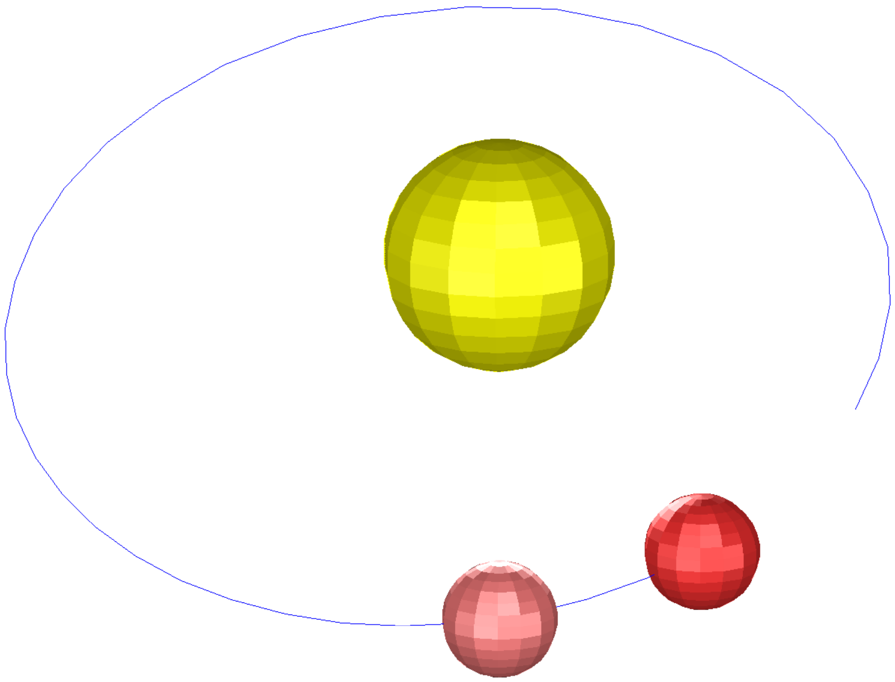
CHALMERS

FINITE ELEMENT CENTER



*PREPRINT 2001–10*

# Multi-Adaptive Galerkin Methods for ODEs II: Applications

Anders Logg

*Chalmers Finite Element Center*
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg Sweden 2001

CHALMERS FINITE ELEMENT CENTER

Preprint 2001–10

# Multi-Adaptive Galerkin Methods for ODEs II: Applications

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 II: Applications**
Anders Logg
NO 2001–10
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 II: APPLICATIONS

ANDERS LOGG

ABSTRACT. We present computational results for a variety of applications, solved using the multi-adaptive Galerkin methods, mcG($q$) and mdG($q$), presented in [7]. Generalizing the standard Galerkin methods, we allow individual time-steps for the different components of a system of ODEs. The individual time-steps are determined in an adaptive feed-back process, with the objective of efficient and reliable error control. Examples include the Lorenz system, the Solar System, stiff chemical reaction problems, and a number of time-dependent PDEs, such as the heat equation, the wave equation and convection-diffusion-reaction problems.

## 1. INTRODUCTION

In this paper we apply the multi-adaptive Galerkin methods, mcG($q$) and mdG($q$), presented in [7] to a variety of problems, chosen to illustrate the potential of the multi-adaptive methods. Throughout this paper, we solve the ODE initial value problem

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

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

We refer to [7] for a detailed description of the multi-adaptive methods. We here recall that each component $U_i(t)$ of the approximate solution $U(t)$ is a piecewise polynomial of degree $q_i = q_i(t)$ on a partition of $(0,T]$ into $M_i$ subintervals of lengths $k_{ij} = t_{ij} - t_{i,j-1}$, $j = 1,\ldots,M_i$. On interval $I_{ij} = (t_{i,j-1}, t_{ij}]$, component $i$ is thus a polynomial of degree $q_{ij}$.

The test problems presented below have been computed with the multi-adaptive ODE-solver *Tanganyika* (see [8]) on a regular workstation. A few of the problems are included in the publicly available (GPL) distribution of Tanganyika for Linux/Unix, and our intention is to include more problems in future versions.

*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, applications, Lorenz, Solar System, Burger.

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

## 2. The Harmonic Oscillator

We start with a basic example, the harmonic oscillator:

$$\left\{\begin{array}{rcll} \dot{u}_1 &=& u_2, & \text{on } (0,T], \\ \dot{u}_2 &=& -u_1, & \text{on } (0,T], \\ u(0) &=& (0,1), & \end{array}\right. \tag{2}$$

with exact solution $u(t) = (\sin(t), \cos(t))$. We solve this problem using the mcG(1) method, and present the solution in Figure 1, together with the error and the time-step sequences used for the two components. As one may expect, the error grows linearly with time. Since the continuous method preserves energy (see [7]), the computed solution has correct amplitude but the phase error grows.

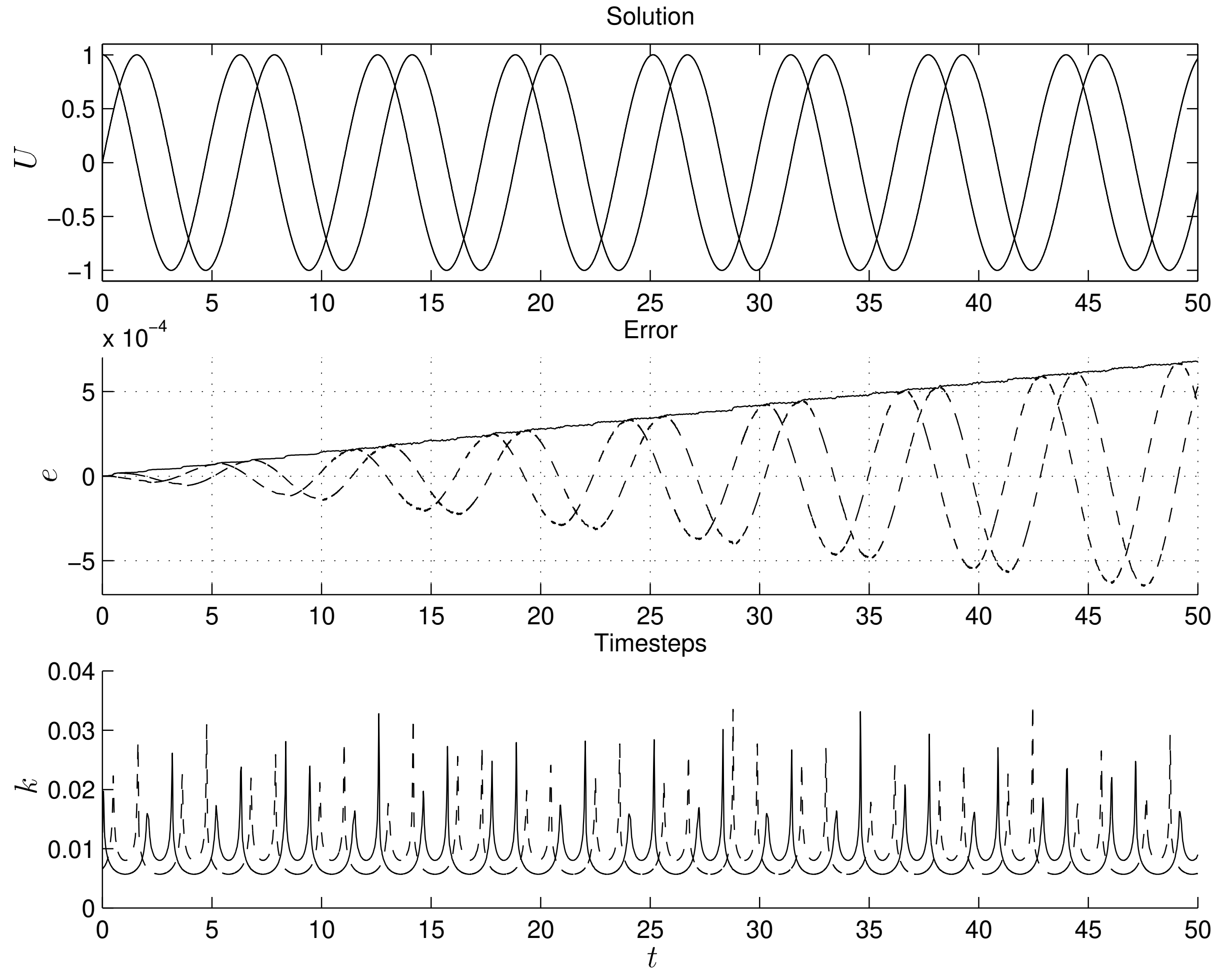



**Figure 1:** The mcG(1) solution of (2) together with the error growth and the multi-adaptive time-steps used for computing the solution.

We see that the time-steps for the two components are different. This is because, even for this simple problem, the components have different variations in time, and the time-steps are large for the mcG(1) method when the variation is close to linear and the residual is small.

The individual time-steps are thus large whenever the individual residuals are small. Examining Figure 1 more closely, we see that the individual time-steps are large also at other occasions. This happens when the local stability weight is small. As discussed in [7], there are different variants of the a posteriori error estimates on which we base the adaptive time-steps. One variant is to keep the local weights and try to equidistribute the contribution to the error over the intervals. In Figure 2 we plot residuals, local stability weights and time-steps. As can be seen, the time-steps are large whenever the residual *or* the stability weight is small. (The exact phase of the stability weights depend very much on the choice of data for the dual problem, i.e. which quantity of the solution we wish to control at final time.)

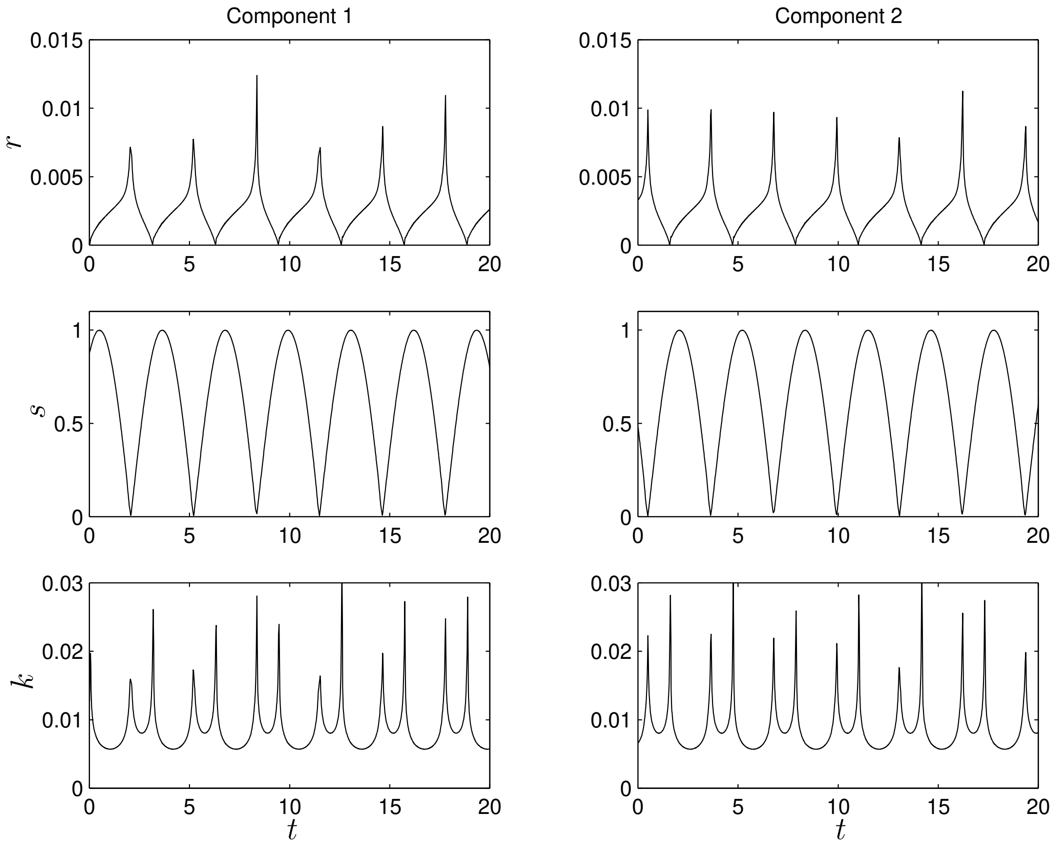



**Figure 2:** Residuals, local stability weights and time-steps for the two components of the harmonic oscillator. The time-steps will be large when the residual is small, or when the local stability weight is small.

2.1. **Comparison with the actual error.** To check the quality of the a posteriori error estimate, we compare the error estimate with the actual error. We do this for a couple of different methods, cG(1), cG(2) and cG(3) and for a number of different tolerances, using multi-adaptive time-steps. The correspondence between error and error estimate is very good (see Figure 3), especially for the cG(1) method.

For other problems we may not always expect as tight estimates as presented here. The quality of the error estimate depends very much on how much you are willing to pay to compute the estimate. If we want $l^2$-norm error control of the error, we have to guess the data for the dual. For the simple problem of this section, we will at most be off by about 50% if we make a completely wrong guess. For other problems, the situation may be worse (or better). The quality of the estimate depends also on which part of the error dominates. If the Galerkin error dominates, we can expect a sharper estimate.

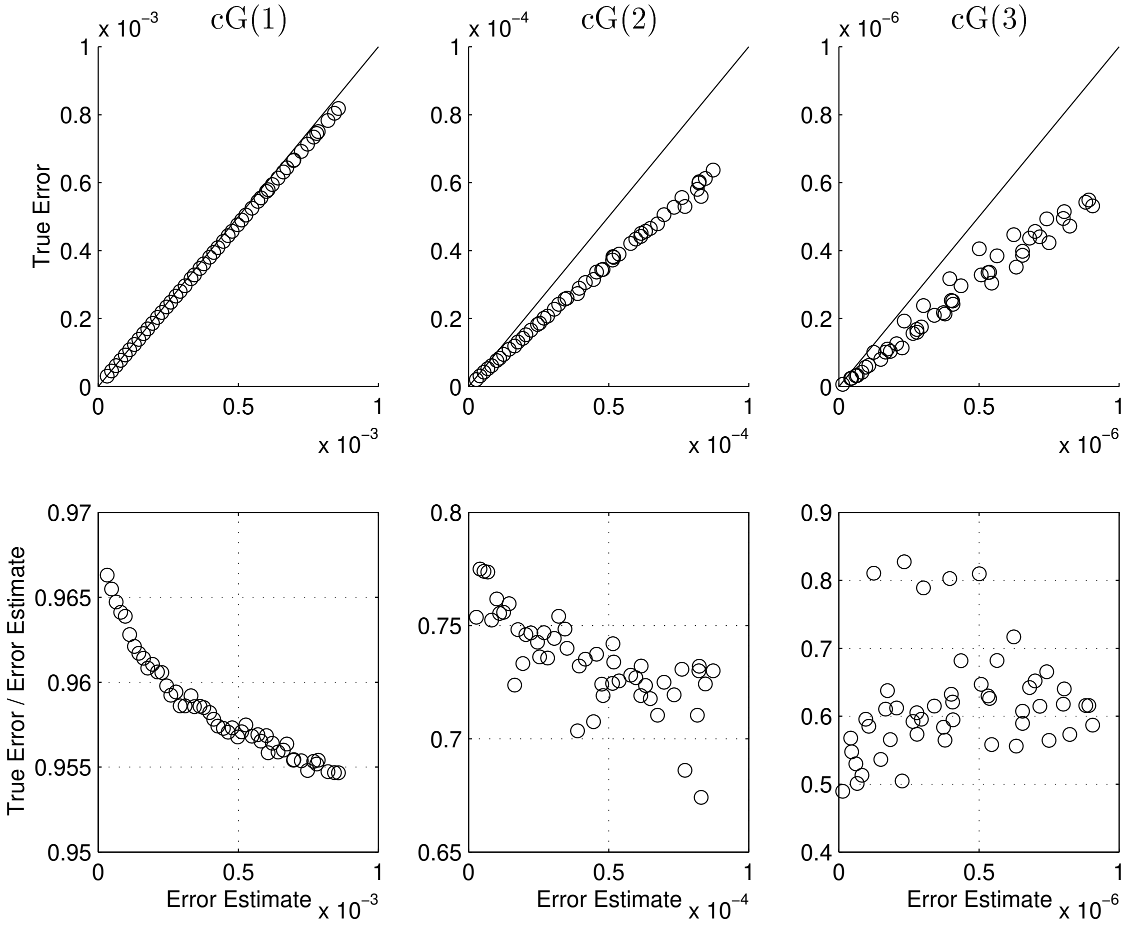



**Figure 3:** Errors at different tolerances for the solution of the harmonic oscillator problem with different methods, multi-adaptive cG(1), cG(2) and cG(3).

To check the error estimates further, we solve again the simple linear test problem for $T = 10$ with a number of different methods, and for a number of different tolerance levels. We do this simply by choosing different fixed and constant values for the time-steps. Although this means we have the same time-step for all components, the method is still the same multi-adaptive method; we do not take any advantage of the fact that we have uniform time-steps. From the solvers point of view, this does not matter. Running the test[1] for $q \leq 10$, we get the results presented in Tables 1 and 2.

To eliminate the uncertainty of guessing data for the dual, i.e. choosing a value for $\varphi(T)$ with $\|\varphi(T)\| = 1$, we choose here $\varphi(T) = (1,0)$ to have error control of the first component, i.e. $|||e||| = |e_1(T)|$ with the notation used in [7]. In Tables 1 and 2 we use the notation $|e| = |e_1(T)|$. This error is compared to the error estimates $\bar{E}_0 = E_0 + E_C + E_Q$ and $\bar{E}_2 = E_2 + E_C + E_Q$, as defined in [7].

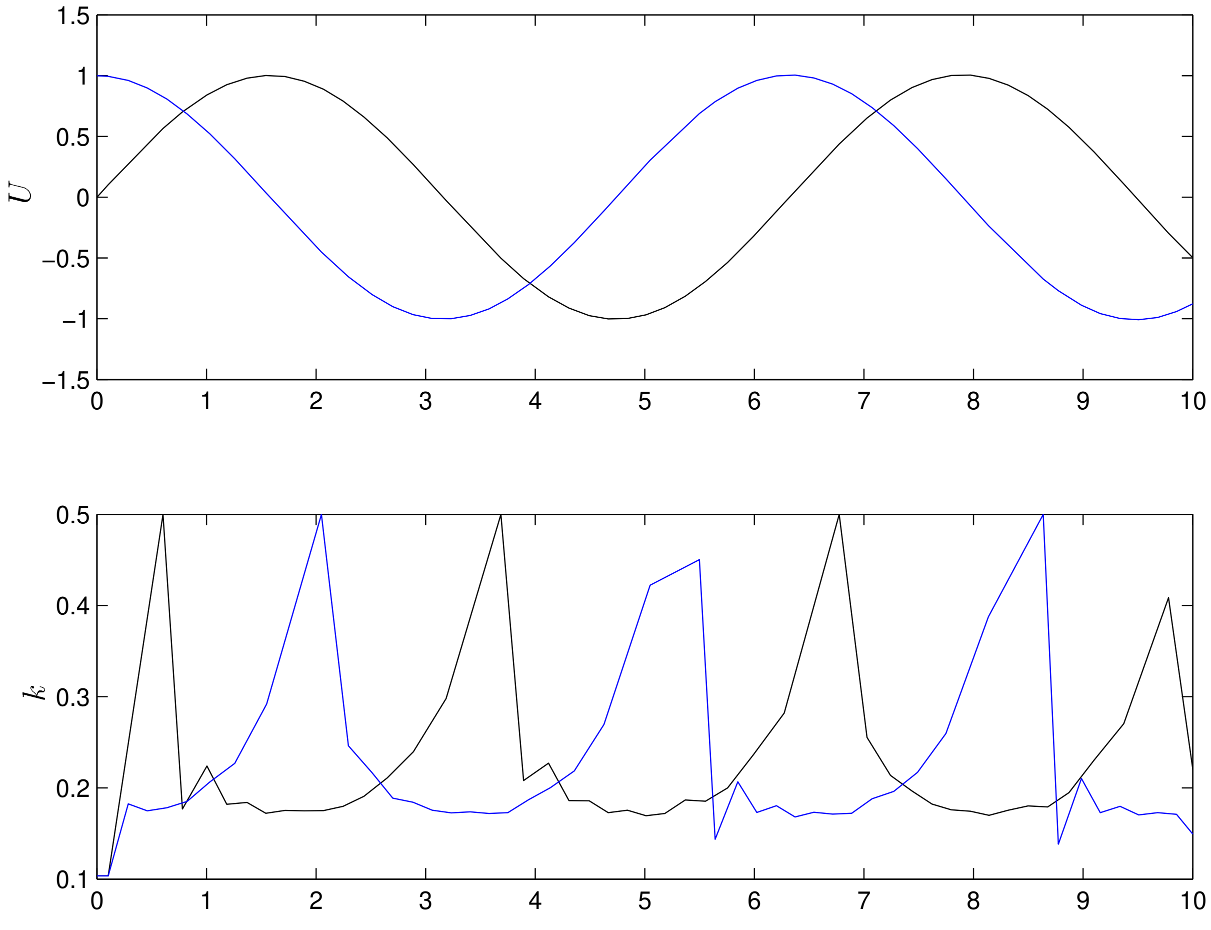



**Figure 4:** Solution and time-steps for the mcG(1) solution of the test problem with TOL = 0.1.

[1] Try this yourself by running '`./checkerror constant`' in the `src/demo/` directory of *Tanganyika*.

| $k$ | $\|e\|$ | $\bar{E}_0$ | $\bar{E}_2$ | $\|e\|/\bar{E}_0$ | $\|e\|/\bar{E}_2$ | $q$ | $2q$ |
|---|---|---|---|---|---|---|---|
| 1.000e-01 | 7.00e-03 | 7.01e-03 | 2.26e-02 | 0.998 | 0.309 | 1 | 2 |
| 5.000e-02 | 1.75e-03 | 1.75e-03 | 5.59e-03 | 1.000 | 0.313 | 1 | 2 |
| 2.500e-02 | 4.37e-04 | 4.37e-04 | 1.39e-03 | 1.000 | 0.315 | 1 | 2 |
| 1.000e-01 | 1.22e-06 | 1.23e-06 | 1.46e-05 | 0.998 | 0.084 | 2 | 4 |
| 5.000e-02 | 7.46e-08 | 7.47e-08 | 8.98e-07 | 0.999 | 0.083 | 2 | 4 |
| 2.500e-02 | 4.61e-09 | 4.61e-09 | 5.57e-08 | 1.000 | 0.083 | 2 | 4 |
| 1.000e-01 | 7.33e-08 | 7.42e-08 | 7.72e-08 | 0.988 | 0.950 | 3 | 6 |
| 5.000e-02 | 2.25e-09 | 2.25e-09 | 2.30e-09 | 0.998 | 0.978 | 3 | 6 |
| 2.500e-02 | 7.16e-11 | 6.93e-11 | 7.00e-11 | 1.033 | 1.022 | 3 | 6 |
| 1.000e-01 | 7.35e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 4 | 8 |
| 5.000e-02 | 2.24e-09 | 2.25e-09 | 2.25e-09 | 0.993 | 0.993 | 4 | 8 |
| 2.500e-02 | 6.82e-11 | 6.95e-11 | 6.95e-11 | 0.981 | 0.981 | 4 | 8 |
| 1.000e-01 | 7.34e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 5 | 10 |
| 5.000e-02 | 2.24e-09 | 2.25e-09 | 2.25e-09 | 0.995 | 0.995 | 5 | 10 |
| 2.500e-02 | 7.07e-11 | 6.93e-11 | 6.93e-11 | 1.020 | 1.020 | 5 | 10 |
| 1.000e-01 | 7.34e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 6 | 12 |
| 5.000e-02 | 2.23e-09 | 2.25e-09 | 2.25e-09 | 0.991 | 0.991 | 6 | 12 |
| 2.500e-02 | 6.80e-11 | 6.95e-11 | 6.95e-11 | 0.979 | 0.979 | 6 | 12 |
| 1.000e-01 | 7.35e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 7 | 14 |
| 5.000e-02 | 2.24e-09 | 2.25e-09 | 2.25e-09 | 0.996 | 0.996 | 7 | 14 |
| 2.500e-02 | 6.75e-11 | 6.95e-11 | 6.95e-11 | 0.971 | 0.971 | 7 | 14 |
| 1.000e-01 | 7.35e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 8 | 16 |
| 5.000e-02 | 2.25e-09 | 2.25e-09 | 2.25e-09 | 0.997 | 0.997 | 8 | 16 |
| 2.500e-02 | 7.14e-11 | 6.95e-11 | 6.95e-11 | 1.028 | 1.028 | 8 | 16 |
| 1.000e-01 | 7.34e-08 | 7.41e-08 | 7.41e-08 | 0.991 | 0.991 | 9 | 18 |
| 5.000e-02 | 2.25e-09 | 2.25e-09 | 2.25e-09 | 0.996 | 0.996 | 9 | 18 |
| 2.500e-02 | 7.06e-11 | 6.93e-11 | 6.93e-11 | 1.019 | 1.019 | 9 | 18 |
| 1.000e-01 | 7.35e-08 | 7.41e-08 | 7.41e-08 | 0.992 | 0.992 | 10 | 20 |
| 5.000e-02 | 2.24e-09 | 2.25e-09 | 2.25e-09 | 0.995 | 0.995 | 10 | 20 |
| 2.500e-02 | 7.23e-11 | 6.93e-11 | 6.93e-11 | 1.043 | 1.043 | 10 | 20 |

**Table 1:** Errors and error estimates for the mcG($q$) method with $1 \leq q \leq 10$ for the harmonic oscillator.

| $k$ | $\|e\|$ | $\bar{E}_0$ | $\bar{E}_2$ | $\|e\|/\bar{E}_0$ | $\|e\|/\bar{E}_2$ | $q$ | $2q+1$ |
|---|---|---|---|---|---|---|---|
| 1.000e-01 | 2.30e-01 | 2.55e-01 | 6.06e-01 | 0.904 | 0.380 | 0 | 1 |
| 5.000e-02 | 1.26e-01 | 1.33e-01 | 3.30e-01 | 0.945 | 0.381 | 0 | 1 |
| 2.500e-02 | 6.54e-02 | 6.74e-02 | 1.71e-01 | 0.971 | 0.383 | 0 | 1 |
| 1.000e-01 | 7.85e-05 | 7.45e-05 | 4.99e-04 | 1.054 | 0.157 | 1 | 3 |
| 5.000e-02 | 9.63e-06 | 9.35e-06 | 6.22e-05 | 1.030 | 0.155 | 1 | 3 |
| 2.500e-02 | 1.19e-06 | 1.17e-06 | 7.73e-06 | 1.021 | 0.154 | 1 | 3 |
| 1.000e-01 | 7.81e-08 | 8.12e-08 | 1.84e-07 | 0.962 | 0.423 | 2 | 5 |
| 5.000e-02 | 2.43e-09 | 2.55e-09 | 5.73e-09 | 0.956 | 0.424 | 2 | 5 |
| 2.500e-02 | 8.07e-11 | 8.10e-11 | 1.79e-10 | 0.997 | 0.450 | 2 | 5 |
| 1.000e-01 | 7.33e-08 | 7.01e-08 | 7.01e-08 | 1.045 | 1.045 | 3 | 7 |
| 5.000e-02 | 2.26e-09 | 2.27e-09 | 2.27e-09 | 0.995 | 0.995 | 3 | 7 |
| 2.500e-02 | 6.88e-11 | 6.97e-11 | 6.97e-11 | 0.987 | 0.987 | 3 | 7 |
| 1.000e-01 | 7.35e-08 | 7.00e-08 | 7.00e-08 | 1.050 | 1.050 | 4 | 9 |
| 5.000e-02 | 2.25e-09 | 2.26e-09 | 2.26e-09 | 0.995 | 0.995 | 4 | 9 |
| 2.500e-02 | 7.12e-11 | 7.13e-11 | 7.13e-11 | 0.998 | 0.998 | 4 | 9 |
| 1.000e-01 | 7.34e-08 | 6.99e-08 | 6.99e-08 | 1.050 | 1.050 | 5 | 11 |
| 5.000e-02 | 2.26e-09 | 2.27e-09 | 2.27e-09 | 0.995 | 0.995 | 5 | 11 |
| 2.500e-02 | 6.80e-11 | 7.09e-11 | 7.09e-11 | 0.959 | 0.959 | 5 | 11 |
| 1.000e-01 | 7.34e-08 | 7.00e-08 | 7.00e-08 | 1.049 | 1.049 | 6 | 13 |
| 5.000e-02 | 2.24e-09 | 2.26e-09 | 2.26e-09 | 0.994 | 0.994 | 6 | 13 |
| 2.500e-02 | 6.70e-11 | 7.15e-11 | 7.15e-11 | 0.937 | 0.937 | 6 | 13 |
| 1.000e-01 | 7.34e-08 | 6.99e-08 | 6.99e-08 | 1.050 | 1.050 | 7 | 15 |
| 5.000e-02 | 2.24e-09 | 2.25e-09 | 2.25e-09 | 0.993 | 0.993 | 7 | 15 |
| 2.500e-02 | 7.00e-11 | 7.01e-11 | 7.01e-11 | 0.998 | 0.998 | 7 | 15 |
| 1.000e-01 | 7.34e-08 | 6.99e-08 | 6.99e-08 | 1.050 | 1.050 | 8 | 17 |
| 5.000e-02 | 2.23e-09 | 2.26e-09 | 2.26e-09 | 0.987 | 0.987 | 8 | 17 |
| 2.500e-02 | 7.13e-11 | 7.14e-11 | 7.14e-11 | 0.998 | 0.998 | 8 | 17 |
| 1.000e-01 | 7.35e-08 | 6.99e-08 | 6.99e-08 | 1.050 | 1.050 | 9 | 19 |
| 5.000e-02 | 2.25e-09 | 2.26e-09 | 2.26e-09 | 0.995 | 0.995 | 9 | 19 |
| 2.500e-02 | 7.13e-11 | 7.14e-11 | 7.14e-11 | 0.998 | 0.998 | 9 | 19 |
| 1.000e-01 | 7.35e-08 | 6.99e-08 | 6.99e-08 | 1.050 | 1.050 | 10 | 21 |
| 5.000e-02 | 2.25e-09 | 2.26e-09 | 2.26e-09 | 0.995 | 0.995 | 10 | 21 |
| 2.500e-02 | 7.08e-11 | 7.10e-11 | 7.10e-11 | 0.998 | 0.998 | 10 | 21 |

**Table 2:** Errors and error estimates for the mdG($q$) method with $0 \leq q \leq 10$ for the harmonic oscillator.

2.2. **A posteriori error estimates.** The Galerkin part of the sharpest error estimate, $\bar{E}_0$, is

$$E_0 = \left| \sum_{i=1}^{N} \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(\varphi_i - \pi_k \varphi_i)\ dt \right|, \tag{3}$$

for both the mcG($q$) and the mdG($q$) method, for the special choices of interpolants discussed in [7]. For the mdG($q$) method this means that the jump terms disappear, and for both methods we thus have

$$\left| \int_{I_{ij}} R_i(\varphi_i - \pi_k \varphi_i)\ dt \right| = \int_{I_{ij}} |R_i||\varphi_i - \pi_k \varphi_i|\ dt = C'_{q_{ij}} k_{ij} |R_i(t_{ij}^-)||\varphi_i(t_{ij}^-) - \pi_k \varphi_i(t_{ij}^-)|, \tag{4}$$

where the $\{C'_q\}$ are computable constants, different for the two methods. We also keep track of the actual sign of $\int_{I_{ij}} R_i(\varphi_i - \pi_k \varphi_i)\ dt$ so that we may actually take into account cancellation over intervals and between different components. In this way we are able to evaluate the error exactly, including its sign, if we know the dual well enough. Note that this does not mean that the dual has to be solved on a finer grid or with a higher-order method — although for the discontinuous method it is natural to solve the dual with one order higher polynomials. By choosing the interpolant in this special way we avoid the problem of having to evaluate $\int_{I_{ij}} R_i \varphi_i\ dt$ accurately, which is otherwise necessary if we want to introduce as few inequalities as possible in the error estimate. Evaluating this quantity directly is difficult, since the residual is, by the Galerkin orthogonality, orthogonal to a large part of the dual.

The Galerkin part of the second error estimate, $\bar{E}_2$, is

$$E_2 = \sum_{i=1}^{N} \sum_{j=1}^{M_i} C_{q_{ij}-1} s_{ij}^{[q_{ij}]} k_{ij}^{q_{ij}+1} r_{ij}, \tag{5}$$

for the mcG($q$) method, while for the mdG($q$) method we have

$$E_2 = \sum_{i=1}^{N} \sum_{j=1}^{M_i} C_{q_{ij}} s_{ij}^{[q_{ij}+1]} k_{ij}^{q_{ij}+2} \bar{r}_{ij}, \tag{6}$$

for certain interpolation constants $\{C_q\}$, residuals $\{r_{ij}, \bar{r}_{ij}\}$, and stability weights $\{s_{ij}^{[p]}\}$, as described in [7]. Here we have introduced a number of inequalities and on top of that interpolation estimates. We may thus expect this second estimate to be less precise than the first estimate. This is also the case, according to the results presented in Tables 1 and 2. For smaller time-steps and higher-order methods, this agrees increasingly well with the more delicate estimate, the reason being that the computational error (see [7]) dominates, and this is estimated in the same way in both $\bar{E}_0$ and $\bar{E}_2$.

Now, why do we keep the time-steps constant and equal? The answer is, to eliminate a part of the error that is difficult to estimate, namely *quadrature errors*. Using a quadrature rule that agrees with the order of the method, as described in [7], we should be able to keep this error zero. If, however, we allow for different time-steps for different components, we will get quadrature errors. To see this, think of the simplest possible case, a system of two ODEs, and time-steps $k_{1j} = \{1\}$ and $k_{2j} = \{0.5, 0.5\}$ on $[0, 1]$. Computing $\int_{I_{11}} f_1(U, t)\ dt = \int_0^1 f_1(U, t)\ dt$ with a quadrature rule that fits the first component, we will not capture the variation of the second component, since this component, although piecewise linear (say), will not be piecewise linear on the partition of the first component, but rather on its own partition. Individual time-steps will thus result in additional quadrature errors if we don't do something about the quadrature. Adaptive quadrature and estimation of quadrature errors are discussed in [7], but although we may, in principle, estimate the quadrature errors, we may not expect these estimates to be of the same quality as the rest of the error.

To illustrate this, we solve the test problem again, using the mcG(1) method with TOL $= 0.1$. We obtain the solution and time-step sequences presented in Figure 2.1. If the time-step sequences seem a little wiggly, this is because the time-steps for this tolerance are quite large compared to the period of the oscillations. As noted before, the time-steps are large when the solution is close to linear.

Plotting the residual — see Figure 5 — we see that it behaves as we expect it to do, as a first-order Legendre polynomial, i.e. being zero in the middle of the interval, on every local interval. We also note that the residual is not exactly a Legendre polynomial, since the time-steps are no longer equal. The projection of the residuals onto the individual trial spaces are, however, Legendre polynomials, as we noted in [7].

Choosing now the interpolant in a suitable way, for the mcG(1) method as the piecewise constant polynomial that interpolates the dual at the midpoint of every interval, the product $R_i(\varphi_i - \pi_k\varphi_i)$ does not change sign on any of the local intervals, provided the dual can be well-represented locally by a first-order polynomial. This is also the case, according to Figure 5.

Assuming exact quadrature and exact solution of the discrete equations, the Galerkin orthogonality should hold, which for the mcG(1) solution means that we should have

$$\int_{I_{ij}} R_i\ dt = 0, \tag{7}$$

for every local interval $I_{ij}$, so that the functions

$$\mathcal{R}_i(t) = \int_0^t R_i(s)\ ds, \tag{8}$$

should be zero at the nodal points, as it is for the solution with constant time-steps — see Figure 7. This will not hold if quadrature and computational errors are not zero, as is evident from Figure 6. The functions $\mathcal{R}_i$ are not zero at the nodal points, and so the Galerkin orthogonality does not hold. Correspondingly, the quadrature and computational parts of the a posteriori error estimates presented in [7] are non-zero, and we should still be able to estimate the error properly. The error (for the first component at time $t = T$) is now $|e| = 0.032$, and the solver reports error estimates $E_0 = 0.036$ and $E_2 = 0.094$.

The conclusion is thus, that by keeping other errors than the Galerkin error small, we get very good estimates for the error, since the pure Galerkin error is easy to estimate. Allowing also computational errors, including round-off error, and quadrature errors on the other hand, the error will be more difficult to estimate well. All sources of the error are equally important and moving from one problem to the other, any of these may be the most dominant and so, the most important. Later, we will consider the Lorenz system, for which the computational error, in this case accumulated finite precision arithmetic round-off error, is the most important to take into consideration, and for which the Galerkin error is negligible.

