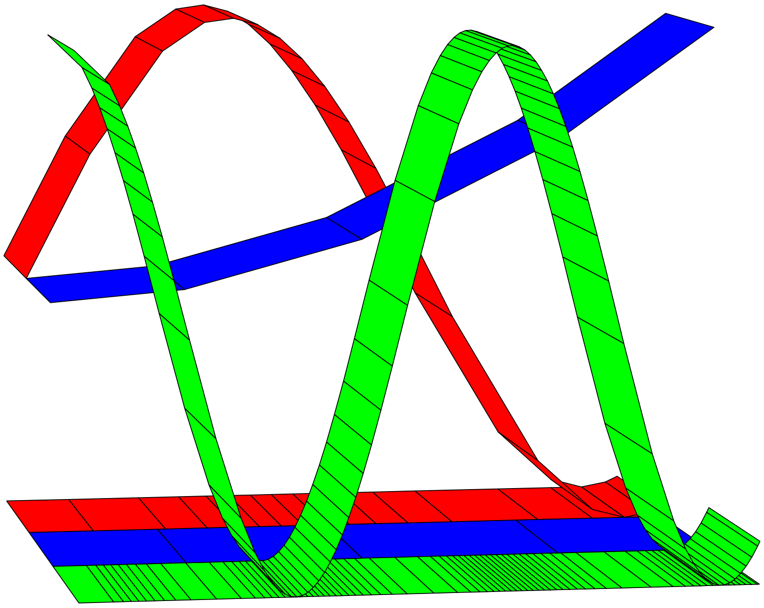
CHALMERS

FINITE ELEMENT CENTER



PREPRINT 2001–09

Multi-Adaptive Galerkin Methods for ODEs I: Theory & Algorithms

Anders Logg

Chalmers Finite Element Center
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg Sweden 2001

CHALMERS FINITE ELEMENT CENTER

Preprint 2001–09

Multi-Adaptive Galerkin Methods for ODEs I: Theory & Algorithms

Anders Logg

CHALMERS

Chalmers Finite Element Center
Chalmers University of Technology
SE–412 96 Göteborg Sweden
Göteborg, April 2001

Multi-Adaptive Galerkin Methods for ODEs I:
Theory & Algorithms
Anders Logg
NO 2001–09
ISSN 1404–4382

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

Printed in Sweden
Chalmers University of Technology
Göteborg, Sweden 2001

MULTI-ADAPTIVE GALERKIN METHODS FOR ODES I: THEORY & ALGORITHMS

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, a priori and a posteriori error estimates, adaptive algorithms for time-stepping and global error control, iterative solution methods for the discrete equations, together with basic features of the implementation *Tanganyika*. In the accompanying paper [31], we present numerical results for a variety of applications, including chemical reaction problems, the Lorenz system and the Solar System.

1. INTRODUCTION

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

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

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 use the term *multi-adaptivity* to describe methods with individual time-stepping for the different components $u_i(t)$ of the solution vector $u(t) = (u_i(t))$, including (i) time-step length, (ii) order and (iii) quadrature, all chosen adaptively in a computational feed-back process. In the companion paper [31], 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.

Date: April 27, 2001.

Key words and phrases. Multi-adaptivity, individual time-steps, local time-steps, ODE, continuous Galerkin, discontinuous Galerkin, global error control, adaptivity, mcgq, mdgq.

Anders Logg, Department of Mathematics, Chalmers University of Technology, SE–412 96 Göteborg, Sweden. *E-mail address*: `logg@math.chalmers.se`.

Surprisingly, individual time-stepping for ODEs has received little attention in the large literature on numerical methods for ODEs, see e.g. [5, 22, 23, 4, 39]. For specific applications, such as the n-body problem, methods with individual time-stepping have been used, see e.g. the paper [7] by Dawson and Kirby and the references therein, [33, 2, 6], or [28], but a general methodology has been lacking. Our aim is to fill this gap.

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 [8, 17, 25, 19, 18, 20]. The techniques for error analysis used in these references, developed by Johnson and coworkers, see e.g. [12, 13, 11, 14, 15, 16], and [9] in particular, naturally carries over to the multi-adaptive methods.

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, $i = 1, \dots, N$, a partition of the time-interval $(0, T]$ into M_i subintervals, $I_{ij} = (t_{i,j-1}, t_{ij}]$, $j = 1, \dots, M_i$, 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 synchronized time-levels common to all components, as illustrated in Figure 1.

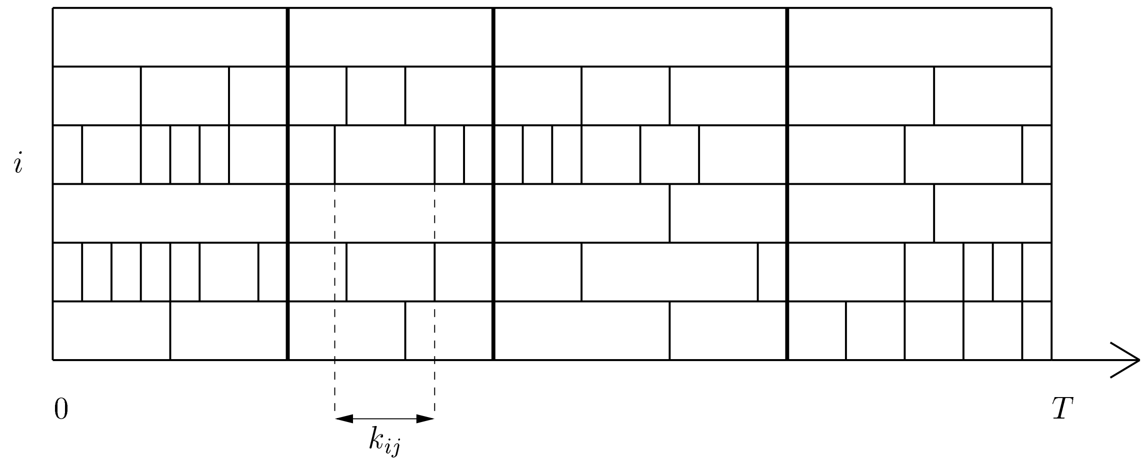


Figure 1: Individual time-discretizations for different components.

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 $\mathrm{TOL} > 0$, 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 [34]).

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

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

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 e.g. the form

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

where the $\{S_i(T)\}_{i=1}^N$ 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*

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

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 below, and is publicly (GPL) available for Linux/Unix [32].

2.5. **Applications.** We have applied mcG(q) and mdG(q) to a variety of problems to illustrate their potential, see [31] for a detailed presentation. (See also [30] and [29].) We discuss here briefly two key problems, *the Lorenz system* and *the Solar System*, posing fundamental questions concerning limits of computability and predictability.

For the Lorenz system, which has exponentially increasing stability factors, with $S_i(50) \sim 10^{16}$, we compare cG(q) methods with $1 \leq q \leq 15$, using double precision. We find that cG(15) computes accurately on $[0, 48]$, while cG(1) reaches only $T = 25$. Using MATLAB's `ode45`, we reach $T = 39$. In this case the machine precision sets the limit and a high-order method with large time-steps is the winner.

For the Solar System, including the Sun, the Moon, and the nine planets, we compute limits of predictability depending on errors in given data, such as initial position, velocity, and the gravitational constant. We find that, assuming initial data is known to five digits, the position of the Moon can be accurately predicted only a few years ahead, and that the position of Mercury can be accurately predicted on the order of 500 years. We also study special events, such as a large comet passing close to Earth, which exhibits interesting stability features, among other things.

In both cases we solve the dual problem and collect extensive information on the stability features of the systems.

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 thesis, an essential difference is thus the technique to estimate the global error: 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. [34] 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 in this thesis 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 [3], 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: Find $U \in V$ with $U(0) = u_0$, such that

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

where

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

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 $i = 1,\dots,N$, $j = 1,\dots,M_i$, find $U_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij})$ with $U_i(t_{i,j-1})$ given, such that

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

We notice the presence of the vector $U(t) = (U_1(t),\dots,U_N(t))$ in the local problem for U_i 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).

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

$$\xi_{ijm} = \xi_{ij0} + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt,\ m = 1,\dots,q_{ij}, \tag{4.4}$$

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

$$\xi_{ijm} = \xi_{ij0} + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]}\ f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})), \tau_{ij}^{-1}(s_n^{[q_{ij}]})),\ m = 1,\dots,q_{ij}, \tag{4.5}$$

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

4.2. **The** mdG(q) **method.** The mdG(q) method in local form, corresponding to (4.3), 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

$$[U_i]_{i,j-1} v(t_{i,j-1}) + \int_{I_{ij}} \dot{U}_i v \; dt = \int_{I_{ij}} f_i(U, \cdot) v \; dt \;\forall v \in \mathcal{P}^{q_{ij}}(I_{ij}), \tag{4.6}$$

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

$$V = W = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}), j = 1, \ldots, M_i, \; i = 1, \ldots, N\}. \tag{4.7}$$

Making an Ansatz for the solution in terms of some nodal basis, we get, as for the continuous method, the following explicit version of (4.6) on every local interval:

$$\xi_{ijm} = \xi_{ij0}^- + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t)) \; f_i(U(t), t) \; dt, \; m = 0, \ldots, q_{ij}, \tag{4.8}$$

or, applying nodal quadrature,

$$\xi_{ijm} = \xi_{ij0}^- + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} \; f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})), \tau_{ij}^{-1}(s_n^{[q_{ij}]})), \; m = 0, \ldots, q_{ij}, \tag{4.9}$$

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 (4.5) and (4.9), 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 q:th-order Legendre polynomial on the interval; and for the mdG(q) method, we choose the *Radau quadrature points*, i.e. the zeros of $P_q + P_{q+1}$ on the interval (with time reversed so as to include the *right* end-point). See the Appendix for a detailed discussion on this subject. The resulting methods are related to the implicit Runge-Kutta methods referred to as Lobatto and Radau methods, see e.g. [4].

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. We here state the corresponding properties for the mcG(q) and mdG(q) methods, and give the proofs in the Appendix.

Assuming that the distances between adjacent synchronized time-levels, i.e. the lengths of the time-slabs, are small enough in terms of the right-hand side f of (1.1), we have the following bound for the error $e(t) = U(t) - u(t)$ at final time for the mcG(q) method:

$$\|e(T)\| \leq \int_0^T (k^{2q}, w)\, dt, \tag{5.1}$$

where the weight $w = (w_i) = w(t) \geq 0$ depends on f and u (and derivatives of u), but does not depend on the time-steps $k = (k_{ij})$. The notation of (5.1) is to be interpreted component-wise, i.e. $(k^{2q})_i = k_i^{2q_i} = k_{ij}^{2q_{ij}}$ on I_{ij}. The mcG(q) method is thus of order $2q$.

With the same assumptions as above, we have the following bound for the mdG(q) method:

$$\|e(T)\| \leq \int_0^T (k^{2q+1}, w)\, dt, \tag{5.2}$$

and thus the mdG(q) method is of order $2q+1$.

5.2. Energy conservation for mcG(q).

The standard cG(q) method is energy-conserving. 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

$$\ddot{x} = -\nabla_x P(x), \tag{5.3}$$

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 (5.3) as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ F(u) \end{bmatrix} = \begin{bmatrix} f_u(v) \\ f_v(u) \end{bmatrix} = f(u, v). \tag{5.4}$$

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

$$E(t) = K(t) + P(x(t)), \tag{5.5}$$

with

$$K(t) = \frac{1}{2}\|\dot{x}(t)\|^2 = \frac{1}{2}\|v(t)\|^2. \tag{5.6}$$

Multiplying (5.3) with $\dot{x}$ it is easy to see that energy is conserved for the continuous problem, i.e. $E(t) = E(0)\ \forall t \in [0,T]$. We now prove the corresponding property for the discrete mcG(q) solution of (5.4).

Theorem 5.1. *The multi-adaptive continuous Galerkin method conserves energy in the following sense: Let (U,V) be the* mcG(q) *solution to (5.4). 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*

$$K(\bar{t}) + P(\bar{t}) = K(0) + P(0), \tag{5.7}$$

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

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

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

Remark 5.1. *Energy conservation requires exact integration of the right-hand side f, or at least that $\int_0^t (\dot{U},\dot{V})\ dt + (P(t) - P(0)) = 0$. This might not hold if we do not have good enough quadrature, or take special care of this, e.g. as proposed by Hansbo in* [24].

5.3. **Monotonicity.** We shall prove that the mdG(q) method is B-stable (see [4]).

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

$$(f(u,\cdot) - f(v,\cdot), u - v) \leq 0\ \forall u,v \in \mathbb{R}^N, \tag{5.8}$$

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

$$\|U(\bar{t}^-) - V(\bar{t}^-)\| \leq \|U(0^-) - V(0^-)\|. \tag{5.9}$$

Proof. Choosing the test function as $v = W = U - V$ in (4.6) for U and V, summing over the local intervals and subtracting the two equations, we have

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

Noting that

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

we get

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

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 start by defining the dual linearized problem, and 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. We return to the discrete dual for the a priori error analysis in the Appendix.

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 note that

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

The continuous dual problem is now the following system of ODEs:

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

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.

6.2. **Error representation.** The basis for the error analysis is the following error representation, expressing the error of any approximate solution $U(t)$ in terms of the residual via the dual solution $\varphi(t)$.

Theorem 6.1. *Let U be an approximate and u the exact solution 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 inner of the partitions $\cup_j I_{ij}$, $i = 1,\dots,N$. Assume also that U is right-continuous at T. Then the error $e = U - u$ satisfies*

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

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

$$\begin{aligned}
(e(T),\varphi_T) + \textstyle\int_0^T (e,g)\ dt &= \textstyle\sum_{ij}\left[\int_{I_{ij}} (\dot{e}_i + f_i(u,\cdot) - f_i(U,\cdot))\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1})\right] \\
&= \textstyle\sum_{ij}\left[\int_{I_{ij}} (\dot{U}_i - f_i(U,\cdot))\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1})\right] \\
&= \textstyle\sum_{ij}\left[\int_{I_{ij}} R_i(U,\cdot)\varphi_i\ dt + [U_i]_{i,j-1}\varphi_i(t_{i,j-1})\right],
\end{aligned}$$

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

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

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

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

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 orthogonality. We first prove the following interpolation estimate.

