becomes very difficult or impossible, and the methods for adaptive time-stepping and error control presented below based on solving the dual problem seem to bring clear advantages in efficiency and reliability.

For a presentation of the traditional approach to error estimation in ODE codes, we refer to [2], where the following rather pessimistic view is presented: *Here we just note that a precise error bound is often unknown and not really needed.* We take the opposite view: *global error control is always needed and often possible to obtain at a reasonable cost.* We hope that multi-adaptivity will bring new life to the discussion on efficient and reliable error control for ODEs.

4. Multi-adaptive Galerkin. In this section we present the multi-adaptive Galerkin methods, mcG(q) and mdG(q), based on the discretization presented in section 2.1.

4.1. The mcG(q) method. The mcG(q) method for (1.1) reads as follows: Find $U \in V$ with $U(0) = u_0$, such that

$$\int_0^T (\dot{U}, v)\, dt = \int_0^T (f(U,\cdot), v)\, dt \qquad \forall v \in W, \tag{4.1}$$

where

$$\begin{array}{rcl} V & = & \{v \in [C([0,T])]^N : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}),\ j = 1,\ldots,M_i,\ i = 1,\ldots,N\}, \\ W & = & \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}-1}(I_{ij}),\ j = 1,\ldots,M_i,\ i = 1,\ldots,N\}, \end{array} \tag{4.2}$$

and where $\mathcal{P}^q(I)$ denotes the linear space of polynomials of degree $\leq q$ on I. The trial functions in V are thus continuous piecewise polynomials, locally of degree q_{ij}, and the test functions in W are discontinuous piecewise polynomials that are locally of degree $q_{ij} - 1$.

Noting that the test functions are discontinuous, we can rewrite the global problem (4.1) 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, such that

$$\int_{I_{ij}} \dot{U}_i v\, dt = \int_{I_{ij}} f_i(U,\cdot) v\, dt \qquad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}). \tag{4.3}$$

We notice the presence of the vector $U(t) = (U_1(t), \ldots, U_N(t))$ in the local problem for $U_i(t)$ on I_{ij}. If thus component $U_{i_1}(t)$ couples to component $U_{i_2}(t)$ through f, this means that in order to solve the local problem for component $U_{i_1}(t)$ we need to know the values of component $U_{i_2}(t)$ and vice versa. The solution is thus implicitly defined by (4.3). Notice also that if we define the *residual* R of the approximate solution U as $R_i(U,t) = \dot{U}_i(t) - f_i(U(t),t)$, we can rewrite (4.3) as

$$\int_{I_{ij}} R_i(U,\cdot)v \; dt = 0 \qquad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \tag{4.4}$$

i.e., the residual is orthogonal to the test space on every local interval. We refer to this as the *Galerkin orthogonality* for the mcG(q) method.

Making an ansatz for every component $U_i(t)$ on every local interval I_{ij} in terms of a nodal basis for $\mathcal{P}^{q_{ij}}(I_{ij})$ (see the appendix), we can rewrite (4.3) as

$$\xi_{ijm} = \xi_{ij0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) \; f_i(U(t),t) \; dt, \qquad m = 1, \ldots, q_{ij}, \tag{4.5}$$

where $\{\xi_{ijm}\}_{m=0}^{q_{ij}}$ are the nodal degrees of freedom for $U_i(t)$ on the interval I_{ij}, $\{w_m^{[q]}\}_{m=1}^{q} \subset \mathcal{P}^{q-1}(0,1)$ are corresponding polynomial weight functions, and τ_{ij} maps I_{ij} to $(0,1]$: $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. Here we assume that the solution is expressed in terms of a nodal basis with the end-points included, so that by the continuity requirement $\xi_{ij0} = \xi_{i,j-1,q_{i,j-1}}$.

Finally, evaluating the integral in (4.5) using nodal quadrature, we obtain a fully discrete scheme in the form of an implicit Runge–Kutta method: For $i = 1, \ldots, N$, $j = 1, \ldots, M_i$, find $\{\xi_{ijm}\}_{m=0}^{q_{ij}}$, with ξ_{ij0} given by the continuity requirement, such that

$$\xi_{ijm} = \xi_{ij0} + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} \; f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})), \tau_{ij}^{-1}(s_n^{[q_{ij}]})), \quad m = 1, \ldots, q_{ij}, \tag{4.6}$$

for certain weights $\{w_{mn}^{[q]}\}$ and certain nodal points $\{s_n^{[q]}\}$ (see the appendix).

4.2. The mdG(q) method. The mdG(q) method in local form, corresponding to (4.3), reads as follows: For $i = 1, \ldots, N$, $j = 1, \ldots, M_i$, find $U_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$, such that

$$[U_i]_{i,j-1} v(t_{i,j-1}^+) + \int_{I_{ij}} \dot{U}_i v \; dt = \int_{I_{ij}} f_i(U,\cdot)v \; dt \qquad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}), \tag{4.7}$$

where $[\cdot]$ denotes the jump, i.e., $[v]_{ij} = v(t_{ij}^+) - v(t_{ij}^-)$, and the initial condition is specified for $i = 1, \ldots, N$, by $U_i(0^-) = u_i(0)$. On a global level, the trial and test spaces are given by

$$V = W = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}), j = 1, \ldots, M_i, \; i = 1, \ldots, N\}. \tag{4.8}$$

In the same way as for the continuous method, we define the residual R of the approximate solution U as $R_i(U,t) = \dot{U}_i(t) - f_i(U(t),t)$, defined on the inner of every local interval I_{ij}, and we rewrite (4.7) in the form

$$[U_i]_{i,j-1} v(t_{i,j-1}^+) + \int_{I_{ij}} R_i(U,\cdot)v \; dt = 0 \qquad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}). \tag{4.9}$$

We refer to this as the Galerkin orthogonality for the mdG(q) method. Notice that this is similar to (4.4) if we extend the integral in (4.4) to include the left end-point of the interval I_{ij}. (The derivative of the discontinuous solution is a Dirac delta function at the end-point.)

Making an ansatz for the solution in terms of some nodal basis, we get, as for the continuous method, the following explicit version of (4.7) on every local interval:

$$\xi_{ijm} = \xi_{ij0}^- + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) \; f_i(U(t),t) \; dt, \qquad m = 0, \ldots, q_{ij}, \tag{4.10}$$

or, applying nodal quadrature,

$$\xi_{ijm} = \xi_{ij0}^- + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} \; f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})), \tau_{ij}^{-1}(s_n^{[q_{ij}]})), \qquad m = 0, \ldots, q_{ij}, \tag{4.11}$$

where the weight functions, the nodal points, and the weights are not the same as for the continuous method.

4.3. The multi-adaptive mcG(q)-mdG(q) method. The discussion above for the two methods extends naturally to using different methods for different components. Some of the components could therefore be solved for using the mcG(q) method, while for others we use the mdG(q) method. We can even change methods between different intervals.

Although the formulation thus includes adaptive orders and methods, as well as adaptive time-steps, our focus will be mainly on adaptive time-steps.

4.4. Choosing basis functions and quadrature. What remains in order to implement the two methods specified by (4.6) and (4.11) is to choose basis functions and quadrature. For simplicity and efficiency reasons, it is desirable to let the nodal points for the nodal basis coincide with the quadrature points. It turns out that for both methods, the mcG(q) and the mdG(q) methods, this is possible to achieve in a natural way. We thus choose the $q+1$ *Lobatto quadrature points* for the mcG(q) method, i.e., the zeros of $xP_q(x) - P_{q-1}(x)$, where P_q is the qth-order Legendre polynomial on the interval; for the mdG(q) method, we choose the *Radau quadrature points*, i.e., the zeros of $P_q(x) + P_{q+1}(x)$ on the interval (with time reversed so as to include the *right* end-point). See [28] for a detailed discussion on this subject. The resulting discrete schemes are related to the implicit Runge–Kutta methods referred to as Lobatto and Radau methods; see, e.g., [3].

5. Basic properties of the multi-adaptive Galerkin methods. In this section we examine some basic properties of the multi-adaptive methods, including order, energy conservation, and monotonicity.

5.1. Order. The standard cG(q) and dG(q) methods are of order $2q$ and $2q+1$, respectively. The corresponding properties hold for the multi-adaptive methods, i.e., mcG(q) is of order $2q$ and mdG(q) is of order $2q+1$, assuming that the exact solution u is smooth. We examine this in more detail in subsequent papers.

5.2. Energy conservation for mcG(q). The standard cG(q) method is energy-conserving for Hamiltonian systems. We now prove that also the mcG(q) method has this property, with the natural restriction that we should use the same time-steps for every pair of positions and velocities. We consider a Hamiltonian system,

$$\ddot{x} = -\nabla_x P(x), \tag{5.1}$$

on $(0,T]$ with $x(t) \in \mathbb{R}^N$, together with initial conditions for x and $\dot{x}$. Here $\ddot{x}$ is the acceleration, which by Newton's second law is balanced by the force $F(x) = -\nabla_x P(x)$ for some potential field P. With $u = x$ and $v = \dot{x}$ we rewrite (5.1) as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ F(u) \end{bmatrix} = \begin{bmatrix} f_u(v) \\ f_v(u) \end{bmatrix} = f(u,v). \tag{5.2}$$

The total energy $E(t)$ is the sum of the kinetic energy $K(t)$ and the potential energy $P(x(t))$,

$$E(t) = K(t) + P(x(t)), \tag{5.3}$$

with

$$K(t) = \frac{1}{2}\|\dot{x}(t)\|^2 = \frac{1}{2}\|v(t)\|^2. \tag{5.4}$$

Multiplying (5.1) with $\dot{x}$ it is easy to see that energy is conserved for the continuous problem, i.e., $E(t) = E(0)$ for all $t \in [0,T]$. We now prove the corresponding property for the discrete mcG(q) solution of (5.2).

THEOREM 5.1. *The multi-adaptive continuous Galerkin method conserves energy in the following sense: Let (U,V) be the* mcG(q) *solution to* (5.2) *defined by* (4.3). *Assume that the same time-steps are used for every pair of positions and corresponding velocities. Then at every synchronized time-level $\bar{t}$, such as, e.g., T, we have*

$$K(\bar{t}) + P(\bar{t}) = K(0) + P(0), \tag{5.5}$$

with $K(t) = \frac{1}{2}\|V(t)\|^2$ and $P(t) = P(U(t))$.

Proof. If every pair of positions and velocities have the same time-step sequence, then we may choose $\dot{V}$ as a test function in the equations for U, to get

$$\int_0^{\bar{t}} (\dot{U}, \dot{V})\, dt = \int_0^{\bar{t}} (V, \dot{V})\, dt = \frac{1}{2}\int_0^{\bar{t}} \frac{d}{dt}\|V\|^2\, dt = K(\bar{t}) - K(0).$$

Similarly, $\dot{U}$ may be chosen as a test function in the equations for V to get

$$\int_0^{\bar{t}} (\dot{V}, \dot{U})\, dt = \int_0^{\bar{t}} -\nabla P(U)\dot{U}\, dt = -\int_0^{\bar{t}} \frac{d}{dt} P(U)\, dt = -(P(\bar{t}) - P(0)),$$

and thus $K(\bar{t}) + P(\bar{t}) = K(0) + P(0)$. □

Remark 5.1. Energy conservation requires exact integration of the right-hand side f (or at least that $\int_0^{\bar{t}} (\dot{U}, \dot{V})\, dt + (P(t) - P(0)) = 0$) but can also be obtained in the case of quadrature; see [23].

5.3. Monotonicity. We shall prove that the mdG(q) method is B-stable (see [3]).

THEOREM 5.2. *Let U and V be the* mdG(q) *solutions of* (1.1) *with initial data $U(0^-)$ and $V(0^-)$, respectively, defined by* (4.7) *on the same partition. If the right-hand side f is monotone, i.e.,*

$$(f(u,\cdot) - f(v,\cdot), u - v) \leq 0 \qquad \forall u, v \in \mathbb{R}^N, \tag{5.6}$$

then, at every synchronized time-level $\bar{t}$, such as, e.g., T, we have

$$\|U(\bar{t}^-) - V(\bar{t}^-)\| \leq \|U(0^-) - V(0^-)\|. \tag{5.7}$$

Proof. Choosing the test function as $v = W = U - V$ in (4.7) for U and V, summing over the local intervals, and subtracting the two equations, we have

$$\sum_{ij} \left[[W_i]_{i,j-1} W^+_{i,j-1} + \int_{I_{ij}} \dot{W}_i W_i\, dt \right] = \int_0^T (f(U,\cdot) - f(V,\cdot), U - V)\, dt \leq 0.$$

Noting that

$$\begin{aligned} [W_i]_{i,j-1} W^+_{i,j-1} + \textstyle\int_{I_{ij}} \dot{W}_i W_i\, dt &= \tfrac{1}{2}(W^+_{i,j-1})^2 + \tfrac{1}{2}(W^-_{ij})^2 - W^-_{i,j-1} W^+_{i,j-1} \\ &= \tfrac{1}{2}[W_i]^2_{i,j-1} + \tfrac{1}{2}\left((W^-_{ij})^2 - (W^-_{i,j-1})^2\right), \end{aligned}$$

we get

$$-\frac{1}{2}\|W(0^-)\|^2 + \frac{1}{2}\|W(T^-)\|^2 \leq \sum_{ij} [W_i]_{i,j-1} W^+_{i,j-1} + \int_{I_{ij}} \dot{W}_i W_i\, dt \leq 0,$$

so that

$$\|W(T^-)\| \leq \|W(0^-)\|.$$

The proof is completed noting that the same analysis applies with T replaced by any other synchronized time-level $\bar{t}$. □

Remark 5.2. The proof extends to the fully discrete scheme, using the positivity of the quadrature weights.

6. A posteriori error analysis. In this section we prove a posteriori error estimates for the multi-adaptive Galerkin methods, including quadrature and discrete solution errors. Following the procedure outlined in the introduction, we first define the dual linearized problem and then derive a representation formula for the error in terms of the dual and the residual.

6.1. The dual problem. The dual problem comes in two different forms: a continuous and a discrete. For the a posteriori error analysis of this section, we will make use of the continuous dual. The discrete dual problem is used to prove a priori error estimates.

To set up the continuous dual problem, we define, for given functions $v_1(t)$ and $v_2(t)$,

$$J^*(v_1(t), v_2(t), t) = \left(\int_0^1 \frac{\partial f}{\partial u}(sv_1(t) + (1-s)v_2(t), t)\, ds \right)^*, \tag{6.1}$$

where * denotes the transpose, and we note that

$$\begin{aligned} J(v_1, v_2, \cdot)(v_1 - v_2) &= \textstyle\int_0^1 \frac{\partial f}{\partial u}(sv_1 + (1-s)v_2, \cdot)\, ds\ (v_1 - v_2) \\ &= \textstyle\int_0^1 \frac{\partial f}{\partial s}(sv_1 + (1-s)v_2, \cdot)\, ds = f(v_1,\cdot) - f(v_2,\cdot). \end{aligned} \tag{6.2}$$

The continuous dual problem is then defined as the following system of ODEs:

$$\begin{cases} -\dot{\varphi} &= J^*(u, U, \cdot)\varphi + g \quad \text{on } [0,T), \\ \varphi(T) &= \varphi_T, \end{cases} \tag{6.3}$$

with data φ_T and right-hand side g. Choosing the data and right-hand side appropriately, we obtain error estimates for different quantities of the computed solution. We shall assume below that the dual solution has q continuous derivatives ($\varphi_i^{(q_{ij})} \in C(I_{ij})$ locally on interval I_{ij}) for the continuous method and $q+1$ continuous derivatives ($\varphi_i^{(q_{ij}+1)} \in C(I_{ij})$ locally on interval I_{ij}) for the discontinuous method.

6.2. Error representation. The basis for the error analysis is the following error representation, expressing the error of an approximate solution $U(t)$ in terms of the residual $R(U,t)$ via the dual solution $\varphi(t)$. We stress that the result of the theorem is valid for any piecewise polynomial approximation of the solution to the initial value problem (1.1) and thus in particular the mcG(q) and mdG(q) approximations.

THEOREM 6.1. *Let U be a piecewise polynomial approximation of the exact solution u of* (1.1)*, and let φ be the solution to* (6.3) *with right-hand side $g(t)$ and initial data φ_T, and define the residual of the approximate solution U as $R(U,t) = \dot{U}(t) - f(U(t),t)$, defined on the open intervals of the partitions $\cup_j I_{ij}$ as*

$$R_i(U,t) = \dot{U}_i(t) - f_i(U(t),t), \qquad t \in (k_{i,j-1}, k_{ij}),$$

$j = 1,\dots,M_i$, $i = 1,\dots,N$. Assume also that U is right-continuous at T. Then the error $e = U - u$ satisfies

(6.4)
$$L_{\varphi_T,g}(e) \equiv (e(T),\varphi_T) + \int_0^T (e,g)\ dt = \sum_{i=1}^N \sum_{j=1}^{M_i} \left[\int_{I_{ij}} R_i(U,\cdot)\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right].$$

Proof. By the definition of the dual problem, we have using (6.2)

$$\begin{aligned} \textstyle\int_0^T (e,g)\ dt &= \textstyle\int_0^T (e, -\dot\varphi - J^*(u,U,\cdot)\varphi)\ dt \\ &= \textstyle\sum_{ij} \int_{I_{ij}} -e_i\dot\varphi_i\ dt + \int_0^T (-J(u,U,\cdot)e,\varphi)\ dt \\ &= \textstyle\sum_{ij} \int_{I_{ij}} -e_i\dot\varphi_i\ dt + \int_0^T (f(u,\cdot) - f(U,\cdot),\varphi)\ dt \\ &= \textstyle\sum_{ij} \int_{I_{ij}} -e_i\dot\varphi_i\ dt + \sum_{ij} \int_{I_{ij}} (f_i(u,\cdot) - f_i(U,\cdot))\varphi_i\ dt. \end{aligned}$$

Integrating by parts, we get

$$\int_{I_{ij}} -e_i\dot\varphi_i\ dt = e_i(t_{i,j-i}^+)\varphi(t_{i,j-1}) - e_i(t_{ij}^-)\varphi(t_{ij}) + \int_{I_{ij}} \dot e_i\varphi_i\ dt,$$

so that

$$\begin{aligned} \textstyle\sum_{ij} \int_{I_{ij}} -e_i\dot\varphi_i\ dt &= \textstyle\sum_{ij} [e_i]_{i,j-1}\varphi_i(t_{i,j-1}) - (e(T^-),\varphi_T) + \int_0^T (\dot e,\varphi)\ dt \\ &= \textstyle\sum_{ij} [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) - (e(T),\varphi_T) + \int_0^T (\dot e,\varphi)\ dt. \end{aligned}$$

Thus, with $L_{\varphi_T,g}(e) = (e(T),\varphi_T) + \int_0^T (e,g)\ dt$, we have

$$\begin{aligned} L_{\varphi_T,g}(e) &= \textstyle\sum_{ij} \left[\int_{I_{ij}} (\dot e_i + f_i(u,\cdot) - f_i(U,\cdot))\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right] \\ &= \textstyle\sum_{ij} \left[\int_{I_{ij}} (\dot U_i - f_i(U,\cdot))\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right] \\ &= \textstyle\sum_{ij} \left[\int_{I_{ij}} R_i(U,\cdot)\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right], \end{aligned}$$

which completes the proof. □

We now apply this theorem to represent the error in various norms. As before, we let $\|\cdot\|$ denote the Euclidean norm on $\mathbb{R}^N$ and define $\|v\|_{L^1([0,T],\mathbb{R}^n)} = \int_0^T \|v\|\ dt$.

COROLLARY 6.2. *If $\varphi_T = e(T)/\|e(T)\|$ and $g = 0$, then*

$$\|e(T)\| = \sum_{i=1}^N \sum_{j=1}^{M_i} \left[\int_{I_{ij}} R_i(U,\cdot)\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right]. \tag{6.5}$$

COROLLARY 6.3. *If $\varphi_T = 0$ and $g(t) = e(t)/\|e(t)\|$, then*

$$\|e\|_{L^1([0,T],\mathbb{R}^N)} = \sum_{i=1}^N \sum_{j=1}^{M_i} \left[\int_{I_{ij}} R_i(U,\cdot)\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1}) \right]. \tag{6.6}$$

6.3. Galerkin errors. To obtain expressions for the Galerkin errors, i.e., the errors of the mcG(q) or mdG(q) approximations, assuming exact quadrature and exact solution of the discrete equations, we use two ingredients: the error representation of Theorem 6.1 and the Galerkin orthogonalities, (4.4) and (4.9). We first prove the following interpolation estimate.

LEMMA 6.4. *If $f \in C^{q+1}([a,b])$, then there is a constant C_q, depending only on q, such that*

$$|f(x) - \pi^{[q]}f(x)| \le C_q k^{q+1} \frac{1}{k} \int_a^b |f^{(q+1)}(y)|\ dy \qquad \forall x \in [a,b], \tag{6.7}$$

where $\pi^{[q]}f(x)$ is the qth-order Taylor expansion of f around $x_0 = (a+b)/2$, $k = b-a$, and $C_q = 1/(2^q q!)$.

Proof. Using Taylor's formula with the remainder in integral form, we have

$$\begin{aligned} |f(x) - \pi^{[q]}f(x)| &= \left| \frac{1}{q!} \int_{x_0}^x f^{(q+1)}(y)(y - x_0)^{(q)}\ dy \right| \\ &\le \frac{1}{2^q q!} k^{q+1} \frac{1}{k} \int_a^b |f^{(q+1)}(y)|\ dy. \end{aligned}$$ □

Note that since we allow the polynomial degree to change between different components and between different intervals, the interpolation constant will change in the same way. We thus have $C_{q_i} = C_{q_i}(t) = C_{q_{ij}}$ for $t \in I_{ij}$.

We can now prove a posteriori error estimates for the mcG(q) and mdG(q) methods. The estimates come in a number of different versions. We typically use E_2 or E_3 to adaptively determine the time-steps and E_0 or E_1 to evaluate the error. The quantities E_4 and E_5 may be used for qualitative estimates of error growth. We emphasize that all of the estimates derived in Theorems 6.5 and 6.6 below may be of use in an actual implementation, ranging from the very sharp estimate E_0 containing only local quantities to the more robust estimate E_5 containing only global quantities.

THEOREM 6.5. *The* mcG(q) *method satisfies the following estimates:*

$$|L_{\varphi_T,g}(e)| = E_0 \le E_1 \le E_2 \le E_3 \le E_4 \tag{6.8}$$

and

$$|L_{\varphi_T,g}(e)| \le E_2 \le E_5, \tag{6.9}$$

where

$$\begin{aligned} E_0 &= \textstyle\left| \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\ dt \right|, \\ E_1 &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} |R_i(U,\cdot)||\varphi_i - \pi_k\varphi_i|\ dt, \\ E_2 &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} C_{q_{ij}-1} k_{ij}^{q_{ij}+1}\ r_{ij} s_{ij}^{[q_{ij}]}, \\ E_3 &= \textstyle\sum_{i=1}^N S_i^{[q_i]} \max_{[0,T]} \{C_{q_i-1} k_i^{q_i} r_i\}, \\ E_4 &= \textstyle S^{[q],1}\sqrt{N} \max_{i,[0,T]} \{C_{q_i-1} k_i^{q_i} r_i\}, \\ E_5 &= \textstyle S^{[q],2} \|C_{q-1} k^q R(U,\cdot)\|_{L^2(\mathbb{R}^N \times [0,T])}, \end{aligned} \tag{6.10}$$

with C_q as in Lemma 6.4, *$k_i(t) = k_{ij}$, $r_i(t) = r_{ij}$, and $s_i^{[q_i]}(t) = s_{ij}^{[q_{ij}]}$ for $t \in I_{ij}$,*

$$\begin{aligned} r_{ij} &= \tfrac{1}{k_{ij}} \textstyle\int_{I_{ij}} |R_i(U,\cdot)|\, dt, \quad & s_{ij}^{[q_{ij}]} &= \tfrac{1}{k_{ij}} \textstyle\int_{I_{ij}} |\varphi^{(q_{ij})}|\, dt, \\ S_i^{[q_i]} &= \textstyle\int_0^T |\varphi_i^{(q_i)}|\, dt, & S^{[q],1} &= \textstyle\int_0^T \|\varphi^{(q)}\|\, dt, \\ S^{[q],2} &= \left(\textstyle\int_0^T \|\varphi^{(q)}\|^2\, dt\right)^{1/2}, \end{aligned} \tag{6.11}$$

and where $\pi_k\varphi$ is any test space approximation of the dual solution φ. Expressions such as $C_{q-1}k^qR$ are defined componentwise, i.e., $(C_{q-1}k^qR(U,\cdot))_i = C_{q_{ij}-1}k_{ij}^{q_{ij}}R_i(U,\cdot)$ for $t \in I_{ij}$.

Proof. Using the error representation of Theorem 6.1 and the Galerkin orthogonality (4.4), noting that the jump terms disappear since U is continuous, we have

$$|L_{\varphi_T,g}(e)| = \left|\sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\, dt\right| = E_0,$$

where $\pi_k\varphi$ is any test space approximation of φ. By the triangle inequality, we have

$$E_0 \le \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} |R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)|\, dt = E_1.$$

Choosing $\pi_k\varphi_i$ as in Lemma 6.4 on every interval I_{ij}, we have

$$\begin{aligned} E_1 &\le \sum_{ij} C_{q_{ij}-1} k_{ij}^{q_{ij}} \int_{I_{ij}} |R_i(U,\cdot)|\, dt\, \frac{1}{k_{ij}} \int_{I_{ij}} |\varphi_i^{(q_{ij})}|\, dt \\ &= \sum_{ij} C_{q_{ij}-1} k_{ij}^{q_{ij}+1}\, r_{ij} s_{ij}^{[q_{ij}]} = E_2. \end{aligned}$$

Continuing, we have

$$\begin{aligned} E_2 &\le \textstyle\sum_{i=1}^N \max_{[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \sum_{j=1}^{M_i} k_{ij} s_{ij}^{[q_{ij}]} \\ &= \textstyle\sum_{i=1}^N \max_{[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \sum_{j=1}^{M_i} \int_{I_{ij}} |\varphi_i^{(q_{ij})}|\, dt \\ &= \textstyle\sum_{i=1}^N \max_{[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \int_0^T |\varphi_i^{(q_i)}|\, dt \\ &= \textstyle\sum_{i=1}^N S_i^{[q_i]} \max_{[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} = E_3, \end{aligned}$$

and, finally,

$$\begin{aligned} E_3 &\le \max_{i,[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \textstyle\sum_{i=1}^N \int_0^T |\varphi_i^{(q_i)}|\, dt \\ &\le \max_{i,[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \sqrt{N} \textstyle\int_0^T \|\varphi^{(q)}\|\, dt \\ &= \max_{i,[0,T]} \{C_{q_i-1}k_i^{q_i} r_i\} \sqrt{N} S^{[q],1} = E_4. \end{aligned}$$

As an alternative we can use Cauchy's inequality in a different way. Continuing from E_2, we have

$$\begin{aligned} E_2 &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} C_{q_{ij}-1} k_{ij}^{q_{ij}+1}\, r_{ij} s_{ij}^{[q_{ij}]} \\ &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} C_{q_{ij}-1} k_{ij}^{q_{ij}} s_{ij}^{[q_{ij}]} \int_{I_{ij}} |R_i(U,\cdot)|\, dt \\ &= \textstyle\sum_{i=1}^N \int_0^T C_{q_i-1} k_i^{q_i} |R_i(U,\cdot)| s_i^{[q_i]}\, dt \\ &= \textstyle\int_0^T (C_{q-1}k^q |R(U,\cdot)|, s^{[q]})\, dt \\ &\le \textstyle\int_0^T \|C_{q-1}k^qR(U,\cdot)\| \|s^{[q]}\|\, dt \\ &\le \left(\textstyle\int_0^T \|C_{q-1}k^qR(U,\cdot)\|^2\, dt\right)^{1/2} \left(\textstyle\int_0^T \|s^{[q]}\|^2\, dt\right)^{1/2}, \end{aligned}$$

where $|R(U,\cdot)|$ denotes the vector-valued function with components $|R|_i = |R_i| = |R_i(U,\cdot)|$. Noting now that s is the L^2-projection of $|\varphi^{(q)}|$ onto the piecewise constants on the partition, we have

$$\left(\int_0^T \|s^{[q]}\|^2\, dt\right)^{1/2} \le \left(\int_0^T \|\varphi^{(q)}\|^2\, dt\right)^{1/2},$$

so that

$$|L_{\varphi_T,g}(e)| \le \|C_{q-1}k^qR(U,\cdot)\|_{L^2(\mathbb{R}^N\times[0,T])} \|\varphi^{[q]}\|_{L^2(\mathbb{R}^N\times[0,T])} = E_5,$$

completing the proof. □

The proof of the estimates for the mdG(q) method is obtained similarly. Since in the discontinuous method the test functions are on every interval of one degree higher order than the test functions in the continuous method, we can choose a better interpolant. Thus, in view of Lemma 6.4, we obtain an extra factor k_{ij} in the error estimates.

THEOREM 6.6. *The* mdG(q) *method satisfies the following estimates:*

$$|L_{\varphi_T,g}(e)| = E_0 \le E_1 \le E_2 \le E_3 \le E_4 \tag{6.12}$$

and

$$|L_{\varphi_T,g}(e)| \le E_2 \le E_5, \tag{6.13}$$

where

$$\begin{aligned} E_0 &= \left|\textstyle\sum_{ij} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\, dt + [U_i]_{i,j-1}(\varphi_i(t_{i,j-1}) - \pi_k\varphi_i(t_{i,j-1}^+))\right|, \\ E_1 &= \textstyle\sum_{ij} \int_{I_{ij}} |R_i(U,\cdot)||\varphi_i - \pi_k\varphi_i|\, dt + |[U_i]_{i,j-1}||\varphi_i(t_{i,j-1}) - \pi_k\varphi_i(t_{i,j-1}^+)|, \\ E_2 &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} C_{q_{ij}} k_{ij}^{q_{ij}+2}\, \bar{r}_{ij} s_{ij}^{[q_{ij}+1]}, \\ E_3 &= \textstyle\sum_{i=1}^N S_i^{[q_i+1]} \max_{[0,T]} \left\{C_{q_i} k_i^{q_i+1} \bar{r}_i\right\}, \\ E_4 &= S^{[q+1],1} \sqrt{N} \max_{i,[0,T]} \left\{C_{q_i} k_i^{q_i+1} \bar{r}_i\right\}, \\ E_5 &= S^{[q+1],2} \|C_q k^{q+1} \bar{R}(U,\cdot)\|_{L^2(\mathbb{R}^N\times[0,T])}, \end{aligned} \tag{6.14}$$

with

$$\bar{r}_{ij} = \frac{1}{k_{ij}} \int_{I_{ij}} |R_i(U,\cdot)|\, dt + \frac{1}{k_{ij}} |[U_i]_{i,j-1}|, \qquad \bar{R}_i(U,\cdot) = |R_i(U,\cdot)| + \frac{1}{k_{ij}} |[U_i]_{i,j-1}|, \tag{6.15}$$

and we otherwise use the notation of Theorem 6.5.

Proof. As in the proof for the continuous method, we use the error representation of Theorem 6.1 and the Galerkin orthogonality (4.9) to get

$$|L_{\varphi_T,g}(e)| = \left|\sum_{ij} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\, dt + [U_i]_{i,j-1}(\varphi_i(t_{i,j-1}) - \pi_k\varphi_i(t_{i,j-1}^+))\right| = E_0.$$

By Lemma 6.4 we obtain

$$\begin{aligned} E_0 &\leq \textstyle\sum_{ij} \int_{I_{ij}} |R_i(U,\cdot)||\varphi_i - \pi_k\varphi_i|\ dt + |[U_i]_{i,j-1}||\varphi_i(t_{i,j-1}) - \pi_k\varphi_i(t^+_{i,j-1})| = E_1 \\ &\leq \textstyle\sum_{ij} C_{q_{ij}} k_{ij}^{q_{ij}+1} \left(\int_{I_{ij}} |R_i(U,\cdot)|\ dt + |[U_i]_{i,j-1}| \right) \frac{1}{k_{ij}} \int_{I_{ij}} |\varphi_i^{(q_{ij}+1)}|\ dt \\ &\leq \textstyle\sum_{ij} C_{q_{ij}} k_{ij}^{q_{ij}+2} \bar{r}_{ij} \bar{s}_{ij}^{[q_{ij}+1]} = E_2. \end{aligned}$$

Continuing now in the same way as for the continuous method, we have $E_2 \leq E_3 \leq E_4$ and $E_2 \leq E_5$. □

Remark 6.1. When evaluating the expressions E_0 or E_1, the interpolant $\pi_k\varphi$ does not have to be chosen as in Lemma 6.4. This is only a convenient way to obtain the interpolation constant. In section 6.6 below we discuss a more convenient choice of interpolant.

Remark 6.2. If we replace $\frac{1}{k_{ij}} \int_{I_{ij}} |R_i|\ dt$ by $\max_{I_{ij}} |R_i|$, we may replace C_q by a smaller constant C'_q. The value of the constant thus depends on the specific way the residual is measured.

6.4. Computational errors. The error estimates of Theorems 6.5 and 6.6 are based on the Galerkin orthogonalities (4.4) and (4.9). If the corresponding discrete equations are not solved exactly, there will be an additional contribution to the total error. Although Theorem 6.1 is still valid, the first steps in Theorems 6.5 and 6.6 are not. Focusing on the continuous method, the first step in the proof of Theorem 6.5 is the subtraction of a test space interpolant. This is possible, since by the Galerkin orthogonality we have

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)\pi_k\varphi_i\ dt = 0$$

for all test space interpolants $\pi_k\varphi$. If the residual is no longer orthogonal to the test space, we add and subtract this term to get to the point where the implications of Theorem 6.5 are valid for one of the terms. Assuming now that φ varies slowly on each subinterval, we estimate the remaining extra term as follows:

(6.16)
$$\begin{aligned} E_C &= \textstyle\left| \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)\pi_k\varphi_i\ dt \right| \leq \sum_{i=1}^N \sum_{j=1}^{M_i} \left| \int_{I_{ij}} R_i(U,\cdot)\pi_k\varphi_i\ dt \right| \\ &\approx \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} k_{ij}|\bar{\varphi}_{ij}| \frac{1}{k_{ij}} \left| \int_{I_{ij}} R_i(U,\cdot)\ dt \right| = \sum_{i=1}^N \sum_{j=1}^{M_i} k_{ij}|\bar{\varphi}_{ij}||\mathcal{R}^{\mathcal{C}}_{ij}| \\ &\leq \textstyle\sum_{i=1}^N \bar{S}_i^{[0]} \max_j |\mathcal{R}^{\mathcal{C}}_{ij}|, \end{aligned}$$

where $\bar{\varphi}$ is a piecewise constant approximation of φ (using, say, the mean values on the local intervals),

$$\bar{S}_i^{[0]} = \sum_{j=1}^{M_i} k_{ij}|\bar{\varphi}_{ij}| \approx \int_0^T |\varphi_i|\ dt = S_i^{[0]} \tag{6.17}$$

is a stability factor, and we define the *discrete* or *computational* residual as

$$\mathcal{R}^{\mathcal{C}}_{ij} = \frac{1}{k_{ij}} \int_{I_{ij}} R_i(U,\cdot)\ dt = \frac{1}{k_{ij}} \left((\xi_{ijq} - \xi_{ij0}) - \int_{I_{ij}} f_i(U,\cdot)\ dt \right). \tag{6.18}$$

More precise estimates may be used if needed.

For the mdG(q) method, the situation is similar with the computational residual now defined as

$$\mathcal{R}^{\mathcal{C}}_{ij} = \frac{1}{k_{ij}} \left((\xi_{ijq} - \bar{\xi}_{ij0}) - \int_{I_{ij}} f_i(U,\cdot)\ dt \right). \tag{6.19}$$

Thus, to estimate the computational error, we evaluate the computational residuals and multiply with the computed stability factors.

6.5. Quadrature errors. We now extend our analysis to take into account also quadrature errors. We denote integrals evaluated by quadrature with $\tilde{\int}$. Starting from the error representation as before, we have for the mcG(q) method

(6.20)
$$\begin{aligned} L_{\varphi_T,g}(e) &= \textstyle\int_0^T (R,\varphi)\ dt \\ &= \textstyle\int_0^T (R,\varphi - \pi_k\varphi)\ dt + \int_0^T (R,\pi_k\varphi)\ dt \\ &= \textstyle\int_0^T (R,\varphi - \pi_k\varphi)\ dt + \tilde{\int}_0^T (R,\pi_k\varphi)\ dt + \left[\int_0^T (R,\pi_k\varphi)\ dt - \tilde{\int}_0^T (R,\pi_k\varphi)\ dt \right] \\ &= \textstyle\int_0^T (R,\varphi - \pi_k\varphi)\ dt + \tilde{\int}_0^T (R,\pi_k\varphi)\ dt + \left(\tilde{\int}_0^T - \int_0^T \right) (f(U,\cdot),\pi_k\varphi)\ dt \end{aligned}$$

if the quadrature is exact for $\dot{U}v$ when v is a test function. The first term of this expression was estimated in Theorem 6.5 and the second term is the computational error discussed previously (where $\tilde{\int}$ denotes that in a real implementation, (6.18) is evaluated using quadrature). The third term is the quadrature error, which may be nonzero even if f is linear, if the time-steps are different for different components. To estimate the quadrature error, notice that

$$\begin{aligned} \left(\tilde{\int}_0^T - \int_0^T \right) (f(U,\cdot),\pi_k\varphi)\ dt &= \textstyle\sum_{ij} \left(\tilde{\int}_{I_{ij}} - \int_{I_{ij}} \right) f_i(U,\cdot)\pi_k\varphi_i\ dt \\ &\approx \textstyle\sum_{ij} k_{ij}\bar{\varphi}_{ij}\mathcal{R}^{\mathcal{Q}}_{ij} \leq \sum_{i=1}^N \bar{S}_i^{[0]} \max_j |\mathcal{R}^{\mathcal{Q}}_{ij}|, \end{aligned} \tag{6.21}$$

where $\{\bar{S}_i^{[0]}\}_{i=1}^N$ are the same stability factors as in the estimate for the computational error and

$$\mathcal{R}^{\mathcal{Q}}_{ij} = \frac{1}{k_{ij}} \left(\tilde{\int}_{I_{ij}} f_i(U,\cdot)\ dt - \int_{I_{ij}} f_i(U,\cdot)\ dt \right) \tag{6.22}$$

is the *quadrature residual*. A similar estimate holds for the mdG(q) method.

We now make a few comments on how to estimate the quadrature residual. The Lobatto quadrature of the mcG(q) method is exact for polynomials of degree less than or equal to $2q-1$, and we have an order $2q$ estimate for $\tilde{\int} - \int$ in terms of $f^{(2q)}$, and so we make the assumption $\mathcal{R}^{\mathcal{Q}}_{ij} \propto k_{ij}^{2q_{ij}}$. If, instead of using the standard quadrature rule over the interval with quadrature residual $\mathcal{R}^{\mathcal{Q}_0}_{ij}$, we divide the interval into 2^m parts and use the quadrature on every interval, summing up the result, we will get a different quadrature residual, namely

$$\mathcal{R}^{\mathcal{Q}_m} = \frac{1}{k} C 2^m (k/2^m)^{2q+1} = 2^{m(-2q)} C k^{2q} = 2^{-2q} \mathcal{R}^{\mathcal{Q}_{m-1}}, \tag{6.23}$$

where we have dropped the $_{ij}$ subindices. Thus, since $|\mathcal{R}^{\mathcal{Q}_m}| \le |\mathcal{R}^{\mathcal{Q}_m} - \mathcal{R}^{\mathcal{Q}_{m+1}}| + |\mathcal{R}^{\mathcal{Q}_{m+1}}| = |\mathcal{R}^{\mathcal{Q}_m} - \mathcal{R}^{\mathcal{Q}_{m+1}}| + 2^{-2q}|\mathcal{R}^{\mathcal{Q}_m}|$, we have the estimate

$$|\mathcal{R}^{\mathcal{Q}_m}| \le \frac{1}{1-2^{-2q}}|\mathcal{R}^{\mathcal{Q}_m} - \mathcal{R}^{\mathcal{Q}_{m+1}}|. \tag{6.24}$$

Thus, by computing the integrals at two or more dyadic levels, we may estimate quadrature residuals and thus the quadrature error.

For the mdG(q) method the only difference is that the basic quadrature rule is one order better, i.e., instead of $2q$ we have $2q+1$, so that

$$|\mathcal{R}^{\mathcal{Q}_m}| \le \frac{1}{1-2^{-1-2q}}|\mathcal{R}^{\mathcal{Q}_m} - \mathcal{R}^{\mathcal{Q}_{m+1}}|. \tag{6.25}$$

6.6. Evaluating E_G. We now present an approach to estimating the quantity $\int_0^T (R(U,\cdot), \varphi - \pi_k\varphi)\ dt$ by direct evaluation, with φ a computed dual solution and $\pi_k\varphi$ a suitably chosen interpolant. In this way we avoid introducing interpolation constants and computing derivatives of the dual. Note, however, that although we do not explicitly compute any derivatives of the dual, the regularity assumed in section 6.1 for the dual solution is still implicitly required for the computed quantities to make sense. Starting now with

$$E_G = \left| \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\ dt \right| \tag{6.26}$$

for the continuous method, we realize that the best possible choice of interpolant, if we want to prevent cancellation, is to choose $\pi_k\varphi$ such that $R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i) \ge 0$ (or ≤ 0) on every local interval I_{ij}. With such a choice of interpolant, we would have

$$E_G = \left| \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)\ dt \right| = \sum_{i=1}^N \sum_{j=1}^{M_i} \alpha_{ij} \int_{I_{ij}} |R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)|\ dt \tag{6.27}$$

with $\alpha_{ij} = \pm 1$. The following lemmas give us an idea of how to choose the interpolant.

LEMMA 6.7. *If, for $i = 1, \ldots, N$, $f_i = f_i(U(t),t) = f_i(U_i(t),t)$ and f_i is linear or, alternatively, $f = f(U(t),t)$ is linear and all components have the same time-steps and order, then every component $R_i(U,\cdot)$ of the* mcG(q) *residual is a Legendre polynomial of order q_{ij} on I_{ij}, for $j = 1, \ldots, M_i$.*

Proof. On every interval I_{ij} the residual component $R_i(U,\cdot)$ is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$. Since the conditions assumed in the statement of the lemma guarantee that the residual is a polynomial of degree q_{ij} on every interval I_{ij}, it is clear that on every such interval it is the q_{ij}th-order Legendre polynomial (or a multiple thereof). □

Even if the rather strict conditions of this lemma do not hold, we can say something similar. The following lemma restates this property in terms of approximations of the residual.

LEMMA 6.8. *Let $\tilde{R}$ be the local L^2-projection of the* mcG(q) *residual R onto the trial space, i.e., $\tilde{R}_i(U,\cdot)|_{I_{ij}}$ is the $L^2(I_{ij})$-projection onto $\mathcal{P}^{q_{ij}}(I_{ij})$ of $R_i(U,\cdot)|_{I_{ij}}$, $j = 1, \ldots, M_i$, $i = 1, \ldots, N$. Then every $\tilde{R}_i(U,\cdot)|_{I_{ij}}$ is a Legendre polynomial of degree q_{ij}.*

Proof. Since $\tilde{R}_i(U,\cdot)$ is the L^2-projection of $R_i(U,\cdot)$ onto $\mathcal{P}^{q_{ij}}(I_{ij})$ on I_{ij}, we have

$$\int_{I_{ij}} \tilde{R}_i(U,\cdot)v\ dt = \int_{I_{ij}} R_i(U,\cdot)v\ dt = 0$$

for all $v \in \mathcal{P}^{q_{ij}-1}(I_{ij})$, so that $\tilde{R}_i(U,\cdot)$ is the q_{ij}th-order Legendre polynomial on I_{ij}. □

To prove the corresponding results for the discontinuous method, we first note some basic properties of Radau polynomials.

LEMMA 6.9. *Let P_q be the qth-order Legendre polynomial on $[-1,1]$. Then the qth-order Radau polynomial, $Q_q(x) = (P_q(x) + P_{q+1}(x))/(x+1)$, has the following property:*

$$I = \int_{-1}^1 Q_q(x)(x+1)^p\ dx = 0 \tag{6.28}$$

for $p = 1, \ldots, q$. Conversely, if f is a polynomial of degree q on $[-1,1]$ and has the property (6.28)*, i.e., $\int_{-1}^1 f(x)(x+1)^p\ dx = 0$ for $p = 1, \ldots, q$, then f is a Radau polynomial.*

Proof. We can write the qth-order Legendre polynomial on $[-1,1]$ as $P_q(x) = \frac{1}{q!2^q} D^q((x^2-1)^q)$. Thus, integrating by parts, we have

$$\begin{aligned} I &= \textstyle\int_{-1}^1 \frac{P_q(x)+P_{q+1}(x)}{x+1}(x+1)^p\ dx \\ &= \textstyle\frac{1}{q!2^q}\int_{-1}^1 D^q((x^2-1)^q + x(x^2-1)^q)(x+1)^{p-1}\ dx \\ &= \textstyle\frac{1}{q!2^q}\int_{-1}^1 D^q((x+1)(x^2-1)^q)(x+1)^{p-1}\ dx \\ &= \textstyle\frac{1}{q!2^q}(-1)^p\int_{-1}^1 D^{q-p}((x+1)(x^2-1)^q)D^p(x+1)^{p-1}\ dx = 0, \end{aligned}$$

since $D^l((x+1)(x^2-1)^q)$ is zero at -1 and 1 for $l < q$. Assume now that f is a polynomial of degree q on $[-1,1]$ with the property (6.28). Since $\{(x+1)^p\}_{p=1}^q$ are linearly independent on $[-1,1]$ and orthogonal to the Radau polynomial Q_q, $\{Q_q(x), (x+1), (x+1)^2, \ldots, (x+1)^q\}$ form a basis for $\mathcal{P}^q([-1,1])$. If then f is orthogonal to the subspace spanned by $\{(x+1)^p\}_{p=1}^q$, we must have $f = cQ_q$ for some constant c, and the proof is complete. □

LEMMA 6.10. *If, for $i = 1, \ldots, N$, $f_i = f_i(U(t),t) = f_i(U_i(t),t)$ and f_i is linear or, alternatively, $f = f(U(t),t)$ is linear and all components have the same time-steps and order, then every component $R_i(U,\cdot)$ of the* mdG(q) *residual is a Radau polynomial of order q_{ij} on I_{ij} for $j = 1, \ldots, M_i$.*

Proof. Note first that by assumption the residual $R_i(U,\cdot)$ is a polynomial of degree q_{ij} on I_{ij}. By the Galerkin orthogonality, we have

$$0 = \int_{I_{ij}} R_i(U,\cdot)v\ dt + [U_i]_{i,j-1}v(t^+_{i,j-1}) \qquad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}),$$

which holds especially for $v(t) = (t - t_{i,j-1})^p$ with $p = 1, \ldots, q$, for which the jump terms disappear. Rescaling to $[-1,1]$, it follows from Lemma 6.9 that the residual $R_i(U,\cdot)$ must be a Radau polynomial on I_{ij}. □

Also for the discontinuous method there is a reformulation in terms of approximations of the residual.

LEMMA 6.11. *Let $\tilde{R}$ be the local L^2-projection of the* mdG(q) *residual R onto the trial space, i.e., $\tilde{R}_i(U,\cdot)|_{I_{ij}}$ is the $L^2(I_{ij})$-projection onto $\mathcal{P}^{q_{ij}}(I_{ij})$ of $R_i(U,\cdot)|_{I_{ij}}$,*

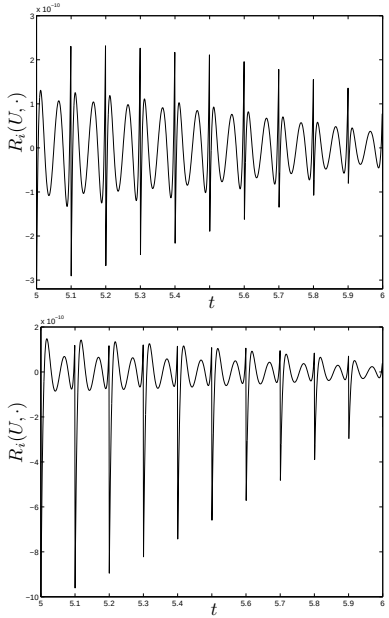

FIG. 6.1. *The Legendre-polynomial residual of the* mcG(q) *method (left) and the Radau-polynomial residual of the* mdG(q) *method (right), for polynomials of degree five, i.e., methods of order* 10 *and* 11*, respectively.*

$j = 1, \ldots, M_i$, $i = 1, \ldots, N$. *Then every* $\tilde{R}_i(U,\cdot)|_{I_{ij}}$ *is a Radau polynomial of degree* q_{ij}.

Proof. Since $\tilde{R}_i(U,\cdot)$ is the L^2-projection of $R_i(U,\cdot)$ onto $\mathcal{P}^{q_{ij}}(I_{ij})$ on I_{ij}, it follows from the Galerkin orthogonality that

$$\int_{I_{ij}} \tilde{R}_i(U,\cdot)v \; dt = \int_{I_{ij}} R_i(U,\cdot)v \; dt = 0$$

for any $v(t) = (t - t_{i,j-1})^p$ with $1 \le p \le q$. From Lemma 6.9 it then follows that $\tilde{R}_i(U,\cdot)$ is a Radau polynomial on I_{ij}. □

We thus know that the mcG(q) residuals are (in the sense of Lemma 6.8) Legendre polynomials on the local intervals and that the mdG(q) residuals are (in the sense of Lemma 6.11) Radau polynomials. This is illustrated in Figure 6.1.

From this information about the residual, we now choose the interpolant. Assume that the polynomial order of the method on some interval is q for the continuous method. Then the dual should be interpolated by a polynomial of degree $q-1$, i.e., we have freedom to interpolate at exactly q points. Since a qth-order Legendre polynomial has q zeros on the interval, we may choose to interpolate the dual exactly at those

points where the residual is zero. This means that if the dual can be approximated well enough by a polynomial of degree q, the product $R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)$ does not change sign on the interval.

For the discontinuous method, we should interpolate the dual with a polynomial of degree q, i.e., we have freedom to interpolate at exactly $q+1$ points. To get rid of the jump terms that are present in the error representation for the discontinuous method, we want to interpolate the dual at the beginning of every interval. This leaves q degrees of freedom. We then choose to interpolate the dual at the q points within the interval where the Radau polynomial is zero.

As a result, we may choose the interpolant in such a way that we have

$$|L_{\varphi T,g}(e)| = \left| \sum_{ij} \int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i) \; dt \right| = \sum_{ij} \alpha_{ij} \int_{I_{ij}} |R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)| \; dt, \tag{6.29}$$

with $\alpha_{ij} = \pm 1$, for both the mcG(q) method and the mdG(q) method (but the interpolants are different). Notice that the jump terms for the discontinuous method have disappeared.

There is now a simple way to compute the integrals $\int_{I_{ij}} R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i) \; dt$. Since the integrands are, in principle, products of two polynomials for which we know the positions of the zeros, the product is a polynomial with known properties. There are then constants C_q (which can be computed numerically), depending on the order and the method, such that

$$\int_{I_{ij}} |R_i(U,\cdot)(\varphi_i - \pi_k\varphi_i)| \; dt = C_{q_{ij}} k_{ij} |R_i(U, t_{ij}^-)| |\varphi_i(t_{ij}) - \pi_k\varphi_i(t_{ij}^-)|. \tag{6.30}$$

Finally, note that there are "computational" counterparts also for the estimates of type E_3 in Theorems 6.5 and 6.6, namely

$$\begin{aligned} |L_{\varphi T,g}(e)| &\le \textstyle\sum_{ij} \int_{I_{ij}} |R_i(U,\cdot)| |\varphi_i - \pi_k\varphi_i| \; dt \\ &= \textstyle\sum_{ij} C'_{q_{ij}} k_{ij}^{q_{ij}} |R_i(U, t_{ij}^-)| \int_{I_{ij}} \frac{1}{k_{ij}^{q_{ij}}} |\varphi_i - \pi_k\varphi_i| \; dt \\ &\le \textstyle\sum_{i=1}^N \tilde{S}_i \max_{j=1,\ldots,M_i} C'_{q_{ij}} k_{ij}^{q_{ij}} |R_i(U, t_{ij}^-)|, \end{aligned} \tag{6.31}$$

with $\tilde{S}_i = \int_0^T \frac{1}{k_i^{q_i}} |\varphi_i - \pi_k\varphi_i| \; dt$ for the continuous method and similarly for the discontinuous method.

6.7. The total error. The total error is composed of three parts—the Galerkin error, E_G, the computational error, E_C and the quadrature error, E_Q:

$$|L_{\varphi T,g}(e)| \le E_G + E_C + E_Q. \tag{6.32}$$

As an example, choosing estimate E_3 of Theorems 6.5 and 6.6 we have the following (approximate) error estimate for the mcG(q) method:

$$|L_{\varphi T,g}(e)| \le \sum_{i=1}^N \left[S_i^{[q_i]} \max_{[0,T]} \{ C_{q_i-1} k_i^{q_i} r_i \} + \bar{S}_i^{[0]} \max_{[0,T]} |\mathcal{R}_i^{\mathcal{C}}| + \bar{S}_i^{[0]} \max_{[0,T]} |\mathcal{R}_i^{\mathcal{Q}}| \right]; \tag{6.33}$$

for the mdG(q) method we have

$$|L_{\varphi_T,g}(e)| \le \sum_{i=1}^{N} \left[S_i^{[q_i+1]} \max_{[0,T]} \left\{ C_{q_i} k_i^{q_i+1} \bar{r}_i \right\} + \bar{S}_i^{[0]} \max_{[0,T]} |\mathcal{R}_i^{\mathcal{C}}| + \bar{S}_i^{[0]} \max_{[0,T]} |\mathcal{R}_i^{\mathcal{Q}}| \right]. \tag{6.34}$$

These estimates containing Galerkin errors, computational errors, and quadrature errors also include numerical round-off errors (included in the computational error). *Modelling errors* could also be similarly accounted for since these are closely related to quadrature errors, in that both errors can be seen as arising from integrating the wrong right-hand side.

The true global error may thus be estimated in terms of computable stability factors and residuals. We expect the estimate for the Galerkin error, E_G, to be quite sharp, while E_C and E_Q may be less sharp. Even sharper estimates are obtained using estimates E_0, E_1, or E_2 of Theorems 6.5 and 6.6.

6.8. An a posteriori error estimate for the dual. We conclude this section by proving a computable a posteriori error estimate for the dual problem. To compute the stability factors used in the error estimates presented above, we solve the dual problem numerically, and we thus face the problem of estimating the error in the stability factors.

To demonstrate how relative errors of stability factors can be estimated using the same technique as above, we compute the relative error for the stability factor $S_\varphi(T)$, defined as

$$S_\varphi(T) = \sup_{\|\varphi(T)\|=1} \int_0^T \|\varphi\| \, dt \tag{6.35}$$

for a computed approximation Φ of the dual solution φ.

To estimate the relative error of the stability factor, we use the error representation of Theorem 6.1 to represent the $L^1([0,T],\mathbb{R}^N)$-error of Φ in terms of the residual of Φ and the dual of the dual, ω. In [28] we prove the following lemma, from which the estimate follows.

LEMMA 6.12. *Let φ be the dual solution with stability factor $S_\varphi(t)$, i.e., with data $\|\varphi(t)\| = 1$ specified at time t, and let ω be the dual of the dual. We then have the following estimate:*

$$\|\omega(t)\| \le S_\varphi(T-t) \qquad \forall t \in [0,T]. \tag{6.36}$$

THEOREM 6.13. *Let Φ be a continuous approximation of the dual solution with residual R_Φ, and assume that $S_\varphi(t)/S_\varphi(T)$ is bounded by C on $[0,T]$. Then the following estimate holds for the relative error of the stability factor $S_\Phi(T)$:*

$$|S_\Phi(T) - S_\varphi(T)|/S_\varphi(T) \le C \int_0^T \|R_\Phi\| \, dt, \tag{6.37}$$

and for many problems we may take $C = 1$.

Proof. By Corollary 6.3, we have an expression for the $L^1([0,T],\mathbb{R}^N)$-error of the dual, so that

$$\begin{aligned} |S_\Phi(T) - S_\varphi(T)| &= \left| \int_0^T \|\Phi\| \, dt - \int_0^T \|\varphi\| \, dt \right| \\ &= \left| \int_0^T (\|\Phi\| - \|\varphi\|) \, dt \right| \le \int_0^T \|\Phi - \varphi\| \, dt \\ &= \|\Phi - \varphi\|_{L^1([0,T],\mathbb{R}^n)} = \int_0^T (R_\Phi, \omega(T-\cdot)) \, dt \\ &\le \int_0^T \|R_\Phi\| \|\omega(T-\cdot)\| \, dt. \end{aligned} \tag{6.38}$$

With C defined as above it now follows by Lemma 6.12 that

$$|S_\Phi(T) - S_\varphi(T)| \le C \int_0^T \|R_\Phi\| \, dt \; S_\varphi(T),$$

and the proof is complete. □

Remark 6.3. We also have to take into account quadrature errors when evaluating (6.35). This can be done in many ways; see, e.g., [9].

Appendix A. Derivation of the methods.

This section contains some details left out of the discussion of section 4.

A.1. The mcG(q) method. To rewrite the local problem in a more explicit form, let $\{s_n\}_{n=0}^q$ be a set of nodal points on $[0,1]$, with $s_0 = 0$ and $s_q = 1$. A good choice for the cG(q) method is the Lobatto points of $[0,1]$. Now, let τ_{ij} be the linear mapping from the interval I_{ij} to $(0,1]$, defined by

$$\tau_{ij}(t) = \frac{t - t_{i,j-1}}{t_{ij} - t_{i,j-1}}, \tag{A.1}$$

and let $\{\lambda_n^{[q]}\}_{n=0}^q$ be the $\{s_n\}_{n=0}^q$ Lagrange basis functions for $\mathcal{P}^q([0,1])$ on $[0,1]$, i.e.,

$$\lambda_n^{[q]}(s) = \frac{(s-s_0)\cdots(s-s_{n-1})(s-s_{n+1})\cdots(s-s_q)}{(s_n-s_0)\cdots(s_n-s_{n-1})(s_n-s_{n+1})\cdots(s_n-s_q)}. \tag{A.2}$$

We can then express U_i on I_{ij} in the form

$$U_i(t) = \sum_{n=0}^{q} \xi_{ijn} \lambda_n^{[q_{ij}]}(\tau_{ij}(t)), \tag{A.3}$$

and choosing the $\lambda_m^{[q-1]}$ as test functions we can formulate the local problem (4.3) as follows: Find $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, with $\xi_{ij0} = \xi_{i,j-1,q_{i,j-1}}$, such that for $m = 0, \ldots, q_{ij} - 1$

$$\int_{I_{ij}} \sum_{n=0}^{q_{ij}} \xi_{ijn} \frac{d}{dt} \left[\lambda_n^{[q_{ij}]}(\tau_{ij}(t)) \right] \lambda_m^{[q_{ij}-1]}(\tau_{ij}(t)) \, dt = \int_{I_{ij}} f_i(U(t),t) \lambda_m^{[q_{ij}-1]}(\tau_{ij}(t)) \, dt. \tag{A.4}$$

To simplify the notation, we drop the $_{ij}$ subindices and assume that the time-interval is $[0,k]$, keeping in mind that, although not visible, all other components are present in f. We thus seek to determine the coefficients $\{\xi_n\}_{n=1}^q$ with ξ_0 given, such that for $m = 1, \ldots, q$ we have

$$\sum_{n=0}^{q} \xi_n \frac{1}{k} \int_0^k \dot{\lambda}_n^{[q]}(\tau(t)) \lambda_{m-1}^{[q-1]}(\tau(t)) \, dt = \int_0^k f \lambda_{m-1}^{[q-1]}(\tau(t)) \, dt, \tag{A.5}$$

or simply

$$\sum_{n=1}^{q} a_{mn}^{[q]} \xi_n = b_m, \tag{A.6}$$

where

$$a_{mn}^{[q]} = \int_0^1 \dot{\lambda}_n^{[q]}(t) \lambda_{m-1}^{[q-1]}(t)\ dt \tag{A.7}$$

and

$$b_m = \int_0^k f \lambda_{m-1}^{[q-1]}(\tau(t))\ dt - a_{m0}\xi_0. \tag{A.8}$$

We explicitly compute the inverse $\bar{A}^{[q]} = (\bar{a}_{mn}^{[q]})$ of the matrix $A^{[q]} = (a_{mn}^{[q]})$. Thus, switching back to the full notation, we get

$$\xi_{ijm} = -\xi_0 \sum_{n=1}^{q} \bar{a}_{mn}^{[q]} a_{n0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt, \qquad m = 1, \ldots, q_{ij}, \tag{A.9}$$

where the *weight functions* $\{w_m^{[q]}\}_{m=1}^q$ are given by

$$w_m^{[q]} = \sum_{n=1}^{q} \bar{a}_{mn}^{[q]} \lambda_{n-1}^{[q-1]}, \qquad m = 1, \ldots, q. \tag{A.10}$$

Following Lemma A.1 below, this relation may be somewhat simplified.

LEMMA A.1. *For the* mcG(q) *method, we have*

$$\sum_{n=1}^{q} \bar{a}_{mn}^{[q]} a_{n0} = -1.$$

Proof. Assume the interval to be $[0,1]$. The value is independent of f so we may take $f = 0$. We thus want to prove that if $f = 0$, then $\xi_n = \xi_0$ for $n = 1, \ldots, q$, i.e., $U = U_0$ on $[0,1]$ since $\{\lambda_n^{[q]}\}_{n=0}^q$ is a nodal basis for $\mathcal{P}^q([0,1])$. Going back to the Galerkin orthogonality (4.4), this amounts to showing that if

$$\int_0^1 \dot{U} v\ dt = 0 \qquad \forall v \in \mathcal{P}^{q-1}([0,1]),$$

with $U \in \mathcal{P}^q([0,1])$, then U is constant on $[0,1]$. This follows by taking $v = \dot{U}$. Thus, $\xi_n = \xi_0$ for $n = 1, \ldots, q$, so that the value of $\sum_{n=1}^q \bar{a}_{mn}^{[q]} a_{n0}$ must be -1. This completes the proof. □

The mcG(q) method thus reads as follows: For every local interval I_{ij}, find $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, with $\xi_{ij0} = \xi_{i,j-1,q_{i,j-1}}$, such that

$$\xi_{ijm} = \xi_{ij0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt, \qquad m = 1, \ldots, q_{ij}, \tag{A.11}$$

for certain weight functions $\{w_n^{[q]}\}_{m=1}^q \subset \mathcal{P}^{q-1}(0,1)$, and where the initial condition is specified by $\xi_{i00} = u_i(0)$ for $i = 1, \ldots, N$.

The weight functions may be computed analytically for small q, and for general q they are easy to compute numerically.

A.2. The mdG(q) method. We now make the same ansatz as for the continuous method,

$$U_i(t) = \sum_{n=0}^{q} \xi_{ijn} \lambda_n^{[q_{ij}]}(\tau_{ij}(t)), \tag{A.12}$$

where the difference is that we now have $q+1$ degrees of freedom on every interval, since we no longer have the continuity requirement for the trial functions. We make the assumption that the nodal points for the nodal basis functions are chosen so that

$$s_q = 1, \tag{A.13}$$

i.e., the end-point of every subinterval is a nodal point for the basis functions.

With this ansatz, we get the following set of equations for determining $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$:

$$\left(\sum_{n=0}^{q_{ij}} \xi_{ijn} \lambda_n^{[q_{ij}]}(0) - \xi_{ij0}^- \right) \lambda_m^{[q_{ij}]}(0) + \int_{I_{ij}} \sum_{n=0}^{q_{ij}} \xi_{ijn} \frac{d}{dt} \left[\lambda_n^{[q_{ij}]}(\tau_{ij}(t)) \right] \lambda_m^{[q_{ij}]}(\tau_{ij}(t))\ dt \tag{A.14}$$

$$= \int_{I_{ij}} f_i(U(t),t) \lambda_m^{[q_{ij}]}(\tau_{ij}(t))\ dt \tag{A.15}$$

for $m = 0, \ldots, q_{ij}$, where we use ξ_{ij0}^- to denote $\xi_{i,j-1,q_{i,j-1}}$, i.e., the value at the right end-point of the previous interval. To simplify the notation, we drop the subindices $_{ij}$ again and rewrite to $[0,k]$. We thus seek to determine the coefficients $\{\xi_n\}_{n=0}^q$ such that for $m = 0, \ldots, q$ we have

$$\left(\sum_{n=0}^{q} \xi_n \lambda_n^{[q]}(0) - \xi_0^- \right) \lambda_m^{[q]}(0) + \sum_{n=0}^{q} \xi_n \frac{1}{k} \int_0^k \dot{\lambda}_n^{[q]}(\tau(t)) \lambda_m^{[q]}(\tau(t))\ dt = \int_0^k f \lambda_m^{[q]}(\tau(t))\ dt, \tag{A.16}$$

or simply

$$\sum_{n=0}^{q} a_{mn}^{[q]} \xi_n = b_m, \tag{A.17}$$

where

$$a_{mn}^{[q]} = \int_0^1 \dot{\lambda}_n^{[q]}(t) \lambda_m^{[q]}(t)\ dt + \lambda_n^{[q]}(0) \lambda_m^{[q]}(0) \tag{A.18}$$

and

$$b_m^{[q]} = \int_0^k f \lambda_m^{[q]}(\tau(t))\ dt + \xi_0^- \lambda_m^{[q]}(0). \tag{A.19}$$

Now, let $A^{[q]}$ be the $(q+1)\times(q+1)$ matrix $A^{[q]} = (a_{mn}^{[q]})$ with inverse $\bar{A}^{[q]} = (\bar{a}_{mn}^{[q]})$. Then, switching back to the full notation, we have

$$\xi_{ijm} = \xi_{ij0}^- \sum_{n=0}^{q} \bar{a}_{mn}^{[q]} \lambda_n^{[q]}(0) + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt, \qquad m = 0, \ldots, q_{ij}, \tag{A.20}$$

where the *weight functions* $\{w_n^{[q]}\}_{n=0}^q$ are given by

$$w_m^{[q]} = \sum_{n=0}^{q} \bar{a}_{mn}^{[q]} \lambda_n^{[q]}, \qquad m = 0, \ldots, q. \tag{A.21}$$

As for the continuous method, this may be somewhat simplified.