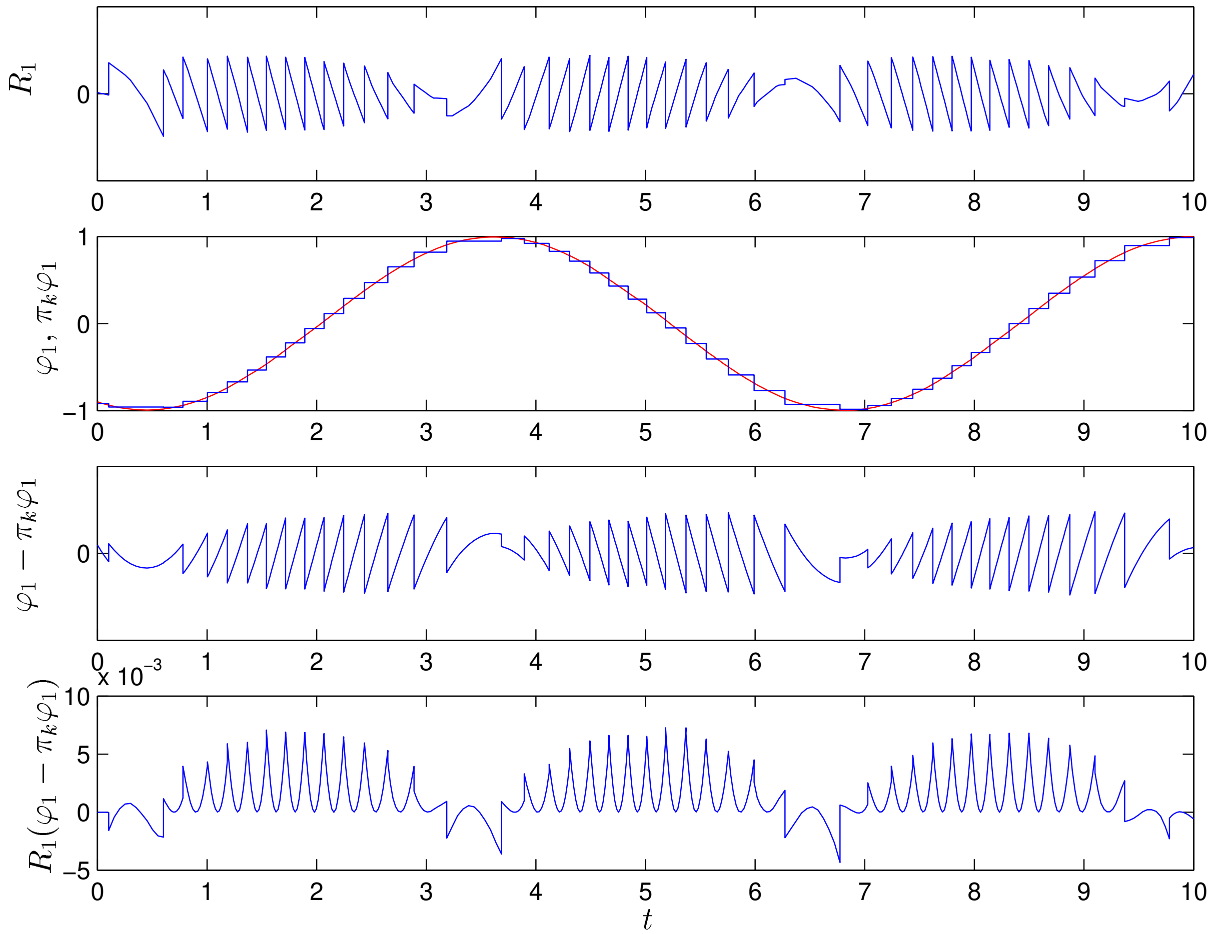



**Figure 5:** The residual $R_1$, the dual $\varphi_1$ and its piecewise constant interpolant $\pi_k\varphi_1$ for the first component of the mcG(1) solution of the test problem with $\mathrm{TOL} = 0.1$.

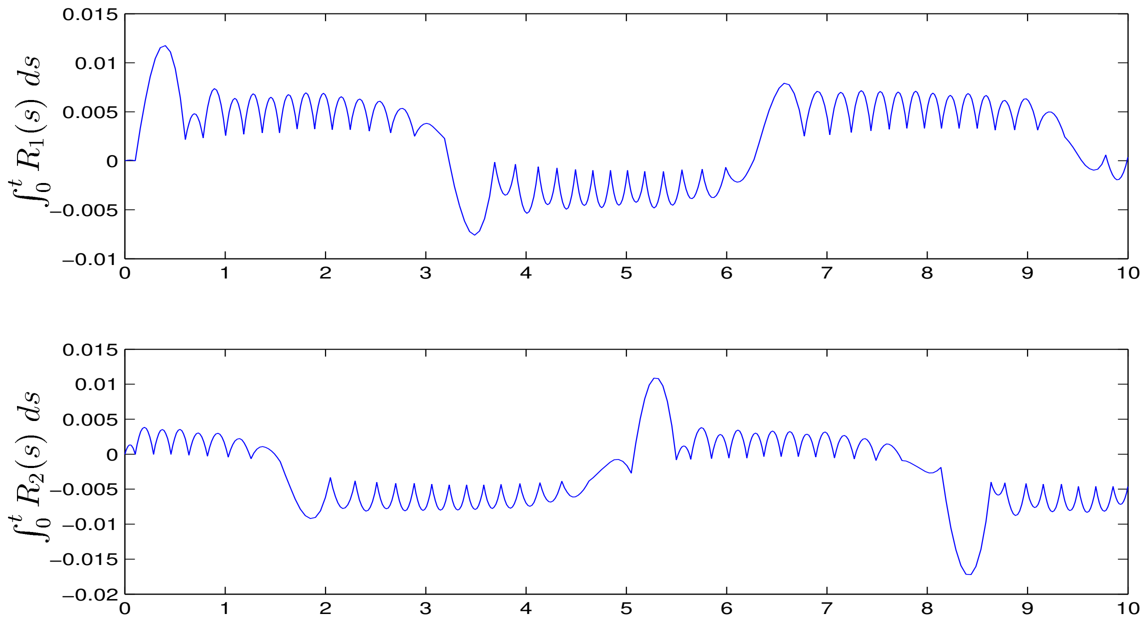



**Figure 6:** Integrated residuals for the two components of the mcG(1) solution to the test problem with TOL $= 0.1$.

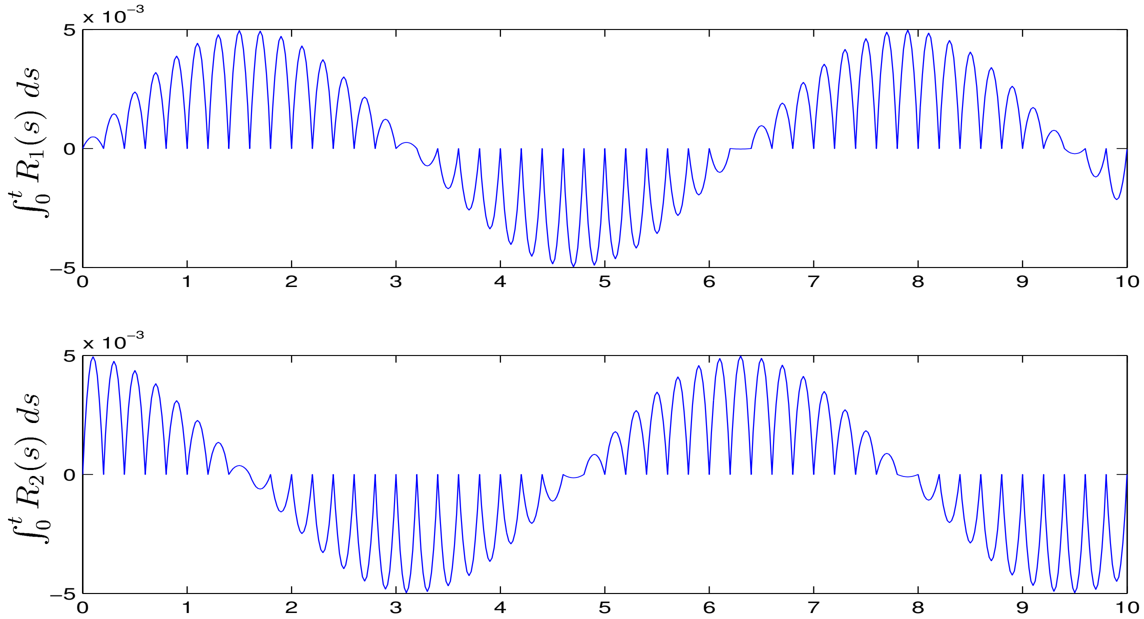



**Figure 7:** Integrated residuals for the first component of the mcG(1) solution of the test problem with $k = 0.2$.

## 3. A Simple Non-Linear Problem

We now compare the error estimate with the actual error in following non-linear problem,

$$\left\{\begin{array}{rcl} \dot{u}_1 & = & u_1, \\ \dot{u}_2 & = & u_2 + u_1 u_1, \\ \dot{u}_3 & = & u_3 + u_1 u_2, \\ \dot{u}_4 & = & u_4 + u_1 u_3 + u_2 u_2, \\ \dot{u}_5 & = & u_5 + u_1 u_4 + u_2 u_3, \end{array}\right. \tag{9}$$

where $u(0) = (1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4})$ with exact solution $u(t) = (e^t, e^{2t}, \frac{1}{2}e^{3t}, \frac{1}{2}e^{4t}, \frac{1}{4}e^{5t})$. We solve using the mcG(1) method with different tolerances. Again we have a good correspondence between error estimate and actual error.

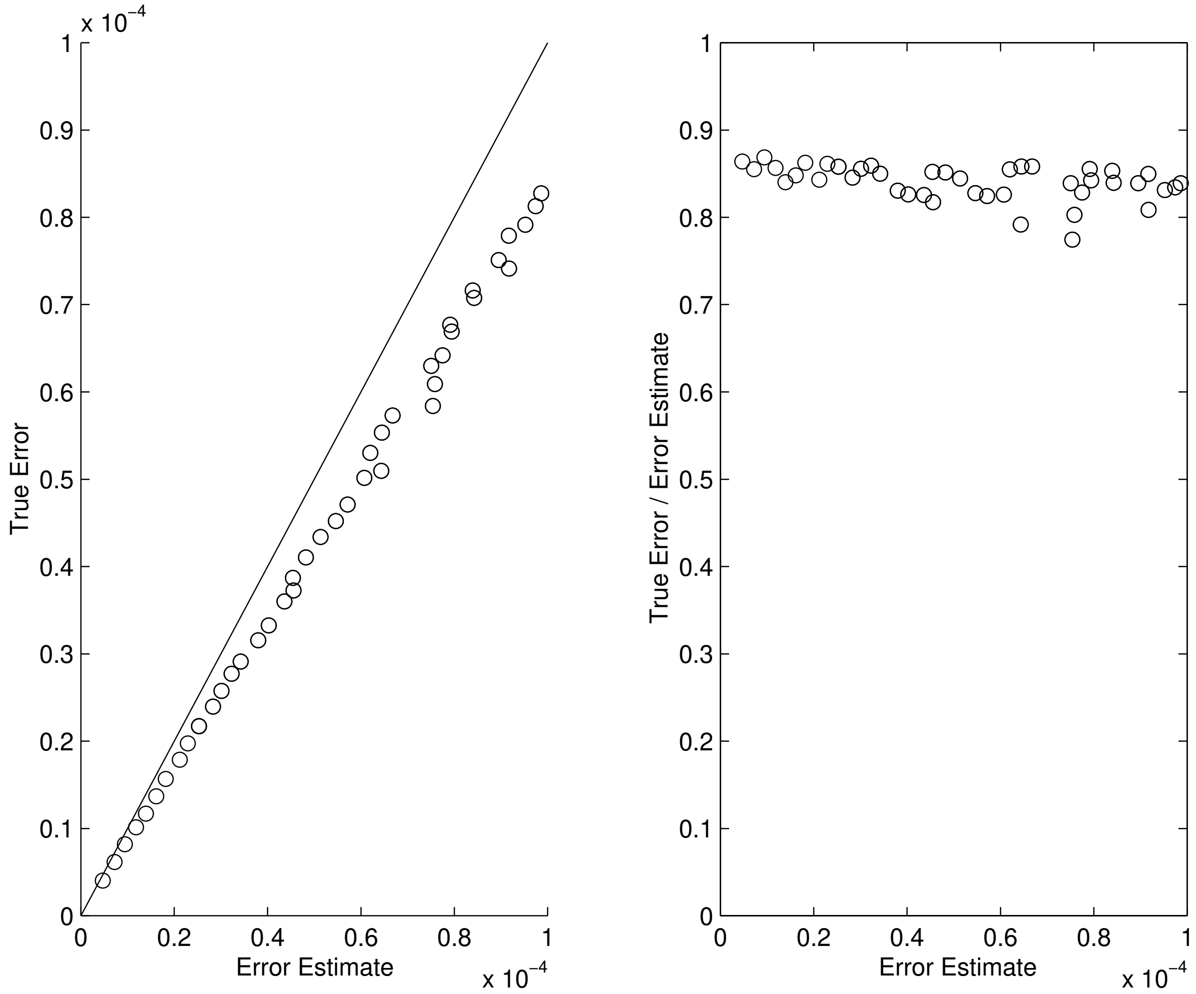



**Figure 8:** Errors at different tolerance levels for the multi-adaptive solution of (9).

## 4. The Lorenz System

We consider now the famous Lorenz system,

$$\left\{\begin{array}{rcl} \dot{x} &=& \sigma(y-x), \\ \dot{y} &=& rx - y - xz, \\ \dot{z} &=& xy - bz, \end{array}\right. \tag{10}$$

with the usual data $(x(0), y(0), z(0)) = (1, 0, 0)$, $\sigma = 10$, $b = 8/3$ and $r = 28$, see e.g. [2]. The solution $u(t) = (x(t), y(t), z(t))$ is very sensitive to perturbations and is often described as being "chaotic". With our perspective this is reflected by stability factors with rapid growth in time.

The computational challenge is to solve the Lorenz system accurately on a time interval $[0, T]$ with $T$ as large as possible. We investigate the computability of the Lorenz system by solving the dual problem and computing stability factors, to find out the maximum value of $T$.

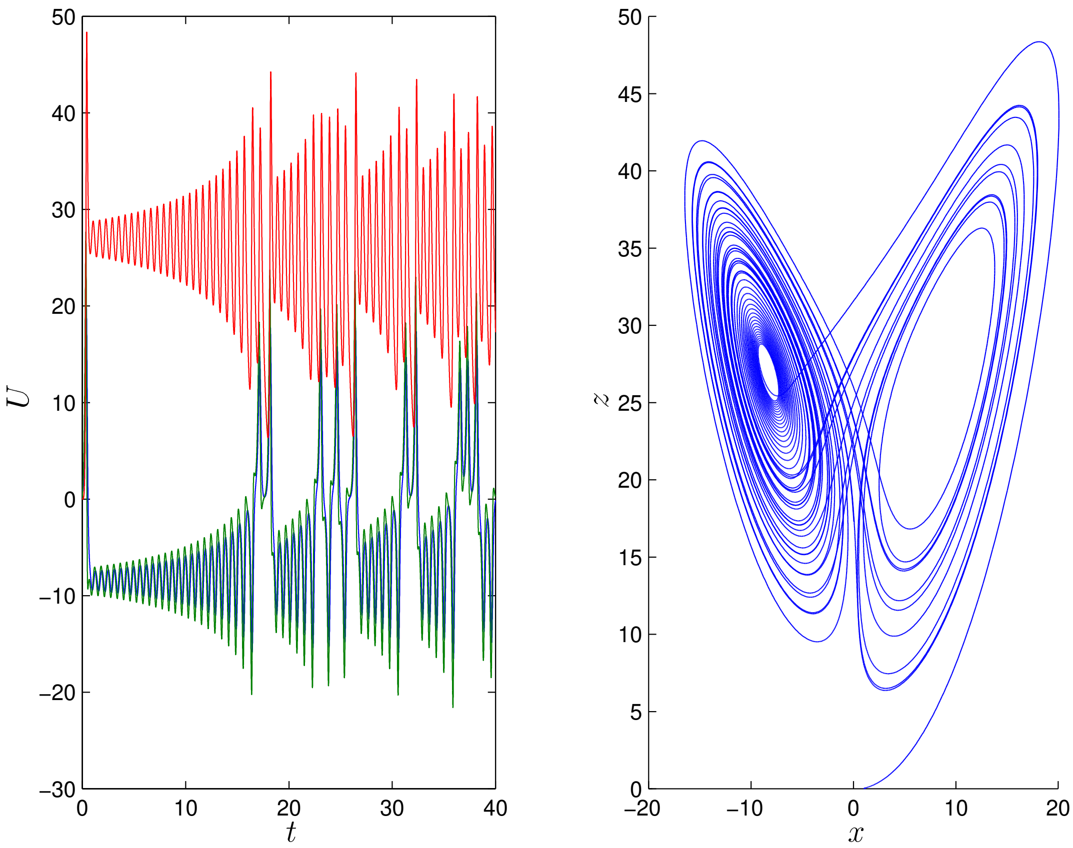



**Figure 9:** On the right is the trajectory of the Lorenz system for final time $T = 40$, computed with the multi-adaptive cG(5) method. On the left is a plot of the time variation of the three components.

4.1. **A simple experiment.** To begin with, we examine solutions of the Lorenz system for a number of different methods. In Figure 10 we plot solutions for the $x$-component of the Lorenz system on $[0, 40]$. The time-step is the same for all components and constant equal to $k = 0.001$. Some time after $t = 20$, the second-order accurate cG(1) solution is no longer correct. (A good thing about the Lorenz system is that when the solution once starts to be incorrect, it quickly becomes very inaccurate also in picture-norm.)

Increasing the order of the method until we finally reach the 20:th and 21:st order methods cG(10) and dG(10), we get more and more accurate solutions, and are able to solve until $T = 40$. Increasing the order further does not increase the accuracy of the solution, since the error is now dominated by the computational error caused by finite precision arithmetic round-off error.

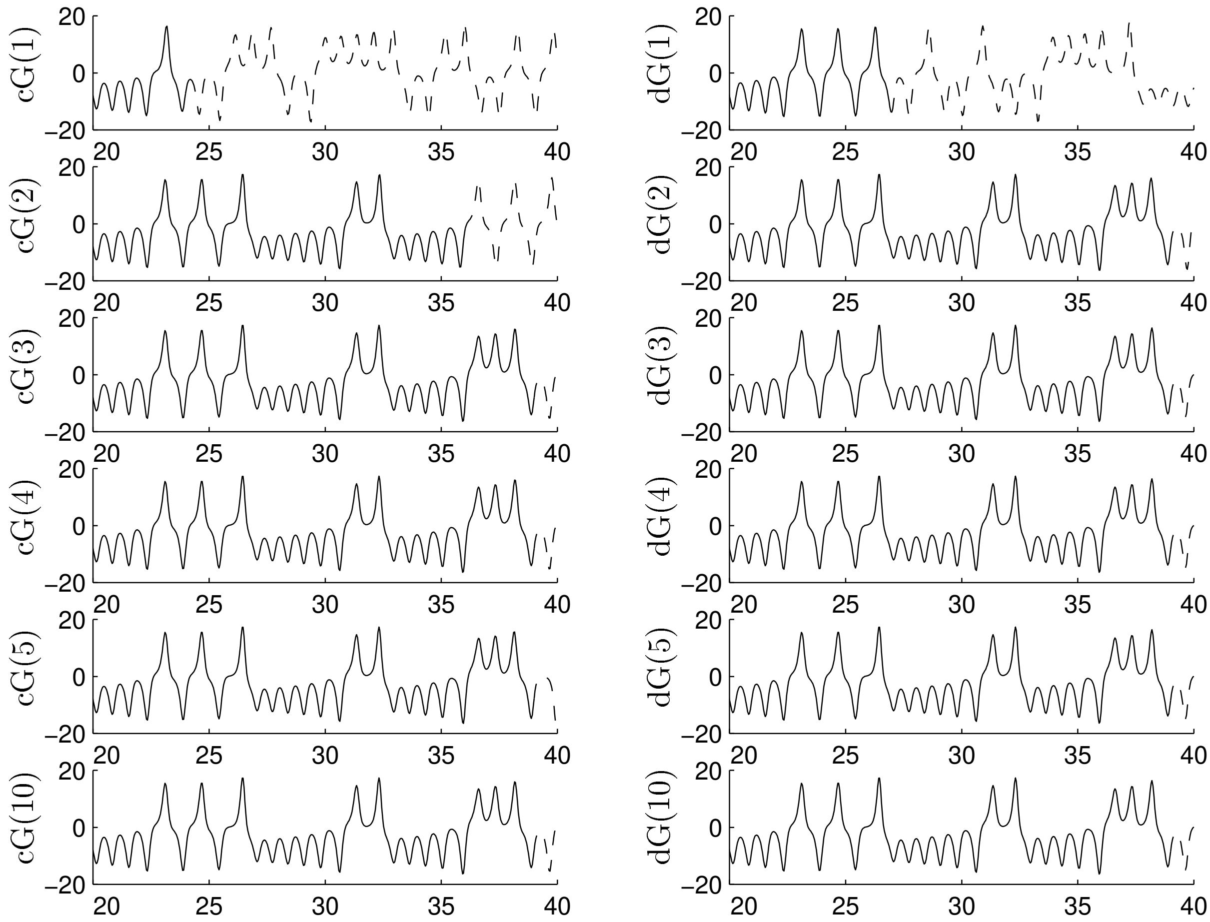



**Figure 10:** These plots show solutions, computed with different methods with constant time-step $k = 0.001$, for the $x$-component of the Lorenz system from time $t = 20$ to final time $T = 40$.

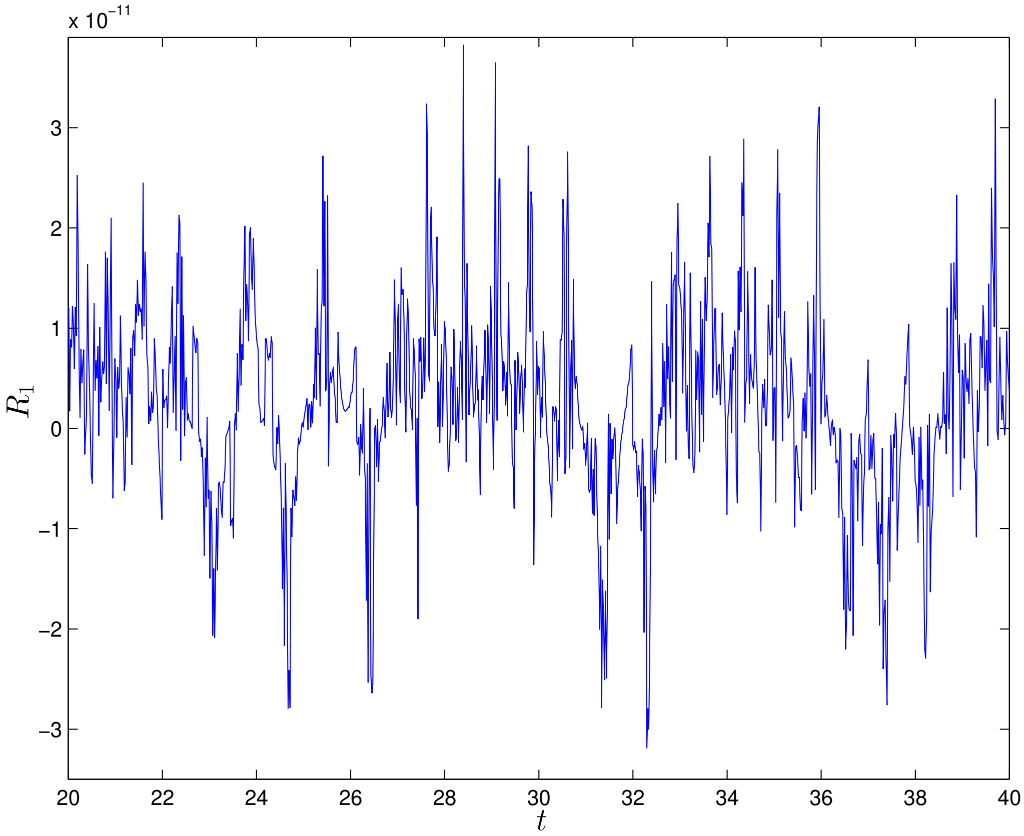



**Figure 11:** The cG(5) residual for the $x$-component of the Lorenz system.

4.2. **Reaching further.** As is evident in Figure 11 the residual is small already for the 10:th-order method cG(5), and this residual should be weighted with $k^5 = 10^{-15}$ in the error estimate. Also when multiplying with the stability factor we will get far below a tolerance of say TOL $= 0.1$. The Galerkin error is thus small enough. To get further than $T = 40$, we must thus do something else than decreasing the time-step or increasing the order of the method. We must decrease the computational error, the accumulated numerical round-off error.

A simple view of this is that at every time-step we will make a relative error of at least $10^{-16}$. These errors will accumulate at a rate determined by the stability properties of the dual. To decrease this error we must thus not decrease the time-step, but instead *increase the time-step*! In this way we will take fewer time-steps and so decrease the computational error. Since increasing the time-step we will increase the Galerkin error, we may perhaps have to increase also the order of the method to keep the Galerkin error small.

In Figure 12 we present solutions with $k = 0.1$. We are now able to solve until about $T = 50$, using as few as 500 time-steps. The higher order methods all agree until some point close to $t = 48$, and not even the 30:th-order method mcG(15) is able to get any further. This indicates that again we have reached a limit.

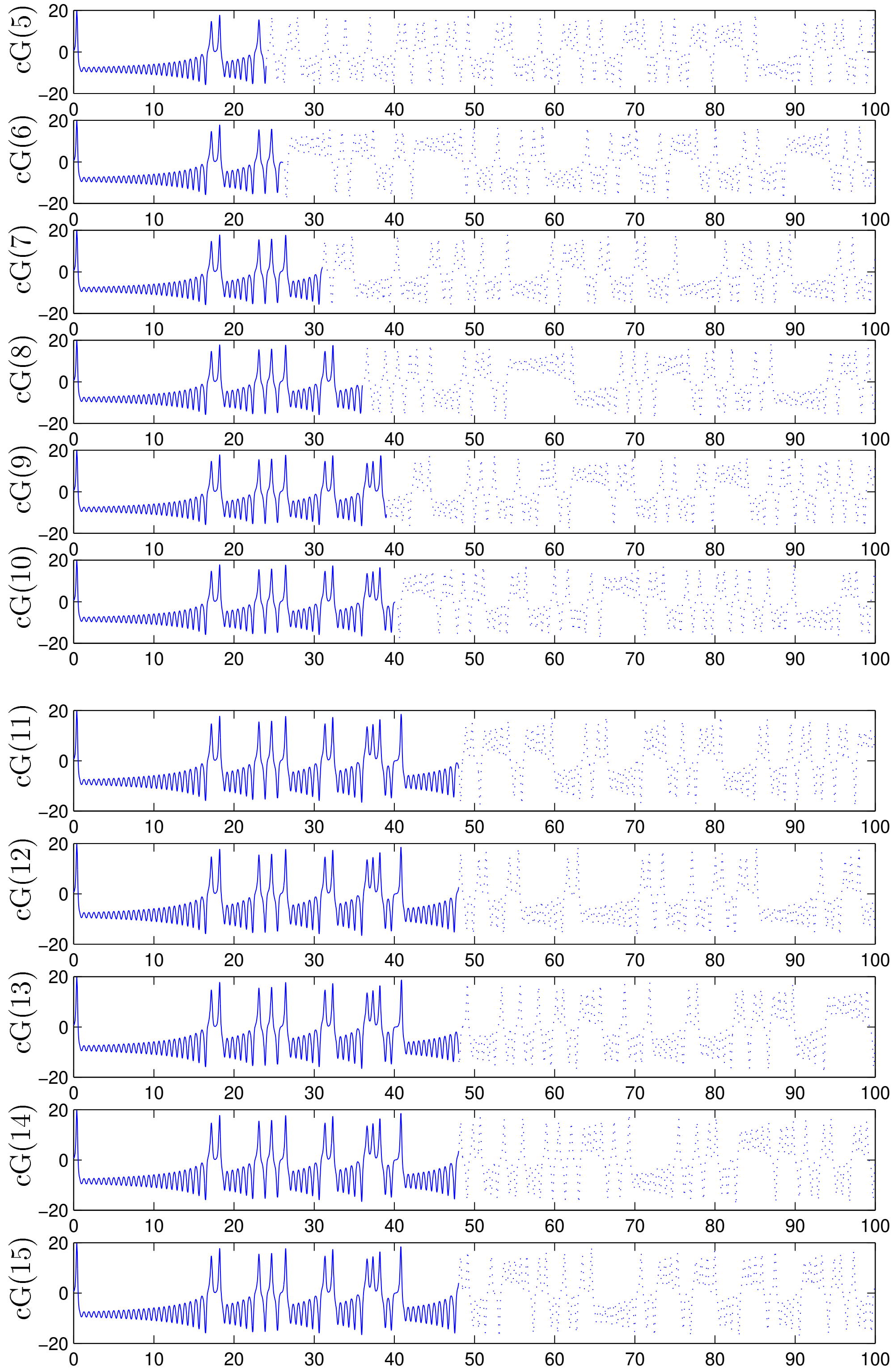



**Figure 12:** Solutions for the $x$-component of the Lorenz system with methods of different order, using a constant time-step $k = 0.1$.

4.3. **Comparison with `ode45`.** We compare these results with a standard ODE-solver, the Runge-Kutta solver `ode45`, that is shipped with the commercial program MATLAB. In this context `ode45` is a low-order solver, and we do not reach time $T = 40$ no matter how small a "tolerance" we choose. Decreasing the time-step further will only increase the numerical error, since the number of time-steps will then increase.

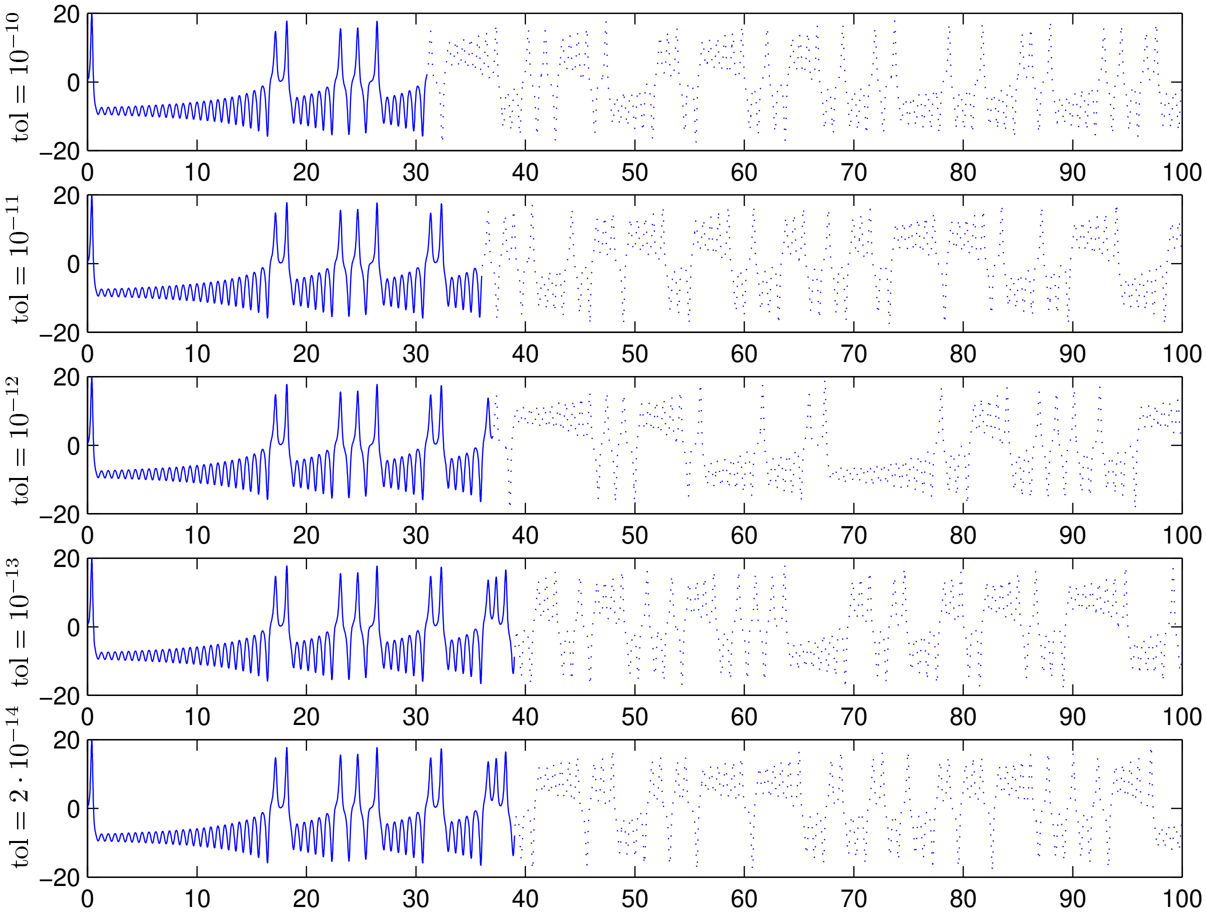



**Figure 13:** The `ode45` solution for the $x$-component of the Lorenz system at a number of different tolerance levels.

4.4. **The dual solution.** We now turn to the solution of the dual problem. The stability factors grow, as we shall see, exponentially with time, and so does the dual (backward in time).

For a small ODE-system such as the Lorenz system, which has only three components, we can afford, from a computational and storage point of view, to compute the fundamental solution to the linear dual problem, i.e. the matrix $\Phi(t)$ such that the dual $\varphi$ solved with

$\varphi(T) = \varphi_T$ is given by $\varphi(t) = \Phi(t)\varphi_T$. In Figure 14 we plot (the $l^2$-norms of the three rows of) the fundamental solution and its third derivative as function of time.

The time-steps for the solution of the dual are based on an a posteriori error estimate for the dual, presented in [7], to give a relative error for the global stability factor smaller than 10%. Here we have used a simple approximation of the matrix exponential for solving the linear dual problem, and so the time-steps are the same for all components of the dual. Note that for the dual, there are no stability factors present in the estimate for the relative error, and so the time-steps are completely determined by the regularity of the solution. This is also evident when examining Figure 14; about 10 time-steps per half period are needed to resolve the dual.

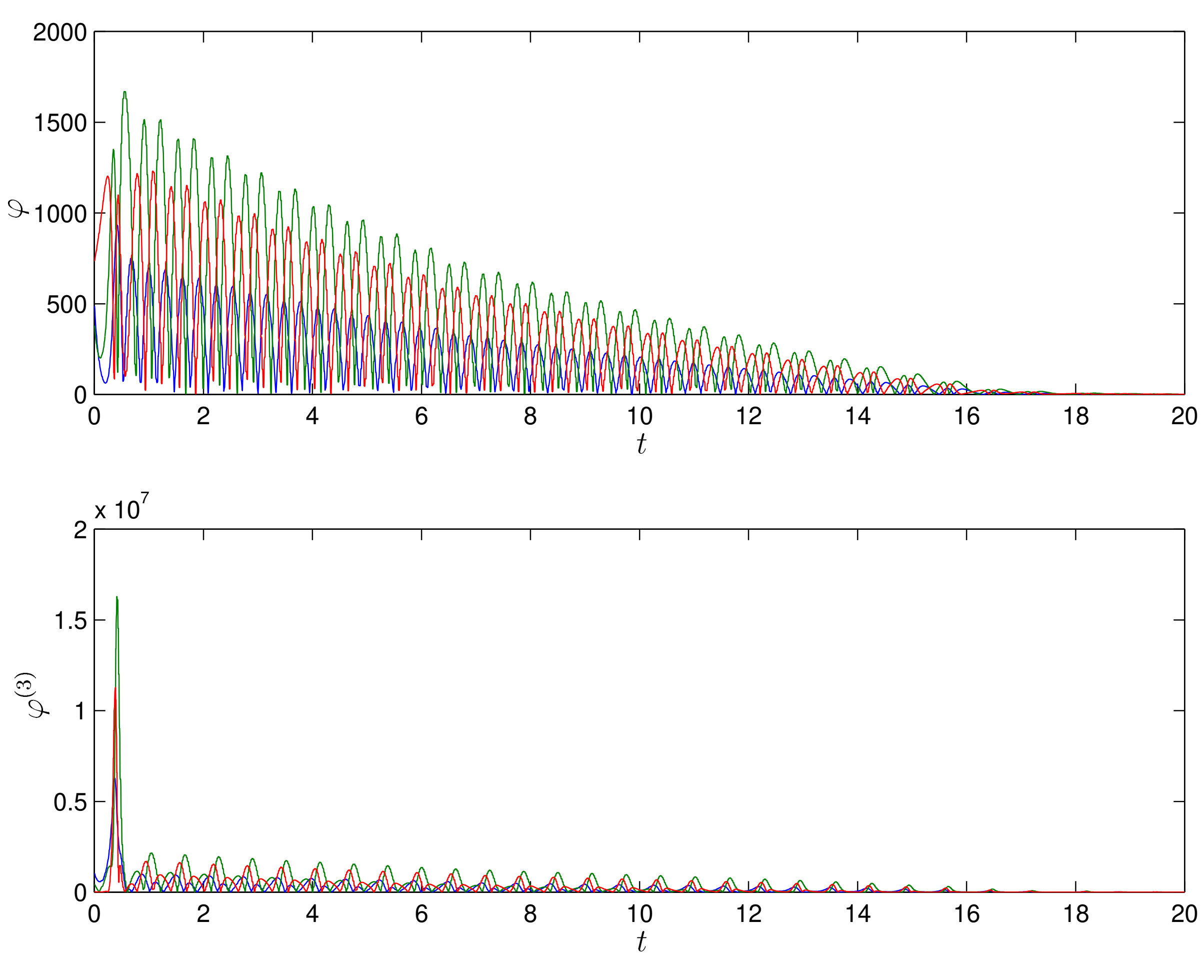



**Figure 14:** The dual for the Lorenz system (above) and its third derivative (below).

4.5. **Stability factors.** We now turn to the computation of stability factors for the Lorenz system, such as

$$S^{[q]}(T) = \max_{\|v\|=1} \int_0^T \|\varphi^{(q)}(t)\| \, dt, \tag{11}$$

where $\varphi$ is an (approximate) solution of the dual problem with $\varphi(T) = v$. Letting $\Phi$ be an (approximate) fundamental solution of the dual problem, we have

$$\max_{\|v\|=1} \int_0^T \|\Phi^{(q)}(t)v\| \, dt \leq \int_0^T \max_{\|v\|=1} \|\Phi^{(q)}(t)v\| \, dt = \int_0^T \|\Phi^{(q)}(t)\| \, dt = \bar{S}^{[q]}(T), \tag{12}$$

and thus computing $\bar{S}^{[q]}(T)$ gives a bound for $S^{[q]}(T)$, which for Lorenz system turns out to be quite sharp. By computing the fundamental solution we avoid computing the maximum in (11).

