Analysis and Implementation of Adaptive Explicit Two-Step Methods

Kjartan Kari Gardasson Myrdal

Master's thesis
2015:E51

Lund University

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

CENTRUM SCIENTIARUM MATHEMATICARUM

Abstract

Recently a new way of constructing variable step-size multistep methods has been proposed, that parametrizes the entire domain of multistep methods. In the presented work the case of explicit two-step methods is looked at, analyzed and related to the already known theory of multistep methods. The error coefficient is derived as a function of the step-size ratio and an upper limit to the method domain due to zero stability is found. The theory is used to introduce an implementation of a variable step-size methods from a pair of explicit two-step methods and optimal parameters then chosen empirically. The chosen method is tested on benchmark problems.

Popular scientific summary

Differential equations are a common type of mathematical problems to be encountered in many fields of study, such as a change of stock price in economics, a chemical reaction in chemistry or movement of an object in a gravitational field in physics. A differential equation is a mathematical model that describes the rate of change of a system. Knowing some starting value for the system, referred to as the initial value, the differential equation describes how the value will change.

The predator-prey equations, also known as the Lotka-Volterra equations, are a famous example that describes the change in the populations of two distinct species of animals where one is the predator and the other is the prey. The differential equation would then be the rate of change over time of the two populations as they depend on each other. If there is plenty of prey, the predator population will grow, but as it gets larger, the prey will be eaten up faster and the population will shrink. As the prey population gets smaller, the predator population will follow due to lack of food. When the predator population gets smaller again the prey population will grow again and the system is back to the original populations.

When we talk about solving a differential equation, we wish to find a new mathematical model that allows us to find the future value of a system, knowing the initial value. While such a solution would be optimal, a mathematical model of this type cannot always be found. For the cases when finding such solution is not possible, we rely on computers to estimate the solution using numerical methods. A numerical method is a computer algorithm that with an approximation forwards the solution in small iterative steps. A numerical solution however contains an error from the computation and the task of getting the best solution is to minimize the error, but at the same time keep the computational time to the minimum. For this reason there exists broad range of numerical methods as different problems have different requirements. One way of getting less error is to make the step-size of the method smaller. That however would mean more steps to reach the final destination and therefore longer computational time. As the same accuracy is not always needed throughout the entire solution, most modern methods allow the step-size to change between steps. Those methods are referred to as variable step-size methods.

One type of methods is known as multistep methods as they use the information from multiple previous steps to further the solution to the next step. A few such methods have been well known for quite some time but a recent development in the field has made it possible to construct all possible variable step-size multistep methods. In this thesis the multistep methods that use the information from the previous two steps to compute the new step, known as two-step methods, were considered. As multistep methods have already been studied in detail, the work presented here was on analyzing and generalizing the already known theory for the new way of constructing methods. Furthermore the theoretical results were used to construct and implement a variable step-size method. The successful implementation was applied to test problems for demonstration.

Contents

Chapter 1

Introduction

1.1 Differential equations and the multistep methods

Ordinary differential equations are a common type of computational problems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}), \tag{1.1}$$

with some initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. For that reason, much effort has gone into constructing numerical solvers. The same solvers can then in many cases be adapted to similar problems, such as partial differential equations and differential algebraic equations. One family of numerical solvers is referred to as *multistep methods* and are of the form

$$\alpha_k \boldsymbol{x}_n + \alpha_{k-1}\boldsymbol{x}_{n-1} + ... + \alpha_0 \boldsymbol{x}_{n-k} = h(\beta_k \dot{\boldsymbol{x}}_n + \beta_{k-1}\dot{\boldsymbol{x}}_{n-1} + ... + \beta_0 \dot{\boldsymbol{x}}_{n-k}), \tag{1.2}$$

where $\boldsymbol{x}_i$, $\dot{\boldsymbol{x}}_i$ are discrete values or estimates of (1.1), $h = t_n - t_{n-1}$ is the step-size and the α's and the β's are coefficients of the method. A multistep method is said to be a k-step method if k is the highest index such that $\alpha_k \neq 0$ and $|\alpha_0| + |\beta_0| > 0$. The methods can be divided further into two categories, where methods that have $\beta_k = 0$ are called explicit and methods where $\beta_k \neq 0$ are implicit. While explicit methods can compute solutions directly, implicit methods need to be combined with an iterative solver for non-linear equations, for example Newton's method or else be used in tandem with an explicit method in a predictor-corrector scheme.

Example 1.1 (Construction of explicit Adams method [8]). *The k-step explicit Adams methods (Adams–Bashforth methods) are amongst the well-known multistep methods. The methods can be constructed as follows. In order to find the solution at t_n, knowing the value at t_{n-1}, consider the equation*

$$\boldsymbol{x}(t_n) = \boldsymbol{x}(t_{n-1}) + \int_{t_{n-1}}^{t_n} \boldsymbol{f}(t)dt.$$

Since we don't know the solution to the integral, we wish to approximate it. Using previously approximated values $\boldsymbol{x}_i$ and $\boldsymbol{f}_i = \boldsymbol{f}(t_i, \boldsymbol{x}_i)$ where $i = n-1, .., n-k$, an interpolation polynomial can be constructed

$$\boldsymbol{p}(t) = \boldsymbol{p}(t_{n-1} + sh) = \sum_{j=0}^{k-1} (-1)^j \binom{-s}{j} \nabla^j \boldsymbol{f}_{n-1}$$

and placed into the integral instead of $\boldsymbol{f}(t)$. The approximated solution to the initial value problem becomes then

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} \boldsymbol{p}(t)dt = \boldsymbol{x}_{n-1} + h\sum_{j=0}^{k-1} \gamma_j \nabla^j \boldsymbol{f}_{n-1},$$

where

$$\gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds.$$

1.2 Existence of a solution

For an initial value problem to have a solution, there are certain properties that the function $\boldsymbol{f}(t, \boldsymbol{x})$ in eq. (1.1) must hold [1].

Definition 1.2. *Let $\boldsymbol{f}(t, \boldsymbol{x})$ be a function as in the initial value problem* (1.1), *then $\boldsymbol{f}$ is said to satisfy the* Lipschitz condition *and be* Lipschitz continuous *for the variable $\boldsymbol{x}$ on the set $\Omega \subseteq \mathbb{R} \times \mathbb{R}^n$ if*

$$||\boldsymbol{f}(t, \boldsymbol{x}_1) - \boldsymbol{f}(t, \boldsymbol{x}_2)|| \leq L||\boldsymbol{x}_1 - \boldsymbol{x}_2||,$$

holds for all $(t, \boldsymbol{x}_1), (t, \boldsymbol{x}_2) \in \Omega$ and some Lipschitz constant $L < \infty$.

The Lipschitz continuity allows us then to state the theorem commonly named after Émile Picard and Ernst Lindelöf.

Theorem 1.3. *For an initial value problem on the form* (1.1), *let $\boldsymbol{f}(t, \boldsymbol{x})$ be a continuous function and so that the Lipschitz condition holds, in a neighborhood around $(t_0, \boldsymbol{x}_0) \in \mathbb{R} \times \mathbb{R}^n$. Then there is an open interval around t_0 where there exists a unique solution to the problem.*

As existence and uniqueness of a solution is necessary for us to even consider the problem, those conditions will be assumed for all problems in the following chapters.

1.3 Varying the step-size

When solving a problem of the form (1.1) using a numerical method, the choice of a step-size h plays an important role in the solution. Choosing a too large step-size can cause the method to not converge to the solution at all. Assuming that the solution converges, the task is then to estimate the step-size needed for the required accuracy of the solution. While taking smaller step-sizes results in a smaller error, there is a trade off between accuracy and efficiency. Smaller step-sizes mean more steps in order to reach the same end value. For some problems the computation can be extremely time consuming and in some cases taking too small steps could lead to the computation not ending within a reasonable time. Commonly, the problems being solved have parts along the trajectory where the resolution needed varies. In order to achieve good accuracy the step-size needs to be selected for the parts where the most resolution is needed. On other parts of the solution this fine step-size is not needed and unnecessary computation is carried out.

One way to tackle this problem is to allow the step-size to vary along the solution. Accompanying the method with an error estimator allows computations to control the step-size needed at each iteration in order to achieve a certain accuracy. The form of the method needs to be altered to

$$x_n + \sum_{i=1}^{k} \alpha_{k-i,n} x_{n-i} = h_n \sum_{i=0}^{k} \beta_{k-i,n} \dot{x}_{n-i}. \tag{1.3}$$

Here we have normalized the method so $\alpha_{k,n} = 1$ for all n. This can be done without any loss of generality. The possibility of varying the step-size has become so widespread that by today's standard a method is considered of little use if the step-size cannot be varied.

Example 1.4. *An elliptic Kepler orbit in a plane around the origin can be described with the second order differential equation*

$$\ddot{\boldsymbol{r}} = -f(r)\hat{\boldsymbol{r}},$$

where $\hat{\boldsymbol{r}}$ denotes a unit vector in the direction of $\boldsymbol{r}$ and $f(r) = \mu r^{-2}$ is the inverse square force with a constant μ only depending on units and the source of attraction. By introducing velocity $\boldsymbol{v}$, the system can be reduced to the first order system

$$\begin{aligned}\dot{\boldsymbol{r}} &= \boldsymbol{v},\\ \dot{\boldsymbol{v}} &= -f(r)\hat{\boldsymbol{r}}\end{aligned}$$

and a solution to the trajectory can be given by

$$r = \frac{c^2/\mu}{1 + e\cos\phi},$$

where $c = |\boldsymbol{r} \times \boldsymbol{v}|$ is the angular velocity, ϕ is the angle between the semi-major axis and $\boldsymbol{r}$ and $0 < e < 1$ is the eccentricity of the orbit. Now, setting the values for a problem with a high eccentricity, $\mu = 1$, $r = 1$ and $v = 0.16$ with the velocity perpendicular to $\boldsymbol{r}$. Then $e = 0.6$ and the ellipse has a very flat shape.

We solve the problem now using explicit two-step Adams method discussed in example 1.1 with two different step-sizes $h = 0.0004$ and $h = 0.00005$. In figure 1.1 we can see that the bigger step-size manages to approximate the solution closely from the initial value at the aphelion of the orbit and all the way to the perihelion. Close to the origin at the perihelion however the speed is high and the curvature is steep so the particle reaches escape velocity and shoots off, while an elliptical trajectory was expected. The smaller step-size manages to solve for the ellipse but needs 8 times as many steps to do so. It would therefore be optimal if for part of the trajectory a bigger step-size could be used while at the perihelion the step-size could be lowered.

1.4 Parametric multistep methods

Some well known multistep methods such as explicit Adams methods discussed in example 1.1, Nyström's methods, the backwards difference formulas (BDF) and Milne-Simpson methods can be found both in the literature [8], [10], [12] and widespread

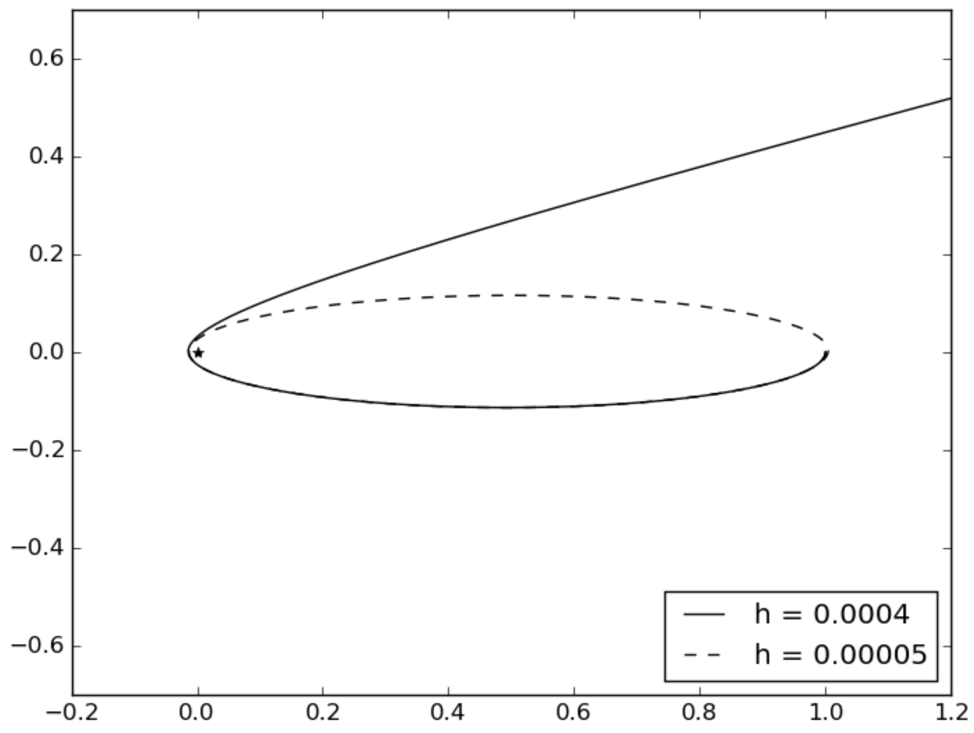

Figure 1.1: A Kepler orbit with high eccentricity $e = 0.6$ solved twice using explicit two-step Adams method with two different fixed step-sizes. With $h = 0.0004$ the computation makes it to the perihelion but then shoots of while with $h = 0.00005$ the method can solve a full orbit.

