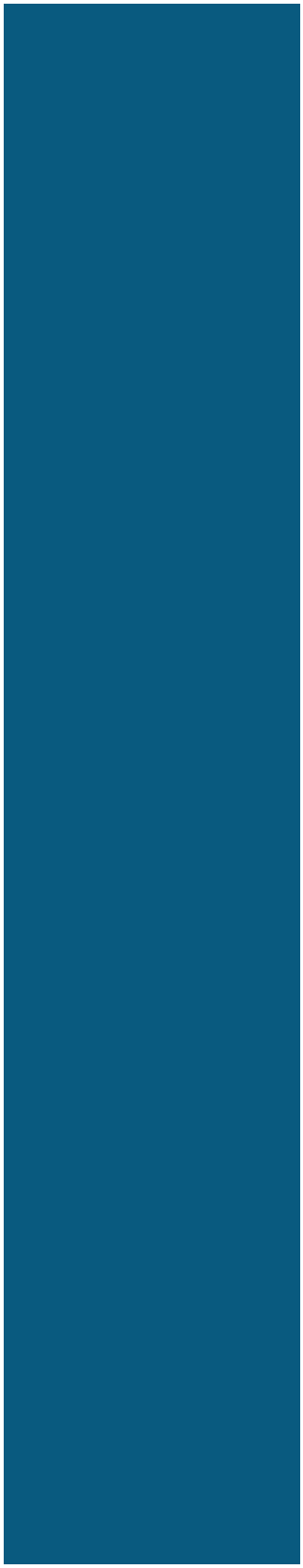

© Anders Logg 1998

The front page shows three different discretizations for a multidimensional system of equations. The first one is a non-adaptive discretization, the second one is an adaptive discretization and the third one is a *multi-adaptive* discretization.

This document was generated with LaTeX on Solaris 2.6 at `dd.chalmers.se`. The font is Palatino 10pt.

This report, as well as the code implementing the method proposed in it, are available for download at `http://www.dd.chalmers.se/~f95logg/Tanganyika/`.

Computations have been made with the Tanganyika multi-adaptive ODE-solver library (available for download at `http://www.dd.chalmers.se/~f95logg/Tanganyika/`) on Linux 2.0 (Intel Pentium 200MHz) and on Solaris 2.6 (Sun Ultra 1 Model 170).

Göteborg, October 3, 1998

Abstract

In this work I present a *multi-adaptive* finite element method for initial value problems for ordinary differential equations, including an a posteriori estimate of the error.

The method is multi-adaptive in the sense that the resolution of the time discretization is chosen individually for each component of the system of ordinary differential equations, based on an estimation of the error.

The method has been successfully implemented in the *Tanganyika* library, available for download. Included are a few example computations made with this library, as well as instructions for downloading and using the package.

Acknowledgements

I wish to thank

- **Claes Johnson**, my advisor, for his continuous encouragement and support in the making of this project;
- **Rickard Lind**, **Mathias Brossard** and **Andreas Brinck** for beta-testing the programs;
- **Jim Tilander** for his expertise help with C++;
- **Greger Cronquist** for some useful hints on the typesetting;
- **Göran Christiansson** for proof-reading the manuscript;
- **Anna** for letting me do this all summer.

Contents

1 Introduction **7**
- 1.1 Quantitative Error Control . . . 8
- 1.2 Multi-Adaptivity . . . 9

2 The Method – Multi-Adaptive Galerkin **10**
- 2.1 Equation . . . 10
- 2.2 Finite Element Formulation . . . 11
  - 2.2.1 Details . . . 12
  - 2.2.2 Even more Flexibility . . . 13
- 2.3 Error Estimation . . . 13
  - 2.3.1 The Constant C_q . . . 14
  - 2.3.2 A Correction of the Error Estimate . . . 15
  - 2.3.3 Other Error Contributions . . . 16
- 2.4 Adaptivity . . . 16
  - 2.4.1 Moderating the Choice of Timesteps . . . 18
  - 2.4.2 Choosing Data for the Dual Problem . . . 18

3 The Implementation – Tanganyika **19**
- 3.1 Individual Stepping . . . 19
  - 3.1.1 Organization, Book-Keeping . . . 20
- 3.2 Quadrature . . . 20

