Numerical methods for solving ODE flow

Numerical Methods for Solving ODE Flow

Bratislav Tasić

Printed by Eindhoven University Press

Cover design: JWL Producties

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Tasić, Bratislav

Numerical Methods for Solving ODE Flow
by Bratislav Tasić. -
Eindhoven : Technische Universiteit Eindhoven, 2004. Proefschrift. -
ISBN 90.386.0952.3

NUR 918

Subject headings: ordinary differential equations / flow /
numerical methods / Euler methods / Runge-Kutta methods /
linear multistep methods / inverse interpolation

2000 Mathematics Subject Classification: 65L05, 65L06, 65L20, 65L70

Numerical Methods for Solving ODE Flow

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College
voor Promoties in het openbaar te verdedigen
op maandag 13 december 2004 om 16.00 uur

door

Bratislav Tasić

geboren te Niš, Servië en Montenegro

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. R.M.M. Mattheij
en
prof.dr. M. Hermann

To my parents

Mojim roditeljima

Preface

Nature is a temple whose living colonnades
Breathe forth a mystic speech in fitful sighs;
Man wanders among symbols in those glades
Where all things watch him with familiar eyes.

Like dwindling echoes gathered far away
Into a deep and thronging unison
Huge as the night or as the light of day,
All scents and sounds and colors meet as one.

Perfumes there are as sweet as the oboe's sound,
Green as the prairies, fresh as a child's caress,
- And there are others, rich, corrupt, profound

And of an infinite pervasiveness,
Like myrrh, or musk, or amber, the excite
The ecstasies of sense, the soul's delight.

La Nature est un temple où de vivants piliers
Laissent parfois sortir de confuses paroles;
L'homme y passe ŕ travers des foręts de symboles
Qui l'observent avec des regards familiers.

Comme de longs échos qui de loin se confondent
Dans une ténébreuse et profonde unité,
Vaste comme la nuit et comme la clarté,
Les parfums, les couleurs et les sons se répondent.

Il est des parfums frais comme des chairs d'enfants,
Doux comme les hautbois, verts comme les prairies,
Et d'autres, corrompus, riches et triomphants,

Ayant l'expansion des choses infinies,
Comme l'ambre, le musc, le benjoin et l'encens
Qui chantent les transports de l'esprit et des sens.

Charles Baudelaire, *Correspondances*

Once I heard that all good things must have an end. It has been four years ago since I was offered to perform a PhD research, as a member of Scientific Computing Group at the Eindhoven University of Technology, on new numerical methods for solving flows of ordinary differential equations. Having an electrical engineering and automatic control background, I was familiar with differential equations as they were used for mathematical modelling of various physical processes. However, numerical methods for solving them were always (in my eyes) a powerful "magical" tool, on which you could almost always rely to obtain results you need.

During my research period I have realized that this magic is coming from amazingly rich and profound numerical theory, accompanied by numerous practical realizations in widely used software packages. To that end I have to admit that, although not easy at first, it was pleasant and fulfilling to find and analyse a new approach for solving ODE problems.

The basic idea of the method originates from problems in fluid dynamics and it was introduced in work of prof.dr. R.M.M. Mattheij and dr. K. Laevsky concerning a glass production process. In this thesis the method principle and the application area are extended. This new approach shows a lot of potential and hopefully it may become an integral part of the family of popular numerical methods that are used for solving differential equations. I strongly believe that there is no such a joy for a numerical mathematician than to see his ideas implemented in various practical applications.

Acknowledgements

According to Umberto Eco, a true virtue of a serious scientist is the ability to formulate the acknowledgements at the end of his work. If one does not have anyone to thank for, the whole work becomes questionable. Hence I would like to take this opportunity to show my gratitude to a number of people who essentially contributed in efforts to bring this thesis to its end.

First, it is a great pleasure to thank prof.dr. R.M.M. Mattheij for offering me the opportunity to become involved in such an interesting topic and for all his trust, advisory and guidance in supervising my research. Many thanks go to prof.dr. M. Hermann and prof.dr. J.G. Verwer for their efforts for improving results obtained in this thesis. I would especially like to thank dr. H.G. ter Morsche with whom I had prodigious discussions about the interpolation topic and who helped to extend my knowledge in this area.