Lemma 6.1. *If $f \in C^{q+1}([a,b])$ then there is a constant C_q, depending only on q, such that*

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

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

Proof. Using Taylors formula with the remainder in integral form, we have

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

□

We can now prove a posteriori error estimates for the mcG(q) and mdG(q) methods. We denote the (absolute value of the) left-hand side of (6.4) by $|||e|||$. 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.

Theorem 6.2. *The* mcG(q) *method satisfies the following estimates:*

$$|||e||| = E_0 \le E_1 \le E_2 \le E_3 \le E_4, \tag{6.8}$$

and

$$|||e||| \le E_2 \le E_5, \tag{6.9}$$

where

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

with $r_i(t) = r_{ij}$ and $k_i(t) = k_{ij}$ for $t \in I_{ij}$, and

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

and where $\pi_k\varphi$ *is any test space approximation of the dual solution* φ*. Expressions such as* k^qR *are defined component-wise, i.e.* $(k^qR)_i = k_i(t)^{q_i(t)}R_i(t) = k_{ij}^{q_{ij}}R_i(t)$ *for* $t \in I_{ij}$*.*

Proof. Using the error representation and the Galerkin orthogonality, noting that the jump terms disappear since U is continuous, we have

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

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

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

Choosing $\pi_k\varphi_i$ as in Lemma 6.1 on every interval I_{ij}, we have

$$\begin{array}{rcl} E_1 & \leq & \sum_{ij} C_{q_{ij}-1}k_{ij}^{q_{ij}} \int_{I_{ij}} |R_i(U)|\ dt\ \frac{1}{k_{ij}}\int_{I_{ij}} |\varphi_i^{(q_{ij})}|\ dt = \sum_{ij} C_{q_{ij}-1}k_{ij}^{q_{ij}+1}\ r_{ij}s_{ij}^{[q_{ij}]} = E_2 \\ & \leq & \sum_{i=1}^N\sum_{j=1}^{M_i}\int_{I_{ij}} |\varphi_i^{(q_i)}|\ dt \max_{[0,T]}\left\{C_{q_i-1}k_i^{q_i}r_i\right\} = \sum_{i=1}^N S_i^{[q_i]}\max_{[0,T]}\left\{C_{q_i-1}k_i^{q_i}r_i\right\} \\ & = & E_3 \leq \max_{i,[0,T]}\left\{C_{q_i-1}k_i^{q_i}r_i\right\}\sum_{i=1}^N\int_0^T |\varphi_i^{(q_i)}|\ dt \\ & \leq & \max_{i,[0,T]}\left\{C_{q_i-1}k_i^{q_i}r_i\right\}\sqrt{N}\int_0^T \|\varphi^{(q)}\|\ dt = \max_{i,[0,T]}\left\{C_{q_i-1}k_i^{q_i}r_i\right\}\sqrt{N}S^{[q],1} = E_4. \end{array}$$

As an alternative, continuing from E_2, we have

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

Noting now that s is the L^2-projection of $|\varphi^{(q)}|$ onto the piecewise constants on the partition, we have

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

so that

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

completing the proof. ∎

Theorem 6.3. *The* mdG(q) *method satisfies the following estimates:*

$$|||e||| = E_0 \leq E_1 \leq E_2 \leq E_3 \leq E_4, \tag{6.12}$$

and

$$|||e||| \leq E_2 \leq E_5, \tag{6.13}$$

where

(6.14)

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

with

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

and we otherwise use the notation of Theorem 6.2.

Proof. As in the proof for the continuous method, we use the Galerkin orthogonality to get

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

By Lemma 6.1 we obtain

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

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, we don't have to choose $\pi_k\varphi$ as in Lemma 6.1. This is only a convenient way to obtain the interpolation constant.*

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. Different ways of measuring the residual give different constants.*

6.4. **Computational errors.** The error estimates of Theorems 6.2 and 6.3 are based on the Galerkin orthogonalities (4.3) and (4.6). If the corresponding discrete equations are not solved exactly, the residual will not be orthogonal to the test space, and there will be an additional contribution to the total error. For the mcG(q) method, this extra contribution may be estimated as follows:

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

where

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

is a stability factor, and we approximated the *discrete* or *computational* residual by

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

assuming that φ varies slowly on each sub-interval. More precise estimates may be used if needed.

For the mdG(q) method, the situation is similar with the computational residual now approximated by

$$
\mathcal{R}^{\mathcal{C}}_{ij} = \frac{1}{k_{ij}}\left([U_i]_{i,j-1} + \int_{I_{ij}} R_i(U)dt\right) = \frac{1}{k_{ij}}\left((\xi_{ijq}-\xi_{ij0}^-) - \int_{I_{ij}} f_i(U,\cdot) \; dt\right).
\tag{6.19}
$$

Thus, to estimate the computational error, we may 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

$$
\begin{array}{rcl}
|||e||| & = & \int_0^T (R,\varphi) \; dt \\
& = & \int_0^T (R,\varphi-\pi_k\varphi) \; dt + \int_0^T (R,\pi_k\varphi) \; dt \\
& = & \int_0^T (R,\varphi-\pi_k\varphi) \; dt + \tilde{\int}_0^T (R,\pi_k\varphi) \; dt + \left[\int_0^T (R,\pi_k\varphi) \; dt - \tilde{\int}_0^T (R,\pi_k\varphi) \; dt\right] \\
& = & \int_0^T (R,\varphi-\pi_k\varphi) \; dt + \tilde{\int}_0^T (R,\pi_k\varphi) \; dt + \left(\tilde{\int}_0^T - \int_0^T\right)(f(U,\cdot),\pi_k\varphi) \; dt,
\end{array}
\tag{6.20}
$$

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 6.2 and the second term is the computational error discussed previously (where we evaluate $\tilde{\int}$ rather than $\int$). 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

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

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

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

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 $\mathcal{R}_{ij}^{\mathcal{Q}} \propto k_{ij}^{2q_{ij}}$. If instead of using the standard quadrature rule over the interval with quadrature residual $\mathcal{R}_{ij}^{\mathcal{Q}_0}$, 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

$$\mathcal{R}^{\mathcal{Q}_m} = \frac{1}{k}C2^m(k/2^m)^{2q+1} = 2^{m(-2q)}Ck^{2q} = 2^{-2q}2^{(m-1)(-2q)}Ck^{2q} = 2^{-2q}\mathcal{R}^{\mathcal{Q}_{m-1}}, \tag{6.23}$$

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

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

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

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

6.6. **Evaluating E_G.** We now present an approach to estimating the quantity $\int_0^T(R,\varphi - \pi_k\varphi)\ dt$ 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. Starting with

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

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 $R_i(\varphi_i - \pi_k \varphi_i) \geq 0$ (or ≤ 0) on every local interval I_{ij}. With such a choice of the interpolant, we would have

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

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

Lemma 6.2. *If, for $i = 1, \ldots, N$, $f_i = f_i(U_i(t), t)$ 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 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 is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$. Since the conditions assumed in the statement of the lemma guarantees 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). $\square$

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

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

$$\int_{I_{ij}} \tilde{R}_i v\ dt = \int_{I_{ij}} R_i v\ dt = 0$$

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

Lemma 6.4. *Let P_q be the q:th-order Legendre polynomial on $[-1, 1]$. Then the q:th-order Radau polynomial, $Q_q(x) = (P_q(x) + P_{q+1}(x))/(x + 1)$, has the following property:*

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

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 q:th-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

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

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 6.5. *If, for $i = 1,\ldots,N$, $f_i = f_i(U_i(t),t)$ 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 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 is a polynomial of degree q_{ij} on I_{ij}. By the Galerkin orthogonality, we have

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

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 6.4 that the residual R_i 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 6.6. *Let $\tilde{R}$ be the local L^2-projection of the* mdG(q) *residual R onto the trial space, i.e. $\tilde{R}_i|_{I_{ij}}$ is the $L^2(I_{ij})$-projection onto $\mathcal{P}^{q_{ij}}(I_{ij})$ of $R_i|_{I_{ij}}$, $j = 1,\ldots,M_i$, $i = 1,\ldots,N$. Then every $\tilde{R}_i|_{I_{ij}}$ is a Radau polynomial of degree q_{ij}.*

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

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

for any $v(t) = (t - t_{i,j-1})^p$ with $1 \le p \le q$. From Lemma 6.4 it then follows that $\tilde{R}_i$ is a Radau polynomial on I_{ij}. □

We thus know that the mcG(q) residuals are (in some sense) Legendre polynomials on the local intervals, and that the mdG(q) residuals are (in some sense) Radau polynomials. This is illustrated in Figure 2.

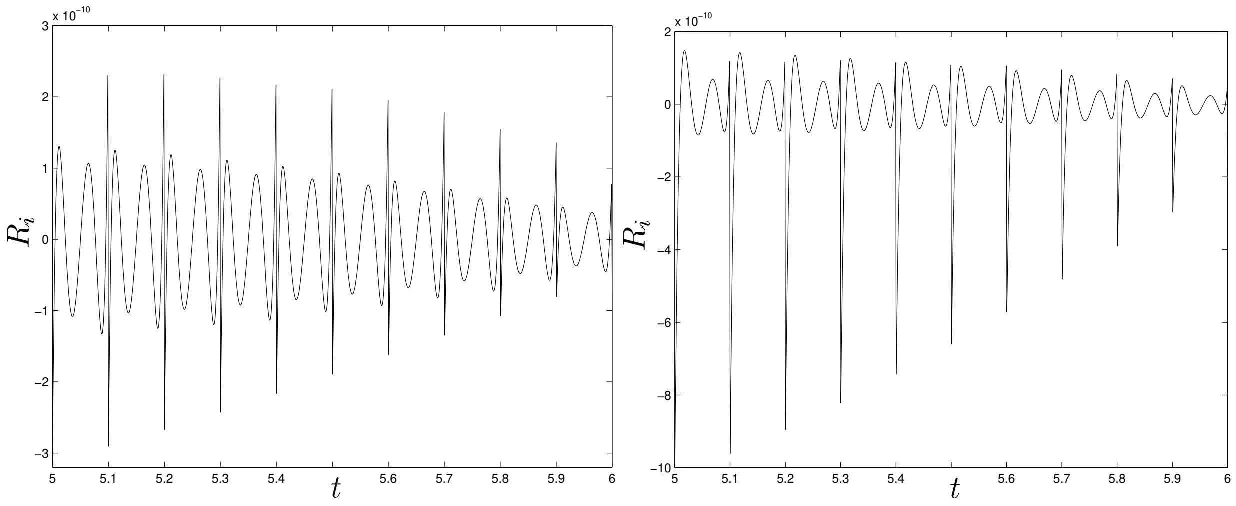


Figure 2: The Legendre-polynomial residual of the mcG(q) method (left) and the Radau-polynomial residual of the mdG(q) method (right), for polynomials of degree five, i.e. methods of order 10 and 11 respectively.

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 q:th-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 $R_i(\varphi_i - \pi_k \varphi_i)$ 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

$$|||e||| = \left| \sum_{ij} \int_{I_{ij}} R_i(\varphi_i - \pi_k \varphi_i)\ dt \right| = \sum_{ij} \alpha_{ij} \int_{I_{ij}} |R_i(\varphi_i - \pi_k \varphi_i)|\ dt, \tag{6.29}$$

with $\alpha_{ij} = \pm 1$, both for 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 $\int_{I_{ij}} R_i(\varphi_i - \pi_k \varphi_i)\ dt$. 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, depending on the order and the method, such that

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

The constants C_q may be computed numerically beforehand.

Finally, note that there are "computational" counterparts also for the estimates of type E_3 in Theorems 6.2 and 6.3, namely

$$\begin{aligned} |||e||| &\leq \textstyle\sum_{ij}\int_{I_{ij}} |R_i||\varphi_i - \pi_k\varphi_i|\, dt = \sum_{ij} C'_{q_{ij}} k_{ij}^{q_{ij}} |R_i(t_{ij}^-)| \int_{I_{ij}} \frac{1}{k_{ij}^{q_{ij}}} |\varphi_i - \pi_k\varphi_i|\, dt \\ &\leq \textstyle\sum_{i=1}^N \tilde{S}_i \max_{j=1,\ldots,M_i} C'_{q_{ij}} k_{ij}^{q_{ij}} |R_i(t_{ij}^-)|, \end{aligned} \tag{6.31}$$

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:

$$|||e||| \leq E_G + E_C + E_Q. \tag{6.32}$$

Choosing estimate E_3 of Theorems 6.2 and 6.3 we then have the following (approximate) error estimate for the mcG(q) method:

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

and for the mdG(q) method we have

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

This estimate includes also numerical round-off errors (included in the computational error) and modelling errors (included in the quadrature error).

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.

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. To compute stability factors 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 the Appendix we prove the following lemma, from which the estimate follows.

Lemma 6.7. *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 as defined in the Appendix. We then have the following estimate:*

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

Theorem 6.4. *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 $S_\Phi(T)$:*

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

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

Proof. By corollary 6.2, we have an expression for the $L^1([0,T],\mathbb{R}^N)$-error of the dual, so that

(6.38)

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

With C defined as above it now follows by Lemma 6.7 that

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

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.* [10].

7. Adaptivity

In this section we describe how to make use of the a posteriori error estimates in an adaptive algorithm.

7.1. **A strategy for adaptive error control.** The goal of the algorithm is to produce an approximate solution within a given tolerance TOL in a given norm $|||\cdot|||$. The adaptive algorithm is based on the a posteriori error estimates presented in the previous section, of the form

$$|||e||| \leq \sum_{i=1}^{N}\sum_{j=1}^{M_i} k_{ij}^{p_{ij}+1} r_{ij} s_{ij}, \tag{7.1}$$

or

$$|||e||| \leq \sum_{i=1}^{N} S_i \max_j k_{ij}^{p_{ij}} r_{ij}, \tag{7.2}$$

where we refer to $\{s_{ij}\}$ as *stability weights* and to $\{S_i\}$ as *stability factors*.