numerical software [4], [11]. All those methods are derived by means of some numerical integration with difference methods with a constant step-size. Many of them have further been extended to variable step-size methods by means of polynomial interpolation to make new grid point values for the new step-size or Nordsieck's approach [8]. In [2] Arévalo, Führer, Selva introduce a construction of implicit multistep methods of maximal order using collocation polynomials. Arévalo extends this approach in [3] using the state slack and the derivative slack conditions

$$s_j = P(t_j) - x_j$$

and

$$\dot{s}_j = \dot{P}(t_j) - \dot{x}_j.$$

along with the orthogonality condition that

$$[\cos\theta,\, \sin\theta] \cdot \begin{bmatrix} s_j \\ h_{j+1}\dot{s}_j \end{bmatrix} = 0,$$

for a polynomial $P(t)$ to hold for $j = n-k, .., n-2$. This leads to interpolation conditions for the polynomial defining explicit methods of maximal order

$$P \in \Pi_k \tag{1.4a}$$

$$P(t_{n-1}) - x_{n-1} = 0 \tag{1.4b}$$

$$\dot{P}(t_{n-1}) - \dot{x}_{n-1} = 0 \tag{1.4c}$$

$$P(t_{n-j}) - x_{n-j} + \tan\theta_j h_{n-j+1}(\dot{P}(t_{n-j}) - \dot{x}_{n-j}) = 0; \quad j = 2, \dots, k, \tag{1.4d}$$

optionally replacing some but not all (1.4d) with

$$\dot{P}(t_{n-j}) - \dot{x}_{n-j} = 0, \tag{1.4e}$$

where Π_k is the space of all polynomials of order k. The new step can be computed then as $x_n = P(t_n)$. In this thesis we will use the condition for parametric explicit methods in (1.4) to analyze all possible explicit two-step methods of order two and use that in an attempt to implement a variable step-size method combined with an error estimator. As explicit two-step methods are defined with a single parameter θ the subscript indexing will be dropped.

1.5 A good foundation

A two volume set called Solving ordinary differential equations by Hairer, Nørsett, Wanner [7], [8] has a comprehensive coverage of the theory of numerical methods for ODEs. The first volume covers explicit methods and the second one implicit methods and stiff problems. As those books thoroughly cover the theory on multistep methods, the analysis of the parametric variable step-size multistep methods in the following chapter will heavily rely on them.

In chapter 2 the theory will be discussed and the methods analyzed, in chapter 3 the code implementation of the methods will be covered along with some theory of how a step-size controller is implemented, then in chapter 4 some computed solution using the method will be displayed and the effect of different parameter values discussed and finally in chapter 5 we will summarize and discuss future research possibilities.

Chapter 2

Analysis of a method

The aim of this chapter is to analyze the family of all variable step-size explicit two-step methods. There are several conditions a method needs to satisfy in order to be valid and an attempt will be made to cover all aspects of the theory. In some cases the two-step methods will only be considered but when appropriate, the theory will be stated in general for all multistep methods.

2.1 Coefficients of an explicit two-step method

The procedure of getting from the conditions in eq. (1.4) to the coefficients of a k-step method is more or less the same, regardless of the value of k and whether or not the method is implicit or explicit. To begin with, some form of a second order polynomial $P(t)$ needs to be chosen. The form used here is

$$P(t) = \frac{a}{2}(t - t_{n-1})^2 + b(t - t_{n-1}) + c,$$

with the first derivative

$$\dot{P}(t) = a(t - t_{n-1}) + b.$$

The polynomial is placed into the conditions of eq. (1.4) and the resulting formulas used to solve for a, b and c. Once those coefficients have been found, the polynomial can be evaluated at the new step

$$P(t_n) = \frac{a}{2}{h_n}^2 + bh_n + c,$$

where $h_n = t_n - t_{n-1}$. The resulting equation will contain terms of $\boldsymbol{x}_i$'s and $\dot{\boldsymbol{x}}_i$'s. Collecting the terms for each one of the $\boldsymbol{x}_i$'s and $\dot{\boldsymbol{x}}_i$'s will give the corresponding coefficient of the multistep method as shown in eq. (1.3). For the explicit two-step method the conditions are few and the equations short but this is the only case where there's only one parameter θ. For methods with more conditions, it can become tedious to keep track of terms and signs. To avoid human error a computer algebra system software is used. For the results here, Maple 17 was chosen for that purpose. The result is four coefficients that are functions of the method parameter

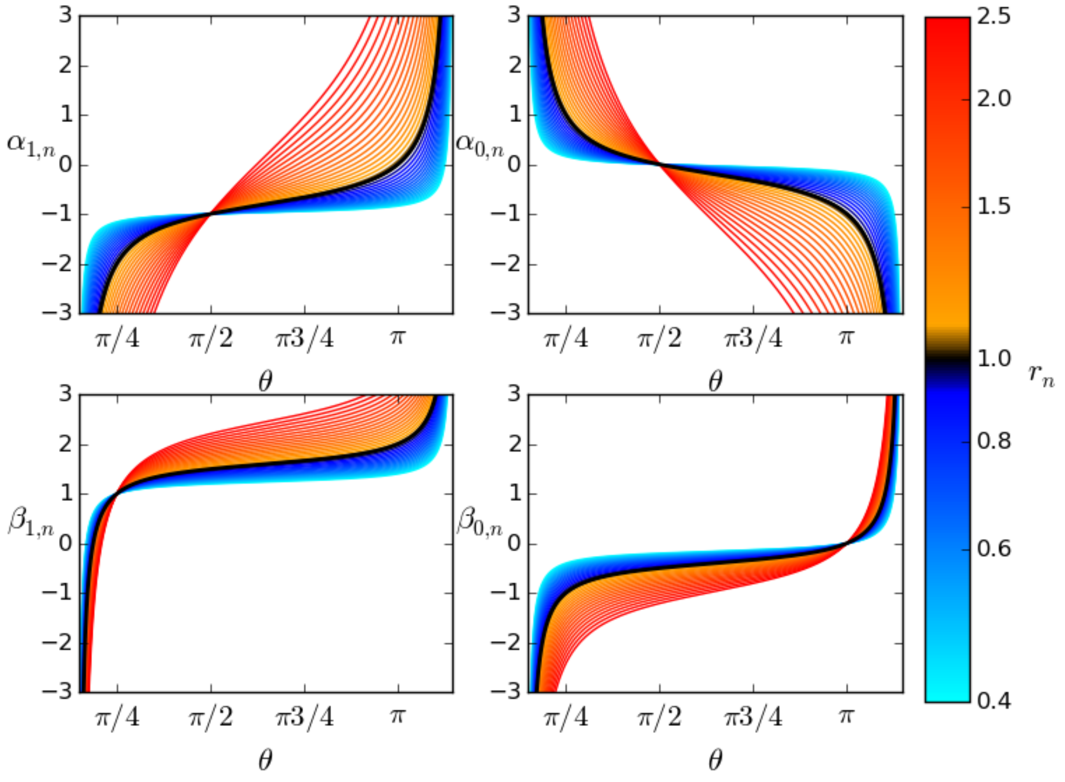

Figure 2.1: The four coefficients for the explicit two step method, $\alpha_{1,n}$, $\alpha_{0,n}$, $\beta_{1,n}$ and $\beta_{0,n}$ as functions of the parameter θ. A color scale is used to show how the coefficients vary with respect to the step-size ratio r_n and the underlying constant step-size method $r_n = 1$ is emphasized with a thicker black line.

θ and the step-size ratio $r_n = h_n/h_{n-1}$

$$\alpha_{1,n}(\theta, r_n) = \frac{{r_n}^2\cos\theta\ - \cos\theta + 2\sin\theta}{\cos\theta - 2\sin\theta} \tag{2.1a}$$

$$\alpha_{0,n}(\theta, r_n) = -\frac{{r_n}^2\cos\theta}{\cos\theta - 2\sin\theta} \tag{2.1b}$$

$$\beta_{1,n}(\theta, r_n) = \frac{r_n\cos\theta\ - r_n\sin\theta + \cos\theta - 2\sin\theta}{\cos\theta - 2\sin\theta} \tag{2.1c}$$

$$\beta_{0,n}(\theta, r_n) = \frac{r_n\sin\theta}{\cos\theta - 2\sin\theta}. \tag{2.1d}$$

Figure 2.1 contains plots of all four coefficients, demonstrating the effect of varying the step size. Since θ only occurs within trigonometric functions, all the coefficients must be periodic with period 2π. Not as obvious is that the functions have a smaller period of π. Furthermore they all have the same denominator that is equal to zero at $\arctan(\frac{1}{2}) \pm n\pi$ for all $n \in \mathbb{N}$. This leads to divergence of the coefficients since none of the numerators are equivalent to zero at those values. Knowing the period and where the coefficients go towards $\pm\infty$, it is sufficient to consider one of the open intervals $(\arctan(\frac{1}{2}) + (n-1)\pi, \arctan(\frac{1}{2}) + n\pi)$. With the smaller domain of a single interval it is worth noting that all four of the coefficients are continuous functions, both for θ and r_n. For later analysis this will be an important property.

Two polynomials that are constructed from the coefficients of a multistep method are referred to as *the generating polynomials* of a method. Those are

$$\rho(z) = \sum_{j=0}^{k} \alpha_k z^k \quad \text{and} \quad \sigma(z) = \sum_{j=0}^{k} \beta_k z^k. \tag{2.2}$$

The generating polynomials serve an important role in the analysis of a method and will be used multiple times later.

Having a well defined method domain, the first question that comes to mind is whether any of the well known methods are contained within it.

Proposition 2.1. *The methods given by equations* (2.1) *with* $\theta = \pi/2$ *is* $\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + h_{n-1}[(\frac{1}{2}r_n + 1)\boldsymbol{f}_{n-1} - \frac{1}{2}r_n\boldsymbol{f}_{n-2}]$ *and with* $\theta = \pi$ *is* $\boldsymbol{x}_n = \boldsymbol{x}_{n-2} + h_{n-1}(r_n + 1)\boldsymbol{f}_{n-1}$. *Setting* $r_n = 1$ *those methods are equivalent to two-step explicit Adams method and Nyström method respectively [8].*

2.2 Order and error coefficient

One way of determining that a variable step size multistep methods is *consistent of order p*, is to apply the method to polynomials of order $\leq p$ and check that an exact solution is obtained. Although convenient, other equivalent order conditions are needed for further analysis of the methods. For the constant step size case, (Theorem III.2.4 given by Hairer et al. [8]), there are three equivalent conditions given for a method to be of order p. We are interested in the first case and wish to show that a generalized formulation is equivalent to the polynomial condition already mentioned. First we define the local error.

Definition 2.2. The local error *of a variable multistep method is*

$$\boldsymbol{x}(t_n) - \boldsymbol{x}_n \tag{2.3}$$

where $\boldsymbol{x}(t_n)$ *is the exact solution to the eq.* (1.1) *and* $\boldsymbol{x}_n$ *is the computed solution from equation* (1.3)*, using as initial values the exact solutions* $\boldsymbol{x}(t_{n-i})$*, for* $i = 1, ..., k$.

In association with the local error we define the difference operator

$$L(\boldsymbol{x}, t_{n-k}, h_n) = \sum_{i=0}^{k} \left[\alpha_i \boldsymbol{x}(t_{n-k} + \sum_{j=1}^{j\leq i} (R_{n-k+j}^{n})^{-1} h_n) - h_n \beta_i \boldsymbol{x}'(t_{n-k} + \sum_{j=1}^{j\leq i} (R_{n-k+j}^{n})^{-1} h_n) \right], \tag{2.4}$$

with the shorthand notation $R_j^k = \prod_{i=j}^{k} r_i = \frac{h_k}{h_j}$.

Definition 2.3. *A method on the form* (1.3) *is said to be consistent of order p if* $L(\boldsymbol{x}, t_{n-k}, h_n) = \mathcal{O}(h_n^{p+1})$ *or the local error is* $\mathcal{O}(h_n^{p+1})$. *A method is called consistent if it is at least of order 1.*

The order of a method can be verified using the following theorem.

