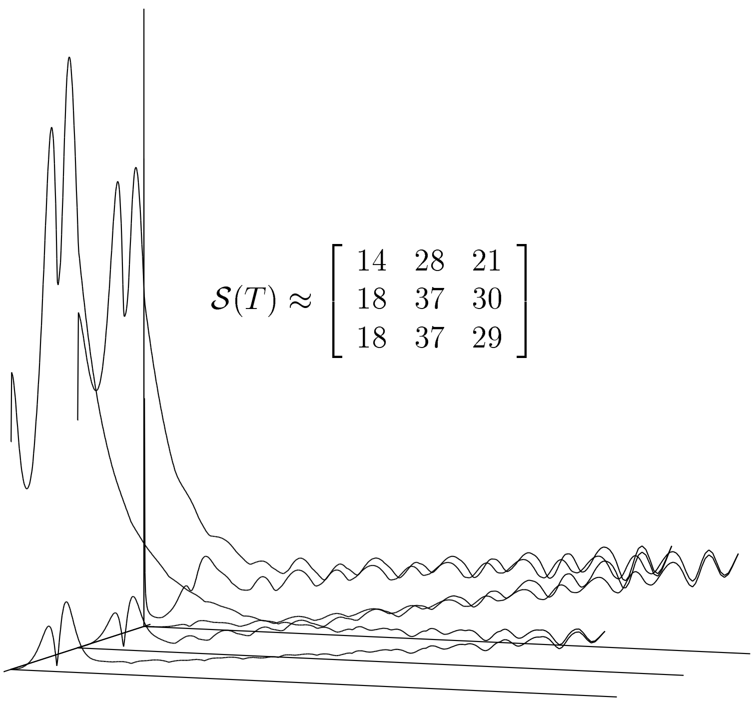
CHALMERS



$$\mathcal{S}(T) \approx \begin{bmatrix} 14 & 28 & 21 \\ 18 & 37 & 30 \\ 18 & 37 & 29 \end{bmatrix}$$

PREPRINT 2000–003

Multi-Adaptive Error Control for ODEs

Anders Logg

Chalmers Finite Element Center
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg Sweden 2000

Preprint Chalmers Finite Element Center

Multi-Adaptive Error Control for ODEs

Anders Logg

CHALMERS

Chalmers Finite Element Center
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Göteborg, March 2000

Multi-Adaptive Error Control for ODEs
Anders Logg
NO 2000–003
ISSN 0347–2809

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 2000

Multi-Adaptive Error Control for ODEs

Anders Logg

March 13, 2000

Abstract

In this report we present a *multi-adaptive* continuous Galerkin method for the solution of initial value problems for ODEs. The method is multi-adaptive in the sense that the timesteps for the different components are chosen individually and adaptively, as based on an a posteriori estimate of the maximum global error.

It is also shown how the computation can be done in *one sweep*, with fixed and correct data for the dual problem and without having to iterate and alternate between the primal and the dual problems, and also without an excessive amount of work needed for the error control.

Note

This report has been previously printed as Oxford University Computing Laboratory Technical Report 98/20 in 1998, and is now being printed again for sake of availability. This version is identical to the original report, except for a few minor changes.

Contents

1 Introduction

Initial value problems for ordinary differential equations, i.e.

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

together with some initial condition, are an important class of problems, both in their own right — arising in different fields of science (biology, chemistry, finance and, of course, physics, to name just a few) — but also considered as space discretizations of PDEs.

Thus, solving equation (1) is an important issue. The numerical method for this equation (and others) should ideally have the following properties:

> Given a tolerance $\mathrm{TOL} > 0$ and a norm $||\cdot||$, the numerical method should produce an approximation U of the true solution u, such that:
>
> - $||e|| \leq \mathrm{TOL}$, where $e = U - u$;
> - the computational cost for obtaining the approximation U at this tolerance level is minimal.

This turns out to be rather difficult in reality. Thus, traditional ODE-solvers usually work by trying to achieve the following:

> Given a norm $||\cdot||$, the numerical method should produce an approximation U of the true solution u, such that:
>
> - $||e|| \to 0$ as the amount of work goes to infinity;
> - the computational cost for obtaining the approximation U is minimal at every (unknown) tolerance level.

The concept of controlling the error e to within a given tolerance is thus thrown out the window and we cannot infer the actual size of the error! A better approach, the one that is followed in this report and is described in [2], is to do the following:

Given a tolerance $\mathrm{TOL} > 0$ and a norm $|| \cdot ||$, the numerical method should produce an approximation U of the true solution u, such that:

- $||e|| \leq \mathrm{TOL}$, where $e = U - u$;
- the computational cost for obtaining the approximation U at this tolerance level is minimal, taking into account the extra amount of work *needed* to be able to control the error.

1.1 Quantitative error control

Finite elements present a general framework for solving differential equations, such as e.g. initial value problems for ordinary differential equations, considered in this report. Depending on the choice of basis functions, normally piecewise polynomials of different kinds, the result is a new method for solving the initial value problem. These methods include $\mathrm{cG}(1)$, $\mathrm{cG}(2)$, $\ldots$, $\mathrm{dG}(0)$, $\mathrm{dG}(1)$, $\ldots$; Galerkin methods using continuous and discontinuous piecewise polynomials respectively. Using a posteriori estimates of the error, i.e. error estimates based on the computed solution, it is possible to accurately control the size of the global error.

Efficieny is obtained by *adaptivity*, investing the computational effort where it is most needed. For initial value problems this means adjusting the size of the timestep, thus choosing the timestep to be small where the solution is especially sensitive to errors in the numerical method.

Proper a posteriori error control requires knowledge of the stability of the problem. Stability properties are in general obtained by solving a so-called dual problem. Thus, error control requires some extra effort from the solver, which in most cases is comparable to the effort of solving the problem itself.

Work on quantitative error-control during the last ten years has resulted not only in extensive theoretical results, but also in working implementations of the methods, such as e.g. CARDS (solver of initial value problems for ordinary differential equations – see [8]) and FEMLAB (solver of partial differential equations).

The current approach to quantitative error control originated with the article by Johnson ([10]) in 1988, discussing error estimation for the $\mathrm{dG}(0)$ and $\mathrm{dG}(1)$ methods. Error estimation for these methods are further discussed by Estep in [5]. The $\mathrm{cG}(q)$ method, which is the basis for the multi-adaptive method presented in this report, is discussed in [6].

A comprehensive and major article on adaptive methods for differential equations is [2]. A general and non-technical discussion on error control and adaptivity is [7].

1.2 Multi-adaptivity

> *It is desirable, in short, that in things which do not primarily concern others, individuality should assert itself.*
> John Stuart Mill, On Liberty (1909).

If we view a system of ODEs as the representation of a mechanical system and notice that different parts, components, of such a system may behave very differently – some parts oscillating very rapidly and others slowly, perhaps undergoing even uniform motion – we realize that different components of an ODE-system may be differently sensitive to the resolution of the discretization. There is obviously a need for multi-adaptivity, *allowing individual components of an ODE-system to use individual timesteps*.

Traditionally, the same timestep is used for all components of an ODE-system. The novelty of multi-adaptivity is thus allowing individual adaption of the timesteps for the different components.

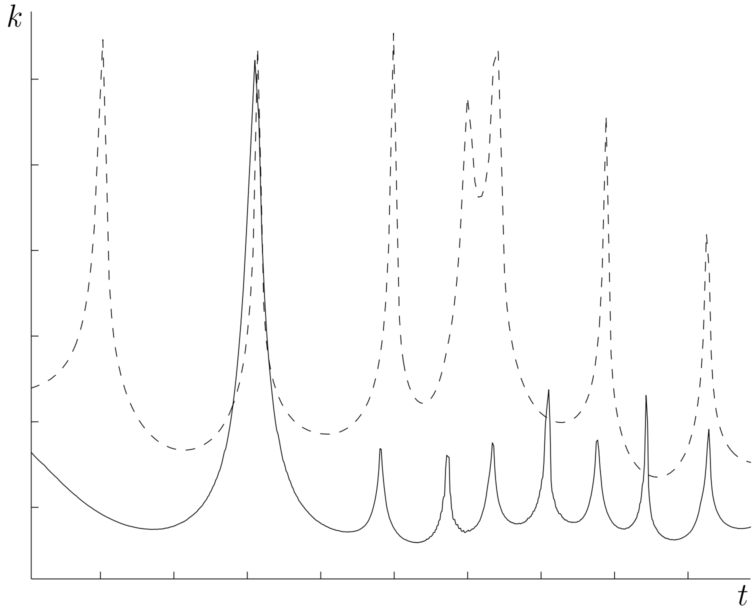


Figure 1: These are the actual timesteps used for an example computation on a simple two-dimensional system.

1.3 One-sweep computation

The a posteriori error estimate is usually of the form

$$||e|| \leq S \max\{k^q |R|\}, \tag{2}$$

where $k = k(t)$ is the timestep, q the order of the method and R the *residual*; the residual measures how well the approximate solution satisfies the differential equation:

$$R(U) = f(U, \cdot) - \dot{U}. \tag{3}$$

Notice that the residual is a known quantity, compared to the so-called truncation error, which is not.

The quantity S is a stability factor, which measures the accumulation of errors in the numerical method, and this quantity depends both on the actual equation being solved and also on the computation. It is important to notice that the size of the stability factor reflects a *global* property, which is necessary if it is supposed to say anything about the accumulation of errors.

In the following sections it will be more clear how this stability factor, or as it turns out for multi-adaptivity — *factors*, are defined, but for now we just assume the generic form (2) for the error estimate.

A difficulty arises from the fact that the stability factor is computed from solving the linearized adjoint problem, the *dual* problem. As is discussed further below, the dual cannot be solved until the solution to the primal has been computed. This means that we do not know all quantities of the error estimate (2) when solving the primal. Since we also base the choice of timesteps for the primal on this expression, we have to make an initial guess for the stability factor S, e.g. $S = 1$. Once the computation is done, we can then solve the dual and compute the value of S and thus obtain the error estimate.

The usual way this is done (see e.g. [8]) is thus to choose an initial value of S and use this for the computation. The error estimate is then compared to the tolerance and if too large, the computation is done once again with smaller timesteps. This procedure is repeated until we finally end up with an error estimate small enough.

This refinement procedure is expensive, since we have to do the computation over and over again. It would surely be nice to be able to do *one* computation of the solution, a *one-sweep* computation, starting at the beginning, ending at the end, with an error estimate smaller than and close to the tolerance.

1.4 Data for the dual

Choosing data, i.e. the data corresponding to an initial value, for the dual problem is another difficulty one has to deal with in order to get a proper error estimate. The dual problem solved in the usual way uses the *true error* as input. The true error is, of course, unknown, so we have to make a clever guess for the (normalized) value of the true error. This can be done by comparing solutions at different tolerance levels or by just assuming that the stability factors are not very sensitive to the data for the dual problem. In fact, this is presented as a conjecture in [8].

It is, however, possible to exactly specify the data for the dual problem without making any guesses. The price one has to pay is to solve more equations, a number of dual problems, with different data. As is shown below, this is not more expensive than solving one dual problem, simply because the dual is linear.

1.5 Tanganyika

The multi-adaptive method has been implemented as a library, the *Tanganyika* multi-adaptive ODE-solver library ([11]), which is available for download through

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

together with its X-interface, *Antananarive*. Though the solver is fully multi-adaptive, the current version does not approximate the stability factors in the way it is described in this report. Instead the full dual problem is solved and resolved, alternating with solutions of the primal, until the error estimate is smaller than the tolerance.

2 Multi-adaptivity

This section describes the multi-adaptive method, for which the basis is a generalization of the continuous Galerkin method, $\mathrm{cG}(q)$, described in e.g. [3].

2.1 Equation

The problem to be solved is

$$\left\{ \begin{array}{rcl} \frac{du}{dt}(t) & = & f(u,t),\ t \in (0,T], \\ u(0) & = & u_0, \end{array} \right. \tag{4}$$

where $f = (f_1, \ldots, f_N)$ is some function depending on the solution, $u = (u_1, \ldots, u_N)$, and t, which may represent time.

2.2 Method

To define the multi-adaptive $\mathrm{cG}(q)$ method, we start out from the weak (variational) formulation of equation (4) and introduce the *trial space*, V_k^N, and the *test space*, W_k^N, of functions on $[0,T]$, where

$$\begin{array}{rcl} V_k^N & = & \{v : v_i \in \mathcal{P}^{q_i}(I_{ij}), j = 1, \ldots, M_i,\ v_i \text{ is continuous},\ i = 1, \ldots, N\}, \\ W_k^N & = & \{v : v_i \in \mathcal{P}^{q_i - 1}(I_{ij}), j = 1, \ldots, M_i, i = 1, \ldots, N\}, \end{array}$$

for some set of positive integers $\{q_i\}_{i=1}^N$.

Thus $v \in V_k^N$ means that all its components $\{v_i\}_{i=1}^N$ are continuous and piecewise polynomial on the intervals $\{I_{ij}\}_{j=1}^{M_i},\ i = 1, \ldots, N$ respectively, and $v \in W_k^N$ means that all its components $\{v_i\}_{i=1}^N$ are in general discontinuous and piecewise polynomial (of one degree less) on the same intervals.

The multi-adaptive $\mathrm{cG}(q)$ method is then:

Find $U \in V_k^N$ such that $U(0) = u_0$ and

$$\int_0^T (\dot{U}, v) = \int_0^T (f(U, \cdot), v) \quad \forall v \in W_k^N. \tag{5}$$

The discontinuity of the test functions means we may rewrite this as

Find $\{\xi_{ijk}\}_{k=0}^{q_i}$ such that:

$$\int_{I_{ij}} \dot{U}_i v = \int_{I_{ij}} f_i(U,\cdot)v \quad \forall v \in \mathcal{P}^{q_i-1}(I_{ij}), \ \ j=1,\dots,M_i, i=1,\dots,N, \tag{6}$$

$U(0) = u_0$ and U is continuous,

where the $\{\xi_{ijk}\}$ are the parameters determining the piecewise polynomials $\{U_i\}$. Note that there are $(q+1)$ parameters determining a polynomial of degree q, so the index k is from zero to q.

Finding the parameters $\{\xi_{ijk}\}$ in agreement with (6) yields the desired solution. What remains is to find the proper discretization $\{I_{ij}\}$, or, equivalently, the *timesteps* $\{k_{ij}\}$. To choose the timesteps, we need an error estimate, which will be the basis for the adaptivity. By means of this error estimate, the discretization will be chosen in a way to give a resulting final error smaller than the specified tolerance.