We use (7.2) to determine the individual time-step sequences, i.e. we seek to choose the time-steps as

$$k_{ij} = \left(\frac{\mathrm{TOL}/N}{S_i\ r_{ij}}\right)^{1/p_{ij}}. \tag{7.3}$$

We use (7.1) to evaluate the resulting error at the end of the computation, noting that (7.1) is sharper than (7.2).

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

(1) Solve the primal problem with time-steps based on (7.3).
(2) Solve the dual problem and compute stability factors and stability weights.
(3) Compute an error estimate E based on (7.1).
(4) If $E \leq$ TOL then stop, and if not go back to (1).

Although this seems simple enough, there are some difficulties involved. For one thing, we have to choose the proper data for the dual in order to get a meaningful error estimate. Furthermore, choosing the time-steps based on (7.3) may be difficult, since the residual depends implicitly on the time-step. We now discuss these issues.

7.2. **Regulating the time-step.** To avoid the implicit dependence of k_{ij} for r_{ji} in (7.3), we may try replacing (7.3) by

$$k_{ij} = \left(\frac{\mathrm{TOL}/N}{S_i\ r_{i,j-1}}\right)^{1/p_{ij}}. \tag{7.4}$$

Following this strategy, if the time-step on an interval is small (and thus also the residual), the time-step for the next interval will be large, so that (7.4) 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 get a smoother time-step sequence. For (7.4) 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 what we get from (7.4). This works surprisingly well for many problems.

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

$$S_i k_{ij}^{p_{ij}} r_{ij} = \mathrm{TOL}/N, \tag{7.5}$$

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

$$p_{ij} \log k_{ij} + \log r_{ij} = C_i, \tag{7.6}$$

with maximal time-steps $\{k_{ij}\}$, following work by Söderlind and coworkers [21, 38]. This type of regulator performs a little better than the simple approach described above, provided the parameters of the regulator are well-tuned.

7.3. **Choosing data for the dual.** Different choices of data for the dual problem give different error estimates, as described before. The simplest choice is $g = 0$ and $(\varphi_T)_i = \delta_{in}$ for control of the final time error of the n:th component. For control of the l^2-norm of the error at final time, we take $g = 0$ and $\varphi_T = \tilde{e}(T)/\|\tilde{e}(T)\|$ with an approximation $\tilde{e}$ of the error e.

If the data for the dual problem is not correct, the error estimate may also be incorrect: with $\tilde{e}(T)$ orthogonal to the error, the error representation only gives $0 \leq \mathrm{TOL}$. In practice however, the dual — or at least the stability factors — seem 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.

As another option, we may use the fact that the multi-adaptive algorithm tries to equidistribute the error onto the different components. If the error is indeed the same for every component, we should choose the data for the dual problem as $(\varphi_T)_i = \pm 1/\sqrt{N}$ for all components. In this way we only have to make a guess for the sign of the errors of the different components. In practice, however, we may not be sure that the error is the same for every component, certainly not for the initial computation when we don't know the stability factors.

7.4. **Adaptive quadrature.** To control the quadrature error, we make also the quadrature adaptive. Based on the estimates of the quadrature error, we choose a good enough quadrature rule for every interval. Notice that a better quadrature rule may be needed for component i both if $f_i = f_i(U_i, \cdot)$ is nonlinear, or if f_i depends on some other component with smaller time-steps.

7.5. **Adaptive order, q-adaptivity.** Since we allow for different and varying orders, as well as different time-steps, for the different components, the method is also *q-adaptive* (or p-adaptive). At this stage, 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 possible way to choose time-steps and orders is to try to solve over some short interval with different time-steps and orders, optimizing the choice of time-step and order 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.

8. Solving the Discrete Equations

In this section we discuss how to solve the discrete equations (4.5) and (4.9).

8.1. **A simple strategy.** As discussed in Section 4, the nonlinear discrete algebraic equations for the mcG(q) method to be solved on every local interval I_{ij} take the form

$$\xi_{ijm} = \xi_{ij0} + k_{ij} \sum_{n=0}^{q_{ij}} w_{mn}^{[q_{ij}]} \, f_i(U(\tau_{ij}^{-1}(s_n^{[q_{ij}]})), \tau_{ij}^{-1}(s_n^{[q_{ij}]})), \;\; m = 1, \ldots, q_{ij}. \tag{8.1}$$

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 (8.1), i.e. we make guesses 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 (8.1), repeating the procedure until convergence.

Note now that component $U_i(t)$ is coupled to all other components through the right-hand side f. This means that we have to know the solution for all other components in order to compute $U_i(t)$. Since this works also the other way around, we have to know $U_i(t)$ in order to compute the solutions for all other components, and since all other components step with different time-steps, it seems at first impossible to solve the equations in the way that we have described.

As a first simple strategy, though, we try to solve the system of nonlinear equations (8.1) 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. If we thus for component i need to know the value of component l at some time t_i, and we only have values for component l up until time $t_l < t_i$, the strategy is to extrapolate U_l from the interval containing t_l to time t_i, according to the order of U_l 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:

(8.2) *The last component steps first.*

This gives an explicit time-stepping method which works well enough for many problems. Each component is updated individually once, following (8.2), and we never go back to correct mistakes. For some problems however, solving the discrete equations in this explicit fashion is not accurate enough. We now describe how to extend the explicit time-stepping to an iterative process for solving the discrete equations.

8.2. **The time-slab.** The idea is now to in principle follow the strategy described above, with the addition that we go back and redo iterations if necessary. The way we do this is that we arrange 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 (8.1) 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

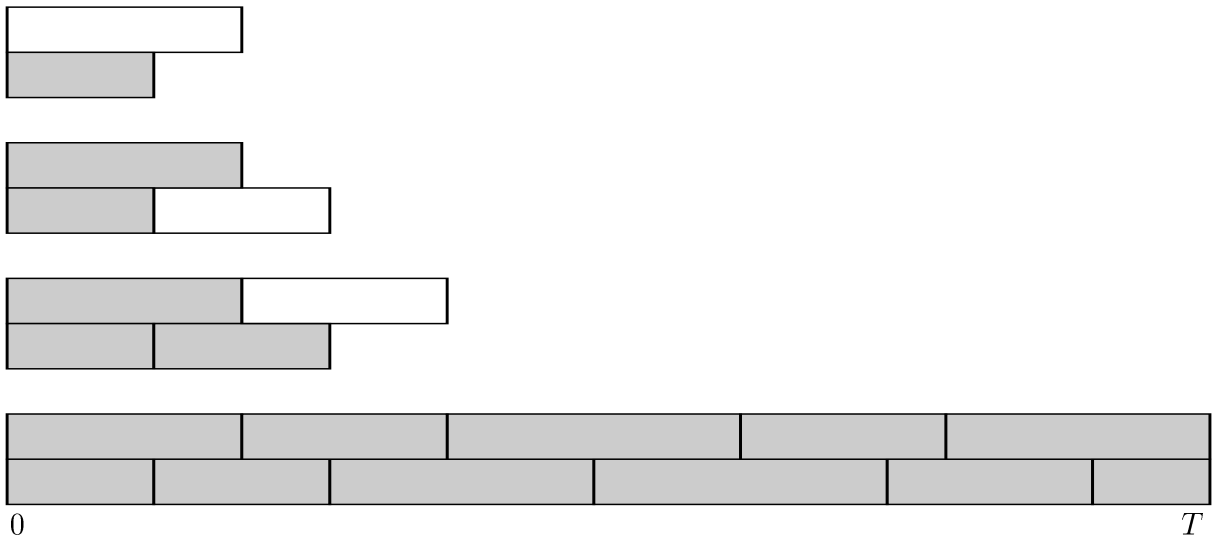


Figure 3: The last component steps first and all needed values are extrapolated or interpolated.

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 on all elements are small enough.

There are now a number of ways to construct the time-slab. One is *dyadic* partitioning, in which we compute new time-steps for all components, based on residuals and stability properties, choose the longest time-step as the length of the new time-slab, and then for every component choose a fraction $1/2^n$ of the length of the time-slab that is smaller than the desired time-step for that component. The advantage of such a partition is that the time-slab has straight edges — we'll explain more what we mean by this below — and that the components have many common nodes. The disadvantage may be that the choice of time-steps is constrained.

Another choice is to choose a *rational* partition of the interval. We choose the longest time-step as the length of the time-slab, and other possible choices are now fractions of this, 1/2, 1/3, 1/4 and so on. This allows for some more freedom when choosing the time-steps, and the time-slab still has straight edges, but we do not have as many common nodes as before.

We choose instead a third alternative: *total freedom.* The time-steps are not constrained at all, except that we match the final time end-point, and 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.

This construction of the time-slab brings with it a number of problems of technical and algorithmic nature. We will not here discuss 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 account for 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 4. We make iterations on the time-slab, starting with the last element and continuing to the right, towards $t = T$. After iterating through the time-slab a few times, the computational (discrete) residuals on all elements are small enough.

For some of the elements, the ones at the front, or *prow*, of the slab, the values have been computed with extrapolated values of many of the other elements. The strategy is now to leave behind *only those elements that are fully covered by all 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 the all elements of the previous time-slab will 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, iteration and formation of a new slab, where the formation part consists of the two steps of decreasing the size of the slab, leaving a number of elements behind, and increasing again the size of the slab, incorporating a number of new elements.

In addition to this, in order to solve the equations on the elements within the slab closest to $t = 0$, i.e. the elements at the *stern* of the slab, we have to know the values of a number of elements that are no longer part of the slab. For this reason, in addition to the slab, we keep a number of elements in a *tail* for each component, just to cover the rear end of the time-slab.

Remark 8.1. *Even if an element within the time-slab is totally covered by all elements, the values on this element may still 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.*

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

Applying Newton's method, we improve the range of allowed time-steps and the convergence rate, but this is costly in terms of memory and computational time, which is especially important for us, since the size of the slab may often be much larger than the number of components, N. We thus look for a cheap way that does not increase the cost of solving the problem, but still has the 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 the equation

$$U_{ij} = U_{i,j-1} + k_{ij} f_i(U(t_{ij}), t_{ij}). \tag{8.3}$$

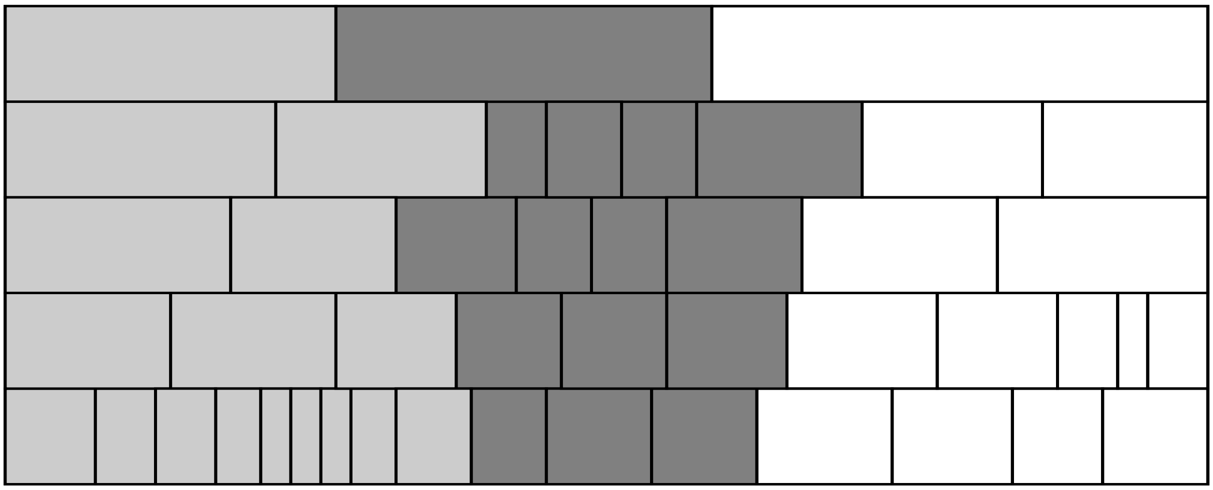


Figure 4: The time-slab used for the multi-adaptive stepping (dark grey). Grey elements are elements that have already been computed.

In order to apply Newton's method we write this as

$$F(V) = 0, \tag{8.4}$$

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

$$V^{n+1} = V^n - (F'(V^n))^{-1} F(V^n). \tag{8.5}$$

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

$$U_{ij}^{n+1} = U_{ij}^n - \frac{U_{ij}^n - U_{i,j-1} - k_{ij} f_i}{1 - k_{ij} \frac{\partial f_i}{\partial u_i}}, \tag{8.6}$$

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

$$U_{ij}^{n+1} = U_{ij}^n - \theta(U_{ij}^n - U_{i,j-1} - k_{ij} f_i) = (1-\theta) U_{ij}^n + \theta(U_{i,j-1} + k_{ij} f_i), \tag{8.7}$$

with

$$\theta = \frac{1}{1 - k_{ij} \frac{\partial f_i}{\partial u_i}}, \tag{8.8}$$

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 don't need to store the Jacobian, we don't have to use any linear algebra, and we still get 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.

8.4. **Explicit or implicit.** As described above, both mcG(q) and mdG(q) are in principle implicit, in the sense that they correspond to the nonlinear equations (8.1) on each time-slab. However, depending on the solution strategy for the non-linear equations (8.1), the resulting fully discrete scheme may be of more or less explicit character. Using a diagonal Newton 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. We now discuss this issue and why in fact we could with our implementation successfully solve also many stiff problems, such as systems of chemical reactions.

8.5. **The stiffness problem.** In a stiff problem the solution varies quickly in transients and slowly outside transients. For accuracy reasons the time-steps will adaptively be kept small inside transients, and will then be within the stability limits of an explicit method, while outside transients one would like to use larger time-steps. Thus Tanganyika has no problem to solve a stiff problem inside transients, and is in fact ideally suited for this purpose by its multi-adaptive feature. Outside the transients the diagonal Newton method handles stiff problems of sufficiently diagonal nature, and 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 couple of large time-steps are followed by some small corrective time-steps, see [31].