Theorem 2.4. *A variable step size multistep method is of order p, if and only if the following two equivalent conditions are satisfied:*

1. $\sum_{i=0}^{k}\alpha_i = 0$ *and* $\sum_{i=0}^{k}\alpha_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^q = q\sum_{i=0}^{k}\beta_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^{q-1}$ *for* $q = 1, ..., p$*;*
2. $Q(t_n) + \sum_{i=0}^{k-1}\alpha_i Q(t_{n-k+i}) = h_n\sum_{i=0}^{k}\beta_i\dot{Q}(t_{n-k+i})$,

for all polynomials $Q(t)$ *of degree* $\le p$*.*

Proof. Consider the difference operator $L(\boldsymbol{x}, t_{n-k}, h_n)$ given by equation (2.4). By expanding every term of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ by its Taylor series from the point t_{n-k}, we get

$$\begin{aligned} L(\boldsymbol{x}, t_{n-k}, h_n) = \sum_{i=1}^{k}\Bigg[\alpha_i\left(\sum_{q\ge 0}\frac{(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^q}{q!}{h_n}^q\boldsymbol{x}^{(q)}(t_{n-k})\right) \\ -h_n\sum_{r\ge 0}\frac{(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^r}{r!}{h_n}^r\boldsymbol{x}^{(r+1)}(t_{n-k})\Bigg]. \end{aligned} \tag{2.5}$$

Now collect by terms of $\boldsymbol{x}^{(q)}(t_{n-k})$ and get

$$\begin{aligned} L(\boldsymbol{x}, t_{n-k}, h_n) = \boldsymbol{x}(t_{n-k})\sum_{i=0}^{k}\alpha_i + \sum_{q=1}^{p}\frac{{h_n}^q}{q!}\boldsymbol{x}^{(q)}(t_{n-k}) \\ \left[\sum_{i=0}^{k}\alpha_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^q - q\sum_{i=0}^{k}\beta_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^{q-1}\right] + \mathcal{O}({h_n}^{p+1}) \end{aligned} \tag{2.6}$$

In order for the remainder to be $\mathcal{O}({h_n}^{p+1})$ the terms in the sums need to add up to zero. This proves the first condition, the second condition follows directly from replacing the function $\boldsymbol{x}(t)$ with a polynomial $Q(t)$ of degree $\le p$. □

This result takes us directly to the case of the coefficients in (2.1).

Corollary 2.5. *All methods defined by coefficients in equations* (2.1) *on the open interval* $(\arctan(\frac{1}{2}), \arctan(\frac{1}{2}) + \pi)$ *are of order two.*

The two other conditions in the theorem by Hairer et al. already mentioned, are skipped here. The reason we are interested in the first condition as it can be used to estimate the error of a method. A common way to estimate the error of a multistep method is to consider the so called *error coefficient*. Using Theorem 2.4, the error coefficient can be derived for the parametric multistep methods for any given θ and r_i's. For a method of order p we have that eq. (2.4) can be written as

$$L(\boldsymbol{x}, t_{n-k}, h_n) = C_{p+1}{h_n}^{p+1}\boldsymbol{x}^{(p+1)}(t_{n-k}) + \mathcal{O}({h_n}^{p+2})$$

where

$$C_{p+1} = \frac{1}{(p+1)!}\left[\sum_{i=0}^{k}\alpha_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^{p+1} - (p+1)\sum_{i=0}^{k}\beta_i(\sum_{j=1}^{j\le i}(R_{n-k+j}^{n})^{-1})^{p}\right]$$

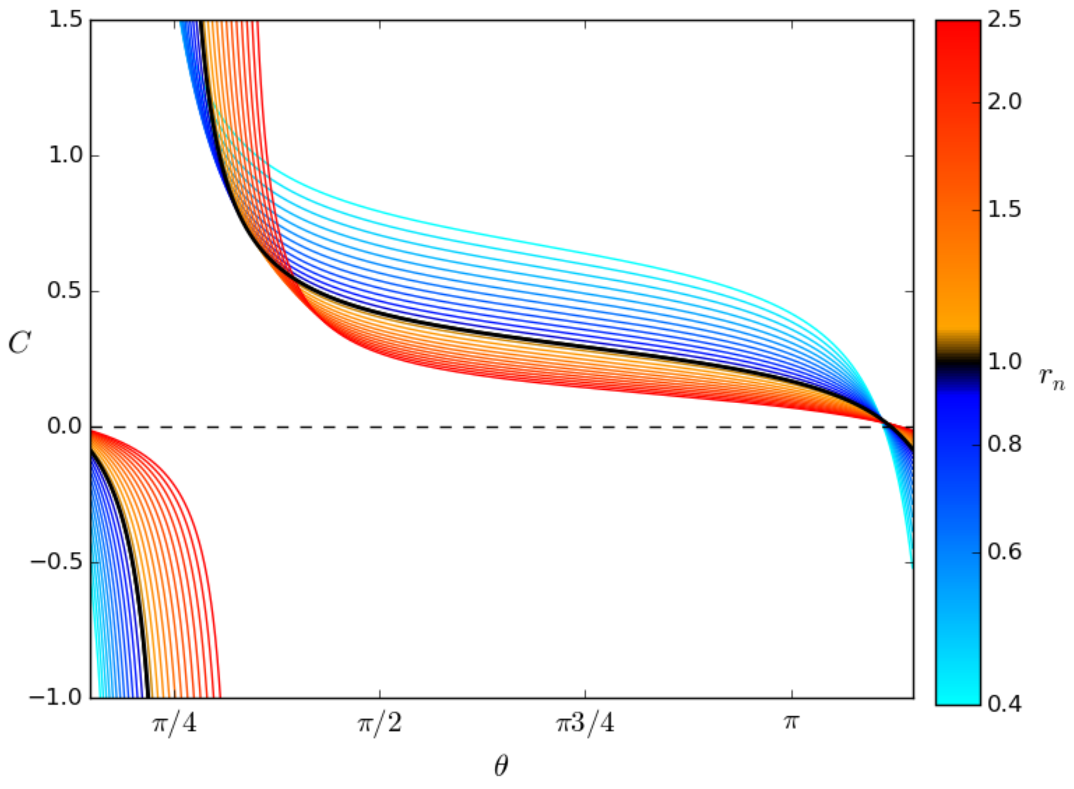

Figure 2.2: The error coefficient as a function of θ. A color map is used to demonstrate the effect of varying the step size.

A normalization for the error coefficient is given in Hairer et al. [8],

$$C = \frac{C_{p+1}}{\sigma(1)}. \tag{2.7}$$

The normalization is the same for the variable step size case. The error constant for the methods described by equations (2.1) is

$$C = \frac{1}{6}\frac{r_n\cos(\theta) - 2r_n\sin(\theta) + cos(\theta) - 3\sin(\theta)}{r_n(\cos(\theta) + \cos(\theta) - 2\sin(\theta))}. \tag{2.8}$$

This is plotted in figure 2.2 as a function of θ with the variation of r_n demonstrated with a color map.

2.3 Stability

A numerical method that is consistent is not guaranteed to converge. *The stability* of the method has to do with how well errors are damped as the computations take place. There are several different notions of stability for numerical methods for ODEs. We will cover two forms of stability, first *zero stability* that handles the boundedness of the round-off error and secondly *absolute stability* which is about rapidly decaying modes of the solution. Zero stability looks at the behavior of the method for a fixed end value t_n as $h \to 0$, while absolute stability looks at the behavior of the methods for a fixed h as $t \to \infty$.

Commonly zero stability is introduced in the literature with an example of an unstable method. This will be skipped here, but an example can be found in Hairer et al. [8] or in the original discussion by Dahlquist [5].

Definition 2.6. *A constant step size multistep method is said to satisfy the* root condition *if the roots of the generating polynomial $\rho(z)$ are on the unit disk in the complex plane and roots on the boundary are simple. A method that satisfies the root condition is said to be* zero-stable.

Since the roots of $\rho(z)$ are dependent on the step size ratio r_n, it's not enough to just consider the roots when $r_n = 1$. In order to generalize the zero stability for a variable step-size, we consider the method in a different notation. Let $X_{n-1} = (\boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-k})^T$ and the numerical solution for $\dot{\boldsymbol{x}} = 0$ on the form

$$X_n = A_{n-1}X_{n-1}, \tag{2.9}$$

where

$$A_n = \begin{pmatrix} -\alpha_{k-1,n} & \cdots & \cdots & -\alpha_{1,n} & -\alpha_{0,n} \\ 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & \vdots \\ & & 1 & 0 & 0 \\ & & & 1 & 0 \end{pmatrix}, \tag{2.10}$$

then instead of the root condition, the stability can be expressed by the boundedness of the matrices.

Definition 2.7. *A variable step size multistep method is called stable, if*

$$||A_{n+l}A_{n+l-1}\cdots A_{n+1}A_n|| \leq M \tag{2.11}$$

for all n and $l \geq 0$.

Fixing the step size ratio at $r_n = 1$ will take us back to the root condition, but with a varying step size there are many other possible solutions to eq. (2.11). In the summary given by Hairer et al. [8], there are three theorems given for a bound on the step size ratio so that the method stays stable. We consider the first two of them.

Theorem 2.8. *A method on the form* (1.3) *that satisfies the conditions that*

- *it is consistent of order $p \geq 0$,*
- *the coefficients $\alpha_{j,,n}$ are continuous for the step size ratios r_n around 1 and*
- *the corresponding constant step size method satisfies the root condition.*

Then there exists an upper and lower bounds ($\omega_n \leq 1 \leq \Omega_n$) such that the method is stable if

$$\omega_n \leq r_n \leq \Omega_n.$$

As has already been noted, all the coefficients in (2.1) are continuous with respect to r_n. In a search for some bounds on r_n we consider a second theorem. First we need the transformation

$$T = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ & 1 & 1 & \cdots & 1 \\ & & 1 & \cdots & 1 \\ & 0 & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}, \quad T^{-1} = \begin{pmatrix} 1 & -1 & & 0 & \\ & 1 & -1 & & \\ & & 1 & \ddots & \\ & 0 & & \ddots & -1 \\ & & & & 1 \end{pmatrix},$$

such that

$$T^{-1}A_nT = \left(\begin{array}{c:c} A_n^* & 0 \\ \hdashline e_{k-1}^T & 1 \end{array}\right)$$

where $e_{k-1}^T = (0, ..., 0, 1)$ and

$$A_n^* = \begin{pmatrix} -\alpha_{k-2,n}^* & \cdots & \cdots & -\alpha_{1,n}^* & -\alpha_{0,n}^* \\ 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & \vdots \\ & & 1 & 0 & 0 \\ & & & 1 & 0 \end{pmatrix}$$

with

$$\begin{aligned} \alpha_{k-2,n}^* &= 1 + \alpha_{k-1,n}, & \alpha_{0,n}^* &= -\alpha_{0,n} \\ \alpha_{k-j-1,n}^* - \alpha_{k-j,n}^* &= \alpha_{k-j,n} & &\text{for } j = 2, ..., k-1. \end{aligned}$$

Theorem 2.9. *A method of the form* (1.3) *of order* $p \geq 0$ *is zero stable if and only if for all* n *and* $l \geq 0$

1. $||A_{n+l}^* \cdots A_{n+1}^* A_n^*|| \leq M_1$
2. $||e_{k-1}^T \sum_{j=n}^{n+l} \prod_{i=n}^{j-1} A_i^*|| \leq M_2$

This way the dimension of the square matrix gets reduced by one and this theorem tells us that for a variable step size two-step explicit method we can return to the root condition, since the only thing that is left in the matrix A, n^* is $\alpha_{0,n}^* = -\alpha_{0,n}$. We see that for

$$\rho(z) = z^2 + z\frac{\cos\theta\, r^2 - \cos\theta + 2\sin\theta}{\cos\theta - 2\sin\theta} - \frac{\cos\theta\, r^2}{\cos\theta - 2\sin\theta} \tag{2.12}$$

the roots are $-\alpha_{0,n}$ and 1. According to Definition 2.6, a parametric method satisfies then the root condition if $|\alpha_{0,n}| < 1$, since the other root is already on the boundary. We consider $|\alpha_{0,n}| = 1$ and solve for r_n and get the equation $r_n = \sqrt{|1 - 2\tan\theta|}$. A corresponding plot is in figure 2.3. Since we are only interested in $r_n > 0$ and the root condition holds for all

$$0 < r_n < \sqrt{|1 - 2\tan\theta|}. \tag{2.13}$$

There is no lower limit to the step-size ratio. While technically the theory above allows for a more free upper bound, it might be hard to find one and specially for