2.2.1 Details

The parameters $\{\xi_{ijk}\}$ may e.g. be the nodal values for a subdivision of the intervals into q_i subintervals. For an interval I_{ij}, let the nodal points of an equipartition of this interval be $\{t_{ijk}\}_{k=0}^{q_i}$. The corresponding nodal (Lagrange) basis functions, $\{\lambda_{ijk} : \mathbf{R} \to \mathbf{R}\}$, are then defined on I_{ij} for $k = 0,\dots,q_i$, by

$$\lambda_{ijk}(t) = \frac{(t-t_{ij0})\cdots(t-t_{ij,k-1})(t-t_{ij,k+1})\cdots(t-t_{ijq_i})}{(t_{ijk}-t_{ij0})\cdots(t_{ijk}-t_{ij,k-1})(t_{ijk}-t_{ij,k+1})\cdots(t_{ijk}-t_{ijq_i})}. \tag{7}$$

On the interval I_{ij}, U_i may then be written (uniquely) as

$$U_i = \sum_{k=0}^{q_i} \xi_{ijk}\lambda_{ijk}, \tag{8}$$

for some values $\{\xi_{ijk}\}$.

Inserting this into (6), computing a few integrals (simple but tedious) and solving the resulting system of linear algebraic equations, yields

$$\left\{ \begin{array}{rcl} \xi_{ij1} &=& \xi_{ij0} + \int_{ij} w_{q_i1}(\tau_{ij}(t)) f_i(U,t)dt, \\ \xi_{ij2} &=& \xi_{ij0} + \int_{ij} w_{q_i2}(\tau_{ij}(t)) f_i(U,t)dt, \\ \vdots && \\ \xi_{ijq_i} &=& \xi_{ij0} + \int_{ij} w_{q_iq_i}(\tau_{ij}(t)) f_i(U,t)dt, \end{array} \right. \tag{9}$$

$w_{11}(\tau) = 1$		
$w_{21}(\tau) = \frac{1}{4}(5 - 6\tau)$	$w_{22}(\tau) = 1$	
$w_{31}(\tau) = \frac{1}{27}(37 - 96\tau + 60\tau^2)$	$w_{32}(\tau) = \frac{1}{27}(26 + 24\tau - 60\tau^2)$	$w_{33}(\tau) = 1$

Table 1: Weight functions for the cG(q) integrals, $q = 1, 2, 3$.

where $\tau_{ij}(t) = \frac{t - t_{i,j-1}}{t_{ij} - t_{i,j-1}}$ and the $\{w_{qk}\}_{k=1}^{q}$ are polynomial weight functions. These are given in Table 1 for $q = 1, 2, 3$.

Note that the weight functions $\{w_{qq}\}$ are unity, which is of course a direct consequence of equation (6), choosing $v = 1$.

2.2.2 Even more flexibility

Note that we could have allowed each component to be piecewise polynomial without beforehand fixing the deqree of the polynomial on the whole of the discretization. We could thus have allowed the polynomial degree to change from one interval to the next. The method would then be even p-adaptive, choosing (in some sense) the best degree of the polynomials for every single interval I_{ij}.

For simplicity, though, the polynomial degrees have been chosen to be $\{q_i\}$ rather than $\{q_{ij}\}$. The difference would be an extra index j on q.

2.3 Individual timestepping in practice

In this section are described details specific to the Tanganyika implementation of the multi-adaptive method. The following is thus a description of one way of implementing the multi-adaptivity and how to actually solve the discrete equations.

The individual stepping is done according to equations (9). This requires knowledge about U, including the values of all other components. These values are evaluated by interpolation (or *extra*polation), according to the order of the method, of the nearest known values of the other components. The solution of the integral equation is performed iteratively for every component. In the Tanganyika library, this is done by fixed point iteration on the equations (9).

The order of the stepping follows one simple principle;

the last component steps first.

The fact that the equations are not solved simultaneously results in non-zero discrete residuals, i.e. the residuals of the discrete equations (9). Since other components are stepped after the values of a certain component have been updated, the values used for updating that component may have changed, and therefore also the information used for the step. Thus the equations (9) might not be solved properly. By iteration over the different components, by choosing smaller timesteps or by some other method, we may be able to do better. Anyhow, the error introduced is measured properly by incorporating the discrete residual in our error estimate.

2.3.1 Organization

Doing the stepping individually rather than stepping all components together requires some book-keeping, keeping track of the positions of all components and which one is to step next.

The individual stepping is done according to figure 2 below. The implementation pretty much follows this sketch.

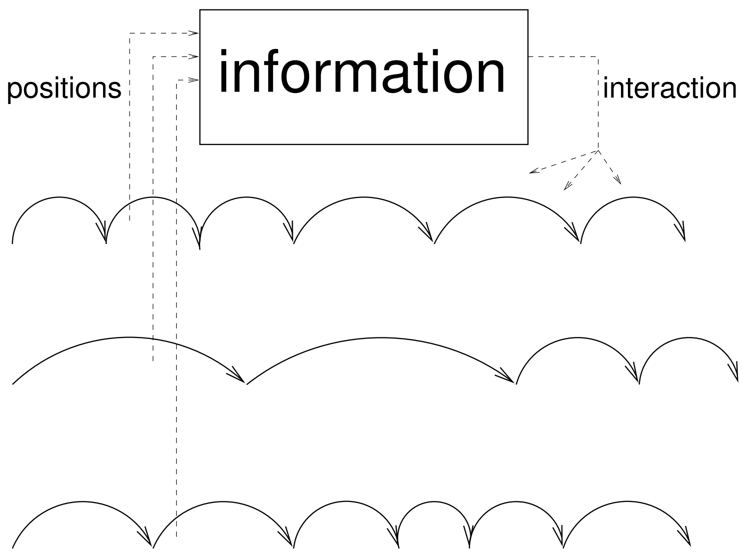


Figure 2: This is how the individual stepping is done. The different components tell/send their respective positions and in turn they get their interactions with (forces from) the other components. Thus, just as in nature itself, progress is made by the exchange of information, small pieces of information (gravitons or perhaps *femions*).

All this can be achieved rather nicely and simply in an object-oriented language such as C++, which is the language of the Tanganyika library.

2.3.2 Quadrature

The integrals of (9) are evaluated by Gaussian (Gauss-Legendre) quadrature.

Since the order of the weight functions for the integrals of a $\mathrm{cG}(q)$ method are $(q-1)$, we expect the total order of the integrands to be of order $q+(q-1)=2q-1$ (and even more if f is of quadratic or higher order). It would thus be wise to use quadrature that is exact at least for polynomials of order $2q-1$, which is exactly the case for Gaussian quadrature with q nodal points.

Thus, midpoint quadrature for $\mathrm{cG}(1)$, two-point Gaussian quadrature for $\mathrm{cG}(2)$ and so on.

3 Error analysis

The error estimate is obtained starting out in the same way as in references [1], [3] and [12].

To estimate the error at final time T in the l^2-norm, the *dual* problem of (4) is introduced. The dual problem is

$$\left\{ \begin{array}{rcl} -\frac{d\varphi}{dt}(t) & = & J^*(u,U,t)\varphi(t),\ t \in [0,T), \\ \varphi(T) & = & \varphi_T, \end{array} \right. \tag{10}$$

where J^* is defined as

$$J^*(u,U,\cdot) = \left(\int_0^1 \frac{\partial f}{\partial u}(su+(1-s)U,\cdot)ds \right)^*, \tag{11}$$

i.e. J^* is the transpose (or more generally, the adjoint) of the Jacobian of f at a mean value of u and U. The data for the dual problem is φ_T and we wait until later to choose the value for φ_T.

Note now that by the chain rule,

$$\begin{array}{rcl} -J(u,U,\cdot)(U-u) & = & \int_0^1 \frac{\partial f}{\partial u}(su+(1-s)U,\cdot)ds(u-U) \\ & = & \int_0^1 \frac{\partial f}{\partial s}(su+(1-s)U,\cdot)ds \\ & = & f(u,\cdot)-f(U,\cdot). \end{array} \tag{12}$$

We may thus write

$$
\begin{array}{rcl}
(e(T),\varphi_T) & = & (e(T),\varphi(T)) - (e(0),\varphi(0)) + \int_0^T (e, -\dot{\varphi} - J^*(u,U,\cdot)\varphi)ds \\
& = & [(e(t),\varphi(t))]_0^T - \int_0^T (e,\dot{\varphi})ds - \int_0^T (e, J^*(u,U,\cdot)\varphi)ds \\
& = & \int_0^T (\dot{e},\varphi)ds - \int_0^T (J(u,U,\cdot)e,\varphi)ds \\
& = & \int_0^T (\dot{e} - J(u,U,\cdot)e,\varphi)ds \\
& = & \int_0^T (\dot{U} - f(u,\cdot) - J(u,U,\cdot)(U-u),\varphi)ds \\
& = & \int_0^T (\dot{U} - f(U,\cdot),\varphi)ds \\
& = & -\int_0^T (R,\varphi)ds,
\end{array}
\tag{13}
$$

where R is the residual, i.e.

$$
R = f(U,\cdot) - \dot{U}. \tag{14}
$$

Using the finite element formulation for some $\overline{\varphi} \in W_k^N$, we continue to get

$$
\begin{array}{rcl}
|(e(T),\varphi_T)| & = & |\int_0^T (R,\varphi - \overline{\varphi})ds| \\
& = & |\sum_{i=1}^N \int_0^T R_i(\varphi_i - \overline{\varphi}_i)ds| \\
& = & |\sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(\varphi_i - \overline{\varphi}_i)ds| \\
& \leq & \sum_{i=1}^N \sum_{j=1}^{M_i} \sup_{I_{ij}} |R_i| \int_{I_{ij}} |\varphi_i - \overline{\varphi}_i|ds.
\end{array}
\tag{15}
$$

This is alright as long as U solves equation (5). If, however, U fails to solve the discrete equation (5), which may be rewritten as

$$
\int_0^T (R,v)\, ds = 0 \;\forall v \in W_k^N, \tag{16}
$$

we have to make a correction for this in our error estimate.

To measure this correction we introduce the *discrete residual*, which should be zero if the discrete equations were solved properly. The following analysis will result in an extra term in the error estimate (15), including the discrete residual together with its proper stability factor, accounting for accumulation of errors due to a non-zero discrete residual.

Defining the discrete residual to be

$$\overline{R}_i(t) = \overline{R}_{ij} = \frac{1}{k_{ij}}\left[\int_{I_{ij}} f_i(U,\cdot)ds - (\xi_{ijq} - \xi_{ij0})\right],\ t \in I_{ij}, j = 1,\dots,M_i,\ i = 1,\dots,N, \tag{17}$$

we get for $\overline{\varphi}_i \in \mathcal{P}^{q_i-1}(I_{ij})$ and some $\eta_{ij} \in I_{ij}$,

$$\int_{I_{ij}} R_i\overline{\varphi}_i ds = \overline{\varphi}_i(\eta_{ij})\int_{I_{ij}} (\dot{U}_i - f_i(U,\cdot))ds = -\overline{\varphi}_i(\eta_{ij})k_{ij}\overline{R}_{ij}. \tag{18}$$

Thus, $\int_{I_{ij}} R_i\overline{\varphi}_i ds$ differs from zero and we get an additional term in our error estimate, continuing from eq (13):

$$\begin{array}{rcl} |(e(T),\varphi_T)| & = & |\int_0^T (R,\varphi)ds| \\ & = & |\sum_{i=1}^N \int_0^T R_i\varphi_i ds| \\ & = & |\sum_{i=1}^N \sum_{j=1}^{M_i}\left[\int_{I_{ij}} R_i\varphi_i ds - \int_{I_{ij}} R_i\overline{\varphi}_i ds - \overline{\varphi}_i(\eta_{ij})k_{ij}\overline{R}_{ij}\right]| \\ & = & |\sum_{i=1}^N \sum_{j=1}^{M_i}\left[\int_{I_{ij}} R_i(\varphi_i - \overline{\varphi}_i)ds - \overline{\varphi}_i(\eta_{ij})k_{ij}\overline{R}_{ij}\right]| \\ & \leq & \sum_{i=1}^N \sum_{j=1}^{M_i}\left[\sup_{I_{ij}}|R_i|\int_{I_{ij}}|\varphi_i - \overline{\varphi}_i|ds + |\overline{R}_{ij}|k_{ij}\sup_{I_{ij}}|\overline{\varphi}_i|\right], \end{array} \tag{19}$$

which is the same as (15) if the discrete residual is zero.

From this point, we may continue in two different ways. We may either want to use the fact that we may write

$$\int_{I_{ij}} |\varphi_i - \overline{\varphi}_i|ds \leq C_{q_i}k_{ij}^{q_i}\int_{I_{ij}} |\varphi_i^{(q)}|ds, \tag{20}$$

introduce the *stability factors*

$$\begin{array}{rcl} S_i(T) & = & \int_0^T |\varphi_i^{(q_i)}|ds,\ i = 1,\dots,N, \\ \overline{S}_i(T) & = & \sum_{j=1}^{M_i} k_{ij}\sup_{I_{ij}}|\overline{\varphi}_i|, i = 1,\dots,N, \end{array} \tag{21}$$

and write the error as

$$|(e(T),\varphi_T)| \leq \sum_{i=1}^N \left(C_iS_i(T)\max_{[0,T]}\{k^{q_i}|R_i|\} + \overline{S}_i(T)\max_{[0,T]}|\overline{R}_i|\right). \tag{22}$$

Or we may wish to write

$$\int_{I_{ij}} |\varphi_i - \overline{\varphi}_i| ds \leq \tilde{C}_{q_i} k_{ij}^{q_i+1} \sup_{I_{ij}} |\varphi_i^{(q)}|, \tag{23}$$

introduce the *stability functions*

$$\begin{array}{rcl} s_i(t) & = & s_{ij} = \sup_{I_{ij}} |\varphi_i^{(q_i)}|, \; t \in I_{ij}, \; j = 1, \ldots, M_i, \; i = 1, \ldots, N, \\ \overline{s}_i(t) & = & \overline{s}_{ij} = \sup_{I_{ij}} |\overline{\varphi}_i|, \; t \in I_{ij}, \; j = 1, \ldots, M_i, \; i = 1, \ldots, N, \end{array} \tag{24}$$

and keep these as weights:

$$|(e(T), \varphi_T)| \leq \sum_{i=1}^{N} \sum_{j=1}^{M_i} \left(\tilde{C}_{q_i} s_{ij} k_{ij}^{q_i+1} \sup_{I_{ij}} |R_i| + \overline{s}_{ij} k_{ij} |\overline{R}_{ij}| \right). \tag{25}$$

This latter expression is used for error control in the multi-adaptive ODE-solver *Tanganyika* described in [11]. The way we shall do it in this report and the way it is done in CARDS (see [8]) is to use (22), since this is less expensive.