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 [26] 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.

8.6. **Extension to DAEs.** In [27] we extend the scope to DAEs, which we approach by regularizing the algebraic equations to stiff ODEs and solving the regularized systems by the methods described above, including the special technique of handling the stiffness.

8.7. **Precalc.** There are many “magic numbers” that need to be computed in order to implement the method, 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 of numerical analysis, see e.g. [37], and once these numbers are known, it is “simple” to implement the method.

8.8. **Solving the dual problem.** In addition to solving the primal problem, i.e. $\dot{u} = f$, we also have to solve the continuous dual problem. 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 and the approximate solution U. If U is sufficiently close to u, which we will assume, we take this mean value to be U. When solving the dual, the primal solution must be accessible, and the Jacobian must be computed numerically by difference quotients if we don't know it explicitly. This makes the computation of the dual solution expensive. There are, however, a couple of advantages as well, as compared to the primal problem. For one thing, we don't have to solve the dual with as high precision as for the primal; a relative error of say 10% may be disastrous for the primal, whereas for the dual this only means that our error estimate will be off by 10%, and we really only need to know the size of the error with at most on digit. Secondly, the dual 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 then solve the discrete equations directly without any iterations.

9. Tanganyika

We conclude by giving a short description of the implementation of the multi-adaptive method: *Tanganyika.* In the accompanying paper [31], we will present numerical results obtained with the multi-adaptive solver for a variety of test problems.

9.1. **Purpose.** The purpose of Tanganyika [32] is to be a working implementation of the multi-adaptive methods. If it may also serve as a general-purpose efficient and reliable ODE solver, that is of course a bonus, although reaching there still requires more of development and testing.

The code is open-source (GPL, see [1]), so feel free to dig in and see for yourself, at

`http://www.phi.chalmers.se/tanganyika/`

Comments are welcome.

All examples in this and the accompanying papers have been computed using this code, perhaps passing a few extra options, some of which may not be documented, to the solver.

9.2. **Structure and implementation.** The solver is implemented as a C/C++ library, and to solve the problem (1.1) the solver needs to know three things: *initial conditions*, *the right-hand side* f, and *the final time* T. In addition to the bare essentials there are a number of optional parameters that the solver accepts, such as e.g. the tolerance (which perhaps is really an essential parameter) and which method and order we want to use (automatic choice of this is thus not yet implemented).

The C++ language makes abstraction easy. The implementation thus closely reflects the description of the method. There are thus classes named `Solution`, `Element`, `cGqElement`, `dGqElement`, `TimeSlab`, `Galerkin`, `Component`, `ErrorControl` and so on.

References

[1] *Gnu gpl*, `http://www.gnu.org/`.

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

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

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

[5] G. Dahlquist, *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*, PhD thesis, Trans. of the Royal Inst. of Techn., Stockholm, Sweden, Number 130, Uppsala, 1958.

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

[7] C. Dawson and R. Kirby, *High resolution schemes for conservation laws with locally varying time steps*, to appear in SIAM J. Sci. Comp.

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

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

[10] ——, *Computational Differential Equations*, Studentlitteratur, Lund, 1st ed., 1996.

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

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

[13] ——, *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.

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

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

[16] 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.

[17] K. Eriksson, C. Johnson, and V. Thomée, *Time discretization of parabolic problems by the discontinuous galerkin method*, RAIRO MAN, 19 (1985), pp. 611–643.

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

[19] 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.

[20] D. Estep and R. Williams, *Accurate parallel integration of large sparse systems of differential equations*, Math. Models. Meth. Appl. Sci. (to appear).

[21] 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.

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

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

[24] P. Hansbo, *A note on energy conservation for hamiltonian systems using continuous time finite elements*, Chalmers Finite Element Center Preprint, 2001–05, available from `http://www.phi.chalmers.se/preprints/` (2001).

[25] 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, No. 4 (1988), pp. 908–926.
[26] C. Johnson and A. Logg, *Explicit time-stepping for stiff odes*, in preparation.
[27] ——, *Multi-adaptive galerkin methods for daes*, in preparation.
[28] O. Kessel-Deynet, *Berücksichtigung ionisierender Strahlung im Smoothed-Particle-Hydrodynamics-Verfahren und Anwendung auf die Dynamik von Wolkenkernen im Strahlungsfeld massiver Sterne*, PhD thesis, Naturwissenschaftlich-Mathematischen Gesamtfakultät, Ruprecht-Karls-Universität, Heidelberg, 1999.
[29] A. Logg, *Multi-adaptive error control for odes*, Oxford University Computing Laboratory, NA Group Report no 98/20, also available from `http://www.phi.chalmers.se/preprints` (1998/2000).
[30] ——, *A multi-adaptive ode-solver*, MSc thesis, Department of mathematics, Chalmers University of Technology, Göteborg, also available from `http://www.phi.chalmers.se/preprints` (1998/2000).
[31] ——, *Multi-adaptive galerkin methods for odes ii: Applications*, Chalmers Finite Element Center Preprint 2001–10, available from `http://www.phi.chalmers.se/preprints/`, (2001).
[32] ——, *Tanganyika, version 1.2*, `http://www.phi.chalmers.se/tanganyika/`, (2001).
[33] 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.
[34] K.-S. Moon, A. Szepessy, R. Tempone, and G. Zourakis, *Adaptive approximation of differential equations based on global and local errors*, Nada Preprint, KTH, TRITA-NA-0006 (2000).
[35] P. Niamsup and V. Phat, *Asymptotic stability of nonlinear control systems described by difference equations with multiple delays*, Electronic Journal of Differential Equations, 11 (2000), pp. 1–17.
[36] M. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1988.
[37] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, PostScript Online Version, `http://www.nr.com`, 2nd ed., 1997.
[38] G. Söderlind, *The automatic control of numerical integration*, CWI Quarterly, 11 (1998), pp. 55–74.
[39] L. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, 1994.
[40] T. Werder, *hp discontinuous galerkin time stepping methods for parabolic problems*, Diploma Thesis, ETH, (2000).

Appendix A. Multi-Adaptive Galerkin Details

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 $[0,1]$. This is discussed in more detail below. 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: 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 $_{ij}$ sub-indices 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 $\bar{A}^{[q]} = \left(\bar{a}_{mn}^{[q]}\right)$ of the matrix $A^{[q]} = \left(a_{mn}^{[q]}\right)$. Thus, switching back to the full notation, we get

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

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

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

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 (4.3), this amounts to showing that if

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

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: 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

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

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. In Table 1 we list some low-order weight functions for an equidistant nodal base. Notice that with $s_q = 1$ we have $w_q^{[q]} = 1$. This agrees with (4.3), taking $v = 1$.

$w_1^{[1]}(\tau) = 1$		
$w_1^{[2]}(\tau) = \frac{1}{4}(5 - 6\tau)$	$w_2^{[2]}(\tau) = 1$	
$w_1^{[3]}(\tau) = \frac{1}{27}(37 - 96\tau + 60\tau^2)$	$w_2^{[3]}(\tau) = \frac{1}{27}(26 + 24\tau - 60\tau^2)$	$w_3^{[3]}(\tau) = 1$

Table 1: Weight functions for the cG(q) method with an equidistant nodal base for $q = 1, 2, 3$.

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 sub-interval is a nodal point for the basis functions.

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

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

for $m = 0, \ldots, q_{ij}$, where we use ξ_{ij0}^- to denote $\xi_{i,j-1,q_{ij}}$, i.e. the value at the right end-point of the previous interval. To simplify the notation, we drop the sub-indices $_{ij}$ 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

$$\left(\sum_{n=0}^{q} \xi_n \lambda_n^{[q]}(0) - \xi_0^- \right) \lambda_m^{[q]}(0) + \sum_{n=0}^{q} \xi_n \frac{1}{k} \int_0^k \dot{\lambda}_n^{[q]}(\tau(t)) \lambda_m^{[q]}(\tau(t)) \, dt = \int_0^k f \lambda_m^{[q]}(\tau(t)) \, dt, \tag{A.15}$$

or simply

$$\sum_{n=0}^{q} a_{mn}^{[q]} \xi_n = b_m, \tag{A.16}$$

where

$$a_{mn}^{[q]} = \int_0^1 \dot{\lambda}_n^{[q]}(t)\lambda_m^{[q]}(t)\ dt + \lambda_n^{[q]}(0)\lambda_m^{[q]}(0), \tag{A.17}$$

and

$$b_m^{[q]} = \int_0^k f\lambda_m^{[q]}(\tau(t))\ dt + \xi_0^- \lambda_m^{[q]}(0). \tag{A.18}$$

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

$$\xi_{ijm} = \xi_{ij0}^- \sum_{n=0}^{q} \bar{a}_{mn}^{[q]}\lambda_n^{[q]}(0) + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt,\ \ m = 0,\ldots,q_{ij}, \tag{A.19}$$

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

$$w_m^{[q]} = \sum_{n=0}^{q} \bar{a}_{mn}^{[q]}\lambda_n^{[q]},\ \ m = 0,\ldots,q. \tag{A.20}$$

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

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

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

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

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: For every local interval I_{ij}, find $\{\xi_{ijn}\}_{n=0}^{q_{ij}}$, such that for $m = 0,\ldots,q_{ij}$ we have

$$\xi_{ijm} = \xi_{ij0}^- + \int_{I_{ij}} w_m^{[q_{ij}]}(\tau_{ij}(t))\ f_i(U(t),t)\ dt, \tag{A.21}$$

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

Remark A.1. *Also for the* mdG(q) *method, we have the property that* $w_q^{[q]} = 1$. *To see this, take* $v = 1$ *in the Galerkin orthogonality.*

A.3. **Choosing basis functions and quadrature.** We now discuss how to choose nodal points for the nodal basis functions, and quadrature points for the quadrature.

Having the same nodal points as quadrature points may increase the efficiency of the solver, since when we then wish to evaluate some function expressed in the nodal basis at a quadrature point, we may just take the nodal value at that point, instead of having to evaluate all basis functions at this specific point and compute the weighted sum. It turns out that it is indeed possible to choose the same quadrature points as nodal points in a good way.

As already mentioned above, we assume that the end-points of every interval are nodal points, both for the continuous and the discontinuous method. This is for practical purposes; the end-point values will have certain significance for both methods. For the continuous method it is desirable to have also the left end-point of every interval as a nodal point. This makes it easy to impose the continuity on the solution, just taking $\xi_{ij0} = \xi_{i,j-1,q_{i,j-1}}$.

If the polynomial order on some interval is q, we have to compute integrals of the right-hand side times polynomial weight functions of degree $q-1$ for the mcG(q) method, and degree q for the mdG(q) method, see (4.4) and (4.8). The main criterion for choosing the quadrature is now that the quadrature should be exact at least if the right-hand side is a polynomial of degree q, i.e. locally $f_i(U(t),t)|_{I_{ij}}$ is a polynomial of degree q. The quadrature should thus be exact for polynomials of degree

$$q_{\mathrm{cG}} = q + q - 1 = 2q - 1 \tag{A.22}$$

for the continuous method and

$$q_{\mathrm{dG}} = q + q = 2q \tag{A.23}$$

for the discontinuous method.

For the continuous method we thus want a quadrature that is exact for polynomials of degree $2q-1$, and where the end-points are quadrature points. Another requirement is that the Lagrange nodal basis functions should be able to represent polynomials of degree q, which means we want $q+1$ nodal points. Now, the best possible quadrature with $q+1$ nodal points, including both end-points of the interval, is the *Lobatto*, or *Gauss-Lobatto*, rule, which turns out to be exact for polynomials of degree less than or equal to $2q-1$. We thus choose the Lobatto points as both quadrature and nodal points. The Lobatto points on $[-1,1]$ are given by -1, 1 and the zeros of the derivative of the q:th-order Legendre polynomial on $[-1,1]$, $P_q'(x)$, which can be expressed as

$$P_q'(x) = \frac{1}{x^2-1}\left[qxP_q(x) - qP_{q-1}(x)\right]. \tag{A.24}$$

We may thus express this as finding the zeros of $xP_q(x) - P_{q-1}(x)$.

For the discontinuous method, we want a quadrature that is exact for polynomials of degree $2q$, i.e. one degree more than for the continuous method. We wish to represent polynomials of the same degree, q, with the nodal basis, and so we have the same number of nodal points, $q+1$. We thus now want a quadrature rule that is one order better with the only difference that now we don't need to include both end-points, only one of them. This turns out to be possible. The best quadrature points with one end-point included are the *Radau*, or *Gauss-Radau*, points of the interval. With this quadrature we can integrate polynomials of degree $2q$ exactly which is just what we want. The Radau quadrature points of $[-1,1]$ are given by -1 and the zeros of

$$\frac{P_q(x)+P_{q+1}(x)}{1+x}, \tag{A.25}$$

where P_q is the q:th-order Legendre polynomial on $[-1,1]$. We may thus express this as finding the zeros of P_q+P_{q+1}. Note now that if we want the right end-point to be included, rather than the left, we have to flip these points to have 1 as a nodal point.

Remark A.2. *Using different time-steps for different components, we will not be able to benefit as much from choosing the same quadrature as nodal points as we would with the same time-steps for all components, since then a quadrature point for one component is not necessarily a nodal point for another component.*

Remark A.3. *One may also think of choosing the basis so as to give as simple coefficients, $\int_I \dot{\lambda}_j\lambda_i\,dt$ or $\int_I \lambda_j\lambda_i\,dt$, as possible. The latter of these turn up in linear problems. As an example, consider the discontinuous method for a linear problem, where a good choice of basis functions is then the Legendre polynomials on the interval, see* [40].