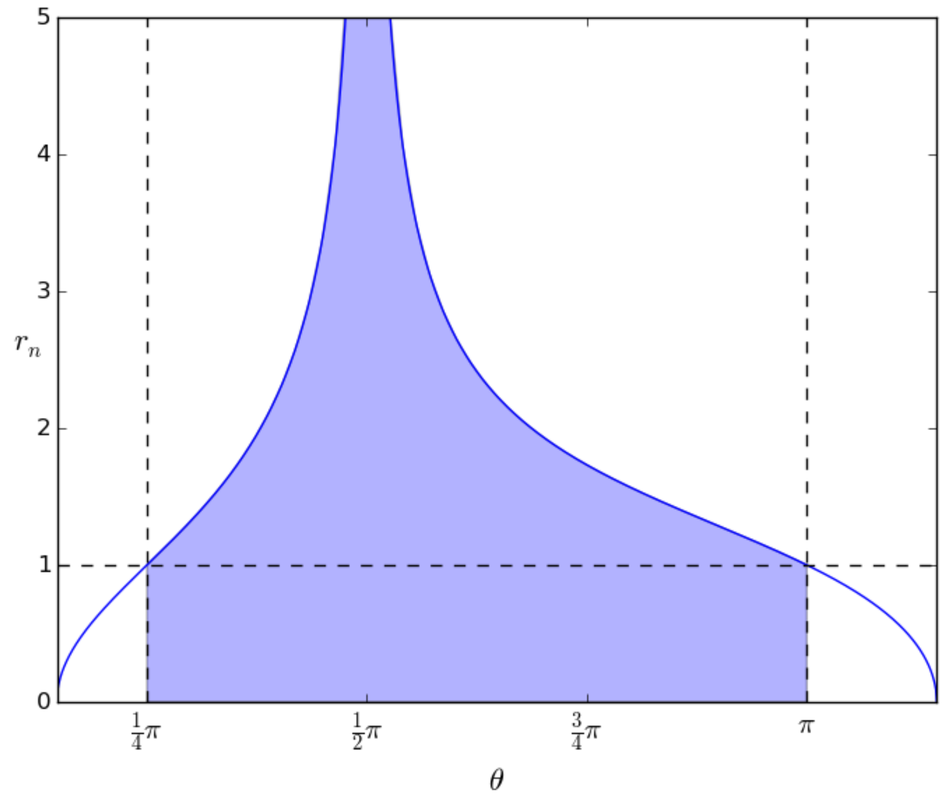

Figure 2.3: The upper limit of r_n as a function of θ to hold the root condition. The shaded blue area shows is the domain of stable methods having an underlying stable constant step size method as described by the theorems.

a generic method that is not constructed for solving only a specific problem. We define the domain of zero stable methods to be

$$Z_{r_n,\theta} = \left\{ r_n \times \theta \in \mathbb{R}^+ \times (\arctan(\frac{1}{2}), \arctan(\frac{1}{2}) + \pi)\, ;\, r_n < \sqrt{|1 - 2\tan\theta|} \right\} \quad (2.14)$$

and then $Z_{1,\theta} = (\pi/4, \pi)$ would be the interval of zero stable methods with a fixed step-size.

While zero stability handles the growth of the round-off error, there is another stability issue to consider of the growth of rapidly decaying modes of the solution of a problem. For explanation and testing of the stability of a method, the linearized and autonomous system with a Jacobian J,

$$\dot{\mathbf{x}} = J\mathbf{x} \quad (2.15)$$

is used. By change of basis to the eigenvectors of J the problem reduces to the test problem

$$\dot{\mathbf{x}} = \lambda\mathbf{x} \quad (2.16)$$

where λ is the corresponding eigenvalue with the problem stability condition that $\mathrm{Re}(\lambda) \leq 0$. Applying a multistep method and solving using Lagrange's method, by setting $y_j = z^j$ results in the characteristics equation

$$\rho(z) - \mu\sigma(z) = 0 \quad (2.17)$$

where $\mu = h\lambda$. As this difference equation is stable only if all z roots of eq. (2.17) are ≤ 1 and multiple roots strictly less than one, the method has a stability region defined by all $\mu \in \mathbb{C}$ such that these conditions are satisfied. The stability region of a method gives the boundaries within which the method will behave "properly", i.e., the solution will be bounded.

By studying the stability region of a method, we can look at the boundary of the region. By rearranging 2.17 and setting $z = e^{i\phi}$, we get

$$\mu = \frac{\rho(e^{i\phi})}{\sigma(e^{i\phi})} \tag{2.18}$$

where $\phi \in [0, 2\pi)$. If the stability region of a method is small, this means that the step size has to remain small.

We end this section by stating the first Dahlquist-barrier.

Theorem 2.10. *A stable k-step method is at most of order p where*

- $p \leq k+2$ *if k is even,*
- $p \leq k+1$ *if k is odd and*
- $p \leq k$ *if $\beta_k/\alpha_k \leq 0$.*

For an explicit method the last condition is always true. So a stable explicit method can never be of a higher order than the number of steps in it.

2.4 Convergence

The convergence is the most important condition of a method, but it relies solely on the order and stability conditions that we have already discussed. The theory of convergence for variable step-size multistep methods has been studied in detail and therefore nothing new or specific for our case that needs to be taken into consideration. We adapt a definition to the notation used here and then restate a theorem on the convergence.

Definition 2.11. *A variable step-size multistep method on the form 1.3 is called* convergent*, if for all IVP's with a unique solution as described in Theorem 1.3 on an interval $[t_0, \hat{t}]$, then*

$$\boldsymbol{x}(\hat{t}) - \boldsymbol{x}_n \to 0$$

for a step size ratio sequence $\{r_i\}_{i=0}^n$ with bounded values such that $h_1 \sum_{i=1}^n R_1^i = \hat{t}$ as $h_1 \to 0$ and $n \to \infty$. The starting values must satisfy

$$\boldsymbol{x}(t_i) - \boldsymbol{x}_i \to 0 \quad \mathit{for} \quad h_1 \to 0, \quad i = 0, 1, ..., k-1$$

Furthermore the method is called convergent of order p if there exists some positive H s.t.

$$||\boldsymbol{x}(\hat{t}) - \boldsymbol{x}_n|| \leq Ch^p \quad \mathit{for} \quad h \leq H,$$

where $h = \max h_j$ and the starting values satisfy

$$||\boldsymbol{x}(t_i) - \boldsymbol{x}_i|| \leq Ch^p \quad \mathit{for} \quad h \leq H, \quad i = 0, 1, ..., k-1.$$

Then we state a theorem from Hairer et al. [8, Theorem 5.8, p. 358]

Theorem 2.12. *For a method of the form 1.3 let*

1. *the method be stable and of order p, with bounded coefficients,*
2. *starting values be of order $\mathcal{O}({h_0}^p)$ and*
3. *the step size ratios be bounded.*

Then the method is convergent of order p.

The order of consistency for the methods defined by the coefficients in (2.1) has already been covered in Corollary 2.5. Combined with Theorem 2.12 we can then state the following.

Corollary 2.13. *Every method defined by the coefficients in* (2.1) *that is stable is also convergent.*

2.5 Boundaries of the method domain

In the beginning of this chapter the function domain for the parametric coefficients in (2.1) was limited to the interval $\theta \in (\arctan(\frac{1}{2}), \arctan(\frac{1}{2}) + \pi)$. As has been covered in the other sections, the domain of valid methods is most likely even smaller. The clearest factor would be the root condition, but methods with an underlying zero stable constant step size method ($r_n = 1$) can only be found on the interval $\theta \in (\pi/4, \pi)$. This cuts of the edge cases of the original interval and restricts the domain further. Note that for the stability condition $\theta = \pi$ can be included as a *weakly stable* case. The other endpoint $\theta = \pi/4$ would however form a double root at 1 for $\rho(z)$ and is therefore not stable.

A factor that might put into question the usability of a method is the corresponding error coefficient. While the error coefficient is only considered an estimation tool, one would want to keep it bounded or else there would be an order reduction. Considering figure 2.2 again, there is a region $\theta \in [\pi/4, 1.5]$ where the error constant diverges for different values of r_n. Although difficult to state anything about the factor of those values on the solution, this is most definitely something to keep in mind when selecting θ.

Another component to be considered is *the reducibility of a method.* A method is said to be reducible if the generating polynomials, $\rho(z)$ and $\sigma(z)$ have a common factor $\phi(z)$. The method defined by the polynomials

$$\rho^*(z) = \frac{\rho(z)}{\phi(z)}, \quad \sigma^*(z) = \frac{\sigma(z)}{\phi(z)} \tag{2.19}$$

is then equivalent to the original method. As $\sigma(z) = \beta_{1,n} z + \beta_{0,n}$, the root of $\sigma(z)$ is

$$-\frac{\beta_{0,n}}{\beta_{1,n}} = -\frac{r_n \sin\theta}{r_n(\cos\theta - \sin\theta) + \cos\theta - 2\sin\theta}.$$

Revisiting the roots of $\rho(z)$, the conditions for reducibility are

$$\frac{r_n \sin\theta}{r_n(\cos\theta - \sin\theta) + \cos\theta - 2\sin\theta} = -1$$

or

$$\frac{r_n \sin\theta}{r_n(\cos\theta - \sin\theta) + \cos\theta - 2\sin\theta} = \frac{{r_n}^2 \cos\theta}{\cos\theta - 2\sin\theta}.$$

The first equation is simple to solve and is $r = 2\tan\theta - 1$. The second one can be rewritten as

$${r_n}^2(1 - \tan\theta) + r_n(1 - 2\tan\theta) + \tan\theta(2\tan\theta - 1) = 0.$$

Solving the quadratic equation for r gives

$$r_n = \frac{(2\tan\theta - 1) \pm \sqrt{8\tan^3\theta - 8\tan^2\theta + 1}}{2(1 - \tan\theta)}.$$

For the case when $\tan\theta = 1$ there is a root when $r_n = 1$. That would be when $\theta = \pi/4$ which has already been discussed as an unstable method.

One aspect of the reducibility of the methods is to consider if one method is simply a multiplication by a constant with another method for two different θ's. Since we have already assumed that $\alpha_{2,n} = 1$ for all methods. Now lets choose arbitrarily one of the other coefficients from eq. (2.1), say $\alpha_{1,n}$. The coefficient $\alpha_{1,n}$ spans the entire $\mathbb{R}$ on the interval $\theta \in (\arctan(\frac{1}{2}), \arctan(\frac{1}{2}) + \pi)$, but knowing the value for those two coefficients the first condition in Theorem 2.4 guaranties a unique value for the rest of the coefficients. As $\alpha_{1,n}$ is strictly growing, each method is unique. This is true for all step-size ratios r_n.

Theorem 2.14 ([3]). *The coefficients in* (2.1) *construct all possible explicit variable step-size two-step multistep methods of order two.*

Example 2.15 (Corroborating the Dahlquist-barrier). *Since we have a way of constructing all two-step methods of order two, trying and see if any of them are of order three as well, would be of interest. If such a method exists and is stable that would mean that the Dahlquist-barrier was incorrect.*

A simple way to find such a method is to consider when the error coefficient in eq. (2.8) *is zero. Solving the equation for r_n gives*

$$r_n = \frac{1 - 3\tan\theta}{2\tan\theta - 1}.$$

For most values of θ the equation is negative, which is not of any interest and only a small part gives a positive value for r_n. Setting $r_n = 1$ we get $\tan\theta = \frac{2}{5}$. That method is not stable for $r_n = 1$. The Dahlquist-barrier has thus been confirmed.

Chapter 3

Implementation of a method

3.1 Code organization

For further investigation of the behavior of the two-step methods, an implementation was made in MATLAB [11] based on the theory previously covered. In the following sections we will have a look at some of the code along with explanations. Only code snippets are included in the text but the entire code can be found at `https://github.com/kjartankg/e2`. Along with the method some test cases were implemented as well. The program code was organized into the following folder and file structure.

```
e2/
 ├── +coeffs/
 │   ├── alpha0.m
 │   ├── alpha1.m
 │   ├── beta0.m
 │   └── beta1.m
 ├── +err/
 │   ├── error_constant.m
 │   ├── error_estimate.m
 │   ├── step_control.m
 │   └── upper_limit.m
 ├── +problems/
 │   ├── logistic_curve.m
 │   ├── logistic_exact.m
 │   ├── lotka_volterra.m
 │   ├── lotka_volterra_inv.m
 │   ├── neg_exp.m
 │   ├── neg_exp_exact.m
 │   ├── oscillatory.m
 │   ├── oscillatory_exact.m
 │   ├── riccati.m
 │   ├── riccati_exact.m
 │   └── vdPol.m
 ├── data/
 ├── gelmm.m
 └── init_two_step.m
```

```
│
├── script_const_step_mge.m
├── script_lotka.m
├── script_var.m
├── script_var_var_c.m
├── script_vdPol.m
├── stepper_const_step.m
├── stepper_var_step_const_c.m
└── stepper_var_step_var_c.m
```

The function files can be divided into four categories: *scripts* where the computation is specified depending on a specific problem and what method should be used, *steppers* that carry out the iteration of the method, *other functions* that the method depends on such as computing the coefficients and error handling functions and then finally there is the *problems* used for testing. In the following sections we will have a closer look at those categories. It should be noted that the program is designed for this project specifically and is not packaged as a computational library for reuse. Furthermore the code is optimized for readability rather than speed and efficiency.

3.2 Executing a computation

A computation is carried out by executing one of the scripts with the appropriate variables that define the problem to be solved. An example of variables could be

```
problem   = @problems.oscillatory;
exact_sol = @problems.oscillatory_exact;
x_start   = 1;
t_start   = 0;
t_end     = 15;
tol       = 1e-5;
```

A different script exists for each one of the different tests. The scrips are responsible for calling the corresponding stepper, that carries out the iteration of the numerical solution. In chapter 4 the corresponding script to each section will be mentioned.