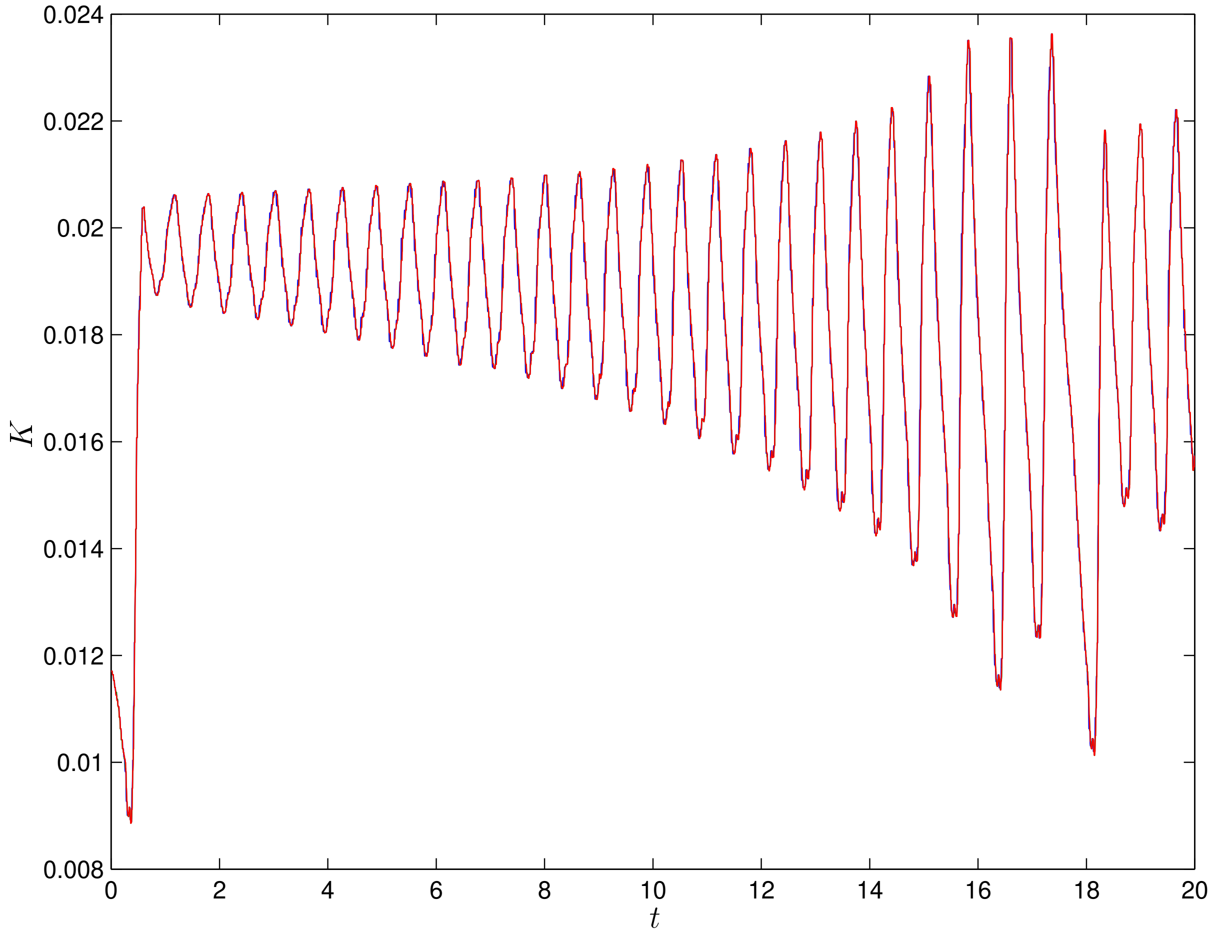



**Figure 15:** Time-steps for the dual problem based on an a posteriori error estimate for the dual.

We compute the variation of the stability factors on $[0, 50]$, see Figure 16, where we plot the stability factor for $q = 0$, corresponding to computational and quadrature errors. The stability factors grow exponentially with time, but not as fast as indicated by an a priori error estimate. An a priori error estimate indicates that the stability factors grow as

$$S^{[q]}(T) \sim e^{\mathcal{A}T}, \tag{13}$$

where $\mathcal{A}$ is some bound for the Jacobian of the right-hand side for the Lorenz system. Making a simple estimate, we take $\mathcal{A} = 50$, so that already at $T = 1$ we have $S(T) \approx 10^{22}$. In view of this, we would not be able to compute even to $T = 1$, and certainly not to $T = 50$ for which $S(T) \approx 10^{1000}$. The point is that although the stability factors grow very rapidly at some occasions, such as nearby the first flip at $T = 18$, they do not grow monotonically, and thus as an average grow at a moderate exponential rate.

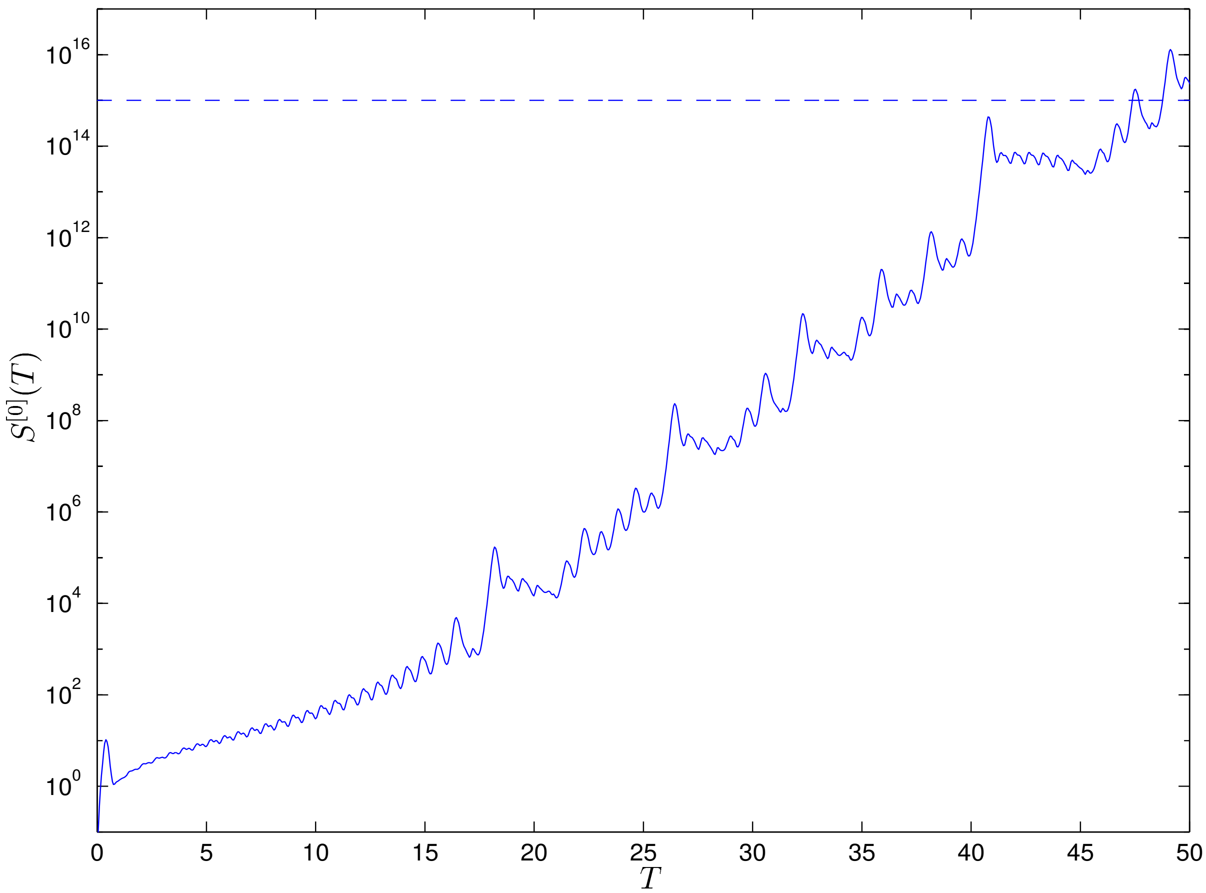



**Figure 16:** The stability factor for the computational and quadrature errors, as function of time for the Lorenz system.

4.6. **Conclusions.** We now make simple estimates for the growth rate of the stability factors, in order to predict how far along it is possible to compute. Fitting simple functions to the stability factors as function of time, we have the following approximations:

$$\tilde{S}^{[q]}(T) \approx 4 \cdot 10^{(q-3)+0.37T}, \tag{14}$$

or, simpler but not as good,

$$\tilde{S}^{[q]}(T) \approx 10^{q+T/3}. \tag{15}$$

From the a posteriori error estimates presented in [7], we find that the computational error can be estimated as

$$E_C \approx S^{[0]}(T) \max_{[0,T]} \|\mathcal{R}^{\mathcal{C}}\|, \tag{16}$$

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

$$\mathcal{R}_i^{\mathcal{C}}(t) = \frac{1}{k_{ij}} \left( U(t_{ij}) - U(t_{i,j-1}) - \int_{I_{ij}} f_i(U,\cdot)\ dt \right) \tag{17}$$

for the mcG($q$) method and similarly for the discontinuous method.

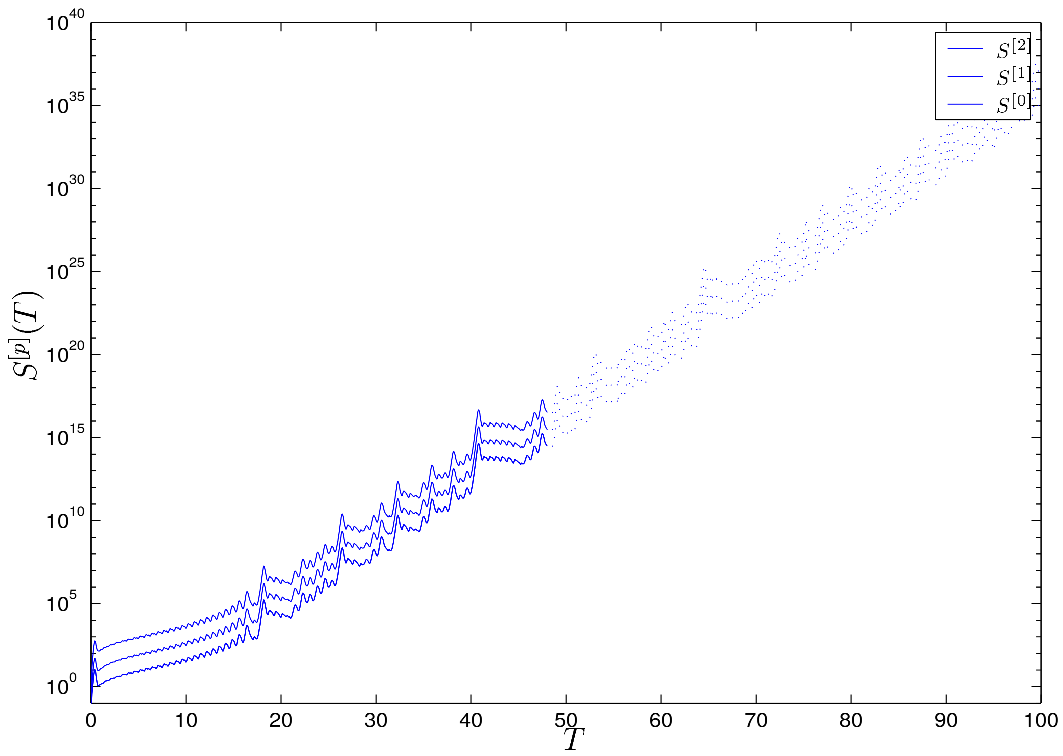



**Figure 17:** Different stability factors, zeroth, first and second order as defined in (12), for the Lorenz system.

With 16 digits of precision, we cannot expect to have a discrete residual smaller than about $\frac{1}{k_{ij}}10^{-16}$, so that, with the approximation above, we have

$$E_C \approx 10^{T/3}\frac{1}{\min k_{ij}}10^{-16} = 10^{T/3-16}\frac{1}{\min k_{ij}}, \tag{18}$$

so that with $k_{ij} = 0.1$ as above we have

$$E_C \approx 10^{T/3-15}. \tag{19}$$

With time-steps $k_{ij} = 0.1$ we thus cannot expect to do much better than $T = 50$, since then we will have an error larger than unity, which we take as a criterion for an incorrect solution. Increasing the time-step further, to say $k_{ij} = 10$, we can compute a little further, but only a couple of time units, since the stability factors grow exponentially. Increasing the time-step even further we will soon have a time-step that is greater than the length of the whole interval, i.e. $k > T$.

Our conclusion is thus that by examining the stability factors, we can say that our computation is probably correct until right before $T = 50$, and that we cannot get much further with 16 digits of precision. (With 32 digits of precision we would reach $T = 100$, and so on.)

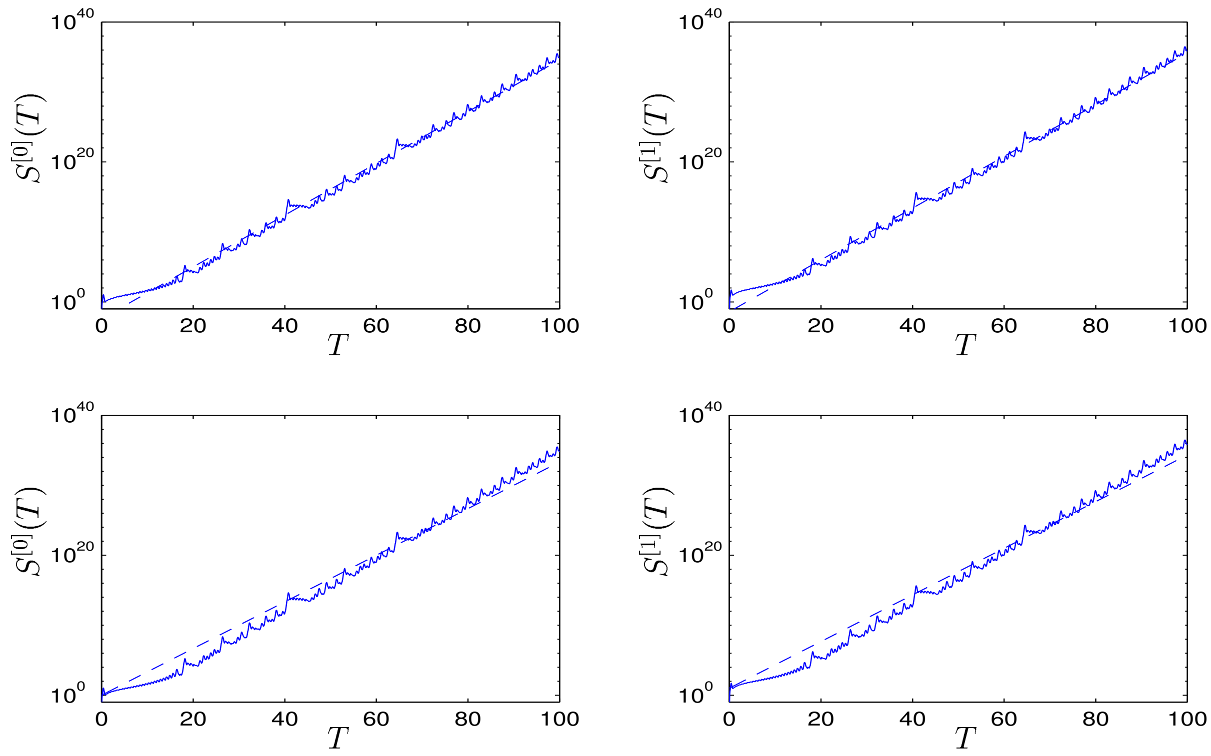



**Figure 18:** Fitting approximations to the Lorenz stability factors; above the approximation (14) and below the approximation (15).

## 5. The Solar System

We now consider the Solar System, including the Sun, the Moon, and the nine planets, which is a particular $n$-body problem of fundamental importance:

$$m_i \ddot{x}_i = \sum_{j \neq i} \frac{G m_i m_j}{|x_j - x_i|^3}(x_j - x_i), \tag{20}$$

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

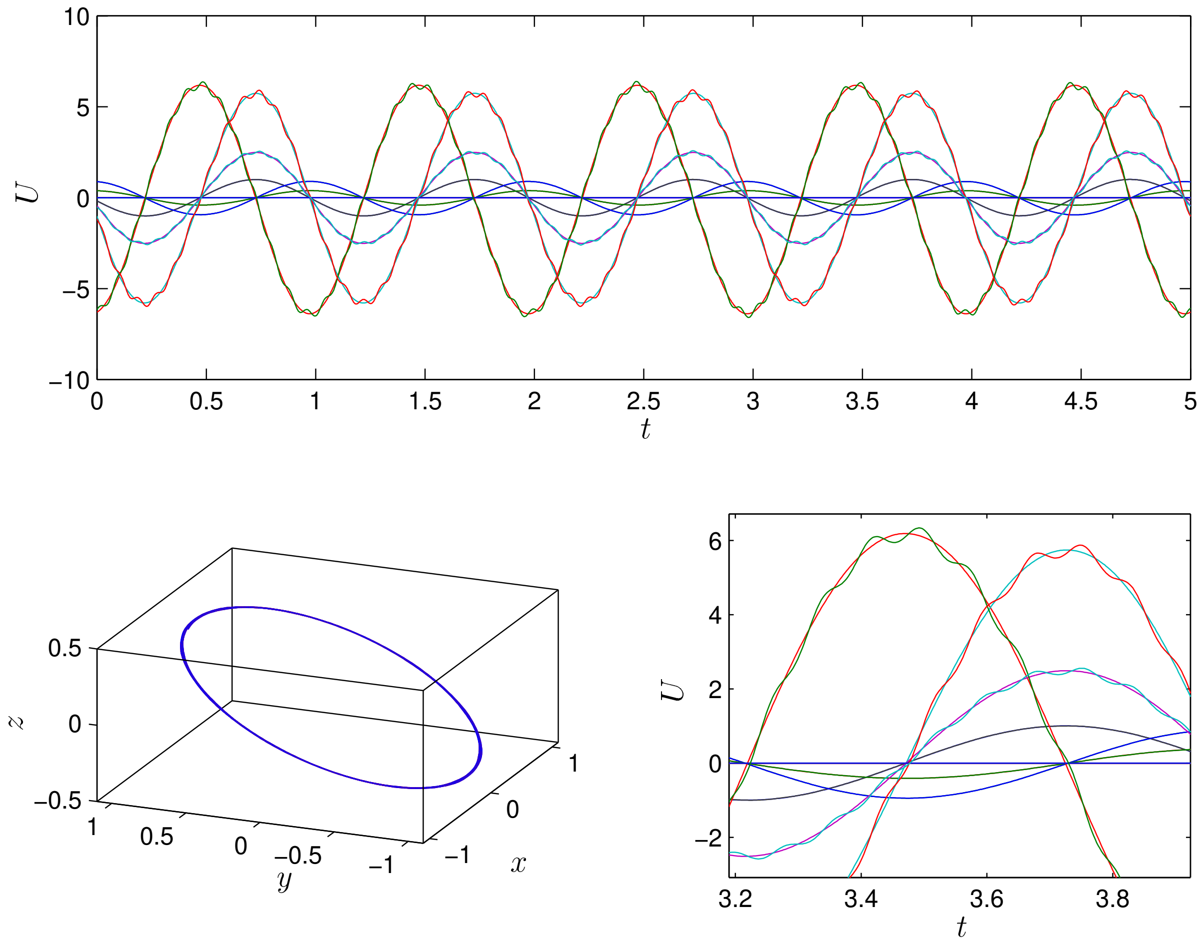



**Figure 19:** The solution for the components of the Earth-Moon system, and the trajectories of Earth and the Moon. Notice that the distance between Earth and the Moon is too small for the differences in positions to be visible in this plot, but that the differences in velocity components are clearly visible.

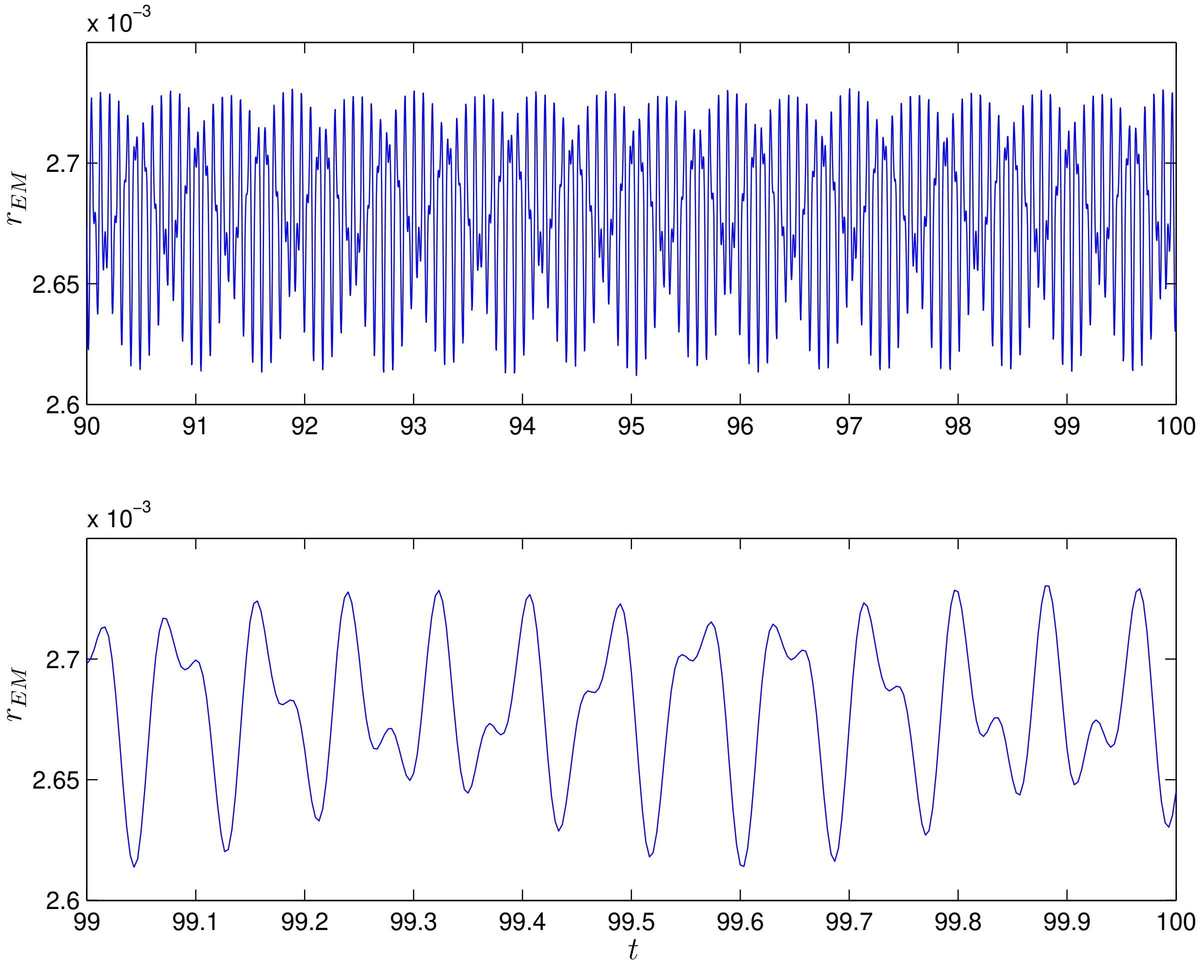



**Figure 20:** The computed distance between Earth and the Moon as function of time.

As initial conditions we take the values at 00.00 GMT on January 1:st 2000, obtained from the US Naval Observatory [1] with initial velocities obtained by fitting a high-degree polynomial to the values of December 1999. The initial data should be correct to five or more digits, which is similar to the available precision for the masses of the planets. We normalize length and time to have the space coordinates per astronomical unit, AU, which is (approximately) the mean distance between the Sun and Earth, the time coordinates per year, and the masses per solar mass. With this normalization, the gravitational constant is $4\pi^2$.

5.1. **Earth and the Moon.** Starting off a little easy, we compute the trajectories for Earth and the Moon with the Sun fixed, dropping the other eight planets. The resulting system of ODEs then consists of $N = 2 \cdot 2 \cdot 3 = 12$ components, i.e. the position and velocity components for the two bodies. In Figure 19 we plot the solution during the first five years, i.e. until January 1:st 2005, and in Figure 20 we plot the variation of the distance between the Moon and Earth as function of time.

We will now consider the predictability of solutions to the Earth-Moon system. Controlling the numerical error on a small tolerance level ($\leq 10^{-6}$), we focus on *data errors* (wrong initial data) and *modelling errors* (wrong equation). Solving the dual of the Earth-Moon system, and computing stability factors, we analyze in detail how these two errors accumulate.

Solving first the dual on $[0, 10]$ with data chosen for control of the $x^1$-coordinate of the Moon at final time, we get the solution given in Figure 21. The dual grows linearly (and oscillates) backward in time.

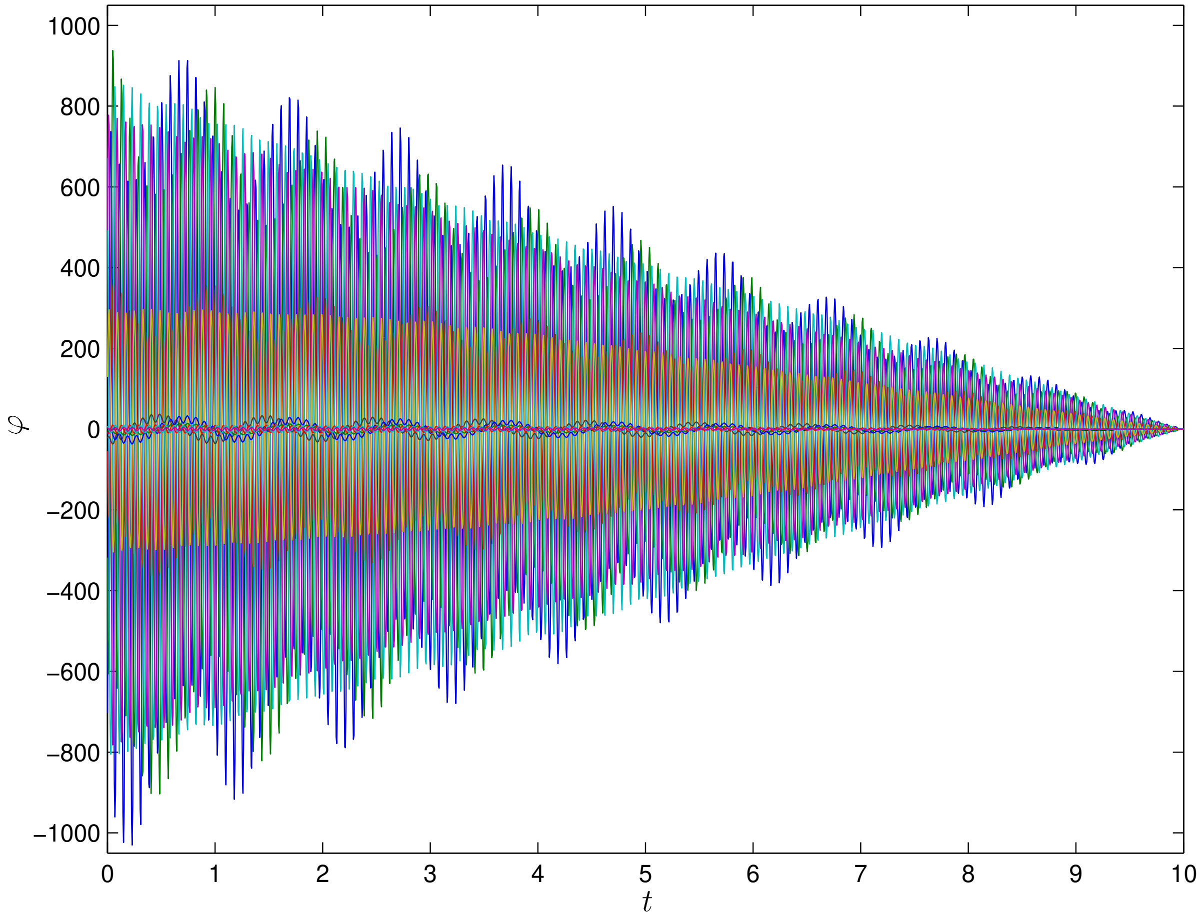



**Figure 21:** The dual of the Earth-Moon system for a specific choice of data (see text).

Neglecting all errors other than errors in initial data, the error at final time, $e(T)$, can be represented as (see [7])

$$(e(T), \varphi(T)) = (e(0), \varphi(0)) \leq \sum_{i=1}^{N} |\varphi_i(0)|\ |e_i(0)| = \sum_{i=1}^{N} S_i^{\mathcal{S}}(T)|e_i(0)|, \tag{21}$$

where $\{S_i^{\mathcal{S}}(T)\}_{i=1}^N$ are *sensitivity* stability factors that measure the influence of error in initial data. With $\varphi_i(T) = 1$ for the $x^1$-coordinate of the Moon and zero for all other components, we have $|e_{x_M^1}| = |(e(0), \varphi(0))|$. We thus expect errors in initial data to also grow linearly, since solving to $T = 20$, the dual will grow twice as large. We verify this simple observation by solving the dual to a number of final times $T' \leq T = 10$. In Figure 22 we plot the different stability factors as function of time.

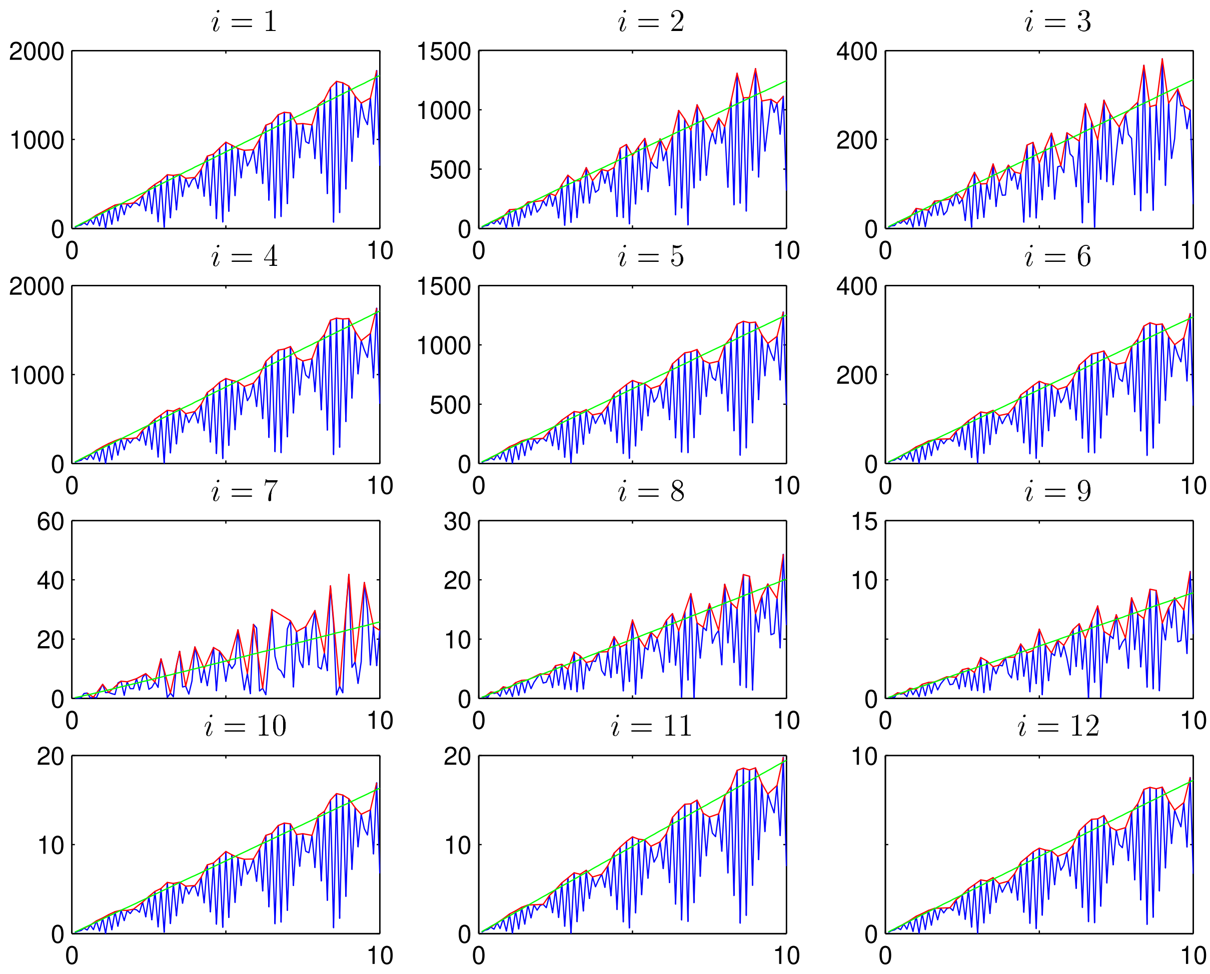



**Figure 22:** Sensitivity (data) stability factors, $\{S_i^{\mathcal{S}}\}_{i=1}^N$, for the Earth-Moon system.

The stability factors grow linearly with time, and thus also the error introduced by errors in initial data. Note also that the stability factors do not grow monotonically, since by the periodicity of the solution, some values for the final time will be more forgiving with certain errors in initial data.

Assuming now that we know initial data for Earth and the Moon with a relative error of $10^{-5}$ for every component, how far do we get? Analyzing the growth of the stability factors in more detail, we fit the parameters of $\{C_i T^{p_i}\}_{i=1}^N$ to the stability factors for the different components and expect to have $p_i \approx 1$ for $i = 1, \ldots, N$. From Table 3, where we also give the errors in initial data, we conclude that

$$|e_{x_M^1}(T)| \leq \sum_{i=1}^{N} S_i^S(T)|e_i(0)| \approx \sum_{i=1}^{N} C_i T|e_i(0)| = T \sum_{i=1}^{N} C_i |e_i(0)| \approx 0.0035\ T. \tag{22}$$

Keeping the error for the $x^1$-coordinate of the Moon less than the radius of its orbit around Earth, which is approximately $r_0 = 0.0026$AU, thus requires $0.0035T < 0.0026$, so that we cannot expect to compute accurately longer than to $T = 0.0026/0.0035 \approx 0.75$, i.e. only *nine months*!

| $i$ | $C_i$ | $p_i$ | $\lvert e_i(0)\rvert$ | $C_i\lvert e_i(0)\rvert$ | Variable |
|---|---|---|---|---|---|
| 1 | 170 | 1.00 | 1.8e-06 | 3.1e-04 | $x^1$-coordinate for Earth |
| 2 | 130 | 0.98 | 8.9e-06 | 1.2e-03 | $x^2$-coordinate for Earth |
| 3 | 35 | 0.99 | 3.8e-06 | 1.3e-04 | $x^3$-coordinate for Earth |
| 4 | 170 | 0.99 | 1.8e-06 | 3.1e-03 | $x^1$-coordinate for the Moon |
| 5 | 130 | 0.99 | 8.8e-06 | 1.1e-03 | $x^2$-coordinate for the Moon |
| 6 | 34 | 0.99 | 3.8e-06 | 1.3e-04 | $x^3$-coordinate for the Moon |
| 7 | 2.4 | 1.03 | 6.3e-05 | 1.5e-04 | $\dot{x}^1$-coordinate for Earth |
| 8 | 2.0 | 1.01 | 1.0e-05 | 2.0e-05 | $\dot{x}^2$-coordinate for Earth |
| 9 | 0.86 | 1.01 | 4.4e-06 | 3.8e-06 | $\dot{x}^3$-coordinate for Earth |
| 10 | 1.6 | 1.00 | 6.2e-05 | 1.0e-04 | $\dot{x}^1$-coordinate for the Moon |
| 11 | 2.0 | 0.99 | 1.2e-05 | 2.4e-05 | $\dot{x}^2$-coordinate for the Moon |
| 12 | 0.88 | 0.99 | 5.1e-06 | 4.5e-06 | $\dot{x}^3$-coordinate for the Moon |

**Table 3:** Estimated values for the individual stability factors, $S_i(t) = C_i T^{p_i}$, together with the individual errors in initial data.

We now proceed to see if this holds in reality, i.e. by comparing two solutions, one with "correct" initial data, and one with some errors introduced. Simply multiplying initial data for all components with 1.00001, the error grows as shown in Figure 23. In this figure we also show the growth of errors introduced by an incorrect gravitational constant $G$.