Concerning the practical part of my research, I would like to thank several people who shared with me their expertise in various subjects. I had many useful discussions with ir. V. Vidojković and ir. D. Milošević about problems from electrical networks. Their help improved the practical aspects of this thesis. I would also like to thank m.sc. B. Veselić for his assistance in the control problem, as well as dr. P. Kagan for helping me use his software for solving FEM problems.

During the time I spent being a member of the Scientific Computing Group I enjoyed a friendly atmosphere surrounded by many intelligent and interesting colleagues. Some of them deserve a special attention for their help on a daily basis. In particular I would like to thank dr.ir. B.J. van der Linden, dr.ir. M.J.H. Anthonissen, dr. W. Drenth, dr.ir. P.C.A. de Haas, dr. J.M.L. Maubach, m.sc. I.A. Lioulina, m.sc. M.G. Graziadei and dr. K. Laevsky.

As part of the small tradition, I would like to greet my friends and members of the small Serbian community in The Netherlands; namely Goran, Nataša, Dragan, Milena, Nenad, Dragana, Vojkan, Maja, Dušan, Aleksandra, Darko, Mirjana, Jelena, Milan, Marija, Igor, Jelena and Dragan. A special greeting goes to my friends in Niš, who are far away from here and yet still very close to me, especially kum Milan and my dear friend Tomislav.

Finally, I would like to thank my family, my father Aleksandar, my mother Gordana, my sister Suzana and my niece Mina, for all their love and support throughout all these years. Their faith encouraged me to overcome problems that I encountered through time and to become a better person.

Contents

CHAPTER 1

Introduction

1.1 Background

The history of ODEs (Ordinary Differential Equations) goes all the way back to the XVIIth century when two great scientists Isaac Newton and Gottfried Leibniz introduced calculus. Soon after, it was noticed that not all ODEs can be solved analytically. In order to overcome this problem, i.e. to find at least an approximate solution, numerical methods for solving ODEs were born. Certainly the most famous ones are the Euler Forward and the Euler Backward methods, named after the remarkable scientist Leonhard Euler. At the end of the XIXth and the beginning of the XXth century more advanced, complex and accurate methods appeared. Most significant contributions were given by Adams and Bashforth developing linear multistep methods (LMMs) and Runge, Kutta and Heun leading to what is now called Runge-Kutta (RK) methods. This represented the beginning of the development of two important classes of numerical methods. Today some of the most interesting methods are BDF (Backward Differential Formula) methods, which are LMMs, and methods based on Gaussian quadrature, such as Gauss, Radau and Lobatto methods. The importance of BDF methods was first noticed in [13] and well-analysed in [21]. The aforementioned IRK methods are firstly introduced in [8,29,30]. Of course, many other numerical methods for solving ODEs were developed and some nice overviews are given in [10,21,23,24,31,33].

Practical applications of ODEs can be found in almost all technical disciplines. For example, mathematical models of electrical circuits, mechanical systems, chemical processes, etc. are described by systems of ODEs. Even generally more complex problems, e.g. modelled by semidiscretised PDEs (Partial Differential Equations), can be solved by some of the numerical methods for solving ODEs. One example of an ODE problem, which we will analyse later, is a deforming material blob (see e.g. [32,38,40,42,44,49]), where the velocity field is determined by a boundary value problem. This problem typically occurs in the glass industry, where one needs to numerically compute a boundary evolution of molten glass. The position vector satisfies an ODE, where the derivative is equal to the velocity. Such ODEs actually have a continuum of initial values rather than a single vector and we would like to track this continuum in time. We will refer to this as *a flow* and to the corresponding ODE problem as *a flow problem*. The blob evolution problem is of special interest here since the velocity field (e.g. coming from the Stokes problem) is obtained numerically in

general and thus not known explicitly. This causes problems in applying ODE solver directly.

Another interesting example of such problems are electrical circuits with nonlinear elements, where the behaviour of the circuit strongly depends on the initial values (e.g. capacitor initial voltage values). To analyse this effect, one again needs to obtain a flow, i.e. to solve the corresponding flow problem. Moreover, transfer functions of some electrical components (e.g. diodes, transistors, thyristors, etc.) are not known in general. One way to obtain them is to perform an experiment where values are measured at some points. The result, of course, is a velocity field, which is known discretely only.