Note that in both cases the different errors have different stability factors, since these errors in general accumulate at different rates.

What remains now is to compute the constants C_q and $\tilde{C}_q$, and to choose φ_T, the data for the dual problem. This will be done in the following two sections.

3.1 The constants C_q and $\tilde{C}_q$

Since we are free to choose the test function $\overline{\varphi}$ as any function in the test space W_k^N, we want to choose the one that in general gives the best, the sharpest, bound on the error. Rather than tackling the problem of finding the best approximations in general, yielding the smallest values for the constants, we show how to compute simple upper bounds for these constants. These bounds are perhaps not the best possible bounds, but they are good enough.

Choosing the test function $\overline{\varphi}$ as the $(q-1)$th-order Taylor-expansion of φ around $(t_{ij} + t_{i,j-1})/2$ on I_{ij} yields

$$C_q = \frac{1}{q! 2^{q-1}}. \tag{26}$$

The proof is simple. Noting that, with

$$\overline{f}_{q-1}(x) = f(x_0) + f'(x_0)(x - x_0) + \ldots + \frac{1}{(q-1)!} f^{(q-1)}(x_0)(x - x_0)^{(q-1)} \tag{27}$$

we have

$$|f(x) - \overline{f}_{q-1}(x)| = |\frac{1}{(q-1)!} \int_{x_0}^{x} f^{(q)}(y)(y - x_0)^{(q-1)} dy| \tag{28}$$

and thus, with $x_0 = (a + b)/2$,

$$\begin{aligned} \int_a^b |f - \overline{f}_{q-1}| dx &= \tfrac{1}{(q-1)!} \int_a^b | \int_{x_0}^x f^{(q)}(y)(y - x_0)^{(q-1)} dy| dx \\ &\leq \tfrac{1}{(q-1)!} \left(\int_a^b |x - x_0|^{q-1} dx \right) \left(\int_a^b |f^{(q)}(x)| dx \right) \\ &= \tfrac{1}{q!} \left(|a - x_0|^q + |b - x_0|^q \right) \int_a^b |f^{(q)}| dx \\ &= \tfrac{2(b-a)^q}{q! 2^q} \int_a^b |f^{(q)}| dx \\ &= \tfrac{(b-a)^q}{q! 2^{q-1}} \int_a^b |f^{(q)}| dx. \end{aligned} \tag{29}$$

Estimating $\tilde{C}_q$ is done in the same way:

$$\int_a^b |f - \overline{f}_{q-1}| dx \leq \tilde{C}_q (b - a)^{q+1} \sup_{(a,b)} |f^{(q)}|, \tag{30}$$

where

$$\tilde{C}_q = \frac{1}{(q+1)! 2^q}, \tag{31}$$

which is obtained as above, choosing $\overline{f}_{q-1}$ to be the $(q-1)$th-order Taylor expansion around the midpoint.

3.2 Choosing data for the dual problem

Note now that the expressions derived for the bound on the error hold for every choice of data for the dual problem, provided the stability properties are computed for the same choice of data. What remains is to choose the data in such a way that the left-hand side, i.e. $|(e(T), \varphi_T)|$, tells us anything about the error.

The way this is usually done is to choose the data to be the normalized error, i.e.

$$\varphi_T = e(T)/||e(T)||_{l^2}, \tag{32}$$

resulting in l^2 norm error control. The difficulty with this approach is obvious: the error is not known, for otherwise we wouldn't have to compute a bound for it.

Another possibility is to choose the data to be $\mathbf{1}_n$, i.e. a unit vector with the nth entry equal to one and all the others zero.

The expression for the error estimate is then:

$$|e_n(T)| \leq \sum_{i=1}^{N} \left(C_i S_i^n(T) \max_{[0,T]} \{k^{q_i}|R_i|\} + \overline{S}_i^n(T) \max_{[0,T]} |\overline{R}_i| \right), \ n = 1, \ldots, N, \tag{33}$$

where the n superscript denotes the stability factors computed for $\varphi_T = \mathbf{1}_n$. To control all the individual component errors, we thus need to solve N *different dual problems*. This however, does not have to be more expensive than solving one dual problem, as will be shown below.

Note now that T does not necessarily have to be the end time value. For error control in the *maximum norm*,

$$||e_n||_\infty = \max_{[0,T]} |e_n|, \tag{34}$$

our final expression for the error estimate is then:

$$||e_n||_\infty \leq \max_{t \in [0,T]} \sum_{i=1}^{N} \left(C_{q_i} S_i^n(t) \max_{[0,t]} \{k^{q_i}|R_i|\} + \overline{S}_i^n(t) \max_{[0,t]} |\overline{R}_i| \right), \ n = 1, \ldots, N, \tag{35}$$

where now the data is specified at time t rather than at T.

3.3 Other error contributions

Other error contributions that are not dealt with here are quadrature errors and numerical errors due the finite precision arithmetic. Note however that the quadrature errors could have been dealt with in the same way as the non-zero discrete residual.

4 Adaptivity

Adaptivity is based on the expression

$$||e|| \leq \text{error estimate} \leq \text{TOL}, \tag{36}$$

where TOL is a given tolerance for the error.

Now let $\text{TOL} = (\text{TOL}_1, \text{TOL}_2, \ldots, \text{TOL}_N)$ be given individual tolerances. What we want to do is to choose the timesteps $\{k_{ij}\}$ such that:

$$||e_n||_\infty \leq \max_{t\in[0,T]} \sum_{i=1}^{N} \left(C_{q_i} S_i^n(t) \max_{[0,t]}\{k^{q_i}|R_i|\} + \overline{S}_i^n(t) \max_{[0,t]} |\overline{R}_i| \right) \leq \text{TOL}_n, \; n = 1, \ldots, N. \tag{37}$$

If we define the *local tolerances* r and $\overline{r}$ by

$$\begin{array}{rcl} C_{q_i} \max_{[0,t]}\{k^{q_i}|R_i|\} & \leq & r_i(t), \; t \in (0,T], \; i = 1, \ldots, N, \\ \max_{[0,t]} |\overline{R}_i| & \leq & \overline{r}_i(t), \; t \in (0,T], \; i = 1, \ldots, N, \end{array} \tag{38}$$

and the stability matrices $\mathcal{S}$ and $\overline{\mathcal{S}}$ by

$$\begin{array}{rcl} \mathcal{S}_{ni}(t) & = & S_i^n(t), \; t \in (0,T], \; i = 1, \ldots, N, \; n = 1, \ldots, N, \\ \overline{\mathcal{S}}_{ni}(t) & = & \overline{S}_i^n(t), \; t \in (0,T], \; i = 1, \ldots, N, \; n = 1, \ldots, N, \end{array} \tag{39}$$

we can write this as

$$\mathcal{S}r + \overline{\mathcal{S}}\overline{r} \leq \text{TOL}. \tag{40}$$

This expression should hold on $[0,T]$. Effectively, this means that the local tolerances at every time t should be chosen in order for this expression to hold with the values of $\mathcal{S}$ and $\overline{\mathcal{S}}$ on the interval $[t,T]$.

Knowing thus the residuals and the stability factors we can choose the proper timesteps, e.g. by choosing

$$\begin{array}{rcl} \mathcal{S}r & \leq & \text{TOL}/2, \\ \overline{\mathcal{S}}\overline{r} & \leq & \text{TOL}/2. \end{array} \tag{41}$$

5 Approximation of the stability factors

In this section, we describe how approximations of the stability factors (the stability matrices) can be computed with little effort.

For simplicity and brevity, we consider here only the case $q = 1$ and the computation of $\mathcal{S}$.

To begin with, we give a short review of some properties for matrix exponentials, since these will be used to compute the stability factors.

5.1 Matrix exponentials

The matrix exponential : $\mathbf{R}^{N\times N} \to \mathbf{R}^{N\times N}$ is defined by

$$\mathrm{e}^{A} = \sum_{n=0}^{\infty} \frac{A^n}{n!}, \tag{42}$$

where A^0 is the identity matrix.

Now recall the following properties of the matrix exponential:

$$\begin{aligned}
\mathrm{e}^{\mathbf{0}} &= I, && (43)\\
\left(\mathrm{e}^{A}\right)^{-1} &= \mathrm{e}^{-A}, && (44)\\
\mathrm{e}^{A}\mathrm{e}^{B} &= \mathrm{e}^{A+B}, \text{ if } A \text{ and } B \text{ commute }, && (45)\\
\frac{d}{dt}(\mathrm{e}^{tA}) &= \mathrm{e}^{tA}A = A\mathrm{e}^{tA}, && (46)
\end{aligned}$$

where A and B are constant matrices, I is the identity matrix, $\mathbf{0}$ the zero matrix and t is a scalar.

These properties all follow easily from the definition (42).

5.1.1 Numerical approximation

The cost of numerically computing the matrix exponential of a matrix is essentially the same as the cost of computing its inverse. The method used in our numerical experiments is essentially Algorithm 11.3.1 in [9]; prescaling, computing a Padé approximation and then rescaling.

5.2 Solving the dual

The equation to be solved, the dual problem (10), can be rewritten as

$$
\begin{cases} -\frac{d\varphi}{dt}(t) & = & A(t)\varphi(t), \ t \in [0,T), \\ \varphi(T) & = & \varphi_T, \end{cases} \tag{47}
$$

where we make the approximation

$$
A = \left(\frac{\partial f}{\partial u}(U, \cdot)\right)^*, \tag{48}
$$

rather than

$$
A = \left(\int_0^1 \frac{\partial f}{\partial u}(su + (1-s)U, \cdot) ds\right)^*, \tag{49}
$$

since the true solution is unknown.

In the case where A is a constant matrix, we can immediately — using the properties stated above for the matrix exponential — write down the solution as

$$
\varphi(t) = \mathrm{e}^{(T-t)A}\varphi_T. \tag{50}
$$

For time-dependent A, viewing A as piecewise constant and passing to the limit (or otherwise), the solution can be written as

$$
\varphi(t) = \lim_{n\to\infty} \Pi_{i=1}^n \mathrm{e}^{k_{ni}A_{ni}} \varphi_T, \tag{51}
$$

where the $\{k_{ni}\}_{i=1}^n$ are positive weights such that

$$
\begin{array}{rcl} \sum_{i=1}^n k_{ni} & = & T - t, \ \forall n \in \mathbf{Z}_+ \text{ and} \\ A_{ni} & = & A(t + \sum_{j=1}^i k_{nj}), \ i = 1, 2, \ldots, n, \ \forall n \in \mathbf{Z}_+. \end{array} \tag{52}
$$

Writing the product as a sum in the exponent, it may be tempting to write this expression as a generalization of (50):

$$
\varphi(t) = \mathrm{e}^{\int_t^T A(s)ds}\varphi_T. \tag{53}
$$

This, however, **is not true**! Though the exponents in two (or any finite number of) adjacent factors in the above product commute approximately

— A_n is close to A_{n+1} — this is not true in the limit. Assume that this was indeed the case. Then we could divide the product into two parts and replace the product with a sum in the exponent for both of them. The remaining product of two matrix exponentials can then not be replaced by the matrix exponential of the sum of the two exponents, since these two may not commute — they may be virtually anything.

For ease of notation we shall write the solution (51) as

$$\varphi(t) = \Pi_t^T \mathrm{e}^{A(s)ds} \varphi_T. \tag{54}$$

5.3 Stability matrices

The $\mathrm{cG}(1)$ stability matrix $\mathcal{S}$ is given by

$$\mathcal{S}_{ni}(t) = S_i^n(t) = \int_0^t |\dot{\varphi}_i| dx, \tag{55}$$

where $\varphi(t) = \mathbf{1}_n$. Differentiating the expression for the dual solution, we obtain the following expression for the stability matrix:

$$\begin{array}{rcl} \mathcal{S}_{ni}(t) & = & \int_0^t |\mathbf{1}_i^* \dot{\varphi}(x)| dx \\ & = & \int_0^t |\mathbf{1}_i^* \dot{\varphi}(x)| dx \\ & = & \int_0^t |\mathbf{1}_i^* (-A(x) \Pi_x^t \mathrm{e}^{A(y)dy} \mathbf{1}_n)| dx \\ & = & \int_0^t |\mathbf{1}_i^* A(x) \Pi_x^t \mathrm{e}^{A(y)dy} \mathbf{1}_n| dx. \end{array} \tag{56}$$

Now, let $|A|$ denote the element-wise modulus of a matrix A. The expression we obtain for the stability matrix $\mathcal{S}$ is then

$$\mathcal{S}(t) = \left(\int_0^t |A(x) \Pi_x^t \mathrm{e}^{A(y)dy}| dx \right)^*. \tag{57}$$

The way the approximation is done, is by direct application of this expression for the stability matrix, using finite timesteps $\{k_i\}$. The approximation we make is thus

$$\mathcal{S}(t_n) \approx \left(\sum_{i=1}^n k_i |A(t_i) \Pi_{j=i}^n \mathrm{e}^{k_j A_j}| \right)^*, \ n = 1, \ldots, M, \tag{58}$$

for the positive sequence of numbers $\{k_i\}_{i=1}^M$. In reality though, some quadrature other than this is applied. This is described in more detail in the sections below.

5.4 Error analysis

We analyze the error of the approximation (58) for the case of using the midpoint values of A in every interval for the finite products and the finite sum.

Notice that we do not claim that these estimates are optimal. Normally, one wouldn't even try to do error estimation for the solution of the dual. However, in order to control the error of the solution itself, it is obvious that we also need to control the error of the solution to the dual, and this is one attempt of doing so.

The error can be divided into two parts, the first one being the error from not solving the dual problem properly (not computing the infinite product) and the second one from not computing the integral for the stability matrix properly (not computing the infinite sum).