LEMMA A.2. *For the* mdG(q) *method, we have*

$$\sum_{n=0}^{q} \bar{a}_{mn}^{[q]} \lambda_n^{[q]}(0) = 1.$$

Proof. As in the proof for the mcG(q) method, assume that the interval is $[0,1]$. Since the value of the expression is independent of f we can take $f = 0$. We thus want to prove that if $f = 0$, then the solution U is constant. By the Galerkin orthogonality, we have

$$[U]_0 v(0) + \int_0^1 \dot{U} v \, dt = 0 \qquad \forall v \in \mathcal{P}^q(0,1),$$

with $U \in \mathcal{P}^q(0,1)$. Taking $v = U - U(0^-)$, we have

$$\begin{aligned} 0 &= ([U]_0)^2 + \textstyle\int_0^1 \dot{U}(U - U(0^-)) \, dt = ([U]_0)^2 + \frac{1}{2}\int_0^1 \frac{d}{dt}(U - U(0^-))^2 \, dt \\ &= \tfrac{1}{2}(U(0^+) - U(0^-))^2 + \tfrac{1}{2}(U(1) - U(0^-))^2, \end{aligned}$$

so that $[U]_0 = 0$. Now take $v = \dot{U}$. This gives $\int_0^1 (\dot{U})^2 \, dt = 0$. Since then both $[U]_0 = 0$ and $\dot{U} = 0$ on $[0,1]$, U is constant and equal to $U(0^-)$, and the proof is complete. □

The mdG(q) method thus reads as follows: For every local interval I_{ij}, find $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, such that for $m = 0, \ldots, q_{ij}$ we have

$$\xi_{ijm} = \xi_{ij0}^- + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) \, f_i(U(t), t) \, dt \tag{A.22}$$

for certain weight functions $\{w_n^{[q]}\}_{n=0}^q \subset \mathcal{P}^q(0,1)$.

REFERENCES

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

[2] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.

[3] J. Butcher, *The Numerical Analysis of Ordinary Differential Equations. Runge-Kutta and General Linear Methods*, John Wiley, Chichester, UK, 1987.

[4] G. Dahlquist, *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*, Kungl. Tekn. Högsk Handl. Stockholm. No., 130 (1959).

[5] R. Davé, J. Dubinski, and L. Hernquist, *Parallel treesph*, New Astronomy, 2 (1997), pp. 277–297.

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

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

[8] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, in Acta Numerica, 1995, Acta Numer., Cambridge University Press, Cambridge, 1995, pp. 105–158.

[9] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Computational Differential Equations*, Cambridge University Press, Cambridge, 1996.

[10] K. Eriksson and C. Johnson, *Adaptive Finite Element Methods for Parabolic Problems* III*: Time Steps Variable in Space*, manuscript, Chalmers University of Technology, Göteborg, Sweden.

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

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

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

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

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

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

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

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

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

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

[21] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations.* I*. Nonstiff Problems*, Springer Ser. Comput. Math. 8, Springer-Verlag, Berlin, 1987.

[22] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations.* II*. Stiff and Differential-Algebraic Problems*, Springer Ser. Comput. Math. 14, Springer-Verlag, Berlin, 1991.

[23] P. Hansbo, *A note on energy conservation for hamiltonian systems using continuous time finite elements*, Comm. Numer. Methods Engrg., 17 (2001), pp. 863–869.

[24] 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 (1988), pp. 908–926.

[25] O. Kessel-Deynet, *Berücksichtigung ionisierender Strahlung im Smoothed-Particle-Hydrodynamics-Verfahren und Anwendung auf die Dynamik von Wolkenkernen im Strahlungsfeld massiver Sterne*, Ph.D. thesis, Naturwissenschaftlich-Mathematischen Gesamtfakultät, Ruprecht-Karls-Universität, Heidelberg, 1999.

[26] A. Logg, *Multi-Adaptive Error Control for ODES*, NA Group Report 98/20, Oxford University Computing Laboratory, Oxford, UK, 1998; also available online from http://www.phi.chalmers.se/preprints/abstracts/preprint-2000-03.html.

[27] A. Logg, *A Multi-Adaptive ODE-Solver*, M.Sc. thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, Sweden, 1998; also available online from http://www.phi.chalmers.se/preprints/abstracts/preprint-2000-02.html.

[28] A. Logg, *Multi-Adaptive Galerkin Methods for ODES* I*: Theory & Algorithms*, Chalmers Finite Element Center Preprint 2001–09, http://www.phi.chalmers.se/preprints/abstracts/preprint-2000-09.html (25 February 2001).

[29] A. Logg, *Multi-adaptive Galerkin methods for ODES* II*: Implementation and applications*, SIAM J. Sci. Comput., submitted.

[30] A. Logg, *Tanganyika, version* 1.2.1, http://www.phi.chalmers.se/tanganyika/ (10 May 2001).

[31] J. Makino and S. Aarseth, *On a hermite integrator with Ahmad–Cohen scheme for gravitational many-body problems*, Publ. Astron. Soc. Japan, 44 (1992), pp. 141–151.

[32] K.-S. Moon, A. Szepessy, R. Tempone, and G. Zourakis, *Adaptive Approximation of Differential Equations Based on Global and Local Errors*, preprint. TRITA-NA-0006 NADA, KTH, Stockholm, Sweden, 2000.

[33] S. Osher and R. Sanders, *Numerical approximations to nonlinear conservation laws with locally varying time and space grids*, Math. Comp., 41 (1983), pp. 321–336.

[34] L. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, London, 1994.

SIAM J. SCI. COMPUT.
Vol. 25, No. 4, pp. 1119–1141

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES II: IMPLEMENTATION AND APPLICATIONS*

ANDERS LOGG†

Abstract. Continuing the discussion of the multi-adaptive Galerkin methods mcG(q) and mdG(q) presented in [A. Logg, *SIAM J. Sci. Comput.*, 24 (2003), pp. 1879–1902], we present adaptive algorithms for global error control, iterative solution methods for the discrete equations, features of the implementation *Tanganyika*, and computational results for a variety of ODEs. Examples include the Lorenz system, the solar system, and a number of time-dependent PDEs.

Key words. multi-adaptivity, individual time-steps, local time-steps, ODE, continuous Galerkin, discontinuous Galerkin, global error control, adaptivity, mcG(q), mdG(q), applications, Lorenz, solar system, Burger

AMS subject classifications. 65L05, 65L07, 65L20, 65L50, 65L60, 65L70, 65L80

DOI. 10.1137/S1064827501389734

1. Introduction.

In this paper we apply the multi-adaptive Galerkin methods mcG(q) and mdG(q), presented in [10], to a variety of problems chosen to illustrate the potential of multi-adaptivity. Throughout this paper, we solve the ODE initial value problem

$$\left\{ \begin{array}{rcl} \dot{u}(t) & = & f(u(t),t),\ t \in (0,T], \\ u(0) & = & u_0, \end{array} \right. \tag{1.1}$$

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

We refer to [10] for a detailed description of the multi-adaptive methods. Here we recall that each component $U_i(t)$ of the approximate solution $U(t)$ is a piecewise polynomial of degree $q_i = q_i(t)$ on a partition of $(0,T]$ into M_i subintervals of lengths $k_{ij} = t_{ij} - t_{i,j-1}$, $j = 1, \ldots, M_i$. On the interval $I_{ij} = (t_{i,j-1}, t_{ij}]$, component $U_i(t)$ is thus a polynomial of degree q_{ij}.

Before presenting the examples, we discuss adaptive algorithms for global error control and iterative solution methods for the discrete equations. We also give a short description of the implementation *Tanganyika*.

2. Adaptivity.

In this section we describe how to use the a posteriori error estimates presented in [10] in an adaptive algorithm.

2.1. A strategy for adaptive error control.

The goal of the algorithm is to produce an approximate solution $U(t)$ to (1.1) within a given tolerance TOL for the error $e(t) = U(t) - u(t)$ in a given norm $\|\cdot\|$. The adaptive algorithm is based on the a posteriori error estimates, presented in [10], of the form

$$\|e\| \leq \sum_{i=1}^{N} \sum_{j=1}^{M_i} k_{ij}^{p_{ij}+1} r_{ij} s_{ij} \tag{2.1}$$

*Received by the editors May 23, 2001; accepted for publication (in revised form) May 1, 2003; published electronically December 5, 2003.
http://www.siam.org/journals/sisc/25-4/38973.html
†Department of Computational Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden (logg@math.chalmers.se).

or

$$\|e\| \leq \sum_{i=1}^{N} S_i \max_j k_{ij}^{p_{ij}} r_{ij}, \tag{2.2}$$

where $\{s_{ij}\}$ are *stability weights*, $\{S_i\}$ are *stability factors* (including interpolation constants), r_{ij} is a local measure of the residual $R_i(U,\cdot) = \dot{U}_i - f(U,\cdot)$ of the approximate solution $U(t)$, and where we have $p_{ij} = q_{ij}$ for mcG(q) and $p_{ij} = q_{ij} + 1$ for mdG(q).

We use (2.2) to determine the individual time-steps, which should then be chosen as

$$k_{ij} = \left(\frac{\mathrm{TOL}/N}{S_i\ r_{ij}} \right)^{1/p_{ij}}. \tag{2.3}$$

We use (2.1) to evaluate the resulting error at the end of the computation, noting that (2.1) is sharper than (2.2).

The adaptive algorithm may then be expressed as follows: Given a tolerance $\mathrm{TOL} > 0$, make a preliminary guess for the stability factors and then

(i) solve the primal problem with time-steps based on (2.3).
(ii) solve the dual problem and compute stability factors and stability weights.
(iii) compute an error estimate E based on (2.1).
(iv) if $E \leq \mathrm{TOL}$, then stop, and if not, go back to (i).

Although this seems simple enough, there are some difficulties involved. For one thing, choosing the time-steps based on (2.3) may be difficult, since the residual depends implicitly on the time-step. Furthermore, we have to choose the proper data for the dual problem to obtain a meaningful error estimate. We now discuss these issues.

2.2. Regulating the time-step.

To avoid the implicit dependence on k_{ij} for r_{ij} in (2.3), we may try replacing (2.3) with

$$k_{ij} = \left(\frac{\mathrm{TOL}/N}{S_i\ r_{i,j-1}} \right)^{1/p_{ij}}. \tag{2.4}$$

Following this strategy, if the time-step on an interval is small (and thus also is the residual), the time-step for the next interval will be large, so that (2.4) introduces unwanted oscillations in the size of the time-step. We therefore try to be a bit more conservative when choosing the time-step to obtain a smoother time-step sequence. For (2.4) to work, the time-steps on adjacent intervals need to be approximately the same, and so we may think of choosing the new time-step as the (geometric) mean value of the previous time-step and the time-step given by (2.4). This works surprisingly well for many problems, meaning that the resulting time-step sequences are comparable to what can be obtained with more advanced regulators.

We have also used standard *PID* (or just *PI*) regulators from control theory with the goal of satisfying

$$S_i k_{ij}^{p_{ij}} r_{ij} = \mathrm{TOL}/N, \tag{2.5}$$

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

$$p_{ij} \log k_{ij} + \log r_{ij} = C_i, \tag{2.6}$$

with maximal time-steps $\{k_{ij}\}$, following work by Söderlind [15] and Gustafsson, Lundh, and Söderlind [6]. This type of regulator performs a little better than the simple approach described above, provided the parameters of the regulator are well tuned.

2.3. Choosing data for the dual. Different choices of data φ_T and g for the dual problem give different error estimates, as described in [10], where estimates for the quantity

$$L_{\varphi_T,g}(e) = (e(T),\varphi_T) + \int_0^T (e,g)\ dt$$

were derived. The simplest choices are $g = 0$ and $(\varphi_T)_i = \delta_{in}$ for control of the final time error of the nth component. For control of the l^2-norm of the error at final time, we take $g = 0$ and $\varphi_T = \bar{e}(T)/\|\bar{e}(T)\|$ with an approximation $\bar{e}$ of the error e. Another possibility is to take $\varphi_T = 0$ and $g_i(t) = \delta_{in}$ for control of the average error in component n.

If the data for the dual problem are incorrect, the error estimate may also be incorrect: with φ_T (or g) orthogonal to the error, the error representation gives only $0 \leq \text{TOL}$. In practice, however, the dual—or at least the stability factors—seems to be quite insensitive to the choice of data for many problems so that it is, in fact, possible to guess the data for the dual.

2.4. Adaptive quadrature. In practice, integrals included in the formulation of the two methods mcG(q) and mdG(q) have to be evaluated using numerical quadrature. To control the resulting quadrature error, the quadrature rule can be chosen adaptively, based on estimates of the quadrature error presented in [10].

2.5. Adaptive order, *q*-adaptivity. The formulations of the methods include the possibility of individual and varying orders for the different components, as well as different and varying time-steps. The method is thus *q-adaptive* (or *p*-adaptive) in the sense that the order can be changed. At this stage, however, lacking a strategy for when to increase the order and when to decrease the time-step, the polynomial orders have to be chosen in an ad hoc fashion for every interval. One way to choose time-steps and orders could be to solve over some short time-interval with different time-steps and orders, and optimize the choice of time-steps and orders with respect to the computational time required for achieving a certain accuracy. If we suspect that the problem will change character, we will have to repeat this procedure at a number of control points.

3. Solving the discrete equations. In this section we discuss how to solve the discrete equations that we obtain when we discretize (1.1) with the multi-adaptive Galerkin methods. We do this in two steps. First, we present a simple explicit strategy, and then we extend this strategy to an iterative method.

3.1. A simple strategy. As discussed in [10], the nonlinear discrete algebraic equations for the mcG(q) method (including numerical quadrature) to be solved on every local interval I_{ij} take the form

$$\xi_{ijm} = \xi_{ij0} + k_{ij}\sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]}\ f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})),\tau_{ij}^{-1}(s_n^{[q_{ij}]})),\ m = 1,\ldots,q_{ij}, \tag{3.1}$$

where $\{\xi_{ijm}\}_{m=1}^{q_{ij}}$ are the degrees of freedom to be determined for component $U_i(t)$ on interval I_{ij}, $\{w_{mn}^{[q_{ij}]}\}_{m=1,n=0}^{q_{ij}}$ are weights, $\{s_n^{[q_{ij}]}\}_{n=0}^{q_{ij}}$ are quadrature points, and τ_{ij} maps I_{ij} to $(0,1]$: $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. The discrete equations for the mdG(q) method are similar in structure and so we focus on the mcG(q) method.

The equations are conveniently written in fixed point form, so we may apply fixed point iteration directly to (3.1); i.e., we make an initial guess for the values of $\{\xi_{ijm}\}_{m=1}^{q_{ij}}$, e.g., $\xi_{ijm} = \xi_{ij0}$ for $m = 1,\ldots,q_{ij}$, and then compute new values for these coefficients from (3.1), repeating the procedure until convergence.

Note that component $U_i(t)$ is coupled to all other components through the right-hand side $f_i = f_i(U,\cdot)$. This means that we have to know the solution for all other components in order to compute $U_i(t)$. Conversely, we have to know $U_i(t)$ to compute the solutions for all other components, and since all other components step with different time-steps, it seems at first very difficult to solve the discrete equations (3.1).

As an initial simple strategy we may try to solve the system of nonlinear equations (3.1) by direct fixed point iteration. All unknown values, for the component itself and all other *needed* components, are interpolated or extrapolated from their latest known values. Thus, if for component i we need to know the value of component l at some time t_i, and we only know values for component l up to time $t_l < t_i$, the strategy is to extrapolate $U_l(t)$ from the interval containing t_l to time t_i, according to the order of $U_l(t)$ on that interval.

In what order should the components now make their steps? Clearly, to update a certain component on a specific interval, we would like to use the best possible values of the other components. This naturally leads to the following strategy:

(3.2) *The last component steps first.*

This means that we should always make a step with the component that is closest to time $t = 0$. Eventually (or after one step), this component catches up with one of the other components, which then in turn becomes the last component, and the procedure continues according to the strategy (3.2), as described in Figure 3.1.

This gives an explicit time-stepping method in which each component is updated individually once, following (3.2), and in which we never go back to correct mistakes. This corresponds to fixed point iterating once on the discrete equations (3.1), which implicitly define the solution. We now describe how to extend this explicit time-stepping strategy to an iterative process, in which the discrete equations are solved to within a prescribed tolerance.

3.2. An iterative method. To extend the explicit strategy described in the previous section to an iterative method, we need to be able to go back and redo iterations if necessary. We do this by arranging the elements—we think of an element as a component $U_i(t)$ on a local interval I_{ij}—in a *time-slab*. This contains a number of elements, a minimum of N elements, and moves forward in time. On every time-slab, we have to solve a large system of equations, namely, the system composed of the element equations (3.1) for every element within the time-slab. We solve this system of equations iteratively, by direct fixed point iteration, or by some other method as described below, starting from the last element in the time-slab, i.e., the one closest to $t = 0$, and continuing forward to the first element in the time-slab, i.e., the one closest to $t = T$. These iterations are then repeated from beginning to end until convergence, which is reached when the computational residuals, as defined in [10], on all elements are small enough.

3.3. The time-slab. The time-slab can be constructed in many ways. One is by *dyadic* partitioning, in which we compute new time-steps for all components, based

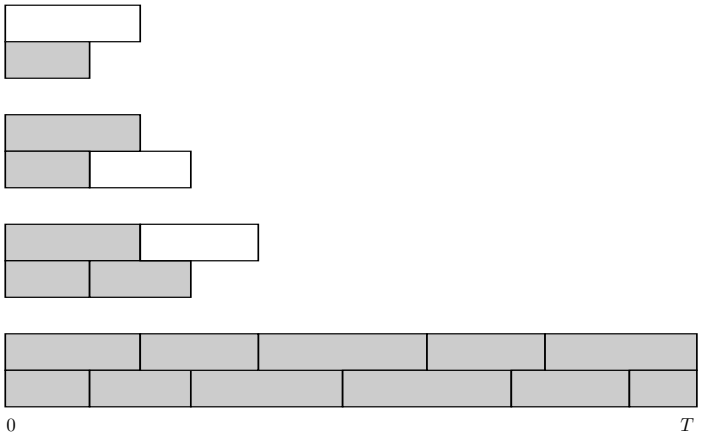

FIG. 3.1. *The last component steps first and all needed values are extrapolated or interpolated.*

on residuals and stability properties, choose the largest time-step K as the length of the new time-slab, and then, for every component, choose the time-step as a fraction $K/2^n$. The advantage of such a partition are that the time-slab has *straight edges*; i.e., for every time-slab there is a common point in time t' (the end-point of the time-slab) which is an end-point for the last element of every component in the time-slab, and that the components have many common nodes. The disadvantage is that the choice of time-steps is constrained.

Another possibility is a *rational* partition of the time-slab. We choose the largest individual time-step K as the length of the time-slab, and time-steps for the remaining components are chosen as fractions of this large time-step, $K/2$, $K/3$, $K/4$, and so on. In this way we increase the set of possible time-step selections, as compared to dyadic partitioning, but the number of common nodes shared between different components is decreased.

A third option is to not impose any constraint at all on the selection of time-steps—except that we match the final time end-point. The time-steps may vary also within the time-slab for the individual components. The price we have to pay is that we have in general no common nodes, and the edges of the time-slab are no longer straight. We illustrate the three choices of partitioning schemes in Figure 3.2. Although dyadic or rational partitioning is clearly advantageous in terms of easier bookkeeping and common nodes, we focus below on unconstrained time-step selection. In this way we stay close to the original, general formulation of the multi-adaptive methods. We refer to this as the *time-crawling* approach.

3.4. The time-crawling approach. The construction of the time-slab brings with it a number of technical and algorithmic problems. We will not discuss here the implementational and data structural aspects of the algorithm—there will be much to keep track of and this has to be done in an efficient way—but we will give a brief

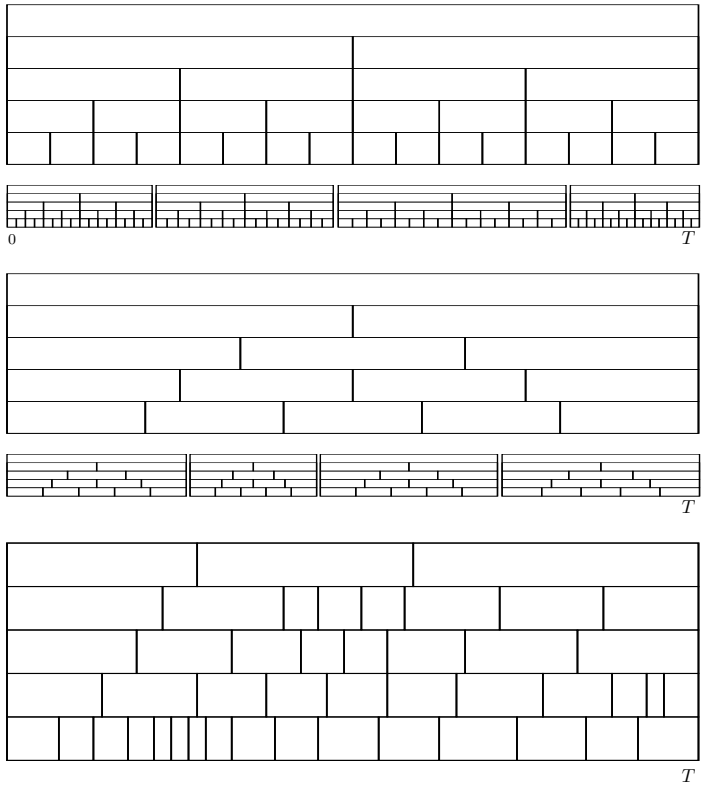

FIG. 3.2. *Different choices of time-slabs. Top: a dyadic partition of the time-slab; middle: a rational partition; bottom: a partition used in the time-crawling approach, where the only restriction on the time-steps is that we match the final time end-point T.*

account of how the time-slab is formed and updated.

Assume that in some way we have formed a time-slab, such as the one in Figure 3.3. We make iterations on the time-slab, starting with the last element and continuing to the right. After iterating through the time-slab a few times, the computational (discrete) residuals, corresponding to the solution of the discrete equations (3.1), on all elements have decreased below a specified tolerance for the computational error, indicating convergence.

For the elements at the front of the slab (those closest to time $t = T$), the values have been computed using extrapolated values of many of the other elements. The strategy now is to leave behind *only those elements that are fully covered by all*

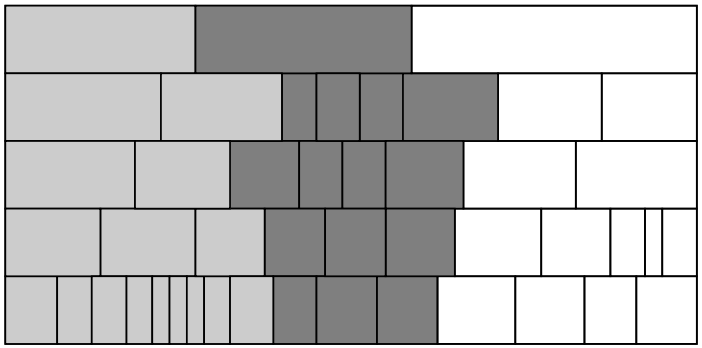

FIG. 3.3. *The time-slab used in the time-crawling approach to multi-adaptive time-stepping (dark grey). Light grey indicates elements that have already been computed.*

other elements. These are then cut off from the time-slab, which then decreases in size. Before starting the iterations again, we have to form a new time-slab. This will contain the elements of the old time-slab that were not removed, and a number of new elements. We form the new time-slab by requiring that all elements of the previous time-slab be totally covered within the new time-slab. In this way we know that every new time-slab will produce at least N new elements. The time-slab is thus crawling forward in time rather than marching.

An implementation of the method then contains the three consecutive steps described above: iterating on an existing time-slab, decreasing the size of the time-slab (cutting off elements at the end of the time-slab, i.e., those closest to time $t = 0$), and incorporating new elements at the front of the time-slab.

Remark 3.1. Even if an element within the time-slab is totally covered by all other elements, the values on this element still may not be completely determined, if they are based on the values of some other element that is not totally covered, or if this element is based on yet another element that is not totally covered, and so on. To avoid this, one can impose the requirement that the time-slabs should have straight edges.

3.5. Diagonal Newton. For stiff problems the time-step condition required for convergence of direct fixed point iteration is too restrictive, and we need to use a more implicit solution strategy.

Applying a full Newton's method, we increase the range of allowed time-steps and also the convergence rate, but this is costly in terms of memory and computational time, which is especially important in our setting, since the size of the slab may often be much larger than the number of components, N (see Figure 3.3). We thus look for a simplified Newton's method which does not increase the cost of solving the problem, as compared to direct fixed point iteration, but still has some advantages of the full Newton's method.

Consider for simplicity the case of the multi-adaptive backward Euler method,

i.e., the mdG(0) method with end-point quadrature. On every element we then want to solve

$$U_{ij} = U_{i,j-1} + k_{ij} f_i(U(t_{ij}), t_{ij}). \tag{3.3}$$

In order to apply Newton's method we write (3.3) as

$$F(V) = 0 \tag{3.4}$$

with $F_i(V) = U_{ij} - U_{i,j-1} - k_{ij} f_i(U(t_{ij}), t_{ij})$ and $V_i = U_{ij}$. Newton's method is then

$$V^{n+1} = V^n - (F'(V^n))^{-1} F(V^n). \tag{3.5}$$

We now simply replace the Jacobian with its diagonal so that for component i we have

$$U_{ij}^{n+1} = U_{ij}^n - \frac{U_{ij}^n - U_{i,j-1} - k_{ij} f_i}{1 - k_{ij} \frac{\partial f_i}{\partial u_i}} \tag{3.6}$$

with the right-hand side evaluated at V^n. We now note that we can rewrite this as

$$U_{ij}^{n+1} = U_{ij}^n - \theta(U_{ij}^n - U_{i,j-1} - k_{ij} f_i) = (1 - \theta) U_{ij}^n + \theta(U_{i,j-1} + k_{ij} f_i) \tag{3.7}$$

with

$$\theta = \frac{1}{1 - k_{ij} \frac{\partial f_i}{\partial u_i}} \tag{3.8}$$

so that we may view the simplified Newton's method as a damped version, with damping θ, of the original fixed point iteration.

The individual damping parameters are cheap to compute. We do not need to store the Jacobian and we do not need linear algebra. We still obtain some of the good properties of the full Newton's method.

For the general mcG(q) or mdG(q) method, the same analysis applies. In this case, however, when we have more degrees of freedom to solve for on every local element, $1 - k_{ij} \frac{\partial f_i}{\partial u_i}$ will be a small local matrix of size $q \times q$ for the mcG(q) method and size $(q+1) \times (q+1)$ for the mdG(q) method.

3.6. Explicit or implicit. Both mcG(q) and mdG(q) are implicit methods since they are implicitly defined by the set of equations (3.1) on each time-slab. However, depending on the solution strategy for these equations, the resulting fully discrete scheme may be of more or less explicit character. Using a diagonal Newton's method as in the current implementation of Tanganyika, we obtain a method of basically explicit nature. This gives an efficient code for many applications, but we may expect to meet difficulties for stiff problems.

3.7. The stiffness problem. In a stiff problem the solution varies quickly inside transients and slowly outside transients. For accuracy the time-steps will be adaptively kept small inside transients and then will be within the stability limits of an explicit method, while outside transients larger time-steps will be used. Outside the transients the diagonal Newton's method handles stiff problems of sufficiently diagonal nature. Otherwise the strategy is to decrease the time-steps whenever needed for stability reasons. Typically this results in an oscillating sequence of time-steps where

a small number of large time-steps are followed by a small number of stabilizing small time-steps.

Our solver Tanganyika thus performs like a modern unstable jet fighter, which needs small stabilizing wing flaps to follow a smooth trajectory. The pertinent question is then the number of small stabilizing time-steps per large time-step. We analyze this question in [3] and show that for certain classes of stiff problems it is indeed possible to successfully use a stabilized explicit method of the form implemented in Tanganyika.

3.8. Preparations. There are many "magic numbers" that need to be computed in order to implement the multi-adaptive methods, such as quadrature points and weights, the polynomial weight functions evaluated at these quadrature points, etc. In Tanganyika, these numbers are computed at the startup of the program and stored for efficient access. Although somewhat messy to compute, these are all computable by standard techniques in numerical analysis; see, e.g., [13].

3.9. Solving the dual problem. In addition to solving the primal problem (1.1), we also have to solve the continuous dual problem to obtain error control. This is an ODE in itself that we can solve using the same solver as for the primal problem.

In order to solve this ODE, we need to know the Jacobian of f evaluated at a mean value of the true solution $u(t)$ and the approximate solution $U(t)$. If $U(t)$ is sufficiently close to u, which we will assume, we approximate the (unknown) mean value by $U(t)$. When solving the dual, the primal solution must be accessible, and the Jacobian must be computed numerically by difference quotients if it is not explicitly known. This makes the computation of the dual solution expensive. Error control can, however, be obtained at a reasonable cost: for one thing, the dual problem does not have to be solved with as high a precision as the primal problem; a relative error of, say, 10% may be disastrous for the primal, whereas for the dual this only means that the error estimate will be off by 10%, which is acceptable. Second, the dual problem is linear, which may be taken into account when implementing a solver for the dual. If we can afford the linear algebra, as we can for reasonably small systems, we can solve the discrete equations directly without any iterations.

4. Tanganyika. We now give a short description of the implementation of the multi-adaptive methods, *Tanganyika*, which has been used to obtain the numerical results presented below.

4.1. Purpose. The purpose of Tanganyika [12] is to be a working implementation of the multi-adaptive methods. The code is open-source (GNU GPL [1]), which means that anyone can freely review the code, which is available at http://www.phi.chalmers.se/tanganyika/. Comments are welcome.

4.2. Structure and implementation. The solver is implemented as a C/C++ library. The C++ language makes abstraction easy, which allows the implementation to follow closely the formulation of the two methods. Different objects and algorithms are thus implemented as C++ classes, including `Solution`, `Element`, `cGqElement`, `dGqElement`, `TimeSlab`, `ErrorControl`, `Galerkin`, `Component`, and so on.

5. Applications. In this section, we present numerical results for a variety of applications. We discuss some of the problems in detail and give a brief overview of the rest. A more extensive account can be found in [11].

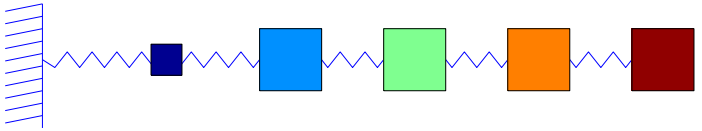

FIG. 5.1. *A mechanical system consisting of $N = 5$ masses attached with springs.*

5.1. A simple test problem. 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 accurately compute the positions (and velocities) of the N point-masses attached with springs of equal stiffness, as in Figure 5.1.

If we choose one of the masses to be much smaller than the others, $m_1 = 10^{-4}$ and $m_2 = m_3 = \cdots = m_N = 1$, then we expect the dynamics of the system to be dominated by the smallest mass, in the sense that the resolution needed to compute the solution will be completely determined by the fast oscillations of the smallest mass.

To compare the multi-adaptive method with a standard method, we first compute with constant time-steps $k = k_0$ using the standard cG(1) method and measure the error, the cpu time needed to obtain the solution, the total number of steps, i.e., $M = \sum_{i=1}^{N} M_i$, and the number of local function evaluations. We then compute the solution with individual time-steps, using the mcG(1) method, choosing the time-steps $k_i = k_0$ for the position and velocity components of the smallest mass, and choosing $k_i = 100k_0$ for the other components (knowing that the frequency of the oscillations scales like $1/\sqrt{m}$). For demonstration purposes, we thus choose the time-steps a priori to fit the dynamics of the system.

We repeat the experiment for increasing values of N (see Figure 5.2) and find that the error is about the same and constant for both methods. As N increases, the total number of time-steps, the number of local function evaluations (including also residual evaluations), and the cpu time increase linearly for the standard method, as we expect. For the multi-adaptive method, on the other hand, the total number of time-steps and local function evaluations remains practically constant as we increase N. The cpu time increases somewhat, since the increasing size of the time-slabs introduces some overhead, although not nearly as much as for the standard method. For this particular problem the gain of the multi-adaptive method is thus a factor N, where N is the size of the system, so that by considering a large-enough system, the gain is arbitrarily large.

5.2. The Lorenz system. We consider now the famous Lorenz system,

$$\left\{ \begin{array}{rcl} \dot{x} &=& \sigma(y - x), \\ \dot{y} &=& rx - y - xz, \\ \dot{z} &=& xy - bz, \end{array} \right. \tag{5.1}$$

with the usual data $(x(0), y(0), z(0)) = (1, 0, 0)$, $\sigma = 10$, $b = 8/3$, and $r = 28$; see [5]. The solution $u(t) = (x(t), y(t), z(t))$ is very sensitive to perturbations and is often described as "chaotic." With our perspective this is reflected by stability factors with rapid growth in time.

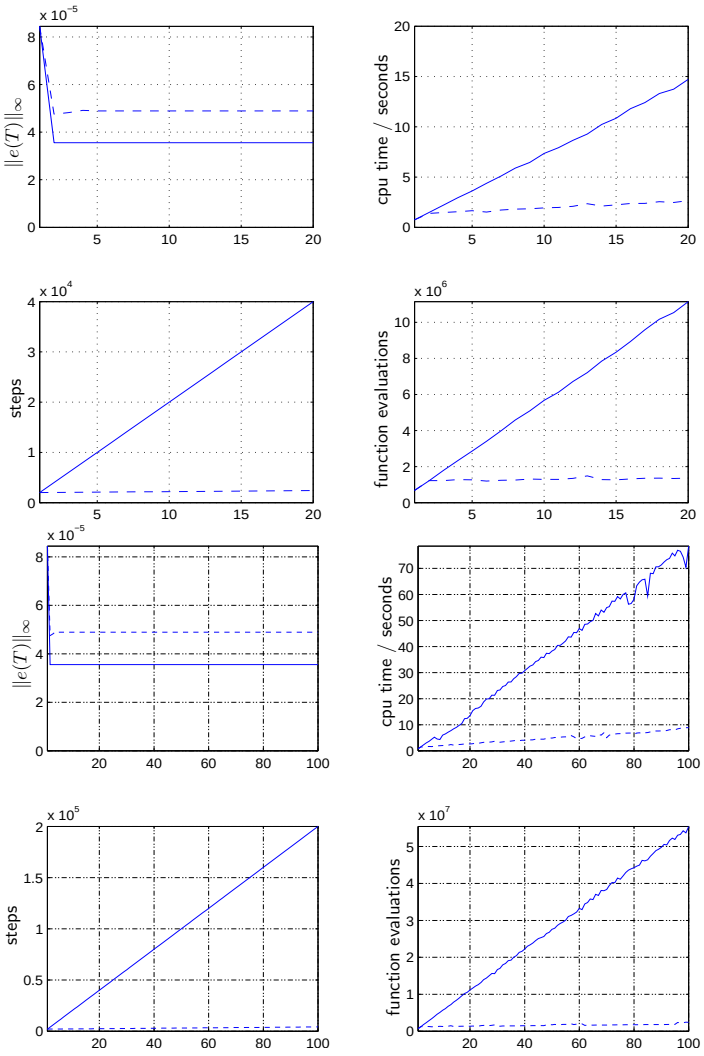

FIG. 5.2. *Error, cpu time, total number of steps, and number of function evaluations as function of the number of masses for the multi-adaptive* cG(1) *method (dashed lines) and the standard* cG(1) *method (solid lines).*

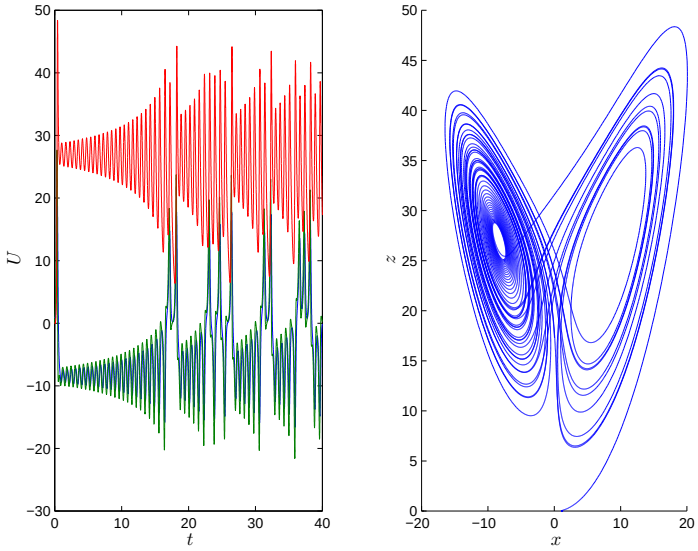

FIG. 5.3. *On the right is the trajectory of the Lorenz system for final time* $T = 40$*, computed with the multi-adaptive* cG(5) *method. On the left is a plot of the time variation of the three components.*

The computational challenge is to solve the Lorenz system accurately on a time-interval $[0, T]$ with T as large as possible. In Figure 5.3 is shown a computed solution which is accurate on the interval $[0, 40]$. We investigate the computability of the Lorenz system by solving the dual problem and computing stability factors to find the maximum value of T. The focus in this section is not on multi-adaptivity—we will use the same time-steps for all components, and so mcG(q) becomes cG(q)—but on higher-order methods and the precise information that can be obtained about the computability of a system from solving the dual problem and computing stability factors.

As an illustration, we present in Figure 5.4 solutions obtained with different methods and constant time-step $k = 0.1$ for all components. For the lower-order methods, cG(5) to cG(11), it is clear that the error decreases as we increase the order. Starting with the cG(12) method, however, the error does not decrease as we increase the order. To explain this, we note that in every time-step a small round-off error of size $\sim 10^{-16}$ is introduced if the computation is carried out in double precision arithmetic. These errors accumulate at a rate determined by the growth of the stability factor for the computational error (see [10]). As we shall see below, this stability factor grows exponentially for the Lorenz system and reaches a value of 10^{16} at final time $T = 50$, and so at this point the accumulation of round-off errors results in a large computational error.

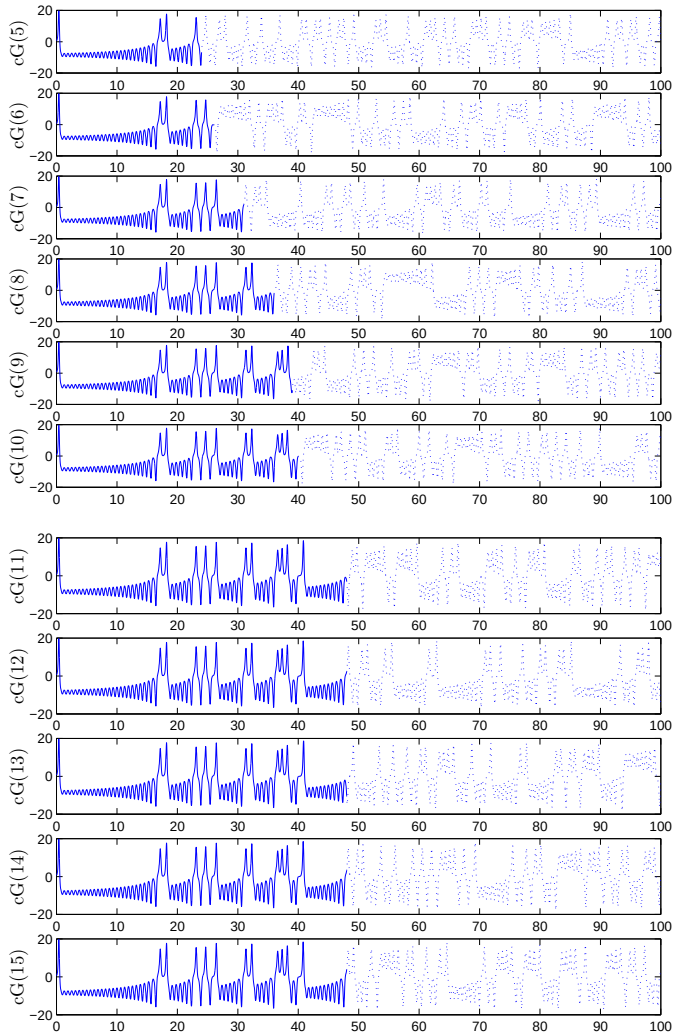

FIG. 5.4. *Solutions for the x-component of the Lorenz system with methods of different order, using a constant time-step $k = 0.1$. Dotted lines indicate the point beyond which the solution is no longer accurate.*

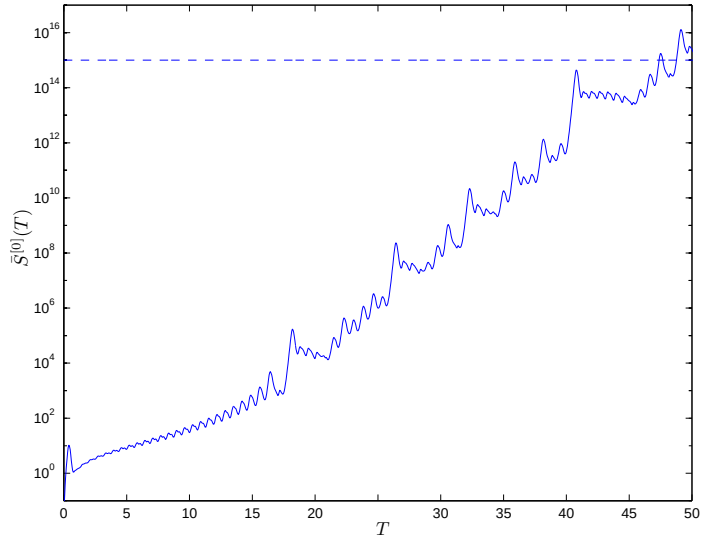

FIG. 5.5. *The stability factor for computational and quadrature errors, as function of time for the Lorenz system.*

5.2.1. Stability factors. We now investigate the computation of stability factors for the Lorenz system. For simplicity we focus on global stability factors, such as

$$S^{[q]}(T) = \max_{\|v\|=1} \int_0^T \|\varphi^{(q)}(t)\| \, dt, \tag{5.2}$$

where $\varphi(t)$ is the solution of the dual problem obtained with $\varphi(T) = v$ (and $g = 0$). Letting $\Phi(t)$ be the fundamental solution of the dual problem, we have

$$\max_{\|v\|=1} \int_0^T \|\Phi^{(q)}(t)v\| \, dt \leq \int_0^T \max_{\|v\|=1} \|\Phi^{(q)}(t)v\| \, dt = \int_0^T \|\Phi^{(q)}(t)\| \, dt = \bar{S}^{[q]}(T). \tag{5.3}$$

This gives a bound $\bar{S}^{[q]}(T)$ for $S^{[q]}(T)$, which for the Lorenz system turns out to be quite sharp and which is simpler to compute since we do not have to compute the maximum in (5.2).

In Figure 5.5 we plot the growth of the stability factor for $q = 0$, corresponding to computational and quadrature errors as described in [10]. The stability factor grows exponentially with time, but not as fast as indicated by an a priori error estimate. An a priori error estimate indicates that the stability factors grow as

$$S^{[q]}(T) \sim \mathcal{A}^q e^{\mathcal{A}T}, \tag{5.4}$$

where $\mathcal{A}$ is a bound for the Jacobian of the right-hand side for the Lorenz system. A simple estimate is $\mathcal{A} = 50$, which already at $T = 1$ gives $S^{[0]}(T) \approx 10^{22}$. In view of this, we would not be able to compute even to $T = 1$, and certainly not to $T = 50$, where we have $S^{[0]}(T) \approx 10^{1000}$. The point is that although the stability factors grow very rapidly on some occasions, such as near the first flip at $T = 18$, the growth is not monotonic. The stability factors thus *effectively* grow at a moderate exponential rate.

5.2.2. Conclusions. To predict the computability of the Lorenz system, we estimate the growth rate of the stability factors. A simple approximation of this growth rate, obtained from numerically computed solutions of the dual problem, is

$$\bar{S}^{[q]}(T) \approx 4 \cdot 10^{(q-3)+0.37T} \tag{5.5}$$

or just

$$\bar{S}^{[q]}(T) \approx 10^{q+T/3}. \tag{5.6}$$

From the a posteriori error estimates presented in [10], we find that the computational error can be estimated as

$$E_C \approx S^{[0]}(T) \max_{[0,T]} \|\mathcal{R}^{\mathcal{C}}\|, \tag{5.7}$$

where the computational residual $\mathcal{R}^{\mathcal{C}}$ is defined as

$$\mathcal{R}_i^{\mathcal{C}}(t) = \frac{1}{k_{ij}} \left(U(t_{ij}) - U(t_{i,j-1}) - \int_{I_{ij}} f_i(U, \cdot)\, dt \right). \tag{5.8}$$

With 16 digits of precision a simple estimate for the computational residual is $\frac{1}{k_{ij}} 10^{-16}$, which gives the approximation

$$E_C \approx 10^{T/3} \frac{1}{\min k_{ij}} 10^{-16} = 10^{T/3-16} \frac{1}{\min k_{ij}}. \tag{5.9}$$

With time-steps $k_{ij} = 0.1$ as above we then have

$$E_C \approx 10^{T/3-15}, \tag{5.10}$$

and so already at time $T = 45$ we have $E_C \approx 1$ and the solution is no longer accurate. We thus conclude by examination of the stability factors that it is difficult to reach beyond time $T = 50$ in double precision arithmetic. (With quadruple precision we would be able to reach time $T = 100$.)

5.3. The solar system. We now consider the solar system, including the Sun, the Moon, and the nine planets, which is a particularly important n-body problem of the form

$$m_i \ddot{x}_i = \sum_{j \neq i} \frac{G m_i m_j}{|x_j - x_i|^3} (x_j - x_i), \tag{5.11}$$

where $x_i(t) = (x_i^1(t), x_i^2(t), x_i^3(t))$ denotes the position of body i at time t, m_i is the mass of body i, and G is the gravitational constant.

As initial conditions we take the values at 00.00 Greenwich mean time on January 1, 2000, obtained from the United States Naval Observatory [2], with initial velocities

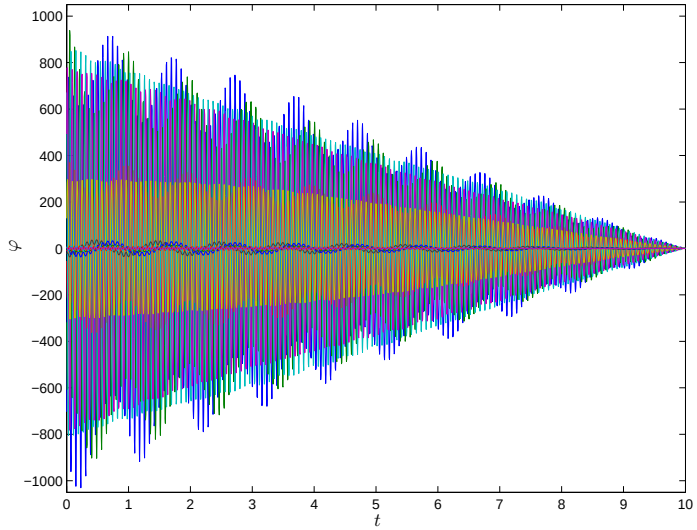

FIG. 5.6. *Part of the dual of the solar system with data chosen for control of the error in the position of the moon at final time.*

obtained by fitting a high-degree polynomial to the values of December 1999. This initial data should be correct to five or more digits, which is similar to the available precision for the masses of the planets. We normalize length and time to have the space coordinates per astronomical unit, AU, which is (approximately) the mean distance between the Sun and Earth, the time coordinates per year, and the masses per solar mass. With this normalization, the gravitational constant is $4\pi^2$.

5.3.1. Predictability. Investigating the *predictability* of the solar system, the question is how far we can accurately compute the solution, given the precision in initial data. In order to predict the accumulation rate of errors, we solve the dual problem and compute stability factors. Assuming the initial data is correct to five or more digits, we find that the solar system is computable on the order of 500 years. Including also the Moon, we cannot compute more than a few years. The dual solution grows linearly backward in time (see Figure 5.6), and so errors in initial data grow linearly with time. This means that for every extra digit of increased precision, we can reach 10 times further.

5.3.2. Computability. To touch briefly on the fundamental question of the *computability* of the solar system, concerning how far the system is computable with correct initial data and correct model, we compute the trajectories for Earth, the Moon, and the Sun over a period of 50 years, comparing different methods. Since errors in initial data grow linearly, we expect numerical errors as well as stability factors to grow quadratically.

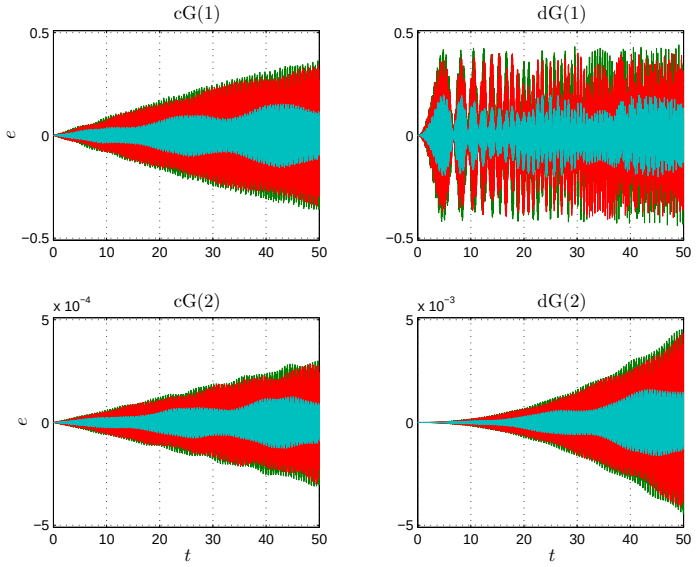

FIG. 5.7. *The growth of the error over* 50 *years for the Earth–Moon–Sun system as described in the text.*

In Figure 5.7 we plot the errors for the 18 components of the solution, computed for $k = 0.001$ with cG(1), cG(2), dG(1), and dG(2). This figure contains much information. To begin with, we see that the error seems to grow linearly for the cG methods. This is in accordance with earlier observations [4, 7] for periodic Hamiltonian systems, recalling that the (m)cG(q) methods conserve energy [10]. The stability factors, however, grow quadratically and thus overestimate the error growth for this particular problem. In an attempt to give an intuitive explanation of the linear growth, we may think of the error introduced at every time-step by an energy-conserving method as a pure phase error, and so at every time-step the Moon is pushed slightly forward along its trajectory (with the velocity adjusted accordingly). Since a pure phase error does not accumulate but stays constant (for a circular orbit), the many small phase errors give a total error that grows linearly with time.

Examining the solutions obtained with the dG(1) and dG(2) methods, we see that the error grows quadratically, as we expect. For the dG(1) solution, the error reaches a maximum level of ~ 0.5 for the velocity components of the Moon. The error in position for the Moon is much smaller. This means that the Moon is still in orbit around Earth, the position of which is still very accurate, but the position relative to Earth, and thus also the velocity, is incorrect. The error thus grows quadratically until it reaches a limit. This effect is also visible for the error of the cG(1) solution; the linear growth flattens out as the error reaches the limit. Notice also that even if

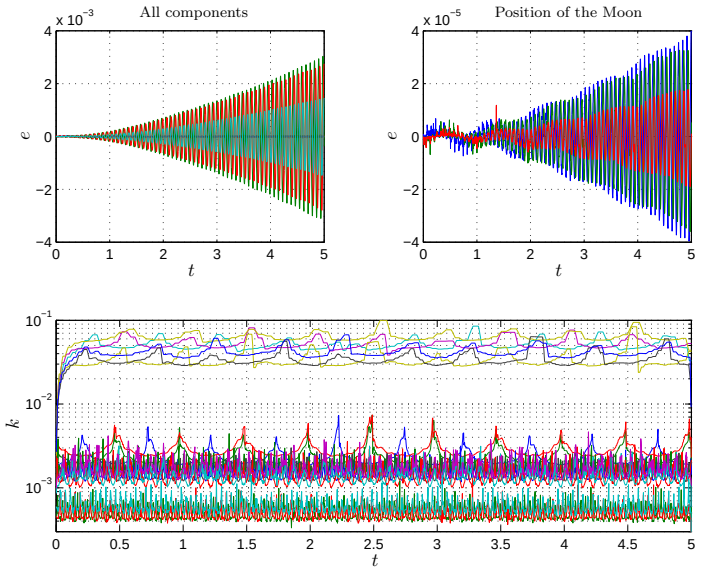

FIG. 5.8. *The growth of the error over* 5 *years for the Earth–Moon–Sun system computed with the* mcG(2) *method, together with the multi-adaptive time-steps.*

the higher-order dG(2) method performs better than the cG(1) method on a short time-interval, it will be outrun on a long enough interval by the cG(1) method, which has linear accumulation of errors (for this particular problem).

5.3.3. Multi-adaptive time-steps. Solving with the multi-adaptive method mcG(2) (see Figure 5.8), the error grows quadratically. We saw in [10] that in order for the mcG(q) method to conserve energy, we require that corresponding position and velocity components use the same time-steps. Computing with different time-steps for all components, as here, we thus cannot expect to have linear error growth. Keeping $k_i^2 r_i \leq \text{tol}$ with $\text{tol} = 10^{-10}$ as here, the error grows as $10^{-4}T^2$ and we are able to reach $T \sim 100$. Decreasing tol to, say, 10^{-18}, we could instead reach $T \sim 10^6$.

We investigated the passing of a large comet close to Earth and the Moon and found that the stability factors increase dramatically at the time t' when the comet comes very close to the Moon. The conclusion is that if we want to compute accurately to a point beyond t', we have to be much more careful than if we want to compute to a point just before t'. This is not evident from the size of residuals or local errors. This is an example of a Hamiltonian system for which the error growth is neither linear nor quadratic.

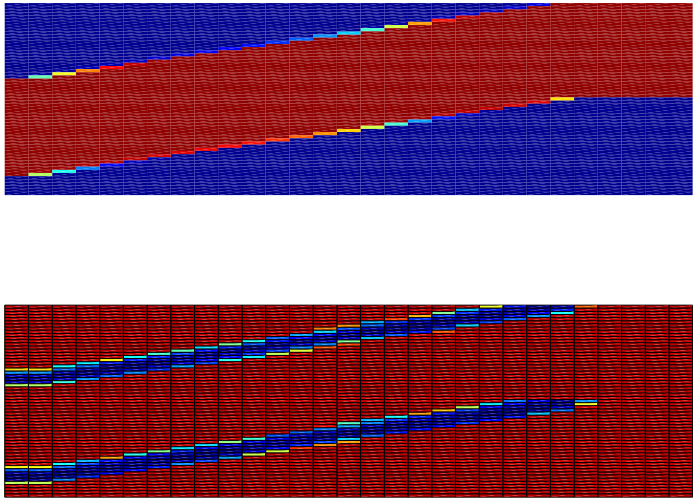

FIG. 5.9. *A space-time plot of the solution (above) and time-steps (below) for the propagating front problem, with time going to the right. The two parts of the plots represent the components for the two species A_1 (lower parts) and A_2 (upper parts).*

5.4. A propagating front problem. The system of PDEs

$$\left\{ \begin{array}{rcl} \dot{u}_1 - \epsilon u_1'' & = & -u_1 u_2^2, \\ \dot{u}_2 - \epsilon u_2'' & = & u_1 u_2^2 \end{array} \right. \tag{5.12}$$

on $(0,1) \times (0,T]$ with $\epsilon = 0.00001$, $T = 100$, and homogeneous Neumann boundary conditions at $x = 0$ and $x = 1$ models isothermal autocatalytic reactions (see [14]) $A_1 + 2A_2 \rightarrow A_2 + 2A_2$. We choose the initial conditions as

$$u_1(x,0) = \left\{ \begin{array}{ll} 0, & x < x_0, \\ 1, & x \geq x_0, \end{array} \right.$$

with $x_0 = 0.2$ and $u_2(x,0) = 1 - u_1(x,0)$. An initial reaction where substance A_1 is consumed and substance A_2 is formed will then occur 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.

Solving with the mcG(2) method, we find that the time-steps are small only close to the reaction front; see Figures 5.9 and 5.10. The reaction front propagates to the right as does the domain of small time-steps.

It is clear that if the reaction front is localized in space and the domain is large, there is a lot to gain by using small time-steps only in this area. To verify this, we

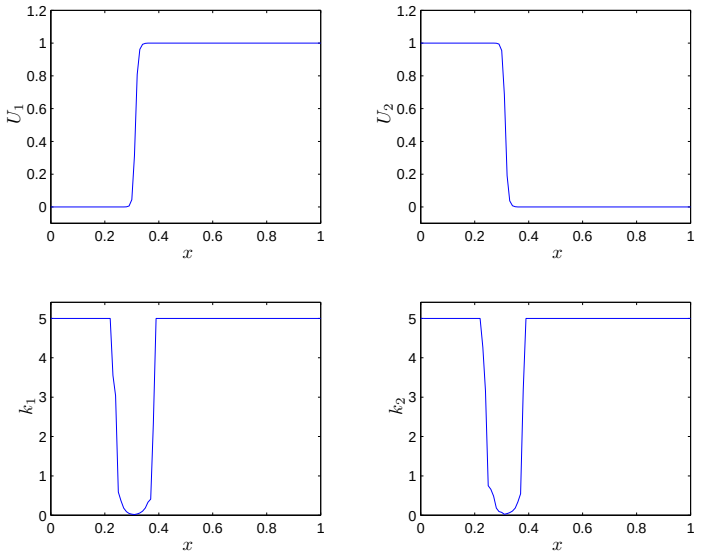

FIG. 5.10. *The concentrations of the two species U_1 and U_2 at time $t = 50$ as function of space (above) and the corresponding time-steps (below).*

compute the solution to within an accuracy of 10^{-7} for the final time error with constant time-steps $k_i(t) = k_0$ for all components and compare with the multi-adaptive solution. Computing on a space grid consisting of 16 nodes on $[0,1]$ (resulting in a system of ODEs with 32 components), the solution is computed in 2.1 seconds on a regular workstation. Computing on the same grid with the multi-adaptive method (to within the same accuracy), we find that the solution is computed in 3.0 seconds. More work is thus required to compute the multi-adaptive solution, and the reason is the overhead resulting from additional bookkeeping and interpolation in the multi-adaptive computation. However, increasing the size of the domain to 32 nodes on $[0,2]$ and keeping the same parameters otherwise, we find the solution is now more localized in the domain and we expect the multi-adaptive method to perform better compared to a standard method. Indeed, the computation using equal time-steps now takes 5.7 seconds, whereas the multi-adaptive solution is computed in 3.4 seconds. In the same way as previously shown in section 5.1, adding extra degrees of freedom does not substantially increase the cost of solving the problem, since the main work is done time-stepping the components, which use small time-steps.

5.5. Burger's equation with moving nodes. As a final example, we present a computation in which we combine multi-adaptivity with the possibility of moving the nodes in a space discretization of a time-dependent PDE. Solving Burger's equation,

$$\dot{u} + \mu u u' - \epsilon u'' = 0, \tag{5.13}$$

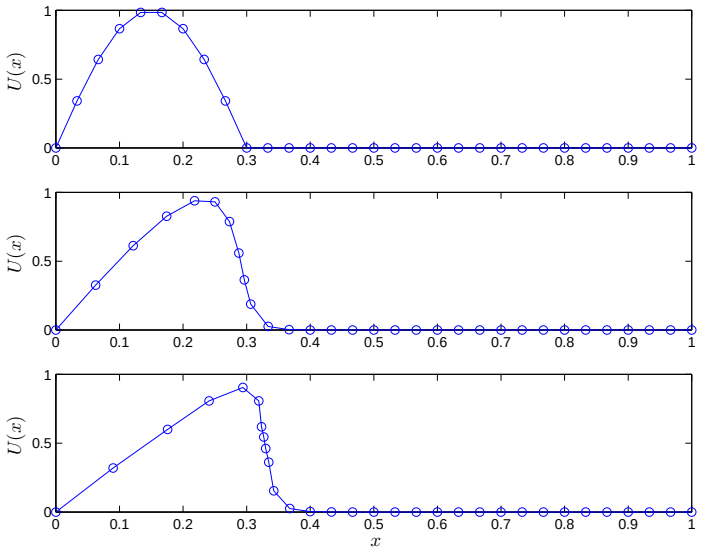

FIG. 5.11. *The solution to Burger's equation as function of space at $t = 0$, $t = T/2$, and $t = T$.*

on $(0, 1) \times (0, T]$ with initial condition

$$u_0(x) = \begin{cases} \sin(\pi x/x_0), & 0 \le x \le x_0, \\ 0 & \text{elsewhere,} \end{cases} \tag{5.14}$$

and with $\mu = 0.1$, $\epsilon = 0.001$, and $x_0 = 0.3$, we find the solution is a shock forming near $x = x_0$. Allowing individual time-steps within the domain, and moving the nodes of the space discretization in the direction of the convection, $(1, \mu u)$, we make the ansatz

$$U(x, t) = \sum_{i=1}^{N} \xi_i(t)\varphi_i(x, t), \tag{5.15}$$

where the $\{\xi_i\}_{i=1}^N$ are the individual components computed with the multi-adaptive method, and the $\{\varphi_i(\cdot, t)\}_{i=1}^N$ are piecewise linear basis functions in space for any fixed t.

Solving with the mdG(0) method, the nodes move into the shock, in the direction of the convection, so what we are really solving is a heat equation with multi-adaptive time-steps along the streamlines; see Figures 5.11 and 5.12.

6. Future work. Together with the companion paper [10] (see also [9, 8]), this paper serves as a starting point for further investigation of the multi-adaptive Galerkin methods and their properties.

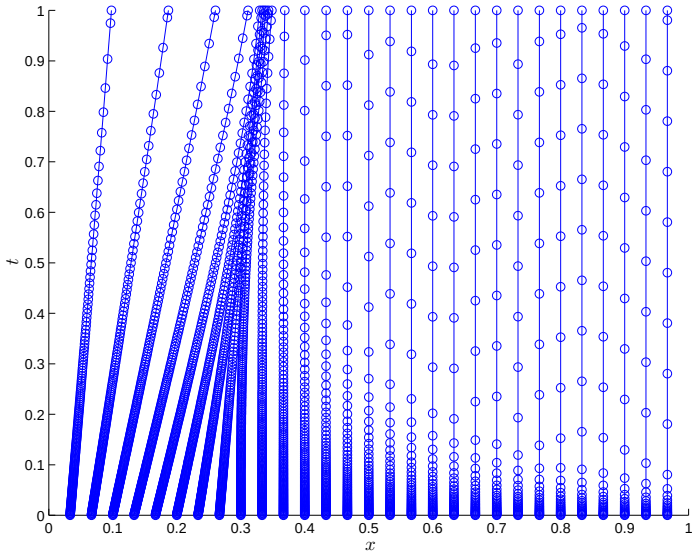

FIG. 5.12. *Node paths for the multi-adaptive moving-nodes solution of Burger's equation.*

Future work will include a more thorough investigation of the application of the multi-adaptive methods to stiff ODEs, as well as the construction of efficient multi-adaptive solvers for time-dependent PDEs, for which memory usage becomes an important issue.