1.2 Problem setting and approach

This thesis is concerned with solving flow problems where, as mentioned before, the velocity field is not known explicitly. To numerically solve a flow problem one needs to employ an appropriate numerical method, according to the nature of the problem (e.g. dimensionality, requested accuracy, stiffness, preservation of the volume, energy etc.). Explicit numerical methods are easy to use, allowing the solution to be computed directly from known values at previous time-levels. However, when solving so-called *stiff problems* with explicit methods (cf. [13]), the numerical computation poses a constraint on the time step in order to produce meaningful (stable) results. For problems where e.g. volume preservation is required, one should use so-called *symplectic methods* (cf. [46]). Many of these symplectic methods are implicit. Hence we are again faced with a problem to use implicit methods.

Although the theory has matured significantly, in terms of variety and quality of numerical methods, some problems are still present. One is caused by the fact that implicit methods lead to a (non)linear system of equations to be solved at every time-level. The "standard" way to solve such a nonlinear system, coming from an implicit time discretisation of an ODE, is to apply Newton iteration. However, this approach introduces some additional problems, like the convergence of the Newton method. For our type of problems an additional concern arises as the iteration function is not known explicitly. Moreover, for iteration one not only needs the explicit representation of the velocity field, but also the inverse Jacobian matrix. This is not straightforward to be carried out. However, if the flow problem is autonomous or if the explicit dependence on time is only present in the forcing term, there is an alternative more effective way, which we propose here.

In this thesis we develop a new method, which we call *the flow method*. As illustrated later, many problems in science and engineering are autonomous. This allows us to implement our method for quite a large class of relevant problems. The method employs existing implicit methods, thereby preserving favourable properties of these methods like $A(\alpha)$-stability or symplecticness. Since all implicit methods, applied to a nonlinear ODE, introduce a nonlinear system to be solved at every time-level and since we would like to avoid using Newton iteration, we apply inverse interpolation. This gives rise to a new method that is effectively explicit.

Since we assume the problem to be autonomous, we can use known values of the velocity field at previous time-levels to approximate the solution at the following time-level. This approximation is done by virtue of inverse interpolation. For this we need to know the approximate velocity field at some spatial points only, which

allows us to determine solution values even where the velocity field is not known explicitly.

Interpolation represents an essential part of the flow method, which of course introduces an additional error. This can be controlled separately and fairly straightforwardly. The theory of interpolation is extensive and a variety of methods can be used. To keep (interpolation) computational costs small, our first choice is piecewise linear interpolation. This method, although being of low order, can produce satisfactory results in many application, as illustrated later. If one chooses to employ higher order implicit methods, the appropriate interpolation method is most likely of the higher order as well. We demonstrate this for a few. Applying the flow method for solving multivariate problems, or use of RK methods of higher order, is not straightforward. This problem will be addressed separately, since interpolation is more complex there. However, problems of interest can successfully be solved also here, as illustrated by several examples.

1.3 Outline of the thesis

In the following chapter we start with some basic ODE theory being important for the analysis in the subsequent chapters. After a short introduction, we discuss some useful properties of ODEs, such as existence and uniqueness conditions of the solution, as well as the stability, see Section 2.2. In Section 2.3 we introduce some (simple) numerical methods. We give the definition and the properties of stiff problems, together with stability properties that determine which numerical methods should be used for solving them. Section 2.5 is devoted to LMMs, among which BDF methods in particular. In Section 2.6 we treat RK methods, in particular implicit methods, such as Gauss, Radau and Lobatto methods and the so-called DIRK (diagonally implicit Runge-Kutta) methods. We conclude this chapter with some implementation aspects of implicit methods. We point out problems in solving nonlinear systems of equations when employing the Newton method.

The flow method involves inverse interpolation (in contrast to an iterative process). Since interpolation is a well-studied subject to date, we give an overview and some important results, which will be used in the rest of the thesis, see Chapter 3. In Section 3.1 a short introduction to approximation and interpolation is given. To determine which interpolation method should be used, one can use some of the characteristics shown in Section 3.2. A short overview of univariate interpolation issues can be found in Section 3.3. A more fundamental part of this thesis is Section 3.4 on multivariate interpolation, in particular piece-wise linear interpolation. As for estimating the error, we give a new proof of an existing result and compare this result with another important one. We also introduce a theorem that connects these two results. The rest of the section is devoted to higher order methods, such as piece-wise cubic, surface spline and biharmonic spline interpolation. Finally, in Section 3.5 we summarise the main aspects of inverse interpolation, that will be used later.