3.3 The Program . . . 21
3.3.1 Language . . . 21
3.3.2 Modularity . . . 21

4 Results **22**
4.1 A First Simple Example . . . 22
4.2 Wave Propagation in an Elastic Medium . . . 24
4.3 Gravitation . . . 26
4.4 The Lorenz System . . . 29
4.5 True Error vs. the Error Estimate . . . 31

5 Conclusion **35**

6 Download **36**

Bibliography **37**

Appendix **39**

A Notation **39**

B Tanganyika User Manual **43**
B.1 Introduction . . . 5
B.2 Download . . . 5
B.3 Installation . . . 5
B.4 The Tanganyika Library . . . 8
B.4.1 What it does . . . 8
B.4.2 How to use it . . . 8
B.4.3 `InitializeSolution()` . . . 9
B.4.4 `ClearSolution()` . . . 12
B.4.5 `Solve()` . . . 12
B.4.6 `Save()` . . . 12
B.5 The Tanganyika X-interface, Antananarive . . . 12
B.5.1 Introduction - What is this program anyway? . . . 12
B.5.2 Using the program - Step by Step . . . 13
B.5.3 Settings . . . 14
B.5.4 Options . . . 14
B.5.5 `.xt`-file syntax . . . 14
B.5.6 Download, Updates, Further Information . . . 16
B.6 GNU General Public License . . . 17

CHAPTER 1

Introduction

Numerical methods for solving initial value problems for ordinary differential equations have been around for a long time and the number of methods is almost as large as the number of equations.

Common methods, such as the ones supplied with Matlab (`ode45()`, `ode23()`, `ode113()`, `ode-whatever()`), are often fast, meaning that they terminate in a short time.

These methods often provide some sort of local error control, where the error is controlled in some way in each integration step. This, however does not mean control of the global error. Although a tolerance is specified, it is not related – otherwise than by some (hopefully) monotonically increasing, and otherwise unknown, function – to the global error of the solution. The program is thus not concerned with the actual value of the error, leaving the user unaware of the quality of the computed solution.

> *In fact, it was wrong.*
> Bill Clinton (1998).

Using such a classical numerical solver usually means solving the problem at a number of different tolerance levels for the local error, and comparisons between these solutions. Error control is thus (perhaps) obtained manually.

This manual effort should also be taken into account when comparing the efficiencies of different solvers.

1.1 Quantitative Error Control

Using a posteriori estimates of the error, i.e. error estimates based on the computed solution, it is possible to accurately control the size of the global error.

Finite elements present a general framework for solving differential equations, such as e.g. initial value problems for ordinary differential equations, considered in this report. Depending on the choice of basis functions, normally piecewise polynomials of different kinds, the result is a new step method for solving the initial value problem. These methods include $\mathrm{cG}(1)$, $\mathrm{cG}(2)$,..., $\mathrm{dG}(0)$, $\mathrm{dG}(1)$,....

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

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

Work on quantitative error-control during the last ten years (see references [1]-[10]) has resulted not only in extensive theoretical results, but also in working implementations of the methods, such as e.g. CARDS (solver of initial value problems for ordinary differential equations – see [8]) and FEMLAB (solver of partial differential equations).

The current approach to quantitative error control was originated with the article by Johnson ([9]) in 1988, discussing error estimation for the $\mathrm{dG}(0)$ and $\mathrm{dG}(1)$ methods. Error estimation for these methods are further discussed by Estep in [5]. The $\mathrm{cG}(q)$ method, which is the basis for the multi-adaptive method presented in this report, is discussed at length in [7]. A more classical approach to error analysis can be found in [11].

A comprehensive and major article on adaptive methods for differential equations is [3]. A general and non-technical discussion on error control and adaptivity is [6].