REFERENCES

[1] *GNU Project Web Server*, http://www.gnu.org/.
[2] *USNO Astronomical Applications Department*, http://aa.usno.navy.mil/.
[3] K. ERIKSSON, C. JOHNSON, AND A. LOGG, *Explicit time-stepping for stiff ODEs*, SIAM J. Sci. Comput., 25 (2003), pp. 1142–1157.
[4] D. ESTEP, *The rate of error growth in Hamiltonian-conserving integrators*, Z. Angew. Math. Phys., 46 (1995), pp. 407–418.
[5] D. ESTEP AND C. JOHNSON, *The computability of the Lorenz system*, Math. Models Methods Appl. Sci., 8 (1998), pp. 1277–1305.
[6] 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.
[7] M. G. LARSON, *Error growth and a posteriori error estimates for conservative Galerkin approximations of periodic orbits in Hamiltonian systems*, Math. Models Methods Appl. Sci., 10 (2000), pp. 31–46.
[8] A. LOGG, *Multi-Adaptive Error Control for ODEs*, Preprint 2000-03, Chalmers Finite Element Center, Chalmers University of Technology, Göteborg, Sweden, 2000. Also available online from http://www.phi.chalmers.se/preprints/.
[9] A. LOGG, *A Multi-Adaptive ODE-Solver*, Preprint 2000-02, Chalmers Finite Element Center, Chalmers University of Technology, Göteborg, Sweden, 2000. Also available online from http://www.phi.chalmers.se/preprints/.

[10] A. Logg, *Multi-adaptive Galerkin methods for ODEs* I, SIAM J. Sci. Comput., 24 (2003), pp. 1879–1902.
[11] A. Logg, *Multi-Adaptive Galerkin Methods for ODEs* II: *Applications*, Preprint 2001-10, Chalmers Finite Element Center, Chalmers University of Technology, Göteborg, Sweden, 2001. Also available online from http://www.phi.chalmers.se/preprints/.
[12] A. Logg, *Tanganyika, Version* 1.2, 2001. http://www.phi.chalmers.se/tanganyika/.
[13] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C. The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, UK, 1992. Also available online from http://www.nr.com.
[14] R. Sandboge, *Adaptive Finite Element Methods for Reactive Flow Problems*, Ph.D. thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, Sweden, 1996.
[15] G. Söderlind, *The automatic control of numerical integration. Solving differential equations on parallel computers*, CWI Quarterly, 11 (1998), pp. 55–74.

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES III: EXISTENCE AND STABILITY

ANDERS LOGG

Abstract. We prove existence and stability of solutions for the multi-adaptive Galerkin methods mcG(q) and mdG(q), and their dual versions mcG(q)* and mdG(q)*, including strong stability estimates for parabolic problems. This paper is the third in a series devoted to multi-adaptive Galerkin methods. In the companion paper [7], we return to the a priori error analysis of the multi-adaptive methods. The stability estimates derived in this paper will then be essential.

1. Introduction

This is part III in a sequence of papers [4, 5] 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 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.

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 this paper, we prove existence and stability of discrete solutions, which we later use together with special interpolation estimates to prove a priori error estimates for the mcG(q) and mdG(q) methods in part IV [7].

Date: March 15, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, a priori error estimates, dual methods, existence, strong stability, parabolic.

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

1.1. **Notation.** For a detailed presentation of the multi-adaptive methods, we refer to [4, 5]. Here, we only give a quick overview of the notation: 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*:

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

We denote the length of a time slab by $K_n = T_n - T_{n-1}$. We also refer to the entire collection of intervals I_{ij} as the partition $\mathcal{T}$.

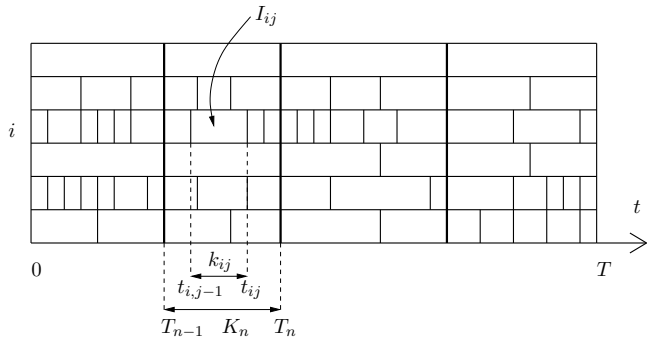

Figure 1. Individual partitions of the interval $(0,T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

1.2. **Outline of the paper.** The first part of this paper is devoted to proving existence of solutions for the multi-adaptive methods mcG(q) and mdG(q), including the dual methods mcG(q)* and mdG(q)* obtained by interchanging trial and test spaces, by proving (relevant) fixed point iterations assuming the time steps are sufficiently small. The proof is constructive and mimics the actual implementation of the methods. The multi-adaptive ODE-solver *Tanganyika*, presented in [5], thus repeats the proof of existence each time it computes a new solution.

In the second part of this paper, we prove stability estimates, including general exponential estimates for mcG(q), mdG(q), mcG(q)*, and mdG(q)*, and strong stability estimates for parabolic problems for the mdG(q) and mdG(q)* methods.

2. Multi-adaptive Galerkin and multi-adaptive dual Galerkin

2.1. **Multi-adaptive continuous Galerkin,** mcG(q)**.** To formulate the mcG(q) method, we define the *trial space* V and the *test space* $\hat{V}$ as

$$V = \{v \in [\mathcal{C}([0,T])]^N : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}),\ j = 1,\ldots,M_i,\ i = 1,\ldots,N\}, \tag{2.1}$$
$$\hat{V} = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}-1}(I_{ij}),\ j = 1,\ldots,M_i,\ i = 1,\ldots,N\},$$

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

We now define the mcG(q) method for (1.1) in the following way: Find $U \in V$ with $U(0) = u_0$, such that

$$\int_0^T (\dot{U}, v)\,dt = \int_0^T (f(U,\cdot), v)\,dt \quad \forall v \in \hat{V}, \tag{2.2}$$

where $(\cdot,\cdot)$ denotes the $\mathbb{R}^N$ inner product. If now for each local interval I_{ij} we take $v_n \equiv 0$ when $n \neq i$ and $v_i(t) = 0$ when $t \notin I_{ij}$, we can rewrite the global problem (2.2) as a sequence 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, such that

$$\int_{I_{ij}} \dot{U}_i v\,dt = \int_{I_{ij}} f_i(U,\cdot)v\,dt \quad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \tag{2.3}$$

where the initial condition is specified for $i = 1,\ldots,N$ by $U_i(0) = u_i(0)$.

We define the *residual* R of the approximate solution U by $R_i(U,t) = \dot{U}_i(t) - f_i(U(t),t)$. In terms of the residual, we can rewrite (2.3) as

$$\int_{I_{ij}} R_i(U,\cdot)v\,dt = 0 \quad \forall v \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \tag{2.4}$$

that is, the residual is orthogonal to the test space on each local interval. We refer to (2.4) 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

$$V = \hat{V} = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}),\ j = 1,\ldots,M_i,\ i = 1,\ldots,N\}, \tag{2.5}$$

that is, both trial and test functions are (possibly discontinuous) piecewise polynomials of degree $q = q_i(t) = q_{ij} \geq 0,\ t \in I_{ij}$, on the partition $\mathcal{T}$. We define the mdG(q) solution $U \in V$ to be left-continuous.

We now define the mdG(q) method for (1.1) in the following way, similar to the definition of the continuous method: Find $U \in V$ with $U(0^-) = u_0$, such that

$$\sum_{i=1}^N \sum_{j=1}^{M_i} \left[[U_i]_{i,j-1} v_i(t_{i,j-1}^+) + \int_{I_{ij}} \dot{U}_i v_i\,dt \right] = \int_0^T (f(U,\cdot), v)\,dt \quad \forall v \in \hat{V}, \tag{2.6}$$

where $[U_i]_{i,j-1} = U_i(t_{i,j-1}^+) - U_i(t_{i,j-1}^-)$ denotes the jump in $U_i(t)$ across the node $t = t_{i,j-1}$.

The mdG(q) method in local form, corresponding to (2.3), 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

$$[U_i]_{i,j-1} v(t_{i,j-1}) + \int_{I_{ij}} \dot{U}_i v\,dt = \int_{I_{ij}} f_i(U,\cdot)v\,dt \quad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}), \tag{2.7}$$

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 by $R_i(U,t) = \dot{U}_i(t) - f_i(U(t),t)$, defined on the inner of each local interval I_{ij}, and rewrite (2.7) in the form

$$[U_i]_{i,j-1} v(t_{i,j-1}^+) + \int_{I_{ij}} R_i(U,\cdot)v\,dt = 0 \quad \forall v \in \mathcal{P}^{q_{ij}}(I_{ij}). \tag{2.8}$$

We refer to (2.8) as the Galerkin orthogonality of the mdG(q) method. Note that the residual has two parts: one interior part R_i and the jump term $[U_i]_{i,j-1}$.

2.3. **The dual problem.** The motivation for introducing the dual problem is for the a priori or a posteriori error analysis of the multi-adaptive methods. For the a posteriori analysis, we formulate a continuous dual problem [4]. For the a priori analysis [7], we formulate a discrete dual problem in terms of the dual multi-adaptive methods mcG(q)* and mdG(q)*.

The discrete dual solution $\Phi : [0,T] \to \mathbb{R}^N$ is a Galerkin approximation of the exact solution $\phi : [0,T] \to \mathbb{R}^N$ of the continuous dual backward problem

$$\begin{aligned} -\dot{\phi}(t) &= J^\top(\pi u, U, t)\phi(t) + g(t), \quad t \in [0,T), \\ \phi(T) &= \psi, \end{aligned} \tag{2.9}$$

where πu is an interpolant or a projection of the exact solution u of (1.1), $g : [0,T] \to \mathbb{R}^N$ is a given function, $\psi \in \mathbb{R}^N$ is a given initial condition, and

$$J^\top(\pi u, U, t) = \left(\int_0^1 \frac{\partial f}{\partial u}(s\pi u(t) + (1-s)U(t), t)\,ds \right)^\top, \tag{2.10}$$

that is, an appropriate mean value of the transpose of the Jacobian of the right-hand side $f(\cdot,t)$ evaluated at $\pi u(t)$ and $U(t)$. Note that by the chain

rule, we have

$$J(\pi u, U, \cdot)(U - \pi u) = f(U, \cdot) - f(\pi u, \cdot). \tag{2.11}$$

2.4. **Multi-adaptive dual continuous Galerkin,** mcG$(q)^*$**.** In the formulation of the dual method of mcG(q), we interchange the trial and test spaces of mcG(q). With the same definitions of V and $\hat{V}$ as in (2.1), we thus define the mcG$(q)^*$ method for (2.9) in the following way: Find $\Phi \in \hat{V}$ with $\Phi(T^+) = \psi$, such that

$$\int_0^T (\dot{v}, \Phi)\, dt = \int_0^T (J(\pi u, U, \cdot)v, \Phi) + L_{\psi,g}(v), \tag{2.12}$$

for all $v \in V$ with $v(0) = 0$, where

$$L_{\psi,g}(v) \equiv (v(T), \psi) + \int_0^T (v, g)\, dt. \tag{2.13}$$

Notice the extra condition that the test functions should vanish at $t = 0$, which is introduced to make the dimension of the test space equal to the dimension of the trial space. Integrating by parts, (2.12) can alternatively be expressed in the form

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \left[-[\Phi_i]_{ij} v_i(t_{ij}) - \int_{I_{ij}} \dot{\Phi}_i v_i \, dt \right] = \int_0^T (J^\top(\pi u, U, \cdot)\Phi + g, v)\, dt. \tag{2.14}$$

2.5. **Multi-adaptive dual discontinuous Galerkin,** mdG$(q)^*$**.** For the discontinuous method, we note that the trial and test spaces are identical. With the same definitions of V and $\hat{V}$ as in (2.5), we define the mdG$(q)^*$ method for (2.9) in the following way: Find $\Phi \in \hat{V}$ with $\Phi(T^+) = \psi$, such that

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \left[[v_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} \dot{v}_i \Phi_i \, dt \right] = \int_0^T (J(\pi u, U, \cdot)v, \Phi)\, dt + L_{\psi,g}(v), \tag{2.15}$$

for all $v \in V$ with $v(0^-) = 0$. Integrating by parts, (2.15) can alternatively be expressed in the form

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \left[-[\Phi_i]_{ij} v_i(t_{ij}^-) - \int_{I_{ij}} \dot{\Phi}_i v_i \, dt \right] = \int_0^T (J^\top(\pi u, U, \cdot)\Phi + g, v)\, dt. \tag{2.16}$$

3. Existence of solutions

To prove existence of the discrete mcG(q), mdG(q), mcG$(q)^*$, and mdG$(q)^*$ solutions defined in the previous section, we formulate fixed point iterations for the construction of solutions. Existence then follows from the Banach fixed point theorem, if the time steps are sufficiently small. The proof is thus constructive and gives a method for computing solutions (see [5]).

3.1. **Multi-adaptive Galerkin in fixed point form.** We start by proving the following simple lemma.

Lemma 3.1. *Let A be a $d \times d$ matrix with elements $A_{mn} = \frac{n}{m+n-1}$, and let B be a $d \times d$ matrix with elements $B_{mn} = \frac{n}{m+n}$, for $m, n = 1, \ldots, d$. Then,* $\det A \neq 0$ *and* $\det B \neq 0$.

Proof. To prove that A is nonsingular, we let $p(t) = \sum_{n=1}^{d} x_n n t^{n-1}$ be a polynomial of degree $d - 1$ on $[0, 1]$. If for $m = 1, \ldots, d$, $\int_0^1 p(t) t^{m-1}\, dt = 0$, it follows that $p \equiv 0$. Thus, $\sum_{n=1}^{d} x_n \frac{n}{m+n-1} = 0$ for $m = 1, \ldots, d$ implies $x = 0$, which proves that $\det A \neq 0$. To prove that B is nonsingular, let again $p(t) = \sum_{n=1}^{d} x_n n t^{n-1}$. If for $m = 1, \ldots, d$ we have $\int_0^1 p(t) t^m\, dt = 0$, take $q(t) = \sum_{m=1}^{d} y_m t^m$, such that p and q have the same zeros on $[0, 1]$ and $pq \geq 0$. Then, $\int_0^1 pq\, dt = 0$ but $pq \geq 0$ on $[0, 1]$ and so $p \equiv 0$. Thus, $\sum_{n=1}^{n} x_n \frac{n}{m+n} = 0$ for $m = 1, \ldots, d$ implies $x = 0$, which proves that $\det B \neq 0$. □

To rewrite the methods in explicit fixed point form, we introduce a simple basis for the trial and test spaces and solve for the degrees of freedom on each local interval.

Lemma 3.2. *The* mcG(q) *method for* (1.1) *in fixed point form reads: For $i = 1, \ldots, N$, $j = 1, \ldots, M_i$, find $\{\xi_{ijn}\}_{n=1}^{q_{ij}}$, such that*

$$\xi_{ijn} = \xi_{ij0} + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i(U(t), t)\, dt, \tag{3.1}$$

with

$$\xi_{ij0} = \begin{cases} \xi_{i,j-1,q_{i,j-1}}, & j > 1, \\ u_i(0), & j = 1, \end{cases}$$

where $\{w_n^{[q_{ij}]}\}_{n=1}^{q_{ij}} \subset \mathcal{P}^{q_{ij}-1}([0, 1])$, $w_{q_{ij}}^{[q_{ij}]} \equiv 1$, and $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. A component $U_i(t)$ of the solution is given on I_{ij} by

$$U_i(t) = \sum_{n=0}^{q_{ij}} \xi_{ijn} \lambda_n^{[q_{ij}]}(\tau_{ij}(t)),$$

where $\{\lambda_n^{[q_{ij}]}\}_{n=0}^{q_{ij}} \subset \mathcal{P}^{q_{ij}}([0, 1])$ is the standard Lagrange basis on $[0, 1]$ with $t = 0$ and $t = 1$ as two of its $q_{ij} + 1 \geq 2$ nodal points.

Proof. Our starting point is the local formulation (2.3). Dropping indices for ease of notation and rescaling to the interval $[0, 1]$, we have

$$\int_0^1 \dot{U} v\, dt = \int_0^1 f v\, dt \quad \forall v \in \mathcal{P}^{q-1}([0, 1]),$$

with $U \in \mathcal{P}^q([0,1])$. Let now $\{\lambda_n^{[q]}\}_{n=0}^q$ be a basis for $\mathcal{P}^q([0,1])$. In terms of this basis, we have $U(t) = \sum_{n=0}^q \xi_n \lambda_n^{[q]}(t)$, and so

$$\sum_{n=0}^{q} \xi_n \int_0^1 \dot{\lambda}_n^{[q]} \lambda_m^{[q-1]} \, dt = \int_0^1 f \lambda_m^{[q-1]} \, dt, \quad m = 0, \ldots, q-1.$$

Since the solution is continuous, the value at $t = 0$ is known from the previous interval (or from the initial condition). This gives

$$U(0) = \sum_{n=0}^{q} \xi_n \lambda_n^{[q]}(0) = \xi_0,$$

if we assume that $\lambda_n^{[q]}(0) = \delta_{0n}$. The remaining q degrees of freedom are then determined by

$$\sum_{n=1}^{q} \xi_n \int_0^1 \dot{\lambda}_n^{[q]} \lambda_{m-1}^{[q-1]} \, dt = \int_0^1 f \lambda_{m-1}^{[q-1]} - \xi_0 \int_0^1 \dot{\lambda}_0^{[q]} \lambda_{m-1}^{[q-1]} \, dt, \quad m = 1, \ldots, q.$$

If $\det\left(\int_0^1 \dot{\lambda}_n^{[q]} \lambda_{m-1}^{[q-1]} \, dt\right) \neq 0$, this system can be solved for the degrees of freedom $(\xi_1, \ldots, \xi_n)$. With $\lambda_n^{[q]}(t) = t^n$, we have

$$\det\left(\int_0^1 \dot{\lambda}_n^{[q]} \lambda_{m-1}^{[q-1]} \, dt\right) = \det\left(\int_0^1 n t^{n-1} t^{m-1} \, dt\right) = \det\left(\frac{n}{m+n-1}\right) \neq 0,$$

by Lemma 3.1. Solving for $(\xi_1, \ldots, \xi_n)$, we obtain

$$\xi_n = \alpha_n^{[q]} \xi_0 + \int_0^1 w_n^{[q]} f \, dt, \quad n = 1, \ldots, q,$$

for some constants $\{\alpha_n^{[q]}\}_{n=1}^q$, where $\{w_n^{[q]}\}_{n=1}^q \subset \mathcal{P}^{q-1}([0,1])$ and ξ_0 is determined from the continuity requirement. For any other basis $\{\lambda_n^{[q]}\}_{n=0}^q$ with $\lambda_n^{[q]}(0) = \delta_{0n}$, we obtain a similar expression for the degrees of freedom by a linear transformation. In particular, let $\{\lambda_n^{[q]}\}_{n=0}^q$ be the Lagrange basis functions for a partition of $[0,1]$ with $t = 0$ as a nodal point. For $f \equiv 0$ it is easy to see that the mcG(q) solution is constant and equal to its initial value. It follows that $\alpha_n^{[q]} = 1$, $n = 1, \ldots, q$, and so

$$\xi_n = \xi_0 + \int_0^1 w_n^{[q]} f \, dt, \quad n = 1, \ldots, q,$$

with $U(1) = \xi_q$ if also $t = 1$ is a nodal point. To see that $w_q^{[q]} \equiv 1$, take $v \equiv 1$ in (2.3). The result now follows by rescaling to I_{ij}. □

Lemma 3.3. *The* mdG(q) *method for* (1.1) *in fixed point form reads: For* $i = 1, \ldots, N$, $j = 1, \ldots, M_i$, *find* $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, *such that*

$$\xi_{ijn} = \xi_{ij0}^- + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i(U(t), t) \, dt, \tag{3.2}$$

with

$$\xi_{ij0}^- = \begin{cases} \xi_{i,j-1,q_{i,j-1}}, & j > 1, \\ u_i(0), & j = 1, \end{cases}$$

where $\{w_n^{[q_{ij}]}\}_{n=0}^{q_{ij}} \subset \mathcal{P}^{q_{ij}}([0,1])$, $w_{q_{ij}}^{[q_{ij}]} \equiv 1$, *and* $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. *A component* $U_i(t)$ *of the solution is given on* I_{ij} *by*

$$U_i(t) = \sum_{n=0}^{q_{ij}} \xi_{ijn} \lambda_n^{[q_{ij}]}(\tau_{ij}(t)),$$

where $\{\lambda_n^{[q_{ij}]}\}_{n=0}^{q_{ij}} \subset \mathcal{P}^{q_{ij}}([0,1])$ *is the standard Lagrange basis on* $[0,1]$ *with* $t = 1$ *as one of its* $q_{ij} + 1 \geq 1$ *nodal points.*

Proof. In as similar way as in the proof of Lemma 3.2, we use (2.7) to obtain

$$(\xi_0 - \xi_0^-)\lambda_m^{[q]}(0) + \sum_{n=0}^{q} \xi_n \int_0^1 \dot{\lambda}_n^{[q]} \lambda_m^{[q]} \, dt = \int_0^1 f \lambda_m^{[q]} \, dt, \quad m = 0, \ldots, q.$$

With $\lambda_n^{[q]}(t) = t^n$, these $1 + q$ equations can be written in the form

$$\begin{aligned} \xi_0 + \sum_{n=1}^{q} \xi_n \int_0^1 n t^{n-1} \, dt &= \int_0^1 f \, dt + \xi_0^-, \\ \sum_{n=1}^{q} \xi_n \int_0^1 n t^{n-1} t^m \, dt &= \int_0^1 f t^m \, dt, \quad m = 1, \ldots, q, \end{aligned}$$

which by Lemma 3.1 has a solution, since

$$\det\left(\int_0^1 n t^{n-1} t^m \, dt\right) = \det\left(\frac{n}{m+n}\right) \neq 0.$$

We thus obtain

$$\xi_n = \alpha_n^{[q]} \xi_0^- + \int_0^1 w_n^{[q]} f \, dt, \quad n = 0, \ldots, q.$$

By the same argument as in the proof of Lemma 3.2, we conclude that when $\{\lambda_n^{[q]}\}_{n=0}^q$ is the Lagrange basis for a partition of $[0,1]$, we have

$$\xi_n = \xi_0^- + \int_0^1 w_n^{[q]} f \, dt, \quad n = 0, \ldots, q,$$

with $U(1) = \xi_q = \xi_0^- + \int_0^1 f \, dt$ if $t = 1$ is a nodal point. The result now follows by rescaling to I_{ij}. □

Lemma 3.4. *The* mcG(q)* *method for* (2.9) *in fixed point form reads: For* $i = 1, \ldots, N$, $j = M_i, \ldots, 1$, *find* $\{\xi_{ijn}\}_{n=0}^{q_{ij}-1}$, *such that*

$$\xi_{ijn} = \psi_i + \int_{t_{ij}}^{T} f_i^*(\Phi, \cdot) \, dt + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i^*(\Phi(t), t) \, dt, \tag{3.3}$$

where $f^*(\Phi,\cdot) = J^\top(\pi u, U, \cdot)\Phi + g$, $\{w_n^{[q_{ij}]}\}_{n=0}^{q_{ij}-1} \subset \mathcal{P}^{q_{ij}}([0,1])$, $w_n^{[q_{ij}]}(0) = 0$, $w_n^{[q_{ij}]}(1) = 1$, $n = 0,\ldots,q_{ij}-1$, *and* $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. *A component* $\Phi_i(t)$ *of the solution is given on* I_{ij} *by*

$$\Phi_i(t) = \sum_{n=0}^{q_{ij}-1} \xi_{ijn} \lambda_n^{[q_{ij}-1]}(\tau_{ij}(t)),$$

where $\{\lambda_n^{[q_{ij}-1]}\}_{n=0}^{q_{ij}-1} \subset \mathcal{P}^{q_{ij}-1}([0,1])$ *is the standard Lagrange basis on* $[0,1]$.

Proof. Our starting point is the definition (2.14). For any $1 \le i \le N$, take $v_n \equiv 0$ when $n \neq i$ and let v_i be a continuous piecewise polynomial that vanishes on $[0, t_{i,j-1}]$ with $v_i \equiv 1$ on $[t_{ij}, T]$, see Figure 3.4. With $f^*(\Phi,\cdot) = J^\top(\pi u, U, \cdot)\Phi + g$, we then have

$$\sum_{l=j}^{M_i} \left[-[\Phi_i]_{il} v_i(t_{il}) - \int_{I_{il}} \dot\Phi_i v_i \, dt \right] = \int_{t_{i,j-1}}^{T} f_i^*(\Phi,\cdot) v_i \, dt.$$

We integrate by parts, moving the derivative onto the test function, to get

$$\begin{aligned} -[\Phi_i]_{il} v_i(t_{il}) - \int_{I_{il}} \dot\Phi_i v_i \, dt &= -[\Phi_i]_{il} v_i(t_{il}) - [\Phi_i v_i]_{t_{i,l-1}^+}^{t_{il}^-} + \int_{I_{il}} \Phi_i \dot v_i \, dt \\ &= \Phi_i(t_{i,l-1}^+) v_i(t_{i,l-1}) - \Phi_i(t_{il}^+) v_i(t_{il}) + \int_{I_{il}} \Phi_i \dot v_i \, dt. \end{aligned}$$

Summing up, noting that $v_i(t_{i,j-1}) = 0$, $\dot v_i = 0$ on $[t_{ij}, T]$ and $\Phi_i(t_{M_i}^+) = \psi_i$, we have

$$-\psi_i + \int_{I_{ij}} \Phi_i \dot v_i \, dt = \int_{t_{i,j-1}}^{T} f_i^*(\Phi,\cdot) v_i \, dt,$$

or

$$\int_{I_{ij}} \Phi_i \dot v_i \, dt = \tilde\Phi_{ij} + \int_{I_{ij}} f_i^*(\Phi,\cdot) v_i \, dt,$$

with $\tilde\Phi_{ij} = \psi_i + \int_{t_{ij}}^{T} f_i^*(\Phi,\cdot)\, dt$. Dropping indices and rescaling to the interval $[0,1]$, we obtain

$$\int_0^1 \Phi \dot v \, dt = \tilde\Phi(1) + \int_0^1 f^* v \, dt,$$

for all $v \in \mathcal{P}^q([0,1])$ with $v(0) = 0$ and $v(1) = 1$. Let now $\{\lambda_n^{[q-1]}\}_{n=0}^{q-1}$ be a basis of $\mathcal{P}^{q-1}([0,1])$ and write $\Phi(t) = \sum_{n=1}^q \xi_n \lambda_{n-1}^{[q-1]}(t)$. For $m = 1,\ldots,q$, we then have

$$\sum_{n=1}^{q} \xi_n \int_0^1 \lambda_{n-1}^{[q-1]}(t)\, m t^{m-1} \, dt = \tilde\Phi(1) + \int_0^1 f^*\, t^m \, dt.$$

If now $\det\left(\int_0^1 \lambda_{n-1}^{[q-1]}(t) m t^{m-1}\, dt\right) \neq 0$, we can solve for $(\xi_1,\ldots,\xi_n)$. With $\lambda_{n-1}^{[q-1]}(t) = t^{n-1}$, we have

$$\det\left(\int_0^1 \lambda_{n-1}^{[q-1]}(t) m t^{m-1}\, dt\right) = \det\left(\int_0^1 m t^{n-1} t^{m-1}\, dt\right) = \det\left(\frac{m}{m+n-1}\right) \neq 0,$$

by Lemma 3.1. Solving for the degrees of freedom, we obtain

$$\xi_n = \alpha_n^{[q]} \tilde\Phi(1) + \int_0^1 w_n^{[q]} f^*\, dt, \quad n = 1,\ldots,q.$$

By a linear transformation, we obtain a similar expression for any other basis of $\mathcal{P}^q([0,1])$. For $f^* \equiv 0$, it is easy to see that the mcG$(q)^*$ solution is constant and equal to its initial value. Thus, when $\{\lambda_n^{[q-1]}\}_{n=0}^{q-1}$ is the standard Lagrange basis for a partition of $[0,1]$, it follows that $\alpha_n^{[q]} = 1$, $n = 1,\ldots,q$, and so

$$\xi_n = \tilde\Phi(1) + \int_0^1 w_n^{[q]} f^*\, dt, \quad n = 1,\ldots,q.$$

We note that $w_n^{[q]}(0) = 0$, $n = 1,\ldots,q$, since each $w_n^{[q]}$ is a linear combination of the functions $\{t^m\}_{m=1}^q$. We also conclude that $w_n^{[q]}(1) = 1$, since $w_n^{[q]}(1) = \alpha_n^{[q]} = 1$, $n = 1,\ldots,q$. The result now follows by rescaling to I_{ij}. We also relabel the degrees of freedom from $(\xi_1,\ldots,\xi_q)$ to $(\xi_0,\ldots,\xi_{q-1})$. □

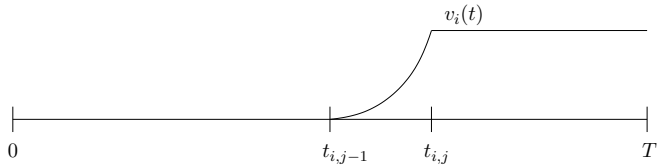

FIGURE 2. The special choice of test function used in the proof of Lemma 3.4.

Lemma 3.5. *The* mdG$(q)^*$ *method for* (2.9) *in fixed point form reads: For* $i = 1,\ldots,N$, $j = M_i,\ldots,1$, *find* $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, *such that*

$$\xi_{ijn} = \xi_{ijq_{ij}}^+ + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i^*(\Phi(t), t)\, dt, \tag{3.4}$$

with

$$\xi_{ijq_{ij}}^+ = \begin{cases} \xi_{i,j+1,0}, & j < M_i, \\ \psi_i, & j = M_i, \end{cases}$$

where $f^*(\Phi,\cdot) = J^\top(\pi u, U, \cdot)\Phi + g$, $\{w_n^{[q_{ij}]}\}_{n=0}^{q_{ij}} \subset \mathcal{P}^{q_{ij}}([0,1])$, *and* $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$. *A component* $\Phi_i(t)$ *of the solution is given on* I_{ij}

by

$$\Phi_i(t) = \sum_{n=0}^{q_{ij}} \xi_{ijn} \lambda_n^{[q_{ij}]}(\tau_{ij}(t)),$$

where $\{\lambda_n^{[q_{ij}]}\}_{n=0}^{q_{ij}} \subset \mathcal{P}^{q_{ij}}([0,1])$ *is the standard Lagrange basis on* $[0,1]$ *with* $t = 0$ *as one of its* $q_{ij} + 1 \geq 1$ *nodal points.*

Proof. The mdG$(q)^*$ method is identical to the mdG(q) method with time reversed. □

Corollary 3.1. *Let* $\mathcal{T}_n$ *be a time slab with synchronized time levels* T_{n-1} *and* T_n*. With time reversed for the dual methods (to simplify the notation), the* mcG(q)*,* mdG(q)*,* mcG$(q)^*$*, and* mdG$(q)^*$ *methods can all be expressed in the form: For all* $I_{ij} \in \mathcal{T}_n$*, find* $\{\xi_{ijn}\}$*, such that*

$$\xi_{ijn} = \tilde{U}_i(T_{n-1}^-) + \int_{T_{n-1}}^{t_{i,j-1}} f_i(U,\cdot)\,dt + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i(U,\cdot)\,dt, \tag{3.5}$$

with a suitable definition of $\tilde{U}_i(T_{n-1}^-)$*. As before,* $\tau_{ij}(t) = (t - t_{i,j-1})/(t_{ij} - t_{i,j-1})$ *and* $\{w_n^{[q_{ij}]}\}$ *is a set of polynomial weight functions on* $[0,1]$*.*

Proof. For mcG(q), mdG(q), and mdG$(q)^*$, the result follows if we take $\tilde{U}(T_{n-1}^-) = U(T_{n-1}^-)$ and note that $w_q^{[q]} \equiv 1$. For mcG$(q)^*$, the result follows if we define $\tilde{U}(T_{n-1}^-) = u_i(0) + \int_0^{T_{n-1}} f_i(U,\cdot)\,dt$. □

3.2. **Fixed point iteration.** We now prove that for each of the four methods, mcG(q), mdG(q), mcG$(q)^*$, and mdG$(q)^*$, the fixed point iterations of Corollary 3.1 converge, proving existence of the discrete solutions.

Theorem 3.1. (Existence of solutions) *Let* $K = \max K_n$ *be the maximum time slab length and define the Lipschitz constant* $L_f > 0$ *by*

$$\|f(x,t) - f(y,t)\|_{l_\infty} \leq L_f \|x - y\|_{l_\infty} \quad \forall t \in [0,T]\ \forall x, y \in \mathbb{R}^N. \tag{3.6}$$

If now

$$KC_q L_f < 1, \tag{3.7}$$

where C_q *is a constant of moderate size, depending only on the order and method, then each of the fixed point iterations,* (3.1)*,* (3.2)*,* (3.3)*, and* (3.4)*, converge to the unique solution of* (2.2)*,* (2.6)*,* (2.12)*, and* (2.15)*, respectively.*

Proof. Let $x = (\ldots, \xi_{ijn}, \ldots)$ be the set of values for the degrees of freedom of $U(t)$ on the time slab $\mathcal{T}_n$ of length $K_n = T_n - T_{n-1} \leq K$. Then, by Corollary 3.1, we can write the fixed point iteration on the time slab in the form

$$\xi_{ijn} = g_{ijn}(x) = \tilde{U}_i(T_{n-1}^-) + \int_{T_{n-1}}^{t_{i,j-1}} f_i(U,\cdot)\,dt + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t)) f_i(U,\cdot)\,dt.$$

Let $V(t)$ be another trial space function on the time slab with degrees of freedom $y = (\ldots, \eta_{ijn}, \ldots)$. Then,

$$\begin{aligned} g_{ijn}(x) - g_{ijn}(y) &= \int_{T_{n-1}}^{t_{i,j-1}} (f_i(U,\cdot) - f_i(V,\cdot))\,dt \\ &\quad + \int_{I_{ij}} w_n^{[q_{ij}]}(\tau_{ij}(t))(f_i(U,\cdot) - f_i(V,\cdot))\,dt, \end{aligned}$$

and so

$$\begin{aligned} \|g(x) - g(y)\|_{l_\infty} &\leq CL_f \int_{T_{n-1}}^{T_n} \|U(t) - V(t)\|_{l_\infty}\,dt \\ &\leq CL_f K \sup_{(T_{n-1},T_n]} \|U(t) - V(t)\|_{l_\infty}. \end{aligned}$$

Noting now that

$$|U_i(t) - V_i(t)| \leq \sum_n |\xi_{ijn} - \eta_{ijn}||\lambda_n^{[q_{ij}]}(t)| \leq C'\|x - y\|_{l_\infty},$$

for $t \in I_{ij}$, we thus obtain

$$\|g(x) - g(y)\|_{l_\infty} \leq CC'L_f K \|x - y\|_{l_\infty}.$$

By Banach's fixed point theorem, we conclude that the fixed point iteration converges to a unique fixed point if $CC'L_f K < 1$. □

4. Stability estimates

In this section, we prove stability estimates for the multi-adaptive methods and the corresponding multi-adaptive dual methods. We consider the linear model problem

$$\begin{aligned} \dot{u}(t) + A(t)u(t) &= 0, \quad t \in (0,T], \\ u(0) &= u_0, \end{aligned} \tag{4.1}$$

where $A = A(t)$ is a piecewise smooth $N \times N$ matrix on $(0,T]$. The dual backward problem of (4.1) for $\phi = \phi(t)$ is then given by

$$\begin{aligned} -\dot{\phi}(t) + A^\top(t)\phi(t) &= 0, \quad t \in [0,T), \\ \phi(T) &= \psi. \end{aligned} \tag{4.2}$$

With $w(t) = \phi(T - t)$, we have $\dot{w}(t) = -\dot{\phi}(T - t) = -A^\top(T - t)w(t)$, and so (4.2) can be written as a forward problem for w in the form

$$\begin{aligned} \dot{w}(t) + B(t)w(t) &= 0, \quad t \in (0,T], \\ w(0) &= w_0, \end{aligned} \tag{4.3}$$

where $w_0 = \psi$ and $B(t) = A^\top(T - t)$. In the following discussion, w represents either u or $\phi(T - \cdot)$ and, correspondingly, W represents either the discrete mc/dG(q) approximation U of u or the discrete mc/dG$(q)^*$ approximation Φ of ϕ.

4.1. **Exponential stability estimates.** The stability estimates are based on the following version of the discrete Grönwall inequality.

Lemma 4.1. (A discrete Grönwall inequality) *Assume that* $z, a : \mathbb{N} \to \mathbb{R}$ *are non-negative,* $a(m) \leq 1/2$ *for all* m*, and* $z(n) \leq C + \sum_{m=1}^{n} a(m)z(m)$ *for all* n*. Then, for* $n = 1, 2, \ldots$*, we have*

$$z(n) \leq 2C \exp\left(\sum_{m=1}^{n-1} 2a(m)\right). \tag{4.4}$$

Proof. By a standard discrete Grönwall inequality, it follows that $z(n) \leq C \exp\left(\sum_{m=0}^{n-1} a(m)\right)$, if $z(n) \leq C + \sum_{m=0}^{n-1} a(m)z(m)$ for $n \geq 1$ and $z(0) \leq C$, see [8]. Here, $(1 - a(n))z(n) \leq C + \sum_{m=1}^{n-1} a(m)z(m)$, and so $z(n) \leq 2C + \sum_{m=1}^{n-1} 2a(m)z(m)$, since $1 - a(n) \geq 1/2$. The result now follows if we take $a(0) = z(0) = 0$. □

Theorem 4.1. (Stability estimate) *Let* W *be the* mcG(q), mdG(q), mcG(q)*, *or* mdG(q)* *solution of* (4.3)*. Then, there is a constant* C_q *of moderate size, depending only on the highest order* $\max q_{ij}$*, such that if*

$$K_n C_q \|B\|_{L_\infty([T_{n-1},T_n],l_p)} \leq 1, \quad n = 1, \ldots, M, \tag{4.5}$$

then

$$\|W\|_{L_\infty([T_{n-1},T_n],l_p)} \leq C_q \|w_0\|_{l_p} \exp\left(\sum_{m=1}^{n-1} K_m C_q \|B\|_{L_\infty([T_{m-1},T_m],l_p)}\right), \tag{4.6}$$

for $n = 1, \ldots, M$*,* $1 \leq p \leq \infty$*.*

Proof. By Corollary 3.1, we can write the mcG(q), mdG(q), mcG(q)*, and mdG(q)* methods in the form

$$\xi_{ijn'} = w_i(0) + \int_0^{t_{i,j-1}} f_i(W, \cdot)\, dt + \int_{I_{ij}} w_{n'}^{[q_{ij}]}(\tau_{ij}(t)) f_i(W, \cdot)\, dt.$$

Applied to the linear model problem (4.3), we have

$$\xi_{ijn'} = w_i(0) - \int_0^{t_{i,j-1}} (BW)_i\, dt - \int_{I_{ij}} w_{n'}^{[q_{ij}]}(\tau_{ij}(t))(BW)_i\, dt,$$

and so

$$\begin{aligned} |\xi_{ijn'}| &\leq |w_i(0)| + \left|\int_0^{t_{i,j-1}} (BW)_i\, dt\right| + \left|\int_{I_{ij}} w_{n'}^{[q_{ij}]}(\tau_{ij}(t))(BW)_i\, dt\right| \\ &\leq |w_i(0)| + C\int_0^{t_{ij}} |(BW)_i|\, dt \leq |w_i(0)| + C\int_0^{T_n} |(BW)_i|\, dt, \end{aligned}$$

where T_n is smallest synchronized time level for which $t_{ij} \leq T_n$. It now follows that for all $t \in [T_{n-1}, T_n]$, we have $|W_i(t)| \leq C|w_i(0)| + C\int_0^{T_n} |(BW)_i|\, dt$,

and so $\|W(t)\|_{l_p}$ is bounded by

$$C\|w_0\|_{l_p} + C\int_0^{T_n} \|BW\|_{l_p}\, dt = C\|w_0\|_{l_p} + C\sum_{m=1}^{n} \int_{T_{m-1}}^{T_m} \|BW\|_{l_p}\, dt.$$

With $\bar{W}_n = \|W\|_{L_\infty([T_{n-1},T_n],l_p)}$, this means that

$$\begin{aligned} \bar{W}_n &\leq C\|w_0\|_{l_p} + C\sum_{m=1}^{n} K_m \|B\|_{L_\infty([T_{m-1},T_m],l_p)} \bar{W}_m \\ &\equiv (C_q/2)\|w_0\|_{l_p} + \sum_{m=1}^{n} K_m (C_q/2) \|B\|_{L_\infty([T_{m-1},T_m],l_p)} \bar{W}_m. \end{aligned}$$

By assumption, $K_m C_q \|B\|_{L_\infty([T_{m-1},T_m],l_p)} \leq 1$ for all m, and so the result follows by Lemma 4.1. □

4.2. **Stability estimates for parabolic problems.** We consider now the parabolic model problem $\dot{u}(t) + Au(t) = 0$, with A a symmetric, positive semidefinite, and constant $N \times N$ matrix, and prove stability estimates for the mdG(q) and mdG(q)* methods. As before, we write the problem in the form (4.3), and note that $B = A = A^\top$. We thus consider the problem: Find $w : [0, T] \to \mathbb{R}^N$, such that

$$\begin{aligned} \dot{w}(t) + Aw(t) &= 0, \quad t \in (0, T], \\ w(0) &= w_0. \end{aligned} \tag{4.7}$$

For the continuous problem (4.7), we have the following standard strong stability estimates, where "strong" indicates control of Aw (or $\dot{w}$) in terms of (the l_2-norm of) the initial data w_0.

Theorem 4.2. (Strong stability for the continuous problem) *The solution* w *of* (4.7) *satisfies for* $T > 0$ *and* $0 < \epsilon < T$ *with* $\|\cdot\| = \|\cdot\|_{l_2}$*,*

$$\|w(T)\|^2 + 2\int_0^T (Aw, w)\, dt = \|w_0\|^2, \tag{4.8}$$

$$\int_0^T t\, \|Aw\|^2\, dt \leq \frac{1}{4}\|w_0\|^2, \tag{4.9}$$

$$\int_\epsilon^T \|Aw\|\, dt \leq \frac{1}{2}(\log(T/\epsilon))^{1/2}\, \|w_0\|. \tag{4.10}$$

Proof. Multiply (4.7) with $v = w$, $v = tAw$, and $v = t^2A^2w$, respectively. See [2] for a full proof. □

We now prove an extension to multi-adaptive time-stepping of the strong stability estimate Lemma 6.1 in [1]. See also Lemma 1 in [3] for a similar estimate. In the proof, we use a special interpolant π, defined on the partition $\mathcal{T}$ as follows. On each local interval, the component $(\pi\varphi)_i$ of the interpolant $\pi\varphi$ of a given function $\varphi : [0, T] \to \mathbb{R}^N$, is defined by the following conditions: $(\pi\varphi_i)|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$ interpolates φ_i at the left end-point $t_{i,j-1}^+$ of I_{ij}

and $\pi\varphi_i - \varphi_i$ is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$. (This is the interpolant denoted by $\pi^{[q]}_{\mathrm{dG}^*}$ in [6].) We also introduce the left-continuous piecewise constant function $\bar{t} = \bar{t}(t)$ defined by $\bar{t}(t) = \min_{ij}\{t_{ij} : t \le t_{ij}\}$. With $\{t_m\}$ the ordered sequence of individual time levels $\{t_{ij}\}$, as illustrated in Figure 3, $\bar{t} = \bar{t}(t)$ is thus the piecewise constant function that takes the value t_m on $(t_{m-1}, t_m]$. We make the following assumption on the partition $\mathcal{T}$:

$$T_{n-1}\int_{T_{n-1}}^{T_n}(Av, Av)\,dt \le \gamma\int_{T_{n-1}}^{T_n}(Av, \pi(\bar{t}Av))\,dt, \quad n = 2, \ldots, M, \tag{4.11}$$

for all functions v in the trial (and test) space V of the mdG(q) and mdG(q)* methods, where $\gamma \ge 1$ is a constant of moderate size if Av is not close to being orthogonal to V. In the case of equal time steps for all components, this estimate is trivially true, because then $\pi Av = Av$ since $Av \in V$ if $v \in V$. Note that we may computationally test the validity of (4.11), see [7].

Theorem 4.3. (Strong stability for the discrete problem) *Let W be the* mdG(q) *or* mdG(q)* *solution of* (4.7)*, computed with the same time step and order for all components on the first time slab $\mathcal{T}_1$. Assume that* (4.11) *holds and that $\sigma K_n \le T_{n-1}$, $n = 2, \ldots, M$, for some constant $\sigma > 1$. Then, there is a constant $C = C(q, \gamma, \sigma)$, such that*

$$\|W(T)\|^2 + 2\int_0^T (AW, W)\,dt + \sum_{i=1}^{N}\sum_{j=1}^{M_i}[W_i]^2_{i,j-1} = \|w_0\|^2, \tag{4.12}$$

(4.13)

$$\sum_{n=1}^{M} T_n\int_{T_{n-1}}^{T_n}\left\{\|\dot{W}\|^2 + \|AW\|^2\right\}dt + \sum_{n=1}^{M} T_n\sum_{ij}[W_i]^2_{i,j-1}/k_{ij} \le C\,\|w_0\|^2,$$

(4.14)

$$\int_0^T\left\{\|\dot{W}\| + \|AW\|\right\}dt + \sum_{n=1}^{N}\left(\sum_{ij}|[W_i]_{i,j-1}|^2\right)^{1/2} \le C\left(\log\frac{T}{K_1} + 1\right)^{1/2}\|w_0\|,$$

where $\|\cdot\| = \|\cdot\|_{l_2}$, $\sum_{ij}$ denotes the sum over all elements within the current time slab $\mathcal{T}_n$, and where in all integrals the domain of integration does not include points where the integrand is discontinuous.

Proof. We follow the proof presented in [1] and make extensions where necessary. With V the trial (and test) space for the mdG(q) method defined in Section 2, the mdG(q) (or mdG(q)*) approximation W of w on a time slab $\mathcal{T}_n$ is defined as follows: Find $W \in V$, such that

$$\sum_{ij}\left([W_i]_{i,j-1}v_i(t^+_{i,j-1}) + \int_{I_{ij}}\dot{W}_i v_i\,dt\right) + \int_{T_{n-1}}^{T_n}(AW, v)\,dt = 0, \tag{4.15}$$

for all test functions $v \in V$, where the sum is taken over all intervals I_{ij} within the time slab $\mathcal{T}_n$. To prove the basic stability estimate (4.12), we

take $v = W$ in (4.15) to get

$$\frac{1}{2}\sum_{ij}[W_i]^2_{i,j-1} + \frac{1}{2}\|W(T_n^-)\|^2 - \frac{1}{2}\|W(T_{n-1}^-)\|^2 + \int_{T_{n-1}}^{T_n}(AW, W)\,dt = 0.$$

The estimate now follows by summation over all time slabs $\mathcal{T}_n$.

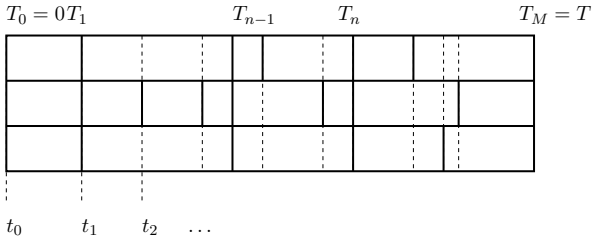

FIGURE 3. The sequence $\{t_m\}$ of individual time levels.

For the proof of (4.13), we would like to take $v = tAW$ in (4.15), but this is not a valid test function. In the proof of Lemma 6.1 in [1], the test function is chosen as $v = T_nAW$, which is not possible in the multi-adaptive case, since A mixes the components of W and as a result, $v_i = T_n(AW)_i$ may not be a test function for component W_i. Instead, we take $v = \pi(\bar{t}AW)$, with π and $\bar{t}$ defined as above, to obtain

$$\sum_{ij}\left([W_i]_{i,j-1}(\bar{t}(AW)_i)(t^+_{i,j-1}) + \int_{I_{ij}}\dot{W}_i\,\bar{t}(AW)_i\,dt\right) + \int_{T_{n-1}}^{T_n}(AW, \pi(\bar{t}AW))\,dt = 0,$$

where we have used the orthogonality condition of the interpolant for $\dot{W}_i \in \mathcal{P}^{q_{ij}-1}(I_{ij})$. Noting that $[W_i]_m = W_i(t^+_m) - W_i(t^-_m) = 0$ if component i has no node at time $t = t_m$, we rewrite the sum as a sum over all intervals $(t_{m-1}, t_m]$ within $(T_{n-1}, T_n]$, in the form

$$\begin{aligned}
&\sum_i\sum_m[W_i]_{m-1}t_m(AW(t^+_{m-1}))_i + \int_{I_m}\dot{W}_i\,t_m(AW)_i\,dt\\
&= \sum_m t_m([W]_{m-1}, AW(t^+_{m-1})) + \frac{t_m}{2}\int_{I_m}\frac{d}{dt}(W, AW)\,dt\\
&= \sum_m\frac{t_m}{2}([W]_{m-1}, A[W]_{m-1})\\
&\quad + \frac{t_m}{2}\left[(W(t^-_m), AW(t^-_m)) - (W(t^-_{m-1}), AW(t^-_{m-1}))\right],
\end{aligned}$$

where, using the notation $k_m = t_m - t_{m-1}$, we note that

$$\begin{aligned}
&\sum_m \frac{t_m}{2}\left[(W(t_m^-), AW(t_m^-)) - (W(t_{m-1}^-), AW(t_{m-1}^-))\right] \\
&= \sum_m \frac{t_m}{2}(W(t_m^-), AW(t_m^-)) - \frac{t_{m-1}}{2}(W(t_{m-1}^-), AW(t_{m-1}^-)) \\
&\quad - \frac{k_m}{2}(W(t_{m-1}^-), AW(t_{m-1}^-)) \\
&= \frac{T_n}{2}(W(T_n^-), AW(T_n^-)) - \frac{T_{n-1}}{2}(W(T_{n-1}^-), AW(T_{n-1}^-)) \\
&\quad - \sum_m \frac{k_m}{2}(W(t_{m-1}^-), AW(t_{m-1}^-)).
\end{aligned}$$

Collecting the terms, we thus have

(4.16)
$$\sum_m t_m([W]_{m-1}, A[W]_{m-1}) + T_n(W(T_n^-), AW(T_n^-)) - T_{n-1}(W(T_{n-1}^-), AW(T_{n-1}^-)) + 2\int_{T_{n-1}}^{T_n} (AW, \pi(\bar{t}AW))\,dt = \sum_m k_m(W(t_{m-1}^-), AW(t_{m-1}^-)).$$

For $n = 1$, we have

$$\int_{T_{n-1}}^{T_n} (AW, \pi(\bar{t}AW))\,dt = T_1 \int_0^{T_1} (AW, \pi(AW))\,dt = T_1 \int_0^{T_1} \|AW\|^2\,dt,$$

since $\pi(AW) = AW$ on $[0, T_1]$, where the same time steps and orders are used for all components. We further estimate the right-hand side of (4.16) as follows:

$$\begin{aligned}
K_1(w_0, Aw_0) &= K_1([W]_0, A[W]_0) - K_1(W(0^+), AW(0^+)) + 2K_1(w_0, AW(0^+)) \\
&\le T_1([W]_0, A[W]_0) + 2K_1(w_0, AW(0^+)) \\
&\le T_1([W]_0, A[W]_0) + \frac{1}{\epsilon}\|w_0\|^2 + \epsilon K_1^2 \|AW(0^+)\|^2 \\
&\le T_1([W]_0, A[W]_0) + \frac{1}{\epsilon}\|w_0\|^2 + \epsilon C_q T_1 \int_0^{T_1} \|AW\|^2\,dt,
\end{aligned}$$

where we have used an inverse estimate for AW on $[0, T_1]$. With $\epsilon = 1/C_q$, we thus obtain the estimate

$$T_1(W(T_1^-), AW(T_1^-)) + T_1 \int_0^{T_1} \|AW\|^2 \le C_q \|w_0\|^2. \tag{4.17}$$

For $n > 1$, it follows by the assumption (4.11), that

$$\int_{T_{n-1}}^{T_n} (AW, \pi(\bar{t}AW))\,dt \ge \gamma^{-1} T_{n-1} \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt$$

and thus

$$\int_{T_{n-1}}^{T_n} (AW, \pi(\bar{t}AW))\,dt \ge \frac{\gamma^{-1}\sigma}{\sigma+1} T_n \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt,$$

where we have also used the assumption $T_{n-1} \ge \sigma K_n$. The terms on the right-hand side of (4.16) are now estimated as follows:

$$\begin{aligned}
&k_m(W(t_{m-1}^-), AW(t_{m-1}^-)) = k_m(W(t_{m-1}^+) - [W]_{m-1}, A(W(t_{m-1}^+) - [W]_{m-1})) \\
&= k_m\left[(W(t_{m-1}^+), AW(t_{m-1}^+)) + ([W]_{m-1}, A[W]_{m-1}) - 2(W(t_{m-1}^+), A[W]_{m-1})\right] \\
&\le k_m\left[(1+\beta)(W(t_{m-1}^+), AW(t_{m-1}^+)) + (1+\beta^{-1})([W]_{m-1}, A[W]_{m-1})\right],
\end{aligned}$$

for any $\beta > 0$. Choosing $\beta = 1/(\sigma - 1)$, we obtain the following bound for $k_m(W(t_{m-1}^-), AW(t_{m-1}^-))$:

$$\begin{aligned}
&\frac{\sigma k_m}{\sigma - 1}(W(t_{m-1}^+), AW(t_{m-1}^+)) + \sigma k_m([W]_{m-1}, A[W]_{m-1}) \\
&\qquad \le \frac{C_q \sigma}{\sigma - 1}\int_{I_m} (W, AW)\,dt + \sigma k_m([W]_{m-1}, A[W]_{m-1}),
\end{aligned}$$

where we have again used an inverse estimate, for W on I_m. From the assumption that $\sigma K_n \le T_{n-1}$ for $n > 1$, it follows that $\sigma k_m \le \sigma K_n \le T_{n-1} \le t_m$, and so

$$\begin{aligned}
&T_n(W(T_n^-), AW(T_n^-)) - T_{n-1}(W(T_{n-1}^-), AW(T_{n-1}^-)) \\
&+ \frac{2\gamma^{-1}\sigma}{\sigma+1} T_n \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt \le \frac{C_q\sigma}{\sigma-1}\int_{T_{n-1}}^{T_n} (W, AW)\,dt.
\end{aligned}$$

Summing over $n > 1$ and using the estimate (4.17) for $n = 1$, we obtain by (4.12),

$$\begin{aligned}
&T\left(W(T^-), AW(T^-)\right) + T_1 \int_0^{T_1} \|AW\|^2\,dt + \frac{2\gamma^{-1}\sigma}{\sigma+1}\sum_{n=2}^{M} T_n \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt \\
&\qquad \le C_q\|w_0\|^2 + \frac{C_q\sigma}{\sigma-1}\int_{T_1}^{T} (W, AW)\,dt \le C_q\left(1 + \frac{\sigma/2}{\sigma-1}\right)\|w_0\|^2,
\end{aligned}$$

which we write as

$$\sum_{n=1}^{M} T_n \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt \le C\,\|w_0\|^2,$$

noting that $T(W(T^-), AW(T^-)) \ge 0$. For the proof of (4.13), it now suffices to prove that

$$\int_{T_{n-1}}^{T_n} \|\dot{W}\|\,dt \le C \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt, \tag{4.18}$$

and

$$\sum_{ij} [W_i]_{i,j-1}^2 / k_{ij} \le C \int_{T_{n-1}}^{T_n} \|AW\|^2\,dt. \tag{4.19}$$

To prove (4.18), we take $v_i = (t - t_{i,j-1})\dot{W}_i/k_{ij}$ on each local interval I_{ij} in (4.15), which gives

$$\begin{aligned}
&\sum_{ij} \int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} \dot{W}_i^2 \, dt = -\sum_{ij} \int_{I_{ij}} (AW)_i \frac{t - t_{i,j-1}}{k_{ij}} \dot{W}_i \, dt \\
&\leq \sum_{ij} \left(\int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} (AW)_i^2 \, dt \right)^{1/2} \left(\int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} \dot{W}_i^2 \, dt \right)^{1/2} \\
&\leq \left(\sum_{ij} \int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} (AW)_i^2 \, dt \right)^{1/2} \left(\sum_{ij} \int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} \dot{W}_i^2 \, dt \right)^{1/2},
\end{aligned}$$

where we have used Cauchy's inequality twice; first on $L_2(I_{ij})$ and then on l_2. Using an inverse estimate for $\dot{W}_i^2$, we obtain

$$\begin{aligned}
\int_{T_{n-1}}^{T_n} \|\dot{W}\|^2 \, dt &= \sum_{ij} \int_{I_{ij}} \dot{W}_i^2 \, dt \leq C_q \sum_{ij} \int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} \dot{W}_i^2 \, dt \\
&\leq C_q \sum_{ij} \int_{I_{ij}} \frac{t - t_{i,j-1}}{k_{ij}} (AW)_i^2 \, dt \\
&\leq C_q \sum_{ij} \int_{I_{ij}} (AW)_i^2 \, dt = C_q \int_{T_{n-1}}^{T_n} \|AW\| \, dt,
\end{aligned}$$

which proves (4.18).

To prove (4.19), we take $v_i = [W_i]_{i,j-1}/k_{ij}$ on each local interval I_{ij} in (4.15), which gives

$$\begin{aligned}
\sum_{ij} [W_i]_{i,j-1}^2/k_{ij} &= -\sum_{ij} \int_{I_{ij}} (\dot{W}_i + (AW)_i) \, [W_i]_{i,j-1}/k_{ij} \, dt \\
&\leq \left(\sum_{ij} \int_{I_{ij}} (\dot{W}_i + (AW)_i)^2 \, dt \right)^{1/2} \left(\sum_{ij} [W_i]_{i,j-1}^2/k_{ij} \right)^{1/2},
\end{aligned}$$

where we have again used Cauchy's inequality twice. We thus have

$$\sum_{ij} [W_i]_{i,j-1}^2/k_{ij} \leq 2 \int_{T_{n-1}}^{T_n} \|\dot{W}\|^2 \, dt + 2 \int_{T_{n-1}}^{T_n} \|AW\|^2 \, dt,$$

and so (4.19) follows, using (4.18). This also proves (4.13).

Finally, to prove (4.14), we use Cauchy's inequality with (4.13) to get

$$\begin{aligned}
\sum_{n=1}^{M} \int_{T_{n-1}}^{T_n} \|\dot{W}\| \, dt &= \sum_{n=1}^{M} \sqrt{K_n/T_n} \sqrt{T_n/K_n} \int_{T_{n-1}}^{T_n} \|\dot{W}\| \, dt \\
&\leq \left(\sum_{n=1}^{M} K_n/T_n \right)^{1/2} \left(\sum_{n=1}^{M} T_n \int_{T_{n-1}}^{T_n} \|\dot{W}\|^2 \, dt \right)^{1/2} \\
&\leq \left(1 + \int_{T_1}^{T} \frac{1}{t} \, dt \right)^{1/2} C \|w_0\| \\
&\leq C \left(\log(T/K_1) + 1 \right)^{1/2} \|w_0\|,
\end{aligned}$$

with a similar estimate for AW. The proof is now complete, noting that

$$\begin{aligned}
\sum_{n=1}^{M} \left(\sum_{ij} |[W_i]_{i,j-1}|^2 \right)^{1/2} &\leq \sum_{n=1}^{M} \left(\frac{K_n}{T_n} T_n \sum_{ij} |[W_i]_{i,j-1}|^2/k_{ij} \right)^{1/2} \\
&\leq \left(\sum_{n=1}^{M} K_n/T_n \right)^{1/2} \left(\sum_{n=1}^{M} T_n \sum_{ij} |[W_i]_{i,j-1}|^2/k_{ij} \right)^{1/2} \\
&\leq C (\log(T/K_1) + 1)^{1/2} \|w_0\|.
\end{aligned}$$

□

References

[1] K. Eriksson and C. Johnson, *Adaptive finite element methods for parabolic problems I: A linear model problem*, SIAM J. Numer. Anal., 28, No. 1 (1991), pp. 43–77.

[2] K. Eriksson, C. Johnson, and A. Logg, *Adaptive computational methods for parabolic problems*, To appear in Encyclopedia of Computational Mechanics, (2004).

[3] 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 (1988), pp. 908–926.

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

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

[6] ———, *Interpolation estimates for piecewise smooth functions in one dimension*, Tech. Rep. 2004–02, Chalmers Finite Element Center Preprint Series, 2004.

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

[8] P. Niamsup and V. N. Phat, *Asymptotic stability of nonlinear control systems described by difference equations with multiple delays*, Electronic Journal of Differential Equations, 11 (2000), pp. 1–17.

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES IV: A PRIORI ERROR ESTIMATES

ANDERS LOGG

Abstract. We prove general order a priori error estimates for the multi-adaptive continuous and discontinuous Galerkin methods mcG(q) and mdG(q). To prove the error estimates, we represent the error in terms of the residual of an interpolant of the exact solution, and a discrete dual solution. The estimates then follow from interpolation estimates, together with stability estimates for the discrete dual solution. For the general non-linear problem, we obtain exponential stability estimates, using a Grönwall argument, and for a parabolic model problem, we show that the stability factor is of unit size.

1. Introduction

This is part IV in a sequence of papers [4, 5, 8] 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 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.

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. In the current paper, we prove a priori error estimates for the mcG(q) and mdG(q) methods.

Date: March 15, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, a priori error estimates.

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

1.1. **Main results.** The main results of this paper are a priori error estimates for the mcG(q) and mdG(q) methods respectively, of the form

$$\|e(T)\|_{l_p} \leq CS(T)\|k^{2q}u^{(2q)}\|_{L_\infty([0,T],l_1)}, \tag{1.2}$$

and

$$\|e(T)\|_{l_p} \leq CS(T)\|k^{2q+1}u^{(2q+1)}\|_{L_\infty([0,T],l_1)}, \tag{1.3}$$

for $p = 2$ or $p = \infty$, where C is an interpolation constant, $S(T)$ is a (computable) stability factor, and $k^{2q}u^{(2q)}$ (or $k^{2q+1}u^{(2q+1)}$) combines local time steps $k = (k_{ij})$ with derivatives of the exact solution u. These estimates state that the mcG(q) method is of order $2q$ and that the mdG(q) method is of order $2q+1$ in the local time step. We refer to Section 5 for the exact results. For the general non-linear problem, we obtain exponential estimates for the stability factor $S(T)$, and for a parabolic model problem we show that the stability factor remains bounded and of unit size, independent of T (up to a logarithmic factor).

1.2. **Notation.** For a detailed description of the multi-adaptive Galerkin methods, we refer the reader to [4, 5, 8]. In particular, we refer to [4] or [8] 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 $K_n = T_n - T_{n-1}$.

1.3. **Outline of the paper.** The outline of the paper is as follows. In Section 2, we first discuss the dual problem that forms the basic tool of the a priori error analysis, and how this differs from the dual problem we formulate in [4] for the a posteriori error analysis. We then, in Section 3, derive a representation of the error in terms of the dual solution and the residual of an interpolant of the exact solution.

In Section 4, we present interpolation results for piecewise smooth functions proved in [7, 6]. We then prove the a priori error estimates in Section 5, starting from the error representation and using the interpolation estimates together with the stability estimates from [8]. Finally, in Section 6, we present some numerical evidence for the a priori error estimates. In

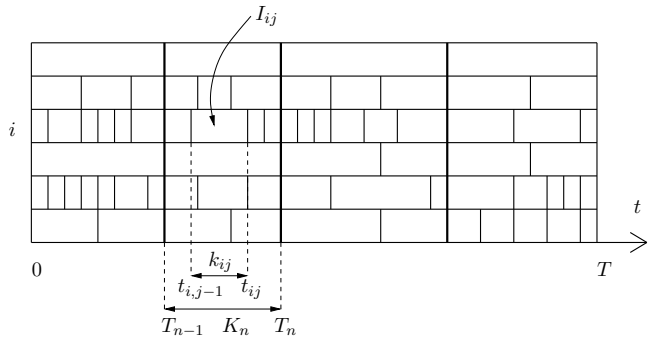

FIGURE 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

particular, we solve a simple model problem and show that we obtain the predicted convergence rates.

2. The dual problem

In [4], we prove *a posteriori* error estimates for the multi-adaptive methods, by deriving a representation for the error $e = U - u$, where $U : [0, T] \to \mathbb{R}^N$ is the computed approximate solution of (1.1), in terms of the residual $R(U, \cdot) = \dot{U} - f(U, \cdot)$ and the solution $\phi : [0, T] \to \mathbb{R}^N$ of the continuous linearized dual problem

$$\begin{aligned} -\dot{\phi}(t) &= J^\top(u, U, t)\phi(t) + g(t), \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{2.1}$$

with given data $g : [0, T] \to \mathbb{R}^N$ and $\psi \in \mathbb{R}^N$, where

$$J^\top(u, U, t) = \left(\int_0^1 \frac{\partial f}{\partial u}(su(t) + (1 - s)U(t), t)\, ds \right)^\top. \tag{2.2}$$

To prove *a priori* error estimates, we derive an error representation in terms of the residual $R(\pi u, \cdot)$ of an interpolant πu of the exact solution u, and a discrete dual solution Φ, following the same approach as in [3] and [1]. The discrete dual solution Φ is defined as a Galerkin solution of the continuous linearized dual problem

$$\begin{aligned} -\dot{\phi}(t) &= J^\top(\pi u, U, t)\phi(t) + g(t), \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{2.3}$$

where we note that J is now evaluated at a mean value of πu and U. We will use the notation $f^*(\phi, \cdot) = J^\top(\pi u, U, \cdot)\phi + g$, to write the dual problem (2.3) in the form

$$\begin{aligned} -\dot{\phi}(t) &= f^*(\phi(t), t), \quad t \in [0, T), \\ \phi(T) &= \psi. \end{aligned} \tag{2.4}$$

We refer to [8] for the exact definition of the discrete dual solution Φ.