Now let $\tilde{A}$ be the piecewise constant interpolant of A that takes the midpoint value of A in every interval, i.e.

$$\tilde{A}(t) = A_i = A(t_i - k_i/2),\ t \in I_i,\ i = 1, \dots, M, \tag{59}$$

and define B and $\tilde{B}$ as

$$\begin{array}{rcl} B(x,t) & = & A(x)\Pi_x^t \mathrm{e}^{A(y)dy},\ x \in [0,t),\ t \in (0,T], \\ \tilde{B}(x,t) & = & \tilde{A}(x)\Pi_x^t \mathrm{e}^{\tilde{A}(y)dy},\ x \in [0,t),\ t \in (0,T]. \end{array} \tag{60}$$

We can then write the error as

$$|\mathcal{S} - \tilde{\tilde{\mathcal{S}}}\ | \leq |\mathcal{S} - \tilde{\mathcal{S}}| + |\tilde{\mathcal{S}} - \tilde{\tilde{\mathcal{S}}}\ |, \tag{61}$$

where

$$\begin{array}{rcl} \mathcal{S}(t) & = & \left(\int_0^t |B(x,t)|dx\right)^*,\ t \in (0,T], \\ \tilde{\mathcal{S}}(t) & = & \left(\int_0^t |\tilde{B}(x,t)|dx\right)^*,\ t \in (0,T], \\ \tilde{\tilde{\mathcal{S}}}\ (t_n) & = & \left(\sum_{i=1}^n |\tilde{B}(x_i - k_i/2, t_n)|k_i\right)^*,\ n = 1, \dots, M. \end{array} \tag{62}$$

5.4.1 The first part of the error

The first part of the error is the error from approximating the infinite product. We estimate this error as

$$
\begin{array}{rcl}
|\mathcal{S}-\tilde{\mathcal{S}}|^*(t) & = & |\int_0^t |B(x,t)|dx - \int_0^t |\tilde{B}(x,t)|dx| \\
& \leq & \int_0^t |B(x,t)-\tilde{B}(x,t)|dx \\
& = & \int_0^t |A(x)\Pi_x^t \mathrm{e}^{A(y)dy} - \tilde{A}(x)\Pi_x^t \mathrm{e}^{\tilde{A}(y)dy}|dx \\
& \leq & \int_0^t |\left(A(x)-\tilde{A}(x)\right)\Pi_x^t \mathrm{e}^{A(y)dy}|dx \\
& & + \int_0^t |\tilde{A}(x)\left(\Pi_x^t \mathrm{e}^{A(y)dy} - \Pi_x^t \mathrm{e}^{\tilde{A}(y)dy}\right)|dx.
\end{array} \tag{63}
$$

Notice now that (for A and $\tilde{A}$ piecewise continuous on $(0,1)$) it follows by repeated addition and subtraction that

$$
\begin{array}{rcl}
\Pi_0^1 \mathrm{e}^{A(y)dy} - \Pi_0^1 \mathrm{e}^{\tilde{A}(y)dy} & = & \lim_{n\to\infty} \Pi_{i=1}^n \mathrm{e}^{\frac{1}{n}A(i/n)} - \lim_{n\to\infty} \Pi_{i=1}^n \mathrm{e}^{\frac{1}{n}\tilde{A}(i/n)} \\
& = & \lim_{n\to\infty} \sum_{i=1}^n \Pi_{j=1}^{i-1} \mathrm{e}^{\frac{1}{n}\tilde{A}(j/n)} (\mathrm{e}^{\frac{1}{n}A(i/n)} - \mathrm{e}^{\frac{1}{n}\tilde{A}(i/n)}) \Pi_{j=i+1}^n \mathrm{e}^{\frac{1}{n}A(j/n)} \\
& = & \int_0^1 \Pi_0^x \mathrm{e}^{\tilde{A}(y)dy} \left(A(x)-\tilde{A}(x)\right) \Pi_x^1 \mathrm{e}^{A(y)dy} dx,
\end{array} \tag{64}
$$

where we have chosen the interval to be $(0,1)$ rather than (x,t) for ease of notation.

We thus have

$$
\begin{array}{rcl}
|\mathcal{S}-\tilde{\mathcal{S}}|^*(t) & \leq & \int_0^t |\left(A(x)-\tilde{A}(x)\right)\Pi_x^t \mathrm{e}^{A(y)dy}|dx \\
& & + \int_0^t |\tilde{A}(x)\int_x^t \Pi_x^y \mathrm{e}^{\tilde{A}(z)dz} \left(A(y)-\tilde{A}(y)\right) \Pi_y^t \mathrm{e}^{A(z)dz} dy|dx
\end{array} \tag{65}
$$

and this is as far as we get without making a few more or less crude approximations. We thus drop the first term in this expression and ignore to what extent the matrices commute, leading to the approximation

$$
|\mathcal{S}-\tilde{\mathcal{S}}|^*(t) \leq \max_{s\in[0,t]}\{\frac{t-s}{2}k(s)\max_{I(s)}|\frac{dA}{dt}|\}\int_0^t |\tilde{A}(x)\frac{1}{t-x}\int_x^t \Pi_x^y \mathrm{e}^{\tilde{A}(z)dz}\Pi_y^t \mathrm{e}^{A(z)dz}dy|dx, \tag{66}
$$

since, for $s\in[0,t]$,

$$
\begin{array}{rcl}
|(t-s)(A(s)-\tilde{A}(s))| & \leq & (t-s)\frac{k(s)}{2}\max_{I(s)}|\frac{dA}{dt}| \\
& \leq & \max_{s\in[0,t]}\{\frac{t-s}{2}k(s)\max_{I(s)}|\frac{dA}{dt}|\},
\end{array} \tag{67}
$$

where the maxima are elementwise, i.e. for $A : t \to \mathbf{R}^{N\times N}$,

$$\mathbf{1}_i^* \max_I\{|A|\}\mathbf{1}_j = \max_I |\mathbf{1}_i^* A \mathbf{1}_j|. \tag{68}$$

Now let

$$\alpha_1(t) = \frac{1}{2}|| \max_{s\in[0,t]}\{(t-s)k(s)\max_{I(s)}|\frac{dA}{dt}|\}||_\infty. \tag{69}$$

Then, assuming that $\tilde{A}$ is close to A, we end up with the approximation

$$\begin{array}{rcl} ||\mathcal{S}-\tilde{\mathcal{S}}||_1(t) & = & ||\mathcal{S}^*-\tilde{\mathcal{S}}^*||_\infty(t) \\ & \lessapprox & \alpha_1(t)||\int_0^t |A(x)\frac{1}{t-x}\int_x^t \Pi_x^y \mathrm{e}^{A(z)dz}\Pi_y^t \mathrm{e}^{A(z)dz}dy|dx||_\infty \\ & = & \alpha_1(t)||\int_0^t |A(x)\frac{1}{t-x}\int_x^t \Pi_x^t \mathrm{e}^{A(z)dz}dy|dx||_\infty \\ & = & \alpha_1(t)||\int_0^t |A(x)\Pi_x^t \mathrm{e}^{A(y)dy}|dx||_\infty \\ & = & \alpha_1(t)||\mathcal{S}^*||_\infty(t) \\ & = & \alpha_1(t)||\mathcal{S}||_1(t), \end{array} \tag{70}$$

since the $||\cdot||_1$ norm is the $||\cdot||_\infty$ norm of the transpose.

5.4.2 The second part of the error

The second part of the error is simply the quadrature error for approximating the integral with the sum, and it is bounded by

$$\begin{array}{rcl} |\tilde{\mathcal{S}}-\tilde{\tilde{\mathcal{S}}}\,|^*(t_n) & = & |\int_0^{t_n}|\tilde{B}(x,t_n)|dx - \sum_{i=1}^n |\tilde{B}(x_i-k_i/2,t_n)|k_i| \\ & = & |\sum_{i=1}^n \int_{I_i}\left(|\tilde{B}(x,t_n)| - |\tilde{B}(x_i-k_i/2,t_n)|\right)dx| \\ & \leq & \sum_{i=1}^n \int_{I_i}|\tilde{B}(x,t_n)-\tilde{B}(x_i-k_i/2,t_n)|dx \\ & \leq & \sum_{i=1}^n \max_{I_i}|\frac{d\tilde{B}}{dt}|\int_{I_i}|x-(x_i-k_i/2)|dx \\ & = & \sum_{i=1}^n \frac{k_i^2}{4}\max_{I_i}|\frac{d\tilde{B}}{dt}|. \end{array} \tag{71}$$

Now, noticing that the derivative of $\tilde{B}$ on I_i is nothing but

$$\frac{d\tilde{B}}{dt}(x,t_n) = -A_i^2\mathrm{e}^{(t_i-x)A_i}\Pi_{j=i+1}^n \mathrm{e}^{k_jA_j} = -A_i\tilde{B}(x,t_n), \tag{72}$$

we can write the error as

$$
\begin{array}{rcl}
|\tilde{\mathcal{S}} - \mathcal{S}|^*(t_n) & \leq & \sum_{i=1}^n \frac{k_i^2}{4} \max_{I_i} \{|A_i||B|\} \\
& \leq & \sum_{i=1}^n \frac{k_i}{4} |A_i| k_i \max_{I_i} \{|B|\} \\
& \leq & \frac{1}{4} \max_{i=1,\dots,n} \{k_i |A_i|\} \sum_{i=1}^n k_i \max_{I_i} \{|B|\} \\
& \approx & \frac{1}{4} \max_{i=1,\dots,n} \{k_i |A_i|\} \mathcal{S}^*(t_n),
\end{array} \tag{73}
$$

where again the maximum is elementwise.

Now let

$$
\alpha_2(t) = \frac{1}{4} || \max_{[0,t]} \{k|\tilde{A}|\}||_\infty, \; t = t_1, \dots, t_M. \tag{74}
$$

Then, taking the norms as before,

$$
||\tilde{\mathcal{S}} - \mathcal{S}||_1(t) \lessapprox \alpha_2(t) ||\mathcal{S}||_1(t), \; t = t_1, \dots, t_M. \tag{75}
$$

5.5 Adaptivity

From the previous section, we know that the relative error in the $||\cdot||_1$ norm for the $\mathrm{cG}(1)$ stability matrix at times $t = t_1, \dots, t_M$ is (approximately) bounded by

$$
\begin{array}{rcl}
\alpha(t) & = & \alpha_1(t) + \alpha_2(t) \\
& = & \frac{1}{2} || \max_{s \in [0,t]} \{(t-s) k(s) \max_{I(s)} |\frac{dA}{dt}|\}||_\infty + \frac{1}{4} || \max_{[0,t]} \{k|\tilde{A}|\}||_\infty .
\end{array} \tag{76}
$$

Given a tolerance, RELTOL, for the maximum relative error of the approximate stability matrix on $[0,T]$, we thus choose

$$
\frac{1}{2} || \max_{t \in [0,T]} \{(T-t) k(t) \max_{I(t)} |\frac{dA}{dt}|\}||_\infty + \frac{1}{4} || \max_{[0,T]} \{k|\tilde{A}|\}||_\infty \leq \mathrm{RELTOL}, \tag{77}
$$

noticing that $\alpha_i(t) \leq \alpha_i(T), \; t \in [0,T], \; i = 1, 2$. This can be done by choosing

$$
\begin{array}{rcl}
\alpha_1(T) & \leq & \mathrm{RELTOL}/2, \\
\alpha_2(T) & \leq & \mathrm{RELTOL}/2.
\end{array} \tag{78}
$$

Since the maxima in time are individual for the different elements, we cannot choose the proper timesteps by evaluating the norms at every timelevel. Instead, noting that the $||\cdot||_\infty$ norm is the same as the maximum row sum, we achieve (78) by choosing $k(t)$ such that

$$\begin{aligned} \tfrac{1}{2}(T-t)k(t)\max_{ij}\left\{N_i^* \max_{I(t)}|\tfrac{dA_{ij}}{dt}|\right\} &\leq \mathrm{RELTOL}/2,\ t\in[0,T], \\ \tfrac{1}{4}k(t)\max_{ij}\left\{N_i^*|\tilde{A}_{ij}(t)|\right\} &\leq \mathrm{RELTOL}/2,\ t\in[0,T], \end{aligned} \tag{79}$$

where N_i^* is the number of nonzero elements in the ith row of A.

6 One-sweep computation

As mentioned earlier, we would like to be able to solve the dual at the same time as we are solving the primal, thus avoiding excessive storage of data that has to be available for the following computation of the dual solution. If we could also cleverly use the information we get from solving the dual simultaneously, we would be able to solve the equations with global error control in a *one-sweep computation*, i.e. a computation starting at time $t = 0$ and ending at time $t = T$.

There are two major concerns with this, and those are how to compute the stability matrix and how to use the information obtained. We discuss these issues in the sections below. Again we only discuss the computation of the stability matrix $\mathcal{S}$ for a $\mathrm{cG}(1)$ computation.

6.1 Efficient computation of stability matrices

As discussed in the previous sections, we compute the approximation of the stability matrix $\mathcal{S}$ as

$$\tilde{\tilde{\mathcal{S}}}\ (t_n) = \left(\sum_{i=1}^{n}|\tilde{B}(x_i - k_i/2, t_n)|k_i\right)^*,\ n = 1,\ldots,M, \tag{80}$$

where

$$\tilde{B}(x,t) = \tilde{A}(x)\Pi_x^t \mathrm{e}^{\tilde{A}(y)dy},\ x\in[0,t),\ t\in(0,T] \tag{81}$$

and $\tilde{A}$ is the piecewise constant interpolant of A that takes the midpoint value in every interval.

What we then need to do at every time t_n is to compute the values of $\tilde{B}$ at times $\{t_1 - k_1/2, \dots, t_n - k_n/2\}$ according to (81) and at the same time compute the sum (80). This can be done easily by using the previous computations at times $\{t_1, \dots, t_{n-1}\}$, realizing that the only thing we have to do is to adjust the previous values of $\tilde{B}$ by a multiplication of the proper matrix exponential from the right.

Thus, essentially, at every timestep for the dual (notice that these may be different from the ones used for the solution of the primal), we have to compute one matrix exponential, do $(n-1)$ matrix multiplications and then the summation. Notice that in this simple way we solve N times M different dual problems, i.e.

$$\left\{ \begin{array}{rcl} -\frac{d\varphi}{dt}(t) & = & J^*(u, U, t)\varphi(t),\ t \in [0, t_m), \\ \varphi(t_m) & = & \mathbf{1}_n, \end{array} \right. \tag{82}$$

for $n = 1, \dots, N,\ m = 1, \dots, M$. The result is maximum norm (both in space, i.e. with respect to the different components, and time) error control, with correct data for the dual problem.

6.2 Extrapolation of stability factors

It is obvious that at time t, we know the value of the stability matrices only in the interval $(0, t]$. On the other hand, we really need to know the values of the stability matrices in the interval $[t, T]$ in order to choose the proper timesteps at time t. It thus seems to be impossible to do one-sweep computations in practice. This is not the case.

What we do is to extrapolate in some way the stability matrices from the interval $(0, t]$. This may of course be done in many different ways and it is not clear which one is the best, simply because the subsequent evolution of the stability matrices may be virtually anything!

However, whichever way we choose to do the extrapolation, at final time T, we will know how well we did doing the extrapolation and we will have a proper estimation of the maximum error.