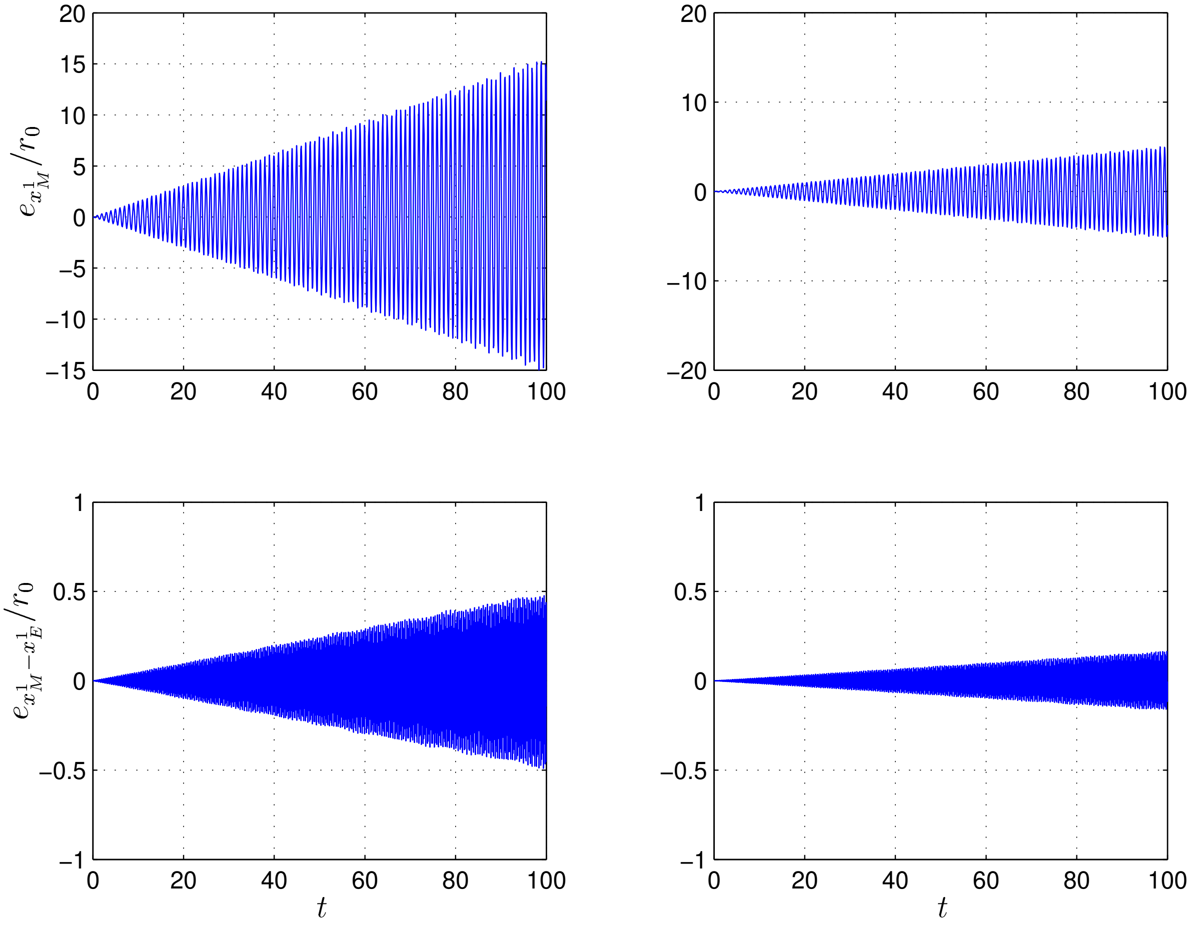



**Figure 23:** Errors in the $x^1$-coordinate of the Moon, $x_M^1$, as function of time (above), and errors in the position relative to Earth, $x_M^1 - x_E^1$ (below), for incorrect initial data (left), and incorrect gravitational constant (right).

We see that the error in position for the Moon grows rapidly, and reaches the limit of unpredictability already after a few years, as we expect. (The prediction of nine months is for worst-case data.) We also note that the error for the relative position (with respect to Earth) for the Moon seems to grow slower. Is it in fact so that even though we cannot compute correctly the absolute position for the Moon more than a few years, the Moon will still be in the correct position relative to Earth, so that we can effectively compute much longer? To investigate this, we solve the dual again, now with data chosen for control of the relative $x^1$-coordinate of the Moon, $x_M^1 - x_E^1$, i.e. we take $\varphi_i(T) = 1$ for the Moon's $x^1$-coordinate, and $\varphi_i(T) = -1$ for Earth's $x^1$-coordinate. Noticing that the dual does not change much, we plot the growth of the error for the two quantities, $x_M^1$ and $x_M^1 - x_E^1$, see Figure 24.

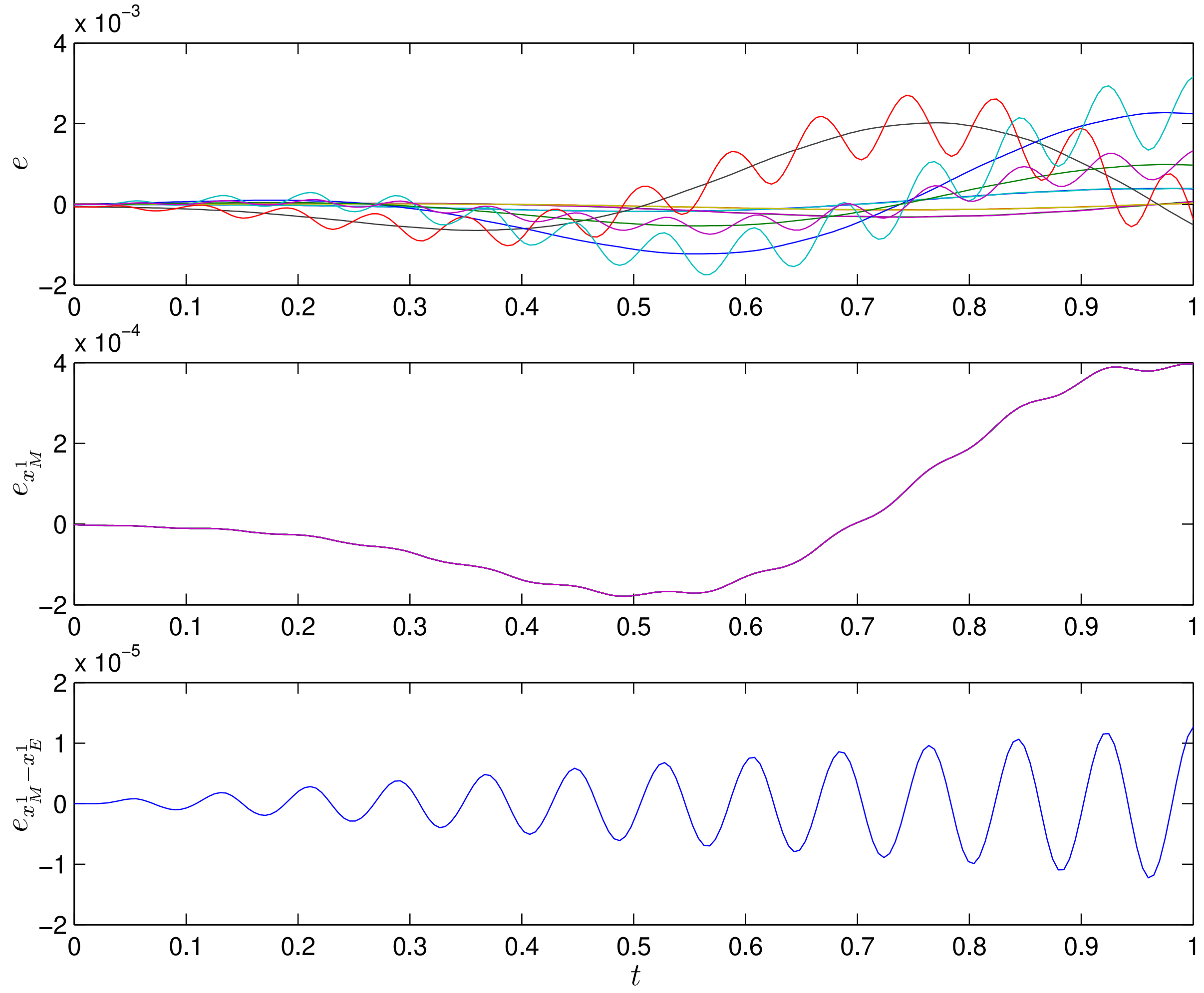



**Figure 24:** The growth of the error for the the two quantities $x_M^1$ and $x_M^1 - x_E^1$.

If the duals, and thus the stability factors, are about the same, why is one of the errors larger than the other? The answer is the choice of error in initial data. Let $\varphi^{[x_M^1]}$ be the dual for control of the error of $x_M^1$ at final time, and let $\varphi^{[x_M^1-x_E^1]}$ be the dual for control of the error of $x_M^1 - x_E^1$ at final time. As noted above, we then have $e_{x_M^1}(T) = (\varphi^{[x_M^1]}(0), e(0))$, and $e_{x_M^1-x_E^1}(T) = (\varphi^{[x_M^1-x_E^1]}(0), e(0))$. Choosing the initial error now as

$$e(0) = C\left(\varphi^{[x_M^1-x_E^1]}(0) - P_{\varphi^{[x_M^1]}(0)}\varphi^{[x_M^1-x_E^1]}(0)\right), \tag{23}$$

for some constant $C$ (chosen so that the largest relative error for the components is the same as before, $10^{-5}$), i.e. subtracting the projection onto $\varphi^{[x_M^1]}(0)$, the situation is the opposite, i.e. large error for $x_M^1 - x_E^1$ and zero error for $x_M^1$, see Figure 25.

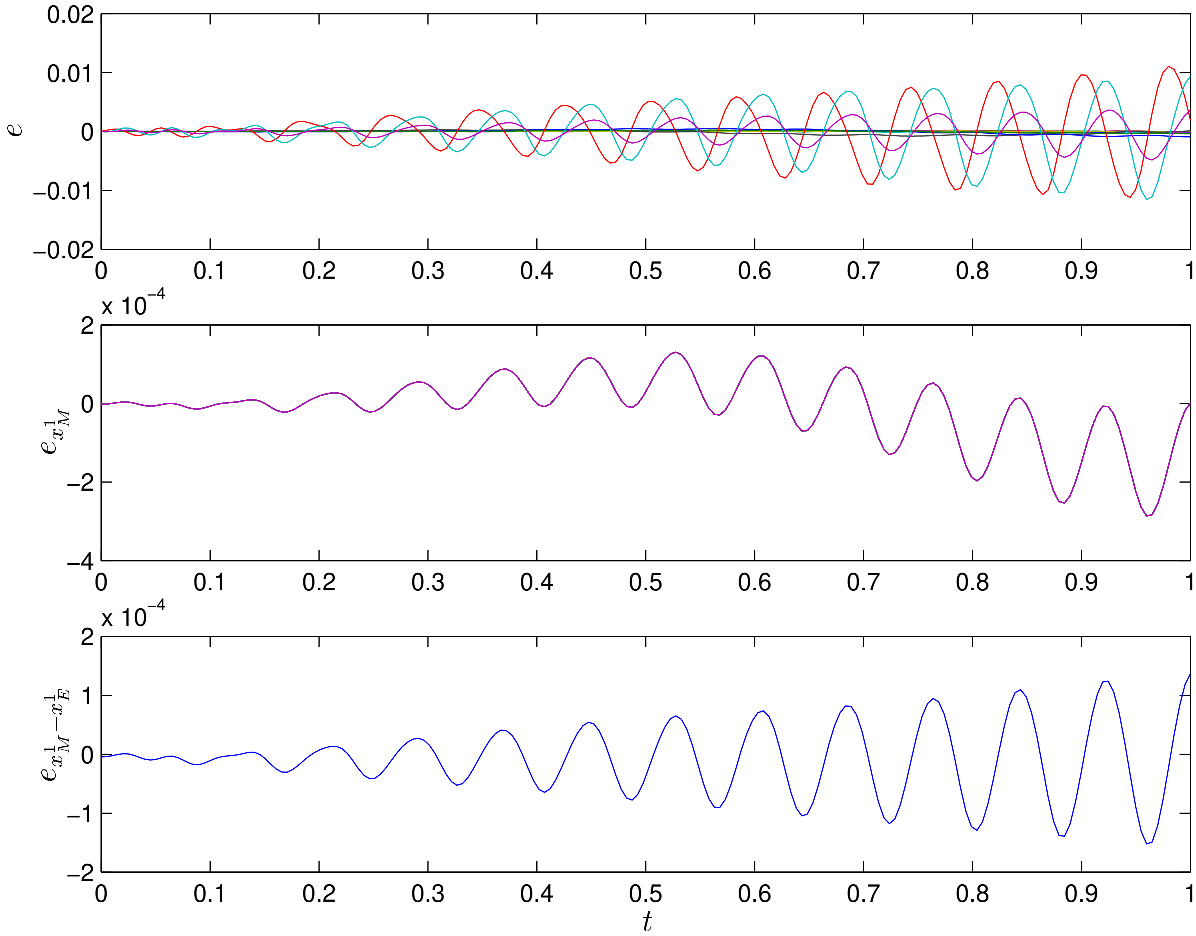



**Figure 25:** The growth of the error for the two quantities $x^1_M$ and $x^1_M - x^1_E$, for the special choice of initial error in (23).

5.2. **The Nine Planets.** We now extend the system to include also the remaining eight planets, in the following order with respect to their mean distance to the Sun: *Mercury*, *Venus*, *(Earth already included)*, *Mars*, *Jupiter*, *Saturn*, *Uranus*, *Neptune* and *Pluto*, and we compute stability factors for error control at final time $T$ of the $x^1$-coordinate for the innermost planet Mercury. As before, the stability factors for error in initial data grow linearly with time, and we make the prediction that we cannot compute accurately further than about 500 years, assuming initial data is correct to five digits. Since the stability factors grow only linearly, we are in a much better situation than for the Lorenz system, for which we have exponentially growing stability factors. This means that if we know initial data with twice as many digits, i.e. with a relative error less than $10^{-10}$, we will reach $10^5$ times further, whereas for the Lorenz system, twice as many digits means we will reach only twice as far.

Examining the growth of the stability factors during the first ten years, see Figure 26, it is clear that the structure is quite intricate. The choice of final time and the distribution of the initial error onto the different components are of great importance for the actual size of the error. It is also evident which components influence Mercury the most, namely the components of Mercury itself, the Sun, and, to some extent, Venus and Jupiter.

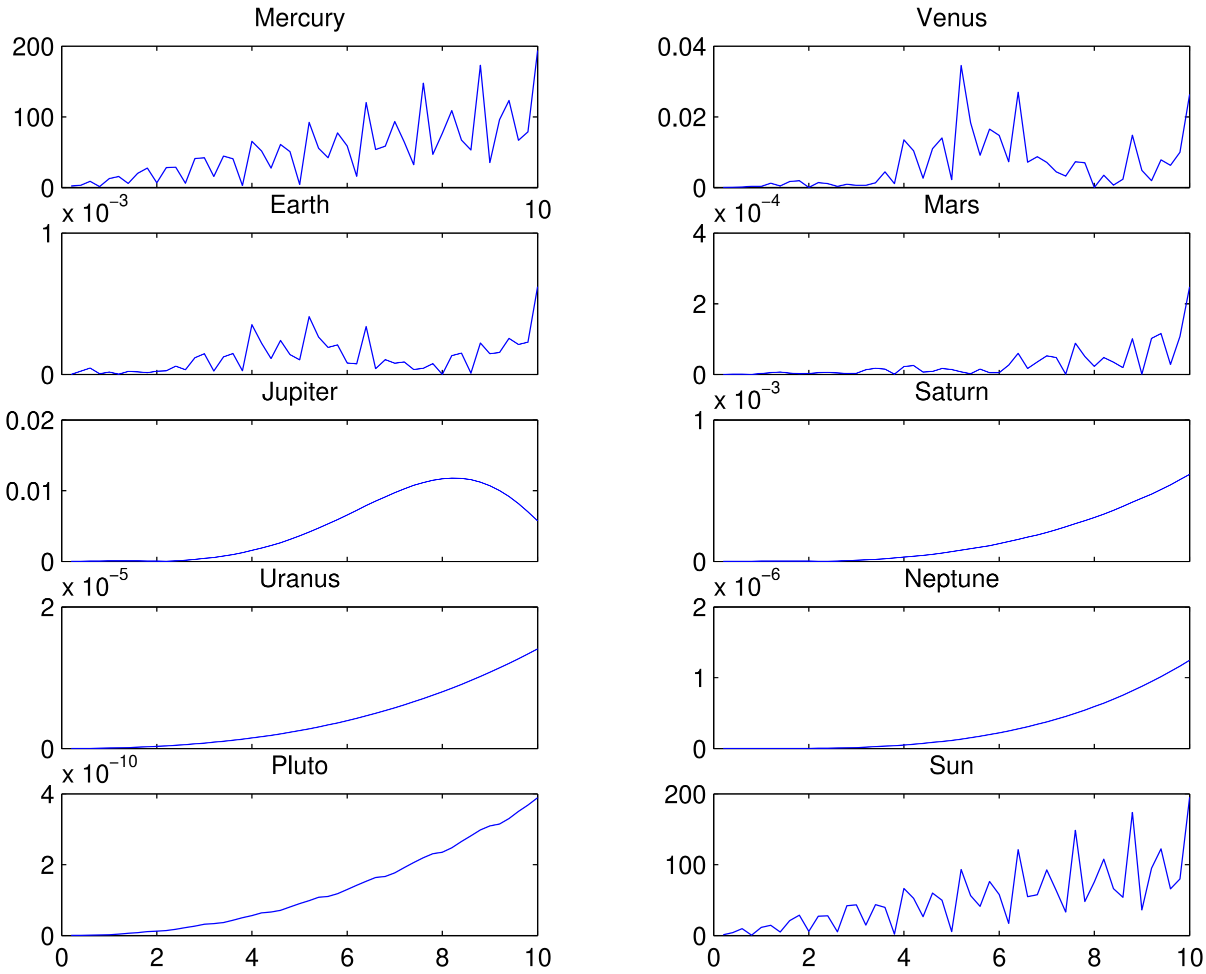



**Figure 26:** Stability factors for the $x^1$-coordinate components of the nine planets and the Sun, w.r.t. the error in the $x^1$-coordinate for Mercury at final time.

Examining now as before the actual size of the error for an actual error in initial data, we multiply initial data for the positions of Mercury and the Sun with 1.00001 (since the contribution to the error introduced by errors in the other components is negligible), and find that after 500 years the error is indeed very large, but perhaps only half as large as we expect it to be. This is again because the estimated value of 500 years is for the worst case error in initial data. In Figure 27 we illustrate the differences in the position of Mercury at final time, $T = 500$, for the two choices of initial data.

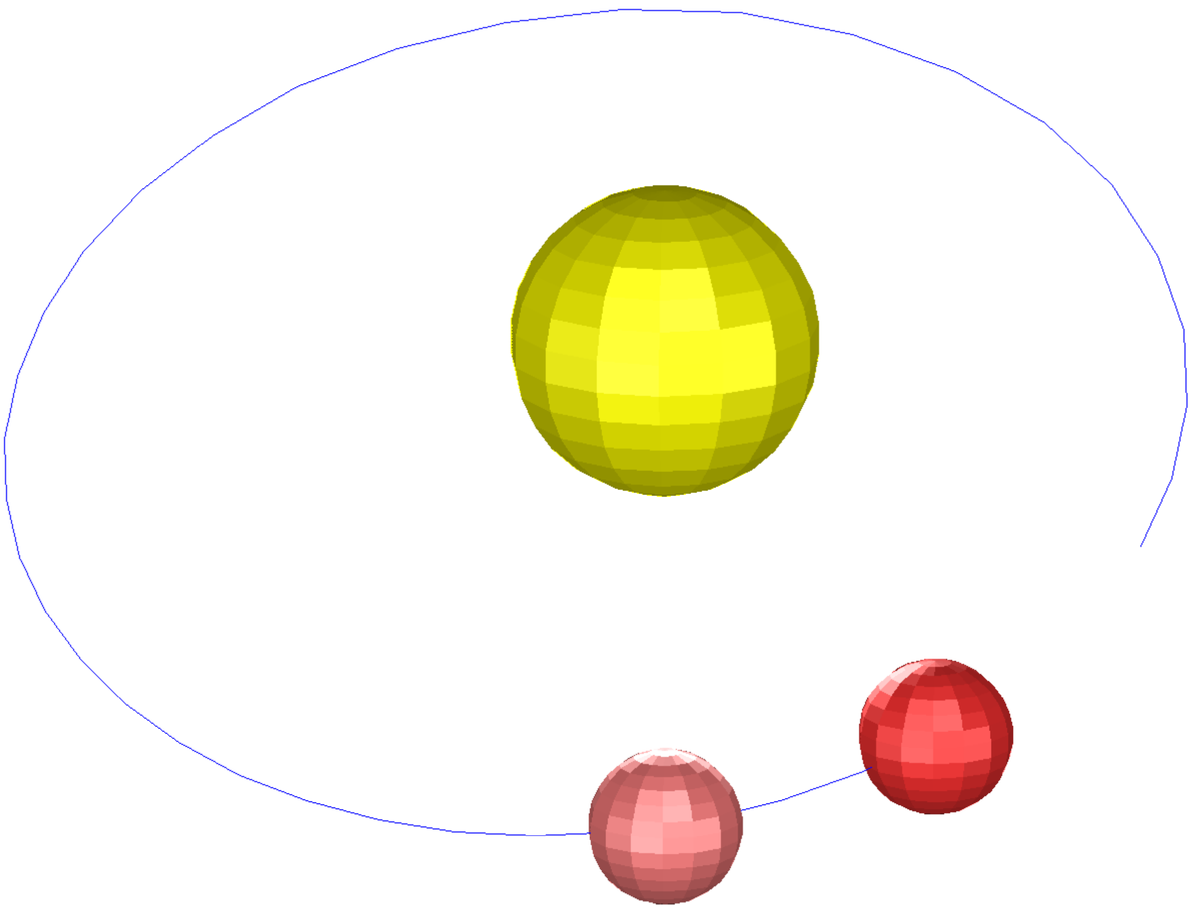


**Figure 27:** Positions of Mercury for two different choices of initial data, with relative difference less than $10^{-5}$, at time $T = 500$, i.e. on January 1:st 2500, 00:00 GMT. (The version of Mercury going a little faster is the one computed with "correct" initial data.)

Finally, we make a comparison of the computational cost between the mcG(1) method and the standard cG(1) method, measured as cpu time, the total number of local time-steps, i.e. $M = \sum_{i=1}^{N} M_i$, and the total number of local function evaluations (including also the evaluation of different residuals).

Starting with the Sun and Mercury, we add the remaining planets one after the other, and expect the cost of the multi-adaptive method to remain close to constant, whereas for the standard cG(1) method, using the same time-steps for all components, we expect the cost to grow linearly with the number of planets.

For demonstration purposes, we take for the multi-adaptive method 75 time-steps per period for each of the planets (with a maximum time-step of 0.1), and for the cG(1) method we take all time-steps equal to the smallest time-step, i.e. 75 time-steps per Mercury year. As a measure of the error, we take the maximum relative error in position for the planets at final time.

We present the results below in Figure 28. The error is about the same for the two methods. (We can expect the error to be somewhat larger for the multi-adaptive method.) The number of local time-steps and local function evaluations clearly behaves as we expect it to do, remaining practically constant for the multi-adaptive method and increasing linearly for the standard method. Although the work in terms of the number of steps and function evaluations remains constant for the multi-adaptive method, the cpu time increases as we increase the number of planets, since the cost for the administration of the different components with their different time-steps increases.

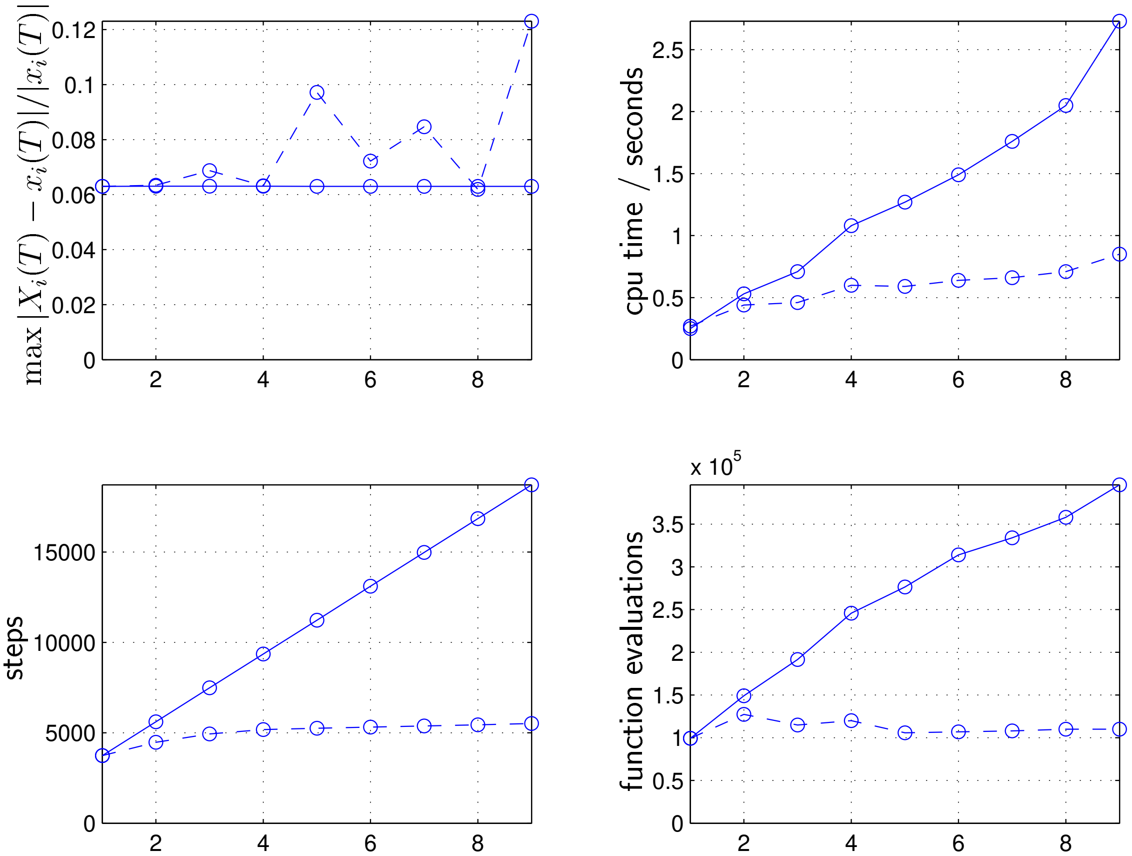



**Figure 28:** A comparison of the mcG(1) method (dashed) with the cG(1) method, starting with the Sun and Mercury, and then adding the rest of the planets, one after the other.

We have above discussed predictability due to data errors. The *computability* of the Solar System will be studied in future work, extending the strategy presented for the Lorenz system: given accurate data, the challenge is to compute on a time interval as long as possible.

## 6. Earth, the Moon and the Comet

Next, we consider the many-body problem in a special setting. Earth and the Moon orbit around the Sun, which we may think of as fixed at the origin. In comes a comet aimed to pass very near the Earth-Moon system. To decrease the sensitivity for the choice of initial data for the comet, and thus to make the aiming a little easier, we take a very heavy comet, more like the size of Jupiter than a regular-sized comet. The events are then as follows: the comet comes in from out of nowhere, going straight towards Earth and the Moon taking their usual route around the Sun. Once the comet gets close enough, it will shoot the Moon out of its orbit around Earth, placing it in a new orbit of its own around the Sun. The equations are, as for the Solar System,

$$m_i \ddot{x}_i = \sum_{j \neq i} \frac{G m_i m_j}{|x_j - x_i|^3}(x_j - x_i), \tag{24}$$

with initial conditions taken as the positions on January 1:st 2000, 00.00 GMT. The hopefully unrealistic data for the comet is taken so as to fit this example.

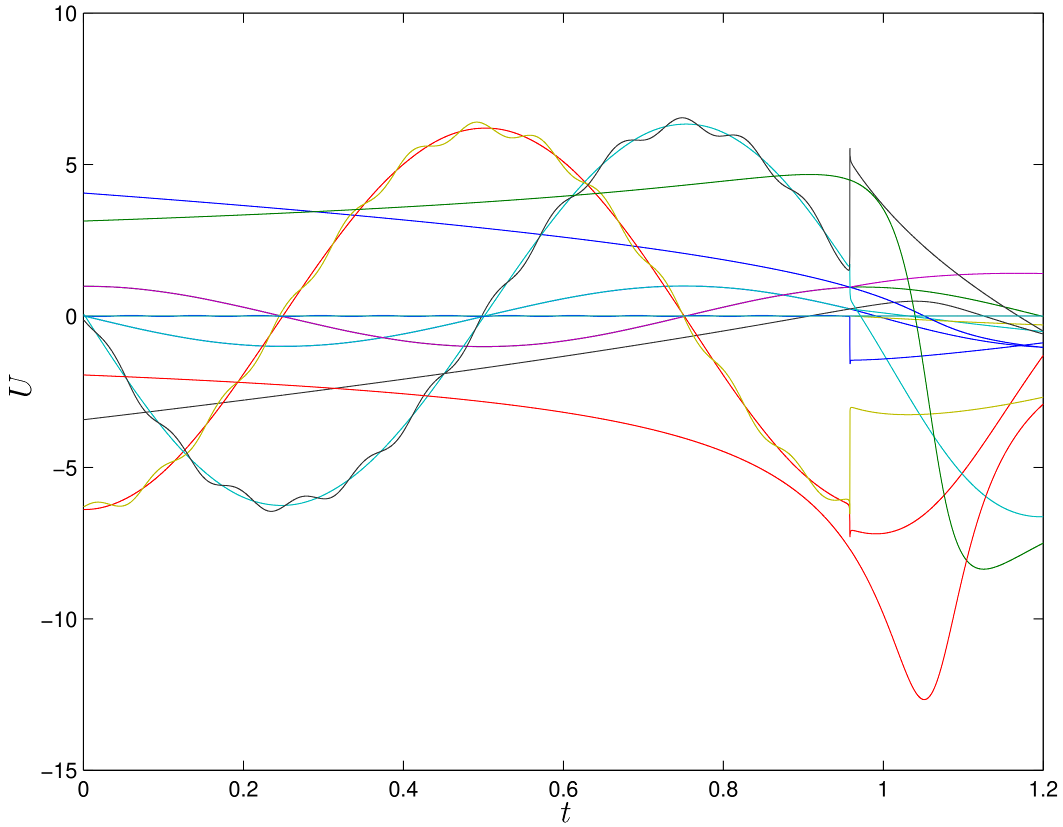



**Figure 29:** Solutions for the positions of the Earth-Moon-Comet system as function of time.

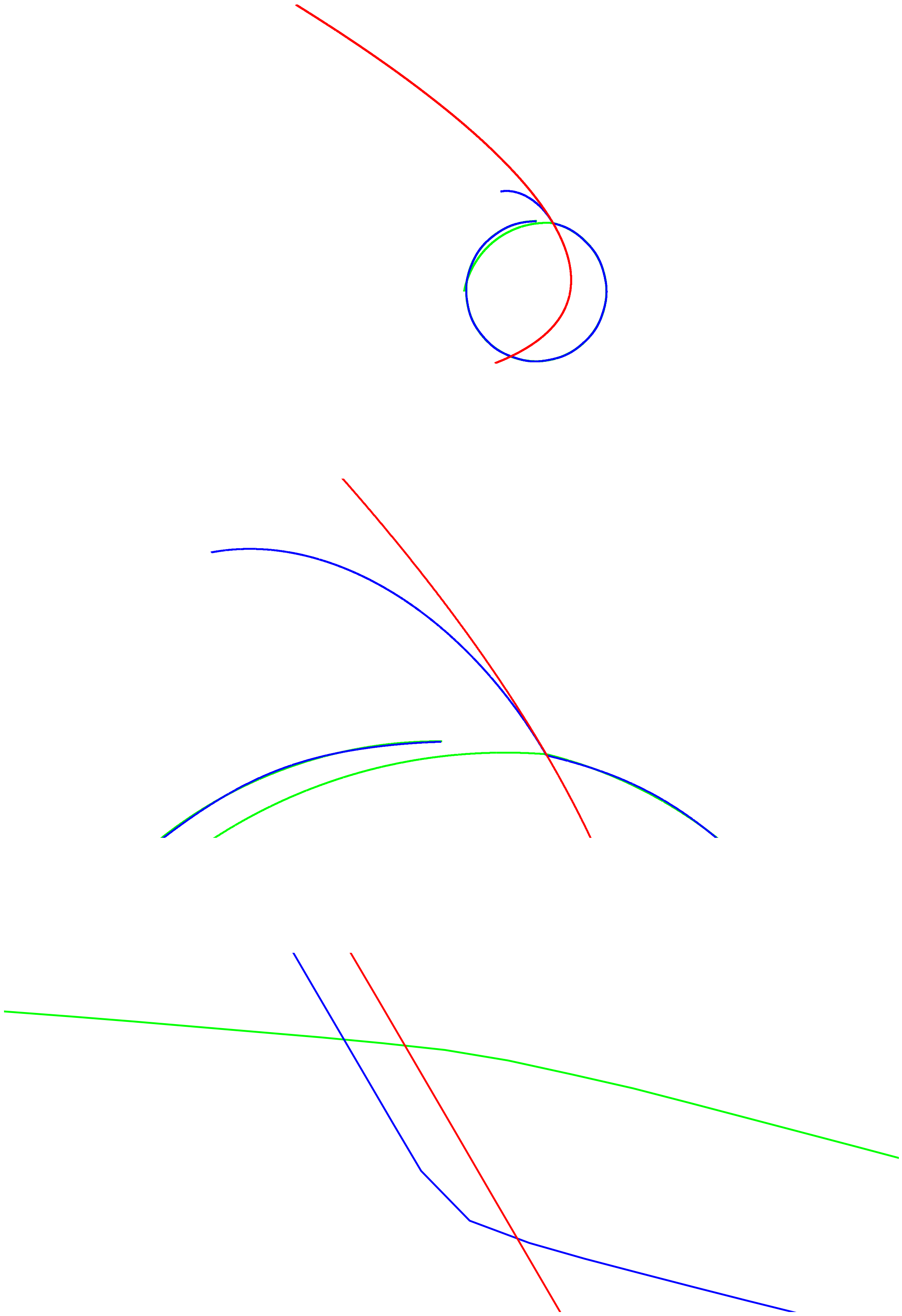


**Figure 30:** Trajectories of Earth (going from the right to the left in the lower plot), the Moon (coming from the right being thrown upwards), and the comet (which is the the straight line coming in from above).

Solving, using the explicit version of the mcG(3) method, to final time $T = 1.2$ (years), we obtain the solution given in Figure 29. At the point where the comet gets very close, there seems to be a sudden jump in the solution. As is evident from Figure 30, this is not a jump, but only a very fast change of velocities for Earth and the Moon. This rapid change is also reflected in the size of the individual time-steps, which decrease rapidly at $t \approx 0.95$, see Figure 31. At this point, the residuals get tremendously large, and in order to keep the error small, the time-steps have to be very small, but only for a short while, and only for some of the components.

6.1. **The dual and the stability factors.** In order to get a better understanding for the problem and for the events occurring near $t = 0.95$, we solve the dual and compute stability factors. In Figure 32 we plot the dual for a particular choice of data. The dual is solved backward in time, starting with $\|\varphi(T)\| = 1$, and evidently stays small until it suddenly starts to increase, and after that increases linearly in size.

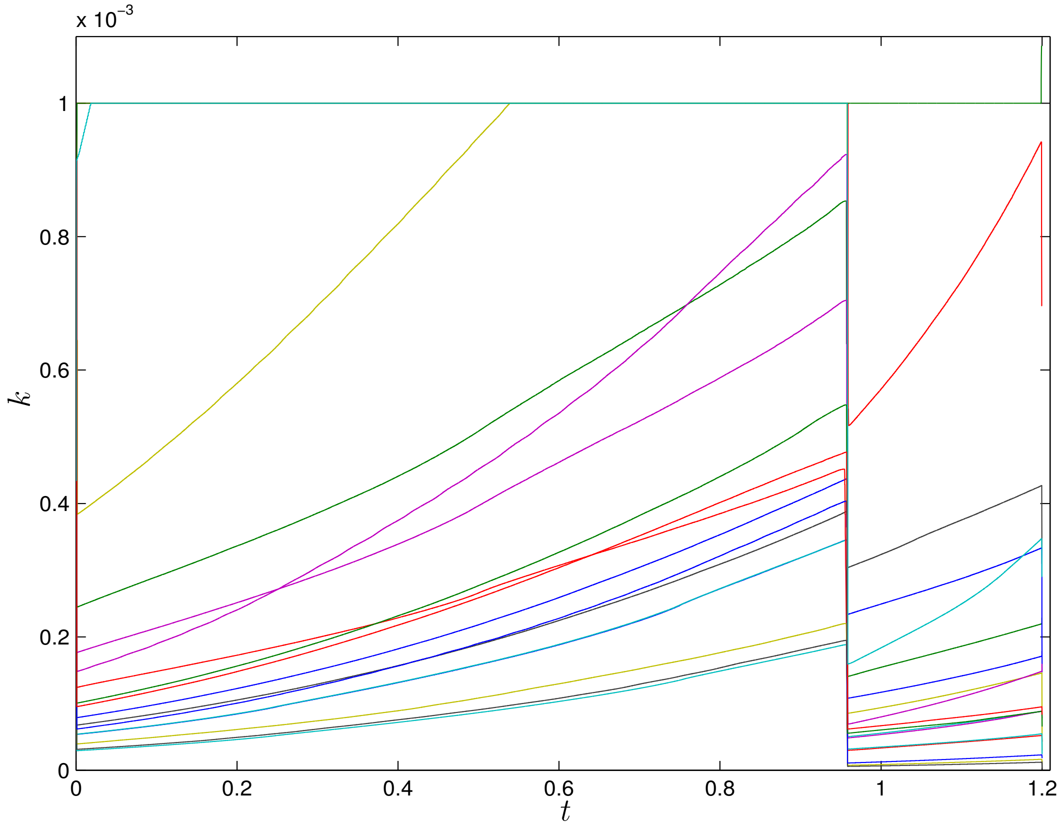



**Figure 31:** Time-steps for the position components of the Earth-Moon-Comet system.

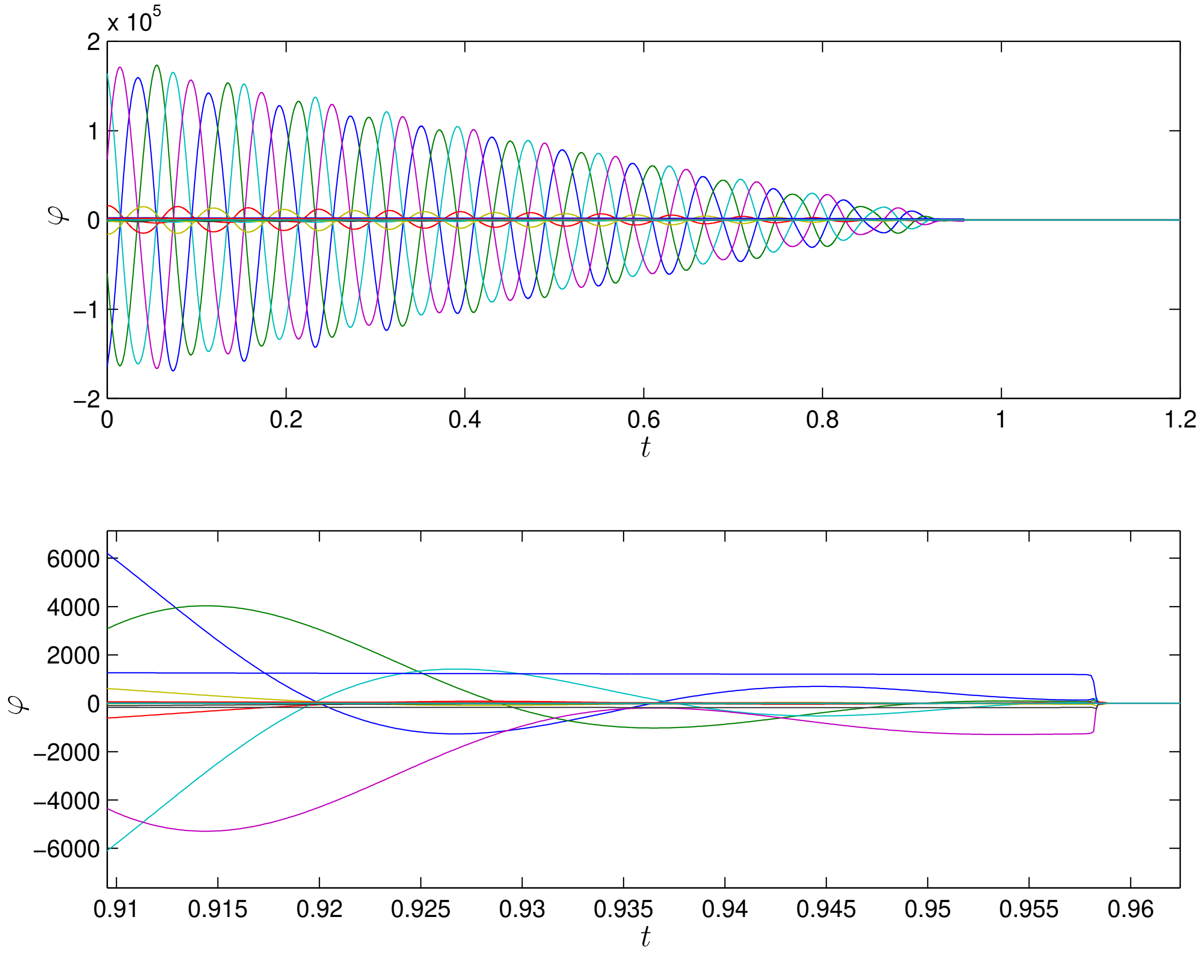



**Figure 32:** The dual for the Earth-Moon-Comet system. At the point where all the action is the dual increases rapidly. (Remember that the dual is solved backward.)