We will also derive a priori error estimates for linear problems of the form

$$\begin{aligned} \dot{u}(t) + A(t)u(t) &= 0, \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{2.5}$$

with $A(t)$ a bounded $N \times N$-matrix, in particular for a parabolic model problem with $A(t)$ a positive semidefinite and symmetric matrix . For the linear problem (2.5), the discrete dual solution Φ is defined as a Galerkin solution of the continuous dual problem

$$\begin{aligned} -\dot{\phi}(t) + A^\top(t)\phi(t) &= g, \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{2.6}$$

which takes the form (2.4) with the notation $f^*(\phi, \cdot) = -A^\top\phi + g$.

3. Error representation

In this section, we derive the error representations on which the a priori error estimates are based. For each of the two methods, mcG(q) and mdG(q), we represent the error in terms of the discrete dual solution Φ and an interpolant πu of the exact solution u of (1.1), using the special interpolants $\pi u = \pi_{\text{cG}}^{[q]} u$ or $\pi u = \pi_{\text{dG}}^{[q]} u$ defined in Section 5 of [7]. The error representations are presented in the general non-linear case, and thus apply to the linear problem (2.5), with corresponding dual problem (2.6), in particular.

We write the error $e = U - u$ as

$$e = \bar{e} + (\pi u - u), \tag{3.1}$$

where $\bar{e} \equiv U - \pi u$ is represented in terms of the discrete dual solution and the residual of the interpolant. An estimate for the second part of the error, $\pi u - u$, follows directly from an interpolation estimate. In Theorem 3.1 below, we derive the error representation for the mcG(q) method, and then derive the corresponding representation for the mdG(q) method in Theorem 3.2.

Theorem 3.1. (Error representation for mcG(q)) *Let U be the* mcG(q) *solution of* (1.1)*, let Φ be the corresponding* mcG(q)* *solution of the dual problem* (2.4)*, and let πu be any trial space approximation of the exact solution u of* (1.1) *that interpolates u at the end-points of every local interval.*

Then,

$$L_{\psi,g}(\bar{e}) \equiv (\bar{e}(T),\psi) + \int_0^T (\bar{e},g)\,dt = -\int_0^T (R(\pi u,\cdot),\Phi)\,dt,$$

where $\bar{e} \equiv U - \pi u$.

Proof. Since $\bar{e}(0) = 0$, we can choose $\bar{e}$ as a test function for the discrete dual. By the definition of the mcG$(q)^*$ solution Φ (see [8]), we thus have

$$\int_0^T (\dot{\bar{e}},\Phi)\,dt = \int_0^T (J(\pi u,U,\cdot)\bar{e},\Phi)\,dt + L_{\psi,g}(\bar{e}),$$

and so, by the definition of J, we have

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \int_0^T (\dot{\bar{e}} - J(\pi u,U,\cdot)\bar{e},\Phi)\,dt = \int_0^T (\dot{\bar{e}} - f(U,\cdot) + f(\pi u,\cdot),\Phi)\,dt \\ &= \int_0^T (R(U,\cdot) - R(\pi u,\cdot),\Phi)\,dt = -\int_0^T (R(\pi u,\cdot),\Phi)\,dt, \end{aligned}$$

since Φ is a test function for U. □

Theorem 3.2. (Error representation for mdG(q)) *Let U be the* mdG(q) *solution of* (1.1)*, let Φ be the corresponding* mdG$(q)^*$ *solution of the dual problem* (2.4)*, and let πu be any trial space approximation of the exact solution u of* (1.1) *that interpolates u at the right end-point of every local interval. Then,*

$$\begin{aligned} L_{\psi,g}(\bar{e}) &\equiv (\bar{e}(T),\psi) + \int_0^T (\bar{e},g)\,dt \\ &= -\sum_{i=1}^N \sum_{j=1}^{M_i} \left[[\pi u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} R_i(\pi u,\cdot)\Phi_i\,dt \right], \end{aligned}$$

where $\bar{e} \equiv U - \pi u$.

Proof. Choosing $\bar{e}$ as a test function for the discrete dual we obtain, by the definition of the mdG$(q)^*$ method (see [8]),

$$\sum_{i=1}^N \sum_{j=1}^{M_i} \left[[\bar{e}_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} \dot{\bar{e}}_i\Phi_i\,dt \right] = \int_0^T (J(\pi u,U,\cdot)\bar{e},\Phi)\,dt + L_{\psi,g}(\bar{e}),$$

and so, by the definition of J, we have

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N \sum_{j=1}^{M_i} \left[[\bar{e}_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} \dot{\bar{e}}_i\Phi_i\,dt \right] - \int_0^T (J(\pi u,U,\cdot)\bar{e},\Phi)\,dt \\ &= \sum_{i=1}^N \sum_{j=1}^{M_i} \left[[U_i - \pi u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} (R_i(U,\cdot) - R_i(\pi u,\cdot))\Phi_i\,dt \right] \\ &= -\sum_{i=1}^N \sum_{j=1}^{M_i} \left[[\pi u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} R_i(\pi u,\cdot)\Phi_i\,dt \right], \end{aligned}$$

since Φ is a test function for U. □

With a special choice of interpolant, $\pi u = \pi_{\rm cG}^{[q]} u$ and $\pi u = \pi_{\rm dG}^{[q]} u$ respectively, we obtain the following versions of the error representations.

Corollary 3.1. (Error representation for mcG(q)) *Let U be the* mcG(q) *solution of* (1.1)*, let Φ be the corresponding* mcG$(q)^*$ *solution of the dual problem* (2.4)*, and let $\pi_{\rm cG}^{[q]} u$ be an interpolant, as defined in* [7]*, of the exact solution u of* (1.1)*. Then,*

$$L_{\psi,g}(\bar{e}) = \int_0^T (f(\pi_{\rm cG}^{[q]} u,\cdot) - f(u,\cdot),\Phi)\,dt.$$

Proof. The residual of the exact solution is zero and so, by Theorem 3.1, we have

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \int_0^T (R(u,\cdot) - R(\pi_{\rm cG}^{[q]} u,\cdot),\Phi)\,dt \\ &= \int_0^T (f(\pi_{\rm cG}^{[q]} u,\cdot) - f(u,\cdot),\Phi)\,dt + \int_0^T (\frac{d}{dt}(u - \pi_{\rm cG}^{[q]} u),\Phi)\,dt, \end{aligned}$$

where we note that

$$\int_0^T (\frac{d}{dt}(u - \pi_{\rm cG}^{[q]} u),\Phi)\,dt = 0,$$

by the construction of the interpolant $\pi_{\rm cG}^{[q]} u$ (Lemma 5.2 in [7]). □

Corollary 3.2. (Error representation for mdG(q)) *Let U be the* mdG(q) *solution of* (1.1)*, let Φ be the corresponding* mdG$(q)^*$ *solution of the dual problem* (2.4)*, and let $\pi_{\rm dG}^{[q]} u$ be an interpolant, as defined in* [7]*, of the exact solution u of* (1.1)*. Then,*

$$L_{\psi,g}(\bar{e}) = \int_0^T (f(\pi_{\rm dG}^{[q]} u,\cdot) - f(u,\cdot),\Phi)\,dt.$$

Proof. The residual of the exact solution is zero, and the jump of the exact solution is zero at every node. Thus, by Theorem 3.2,

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N \sum_{j=1}^{M_i} \left[[u_i - \pi_{\rm dG}^{[q_{ij}]} u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} (R_i(u,\cdot) - R_i(\pi_{\rm dG}^{[q]} u,\cdot))\Phi_i\,dt \right] \\ &= \int_0^T (f(\pi_{\rm dG}^{[q]} u,\cdot) - f(u,\cdot),\Phi)\,dt, \end{aligned}$$

where we have used the fact that $\pi_{\rm dG}^{[q]} u$ interpolates u at the right end-point of every local interval, and thus that

$$\sum_{i=1}^N \sum_{j=1}^{M_i} \left[[u_i(t_{i,j-1}^+) - \pi_{\rm dG}^{[q_{ij}]} u_i(t_{i,j-1}^+)]\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} (\frac{d}{dt}(u_i - \pi_{\rm dG}^{[q_{ij}]} u_i))\Phi_i\,dt \right] = 0,$$

by the construction of the interpolant $\pi_{\rm dG}^{[q]} u$ (Lemma 5.3 in [7]). □

3.1. **A note on quadrature errors.** In the derivation of the error representations, we have used the Galerkin orthogonalities for the mcG(q) and mdG(q) solutions. For the mcG(q) method, we have assumed that

$$\int_0^T (R(U,\cdot),\Phi)\,dt = 0$$

in the proof of Theorem 3.1, and for the mdG(q)method, we have assumed that

$$\sum_{i=1}^{N}\sum_{j=1}^{M_i}\left[[U_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} R_i(U,\cdot)\Phi_i\,dt\right] = 0$$

in the proof of Theorem 3.2. In the presence of quadrature errors, these terms are nonzero. As a result, we obtain additional terms of the form

$$\int_0^T (\tilde{f}(U,\cdot) - f(U,\cdot),\Phi)\,dt,$$

where $\tilde{f}$ is the interpolant of f corresponding the quadrature rule that is used. Typically, Lobatto quadrature (with $q+1$ nodal points) is used for the mcG(q) method, which means that the quadrature error is of order $2(q+1)-2 = 2q$ and so (super-) convergence of order $2q$ is obtained also in the presence of quadrature errors. Similarly for the mdG(q) method, we use Radau quadrature with $q+1$ nodal points, which means that the quadrature error is of order $2(q+1)-1 = 2q+1$, and so the $2q+1$ convergence order of mdG(q) is also maintained under quadrature.

4. Interpolation estimates

To prove the a priori error estimates, starting from the error representations derived in the previous section, we need special interpolation estimates. These estimates are proved in [6], based on the interpolation estimates of [7]. In this section, we present the interpolation estimates, first for the general non-linear problem and then for linear problems, and refer to [7, 6] for the proofs.

4.1. **The general non-linear problem.** In order to prove the interpolation estimates for the general non-linear problem, we need to make the following assumptions: Given a time slab $\mathcal{T}$, assume that for each pair of local intervals I_{ij} and I_{mn} within the time slab, we have

(A1) $$q_{ij} = q_{mn} = \bar{q},$$

and

(A2) $$k_{ij} > \alpha\, k_{mn},$$

for some $\bar{q} \geq 0$ and some $\alpha \in (0,1)$. We also assume that the problem (1.1) is autonomous,

(A3) $$\frac{\partial f_i}{\partial t} = 0, \quad i = 1,\ldots,N,$$

noting that the dual problem nevertheless is non-autonomous in general. Furthermore, we assume that

(A4) $$\|f_i\|_{D^{\bar{q}+1}(\mathcal{T})} < \infty, \quad i = 1,\ldots,N,$$

where $\|\cdot\|_{D^p(\mathcal{T})}$ is defined for $v : \mathbb{R}^N \to \mathbb{R}$ and $p \geq 0$ by $\|v\|_{D^p(\mathcal{T})} = \max_{n=0,\ldots,p}\|D^n v\|_{L_\infty(\mathcal{T},l_\infty)}$, with

(4.5) $$\|D^n v\, w^1\cdots w^n\|_{L_\infty(\mathcal{T})} \leq \|D^n v\|_{L_\infty(\mathcal{T},l_\infty)}\|w^1\|_{l_\infty}\cdots\|w^n\|_{l_\infty}$$

for all $w^1,\ldots,w^n \in \mathbb{R}^N$ and $D^n v$ the nth-order tensor given by

$$D^n v\, w^1\cdots w^n = \sum_{i_1=1}^{N}\cdots\sum_{i_n=1}^{N}\frac{\partial^n v}{\partial x_{i_1}\cdots\partial x_{i_n}}\,w_{i_1}^1\cdots w_{i_n}^n.$$

Furthermore, we choose $\|f\|_{\mathcal{T}} \geq \max_{i=1,\ldots,N}\|f_i\|_{D^{\bar{q}+1}(\mathcal{T})}$, such that

(4.6) $$\|d^p/dt^p(\partial f/\partial u)^\top(x(t))\|_{l_\infty} \leq \|f\|_{\mathcal{T}} C_x^p,$$

for $p = 0,\ldots,\bar{q}$, and

(4.7) $$\|[d^p/dt^p(\partial f/\partial u)^\top(x(t))]_t\|_{l_\infty} \leq \|f\|_{\mathcal{T}}\sum_{n=0}^{p} C_x^{p-n}\|[x^{(n)}]_t\|_{l_\infty},$$

for $p = 0,\ldots,\bar{q}-1$ and any given $x : \mathbb{R} \to \mathbb{R}^N$, where $C_x > 0$ denotes a constant, such that $\|x^{(n)}\|_{L_\infty(\mathcal{T},l_\infty)} \leq C_x^n$, for $n = 1,\ldots,p$. Note that assumption (A4) implies that each f_i is bounded by $\|f\|_{\mathcal{T}}$. We further assume that there is a constant $c_k > 0$, such that

(A5) $$k_{ij}\|f\|_{\mathcal{T}} \leq c_k,$$

for each local interval I_{ij}. We summarize the list of assumptions as follows:

- (A1) the local orders q_{ij} are equal within each time slab;
- (A2) the local time steps k_{ij} are semi-uniform within each time slab;
- (A3) f is autonomous;
- (A4) f and its derivatives are bounded;
- (A5) the local time steps k_{ij} are small.

To derive a priori error estimates for the non-linear problem (1.1), we need to estimate the interpolation error $\pi\varphi_i - \varphi_i$ on a local interval I_{ij}, where φ_i is defined by

(4.9) $$\varphi_i = (J^\top(\pi u, u)\Phi)_i = \sum_{l=1}^{N} J_{li}(\pi u, u)\Phi_l, \quad i = 1,\ldots,N.$$

We note that φ_i may be discontinuous within I_{ij}, if I_{ij} contains a node for some other component, which is generally the case with multi-adaptive time-stepping. This is illustrated in Figure 2. Using assumptions (A1)–(A5), we obtain the following interpolation estimates for the function φ.

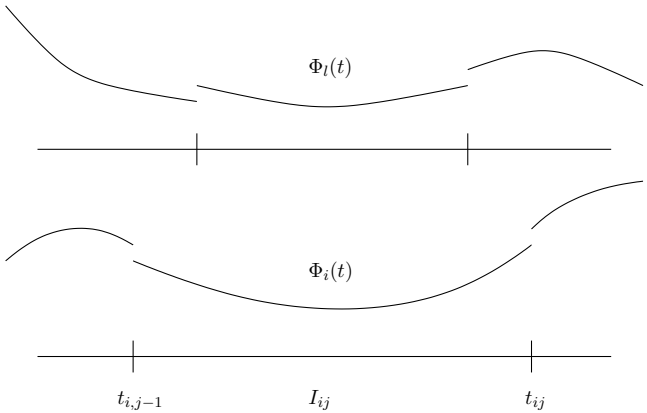

FIGURE 2. If some other component $l \neq i$ has a node within I_{ij}, then Φ_l may be discontinuous within I_{ij}, causing φ_i to be discontinuous within I_{ij}.

Lemma 4.1. (Interpolation estimates for φ) *Let φ be defined as in* (4.9)*. If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\pi_{\mathrm{cG}}^{[q_{ij}-2]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq C k_{ij}^{q_{ij}-1} \|f\|_{\mathcal{T}}^{q_{ij}} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 2, \tag{4.10}$$

and

$$\|\pi_{\mathrm{dG}}^{[q_{ij}-1]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq C k_{ij}^{q_{ij}} \|f\|_{\mathcal{T}}^{q_{ij}+1} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 1, \tag{4.11}$$

for each local interval I_{ij} within the time slab $\mathcal{T}$.

Proof. See [6]. □

4.2. **Linear problems.** For the linear problem (2.5), we make the following basic assumptions: Given a time slab $\mathcal{T}$, assume that for each pair of local intervals I_{ij} and I_{mn} within the time slab, we have

$$q_{ij} = q_{mn} = \bar{q}, \tag{B1}$$

and

$$k_{ij} > \alpha\, k_{mn}, \tag{B2}$$

for some $\bar{q} \geq 0$ and some $\alpha \in (0, 1)$. Furthermore, assume that A has $\bar{q} - 1$ continuous derivatives and let $C_A > 0$ be constant, such that

$$\max\left(\|A^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)}, \|A^{\top(p)}\|_{L_\infty(\mathcal{T},l_\infty)}\right) \leq C_A^{p+1}, \quad p = 0, \ldots, \bar{q}, \tag{B3}$$

for all time slabs $\mathcal{T}$. We further assume that there is a constant $c_k > 0$, such that

$$k_{ij} C_A \leq c_k. \tag{B4}$$

We summarize the list of assumptions as follows:

(B1) the local orders q_{ij} are equal within each time slab;
(B2) the local time steps k_{ij} are semi-uniform within each time slab;
(B3) A and its derivatives are bounded;
(B4) the local time steps k_{ij} are small.

As for the general non-linear problem, we need to estimate the interpolation error $\pi\varphi_i - \varphi_i$ on a local interval I_{ij}, where φ_i is now defined by

$$\varphi_i = (A^\top \Phi)_i = \sum_{l=1}^{N} A_{li} \Phi_l, \quad i = 1, \ldots, N. \tag{4.16}$$

Using assumptions (B1)–(B4), we obtain the following interpolation estimates for the function φ.

Lemma 4.2. (Interpolation estimates for φ) *Let φ be defined as in* (4.16)*. If assumptions* (B1)–(B4) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\pi_{\mathrm{cG}}^{[q_{ij}-2]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq C k_{ij}^{q_{ij}-1} C_A^{q_{ij}} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 2, \tag{4.17}$$

and

$$\|\pi_{\mathrm{dG}}^{[q_{ij}-1]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq C k_{ij}^{q_{ij}} C_A^{q_{ij}+1} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 1, \tag{4.18}$$

for each local interval I_{ij} within the time slab $\mathcal{T}$.

Proof. See [6]. □

5. A priori error estimates

Using the error representations derived in Section 3, the interpolation estimates of the previous section, and the stability estimates from [8], we now derive our main results: a priori error estimates for general order mcG(q) and mdG(q). The estimates are derived first for the general non-linear problem (1.1), then for the general linear problem (2.5), and finally for a parabolic model problem.

5.1. The general non-linear problem.

Theorem 5.1. (A priori error estimate for mcG(q)) *Let U be the* mcG(q) *solution of* (1.1)*, and let Φ be the corresponding* mcG(q)* *solution of the dual problem* (2.4)*. Then, there is a constant $C = C(q) > 0$, such that*

$$|L_{\psi,g}(\bar{e})| \leq C S(T) \|k^{q+1} \bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.1}$$

where $(k^{q+1}\bar{u}^{(q+1)})_i(t) = k_{ij}^{q_{ij}+1}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ *for* $t \in I_{ij}$*, and where the stability factor* $S(T)$ *is given by* $S(T) = \int_0^T \|J^\top(\pi_{\rm cG}^{[q]}u, u, \cdot)\Phi\|_{l_2}\,dt$*. Furthermore, if assumptions* (A1)–(A5) *hold and* $g = 0$ *in (2.4), then there is a constant* $C = C(q, c_k, \alpha) > 0$*, such that*

$$|L_{\psi,g}(\bar{e})| \le C\bar{S}(T)\|k^{2q}\bar{u}^{(2q)}\|_{L_\infty([0,T],l_1)}, \tag{5.2}$$

where $(k^{2q}\bar{u}^{(2q)})_i(t) = k_{ij}^{2q_{ij}}\|f\|_{\mathcal{T}}^{q_{ij}-1}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ *for* $t \in I_{ij}$*, and where the stability factor* $\bar{S}(T)$ *is given by*

$$\bar{S}(T) = \int_0^T \|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt = \sum_{n=1}^M K_n\|f\|_{\mathcal{T}_n}\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

Proof. By Corollary 3.1, we obtain

$$\begin{aligned}L_{\psi,g}(\bar{e}) &= \int_0^T (f(\pi_{\rm cG}^{[q]}u,\cdot) - f(u,\cdot),\Phi)\,dt = \int_0^T (J(\pi_{\rm cG}^{[q]}u,u,\cdot)(\pi_{\rm cG}^{[q]}u - u),\Phi)\,dt\\ &= \int_0^T (\pi_{\rm cG}^{[q]}u - u, J^\top(\pi_{\rm cG}^{[q]}u,u,\cdot)\Phi)\,dt.\end{aligned}$$

By Theorem 5.1 in [7], it now follows that

$$|L_{\psi,g}(\bar{e})| \le C\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}\int_0^T \|J^\top(\pi_{\rm cG}^{[q]}u,u,\cdot)\Phi\|_{l_2}\,dt,$$

which proves (5.1). To prove (5.2), we note that by definition, $\pi_{\rm cG}^{[q_{ij}]}u_i - u_i$ is orthogonal to $\mathcal{P}^{q_{ij}-2}(I_{ij})$ for each local interval I_{ij}, and so, recalling that $\varphi = J^\top(\pi_{\rm cG}^{[q]}u,u,\cdot)\Phi$,

$$\begin{aligned}L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N\sum_{j=1}^{M_i}\int_{I_{ij}}(\pi_{\rm cG}^{[q_{ij}]}u_i - u_i)\varphi_i\,dt\\ &= \sum_{i=1}^N\sum_{j=1}^{M_i}\int_{I_{ij}}(\pi_{\rm cG}^{[q_{ij}]}u_i - u_i)(\varphi_i - \pi_{\rm cG}^{[q_{ij}-2]}\varphi_i)\,dt,\end{aligned}$$

where we take $\pi_{\rm cG}^{[q_{ij}-2]}\varphi_i \equiv 0$ for $q_{ij} = 1$. By Theorem 5.1 in [7] and Lemma 4.1, it now follows that

$$\begin{aligned}|L_{\psi,g}(\bar{e})| &\le \int_0^T |(\pi_{\rm cG}^{[q]}u - u, \varphi - \pi_{\rm cG}^{[q-2]}\varphi)|\,dt\\ &= \int_0^T |(k^{q-1}\|f\|_{\mathcal{T}}^{q-1}(\pi_{\rm cG}^{[q]}u - u), k^{-(q-1)}\|f\|_{\mathcal{T}}^{-(q-1)}(\varphi - \pi_{\rm cG}^{[q-2]}\varphi))|\,dt\\ &\le C\|k^{2q}\bar{u}^{(2q)}\|_{L_\infty([0,T],l_1)}\int_0^T \|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt\\ &= C\bar{S}(T)\|k^{2q}\bar{u}^{(2q)}\|_{L_\infty([0,T],l_1)},\end{aligned}$$

where the stability factor $\bar{S}(T)$ is given by

$$\bar{S}(T) = \int_0^T \|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt = \sum_{n=1}^M K_n\|f\|_{\mathcal{T}_n}\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

□

Theorem 5.2. (A priori error estimate for mdG(q)) *Let* U *be the* mdG(q) *solution of* (1.1)*, and let* Φ *be the corresponding* mdG(q)* *solution of the dual problem* (2.4)*. Then, there is a constant* $C = C(q) > 0$*, such that*

$$|L_{\psi,g}(\bar{e})| \le CS(T)\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.3}$$

where $(k^{q+1}\bar{u}^{(q+1)})_i(t) = k_{ij}^{q_{ij}+1}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ *for* $t \in I_{ij}$*, and where the stability factor* $S(T)$ *is given by* $S(T) = \int_0^T \|J^\top(\pi_{\rm dG}^{[q]}u,u,\cdot)\Phi\|_{l_2}\,dt$*. Furthermore, if assumptions* (A1)–(A5) *hold and* $g = 0$ *in (2.4), then there is a constant* $C = C(q, c_k, \alpha) > 0$*, such that*

$$|L_{\psi,g}(\bar{e})| \le C\bar{S}(T)\|k^{2q+1}\bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)}, \tag{5.4}$$

where $(k^{2q+1}\bar{\bar{u}}^{(2q+1)})_i(t) = k_{ij}^{2q_{ij}+1}\|f\|_{\mathcal{T}}^{q_{ij}}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ *for* $t \in I_{ij}$*, and where the stability factor* $\bar{S}(T)$ *is given by*

$$\bar{S}(T) = \int_0^T \|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt = \sum_{n=1}^M K_n\|f\|_{\mathcal{T}_n}\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

Proof. The estimate (5.3) is obtained in the same way as we obtained the estimate (5.1). To prove (5.4), we note that as in the proof of Theorem 5.1, we obtain $L_{\psi,g}(\bar{e}) = \int_0^T(\pi_{\rm dG}^{[q]}u - u,\varphi)\,dt$. By definition, $\pi_{\rm dG}^{[q_{ij}]}u_i - u_i$ is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$ for each local interval I_{ij}, and so

$$\begin{aligned}L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N\sum_{j=1}^{M_i}\int_{I_{ij}}(\pi_{\rm dG}^{[q_{ij}]}u_i - u_i)\varphi_i\,dt\\ &= \sum_{i=1}^N\sum_{j=1}^{M_i}\int_{I_{ij}}(\pi_{\rm dG}^{[q_{ij}]}u_i - u_i)(\varphi_i - \pi_{\rm dG}^{[q_{ij}-1]}\varphi_i)\,dt,\end{aligned}$$

where we take $\pi_{\rm dG}^{[q_{ij}-1]}\varphi_i = 0$ for $q_{ij} = 0$. By Theorem 5.1 in [7] and Lemma 4.1, it now follows that

$$\begin{aligned}|L_{\psi,g}(\bar{e})| &\le \int_0^T |(\pi_{\rm dG}^{[q]}u - u, \varphi - \pi_{\rm dG}^{[q-1]}\varphi)|\,dt\\ &= \int_0^T |(k^q\|f\|_{\mathcal{T}}^q(\pi_{\rm dG}^{[q]}u - u), k^{-q}\|f\|_{\mathcal{T}}^{-q}(\varphi - \pi_{\rm dG}^{[q-1]}\varphi))|\,dt\\ &\le C\|k^{2q+1}\bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)}\int_0^T \|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt.\end{aligned}$$

□

Using the stability estimates proved in [8], we obtain the following bound for the stability factor $\bar{S}(T)$.

Lemma 5.1. *Assume that $K_n C_q \|f\|_{\mathcal{T}_n} \leq 1$ for all time slabs $\mathcal{T}_n$, with $C_q = C_q(q) > 0$ the constant in Theorem 4.1 of [8], and take $g = 0$. Then,*

$$\bar{S}(T) \leq \|\psi\|_{l_\infty} e^{C_q \|f\|_{[0,T]} T}, \tag{5.5}$$

where $\|f\|_{[0,T]} = \max_{n=1,\ldots,M} \|f\|_{\mathcal{T}_n}$.

Proof. By Theorem 4.1 in [8], we obtain

$$\begin{aligned}\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)} &\leq C_q \|\psi\|_{l_\infty} \exp\left(\sum_{m=n+1}^{M} K_m C_q \|f\|_{\mathcal{T}_m}\right)\\ &\leq C_q \|\psi\|_{l_\infty} e^{C_q \|f\|_{[0,T]}(T-T_n)},\end{aligned}$$

and so

$$\begin{aligned}\bar{S}(T) &= \sum_{n=1}^{M} K_n \|f\|_{\mathcal{T}_n} \|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}\, dt\\ &\leq \|\psi\|_{l_\infty} \sum_{n=1}^{M} K_n C_q \|f\|_{[0,T]} e^{C_q \|f\|_{[0,T]}(T-T_n)}\\ &\leq \|\psi\|_{l_\infty} \int_0^T C_q \|f\|_{[0,T]} e^{C_q \|f\|_{[0,T]} t}\, dt \leq \|\psi\|_{l_\infty} e^{C_q \|f\|_{[0,T]} T}.\end{aligned}$$

□

Finally, we rewrite the estimates 5.1 and 5.2 for special choices of data ψ and g. We first take $\psi = 0$. With $g_n = 0$ for $n \neq i$, $g_i(t) = 0$ for $t \notin I_{ij}$, and

$$g_i(t) = \operatorname{sgn}(\bar{e}_i(t))/k_{ij}, \quad t \in I_{ij},$$

we obtain $L_{\psi,g}(\bar{e}) = \frac{1}{k_{ij}} \int_{I_{ij}} |\bar{e}_i(t)|\, dt$ and so $\|\bar{e}_i\|_{L_\infty(I_{ij})} \leq C L_{\psi,g}(\bar{e})$ by an inverse estimate. By the definition of $\bar{e}$, it follows that $\|e_i\|_{L_\infty(I_{ij})} \leq C L_{\psi,g}(\bar{e}) + C k_{ij}^{q_{ij}+1} \|u_i^{q_{ij}+1}\|_{L_\infty(I_{ij})}$. Note that for this choice of g, we have $\|g\|_{L_1([0,T],l_2)} = \|g\|_{L_1([0,T],l_\infty)} = 1$.

We also make the choice $g = 0$. Noting that $\bar{e}(T) = e(T)$, since $\pi u(T) = u(T)$, we obtain

$$L_{\psi,g}(\bar{e}) = (e(T), \psi) = |e_i(T)|,$$

for $\psi_i = \operatorname{sgn}(e_i(T))$ and $\psi_n = 0$ for $n \neq i$, and

$$L_{\psi,g}(\bar{e}) = (e(T), \psi) = \|e(T)\|_{l_2},$$

for $\psi = e(T)/\|e(T)\|_{l_2}$. Note that for both choices of ψ, we have $\|\psi\|_{l_\infty} \leq 1$.

With these choices of data, we obtain the following versions of the a priori error estimates.

Corollary 5.1. (A priori error estimate for mcG(q)) *Let U be the* mcG(q) *solution of* (1.1). *Then, there is a constant $C = C(q) > 0$, such that*

$$\|e\|_{L_\infty([0,T],l_\infty)} \leq C S(T) \|k^{q+1} \bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.6}$$

where the stability factor $S(T) = \int_0^T \|J^\top(\pi_{\mathrm{cG}}^{[q]} u, u, \cdot)\Phi\|_{l_2}\, dt$ is taken as the maximum over $\psi = 0$ and $\|g\|_{L_1([0,T],l_\infty)} = 1$. Furthermore, if assumptions (A1)–(A5) *and the assumptions of Lemma 5.1 hold, then there is a constant $C = C(q, c_k, \alpha)$, such that*

$$\|e(T)\|_{l_p} \leq C \bar{S}(T) \|k^{2q} \bar{\bar{u}}^{(2q)}\|_{L_\infty([0,T],l_1)}, \tag{5.7}$$

for $p = 2, \infty$, where the stability factor $\bar{S}(T)$ is given by $\bar{S}(T) = e^{C_q \|f\|_{[0,T]} T}$.

Corollary 5.2. (A priori error estimate for mdG(q)) *Let U be the* mdG(q) *solution of* (1.1). *Then, there is a constant $C = C(q) > 0$, such that*

$$\|e\|_{L_\infty([0,T],l_\infty)} \leq C S(T) \|k^{q+1} \bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.8}$$

where the stability factor $S(T) = \int_0^T \|J^\top(\pi_{\mathrm{dG}}^{[q]} u, u, \cdot)\Phi\|_{l_2}\, dt$ is taken as the maximum over $\psi = 0$ and $\|g\|_{L_1([0,T],l_\infty)} = 1$. Furthermore, if assumptions (A1)–(A5) *and the assumptions of Lemma 5.1 hold, then there is a constant $C = C(q, c_k, \alpha)$, such that*

$$\|e(T)\|_{l_p} \leq C \bar{S}(T) \|k^{2q+1} \bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)}, \tag{5.9}$$

for $p = 2, \infty$, where the stability factor $\bar{S}(T)$ is given by $\bar{S}(T) = e^{C_q \|f\|_{[0,T]} T}$.

5.2. **Linear problems.**

Theorem 5.3. (A priori error estimate for mcG(q)) *Let U be the* mcG(q) *solution of* (2.5), *and let Φ be the corresponding* mcG(q)* *solution of the dual problem* (2.6). *Then, there is a constant $C = C(q) > 0$, such that*

$$|L_{\psi,g}(\bar{e})| \leq C S(T) \|k^{q+1} \bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.10}$$

where $(k^{q+1}\bar{u}^{(q+1)})_i(t) = k_{ij}^{q_{ij}+1} \|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ for $t \in I_{ij}$, and where the stability factor $S(T)$ is given by $S(T) = \int_0^T \|A^\top \Phi\|_{l_2}\, dt$. Furthermore, if assumptions (B1)–(B4) *hold and $g = 0$ in (2.6), then there is a constant $C = C(q, c_k, \alpha) > 0$, such that*

$$|L_{\psi,g}(\bar{e})| \leq C \bar{S}(T) \|k^{2q} \bar{\bar{u}}^{(2q)}\|_{L_\infty([0,T],l_1)}, \tag{5.11}$$

where $(k^{2q}\bar{\bar{u}}^{(2q)})_i(t) = k_{ij}^{2q_{ij}} C_A^{q_{ij}-1} \|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ for $t \in I_{ij}$, and where the stability factor $\bar{S}(T)$ is given by

$$\bar{S}(T) = \int_0^T C_A \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\, dt = \sum_{n=1}^{M} K_n C_A \|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

Proof. By Corollary 3.1, we obtain

$$L_{\psi,g}(\bar{e}) = \int_0^T (A(u - \pi_{\rm cG}^{[q]}u), \Phi)\,dt = \int_0^T (u - \pi_{\rm cG}^{[q]}, A^\top\Phi)\,dt.$$

By Theorem 5.1 in [7], it now follows that

$$|L_{\psi,g}(\bar{e})| \le C\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)} \int_0^T \|A^\top\Phi\|_{l_2}\,dt,$$

which proves (5.10). To prove (5.11), we note that by definition, $\pi_{\rm cG}^{[q_{ij}]}u_i - u_i$ is orthogonal to $\mathcal{P}^{q_{ij}-2}(I_{ij})$ for each local interval I_{ij}, and so

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (u_i - \pi_{\rm cG}^{[q_{ij}]}u_i)\varphi_i\,dt \\ &= \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (u_i - \pi_{\rm cG}^{[q_{ij}]}u_i)(\varphi_i - \pi_{\rm cG}^{[q_{ij}-2]}\varphi_i)\,dt, \end{aligned}$$

where $\varphi = A^\top\Phi$. By Theorem 5.1 in [7] and Lemma 4.2, it now follows that

$$\begin{aligned} |L_{\psi,g}(\bar{e})| &\le \int_0^T |(\pi_{\rm cG}^{[q]}u - u, \varphi - \pi_{\rm cG}^{[q-2]}\varphi)|\,dt \\ &= \int_0^T |(k^{q-1}C_A^{q-1}(\pi_{\rm cG}^{[q]}u - u), k^{-(q-1)}C_A^{-(q-1)}(\varphi - \pi_{\rm cG}^{[q-2]}\varphi))|\,dt \\ &\le C\|k^{2q}\bar{\bar{u}}^{(2q)}\|_{L_\infty([0,T],l_1)} \int_0^T C_A\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt \\ &= C\bar{S}(T)\|k^{2q}\bar{\bar{u}}^{(2q)}\|_{L_\infty([0,T],l_1)}, \end{aligned}$$

where the stability factor $\bar{S}(T)$ is given by

$$\bar{S}(T) = \int_0^T C_A\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt = \sum_{n=1}^M K_n C_A\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

□

Theorem 5.4. (A priori error estimate for mdG(q)) *Let U be the* mdG(q) *solution of* (2.5)*, and let Φ be the corresponding* mdG(q)* *solution of the dual problem* (2.6)*. Then, there is a constant $C = C(q) > 0$, such that*

$$|L_{\psi,g}(\bar{e})| \le CS(T)\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.12}$$

where $(k^{q+1}\bar{u}^{(q+1)})_i(t) = k_{ij}^{q_{ij}+1}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ for $t \in I_{ij}$, and where the stability factor $S(T)$ is given by $S(T) = \int_0^T \|A^\top\Phi\|_{l_2}\,dt$. Furthermore, if assumptions (B1)–(B4) *hold and $g = 0$ in (2.6), then there is a constant $C = C(q, c_k, \alpha) > 0$, such that*

$$|L_{\psi,g}(\bar{e})| \le C\bar{S}(T)\|k^{2q+1}\bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)}, \tag{5.13}$$

where $(k^{2q+1}\bar{\bar{u}}^{(2q+1)})_i(t) = k_{ij}^{2q_{ij}+1}C_A^{q_{ij}}\|u_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ for $t \in I_{ij}$, and where the stability factor $\bar{S}(T)$ is given by

$$\bar{S}(T) = \int_0^T C_A\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt = \sum_{n=1}^M K_n C_A\|\Phi\|_{L_\infty(\mathcal{T}_n,l_\infty)}.$$

Proof. The estimate (5.12) is obtained in the same way as we obtained the estimate (5.10). To prove (5.13), we note that as in the proof of Theorem 5.1, we obtain $L_{\psi,g}(\bar{e}) = \int_0^T (u - \pi_{\rm dG}^{[q]}u, \varphi)\,dt$. By definition, $\pi_{\rm dG}^{[q_{ij}]}u_i - u_i$ is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$ for each local interval I_{ij}, and so

$$\begin{aligned} L_{\psi,g}(\bar{e}) &= \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (u_i - \pi_{\rm dG}^{[q_{ij}]}u_i)\varphi_i\,dt \\ &= \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (u_i - \pi_{\rm dG}^{[q_{ij}]}u_i)(\varphi_i - \pi_{\rm dG}^{[q_{ij}-1]}\varphi_i)\,dt. \end{aligned}$$

By Theorem 5.1 in [7] and Lemma 4.2, it now follows that

$$\begin{aligned} |L_{\psi,g}(\bar{e})| &\le \int_0^T |(\pi_{\rm dG}^{[q]}u - u, \varphi - \pi_{\rm dG}^{[q-1]}\varphi)|\,dt \\ &= \int_0^T |(k^qC_A^q(\pi_{\rm dG}^{[q]}u - u), k^{-q}C_A^{-q}(\varphi - \pi_{\rm dG}^{[q-1]}\varphi))|\,dt \\ &\le C\|k^{2q+1}\bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)} \int_0^T C_A\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\,dt. \end{aligned}$$

□

We now use Lemma 5.1 to obtain a bound for the stability factor $\bar{S}(T)$. As for the non-linear problem, we note that for special choices of data ψ and g for the dual problem, we obtain error estimates in various norms, in particular the l_2-norm at final time.

Corollary 5.3. (A priori error estimate for mcG(q)) *Let U be the* mcG(q) *solution of* (2.5)*. Then, there is a constant $C = C(q) > 0$, such that*

$$\|e\|_{L_\infty([0,T],l_\infty)} \le CS(T)\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.14}$$

where the stability factor $S(T) = \int_0^T \|A^\top\Phi\|_{l_2}\,dt$ is taken as the maximum over $\psi = 0$ and $\|g\|_{L_1([0,T],l_\infty)} = 1$. Furthermore, if assumptions (B1)–(B4) *and the assumptions of Lemma 5.1 hold, then there is a constant $C = C(q, c_k, \alpha)$, such that*

$$\|e(T)\|_{l_p} \le C\bar{S}(T)\|k^{2q}\bar{\bar{u}}^{(2q)}\|_{L_\infty([0,T],l_1)}, \tag{5.15}$$

for $p = 2, \infty$, where the stability factor $\bar{S}(T)$ is given by $\bar{S}(T) = e^{C_qC_AT}$.

Corollary 5.4. (A priori error estimate for mdG(q)) *Let U be the* mdG(q) *solution of* (2.5). *Then, there is a constant $C = C(q) > 0$, such that*

$$\|e\|_{L_\infty([0,T],l_\infty)} \le CS(T)\|k^{q+1}\bar{u}^{q+1}\|_{L_\infty([0,T],l_2)}, \tag{5.16}$$

where the stability factor $S(T) = \int_0^T \|A^\top \Phi\|_{l_2}\,dt$ is taken as the maximum over $\psi = 0$ and $\|g\|_{L_1([0,T],l_\infty)} = 1$. Furthermore, if assumptions (B1)–(B4) *and the assumptions of Lemma 5.1 hold, then there is a constant $C = C(q, c_k, \alpha)$, such that*

$$\|e(T)\|_{l_p} \le C\bar{S}(T)\|k^{2q+1}\bar{\bar{u}}^{(2q+1)}\|_{L_\infty([0,T],l_1)}, \tag{5.17}$$

for $p = 2, \infty$, where the stability factor $\bar{S}(T)$ is given by $\bar{S}(T) = e^{C_q C_A T}$.

5.3. **Parabolic problems.** We consider the linear parabolic model problem,

$$\begin{aligned} \dot{u}(t) + A(t)u(t) &= 0, \quad t \in (0,T], \\ u(0) &= u_0, \end{aligned} \tag{5.18}$$

with A a positive semidefinite and symmetric $N \times N$ matrix for all $t \in (0,T]$, and prove an a priori error estimate for the mdG(q) method. The estimate is based on the error representation for the mdG(q) method presented in Section 3 and the strong stability estimate derived in [8]. To prove the a priori error estimate, we need to make the following assumptions. We first assume that q is constant within each time slab, that is, for each pair of intervals I_{ij} and I_{mn} within a given time slab, we have as before

$$q_{ij} = q_{mn} = \bar{q}. \tag{C1}$$

Furthermore, we assume that A is invertible on $(0,T)$ for $q \ge 2$ and refer to this as assumption (C2).

Additional assumptions, (C3)–(C6), are needed for the strong stability estimate of [8]. We first assume that there is a constant $\gamma \ge 1$, such that

(C3)
$$(T - T_n)\int_{T_{n-1}}^{T_n} (Av, Av)\,dt \le \gamma \int_{T_{n-1}}^{T_n} (Av, \pi(\bar{t}Av))\,dt, \quad n = 1, \ldots, M-1,$$

for all trial functions v, that is, all v discontinuous and piecewise polynomial with $v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$, where $\bar{t} = \bar{t}(t)$ is the piecewise constant right-continuous function defined by $\bar{t}(t) = \min_{ij}\{T - t_{i,j-1} : t \ge t_{i,j-1}\}$. If Av is not close to being orthogonal to the trial space, then γ is of moderate size. We also assume that there is a constant $\sigma > 1$, such that

$$\sigma K_n \le (T - T_n), \quad n = 1, \ldots, M-1. \tag{C4}$$

This condition corresponds to the condition $\sigma K_n \le T_{n-1}$ used in the strong stability estimate for the discrete dual problem in [8]. We further assume that all components use the same time step on the last time slab $\mathcal{T}_M$,

$$k_{ij} = K_M \quad \forall\, I_{ij} \in \mathcal{T}_M. \tag{C5}$$

Finally, we assume that A is constant and refer to this as assumption (C6).

Theorem 5.5. (A priori error estimate for parabolic problems) *Let U be the* mdG(q) *solution of* (5.18), *and assume that* (C1) *and* (C2) *hold. Then, there is a constant $C = C(q)$, such that*

$$\|e(T)\|_{l_2} \le CS(T)\max_{[0,T]}\|k^{2q+1}A^q\bar{u}^{(q+1)}\|_{l_2} + \mathcal{E}, \tag{5.23}$$

where $(k^{2q+1}A^q\bar{u}^{(q+1)})_i(t) = k_{ij}^{2q_{ij}+1}\|(A^{q_{ij}}u)_i^{(q_{ij}+1)}\|_{L_\infty(I_{ij})}$ for $t \in I_{ij}$, $\mathcal{E} = 0$ for $q = 0$, and $\mathcal{E} = \int_0^T (\pi_{\mathrm{dG}}^{[q]}A^q u - A^q\pi_{\mathrm{dG}}^{[q]}u, A^{1-q}\Phi)\,dt$ for $q > 0$. The stability factor $S(T)$ is given by

$$S(T) = \int_0^T \|k^{-q}(A^{1-q}\Phi - \pi_{\mathrm{dG}}^{[q-1]}A^{1-q}\Phi)\|_{l_2}\,dt. \tag{5.24}$$

For $q = 0, 1$, we obtain the following analytical bound for $S(T)$, using assumptions (C3)–(C6),

$$S(T) \le C\left(\log\frac{T}{K_M} + 1\right)^{1/2}, \tag{5.25}$$

where $C = C(q, \gamma, \sigma) > 0$.

Proof. With $\psi = e(T)/\|e(T)\|_{l_2}$ and $g = 0$, it follows by Corollary 3.2 that

$$\begin{aligned} \|e(T)\|_{l_2} &= \int_0^T (u - \pi_{\mathrm{dG}}^{[q]}u, A\Phi)\,dt = \int_0^T (A^q(u - \pi_{\mathrm{dG}}^{[q]}u), A^{1-q}\Phi)\,dt \\ &= \int_0^T (A^q u - \pi_{\mathrm{dG}}^{[q]}A^q u + \pi_{\mathrm{dG}}^{[q]}A^q u - A^q\pi_{\mathrm{dG}}^{[q]}u, A^{1-q}\Phi)\,dt \\ &= \int_0^T (A^q u - \pi_{\mathrm{dG}}^{[q]}A^q u, A^{1-q}\Phi)\,dt \\ &\quad + \int_0^T (\pi_{\mathrm{dG}}^{[q]}A^q u - A^q\pi_{\mathrm{dG}}^{[q]}u, A^{1-q}\Phi)\,dt \\ &= \int_0^T (A^q u - \pi_{\mathrm{dG}}^{[q]}A^q u, A^{1-q}\Phi - \pi_{\mathrm{dG}}^{[q-1]}A^{1-q}\Phi)\,dt + \mathcal{E}, \end{aligned}$$

where we have assumed that A is invertible for $q \ge 2$. With $S(T) = \int_0^T \|k^{-q}(A^{1-q}\Phi - \pi_{\mathrm{dG}}^{[q-1]}A^{1-q}\Phi)\|_{l_2}\,dt$, we thus obtain

$$\begin{aligned} \|e(T)\|_{l_2} &\le S(T)\max_{[0,T]}\|k^q(A^q u - \pi_{\mathrm{dG}}^{[q]}A^q u)\|_{l_2} + \mathcal{E} \\ &= S(T)\max_{[0,T]}\left(\sum_{i=1}^N [k_i^{q_i}((A^q u)_i - \pi_{\mathrm{dG}}^{[q_i]}(A^q u)_i)]^2\right)^{1/2} + \mathcal{E} \\ &\le CS(T)\max_{t\in[0,T]}\left(\sum_{i=1}^N [k_i^{2q_i(t)+1}(t)\|(A^q u)_i^{(q_i(t)+1)}\|_{L_\infty(I_{ij}(t))}]^2\right)^{1/2} + \mathcal{E} \\ &= CS(T)\max_{[0,T]}\|k^{2q+1}A^q\bar{u}^{(q+1)}\|_{l_2} + \mathcal{E}. \end{aligned}$$

Note that we use an interpolation estimate for $A^q u$ which is straightforward since the exact solution u is smooth. We conclude by estimating the stability factor $S(T)$ for $q = 0, 1$, using the strong stability estimate for the discrete dual solution Φ. For $q = 0$, it follows directly by Theorem 4.3 in [8], that

$$S(T) = \int_0^T \|A\Phi\|_{l_2}\, dt \le C \left(\log \frac{T}{K_M} + 1\right)^{1/2},$$

and for $q = 1$, we obtain

$$S(T) = \int_0^T \|k^{-1}(\Phi - \pi_{\mathrm{dG}}^{[0]}\Phi)\|_{l_2}\, dt \le C \int_0^T \|\dot{\Phi}\|_{l_2}\, dt,$$

using an interpolation estimate in combination with an inverse estimate, and so the estimate $S(T) \le C \left(\log \frac{T}{K_M} + 1\right)^{1/2}$ follows again by Theorem 4.3 in [8].

□

The stability factor $S(T)$ that appears in the a priori error estimate is obtained from the discrete solution Φ of the dual problem (2.6), and can thus be computed exactly by solving the discrete dual problem. Allowing numerical computation of the stability factor, the additional assumptions (C3)–(C6) needed to obtain the analytical bound for $S(T)$ are no longer needed. Numerical computation of the stability factor also directly reveals whether the problem is parabolic or not; if the stability factor is of unit size and does not grow, then the problem is parabolic by definition, see [2].

6. Numerical examples

We conclude by demonstrating the convergence of the multi-adaptive methods in the case of a simple test problem. We also present some results in support of assumption (C3).

6.1. **Convergence.** Consider the problem

$$\begin{aligned}
\dot{u}_1 &= u_2, \\
\dot{u}_2 &= -u_1, \\
\dot{u}_3 &= -u_2 + 2u_4, \\
\dot{u}_4 &= u_1 - 2u_3, \\
\dot{u}_5 &= -u_2 - 2u_4 + 4u_6, \\
\dot{u}_6 &= u_1 + 2u_3 - 4u_5,
\end{aligned} \tag{6.1}$$

on $[0, 1]$ with initial condition $u(0) = (0, 1, 0, 2, 0, 3)$. The solution is given by $u(t) = (\sin t, \cos t, \sin t + \sin 2t, \cos t + \cos 2t, \sin t + \sin 2t + \sin 4t, \cos t + \cos 2t + \cos 4t)$. For given $k_0 > 0$, we take $k_i(t) = k_0$ for $i = 1, 2$, $k_i(t) = k_0/2$ for $i = 3, 4$, and $k_i(t) = k_0/4$ for $i = 5, 6$, and study the convergence of the error $\|e(T)\|_{l_2}$ with decreasing k_0. From the results presented in Figure 3,

Table 1, and Table 2, it is clear that the predicted order of convergence is obtained, both for mcG(q) and mdG(q).

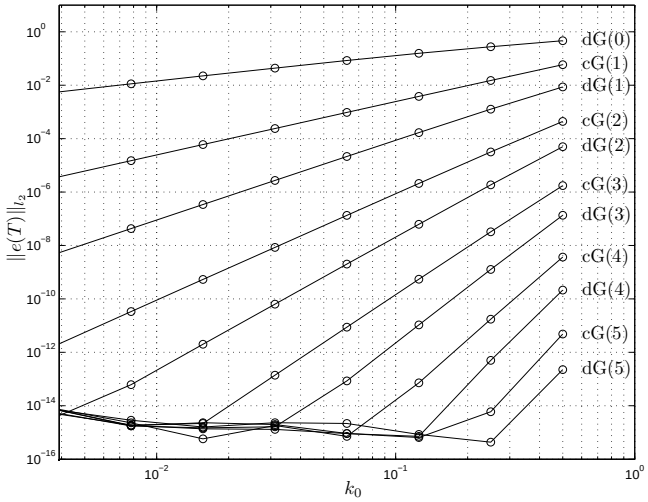

FIGURE 3. Convergence of the error at final time for the solution of the test problem (6.1) with mcG(q) and mdG(q), $q \le 5$.

mcG(q)	1	2	3	4	5
p	1.99	3.96	5.92	7.82	9.67
$2q$	2	4	6	8	10

TABLE 1. Order of convergence p for mcG(q).

mdG(q)	0	1	2	3	4	5
p	0.92	2.96	4.94	6.87	9.10	–
$2q+1$	1	3	5	7	9	11

TABLE 2. Order of convergence p for mdG(q).

6.2. **Numerical evidence for assumption (C3).** The strong stability estimate Theorem 4.3 in [8], which is used in the proof of Theorem 5.5, relies on assumption (C3), which for the dual problem (with time reversed) can be stated in the form

$$T_{n-1}\int_{T_{n-1}}^{T_n}(Av,Av)\,dt \le \gamma\int_{T_{n-1}}^{T_n}(Av,\pi(\bar{t}Av))\,dt, \quad n=2,\ldots,M, \tag{6.2}$$

where $\bar{t}=\bar{t}(t)$ is the piecewise constant left-continuous function defined by $\bar{t}(t)=\min_{ij}\{t_{ij} : t\le t_{ij}\}$. As mentioned, this may fail to hold if Av is close to orthogonal to the trial space. On the other hand, if every pair of components which are coupled through A use approximately the same step size, then $\pi(Av)\approx Av$ and (6.2) holds. We illustrate this in the case of the mdG(0) method, where interpolation is given by taking the right end-point value within each local interval, for A given by

$$A=\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}.$$

We take $k_i(t)=1/i$ for $i=1,\ldots,10$ on $[0,1]$, and randomize the piecewise constant function v on this partition. Examining the quotient

$$\gamma=\max_v \frac{C\int_0^1(Av,Av)\,dt}{\int_0^1(Av,\pi(\bar{t}Av))\,dt},$$

with $\bar{t}(t)=\min_{ij}\{C+t_{ij} : t\le t_{ij}\}$ for C large, we find $\gamma\lessapprox 1.5$. Here, C corresponds to T_{n-1} in (6.2). In Figure 4, we present an example in which $C=100$ and $\gamma=1.05$.

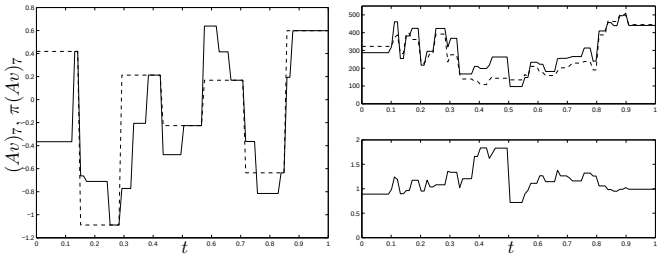

FIGURE 4. Left, we plot a component of the function Av (solid) and its interpolant $\pi(Av)$ (dashed) for the partition discussed in the text. Above right, we plot $C(Av,Av)$ (solid) and $(Av,\pi(\bar{t}Av))$ (dashed), and below right, we plot the corresponding quotient $C(Av,Av)/(Av,\pi(\bar{t}Av))$.

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] K. ERIKSSON, C. JOHNSON, AND A. LOGG, *Adaptive computational methods for parabolic problems*, To appear in Encyclopedia of Computational Mechanics, (2004).

[3] 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 (1988), pp. 908–926.

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

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

[6] ———, *Estimates of derivatives and jumps across element boundaries for multi-adaptive Galerkin solutions of ODEs*, Tech. Rep. 2004–03, Chalmers Finite Element Center Preprint Series, 2004.

[7] ———, *Interpolation estimates for piecewise smooth functions in one dimension*, Tech. Rep. 2004–02, Chalmers Finite Element Center Preprint Series, 2004.

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

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.

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 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*:

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

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

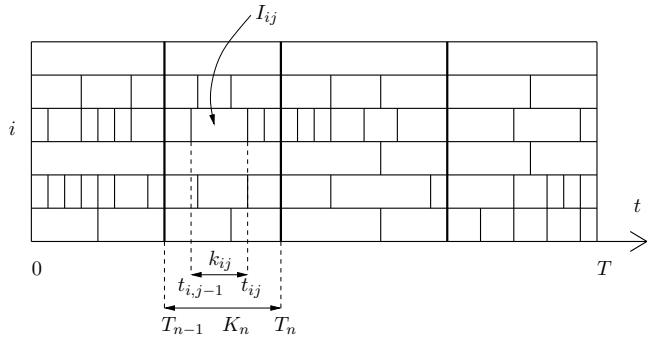

FIGURE 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

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].

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 $\alpha = \text{diag}(\alpha_1, \ldots, \alpha_N)$ 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, 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:

$$\exists\, V = \begin{bmatrix} v^1 \cdots v^N \end{bmatrix} \text{ non-singular} : V^{-1}BV = \text{diag}(\lambda_1, \ldots, \lambda_N),$$

with $\lambda_1, \ldots, \lambda_N$ the eigenvalues of B;
(2) $\text{Re}\,\lambda_i > -1$ for all λ_i;
(3) $|\text{Im}\,\lambda_i|/(1+\text{Re}\,\lambda_i) \leq \tan\beta$, 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.

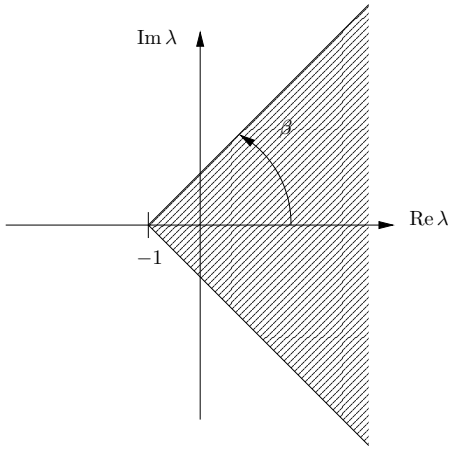

FIGURE 2. The eigenvalues λ of the matrix B are assumed to lie within the shaded sector.

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

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

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

$$\begin{aligned} |\sigma_i|^2 &= (\text{Re}\,\sigma_i)^2 + (\text{Im}\,\sigma_i)^2 = (1 - \alpha(1 + \text{Re}\,\lambda_i))^2 + \alpha^2(\text{Im}\,\lambda_i)^2 \\ &= 1 + \alpha^2\tilde{r}_i^2 - 2\alpha(1 + \text{Re}\,\lambda_i), \end{aligned}$$

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

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

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

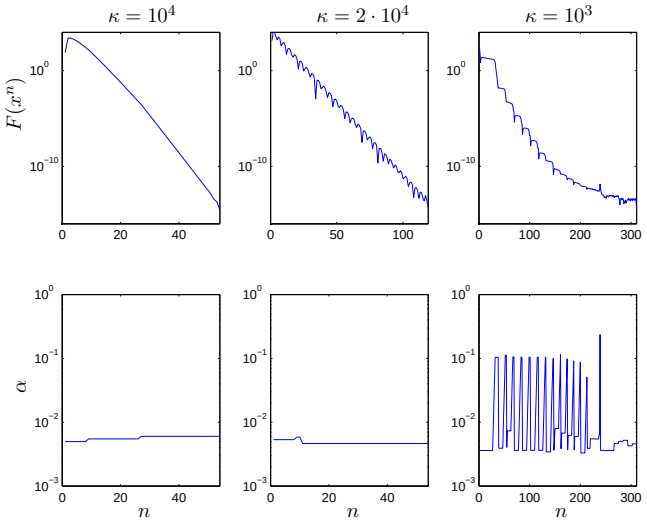

FIGURE 3. Convergence of the stabilized fixed point iteration for the solution of (3.14) with $\kappa = 10^4$, $\kappa = 2 \cdot 10^4$, and $\kappa = 10^3$.

$G_\alpha = I - \alpha(I+B)$, 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 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 $T_{n-1} \le t_{i,j-1} < t_{ij} \le T_n$.

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.

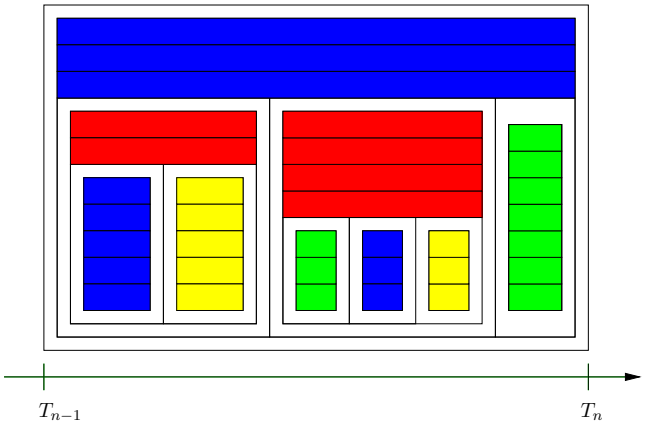

FIGURE 4. The recursive organization of the time slab. Each time slab contains an element group (shaded) and a list of recursively nested time slabs. The root time slab in the figure contains one element group of three elements and three time slabs. The first of these sub slabs contains an element group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

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

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.

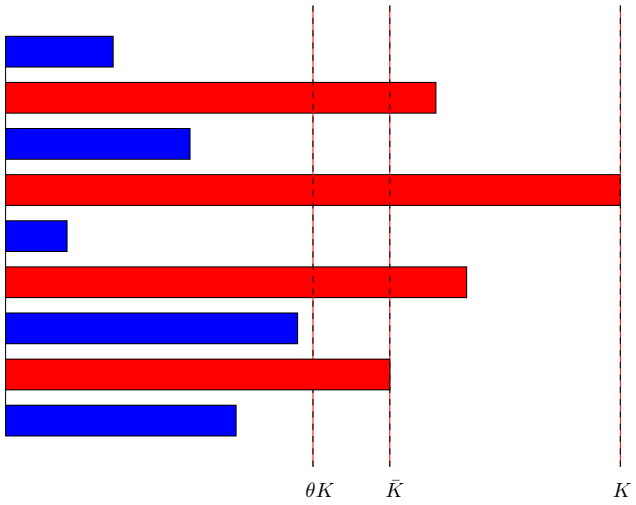

FIGURE 5. The partition of components into groups of small and large time steps for $\theta = 1/2$.

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

The last component steps first. (4.4)

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.

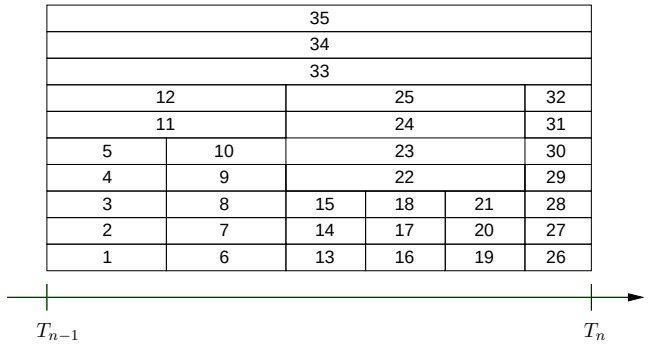

FIGURE 6. Numbering of the elements in the order in which they are created.

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 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*,

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**
	Update(element)
Iterate(element)	**for each** degree of freedom **do**
while *condition 1* **do**	Update(degree of freedom)
Update(element)	**end for**
end while	

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

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.

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).

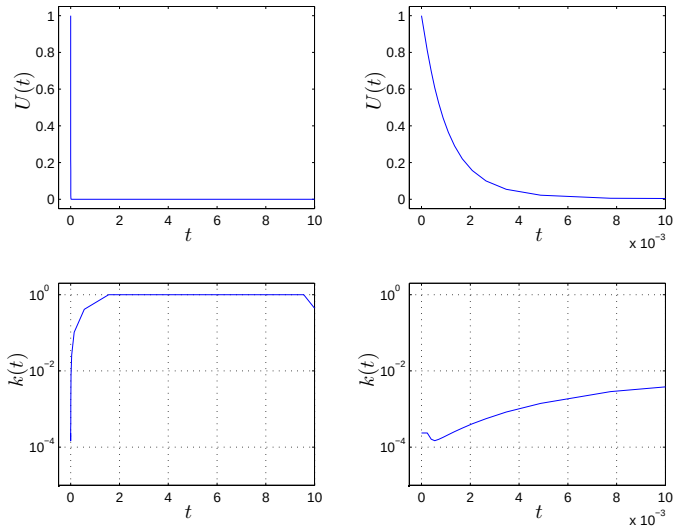

FIGURE 7. Solution and adaptive time step sequence for (5.1).

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).

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.

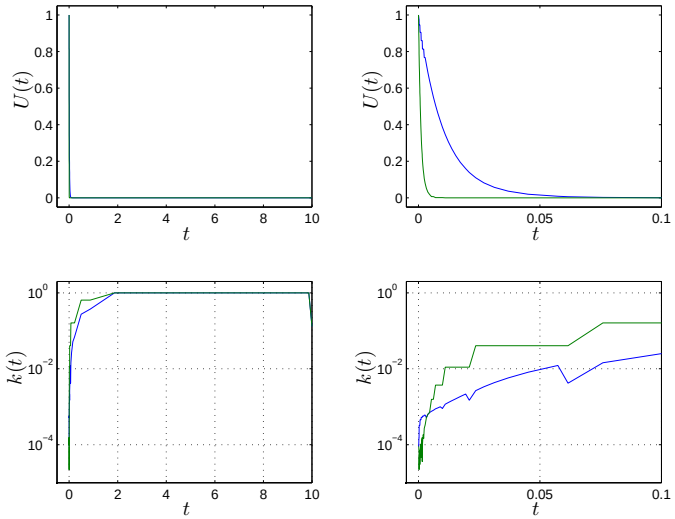

FIGURE 8. Solution and adaptive time step sequence for (5.2).

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.