1.2 Multi-Adaptivity

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

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

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

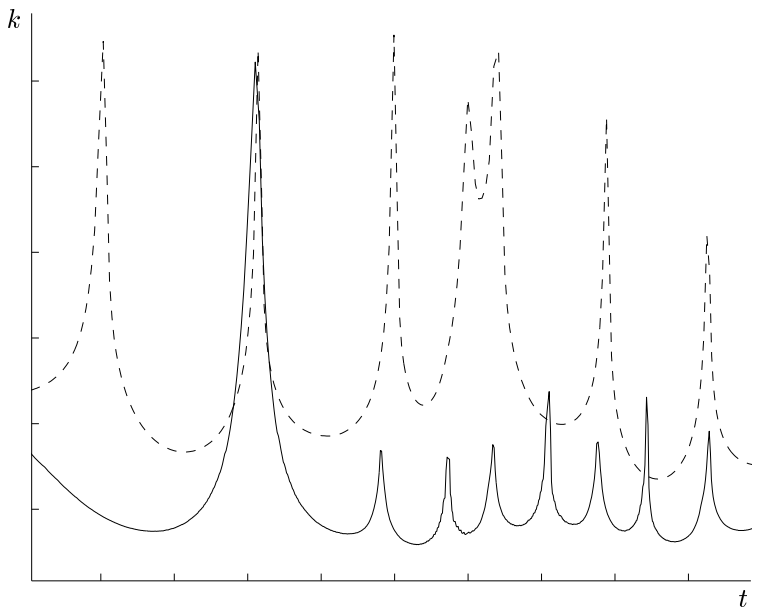

Figure 1.1: These are the actual timesteps used for an example computation on a simple two-dimensional system.

CHAPTER 2

The Method – Multi-Adaptive Galerkin

This chapter describes the multi-adaptive method, complete with an a posteriori error estimate.

The basis for the multi-adaptive method is a generalization of the continuous Galerkin method, $\mathrm{cG}(q)$, described in e.g. [4].

2.1 Equation

The equation to be solved is

$$\begin{cases} \frac{du}{dt}(t) &= f(u,t),\ t \in (0,T], \\ u(0) &= u_0, \end{cases} \tag{2.1}$$

where $f = (f_1, \ldots, f_N)$ is some function[1] depending on the solution $u = (u_1, \ldots, u_N)$ and t, which may represent time.

[1] In order to guarantee the existence of a unique solution, it may be good to know that f is Lipschitz continuous.

2.2 Finite Element Formulation

The weak (variational) formulation of equation (2.1) reads

Find $u = u(t)$ such that $u(0) = u_0$ and

$$\int_0^T (\dot{u}, v) = \int_0^T (f, v) \quad \text{for all test functions } v, \tag{2.2}$$

where $(\cdot, \cdot)$ denotes the usual l^2-inner product.

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

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

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

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

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

$$\int_0^T (\dot{U}, v) = \int_0^T (f, v) \quad \forall v \in W_k^N. \tag{2.3}$$

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

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

$$\int_{I_{ij}} \dot{U}_i v = \int_{I_{ij}} f_i v \quad \forall v \in \mathcal{P}^{q_i - 1}(I_{ij}),\ \ j = 1, \ldots, M_i, i = 1, \ldots, N, \tag{2.4}$$

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

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

Finding the parameters $\{\xi_{ijk}\}$ in agreement with eq. (2.4) yields the desired solution. What remains is to find the proper discretization, $\{I_{ij}\}$, i.e. the *timesteps* $\{k_{ij}\}$. To choose the timesteps, we need an error estimate, which will be the basis for adaptivity. By means of this error estimate, the discretization will be chosen in a way to give a resulting final error smaller than the specified tolerance.

2.2.1 Details

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

$$\lambda_{ijk}(t) = \frac{(t - t_{ij0}) \cdots (t - t_{ij,k-1})(t - t_{ij,k+1}) \cdots (t - t_{ijq_i})}{(t_{ijk} - t_{ij0}) \cdots (t_{ijk} - t_{ij,k-1})(t_{ijk} - t_{ij,k+1}) \cdots (t_{ijk} - t_{ijq_i})}. \quad (2.5)$$

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

$$U_i = \sum_{k=1}^{q_i} \xi_{ijk} \lambda_{ijk}, \quad (2.6)$$

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

Inserting this into eq. (2.4), computing a few integrals (simple but tedious) and solving the resulting system of linear algebraic equations, yields

$$\begin{cases} \xi_{ij1} &= \xi_{ij0} + \int_{ij} w_{q_i 1}(\tau_{ij}(t)) f_i(U, t) dt, \\ \xi_{ij2} &= \xi_{ij0} + \int_{ij} w_{q_i 2}(\tau_{ij}(t)) f_i(U, t) dt, \\ \quad \vdots & \\ \xi_{ijq_i} &= \xi_{ij0} + \int_{ij} w_{q_i q_i}(\tau_{ij}(t)) f_i(U, t) dt, \end{cases} \quad (2.7)$$

where $\tau_{ij}(t) = \frac{t - t_{i,j-1}}{t_{ij} - t_{i,j-1}}$ and the $\{w_{qk}\}_{k=1}^{q}$ are polynomial weight functions. These are given in table 2.1 for $q = 1, 2, 3$.