As noted earlier, we expect the dual for a system of planets to grow linearly backward in time, provided nothing extraordinarily happens. This is also the case; before and after the comet comes in very close the Earth-Moon system, the dual grows linearly, increasing very fast only near the critical point.

From this we may also realize how the stability factors depend on time. If the equations are solved to a point beyond the critical point, the stability factors will be large, whereas if we solve the equations up to a point before the critical point, the dual will never start to grow dramatically, and so the stability factors will be small. The conclusion is thus that the stability factors grow slowly (the sensitivity stability factors grow linearly, and other stability factors grow quadratically) up to the critical point, make a sudden jump, and then grow slowly as before. This is also the case, according to Figure 33.

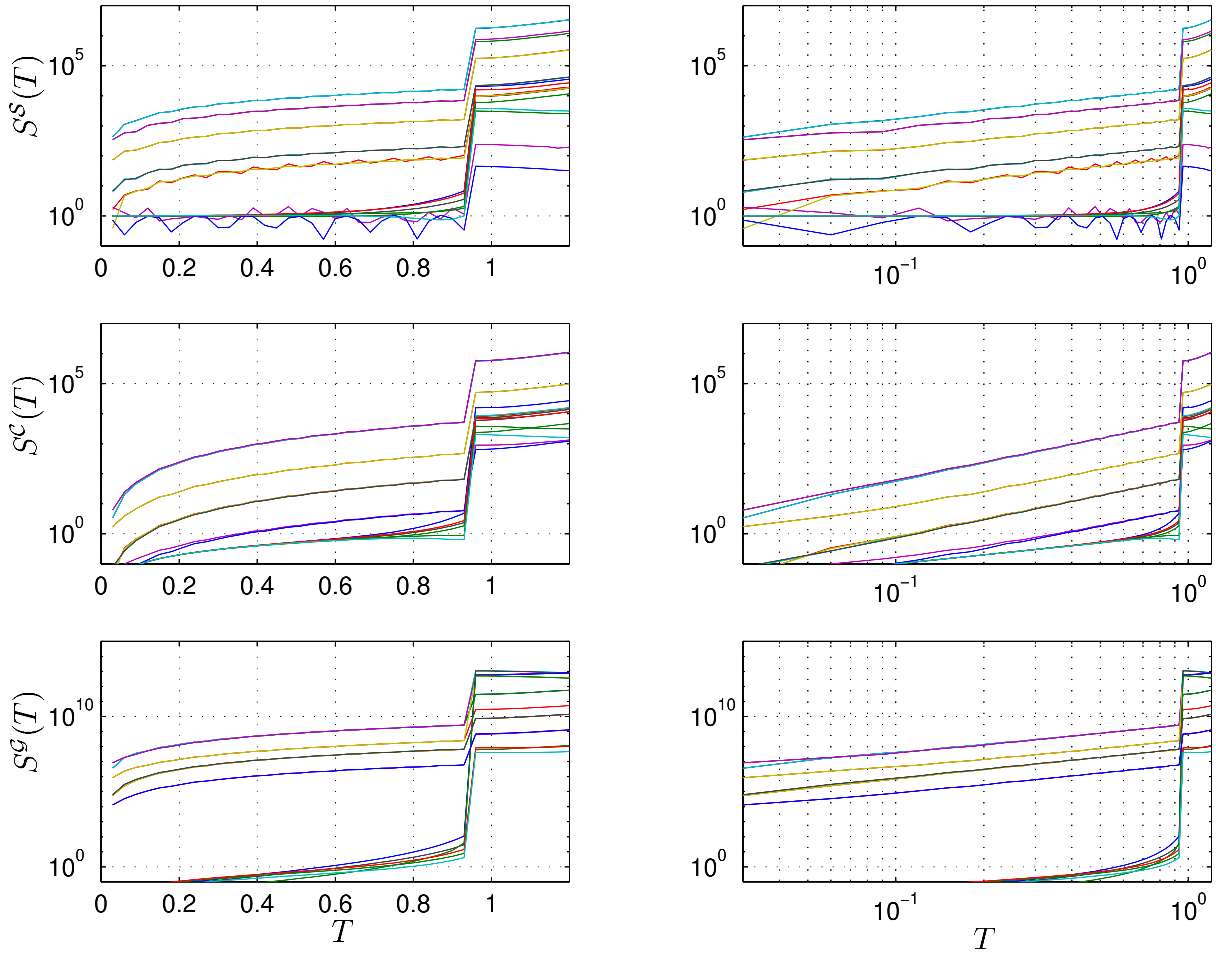



**Figure 33:** Sensitivity, $S^{\mathcal{S}}$, computational, $S^{\mathcal{C}}$, and Galerkin, $S^{\mathcal{G}}$, stability factors as function of time for the different components of the Earth-Moon-Comet system.

The conclusion is then, that if we want to compute beyond the critical point, we have to be more careful, and use much smaller time-step, than if we want to compute to a point before the critical point. This is even clearer if we look at the local stability weights, the integrals of which are the stability factors. In Figure 34, we plot the local stability weights for the different components of the Earth-Moon-Comet system, when solved to a point beyond the critical point. Judging by these local weights, the time-steps must be small all the way up until the critical point, even at the beginning of the interval, long before the critical point. Choosing the time-steps based only on the residual — or the local error — we will miss the importance of having to take smaller time-steps before the critical point than afterwards. The residual is large only at the critical point, and so says nothing about the stability properties of the problem.

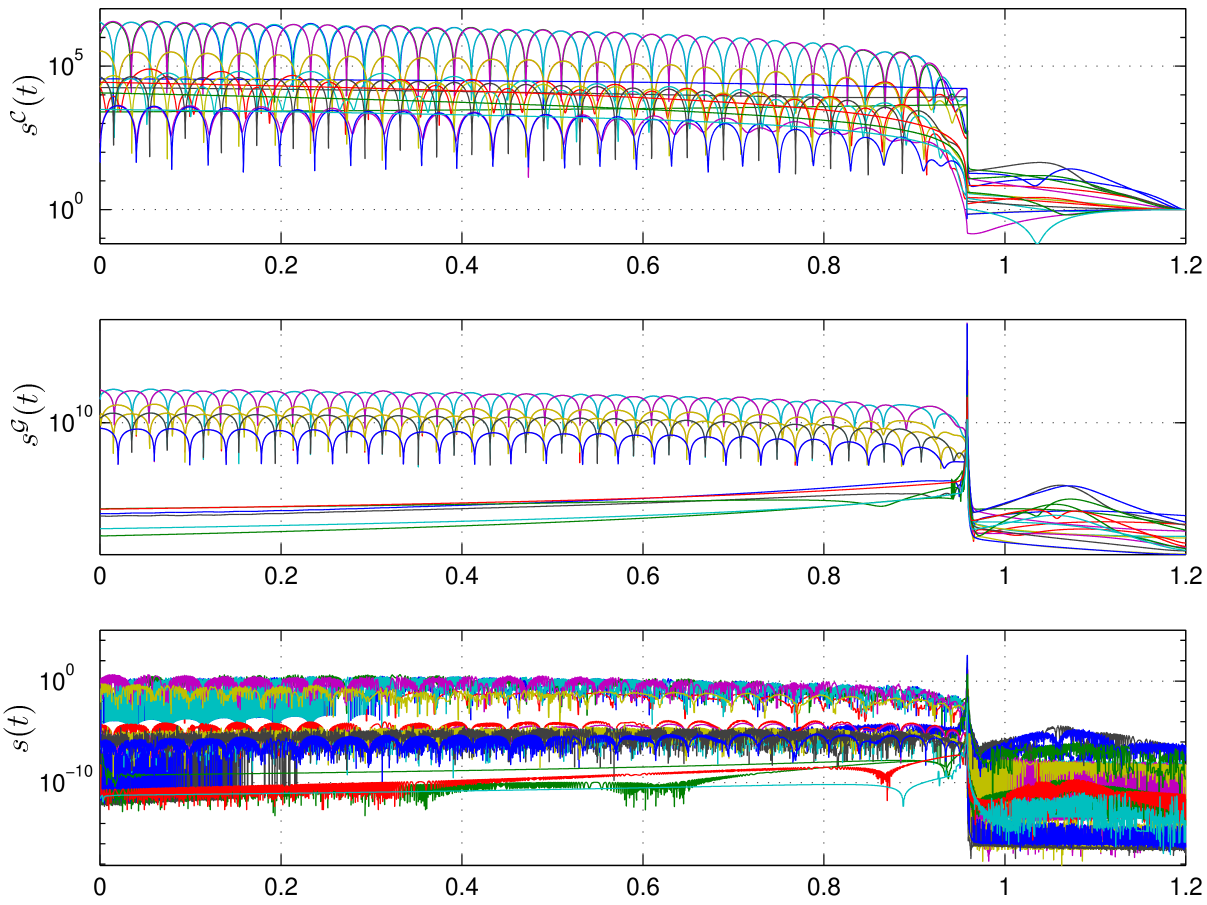



**Figure 34:** Local stability weights for the Earth-Moon-Comet system; $s^{\mathcal{C}}$ is the local computational weight, $s^{\mathcal{G}}$ is the local weight for the Galerkin error, and $s$ is the local weight for the Galerkin error based on evaluating $\varphi - \pi_k \varphi$.

## 7. A Stiff Chemical Reaction Problem

We next consider the "HIRES" problem from [4], which is the following stiff system of ODEs:

$$
\left\{
\begin{array}{rcl}
\dot{u}_1 & = & -1.71\ u_1 + 0.43\ u_2 + 8.32\ u_3 + 0.0007, \\
\dot{u}_2 & = & 1.71\ u_1 - 8.75\ u_2, \\
\dot{u}_3 & = & -10.03\ u_3 + 0.43\ u_4 + 0.035\ u_5, \\
\dot{u}_4 & = & 8.32\ u_2 + 1.71\ u_3 - 1.12\ u_4, \\
\dot{u}_5 & = & -1.745\ u_5 + 0.43\ u_6 + 0.43\ u_7, \\
\dot{u}_6 & = & -280\ u_6 u_8 + 0.69\ u_4 + 1.71\ u_5 - 0.43\ u_6 + 0.69\ u_7, \\
\dot{u}_7 & = & 280\ u_6 u_8 - 1.81\ u_7, \\
\dot{u}_8 & = & -280\ u_6 u_8 + 1.81\ u_7,
\end{array}
\right.
\tag{25}
$$

with $u_0 = (1, 0, 0, 0, 0, 0, 0, 0.0057)$ and $T = 321.8122$.

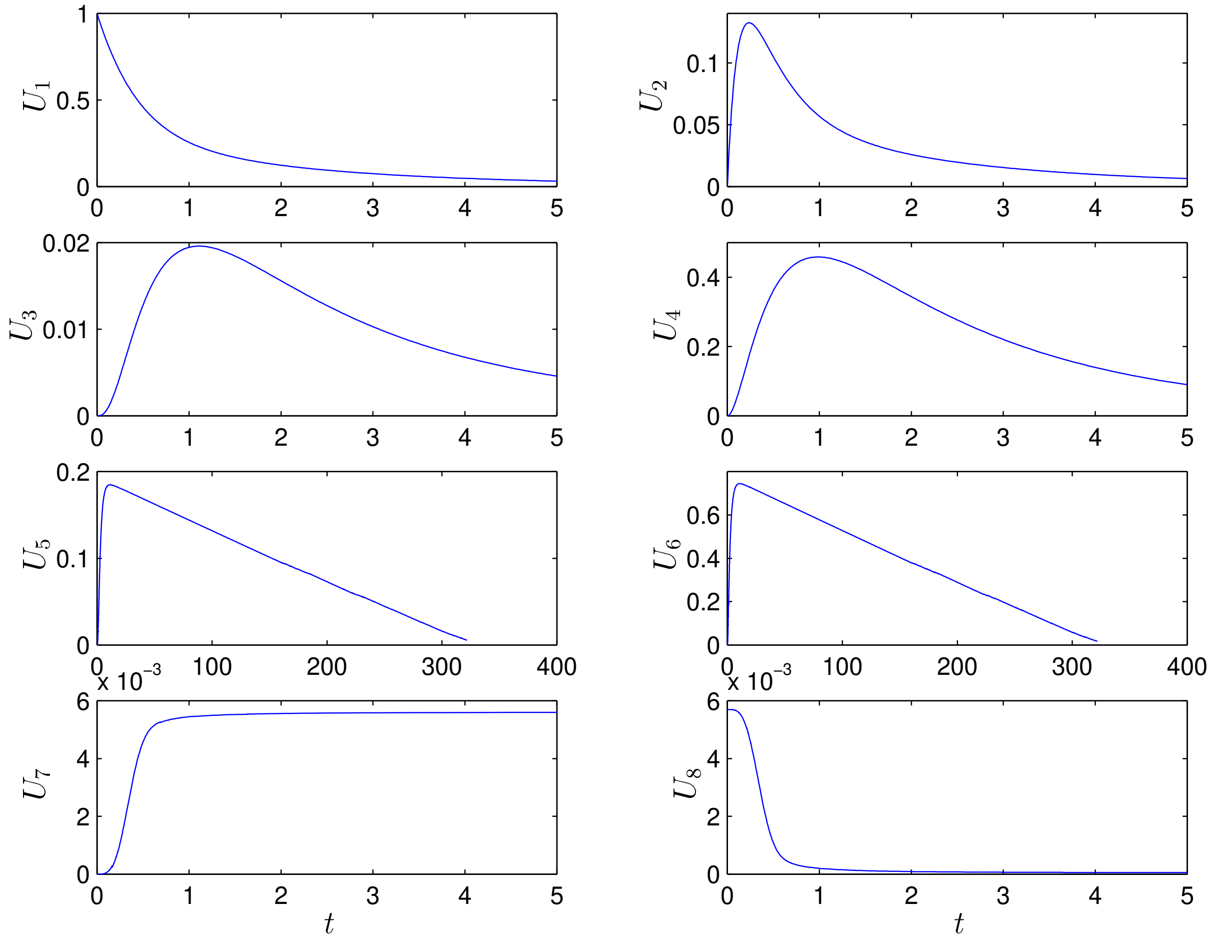



**Figure 35:** Solutions for the eight components of the ODE-system with axes chosen as in [4].

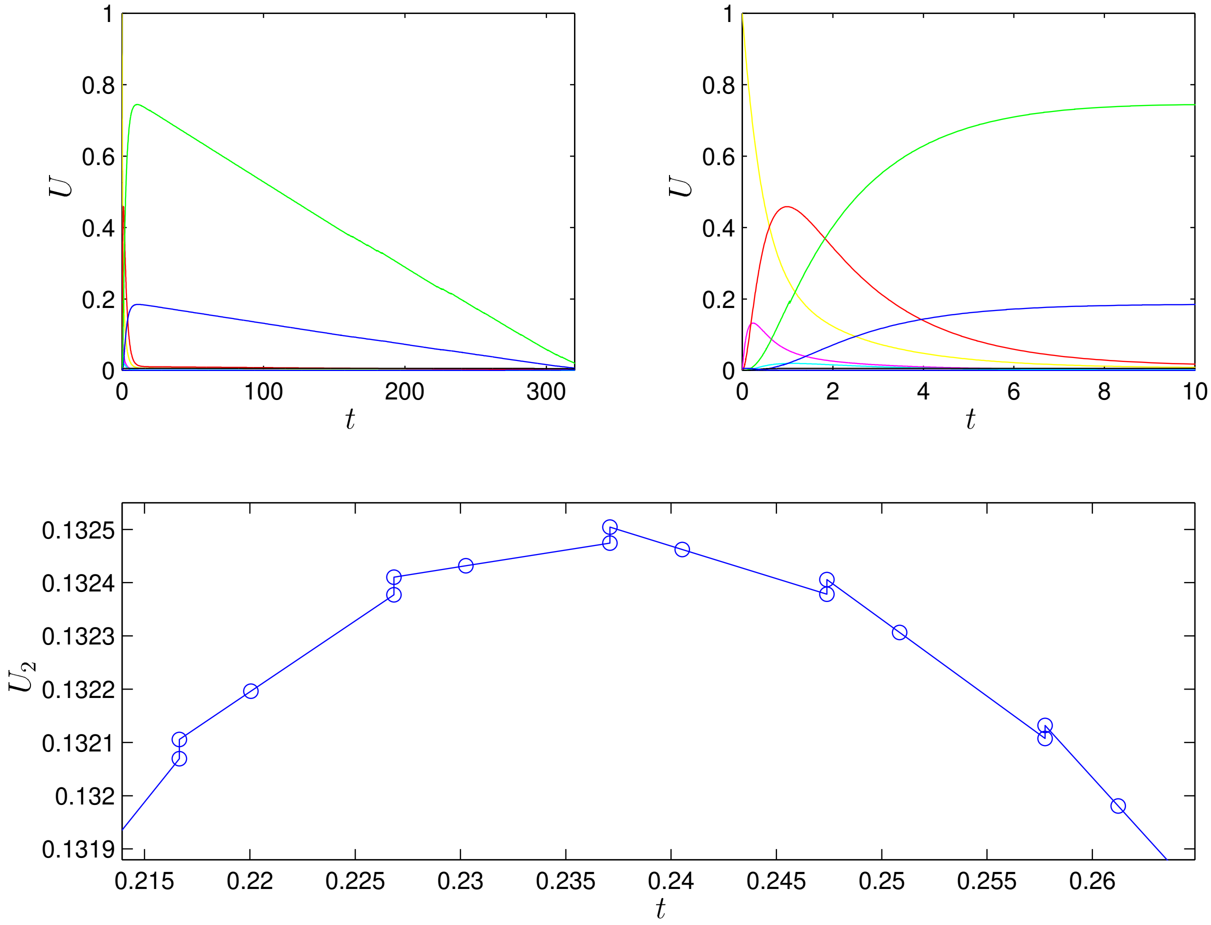



**Figure 36:** The mdG(1) solution for all components of the ODE-system (above), and the solution for the second component at its peak (below). The second and third markings on every interval are the nodal points; the first marking is there to indicate the discontinuity.

Solving using the mdG(1) method, we obtain the solution plotted in Figures 35 and 36, with time-steps as in Figure 37 and residual as in Figure 38. The time-steps are small for all components during the initial transient, after which all time-steps quickly increase to reach a pre-defined maximum time-step of $k = 1$, which is larger than the maximum stability/stiffness-imposed allowed time-step. Since the method is explicit, or, more correctly, the implicit discrete equations are solved by explicit fixed point iteration, we should not be able to use a time-step as large as $k = 1$. The individual Newton scaling makes it possible, however, to do this, as described in [7].

7.1. **The stiffness problem.** For this problem, the simple Newton scaling is not enough to make the iterations converge. To handle this, we must decrease the time-step, but only when necessary. In this way the time-step will oscillate, being sometimes (hopefully most of the time) large and sometimes small, as is described in [7] and is explored further in [3], the idea being that allowing for non-zero discrete residuals, the solution will often be accepted already after the first iteration, so that we manage well using the large time-steps. This will make the continuous and discrete residuals grow gradually, as is evident from Figure 38, so that at some point we will have to decrease the time-step (which then happens automatically since the residual is large) to make the iterations converge, and we obtain small residuals once again. Note that the exact behaviour of this process depends very much of the actual problem, the size of the time-steps, and the tolerances for the total error and the computational error. Ideally, the process of changing between large and small time-steps comes automatically with the usual time-step regulation based on the residual.

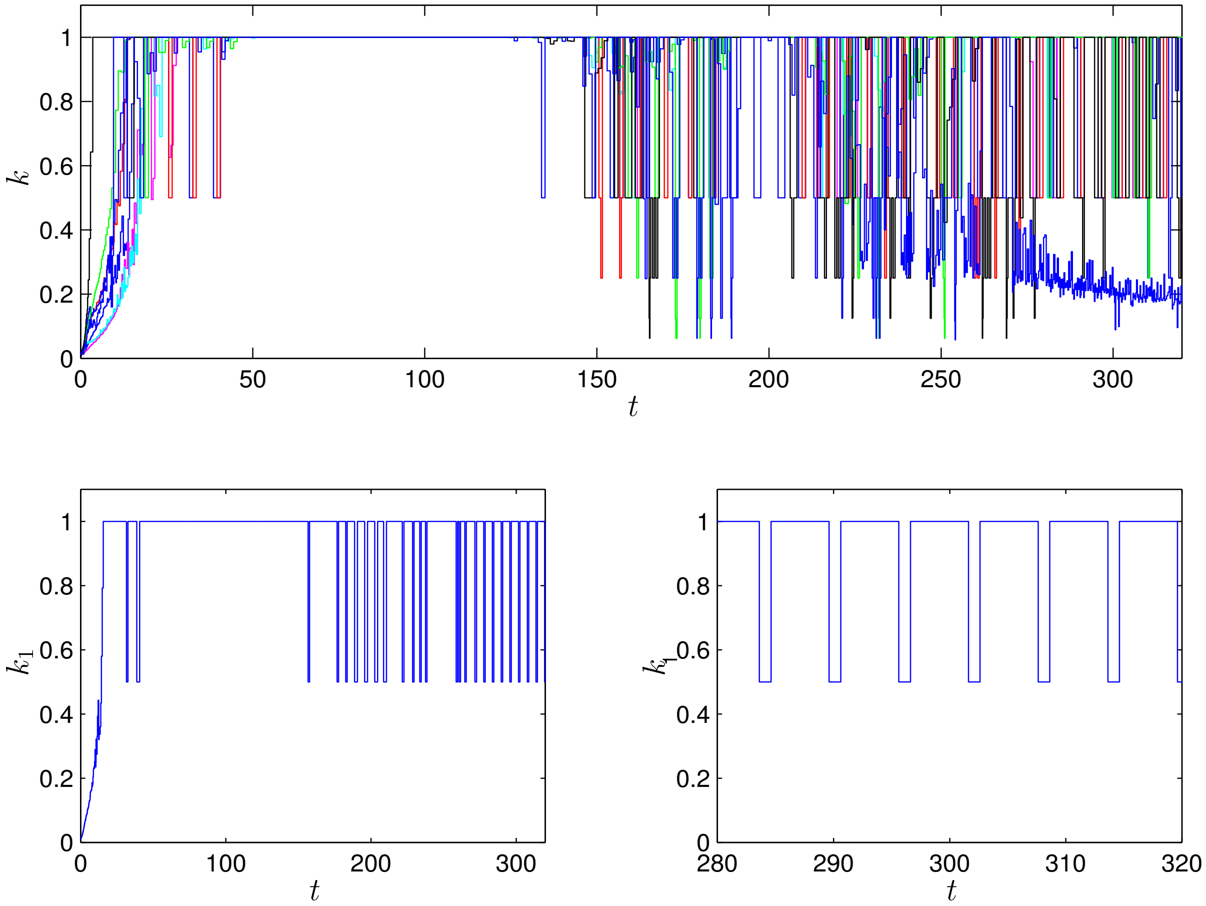



**Figure 37:** The individual time-steps for the eight components of the ODE-system (above), and the time-steps for the first component (below).

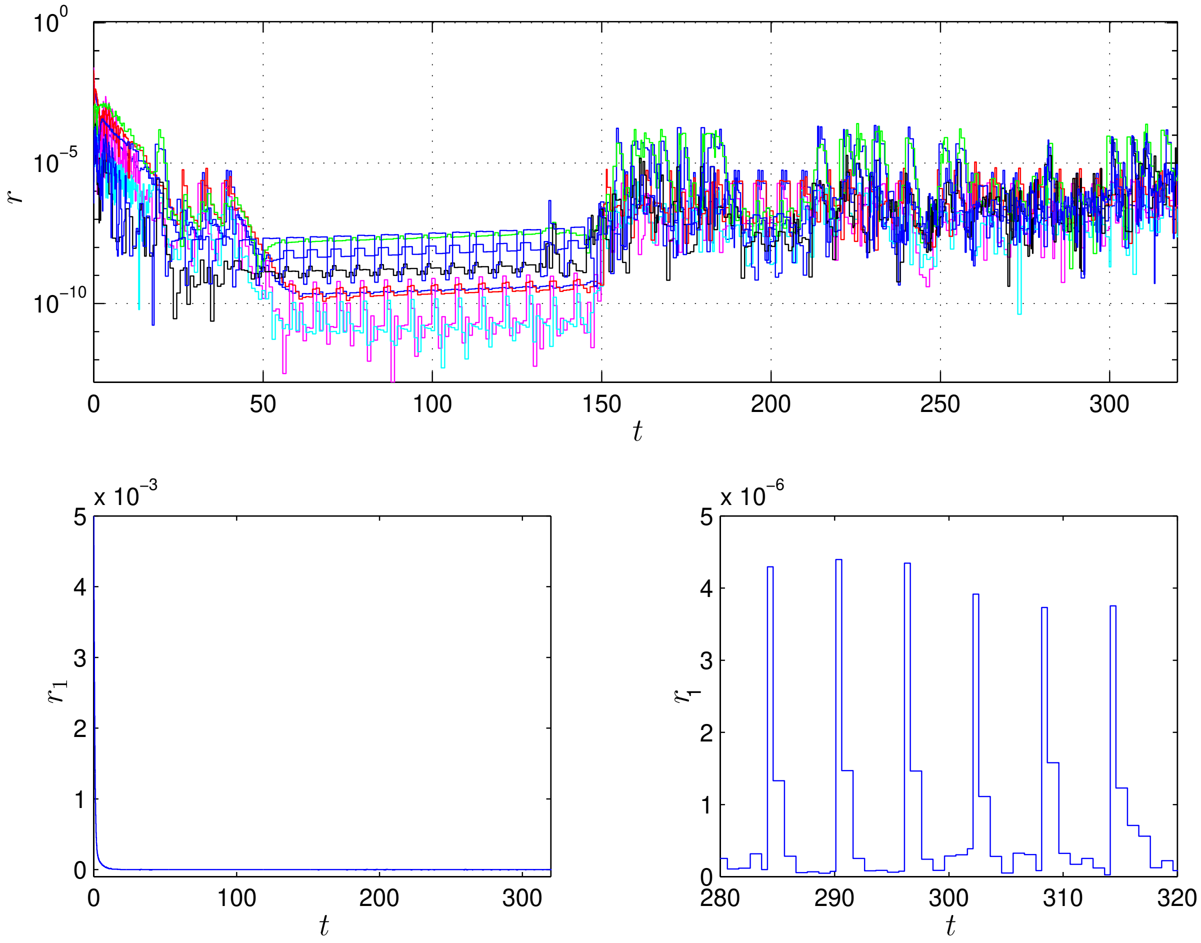



**Figure 38:** The individual residuals for the eight components of the ODE-system (above), and the residual for the first component (below).

7.2. **Stability factors.** After the initial transient, the variation of the solution is simple. This is reflected also in the stability properties of the problem. In Figure 39, we plot the sensitivity stability factors for the components of the ODE-system. These are defined as

$$S_i^{\mathcal{S}}(T) = \|\varphi(0)\|, \tag{26}$$

with $\varphi_j(T) = \delta_{ij}$. Notice that this is somewhat different from the definition in Section 5.1, where the $i$:th sensitivity stability factor measures the influence of an error in the initial value for component $i$, on the error for a given quantity of the solution. Here, the $i$:th sensitivity stability factor measures the influence of a general error in initial data on the error in component $i$. From the discussion in Section 5.1 we know that, assuming the equations are solved exactly, we have, for $i = 1, \ldots, N$,

$$|e_i(T)| \leq S_i^{\mathcal{S}}(T)\|e(0)\|. \tag{27}$$

The sensitivity stability factors for the components that quickly settle to constants quickly decrease below unity (they are all unity at time $t = 0$), whereas the sensitivity stability factors for components five and six increase steadily until finally also these decrease. (Compare with the behaviour of the solution components.)

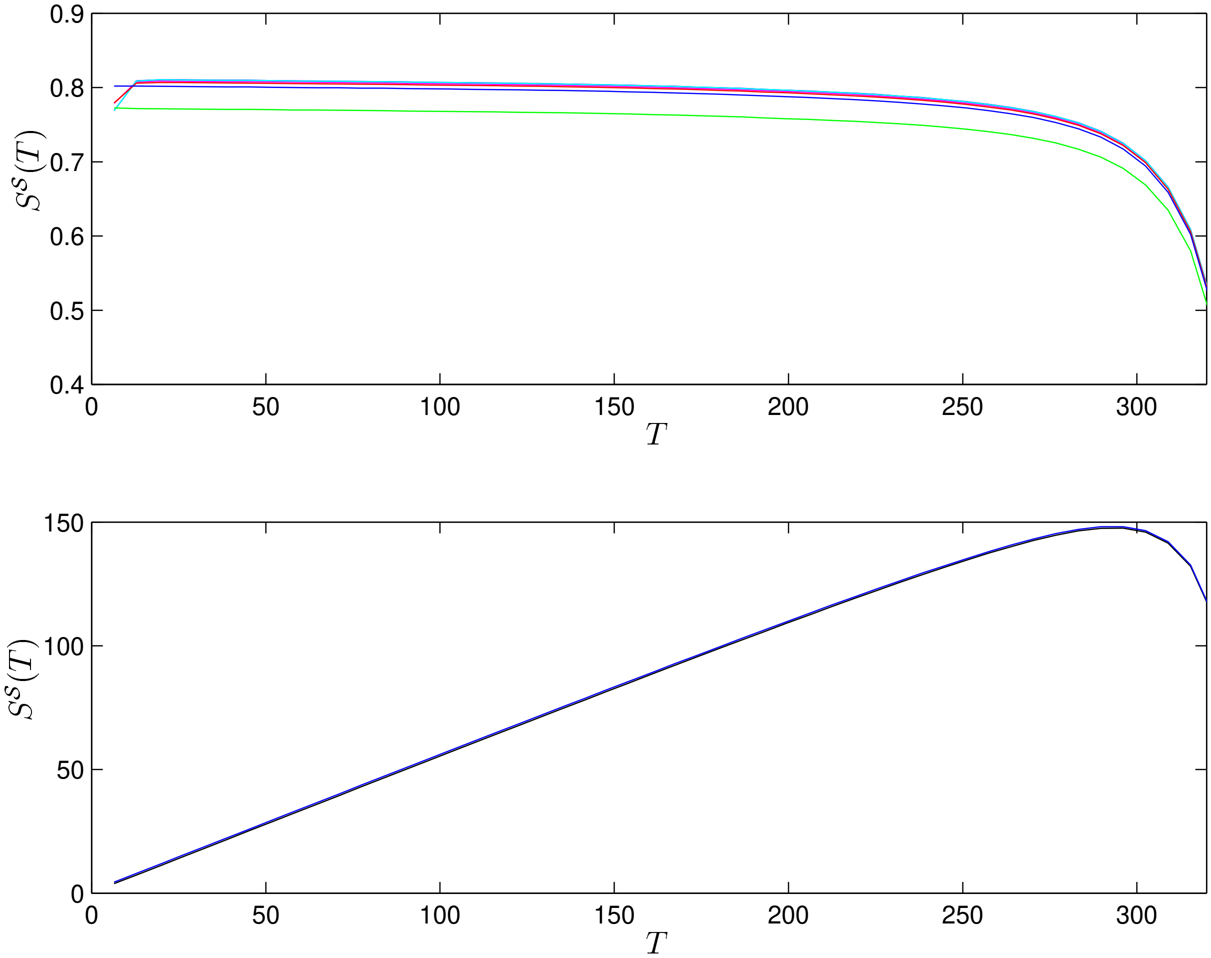



**Figure 39:** Sensitivity stability factors for components $1, 2, 3, 4, 7$ and $8$ (above) and for components $5$ and $6$ (below).

## 8. A Propagating Front Problem

We now consider the following system of partial differential equations:

$$\left\{ \begin{array}{rcl} \dot{u}_1 - \epsilon u_1'' & = & -u_1 u_2^2, \\ \dot{u}_2 - \epsilon u_2'' & = & u_1 u_2^2, \end{array} \right. \tag{28}$$

on $(0,1) \times (0,T]$ with $u_1(\cdot,0) = u_{10}$, $u_2(\cdot,0) = u_{11}$, and homogeneous Neumann boundary conditions at $x = 0$ and $x = 1$. This reaction-diffusion system models an isothermal auto-catalytic reaction, see [9]: Two substances, $A$ and $B$, distributed along $[0,1]$ with concentrations $u_1$ and $u_2$, $A$ reacts to form $B$ with $B$ working as a catalyst,

$$A + 2B \rightarrow B + 2B. \tag{29}$$

With a suitable choice of data conditions we will obtain a reaction front moving from the left to the right. Taking

$$u_{10} = \left\{ \begin{array}{ll} 0, & x < x_0, \\ 1, & x \geq x_0, \end{array} \right. \tag{30}$$

and $u_{20} = 1 - u_{10}$ for some $x_0 \in (0,1)$, the reaction will take place at $x = x_0$ and propagate to the right. For this to work we will need some diffusion.

We take $\epsilon = 0.001$ and solve for $T = 500$, using the cG(2) method after discretizing in space to obtain an ODE. The solution and the individual time-steps are plotted in Figure 40 at two different times. The reaction front propagates to the right, and so do the time-steps, in the sense that these are small only at the front. Outside of the reaction front the time-steps are large and reach some pre-defined maximum level, in this case $k_{\max} = 5$. (The individual Newton-scaling make time-steps as large as this possible.) See Figure 41 for a space-time plot of the solution and the time-steps.

In Figure 42 is another view of the time-steps. These are the time-steps for one of the two species at a specific location within the domain as function of time. The time-steps are clearly localized in both space and time. (In much the same way since the solution has a wave-like behaviour.) In this figure we also plot the corresponding residual as function of time. Keeping in mind that the plot is logarithmic we see that the residual is very small outside the reaction front, so that outside the front the time-steps are large.