The flow method is discussed in Chapter 4. To start with, we first analyse a scalar flow problem and the use of Euler Backward. In Section 4.1 we sketch the method and some notation. Interpolation is an essential ingredient in the flow method, see Section 4.2. Since numerical solution curves should not "intersect", we need a notion of well-posedness of the numerical method, as shown in Section 4.3. The above mentioned interpolation introduces a more complicated local error than usual. An error

analysis of this is given in Section 4.4. The most interesting aspect of our method is its stability. In Section 4.5 it is shown that the stability behaviour is similar to that of the Euler Backward method, despite the fact that our method is explicit. We conclude the chapter with some practical aspects and several numerical examples, see Section 4.6. The flow method can successfully be applied for a class of non-autonomous problems, where the time dependence is present via the forcing term of the velocity field. This is illustrated by an example.

The flow method can also be constructed by employing higher order implicit methods, such as LMMs and RK methods. Chapter 5 shows how this can be done with LMMs, in particular BDF methods. Section 5.1 describes this in general. For LMMs the well-posedness introduces a time-step constraint, which is more strict than for Euler Backward. We investigate this in Section 5.2. The last two sections of this chapter deal with local errors and stability analyses. It is shown that the local error again depends on two components, namely the local discretisation error of the standard LMM and the interpolation error. In the last section we show that the good stability properties of BDF methods are inherited by the flow method based on them, see Section 5.4.

Following the same pattern as in Chapter 5, Chapter 6 analyses the usage of IRK methods. The main difference is coming from the fact that IRK methods are multistage methods, which increase the dimensionality of the nonlinear system to be solved. This leads to multivariate interpolation, even for scalar problems, see Section 6.1. To employ multivariate inverse interpolation one needs to generate a multidimensional grid. This is an essential part of the method and closely related to the well-posedness property of the method. Since we are dealing with a vectorial problem, we need a double concept for well-posedness, viz. the topology preservation in space and the topology preservation in time, see Section 6.2. The section is concluded by an example that shows which IRK methods should be used in order to ensure a well-posed problem. Again we give a local error analysis, where we mainly focus on the interpolation error analysis, see Section 6.3. Finally, a stability analysis is given in Section 6.4, where we show that favourable stability properties of RK methods are preserved.

In the first few chapters the flow method is constructed for scalar ODEs. The extension to vectorial problems turns out not to be straightforward. In Chapter 7 we consider the main aspects of the flow method applied to a general vectorial problem, see Section 7.1. For simplicity we restrict ourselves to the Euler Backward method and the implicit midpoint rule. The implementation of the latter is quite similar to that of the Euler Backward method. Even for higher dimensional stiff problems, the flow method can produce satisfactory results. Using results from Chapter 3, we give a local error analysis in Section 7.2. A stability analysis shows that the flow method again preserves the stability properties of the Euler Backward method, see Section 7.3. To conclude this chapter, several examples of flow problems, with a discretely given velocity field, are solved. The first one is a stiff problem coming from an electrical network with nonlinear elements, where the "velocity field" is obtained experimentally. The second example concerns a problem where the velocity is a numerical solution of a divergence free velocity field, coming from a boundary problem. To solve this problem we apply the flow method based on the implicit midpoint rule and show that a sufficient accuracy can be achieved even for relatively long time integration intervals. The third example deals with a problem from

the control theory, in particular the amplitude synchronisation of a harmonic oscillatory system. The mathematical model of an oscillator is an ODE for which we need a symplectic method. Hence we apply the flow method based on the implicit midpoint rule to show that computing the solution can be achieved with a sufficient accuracy. In the last example we address a problem of the motion of a viscous axisymmetric body, where the velocity field needs to be computed at every time-level by solving a Stokes problem. The physical property of such a problem is the volume preservation in time. Hence a symplectic method needs to be used and we again apply the flow method based on the implicit midpoint rule and obtain sufficiently accurate results even for large time scales.