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

Table 2.1: Weight functions for the cG(q) integrals, $q = 1, 2, 3$.

2.2.2 Even more Flexibility

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

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

2.3 Error Estimation

The error estimate is obtained starting the same way as in references [1], [4] and [10].

To estimate the error at final time T in the l^2-norm, the *dual* problem of eq. (2.1) is introduced. The dual problem is

$$\left\{ \begin{array}{rcl} -\frac{d\varphi}{dt}(t) & = & J^*(u,U,t)\varphi(t),\ t \in [0,T), \\ \varphi(T) & = & e(T)/||e(T)||, \end{array} \right. \tag{2.8}$$

where $e = U - u$ is the error, $||\cdot||$ is the l^2-norm and J^* is defined as

$$J^*(u,U,\cdot) = \left(\int_0^1 \frac{\partial f}{\partial u}(su + (1-s)U, \cdot)ds \right)^*, \tag{2.9}$$

i.e. J^* is the transpose (or more generally, the adjoint) of the Jacobian of f at a mean value of u and U.

Note now that by the chain rule,

$$\begin{array}{rcl} -J(u,U,\cdot)(U-u) & = & \int_0^1 \frac{\partial f}{\partial u}(su + (1-s)U, \cdot)ds(u-U) \\ & = & \int_0^1 \frac{\partial f}{\partial s}(su + (1-s)U, \cdot)ds \\ & = & f(u,\cdot) - f(U,\cdot). \end{array} \tag{2.10}$$

We may thus write

$$
\begin{aligned}
||e(T)|| &= (e(T), e(T))/||e(T)|| \\
&= (e(T), \varphi(T)) \\
&= (e(T), \varphi(T)) - (e(0), \varphi(0)) + \int_0^T (e, -\dot{\varphi} - J^*(u, U, \cdot)\varphi) \\
&= [(e(t), \varphi(t))]_0^T - \int_0^T (e, \dot{\varphi}) - \int_0^T (e, J^*(u, U, \cdot)\varphi) \\
&= \int_0^T (\dot{e}, \varphi) - \int_0^T (J(u, U, \cdot)e, \varphi) \\
&= \int_0^T (\dot{e} - J(u, U, \cdot)e, \varphi) \\
&= \int_0^T (\dot{U} - f(u, \cdot) - J(u, U, \cdot)(U - u), \varphi) \\
&= \int_0^T (\dot{U} - f(U, \cdot), \varphi) \\
&= -\int_0^T (R, \varphi),
\end{aligned}
\tag{2.11}
$$

where R is the residual, i.e.

$$
R = f(U, \cdot) - \dot{U}. \tag{2.12}
$$

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

$$
\begin{aligned}
||e(T)|| &= -\int_0^T (R, \varphi - \overline{\varphi}) \\
&= -\sum_{i=1}^N \int_0^T R_i(\varphi_i - \overline{\varphi}_i) \\
&= -\sum_{i=1}^N \sum_{j=1}^{M_i} \int_{I_{ij}} R_i(\varphi_i - \overline{\varphi}_i) \\
&\leq \sum_{i=1}^N \sum_{j=1}^{M_i} \sup_{I_{ij}} |R_i| \int_{I_{ij}} |\varphi_i - \overline{\varphi}_i| \\
&\leq \sum_{i=1}^N \sum_{j=1}^{M_i} C_{q_i} k_{ij}^{q_i} \sup_{I_{ij}} |R_i| \int_{I_{ij}} |\varphi_i^{(q_i)}|,
\end{aligned}
\tag{2.13}
$$

where the $\{C_{q_i}\}$ are constants.