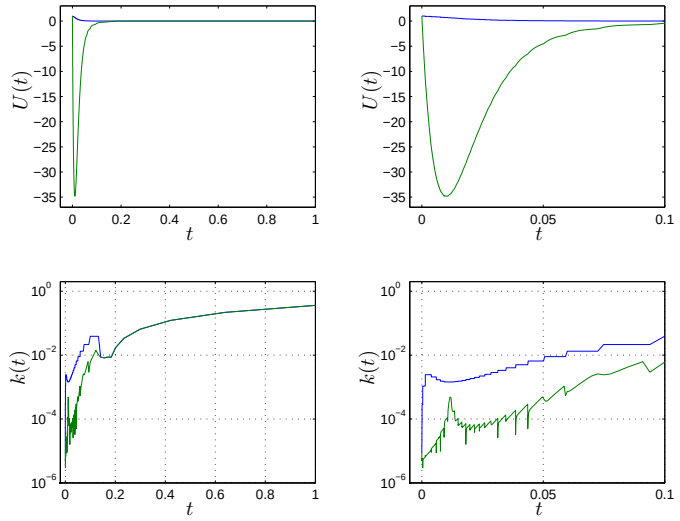

FIGURE 9. Solution and adaptive time step sequence for (5.3).

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:

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

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).

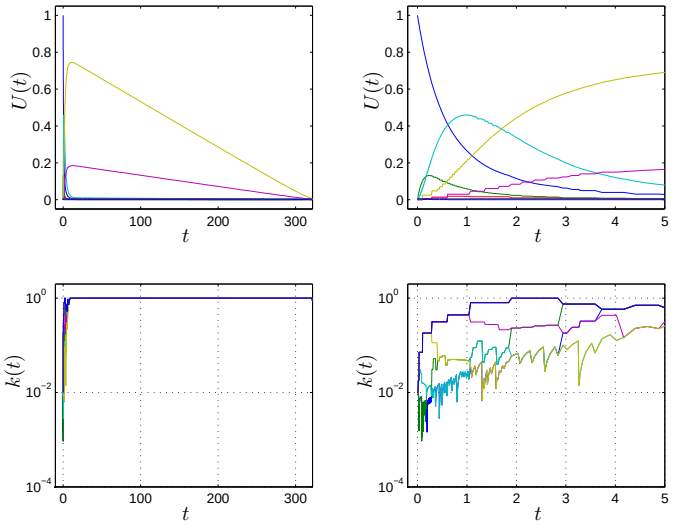

FIGURE 10. Solution and adaptive time step sequence for (5.4).

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:

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

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.

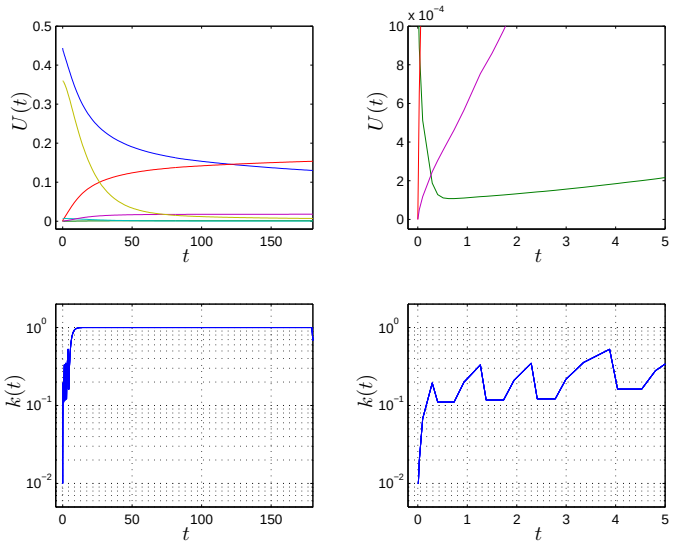

FIGURE 11. Solution and adaptive time step sequence for (5.5).

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.

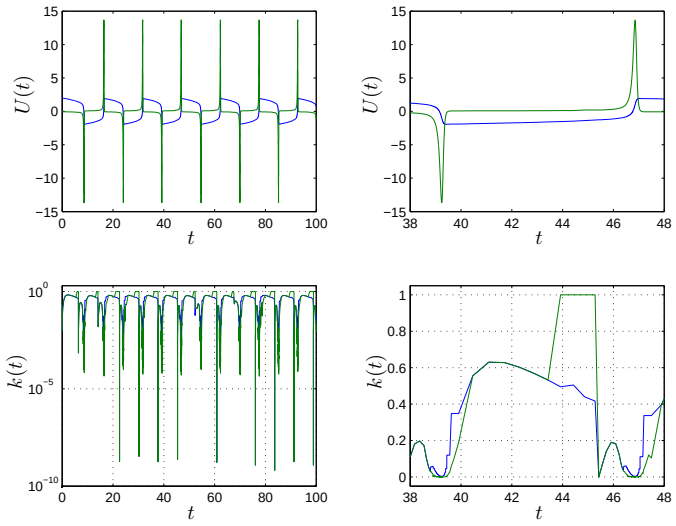

FIGURE 12. Solution and adaptive time step sequence for (5.6).

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.

5.8. **A chemical reaction test problem.** The next problem originating from 1966 (Robertson) is taken from Hairer and Wanner [4]. This problem

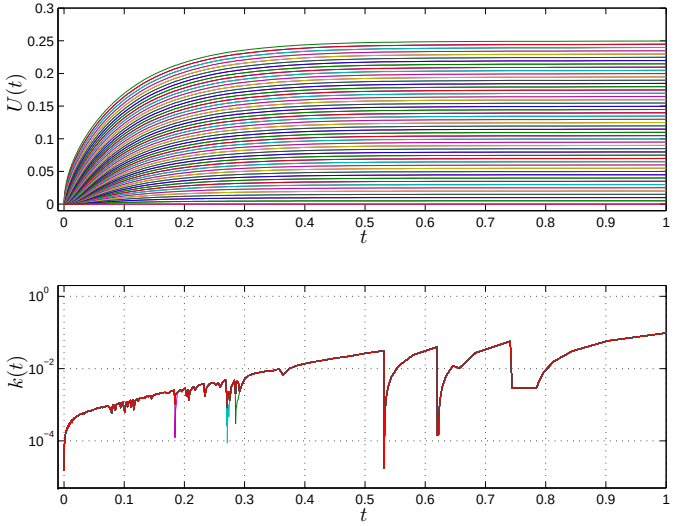

FIGURE 13. Solution and adaptive time step sequence for (5.7).

models the following system of chemical reactions:

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

which gives an initial value problem of the form

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

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

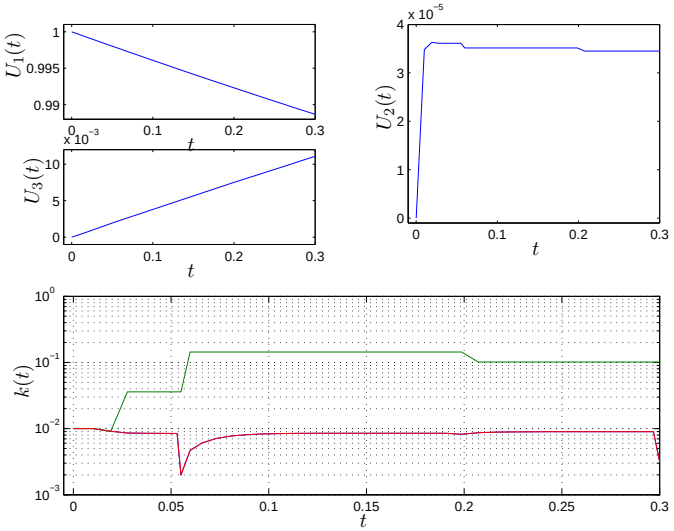

FIGURE 14. Solution and adaptive time step sequence for (5.8).

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

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.

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 $[10, 30]$.

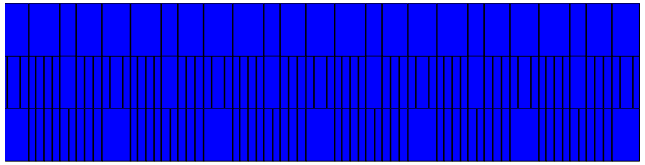

FIGURE 15. The structure of the time slabs on the interval $[10, 30]$ for the solution of (5.9).

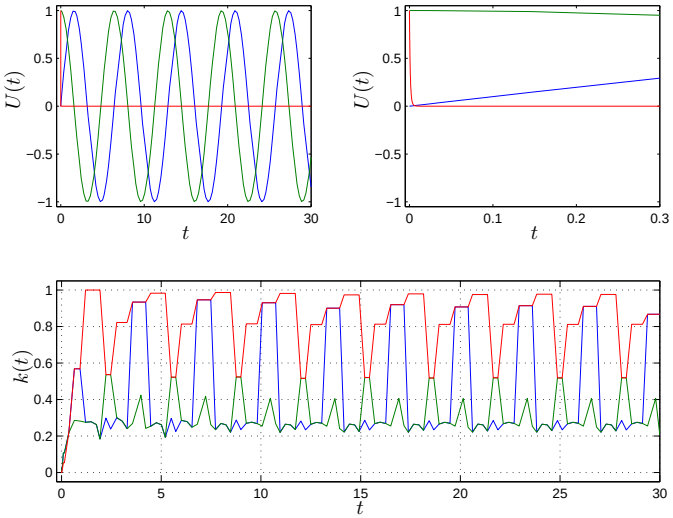

FIGURE 16. Solution and adaptive time step sequence for (5.9).

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 L.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*, submitted to ACM Trans. Math. Softw., (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).

SIAM J. SCI. COMPUT.
Vol. 25, No. 4, pp. 1142–1157

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, so the improvement is large. We illustrate the technique on a number of well-known stiff test problems.

Key words. stiff ODE, explicit methods, explicit Euler, Galerkin methods

AMS subject classifications. 65L05, 65L07, 65L20, 65L50, 65L60, 65L70, 65M12

DOI. 10.1137/S1064827502409626

1. Introduction.

The classical wisdom developed in the 1950s regarding stiff ODEs is that efficient integration requires implicit (A-stable) methods, at least outside of 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 counterbalanced 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 now is 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 of 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, which is the 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 to 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 yields 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, 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. It remains to be seen if the proposed method can also be efficient in comparison to implicit methods.

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 [8, 9]). 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 formation of Jacobians and the solution of the linear system become very expensive.

Possibly, similar techniques may be used also to stabilize multigrid smoothers.

*Received by the editors June 14, 2002; accepted for publication (in revised form) June 2, 2003; published electronically December 5, 2003.
http://www.siam.org/journals/sisc/25-4/40962.html
†Department of Computational Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden (kenneth@math.chalmers.se, claes@math.chalmers.se, logg@math.chalmers.se).

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 of transients, is too restrictive for large λ.

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 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, let $c = 1/2$.

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. 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 $0 \le \lambda_1 \le \cdots \le \lambda_{i-1} \le 2/K$ and $2/K \ll \lambda_i \le \cdots \le \lambda_N$. In this case, the cost is decreased by a factor of

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

In recent independent work by Gear and Kevrekidis [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. That 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, the 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 $|P_c(x)| \le 1$ on $[0, K\lambda_N]$ (see Figure 1). A similar approach is taken in [11].

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

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

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 $k_m = 2/(\lambda_N(1-s_1)) \le K$; that is,

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

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 is a modified version of the simple approach of small and large time steps discussed in the previous section. As discussed above, the problem with this basic method is that the damping is inefficient for intermediate eigenmodes. To solve this problem, we modify the simple method of large and small time steps to include also time steps of intermediate size. A related approach is taken in [3]. Starting with a couple of small time steps of size k, we thus increase the time step gradually by (say) a factor of 2, until we reach the largest step size K. Correspondingly, we decrease the number of time steps at each level by a factor of 2. The polynomial P is now given by

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

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

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

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

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

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

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.

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 of 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 must 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 dyadic damping method 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 $K\lambda_N = 64$. With $K\lambda_N = 64$, we obtain $m = (\pi/4)\sqrt{K\lambda_N} \approx 6$ for the Chebyshev polynomial. For the dyadic damping polynomial, we have $p = \log(K\lambda_N)/\log(2) = 6$, and from Table 1, we obtain $q = 3$.

As another comparison, we 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

$$P_c(z) = \prod_{i=1}^{m} \left(1 + \frac{z/m^2}{1 - s_i}\right) \tag{4.1}$$

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

$$P_d(z) = \prod_{j=q+1}^{p} (1 + \theta_j z) \prod_{i=0}^{q} (1 + \theta_i z)^{2^{q-1}} \tag{4.2}$$

for $\theta_i = 2^i/(2^q(q+1) + 2^{p+1} - 2^{q+1})$. Note that we take $z = -\bar{K}\lambda$, where $\bar{K} = k_1 + \cdots + k_m$ is the sum of all time steps in the sequence, rather than $z = -K\lambda$, where $K \leq \bar{K}$ is the maximum time step in the sequence. To make the comparison fair, we take $m = 5$ for the Chebyshev polynomial and $(p, q) = (3, 1)$ for the dyadic damping polynomial, which gives the same number of time steps ($m = 2^{q+1} - 1 + p - q = 5$) for the two methods.

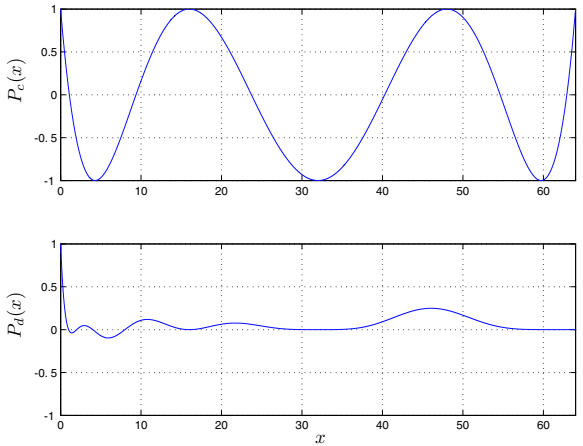

FIG. 1. *A comparison of the two polynomials for $K\lambda_N = 64$: we take $m = 6$ for the shifted Chebyshev polynomial $P_c(x)$ and $(p, q) = (6, 3)$ for the dyadic polynomial $P_d(x)$. With $K = 1$, the costs are 5.3 and 8, respectively.*

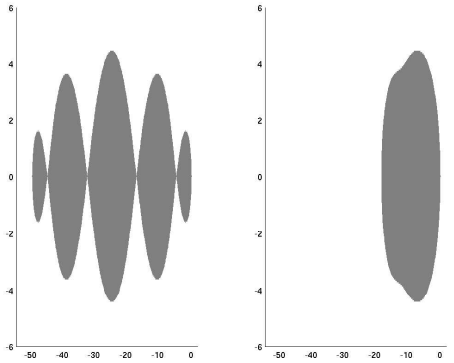

FIG. 2. *A comparison of stability regions for the two polynomials, with $m = 5$ for the shifted Chebyshev polynomial $P_c(z)$ (left) and $(p, q) = (3, 1)$ for the dyadic polynomial $P_d(z)$ (right).*

From Figures 1 and 2, we see that Chebyshev damping can be slightly more efficient than dyadic damping; the cost is smaller for a given value of $K\lambda_N$ (Figure 1) and the stability interval on the negative real axis is larger (Figure 2). However, with dyadic damping we don't need to assume that the eigenvalues of A in (3.1) lie in a narrow strip along the negative real 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 [11].)

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 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 occurs with $U^{n,0} = U^{n-1}$. The fixed point iteration converges for a small enough time step k_n, 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 tol > 0. Usually only a couple of iterations are needed. Estimating the error $e(T) = U(T) - u(T)$ at final time T (see [8]), we have

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

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 $S(T) \le 1$ and $S^0(T) \le 1/\lambda$, which suggests that for a typical stiff problem we can take tol = 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 [8, 9] 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 an 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 $t_n \ge T$:
   (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 also have to solve 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 [4, 10] or [9]. 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) did not 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 [7], including among others the test problems presented in the next section.

We also note that, by (8.1), we have $k_n \leq 2k_{n-1}$, 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 being that most of the damping is made with the smallest time step. For a parabolic problem, we modify the algorithm to increase the time step more carefully, as determined by Table 1, with p given by $p = \log(k_n L)/\log(2)$.

9. Examples. To illustrate the technique, we take a simple standard implementation of the cG(1)-method (with an explicit fixed point solution of the discrete equations) and add a couple of lines to handle the stabilization of stiff problems. We

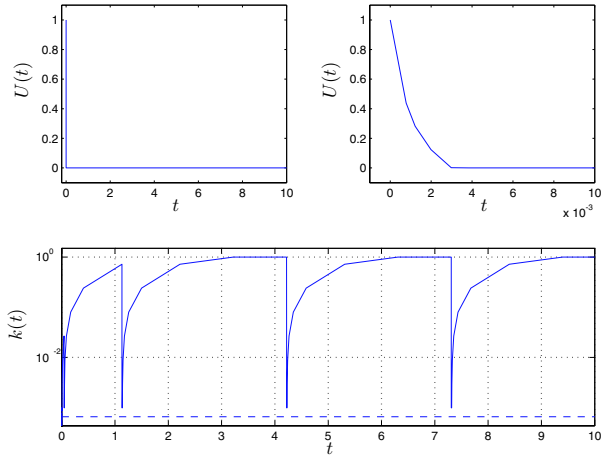

FIG. 3. *Solution and time step sequence for* (9.1), $\alpha/\alpha_0 \approx 1/310$.

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 also includes 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 stabilization again is needed.

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 $A = \mathrm{diag}(100, 1000)$ 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

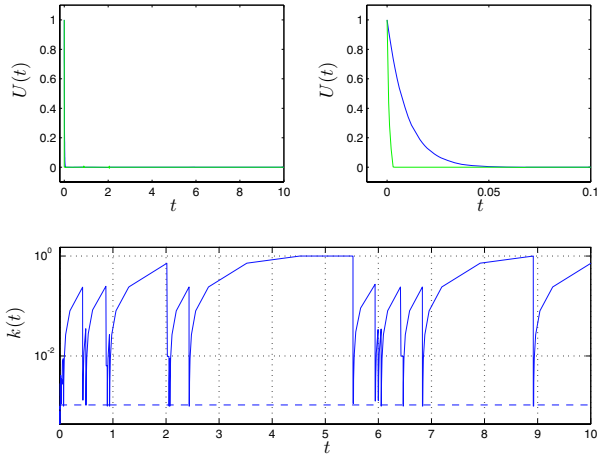

FIG. 4. *Solution and time step sequence for* (9.2), $\alpha/\alpha_0 \approx 1/104$.

$\lambda_2 = 1000$, 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 $\lambda_1 = 100$ (see Figure 4). 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, $\alpha \approx 18$, which gives a cost reduction factor of $\alpha/\alpha_0 \approx 1/104$.

9.3. A linear nonnormal problem. The method behaves similarly even if we make the matrix A highly nonnormal. 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: $\alpha/\alpha_0 \approx 1/180$. The solution and the time step sequence are given in Figure 5.

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 [6]. The problem consists of the following

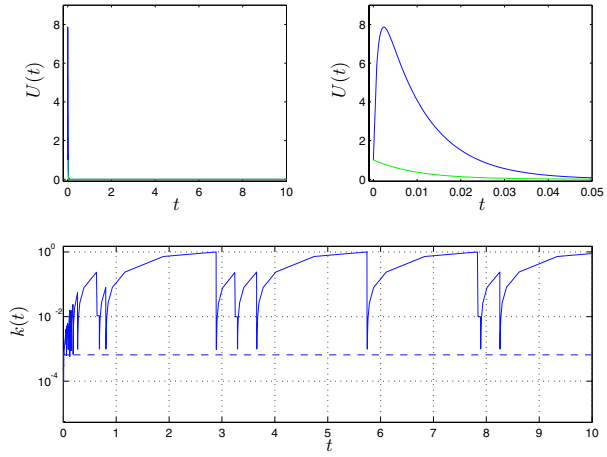

FIG. 5. *Solution and time step sequence for* (9.3), $\alpha/\alpha_0 \approx 1/180$.

eight equations:

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

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 [6]) 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 $\alpha/\alpha_0 \approx 1/33$.

9.5. The Akzo–Nobel problem. The next problem is a version of the "Chemical Akzo–Nobel" problem taken from the ODE test set [6], consisting of the following six equations:

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

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

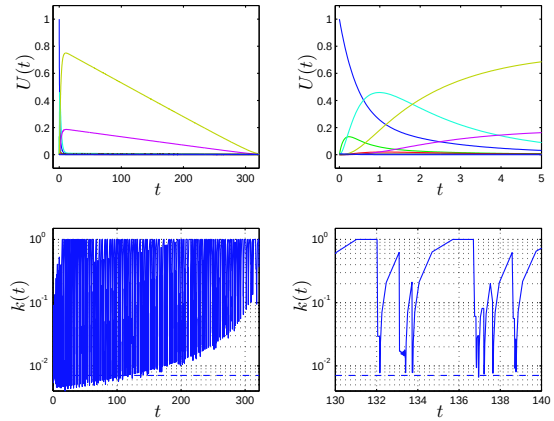

FIG. 6. *Solution and time step sequence for* (9.4), $\alpha/\alpha_0 \approx 1/33$.

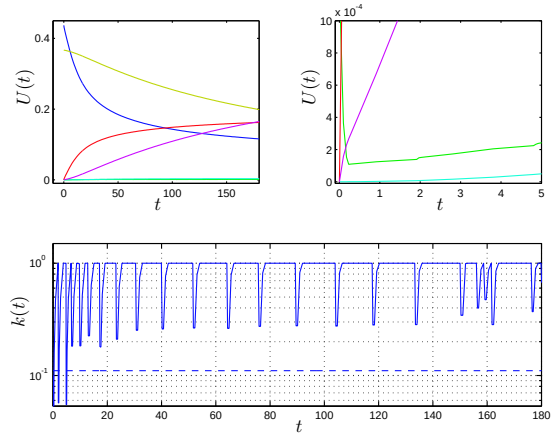

FIG. 7. *Solution and time step sequence for* (9.5), $\alpha/\alpha_0 \approx 1/9$.

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 of the large time steps and the small damping time steps, as well as the number of small damping steps that are needed. In this

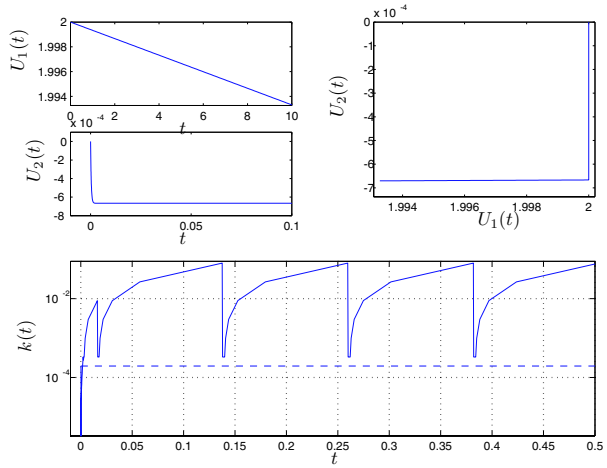

FIG. 8. *Solution and time step sequence for* (9.6), $\alpha/\alpha_0 \approx 1/75$.

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).

9.6. Van der Pol's equation. A stiff problem discussed in the book by Hairer and Wanner [5] is Van der Pol's equation,

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

which we write as

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

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 (see Figure 8). The cost is now $\alpha \approx 140$, and the cost reduction factor is $\alpha/\alpha_0 \approx 1/75$.

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

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

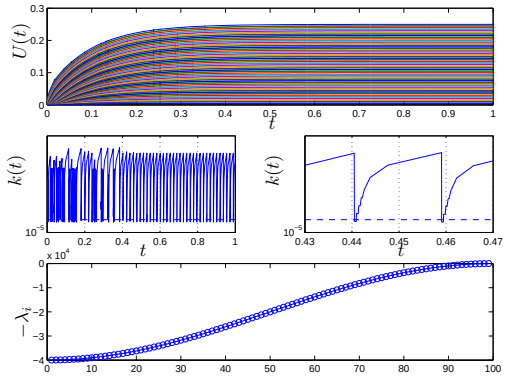

FIG. 9. *Solution and time step sequence for* (9.7), $\alpha/\alpha_0 \approx 1/31$. *Note that the time step sequence in this example is chosen slightly differently than in the previous examples, using the technique of dyadic damping discussed in section* 3. *This is evident upon close inspection of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

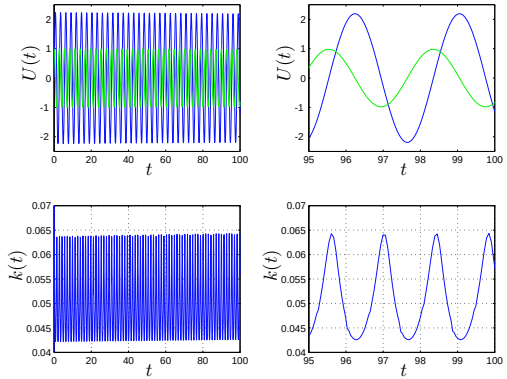

FIG. 10. *Solution and time step sequence for* (9.8), $\alpha/\alpha_0 = 1$.

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

9.8. A nonstiff problem. To show that the method works equally well for nonstiff 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 nonstiff (for reasonable tolerances), no stabilization is needed, and so the solver works as a standard nonstiff cG(1) solver with no overhead. This is also evident from the time step sequence (see Figure 10) which is chosen only to match the size of the (continuous) residual.

REFERENCES

[1] K. ERIKSSON, D. ESTEP, P. HANSBO, AND C. JOHNSON, *Introduction to adaptive methods for differential equations*, in Acta Numerica 1995, Acta. Numer., Cambridge University Press, Cambridge, UK, 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*, SIAM J. Sci. Comput., 24 (2003), pp. 1091–1106.

[3] C. W. GEAR AND I. G. KEVREKIDIS, *Telescopic methods for parabolic differential equations*, J. Comput. Phys., 187 (2003), pp. 95–109.

[4] 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.

[5] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations* II—*Stiff and Differential-Algebraic Problems*, Springer Ser. Comput. Math. 14, Springer-Verlag, New York, 1991.

[6] W. M. LIOEN AND H. J. B. DE SWART, *Test Set for Initial Value Problem Solvers*, Tech. report MAS–R9832, CWI, Amsterdam, 1998.

[7] A. LOGG, *Chalmers Finite Element Center Software*, Chalmers University of Technology, Göteborg, Sweden, 2002. Available online at www.phi.chalmers.se/software/.

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

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

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

[11] J. G. VERWER, *Explicit Runge-Kutta methods for parabolic partial differential equations*, Appl. Numer. Math., 22 (1996), pp. 359–379.

INTERPOLATION ESTIMATES FOR PIECEWISE SMOOTH FUNCTIONS IN ONE DIMENSION

ANDERS LOGG

ABSTRACT. In preparation for a priori error analysis of the multi-adaptive Galerkin methods mcG(q) and mdG(q) presented earlier in a series of papers, we prove basic interpolation estimates for a pair of carefully chosen non-standard interpolants. The main tool in the derivation of these estimates is the Peano kernel theorem. A large part of the paper is non-specific and applies to general interpolants on the real line.

1. INTRODUCTION

The motivation for this paper is to prepare for the a priori error analysis of the multi-adaptive Galerkin methods mcG(q) and mdG(q), presented earlier in [1, 2]. This requires a set of special interpolation estimates for piecewise smooth functions, which we prove in this paper.

Throughout this paper, V denotes the space of piecewise smooth, real-valued functions on $[a, b]$, that is, the set of functions which, for some partition $a = x_0 < x_1 < \ldots < x_n < x_{n+1} = b$ of the interval $[a, b]$ and some $q \geq 0$, are $\mathcal{C}^{q+1}$ and bounded on each of the sub-intervals (x_{i-1}, x_i), $i = 1, \ldots, n + 1$. This is illustrated in Figure 1.

For $v \in V$, we denote by πv a polynomial approximation of v on $[a, b]$, such that $\pi v \approx v$. We refer to πv as an *interpolant* of v.

We are concerned with estimating the size of the interpolation error $\pi v - v$ in the maximum norm, $\|\cdot\| = \|\cdot\|_{L_\infty([a,b])}$, in terms of the regularity of v and the length of the interval, $k = b - a$. Specifically, when $v \in \mathcal{C}^{q+1}([a, b]) \subset V$ for some $q \geq 0$, we obtain estimates of the form

$$\|(\pi v)^{(p)} - v^{(p)}\| \leq C k^{q+1-p} \|v^{(q+1)}\|, \quad p = 0, \ldots, q + 1. \tag{1.1}$$

In the general case, the interpolation estimates include also the size of the jump $[v^{(p)}]_x$ in function value and derivatives at the points of discontinuity within (a, b).

Date: March 15, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, a priori error estimates, Peano kernel theorem, interpolation estimates, piecewise smooth.

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

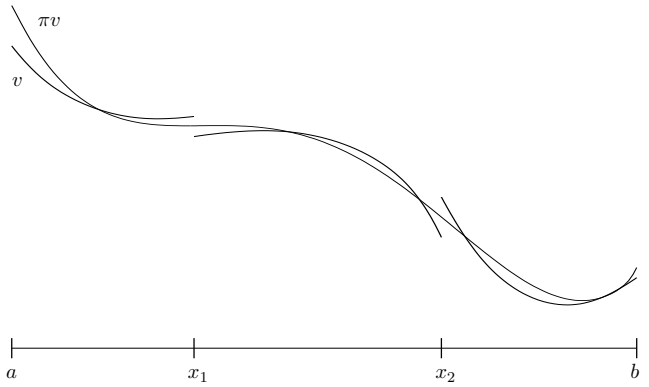

FIGURE 1. A piecewise smooth function v and its interpolant πv.

1.1. **Outline of the paper.** We first assume that $v \in \mathcal{C}^{q+1}([a, b]) \subset V$ and use the Peano kernel theorem to obtain a representation of the interpolation error $\pi v - v$ (Section 2). We then directly obtain interpolation estimates (for $v \in \mathcal{C}^{q+1}([a, b])$) in Section 3.

In Section 4, we generalize the interpolation estimates from Section 3 to v piecewise smooth by constructing a regularized version of v. Finally, in Section 5, we apply the general results of Section 4 to a pair of special interpolants that appear in the a priori error analysis of the mcG(q) and mdG(q) methods.

2. THE PEANO KERNEL THEOREM

The basic tool in our derivation of interpolation estimates is the Peano kernel theorem, which we discuss in this section. In its basic form, the Peano kernel theorem can be stated as follows.

Theorem 2.1. (Peano kernel theorem) *For Λ a bounded and linear functional on V that vanishes on $\mathcal{P}^q([a, b]) \subset V$, define*

$$K(t) = \frac{1}{q!} \Lambda((\cdot - t)_+^q), \tag{2.1}$$

where $v_+(t) = \max(0, v(t))$. In other words, $K(t) = \Lambda w/q!$, with $w(x) = [\max(0, x - t)]^q$. If K has bounded variation and $v \in \mathcal{C}^{q+1}([a, b])$, then

$$\Lambda v = \int_a^b K(t) v^{(q+1)}(t)\, dt. \tag{2.2}$$

Proof. See [3]. □

Let now $v \in V$ and let

$$\pi : V \to \mathcal{P}^q([a,b]) \subset V. \tag{2.3}$$

If we can show that

(1) π is linear on V,
(2) π is exact on $\mathcal{P}^q([a,b]) \subset V$, that is, $\pi v = v$ for all $v \in \mathcal{P}^q([a,b])$, and
(3) π is bounded on V, that is, $\|\pi v\| \le C\|v\|$ for all $v \in V$ for some constant $C > 0$,

then the Peano kernel theorem directly leads to a representation of the interpolation error $\pi v - v$.

Theorem 2.2. (Peano kernel theorem II) *If π is linear and bounded (with constant $C > 0$) on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a bounded function $G : [a,b] \times [a,b] \to \mathbb{R}$, with $|G(x,t)| \le 1 + C$ for all $x, t \in [a,b]$, such that for all $v \in \mathcal{C}^{q+1}([a,b])$,*

$$\pi v(x) - v(x) = \frac{k^{q+1}}{q!}\frac{1}{k}\int_a^b v^{(q+1)}(t)G(x,t)\,dt, \tag{2.4}$$

where $k = b - a$. Furthermore, if g is an integrable, (essentially) bounded function on $[a,b]$, then there is a function $H_g : [a,b] \to \mathbb{R}$, with $|H_g(x)| \le (1+C)\|g\|$ for all $x \in [a,b]$, such that for all $v \in \mathcal{C}^{q+1}([a,b])$,

$$\int_a^b (\pi v(x) - v(x))\, g(x)\,dx = \frac{k^{q+1}}{q!}\int_a^b v^{(q+1)}(x)H_g(x)\,dx. \tag{2.5}$$

Proof. To prove (2.4), we define for any fixed $x \in [a,b]$,

$$\Lambda_x v = \pi v(x) - v(x),$$

which is linear, since π is linear. Furthermore, if $v \in \mathcal{P}^q([a,b])$ then $\pi v = v$ so Λ_x vanishes on $\mathcal{P}^q([a,b])$. From the estimate,

$$|\Lambda_x v| = |\pi v(x) - v(x)| \le |\pi v(x)| + |v(x)| \le (1+C)\|v\|,$$

it follows that Λ_x is bounded. Now,

$$\begin{aligned} K_x(t) &= \frac{1}{q!}\Lambda_x((\cdot - t)_+^q) = \frac{1}{q!}\left[\pi((\cdot - t)_+^q)(x) - (x-t)_+^q\right] \\ &= \frac{k^q}{q!}\left[\pi\left(\left(\frac{\cdot - t}{k}\right)_+^q\right)(x) - \left(\frac{x-t}{k}\right)_+^q\right] = \frac{k^q}{q!}G(x,t), \end{aligned}$$

where $|G(x,t)| \le 1 + C$ for $x, t \in [a,b]$. Thus, by the Peano kernel theorem,

$$\pi v(x) - v(x) = \Lambda_x v = \frac{k^{q+1}}{q!}\frac{1}{k}\int_a^b v^{(q+1)}(t)G(x,t)\,dt.$$

To prove (2.5), define Λ by

$$\Lambda v = \int_a^b (\pi v(x) - v(x))g(x)\,dx,$$

which is linear, bounded, and vanishes on $\mathcal{P}^q([a,b])$. Now,

$$\begin{aligned} K(t) &= \frac{1}{q!}\int_a^b \left[\pi((\cdot - t)_+^q)(x) - (x-t)_+^q\right] g(x)\,dx \\ &= \frac{k^{q+1}}{q!}\frac{1}{k}\int_a^b \left[\pi\left(\left(\frac{\cdot - t}{k}\right)_+^q\right)(x) - \left(\frac{x-t}{k}\right)_+^q\right] g(x)\,dx \\ &= \frac{k^{q+1}}{q!}H_g(t), \end{aligned}$$

where $H_g(t) \le (1+C)\|g\|$ for $t \in [a,b]$. By the Peano kernel theorem, it now follows that

$$\int_a^b (\pi v(x) - v(x))g(x)\,dx = \frac{k^{q+1}}{q!}\int_a^b v^{(q+1)}(t)H_g(t)\,dt.$$

□

In order to derive a representation for derivatives of the interpolation error, we need to investigate the differentiability of the kernel $G(x,t)$.

Lemma 2.1. (Differentiability of G) *If π is linear on V, then the kernel G, defined by*

$$G(x,t) = \pi\left(\left(\frac{\cdot - t}{k}\right)_+^q\right)(x) - \left(\frac{x-t}{k}\right)_+^q, \tag{2.6}$$

has the following properties:

(i) *For any fixed $t \in [a,b]$, $G(\cdot,t) \in \mathcal{C}^{q-1}([a,b])$ and*

$$\frac{\partial^p}{\partial x^p}G(x,t) = k^{-p}G_{x,p}(x,t), \quad p = 0,\ldots,q, \tag{2.7}$$

where each $G_{x,p}$ is bounded on $[a,b] \times [a,b]$ independent of k.

(ii) *For any fixed $x \in [a,b]$, $G(x,\cdot) \in \mathcal{C}^{q-1}([a,b])$ and*

$$\frac{\partial^p}{\partial t^p}G(x,t) = k^{-p}G_{t,p}(x,t), \quad p = 0,\ldots,q, \tag{2.8}$$

where each $G_{t,p}$ is bounded on $[a,b] \times [a,b]$ independent of k.

(iii) *For $x \neq t$ and $p_1, p_2 \ge 0$, we have*

$$\frac{\partial^{p_2}}{\partial x^{p_2}}\frac{\partial^{p_1}}{\partial t^{p_1}}G(x,t) = k^{-(p_1+p_2)}G_{t,x,p_1,p_2}(x,t), \tag{2.9}$$

where each G_{t,x,p_1,p_2} is bounded on $[a,b] \times [a,b]\backslash\{(x,t) : x = t\}$ independent of k.

Proof. Define

$$G(x,t) = \pi\left(\left(\frac{\cdot - t}{k}\right)_+^q\right)(x) - \left(\frac{x-t}{k}\right)_+^q \equiv g(x,t) - h(x,t).$$

We first note that for any fixed $t \in [a,b]$, $h(\cdot,t) \in \mathcal{C}^{q-1}([a,b])$ with

$$\frac{\partial^p}{\partial x^p}h(x,t) = \frac{\partial^p}{\partial x^p}\left(\frac{x-t}{k}\right)_+^q = k^{-p}\frac{q!}{(q-p)!}\left(\frac{x-t}{k}\right)_+^{q-p} = k^{-p}h_{x,p}(x,t),$$

where $h_{x,p}$ is bounded on $[a,b]\times[a,b]$ independent of k. Similarly, we note that for any fixed $x\in[a,b]$, $h(x,\cdot)\in\mathcal{C}^{q-1}([a,b])$ with

$$\frac{\partial^p}{\partial t^p}h(x,t)=\frac{\partial^p}{\partial t^p}\left(\frac{x-t}{k}\right)_+^q=k^{-p}\frac{q!(-1)^p}{(q-p)!}\left(\frac{x-t}{k}\right)_+^{q-p}=k^{-p}h_{t,p}(x,t),$$

where $h_{t,p}$ is bounded on $[a,b]\times[a,b]$ independent of k.

If we now let $\pi_{[0,1]}$ denote the interpolant shifted to $[0,1]$, we can write

$$\begin{aligned}g(x,t) &= \pi\left(\left(\frac{\cdot-t}{k}\right)_+^q\right)(x)=\pi\left(\left(\frac{(\cdot-a)}{k}-\frac{(t-a)}{k}\right)_+^q\right)(x)\\ &= \pi_{[0,1]}\left(\left(\cdot-\frac{(t-a)}{k}\right)_+^q\right)((x-a)/k),\end{aligned}$$

and so, for any fixed $t\in[a,b]$, $g(\cdot,t)\in\mathcal{P}^q([a,b])\subset\mathcal{C}^{q-1}([a,b])$, with

$$\frac{\partial^p}{\partial x^p}g(x,t)=k^{-p}\pi^{(p)}_{[0,1]}\left(\left(\cdot-\frac{(t-a)}{k}\right)_+^q\right)((x-a)/k))=k^{-p}g_{x,p}(x,t),$$

where $g_{x,p}$ is bounded on $[a,b]\times[a,b]$ independent of k. Finally, let $s=(t-a)/k$. Then,

$$g(x,t)=\pi_{[0,1]}\left((\cdot-s)_+^q\right)((x-a)/k)).$$

The degrees of freedom of the interpolant $\pi_{[0,1]}\left((\cdot-s)_+^q\right)$ are determined by the solution of a linear system, and thus each of the degrees of freedom (point values or integrals) will be a linear combination of the degrees of freedom of the function $(\cdot-s)_+^q$, each of which in turn are $\mathcal{C}^{q-1}$ in the s-variable. Hence, $g(x,\cdot)\in\mathcal{C}^{q-1}([a,b])$ for any fixed $x\in[a,b]$, with

$$\begin{aligned}\frac{\partial^p}{\partial t^p}g(x,t) &= \frac{\partial^p}{\partial t^p}\pi_{[0,1]}\left((\cdot-s)_+^q\right)((x-a)/k))\\ &= k^{-p}\frac{\partial^p}{\partial s^p}\pi_{[0,1]}\left((\cdot-s)_+^q\right)((x-a)/k))=k^{-p}g_{t,p}(x,t),\end{aligned}$$

where $g_{t,p}$ is bounded on $[a,b]\times[a,b]$ independent of k. We now take $G_{x,p}=g_{x,p}-h_{x,p}$ and $G_{t,p}=g_{t,p}-h_{t,p}$, which proves (2.7) and (2.8).

To prove (2.9), we note that

$$\frac{\partial^{p_1}}{\partial t^{p_1}}h(x,t)=k^{-p_1}\frac{q!(-1)^{p_1}}{(q-p_1)!}\left(\frac{x-t}{k}\right)_+^{q-p_1},$$

and so, for $x\neq t$,

$$\frac{\partial^{p_2}}{\partial x^{p_2}}\frac{\partial^{p_1}}{\partial t^{p_1}}h(x,t)=k^{-(p_1+p_2)}\frac{q!(-1)^{p_1}}{(q-(p_1+p_2))!}\left(\frac{x-t}{k}\right)_+^{q-(p_1+p_2)},$$

when $p_1+p_2\le q$ and $\frac{\partial^{p_2}}{\partial x^{p_2}}\frac{\partial^{p_1}}{\partial t^{p_1}}h(x,t)=0$ for $p_1+p_2>q$. Furthermore, for any fixed x,

$$\frac{\partial^{p_1}}{\partial t^{p_1}}g(x,t)=k^{-p_1}\frac{\partial^{p_1}}{\partial s^{p_1}}\pi_{[0,1]}\left((\cdot-s)_+^q\right)((x-a)/k)).$$

With $y=(x-a)/k$, we thus have

$$\frac{\partial^{p_2}}{\partial x^{p_2}}\frac{\partial^{p_1}}{\partial t^{p_1}}g(x,t)=k^{-(p_1+p_2)}\frac{\partial^{p_2}}{\partial y^{p_2}}\frac{\partial^{p_1}}{\partial s^{p_1}}\pi_{[0,1]}\left((\cdot-s)_+^q\right)(y).$$

We conclude that

$$\frac{\partial^{p_2}}{\partial x^{p_2}}\frac{\partial^{p_1}}{\partial t^{p_1}}G(x,t)=k^{-(p_1+p_2)}G_{t,x,p_1,p_2}(x,t),$$

where G_{t,x,p_1,p_2} is bounded on $[a,b]\times[a,b]\backslash\{(x,t):x=t\}$. □

By differentiating (2.4), we now obtain the following representation for derivatives of the interpolation error.

Theorem 2.3. (Peano kernel theorem III) *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b])\subset V$, then there is a constant $C>0$, depending only on the definition of the interpolant (and not on k), and functions $G_p:[a,b]\times[a,b]\to\mathbb{R}$, $p=0,\ldots,q$, such that for all $v\in\mathcal{C}^{q+1}([a,b])$,*

(2.10)
$$(\pi v)^{(p)}(x)-v^{(p)}(x)=\frac{k^{q+1-p}}{q!}\frac{1}{k}\int_a^b v^{(q+1)}(t)G_p(x,t)\,dt,\quad p=0,\ldots,q,$$

where for each p, $|G_p(x,t)|\le C$ for all $x,t\in[a,b]\times[a,b]$.

Proof. For $p=0$ the result follows from Theorem 2.2 with $G_0=G$. For $p=1,\ldots,q$, we differentiate (2.4) with respect to x and use Lemma 2.1, to obtain

$$\begin{aligned}(\pi v)^{(p)}(x)-v^{(p)}(x) &= \frac{k^q}{q!}\int_a^b v^{(q+1)}(t)\frac{\partial^p}{\partial x^p}G(x,t)\,dt\\ &= \frac{k^{q-p}}{q!}\int_a^b v^{(q+1)}(t)G_{x,p}(x,t)\,dt.\end{aligned}$$

□

3. Interpolation estimates

Using the results of the previous section, we now obtain estimates for the interpolation error $\pi v-v$. The following corollary is a simple consequence of Theorem 2.3.

Corollary 3.1. *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b])\subset V$, then there is a constant $C=C(q)>0$, such that for all $v\in\mathcal{C}^{q+1}([a,b])$,*

(3.1)
$$\|(\pi v)^{(p)}-v^{(p)}\|\le Ck^{r+1-p}\|v^{(r+1)}\|,$$

for $p=0,\ldots,r+1$, $r=0,\ldots,q$.

Proof. If π is exact on $\mathcal{P}^q([a,b])$, it is exact on $\mathcal{P}^r([a,b])\subseteq\mathcal{P}^q([a,b])$ for $r=0,\ldots,q$. It follows by Theorem 2.3 that for all $v\in\mathcal{C}^{(r+1)}([a,b])$, we have

$$\|(\pi v)^{(p)}-v^{(p)}\|\le Ck^{r+1-p}\|v^{(r+1)}\|,$$

for $p = 0, \ldots, r$. When $r < q$, this holds also for $p = r + 1 \leq q$, and for $p = r + 1 = q$, we note that $\|(\pi v)^{(p)} - v^{(p)}\| = \|v^{(p)}\| = \|v^{(r+1)}\|$. □

This leads to the following estimate for the derivative of the interpolant.

Corollary 3.2. *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a constant $C = C(q) > 0$, such that for all $v \in \mathcal{C}^{q+1}([a,b])$,*

$$\|(\pi v)^{(p)}\| \leq C\|v^{(p)}\|, \tag{3.2}$$

for $p = 0, \ldots, q$.

Proof. It is clear that (3.2) holds when $p = 0$ since π is bounded. For $0 < p \leq q$, we add and subtract $v^{(p)}$, to obtain

$$\|(\pi v)^{(p)}\| \leq \|(\pi v)^{(p)} - v^{(p)}\| + \|v^{(p)}\|.$$

Since π is exact on $\mathcal{P}^q([a,b])$, it is exact on $\mathcal{P}^{p-1}([a,b]) \subset \mathcal{P}^q([a,b])$ for $p \leq q$. It follows by Corollary 3.1, that

$$\|(\pi v)^{(p)}\| \leq Ck^{(p-1)+1-p}\|v^{((p-1)+1)}\| + \|v^{(p)}\| = (C+1)\|v^{(p)}\| = C'\|v^{(p)}\|.$$

□

Finally, we show that it is often enough to show that π is linear on V and exact on $\mathcal{P}^{(q)}([a,b])$, that is, we do not have to show that π is bounded.

Lemma 3.1. *If $\pi : V \to \mathcal{P}^q([a,b])$ is linear on V, and is uniquely determined by n_1 interpolation conditions and $n_2 = q + 1 - n_1$ projection conditions,*

$$\pi v(x_i) = v(x_i), \quad i = 1, \ldots, n_1, \tag{3.3}$$

where each $x_i \in [a,b]$, and

$$\int_a^b \pi v(x) w_i(x)\, dx = \int_a^b v(x) w_i(x)\, dx, \quad i = 1, \ldots, n_2, \tag{3.4}$$

where each w_i is bounded, then π is bounded on V, that is,

$$\|\pi v\| \leq C\|v\| \quad \forall v \in V, \tag{3.5}$$

for some constant $C = C(q) > 0$.

Proof. Define $\|\cdot\|_*$ on $\mathcal{P}^q([a,b])$ by

$$\|v\|_* = \sum_{i=1}^{n_1} |v(x_i)| + \sum_{i=1}^{n_2} \left| \frac{1}{k} \int_a^b v(x) w_i(x)\, dx \right|.$$

Clearly, the triangle inequality holds and $\|\alpha v\|_* = |\alpha|\|v\|_*$ for all $\alpha \in \mathbb{R}$. Furthermore, if $v = 0$ then $\|v\|_* = 0$. Conversely, if $\|v\|_* = 0$ then $v \in \mathcal{P}^q([a,b])$ is the unique interpolant of 0, and so $v = \pi 0 = 0$ by linearity. Thus, $\|\cdot\|_*$ is a norm. Since all norms on a finite dimensional vector space are equivalent, we obtain

$$\|\pi v\| \leq C\|\pi v\|_* = C\|v\|_* \leq C(n_1 + C' n_2)\|v\| = C''\|v\|.$$

□

4. Interpolation of piecewise smooth functions

We now extend the interpolation results of the previous two sections to piecewise smooth functions.

4.1. Extensions of the Peano kernel theorem.

To extend the Peano kernel theorem to piecewise smooth functions, we construct a smooth version v_ϵ of the discontinuous function v, with $v_\epsilon \to v$ as $\epsilon \to 0$. For the construction of v_ϵ, we introduce the function

$$w(x) = Cx \sum_{m=0}^{q+1} \binom{q+1}{m} \frac{(-1)^m x^{2(q+1)-2m}}{2(q+1) - 2m + 1}, \tag{4.1}$$

where

$$C = \left(\sum_{m=0}^{q+1} \binom{q+1}{m} \frac{(-1)^m}{2(q+1) - 2m + 1} \right)^{-1}. \tag{4.2}$$

In Figure 2, we plot the function $w(x)$ on $[-1,1]$ for $q = 0, \ldots, 4$.

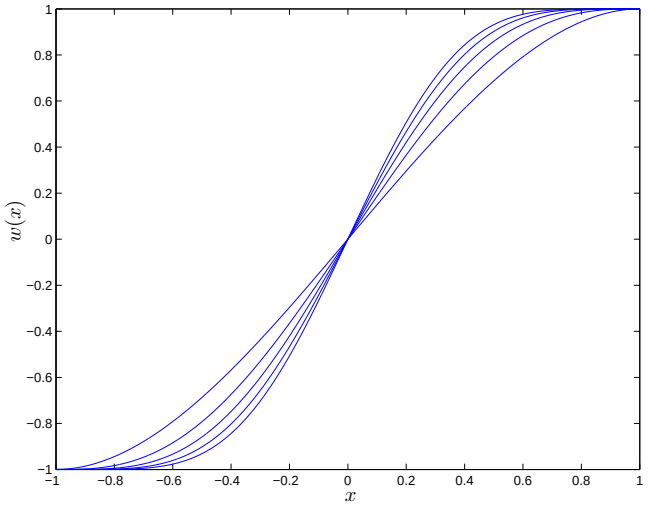

Figure 2. The function $w(x)$ on $[-1,1]$ for $q = 0, \ldots, 4$.

Let also $T_y v$ denote the Taylor expansion of order $q \geq 0$ around a given point $y \in (a,b)$. For $\epsilon > 0$ sufficiently small, we then define

(4.3)
$$v_\epsilon(x) = \begin{cases} \left(1 - \frac{w((x-x_1)/\epsilon)+1}{2}\right) T_{x_1-\epsilon} v(x) + \frac{w((x-x_1)/\epsilon)+1}{2} T_{x_1+\epsilon} v(x), \\ \quad x \in [x_1-\epsilon, x_1+\epsilon], \\ \dots \\ \left(1 - \frac{w((x-x_n)/\epsilon)+1}{2}\right) T_{x_n-\epsilon} v(x) + \frac{w((x-x_n)/\epsilon)+1}{2} T_{x_n+\epsilon} v(x), \\ \quad x \in [x_n-\epsilon, x_n+\epsilon], \end{cases}$$

with $v_\epsilon = v$ on $[a,b] \setminus ([x_1-\epsilon, x_1+\epsilon] \cup \ldots \cup [x_n-\epsilon, x_n+\epsilon])$. As a consequence of the following Lemma, v_ϵ has $q+1$ continuous derivatives on $[a,b]$.

Lemma 4.1. *The weight function w defined by* (4.1) *and* (4.2) *has the following properties:*

$$-1 = w(-1) \leq w(x) \leq w(1) = 1, \quad x \in [-1,1], \tag{4.4}$$

and

$$w^{(p)}(-1) = w^{(p)}(1) = 0, \quad p = 1, \dots, q+1. \tag{4.5}$$

Proof. It is clear from the definition that $w(1) = 1$ and $w(-1) = -1$. Taking the derivative of w, we obtain

$$\begin{aligned} \frac{dw}{dx}(x) &= C \sum_{m=0}^{q+1} \binom{q+1}{m} \frac{(-1)^m \frac{d}{dx} x^{2(q+1)-2m+1}}{2(q+1)-2m+1} \\ &= C \sum_{m=0}^{q+1} \binom{q+1}{m} (-1)^m x^{2(q+1)-2m} \\ &= C(x^2-1)^{q+1} = C(x+1)^{q+1}(x-1)^{q+1}, \end{aligned}$$

and so $w^{(p)}$ is zero at $x = \pm 1$ for $p = 1, \dots, q+1$. Furthermore, since $\frac{dw}{dx}$ has no zeros within $(-1,1)$, w attains its maximum and minimum at $x = 1$ and $x = -1$ respectively. □

This leads to the following extension of Theorem 2.2.

Theorem 4.1. (Peano kernel theorem for piecewise smooth functions) *If π is linear and bounded (with constant $C > 0$) on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a bounded function $G : [a,b] \times [a,b] \to \mathbb{R}$, with $|G(x,t)| \leq 1 + C$ for all $x, t \in [a,b]$, such that for v piecewise $\mathcal{C}^{q+1}$ on $[a,b]$ with discontinuities at $a < x_1 < x_2 < \ldots < x_n < b$, we have*

$$\pi v(x) - v(x) = \frac{k^{q+1}}{q!} \frac{1}{k} \int_a^b v^{(q+1)}(t) G(x,t)\, dt + \sum_{j=1}^{n} \sum_{m=0}^{q} C_{qjm}(x) k^m [v^{(m)}]_{x_j}, \tag{4.6}$$

where $k = b-a$ and for each C_{qjm} we have $|C_{qjm}(x)| \leq C$ for all $x \in [a,b]$. Furthermore, if g is an integrable, (essentially) bounded function on $[a,b]$,

then there is a function $H_g : [a,b] \to \mathbb{R}$, with $|H_g(x)| \leq (1+C)\|g\|$ for all $x \in [a,b]$, such that

$$\begin{aligned} \int_a^b (\pi v(x) - v(x)) g(x)\, dx = \frac{k^{q+1}}{q!} \int_a^b v^{(q+1)}(x) H_g(x)\, dx \\ + \sum_{j=1}^{n} \sum_{m=0}^{q} D'_{qjm}(x) k^{m+1} [v^{(m)}]_{x_j}, \end{aligned} \tag{4.7}$$

where for each D'_{qjm} we have $|D'_{qjm}(x)| \leq C$ for all $x \in [a,b]$.

Proof. Without loss of generality, we can assume that v is discontinuous only at $x_1 \in (a,b)$. Fix $x \in [a,b] \setminus \{x_1\}$, take $0 < \epsilon < |x - x_1|$, and define v_ϵ as in (4.3). Then,

$$\pi v - v = (\pi v - \pi v_\epsilon) + (\pi v_\epsilon - v_\epsilon) + (v_\epsilon - v),$$

where $|\pi v(x) - \pi v_\epsilon(x)| \to 0$ and $|v_\epsilon(x) - v(x)| \to 0$ when $\epsilon \to 0$. Since $v_\epsilon \in \mathcal{C}^{q+1}([a,b])$, we have by Theorem 2.2,

$$\pi v_\epsilon(x) - v_\epsilon(x) = \frac{k^q}{q!} \int_a^b v_\epsilon^{(q+1)}(t) G(x,t)\, dt = I_1 + I_2 + I_3,$$

where

$$\begin{aligned} I_1 &= \frac{k^q}{q!} \int_a^{x_1-\epsilon} v^{(q+1)}(t) G(x,t)\, dt, \\ I_2 &= \frac{k^q}{q!} \int_{x_1-\epsilon}^{x_1+\epsilon} v_\epsilon^{(q+1)}(t) G(x,t)\, dt, \\ I_3 &= \frac{k^q}{q!} \int_{x_1+\epsilon}^{b} v^{(q+1)}(t) G(x,t)\, dt. \end{aligned}$$

Since $v|_{[a,x_1)} \in \mathcal{C}^{[q+1]}([a,x_1))$ and $v|_{(x_1,b]} \in \mathcal{C}^{[q+1]}((x_1,b])$, it is clear that

$$I_1 + I_3 = \frac{k^q}{q!} \int_a^b v^{(q+1)}(t) G(x,t)\, dt + \mathcal{O}(\epsilon).$$

We therefore focus on the remaining term I_2. With $w_\epsilon(t) = \frac{w((t-x_1)/\epsilon)+1}{2}$, we have

$$\begin{aligned} I_2 &= \frac{k^q}{q!} \int_{x_1-\epsilon}^{x_1+\epsilon} v_\epsilon^{(q+1)}(t) G(x,t)\, dt \\ &= \frac{k^q}{q!} \int_{x_1-\epsilon}^{x_1+\epsilon} \left[(1-w_\epsilon) T_{x_1-\epsilon} v + w_\epsilon T_{x_1+\epsilon} v\right]^{(q+1)}(t) G(x,t)\, dt \\ &= \sum_{m=0}^{q+1} \frac{k^q}{q!} \binom{q+1}{m} \int_{x_1-\epsilon}^{x_1+\epsilon} \left[(1-w_\epsilon)^{(m)} T_{x_1-\epsilon}^{(q+1-m)} v + w_\epsilon^{(m)} T_{x_1+\epsilon}^{(q+1-m)} v\right](t) G(x,t)\, dt \\ &= \sum_{m=0}^{q+1} \frac{k^q}{q!} \binom{q+1}{m} I_{2m}, \end{aligned}$$

with obvious notation. Evaluating the integrals I_{2m}, we have for $m = 0$,

$$I_{20} = \int_{x_1-\epsilon}^{x_1+\epsilon} \left[(1-w_\epsilon)^{(0)} T_{x_1-\epsilon}^{(q+1-0)} v + w_\epsilon^{(0)} T_{x_1+\epsilon}^{(q+1-0)} v\right](t) G(x,t)\,dt = \mathcal{O}(\epsilon),$$

while for $m = 1, \ldots, q+1$, we obtain

$$\begin{aligned} I_{2m} &= \int_{x_1-\epsilon}^{x_1+\epsilon} \left[(1-w_\epsilon)^{(m)} T_{x_1-\epsilon}^{(q+1-m)} v + w_\epsilon^{(m)} T_{x_1+\epsilon}^{(q+1-m)} v\right](t) G(x,t)\,dt \\ &= \int_{x_1-\epsilon}^{x_1+\epsilon} w_\epsilon^{(m)}(t)(T_{x_1+\epsilon}^{(q+1-m)} v(t) - T_{x_1-\epsilon}^{(q+1-m)} v(t)) G(x,t)\,dt. \end{aligned}$$

Let now $h = (T_{x_1+\epsilon}^{(q+1-m)} v - T_{x_1-\epsilon}^{(q+1-m)} v) G(x,\cdot)$ and note that by Lemma 2.1, $h \in \mathcal{C}^{q-1}([x_1-\epsilon, x_1+\epsilon])$. Noting also that $w_\epsilon^{(m)}(x_1 \pm \epsilon) = 0$ for $m = 1, \ldots, q+1$, we integrate by parts to obtain

$$\begin{aligned} I_{2m} &= \int_{x_1-\epsilon}^{x_1+\epsilon} w_\epsilon^{(m)}(t) h(t)\,dt \\ &= (-1)^{m-1} \int_{x_1-\epsilon}^{x_1+\epsilon} w_\epsilon'(t) h^{(m-1)}(t)\,dt \\ &= \frac{(-1)^{m-1}}{2\epsilon} \int_{x_1-\epsilon}^{x_1+\epsilon} w'((t-x_1)/\epsilon) h^{(m-1)}(t)\,dt \\ &= \frac{C(-1)^{m-1}}{2\epsilon} \int_{x_1-\epsilon}^{x_1+\epsilon} (((t-x_1)/\epsilon)^2 - 1)^{q+1} h^{(m-1)}(t)\,dt, \\ &= \frac{C(-1)^{m-1}}{2} \int_{-1}^{1} (s^2-1)^{q+1} h^{(m-1)}(x_1+\epsilon s)\,ds, \end{aligned}$$

where C is the constant defined in (4.2). Evaluating the derivatives of h, we obtain

$$\begin{aligned} h^{(m-1)} &= \frac{d^{m-1}}{dt^{m-1}} (T_{x_1+\epsilon}^{(q+1-m)} v - T_{x_1-\epsilon}^{(q+1-m)} v) G(x,\cdot) \\ &= \sum_{j=0}^{m-1} \binom{m-1}{j} (T_{x_1+\epsilon}^{(q+1-m+j)} v - T_{x_1-\epsilon}^{(q+1-m+j)} v) G_t^{(m-1-j)}(x,\cdot) \\ &= \sum_{j=0}^{m-1} \binom{m-1}{j} \left[v^{(q-(m-1-j))}\right]_{x_1} G_t^{(m-1-j)}(x,\cdot) + \mathcal{O}(\epsilon), \end{aligned}$$

where $G_t^{(p)}(x,t)$ denotes $\frac{\partial^p}{\partial t^p} G(x,t)$. Consequently,

$$\begin{aligned} I_2 &= \sum_{m=1}^{q+1} \sum_{j=0}^{m-1} c_{qmj} k^q \left[v^{(q-(m-1-j))}\right]_{x_1} \int_{-1}^{1} (s^2-1)^{q+1} G_t^{(m-1-j)}(x, x_1+\epsilon s)\,ds + \mathcal{O}(\epsilon) \\ &= \sum_{m=1}^{q+1} \sum_{j=0}^{m-1} c_{qmj} k^q \left[v^{(q-(m-1-j))}\right]_{x_1} G_t^{(m-1-j)}(x,x_1) \int_{-1}^{1} (s^2-1)^{q+1}\,ds + \mathcal{O}(\epsilon) \\ &= \sum_{m=0}^{q} c_{qm} k^q G_t^{(q-m)}(x,x_1) \left[v^{(m)}\right]_{x_1} + \mathcal{O}(\epsilon). \end{aligned}$$

Letting $\epsilon \to 0$, we obtain

$$\pi v(x) - v(x) = \frac{k^q}{q!} \int_a^b v^{(q+1)}(t) G(x,t)\,dt + \sum_{m=0}^{q} c_{qm} k^q G_t^{(q-m)}(x,x_1) \left[v^{(m)}\right]_{x_1},$$

for $x \in [a,b] \backslash \{x_1\}$. By continuity this holds also when $x = x_1$. From Lemma 2.1, we know that $G_t^{(q-m)}(x,x_1) = k^{-(q-m)} G_{t,q-m}(x,x_1)$ where $G_{t,q-m}$ is bounded. Hence,

$$\begin{aligned} \pi v(x) - v(x) &= \frac{k^q}{q!} \int_a^b v^{(q+1)}(t) G(x,t)\,dt \\ &\quad + \sum_{m=0}^{q} c_{qm} k^{q-(q-m)} G_{t,q-m}(x,x_1) \left[v^{(m)}\right]_{x_1} \\ &= \frac{k^q}{q!} \int_a^b v^{(q+1)}(t) G(x,t)\,dt + \sum_{m=0}^{q} C_{qm}(x,x_1) k^m \left[v^{(m)}\right]_{x_1}, \end{aligned}$$

where $C_{qm}(x,x_1)$ is bounded on $[a,b] \times [a,b]$ independent of k. The second result, (4.7), is proved similarly. □

We now extend this to a representation of derivatives of the interpolation error, corresponding to Theorem 2.3.

Theorem 4.2. (Peano kernel theorem for piecewise smooth functions II) *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a constant $C > 0$, depending only on the definition of the interpolant (and not on k), and functions $G_p : [a,b] \times [a,b] \to \mathbb{R}$, $p = 0, \ldots, q$, such that for v piecewise $\mathcal{C}^{q+1}$ on $[a,b]$ with discontinuities at $a < x_1 < x_2 < \ldots < x_n < b$, we have*

$$\begin{aligned} (\pi v)^{(p)}(x) - v^{(p)}(x) &= \frac{k^{q+1-p}}{q!} \frac{1}{k} \int_a^b v^{(q+1)}(t) G_p(x,t)\,dt \\ &\quad + \sum_{j=1}^{n} \sum_{m=0}^{q} C_{qjmp}(x) k^{m-p} [v^{(m)}]_{x_j}, \end{aligned} \tag{4.8}$$

for $p = 0, \ldots, q$, with $|G_p(x,t)| \le C$ for all $x, t \in [a,b] \times [a,b]$, and $|C_{qjmp}(x)| \le C$ for all $x \in [a,b] \setminus \{x_1, \ldots, x_n\}$.

Proof. For $p = 0$, the result follows from Theorem 4.1 with $G_0 = G$ and $C_{qjm0} = C_{qjm}$. For $p = 1, \ldots, q$, we differentiate (4.6) with respect to x, to obtain

$$(\pi v)^{(p)}(x) - v^{(p)}(x) = \frac{k^q}{q!} \int_a^b v^{(q+1)}(t) \frac{d^p}{dx^p} G(x,t)\, dt + \sum_{j=1}^{n} \sum_{m=0}^{q} C_{qjm}^{(p)}(x) k^m [v^{(m)}]_{x_j}.$$

From Lemma 2.1, we know that $\frac{d^p}{dx^p} G(x,t) = k^{-p} G_{x,p}(x,t)$, where $G_{x,p}$ is bounded on $[a,b] \times [a,b]$. Furthermore, from the proof of Theorem 4.1, we know that

$$\begin{aligned} C_{qjm}^{(p)}(x) &= \frac{d^p}{dx^p} C_{qjm}(x) = \frac{d^p}{dx^p} c_{qm} G_{t,q-m}(x, x_j) \\ &= c_{qm} k^{q-m} \frac{\partial^p}{\partial x^p} \frac{\partial^{q-m}}{\partial t^{q-m}} G(x, x_j), \end{aligned}$$

and so, by Lemma 2.1,

$$C_{qjm}^{(p)}(x) = c_{qm} k^{q-m} k^{-(q-m+p)} G_{t,x,q-m,p}(x, x_j) = C_{qjmp}(x) k^{-p},$$

where each C_{qjmp} is bounded on $[a,b] \backslash \{x_1, \ldots, x_n\}$. □

4.2. **Interpolation estimates.** The following corollary, corresponding to Corollary 3.1, is a simple consequence of Theorem 4.2.