8.1. **Stability factors.** We also include a plot of some different stability factors for the problem. These are defined as follows,

$$\begin{array}{rcl} S_i^{\mathcal{S}}(T) & = & |\varphi_i(0)|, \\ S_i^{\mathcal{G}}(T) & = & \int_0^T |\varphi_i^{(p_i)}(t)| \, dt, \\ S_i^{\mathcal{C}}(T) & = & \int_0^T |\varphi_i(0)| \, dt, \end{array} \tag{31}$$

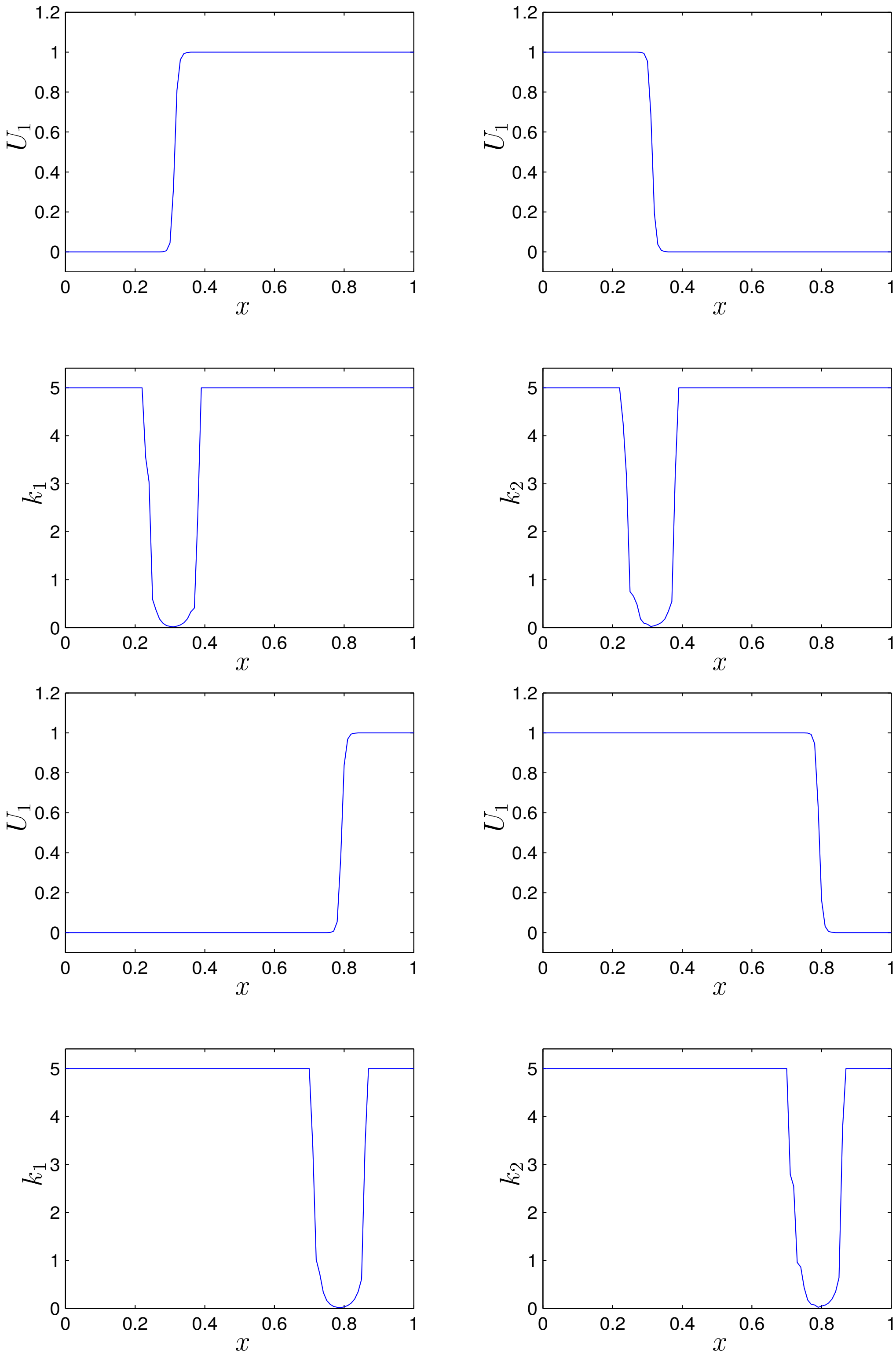



**Figure 40:** The solution and the time-steps at $t = 50$ (above) and $t = 300$ (below).

where the dual is solved backward from $t = T$ with data $\|\varphi(T)\| = 1$, and for the mcG($q$) method we have $p_i = q_i$. We refer to these as *sensitivity*, *Galerkin* and *computational* (or quadrature) stability factors. What is noteworthy is that these all decrease significantly when the reaction has finished and the solution is constant, and we interpret this so, that if we want to have a solution that is correct at a time beyond this point we do not need to be very careful; the solution will anyway be close to true solution which is constant.

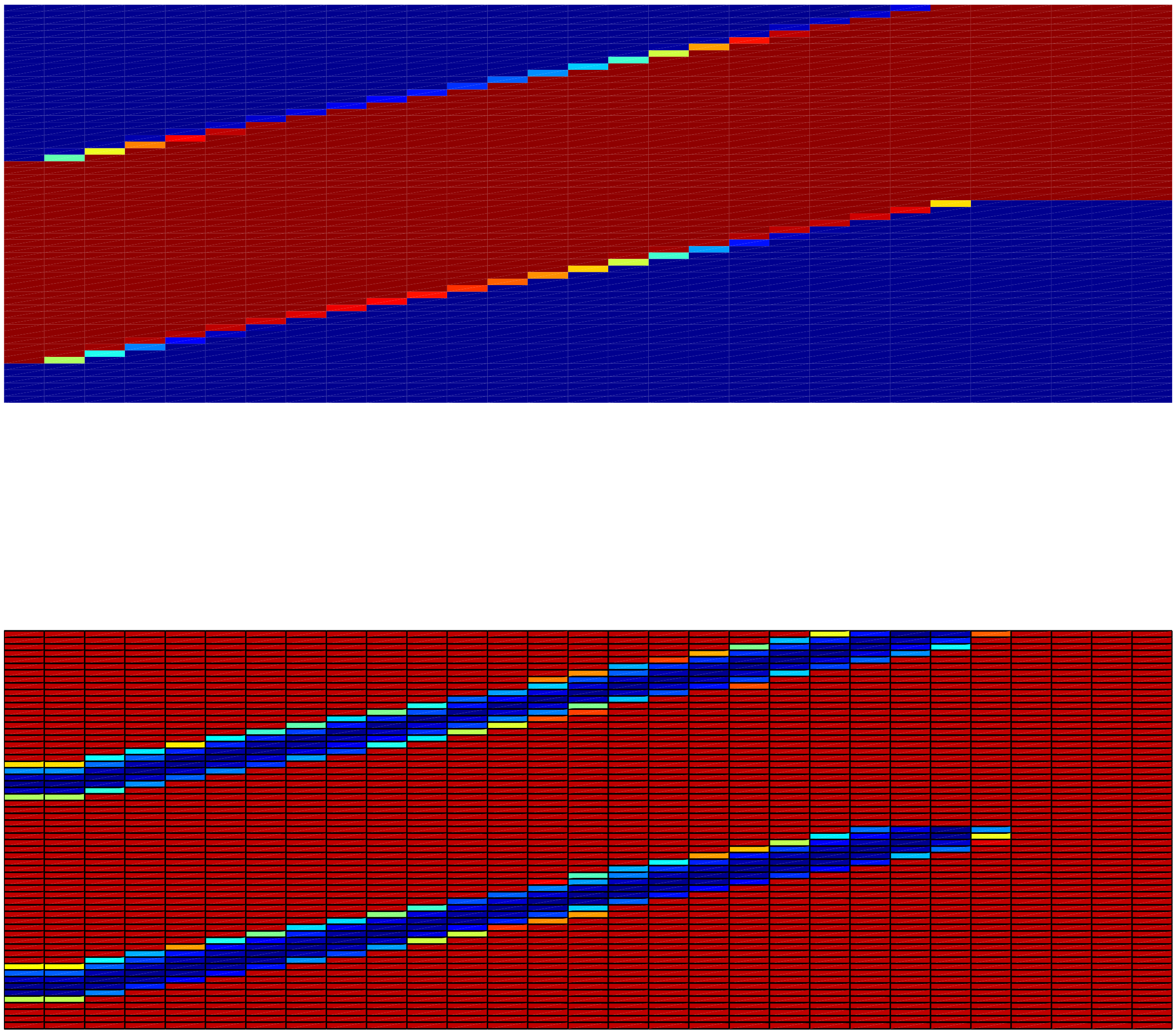


**Figure 41:** A space-time plot of the solution (above) and time-steps (below) for the propagating front problem, with time going to the right. The two parts of the plots represent the components for the two species $A$ (lower parts) and $B$ (upper parts).

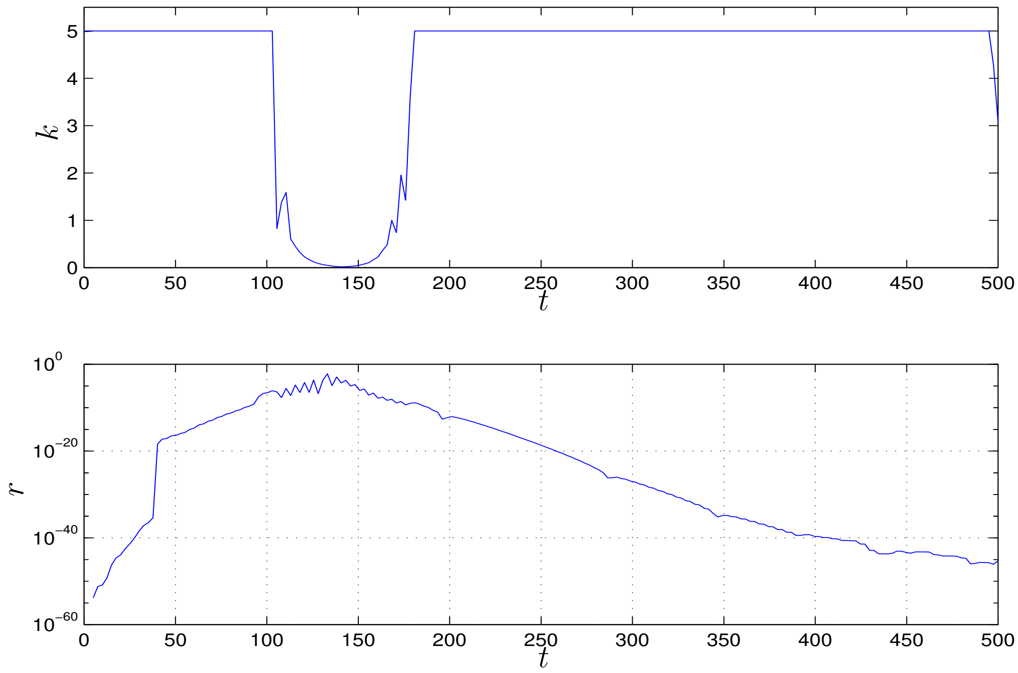



**Figure 42:** Time-steps and residuals for one of the species at a fixed position within the domain.

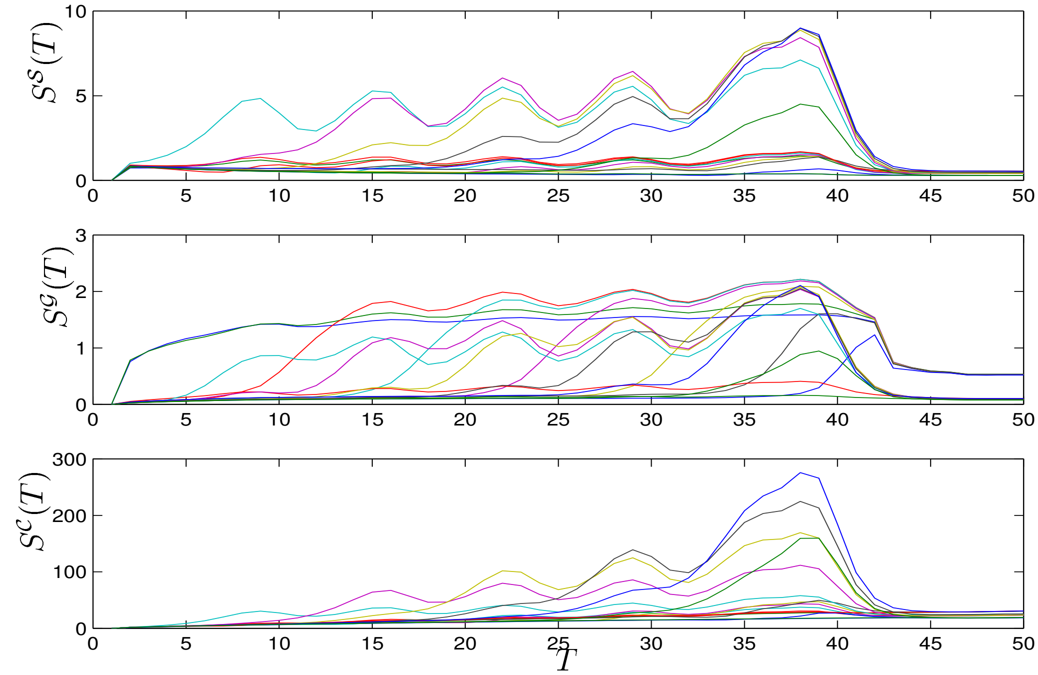



**Figure 43:** Different individual stability factors as function of time as described in the text.

## 9. A Mechanical Multi-Scale System

We consider the mechanical system in Figure 44, consisting of $N$ different point masses $\{m_i\}_{i=1}^N$, connected together with equal linear springs with spring constant $k$, which exhibits a range of different time-scales.

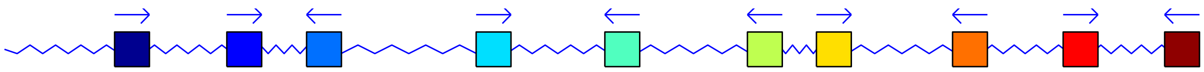


**Figure 44:** The mechanical multi-scale system for $N = 10$. The masses are different but we assume the spring constants to be the same.

Using Newton's second law, we model the system by

$$\left\{ \begin{array}{rcll} m_i \ddot{x}_i & = & k(x_{i+1} - x_i) - kx_i, & i = 1, \\ m_i \ddot{x}_i & = & k(x_{i+1} - x_i) - k(x_i - x_{i-1}), & 1 < i < N, \\ m_i \ddot{x}_i & = & -k(x_i - x_{i-1}), & i = N, \end{array} \right. \tag{32}$$

where $x_i(t)$ is the position of mass $m_i$ at time $t$. These equations can be written as a $2N$ first order system of the form $\dot{u} = f$.

We assume that

$$m_i = p^{-(i-1)}, \tag{33}$$

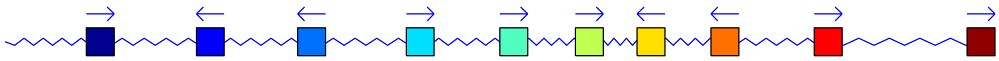
for some $p > 1$. Relating the masses to the frequency $\omega$ of the oscillating motions we have, somewhat simplified, $\omega = \sqrt{k/m}$, so that with e.g. $p = 4$, the frequency of should be doubled for every mass as we move from the left to the right. To obtain larger systems we repeat periodically, setting

$$m_i = p^{-\mathrm{mod}(i-1,n)}, \tag{34}$$

for some $n < N$. For example, if $N = 100$ and $n = 10$, we have

$$\{m_i\}_{i=1}^N = \{1, 1/2, \ldots, 1/2^9, \ldots, 1, 1/2, \ldots, 2^9\}. \tag{35}$$

We will also consider a system consisting of one small mass and many larger masses. In this case the dynamics of the system will be completely dominated by the smallest mass. We expect the multi-adaptive method to perform particularly well on such a problem, using small time-steps only for the smallest mass.



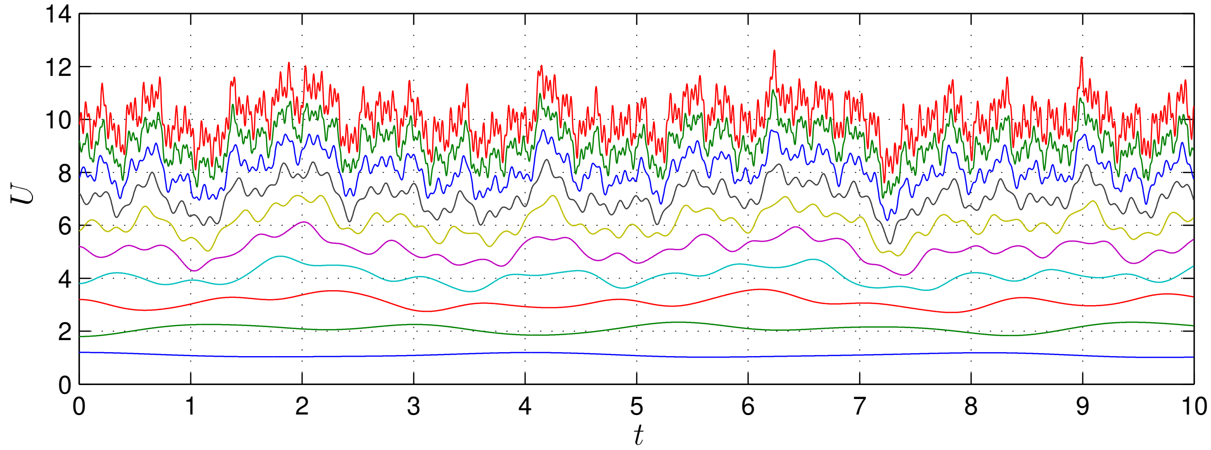



**Figure 45:** The positions of the masses of the mechanical multi-scale system as function of time for $N = 10$, $p = 3$. Notice the presence of many different scales.

9.1. **The solution.** We now choose $p = 3$ and solve for $T = 10$. For this system the difference in scales between the finest scale and the coarsest scale is roughly a factor $\sqrt{3}^9 \approx 140$. The large difference in scales is evident in Figure 45. It is also directly evident in Figure 46, where we plot the multi-adaptive time-steps for the different components.

Ideally we would like to see that the size of the time-steps of a component should precisely reflect the frequency for that component. To see that this is the case, that the residual-based time-steps are chosen in agreement with the choice we would make ourselves if we were to make a clever guess for the time-steps, we plot the mean time-steps for every component in Figure 47. Comparing the mean time-steps we see that they agree well with the predicted factor of $\sqrt{3}$ between adjacent scales.

9.2. **Multi-adaptive time-stepping.** If we use $M$ time-steps for the slowest component and the time-steps for all components are chosen to be the same, the total number of time-steps will be $MN\sqrt{p}^{N-1}$, neglecting the time-stepping of the velocity components. With different time-steps for the different components, the total number of time-steps will be

$$M(1 + p^{1/2} + \ldots + p^{(N-1)/2}) = M\frac{\sqrt{p}^N - 1}{\sqrt{p} - 1} \leq \frac{M}{1 - 1/\sqrt{p}}\sqrt{p}^{N-1}. \tag{36}$$

We thus gain a factor $N(1 - 1/\sqrt{p})$. For $p = 4$ this means a factor $N/2$.

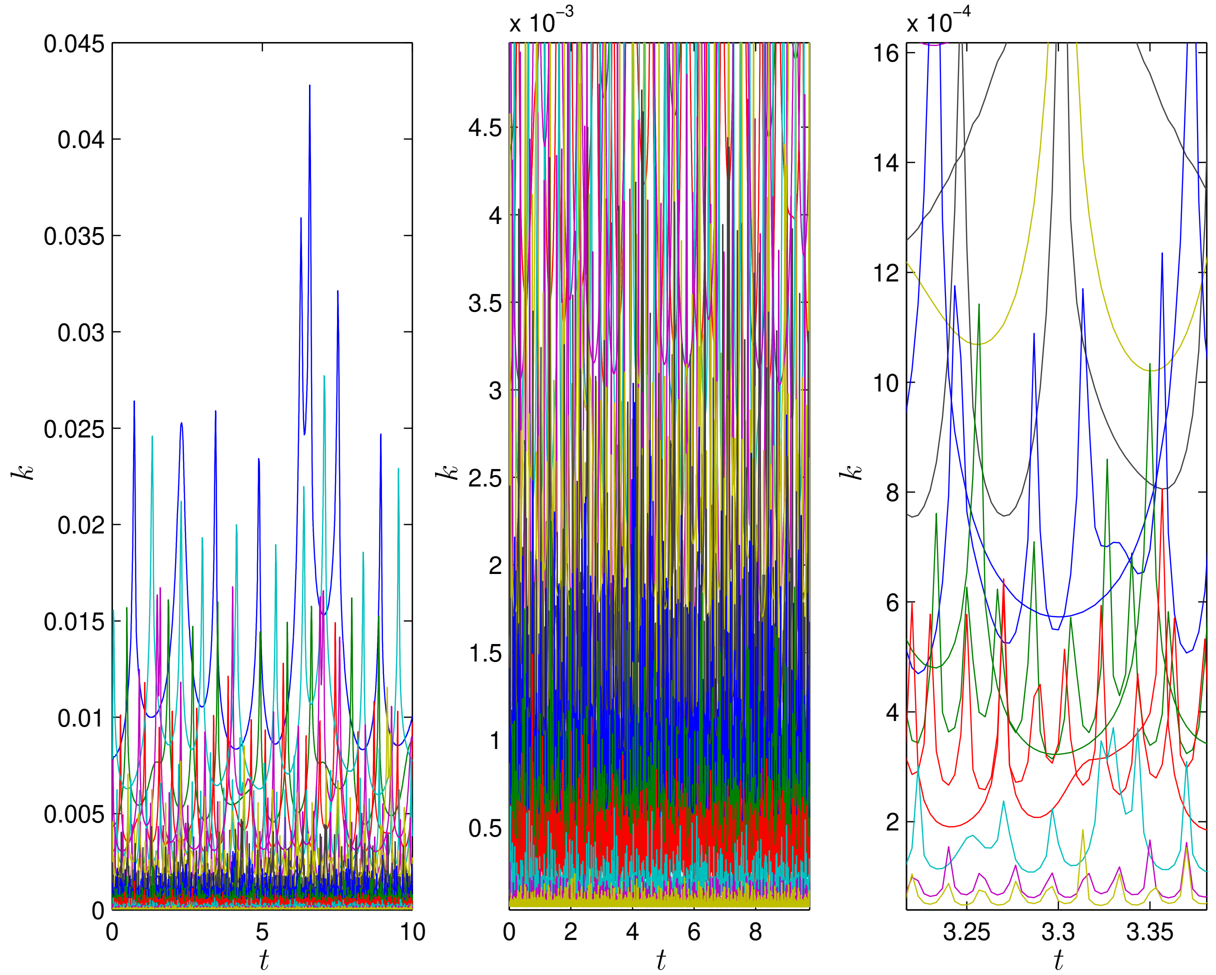



**Figure 46:** The time-steps for the position components of the mechanical multi-scale system as function of time for $N = 10$, $p = 3$. The presence of different scales is reflected by the different time-steps.

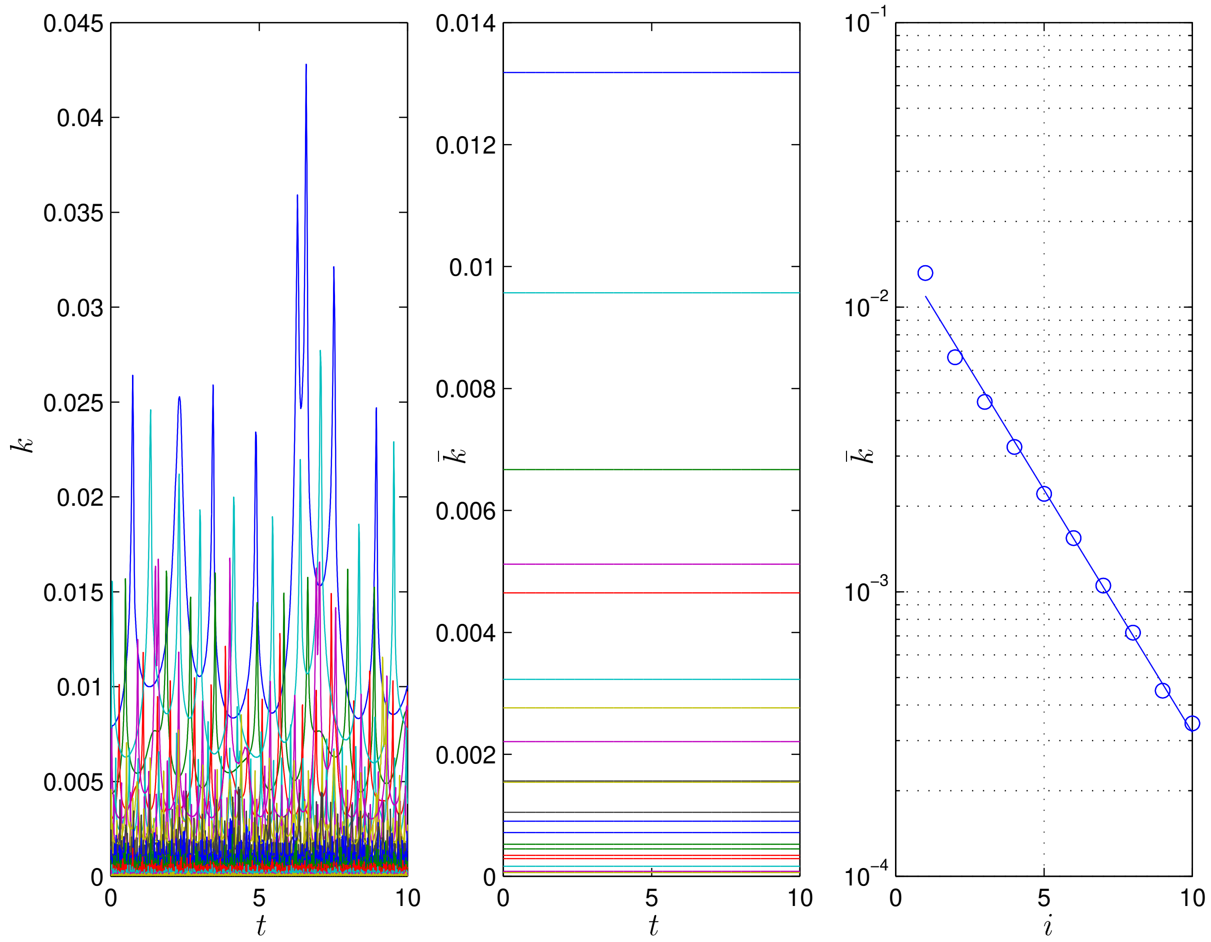



**Figure 47:** The time-steps for the position components of the mechanical multi-scale system as function of time for $N = 10$, $p = 3$ (left), mean time-steps (middle), and mean time-steps as function of component index.

9.3. **Increasing the size of the system.** Increasing now the size of the system to a thousand particles, meaning an ODE-system with 2000 components, we take the masses as in (34), with $n = 10$, i.e. the 1000 masses are ordered in 100 packages of ten. We take $p = 2$ and solve using the mcG(1) method, and obtain the solution presented in Figure 48. Masses of the same sizes sit in similar positions and they have similar, but not the same, trajectories. We thus have 200 components moving with the lowest frequency, 200 components moving with the highest frequency, and 1800 components in between. Note also that positions and velocities for a certain mass need not, and will often not, use the same time-steps. (If we want the method to be energy-conserving we have to use the same time-steps for corresponding position and velocity components, see [7].)

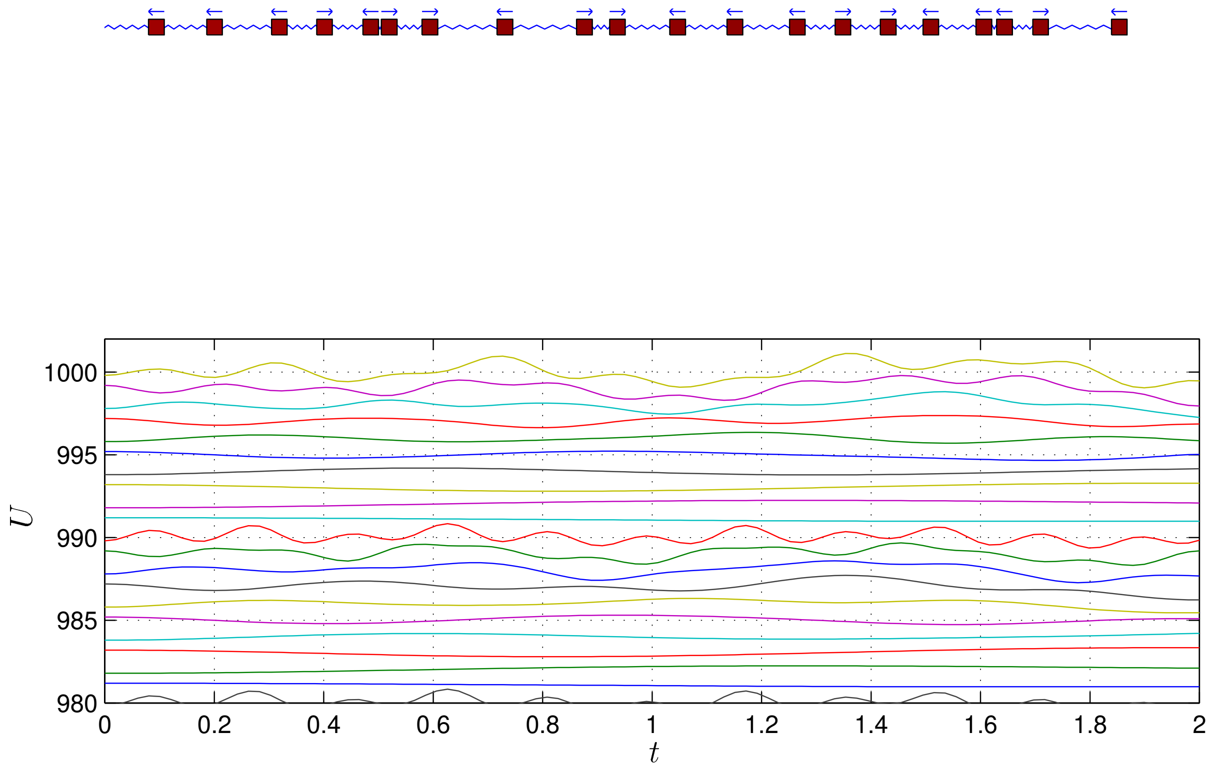



**Figure 48:** The mechanical multi-scale system for $N = 1000$. The plots show the motion and solution for a part of the system.

All 2000 components of the ODE-system thus have their own individual and time-dependent time-steps, residuals, and stability weights. For this particular computation we have chosen to base the time-steps on an equidistribution of the error, i.e. keeping the stability weights within the sum of the a posteriori error estimate, rather than taking a maximum to obtain global stability factors. (See [7] for a discussion of this.)

9.4. **Stability weights.** To investigate the local stability weights we decrease the size of the system to 100 masses, which is large enough for demonstration purposes. Examining Figure 50 we find that, as we expect, the stability weights are ordered in groups. Apart from the grouping based on the size and position of the corresponding mass, the components will also group depending on whether they represent a position or a velocity. To keep the discussion simple, we focus on the components representing positions, i.e. components with indices $1, 2, \ldots, 100$.

We expect these components to group into ten different groups, possibly with the exception of the first and the last components that correspond to the mass attached to the wall and the last little loose mass at the end, since these are special.

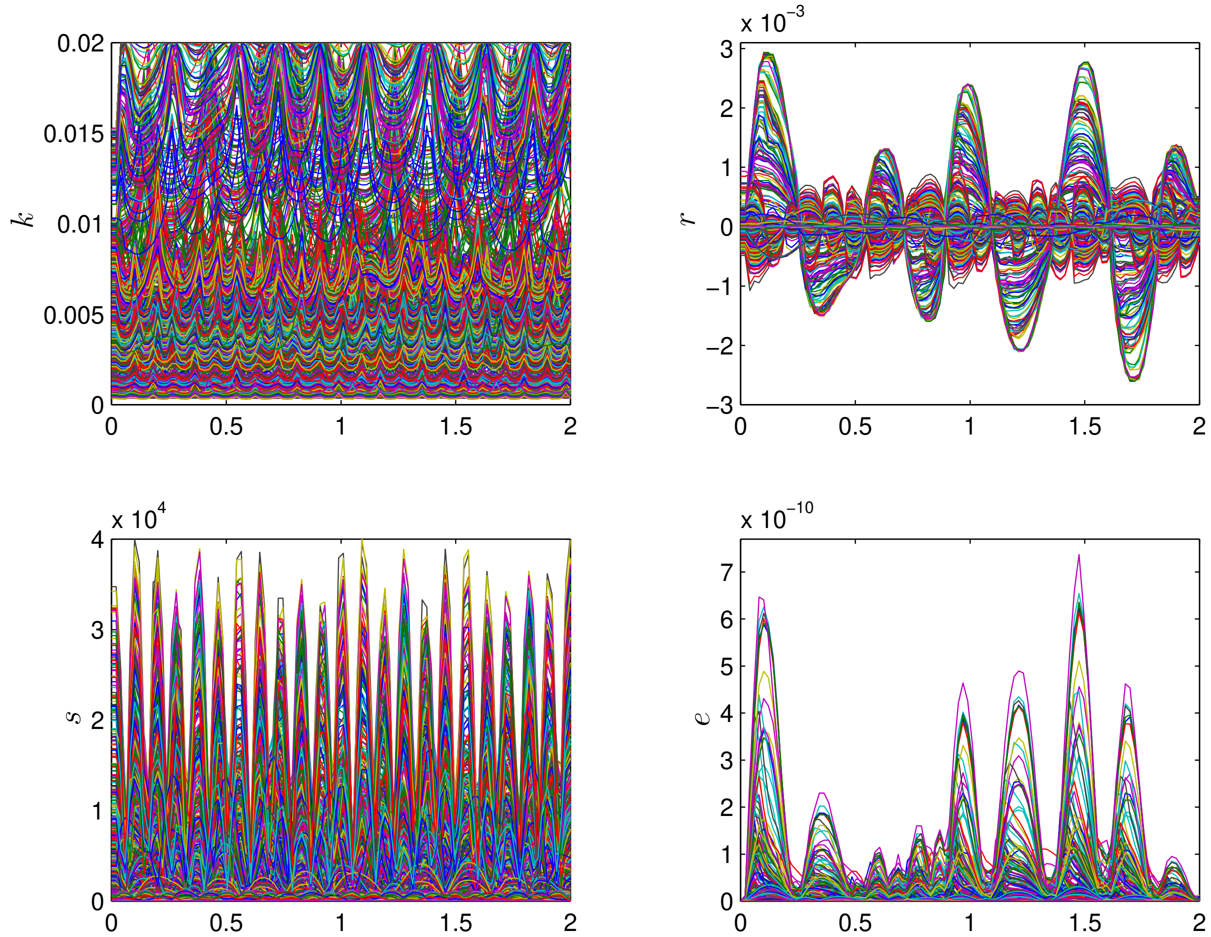



**Figure 49:** Time-steps $(k)$, residuals $(r)$, stability weights $(s)$ and "local errors" $(e)$, i.e. $s_i^{[q_i]} k_i^{q_i} |R_i|$, for the mechanical multi-scale system with $N = 1000$.

Investigating the stability weights more closely, as in Figure 51, the structure becomes more clear. The most clearly visible groups of stability weights are those labelled $A$, $B$, $C$ and $D$ in this figure, and these groups contain the following components:

| | | |
|---|---|---|
| $A$ | : | 10,20,...,90, |
| $B$ | : | 9,19,...,89, |
| $C$ | : | 11,21,...,91, |
| $D$ | : | 8,18,...,98. |

**Table 4:** Grouping of the component stability weights as indicated in Figure 51.

These are components that are close to the high-frequency motion occurring nearby $i = 10, 20, \ldots, 100$. What is missing in this picture is perhaps the last component, number 100. As mentioned before, this one is a bit special since it has only one neighbour. It therefore does not fall into this group, and following the discussion below it will be clear why. The first component is not present since to the left of it is the wall and not a high-frequency component, as is the case for components $11, 21, \ldots, 91$.

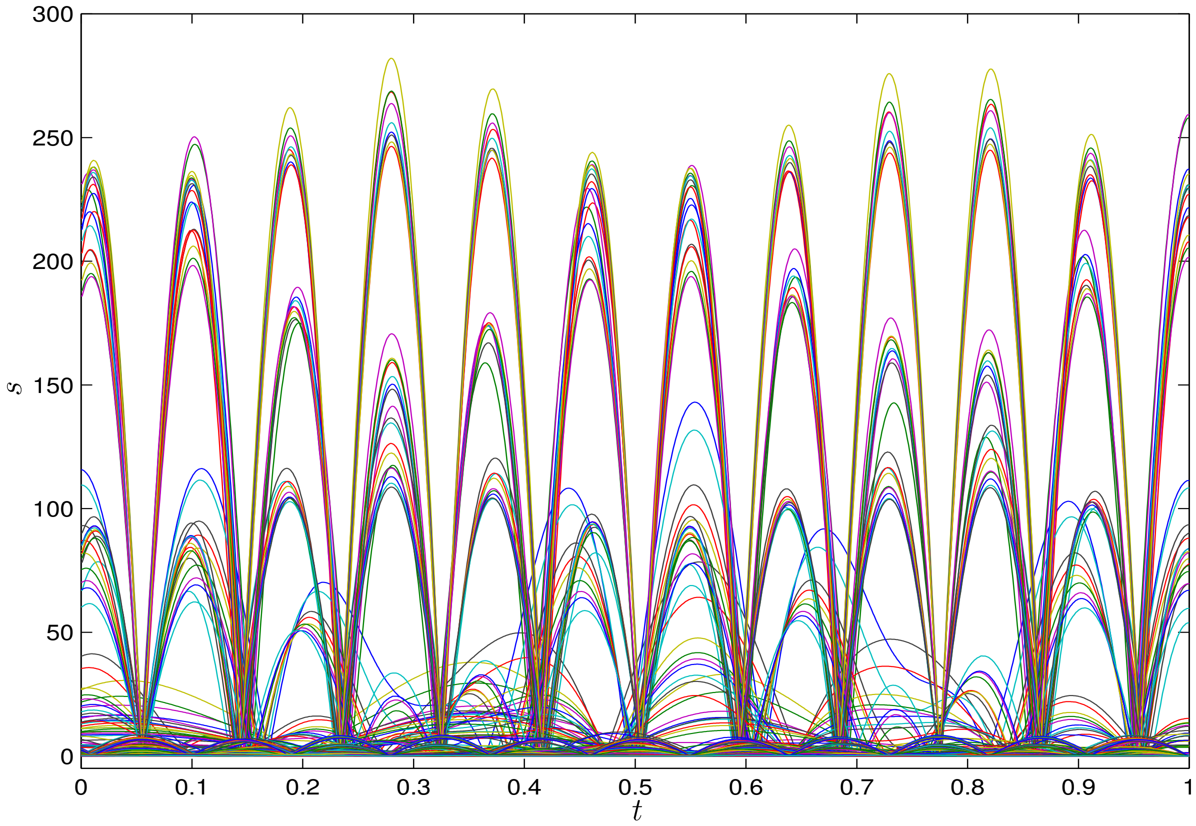



**Figure 50:** Local stability weights for the solution of the mechanical multi-scale system for $N = 100$. The stability weights seem to be ordered in groups.