What we thus seek at time t is the values of the stability matrices in the interval $[t, T]$ that give the worst (largest) possible value of the left-hand side of (40). If we underestimate these values of the stability matrices, the final value of the error may be larger than the given tolerance. On the other hand, overestimating results in excessive computation. In either case, we will know at final time T, and perhaps — if the user so wishes — do the computation once more. Notice that the stability matrices do not have to be computed once more, since the data for the dual is fixed. Thus, at most, the primal has to be solved twice and the stability matrices computed once. (Because of the fixed data for the dual problem, we could also think of storing the values of the stability matrices in a large data bank for different equations, storing in this data bank stability matrices not previously computed, and reusing already computed stability matrices instead of computing them.)

For the example computations in the next section, the method of extrapolation is simply linear extrapolation, together with some additional smoothing. Though this is not the most clever way to do the extrapolation in general, we content ourselves with this simple approach for our example computations.

7 Numerical results

As already mentioned, multi-adaptivity is not yet implemented in the form presented in this report. The numerical examples presented in this section are therefore of two kinds.

The first of these are examples computed with the multi-adaptive Tanganyika ODE-solver library described in [11]. These examples demonstrate the features of the multi-adaptivity and the sharpness of the computed error bounds.

Secondly, we give some examples of stability factors (matrices) computed with the method (by matrix exponentials) described in this report.

7.1 Multi-adaptive multi-sweep computations

The following examples were computed with the Tanganyika multi-adaptive ODE-solver library. For these examples, the dual problem was thus solved in its full, using some guess for the data based on comparisons of solutions. Also, for the error estimates, the derivatives of the dual solution were kept as weights (stability functions), rather than computing stability factors.

Notice that these results were all obtained automatically, the only data specified being the equation (including initial data) and the tolerance. The equations were then solved automatically, including the solution of the dual problem – which was automatically generated by numerical differentiation of the given equation – and the error estimation.

7.1.1 A first simple example

As a first simple example, consider the following system of equations:

$$\begin{cases} \dot{u}_1 &= u_2, \\ \dot{u}_2 &= -u_1, \text{ in } (0,T], \\ u(0) &= (0,1). \end{cases} \tag{83}$$

The solution is of course $u(t) = (\sin(t), \cos(t))$. The equations are solved by the multi-adaptive $\mathrm{cG}(1)$ method with tolerance $8 \cdot 10^{-4}$ and $T = 50$. (The tolerance was actually chosen to be $.001$. The resulting error estimate was, however, $8 \cdot 10^{-4}$.) The true error is, according to figure 3, $6.8 \cdot 10^{-4}$ and the component errors are $5.3 \cdot 10^{-4}$ and $4.2 \cdot 10^{-4}$ respectively.

Note the behaviour of the multi-adaptive method, choosing different timesteps for the two components. The timesteps are chosen on basis of the residuals and stability functions. Note also the approximate equidistribution of the error.

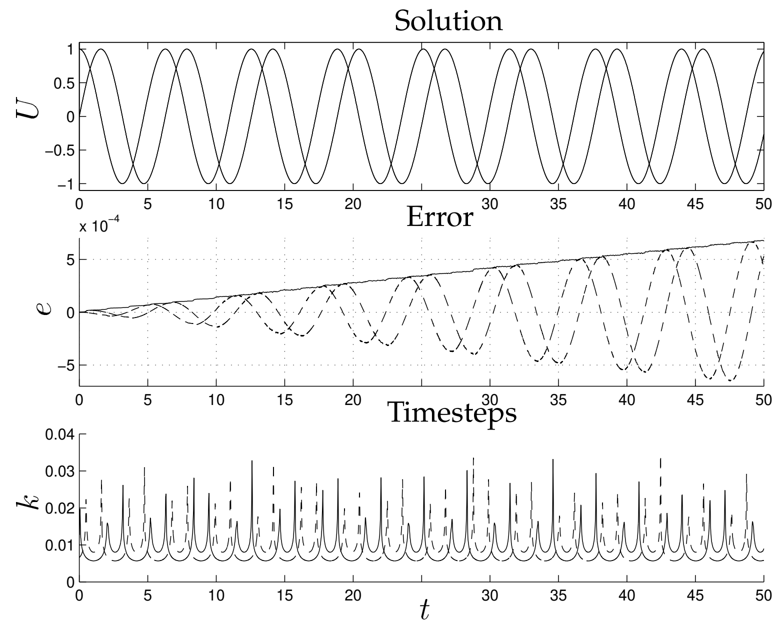


Figure 3: The solution of the simple harmonic oscillator problem, the errors and the timesteps respectively.

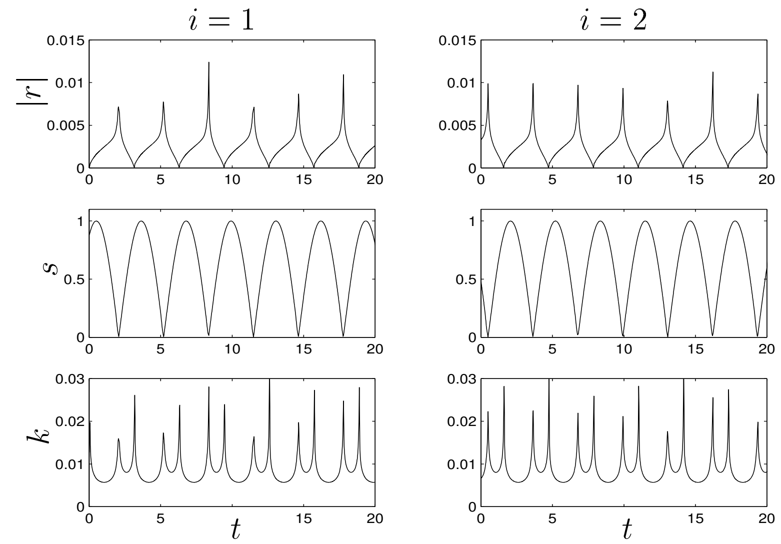


Figure 4: Residuals, stability functions and timesteps for the two components of the harmonic oscillator problem, shown for the interval $(0, 20)$.

Since the solution for this system of equations is known, we are able to compare the error estimate with the true error. Ideally the true error is smaller than and close to the error estimate. Is this the case for the multi-adaptive cG(q) method?

To check the reliability of the solver, the solution of (83) was computed with $T = 100$ at a large number of tolerances. The results are given for cG(q), $q = 1, 2, 3$, in figure 5.

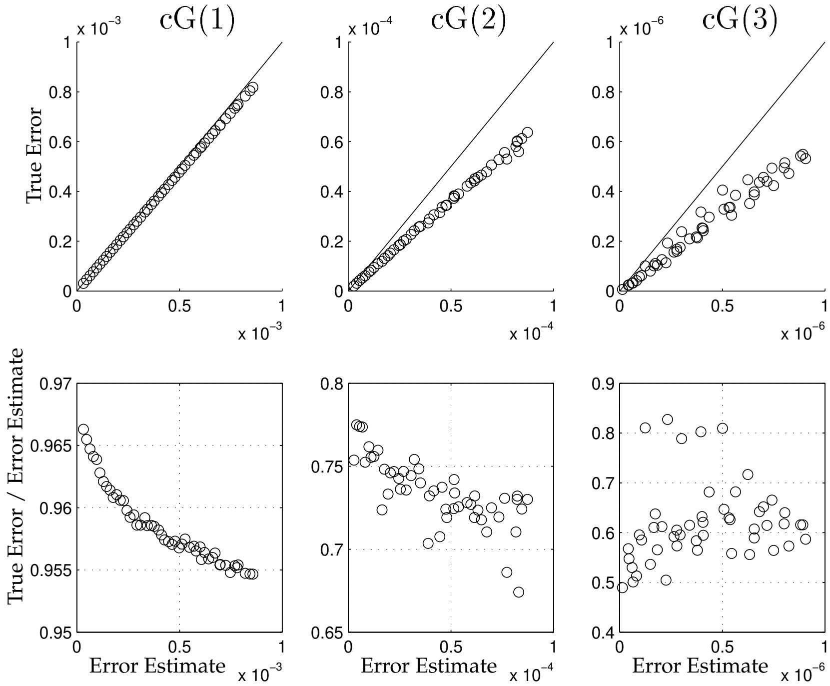


Figure 5: True error vs. error estimate for multi-adaptive cG(1), cG(2) and cG(3) respectively. Solid lines indicate the ideal maximum size of the true error.

As can be seen the true error is smaller than and close to the error estimate for the three methods. For this specific problem at these specific tolerance levels, the error for the cG(1) method is mostly discretization error (arising from the finite element discretization), whereas for the cG(3) method the error is mostly computational (arising from a non-zero discrete residual). For the cG(2) method the situation is somewhere in between. This explains the different variances in error–tolerance correlations for the three methods.

Notice also how sharp the error estimate is, especially for the cG(1) method. Again, this is due to the fact that at this tolerance level, most of the error is the usual finite element discretization error for the $\mathrm{cG}(1)$ method.

For comparison, the same computations were performed with the often used MATLAB ODE-solver, `ode45()`. As can be expected with a solver without global error control, the tolerance is only nominal, in the sense that its correlation to the true error is unknown.

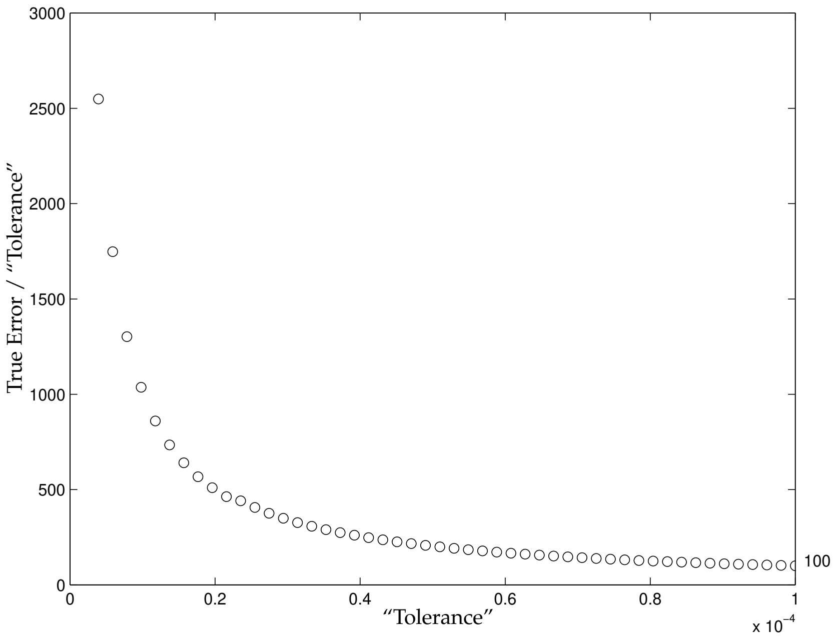


Figure 6: True error / tolerance vs. tolerance for MATLAB:s ODE-solver `ode45()`.

7.1.2 A simple non-linear system

Consider now the following nonlinear system:

$$\begin{cases} \dot{u}_1 &= u_1, \\ \dot{u}_2 &= u_2 + u_1 u_1, \\ \dot{u}_3 &= u_3 + u_1 u_2, \\ \dot{u}_4 &= u_4 + u_1 u_3 + u_2 u_2, \\ \dot{u}_5 &= u_5 + u_1 u_4 + u_2 u_3, \\ u(0) &= (1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}). \end{cases} \tag{84}$$

The solution is (obviously) $u(t) = (e^t, e^{2t}, \frac{1}{2}e^{3t}, \frac{1}{2}e^{4t}, \frac{1}{4}e^{5t})$.

A comparison between true error and error estimate is given in figure 7 for the multi-adaptive $\mathrm{cG}(1)$-method. Also for this nonlinear problem, the true error is smaller than and close to the error estimate, as desired.

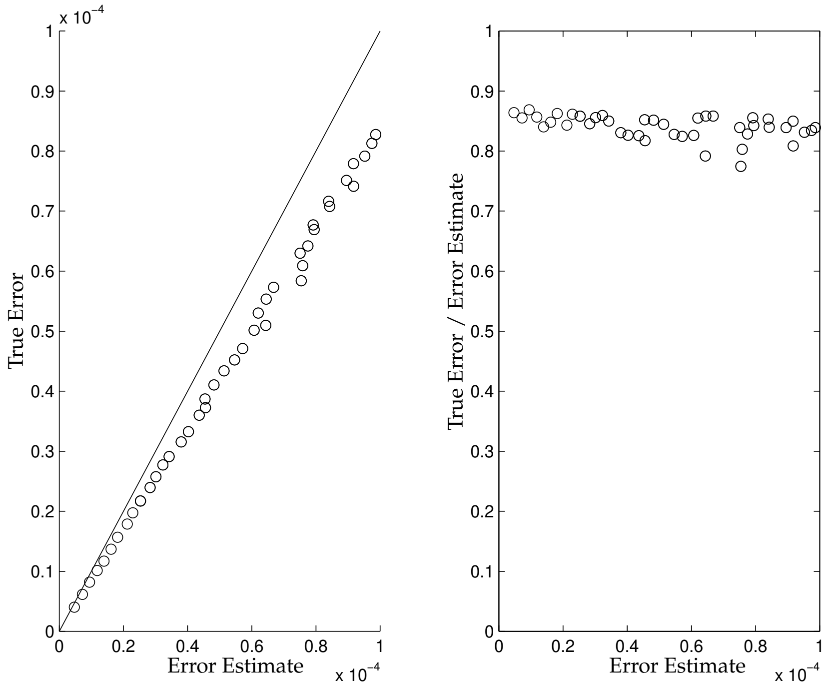


Figure 7: True error vs. error estimate for the multi-adaptive cG(1)-method. The solid lines indicate the ideal maximum size of the true error.

7.1.3 Wave propagation in an elastic medium

As a third example, consider wave propagation in an elastic medium, represented by a number of masses connected together with springs of equal stiffness. The equations are easily obtained from Newton's second law of motion:

$$\left\{\begin{array}{rcl} \ddot{x} & = & Ax, \text{ where} \\ A & = & \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & 0 & \\ & 1 & -2 & 1 & & \\ & & 1 & -2 & \ddots & \\ & 0 & & \ddots & \ddots & 1 \\ & & & & 1 & -2 \end{bmatrix} \end{array}\right. . \tag{85}$$

(This may also be thought of as a FEM space discretization of the wave equation.)

With initial conditions corresponding to all masses but one being at rest at $t = 0$, we expect a propagation of the timesteps. At the beginning the timesteps for the masses at rest may be large. As the oscillations of a mass increase, the corresponding timesteps should decrease and oscillate. This is also the case according to figure 8.