Appendix B. The Discrete Dual Problem

The discrete dual solution Φ is a Galerkin approximation of the continuous problem

$$\left\{ \begin{array}{rcl} -\dot{\varphi} & = & J^*(\pi_k u, U, \cdot)\varphi + g, \\ \varphi(T) & = & \varphi_T, \end{array} \right. \tag{B.1}$$

where $\pi_k u$ is an interpolant or a projection of the true solution u, and where we will choose the Galerkin approximation to suit the needs of the a priori error analysis below.

As everything else is the other way around for the dual problem it should come as no surprise that the test and trial spaces are interchanged. Thus for the mcG(q) method the dual trial space consists of all (possibly discontinuous) piecewise polynomials of degrees $\{q_{ij} - 1\}$, and the test space consists of all continuous piecewise polynomials of degrees $\{q_{ij}\}$, on the same partition as for the forward problem. To ensure that the dimensions of the test and trial spaces are the same, we require that the test functions vanish at the end-point $t = 0$. To see that we get the correct number of degrees of freedom, note first that with M intervals for a certain component, the count is Mq for the trial functions. For the test functions, we have $q + 1$ degrees of freedom per interval, from which we subtract one degree of freedom per interval; one for everyone of the $M - 1$ continuity requirements, and one extra for the requirement that the test functions must vanish at the end-point. The total number of degrees of freedom is thus

$$M(q+1) - (M-1) - 1 = M(q+1) - M = Mq, \tag{B.2}$$

in agreement with the number of degrees of freedom for the trial functions.

The discrete dual problem for the mcG(q) method is then to find $\Phi \in W$, such that

$$-(\Phi(T), v(T)) + \sum_{i=1}^{N} \sum_{j=1}^{M_i} \int_{I_{ij}} \Phi_i \dot{v}_i \; dt = \int_0^T (J^*(\pi_k u, U, \cdot)\Phi + g, v) \; dt, \tag{B.3}$$

for all $v \in V$, where $\Phi(T) = \Phi_T$ and

(B.4)

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

In the mdG(q) method the trial and test spaces are the same. The discrete dual problem for the mdG(q) method is thus to find $\Phi \in W$, such that

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \left[-[\Phi_i]_{ij} v_{ij}^- - \int_{I_{ij}} \dot{\Phi}_i v_i \; dt \right] = \int_0^T (J^*(\pi_k u, U, \cdot)\Phi + g, v) \; dt, \tag{B.5}$$

for all $v \in V$, where $\Phi(T^+) = \Phi_T$ and the trial and test spaces are the same as for the mdG(q) method, i.e.

$$V = W = \{v : v_i|_{I_{ij}} \in \mathcal{P}^{q_{ij}}(I_{ij}), j = 1, \ldots, M_i,\ i = 1, \ldots, N\}. \tag{B.6}$$

Appendix C. Stability Estimates

In this section we prove stability estimates for the multi-adaptive methods and their duals.

C.1. **Stability estimates for the primal methods.** We start by proving basic stability estimates for the the linear problem

$$\left\{ \begin{array}{rcl} u'(t) & = & A(t)u(t), \ t \in (0,T], \\ u(0) & = & u_0, \end{array} \right. \tag{C.1}$$

where $A(t)$ is an $N \times N$ matrix depending on time t. The goal is to estimate the size of the discrete solution U, and its derivatives, in terms of A. As a simple measure of A, we take

$$\mathcal{A} = \max_{0 \le p \le q-1} \max_{i,j} \sup_{t \in [0,T]} |a_{ij}^{(p)}(t)|, \tag{C.2}$$

where $A = (a_{ij})$ and q is the order of the derivative we want to estimate. We use $\{t_m\}$ to denote synchronized time-levels, and denote by $K_m = t_m - t_{m-1}$ the corresponding time-steps.

To begin with, we will need the following variant of the discrete Grönwall inequality.

Lemma C.1. *Assume that $z, a : \mathbb{N} \to \mathbb{R}$ are non-negative, $z(0) \le C$, $a(i) \le 1/2$ for all i, and*

$$z(m) \le C + \sum_{i=1}^{m} a(i)z(i). \tag{C.3}$$

Then, for $m = 1, 2, \ldots$, we have

$$z(m) \le 2C \exp\left(\sum_{i=1}^{m-1} 2a(i)\right). \tag{C.4}$$

Proof. The usual discrete Grönwall inequality takes the form (see e.g. [35])

$$z(m) \le C \exp\left(\sum_{i=0}^{m-1} a(i)\right),$$

under the assumption

$$z(m) \le C + \sum_{i=0}^{m-1} a(i)z(i).$$

Assuming $a(i) \le 1/2$ for all i, we have by (C.3)

$$(1 - a(m))z(m) \le C + \sum_{i=1}^{m-1} a(i)z(i),$$

so that, since $|1-a(m)| \geq 1/2$,

$$z(m) \leq 2C + \sum_{i=1}^{m-1} 2a(i)z(i),$$

and the desired estimate follows. $\square$

We can now prove the stability estimates for the mcG(q) and the mdG(q) method.

Lemma C.2. *Let U be the* mcG(q) *solution of (C.1), and assume for simplicity that $\mathcal{A} \geq 1$. If for some constant C_q, only depending on the highest polynomial order, $C_q N \mathcal{A} K_m \leq 1/2$ for all m, then for $i = 1, \ldots, N$ we have*

$$|U_i^{(p)}(t)| \leq C_q \mathcal{A}^p \exp(C_q N \mathcal{A} t_m) \sum_{i=1}^{N} |u_i(0)|, \;\; p = 0, 1, \ldots, \tag{C.5}$$

where t_m is the next synchronized time-level with $t \leq t_m$.

Proof. By (4.4) we have

$$\xi_{ijm} = u_i(0) + \int_0^{t_{ij}} \gamma_{ijm}(t)(A(t)U(t))_i \; dt, \tag{C.6}$$

for some piecewise polynomial weight functions $\{\gamma_{ijm}\}$, and therefore

$$|\xi_{ijm}| \leq |u_i(0)| + C_q \mathcal{A} \int_0^{t_{ij}} \sum_{n=1}^{N} |U_n(t)| \; dt.$$

We then have, with a new constant C_q (depending on the choice of nodal points),

$$|U_i(t)| \leq |u_i(0)| + C_q \mathcal{A} \int_0^{t_{ij}} \sum_{n=1}^{N} |U_n(t)| \; dt, \;\; t \in [0, t_{ij}].$$

Now choose $t_{ij} = t_m$ for some synchronized time-level t_m, so that this expression holds for all i, and make the definition

$$\bar{U}_i(t_m) = \max_{t \in [0, t_m]} |U_i(t)|,$$

and let

$$\bar{U}(t_m) = \sum_{i=1}^{N} \bar{U}_i(t_m),$$

with $\bar{U}_i$ and $\bar{U}$ piecewise constant on the partition defined by the synchronized time-levels. Then, summing over all components, we get

$$\begin{array}{rcl} \bar{U}(t_m) & \leq & \sum_{i=1}^{N} |u_i(0)| + C_q N \mathcal{A} \int_0^{t_m} \sum_{n=1}^{N} |U_n(t)| \; dt \\ & \leq & \sum_{i=1}^{N} |u_i(0)| + C_q N \mathcal{A} \int_0^{t_m} \bar{U}(t) \; dt \\ & = & \sum_{i=1}^{N} |u_i(0)| + \sum_{n=1}^{m} C_q N \mathcal{A} K_n \bar{U}(t_n), \end{array}$$

where $\{K_m\}$ are the lengths of the time-slabs, i.e. the distances between adjacent synchronized time-levels. Since by assumption $C_q N \mathcal{A} K_m \leq 1/2$ we have, by the discrete Grönwall lemma,

$$\bar{U}(t_m) \leq 2 \sum_{i=1}^{N} |u_i(0)| \exp\left(\sum_{n=1}^{m-1} 2 C_q N \mathcal{A} K_n \right) \leq 2 e^{2 C_q N \mathcal{A} t_m} \sum_{i=1}^{N} |u_i(0)|,$$

so that, in particular,

$$|U_i(t)| \leq 2 e^{2 C_q N \mathcal{A} t_m} \sum_{i=1}^{N} |u_i(0)|,$$

for $0 \leq t \leq t_m$.

Now, in order to control derivatives of U, we choose a local test function v for component i that vanishes outside I_{ij} in the Galerkin orthogonality (4.1) to get

$$\int_{I_{ij}} \dot{U}_i v \; dt = \int_{I_{ij}} (AU)_i v \; dt.$$

Since the test function is one degree lower than U_i for the mcG(q) method, we may choose v as the L^2-projection onto $\mathcal{P}^{q_{ij}-1}(I_{ij})$ of $\mathrm{sgn}(\dot{U}_i)$, which gives

$$\int_{I_{ij}} |\dot{U}_i| \; dt = \int_{I_{ij}} \dot{U}_i \; \mathrm{sgn}(\dot{U}_i) \; dt = \int_{I_{ij}} \dot{U}_i v \; dt = \int_{I_{ij}} (AU)_i v \; dt \leq \mathcal{A} \bar{U}(t_{ij}) \int_{I_{ij}} |v| \; dt.$$

Since v is an L^2-projection of $\mathrm{sgn}(U_i)$, it follows by the Cauchy inequality that $\int_{I_{ij}} |v| \; dt \leq \sqrt{k_{ij}} \left(\int_{I_{ij}} v^2 \; dt \right)^{1/2} \leq \sqrt{k_{ij}} \left(\int_{I_{ij}} (\mathrm{sgn}(U_i))^2 \; dt \right)^{1/2} \leq k_{ij}$, and therefore

$$\frac{1}{k_{ij}} \int_{I_{ij}} |\dot{U}_i| \; dt \leq \mathcal{A} \bar{U}(t_{ij}).$$

By an inverse estimate for polynomials, it follows that

$$\max_{I_{ij}} |\dot{U}_i| \leq C_{q_{ij}} \mathcal{A} \bar{U}(t_{ij}),$$

so that in particular

$$\max_{[0,t_m]} |\dot{U}_i| \leq C_{q_i} \mathcal{A} \bar{U}(t_m),$$

for all synchronized time-levels t_m.

In order to control higher order derivatives, we choose $v \in \mathcal{P}^{q_{ij}-1}(I_{ij})$ so that for all $w \in \mathcal{P}^{q_{ij}-p}$, we have

$$\int_{I_{ij}} \mathrm{sgn}(U_i^{(p)}) \; w \; dt = \int_{I_{ij}} v \; w \; dt.$$

This will determine $q_{ij} + 1 - p$ degrees of freedom for v, leaving $q_{ij} - (q_{ij} + 1 - p) = p - 1$ degrees of freedom. We choose these to have $[(U_i^{(p-n)} - (AU)_i^{(p-n-1)}) v^{(n-1)}]_{t_{i,j-1}}^{t_{ij}} = 0$ for

$n = 1, \ldots, p-1$, and obtain

$$\begin{array}{rcl} \int_{I_{ij}} |U_i^{(p)}| \; dt & = & \int_{I_{ij}} U_i^{(p)} \,\mathrm{sgn}(U_i^{(p)}) \; dt \\ & = & \int_{I_{ij}} U_i^{(p)} v \; dt \\ & = & \sum_{n=1}^{p-1} (-1)^{n-1} [U_i^{(p-n)} v^{(n-1)}]_{t_{i,j-1}}^{t_{ij}} + (-1)^{p-1} \int_{I_{ij}} \dot{U}_i v^{(p-1)} \; dt \\ & = & \sum_{n=1}^{p-1} (-1)^{n-1} [U_i^{(p-n)} v^{(n-1)}]_{t_{i,j-1}}^{t_{ij}} + (-1)^{p-1} \int_{I_{ij}} (AU)_i v^{(p-1)} \; dt \\ & = & \sum_{n=1}^{p-1} (-1)^{n-1} [(U_i^{(p-n)} - (AU)_i^{(p-n-1)}) v^{(n-1)}]_{t_{i,j-1}}^{t_{ij}} + \int_{I_{ij}} ((AU)_i)^{(p-1)} v \; dt \\ & = & \int_{I_{ij}} ((AU)_i)^{(p-1)} v \; dt \\ & \leq & k_{ij} C_p \mathcal{A} \max_{n=0,\ldots,p-1} \overline{U^{(n)}}. \end{array}$$

Thus, by induction, assuming for simplicity that $\mathcal{A} \geq 1$, we get, again by an inverse estimate,

$$|U_i^{(p)}(t)| \leq C_q \mathcal{A}^p \exp(C_q N \mathcal{A} t_m) \sum_{i=1}^{N} |u_i(0)|,$$

for every $t \leq t_m$ (with a new constant C_q). Note that this expression holds also $p > q$, since those derivatives are zero. □

Lemma C.3. *Let U be the* mdG(q) *solution of (C.1), and assume for simplicity that $\mathcal{A} \geq 1$. If for some constant C_q, only depending on the highest polynomial order, $C_q N \mathcal{A} K_m \leq 1/2$ for all m, then for $i = 1, \ldots, N$ we have*

$$|U_i^{(p)}(t)| \leq C_q \mathcal{A}^p \exp(C_q N \mathcal{A} t_m) \sum_{i=1}^{N} |u_i(0)|, \;\; p = 0, 1, \ldots, \tag{C.7}$$

where t_m is the smallest synchronized time-level with $t \leq t_m$.

Proof. Following the proof for the continuous method, since we also for the discontinuous method have an expression of the kind

$$\xi_{ijm} = u_i(0) + \int_0^{t_{ij}} \gamma_{ijm}(t) (A(t) U(t))_i \; dt,$$

it is clear that we will also get the estimate

$$|U_i(t)| \leq 2 e^{2 C_q N \mathcal{A} t_m} \sum_{i=1}^{N} |u_i(0)|,$$

where now C_q is some other constant, depending on the highest order of the method. For estimating higher order derivatives, notice that also for the discontinuous method we have

$$\int_{I_{ij}} \dot{U}_i v \; dt = \int_{I_{ij}} (AU)_i v \; dt,$$

if we take the test function zero at $t_{i,j-1}$. This will take away one degree of freedom, but since the test functions have one extra degree of freedom for the discontinuous method, we are still able to estimate the derivatives as before, possibly with different constants. □

C.2. **Stability estimates for the dual methods.** We now discuss stability estimates for the discrete dual problems, i.e. obtain bounds on Φ in terms of f or J. In the following, we limit ourselves to estimating the final time l_2-error, i.e. the right-hand side data for the discrete dual problems, g, is assumed to be zero. In order to obtain the same estimates as for the primal forward problem, we show that the dual methods, i.e. (B.3) and (B.5), are time-stepping methods in the sense that the degrees of freedom are given by expressions similar to (C.6).