The components of groups $A$, $B$, $C$ and $D$ all have particularly large stability weights. We will now attempt to give an explanation of this and perhaps some further insight to the mystery of stability factors.

The dual problem, from which we compute the stability factors and the stability weights, is the linearized adjoint problem

$$-\dot{\varphi}(t) = J^*(t)\varphi(t), \tag{37}$$

where $J$ is as defined in [7]. Think of it as the Jacobian of the right-hand side.

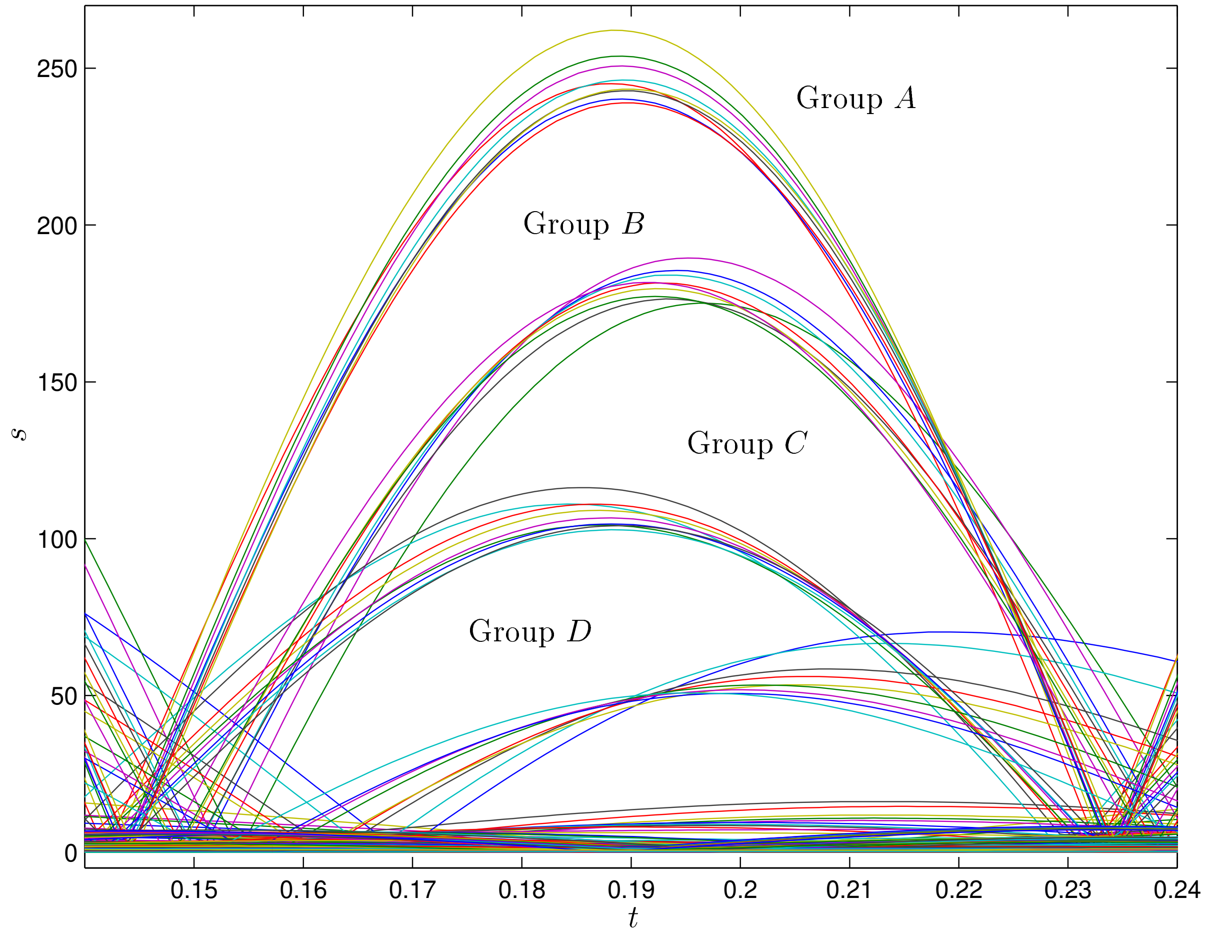



**Figure 51:** Grouping of local stability weights for the solution of the mechanical multi-scale system for $N = 100$.

Now, if component $i$ depends to a large extent on component $j$, there will be a large entry at position $(i, j)$ in the Jacobian. Large entries on row $i$ in its *transpose* $J^*$ thus means that other components (including itself) in the original primal problem depend heavily on this component. Since now large entries on row $i$ of $J^*$ may contribute to making the $i$:th component of the dual large, we conclude that *the local stability factor for a certain component reflects how much that component influences other components in the primal problem*:

$$\dot{u} = \begin{bmatrix} & & \\ & * & \\ & * & \\ & * & \\ & & \end{bmatrix} u \quad \Rightarrow \quad -\dot{\varphi} = \begin{bmatrix} & & \\ & & \\ * & * & * \\ & & \\ & & \end{bmatrix} \varphi. \tag{38}$$

This is in agreement with our example, where the components with the highest stability factors are those that are close to and influence small masses, since these are easiest to influence. Since the time-steps are based on both the stability properties of the dual and on the residuals, the components corresponding to large masses will still use larger time-steps; the motion is slow and the residuals will therefore be small. The time-steps will, however, be smaller than if they had been based on residuals alone. We thus have to be more careful with the slow components than one might think, since small errors for these components may cause large errors for the fast components. (In this case where fast means light and easy to move.) In Figure 52 we compare time-steps and residuals for the fastest and slowest components of the system.

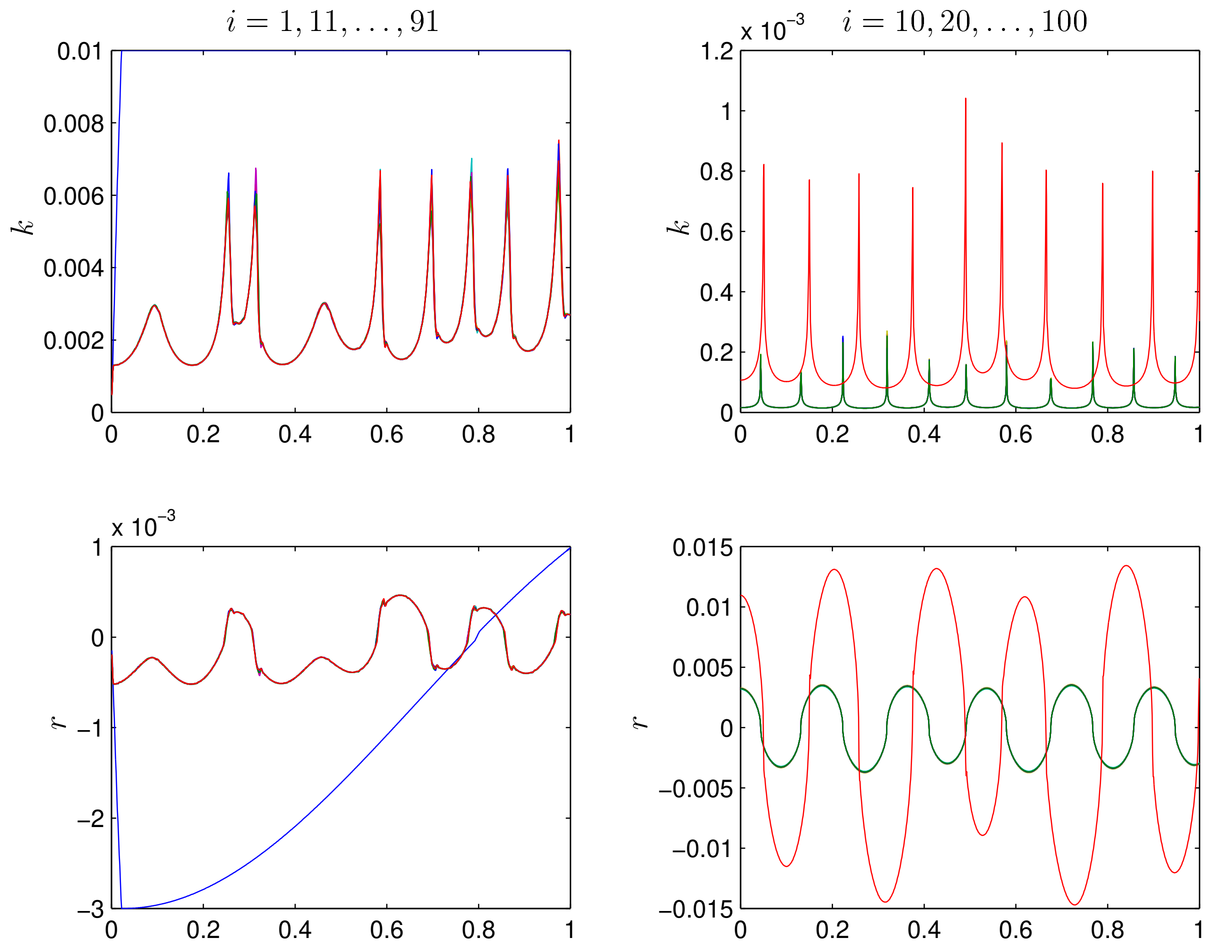



**Figure 52:** Time-steps and residuals for components of the mechanical multi-scale system for $N = 100$; components corresponding to large masses on the left and component corresponding to small masses on the right. The time-steps are almost identical for components at similar positions in the system, with the exception of the first component (which is attached to the wall), and the last component (which does not have a component on its right-hand side).

9.5. **A multi-adaptive test problem.** To further demonstrate the potential of the multi-adaptive methods, and to specify a test problem for multi-adaptive time-stepping, we change the setup of the mechanical multi-scale system considered above.

The problem is as before to compute accurately the positions (and velocities) of the $N$ point-masses connected with springs of equal stiffness as in Figure 53.

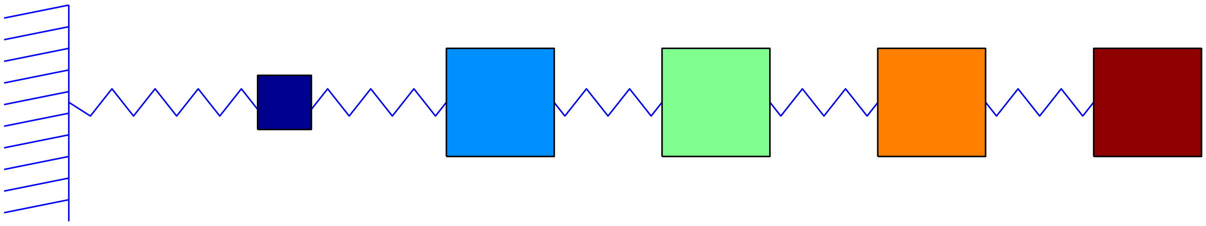


**Figure 53:** A mechanical system consisting of $N = 5$ masses attached together with springs.

Choosing one of the masses much smaller than the other, $m_1 = 10^{-4}$, $m_i = 1$ for $i = 2, \ldots, N$, we expect the dynamics of the system to be dominated by the smallest mass, in the sense that the resolution needed to compute the solution will be completely determined by the fast oscillations of the smallest mass.

To compare the multi-adaptive method with a standard method, we first compute with constant time-steps $k = k_0$ using the standard cG(1) method, and measure the error, the cpu time needed to obtain the solution, the total number of steps, i.e. $M = \sum_{i=1}^{N} M_i$, and the number of local function evaluations. We then compute with individual time-steps, using the mcG($q$) method, choosing the time-steps as $k_i = k_0$ for the position and velocity components of the smallest mass, and as $k_i = 100k_0$ for the other components, since we know that the frequency of the oscillations behaves like $1/\sqrt{m}$. For demonstration purposes, we thus choose the time-steps a priori to fit the dynamics of the system. Comparing the results for the two methods, $k_i = k_0$ for all components and $k_i = k_0$ only for the fast components, we find that the error is basically the same. As $N$ increases the total number of time-steps, the number of local function evaluations (including also residual evaluations), and the cpu time increase linearly for the standard method, as we expect.

For the multi-adaptive method, on the other hand, the total number of time-steps and local function evaluations remain practically constant as we increase $N$. The cpu time increases somewhat, since the increasing size of the time-slabs introduces some overhead, although not nearly as much as for the standard method.

For this particular problem the gain of the multi-adaptive method is thus a factor $N$, where $N$ is the size of the system, so that by considering a large-enough system, the gain is arbitrarily large.

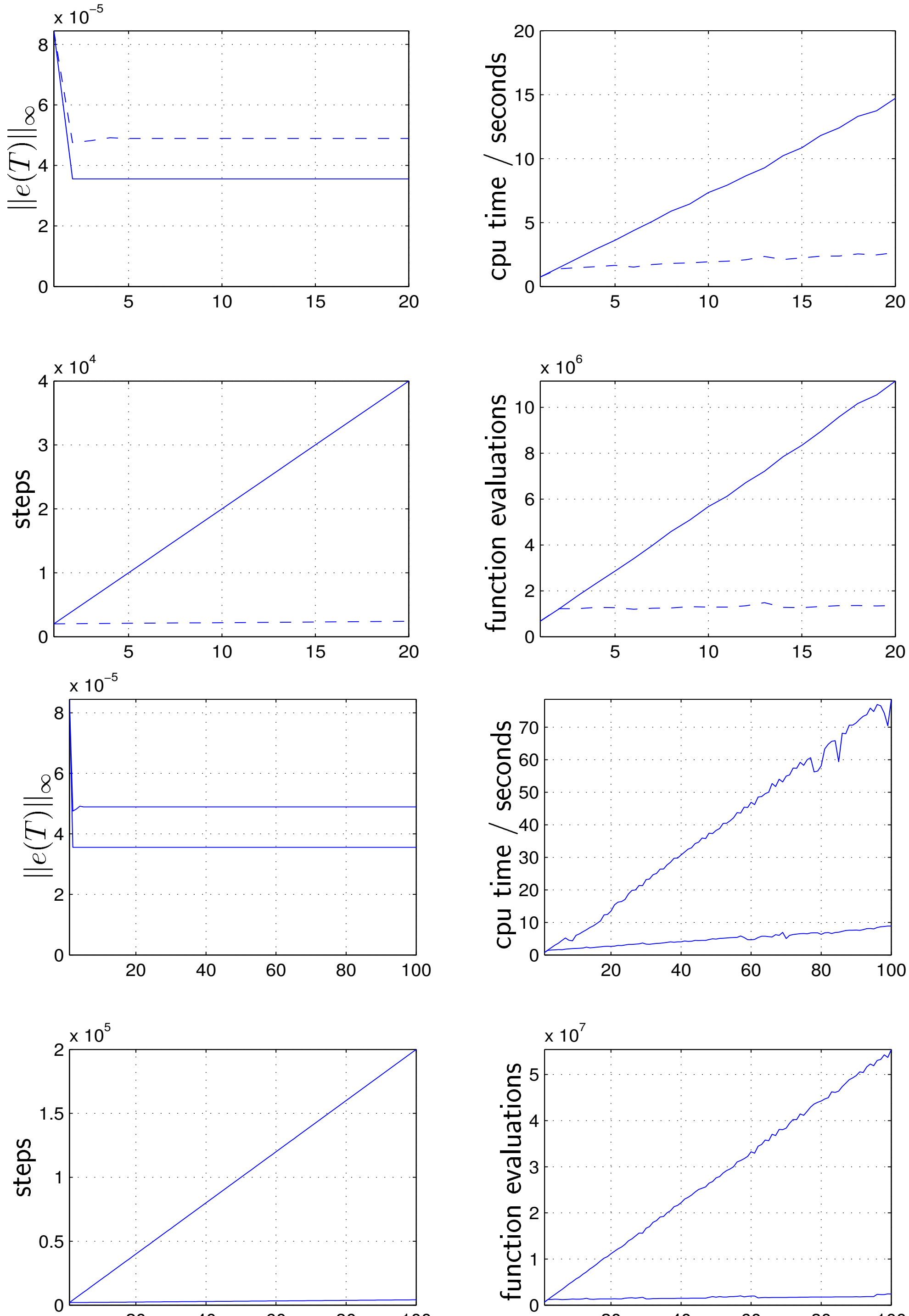



**Figure 54:** Error, cpu time, total number of steps, and number of function evaluations as function of the number of masses, for the multi-adaptive cG(1) method (dashed) and the standard cG(1) method.

## 10. The Wave Equation

A closely related problem is the wave equation,

$$\ddot{u} - u'' = 0, \tag{39}$$

on $(0,1) \times (0,T]$ together with initial and boundary conditions. Starting with a pulse at $x = 1/2$, we expect to see two pulses of half the size propagating in both directions, as in Figure 55.

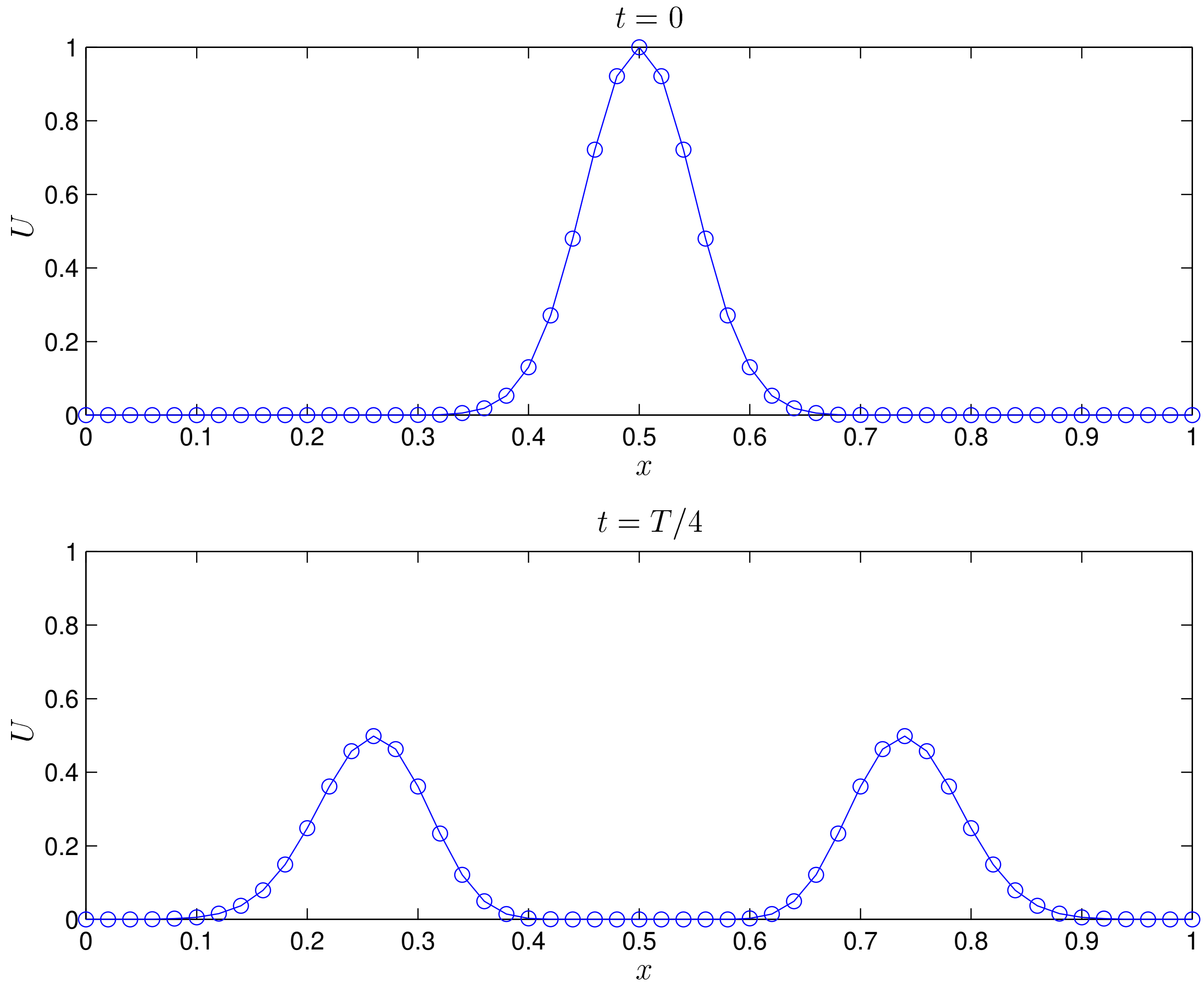



**Figure 55:** The solution will be two waves, travelling in opposite directions.

Solving with the mcG(1) method, with large enough time-steps and coarse enough grid, we get the solution given in Figure 56. As we expect, the solution "wiggles" in the same way as with the cG(1) method (Crank-Nicolson).

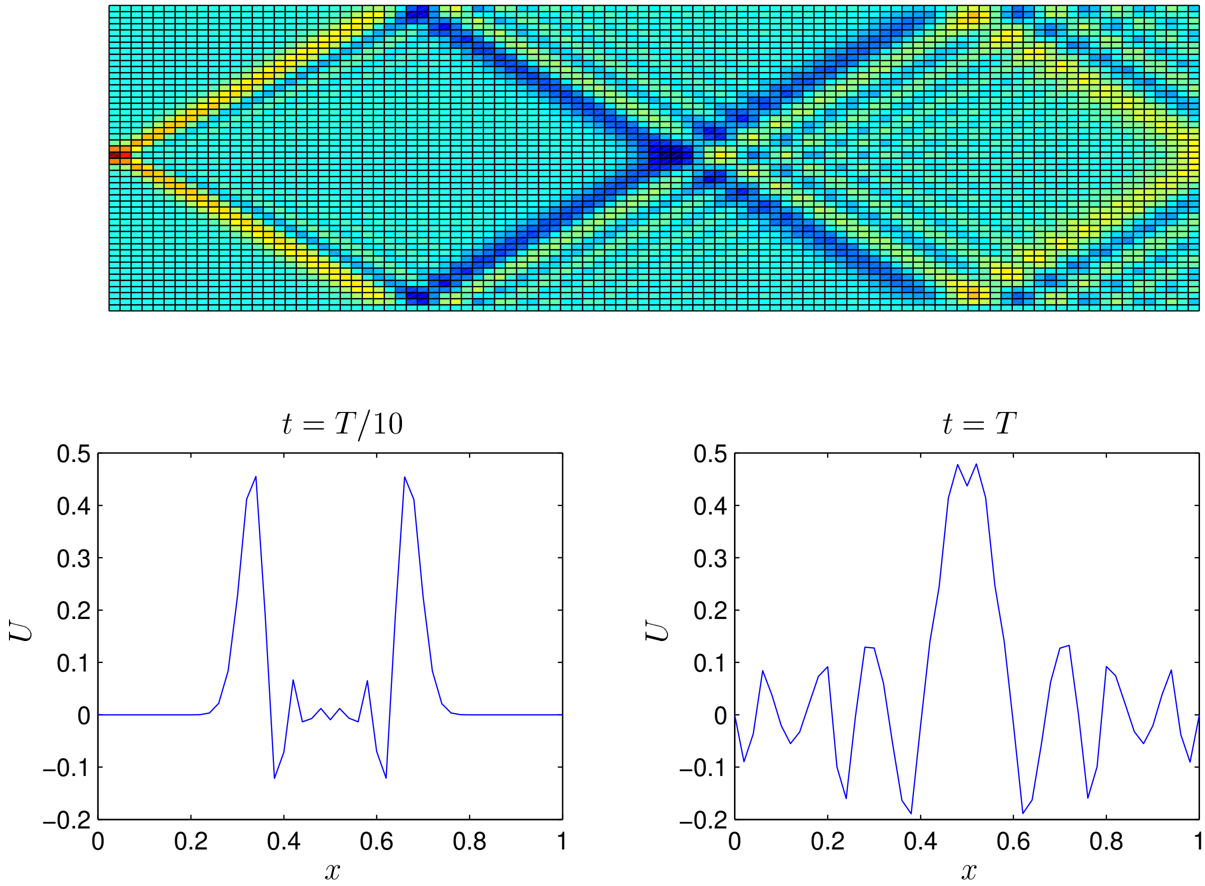



**Figure 56:** The mcG(1) solution at two different times.

We expect the time-steps to be small only near the propagating peaks, and large elsewhere. Examining a plot of the time-steps as function of time and space — see Figure 57 — it is clear that the time-steps are localized in both space and time. To see that the residual-based time-steps are chosen properly, notice the following: By an a priori error estimate, we have for the mcG(1) method something like

$$e \sim k^2 \ddot{u}, \tag{40}$$

which for the wave equation means we have

$$e \sim k^2 u''. \tag{41}$$

We may also realize this directly; the mcG(1) method with its piecewise linear approximation of the solution, should have difficulties at such occasions when and where the second derivative is large. The time-steps should therefore be small whenever $u''$ is large.

Plotting thus the second derivative of the exact solution at $t = 0$ in Figure 58, and comparing to the residuals and the time-steps in Figure 59, we find that the residual reflects the size of $u''$ and that the time-steps are chosen properly. (The residual contains one time-step, so that we have $e \sim kR \sim kku'' = k^2u''$ in agreement with the a priori error estimate.)

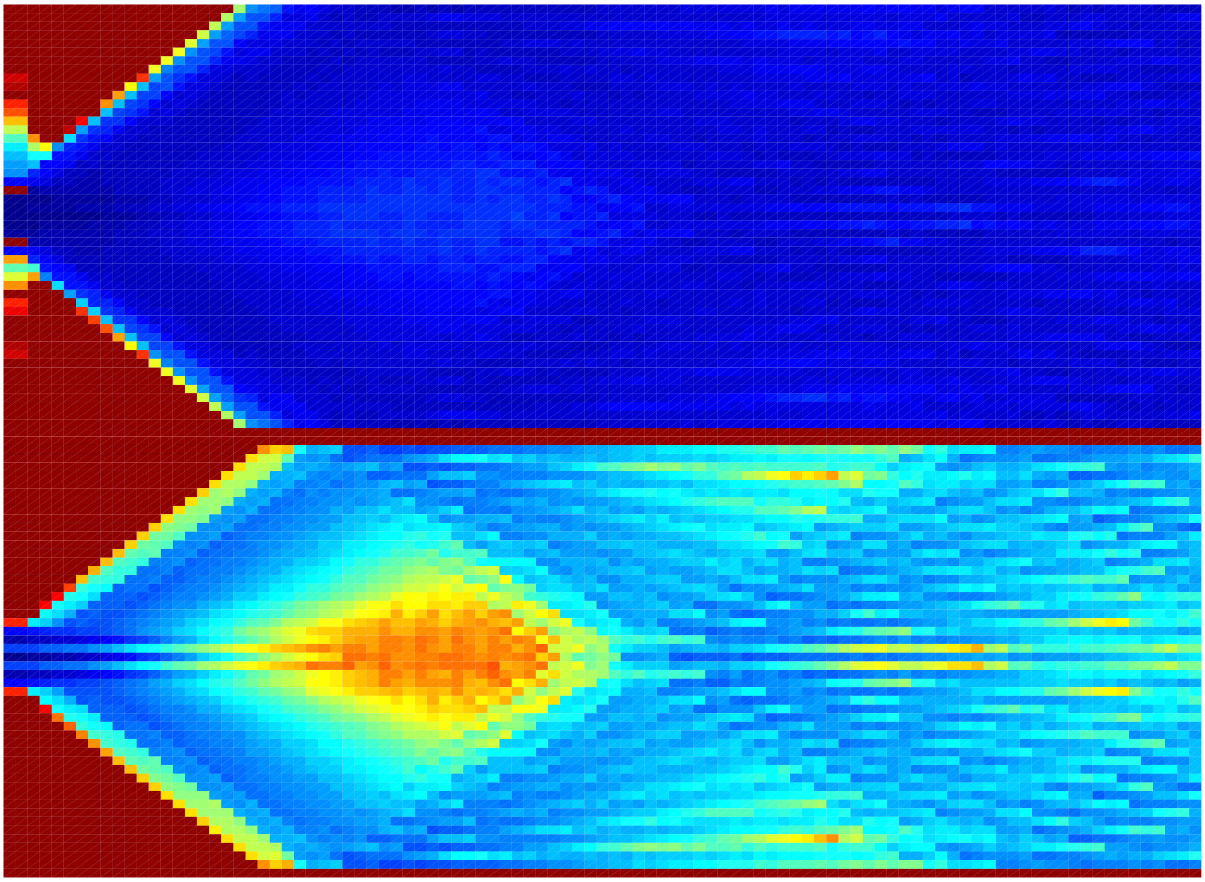


**Figure 57:** Time-steps as function of time (going to the right) and space for the mcG(1) solution of the wave equation, below time-steps for the positions and above time-steps for the velocities.

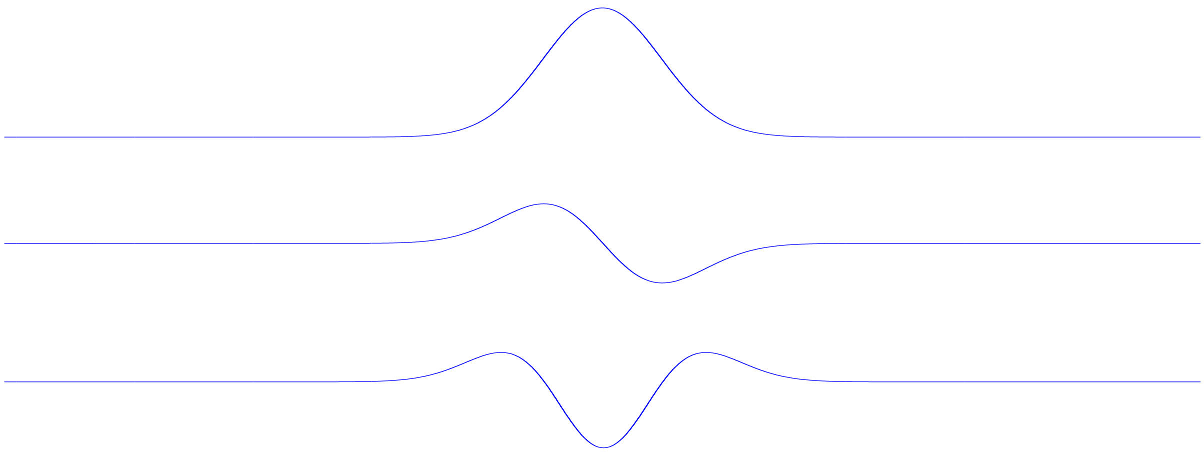


**Figure 58:** The initial condition, its derivative and its second derivative.

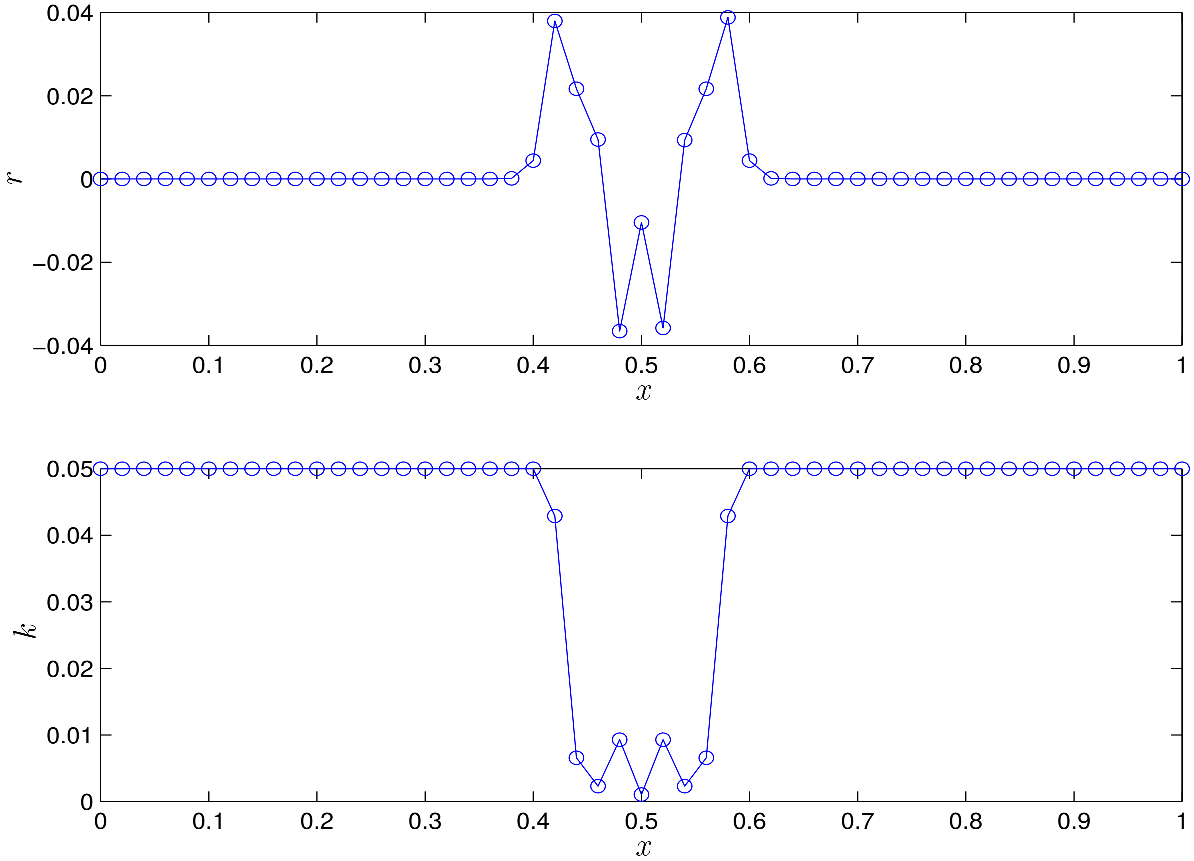



**Figure 59:** Residuals and time-steps at time $t = 0$. The residual resembles the second derivative, as in Figure 58.

## 11. The Heat Equation with Local Heating

As another example of a standard PDE, we solve the heat equation,

$$\dot{u} - \Delta u = f(x,t),$$

on $\Omega \times (0,T]$ with homogeneous Dirichlet boundary conditions, where $\Omega$ is a square in $\mathbb{R}^2$ centred at $(0,0)$.

We discretize in space and solve the resulting ODE with the multi-adaptive method as before. Solving for the right-hand side

$$f(x,t) = (1+t)\sin(t)\exp(-((x/0.1)^2 + (y/0.1)^2)/2),$$

we have a source localized in space and time, and we want to see the time-steps localized in the same way. The solution will be localized to $x = y = 0$, oscillating up and down. We plot the solution at three different times in Figure 61.

The time-steps will behave in much the same way, being quite large, reaching some pre-defined largest time-step, close to the boundary where the solution is constant, and oscillating from small to large in the middle of the domain.

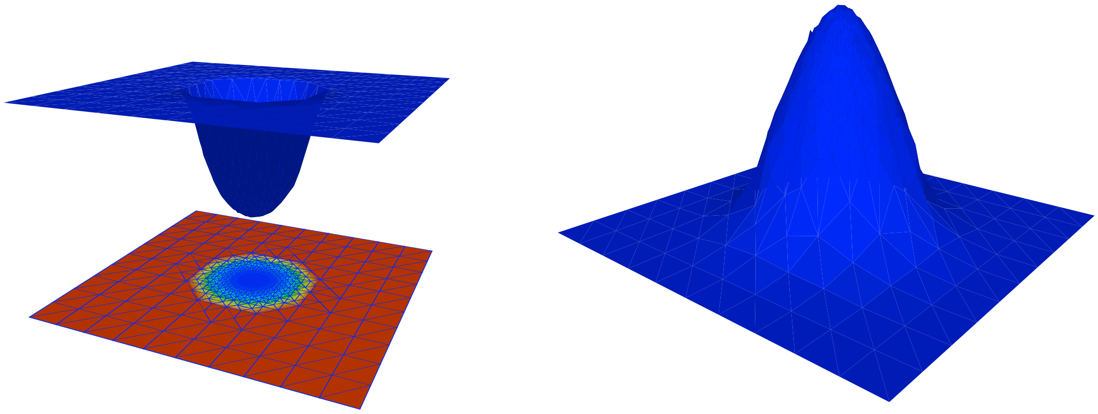


**Figure 60:** Time-steps (left) and residuals (right) for the solution of the heat equation with local heating.

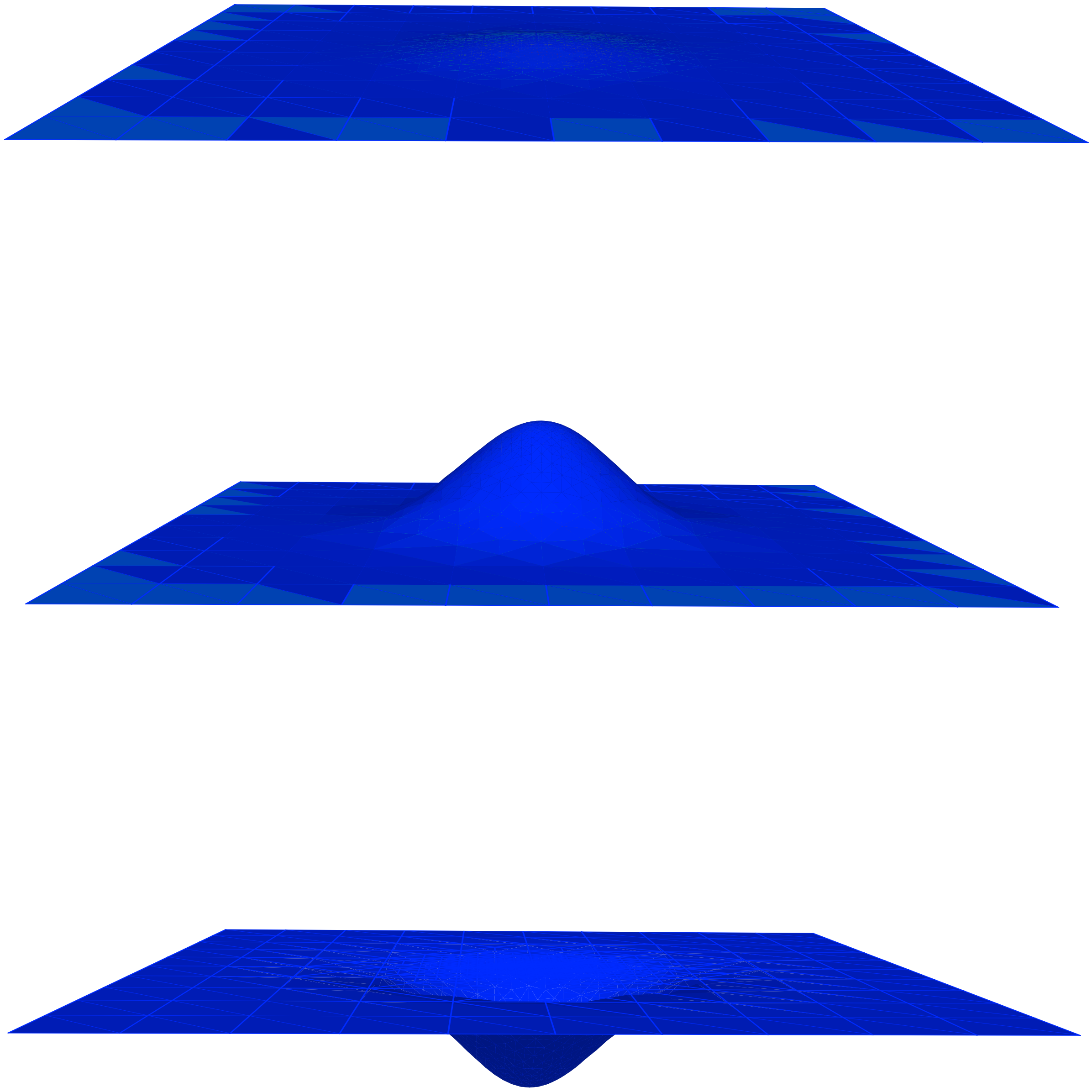


**Figure 61:** The solution to the heat equation with local heating at three different times.

The time-steps vary also in time, being large whenever the solution is close to linear (compare Section 2). In this way, twice every period the time-steps are large, and twice they are small, at least close to $x = y = 0$ where all the action is (see Figure 62).