2.3.1 The Constant C_q

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

$$
C_q = \frac{1}{q! 2^{q-1}}. \tag{2.14}
$$

The proof is simple. Noting that, with

$$
\overline{f}_{q-1}(x) = f(x_0) + f'(x_0)(x - x_0) + \ldots + \frac{1}{(q-1)!} f^{(q-1)}(x_0)(x - x_0)^{(q-1)} \tag{2.15}
$$

we have

$$|f(x) - \overline{f}_{q-1}(x)| = |\frac{1}{(q-1)!}\int_{x_0}^{x} f^{(q)}(y)(y-x_0)^{(q-1)}dy| \tag{2.16}$$

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

$$\begin{aligned}
\int_a^b |f - \overline{f}_{q-1}| &= \tfrac{1}{(q-1)!}\int_a^b |\int_{x_0}^{x} f^{(q)}(y)(y-x_0)^{(q-1)}dy|dx \\
&\leq \tfrac{1}{(q-1)!}\left(\int_a^b |x-x_0|^{q-1}dx\right)\left(\int_a^b |f^{(q)}(x)|dx\right) \\
&= \tfrac{1}{q!}\left(|a-x_0|^q + |b-x_0|^q\right)\int_a^b |f^{(q)}| \\
&= \tfrac{2(b-a)^q}{q!2^q}\int_a^b |f^{(q)}| \\
&= \tfrac{(b-a)^q}{q!2^{q-1}}\int_a^b |f^{(q)}|.
\end{aligned} \tag{2.17}$$

Another useful estimate (see the section on adaptivity below) is

$$\int_a^b |f - \overline{f}_{q-1}| \leq \tilde{C}_q (b-a)^{q+1} \sup_{I_{ij}} |f^{(q)}|, \tag{2.18}$$

where

$$\tilde{C}_q = \frac{1}{(q+1)!2^q}, \tag{2.19}$$

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

2.3.2 A Correction of the Error Estimate

The method to be used is not, because of the difficulty involved with solving eq. (2.7), the true multi-adaptive $\mathrm{cG}(q)$ method, as will be described further in chapter 3.

Not solving the equations properly will introduce the *discrete residual*, which should be zero if the discrete equations, i.e. (2.7), were solved properly. The following analysis will result in an extra term in the error estimate (2.13), including the discrete residual together with its proper stability factor, accounting for accumulation of errors due to a non-zero discrete residual.

Defining the discrete residual to be

$$\overline{R}_{ij} = \overline{R}_i(t) = \int_{I_{ij}} f_i - (\xi_{ijq} - \xi_{ij0}),\ j = 1, \ldots, M_i,\ i = 1, \ldots, N, \tag{2.20}$$

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

$$\int_{I_{ij}} R_i \overline{\varphi}_i = \overline{\varphi}_i(\eta_{ij}) \int_{I_{ij}} (\dot{U}_i - f_i) = -\overline{\varphi}_i(\eta_{ij}) \overline{R}_{ij}. \tag{2.21}$$

Thus, $\int_{I_{ij}} R_i \overline{\varphi}_i$ differs from zero and we get an additional term in our error estimate, continuing from eq (2.11):

$$\begin{aligned}
||e(T)|| &= -\int_0^T (R, \varphi) \\
&= -\sum_{i=1}^N \int_0^T R_i \varphi_i \\
&= -\sum_{i=1}^N \sum_{j=1}^{M_i} \left[\int_{I_{ij}} R_i \varphi_i - \int_{I_{ij}} R_i \overline{\varphi}_i - \overline{\varphi}_i(\eta_{ij}) \overline{R}_{ij} \right] \\
&= -\sum_{i=1}^N \sum_{j=1}^{M_i} \left[\int_{I_{ij}} R_i (\varphi_i - \overline{\varphi}_i) - \overline{\varphi}_i(\eta_{ij}) \overline{R}_{ij} \right] \\
&\leq \sum_{i=1}^N \sum_{j=1}^{M_i} \left[C_{q_i} k_{ij}^{q_i} \sup_{I_{ij}} |R_i| \int_{I_{ij}} |\varphi_i^{(q_i)}| + |\overline{R}_{ij}| \sup_{I_{ij}} |\overline{\varphi}_i| \right] \\
&\approx \sum_{i=1}^N \sum_{j=1}^{M_i} \left[C_{q_i} k_{ij}^{q_i} \sup_{I_{ij}} |R_i| \int_{I_{ij}} |\varphi_i^{(q_i)}| + |\overline{R}_{ij}| \sup_{I_{ij}} |\varphi_i| \right],
\end{aligned} \tag{2.22}$$

if we choose $\overline{\varphi}$ close to φ.