Consider component i of the discrete dual for the continuous method. Choosing the test function $v_j = 0$ for $j \neq i$ and v_i as the test function w that vanishes outside $[t_{i,j-1}, T]$ and is equal to one on $[t_{ij}, T]$ in (B.3), we have

$$-\Phi_i(T) + \int_{I_{ij}} \Phi_i \dot{w}\ dt = \int_{t_{ij}}^{T} B_i\ dt + \int_{I_{ij}} B_i w\ dt, \tag{C.8}$$

with $B_i(t) = (J^*(\pi_k u(t), U(t), t)\Phi(t))_i$. If we now make an Ansatz for the discrete dual Φ_i, as we did for the forward method, and vary the test function w over I_{ij} with $w(t_{i,j-1}) = 0$ and $w(t_{ij}) = 1$, we get expressions for the degrees of freedom for Φ_i on I_{ij}, $\{\eta_{ijm}\}_{m=0}^{q_{ij}-1}$, of the form

$$\eta_{ijm} = \Phi_i(T) + \int_{t_{i,j-1}}^{T} \gamma_{ijm} B_i\ dt, \tag{C.9}$$

for $m = 0, \ldots, q_{ij} - 1$, where $\{\gamma_{ijm}\}$ are polynomial weight functions. In the same way as before for the forward method, we thus get a similar stability estimate for the dual backward method.

For the dual backward method of the discontinuous method, we will, since the test and trial spaces are the same, get exactly the same estimate as before for the forward method.

We thus have stability estimates also for the dual methods, and we state these in the simplest possible way.

Lemma C.4. *If the Jacobian elements $\{J_{ij}(t)\}$ and their derivatives are bounded by $\mathcal{A}$ on $[0, T]$, and the lengths of the time-slabs, $\{K_m\}$, are small enough in terms of $\mathcal{A}$, then the discrete duals and their derivatives, both for the* mcG(q) *and the* mdG(q) *method, are bounded on $[0, T]$ in terms of $\mathcal{A}$.*

Appendix D. A Priori Error Analysis

In this section we prove a priori error estimates for the multi-adaptive Galerkin methods. We obtain estimates showing that the mcG(q) method is of order $2q$ and the mdG(q) method is of order $2q+1$.

D.1. **Introduction.** The goal here is to obtain estimates for the error in terms of the time-steps k, i.e. estimates of the form

$$|||e(t)||| \leq \int_0^T (k^p, w)\ dt, \tag{D.1}$$

for some integer $p > 0$, and where w does not depend on the computation.

D.2. **Error representation.** We split the error into two parts,

$$e = U - u = (U - \pi_k u) + (\pi_k u - u) = \bar{e} + (\pi_k u - u),$$

where $\bar{e}$ denotes the *discrete error* $U - \pi_k u$ for $\pi_k u$ some *trial* space approximation of the exact solution. We will choose $\pi_k u$ in a special way below, and we will refer to it simply as the interpolant. The idea is that the second part of the error is easy to estimate in terms of derivatives of u, so if we can estimate $\bar{e}$ we are done.

The following two theorems represent this part of the error in terms of the residual of the interpolant. This is the main part of the a priori error analysis, together with the stability estimates for the discrete duals.

Theorem D.1. *For the multi-adaptive continuous Galerkin method,* mcG(q)*, the discrete error $\bar{e} = U - \pi_k u$, where $\pi_k u$ is a trial space interpolant of the exact solution, can be represented in terms of the residual of the trial space interpolant and the discrete dual Φ as*

$$(\bar{e}(T), \Phi_T) + \int_0^T (\bar{e}, g)\ dt = -\int_0^T (R(\pi_k u, \cdot), \Phi)\ dt,$$

if $\pi_k u$ interpolates u at end-points of every local interval.

Proof. Since $\bar{e}(0) = 0$ and $\bar{e}$ is a continuous piecewise polynomial of orders $\{q_{ij}\}$, it is a test function for the discrete dual, and thus

$$(\bar{e}(T), \Phi_T) + \int_0^T (\bar{e}, g)\ dt = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \int_{I_{ij}} \dot{\bar{e}}_i \Phi_i\ dt + \int_0^T (\bar{e}, -J^*(\pi_k u, U, \cdot)\Phi)\ dt,$$

where

$$\begin{aligned} \textstyle\int_0^T (\bar{e}, -J^*(\pi_k u, U, \cdot)\Phi)\ dt &= \textstyle\int_0^T (-J(\pi_k u, U, \cdot)\bar{e}, \Phi)\ dt \\ &= \textstyle\int_0^T (f(\pi_k u, \cdot) - f(U, \cdot), \Phi)\ dt \\ &= \textstyle\sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (f_i(\pi_k u, \cdot) - f_i(U, \cdot))\Phi_i\ dt. \end{aligned}$$

Since now Φ is a test function for the forward problem, it is orthogonal to the residual of U, so that

$$\begin{aligned}\int_{I_{ij}} [\dot{\bar{e}}_i\Phi_i + (f_i(\pi_k u,\cdot) - f_i(U,\cdot))\Phi_i]\ dt &= \int_{I_{ij}} \left[\dot{U}_i - \tfrac{d}{dt}\pi_k u_i + f_i(\pi_k u,\cdot) - f_i(U,\cdot)\right]\Phi_i\ dt \\ &= \int_{I_{ij}} [-R_i(\pi_k u,\cdot) + R_i(U,\cdot)]\,\Phi_i\ dt \\ &= -\int_{I_{ij}} R_i(\pi_k u,\cdot)\Phi_i\ dt.\end{aligned}$$

Summing up, noticing that the residual of the interpolant is well-defined except at the nodes where it is discontinuous, we have

$$(\bar{e}(T),\Phi_T) + \int_0^T (\bar{e},g)\ dt = -\int_0^T (R(\pi_k u,\cdot),\Phi)\ dt,$$

which completes the proof. $\square$

Theorem D.2. *For the multi-adaptive discontinuous Galerkin method,* mdG(q), *the discrete error $\bar{e} = U - \pi_k u$, where $\pi_k u$ is a trial space interpolant of the exact solution, can be represented in terms of the residual of the trial space interpolant and the discrete dual Φ as*

$$(\bar{e}(T),\Phi_T) + \int_0^T (\bar{e},g)\ dt = \sum_{i=1}^{N}\sum_{j=1}^{M_i}\left[-[\pi_k u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} R_i(\pi_k u)\Phi_i\ dt\right],$$

if $\pi_k u$ interpolates u at the right end-point of every interval.

Proof. We now try to do the same thing as in the previous proof, having only to keep track of a few extra terms. Since all four trial and test spaces are the same for the primal and the dual problem, it is clear that $\bar{e}$ is a test function for the discrete dual, and thus

$$\int_0^T (\bar{e},g)\ dt = \sum_{ij}\left[-[\Phi_i]_{ij}\bar{e}_{ij}^- + \int_{I_{ij}} -\bar{e}\dot{\Phi}_i\ dt\right] + \int_0^T (\bar{e}, -J^*(\pi_k u, U,\cdot)\Phi)\ dt.$$

As before, we have

$$\int_0^T (\bar{e}, -J^*(\pi_k u, U,\cdot)\Phi)\ dt = \sum_{ij}\int_{I_{ij}} (f_i(\pi_k u,\cdot) - f_i(U,\cdot))\Phi_i\ dt.$$

Now, integrating by parts, we get

$$-[\Phi_i]_{ij}\bar{e}_{ij}^- + \int_{I_{ij}} -\bar{e}_i\dot{\Phi}_i\ dt \;=\; \bar{e}_i(t_{i,j-1}^+)\Phi_i(t_{i,j-1}^+) - \bar{e}_i(t_{ij}^-)\Phi_i(t_{ij}^+) + \int_{I_{ij}} \dot{\bar{e}}_i\Phi_i\ dt,$$

so that, since $\bar{e}(0^-) = 0$ and $\bar{e}$ is right-continuous at T,

$$\sum_{ij}\left[-[\Phi_i]_{ij}\bar{e}_{ij}^- + \int_{I_{ij}} -\bar{e}_i\dot{\Phi}_i\ dt\right] \;=\; -(\bar{e}(T),\Phi_T) + \sum_{ij}\left[[\bar{e}_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} \dot{\bar{e}}_i\Phi_i\ dt\right].$$

Summing up and using the Galerkin orthogonality for U, we have, noting that $\dot{U}_i - \frac{d}{dt}\pi_k u_i + f_i(\pi_k u, \cdot) - f_i(U, \cdot) = R_i(U, \cdot) - R_i(\pi_k u, \cdot)$,

$$\begin{aligned}(\bar{e}(T), \Phi_T) + \int_0^T (\bar{e}, g)\ dt &= \sum_{ij} \left[[\bar{e}_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) + \int_{I_{ij}} \left(-R_i(\pi_k u, \cdot) + R_i(U, \cdot)\right) \Phi_i\ dt \right] \\ &= \sum_{ij} \left[-[\pi_k u_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} R_i(\pi_k u) \Phi_i\ dt \right],\end{aligned}$$

completing the proof. □

D.3. **Interpolation estimates.** What follows here is some further preparation for the a priori error estimates, for which we need expressions for $u - \pi_k u$ with $\pi_k u$ some trial space approximation of u chosen in a special way. For now, assume that f is a real-valued scalar function on the interval $[a, b]$.

For the mcG(q) method, we define the following interpolant:

$$\text{(D.2)} \qquad \begin{cases} \pi^{[q]} f \in \mathcal{P}^q(a, b), \\ \pi^{[q]}(a) = f(a) \text{ and } \pi^{[q]}(b) = f(b), \\ \int_a^b (f - \pi^{[q]} f) v\ dx = 0\ \forall v \in \mathcal{P}^{q-2}(a, b), \end{cases}$$

i.e. $\pi^{[q]} f$ is a polynomial of degree q that interpolates f at the end-points. The further $q + 1 - 2 = q - 1$ degrees of freedom are determined by $q - 1$ projection conditions. Notice that for $q = 1$ only the interpolation conditions are necessary to determine the interpolant.

For the mdG(q) method, we define the following interpolant:

$$\text{(D.3)} \qquad \begin{cases} \pi^{[q]} f \in \mathcal{P}^q(a, b), \\ \pi^{[q]}(b) = f(b), \\ \int_a^b (f - \pi^{[q]} f) v\ dx = 0\ \forall v \in \mathcal{P}^{q-1}(a, b), \end{cases}$$

i.e. $\pi^{[q]} f$ is a polynomial of degree q that interpolates f at the right end-point. The further $q + 1 - 1 = q$ degrees of freedom are determined by q projection conditions. Again, notice that for the lowest degree polynomial of the method, in this case $q = 0$, only the interpolation condition is necessary to determine the interpolant.

We will use the *Peano kernel theorem* to represent $f - \pi^{[q]} f$ in terms of derivatives of f. To do so, we must first investigate a few important properties of the two interpolants.

Lemma D.1. *For both types of interpolants, if $f \in \mathcal{P}^q(0, 1)$ then $\pi^{[q]} f = f$.*

Proof. For the mcG(q) interpolant, $f - \pi^{[q]} f$ is zero at 0 and 1 and is a polynomial of degree q, so that it has a maximum of $q - 2$ zeros on $[0, 1]$. We may thus choose the test function $v \in \mathcal{P}^{q-2}(0, 1)$ such that it has the same sign as $f - \pi^{[q]} f$. If then $f(x) - \pi^{[q]} f(x) \neq 0$ at some point, we have

$$\int_0^1 (f - \pi^{[q]} f) v\ dx > 0,$$

in contradiction to the property of the interpolant that $f - \pi^{[q]} f$ should be orthogonal to $\mathcal{P}^{q-2}(0, 1)$. Thus $f - \pi^{[q]} f = 0$ and $\pi^{[q]} f = f$.