Three different steppers were implemented for different purposes. All of them take as input a function `f`, initial value `x_0`, time start value `t_0`, a time end value `t_n` and the parameter `theta` that defines the method coefficients used. The simplest stepper iterates with constant step-sizes using a for loop. On top on the already mentioned inputs it takes the number of steps `n`. The other two steppers are very similar to one another since they both are variable step-size iterators using the same method to estimate the error. The only difference is that one uses always the same error coefficient, as it would be for the underlying constant step-size method, while the other one updates the error coefficient every iteration depending on the step-size ratio r_n. Those steppers have a second method parameter `eta`, that defines the error comparison method.

```
function [data] = stepper_var_step_const_c(f, x_0, ...
                                 t_0, t_n, tol, theta, eta)
```

The constant step-size stepper returns a single vector containing the computed value for each iteration while the other steppers have multiple return values packed into a single variable called `data`.

```
    data.x_all   = x_all(1:step,:);
    data.dx_all  = dx_all(1:step,:);
    data.t_all   = t_all(1:step);
    data.h_all   = h_all(1:step);
    data.r_all   = r_all(1:step);
    data.est_err_all = est_err_all(1:step);
```

3.3 Initializing the multistep method

Since problems are usually only presented with an initial value for a single point and a multistep method needs more starting values, some other method is needed to compute the first few steps. Methods for starting up multistep methods is a study of its own and outside the scope of this thesis. For that reason and for simplicity, the built-in MATLAB solver `ode45` is used to start all computations. This solver uses an explicit Runge-Kutta formula based on the Dormand-Prince pair [6].

```
function ys = init_two_step(f, y_0, t_0, h)
    [t, y] = ode45(f,[t_0, t_0 + h], y_0);
    ys     = [reshape(y(end, :), [], 1) y_0];
end
```

The initial values are passed to the built-in function and then a vector of two first values are passed back to be used to start a two-step method.

3.4 Evaluating the next step

In order to evaluate the next step, the coefficients of the method are needed. Constant step-size methods can be initialized once before the computation starts but variable step-size methods need the coefficients to be reevaluated for each step with respect to the step-size ratio.

```
        % Calculate the coefficients for the given step.
        neg_alphas     = - [coeffs.alpha1(r, theta); ...
                            coeffs.alpha0(r, theta)];
        betas          = [coeffs.beta1(r, theta); ...
                          coeffs.beta0(r, theta)];

        neg_alphas_err = - [coeffs.alpha1(r, eta); ...
                            coeffs.alpha0(r, eta)];
        betas_err      = [coeffs.beta1(r, eta); ...
                          coeffs.beta0(r, eta)];
```

Each coefficient has a separate function to be computed. As an example, here is the function for α_1.

```
function a = alpha1(r, theta)
    a = ((r^2 - 1) * cos(theta) + 2 * sin(theta)) / ...
        (cos(theta) - 2 * sin(theta));
end
```

Having obtained the coefficients, the solutions and derivatives, the next step can be evaluated using a general explicit linear multistep method function.

```
function x_new = gelmm(xs, dxs, h, neg_alphas, betas)
    x_new = xs * neg_alphas + h * (dxs * betas);
end
```

The solutions and derivatives of the two last steps are stored in vectors.

```
        % Updating the variables
        xs  = [x_new xs(:,1:(end-1))];
        dxs = [f(t, x_new) dxs(:, 1:(end-1))];
```

The only thing left then of the iteration is to store the new data and compute the next step-size ratio based on the error estimation.

3.5 Error control

In order to control the error, there are mainly two things that need to be done. The first part is to estimate the error and the second one is to choose a new step-size so that the error bound is met. In total there are four functions that take care of this process. For the error estimation, the error coefficient (2.8) discussed in the previous chapter is computed for two separate two-step methods defined by the variables `theta` and `eta`.

```
function c = error_constant(r, theta)
    c = (1/6)*(cos(theta)*r-2*r*sin(theta)+cos(theta)- ...
        3*sin(theta)) / (r*(cos(theta)*r+cos(theta) - ...
        2*sin(theta)));
end
```

The solution from the method defined by `theta` will be the main result while the solution computed with `eta` is only used for comparison.

```
% Calculating new values.
x_new = gelmm(xs, dxs, h, neg_alphas, betas);
x_err = gelmm(xs, dxs, h, neg_alphas_err, betas_err);
```

With the two solutions the error can now be estimated using [13]

$$err = \left| \frac{C_\theta}{C_\theta - C_\eta} \right| \times \|\mathbf{x}_\theta - \mathbf{x}_\eta\| \tag{3.1}$$

where err is the error and can be implemented as

```
function err = error_estimate(c, c_err, x, x_err)
    err = abs(c/(c-c_err))*norm(x_err - x);
end
```

The error estimate can then be used to compute the step-size ratio for the next step. The simplest step-size controller is

$$r_n = \left(\frac{TOL}{err}\right)^{1/(k+1)},$$

where r_n is the step-size ratio as before and TOL is the given error tolerance. This step-size controller has a tendency to oscillate around an appropriate step-size, which is an unwanted behavior. As a solution to that, other step-size controllers have been made that use information from past step-size ratios in order to damp the oscillation [14]. The one used here is

$$r_n = \left(\left(\frac{TOL}{err}\right)^{1/(k+1)}\right)^{1/6} \left(\left(\frac{TOL}{err_{old}}\right)^{1/(k+1)}\right)^{1/6},$$

where err_{old} is the estimated error from the previous step. This can then be computed with the following code.

```
function r = step_control(tol, err, err_old, ord)
    parvec = [1/6 1/6];
    r = prod([(tol/err)^(1/(ord+1)) ...
                  (tol/err_old)^(1/(ord+1))] .^ parvec);
end
```

Not all step-size ratios are acceptable with respect to stability. As discussed in section 2.3 the step-size ratio must be bounded as in eq. 2.13. In order to guaranty stability the step-size ratio is passed on to the function `upper_limit` along with the method parameters.

```
function r = upper_limit(r, thetas)

    % Some factor to make sure that the value is
    % below the upper limit.
    inner_point_factor = .99;

    % The smaller value of the previously computed
    % step-size ratio or the upper limit is chosen.
    r = min([r sqrt(abs(1 - 2 * tan(thetas))) * ...
                inner_point_factor]);
end
```

Then the next step-size can be computed. This takes place in the beginning of each iteration. In case the new step-size reaches beyond the final time value, the following control flow takes place.

```
        if t_n < t + h * r
            h_end      = t_n - t;
            r          = h_end / h;
            h          = h_end;
            isNotDone  = 0;
        else
            h = h * r;
        end
```

3.6 Test problems

For testing purposes some problems were implemented as well. The problems can be grouped into two different categories. First kind taken from [9], are simple single variable problems with a known analytical solution. The second category contains problems more commonly used to show the usability of a variable step-size method. In the first category there are the following problems:

- The negative exponential
  $$\dot{x} = -x, \quad x(0) = 1 \tag{3.2}$$
  with the solution $x(t) = Ce^{-t}$, where $C = 1$ and final time value $t_n = 3$.
- A special case of the Riccati equation
  $$\dot{x} = -\frac{x^3}{2}, \quad x(0) = 1 \tag{3.3}$$
  with the solution $x = 1/\sqrt{t+C}$, where $C = 1$ and final time value $t_n = 6$.
- An oscillatory problem
  $$\dot{x} = x\cos t, \quad x(0) = 1 \tag{3.4}$$
  with the solution $x = Ce^{\sin t}$, where $C = 1$ and final time value $t_n = 4\pi$.
- A logistic curve.
  $$\dot{x} = \frac{x}{4}\left(1 - \frac{x}{20}\right), \quad x(0) = 1 \tag{3.5}$$
  with the solution $x = \frac{20}{1+19Ce^{-t/4}}$, where $C = 1$ and final time value $t_n = 20$.

For those problems the exact solution is implemented as well. In the second category are the following two problems.

- Non-stiff Van der Pol oscillation
  $$\dot{x}_1 = x_2 \tag{3.6a}$$
  $$\dot{x}_2 = \mu(1 - {x_1}^2)x_2 - x_1 \tag{3.6b}$$
  with a period of approximately 2μ. Initial values are $\boldsymbol{x} = (2, 0)$, $t_0 = 0$, final time value is $t_n = 60$ and $\mu = 10$.
- Lotka–Volterra equation
  $$\dot{x}_1 = ax_1 - bx_1x_2 \tag{3.7a}$$
  $$\dot{x}_2 = cx_1x_2 - dx_2 \tag{3.7b}$$
  where $a = 3$, $b = 9$, $c = 15$ and $d = 15$ are coefficients of the problem. Initial values are $\boldsymbol{x} = (1, 1)$, $t_0 = 0$ and final time value $t_n = 5$. Lotka–Volterra has an invariant
  $$H(x_1, x_2) = cx_1 + bx_2 - d\log x_1 - a\log x_2. \tag{3.8}$$

All problems are implemented on the same form. As an example here is the negative exponential.

```
function dx = neg_exp(t,x)
    dx = -x;
end
```

The analytical solution to the problem is implemented as well.

```
function x = neg_exp_exact(t)
    x = exp(-t);
end
```

Chapter 4

Computational results

4.1 Constant step-size for different θ's

While the goal is to make a variable step-size method, the first implementation tested will be of the constant step-size methods defined by different θ's. Since the method's coefficients are smooth with respect to both the parameter θ and the step-size ratio r_n, it can be expected that the characteristics of a method with r_n close to 1 is similar to the characteristics of the underlying constant step-size method.

In order to measure the accuracy of a method, *the mean global error* will be used, which is given by

$$MGE = \frac{\sum_{i=1}^{n} \|\boldsymbol{x}(t_i) - \boldsymbol{x}_i\|}{n}, \tag{4.1}$$

where n is the number of steps. As the global error can vary greatly for different time values, using only the global error at the final value is therefore not reliable for evaluating the accuracy of a method. That is why the mean global error is used instead.

The script was placed in file `script_const_step.m`. The test was carried out by choosing 500 uniformly spaced θ's, over the open interval of zero stable fixed step-size methods, $Z_{1,\theta} = (\pi/4, \pi)$. For each one of the θ's, the computation was repeated for different number of steps, being doubled each time on the form $n = 2^k$ for $k = 4, ..., 11$. The methods were all applied to the four test problems negative exponential, special case of the Riccati equation, oscillatory problem and logistic curve as described in section 3.6.

The results for all the different problems showed similar trends. For demonstration the data for the logistic equation (3.5) was chosen to be visualized. Two figures were made. In figure 4.1 the mean global error (MGE) is plotted against the parameter θ. Each line is for different number of steps and therefore different step-size as presented on the color bar. The error seems to somewhat follow the expected trend of the error coefficient in figure 2.2 with a lower error as θ gets higher in the stable interval. A few values of θ were chosen out of the computation and the same data then replotted in figure 4.2 with the mean global error against the step-size. This demonstrates the second order convergence of the methods.

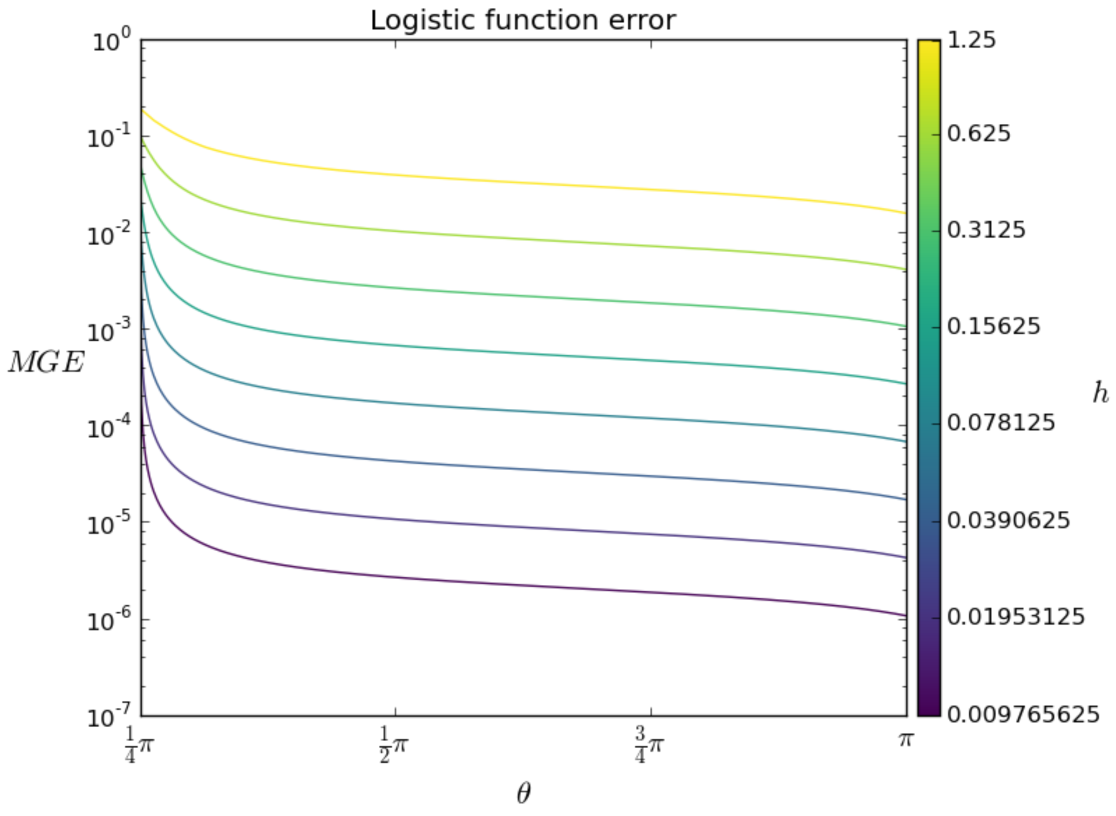

Figure 4.1: The mean global error against the parameter θ on the stable interval $Z_{1,\theta} = (\pi/4, \pi)$, that defines the constant step size methods used. The problem solved is the logistic curve (3.5). Each line represents a different number of steps and thus different step-size. The steps were doubled from 16 and up to 2048. The error scale is logarithmic.