Corollary 4.1. *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a constant $C = C(q) > 0$, such that for all v piecewise $\mathcal{C}^{q+1}$ on $[a,b]$ with discontinuities at $a < x_1 < \ldots < x_n < b$,*

$$\|(\pi v)^{(p)} - v^{(p)}\| \leq C k^{r+1-p} \|v^{(r+1)}\| + C \sum_{j=1}^{n} \sum_{m=0}^{r} k^{m-p} \left| \left[v^{(m)} \right]_{x_j} \right|, \tag{4.9}$$

for $p = 0, \ldots, r+1$, $r = 0, \ldots, q$.

Proof. See proof of Corollary 3.1. □

We also obtain the following estimate for derivatives of the interpolant, corresponding to Corollary 3.2.

Corollary 4.2. *If π is linear and bounded on V, and is exact on $\mathcal{P}^q([a,b]) \subset V$, then there is a constant $C = C(q) > 0$ and a constant $C' = C'(q,n) > 0$, such that for all v piecewise $\mathcal{C}^{q+1}$ on $[a,b]$ with discontinuities at $a < x_1 < \ldots < x_n < b$,*

$$\begin{aligned} \|(\pi v)^{(p)}\| &\leq C \|v^{(p)}\| + C \sum_{j=1}^{n} \sum_{m=0}^{p-1} k^{m-p} \left| \left[v^{(m)} \right]_{x_j} \right| \\ &\leq C' \left(\|v^{(p)}\| + \sum_{m=0}^{p-1} k^{m-p} \|v^{(m)}\| \right), \end{aligned} \tag{4.10}$$

for $p = 0, \ldots, q$.

Proof. It is clear that (4.10) holds when $p = 0$, since π is bounded. As in the proof of Corollary 3.2, we add and subtract $v^{(p)}$ for $0 < p \leq q$, to obtain

$$\|(\pi v)^{(p)}\| \leq \|(\pi v)^{(p)} - v^{(p)}\| + \|v^{(p)}\|.$$

Since π is exact on $\mathcal{P}^q([a,b])$, it is exact on $\mathcal{P}^{p-1}([a,b]) \subset \mathcal{P}^q([a,b])$ for $p \leq q$. It follows by Corollary 4.1 that

$$\begin{aligned} \|(\pi v)^{(p)}\| &\leq C k^{(p-1)+1-p} \|v^{((p-1)+1)}\| + C \sum_{j=1}^{n} \sum_{m=0}^{p-1} k^{m-p} \left| \left[v^{(m)} \right]_{x_j} \right| + \|v^{(p)}\| \\ &\leq C \|v^{(p)}\| + C \sum_{j=1}^{n} \sum_{m=0}^{p-1} k^{m-p} \left| \left[v^{(m)} \right]_{x_j} \right| \\ &\leq C \|v^{(p)}\| + 2Cn \sum_{m=0}^{p-1} k^{m-p} \|v^{(m)}\| \\ &\leq C' \left(\|v^{(p)}\| + \sum_{m=0}^{p-1} k^{m-p} \|v^{(m)}\| \right). \end{aligned}$$

□

5. Two special interpolants

In this section, we use the results of Sections 2–4 to prove interpolation estimates for two special interpolants.

For the mcG(q) method, we define the following interpolant:

$$\begin{aligned} &\pi_{\text{cG}}^{[q]} : V \to \mathcal{P}^q([a,b]), \\ &\pi_{\text{cG}}^{[q]} v(a) = v(a) \text{ and } \pi_{\text{cG}}^{[q]} v(b) = v(b), \\ &\int_a^b (v - \pi_{\text{cG}}^{[q]} v) w \, dx = 0 \quad \forall w \in \mathcal{P}^{q-2}([a,b]), \end{aligned} \tag{5.1}$$

where V denotes the set of functions that are piecewise $\mathcal{C}^{q+1}$ and bounded on $[a,b]$. In other words, $\pi_{\text{cG}}^{[q]} v$ is the polynomial of degree q that interpolates v at the end-points of the interval $[a,b]$ and additionally satisfies $q-1$ projection conditions. This is illustrated in Figure 3. We also define the dual interpolant $\pi_{\text{cG}^*}^{[q]}$ as the standard L_2-projection onto $\mathcal{P}^{q-1}([a,b])$.

For the mdG(q) method, we define the following interpolant:

$$\begin{aligned} &\pi_{\text{dG}}^{[q]} : V \to \mathcal{P}^q([a,b]), \\ &\pi_{\text{dG}}^{[q]} v(b) = v(b), \\ &\int_a^b (v - \pi_{\text{dG}}^{[q]} v) w \, dx = 0 \quad \forall w \in \mathcal{P}^{q-1}([a,b]), \end{aligned} \tag{5.2}$$

that is, $\pi_{\text{dG}}^{[q]} v$ is the polynomial of degree q that interpolates v at the right end-point of the interval $[a,b]$ and additionally satisfies q projection conditions. This is illustrated in Figure 4. The dual interpolant $\pi_{\text{dG}^*}^{[q]}$ is defined

similarly, with the only difference that we use the left end-point $x = a$ for interpolation.

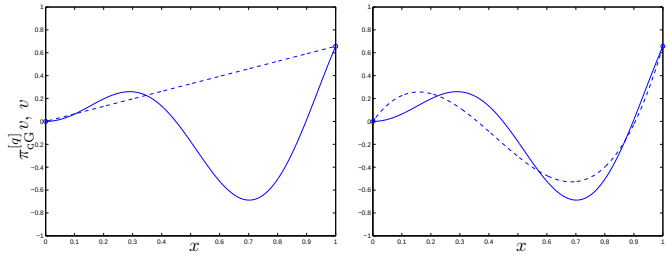

FIGURE 3. The interpolant $\pi_{\mathrm{cG}}^{[q]} v$ (dashed) of the function $v(x) = x\sin(7x)$ (solid) on $[0, 1]$ for $q = 1$ (left) and $q = 3$ (right).

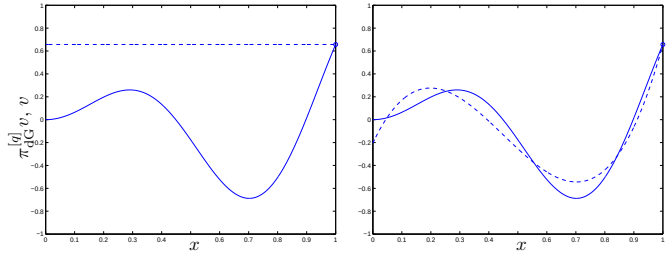

FIGURE 4. The interpolant $\pi_{\mathrm{dG}}^{[q]} v$ (dashed) of the function $v(x) = x\sin(7x)$ (solid) on $[0, 1]$ for $q = 0$ (left) and $q = 3$ (right).

It is clear that both $\pi_{\mathrm{cG}}^{[q]}$ and $\pi_{\mathrm{dG}}^{[q]}$ are linear and so, by Lemma 3.1, we only have to show that $\pi_{\mathrm{cG}}^{[q]}$ and $\pi_{\mathrm{dG}}^{[q]}$ are exact on $\mathcal{P}^q([a, b])$.

Lemma 5.1. *The two interpolants $\pi_{\mathrm{cG}}^{[q]}$ and $\pi_{\mathrm{dG}}^{[q]}$ are exact on $\mathcal{P}^q([a, b])$, that is,*

$$\pi_{\mathrm{cG}}^{[q]} v = v \quad \forall v \in \mathcal{P}^q([a, b]), \tag{5.3}$$

and

$$\pi_{\mathrm{dG}}^{[q]} v = v \quad \forall v \in \mathcal{P}^q([a, b]). \tag{5.4}$$

Proof. To prove (5.3), take $v \in \mathcal{P}^q([a, b])$ and note that $p = \pi_{\mathrm{cG}}^{[q]} v - v \in \mathcal{P}^q([a, b])$. Since $p(a) = p(b) = 0$, p has at most $q - 2$ zeros within (a, b) and so we can take $w \in \mathcal{P}^{q-2}([a, b])$ with $pw \geq 0$ on $[a, b]$. By definition, $\int_a^b pw\,dx = 0$, and so we conclude that $p = 0$.

To prove (5.4), take $p = \pi_{\mathrm{dG}}^{[q]} v - v \in \mathcal{P}^q([a, b])$. Then, $p(b) = 0$ and so p has at most $q - 1$ zeros within (a, b). Take now $w \in \mathcal{P}^{q-1}([a, b])$ with $pw \geq 0$ on $[a, b]$. By definition, $\int_a^b pw\,dt = 0$, and so again we conclude that $p = 0$. □

The desired interpolation estimates now follow Corollaries 3.1, 3.2, 4.1, and 4.2.

Theorem 5.1. (Estimates for $\pi_{\mathrm{cG}}^{[q]}$ and $\pi_{\mathrm{dG}}^{[q]}$) *For any $q \geq 1$, there is a constant $C = C(q) > 0$, such that for all $v \in \mathcal{C}^{r+1}([a, b])$,*

$$\|(\pi_{\mathrm{cG}}^{[q]} v)^{(p)} - v^{(p)}\| \leq Ck^{r+1-p}\|v^{(r+1)}\|, \tag{5.5}$$

for $p = 0, \ldots, r + 1$, $r = 0, \ldots, q$, and

$$\|(\pi_{\mathrm{cG}}^{[q]} v)^{(p)}\| \leq C\|v^{(p)}\|, \tag{5.6}$$

for $p = 0, \ldots, q$. Furthermore, for any $q \geq 0$, there is a constant $C = C(q) > 0$, such that for all $v \in \mathcal{C}^{r+1}([a, b])$,

$$\|(\pi_{\mathrm{dG}}^{[q]} v)^{(p)} - v^{(p)}\| \leq Ck^{r+1-p}\|v^{(r+1)}\|, \tag{5.7}$$

for $p = 0, \ldots, r + 1$, $r = 0, \ldots, q$, and

$$\|(\pi_{\mathrm{dG}}^{[q]} v)^{(p)}\| \leq C\|v^{(p)}\|, \tag{5.8}$$

for $p = 0, \ldots, q$.

Theorem 5.2. (Estimates for $\pi_{\mathrm{cG}}^{[q]}$ and $\pi_{\mathrm{dG}}^{[q]}$ II) *For any $q \geq 1$ and any $n \geq 0$, there is a constant $C = C(q) > 0$, such that for all v piecewise $\mathcal{C}^{r+1}$ on $[a, b]$ with discontinuities at $a < x_1 < \ldots < x_n < b$,*

$$\|(\pi_{\mathrm{cG}}^{[q]} v)^{(p)} - v^{(p)}\| \leq Ck^{r+1-p}\|v^{(r+1)}\| + C\sum_{j=1}^{n}\sum_{m=0}^{r} k^{m-p}\left|\left[v^{(m)}\right]_{x_j}\right|, \tag{5.9}$$

for $p = 0, \ldots, r + 1$, $r = 0, \ldots, q$, and

$$\|(\pi_{\mathrm{cG}}^{[q]} v)^{(p)}\| \leq C\|v^{(p)}\| + C\sum_{j=1}^{n}\sum_{m=0}^{p-1} k^{m-p}\left|\left[v^{(m)}\right]_{x_j}\right|, \tag{5.10}$$

for $p = 0, \ldots, q$. Furthermore, for any $q \geq 0$ and any $n \geq 0$, there is a constant $C = C(q) > 0$, such that for all v piecewise $\mathcal{C}^{r+1}$ on $[a, b]$ with discontinuities at $a < x_1 < \ldots < x_n < b$,

$$\|(\pi_{\mathrm{dG}}^{[q]} v)^{(p)} - v^{(p)}\| \leq Ck^{r+1-p}\|v^{(r+1)}\| + C\sum_{j=1}^{n}\sum_{m=0}^{r} k^{m-p}\left|\left[v^{(m)}\right]_{x_j}\right|, \tag{5.11}$$

for $p = 0, \ldots, r+1$, $r = 0, \ldots, q$, *and*

$$\|(\pi_{\mathrm{dG}}^{[q]} v)^{(p)}\| \leq C\|v^{(p)}\| + C \sum_{j=1}^{n} \sum_{m=0}^{p-1} k^{m-p} \left| \left[v^{(m)} \right]_{x_j} \right|, \tag{5.12}$$

for $p = 0, \ldots, q$.

Note that the corresponding estimates hold for the dual versions of the two interpolants, $\pi_{\mathrm{cG}^*}^{[q]}$ and $\pi_{\mathrm{dG}^*}^{[q]}$, the only difference being that $r \leq q-1$ for $\pi_{\mathrm{cG}^*}^{[q]}$.

Finally, we note the following properties of the two interpolants, which is of importance for the a priori error analysis.

Lemma 5.2. *For any* $v \in \mathcal{C}([a,b])$ *and any* $q \geq 1$, *we have*

$$\int_a^b \left(\frac{d}{dx}(v - \pi_{\mathrm{cG}}^{[q]} v)\right) w \, dx = 0 \quad \forall w \in \mathcal{P}^{q-1}([a,b]). \tag{5.13}$$

Proof. For any $w \in \mathcal{P}^{q-1}([a,b])$, we integrate by parts to get

$$\int_a^b \left(\frac{d}{dx}(v - \pi_{\mathrm{cG}}^{[q]} v)\right) w \, dx = \left[(v - \pi_{\mathrm{cG}}^{[q]} v) w \right]_a^b - \int_a^b (v - \pi_{\mathrm{cG}}^{[q]} v) \, w' \, dx = 0,$$

by the definition of $\pi_{\mathrm{cG}}^{[q]}$. □

Lemma 5.3. *For any* $v \in \mathcal{C}([a,b])$ *and any* $q \geq 0$, *we have*

$$[v(a) - \pi_{\mathrm{dG}}^{[q]} v(a)] w(a) + \int_a^b \left(\frac{d}{dx}(v - \pi_{\mathrm{dG}}^{[q]} v)\right) w \, dx = 0 \quad \forall w \in \mathcal{P}^q([a,b]). \tag{5.14}$$

Proof. Integrate by parts as in the proof of Lemma 5.2 and use the definition of $\pi_{\mathrm{dG}}^{[q]}$. □

References

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

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

[3] M.J.D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1988.

ESTIMATES OF DERIVATIVES AND JUMPS ACROSS ELEMENT BOUNDARIES FOR MULTI-ADAPTIVE GALERKIN SOLUTIONS OF ODES

ANDERS LOGG

Abstract. As an important step in the a priori error analysis of the multi-adaptive Galerkin methods mcG(q) and mdG(q), we prove estimates of derivatives and jumps across element boundaries for the multi-adaptive discrete solutions. The proof is by induction and is based on a new representation formula for the solutions.

1. Introduction

In [3], we proved special interpolation estimates as a preparation for the derivation of a priori error estimates for the multi-adaptive Galerkin methods mcG(q) and mdG(q), presented earlier in [1, 2]. As further preparation, we here derive estimates for derivatives, and jumps in function value and derivatives for the multi-adaptive solutions.

We first derive estimates for the general non-linear problem,

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

where $u : [0, T] \to \mathbb{R}^N$ is the solution to be computed, $u_0 \in \mathbb{R}^N$ a given initial 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. We also derive estimates for the linear problem,

$$\begin{aligned} \dot{u}(t) + A(t)u(t) &= 0, \quad t \in (0, T], \\ u(0) &= u_0, \end{aligned} \tag{1.2}$$

with $A(t)$ a bounded $N \times N$-matrix.

Furthermore, we prove the corresponding estimates for the discrete dual solution Φ, corresponding to (1.1) or (1.2). For the non-linear problem (1.1), the discrete dual solution Φ is defined as a Galerkin solution of the continuous linearized dual problem

$$\begin{aligned} -\dot{\phi}(t) &= J^\top(\pi u, U, t)\phi(t) + g(t), \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{1.3}$$

with given data $g : [0, T) \to \mathbb{R}^N$ and $\psi \in \mathbb{R}^N$, where

$$J^\top(\pi u, U, t) = \left(\int_0^1 \frac{\partial f}{\partial u}(s\pi u(t) + (1 - s)U(t), t)\, ds \right)^\top \tag{1.4}$$

is the transpose of the Jacobian of the right-hand side f, evaluated at an appropriate mean value of the approximate Galerkin solution U of (1.1) and an interpolant πu of the exact solution u. We will use the notation

$$f^*(\phi, \cdot) = J^\top(\pi u, U, \cdot)\phi + g, \tag{1.5}$$

to write the dual problem (1.3) in the form

$$\begin{aligned} -\dot{\phi}(t) &= f^*(\phi(t), t), \quad t \in [0, T), \\ \phi(T) &= \psi. \end{aligned} \tag{1.6}$$

We remind the reader that the discrete dual solution Φ is a Galerkin approximation, given by the mcG(q)* or mdG(q)* method defined in [4], of the exact solution ϕ of (1.3), and refer to [4] for the exact definition.

For the linear problem (1.2), the discrete dual solution Φ is defined as a Galerkin solution of the continuous dual problem

$$\begin{aligned} -\dot{\phi}(t) + A^\top(t)\phi(t) &= g, \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{1.7}$$

or $-\dot{\phi}(t) = f^*(\phi(t), t)$, with the notation $f^*(\phi, \cdot) = -A^\top\phi + g$.

1.1. Notation.

For a detailed description of the multi-adaptive Galerkin methods, we refer the reader to [1, 2, 6, 4, 5]. In particular, we refer to [1] or [4] for the exact 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*:

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

We denote the length of a time slab by $K_n = T_n - T_{n-1}$. For a given local interval I_{ij}, we denote the time slab $\mathcal{T}$, for which $I_{ij} \in \mathcal{T}$, by $\mathcal{T}(i, j)$.

Date: March 15, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, a priori error estimates, linear, parabolic.

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

Since different components use different time steps, a local interval I_{ij} may contain nodal points for other components, that is, some $t_{i'j'} \in (t_{i,j-1}, t_{ij})$. We denote the set of such internal nodes on each local interval I_{ij} by $\mathcal{N}_{ij}$.

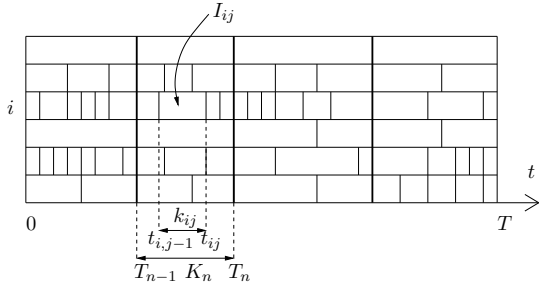

FIGURE 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

1.2. **Outline of the paper.** In Section 2, we show that the multi-adaptive Galerkin solutions (including discrete dual solutions) can be expressed as certain interpolants. It is known before [1] that the mcG(q) solution of (1.1) satisfies the relation

$$U_i(t_{ij}) = (u_0)_i + \int_0^{t_{ij}} f_i(U(t), t)\,dt, \quad j = 1, \ldots, M_i, \quad i = 1, \ldots, N, \tag{1.8}$$

with a similar relation for the mdG(q) solution, but this does not hold with t_{ij} replaced by an arbitrary $t \in [0, T]$. However, we prove that

$$U(t) = \pi_{\text{cG}}^{[q]} \left[u_0 + \int_0^{\cdot} f(U(s), s)\,ds \right](t), \tag{1.9}$$

for all $t \in [0, T]$, with $\pi_{\text{cG}}^{[q]}$ a special interpolant. This new way of expressing the multi-adaptive Galerkin solutions is a powerful tool and it is used extensively throughout the remainder of the paper.

In Section 3, we prove a chain rule for higher-order derivatives, which we use in Section 4, together with the representations of Section 2, to prove the desired estimates for the non-linear problem (1.1) by induction. Finally, in Section 5, we prove the corresponding estimates for linear problems.

2. A representation formula for the solutions

The proof of estimates for derivatives and jumps of the multi-adaptive Galerkin solutions is based on expressing the solutions as certain interpolants. These representations are obtained as follows. Let U be the mcG(q)

or mdG(q) solution of (1.1) and define for $i = 1, \ldots, N$,

$$\tilde{U}_i(t) = u_i(0) + \int_0^t f_i(U(s), s)\,ds. \tag{2.1}$$

Similarly, for Φ the mcG(q)* or mdG(q)* solution of (1.6), we define for $i = 1, \ldots, N$,

$$\tilde{\Phi}_i(t) = \psi_i + \int_t^T f_i^*(\Phi(s), s)\,ds. \tag{2.2}$$

We note that $\dot{\tilde{U}} = f(U, \cdot)$ and $-\dot{\tilde{\Phi}} = f^*(\Phi, \cdot)$.

It now turns out that U can be expressed as an interpolant of $\tilde{U}$. Similarly, Φ can be expressed as an interpolant of $\tilde{\Phi}$. We derive these representations in Theorem 2.1 below for the mcG(q) and mcG(q)* methods, and in Theorem 2.2 for the mdG(q) and mdG(q)* methods. We remind the reader about the special interpolants $\pi_{\text{cG}}^{[q]}$, $\pi_{\text{cG}^*}^{[q]}$, $\pi_{\text{dG}}^{[q]}$, and $\pi_{\text{dG}^*}^{[q]}$, defined in [3].

Theorem 2.1. *The* mcG(q) *solution* U *of* (1.1) *can expressed in the form*

$$U = \pi_{\text{cG}}^{[q]} \tilde{U}. \tag{2.3}$$

Similarly, the mcG(q)* *solution* Φ *of* (1.6) *can be expressed in the form*

$$\Phi = \pi_{\text{cG}^*}^{[q]} \tilde{\Phi}, \tag{2.4}$$

that is, $U_i = \pi_{\text{cG}}^{[q_{ij}]} \tilde{U}_i$ *and* $\Phi_i = \pi_{\text{cG}^*}^{[q_{ij}]} \tilde{\Phi}_i$ *on each local interval* I_{ij}.

Proof. To prove (2.3), we note that if U is the mcG(q) solution of (1.1), then on each local interval I_{ij}, we have

$$\int_{I_{ij}} \dot{U}_i v_m\,dt = \int_{I_{ij}} f_i(U, \cdot) v_m\,dt, \quad m = 0, \ldots, q_{ij} - 1,$$

with $v_m(t) = ((t - t_{i,j-1})/k_{ij})^m$. On the other hand, by the definition of $\tilde{U}$, we have

$$\int_{I_{ij}} \dot{\tilde{U}}_i v_m\,dt = \int_{I_{ij}} f_i(U, \cdot) v_m\,dt, \quad m = 0, \ldots, q_{ij} - 1.$$

Integrating by parts and subtracting, we obtain

$$-\left[(U_i - \tilde{U}_i) v_m\right]_{t_{i,j-1}}^{t_{ij}} + \int_{I_{ij}} \left(U_i - \tilde{U}_i\right) \dot{v}_m\,dt = 0,$$

and thus, since $U_i(t_{i,j-1}) - \tilde{U}_i(t_{i,j-1}) = U_i(t_{ij}) - \tilde{U}_i(t_{ij}) = 0$,

$$\int_{I_{ij}} \left(U_i - \tilde{U}_i\right) \dot{v}_m\,dt = 0.$$

By the definition of the mcG(q)-interpolant $\pi_{\text{cG}}^{[q]}$, it now follows that $U_i = \pi_{\text{cG}}^{[q_{ij}]} \tilde{U}_i$ on I_{ij}.

To prove (2.4), we note that with Φ the mcG$(q)^*$ solution of (1.6), we have

$$-(\psi, v(T)) + \sum_{i=1}^{N}\sum_{j=1}^{M_i}\int_{I_{ij}} \Phi_i \dot{v}_i \, dt = \int_0^T (f^*(\Phi,\cdot), v)\, dt, \tag{2.5}$$

for all continuous test functions v of order $q = \{q_{ij}\}$ vanishing at $t = 0$. On the other hand, by the definition of $\tilde{\Phi}$, it follows that

$$-\int_{I_{ij}} \dot{\tilde{\Phi}}_i v_i \, dt = \int_{I_{ij}} f_i^*(\Phi,\cdot) v_i \, dt.$$

Integrating by parts, we obtain

$$-\left[\tilde{\Phi}_i v_i\right]_{t_{i,j-1}}^{t_{ij}} + \int_{I_{ij}} \tilde{\Phi}_i \dot{v}_i \, dt = \int_{I_{ij}} f_i^*(\Phi,\cdot) v_i \, dt,$$

and thus

$$-(\psi, v(T)) + \sum_{i=1}^{N}\sum_{j=1}^{M_i}\int_{I_{ij}} \tilde{\Phi}_i \dot{v}_i \, dt = \int_0^T (f^*(\Phi,\cdot), v)\, dt, \tag{2.6}$$

since $v(0) = 0$ and both $\tilde{\Phi}$ and v are continuous. Subtracting (2.5) and (2.6), it now follows that

$$\sum_{i=1}^{N}\sum_{j=1}^{M_i}\int_{I_{ij}} (\Phi_i - \tilde{\Phi}_i)\dot{v}_i \, dt = 0,$$

for all test functions v. We now take $\dot{v}_i = 0$ except on I_{ij}, and $\dot{v}_n = 0$ for $n \neq i$, to obtain

$$\int_{I_{ij}} (\Phi_i - \tilde{\Phi}_i) w \, dt = 0 \quad \forall w \in \mathcal{P}^{q_{ij}-1}(I_{ij}),$$

and so $\Phi_i = P^{[q_{ij}-1]}\tilde{\Phi}_i \equiv \pi_{\mathrm{cG}^*}^{[q_{ij}]}\tilde{\Phi}_i$ on I_{ij}. □

Theorem 2.2. *The* mdG(q) *solution* U *of* (1.1) *can expressed in the form*

$$U = \pi_{\mathrm{dG}}^{[q]}\tilde{U}. \tag{2.7}$$

Similarly, the mdG$(q)^*$ *solution* Φ *of* (1.6) *can be expressed in the form*

$$\Phi = \pi_{\mathrm{dG}^*}^{[q]}\tilde{\Phi}, \tag{2.8}$$

that is, $U_i = \pi_{\mathrm{dG}}^{[q_{ij}]}\tilde{U}_i$ *and* $\Phi_i = \pi_{\mathrm{dG}^*}^{[q_{ij}]}\tilde{\Phi}_i$ *on each local interval* I_{ij}.

Proof. To prove (2.7), we note that if U is the mdG(q) solution of (1.1), then on each local interval I_{ij}, we have

$$\int_{I_{ij}} \dot{U}_i v_m \, dt = \int_{I_{ij}} f_i(U,\cdot) v_m \, dt, \quad m = 1, \ldots, q_{ij},$$

with $v_m(t) = ((t - t_{i,j-1})/k_{ij})^m$. On the other hand, by the definition of $\tilde{U}$, we have

$$\int_{I_{ij}} \dot{\tilde{U}}_i v_m \, dt = \int_{I_{ij}} f_i(U,\cdot) v_m \, dt, \quad m = 1, \ldots, q_{ij}.$$

Integrating by parts and subtracting, we obtain

$$\int_{I_{ij}} \left(U_i - \tilde{U}_i\right) \dot{v}_m \, dt - \left(U_i(t_{ij}^-) - \tilde{U}(t_{ij})\right) = 0,$$

and thus, since $U_i(t_{ij}^-) = \tilde{U}_i(t_{ij})$,

$$\int_{I_{ij}} \left(U_i - \tilde{U}_i\right) \dot{v}_m \, dt = 0.$$

By the definition of the mdG(q)-interpolant $\pi_{\mathrm{dG}}^{[q]}$, it now follows that $U_i = \pi_{\mathrm{dG}}^{[q_{ij}]}\tilde{U}_i$ on I_{ij}.

The representation (2.8) of the dual solution follows directly, since the mdG$(q)^*$ method is identical to the mdG(q) method with time reversed. □

Remark 2.1. *The representations of the multi-adaptive Galerkin solutions as certain interpolants are presented here for the general non-linear problem* (1.1)*, but apply also to the linear problem* (1.2).

3. A chain rule for higher-order derivatives

To estimate higher-order derivatives, we face the problem of taking higher-order derivatives of $f(U(t), t)$ with respect to t. In this section, we derive a generalized version of the chain rule for higher-order derivatives. We also prove a basic estimate for the jump in a composite function.

Lemma 3.1. (Chain rule) *Let* $v : \mathbb{R}^N \to \mathbb{R}$ *be* $p > 0$ *times differentiable in all its variables, and let* $x : \mathbb{R} \to \mathbb{R}^N$ *be* p *times differentiable, so that*

$$v \circ x : \mathbb{R} \to \mathbb{R} \tag{3.1}$$

is p *times differentiable. Furthermore, let* $D^n v$ *denote the* n*th order tensor defined by*

$$D^n v \, w^1 \cdots w^n = \sum_{i_1=1}^{N} \cdots \sum_{i_n=1}^{N} \frac{\partial^n v}{\partial x_{i_1} \cdots \partial x_{i_n}} w_{i_1}^1 \cdots w_{i_n}^n,$$

for $w^1, \ldots, w^n \in \mathbb{R}^N$. *Then,*

$$\frac{d^p(v \circ x)}{dt^p} = \sum_{n=1}^{p} D^n v(x) \sum_{n_1,\ldots,n_n} C_{p,n_1,\ldots,n_n} x^{(n_1)} \cdots x^{(n_n)}, \tag{3.2}$$

where for each n *the sum* $\sum_{n_1,\ldots,n_n}$ *is taken over* $n_1 + \ldots + n_n = p$ *with* $n_i \geq 1$.

Proof. Repeated use of the chain rule and Leibniz rule gives

$$\begin{aligned}\frac{d^p(v\circ x)}{dt^p} &= \frac{d^{p-1}}{dt^{p-1}}Dv(x)x^{(1)} = \frac{d^{p-2}}{dt^{p-2}}\left[D^2v(x)x^{(1)}x^{(1)} + Dv(x)x^{(2)}\right]\\ &= \frac{d^{p-3}}{dt^{p-3}}\left[D^3v(x)x^{(1)}x^{(1)}x^{(1)} + D^2v(x)x^{(2)}x^{(1)} + \ldots + Dv(x)x^{(3)}\right]\\ &= \sum_{n=1}^{p} D^nv(x)\sum_{n_1,\ldots,n_n} C_{p,n_1,\ldots,n_n}x^{(n_1)}\cdots x^{(n_n)},\end{aligned}$$

where for each n the sum is taken over $n_1+\ldots+n_n = p$ with $n_i \geq 1$. □

To estimate the jump in function value and derivatives for the composite function $v\circ x$, we will need the following lemma.

Lemma 3.2. *With* $[A] = A^+ - A^-$, $\langle A\rangle = (A^+ + A^-)/2$ *and* $|A| = \max(|A^+|, |A^-|)$, *we have*

$$[AB] = [A]\langle B\rangle + \langle A\rangle[B], \tag{3.3}$$

and

$$|[A_1A_2\cdots A_n]| \leq \sum_{i=1}^{n} |[A_i]|\ \Pi_{j\neq i}|A_i|. \tag{3.4}$$

Proof. The proof of (3.3) is straightforward:

$$\begin{aligned}[A]\langle B\rangle + \langle A\rangle[B] &= (A^+ - A^-)(B^+ + B^-)/2 + (A^+ + A^-)(B^+ - B^-)/2\\ &= A^+B^+ - A^-B^- = [AB].\end{aligned}$$

It now follows that

$$\begin{aligned}|[A_1A_2\cdots A_n]| &= |[A_1(A_2\cdots A_n)\,]| = |\,[A_1]\langle A_2\cdots A_n\,\rangle + \langle A_1\rangle[A_2\cdots A_n]\,|\\ &\leq |[A_1]|\cdot|A_2|\cdots|A_n| + |A_1|\cdot|[A_2\cdots A_n]|\\ &\leq \sum_{i=1}^{n}|[A_i]|\ \Pi_{j\neq i}|A_i|.\end{aligned}$$

□

Using Lemma 3.1 and 3.2, we now prove basic estimates of derivatives and jumps for the composite function $v\circ x$. We will use the following notation: For $n\geq 0$, let $\|D^nv\|_{L_\infty(\mathbb{R},l_\infty)}$ be defined by

$$\|D^nv\,w^1\cdots w^n\|_{L_\infty(\mathbb{R})} \leq \|D^nv\|_{L_\infty(\mathbb{R},l_\infty)}\|w^1\|_{l_\infty}\cdots\|w^n\|_{l_\infty} \tag{3.5}$$

for all $w^1,\ldots,w^n\in\mathbb{R}^N$, with $\|D^nv\|_{L_\infty(\mathbb{R},l_\infty)} = \|v\|_{L_\infty(\mathbb{R})}$ for $n=0$, and define

$$\|v\|_{D^p(\mathbb{R})} = \max_{n=0,\ldots,p}\|D^nv\|_{L_\infty(\mathbb{R},l_\infty)}. \tag{3.6}$$

Lemma 3.3. *Let* $v:\mathbb{R}^N\to\mathbb{R}$ *be* $p\geq 0$ *times differentiable in all its variables, let* $x:\mathbb{R}\to\mathbb{R}^N$ *be* p *times differentiable, and let* $C_x>0$ *be a constant, such that* $\|x^{(n)}\|_{L_\infty(\mathbb{R},l_\infty)}\leq C_x^n$, *for* $n=1,\ldots,p$. *Then, there is a constant* $C=C(p)>0$, *such that*

$$\left\|\frac{d^p(v\circ x)}{dt^p}\right\|_{L_\infty(\mathbb{R})} \leq C\|v\|_{D^p(\mathbb{R})}C_x^p. \tag{3.7}$$

Proof. We first note that for $p=0$, (3.7) follows directly by the definition of $\|v\|_{D^p(\mathbb{R})}$. For $p>0$, we obtain by Lemma 3.1,

$$\left|\frac{d^p(v\circ x)}{dt^p}\right| \leq C\sum_{n=1}^{p}\sum_{n_1,\ldots,n_n}\left|D^n\,v(x)x^{(n_1)}\cdots x^{(n_n)}\right| \leq C\|v\|_{D^p(\mathbb{R})}C_x^p.$$

□

Lemma 3.4. *Let* $v:\mathbb{R}^N\to\mathbb{R}$ *be* $p+1\geq 1$ *times differentiable in all its variables, let* $x:\mathbb{R}\to\mathbb{R}^N$ *be* p *times differentiable, except possibly at some* $t\in\mathbb{R}$, *and let* $C_x>0$ *be a constant, such that* $\|x^{(n)}\|_{L_\infty(\mathbb{R},l_\infty)}\leq C_x^n$ *for* $n=1,\ldots,p$. *Then, there is a constant* $C=C(p)>0$, *such that*

$$\left|\left[\frac{d^p(v\circ x)}{dt^p}\right]_t\right| \leq C\|v\|_{D^{p+1}(\mathbb{R})}\sum_{n=0}^{p}C_x^{p-n}\|[x^{(n)}]_t\|_{l_\infty}. \tag{3.8}$$

Proof. We first note that for $p=0$, we have

$$\begin{aligned}\left|\left[\frac{d^p(v\circ x)}{dt^p}\right]_t\right| &= |[(v\circ x)]_t| = \left|v(x(t^+)) - v(x(t^-))\right|\\ &\leq \|Dv\|_{L_\infty(\mathbb{R},l_\infty)}\,\|[x]_t\|_{l_\infty},\end{aligned}$$

and so (3.8) holds for $p=0$. For $p>0$, we obtain by Lemma 3.1 and Lemma 3.2,

$$\begin{aligned}&\left|\left[\frac{d^p(v\circ x)}{dt^p}\right]_t\right| \leq C\sum_{n=1}^{p}\sum_{n_1,\ldots,n_n}\left|\left[D^nv(x)\,x^{(n_1)}\cdots x^{(n_n)}\right]_t\right|\\ &\leq C\sum_{n=1}^{p}\sum_{n_1,\ldots,n_n}\|D^{n+1}v\|_{L_\infty(\mathbb{R},l_\infty)}\|[x]_t\|_{l_\infty}C_x^p +\\ &\quad + \|D^nv\|_{L_\infty(\mathbb{R},l_\infty)}(\|[x^{(n_1)}]_t\|_{l_\infty}C_x^{p-n_1} + \ldots + \|[x^{(n_n)}]_t\|_{l_\infty}C_x^{p-n_n})\\ &\leq C\|v\|_{D^{p+1}(\mathbb{R})}\sum_{n=0}^{p}C_x^{p-n}\|[x^{(n)}]_t\|_{l_\infty}.\end{aligned}$$

□

4. Estimates of derivatives and jumps for the non-linear problem

We now derive estimates of derivatives and jumps for the multi-adaptive solutions of the general non-linear problem (1.1). To obtain the estimates

for the multi-adaptive solutions U and Φ, we first derive estimates for the functions $\tilde{U}$ and $\tilde{\Phi}$ defined in Section 2. These estimates are then used to derive estimates for U and Φ.

4.1. **Assumptions.** We make the following basic assumptions: Given a time slab $\mathcal{T}$, assume that for each pair of local intervals I_{ij} and I_{mn} within the time slab, we have

$$q_{ij} = q_{mn} = \bar{q}, \tag{A1}$$

and

$$k_{ij} > \alpha\, k_{mn}, \tag{A2}$$

for some $\bar{q} \geq 0$ and some $\alpha \in (0, 1)$. We also assume that the problem (1.1) is autonomous,

$$\frac{\partial f_i}{\partial t} = 0, \quad i = 1, \ldots, N. \tag{A3}$$

Note that dual problem is in general non-autonomous. Furthermore, assume that

$$\|f_i\|_{D^{\bar{q}+1}(\mathcal{T})} < \infty, \quad i = 1, \ldots, N, \tag{A4}$$

and take $\|f\|_{\mathcal{T}} \geq \max_{i=1,\ldots,N} \|f_i\|_{D^{\bar{q}+1}(\mathcal{T})}$, such that

$$\|d^p/dt^p(\partial f/\partial u)^\top(x(t))\|_{l_\infty} \leq \|f\|_{\mathcal{T}} C_x^p, \tag{4.5}$$

for $p = 0, \ldots, \bar{q}$, and

$$\|[d^p/dt^p(\partial f/\partial u)^\top(x(t))]_t\|_{l_\infty} \leq \|f\|_{\mathcal{T}} \sum_{n=0}^{p} C_x^{p-n} \|[x^{(n)}]_t\|_{l_\infty}, \tag{4.6}$$

for $p = 0, \ldots, \bar{q} - 1$, with the notation of Lemma 3.3 and Lemma 3.4. Note that assumption (A4) implies that each f_i is bounded by $\|f\|_{\mathcal{T}}$. We further assume that there is a constant $c_k > 0$, such that

$$k_{ij}\|f\|_{\mathcal{T}} \leq c_k, \tag{A5}$$

for each local interval I_{ij}. We summarize the list of assumptions as follows:

- (A1) the local orders q_{ij} are equal within each time slab;
- (A2) the local time steps k_{ij} are semi-uniform within each time slab;
- (A3) f is autonomous;
- (A4) f and its derivatives are bounded;
- (A5) the local time steps k_{ij} are small.

4.2. **Estimates for U.** To simplify the estimates, we introduce the following notation: For given $p > 0$, let $C_{U,p} \geq \|f\|_{\mathcal{T}}$ be a constant, such that

$$\|U^{(n)}\|_{L_\infty(\mathcal{T},l_\infty)} \leq C_{U,p}^n, \quad n = 1, \ldots, p. \tag{4.8}$$

For $p = 0$, we define $C_{U,0} = \|f\|_{\mathcal{T}}$. Temporarily, we will assume that there is a constant $c_k' > 0$, such that for each p,

$$k_{ij} C_{U,p} \leq c_k'. \tag{A5$'$}$$

This assumption will be removed below in Theorem 4.1. In the following lemma, we use assumptions (A1), (A3), and (A4) to derive estimates for $\tilde{U}$ in terms of $C_{U,p}$ and $\|f\|_{\mathcal{T}}$.

Lemma 4.1. (Derivative and jump estimates for $\tilde{U}$) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.1) *and define $\tilde{U}$ as in* (2.1)*. If assumptions* (A1)*,* (A3)*, and* (A4) *hold, then there is a constant $C = C(\bar{q}) > 0$, such that*

$$\|\tilde{U}^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \leq C C_{U,p-1}^p, \quad p = 1, \ldots, \bar{q} + 1, \tag{4.10}$$

and

$$\|[\tilde{U}^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \leq C \sum_{n=0}^{p-1} C_{U,p-1}^{p-n} \|[U^{(n)}]_{t_{i,j-1}}\|_{l_\infty}, \quad p = 1, \ldots, \bar{q} + 1, \tag{4.11}$$

for each local interval I_{ij}, where $t_{i,j-1}$ is an internal node of the time slab $\mathcal{T}$.

Proof. By definition, $\tilde{U}_i^{(p)} = \frac{d^{p-1}}{dt^{p-1}} f_i(U)$, and so the results follow directly by Lemma 3.3 and Lemma 3.4, noting that $\|f\|_{\mathcal{T}} \leq C_{U,p-1}$. □

By Lemma 4.1, we now obtain the following estimate for the size of the jump in function value and derivatives for U.

Lemma 4.2. (Jump estimates for U) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.1)*. If assumptions* (A1)–(A5) *and* (A5$'$) *hold, then there is a constant $C = C(\bar{q}, c_k, c_k', \alpha) > 0$, such that*

$$\|[U^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \leq C k_{ij}^{r+1-p} C_{U,r}^{r+1}, \quad p = 0, \ldots, r + 1, \quad r = 0, \ldots, \bar{q}, \tag{4.12}$$

for each local interval I_{ij}, where $t_{i,j-1}$ is an internal node of the time slab $\mathcal{T}$.

Proof. The proof is by induction. We first note that at $t = t_{i,j-1}$, we have

$$\begin{aligned} [U_i^{(p)}]_t &= \left(U_i^{(p)}(t^+) - \tilde{U}_i^{(p)}(t^+)\right) + \left(\tilde{U}_i^{(p)}(t^+) - \tilde{U}_i^{(p)}(t^-)\right) + \left(\tilde{U}_i^{(p)}(t^-) - U_i^{(p)}(t^-)\right) \\ &\equiv e_+ + e_0 + e_-. \end{aligned}$$

By Theorem 2.1 (or Theorem 2.2), U is an interpolant of $\tilde{U}$ and so, by Theorem 5.2 in [3], we have

$$|e_+| \leq C k_{ij}^{r+1-p} \|\tilde{U}_i^{(r+1)}\|_{L_\infty(I_{ij})} + C \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{r} k_{ij}^{m-p} |[\tilde{U}_i^{(m)}]_x|,$$

for $p = 0, \ldots, r+1$ and $r = 0, \ldots, \bar{q}$. Note that the second sum starts at $m = 1$ rather than at $m = 0$, since $\tilde{U}$ is continuous. Similarly, we have

$$|e_-| \le Ck_{i,j-1}^{r+1-p} \|\tilde{U}_i^{(r+1)}\|_{L_\infty(I_{i,j-1})} + C \sum_{x \in \mathcal{N}_{i,j-1}} \sum_{m=1}^{r} k_{i,j-1}^{m-p} |[\tilde{U}_i^{(m)}]_x|.$$

To estimate e_0, we note that $e_0 = 0$ for $p = 0$, since $\tilde{U}$ is continuous. For $p = 1, \ldots, \bar{q}+1$, Lemma 4.1 gives

$$|e_0| = |[\tilde{U}_i^{(p)}]_t| \le C \sum_{n=0}^{p-1} C_{U,p-1}^{p-n} \|[U^{(n)}]_t\|_{l_\infty}.$$

Using assumption (A2), and the estimates for e_+, e_0, and e_-, we obtain for $r = 0$ and $p = 0$,

$$\begin{aligned} |[U_i]_t| &\le Ck_{ij} \|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} + 0 + Ck_{i,j-1} \|\dot{\tilde{U}}_i\|_{L_\infty(I_{i,j-1})} \\ &\le C(1 + \alpha^{-1}) k_{ij} C_{U,0} = Ck_{ij} C_{U,0}. \end{aligned}$$

It now follows by assumption (A5), that for $r = 0$ and $p = 1$,

$$\begin{aligned} |[\dot{U}_i]_t| &\le C \|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} + CC_{U,0} \|[U]_t\|_{l_\infty} + C \|\dot{\tilde{U}}_i\|_{L_\infty(I_{i,j-1})} \\ &\le C(1 + k_{ij} C_{U,0}) C_{U,0} \le CC_{U,0}. \end{aligned}$$

Thus, (4.12) holds for $r = 0$. Assume now that (4.12) holds for $r = \bar{r} - 1 \ge 0$. Then, by Lemma 4.1 and assumption (A5′), it follows that

$$\begin{aligned} |e_+| &\le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} + C \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{\bar{r}} k_{ij}^{m-p} \sum_{n=0}^{m-1} C_{U,m-1}^{m-n} \|[U^n]_x\|_{l_\infty} \\ &\le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} + C \sum k_{ij}^{m-p} C_{U,m-1}^{m-n} k_{ij}^{(\bar{r}-1)+1-n} C_{U,\bar{r}-1}^{(\bar{r}-1)+1} \\ &\le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \left(1 + \sum (k_{ij} C_{U,\bar{r}-1})^{m-1-n}\right) \le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1}. \end{aligned}$$

Similarly, we obtain the estimate $|e_-| \le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1}$. Finally, we use Lemma 4.1 and assumption (A5′), to obtain the estimate

$$\begin{aligned} |e_0| &\le C \sum_{n=0}^{p-1} C_{U,p-1}^{p-n} \|[U^n]_t\|_{l_\infty} \le C \sum_{n=0}^{p-1} C_{U,p-1}^{p-n} k_{ij}^{(\bar{r}-1)+1-n} C_{U,\bar{r}-1}^{(\bar{r}-1)+1} \\ &= Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \sum_{n=0}^{p-1} (k_{ij} C_{U,\bar{r}})^{p-1-n} \le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1}. \end{aligned}$$

Summing up, we thus obtain $|[U_i^{(p)}]_t| \le |e_+| + |e_0| + |e_-| \le Ck_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1}$, and so (4.12) follows by induction. □

By Lemma 4.1 and Lemma 4.2, we now obtain the following estimate for derivatives of the solution U.

Theorem 4.1. (Derivative estimates for U) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.1). *If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|U^{(p)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le C \|f\|_{\mathcal{T}}^p, \quad p = 1, \ldots, \bar{q}. \tag{4.13}$$

Proof. By Theorem 2.1 (or Theorem 2.2), U is an interpolant of $\tilde{U}$ and so, by Theorem 5.2 in [3], we have

$$\begin{aligned} \|U_i^{(p)}\|_{L_\infty(I_{ij})} &= \|(\pi \tilde{U}_i)^{(p)}\|_{L_\infty(I_{ij})} \\ &\le C' \|\tilde{U}_i^{(p)}\|_{L_\infty(I_{ij})} + C' \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{p-1} k_{ij}^{m-p} |[\tilde{U}_i^{(m)}]_x|, \end{aligned}$$

for some constant $C' = C'(\bar{q})$. For $p = 1$, we thus obtain the estimate

$$\|\dot{U}_i\|_{L_\infty(I_{ij})} \le C' \|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} = C' \|f_i(U)\|_{L_\infty(I_{ij})} \le C' \|f\|_{\mathcal{T}},$$

by assumption (A4), and so (4.13) holds for $p = 1$.

For $p = 2, \ldots, \bar{q}$, assuming that (A5′) holds for $C_{U,p-1}$, we use Lemma 4.1, Lemma 4.2 (with $r = p-1$), and assumption (A2), to obtain

$$\begin{aligned} \|U_i^{(p)}\|_{L_\infty(I_{ij})} &\le CC_{U,p-1}^p + C \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{p-1} k_{ij}^{m-p} \sum_{n=0}^{m-1} C_{U,m-1}^{m-n} \|[U^{(n)}]_x\|_{l_\infty} \\ &\le CC_{U,p-1}^p + C \sum k_{ij}^{m-p} C_{U,m-1}^{m-n} k_{ij}^{(p-1)+1-n} C_{U,p-1}^{(p-1)+1} \\ &\le CC_{U,p-1}^p \left(1 + \sum (k_{ij} C_{U,m-1})^{m-n}\right) \le CC_{U,p-1}^p, \end{aligned}$$

where $C = C(\bar{q}, c_k, c_k', \alpha)$. This holds for all components i and all local intervals I_{ij} within the time slab $\mathcal{T}$, and so

$$\|U^{(p)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le CC_{U,p-1}^p, \quad p = 1, \ldots, \bar{q},$$

where by definition $C_{U,p-1}$ is a constant, such that $\|U^{(n)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le C_{U,p-1}^n$ for $n = 1, \ldots, p-1$. Starting at $p = 1$, we now define $C_{U,1} = C_1 \|f\|_{\mathcal{T}}$ with $C_1 = C' = C'(\bar{q})$. It then follows that (A5′) holds for $C_{U,1}$ with $c_k' = C' c_k$, and thus

$$\|U^{(2)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le CC_{U,2-1}^2 = CC_{U,1}^2 \equiv C_2 \|f\|_{\mathcal{T}}^2,$$

where $C_2 = C_2(\bar{q}, c_k, \alpha)$. We may thus define $C_{U,2} = \max(C_1 \|f\|_{\mathcal{T}}, \sqrt{C_2} \|f\|_{\mathcal{T}})$. Continuing, we note that (A5′) holds for $C_{U,2}$, and thus

$$\|U^{(3)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le CC_{U,3-1}^3 = CC_{U,2}^3 \equiv C_3 \|f\|_{\mathcal{T}}^3,$$

where $C_3 = C_3(\bar{q}, c_k, \alpha)$. In this way, we obtain a sequence of constants $C_1, \ldots, C_{\bar{q}}$, depending only on $\bar{q}$, c_k, and α, such that $\|U^{(p)}\|_{L_\infty(\mathcal{T}, l_\infty)} \le C_p \|f\|_{\mathcal{T}}^p$ for $p = 1, \ldots, \bar{q}$, and so (4.13) follows if we take $C = \max_{i=1,\ldots,\bar{q}} C_i$. □

Having now removed the additional assumption (A5′), we obtain the following version of Lemma 4.2.

Theorem 4.2. (Jump estimates for U) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.1). *If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|[U^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \le C k_{ij}^{\bar{q}+1-p} \|f\|_{\mathcal{T}}^{\bar{q}+1}, \quad p = 0, \dots, \bar{q}, \tag{4.14}$$

for each local interval I_{ij}, where $t_{i,j-1}$ is an internal node of the time slab $\mathcal{T}$.

4.3. **Estimates for Φ.** To obtain estimates corresponding to those of Theorem 4.1 and Theorem 4.2 for the discrete dual solution Φ, we need to consider the fact that $f^* = f^*(\phi, \cdot) = J^\top \phi$ is linear and non-autonomous. To simplify the estimates, we introduce the following notation: For given $p \ge 0$, let $C_{\Phi,p} \ge \|f\|_{\mathcal{T}}$ be a constant, such that

$$\|\Phi^{(n)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C_{\Phi,p}^n \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad n = 0, \dots, p. \tag{4.15}$$

Temporarily, we will assume that for each p there is a constant $c_k'' > 0$, such that

$$k_{ij} C_{\Phi,p} \le c_k''. \tag{A5″}$$

This assumption will be removed below in Theorem 4.3. Now, to obtain estimates for Φ, we first need to derive estimates of derivatives and jumps for J.

Lemma 4.3. *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.1), *and let πu be an interpolant, of order $\bar{q}$, of the exact solution u of* (1.1). *If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\left\| \frac{d^p J^\top(\pi u, U)}{dt^p} \right\|_{L_\infty(\mathcal{T},l_\infty)} \le C \|f\|_{\mathcal{T}}^{p+1}, \quad p = 0, \dots, \bar{q}, \tag{4.17}$$

and

$$\left\| \left[\frac{d^p J^\top(\pi u, U)}{dt^p} \right]_{t_{i,j-1}} \right\|_{l_\infty} \le C k_{ij}^{\bar{q}+1-p} \|f\|_{\mathcal{T}}^{\bar{q}+2}, \quad p = 0, \dots, \bar{q}-1, \tag{4.18}$$

for each local interval I_{ij}, where $t_{i,j-1}$ is an internal node of the time slab $\mathcal{T}$.

Proof. Since f is autonomous by assumption (A3), we have

$$J(\pi u(t), U(t)) = \int_0^1 \frac{\partial f}{\partial u}(s\pi u(t) + (1-s)U(t))\, ds = \int_0^1 \frac{\partial f}{\partial u}(x_s(t))\, ds,$$

with $x_s(t) = s\pi u(t) + (1-s)U(t)$. Noting that $\|u^{(n)}(t)\|_{l_\infty} \le C\|f\|_{\mathcal{T}}^n$ by (1.1), it follows by Theorem 4.1 and an interpolation estimate, that $\|x_s^{(n)}(t)\|_{l_\infty} \le C\|f\|_{\mathcal{T}}^n$, and so (4.17) follows by assumption (A4).

At $t = t_{i,j-1}$, we obtain, by Theorem 4.2 and an interpolation estimate,

$$\begin{aligned}
|[x_{si}^{(n)}]_t| &\le s\,|[(\pi u_i)^{(n)}]_t| + (1-s)\,|[U_i^{(n)}]_t| \le |[(\pi u_i)^{(n)}]_t| + |[U_i^{(n)}]_t| \\
&\le |(\pi u_i)^{(n)}(t^+) - u_i^{(n)}(t)| + |u_i^{(n)}(t) - (\pi u_i)^{(n)}(t^-))| + C k_{ij}^{\bar{q}+1-n} \|f\|_{\mathcal{T}}^{\bar{q}+1} \\
&\le C k_{ij}^{\bar{q}+1-n} \|u_i^{(\bar{q}+1)}\|_{L_\infty(I_{ij})} + C k_{i,j-1}^{\bar{q}+1-n} \|u_i^{(\bar{q}+1)}\|_{L_\infty(I_{i,j-1})} + C k_{ij}^{\bar{q}+1-n} \|f\|_{\mathcal{T}}^{\bar{q}+1} \\
&\le C k_{ij}^{\bar{q}+1-n} \|f\|_{\mathcal{T}}^{\bar{q}+1},
\end{aligned}$$

where we have also used assumption (A2). With similar estimates for other components which are discontinuous at $t = t_{i,j-1}$, the estimate (4.18) now follows by assumptions (A4) and (A5). □

Using these estimates for $J^\top$, we now derive estimates for $\tilde{\Phi}$, corresponding to the estimates for $\tilde{U}$ in Lemma 4.1.

Lemma 4.4. (Derivative and jump estimates for $\tilde{\Phi}$) *Let Φ be the* mcG(q)* *or* mdG(q)* *solution of* (1.3) *with $g = 0$, and define $\tilde{\Phi}$ as in* (2.2). *If assumptions* (A1)–(A5) *and* (A5″) *hold, then there is a constant $C = C(\bar{q}, c_k, c_k'', \alpha) > 0$, such that*

$$\|\tilde{\Phi}^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C C_{\Phi,p-1}^p \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 1, \dots, \bar{q}+1, \tag{4.19}$$

and

$$\|[\tilde{\Phi}^{(p)}]_{t_{ij}}\|_{l_\infty} \le C k_{ij}^{\bar{q}+2-p} \|f\|_{\mathcal{T}}^{\bar{q}+2} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C \sum_{n=0}^{p-1} \|f\|_{\mathcal{T}}^{p-n} \|[\Phi^{(n)}]_{t_{ij}}\|_{l_\infty}, \tag{4.20}$$

$p = 1, \dots, \bar{q}$, *for each local interval I_{ij}, where t_{ij} is an internal node of the time slab $\mathcal{T}$.*

Proof. By definition, $\dot{\tilde{\Phi}} = -f^*(\Phi, \cdot) = -J(\pi u, U)^\top \Phi$. It follows that

$$\tilde{\Phi}^{(p)} = -\frac{d^{p-1}}{dt^{p-1}} J^\top \Phi = -\sum_{n=0}^{p-1} \binom{p-1}{n} \left(\frac{d^{p-1-n}}{dt^{p-1-n}} J^\top \right) \Phi^{(n)},$$

and so, by Lemma 4.3,

$$\|\tilde{\Phi}^{(p)}(t)\|_{l_\infty} \le C \sum_{n=0}^{p-1} \|f\|_{\mathcal{T}}^{p-n} C_{\Phi,p-1}^n \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \le C C_{\Phi,p-1}^p \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)},$$

for $0 \le p-1 \le \bar{q}$. To estimate the jump at $t = t_{ij}$, we use Lemma 3.2, Lemma 4.3, and assumption (A5″), to obtain

$$\begin{aligned}
\|[\tilde{\Phi}^{(p)}]_t\|_{l_\infty} &\le C\sum_{n=0}^{p-1}\left\|\left[\left(\frac{d^{p-1-n}}{dt^{p-1-n}}J^\top\right)\Phi^{(n)}\right]_t\right\|_{l_\infty} \\
&\le C\sum_{n=0}^{p-1}\left(k_{ij}^{\bar{q}+1-(p-1-n)}\|f\|_{\mathcal{T}}^{\bar{q}+2}C_{\Phi,p-1}^n\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + \|f\|_{\mathcal{T}}^{p-n}\|[\Phi^{(n)}]_t\|_{l_\infty}\right) \\
&\le Ck_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\sum_{n=0}^{p-1}k_{ij}^nC_{\Phi,p-1}^n\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum_{n=0}^{p-1}\|f\|_{\mathcal{T}}^{p-n}\|[\Phi^{(n)}]_t\|_{l_\infty} \\
&\le Ck_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum_{n=0}^{p-1}\|f\|_{\mathcal{T}}^{p-n}\|[\Phi^{(n)}]_t\|_{l_\infty},
\end{aligned}$$

for $0 \le p-1 \le \bar{q}-1$. □

Our next task is to estimate the jump in the discrete dual solution Φ itself, corresponding to Lemma 4.2.

Lemma 4.5. (Jump estimates for Φ) *Let Φ be the* mcG$(q)^*$ *or* mdG$(q)^*$ *solution of* (1.3) *with $g = 0$. If assumptions* (A1)–(A5) *and* (A5″) *hold, then there is a constant $C = C(\bar{q}, c_k, c_k'', \alpha) > 0$, such that*

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le Ck_{ij}^{r+1-p}C_{\Phi,r}^{r+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0,\dots,r+1, \tag{4.21}$$

with $r = 0,\dots,\bar{q}-1$ for the mcG(q) *method and $r = 0,\dots,\bar{q}$ for the* mdG(q) *method, for each local interval I_{ij}, where t_{ij} is an internal node of the time slab $\mathcal{T}$.*

Proof. The proof is by induction. We first note that at $t = t_{ij}$, we have

$$\begin{aligned}
[\Phi_i^{(p)}]_t &= \left(\Phi_i^{(p)}(t^+) - \tilde{\Phi}_i^{(p)}(t^+)\right) + \left(\tilde{\Phi}_i^{(p)}(t^+) - \tilde{\Phi}_i^{(p)}(t^-)\right) + \left(\tilde{\Phi}_i^{(p)}(t^-) - \Phi_i^{(p)}(t^-)\right) \\
&\equiv e_+ + e_0 + e_-.
\end{aligned}$$

By Theorem 2.1 (or Theorem 2.2), Φ is an interpolant of $\tilde{\Phi}$; if Φ is the mcG$(q)^*$ solution, then Φ_i is the $\pi_{\mathrm{cG}^*}^{[q_{ij}]}$-interpolant of $\tilde{\Phi}_i$ on I_{ij}, and if Φ is the mdG$(q)^*$ solution, then Φ_i is the $\pi_{\mathrm{dG}^*}^{[q_{ij}]}$-interpolant of $\tilde{\Phi}_i$. It follows that

$$|e_-| \le Ck_{ij}^{r+1-p}\|\tilde{\Phi}_i^{(r+1)}\|_{L_\infty(I_{ij})} + C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=1}^{r}k_{ij}^{m-p}|[\tilde{\Phi}_i^{(m)}]_x|, \quad p = 0,\dots,r+1,$$

where $r = 0,\dots,\bar{q}-1$ for the mcG$(q)^*$ solution and $r = 0,\dots,\bar{q}$ for the mdG$(q)^*$solution. Similarly, we have

$$|e_+| \le Ck_{i,j+1}^{r+1-p}\|\tilde{\Phi}_i^{(r+1)}\|_{L_\infty(I_{i,j+1})} + C\sum_{x\in\mathcal{N}_{i,j+1}}\sum_{m=1}^{r}k_{i,j+1}^{m-p}|[\tilde{\Phi}_i^{(m)}]_x|,$$

for $p = 0,\dots,r+1$. To estimate e_0, we note that $e_0 = 0$ for $p = 0$, since $\tilde{\Phi}$ is continuous. For $p = 1,\dots,\bar{q}$, Lemma 4.4 gives

(4.22)

$$|e_0| = |[\tilde{\Phi}_i^{(p)}]_t| \le Ck_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum_{n=0}^{p-1}\|f\|_{\mathcal{T}}^{p-n}\|\,|[\Phi^{(n)}]_t|\,\|_{l_\infty}.$$

Using assumption (A2), and the estimates for e_+, e_0, and e_-, we obtain for $r = 0$ and $p = 0$,

$$\begin{aligned}
|[\Phi_i]_t| &\le Ck_{i,j+1}\|\dot{\tilde{\Phi}}_i\|_{L_\infty(I_{i,j+1})} + 0 + Ck_{ij}\|\dot{\tilde{\Phi}}_i\|_{L_\infty(I_{ij})} \\
&\le C(\alpha^{-1}+1)k_{ij}C_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} = Ck_{ij}C_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.
\end{aligned}$$

For $r = 0$ and $p = 1$, it follows by (4.22), noting that $k_{ij}^{\bar{q}+2-1}\|f\|_{\mathcal{T}}^{\bar{q}+2} \le C\|f\|_{\mathcal{T}} = CC_{\Phi,0}$, and assumption (A2), that $|e_0| \le CC_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\|f\|_{\mathcal{T}}\|[\Phi]_t\|_{l_\infty} \le CC_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}$, and so,

$$\begin{aligned}
|[\dot{\Phi}_i]_t| &\le C\|\dot{\tilde{\Phi}}_i\|_{L_\infty(I_{i,j+1})} + CC_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\|\dot{\tilde{\Phi}}_i\|_{L_\infty(I_{ij})} \\
&\le CC_{\Phi,0}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.
\end{aligned}$$

Thus, (4.21) holds for $r = 0$. Assume now that (4.21) holds for $r = \bar{r}-1 \ge 0$. Then, by Lemma 4.4 and assumption (A5), it follows that

$$\begin{aligned}
|e_-| &\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\quad + C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=1}^{r}k_{ij}^{m-p}\left(k_{ij}^{\bar{q}+2-m}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + \sum_{n=0}^{m-1}\|f\|_{\mathcal{T}}^{m-n}\|[\Phi^{(n)}]_t\|_{l_\infty}\right) \\
&\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum\left(k_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\right. \\
&\quad \left. + \sum\|f\|_{\mathcal{T}}^{m-n}k_{ij}^{m-p+(\bar{r}-1)+1-n}C_{\Phi,\bar{r}-1}^{(\bar{r}-1)+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\right) \\
&\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum\left(k_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\right. \\
&\quad \left. + \sum k_{ij}^{\bar{r}+1-p+m-1-n}\|f\|_{\mathcal{T}}^{m-1-n}C_{\Phi,\bar{r}-1}^{\bar{r}+1}\right)\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.
\end{aligned}$$

Similarly, we obtain the estimate

$$|e_+| \le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.$$

Again using the assumption that (4.21) holds for $r = \bar{r} - 1$, we obtain

$$\begin{aligned}
|e_0| &\le Ck_{ij}^{\bar{q}+2-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\quad + C\sum_{n=0}^{p-1}\|f\|_{\mathcal{T}}^{p-n}k_{ij}^{(\bar{r}-1)+1-n}C_{\Phi,\bar{r}-1}^{(\bar{r}-1)+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}-1}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\left(1+\sum_{n=0}^{p-1}(k_{ij}\|f\|_{\mathcal{T}})^{p-1-n}\right) \\
&\le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}-1}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.
\end{aligned}$$

We thus have $|[\Phi_i^{(p)}]_t| \le |e_+| + |e_0| + |e_-| \le Ck_{ij}^{\bar{r}+1-p}C_{\Phi,\bar{r}}^{\bar{r}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}$, and so (4.21) follows by induction. □

Next, we prove an estimate for the derivatives of the discrete dual solution Φ, corresponding to Theorem 4.1.