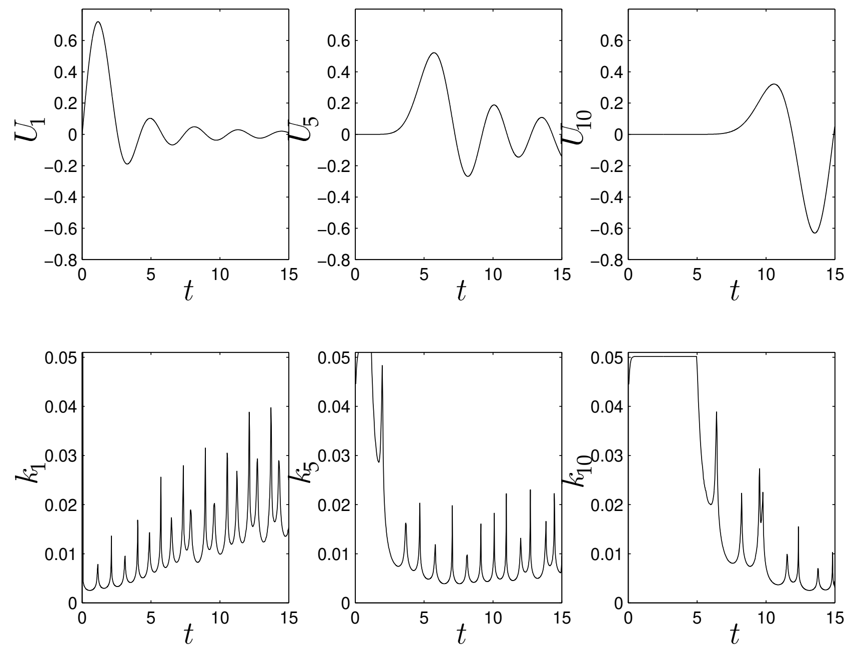


Figure 8: Solutions for components 1,5 and 10 of a system consisting of 10 masses and 11 springs, together with their respective timesteps, solved at $\mathrm{TOL} = 5 \cdot 10^{-4}$ with the multi-adaptive $\mathrm{cG}(1)$ method.

7.1.4 Gravitation

As a fourth example, consider a system of three bodies (planets) in a somewhat complicated situation where one of the planets is in orbit around a larger one, and a third even smaller planet comes in making sort of a weird sling-shot around the smaller planet.

The forces involved are $1/r^2$ and for a certain choice of initial conditions, the solution is as depicted in figure 9 below for $\mathrm{TOL} = .001$, solved with the multi-adaptive $\mathrm{cG}(2)$ method.

As one might expect, the three bodies are differently sensitive to the resolution of the discretization. This is also evident in figure 11, where are drawn the timesteps for the components corresponding to the x-coordinates of the three planets. (The problem is in two dimensions so there is a total number of 12 components.) In this figure are also the number of timesteps used for the different components. The larger planet, corresponding to components 1,2,7 and 8, obviously doesn't require as many steps as the two smaller ones. The largest number of steps is, according to this figure, needed to resolve the y-velocities of the smallest planet, which is not too strange, considering the main acceleration is in the y-direction at the critical point.

It is obviously crucial for the timesteps (of the involved components) to be small just when the smallest planet makes the sling-shot. This is realized in the adaptive algorithm by extremely large values of the stability functions for the involved components, as is shown in figure 10.

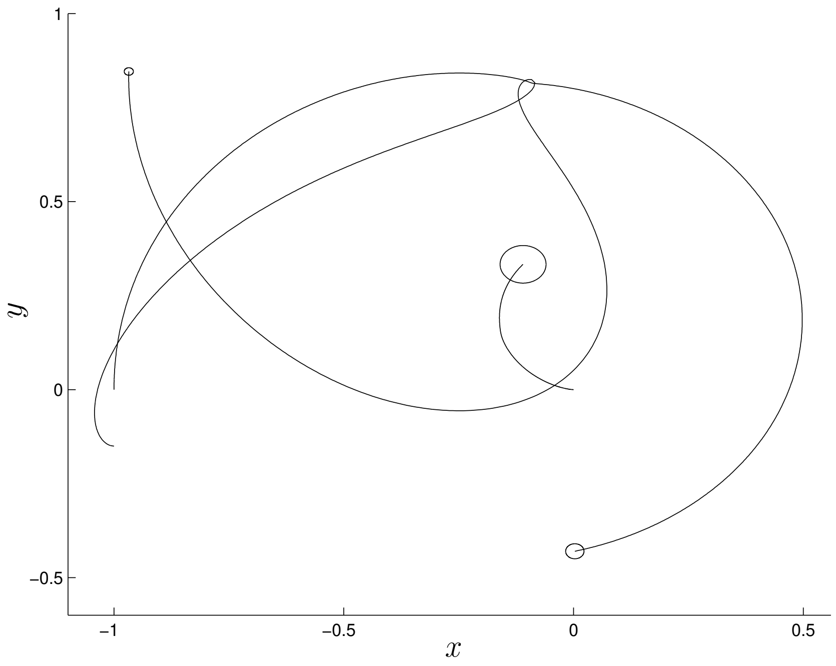


Figure 9: Orbits for the three planets. The circles drawn represent the planets at time $t = T$.

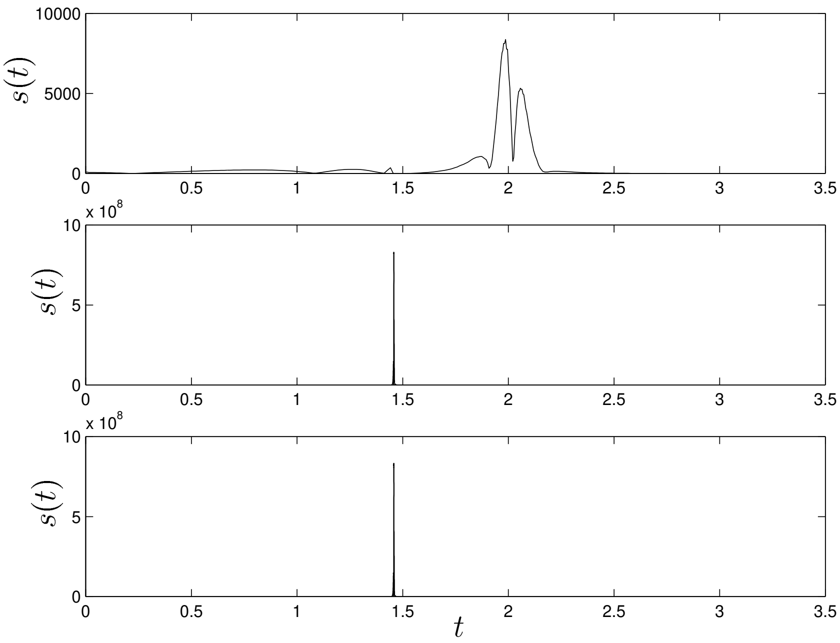


Figure 10: Stability functions for the x-components of the three planets.

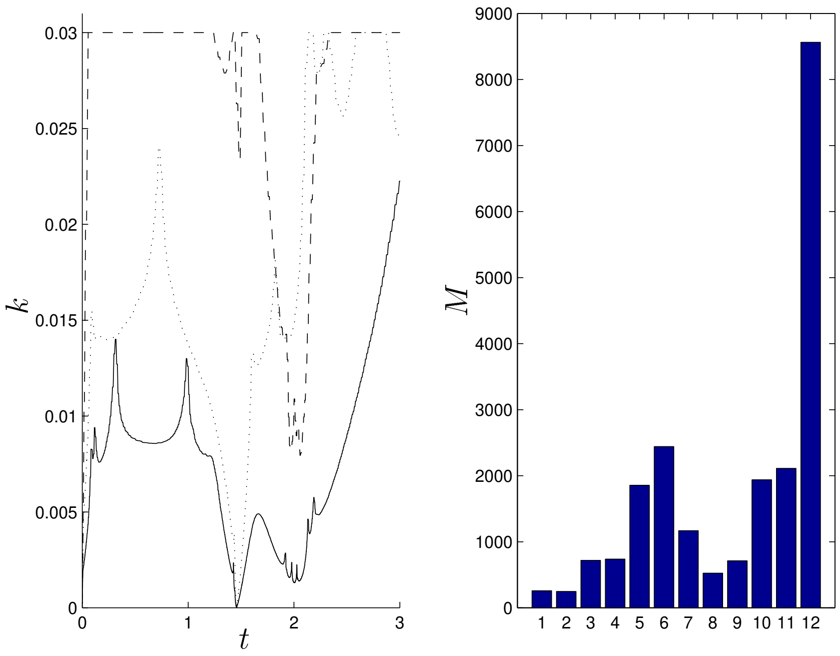


Figure 11: Timesteps (left) and the number of timesteps (right) for the 12 different components of the three-body problem.

7.1.5 The Lorenz system

As a fifth and final example, consider the Lorenz system given by the equations

$$\begin{cases} \dot{x} &= \sigma(y-x), \qquad t \in (0,T], \\ \dot{y} &= rx - y - xz, \quad t \in (0,T], \\ \dot{z} &= xy - bz, \qquad t \in (0,T], \\ x(0) = x_0, & y(0) = y_0, z(0) = z_0, \end{cases} \tag{86}$$

where $\sigma = 10$, $b = 8/3$ and $r = 28$, and $(x_0, y_0, z_0) = (1, 0, 0)$.

The solution at $\mathrm{TOL} = 2.5 \cdot 10^{-5}$ and $T = 10$ is shown in figure 12, together with the timesteps used for the computation. The "chaotic", flipping, behaviour of the Lorenz system is not evident in this figure, since T is too small. The purpose of this example is however not to illustrate certain characteristics of the Lorenz system, but to illustrate the use of multiadaptivity for the three components.

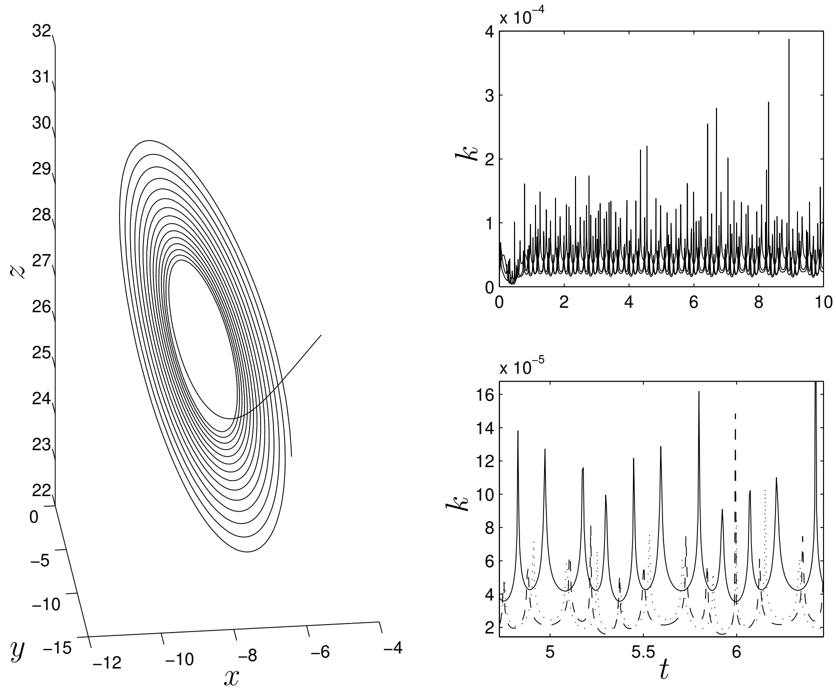


Figure 12: At the left is the solution of the Lorenz system, solved with the multiadaptive $\mathrm{cG}(1)$ method at $\mathrm{TOL} = 2.5 \cdot 10^{-5}$ and with final time $T = 10$. At the right are the timesteps used for the computation.

7.2 Computation and extrapolation of stability matrices

The following are examples of the computation of stability matrices (the $\mathrm{cG}(1)$ stability matrix for the accumulation of the Galerkin discretization error) by the method of matrix exponentials for a number of different model problems, most of which are taken (with some modification) from Don Estep's *Stability Factor Gallery* ([4]).

We give the solutions to the primal problem, the solutions to the dual for $\varphi(T) = \mathbf{1}_1$, the evolution of the stability factors for this choice of data, the extrapolations of these stability factors to time $t = T$ and finally the adaptive timesteps used for the computation of the stability matrix as function of time.

Notice that although we only plot the stability factors for this specific choice of data, the full stability matrix is computed.

For some of these examples, the extrapolation of the stability matrix works fine, and for others the extrapolation fails. This can be blaimed on the somewhat simple method used for extrapolation. Notice also that the error estimate for the computation of the stability matrix seems to be somewhat pessimistic. We could have taken larger timesteps than what follows from the error estimate.

All examples were computed with relative tolerance $\mathrm{RELTOL} = 50\%$.

7.2.1 The harmonic oscillator

We start out with the simple linear problem

$$\left\{ \begin{array}{rcl} \dot{u}_1 & = & u_2, \\ \dot{u}_2 & = & -u_1, \text{ in } (0, T], \\ u(0) & = & (0, 1), \end{array} \right. \tag{87}$$

where we choose $T = 20$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \left[\begin{array}{cc} 12.6 & 12.9 \\ 12.9 & 12.6 \end{array} \right] \tag{88}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}} - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 0.40\% \leq \mathrm{RELTOL}. \tag{89}$$

For this simple problem the extrapolation works as desired, resulting in a one-sweep computation.