4.2 Variable step-size with θ vs. η

In section 3.5, we discussed how the error could be estimated using two separate methods and by comparing the difference of their solutions with the difference of their error coefficients. The error estimate could then be used to set a new step-size in order to hold the local error close to the given tolerance. As the minimum requirement for a valid parameter is that the underlying constant step-size method is zero stable, we will restrict the parameters to $Z_{1,\theta} = (\pi/4, \pi)$. As we need two parameters for the error estimation pair the method domain becomes $Z_{1,\theta} \times Z_{1,\theta}$.

The computations were performed using the script `script_var.m`. In order to test different pairs, 128 values were uniformly chosen from the interval $(\pi/4, \pi)$. Each value was then paired with each one of the 127 other parameters. The pairs were used to carry out a computation and each time the mean global error was computed and stored along with the number of steps needed. The problem chosen to be the most suitable for this test was the oscillatory problem (3.4) as the step-size must be changed repeatedly as the computation goes over more than one period. The problem was set up as follows

```
problem   = @problems.oscillatory;
exact_sol = @problems.oscillatory_exact;
```

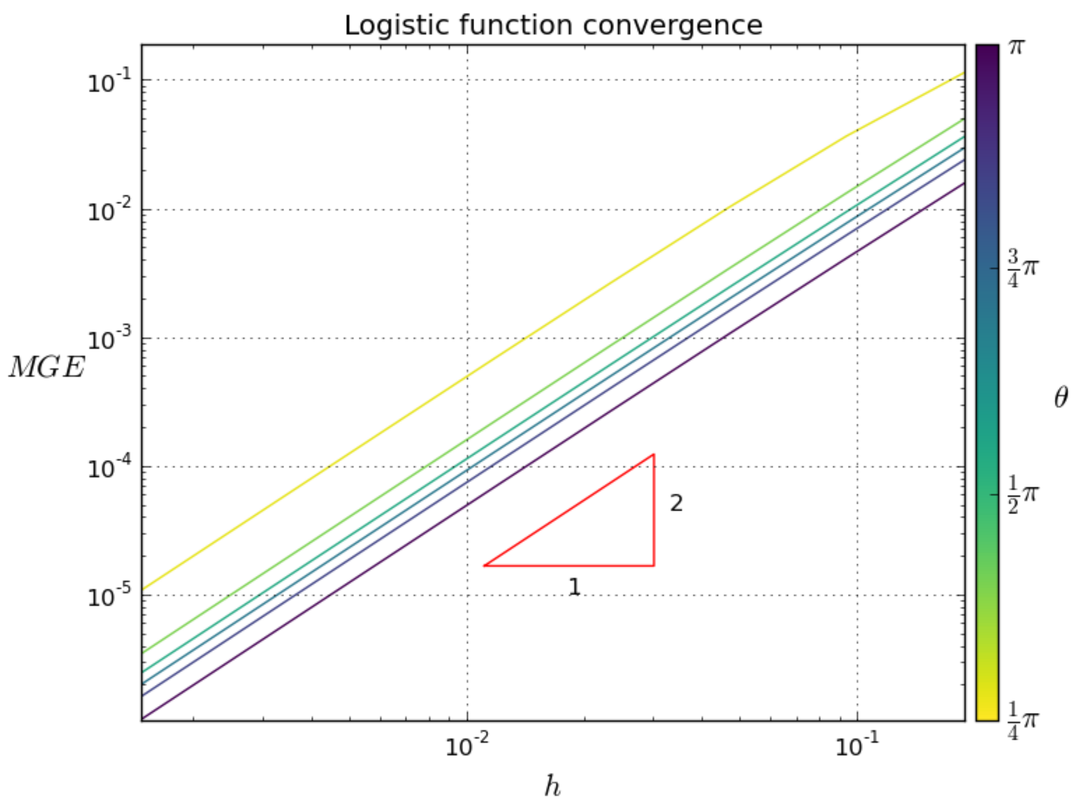

Figure 4.2: Same data as in figure 4.1 showing the order two convergence of 5 of the 500 methods that a solutions was computed with. The plot is of the mean global error against step-size. Both scales are logarithmic. A red triangle with slope 2 is included for comparison of slope.

```
x_start    = 1;
t_start    = 0;
t_end      = 15;
tol        = 1e-4;
```

The results were collected into two 128×128 matrices with empty diagonals.

Due to the upper limit on the step-size ratio described by eq. (2.13), the methods defined by the parameters close to the boundaries of $Z_{1,\theta}$ have a range for r_n that is too limited to be considered. As the result values from the computation close to the borders take extreme values both in accuracy and number of steps due to this limit, it's beneficial to trim away those values before visualizing them. We define the trimmed region of the matrices

$$A = X(7 : 122, 7 : 122),$$

where X is either one of the result matrices and following MATLAB notation of indexing. This cuts away the first and last 6 values on both axes. While including those values would be perfectly fine, the benefit of removing those rows and columns gives the remaining values more range on the color scale when plotted. In figures 4.3 and 4.4 the plots of each one of the result matrices restricted to the region A can be found. Considering fist the mean global error in figure 4.3, while the MGE

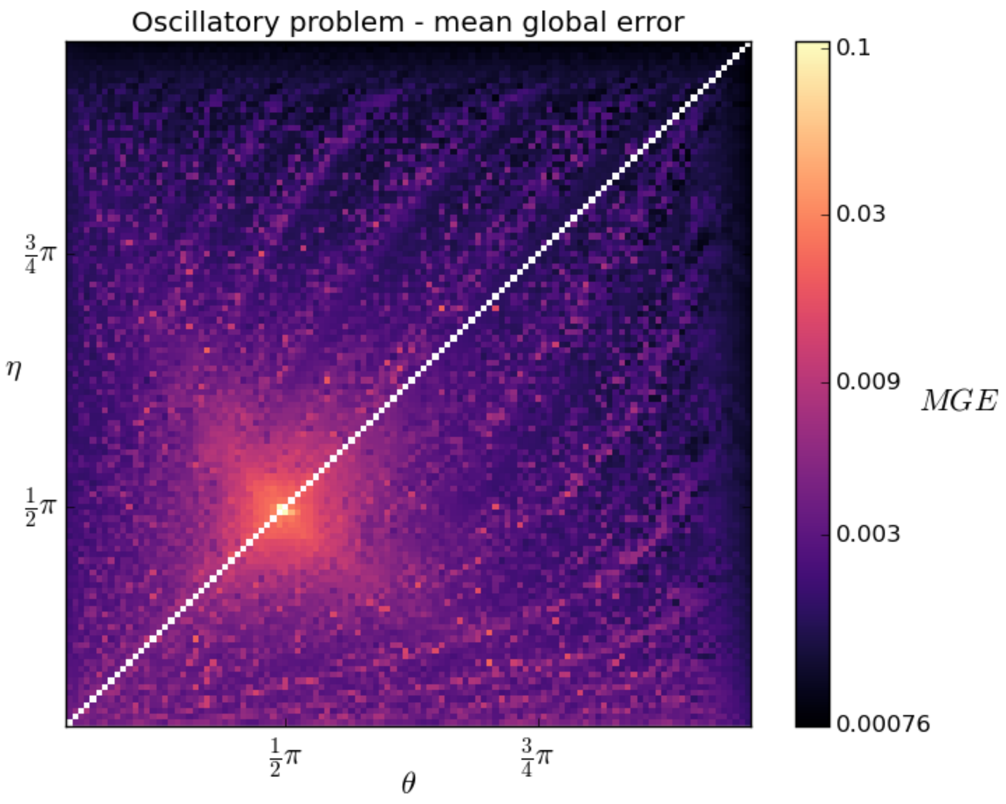

Figure 4.3: The mean global error for each pair (θ, η) restricted to the region A. The diagonal is empty since a parameter can't be paired with itself and on the plot the diagonal is filled in with white. Even though the result contains noise, there is a clear trend of the MGE getting lower as θ and η get larger. The error spikes when both parameters are close to $\pi/2$.

is clearly not a smooth function on the domain of parameter pairs, there is clearly a trend of the error getting lower as both θ and η become larger. The error spikes when both parameters are in the neighborhood of $\pi/2$. The number of steps needed, as seen in figure 4.4 has less noise to it. If we were to consider the number of steps alone, we should choose θ high but η low.

Taking into account both the MGE and the number of steps, it seems to be good to choose a method close to the upper limit and close to the diagonal. In order to choose appropriate pair (θ, η), we could first estimate the maximum step-size ratio r_n needed and from eq. (2.13) choose θ as large as possible. With θ chosen we can then pick η close to θ but smaller. Lets say for example that we want $\max(r_n) \approx 1.2$ to be our upper limit, then from eq. (2.13) we have

$$\theta = \pi + \arctan\left(-\frac{1.2^2 - 1}{2}\right) \approx 2.9$$

and $\eta = 2.85$ would be an option for a pair. To see if this is a reasonable choice of parameters, we must compare the accuracy and the steps needed for those values. As the accuracy for different methods in the same region can vary as we have seen in figure 4.3, we consider some neighborhood of the chosen parameters rather than just a single result. We define a new sub region of the solution matrices, containing

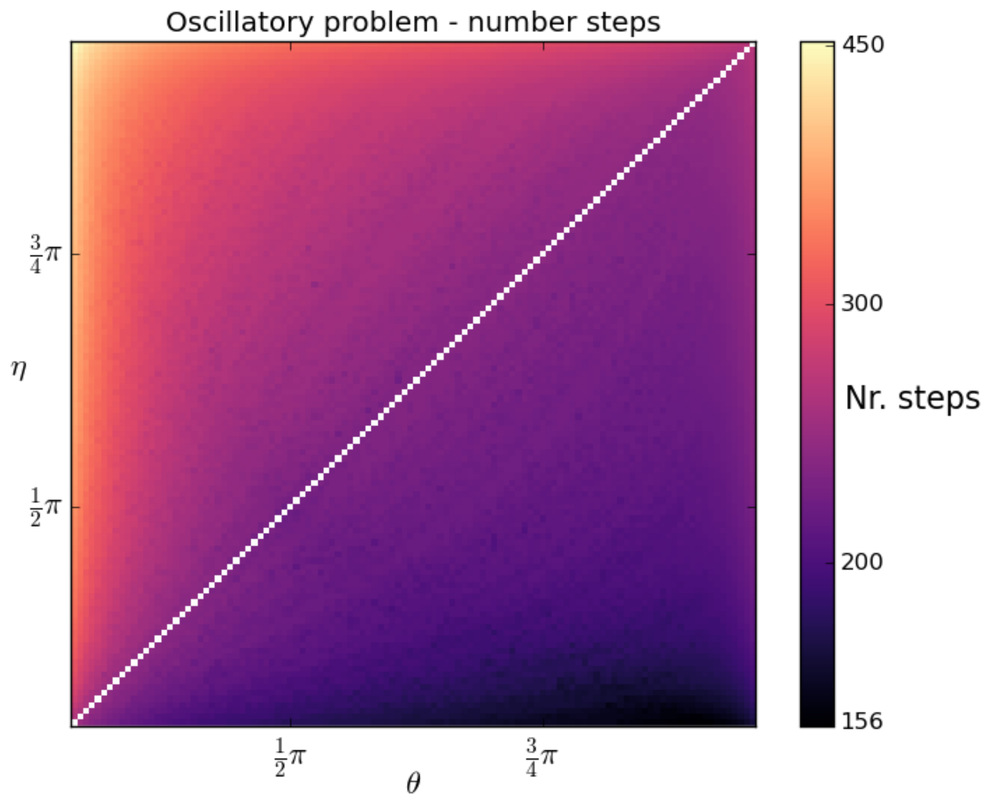

Figure 4.4: The number of steps used for each pair (θ, η) restricted to the region A. While the number of steps is lowest in the lower right corner, one has to take into consideration the accuracy as well, as visualized in figure 4.3. Taking both accuracy and number of steps into consideration the upper right corner seems to be better, but not too close to the borders.

the chosen θ and η along with some of their neighborhood. The region chosen is then

$$B = X(114 : 116, 109 : 113).$$

We do not want the quality of our chosen parameters to be too sensitive to small deviations and as we have already discussed, the MGE in figure 4.3 has noise in it. This is why we are interested in the neighborhood around the chosen parameters and not just the solution of the specific method.