Theorem 4.3. (Derivative estimates for Φ) *Let Φ be the* mcG$(q)^*$ *or* mdG$(q)^*$ *solution of* (1.3) *with $g = 0$. If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\Phi^{(p)}\|_{L_\infty(\mathcal{T},\infty)} \le C\|f\|_{\mathcal{T}}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0,\dots,\bar{q}. \tag{4.23}$$

Proof. By Theorem 2.1 (or Theorem 2.2), Φ is an interpolant of $\tilde{\Phi}$, and so, by Theorem 5.2 in [3], we have

$$\begin{aligned}
\|\Phi_i^{(p)}\|_{L_\infty(I_{ij})} &= \|(\pi\tilde{\Phi}_i)^{(p)}\|_{L_\infty(I_{ij})} \\
&\le C'\|\tilde{\Phi}_i^{(p)}\|_{L_\infty(I_{ij})} + C'\sum_{x\in\mathcal{N}_{ij}}\sum_{m=1}^{p-1}k_{ij}^{m-p}|[\tilde{\Phi}_i^{(m)}]_x|,
\end{aligned}$$

for some constant $C' = C'(\bar{q}) > 0$. For $p = 1$, we thus obtain the estimate

$$\begin{aligned}
\|\dot{\Phi}_i\|_{L_\infty(I_{ij})} &\le C'\|\dot{\tilde{\Phi}}_i\|_{L_\infty(I_{ij})} = C'\|f_i^*(\Phi)\|_{L_\infty(I_{ij})} \\
&= C'\|J^\top\Phi\|_{L_\infty(I_{ij})} \le C'\|f\|_{\mathcal{T}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)},
\end{aligned}$$

by assumption (A4), and so (4.23) holds for $p = 1$.

For $p = 2,\dots,\bar{q}$, assuming that (A5″) holds for $C_{\Phi,p-1}$, we use Lemma 4.4, Lemma 4.5 (with $r = p-1$) and assumption (A2), to obtain

$$\begin{aligned}
&\|\Phi_i^{(p)}\|_{L_\infty(I_{ij})} \le CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&+ C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=1}^{p-1}k_{ij}^{m-p}\left(k_{ij}^{\bar{q}+2-m}\|f\|_{\mathcal{T}}^{\bar{q}+2}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + \sum_{n=0}^{m-1}\|f\|_{\mathcal{T}}^{m-n}\|[\Phi^{(n)}]_x\|_{l_\infty}\right) \\
&\le CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum k_{ij}^{m-p}\|f\|_{\mathcal{T}}^{m-n}k_{ij}^{(p-1)+1-n}C_{\Phi,p-1}^{(p-1)+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\sum(k_{ij}\|f\|_{\mathcal{T}})^{m-n} \\
&\le CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)},
\end{aligned}$$

where we have used the fact that

$$k_{ij}^{m-p}k_{ij}^{\bar{q}+2-m}\|f\|_{\mathcal{T}}^{\bar{q}+2} = \|f\|_{\mathcal{T}}^p(k_{ij}\|f\|_{\mathcal{T}})^{\bar{q}+2-p} \le CC_{\Phi,p-1}^p,$$

and where $C = C(\bar{q}, c_k, c_k'', \alpha)$. Continuing now in the same way as in the proof of Theorem 4.1, we obtain

$$\|\Phi^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C\|f\|_{\mathcal{T}}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 1,\dots,\bar{q},$$

for $C = C(\bar{q}, c_k, \alpha)$, which (trivially) holds also when $p = 0$. □

Having now removed the additional assumption (A5″), we obtain the following version of Lemma 4.5.

Theorem 4.4. (Jump estimates for Φ) *Let Φ be the* mcG$(q)^*$ *or* mdG$(q)^*$ *solution of* (1.3) *with $g = 0$. If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le Ck_{ij}^{\bar{q}-p}\|f\|_{\mathcal{T}}^{\bar{q}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0,\dots,\bar{q}-1, \tag{4.24}$$

for the mcG$(q)^*$ *solution, and*

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le Ck_{ij}^{\bar{q}+1-p}\|f\|_{\mathcal{T}}^{\bar{q}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0,\dots,\bar{q}, \tag{4.25}$$

for the mdG$(q)^*$ *solution. This holds for each local interval I_{ij}, where t_{ij} is an internal node of the time slab $\mathcal{T}$.*

4.4. **A special interpolation estimate.** In the derivation of a priori error estimates, we face the problem of estimating the interpolation error $\pi\varphi_i - \varphi_i$ on a local interval I_{ij}, where φ_i is defined by

$$\varphi_i = (J^\top(\pi u, u)\Phi)_i = \sum_{l=1}^N J_{li}(\pi u, u)\Phi_l, \quad i = 1,\dots,N. \tag{4.26}$$

We note that φ_i may be discontinuous within I_{ij}, if other components have nodes within I_{ij}, see Figure 2, since then some Φ_l (or some J_{li}) may be discontinuous within I_{ij}. To estimate the interpolation error, we thus need to estimate derivatives and jumps of φ_i, which requires estimates for both J_{li} and Φ_l.

In Lemma 4.3 we have already proved an estimate for $J^\top$ when f is linearized around πu and U, rather than around πu and u as in (4.26). Replacing U by u, we obtain the following estimate for $J^\top$.

Lemma 4.6. *Let πu be an interpolant, of order $\bar{q}$, of the exact solution u of* (1.1). *If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\left\|\frac{d^pJ^\top(\pi u, u)}{dt^p}\right\|_{L_\infty(\mathcal{T},l_\infty)} \le C\|f\|_{\mathcal{T}}^{p+1}, \quad p = 0,\dots,\bar{q}, \tag{4.27}$$

and

$$\left\|\left[\frac{d^pJ^\top(\pi u, u)}{dt^p}\right]_{t_{i,j-1}}\right\|_{l_\infty} \le Ck_{ij}^{\bar{q}+1-p}\|f\|_{\mathcal{T}}^{\bar{q}+2}, \quad p = 0,\dots,\bar{q}-1, \tag{4.28}$$

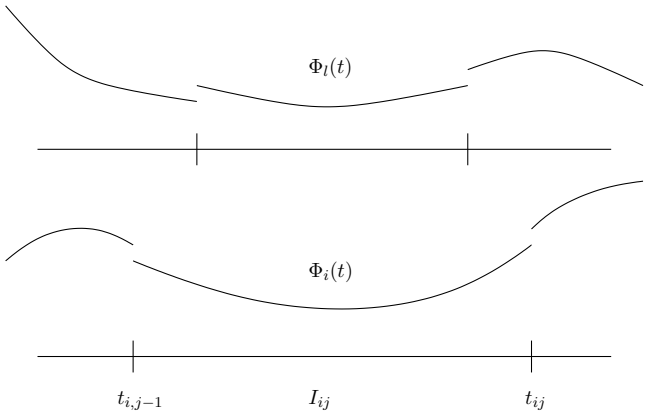

FIGURE 2. If some other component $l \neq i$ has a node within I_{ij}, then Φ_l may be discontinuous within I_{ij}, causing φ_i to be discontinuous within I_{ij}.

for each local interval I_{ij}, where $t_{i,j-1}$ is an internal node of the time slab $\mathcal{T}$.

Proof. See proof of Lemma 4.3. □

From Lemma 4.6 and the estimates for Φ derived in the previous section, we now obtain the following estimates for φ.

Lemma 4.7. (Estimates for φ) *Let φ be defined as in* (4.26)*. If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\varphi_i^{(p)}\|_{L_\infty(I_{ij})} \leq C\|f\|_{\mathcal{T}}^{p+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, q_{ij}, \tag{4.29}$$

and

$$|[\varphi_i^{(p)}]_x| \leq Ck_{ij}^{r_{ij}-p}\|f\|_{\mathcal{T}}^{r_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \quad \forall x \in \mathcal{N}_{ij}, \quad p = 0, \ldots, q_{ij} - 1, \tag{4.30}$$

with $r_{ij} = q_{ij}$ for the mcG(q) *method and $r_{ij} = q_{ij} + 1$ for the* mdG(q) *method. This holds for each local interval I_{ij} within the time slab $\mathcal{T}$.*

Proof. Differentiating, we have $\varphi_i^{(p)} = \sum_{n=0}^p \binom{p}{n} \frac{d^{p-n}J^\top(\pi u,u)}{dt^{p-n}}\Phi^{(n)}$, and so, by Theorem 4.3 and Lemma 4.6, we obtain

$$\begin{aligned}\|\varphi_i^{(p)}\|_{L_\infty(I_{ij})} &\leq C\sum_{n=0}^p \|f\|_{\mathcal{T}}^{(p-n)+1}\ \|f\|_{\mathcal{T}}^n\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\ &= C\sum_{n=0}^p \|f\|_{\mathcal{T}}^{p+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} = C\|f\|_{\mathcal{T}}^{p+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.\end{aligned}$$

To estimate the jump in $\varphi_i^{(p)}$, we use Lemma 3.2, Theorem 4.3, Theorem 4.4, and Lemma 4.6, to obtain

$$\begin{aligned}|[\varphi_i^{(p)}]_x| &= \left|\left[\sum_{n=0}^p \binom{p}{n}\frac{d^{p-n}J^\top}{dt^{p-n}}\Phi^{(n)}\right]_x\right| \leq C\sum_{n=0}^p \left|\left[\frac{d^{p-n}J^\top}{dt^{p-n}}\Phi^{(n)}\right]_x\right| \\ &\leq C\sum_{n=0}^p (k_{ij}^{q_{ij}+1-(p-n)}\|f\|_{\mathcal{T}}^{q_{ij}+2}\|f\|_{\mathcal{T}}^n \\ &\quad + \|f\|_{\mathcal{T}}^{(p-n)+1}k_{ij}^{q_{ij}-n}\|f\|_{\mathcal{T}}^{q_{ij}})\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\leq Ck_{ij}^{q_{ij}-p}\|f\|_{\mathcal{T}}^{q_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}\sum_{n=0}^p (k_{ij}\|f\|_{\mathcal{T}})^{n+1} + (k_{ij}\|f\|_{\mathcal{T}})^{p-n} \\ &\leq Ck_{ij}^{q_{ij}-p}\|f\|_{\mathcal{T}}^{q_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)},\end{aligned}$$

for the mcG(q) method. For the mdG(q) method, we obtain one extra power of $k_{ij}\|f\|_{\mathcal{T}}$. □

Using the interpolation estimates of [3], together with Lemma 4.7, we now obtain the following important interpolation estimates for φ.

Lemma 4.8. (Interpolation estimates for φ) *Let φ be defined as in* (4.26)*. If assumptions* (A1)–(A5) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\pi_{\text{cG}}^{[q_{ij}-2]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq Ck_{ij}^{q_{ij}-1}\|f\|_{\mathcal{T}}^{q_{ij}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 2, \tag{4.31}$$

and

$$\|\pi_{\text{dG}}^{[q_{ij}-1]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \leq Ck_{ij}^{q_{ij}}\|f\|_{\mathcal{T}}^{q_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \geq 1, \tag{4.32}$$

for each local interval I_{ij} within the time slab $\mathcal{T}$.

Proof. To prove (4.31), we use Theorem 5.2 in [3], with $r = q_{ij}-2$ and $p = 0$, together with Lemma 4.7, to obtain the following bound for $\|\pi_{\text{cG}}^{[q_{ij}-2]}\varphi_i -$

$\varphi_i\|_{L_\infty(I_{ij})}$:

$$
\begin{aligned}
&Ck_{ij}^{(q_{ij}-2)+1}\|\varphi_i^{((q_{ij}-2)+1)}\|_{L_\infty(I_{ij})} + C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=0}^{q_{ij}-2} k_{ij}^m\,|[\varphi_i^{(m)}]_x\,| \\
&\le Ck_{ij}^{q_{ij}-1}\|f\|_{\mathcal{T}}^{q_{ij}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=0}^{q_{ij}-2} k_{ij}^m k_{ij}^{q_{ij}-m}\|f\|_{\mathcal{T}}^{q_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&= Ck_{ij}^{q_{ij}-1}\|f\|_{\mathcal{T}}^{q_{ij}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} + Ck_{ij}^{q_{ij}}\|f\|_{\mathcal{T}}^{q_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le Ck_{ij}^{q_{ij}-1}\|f\|_{\mathcal{T}}^{q_{ij}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.
\end{aligned}
$$

The estimate for $\pi_{\mathrm{dG}}^{[q_{ij}-1]}\varphi_i - \varphi_i$ is obtained similarly. □

5. Estimates of derivatives and jumps for linear problems

We now derive estimates for derivatives and jumps for the multi-adaptive solutions of the linear problem (1.2). Assuming that the problem is linear, but non-autonomous, the estimates are obtained in a slightly different way compared to the estimates of the previous section.

5.1. **Assumptions.** We make the following basic assumptions: Given a time slab $\mathcal{T}$, assume that for each pair of local intervals I_{ij} and I_{mn} within the time slab, we have

$$q_{ij} = q_{mn} = \bar{q}, \tag{B1}$$

and

$$k_{ij} > \alpha\; k_{mn}, \tag{B2}$$

for some $\bar{q} \ge 0$ and some $\alpha \in (0,1)$. Furthermore, assume that A has $\bar{q}-1$ continuous derivatives and let $C_A > 0$ be constant, such that

$$\max\left(\|A^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)}, \|A^{\top(p)}\|_{L_\infty(\mathcal{T},l_\infty)}\right) \le C_A^{p+1}, \quad p = 0,\ldots,\bar{q}, \tag{B3}$$

for all time slabs $\mathcal{T}$. We further assume that there is a constant $c_k > 0$, such that

$$k_{ij}C_A \le c_k. \tag{B4}$$

We summarize the list of assumptions as follows:

- (B1) the local orders q_{ij} are equal within each time slab;
- (B2) the local time steps k_{ij} are semi-uniform within each time slab;
- (B3) A and its derivatives are bounded;
- (B4) the local time steps k_{ij} are small.

5.2. **Estimates for U and Φ.** To simplify the estimates, we introduce the following notation: For given $p > 0$, let $C_{U,p} \ge C_A$ be a constant, such that

$$\|U^{(n)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C_{U,p}^n\|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad n = 0,\ldots,p, \tag{5.5}$$

For $p = 0$, we define $C_{U,0} = C_A$. Temporarily, we will assume that there is a constant $c_k' > 0$, such that for each p,

$$k_{ij}C_{U,p} \le c_k'. \tag{B4$'$}$$

This assumption will be removed below in Theorem 5.1. We similarly define the constant $C_{\Phi,p}$, with $k_{ij}C_{\Phi,p} \le c_k'$. In the following lemma, we use assumptions (B1) and (B3) to derive estimates for $\tilde{U}$ and $\tilde{\Phi}$.

Lemma 5.1. (Estimates for $\tilde{U}$ and $\tilde{\Phi}$) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.2) *and define $\tilde{U}$ as in* (2.1)*. If assumptions* (B1) *and* (B3) *hold, then there is a constant $C = C(\bar{q}) > 0$, such that*

$$\|\tilde{U}^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 1,\ldots,\bar{q}+1, \tag{5.7}$$

and

$$\|[\tilde{U}^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \le C\sum_{n=0}^{p-1} C_A^{p-n}\|[U^{(n)}]_{t_{i,j-1}}\|_{l_\infty}, \quad p = 1,\ldots,\bar{q}. \tag{5.8}$$

Similarly, for Φ the mcG(q)* *or* mdG(q)* *solution of* (1.7) *with $g = 0$, and with $\tilde{\Phi}$ defined as in* (2.2)*, we obtain*

$$\|\tilde{\Phi}^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le CC_{\Phi,p-1}^p\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 1,\ldots,\bar{q}+1, \tag{5.9}$$

and

$$\|[\tilde{\Phi}^{(p)}]_{t_{ij}}\|_{l_\infty} \le C\sum_{n=0}^{p-1} C_A^{p-n}\|[\Phi^{(n)}]_{t_{ij}}\|_{l_\infty}, \quad p = 1,\ldots,\bar{q}. \tag{5.10}$$

Proof. By (2.1), it follows that $\dot{\tilde{U}} = -AU$, and so

$$\tilde{U}^{(p)} = \sum_{n=0}^{p-1}\binom{p-1}{n}A^{(p-1-n)}U^{(n)}.$$

It now follows by assumptions (B1) and (B3), that

$$\|\tilde{U}^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C\sum_{n=0}^{p-1} C_A^{p-n}\,C_{U,p-1}^n\|U\|_{L_\infty(\mathcal{T},l_\infty)} \le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)}.$$

Similarly, we obtain $\|[\tilde{U}^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \le C\sum_{n=0}^{p-1} C_A^{p-n}\|[U^{(n)}]_{t_{i,j-1}}\|_{l_\infty}$. The corresponding estimates for $\tilde{\Phi}$ follow similarly. □

By Lemma 5.1, we now obtain the following estimate for the size of the jump in function value and derivatives for U and Φ.

Lemma 5.2. (Jump estimates for U and Φ) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.2)*, and let Φ be the corresponding* mcG(q)* *or* mdG(q)* *solution of* (1.7) *with $g = 0$. If assumptions* (B1)–(B4) *and* (B4′) *hold, then there is a constant $C = C(\bar{q}, c_k, c_k', \alpha) > 0$, such that*

$$\|[U^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \le C k_{ij}^{r+1-p} C_{U,r}^{r+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, r+1, \tag{5.11}$$

$r = 0, \ldots, \bar{q}$, *and*

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le C k_{ij}^{r+1-p} C_{\Phi,r}^{r+1} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, r+1, \tag{5.12}$$

with $r = 0, \ldots, \bar{q}-1$ for the mcG(q)* *solution and $r = 0, \ldots, \bar{q}$ for the* mdG(q)* *solution, for each local interval I_{ij}, where $t_{i,j-1}$ and t_{ij}, respectively, are internal nodes of the time slab $\mathcal{T}$.*

Proof. The proof is by induction and follows those of Lemma 4.2 and Lemma 4.5. We first note that at $t = t_{i,j-1}$, we have

$$\begin{aligned}[U_i^{(p)}]_t &= \left(U_i^{(p)}(t^+) - \tilde{U}_i^{(p)}(t^+)\right) + \left(\tilde{U}_i^{(p)}(t^+) - \tilde{U}_i^{(p)}(t^-)\right) + \left(\tilde{U}_i^{(p)}(t^-) - U_i^{(p)}(t^-)\right) \\ &= e_+ + e_0 + e_-.\end{aligned}$$

Now, U is an interpolant of $\tilde{U}$ and so, by Theorem 5.2 in [3], it follows that

$$|e_+| \le C k_{ij}^{r+1-p} \|\tilde{U}_i^{(r+1)}\|_{L_\infty(I_{ij})} + C \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{r} k_{ij}^{m-p} |[\tilde{U}_i^{(m)}]_x|,$$

for $p = 0, \ldots, r+1$ and $r = 0, \ldots, \bar{q}$. Note that the second sum starts at $m = 1$ rather than at $m = 0$, since $\tilde{U}$ is continuous. Similarly, we have

$$|e_-| \le C k_{i,j-1}^{r+1-p} \|\tilde{U}_i^{(r+1)}\|_{L_\infty(I_{i,j-1})} + C \sum_{x \in \mathcal{N}_{i,j-1}} \sum_{m=1}^{r} k_{i,j-1}^{m-p} |[\tilde{U}_i^{(m)}]_x|.$$

To estimate e_0, we note that $e_0 = 0$ for $p = 0$, since $\tilde{U}$ is continuous. For $p = 1, \ldots, \bar{q}$, Lemma 5.1 gives

$$|e_0| = |[\tilde{U}_i^{(p)}]_t| \le C \sum_{n=0}^{p-1} C_A^{p-n} \|[U^{(n)}]_t\|_{l_\infty}.$$

Using assumption (B2), and the estimates for e_+, e_0, and e_-, we obtain for $r = 0$ and $p = 0$,

$$\begin{aligned}|[U_i]| &\le C k_{ij} \|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} + 0 + C k_{i,j-1} \|\dot{\tilde{U}}_i\|_{L_\infty(I_{i,j-1})} \\ &\le C(1 + \alpha^{-1}) k_{ij} C_{U,0} \|U\|_{L_\infty(\mathcal{T},l_\infty)} = C k_{ij} C_{U,0} \|U\|_{L_\infty(\mathcal{T},l_\infty)}.\end{aligned}$$

It now follows by assumption (B4), that for $r = 0$ and $p = 1$,

$$\begin{aligned}|[\dot{U}_i]_t| &\le C \|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} + C C_A \|[U]_t\|_{l_\infty} + C \|\dot{\tilde{U}}_i\|_{L_\infty(I_{i,j-1})} \\ &\le C(1 + k_{ij} C_{U,0}) C_{U,0} \|U\|_{L_\infty(\mathcal{T},l_\infty)} \le C C_{U,0} \|U\|_{L_\infty(\mathcal{T},l_\infty)}.\end{aligned}$$

Thus, (5.11) holds for $r = 0$. Assume now that (5.11) holds for $r = \bar{r} - 1 \ge 0$. Then, by Lemma 5.1 and assumption (B4′), it follows that

$$\begin{aligned}|e_+| &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)} + C \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{\bar{r}} k_{ij}^{m-p} \sum_{n=0}^{m-1} C_A^{m-n} \|[U^{(n)}]_t\|_{l_\infty} \\ &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\quad + C \sum k_{ij}^{m-p} C_A^{m-n} k_{ij}^{(\bar{r}-1)+1-n} C_{U,\bar{r}-1}^{(\bar{r}-1)+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \left(1 + \sum (k_{ij} C_{U,\bar{r}})^{m-1-n}\right) \|U\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)}.\end{aligned}$$

Similarly, we obtain the estimate $|e_-| \le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)}$. Finally, we use Lemma 5.1 and (B4′), to obtain the estimate

$$\begin{aligned}|e_0| = |[\tilde{U}_i^{(p)}]_t| &\le C \sum_{n=0}^{p-1} C_A^{p-n} \|[U^{(n)}]_t\|_{l_\infty} \\ &\le C \sum_{n=0}^{p-1} C_A^{p-n} k_{ij}^{(\bar{r}-1)+1-n} C_{U,\bar{r}-1}^{(\bar{r}-1)+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \sum_{n=0}^{p-1} (k_{ij} C_{U,\bar{r}-1})^{p-1-n} \|U\|_{L_\infty(\mathcal{T},l_\infty)} \\ &\le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)}.\end{aligned}$$

Summing up, we thus obtain

$$|[U_i^{(p)}]_t| \le |e_+| + |e_0| + |e_-| \le C k_{ij}^{\bar{r}+1-p} C_{U,\bar{r}}^{\bar{r}+1} \|U\|_{L_\infty(\mathcal{T},l_\infty)},$$

and so (5.11) follows by induction. The estimates for Φ follow similarly. □

Theorem 5.1. (Derivative estimates for U and Φ) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.2)*, and let Φ be the corresponding* mcG(q)* *or* mdG(q)* *solution of* (1.7) *with $g = 0$. If assumptions* (B1)–(B4) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|U^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C C_A^p \|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, \bar{q}, \tag{5.13}$$

and

$$\|\Phi^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le C C_A^p \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, \bar{q}. \tag{5.14}$$

Proof. Since U is an interpolant of $\tilde{U}$, it follows by Theorem 5.2 in [3], that $\|U_i^{(p)}\|_{L_\infty(I_{ij})} = \|(\pi\tilde{U}_i)^{(p)}\|_{L_\infty(I_{ij})}$ is bounded by

$$C' \|\tilde{U}_i^{(p)}\|_{L_\infty(I_{ij})} + C' \sum_{x \in \mathcal{N}_{ij}} \sum_{m=1}^{p-1} k_{ij}^{m-p} |[\tilde{U}_i^{(m)}]_x|,$$

for some constant $C' = C'(\bar{q})$. For $p = 1$, we thus obtain the estimate

$$\|\dot{U}_i\|_{L_\infty(I_{ij})} \le C'\|\dot{\tilde{U}}_i\|_{L_\infty(I_{ij})} = C'\|(AU)_i\|_{L_\infty(I_{ij})} \le C'C_A\|U\|_{L_\infty(\mathcal{T},l_\infty)},$$

and so (5.13) holds for $p = 1$.

For $p = 2, \ldots, \bar{q}$, assuming that (B4′) holds for $C_{U,p-1}$, we use Lemma 5.1, Lemma 5.2 (with $r = p-1$) and assumption (B2), to obtain

$$\begin{aligned}
\|U_i^{(p)}\|_{L_\infty(I_{ij})} &\le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\quad + C\sum_{x\in\mathcal{N}_{ij}}\sum_{m=1}^{p-1} k_{ij}^{m-p}\sum_{n=0}^{m-1} C_A^{m-n}\|[U^{(n)}]_x\|_{l_\infty} \\
&\le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\quad + C\sum k_{ij}^{m-p}C_A^{m-n}k_{ij}^{(p-1)+1-n}C_{U,p-1}^{(p-1)+1}\|U\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)}\left(1+\sum(k_{ij}C_A)^{m-n}\right) \\
&\le CC_{U,p-1}^p\|U\|_{L_\infty(\mathcal{T},l_\infty)},
\end{aligned}$$

where $C = C(\bar{q}, c_k, c_k', \alpha)$. It now follows in the same way as in the proof of Theorem 4.1, that

$$\|U^{(p)}\|_{L_\infty(\mathcal{T},l_\infty)} \le CC_A^p\|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 1, \ldots, \bar{q},$$

for $C = C(\bar{q}, c_k, \alpha)$, which (trivially) holds also when $p = 0$. The estimate for Φ follows similarly. □

Having now removed the additional assumption (B4′), we obtain the following version of Lemma 5.2.

Theorem 5.2. (Jump estimates for U and Φ) *Let U be the* mcG(q) *or* mdG(q) *solution of* (1.2)*, and let Φ be the corresponding* mcG(q)* *or* mdG(q)* *solution of* (1.7) *with $g = 0$. If assumptions* (B1)–(B4) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|[U^{(p)}]_{t_{i,j-1}}\|_{l_\infty} \le Ck_{ij}^{\bar{q}+1-p}C_A^{\bar{q}+1}\|U\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, \bar{q}. \tag{5.15}$$

Furthermore, we have

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le Ck_{ij}^{\bar{q}-p}C_A^{\bar{q}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, \bar{q}, \tag{5.16}$$

for the mcG(q)* *solution and*

$$\|[\Phi^{(p)}]_{t_{ij}}\|_{l_\infty} \le Ck_{ij}^{\bar{q}+1-p}C_A^{\bar{q}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, \bar{q}, \tag{5.17}$$

for the mdG(q)* *solution. This holds for each local interval I_{ij}, where $t_{i,j-1}$ and t_{ij}, respectively, are internal nodes of the time slab $\mathcal{T}$.*

5.3. **A special interpolation estimate.** As for the general non-linear problem, we need to estimate the interpolation error $\pi\varphi_i - \varphi_i$ on a local interval I_{ij}, where φ_i is now defined by

$$\varphi_i = (A^\top\Phi)_i = \sum_{l=1}^N A_{li}\Phi_l, \quad i = 1, \ldots, N. \tag{5.18}$$

As noted above, φ_i may be discontinuous within I_{ij}, if I_{ij} contains nodes for other components. We first prove the following estimates for φ.

Lemma 5.3. (Estimates for φ) *Let φ be defined as in* (5.18)*. If assumptions* (B1)–(B4) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\varphi_i^{(p)}\|_{L_\infty(I_{ij})} \le CC_A^{p+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad p = 0, \ldots, q_{ij}, \tag{5.19}$$

and

$$|[\varphi_i^{(p)}]_x| \le Ck_{ij}^{r_{ij}-p}C_A^{r_{ij}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \quad \forall x \in \mathcal{N}_{ij}, \quad p = 0, \ldots, q_{ij}-1, \tag{5.20}$$

with $r_{ij} = q_{ij}$ for the mcG(q) *method and $r_{ij} = q_{ij} + 1$ for the* mdG(q) *method. This holds for each local interval I_{ij} within the time slab $\mathcal{T}$.*

Proof. Differentiating φ_i, we have

$$\varphi_i^{(p)} = \frac{d^p}{dt^p}(A^\top\Phi)_i = \sum_{n=0}^p \binom{p}{n}(A^{\top(p-n)}\Phi^{(n)})_i$$

and so, by Theorem 5.1, we obtain

$$\|\varphi_i^{(p)}\|_{L_\infty(I_{ij})} \le C\sum_{n=0}^p C_A^{(p-n)+1}C_A^n\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} = CC_A^{p+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}.$$

To estimate the jump in $\varphi_i^{(p)}$, we use Theorem 5.2, to obtain

$$\begin{aligned}
|[\varphi_i^{(p)}]_x| &\le C\sum_{n=0}^p |(A^{\top(p-n)}[\Phi^{(n)}]_x)_i| \le C\sum_{n=0}^p C_A^{(p-n)+1}\|[\Phi^{(n)}]_x\|_{l_\infty} \\
&\le C\sum_{n=0}^p C_A^{(p-n)+1}k_{ij}^{\bar{q}-n}C_A^{\bar{q}}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le Ck_{ij}^{\bar{q}-p}C_A^{\bar{q}+1}\sum_{n=0}^p(k_{ij}C_A)^{p-n}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)} \\
&\le Ck_{ij}^{\bar{q}-p}C_A^{\bar{q}+1}\|\Phi\|_{L_\infty(\mathcal{T},l_\infty)},
\end{aligned}$$

for the mcG(q) method. For the mdG(q) method, we obtain one extra power of $k_{ij}C_A$. □

Using Lemma 5.3 and the interpolation estimates from [3], we now obtain the following interpolation estimates for φ.

Lemma 5.4. (Interpolation estimates for φ) *Let φ be defined as in* (5.18). *If assumptions* (B1)–(B4) *hold, then there is a constant $C = C(\bar{q}, c_k, \alpha) > 0$, such that*

$$\|\pi_{\mathrm{cG}}^{[q_{ij}-2]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \le C k_{ij}^{q_{ij}-1} C_A^{q_{ij}} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \ge 2, \tag{5.21}$$

and

$$\|\pi_{\mathrm{dG}}^{[q_{ij}-1]}\varphi_i - \varphi_i\|_{L_\infty(I_{ij})} \le C k_{ij}^{q_{ij}} C_A^{q_{ij}+1} \|\Phi\|_{L_\infty(\mathcal{T},l_\infty)}, \quad q_{ij} = \bar{q} \ge 1, \tag{5.22}$$

for each local interval I_{ij} within the time slab $\mathcal{T}$.

Proof. See proof of Lemma 4.8. □

References

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

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

[3] ———, *Interpolation estimates for piecewise smooth functions in one dimension*, Tech. Rep. 2004–02, Chalmers Finite Element Center Preprint Series, 2004.

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

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

[6] ———, *Multi-adaptive time-integration*, Applied Numerical Mathematics, 48 (2004), pp. 339–354.

ALGORITHMS FOR MULTI-ADAPTIVE TIME-STEPPING

JOHAN JANSSON AND ANDERS LOGG

Abstract. Multi-adaptive Galerkin methods are extensions of the standard continuous and discontinuous Galerkin methods for the numerical solution of initial value problems for ordinary or partial differential equations. In particular, the multi-adaptive methods allow individual time steps to be used for different components or in different regions of space. We present algorithms for multi-adaptive time-stepping, including the recursive construction of time slabs, regulation of the individual time steps, adaptive fixed point iteration on time slabs, and the automatic generation of dual problems. An example is given for the solution of a nonlinear partial differential equation in three dimensions.

1. Introduction

We have earlier in a sequence of papers [14, 15, 16, 17, 13] introduced the multi-adaptive Galerkin methods mcG(q) and mdG(q) for the 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.

In the current paper, we discuss important aspects of the implementation of multi-adaptive Galerkin methods. Some of these aspects are discussed in [15] and [13], but with technical details left out. We now provide these details.

1.1. **Implementation.** The algorithms presented in this paper are implemented in the multi-adaptive ODE-solver of **DOLFIN** [11, 12], which is the C++ implementation of the new open-source software project **FEniCS** [3] for the automation of Computational Mathematical Modeling (CMM).

Date: April 15, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, C++, implementation, algorithms.

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.

The multi-adaptive solver in **DOLFIN** is based on the original implementation *Tanganyika*, presented in [15], but has been completely rewritten for **DOLFIN**.

The multi-adaptive solver is actively developed by the authors, with the intention of providing the next standard for the solution of initial value problems. This will be made possible through the combination of an efficient forward integrator, automatic and reliable error control, and a simple and intuitive user interface.

1.2. **Obtaining the software. DOLFIN** is licensed under the GNU General Public License [8], which means that anyone is free to use or modify the software, provided these rights are preserved.

The source code of **DOLFIN**, including numerous example programs, is available at the web page `http://www.phi.chalmers.se/dolfin/`, and each new release is announced on freshmeat.net. Alternatively, the source code can be obtained through anonymous CVS as explained on the web page. Comments and contributions are welcome.

1.3. **Notation.** For a detailed description of the multi-adaptive Galerkin methods, we refer the reader to [14, 15, 16, 17, 13]. In particular, we refer to [14] or [16] 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*:

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

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

1.4. **Outline of the paper.** We first present the user interface of the multi-adaptive solver in Section 2, before we discuss the algorithms of multi-adaptive time-stepping in Section 3. In Section 4, we present numerical results for the bistable equation as an example of multi-adaptive time-stepping for a partial differential equation.

2. User interface

Potential usage of the multi-adaptive solver ranges from a student or teacher wanting to solve a fixed ODE in an educational setting, to a PDE package using it as an internal solver module. The user interface of the

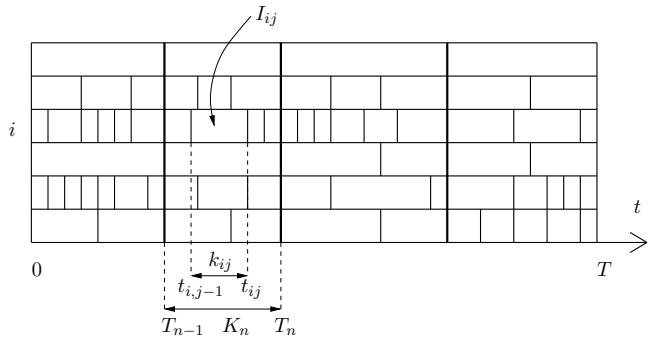

FIGURE 1. Individual partitions of the interval $(0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

multi-adaptive solver is specified in terms of an ODE base class consisting of a right hand side f, a time interval $[0, T]$, and initial value u_0, as shown in Figure 2.

To solve an ODE, the user implements a subclass which inherits from the ODE base class. As an example, we present in Figure 2 and Table 1 the implementation of the harmonic oscillator

$$\begin{aligned} \dot{u}_1 &= u_2, \\ \dot{u}_2 &= -u_1, \end{aligned} \tag{2.1}$$

on $[0, 10]$ with initial value $u(0) = (0, 1)$.

3. Multi-adaptive time-stepping

We present below a collection of the key algorithms for multi-adaptive time-stepping. The algorithms are given in pseudo-code and where appropriate we give remarks on how the algorithms have been implemented using C++ in **DOLFIN**. In most cases, we present simplified versions of the algorithms with focus on the most essential steps.

3.1. **General algorithm.** The general multi-adaptive time-stepping algorithm (Algorithm 1) is based on the algorithm CreateTimeSlab (Algorithm 3) and the algorithms for adaptive fixed point iteration on time slabs discussed below in Section 3.3. Starting at $t = 0$, the algorithm creates a sequence of time slabs until the given end time T is reached. The end time T is given as an argument to CreateTimeSlab, which creates a time slab covering an interval $[T_{n-1}, T_n]$ such that $T_n \leq T$. CreateTimeSlab returns the end time T_n of the created time slab and the integration continues until

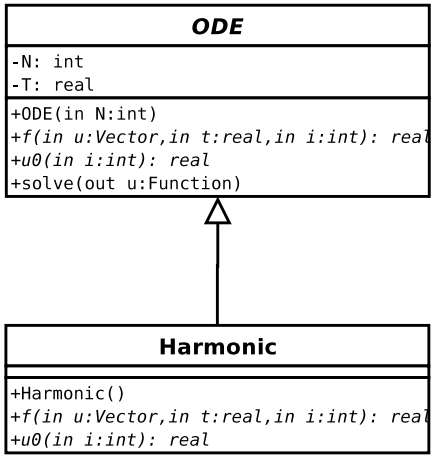

FIGURE 2. UML class diagram showing the base class interface of the multi-adaptive ODE-solver together with a subclass representing the ODE (2.1).

$T_n = T$. For each time slab, adaptive fixed point iteration is performed on the time slab until the discrete equations given by the mcG(q) or mdG(q) method have converged.

Algorithm 1 U = Integrate(ODE)

```
t ← 0
while t < T do
  {time slab, t} ← CreateTimeSlab({1, ..., N}, t, T)
  Iterate(time slab)
end while
```

In **DOLFIN**, Algorithm 1 is implemented by the class `TimeStepper`, which can be used in two different ways (see Table 2). In the standard case, the function `TimeStepper::solve()` is called to integrate the given ODE on the interval $[0, T]$. The class `TimeStepper` also provides the alternative interface `TimeStepper::step()`, that can be used to integrate the solution one time slab at a time. This is useful in interactive applications, where the right-hand side f of (1.1) needs to be modified in each step of the integration.

The basic forward integrator, Algorithm 1, can be used as the main component of an adaptive algorithm with automated error control of the computed solution (Algorithm 2). This algorithm first estimates the individual

```
Harmonic::Harmonic : ODE(2)
{
  T = 10.0;
}

real Harmonic::u0(int i)
{
  if (i == 0)
    return 0;
  if (i == 1)
    return 1;
}

real Harmonic::f(Vector u, real t, int i)
{
  if (i == 0)
    return u(1);
  if (i == 1)
    return -u(0);
}
```

TABLE 1. Sketch of the C++ implementation of the harmonic oscillator (2.1). Note that C++ indexing starts at 0.

```
class TimeStepper
{
  TimeStepper(ODE ode, Function u);

  static void solve(ODE ode, Function u);
  real step();
}
```

TABLE 2. Sketch of the C++ interface of the multi-adaptive time-stepper, Algorithm 1.

stability factors $\{S_i(T)\}_{i=1}^N$, which together with the local residuals determine the multi-adaptive time steps. (See [4, 5, 2] or [14] for a discussion on duality-based a posteriori error estimation.) The preliminary estimates for the stability factors can be based on previous experience, i.e., if we have solved a similar problem before, but usually we take $S_i(T) = 1$ for $i = 1, \ldots, N$.

In each iteration, the *primal problem* (1.1) is solved using Algorithm 1. An ODE of the form (1.1) representing the *dual problem* is then created, as discussed below in Section 3.7, and solved using Algorithm 1. It is important to note that both the primal and the dual problems are solved using the same

algorithm, but with different time steps and, possibly, different tolerances, methods, and orders. When the solution of the dual problem has been computed, the stability factors $\{S_i(T)\}_{i=1}^N$ and the error estimate can be computed.

Algorithm 2 U = Solve(ODE, TOL)

estimate stability factors
repeat
  U = Integrate(ODE)
  create dual problem ODE*
  Φ = Integrate(ODE*)
  compute stability factors
  compute error estimate E
until $E \leq$ TOL

3.2. **Recursive construction of time slabs.** In each step of Algorithm 1, a new time slab is created between two synchronized time levels T_{n-1} and 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 3.

To create a time slab, we first compute the desired time steps for all components as discussed below in Section 3.5. A threshold θK is then computed based on the maximum time step K and a fixed parameter θ controlling the density of the time slab. The components are then partitioned into two sets based on the threshold, see Figure 4. For each component in the group with large time steps, an element is created and added to the element group of the time slab. The remaining components with small time steps are processed by a recursive application of this algorithm for the construction of time slabs. For a more detailed discussion of the construction, see [13].

We organize the recursive construction of time slabs as described by Algorithms 3, 4, 5, and 6. The recursive construction simplifies the implementation; each recursively nested time slab can be considered as a sub system of the ODE. Note that the group of recursively nested time slabs for components in group I_0 is created before the element group containing elements for components in group I_1. The tree of time slabs is thus created recursively *depth-first*, which means in particular that the element for the component with the smallest time step is created first.

Algorithm 4 for the partition of components can be implemented efficiently using the function `std::partition()`, which is part of the Standard C++ Library.

Algorithm 3 {time slab, T_n} = CreateTimeSlab(components, T_{n-1}, T)

$\{I_0, I_1, K\} \leftarrow$ Partition(components)
if $T_{n-1} + K < T$ **then**
  $T_n \leftarrow T_{n-1} + K$
else
  $T_n \leftarrow T$
end if
time slabs $\leftarrow$ CreateTimeSlabs(I_0, T_{n-1}, T_n)
element group $\leftarrow$ CreateElements(I_1, T_{n-1}, T_n)
time slab $\leftarrow$ {time slabs, element group}

Algorithm 4 $\{I_0, I_1, K\}$ = Partition(components)

$I_0 \leftarrow \emptyset$
$I_1 \leftarrow \emptyset$
$K \leftarrow$ *maximum time step within* components
for each component **do**
  $k \leftarrow$ *time step of* component
  if $k < \theta K$ **then**
    $I_0 \leftarrow I_0 \cup$ {component}
  else
    $I_1 \leftarrow I_1 \cup$ {component}
  end if
end for
$\bar{K} \leftarrow$ *minimum time step within* I_1
$K \leftarrow \bar{K}$

Algorithm 5 time slabs = CreateTimeSlabs(components, T_{n-1}, T_n)

time slabs $\leftarrow \emptyset$
$t \leftarrow T_{n-1}$
while $t < T$ **do**
  {time slab, t} $\leftarrow$ CreateTimeSlab(components, t, T_n)
  time slabs $\leftarrow$ time slabs $\cup$ time slab
end while

Algorithm 6 elements = CreateElements(components, T_{n-1}, T_n)

elements $\leftarrow \emptyset$
for each component **do**
  create element *for* component *on* $[T_{n-1}, T_n]$
  elements $\leftarrow$ elements $\cup$ element
end for

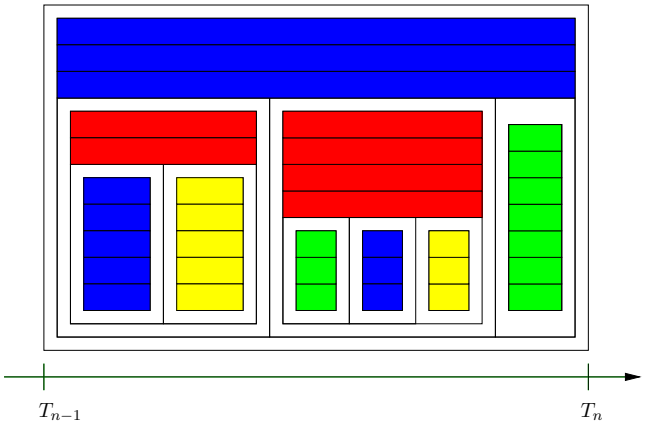

FIGURE 3. The recursive organization of the time slab. Each time slab contains an element group and a list of recursively nested time slabs. The root time slab in the figure contains one element group of three elements and three time slabs. The first of these sub slabs contains an element group of two elements and two nested time slabs, and so on. The root time slab recursively contains a total of nine element groups and 35 elements.

3.3. **Adaptive fixed point iteration on time slabs.** As discussed in [13], the discrete equations given by the mcG(q) or mdG(q) method on each time slab are solved using adaptive fixed point iteration. For the fixed point iteration, each time slab is viewed as a discrete system of equations the form

$$F(x) = 0 \tag{3.1}$$

for the degrees of freedom x of the solution U on the time slab. This system is partitioned into coupled sub systems which each take the form (3.1), one for each element group. Similarly, each element group is naturally partitioned into sub systems, one for each element within the element group.

The general algorithm for nested fixed point iteration on the sub systems of a time slab is based on the principle that each iteration on a given system consists of fixed point iteration on each sub system, as outlined in Table 3. By modifying the condition (*1*, *2*, or *3*) for fixed point iteration on each level of iteration, different versions of the overall iterative algorithm are obtained. In [13], four different strategies for the adaptive fixed point iteration are discussed: *non-stiff iteration*, *adaptive level 1 iteration*, *adaptive level 2 iteration*, and *adaptive level 3 iteration*. By monitoring the convergence at

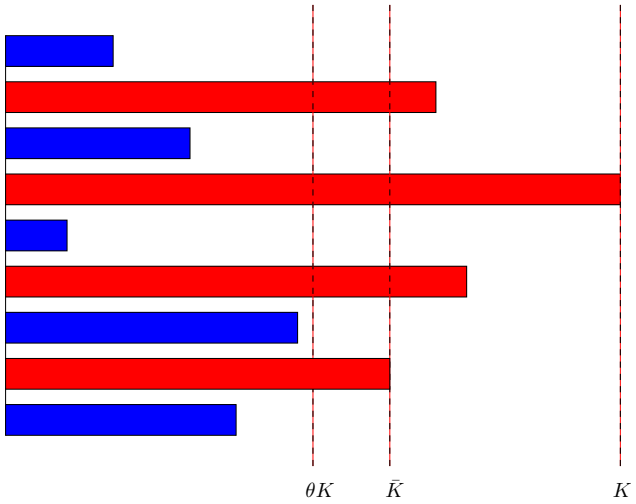

FIGURE 4. The partition of components into groups of small and large time steps for $\theta = 1/2$.

the different levels of iteration, the appropriate version of the adaptive fixed point iteration is chosen, depending on the stiffness of the problem.

Table 3 shows a simplified version of the fixed point iteration. The full algorithm also needs to monitor the convergence rate, stabilize the iterations, and (possibly) change strategy, as shown in Algorithm 7 (Iterate) and Algorithm 8 (Update). Both algorithms return the increment d for the degrees of freedom of the current (sub) system. In the case of Algorithm 7, the increment is computed as the maximum increment over the iterations for the system, and in Algorithm 8, the increment is computed as the l_2-norm of the increments for the set of sub systems.

Since we sometimes need to change the strategy for the fixed point iteration depending on the stiffness of the problem, the fixed point iteration is naturally implemented as a *state machine*, where each state has different versions of the algorithms Converged, Diverged, Stabilize, and Update.

In **DOLFIN**, the state machine is implemented as shown in Figure 5, following the design pattern for a state machine suggested in [9]. The class `FixedPointIteration` implements Algorithm 7 and the class `Iteration` serves as a base class (interface) for the subclasses `NonStiffIteration`, `AdaptiveIterationLevel1`, `AdaptiveIterationLevel2`, and `AdaptiveIterationLevel3`. Each of these subclasses implement the interface specified

Iterate(time slab)
while *condition 3* **do**
  Update(time slab)
end while
	Update(time slab)
	for each element group **do**
	  Iterate(element group)
	end for

Iterate(element group)
while *condition 2* **do**
  Update(element group)
end while
	Update(element group)
	for each element **do**
	  Iterate(element)
	end for

Iterate(element)
while *condition 1* **do**
  Update(element)
end while
	Update(element)
	for each degree of freedom **do**
	  Update(degree of freedom)
	end for

TABLE 3. Nested fixed point iteration on the time slab.

Algorithm 7 $d = \text{Iterate(system)}$

```
d ← 0
loop
  for n = 1, ..., n_max do
    if Converged(system) then
      return
    end if
    if Diverged(system) then
      ChangeState()
      break
    end if
    Stabilize(system)
    d ← max(d, Update(system))
  end for
end loop
```

Algorithm 8 $d = \text{Update(system)}$

```
d ← 0
for each sub system do
  d_i ← Iterate(sub system)
  d ← d + d_i^2
end for
d ← √d
```

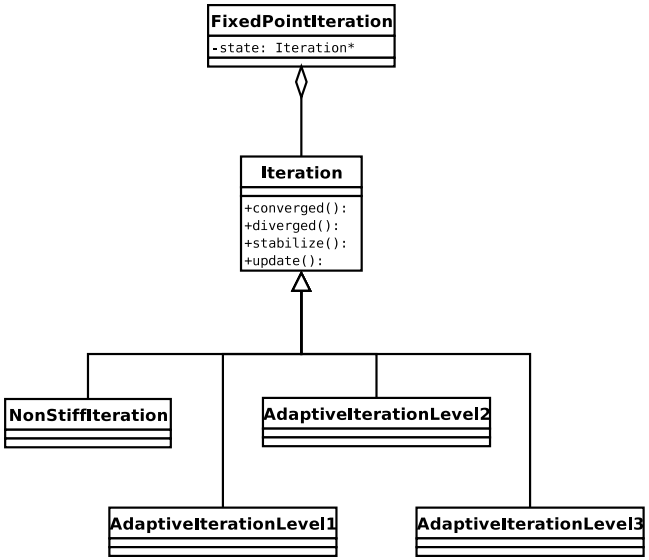

FIGURE 5. UML class diagram showing the implementation of the state machine for adaptive fixed point iteration in **DOLFIN**.

by the base class `Iteration`, in particular the functions `Iteration::converged()`, `Iteration::diverged()`, `Iteration::stabilize()`, and `Iteration::update()` for each level of iteration (element level, element group level, and time slab level). To change the state, the object pointed to by the pointer `state` is deleted and a new object is allocated to the pointer, with the type of the new object determined by the new state. This simplifies the implementation of the state machine and makes it possible to separate the implementation of the different states.

3.4. **Cumulative power iteration.** In each iteration of Algorithm 7, the amount of damping for the fixed point iteration is determined by the algorithm Stabilize. If the convergence rate of the iterations is not satisfactory, then the appropriate damping factor α and the number of stabilizing iterations m are determined by *cumulative power iteration*, as discussed in [13]; the divergence rate ρ is first determined by Algorithm 9, and then α and m

are determined according to

$$\alpha = \frac{1/\sqrt{2}}{1+\rho} \tag{3.2}$$

and

$$m = \log \rho. \tag{3.3}$$

The iteration continues until the divergence rate ρ has converged to within some tolerance tol, typically tol = 10%.

Algorithm 9 ρ = ComputeDivergence(system)

$d_1 \leftarrow$ Update(system)
$d_2 \leftarrow$ Update(system)
$\rho_2 \leftarrow d_2/d_1$
$n \leftarrow 2$
repeat
  $d_1 \leftarrow d_2$
  $d_2 \leftarrow$ Update(system)
  $\rho_1 \leftarrow \rho_2$
  $\rho_2 \leftarrow \rho_2^{(n-1)/n} \cdot (d_2/d_1)^{1/n}$
  $n \leftarrow n + 1$
until $|\rho_2 - \rho_1| < \text{tol} \cdot \rho_1$
$\rho \leftarrow \rho_2$

3.5. **Controlling the individual time steps.** The individual and adaptive time steps k_{ij} are determined during the recursive construction of time slabs based on an a posteriori error estimate for the global error $\|e(T)\|_{l_2}$ at final time, as described in [14, 15]. The a posteriori error estimate is expressed in terms of the individual stability factors $\{S_i(T)\}_{i=1}^N$, the local time steps $\{k_{ij}\}_{j=1,i=1}^{M_i,N}$, and the local residuals $\{r_{ij}\}_{j=1,i=1}^{M_i,N}$. The a posteriori error estimate takes the form

$$\|e(T)\|_{l_2} \leq \sum_{i=1}^{N} S_i(T) \max_{j=1,\ldots,M_i} k_{ij}^{p_{ij}} r_{ij}, \tag{3.4}$$

with $p_{ij} = q_{ij}$ for the mcG(q) method and $p_{ij} = q_{ij} + 1$ for the mdG(q) method.

Basing the time step for interval I_{ij} on the residual of the previous interval $I_{i,j-1}$, since the residual of interval I_{ij} is unknown until the solution on that interval has been computed, we should thus choose the local time step k_{ij} according to

$$k_{ij} = \left(\frac{\text{TOL}}{N\, S_i(T) r_{i,j-1}} \right)^{1/p_{ij}}, \quad j = 2, \ldots, M_i, \quad i = 1, \ldots, N, \tag{3.5}$$

where TOL is a given tolerance.

However, the time steps can not be based directly on (3.5), since that leads to oscillations in the time steps. If $r_{i,j-1}$ is small, then k_{ij} will be large, and as a result r_{ij} will also be large. Consequently, $k_{i,j+1}$ and $r_{i,j+1}$ will be small, and so on. To avoid these oscillations, we adjust the time step $k_{\text{new}} = k_{ij}$ according to Algorithm 10, which determines the new time step as the harmonic mean value of the previous time step and the time step determined by (3.5).

Alternatively, the time steps can be determined using control theory, as suggested in [10, 18]. Typically, a standard PID controller is used to determine the time steps with the goal of satisfying $k_{ij}^{p_{ij}} r_{ij} = \text{TOL}/(N\, S_i(T))$ on $[0, T]$ for $i = 1, \ldots, N$. However, since multi-adaptive time-stepping requires an individual controller for each component, the current implementation of **DOLFIN** determines the time steps according to Algorithm 10.

Algorithm 10 $k = \text{Controller}(k_{\text{new}}, k_{\text{old}}, k_{\max})$

$k \leftarrow 2k_{\text{old}}k_{\text{new}}/(k_{\text{old}} + k_{\text{new}})$
$k \leftarrow \min(k, k_{\max})$

The initial time steps $k_{11} = \cdots = k_{N1} = K_1$ are chosen equal for all components and are determined iteratively for the first time slab. The size K_1 of the first time slab is first initialized to some default value, possibly based on the length T of the time interval, and then adjusted until the local residuals are sufficiently small for all components.

3.6. **Implementation of general elements.** As described in [15], the system of equations to be solved for the degrees of freedom $\{\xi_{ijm}\}$ on each element takes the form

$$\xi_{ijm} = \xi_{ij0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) f_i(U(t), t)\, dt, \quad m = 1, \ldots, q_{ij}, \tag{3.6}$$

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

The integral in (3.6) is computed using quadrature, and thus the weight functions $\{w_m^{[q_{ij}]}\}_{m=1}^{q_{ij}}$ need to be evaluated at a set of quadrature points $\{s_n\} \subset [0, 1]$. In **DOLFIN**, these values are computed and tabulated each time a new type of element is created. If the same method is used for all components throughout the computation, then this computation is carried out only once.

For the mcG(q) method, Lobatto quadrature with $n = q + 1$ quadrature points is used. The $n \geq 2$ Lobatto quadrature points are defined on $[-1, 1]$ as the two end-points together with the roots of the derivative P'_{n-1} of the $(n-1)$th-order Legendre polynomial. The quadrature points are computed

in **DOLFIN** using Newton's method to find the roots of P'_{n-1} on $[-1, 1]$, and are then rescaled to the interval $[0, 1]$.

Similarly, Radau quadrature with $n = q + 1$ quadrature points is used for the mdG(q) method. The $n \geq 1$ Radau points are defined on $[-1, 1]$ as the roots of $Q_n = P_{n-1} + P_n$, where P_{n-1} and P_n are Legendre polynomials. Note that the left end-point is always a quadrature point. As for the mcG(q) method, Newton's method is used to find the roots of Q_n on $[-1, 1]$. The quadrature points are then rescaled to $[0, 1]$, with time reversed to include the right end-point.

Since Lobatto quadrature with n quadrature points is exact for polynomials of degree $p \leq 2n - 3$ and Radau quadrature with n quadrature points is exact for polynomials of degree $p \leq 2n - 2$, both quadrature rules are exact for polynomials of degree $n - 1$ for $n \geq 2$ and $n \geq 1$, respectively. With both quadrature rules, the integral of the Legendre polynomial P_p on $[-1, 1]$ should thus be zero for $p = 0, \ldots, n - 1$. This defines a linear system, which is solved to obtain the quadrature weights.

After the quadrature points $\{s_n\}_{n=0}^{q_{ij}}$ have been determined, the polynomial weight functions $\{w_m^{[q_{ij}]}\}_{m=1}^{q_{ij}}$ are computed as described in [14] (again by solving a linear system) and then evaluated at the quadrature points. Multiplying these values with the quadrature weights, we rewrite (3.6) in the form

$$\xi_{ijm} = \xi_{ij0} + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} f_i(U(t_{i,j-1} + s_n k_{ij}), t_{i,j-1} + s_n k_{ij}), \quad m = 1, \ldots, q_{ij}. \tag{3.7}$$

General order mcG(q) and mdG(q) have been implemented in **DOLFIN**. The two methods are implemented by the two classes `cGqElement` and `dGqElement`, implementing the interface specified by the common base class `Element`. Both classes take the order q as an argument to its constructor and implement the appropriate version of (3.7).

3.7. **Automatic generation of the dual problem.** The dual problem of (1.1) for $\phi = \phi(t)$ that is solved to obtain stability factors and error estimates is given by

$$\begin{aligned} -\dot{\phi}(t) &= J(U, t)^\top \phi(t), \quad t \in [0, T), \\ \phi(T) &= \psi, \end{aligned} \tag{3.8}$$

where $J(U, t)$ denotes the Jacobian of the right-hand side f of (1.1) at time t. Note that we need to linearize around the computed solution U, since the exact solution u of (1.1) is not known. To solve this backward problem over $[0, T)$ using the forward integrator Algorithm 1, we rewrite (3.8) as a forward problem. With $w(t) = \phi(T - t)$, we have $\dot{w} = -\dot{\phi}(T - t) = J^\top(U, T - t)w(t)$,

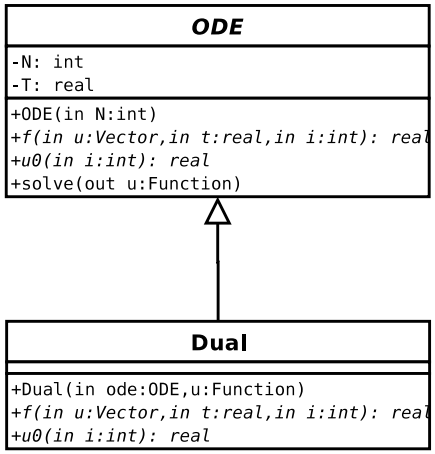

FIGURE 6. The dual problem is implemented as a subclass of the common base class for ODEs of the form (1.1).

and so (3.8) can be written as a forward problem for w in the form

$$\begin{aligned} \dot{w}(t) &= f^*(w(t),t) \equiv J(U,T-t)^\top w(t), \quad t \in (0,T], \\ w(0) &= \psi. \end{aligned} \tag{3.9}$$

In **DOLFIN**, the initial value problem for w is implemented as a subclass of the ODE base class, as shown in Figure 6, which makes it possible to solve the dual problem using the same algorithm as the primal problem.

The constructor of the dual problem takes as arguments the primal problem (1.1) and the computed solution U of the primal problem. The right-hand side f^* of the dual problem is automatically generated by numerical differentiation of the right-hand side of the primal problem.

3.8. **Interpolation of the solution.** To update the degrees of freedom for an element according to (3.7), the appropriate component f_i of the right-hand side of (1.1) needs to be evaluated at the set of quadrature points. In order for f_i to be evaluated, each component U_j of the computed solution U on which f_i depends has to be evaluated at the quadrature points. We let $\mathcal{S}_i \subseteq \{1,\ldots,N\}$ denote the *sparsity pattern* of component U_i, i.e., the set of components on which f_i depend,

$$\mathcal{S}_i = \{j \in \{1,\ldots,N\} : \partial f_i/\partial u_j \neq 0\}. \tag{3.10}$$

Thus, to evaluate f_i at a given time t, only the components U_j for $j \in S_i$ need to be evaluated at t, see Algorithm 11. This is of particular importance for

problems of sparse structure and makes it possible to use the multi-adaptive solver for the integration of time-dependent PDEs, see Section 4. The sparsity pattern $\mathcal{S}_i$ is automatically detected by the solver. Alternatively, the sparsity pattern can be specified by a (sparse) matrix.

Algorithm 11 $y = \text{EvaluateRightHandSide}(i,t)$

for $j \in \mathcal{S}_i$ **do**
  $x(j) \leftarrow U_j(t)$
end for
$y \leftarrow f_i(x,t)$

The key part of Algorithm 11 is the evaluation of a given component U_i at a given time t. To evaluate $U_i(t)$, the solver first needs to find the element $(I_{ij}, U_i|_{I_{ij}})$ satisfying $t \in I_{ij}$. The local polynomial $U_i|_{I_{ij}}$ is then evaluated (interpolated) at the given time t. During the construction of a time slab, an element such that $t \in I_{ij}$ might not exists, in which case the last element of component U_i is used and extrapolated to the given time t.

To find the element $(I_{ij}, U_i|_{I_{ij}})$ such that $t \in I_{ij}$, the function `std::upper_bound()` is used. This function is part of the Standard C++ Library and uses binary search to find the appropriate element from the ordered sequence of elements for component U_i, which means that the complexity of finding the element is logarithmic. In addition, the speed of the evaluation is increased by caching the latest used element and each time checking this element before the binary search is performed.

3.9. **Storing the solution.** Since the computed solution U of the primal problem (1.1) is needed for the computation of the discrete solution Φ of the dual problem (3.8), the solution needs to be stored to allow U to be evaluated at any given $t \in [0,T]$.

The solution is stored on disk in a temporary file created by the function `tmpfile()`, which is part of the C standard I/O library. The solution is written in blocks to the file in binary format. During the computation, a cache of blocks is kept in main memory to allow efficient evaluation of the solution. The number of blocks kept in memory depends on the amount of memory available. For a sufficiently large problem, only one block will be kept in memory. Each time a value is requested which is not in one of the available blocks, one of these blocks is dropped and the appropriate block is fetched from file.

4. Solving the bistable equation

As an example of multi-adaptive time-stepping, we solve the bistable equation on the unit cube,

$$\begin{aligned} \dot{u} - \epsilon \Delta u &= u(1-u^2) \quad \text{in } \Omega \times (0,T], \\ \partial_n u &= 0 \quad \text{on } \partial\Omega \times (0,T], \\ u(\cdot,0) &= u_0 \quad \text{in } \Omega, \end{aligned} \tag{4.1}$$

with $\Omega = (0,1) \times (0,1) \times (0,1)$, $\epsilon = 0.0001$, final time $T = 100$, and with random initial data $u_0 = u_0(x)$ distributed uniformly on $[-1,1]$.

The bistable equation has been studied extensively before [7, 6] and has interesting stability properties. In particular, it has two stable steady-state solutions, $u = 1$ and $u = -1$, and one unstable steady-state solution, $u = 0$. From (4.1), it is clear that the solution increases in regions where it is positive and decreases in regions where it is negative. Because of the diffusion, neighboring regions will compete until finally the solution has reached one of the two stable steady states. Since this action is local on the interface between positive and negative regions, the bistable equation is an ideal example for multi-adaptive time-stepping.

To solve the bistable equation (4.1) using multi-adaptive time-stepping, we discretize in space using the standard cG(1) method [5]. Lumping the mass matrix, we obtain an ODE initial value problem of the form (1.1), which we solve using the multi-adaptive mcG(1) method. We refer to the overall method thus obtained as the cG(1)mcG(1) method.

The solution was computed on a uniformly refined tetrahedral mesh with mesh size $h = 1/64$. This mesh consists of $1,572,864$ tetrahedrons and has $N = 274,625$ nodes. In Figure 7, we plot the initial value used for the computation, and in Figure 8 the solution at final time $T = 100$. We also plot the solution and the multi-adaptive time steps at time $t = 10$ in Figure 9 and Figure 10, and note that the time steps are small in regions where there is strong competition between the two stable steady-state solutions, in particular in regions with where the curvature of the interface is small.

The computation was carried out using **DOLFIN** version 0.4.9, which includes the bistable equation as one of numerous multi-adaptive test problems. The visualization of the solution was made using OpenDX [1] version 4.3.2. Animations of the solution and the multi-adaptive time steps are available in the gallery on the **DOLFIN** web page [11].

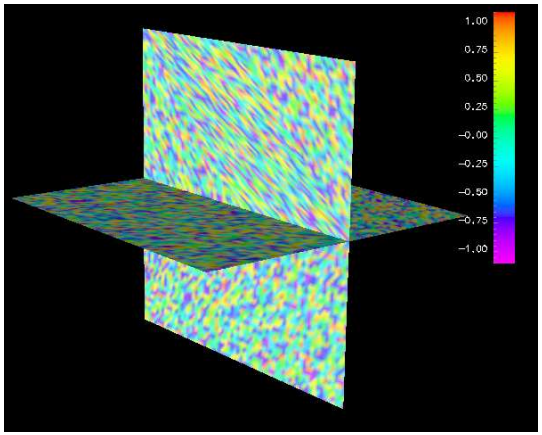

FIGURE 7. Initial data for the solution of the bistable equation (4.1).

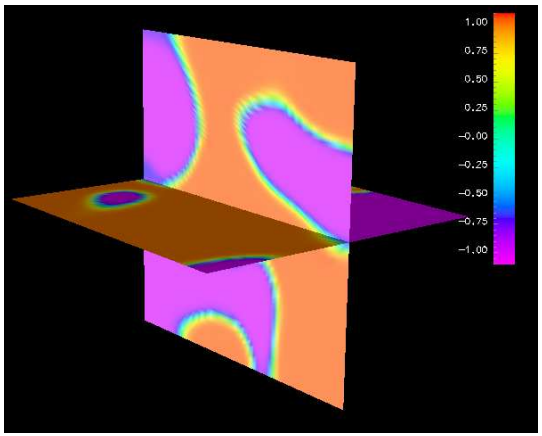

FIGURE 8. Solution of the bistable equation (4.1) at final time $T = 100$.