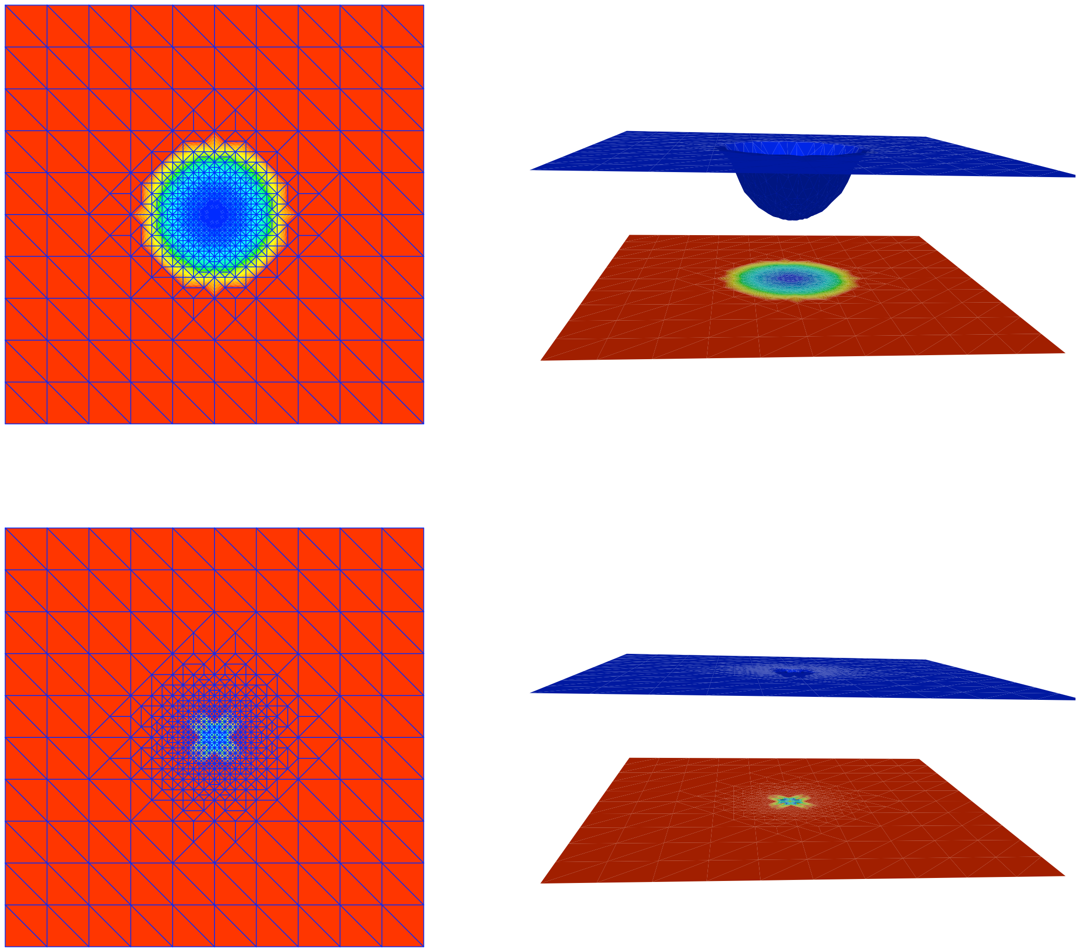


**Figure 62:** Time-steps for the solution of the heat equation at two different times.

## 12. Burger's Equation with Moving Nodes

We turn now to Burger's equation,

$$\dot{u} + \mu u u' - \epsilon u'' = 0, \tag{42}$$

on $(0,1) \times (0,T]$ with initial condition

$$u_0(x) = \begin{cases} \sin(\pi x / x_0), & 0 \le x \le x_0, \\ 0, & \text{elsewhere}, \end{cases} \tag{43}$$

and where $\mu = 0.1$, $\epsilon = 0.001$ and $x_0 = 0.3$.

The solution is a shock forming near $x = x_0$, and so we expect to have small time-steps close to the shock and large time-steps outside.

12.1. **Moving nodes.** To make things a little more interesting, we will allow some more freedom for the discretization, namely *moving nodes*, i.e. nodes with time-dependent positions. We may think of this as following particle paths or streamlines. The idea is that if we move the nodes in the direction of the convection, the resulting PDE (ODE) is essentially the heat equation, so if we can remove the convective term by moving the nodes, we can solve a simpler equation.

To formulate the multi-adaptive method for PDEs with moving nodes, consider the problem

$$\dot{u} + A(u)u = f(x,t), \text{ in } \Omega \times (0,T], \tag{44}$$

for some linear differential operator $A(u)$ on some domain $\Omega$. We consider now the multi-adaptive moving-nodes method with piecewise linears in space, for which the Ansatz is

$$U(x,t) = \sum_{i=1}^{N} \xi_i(t) \varphi_i(x,t), \tag{45}$$

where the $\{\xi_i\}_{i=1}^N$ are functions of time, and $\{\varphi_i(\cdot,t)\}_{i=1}^N$ are piecewise linear basis functions in space for any fixed $t$. This is explained in Figure 63. We could also think of adding and deleting nodes as we go along, but we keep the number of nodes is fixed here.

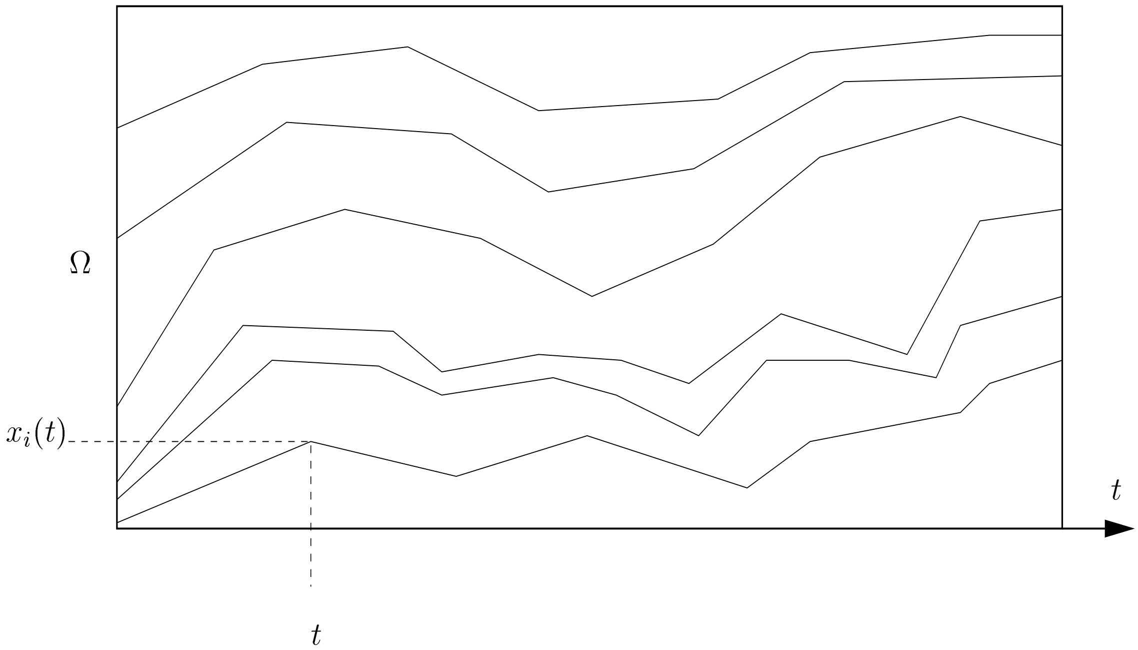



**Figure 63:** Moving nodes with piecewise linear node position functions for $\Omega = (0, 1)$.

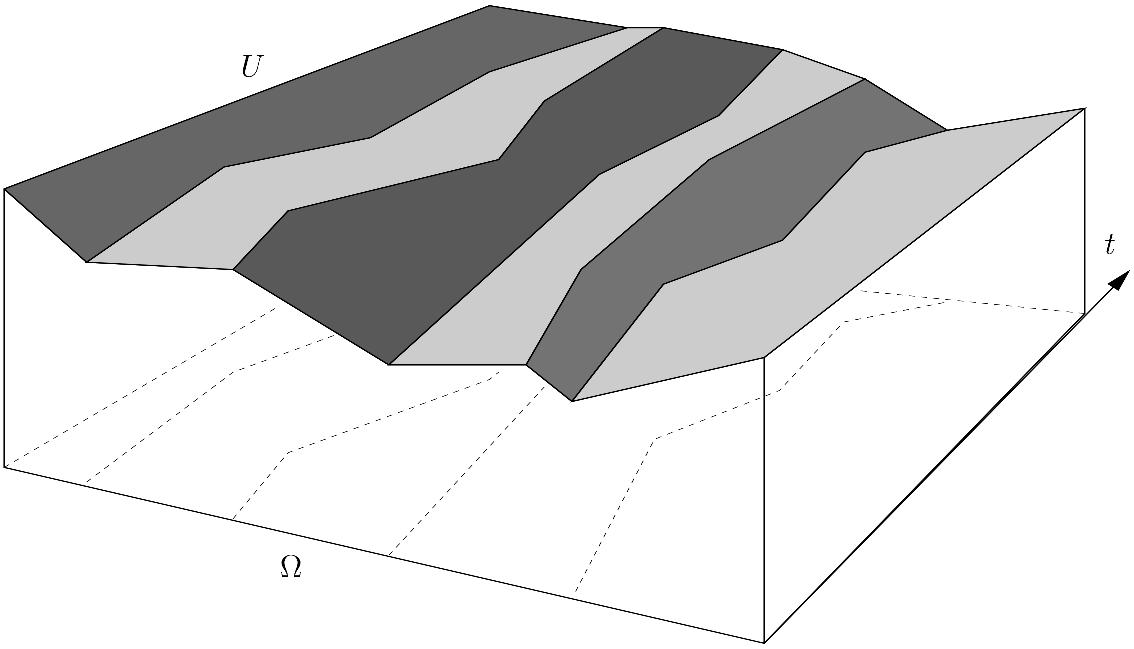



**Figure 64:** A multi-adaptive solution with moving nodes. The solution is piecewise linear in space at every fixed $t$ and piecewise polynomial (in this case piecewise linear) along the particle paths.

Inserting the Ansatz into (44) and testing with the basis functions gives, for $i = 1, \ldots, N$,

$$\int_\Omega \frac{d}{dt}\left(\sum_{j=1}^N \xi_j \varphi_j\right) \varphi_i \, dx + \int_\Omega \left(A(U)\sum_{j=1}^N \xi_j \varphi_j\right) \varphi_i \, dx = \int_\Omega f \varphi_i \, dx, \tag{46}$$

which should hold on $(0, T]$. Rearranging, we get

$$\sum_{j=1}^N \dot{\xi}_j \int_\Omega \varphi_i \varphi_j \, dx + \sum_{j=1}^N \xi_j \left[\int_\Omega \varphi_i A(U) \varphi_j \, dx + \int_\Omega \varphi_i \dot{\varphi}_j \, dx\right] = \int_\Omega f \varphi_i \, dx, \tag{47}$$

or, finally,

$$M(t)\dot{\xi}(t) + (S(t) + T(t))\xi(t) = b(t), \tag{48}$$

where

$$M(t) = \left(\int_\Omega \varphi_i(x,t)\varphi_j(x,t) \, dx\right) \tag{49}$$

is the *mass matrix*,

$$S(t) = \left(\int_\Omega \varphi_i(x,t)(A(U(x,t))\varphi_j(x,t)) \, dx\right) \tag{50}$$

is the *stiffness matrix*[2],

$$T(t) = \left(\int_\Omega \varphi_i(x,t)\dot{\varphi}_j(x,t) \, dx\right) \tag{51}$$

is the *translation matrix* and $b(t) = \left(\int_\Omega f(x,t)\varphi_i \, dx\right)$ is the right-hand side. This equation is now in the form of (1), except for the mass matrix, which we may choose to treat in different ways, the simplest being to replace it with the diagonal lumped mass matrix. We may thus apply the multi-adaptive method to (48) to have individual time-steps with moving nodes. Such a solution is at every fixed $t$ piecewise linear in space, and piecewise polynomial of degree according to the method along the particle paths, see Figure 64.

For this method to work well, it is crucial that we can compute the element-interactions, i.e. the mass, stiffness and translation matrices on the fly. This is simple to do in one dimension, where we can write down explicit formulae for the matrix elements. As an example we take the translation matrix. This is a tridiagonal matrix with elements

$$\left\{ \begin{array}{rcl} t_{ii} & = & 1/6 \ (\dot{x}_{i+1} - \dot{x}_{i-1}), \\ t_{i,i-1} & = & 1/6 \ (\dot{x}_{i-1} + 2\dot{x}_i), \\ t_{i,i+1} & = & 1/6 \ (-\dot{x}_{i+1} - 2\dot{x}_i), \end{array} \right. \tag{52}$$

for homogeneous Dirichlet boundary conditions, where $x_i(t)$ is the position of node $i$ at time $t$.

This method has been implemented for one-dimensional problems, and has been successfully applied to Burger's equation and a number of linear convection-diffusion problems.

[2]The expression for the stiffness matrix is to be interpreted in a suitable way. If e.g. $A(U) = -\Delta$, we integrate by parts to place one of the derivatives on the test function.

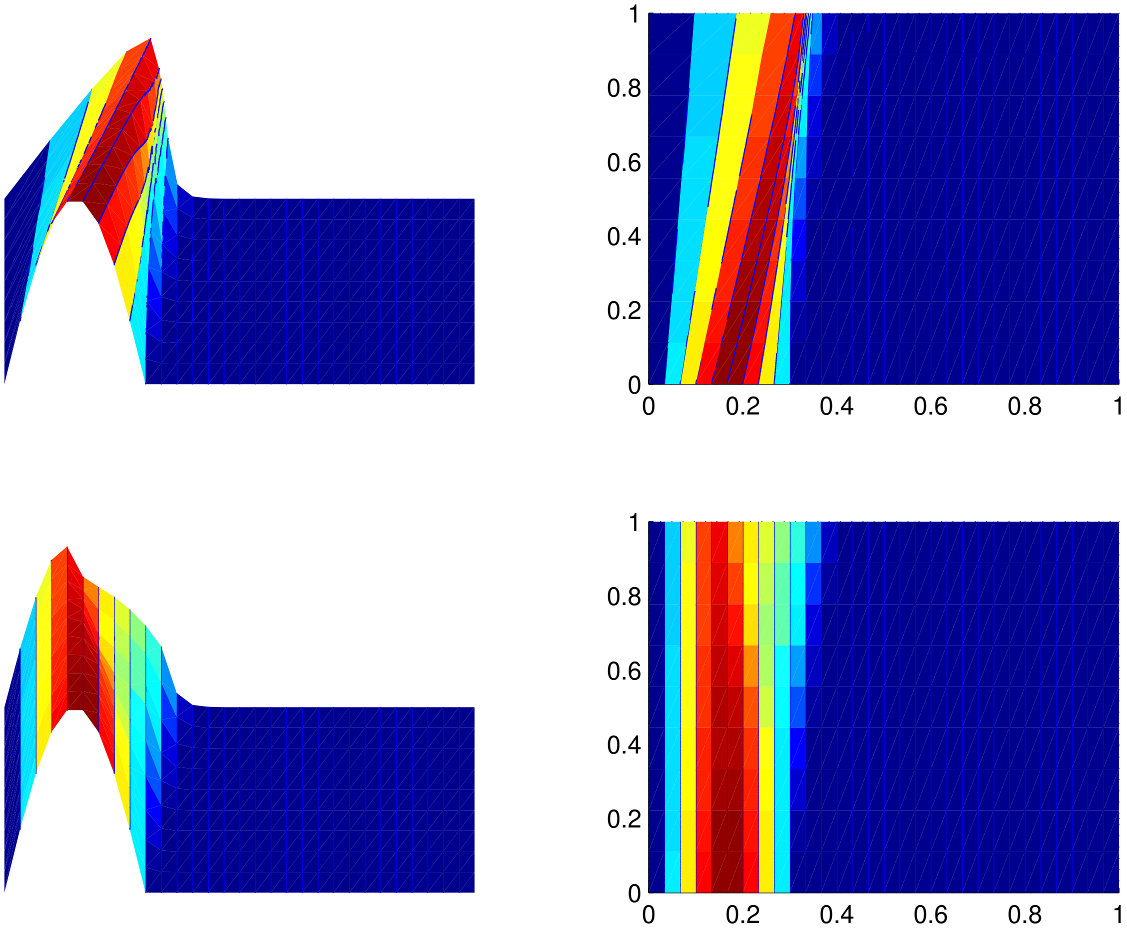



**Figure 65:** Above the solution as function of space and time, and below the solution as function of the local coordinates and time, i.e. the same plot as above but keeping the nodes at their original positions in the plot.

For convection-diffusion problems, notice that if we choose the particle paths correctly, i.e. in the direction of the streamlines, we get the desired set of equations along these streamlines. To see this, consider the problem

$$\dot{u}(x,t) + b(x,t)u'(x,t) - \epsilon u''(x,t) = f(x,t) \tag{53}$$

on $(0,1) \times (0,T]$, for which the contribution from the convection,

$$\int_0^1 b(x,t)\varphi_j'(x,t)\varphi_i(x,t)\, dx, \tag{54}$$

is balanced by the corresponding term from the translation matrix, so that the result is that the convection effectively disappears from the equation. To verify this, consider the basis function $\varphi_i$ on $(x_{i-1}, x_i)$, where $\varphi_i(x,t) = (x - x_{i-1}(t))/(x_i(t) - x_{i-1}(t))$, so that

$$\dot{\varphi}_i(x,t) = -\frac{1}{h_i}\left[\dot{x}_{i-1}\frac{x_i - x}{h_i} + \dot{x}_i\frac{x - x_{i-1}}{h_i}\right] = -\frac{1}{h_i(t)}\bar{\dot{x}}(x,t), \tag{55}$$

where $h_i(t) = x_i(t) - x_{i-1}(t)$ and $\bar{\dot{x}}(x,t)$ is a measure of $\dot{x}_i$ in between nodes. To have balance between the convection and the terms from the translation of the basis functions, $b(x,t)\varphi_j'(x,t)$ should balance $-\dot{\varphi}_j(x,t)$, which is thus the case if $b(x,t) = \bar{\dot{x}}(x,t)$. Thus, choosing the particle paths in accordance with the field $b$, the convection and the translation parts will (almost) cancel out.

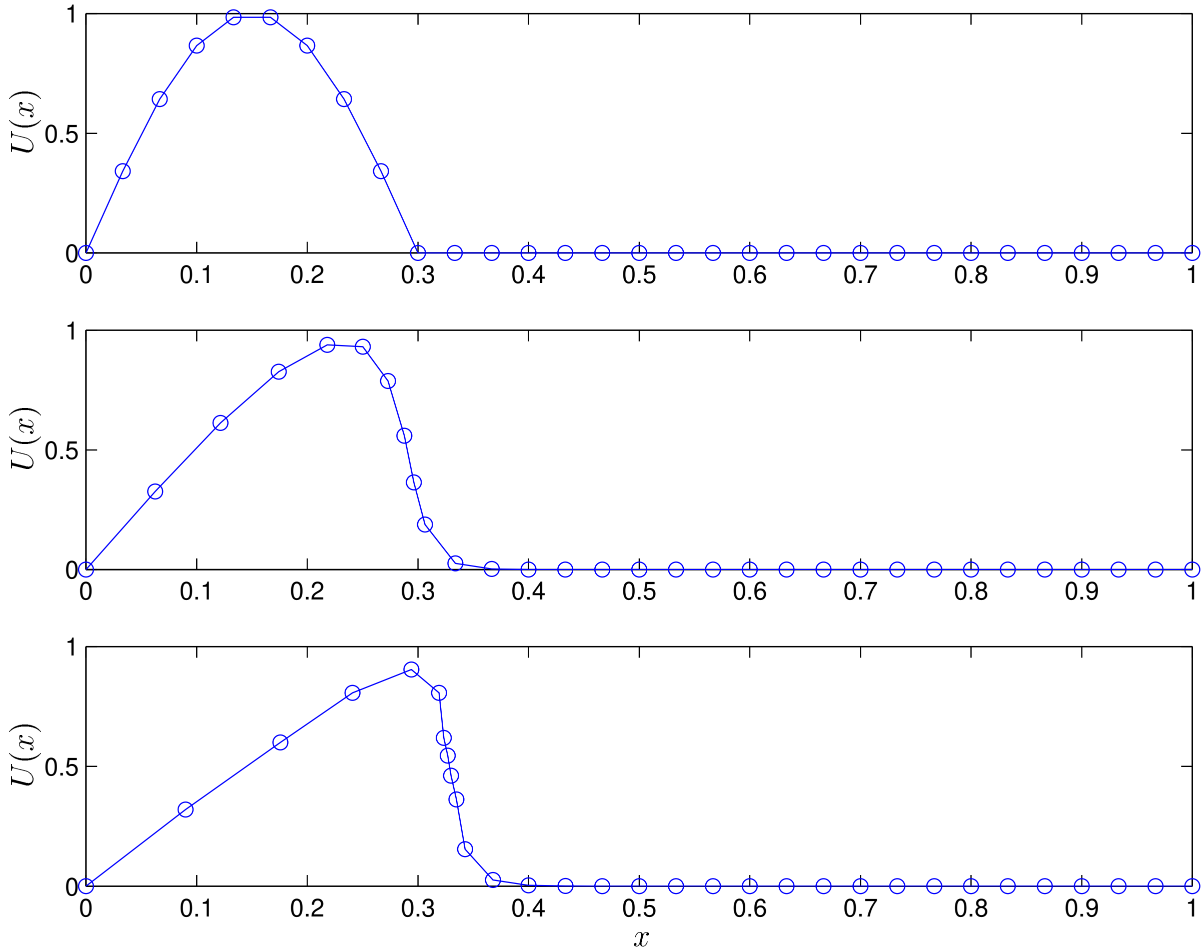



**Figure 66:** The solution as function of space at $t = 0$, $t = T/2$ and $t = T$.

12.2. **The solution.** Solving now with the mdG(0) method to $T = 1$, we get the solution presented in Figure 65.

The solution is obtained by starting with a uniform grid at $t = 0$, dividing the interval $[0, 1]$ into 30 subintervals. The nodes are then moved in the direction of the convection, i.e. $(\mu u, 1)$, taking care the node paths do not intersect. In this way, the resulting ODE will contain very little of the convection, being dominated by the diffusion term, so that we in fact are back to solving the heat equation. This is seen in Figure 65, where we together with the solution also plot the solution expressed in the original coordinates, i.e. the coordinates are put back to their original position.

As is evident in Figure 66, the nodes move in the direction of the convection into the front, to automatically give a better resolution of the front.

In Figure 67, we plot the node paths. Not only do the nodes move into the front to provide a better resolution, but the time-steps are also larger outside the front where nothing happens.

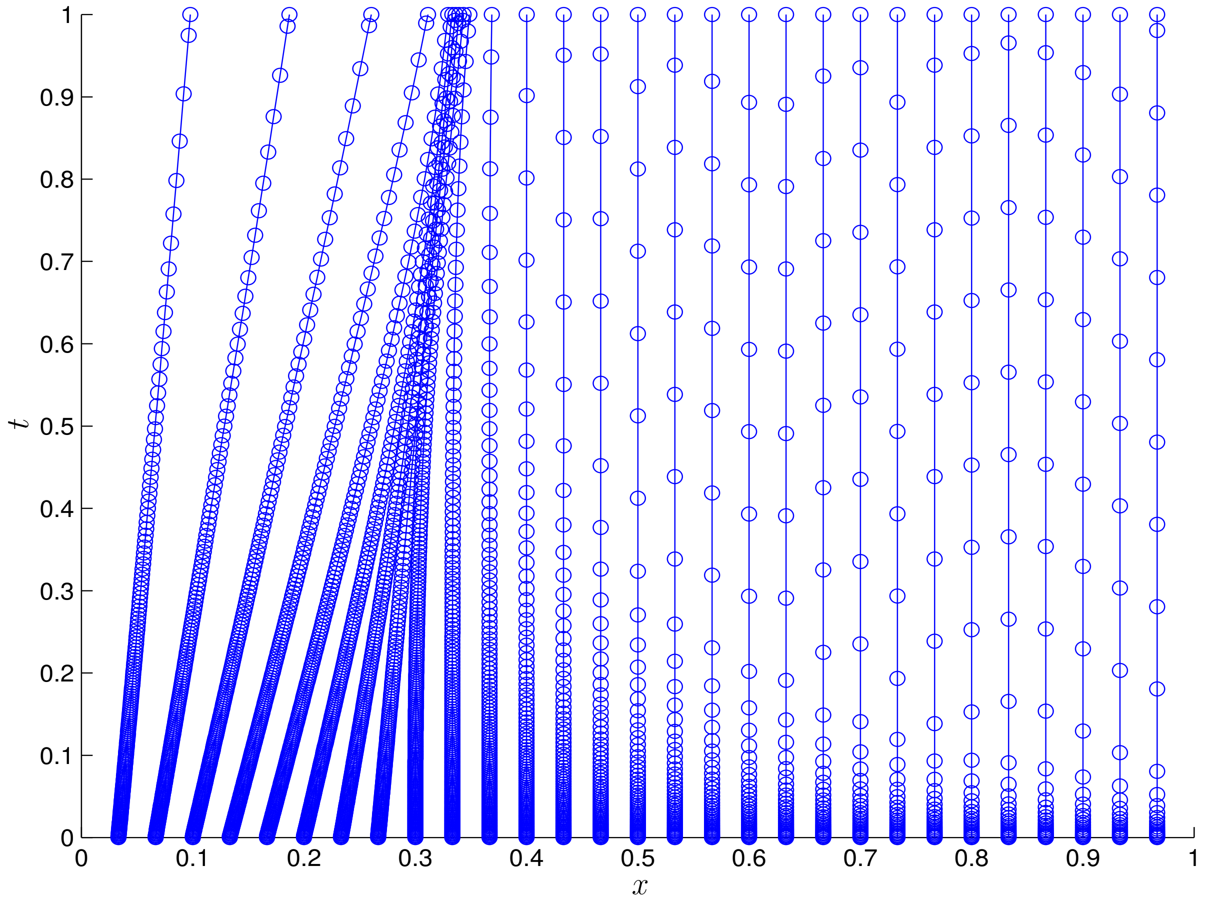



**Figure 67:** Node paths for the multi-adaptive moving-nodes solution of Burger's equation.

## 13. A Convection-Diffusion Problem with Moving Boundary

A related problem to the Burger's equation is the following convection-diffusion problem,

$$\dot{u} + \beta u' - \epsilon u'' = f, \tag{56}$$

where we take

$$f = 1 \text{ for } |x - 0.5| < 0.1 \text{ and } t < 1, \tag{57}$$

and zero elsewhere. The difference from Burger's equation is that the convection, $\beta = \beta(x,t)$ is given, and we take

$$\beta(x,t) = 0.1 \sin t. \tag{58}$$

We solve for $T = 4$ and $\epsilon = 0.001$ using the mdG(0) method. To make things a little more interesting we also have a time-dependent domain, letting also the boundary nodes follow the flow.

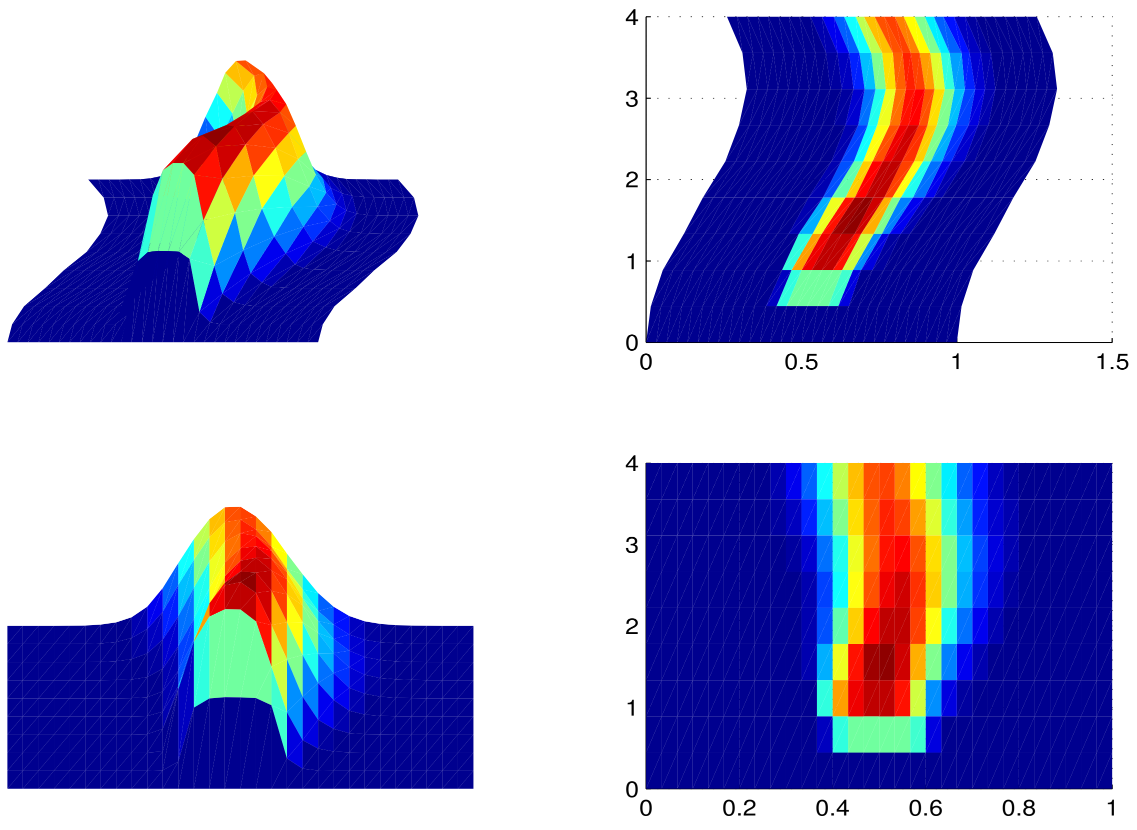



**Figure 68:** Above the solution as function of space and time, below the solution as function of the local coordinates and time, i.e. the same plot as above but keeping the nodes at the original position in the plot.

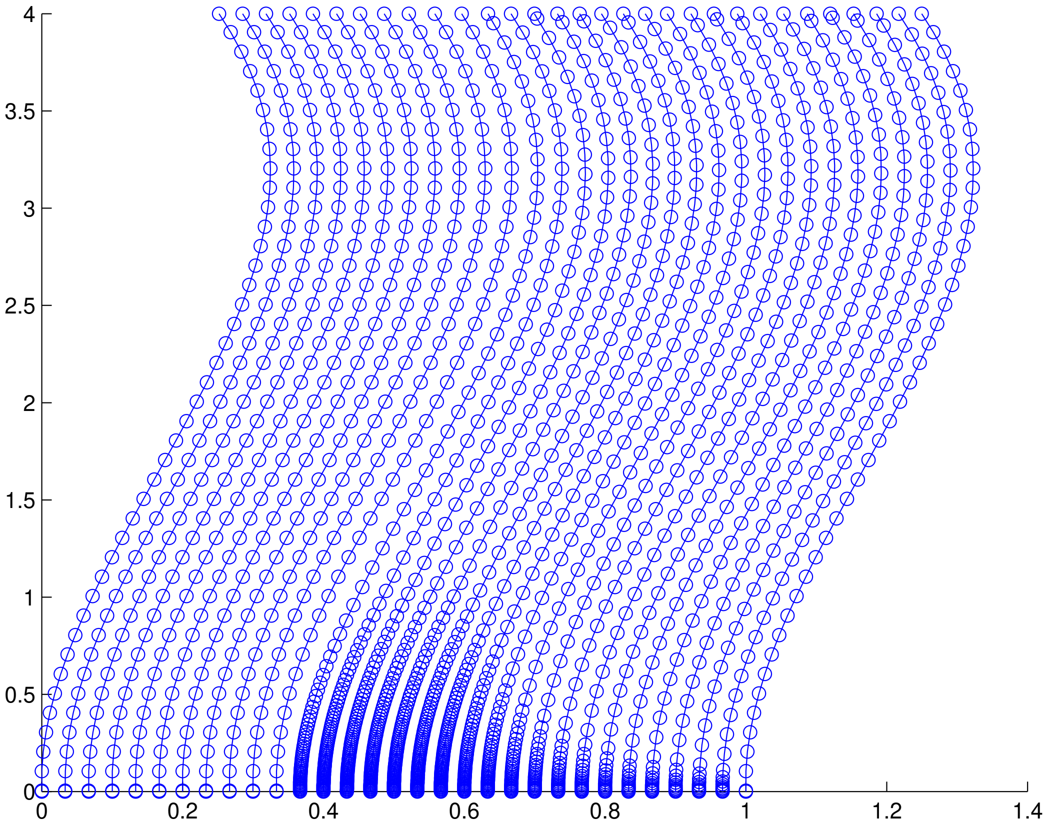



**Figure 69:** Node paths. Notice the localization in space and time of the smaller time-steps.

As for Burger's equation, moving the nodes transforms the problem into something more like the heat equation, which is evident in Figure 68, where we also plot the solution with the nodes moved back to their original positions.

Looking at Figure 69 showing the node-paths for the solution, we see that the small time-steps are clearly localized to the beginning of the time interval, and also localized to nearby $x = 0.5$. Noteworthy is that the time-steps are also somewhat smaller in the region to the right of $x = 0.5$. This is natural, since the convection is directed towards the right close to $t = 0$.

## 14. Future Work

Together with the companion paper [7] (and [6, 5]), this paper serves as a starting point for further investigation of the multi-adaptive Galerkin methods and their properties.

Future work will include a more extensive investigation of the computability of the Solar System, and application of the multi-adaptive methods to DAEs and large systems of chemical reactions. We will also consider in more detail the extension of the multi-adaptive methods to time-dependent PDEs.

## References

[1] *Usno astronomical applications department*, `http://aa.usno.navy.mil/`.

[2] D. Estep and C. Johnson, *The computability of the lorenz system*, Deparment of Mathematics, Chalmers, Preprint, 1994–33 (1994).

[3] C. Johnson and A. Logg, *Explicit time-stepping for stiff odes*, in preparation.

[4] W. Lioen and H. de Swart, *Test set for initial value problem solvers*, CWI Report MAS–R9832, ISSN 1386–3703, Amsterdam, (1998).

[5] 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).

[6] ——, *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).

[7] ——, *Multi-adaptive galerkin methods for odes i: Theory & algorithms*, Chalmers Finite Element Center Preprint 2001–09, available from `http://www.phi.chalmers.se/preprints/`, (2001).

[8] ——, *Tanganyika, version 1.2*, `http://www.phi.chalmers.se/tanganyika/`, (2001).

[9] 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


**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

**2001–05** *A Note on Energy Conservation for Hamiltonian Systems Using Continuous Time Finite Elements*
Peter Hansbo

**2001–06** *Stationary Level Set Method for Modelling Sharp Interfaces in Groundwater Flow*
Nahidh Sharif and Nils-Erik Wiberg

**2001–07** *Integration methods for the calculation of the magnetostatic field due to coils*
Marzia Fontana

**2001–08** *Adaptive finite element computation of 3D magnetostatic problems in potential formulation*
Marzia Fontana

**2001–09** *Multi-Adaptive Galerkin Methods for ODEs I: Theory & Algorithms*
Anders Logg

**2001–10** *Multi-Adaptive Galerkin Methods for ODEs II: Applications*
Anders Logg

These preprints can be obtained from

`www.phi.chalmers.se/preprints`