In table 4.1 the minimum, mean and maximum values of the MGE and number of steps are listed for the reions A and B. The methods contained inside B clearly have a good accuracy and small variance of the MGE compared to all methods in A. The number of steps used by the methods in B is above the average for the region A, but again with a small variance and still far from the maximum in A. Bear in mind that it's unreasonable to expect a method to exist that has both the minimum number of steps and minimum MGE in A. The methods contained in the region B can therefore, most likely be accepted as reasonable methods out of A in general.

Table 4.1: A comparison between the *MGE* and number of steps used for the two different regions A and B. The methods are variable step-size but the error coefficient is kept constant.

		min	mean	max
A	*MGE*	0.00076	0.00317	0.10519
	Nr. steps	155	244.46	453
B	*MGE*	0.00102	0.00120	0.00156
	Nr. steps	248	252.47	257

Table 4.2: A comparison between the *MGE* and number of steps used for the two different regions A and B. The methods are variable step-size and the error coefficient is varied as well. Comparing with table 4.1, for this method it does not pay off to vary the error coefficient.

		min	mean	max
A	*MGE*	0.00076	0.00476	0.32679
	Nr. steps	150	242.93	453
B	*MGE*	0.00109	0.00120	0.00142
	Nr. steps	248	252.4	257

4.3 Varying C with respect to r_n

The standard way of using an error coefficient C, as the one in eq. (2.8) is to disregard the current step-size ratio and simply put $r_n = 1$ when evaluating the coefficient. This is what was done in the previous section for testing different pairs of parameters for a variable step-size method. Since we have an equation for the error coefficient that does include the step-size ratio, we now wish to compare what difference it makes to vary the error coefficient with respect to r_n.

The script `script_var_var_c.m` was used for this test. The computation from the previous section was repeated by reevaluating C at each step with respect to r_n. Plotting the results as before gave figures very similar to 4.3 and 4.4 and therefore no reason to display them here as well. A better way of comparing the two different alternatives is to repeat table 4.1 for the new results. Those values can be found in table 4.2.

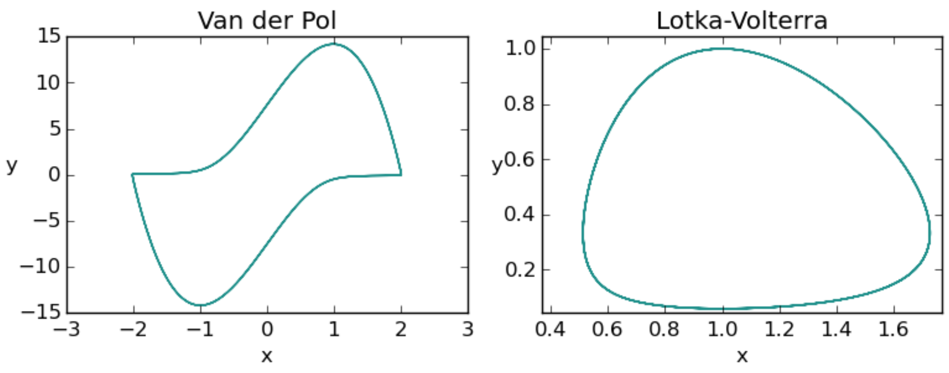

Figure 4.5: A phase plot of the Van der Pol oscillation on the left and the Lotka-Volterra equation on the right.

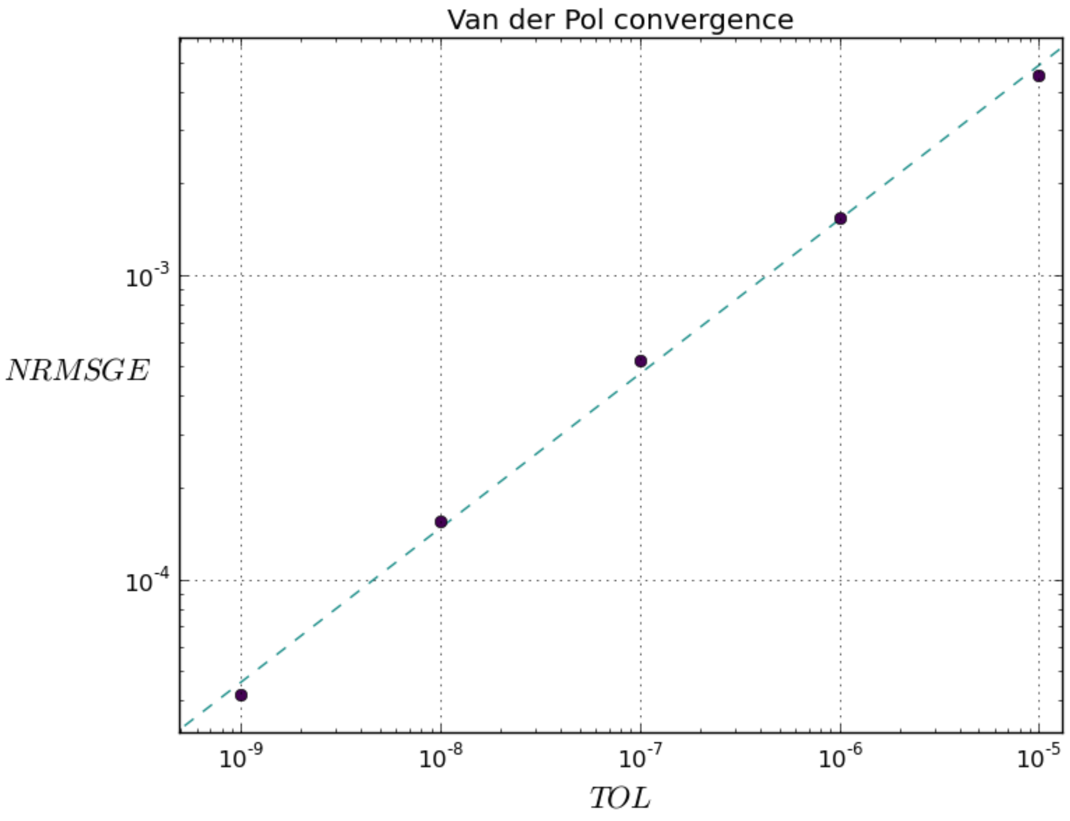

Figure 4.6: The convergence of the normalized root mean square global error for the Van der Pol oscillation, as related to the given tolerance. The five data points are plotted in as circles and the linear regression for the result is represented by a dotted line. The slope of the line is 0.507.

Comparing tables 4.1 and 4.2, we see that the values don't differ that much. What is more interesting is that when C is varied, the solvers seem to be even worse. There doesn't seem to be any reason to vary C like this and the standard way of doing this is better.

The error coefficient only plays a role in the error estimator which estimates the local error. The quality of a method is however measured here using the mean global error. As the error estimator is only one component of a variable step-size method, the mean global error doesn't necessarily reflect the quality of the error estimation. The method itself along with the step-size controller also plays a big role in the final result. A better test for the error estimator would therefore be to consider the difference of the true local error against the estimated error. The analysis of such experiment has however proven to be more difficult the expected at first glance and will not be covered here.

4.4 Testing on harder problems

Using the results from the previous sections in this chapter, we continue now with a test of a variable step-size method on less trivial problems than those used before. The problems used here are the Van der Pol equation (3.6) and the Lotka-Volterra

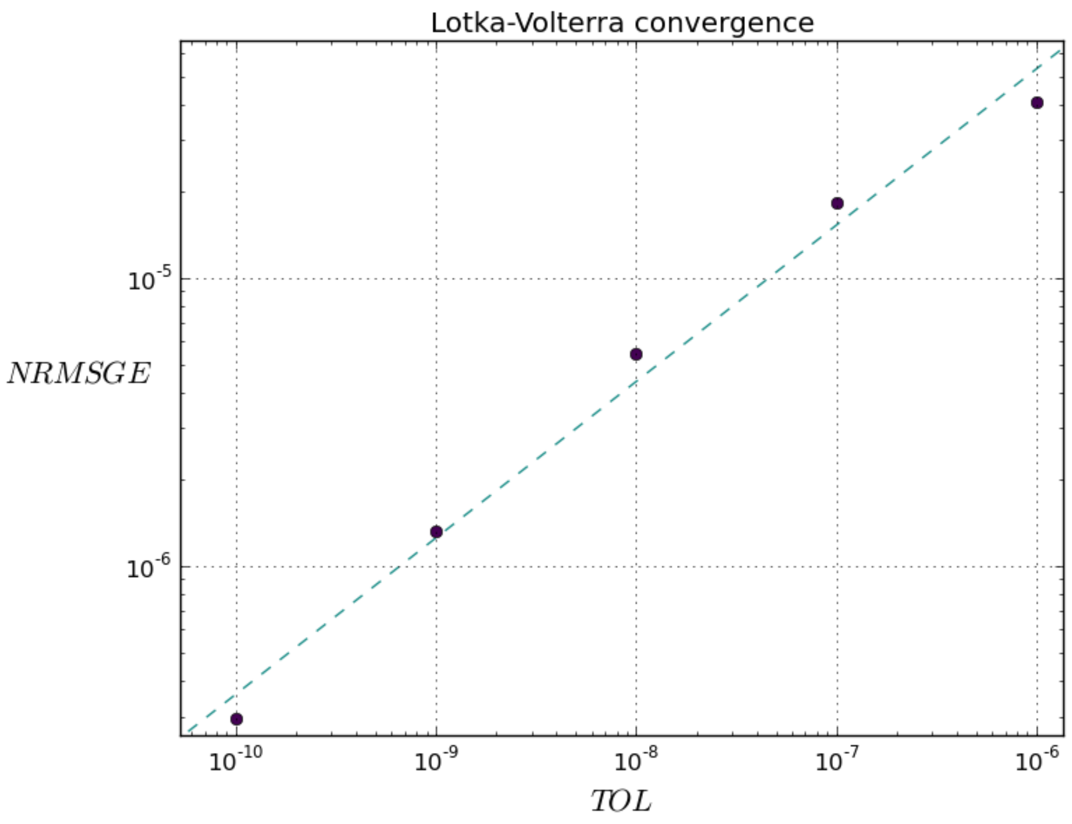

Figure 4.7: The convergence of the normalized root mean square global error for the Lotka-Volterra equation, as related to the given tolerance. The five data points are plotted in as circles and the linear regression for the result is represented by a dotted line. The slope of the line is 0.542.

equation (3.7). A phase plot of the two problems can be found in figure 4.5. As the exact solution is not available for the two problems considered here, the built-in solver `ode113` in MATLAB was used for comparison with the relative tolerance set at 10^{-13} and the absolute tolerance at 10^{-16}.

The corresponding scripts is `script_lotka.m` and `script_vdPol.m`. The problems were solved with the parameter pair $(\theta = 2.9, \eta = 2.85)$ and the non-varying C stepper was used. The problems were then solved for different tolerance values. In the case of the Van der Pol equation, the tolerances used were `[1e-5 1e-6 1e-7 1e-8 1e-9]` and for the Lotka-Volterra equation `[1e-6 1e-7 1e-8 1e-9 1e-10]`.