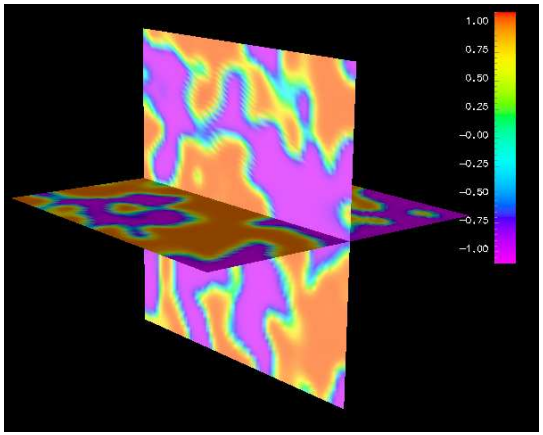

FIGURE 9. Solution of the bistable equation (4.1) at time $t = 10$.

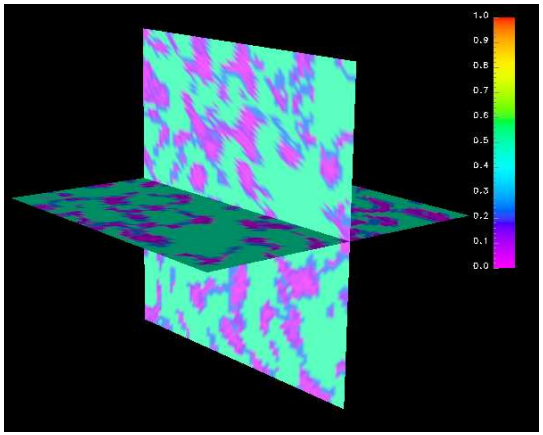

FIGURE 10. Multi-adaptive time steps at time $t = 10$ for the solution of the bistable equation (4.1).

REFERENCES

[1] *OpenDX*, http://www.opendx.org/.

[2] R. BECKER AND R. RANNACHER, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica, 10 (2001).

[3] T. DUPONT, J. HOFFMAN, C. JOHNSON, R.C. KIRBY, M.G. LARSON, A. LOGG, AND L.R. SCOTT, *The FEniCS project*, Tech. Rep. 2003–21, Chalmers Finite Element Center Preprint Series, 2003.

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

[5] ———, *Computational Differential Equations*, Cambridge University Press, 1996.

[6] D. ESTEP, *An analysis of numerical approximations of metastable solutions of the bistable equation*, Nonlinearity, 7 (1994), pp. 1445–1462.

[7] D. ESTEP, M. LARSON, AND R. WILLIAMS, *Estimating the error of numerical solutions of systems of nonlinear reaction–diffusion equations*, Memoirs of the American Mathematical Society, 696 (2000), pp. 1–109.

[8] FREE SOFTWARE FOUNDATION, *GNU GPL*, http://www.gnu.org/copyleft/gpl.html.

[9] E. GAMMA, R. HELM, R. JOHNSON, AND J. VLISSIDES, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

[10] 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.

[11] J. HOFFMAN AND A. LOGG ET AL., *DOLFIN*, http://www.phi.chalmers.se/dolfin/.

[12] J. HOFFMAN AND A. LOGG, *DOLFIN: Dynamic Object oriented Library for FINite element computation*, Tech. Rep. 2002–06, Chalmers Finite Element Center Preprint Series, 2002.

[13] J. JANSSON AND A. LOGG, *Multi-adaptive Galerkin methods for ODEs V: Stiff problems*, submitted to BIT, (2004).

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

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

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

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

[18] G. SÖDERLIND, *Digital filters in adaptive time-stepping*, ACM Trans. Math. Softw., 29 (2003), pp. 1–26.

SIMULATION OF MECHANICAL SYSTEMS WITH INDIVIDUAL TIME STEPS

JOHAN JANSSON AND ANDERS LOGG

Abstract. The simulation of a mechanical system involves the formulation of a differential equation (modeling) and the solution of the differential equation (computing). The solution method needs to be efficient as well as accurate and reliable. This paper discusses multi-adaptive Galerkin methods in the context of mechanical systems. The primary type of mechanical system studied is an extended mass–spring model. A multi-adaptive method integrates the mechanical system using individual time steps for the different components of the system, adapting the time steps to the different time scales of the system, potentially allowing enormous improvement in efficiency compared to traditional mono-adaptive methods.

1. Introduction

Simulation of mechanical systems is an important component of many technologies of modern society. It appears in industrial design, for the prediction and verification of mechanical products. It appears in virtual reality, both for entertainment in the form of computer games and movies, and in the simulation of realistic environments such as surgical training on virtual and infinitely resurrectable patients. Common to all these applications is that the computation time is critical. Often, an application is real-time, which means that the time inside the simulation must reasonably match the time in the real world.

Simulating a mechanical system involves both *modeling* (formulating an equation describing the system) and *computation* (solving the equation). The model of a mechanical system often takes the form of an initial value problem for a system of ordinary differential equations of the form

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

Date: April 27, 2004.

Key words and phrases. Multi-adaptivity, individual time steps, local time steps, ODE, continuous Galerkin, discontinuous Galerkin, mcgq, mdgq, mechanical system, mass–spring model.

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.

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 simulation of a mechanical system thus involves the formulation of a model of the form (1.1) and the solution of (1.1) using a time-stepping method. We present below *multi-adaptive Galerkin* methods for the solution of (1.1) with individual time steps for the different parts of the mechanical system.

1.1. Mass–spring systems.

A mass–spring system consists of a set of point masses connected by springs, typically governed by Hooke's law with other laws optionally present, such as viscous damping and external forces. Mass–spring systems appear to encompass most of the behaviors of elementary mechanical systems and thus represent a simple, intuitive, and powerful model for the simulation of mechanical systems. This is the approach taken in this paper.

However, to obtain a physically accurate model of a mechanical system, we believe it is necessary to solve a system of partial differential equations properly describing the mechanical system, in the simplest case given by the equations of linear elasticity. Discretizing the system of PDEs in space, for example using the Galerkin finite element method, an initial value problem for a system of ODEs of the form (1.1) is obtained. The resulting system can be interpreted as a mass–spring system and thus the finite element method in combination with a PDE model represents a systematic methodology for the generation of a mass–spring model of a given mechanical system.

1.2. Time-stepping methods.

Numerical methods for the (approximate) solution of (1.1) are almost exclusively based on time-stepping, i.e., the step-wise integration of (1.1) to obtain an approximation U of the solution u satisfying

$$u(t_j) = u(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(u(t),t)\,dt, \quad j = 1,\ldots,M, \tag{1.2}$$

for a partition $0 = t_0 < t_1 < \cdots < t_M = T$ of $[0,T]$. The approximate solution $U \approx u$ is obtained by an appropriate approximation of the integral $\int_{t_{j-1}}^{t_j} f(u(t),t)\,dt$.

Selecting the appropriate size of the time steps $\{k_j = t_j - t_{j-1}\}_{j=1}^M$ is essential for efficiency and accuracy. We want to compute the solution U using as little work as possible, which means using a small number of large time steps. At the same time, we want to compute an accurate solution U which is close to the exact solution u, which means using a large number of small time steps. Often, the accuracy requirement is given in the form of a *tolerance* TOL for the size of the *error* $e = U - u$ in a suitable norm. The competing goals of efficiency and accuracy can be met using an *adaptive*

algorithm, determining a sequence of time steps $\{k_j\}_{j=1}^M$ which produces an approximate solution U satisfying the given tolerance with minimal work.

Galerkin finite element methods present a general framework for the numerical solution of (1.1), including adaptive algorithms for the automatic construction of an optimal time step sequence, see [7, 8]. The Galerkin finite element method for (1.1) reads: Find $U \in V$, such that

$$\int_0^T (\dot{U}, v)\,dt = \int_0^T (f, v)\,dt \quad \forall v \in \hat{V}, \tag{1.3}$$

where $(\cdot,\cdot)$ denotes the $\mathbb{R}^N$ inner product and $(V, \hat{V})$ denotes a suitable pair of finite dimensional subspaces (the *trial* and *test spaces*).

Typical choices of approximating spaces include

$$\begin{aligned} V &= \{v \in [\mathcal{C}([0,T])]^N : v|_{I_j} \in [\mathcal{P}^q(I_j)]^N, \quad j = 1,\ldots,M\}, \\ \hat{V} &= \{v : v|_{I_j} \in [\mathcal{P}^{q-1}(I_j)]^N, \quad j = 1,\ldots,M\}, \end{aligned} \tag{1.4}$$

i.e., V represents the space of continuous and piecewise polynomial vector-valued functions of degree $q \geq 1$ and $\hat{V}$ represents the space of discontinuous piecewise polynomial vector-valued functions of degree $q-1$ on a partition of $[0,T]$. We refer to this as the cG(q) method. With both V and $\hat{V}$ representing discontinuous piecewise polynomials of degree $q \geq 0$, we obtain the dG(q) method. Early work on the cG(q) and dG(q) methods include [6, 19, 10, 9].

By choosing a constant test function v in (1.3), it follows that both the cG(q) and dG(q) solutions satisfy the relation

$$U(t_j) = U(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(U(t), t)\,dt, \quad j = 1,\ldots,M, \tag{1.5}$$

corresponding to (1.2).

In the simplest case of the dG(0) method, we note that $\int_{t_{j-1}}^{t_j} f(U(t),t)\,dt \approx k_j f(U(t_j), t_j)$, since U piecewise constant, with equality if f does not depend explicitly on t. We thus obtain the method

$$U(t_j) = U(t_{j-1}) + k_j f(U(t_j), t_j), \quad j = 1,\ldots,M, \tag{1.6}$$

which we recognize as the backward (or implicit) Euler method. In general, a cG(q) or dG(q) method corresponds to an implicit Runge–Kutta method, with details depending on the choice of quadrature for the approximation of the integral of $f(U,\cdot)$.

1.3. **Multi-adaptive time-stepping.** Standard methods for the discretization of (1.1), including the cG(q) and dG(q) methods, require that the same time steps $\{k_j\}_{j=1}^M$ are used for all components $U_i = U_i(t)$ of the approximate solution U of (1.1). This can be very costly if the system exhibits multiple time scales of different magnitudes. If the different time scales are localized to different components, efficient representation and computation of the solution thus requires that this difference in time scales is reflected

in the choice of approximating spaces $(V, \hat{V})$. We refer to the resulting methods, recently introduced in a series of papers [20, 21, 22, 23, 16], as *multi-adaptive Galerkin methods*.

Surprisingly, individual time-stepping (multi-adaptivity) has previously received little attention in the large literature on numerical methods for ODEs, see e.g. [3, 12, 13, 2, 27, 1], but has been used to some extent for specific applications, including specialized integrators for the n-body problem [24, 4, 26], and low-order methods for conservation laws [25, 18, 5].

1.4. **Obtaining the software.** The examples presented below have been obtained using **DOLFIN** version 0.4.11 [14]. **DOLFIN** is licensed under the GNU General Public License [11], which means that anyone is free to use or modify the software, provided these rights are preserved. The source code of **DOLFIN**, including numerous example programs, is available at the **DOLFIN** web page, `http://www.phi.chalmers.se/dolfin/`, and each new release is announced on freshmeat.net. Alternatively, the source code can be obtained through anonymous CVS as explained on the web page. Comments and contributions are welcome.

The mechanical systems presented in the examples have been implemented using *Ko*, which is a software system for the simulation of mass–spring models, based on **DOLFIN**s multi-adaptive ODE-solver. Ko will be released shortly under the GNU General Public License and will be available at `http://www.phi.chalmers.se/ko/`.

1.5. **Outline of the paper.** We first describe the basic mass–spring model in Section 2 and then give a short introduction to multi-adaptive Galerkin methods in Section 3. In Section 4, we discuss the interface of the multi-adaptive solver and its application to mass–spring models. In Section 5, we investigate and analyze the performance of the multi-adaptive methods for a set of model problems. Finally, we present in Section 6 results for a number of large mechanical systems to demonstrate the potential and applicability of the proposed methods.

2. Mass–spring model

We have earlier in [17] described an extended mass–spring model for the simulation of systems of deformable bodies.

The mass–spring model represents bodies as systems of discrete *mass elements*, with the forces between the mass elements transmitted using explicit *spring connections*. (Note that "spring" is a historical term, and is not limited to pure Hookean interactions.) Given the forces acting on an element, we can determine its motion from Newton's second law,

$$F = ma, \tag{2.1}$$

where F denotes the force acting on the element, m is the mass of the element, and $a = \ddot{x}$ is the acceleration of the element with $x = (x_1, x_2, x_3)$

A *mass element* e is a set of parameters $\{x,\ v,\ m,\ r,\ C\}$:

x	:	position
v	:	velocity
m	:	mass
r	:	radius
C	:	a set of spring connections

A *spring connection* c is a set of parameters $\{e_1,\ e_2,\ \kappa,\ b,\ l_0,\ l_f\}$:

e_1	:	the first mass element connected to the spring
e_2	:	the second mass element connected to the spring
κ	:	Hooke spring constant
b	:	damping constant
l_0	:	rest length
l_f	:	fracture length

TABLE 1. Descriptions of the basic elements of the mass–spring model: mass elements and spring connections.

the position of the element. The motion of the entire body is then implicitly described by the motion of its individual mass elements.

The force given by a standard spring is assumed to be proportional to the elongation of the spring from its rest length. We extend the standard model with contact, collision and fracture, by adding a radius of interaction to each mass element, and dynamically creating and destroying spring connections based on contact and fracture conditions.

In Table 1 and Figure 1, we give the basic properties of the mass–spring model consisting of mass elements and spring connections. With these definitions, a mass–spring model may thus be given by just listing the mass elements and spring connections of the model.

3. Multi-adaptive Galerkin methods

The multi-adaptive methods mcG(q) and mdG(q) used for the simulation are obtained as extensions of the standard cG(q) and dG(q) methods by enriching the trial and test spaces $(V, \hat{V})$ of (1.3) to allow each component U_i of the approximate solution U to be piecewise polynomial on an individual partition of $[0, T]$.

3.1. **Definition of the methods.** To give the definition of the multi-adaptive Galerkin methods, we introduce the following notation: 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* $k_{ij} = t_{ij} - t_{i,j-1}$ for $j = 1, \ldots, M_i$. This is illustrated in Figure 2. We also assume that the interval $[0, T]$ is partitioned into blocks between certain synchronized time

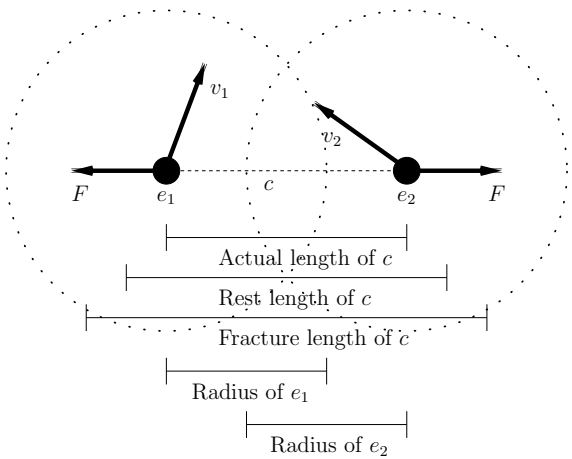

FIGURE 1. Schema of two mass elements e_1 and e_2, a spring connection c, and important quantities.

levels $0 = T_0 < T_1 < \cdots < T_M = T$. 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*.

With this notation, we can write the mcG(q) method for (1.1) in the following form: Find $U \in V$, such that

$$\int_0^T (\dot{U}, v)\, dt = \int_0^T (f, v)\, dt \quad \forall v \in \hat{V}, \tag{3.1}$$

where the trial space V and test space $\hat{V}$ are given by

$$V = \{v \in [\mathcal{C}([0,T])]^N : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}), \quad j = 1, \ldots, M_i, \quad i = 1, \ldots, N\},$$
$$\hat{V} = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}-1}(I_{ij}), \quad j = 1, \ldots, M_i, \quad i = 1, \ldots, N\}. \tag{3.2}$$

The mcG(q) method is thus obtained as a simple extension of the standard cG(q) method by allowing each component to be piecewise polynomial on an individual partition of $[0, T]$. Similarly, we obtain the mdG(q) method as a simple extension of the standard dG(q) method. For a detailed description of the multi-adaptive Galerkin methods, we refer the reader to [20, 21, 22, 23, 16]. In particular, we refer to [20] or [22] for the full definition of the methods.

3.2. **Adaptivity.** The individual time steps $\{k_{ij}\}_{j=1,i=1}^{M_i,N}$ are chosen adaptively based on an a posteriori error estimate for the global error $e = U - u$

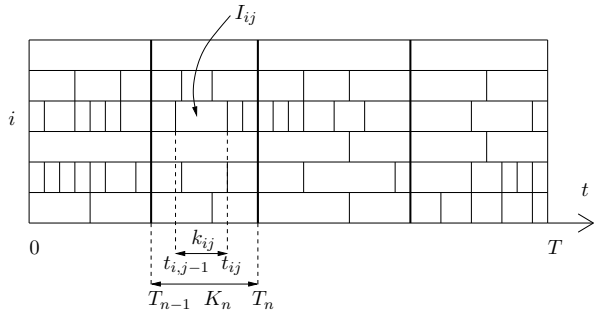

FIGURE 2. Individual partitions of the interval $[0, T]$ for different components. Elements between common synchronized time levels are organized in time slabs. In this example, we have $N = 6$ and $M = 4$.

at final time $t = T$, as discussed in [20, 21]. The a posteriori error estimate for the mcG(q) method takes the form

$$\|e(T)\|_{l_2} \leq C_q \sum_{i=1}^{N} S_i^{[q_i]}(T) \max_{[0,T]} |k_i^{q_i} R_i(U, \cdot)|, \tag{3.3}$$

where C_q is an interpolation constant, $S_i^{[q_i]}(T)$ are the individual *stability factors*, $k_i = k_i(t)$ are the individual time steps, and $R_i(U, \cdot) = \dot{U}_i - f_i(U, \cdot)$ are the individual *residuals* for $i = 1, \ldots, N$. The individual stability factors $S_i^{[q_i]}(T)$, measuring the effect of local errors introduced by a nonzero local residual on the global error, are obtained from the solution ϕ of an associated *dual problem*, see [7] or [20].

Thus, to determine the individual time steps, we measure the individual residuals and take each individual time step k_{ij} such that

$$k_{ij}^{q_{ij}} \max_{I_{ij}} |R_i(U, \cdot)| = \mathrm{TOL}/(N C_q S_i^{[q_i]}(T)), \tag{3.4}$$

where TOL is a tolerance for the error at final time. See [21] or [15] for a detailed discussion of the algorithm for the automatic selection of the individual time steps.

3.3. **Iterative methods.** The system of discrete nonlinear equations defined by (3.1) is solved by fixed point iteration on time slabs, as described in [16]. For a stiff problem, the fixed point iteration is automatically stabilized by introducing a damping parameter which is adaptively determined through feed-back from the computation. We refer to this as *adaptive fixed point iteration*.

4. Multi-adaptive simulation of mass–spring systems

The simulation of a mechanical system involves the formulation of a differential equation (modeling) and the solution of the differential equation (computing). Having defined these two components in the form of the mass–spring model presented in Section 2 and the multi-adaptive solver presented in Section 3, we comment briefly on the user interface of the multi-adaptive solver.

The user interface of the multi-adaptive solver is specified in terms of an ODE base class consisting of a right hand side f, a time interval $[0, T]$, and an initial value u_0, as shown in Table 2. To solve an ODE, the user implements a subclass which inherits from the ODE base class.

```
class ODE
{
  ODE(int N);

  virtual real u0(int i);
  virtual real f(Vector u, real t, int i);
}
```

TABLE 2. Sketch of the C++ interface of the multi-adaptive ODE-solver.

The mass–spring model presented above has been implemented using Ko, a software system for the simulation and visualization of mass–spring models. Ko automatically generates a mass–spring model from a geometric representation of a given system, as shown in Figure 3. The mass–spring model is then automatically translated into a system of ODEs of the form (1.1). Ko specifies the ODE system as an ODE subclass and uses **DOLFIN** to compute the solution.

Ko represents a mass–spring model internally as lists of mass elements and spring connections. To evaluate the right-hand side f of the corresponding ODE system, a translation or mapping is thus needed between a given mass element and a component number in the system of ODEs. This mapping may take a number different forms; Ko uses the mapping presented in Algorithm 1.

5. Performance

We consider a simple model problem consisting of a long string of n point masses connected with springs as shown in Figure 4. The first mass on the left is connected to a fixed support through a hard spring with large spring constant $\kappa \gg 1$. All other springs are connected together with soft springs with spring constant $\kappa = 1$. As a result, the first mass oscillates at a high frequency, with the rest of the masses oscillating slowly. In Figure 5, we plot the coordinates for the first three masses on $[0, 1]$.

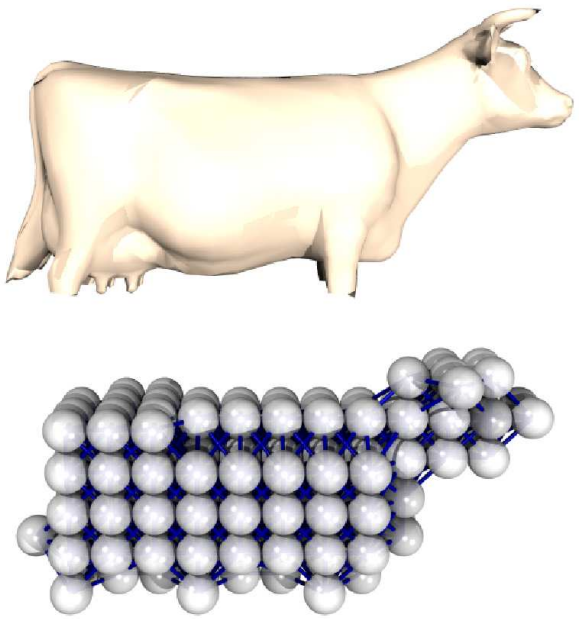

FIGURE 3. A geometric representation of a cow is automatically translated into a mass–spring model.

Algorithm 1 FromComponents(Vector u, Mass m)

$i \leftarrow \text{index}(m)$
$N \leftarrow \text{size}(u)$

$m.x_1 \leftarrow u(3(i-1)+1)$
$m.x_2 \leftarrow u(3(i-1)+2)$
$m.x_3 \leftarrow u(3(i-1)+3)$

$m.v_1 \leftarrow u(N/2+3(i-1)+1)$
$m.v_2 \leftarrow u(N/2+3(i-1)+2)$
$m.v_3 \leftarrow u(N/2+3(i-1)+3)$

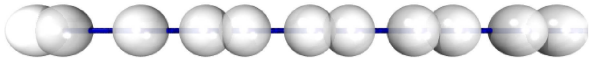

FIGURE 4. The mechanical system used for the performance test. The system consists of a string of masses, fixed at the left end. Each mass has been slightly displaced to initialize the oscillations.

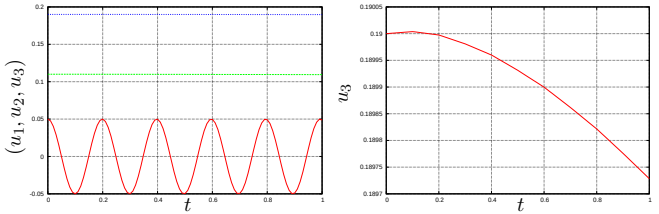

FIGURE 5. Coordinates for the first three masses of the simple model problem (left) and for the third mass (right).

To compare the performance of the multi-adaptive solver (in the case of the mcG(1) method) with a mono-adaptive method (the cG(1) method), we choose a fixed small time step k for the first mass and a fixed large time step $K > k$ for the rest of the masses in the multi-adaptive simulation, and use the same small time step k for all masses in the mono-adaptive simulation. We let $M = K/k$ denote the number of small time steps per each large time step.

We run the test for $M = 100$ and $M = 200$ with large spring constant $\kappa = 10M$ for the hard spring connecting the first mass to the fixed support. We use a large time step of size $K = 0.1$ and, consequently, a small time step of size $k = 0.1/M$. The computation time T_c is recorded as function of the number of masses n.

As shown in Figure 6, the computation time for the multi-adaptive solver grows slowly with the number of masses n, practically remaining constant; small time steps are used only for the first rapidly oscillating mass and so the work is dominated by frequently updating the first mass, independent of the total number of masses. On the other hand, the work for the mono-adaptive method grows linearly with the total number of masses, as expected.

More specifically, the complexity of the mono-adaptive method may be expressed in terms of M and n as follows:

$$T_c(M, n) = C_1 + C_2 M n, \tag{5.1}$$

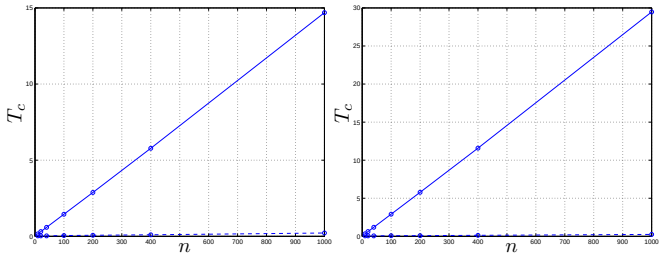

FIGURE 6. Computation time T_c as function of the number of masses n for the multi-adaptive solver (dashed) and a mono-adaptive method (solid), with $M = 100$ (left) and $M = 200$ (right).

while for the multi-adaptive solver, we obtain

$$T_c(M, n) = C_3 M + C_4 n. \tag{5.2}$$

Our general conclusion is that the multi-adaptive solver is more efficient than a mono-adaptive method for the simulation of a mechanical system if M is large, i.e., when small time steps are needed for a part of the system, and if n is large, i.e, if large time steps may be used for a large part of the system.

The same result is obtained if we add damping to the system in the form of a damping constant of size $b = 100$ for the spring connections between the slowly oscillating masses, resulting in gradual damping of the slow oscillations, while keeping the rapid oscillations of the first mass largely unaffected. With $b = 100$, adaptive fixed point iteration is automatically activated for the solution of the discrete equations, as discussed in Section 3.3.

6. Large problems and applications

To demonstrate the potential and applicability of the proposed mass–spring model and the multi-adaptive solver, we present results for a number of large mechanical systems.

6.1. **Oscillating tail.** For the first example, we take the mass–spring model of Figure 3 representing a heavy cow and add a light mass representing its tail, as shown in Figure 7. The cow is given a constant initial velocity and the tail is given an initial push to make it oscillate. A sequence of frames from an animation of the multi-adaptive solution is given in Figure 8.

We compare the performance of the multi-adaptive solver (in the case of the mcG(1) method) with a mono-adaptive method (the cG(1) method) using the same time steps for all components. We also make a comparison with

a simple non-adaptive implementation of the cG(1) method, with minimal overhead, using constant time steps equal to the smallest time step selected by the mono-adaptive method.

As expected, the multi-adaptive solver automatically selects small time steps for the oscillating tail and large time steps for the rest of the system. In Figure 9, we plot the time steps as function of time for relevant components of the system. We also plot the corresponding solutions in Figure 11. In Figure 10, we plot the time steps used in the mono-adaptive simulation.

The computation times are given in Table 3. The speed-up of the multi-adaptive method compared to the mono-adaptive method is a factor 70. Compared to the simple non-adaptive implementation of the cG(1) method, using a minimal amount of work, the speed-up is a factor 3. This shows that the speed-up of a multi-adaptive method can be significant. It also shows that the overhead is substantial for the current implementation of the multi-adaptive solver, including the organization of the multi-adaptive time slabs, interpolation of solution values within time slabs, and the evaluation of residuals for multi-adaptive time-stepping. However, we believe it is possible to remove a large part of this overhead.

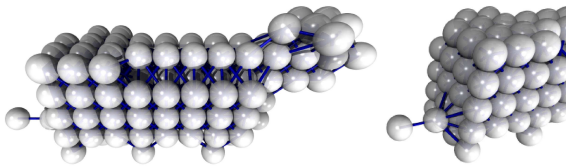

FIGURE 7. A cow with an oscillating tail (left) with details of the tail (right).

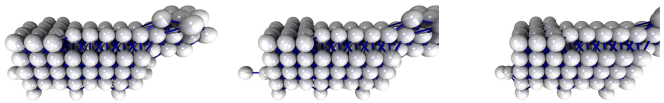

FIGURE 8. The tail oscillates rapidly while the rest of the cow travels at a constant velocity to the right.

6.2. **Local manipulation.** For the next example, we fix a damped cow shape at one end and repeatedly push the other end with a manipulator in the form of a large sphere, as illustrated in Figure 12.

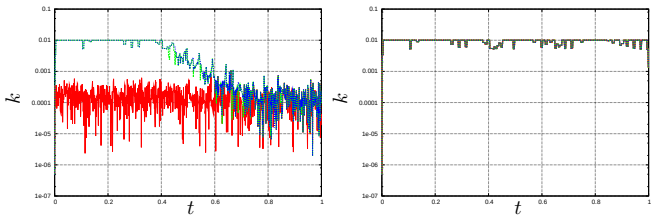

FIGURE 9. Multi-adaptive time steps used in the simulation of the cow with oscillating tail. The plot on the left shows the time steps for components 481–483 corresponding to the velocity of the tail, and the plot on the right shows the time steps for components 13–24 corresponding to the positions for a set of masses in the interior of the cow.

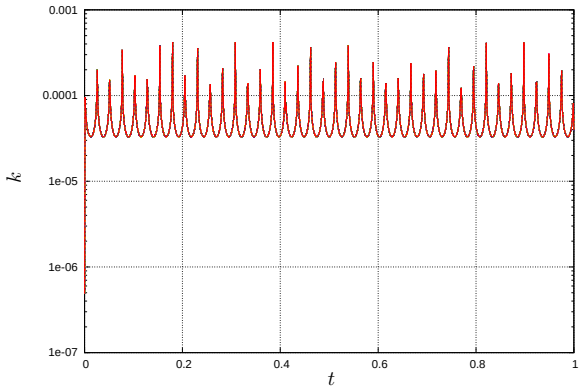

FIGURE 10. Mono-adaptive time steps for the cow with oscillating tail.

As shown in Figure 13, the multi-adaptive solver automatically selects small time steps for the components directly affected by the manipulation. This test problem also illustrates the basic use of adaptive time-stepping; the time steps are drastically decreased at each impact to accurately track the effect of the impact.

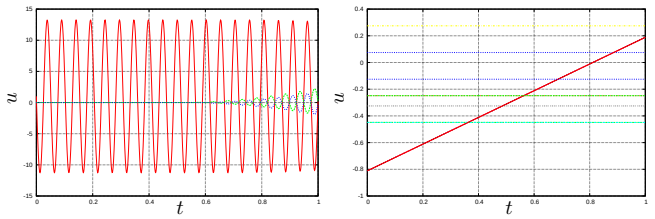

FIGURE 11. Solution for relevant components of the cow with oscillating tail. The plot on the left shows the solution for components 481–483 corresponding to the velocity of the tail, and the plot on the right shows the solution for components 13–24 corresponding to the positions for a set of masses in the interior of the cow.

Algorithm	Time / s
Multi-adaptive	40
Mono-adaptive	2800
Non-adaptive	130

TABLE 3. Computation times for the simulation of the cow with oscillating tail for three different algorithms: multi-adaptive mcG(1), mono-adaptive cG(1), and a simple implementation of non-adaptive cG(1) with fixed time steps and minimal overhead.

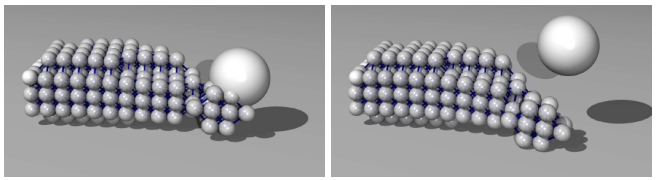

FIGURE 12. A cow shape is locally manipulated. Small time steps are automatically selected for the components affected by the local manipulation, with large time steps for the rest of the system.

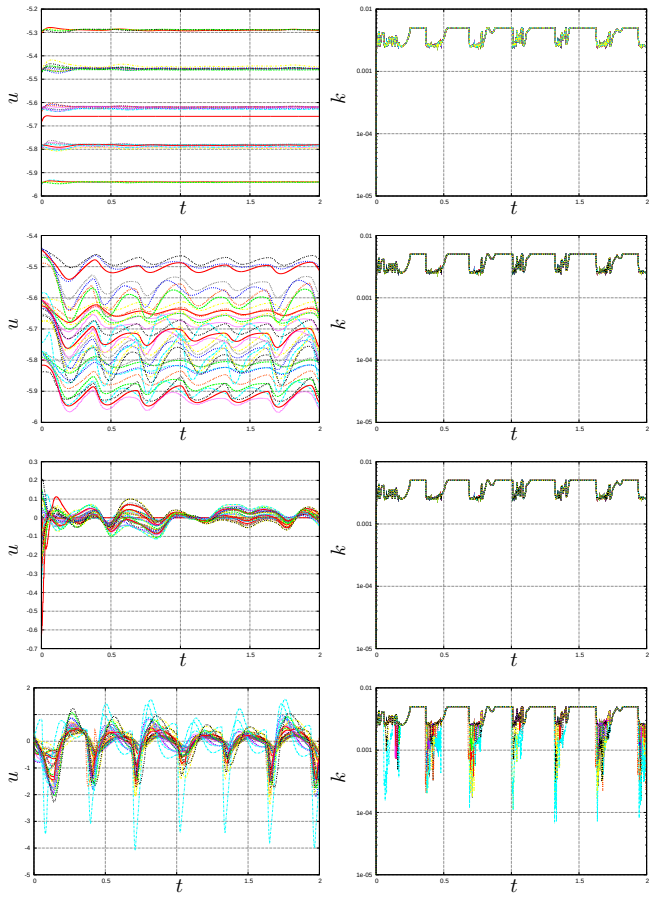

FIGURE 13. Solution (left) and multi-adaptive time steps (right) for selected components of the manipulated cow. The two top rows correspond to the positions of the left- and right-most masses, respectively, and the two rows below correspond to the velocities of the left- and right-most masses, respectively.

6.3. **A stiff beam.** Our final example demonstrates the applicability of the multi-adaptive solver to a stiff problem consisting of a block being dropped onto a stiff beam, as shown in Figure 14. The material of both the block and the beam is very hard and very damped, with spring constant $\kappa = 10^7$ and damping constant $b = 2 \cdot 10^5$ for each spring connection. The multi-adaptive time steps for the simulation are shown in Figure 15. Note that the time steps are drastically reduced at the time of impact, with large time steps before and after the impact.

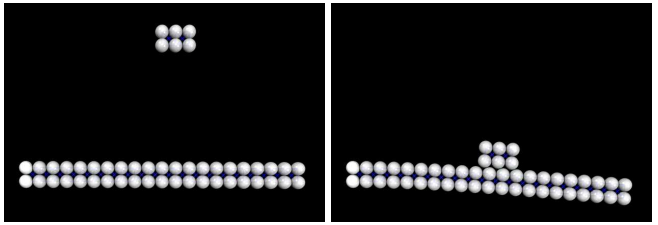

FIGURE 14. A block is dropped onto a beam. The material of both the block and the beam is very hard and very damped, with spring constant $\kappa = 10^7$ and damping constant $b = 2 \cdot 10^5$.

7. Conclusions

From the results presented above, we make the following conclusions regarding multi-adaptive time-stepping:

- A multi-adaptive method outperforms a mono-adaptive method for systems containing different time scales if there is a significant separation of the time scales and if the fast time scales are localized to a relatively small part of the system.
- Multi-adaptive time-stepping, and in particular the current implementation, works in practice for large and realistic problems.

References

[1] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, 1998.
[2] J. Butcher, *The Numerical Analysis of Ordinary Differential Equations — Runge–Kutta and General Linear Methods*, Wiley, 1987.
[3] G. Dahlquist, *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*, PhD thesis, Stockholm University, 1958.
[4] R. Davé, J. Dubinski, and L. Hernquist, *Parallel treeSPH*, New Astronomy, 2 (1997), pp. 277–297.
[5] C. Dawson and R.C. Kirby, *High resolution schemes for conservation laws with locally varying time steps*, SIAM J. Sci. Comput., 22, No. 6 (2001), pp. 2256–2281.

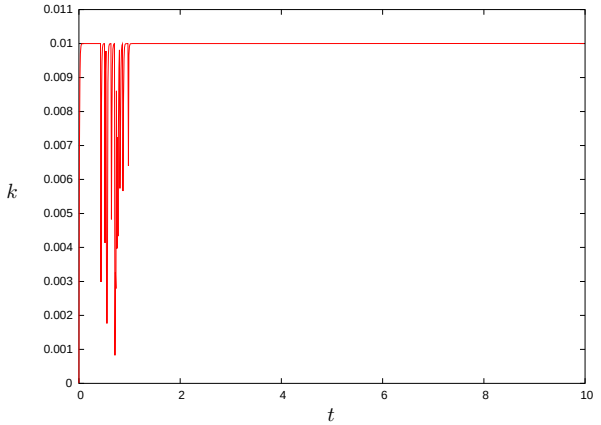

FIGURE 15. Multi-adaptive time steps for the block and beam. Note that the time steps are drastically reduced at the time of impact. The maximum time step is set to 0.01 to track the contact between the block and the beam.

[6] M. DELFOUR, W. HAGER, AND F. TROCHU, *Discontinuous Galerkin methods for ordinary differential equations*, Math. Comp., 36 (1981), pp. 455–473.
[7] K. ERIKSSON, D. ESTEP, P. HANSBO, AND C. JOHNSON, *Introduction to adaptive methods for differential equations*, Acta Numerica, (1995), pp. 105–158.
[8] ———, *Computational Differential Equations*, Cambridge University Press, 1996.
[9] D. ESTEP, *A posteriori error bounds and global error control for approximations of ordinary differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 1–48.
[10] D. ESTEP AND D. FRENCH, *Global error control for the continuous Galerkin finite element method for ordinary differential equations*, M^{2}AN, 28 (1994), pp. 815–852.
[11] FREE SOFTWARE FOUNDATION, *GNU GPL*, http://www.gnu.org/copyleft/gpl.html.
[12] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations I — Nonstiff Problems*, Springer Series in Computational Mathematics, vol 8, 1991.
[13] ———, *Solving Ordinary Differential Equations II — Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, vol 14, 1991.
[14] J. HOFFMAN AND A. LOGG ET AL., *DOLFIN*, http://www.phi.chalmers.se/dolfin/.
[15] J. JANSSON AND A. LOGG, *Algorithms for multi-adaptive time-stepping*, submitted to ACM Trans. Math. Softw., (2004).
[16] ———, *Multi-adaptive Galerkin methods for ODEs V: Stiff problems*, submitted to BIT, (2004).
[17] J. JANSSON AND J. VERGEEST, *A discrete mechanics model for deformable bodies*, Computer-Aided Design, 34 (2002).
[18] J.E. FLAHERTY, R.M. LOY, M.S. SHEPHARD, B.K. SZYMANSKI, J.D. TERESCO, AND L.H. ZIANTZ, *Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 139–152.

[19] 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 (1988), pp. 908–926.
[20] A. LOGG, *Multi-adaptive Galerkin methods for ODEs I*, SIAM J. Sci. Comput., 24 (2003), pp. 1879–1902.
[21] ———, *Multi-adaptive Galerkin methods for ODEs II: Implementation and applications*, SIAM J. Sci. Comput., 25 (2003), pp. 1119–1141.
[22] ———, *Multi-adaptive Galerkin methods for ODEs III: Existence and stability*, Submitted to SIAM J. Numer. Anal., (2004).
[23] ———, *Multi-adaptive Galerkin methods for ODEs IV: A priori error estimates*, Submitted to SIAM J. Numer. Anal., (2004).
[24] J. MAKINO AND S. AARSETH, *On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems*, Publ. Astron. Soc. Japan, 44 (1992), pp. 141–151.
[25] S. OSHER AND R. SANDERS, *Numerical approximations to nonlinear conservation laws with locally varying time and space grids*, Math. Comp., 41 (1983), pp. 321–336.
[26] S.G. ALEXANDER AND C.B. AGNOR, *n-body simulations of late stage planetary formation with a simple fragmentation model*, ICARUS, 132 (1998), pp. 113–124.
[27] L. SHAMPINE, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, 1994.

COMPUTATIONAL MODELING OF DYNAMICAL SYSTEMS

JOHAN JANSSON, CLAES JOHNSON, AND ANDERS LOGG

Abstract. In this short note, we discuss the basic approach to computational modeling of dynamical systems. If a dynamical system contains multiple time scales, ranging from very fast to slow, computational solution of the dynamical system can be very costly. By resolving the fast time scales in a short time simulation, a model for the effect of the small time scale variation on large time scales can be determined, making solution possible on a long time interval. This process of computational modeling can be completely automated. Two examples are presented, including a simple model problem oscillating at a time scale of 10^{-9} computed over the time interval $[0, 100]$, and a lattice consisting of large and small point masses.

1. Introduction

We consider a dynamical system 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. We consider a situation where the exact solution u varies on different time scales, ranging from very fast to slow. Typical examples include meteorological models for weather prediction, with fast time scales on the range of seconds and slow time scales on the range of years, protein folding represented by a molecular dynamics model of the form (1.1), with fast time scales on the range of femtoseconds and slow time scales on the range of microseconds, or turbulent flow with a wide range of time scales.

To make computation feasible in a situation where computational resolution of the fast time scales would be prohibitive because of the small time steps required, the given model (1.1) containing the fast time scales needs to

Date: April 22, 2004.

Key words and phrases. Modeling, dynamical system, reduced model, automation.

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

be replaced with a *reduced model* for the variation of the solution u of (1.1) on resolvable time scales. As discussed below, the key step is to correctly model the effect of the variation at the fast time scales on the variation on slow time scales.

The problem of model reduction is very general and various approaches have been taken, see e.g. [8] and [6]. We present below a new approach to model reduction, based on resolving the fast time scales in a short time simulation and determining a model for the effect of the small time scale variation on large time scales. This process of computational modeling can be completely *automated* and the validity of the reduced model can be evaluated a posteriori.

2. A simple model problem

We consider a simple example illustrating the basic aspects: Find $u = (u_1, u_2) : [0, T] \to \mathbb{R}^2$, such that

$$\begin{aligned} \ddot{u}_1 + u_1 - u_2^2/2 = 0 \quad &\text{on } (0, T], \\ \ddot{u}_2 + \kappa u_2 = 0 \quad &\text{on } (0, T], \\ u(0) = (0, 1) \quad \dot{u}(0) &= (0, 0), \end{aligned} \tag{2.1}$$

which models a moving unit point mass M_1 connected through a soft spring to another unit point mass M_2, with M_2 moving along a line perpendicular to the line of motion of M_1, see Figure 1. The second point mass M_2 is connected to a fixed support through a very stiff spring with spring constant $\kappa = 10^{18}$ and oscillates rapidly on a time scale of size $1/\sqrt{\kappa} = 10^{-9}$. The oscillation of M_2 creates a force $\sim u_2^2$ on M_1 proportional to the elongation of the spring connecting M_2 to M_1 (neglecting terms of order u_2^4).

The short time scale of size 10^{-9} requires time steps of size $\sim 10^{-10}$ for full resolution. With $T = 100$, this means a total of $\sim 10^{12}$ time steps for solution of (2.1). However, by replacing (2.1) with a reduced model where the fast time scale has been removed, it is possible to compute the (averaged) solution of (2.1) with time steps of size ~ 0.1 and consequently only a total of 10^3 time steps.

3. Taking averages to obtain the reduced model

Having realized that point-wise resolution of the fast time scales of the exact solution u of (1.1) may sometimes be computationally very expensive or even impossible, we seek instead to compute a time average $\bar{u}$ of u, defined by

$$\bar{u}(t) = \frac{1}{\tau} \int_{-\tau/2}^{\tau/2} u(t + s)\, ds, \quad t \in [\tau/2, T - \tau/2], \tag{3.1}$$

where $\tau > 0$ is the size of the average. The average $\bar{u}$ can be extended to $[0, T]$ in various ways. We consider here a constant extension, i.e., we let $\bar{u}(t) = \bar{u}(\tau/2)$ for $t \in [0, \tau/2)$, and let $\bar{u}(t) = \bar{u}(T - \tau/2)$ for $t \in (T - \tau/2, T]$.

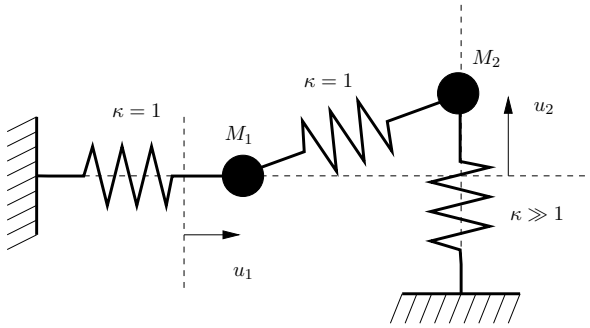

FIGURE 1. A simple mechanical system with large time scale ~ 1 and small time scale $\sim 1/\sqrt{\kappa}$.

We now seek a dynamical system satisfied by the average $\bar{u}$ by taking the average of (1.1). We obtain

$$\dot{\bar{u}}(t) = \bar{\dot{u}}(t) = \overline{f(u,\cdot)}(t) = f(\bar{u}(t),t) + (\overline{f(u,\cdot)}(t) - f(\bar{u}(t),t)),$$

or

$$\dot{\bar{u}}(t) = f(\bar{u}(t),t) + \bar{g}(u,t), \tag{3.2}$$

where the *variance* $\bar{g}(u,t) = \overline{f(u,\cdot)}(t) - f(\bar{u}(t),t)$ accounts for the effect of small scales on time scales larger than τ. (Note that we may extend (3.2) to $(0,T]$ by defining $\bar{g}(u,t) = -f(\bar{u}(t),t)$ on $(0,\tau/2] \cup (T-\tau/2,T]$.)

We now seek to model the variance $\bar{g}(u,t)$ in the form $\bar{g}(u,t) \approx \tilde{g}(\bar{u}(t),t)$ and replace (3.2) and thus (1.1) by

$$\begin{aligned} \dot{\tilde{u}}(t) &= f(\tilde{u}(t),t) + \tilde{g}(\tilde{u}(t),t), \quad t \in (0,T], \\ \tilde{u}(0) &= \bar{u}_0, \end{aligned} \tag{3.3}$$

where $\bar{u}_0 = \bar{u}(0) = \bar{u}(\tau/2)$. We refer to this system as the *reduced model* with *subgrid model* $\tilde{g}$ corresponding to (1.1).

To summarize, if the solution u of the full dynamical system (1.1) is computationally unresolvable, we aim at computing the average $\bar{u}$ of u. However, since the variance $\bar{g}$ in the averaged dynamical system (3.2) is unknown, we need to solve the reduced model (3.3) for $\tilde{u} \approx \bar{u}$ with an approximate subgrid model $\tilde{g} \approx \bar{g}$. Solving the reduced model (3.3) using e.g. a Galerkin finite element method, we obtain an approximate solution $U \approx \tilde{u} \approx \bar{u}$. Note that we may not expect U to be close to u point-wise in time, while we hope that U is close to $\bar{u}$ point-wise.

4. Modeling the variance

There are two basic approaches to the modeling of the variance $\bar{g}(u,t)$ in the form $\tilde{g}(\tilde{u}(t),t)$; (i) scale-extrapolation or (ii) local resolution. In (i), a sequence of solutions is computed with increasingly fine resolution, but without resolving the fastest time scales. A model for the effects of the fast unresolvable scales is then determined by extrapolation from the sequence of computed solutions, see e.g. [3]. In (ii), the approach followed below, the solution u is computed accurately over a short time period, resolving the fastest time scales. The reduced model is then obtained by computing the variance

$$\bar{g}(u,t) = \overline{f(u,\cdot)}(t) - f(\bar{u}(t),t) \tag{4.1}$$

and then determining $\tilde{g}$ for the remainder of the time interval such that $\tilde{g}(\tilde{u}(t),t) \approx \bar{g}(u,t)$.

For the simple model problem (2.1), which we can write in the form (1.1) by introducing the two new variables $u_3 = \dot{u}_1$ and $u_4 = \dot{u}_2$ with

$$f(u,\cdot) = (u_3, u_4, -u_1 + u_2^2/2, -\kappa u_2),$$

we note that $\bar{u}_2 \approx 0$ (for $\sqrt{\kappa}\tau$ large) while $\overline{u_2^2} \approx 1/2$. By the linearity of f_1, f_2, and f_4, the (approximate) reduced model takes the form

$$\begin{aligned} \ddot{\tilde{u}}_1 + \tilde{u}_1 - 1/4 = 0 \quad &\text{on } (0,T], \\ \ddot{\tilde{u}}_2 + \kappa \tilde{u}_2 = 0 \quad &\text{on } (0,T], \\ \tilde{u}(0) = (0,0), \quad \dot{\tilde{u}}(0) &= (0,0), \end{aligned} \tag{4.2}$$

with solution $\tilde{u}(t) = (\frac{1}{4}(1-\cos t), 0)$.

In general, the reduced model is constructed with subgrid model $\tilde{g}$ varying on resolvable time scales. In the simplest case, it is enough to model $\tilde{g}$ with a constant and repeatedly checking the validity of the model by comparing the reduced model (3.3) with the full model (1.1) in a short time simulation. Another possibility is to use a piecewise polynomial representation for the subgrid model $\tilde{g}$.

5. Solving the reduced system

Although the presence of small scales has been decreased in the reduced system (3.3), the small scale variation may still be present. This is not evident in the reduced system (4.2) for the simple model problem (2.1), where we made the approximation $\tilde{u}_2(0) = 0$. In practice, however, we compute $\tilde{u}_2(0) = \frac{1}{\tau}\int_0^\tau u_2(t)\,dt = \frac{1}{\tau}\int_0^\tau \cos(\sqrt{\kappa}t)\,dt \sim 1/(\sqrt{\kappa}\tau)$ and so $\tilde{u}_2$ oscillates at the fast time scale $1/\sqrt{\kappa}$ with amplitude $1/(\sqrt{\kappa}\tau)$.

To remove these oscillations, the reduced system needs to be stabilized by introducing damping of high frequencies. Following the general approach presented in [5], a least squares stabilization is added in the Galerkin formulation of the reduced system (3.3) in the form of a modified test function.

As a result, damping is introduced for high frequencies without affecting low frequencies.

Alternatively, components such as u_2 in (4.2) may be *inactivated*, corresponding to a subgrid model of the form $\tilde{g}_2(\tilde{u},\cdot) = -f_2(\tilde{u},\cdot)$. We take this simple approach for the example problems presented below.

6. Error analysis

The validity of a proposed subgrid model may be checked a posteriori. To analyze the modeling error introduced by approximating the variance $\bar{g}$ with the subgrid model $\tilde{g}$, we introduce the *dual problem*

$$\begin{aligned} -\dot{\phi}(t) &= J(\bar{u},U,t)^\top \phi(t), \quad t \in [0,T), \\ \phi(T) &= \psi, \end{aligned} \tag{6.1}$$

where J denotes the Jacobian of the right-hand side of the dynamical system (1.1) evaluated at a mean value of the average $\bar{u}$ and the computed numerical (finite element) solution $U \approx \tilde{u}$ of the reduced system (3.3),

$$J(\bar{u},U,t) = \int_0^1 \frac{\partial f}{\partial u}(s\bar{u}(t) + (1-s)U(t),t)\,ds, \tag{6.2}$$

and where ψ is initial data for the backward dual problem.

To estimate the error $\bar{e} = U - \bar{u}$ at final time, we note that $\bar{e}(0) = 0$ and $\dot{\phi} + J(\bar{u},U,\cdot)^\top \phi = 0$, and write

$$\begin{aligned} (\bar{e}(T),\psi) &= (\bar{e}(T),\psi) - \int_0^T (\dot{\phi} + J(\bar{u},U,\cdot)^\top \phi, \bar{e})\,dt \\ &= \int_0^T (\phi, \dot{\bar{e}} - J\bar{e})\,dt = \int_0^T (\phi, \dot{U} - \dot{\bar{u}} - f(U,\cdot) + f(\bar{u},\cdot))\,dt \\ &= \int_0^T (\phi, \dot{U} - f(U,\cdot) - \tilde{g}(U,\cdot))\,dt + \int_0^T (\phi, \tilde{g}(U,\cdot) - \bar{g}(u,\cdot))\,dt \\ &= \int_0^T (\phi, \tilde{R}(U,\cdot))\,dt + \int_0^T (\phi, \tilde{g}(U,\cdot) - \bar{g}(u,\cdot))\,dt. \end{aligned}$$

The first term, $\int_0^T (\phi, \tilde{R}(U,\cdot))\,dt$, in this *error representation* corresponds to the *discretization error* $U - \tilde{u}$ for the numerical solution of (3.3). If a Galerkin finite element method is used (see [1, 2]), the *Galerkin orthogonality* expressing the orthogonality of the residual $\tilde{R}(U,\cdot) = \dot{U} - f(U,\cdot) - \tilde{g}(U,\cdot)$ to a space of test functions can be used to subtract a test space interpolant $\pi\phi$ of the dual solution ϕ. In the simplest case of the cG(1) method for a partition of the interval $(0,T]$ into M subintervals $I_j = (t_{j-1},t_j]$, each of length $k_j = t_j - t_{j-1}$, we subtract a piecewise constant interpolant to obtain

$$\begin{aligned} \int_0^T (\phi, \tilde{R}(U,\cdot))\,dt &= \int_0^T (\phi - \pi\phi, \tilde{R}(U,\cdot))\,dt \\ &\le \sum_{j=1}^M k_j \max_{I_j} \|\tilde{R}(U,\cdot)\|_{l_2} \int_{I_j} \|\dot{\phi}\|_{l_2}\,dt \\ &\le S^{[1]}(T) \max_{[0,T]} \|k\tilde{R}(U,\cdot)\|_{l_2}, \end{aligned}$$

where the *stability factor* $S^{[1]}(T) = \int_0^T \|\dot{\phi}\|_{l_2}\,dt$ measures the sensitivity to discretization errors for the given output quantity $(\bar{e}(T),\psi)$.

The second term, $\int_0^T (\phi, \tilde{g}(U,\cdot) - \bar{g}(u,\cdot))\,dt$, in the error representation corresponds to the *modeling error* $\tilde{u} - \bar{u}$. The sensitivity to modeling errors is measured by the stability factor $S^{[0]}(T) = \int_0^T \|\phi\|_{l_2}\,dt$. We notice in particular that if the stability factor $S^{[0]}(T)$ is of moderate size, a reduced model of the form (3.3) for $\tilde{u} \approx \bar{u}$ may be constructed.

We thus obtain the error estimate

$$|(\bar{e}(T),\psi)| \le S^{[1]}(T) \max_{[0,T]} \|k\tilde{R}(U,\cdot)\|_{l_2} + S^{[0]}(T) \max_{[0,T]} \|\tilde{g}(U,\cdot) - \bar{g}(u,\cdot)\|_{l_2}, \tag{6.3}$$

including both discretization and modeling errors. The initial data ψ for the dual problem (6.1) is chosen to reflect the desired output quantity, e.g. $\psi = (1,0,\ldots,0)$ to measure the error in the first component of U.

To estimate the modeling error, we need to estimate the quantity $\tilde{g} - \bar{g}$. This estimate is obtained by repeatedly solving the full dynamical system (1.1) at a number of control points and comparing the subgrid model $\tilde{g}$ with the computed variance $\bar{g}$. As initial data for the full system at a control point, we take the computed solution $U \approx \bar{u}$ at the control point and add a perturbation of appropriate size, with the size of the perturbation chosen to reflect the initial oscillation at the fastest time scale.

7. Numerical results

We present numerical results for two model problems, including the simple model problem (2.1), computed with DOLFIN version 0.4.10 [4]. With the option *automatic modeling* set, DOLFIN automatically creates the reduced model (3.3) for a given dynamical system of the form (1.1) by resolving the full system in a short time simulation and then determining a constant subgrid model $\bar{g}$. Components with constant average, such as u_2 in (2.1), are automatically marked as inactive and are kept constant throughout the simulation. The automatic modeling implemented in DOLFIN is rudimentary and many improvements are possible, but it represents a first attempt at the automation of modeling, following the directions for the *automation of computational mathematical modeling* presented in [7].

7.1. **The simple model problem.** The solution for the two components of the simple model problem (2.1) is shown in Figure 2 for $\kappa = 10^{18}$ and $\tau = 10^{-7}$. The value of the subgrid model $\bar{g}_1$ is automatically determined to $0.2495 \approx 1/4$.

7.2. **A lattice with internal vibrations.** The second example is a lattice consisting of a set of p^2 large and $(p-1)^2$ small point masses connected by springs of equal stiffness $\kappa = 1$, as shown in Figure 3 and Figure 4. Each large point mass is of size $M = 100$ and each small point mass is of size $m = 10^{-12}$, giving a large time scale of size ~ 10 and a small time scale of size $\sim 10^{-6}$.

The fast oscillations of the small point masses make the initially stationary structure of large point masses contract. Without resolving the fast time scales and ignoring the subgrid model, the distance D between the lower left large point mass at $x = (0, 0)$ and the upper right large point mass at $x = (1, 1)$ remains constant, $D = \sqrt{2}$. In Figure 5, we show the computed solution with $\tau = 10^{-4}$, which manages to correctly capture the oscillation in the diameter D of the lattice as a consequence of the internal vibrations at time scale 10^{-6}.

With a constant subgrid model $\bar{g}$ as in the example, the reduced model stays accurate until the configuration of the lattice has changed sufficiently. When the change becomes too large, the reduced model can no longer give an accurate representation of the full system, as shown in Figure 6. At this point, the reduced model needs to be reconstructed in a new short time simulation.

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] ——, *Computational Differential Equations*, Cambridge University Press, 1996.

[3] J. Hoffman, *Computational Modeling of Complex Flows*, PhD thesis, Chalmers University of Technology, 2002.

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

[5] J. Hoffman and C. Johnson, *Computability and adaptivity in CFD*, to appear in Encyclopedia of Computational Mechanics, (2004).

[6] H.-O. Kreiss, *Problems with different time scales*, Acta Numerica, 1 (1991).

[7] A. Logg, *Automation of Computational Mathematical Modeling*, PhD thesis, Chalmers University of Technology, 2004.

[8] A. Ruhe and D. Skoogh, *Rational Krylov algorithms for eigenvalue computation and model reduction*, in Applied Parallel Computing — Large Scale Scientific and Industrial Problems, B. Kågström, J. Dongarra, E. Elmroth, and J. Waśniewski, eds., Lecture Notes in Computer Science, No. 1541, 1998, pp. 491–502.

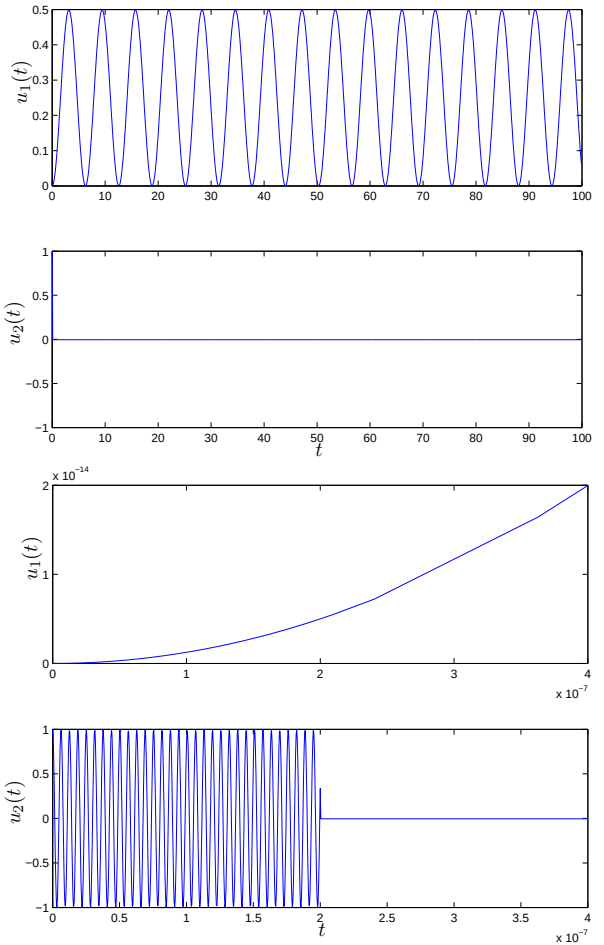

Figure 2. The solution of the simple model problem (2.1) on $[0, 100]$ (above) and on $[0, 4 \cdot 10^{-7}]$ (below). The automatic modeling is activated at time $t = 2\tau = 2 \cdot 10^{-7}$.

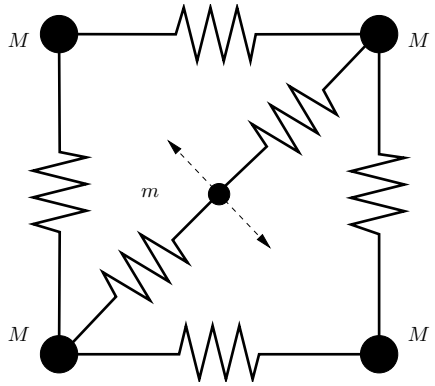

FIGURE 3. Detail of the lattice. The arrows indicate the direction of vibration perpendicular to the springs connecting the small mass to the large masses.

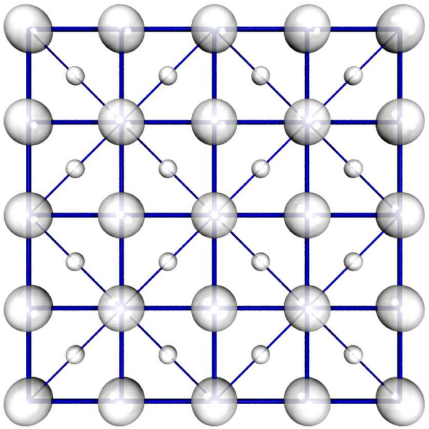

FIGURE 4. Lattice consisting of p^2 large masses and $(p-1)^2$ small masses.

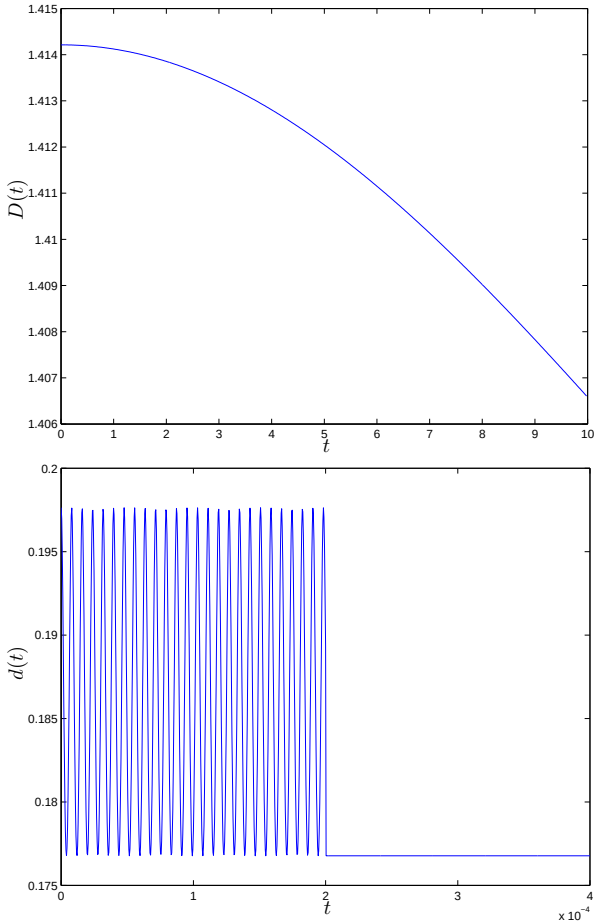

FIGURE 5. Distance D between the lower left large mass and the upper right large mass (above) and the distance d between the lower left large mass and the lower left small mass (below) as function of time on $[0, 10]$ and on $[0, 4 \cdot 10^{-4}]$, respectively.

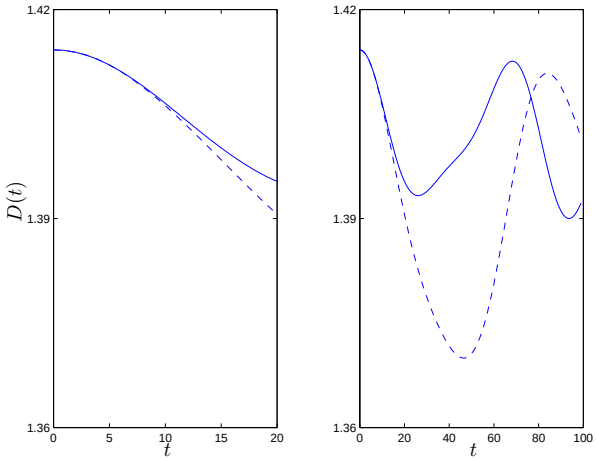

FIGURE 6. The diameter D of the lattice as function of time on $[0, 20]$ (left) and on $[0, 100]$ (right) for $m = 10^{-4}$ and $\tau = 1$. The solid line represents the diameter for the solution of the reduced system (3.3) and the dashed line represents the solution of the full system (1.1).