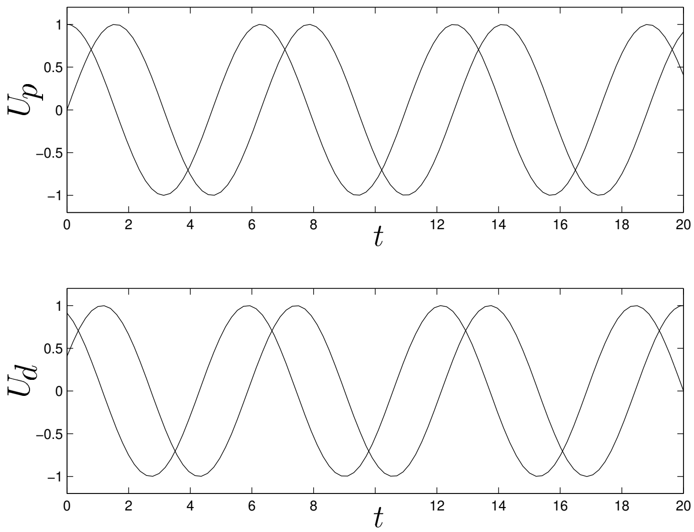


Figure 13: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

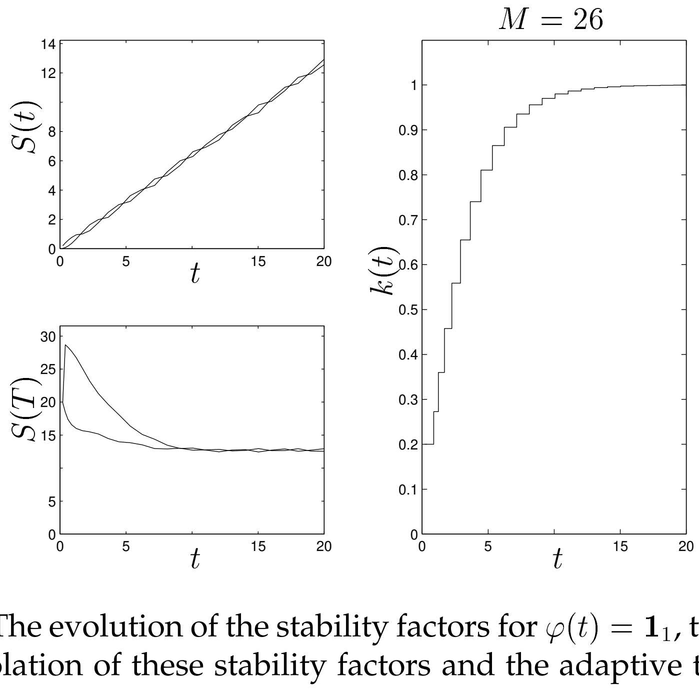


Figure 14: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.2 Exponential growth

Consider now a problem with exponentially growing stability factors:

$$\begin{cases} \dot{u}_1 &= u_1 + u_2, \\ \dot{u}_2 &= 2(u_1 + u_2), \text{ in } (0, T], \\ u(0) &= (1, 0), \end{cases} \tag{90}$$

where we choose $T = 3$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \begin{bmatrix} 2700 & 2700 \\ 5400 & 5400 \end{bmatrix} \tag{91}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}}\; - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 9.6\% \leq \mathrm{RELTOL}. \tag{92}$$

For this problem the extrapolation does not work as desired. (However, it would have been possible to do a better job with the extrapolation, covering also the case of exponential growth.) The result is that we have to do the computation of the solution to the primal once more, this time knowing everything we need to know.

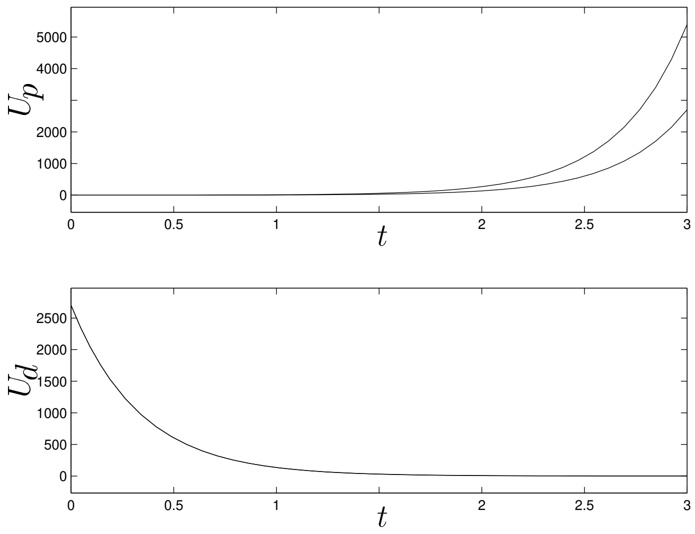


Figure 15: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

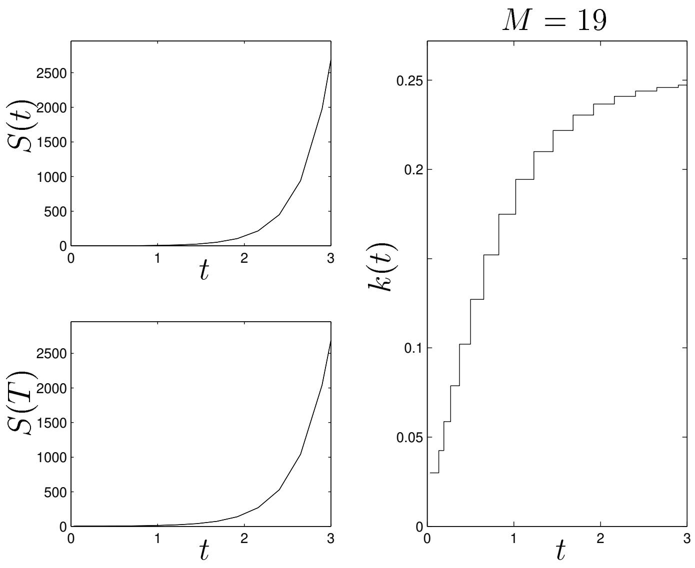


Figure 16: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.3 From instability to stability

Consider now the following system of equations:

$$\begin{cases} \dot{u}_1 &= -100(u_1 u_2)^2, \\ \dot{u}_2 &= 100(u_1 u_2)^2, \text{ in } (0,T], \\ u(0) &= (1, 0.1), \end{cases} \tag{93}$$

where we choose $T = 1$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}(T) = \begin{bmatrix} 1.0 & 0.034 \\ 1.0 & 0.034 \end{bmatrix} \tag{94}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}} - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 0.11\% \leq \mathrm{RELTOL}. \tag{95}$$

For this problem the extrapolation works as it should, overestimating the maximum over the remaining interval for every stability factor.

Notice how the stability factors settle to the constant values (of one and zero) after the initial peaks. (The initial value $(1, 0.1)$ is a perturbation of the unstable steady solution $(1, 0)$, whereas the solution at time T approaches the stable steady solution $(0, 1)$.)

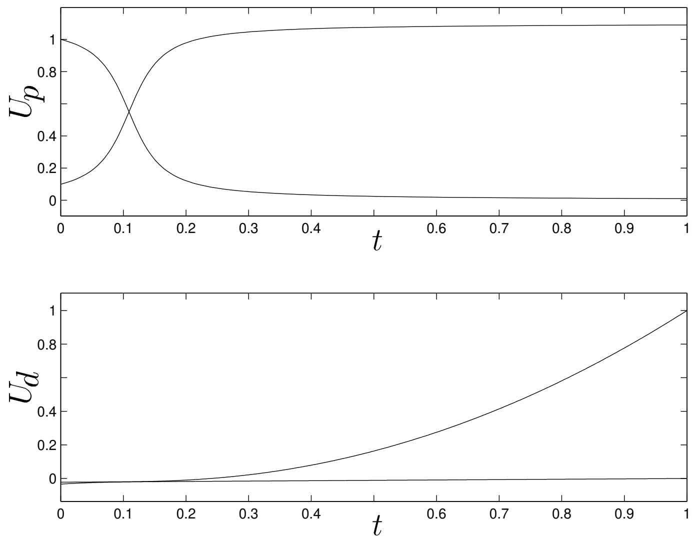


Figure 17: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

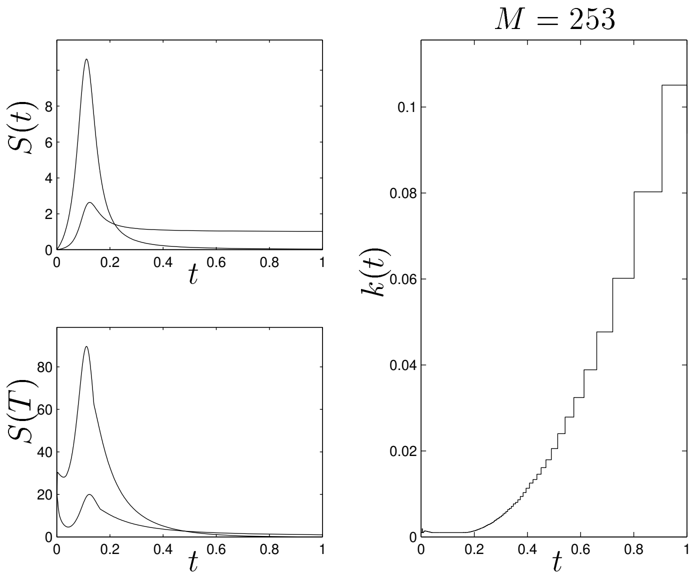


Figure 18: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.4 Signal transmission in the nerve of a giant squid

Consider now the following system of equations:

$$\left\{ \begin{array}{rcl} \dot{u_1} & = & -u_1(u_1 - \frac{1}{4})(u_1 - 1) - u_2, \\ \dot{u_2} & = & \frac{1}{10}u_1 - u_2, \text{ in } (0, T], \\ u(0) & = & (0.1, 0), \end{array} \right. \tag{96}$$

where we choose $T = 15$.

These are the Fitz-Hugh-Nagumo equations, which are a model for the Hodgkin-Huxley equations. These in turn are a model for the electromagnetic signal transmission in the nerve of a giant squid ([4]).

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\ (T) = \begin{bmatrix} 0.99 & 1.2 \\ 0.12 & 1.1 \end{bmatrix} \tag{97}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}}\ - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 6.8\% \leq \mathrm{RELTOL}. \tag{98}$$

Also for this problem the extrapolation works as it should.

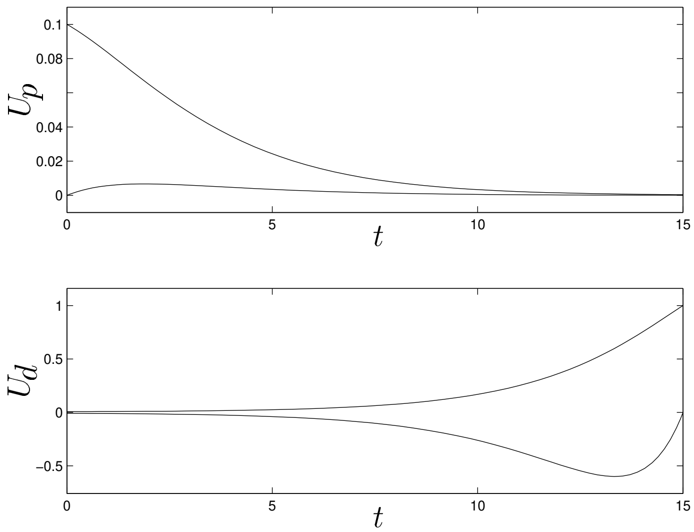


Figure 19: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

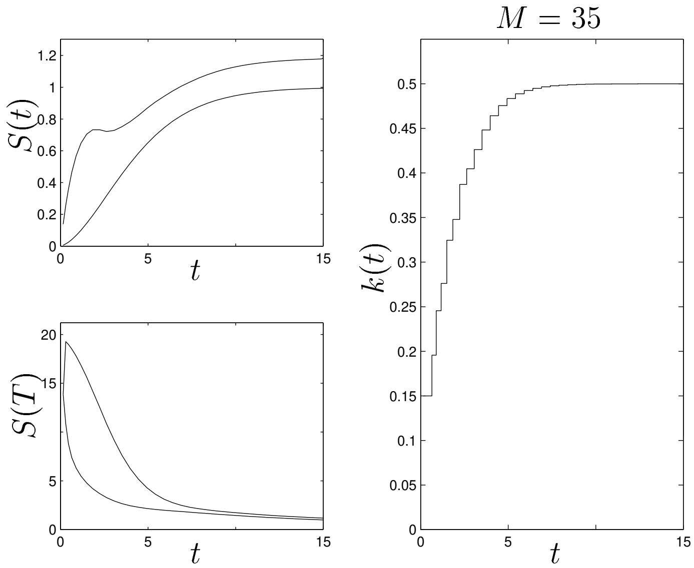


Figure 20: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.5 The Duffing problem

The following is the Duffing problem with a periodic force:

$$\left\{ \begin{array}{rcl} \dot{u}_1(t) & = & u_2(t), \\ \dot{u}_2(t) & = & u_1(t) - u_1^3(t) - 0.15u_2(t) + 0.3\cos t,\ t \in (0, T], \\ u(0) & = & (0, 0), \end{array} \right. \tag{99}$$

where we choose $T = 10$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \left[\begin{array}{cc} 14 & 11 \\ 13 & 8.5 \end{array} \right] \tag{100}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}} - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 2.5\% \leq \mathrm{RELTOL}. \tag{101}$$

For this problem the extrapolation does not work well enough for a one-sweep computation.

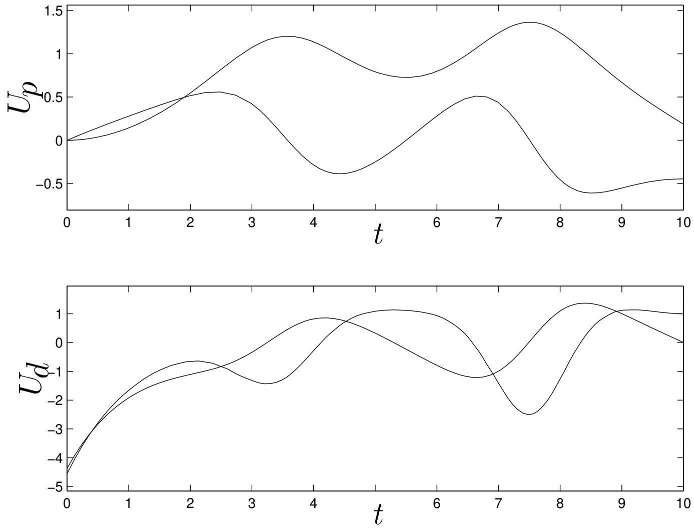


Figure 21: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

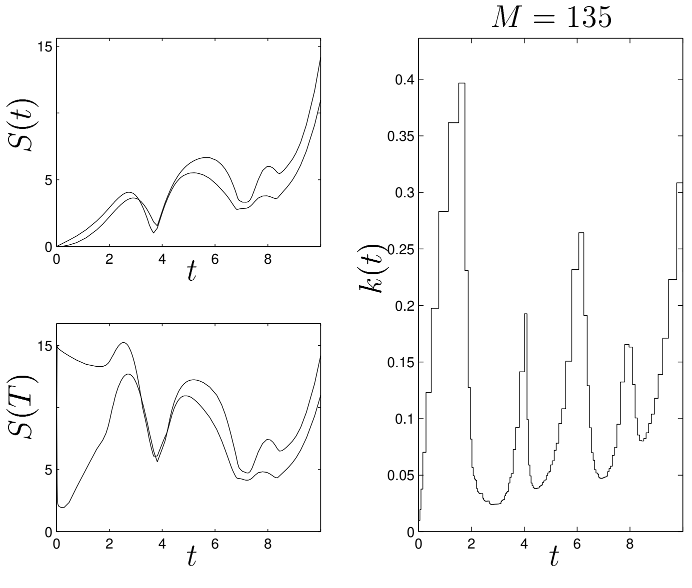


Figure 22: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.6 A nonlinear problem

The following is some nonlinear system of equations, chosen at random:

$$\left\{ \begin{array}{rcl} \dot{u_1} & = & u_2 - u_3, \\ \dot{u_2} & = & -u_1 - u_3, \\ \dot{u_3} & = & (u_1 + u_2)^3 - u_3, \ \text{in } (0, T], \\ u(0) & = & (1, 0, 0), \end{array} \right. \tag{102}$$

where we choose $T = 10$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \begin{bmatrix} 6.2 & 5.5 & 5.3 \\ 5.3 & 4.9 & 4.5 \\ 1.2 & 1.1 & 1.6 \end{bmatrix} \tag{103}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}} - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 1.2\% \leq \mathrm{RELTOL}. \tag{104}$$

For this problem the extrapolation works reasonably well.