For the mdG(q) interpolant, $f - \pi^{[q]} f$ is zero at 1 and is a polynomial of degree q with $q-1$ zeros on $[0,1]$. Choosing now the test function $v \in \mathcal{P}^{q-1}(0,1)$ with the same sign as $f - \pi^{[q]} f$, we see as before that we must have $\pi^{[q]} f = f$. □

Lemma D.2. *For both types of interpolants, if f is continuous on $[0,1]$, then there is a constant C_q (different for the two types of interpolants), only depending on q, such that*

$$|\pi^{[q]} f(x)| \leq C_q |f(x)| \ \forall x \in [0,1]. \tag{D.4}$$

Proof. To estimate the mcG(q) interpolant, let $\lambda_0(x) = 1 - x$ and $\lambda_1(x) = x$. Then

$$f(x) = f_0(x) + f(0)\lambda_0(x) + f(1)\lambda_1(x),$$

where $f_0 = f - f(0)\lambda_0 - f(1)\lambda_1$, so that, since $\pi^{[q]}$ is linear,

$$|\pi^{[q]} f(x)| \quad \leq \quad |\pi^{[q]} f_0(x)| + \|f\|_{[0,1]},$$

where $\|f\|_{[0,1]} = \max_{[0,1]} |f|$. It thus suffices to estimate the interpolant of the function f_0 that is zero at the end-points. Write the interpolant as $\pi^{[q]} f_0(x) = x(1-x)p(x)$ with $p \in \mathcal{P}^{q-2}(0,1)$. Then, by the Cauchy inequality, we have

$$\begin{aligned}
\int_0^1 |\pi^{[q]} f_0(x)| \, dx &= \int_0^1 x(1-x)|p(x)| \, dx = \int_0^1 \sqrt{x(1-x)}\sqrt{x(1-x)}|p(x)| \, dx \\
&\leq \left(\int_0^1 x(1-x) \, dx\right)^{1/2} \left(\int_0^1 (x(1-x)p(x))(p(x)) \, dx\right)^{1/2} \\
&= \tfrac{1}{\sqrt{6}} \left(\int_0^1 \pi^{[q]} f_0(x) p(x) \, dx\right)^{1/2} = \tfrac{1}{\sqrt{6}} \left(\int_0^1 f_0(x)p(x) \, dx\right)^{1/2} \\
&\leq \sqrt{\tfrac{\|f_0\|_{[0,1]}}{6}} \left(\int_0^1 |p(x)| \, dx\right)^{1/2} \leq \sqrt{\tfrac{\|f_0\|_{[0,1]}}{6}} \sqrt{c_q \|\pi^{[q]} f_0\|_{[0,1]}},
\end{aligned}$$

where c_q is a constant such that

$$\max_{x\in[0,1]} |p(x)| \leq c_q \max_{x\in[0,1]} x(1-x)|p(x)| \ \forall p \in \mathcal{P}^{q-2}.$$

To obtain a value for this constant, note that the inequality holds if $p(x) = 0 \ \forall x \in [0,1]$. If not, take $x' \in (0,1)$ with $p(x') \neq 0$. Then, assuming that the maximum is attained at $x = \bar{x}$,

$$\begin{aligned}
\max_{x\in[0,1]} |p(x)| &= \tfrac{|p(\bar{x})|}{x'(1-x')|p(x')|} x'(1-x')|p(x')| \leq \tfrac{|p(\bar{x})|}{x'(1-x')|p(x')|} \max_{x\in[0,1]} x(1-x)|p(x)| \\
&\leq c_q \max_{x\in[0,1]} x(1-x)|p(x)|,
\end{aligned}$$

with

$$c_q = \frac{1}{\min_{\|p\|_{[0,1]}=1} \max_{x\in[0,1]} x(1-x)|p(x)|}.$$

The minimal maximum is obtained if the maximum of $|p|$ is placed at one of the end-points where $x(1-x) = 0$ and $|p(x)|$ decreases as fast as possible when $x(1-x)$ increases. Thus taking $p(x) = x^{q-2}$ we obtain a maximum at $x = 1 - 1/q$, giving

$$c_q = \frac{1}{(1-1/q)^{q-1}\frac{1}{q}} \leq 4q.$$

Since all norms on finite dimensional spaces are equivalent, there is a constant c_q' such that

$$\|\pi^{[q]} f_0\|_{[0,1]} \leq c_q' \int_0^1 |\pi^{[q]} f_0(x)|\, dx \leq c_q' \sqrt{\tfrac{\|f_0\|_{[0,1]}}{6}} \sqrt{c_q \|\pi^{[q]} f_0\|_{[0,1]}},$$

so that, finally,

$$\|\pi^{[q]} f_0\|_{[0,1]} \leq C_q \|f_0\|_{[0,1]},$$

with $C_q = (c_q')^2 c_q/6$.

For the mdG(q) interpolant, it suffices to estimate the interpolant of the function f_0 that is zero a the right end-point, i.e. $f_0(1) = 0$. We write the interpolant as $\pi^{[q]} f_0 = (1-x)p(x)$ with $p \in \mathcal{P}^{q-1}(0,1)$. By the Cauchy inequality we then have

$$\begin{aligned}
\int_0^1 |\pi^{[q]} f_0(x)|\, dx &= \int_0^1 (1-x)|p(x)|\, dx = \int_0^1 \sqrt{(1-x)}\sqrt{(1-x)}|p(x)|\, dx \\
&\leq \left(\int_0^1 (1-x)\, dx\right)^{1/2} \left(\int_0^1 ((1-x)p(x))(p(x))\, dx\right)^{1/2} \\
&= \tfrac{1}{\sqrt{2}} \left(\int_0^1 \pi^{[q]} f_0(x) p(x)\, dx\right)^{1/2} = \tfrac{1}{\sqrt{2}} \left(\int_0^1 f_0(x)p(x)\, dx\right)^{1/2} \\
&\leq \sqrt{\tfrac{\|f\|_{[0,1]}}{2}} \left(\int_0^1 |p(x)|\, dx\right)^{1/2} \leq \sqrt{\tfrac{\|f_0\|_{[0,1]}}{2}} \sqrt{c_q \|\pi^{[q]} f_0\|_{[0,1]}},
\end{aligned}$$

where c_q is a constant such that

$$\max_{x\in[0,1]} |p(x)| \leq c_q \max_{x\in[0,1]} (1-x)|p(x)| \ \forall p \in \mathcal{P}^{q-1}.$$

To obtain a value for this constant, note that the inequality holds if $p(x) = 0\ \forall x \in [0,1]$. If not, take $x' \in (0,1)$ with $p(x') \neq 0$. Then, again assuming that the maximum is attained at $x = \bar{x}$,

$$\begin{aligned}
\max_{x\in[0,1]} |p(x)| &= \tfrac{|p(\bar{x})|}{(1-x')|p(x')|}(1-x')|p(x')| \leq \tfrac{|p(\bar{x})|}{(1-x')|p(x')|} \max_{x\in[0,1]}(1-x)|p(x)| \\
&\leq c_q \max_{x\in[0,1]}(1-x)|p(x)|,
\end{aligned}$$

with

$$c_q = \frac{1}{\min_{\|p\|_{[0,1]}=1} \max_{x\in[0,1]}(1-x)|p(x)|}.$$

The minimal maximum is obtained if the maximum of $|p|$ is placed at the right end-point where $1-x=0$ and $|p(x)|$ decreases as fast as possible when $1-x$ increases. Thus taking $p(x) = x^{q-1}$ we obtain a maximum at $x = 1 - 1/q$, giving

$$c_q = \frac{1}{(1-1/q)^{q-1}\frac{1}{q}} \leq 4q,$$

as before. Continuing now as above for the mcG(q) interpolant, we obtain a similar estimate for the mdG(q) interpolant, completing the proof. □

The Peano kernel theorem can be stated in the following way (see e.g. [36]): For Λ a bounded linear functional that vanishes on $\mathcal{P}^q(a,b)$, define

$$K(t) = \frac{1}{q!}\Lambda(\cdot - t)_+^q, \tag{D.5}$$

where $v_+ = \max(0, v)$. If K has bounded variation and $f \in C^{q+1}([a,b])$, then

$$\Lambda f = \int_a^b K(t) f^{(q+1)}(t)\ dt. \tag{D.6}$$

We now apply this to the special choices of interpolants for the mcG(q) and mdG(q) methods discussed above.

Theorem D.3. *Assume that $f \in C^{q+1}([a,b])$. Let $k = b - a$. Then there is a function $G(x,t)$, with $|G(x,t)| \leq (1 + C_q)$, such that*

$$f(x) - \pi^{[q]} f(x) = \frac{k^{q+1}}{q!} \frac{1}{k} \int_a^b f^{(q+1)}(t) G(x,t)\ dt. \tag{D.7}$$

Furthermore, if g is an integrable (essentially) bounded function on $[a,b]$, then there is a function $H_g(t)$, with $|H_g| \leq (1 + C_q) \operatorname{ess\,sup}_{[0,1]} |g|$, such that

$$\int_a^b (f(x) - \pi^{[q]} f(x)) g(x)\ dx = \frac{k^{q+1}}{q!} \int_a^b f^{(q+1)}(x) H_g(x)\ dx. \tag{D.8}$$

These representations hold for both types of interpolants.

Proof. For the proofs, we only need the linearity, and the two lemmas D.1 and D.2. What we say therefore holds for both types of interpolants. For simplicity, we assume the interval in question is $[0,k]$.

(i) For any fixed x, define

$$\Lambda_x f = f(x) - \pi^{[q]} f(x).$$

Since $\pi^{[q]}$ is linear, this functional is linear, and by lemma D.1 it vanishes on $\mathcal{P}^q(0,k)$. By Lemma D.2,

$$|\Lambda_x f| = |f(x) - \pi^{[q]} f(x)| \leq |f(x)| + |\pi^{[q]} f(x)| \leq (1 + C_q) \max_{[0,k]} |f|,$$

so that the functional is also bounded. Furthermore,

$$K_x(t) = \frac{1}{q!} \Lambda_x (\cdot - t)_+^q = \frac{1}{q!} \left[(x-t)_+^q - \pi^{[q]} (x-t)_+^q \right],$$

so that K_x is of bounded variation. Now,

$$K_x(t) \;=\; \tfrac{1}{q!} \left[(x-t)_+^q - \pi^{[q]} (x-t)_+^q \right] = \tfrac{k^q}{q!} \left[\left(\tfrac{x-t}{k}\right)_+^q - \pi^{[q]} \left(\tfrac{x-t}{k}\right)_+^q \right] = \tfrac{k^q}{q!} G(x,t),$$

where $|G(x,t)| \leq 1 + C_q$ for $x, t \in [0,k]$. Thus, by the Peano kernel theorem,

$$f(x) - \pi^{[q]} f(x) = \Lambda_x f = \tfrac{k^q}{q!} \textstyle\int_0^k f^{(q+1)}(t) G(x,t)\ dt.$$

(ii) To prove (D.8) define

$$\Lambda f = \int_0^k (f(x) - \pi^{[q]} f(x)) g(x)\ dx.$$

The functional Λ is then linear, bounded and vanishes on $\mathcal{P}^q(0,k)$. Now,

$$\begin{aligned} K(t) &= \tfrac{1}{q!}\int_0^k \left[(x-t)_+^q - \pi^{[q]}(x-t)_+^q\right] g(x)\ dx \\ &= \tfrac{k^{q+1}}{q!}\tfrac{1}{k}\int_0^k \left[\left(\tfrac{x-t}{k}\right)_+^q - \pi^{[q]}\left(\tfrac{x-t}{k}\right)_+^q\right] g(x)\ dx = \tfrac{k^{q+1}}{q!}H_g(t), \end{aligned}$$

where $H_g(t) \leq (1+C_q)\mathrm{ess\,sup}_{[0,k]}|g|$ for $t \in [0,k]$. Thus, again by the Peano kernel theorem,

$$\int_0^k (f(x) - \pi^{[q]}f(x))g(x)\ dx = \frac{k^{q+1}}{q!}\int_0^k f^{(q+1)}(t)H_g(t)\ dt,$$

completing the proof. □

D.4. **A priori error estimates.** We now make use of the error representations and the stability estimates in order to obtain a priori error estimates for the multi-adaptive methods.

Theorem D.4. *Assume that the exact solution u has $q+1$ continuous derivatives, i.e. locally $u_i^{(q_{ij}+1)}$ is continuous on I_{ij}. Assume also that the Jacobian elements $\{J_{ij}\}$ and their derivatives are bounded by $\mathcal{A}$ on $[0,T]$, and that all K_m are small enough in terms of $\mathcal{A}$. Then the following a priori error estimate holds for the* mcG(q) *method:*

$$\|e(T)\| \leq \int_0^T (k^{2q}, w)\ dt, \tag{D.9}$$

where $w = w(u,t) \geq 0$ does not depend on k. The mcG(q) *method is thus of order $2q$.*

Proof. Starting from the error representation of Theorem D.1, noting that the residual of the exact solution is zero, we have, with $g = 0$ and $\Phi_T = \bar{e}(T)/\|\bar{e}(T)\|$,

$$\begin{aligned} \|\bar{e}(T)\| &= -\int_0^T (R(\pi_k u,\cdot),\Phi)\ dt = -\int_0^T (R(\pi_k u,\cdot) - R(u,\cdot),\Phi)\ dt \\ &= \int_0^T (f(\pi_k u,\cdot) - f(u,\cdot),\Phi)\ dt + \int_0^T (\tfrac{d}{dt}(u - \pi_k u),\Phi)\ dt. \end{aligned}$$

Since $\pi_k u$ interpolates u at the end-points of $\{I_{ij}\}$ we can integrate by parts, to get

$$\begin{aligned} \int_0^T (\tfrac{d}{dt}(u-\pi_k u),\Phi)\ dt &= \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} \tfrac{d}{dt}(u_i - \pi_k u_i)\Phi_i\ dt \\ &= \sum_{i=1}^N \sum_{j=1}^{M_i} -\int_{I_{ij}} (u_i - \pi_k u_i)\dot{\Phi}_i\ dt = 0, \end{aligned}$$

which clearly holds for $q_{ij} = 1$, since then $\dot{\Phi}_i = 0$, and also for $q_{ij} > 1$, since then $u_i - \pi_k u_i$ is orthogonal to $\mathcal{P}^{q_{ij}-2}(I_{ij})$ on I_{ij}. Linearizing, with J as in (6.1), we have

$$\begin{aligned} \int_0^T (f(\pi_k u,\cdot) - f(u,\cdot),\Phi)\ dt &= \int_0^T (J(\pi_k u,u,\cdot)(\pi_k u - u),\Phi)\ dt \\ &= \int_0^T (\pi_k u - u, J^*(\pi_k u,u,\cdot)\Phi)\ dt = \int_0^T (\pi_k u - u, B(\Phi))\ dt, \end{aligned}$$

where $B(\Phi) = J^*(\pi_k u,u,\cdot)\Phi$. We now again use the orthogonality of $\pi_k u - u$ and subtract a interpolant of B (noting that this interpolant is zero for $q=1$), to get

$$\begin{aligned} \int_0^T (f(\pi_k u,\cdot) - f(u,\cdot),\Phi)\ dt &= \int_0^T (\pi_k^{[q]}u - u, B(\Phi) - \pi_k^{[q-2]}B(\Phi))\ dt \\ &= \sum_{ij}\int_{I_{ij}} (\pi_k^{[q_{ij}]}u_i - u_i)(B_i(\Phi) - \pi_k^{[q_{ij}-2]}B_i(\Phi))\ dt \\ &= \sum_{ij} a_{ij}. \end{aligned}$$