2.3.3 Other Error Contributions

Other error contributions that are not dealt with here are quadrature errors and numerical errors due the finite precision arithmetic.

2.4 Adaptivity

Introducing the *stability function*, defined by

$$s_i(t) = s_{ij} = \sup_{I_{ij}} |\varphi_i^{(q_i)}|,\ t \in I_{ij},\ j = 1, \ldots, M_i,\ i = 1, \ldots, N \tag{2.23}$$

and the *stability factor*, defined by

$$S_i(T) = \int_0^T |\varphi_i^{(q_i)}|, \ i = 1, \ldots, N, \tag{2.24}$$

the error estimate (2.13) may be written in two alternative ways as

$$\begin{array}{rcl} ||e(T)|| & \leq & \sum_{i=1}^N \sum_{j=1}^{M_i} \tilde{C}_{q_i} s_{ij} k_{ij}^{q_i+1} \sup_{I_{ij}} |R_i|, \\ ||e(T)|| & \leq & \sum_{i=1}^N C_{q_i} S_i \sup_{(0,T)} (k_i^{q_i} |R_i|). \end{array} \tag{2.25}$$

The stability properties are obtained by numerical approximation (by the multi-adaptive $\mathrm{cG}(q)$ method) of the solution of the dual problem.

Notice that the error contribution from the non-zero discrete residual is not included in these expressions, since I have chosen to base the adaptivity on the Galerkin discretizational error alone. However, the contribution from the non-zero discrete residual is of course included in the computation of the error estimate and thus, indirectly, also in the adaptive procedure.

Adaptivity is then based on the expression

$$||e(T)|| \leq \text{error estimate} = \mathrm{TOL}, \tag{2.26}$$

where TOL is a given tolerance for the error of the solution at time $t = T$.

The discretization is now chosen by *equidistribution* of the error, both onto the different components and onto the different intervals, i.e.

$$\tilde{C}_{q_i} s_{ij} k_{ij}^{q_i+1} \sup_{I_{ij}} |R_i| = \frac{\mathrm{TOL}}{NMi}. \tag{2.27}$$

Alternatively, we may whish to do

$$C_{q_i} S_i \sup_{(0,T)} (k_i^{q_i} |R_i|) = \frac{\mathrm{TOL}}{N}. \tag{2.28}$$

Knowing thus the residuals and the stability functions (or factors) we may choose the proper timesteps. This is done in a way that is iterative in two respects. Firstly, the timestep for an interval is chosen based on the residual in the previous interval. Secondly, the $\{M_i\}$, are not known until the end of the computation. The values $\{M_i\}$ are then a more or less clever guess based on a previous computation. Of course, having computed the solution, we don't have to guess these values to compute an error estimate.

2.4.1 Moderating the Choice of Timesteps

Choosing timesteps as described in the previous section without any extra moderation may cause problems. If the residual in one interval is small, the timestep of the next interval will be large. A large timestep will (often) result in a large residual, which in turn in the same way means the timestep of the next interval will be small. There is thus a chance the timestep will oscillate if it is only based on the residual of the last interval. What needs to be done is to make sure the timesteps (and thus also the residuals) don't differ too much between adjacent intervals. This may be done in a lot of different ways, e.g. by choosing the (harmonic) mean of the previous timestep and the value of the new timestep, as based on the residual. (The Tanganyika library uses a somewhat more sophisticated moderation of the timesteps.)

2.4.2 Choosing Data for the Dual Problem

According to eq. (2.8), we need to know the true error in order to solve the dual problem. If we indeed knew the true error, we would not have to bother with any of this, and since the true error is unknown, we have to make a clever guess. We now discover another benefit of multi-adaptivity – it makes it easier for us to estimate the data for the dual problem! Since we equidistribute the error onto the different components, an estimation of the proper data for the dual problem should be $\pm 1/\sqrt{N}$, N being the dimension, for the different components. The signs for the different components may be obtained by solving at different tolerance levels.