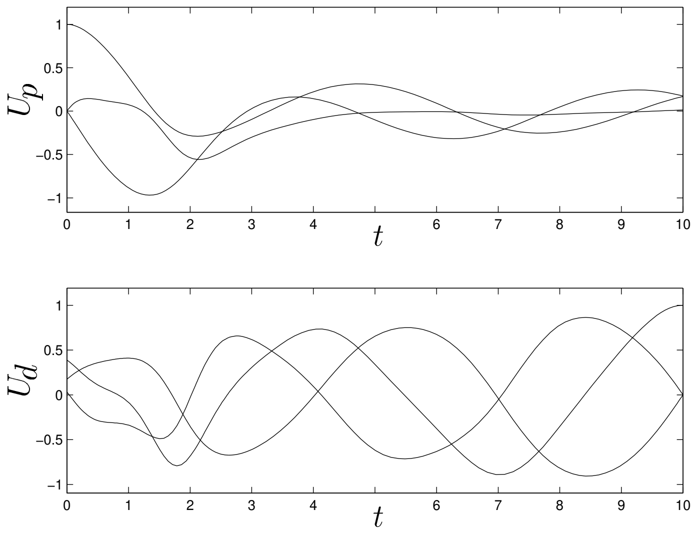


Figure 23: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

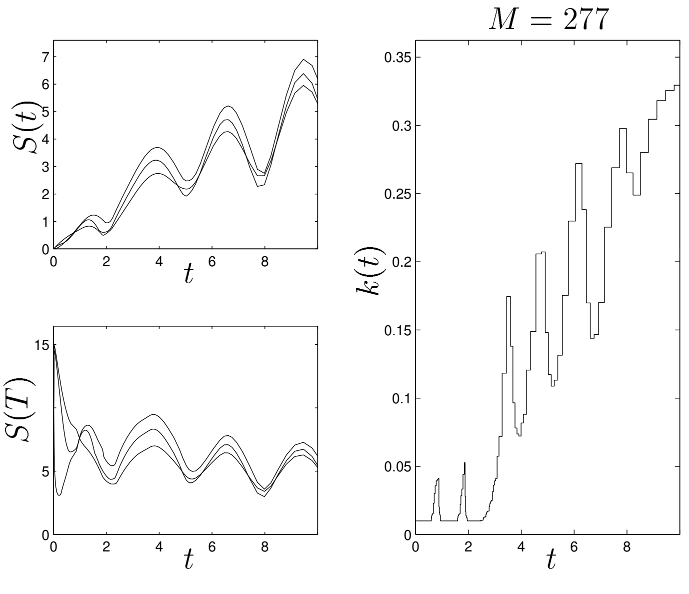


Figure 24: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

7.2.7 More on the Lorenz system

Consider again the Lorenz system:

$$\left\{\begin{array}{rcll} \dot{x} &=& \sigma(y-x), & t \in (0,T], \\ \dot{y} &=& rx - y - xz, & t \in (0,T], \\ \dot{z} &=& xy - bz, & t \in (0,T], \\ \multicolumn{4}{l}{x(0) = x_0, y(0) = y_0, z(0) = z_0,} \end{array}\right. \tag{105}$$

where $\sigma = 10$, $b = 8/3$ and $r = 28$, $(x_0, y_0, z_0) = (1, 0, 0)$ and $T = 5$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \begin{bmatrix} 16 & 28 & 20 \\ 22 & 40 & 30 \\ 18 & 36 & 29 \end{bmatrix} \tag{106}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}} - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 13\% \leq \mathrm{RELTOL}. \tag{107}$$

The extrapolation works reasonably well, although we wouldn't have ended up performing a one-sweep computation for this problem, if we really wanted the resulting error estimate to be smaller than the given tolerance. (Perhaps we could have tolerated the resulting error estimate to be close to but somewhat larger than the tolerance.)

For this problem we have chosen the timestep to be constant $k = T/100$. This is to demonstrate that the error bound is too pessimistic — based on this bound, the number of timesteps would be about ten times as large, which is obviously not necessary.

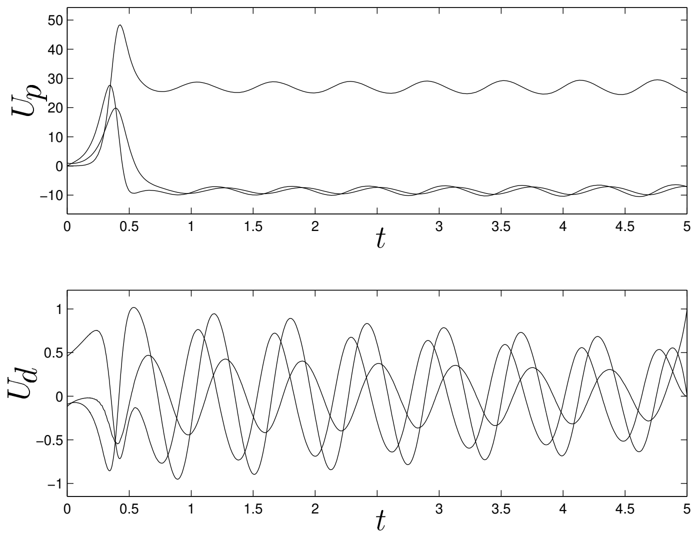


Figure 25: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

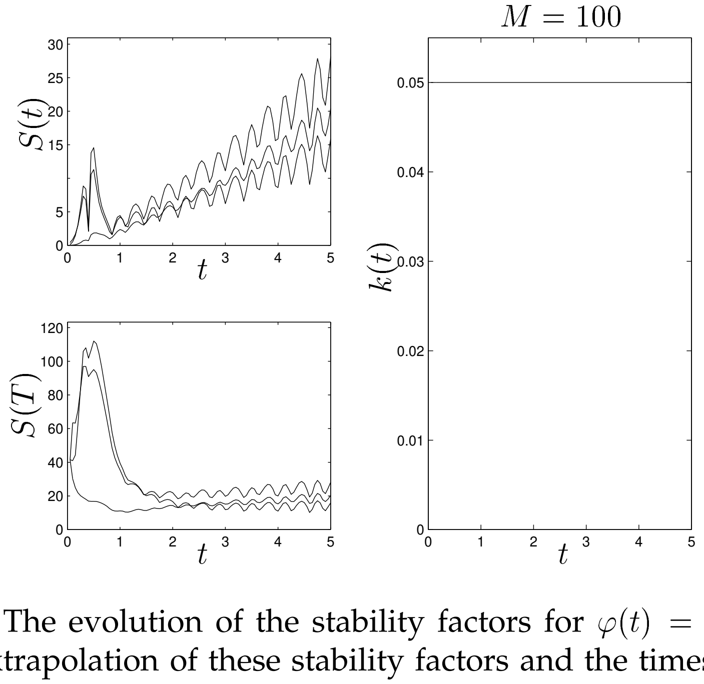


Figure 26: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the timesteps for the computation.

7.2.8 Gravitation again

Finally, we investigate the following 2-dimensional 2-body problem:

$$\left\{\begin{array}{rcl} \dot{u_1} &=& u_3, \\ \dot{u_2} &=& u_4, \\ \dot{u_3} &=& \frac{-u_1}{\sqrt{u_1^2+u_2^2}^3}, \\ \dot{u_4} &=& \frac{-u_2}{\sqrt{u_1^2+u_2^2}^3}, \text{ in } (0,T], \\ u(0) &=& (0,1,1,0), \end{array}\right. \tag{108}$$

where we choose $T = 3$.

The approximation of the $\mathrm{cG}(1)$ stability matrix is then:

$$\tilde{\tilde{\mathcal{S}}}\,(T) = \left[\begin{array}{cccc} 7.1 & 11 & 8.9 & 5.5 \\ 4.0 & 3.6 & 2.8 & 3.3 \\ 4.9 & 5.8 & 3.5 & 3.4 \\ 8.0 & 11 & 8.4 & 6.6 \end{array}\right] \tag{109}$$

and the relative error at final time is

$$\frac{||\tilde{\tilde{\mathcal{S}}}\, - \mathcal{S}||_1}{||\mathcal{S}||_1}(T) \approx 1.7\% \leq \mathrm{RELTOL}. \tag{110}$$

For this problem the extrapolation does not work as desired, which cannot be expected with the stability factors growing slowly at first and then faster. For this system, we would thus have to solve the primal once again.

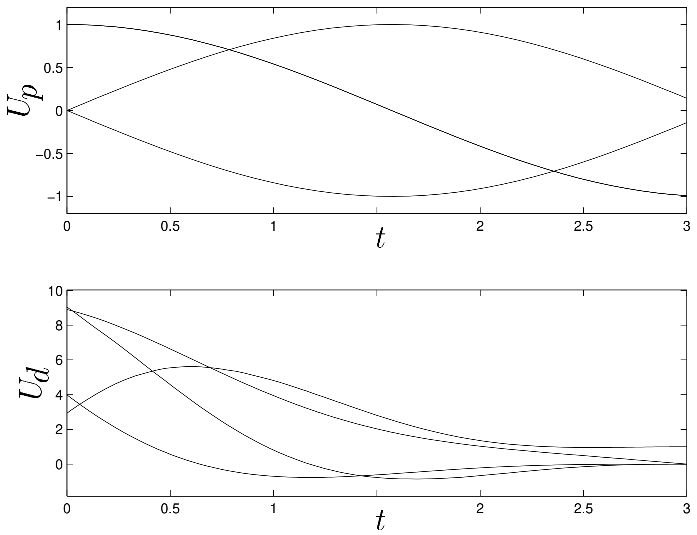


Figure 27: The solution of the primal and the solution of the dual for data $\varphi(T) = \mathbf{1}_1$.

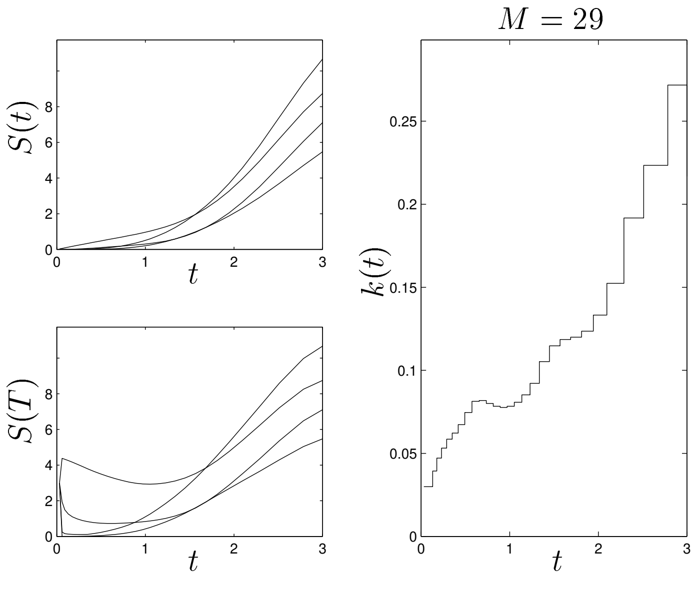


Figure 28: The evolution of the stability factors for $\varphi(t) = \mathbf{1}_1$, together with the extrapolation of these stability factors and the adaptive timesteps for the computation.

8 Conclusions

We have introduced the multi-adaptive method for ODEs and demonstrated the use of it for a number of different model problems. The correlation between error and error estimate is as desired (with the true error smaller than and close to the error estimate), at least for our simple model problems.

We have also shown how to do the error control with fixed and correct data for the dual, and how the resulting dual problems can be solved efficiently and simultaneously by the use of matrix exponentials. At the same time, we also get the stability factors as function of time, resulting in maximum norm control in time of the error.

Furthermore, we have shown how the computation of the stability factors can be done forward in time, simultaneously with the solution of the primal, resulting in one-sweep computation, provided we do a good job extrapolating the stability factors. (Otherwise the result is we have to do *one* more solution of the primal.)

What remains is to improve the error control and the adaptivity for the computation of the stability factors, and to refine the method of extrapolation.

Acknowledgements

I would like to thank Claes Johnson (Chalmers University of Technology) and Endre Süli (Oxford University Computing Laboratory) for their comments and their encouragement.

References

[1] E. BURMAN, *Adaptive Finite Element Methods for Compressible Two-Phase Flow*, PhD thesis, Department of mathematics, Chalmers University of Technology, Göteborg, 1998.

[2] 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.

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

[4] D. ESTEP, *Stability factor gallery*, `http://www.math.gatech.edu/~estep/math_stabgal.html` `[981130]`.

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

[6] D. ESTEP AND D. FRENCH, *Global error control for the continuous galerkin finite element method for ordinary differential equations*, M^2AN, 28 (1994), pp. 815–852.

[7] D. ESTEP, S. V. LUNEL, AND R. WILLIAMS, *Error estimation for numerical differential equations*, `http://www.cacr.caltech.edu/publications/techpubs/` `[980930]`, (1995).

[8] D. ESTEP AND R. WILLIAMS, *Accurate parallel integration of large sparse systems of differential equations*, Math. Models. Meth. Appl. Sci. (to appear).

[9] G. GOLUB AND C. V. LOAN, *Matrix Computations*, North Oxford Academic, Oxford, 2nd ed., 1986.

[10] 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.

[11] A. LOGG, *A Multi-Adaptive ODE-Solver*, MSc thesis, Department of mathematics, Chalmers University of Technology, Göteborg, 1998.

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

Chalmers Finite Element Center Preprints

These preprints can be obtained from

`www.phi.chalmers.se/preprints`