By Theorem D.3, we have

$$\begin{array}{rcl} a_{ij} & = & \int_{I_{ij}} (\pi_k^{[q_{ij}]} u_i - u_i)(B_i(\Phi) - \pi_k^{[q_{ij}-2]} B_i(\Phi))\ dt \\ & = & \int_{I_{ij}} k_{ij}^{2q_{ij}} \frac{1}{k_{ij}} \int_{I_{ij}} G_{q_{ij}}(x,t) u_i^{(q_{ij}+1)}(x)\ dx\ \frac{1}{k_{ij}} \int_{I_{ij}} G_{q_{ij}-2}(y,t) B_i^{(q_{ij}-1)}(y)\ dy\ dt \\ & \leq & \int_{I_{ij}} C_{q_{ij}} k_{ij}^{2q_{ij}} \frac{1}{k_{ij}} \int_{I_{ij}} |u_i^{(q_{ij}+1)}(x)|\ dx\ \frac{1}{k_{ij}} \int_{I_{ij}} |B_i^{(q_{ij}-1)}(y)|\ dy\ dt \\ & \leq & \int_{I_{ij}} k_{ij}^{2q_{ij}} w_{ij}\ dt, \end{array}$$

with

$$w_i(t) = w_{ij} = C_{q_{ij}} \frac{1}{k_{ij}} \int_{I_{ij}} |u_i^{(q_{ij}+1)}(x)|\ dx\ \frac{1}{k_{ij}} \int_{I_{ij}} |B_i^{(q_{ij}-1)}(y)|\ dy,$$

for $t \in I_{ij}$. From the discussion of Section C.2 it follows that this w is bounded if all K_m are small enough. Thus, since $\pi_k u$ interpolates u at T,

$$\|e(T)\| = \|\bar{e}(T)\| \leq \sum_{ij} \int_{I_{ij}} k_{ij}^{2q_{ij}} w_{ij}\ dt = \int_0^T (k^{2q}, w)\ dt.$$

□

Theorem D.5. *Assume that the exact solution u has $q+1$ continuous derivatives, i.e. locally $u_i^{(q_{ij}+1)}$ is continuous on I_{ij}. Assume also that the Jacobian elements $\{J_{ij}\}$ and their derivatives are bounded by $\mathcal{A}$ on $[0,T]$, and that all K_m are small enough in terms of $\mathcal{A}$. Then (assuming U is right-continuous at T) the following a priori error estimate holds for the* mdG(q) *method:*

$$\|e(T)\| \leq \int_0^T (k^{2q+1}, w)\ dt, \tag{D.10}$$

where $w = w(u,t) \geq 0$ does not depend on k. The mdG(q) *method is thus of order $2q+1$.*

Proof. Starting from the error representation of Theorem D.2 in the same way as for the continuous method, noting that the residual of the exact solution is zero as well as the jump terms, we have, with $g = 0$ and $\Phi_T = \bar{e}(T)/\|\bar{e}(T)\|$,

$$\begin{array}{rcl} \|\bar{e}(T)\| & = & \sum_{i=1}^N \sum_{j=1}^{M_i} \left[-[\pi_k u_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} R_i(\pi_k u) \Phi_i\ dt \right] \\ & = & \sum_{ij} \left[-[\pi_k u_i - u_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} (R_i(\pi_k u) - R_i(u)) \Phi_i\ dt \right] \\ & = & \sum_{ij} \Big[\Big(-[\pi_k u_i - u_i]_{i,j-1} \Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} \frac{d}{dt}(\pi_k u_i - u_i) \Phi_i\ dt \Big) \\ & & \quad + \int_{I_{ij}} (f_i(\pi_k u, \cdot) - f_i(u, \cdot)) \Phi_i\ dt \Big] \\ & = & I + II. \end{array}$$

We integrate the first of these expressions by parts, using the fact that $(\pi_k u_i - u_i)$ is orthogonal to $\mathcal{P}^{q_{ij}-1}(I_{ij})$, to get

$$\begin{array}{rcl} I & = & \sum_{ij}\left[-[\pi_k u_i - u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) - \int_{I_{ij}} \frac{d}{dt}(\pi_k u_i - u_i)\Phi_i \, dt\right] \\ & = & \sum_{ij}\left[-[\pi_k u_i - u_i]_{i,j-1}\Phi_i(t_{i,j-1}^+) - [(\pi_k u_i - u_i)\Phi_i]_{t_{i,j-1}^+}^{t_{ij}^-}\right] \\ & = & \sum_{ij}\left[(\pi_k u_i - u_i)(t_{i,j-1}^-)\Phi_i(t_{i,j-1}^+) - (\pi_k u_i - u_i)(t_{ij}^-)\Phi_i(t_{ij}^-)\right] \\ & = & 0, \end{array}$$

since $\pi_k u_i$ interpolates u_i at the right end-point of every interval. Thus, linearizing, with J as in (6.1), we have

$$\begin{array}{rcl} \|\bar{e}(T)\| & = & \sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} (f_i(\pi_k u, \cdot) - f_i(u, \cdot))\Phi_i \, dt = \int_0^T (f(\pi_k u, \cdot) - f(u, \cdot), \Phi) \, dt \\ & = & \int_0^T (J(\pi_k u, u, \cdot)(\pi_k u - u), \Phi) \, dt = \int_0^T (\pi_k u - u, J^*(\pi_k u, u, \cdot)\Phi) \, dt \\ & = & \int_0^T (\pi_k u - u, B(\Phi)) \, dt, \end{array}$$

where $B(\Phi) = J^*(\pi_k u, u, \cdot)\Phi$. Continuing in the same way as for the continuous method, using the orthogonality of $\pi_k u - u$ to subtract an interpolant of $B(\Phi)$, and using Lemma D.3 to represent differences in terms of derivatives, we end up with the same estimate as before, only now we may subtract interpolants of $B(\Phi)$ that are one degree better than for the continuous method, so that now

$$\|e(T)\| = \|\bar{e}(T)\| \leq \int_0^T (k^{2q+1}, w) \, dt.$$

$\square$

Appendix E. A Stability Estimate for the Dual of the Dual

To simplify matters, we write the dual problem as

$$\left\{ \begin{array}{rcl} \dot{\psi}(t) & = & A(t)\psi(t), \\ \psi(0) & = & v, \end{array} \right. \tag{E.1}$$

where $\psi(t) = \varphi(T-t)$, $A(t) = J^*(u(T-t), U(T-t), T-t)$, $v \in \mathbb{R}^N$ is some normalized data for the dual, i.e. $\|v\| = 1$, and we assume A to be piecewise Lipschitz-continuous. The dual problem for this dual problem may then be written as

$$\left\{ \begin{array}{rcl} -\dot{\omega}(t) & = & A^*(t)\omega(t) + e_\Psi/\|e_\Psi\|, \\ \omega(T) & = & 0. \end{array} \right. \tag{E.2}$$

for error control in $L^1([0,T], \mathbb{R}^n)$ of an approximation Ψ of the dual ψ.

To prepare for the estimate we explore a few basic properties of the solution operator $E_{t_1}^A(t_2)$ of (E.1).

Lemma E.1. *Let $E_{t_1}^A(t_2)$ be the solution operator of*

$$\dot{v}(t) = A(t)v(t), \;\; t \in (0,T), \tag{E.3}$$

defined by $v(t_2) = E_{t_1}^A(t_2)v(t_1)$ for all $t_1, t_2 \in (0,T)$ with $t_1 \leq t_2$. For $t_2 < t_1$ we make the definition $E_{t_1}^A(t_2) = E_{T-t_1}^{A(T-\cdot)}(T-t_2)$. This solution operator has the following properties:

$$\left(E_{t_1}^A(t_2)\right)^* = E_{t_2}^{A^*}(t_1), \tag{E.4}$$

$$E_{t_1}^A(t_2) = E_t^A(t_2)E_{t_1}^A(t), \tag{E.5}$$

for any $t_1, t_2, t \in (0,T)$.

Proof. We give an intuitive proof based on expressing the solution operator as a product of matrix exponentials. Since A is piecewise Lipschitz-continuous, it can be approximated by some piecewise constant $\tilde{A}$ such that, for any $\epsilon > 0$, $\|\tilde{A}(t) - A(t)\| \leq \epsilon$ on $[0,T]$. By a Grönwall argument it is then clear that we may also approximate $E_{t_1}^A(t_2)$ with $E_{t_1}^{\tilde{A}}(t_2)$ arbitrarily well. It thus suffices to consider a piecewise constant A. For A piecewise constant and right-continuous, and we may take A to be piecewise constant on a uniform partition of $[t_1, t_2]$, we have, for $t_1 \leq t_2$,

$$E_{t_1}^A(t_2) = \exp(kA(t_2))\exp(kA(t_2-k))\cdots\exp(kA(t_1+k)),$$

for some $k = (t_2 - t_1)/M$. The two stated properties of A now follow directly, noting that for $t_1 > t_2$, we have

$$\begin{array}{rcl} E_{t_1}^A(t_2) & = & \exp(kA(T-(T-t_2-k)))\cdots\exp(kA(T-(T-t_1))) \\ & = & \exp(kA(t_2+k))\cdots\exp(kA(t_1)). \end{array}$$

$\square$

Lemma 6.7. *Let φ be the dual, with stability factor $S_\varphi(t)$ (i.e. with data $\|v\| = 1$ specified at time t), and let ω be the dual of the dual as defined in (E.2). Then the following estimate holds:*

$$\|\omega(t)\| \leq S_\varphi(T-t). \tag{E.6}$$

Proof. Using the Duhamel representation formula, we can write the dual of the dual as

$$\omega(t) = \int_t^T E_T^{A^*}(t)(E_T^{A^*}(s))^{-1} e_\psi(s)/\|e_\psi(s)\| \; ds,$$

so that

$$\|\omega(t)\| = \int_t^T (E_T^{A^*}(t)(E_T^{A^*}(s))^{-1} e_\psi(s)/\|e_\psi(s)\|, w_t) \; ds,$$

with $w_t = \omega(t)/\|\omega(t)\|$. Thus, using the properties of the solution operator discussed in the previous lemma, we have

$$\begin{array}{rcl} \|\omega(t)\| &=& \int_t^T (E_T^{A^*}(t)(E_T^{A^*}(s))^{-1} e_\psi(s)/\|e_\psi(s)\|, w_t) \; ds \\ &=& \int_t^T (E_s^{A^*}(t)E_T^{A^*}(s)(E_T^{A^*}(s))^{-1} e_\psi(s)/\|e_\psi(s)\|, w_t) \; ds \\ &=& \int_t^T (E_s^{A^*}(t) e_\psi(s)/\|e_\psi(s)\|, w_t) \; ds = \int_t^T (e_\psi(s)/\|e_\psi(s)\|, (E_s^{A^*}(t))^* w_t) \; ds \\ &=& \int_t^T (e_\psi(s)/\|e_\psi(s)\|, E_t^A(s) w_t) \; ds \leq \int_t^T \|E_t^A(s) w_t\| \; ds = S_{w_t}(T-t), \end{array}$$

with

$$S_{w_t}(T-t) = \int_t^T \|\psi(s)\| \; ds = \int_t^T \|\varphi(T-s)\| \; ds = \int_0^{T-t} \|\varphi(s)\| \; ds,$$

and $\varphi(T-t) = \psi(t) = w_t$. With $S_\varphi(T-t) = \sup \int_0^{T-t} \|\varphi\| \; ds$ and the supremum taken over all $w_t \in \mathbb{R}^n$ with $\|w_t\| = 1$, we thus have

$$\|\omega(t)\| \leq S_\varphi(T-t).$$

□

Chalmers Finite Element Center Preprints

2000–01 *Adaptive Finite Element Methods for the Unsteady Maxwell's Equations*
Johan Hoffman

2000–02 *A Multi-Adaptive ODE-Solver*
Anders Logg

2000–03 *Multi-Adaptive Error Control for ODEs*
Anders Logg

2000–04 *Dynamic Computational Subgrid Modeling* (Licentiate Thesis)
Johan Hoffman

2000–05 *Least-Squares Finite Element Methods for Electromagnetic Applications* (Licentiate Thesis)
Rickard Bergström

2000–06 *Discontinuous Galerkin Methods for Incompressible and Nearly Incompressible Elasticity by Nitsche's Method*
Peter Hansbo and Mats G. Larson

2000–07 *A Discountinuous Galerkin Method for the Plate Equation*
Peter Hansbo and Mats G. Larson

2000–08 *Conservation Properties for the Continuous and Discontinuous Galerkin Methods*
Mats G. Larson and A. Jonas Niklasson

2000–09 *Discontinuous Galerkin and the Crouzeix-Raviart element: Application to elasticity*
Peter Hansbo and Mats G. Larson

2000–10 *Pointwise A Posteriori Error Analysis for an Adaptive Penalty Finite Element Method for the Obstacle Problem*
Donald A. French, Stig Larson and Ricardo H. Nochetto

2000–11 *Global and Localised A Posteriori Error Analysis in the Maximum Norm for Finite Element Approximations of a Convection-Diffusion Problem*
Mats Boman

2000–12 *A Posteriori Error Analysis in the Maximum Norm for a Penalty Finite Element Method for the Time-Dependent Obstacle Problem*
Mats Boman

2000–13 *A Posteriori Error Analysis in the Maximum Norm for Finite Element Approximations of a Time-Dependent Convection-Diffusion Problem*
Mats Boman

2001–01 *A Simple Nonconforming Bilinear Element for the Elasticity Problem*
Peter Hansbo and Mats G. Larson

2001–02 *The $\mathcal{LL}^*$ Finite Element Method and Multigrid for the Magnetostatic Problem*
Rickard Bergström, Mats G. Larson, and Klas Samuelsson

2001–03 *The Fokker-Planck Operator as an Asymptotic Limit in Anisotropic Media*
Mohammad Asadzadeh

2001–04 *A Posteriori Error Estimation of Functionals in Elliptic Problems: Experiments*
Mats G. Larson and A. Jonas Niklasson

These preprints can be obtained from

`www.phi.chalmers.se/preprints`