Since, however, we don't know the stability properties of the problem until the computation is done, we cannot expect the errors of an initial computation to be fully equidistributed onto the different components. Hence, we cannot expect $\pm 1/\sqrt{N}$ to always work as data for the components of the dual problem. Again, proper data is obtained by e.g. solving at different tolerance levels.

CHAPTER 3

The Implementation – Tanganyika

This section describes the actual implementation of the method described in the previous section.

3.1 Individual Stepping

The individual stepping is done according to eq. (2.7). This requires knowledge about U, including the values of all other components. These values are evaluated by interpolation (or *extra*polation), according to the order of the method, of the nearest known values of the other components. The solution of the integral equation is done iteratively for every component.

The order of the stepping follows one simple principle;

the last component steps first.

It is the fact that the equations are not solved simultaneously that results in non-zero discrete residuals.

3.1.1 Organization, Book-Keeping

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

The individual stepping is done according to figure 3.1 below. The implementation pretty much follows this scetch.

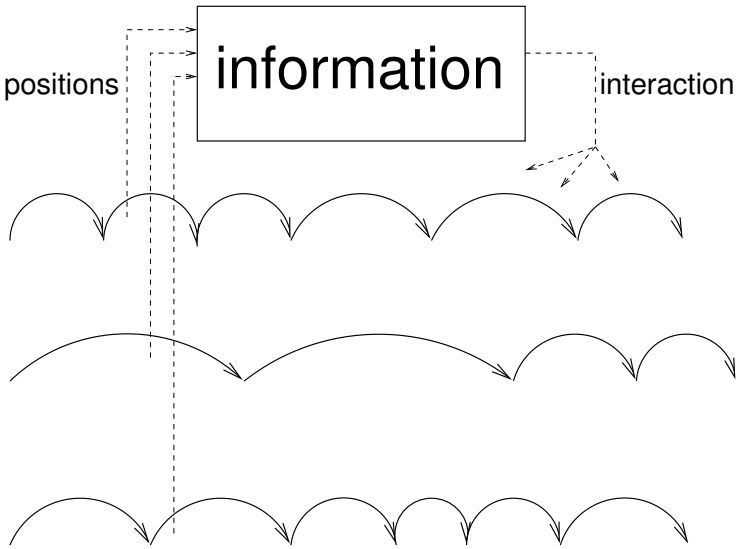

Figure 3.1: This is how the individual stepping is done. The different components tell/send their respective positions and in turn they get their interactions with (forces from) the other components. Thus, just as in nature itself, progress is made by the exchange of information, small pieces of information (gravitons or perhaps *femions*).

3.2 Quadrature

The integrals of eq. (2.7) are evaluated by Gaussian (Gauss-Legendre) quadrature.

Since the order of the weight functions for the integrals of a $\mathrm{cG}(q)$ method are $(q-1)$, we expect the total order of the integrands to be of order $q+(q-1)=$

$2q - 1$ (and even more if f is of quadratic or higher order). It would thus be wise to use quadrature that is exact at least for polynomials of order $2q - 1$, which is exactly the case for Gaussian quadrature with q nodal points.

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

3.3 The Program

The method has been implemented as a library, called *Tanganyika*. To use the library functions, all one needs to do is to

```
#include <tanganyika.h>
```

in one's C/C++ program. For more details, refer to the *Tanganyika User Manual* included in Appendix B. For *even more* details (all!) download the source code – see chapter 6.

3.3.1 Language

The language of the Tanganyika library is C++, although its interface is pure C. An object-oriented programming language such as C++ is obviously well-suited for such a program like the Tanganyika library, viewing the different objects as classes; Solution, Component, Element, etc.

3.3.2 Modularity

A nice feature of the C++ programming language is the use of class derivation and inheritance, enabling a modular implementation of the different methods. Implemented in the current version (1.0) of the library are $\mathrm{cG}(1)$, $\mathrm{cG}(2)$ and $\mathrm{cG}(3)$, but the implementation of another method, such as e.g. $\mathrm{dG}(0)$, would require only the implementation of a new subclass, specifying only what differs from the already existing methods. (This would in reality mean perhaps 50 lines of code.)