Both problems are two dimensional and therefore we need to replace the mean global error with something else. We define *the normalized root mean square global error* as

$$NRMSGE = \frac{1}{t_n - t_0}\sqrt{\sum_{i=1}^{n} h_i |\mathbf{x}_i - \mathbf{x}(t_i)|^2}. \tag{4.2}$$

In order to understand the relationship between the tolerance and the normalized root mean square global error, we must start at the error estimator that estimates the local error e_{local} while we are measuring the global error e_{global}. For a second order method we have that $e_{local} \approx TOL \propto h^3$. This means that $e_{global} \propto h^2 \propto TOL^{2/3}$

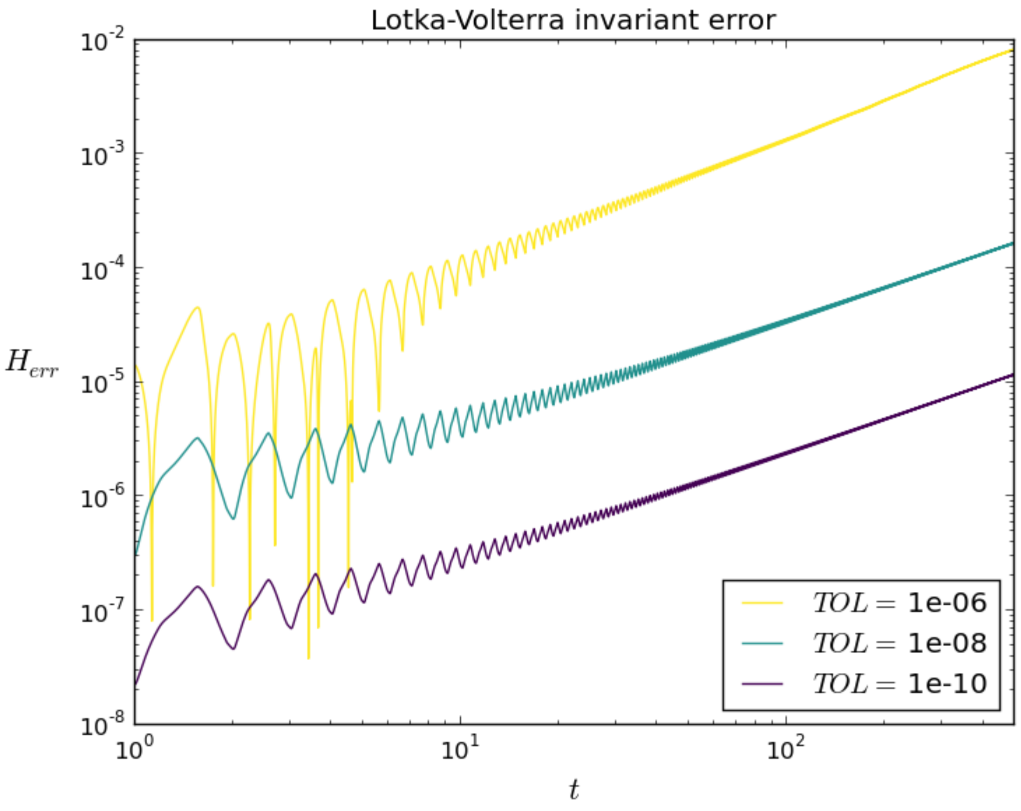

Figure 4.8: The error of the invariant for Lotka-Volterra as given by eq. (4.4) over time both on log scales, for three different tolerance values. The invariant slowly drifts away from the initial value and the error grows.

and thus

$$NRMSGE \propto e_{global} \propto TOL^{2/3}. \tag{4.3}$$

This means that when $NRMSGE$ is plotted against TOL on logarithmic scales, we expect linear relationship with the slope $2/3 = 0.\overline{66}$. In figure 4.6 we can see the convergence rate of $NRMSGE$ against the tolerance for the Van der Pol equation. A linear regression was done on the logarithmic values of the five data points which gave the slope 0.507. Same kind of plot for the Lotka-Volterra equation can be found in figure 4.7 and slope of the linear regression is 0.542. Both those slopes are under the theoretical slope that we expected. This is however not so surprising since the step-size controller doesn't always succeed in holding the local error close to the tolerance and therefore some unforeseen factors occur in the results.

Finally in this section we have figure 4.8 showing the error of the invariant (4.4) of the Lotka-Volterra equation. The error is computed by

$$H_{err} = |H/H_0 - 1| \tag{4.4}$$

where H_0 is the invariant at the initial value. As we see in figure 4.8, there is a clear drift of the invariant. This is however to be expected for a numerical solution. Furthermore the error gets less as the tolerance is decreased.

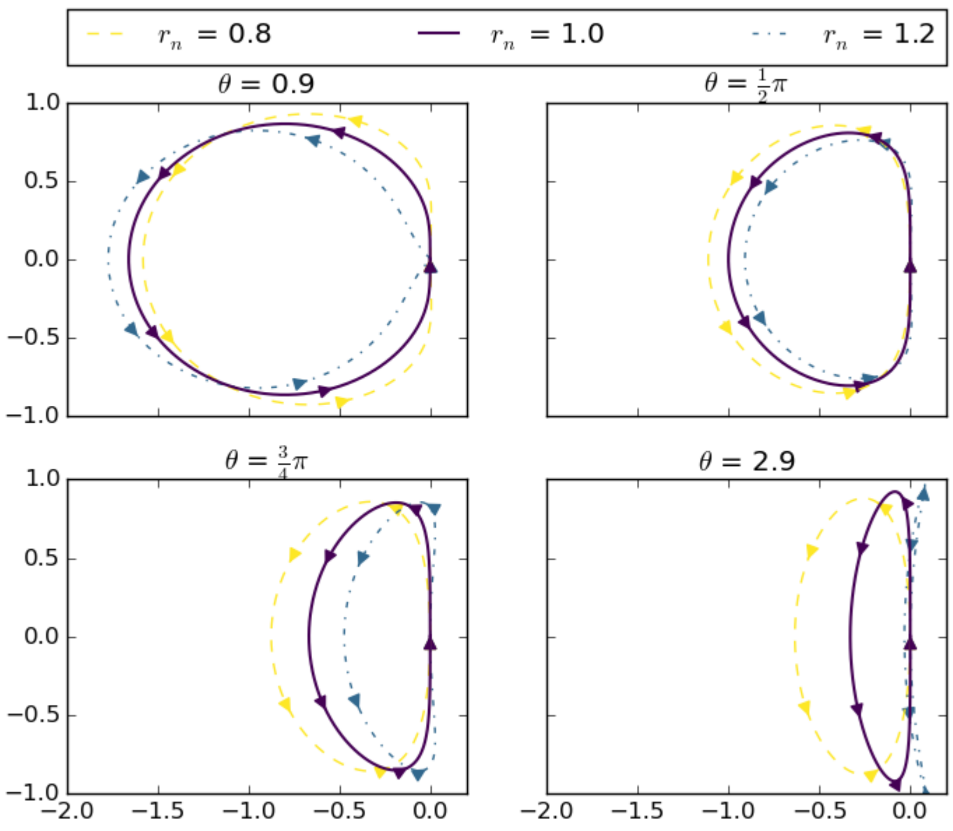

Figure 4.9: The root locus curve for the stability region of four different values of θ. Each plot contains three curves showing the effect of the step-size ratio on the stability region. The stability region gets smaller as θ gets larger, but for a small value of θ and large step-size ratio, the region tends to diverge from the imaginary axis.

4.5 Finding stability regions

What we have not considered so far for the variable step-size methods in this chapter is the stability regions. Although stability regions are of higher interest when dealing with stiff problems, we would like to study how these regions are affected by the choice of θ. The process of computing the root locus curve is more or less the same for all cases of multistep methods. Equation (2.18) is applied to the unit circle in the complex plane and the resulting curve plotted. Such a plot can be found in figure 4.9 for four different θ's and three different step-size ratios. In order to stay within the zero stability region for all cases, the outer θ values chosen were 0.9 and 2.9 along with $\frac{1}{2}\pi$ (Adams method) and $\frac{3}{4}\pi$ in between. The step-size ratios are 0.8, 1.0 and 1.2. The resulting plots can be found in figure 4.9.

The stability regions clearly get smaller as θ gets larger. For $\theta = 2.9$ the regions get quite small and in the case of step-size ratio $r = 1.2$ the range of $h\lambda$ is very limited. On the other side, for $\theta = 0.9$ the region is inconvenient as well for $r = 1.2$. As the region quickly diverges from the imaginary axis except for the origin, the method would not be stable for a linearized problem whose eigenvalue λ that lies on or is close to the imaginary axis.

Chapter 5

Discussion

5.1 What has been covered

Using the interpolation conditions in eq. (1.4), the coefficients for a multistep method as a function of the parameter θ and the step-size ratio r_n were constructed in section 2.1. Those coefficients were then used to analyze the potential methods defined by them and for implementation.

The order of convergence was derived in similar way to what has been done for fixed step-size methods and the condition used to find a error coefficient dependent on the step-size ratio. A condition was found for the zero stability based on the roots of the generating polynomial $\rho(z)$ and absolute stability conditions were discussed. Finally in chapter 2 the convergence of a method was related to the consistency and the stability of a method and the overall limits of the domain of two-step methods discussed.

In chapter 3 an implementation was discussed that uses two parameters to form a variable step-size method. Using the implementation, there were four computational tests performed in chapter 4. An initial test of the underlying fixed step-size method was done which showed the expected behavior as predicted by the error coefficient. The variable step-size implementation was then used to compare the efficiency for different parametric values of the method pair. The variable step-size method defined by the parametric pair (θ, η) was tested on a single problem for multiple values and conclusion reached that picking θ as high as the zero stability allows, paired with an η smaller but close to θ was a reasonable way of choosing the pair. The computation was repeated with an error coefficient that was updated according to the step-size ratio r_n and showed that for this implementation of the method it does not pay of to vary the error coefficient. The chosen method pair with fixed error coefficients was then used for a test on the benchmark problems, non-stiff Van der Pol oscillation and Lotka-Volterra equation and the convergence of global error related to the chosen tolerance considered.

Finally the stability region as it depends on the parameter θ was considered which showed that higher values for θ results in smaller stability regions. As the equation 2.18, describing the stability region is smooth for θ, the shrinking and growing of the region behaves accordingly.

5.2 Further work on explicit two-step methods

As mentioned in section 1.5, there already exists a comprehensive collection of theory on multistep methods. This simplifies the process of making sure that we have covered every aspect needed. For most parts there is not much more to do with the exception of zero stability. While the conditions in eq. (2.13) guarantee zero stability, there's nothing that says that same method with bigger step-size ratio is not stable as well. By manipulating the sequence of matrix multiplication given by the condition in definition 2.7 by choices of step-size ratios sequences, the condition could still be met even though the root condition does not hold for all matrices.

The way of choosing the parameter pair as was done in section 4.2, by experimenting on a single problem has a clear limiting factor to it. What the test showed however is that the accuracy and the effort needed to reach it did not vary enough that it would pay off to go into too much detail in choosing the parameters. One surprising aspect of the results was that when θ and η were close to one another, with the exception of the neighborhood of $\pi/2$, they gave better estimation of the local error. The numerator in eq. (3.1) will become smaller as the parameters are closer to one another and one could suspect that this would introduce an unwanted sensitivity in the error estimator. This however seems not to be the case. A better way to consider the optimal pair would be to compare the error estimator with the true local error. An attempt was made to do this comparison for this work but was dropped after it turned out to be a bigger project than originally anticipated.

While the method formulation gives a convenient way of getting a variable step-size method by simply choosing a parameter in the stable region, there are other ways this work can be used. As the domain of all possible explicit two-step methods of maximal order can be reached from the parameter θ and the step-size ratio r_n, one could consider other relationships between the two variables. Say for example that we want the error coefficient to be fixed for all r_n, then that gives a condition for θ and defines a new variable step-size method. Same way the root of the method could be fixed and give a relationship between θ and r_n and thus free the method from stability restrictions.

5.3 Other k-step methods

While explicit two-step methods is the only case discussed in this thesis, the work of Arévalo [3] allows for any number of steps in k-step methods, both explicit and implicit. As all other methods than the explicit two-step ones have more that one parameter, the work of analyzing their domain can be a lot more difficult. For instance it might not always be possible to find an explicit condition for zero-stability. Higher order methods, implicit in particular, are of more interest and therefore the work of analyzing them of high importance. Hopefully the work done here will shed some light on what can be expected to be found for the higher order cases as well and simplify their analysis.

Bibliography

[1] K. G. Andersson and L.-C. Böiers, *Ordinära differentialekvationer*. Lund: Studentlitteratur AB, 1989, 1992.

[2] C. Arévalo, C. Führer, and M. Selva, "A collocation formulation of multistep methods for variable step-size extensions," *Applied numerical mathematics*, vol. 42, no. 1, pp. 5–16, 2002.

[3] C. Arévalo, "Parametric variable step-size formulation of mutlistep methods," Unpublished.

[4] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, "Vode: A variable-coefficient ode solver," *SIAM journal on scientific and statistical computing*, vol. 10, no. 5, pp. 1038–1051, 1989.

[5] G. Dahlquist, "Convergence and stability in the numerical integration of ordinary differential equations," *Math. Scand*, vol. 4, no. 1, pp. 33–53, 1956.

[6] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[7] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II*. Springer, 2010.

[8] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations*. Ser. Springer series in computational mathematics: 8. Berlin ; New York : Springer-Vlg, cop. 1987, 1987.

[9] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick, "Comparing numerical methods for ordinary differential equations," *SIAM Journal on Numerical Analysis*, vol. 9, no. 4, pp. 603–637, 1972.

[10] E. Isaacson and H. Keller, *Analysis of Numerical Methods*, ser. Dover Books on Mathematics. Dover Publications, 1994.

[11] MATLAB, *Version 8.3.0.532 (R2014a)*. Natick, Massachusetts: The MathWorks Inc., 2014.

[12] T. Sauer, *Numerical Analysis*, ser. Always learning. Pearson, 2011.

[13] L. Shampine, *Numerical Solution of Ordinary Differential Equations*, ser. Chapman & Hall mathematics v. 4. Taylor & Francis, 1994.

[14] G. Söderlind, "Digital filters in adaptive time-stepping," *ACM Trans. Math. Softw.*, vol. 29, no. 1, pp. 1–26, Mar. 2003.

Master's Theses in Mathematical Sciences 2015:E51
ISSN 1404-6342

LUNFNA-3020-2015

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/