In the final chapter the most important conclusions are addressed, together with some recommendations for the future research. An idea is presented how to extend the application area of the flow method, which would include general non-autonomous problems.

CHAPTER 2

Ordinary differential equations

In this chapter we will give an overview of the ODE theory which will be useful in the following chapters. First, we give some definitions related to ODE problem settings, namely IVP and the flow problem, followed by the existence, uniqueness and stability properties of the ODE solution. The main part of this chapter is related to the numerical methods for solving ODE problems, so they are introduced in Section 2.3. Since we are mainly interested in stiff problems, a short overview is given in Section 2.4. The number of numerical methods is extensive, so we introduce few of them, namely linear multistep methods and Runge-Kutta methods in Sections 2.5 and 2.6. Finally, since implicit numerical methods involve solving systems of nonlinear equations, we address this problem in Section 2.7.

2.1 Introduction to ODEs

Coming from classical mechanics first *ordinary differential equations* (*ODEs*) were developed to model mechanical systems. For example, consider a particle with unit mass and denote

$$\begin{aligned} t &: \text{ time,} \\ \mathbf{x}(t) &: \text{ vector of spatial coordinates, i.e. position,} \\ \dot{\mathbf{x}}(t) &: \text{ time derivative of } \mathbf{x}\text{, i.e. velocity.} \end{aligned}$$

Assuming that the velocity is prescribed by a function $\mathbf{u}(t, \mathbf{x})$ at any time t and position $\mathbf{x}$, then the relation

$$\dot{\mathbf{x}} = \mathbf{u}(t, \mathbf{x}), \tag{2.1.1}$$

describes the trajectories of such particle. This relation is *a first order ODE*, since it involves the first derivative only. A specific trajectory of this ODE may be defined by prescribing *an initial condition* for the position, at some time point $t = t_0$, say

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{2.1.2}$$

Relations (2.1.1) and (2.1.2) constitute a so-called *initial value problem* (IVP). Often, one assumes $t_0 = 0$ for the initial time point and this assumption will be used throughout this thesis.

Letting $X \subset \mathbb{R}^N$ be an open, non-empty set and assuming that $\mathbf{x}$ represents a position in $\mathbb{R}^N$, a mapping $\mathbf{u} : X \to \mathbb{R}^N$ is called *a velocity field* (also *a vector field*) on X and has a form

$$\mathbf{u}(\mathbf{x}) := \begin{bmatrix} u_1(x_1, \ldots, x_N) \\ \vdots \\ u_N(x_1, \ldots, x_N) \end{bmatrix}. \tag{2.1.3}$$

The set X is called *the state space*. It will be assumed that $\mathbf{u}$ is continuous, although not necessarily explicitly known (as shown later). If additionally $\mathbf{u}$ is differentiable, the derivative of $\mathbf{u}$ is given by

$$D\mathbf{u}(\mathbf{x}) := \frac{\partial \mathbf{u}}{\partial \mathbf{x}} := \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_N} \\ \vdots & & \\ \frac{\partial u_N}{\partial x_1} & & \frac{\partial u_N}{\partial x_N} \end{bmatrix}. \tag{2.1.4}$$

This derivative is called *the Jacobian matrix* and its determinant $\det[D\mathbf{u}(\mathbf{x})]$ *the Jacobian.* If $\mathbf{u}$ is of the form (2.1.3), i.e. if it does not explicitly depend on time, the corresponding ODE is called *autonomous* and the corresponding IVP is defined by

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}), \\ \mathbf{x}(t_0) = \mathbf{x}_0. \end{cases} \tag{2.1.5}$$

The velocity field may also explicitly depend on time, i.e. it may be of the form $\mathbf{u} := \mathbf{u}(t, \mathbf{x})$. If t takes values from an interval J, we call $J \times X$ *the time-state space.*

It is sometimes useful to denote the solution as

$$\mathbf{x}(t) = \Psi(t; t_0, \mathbf{x}_0), \tag{2.1.6}$$

which emphasizes the dependency of the solution on its initial value. Now the useful concept can be introduced by the following:

- *The orbit* or *trajectory* of the IVP is the curve

$$\{\Psi(t; t_0, \mathbf{x}_0) \mid t \in J\}, \tag{2.1.7}$$

  in the state-space. The corresponding *positive orbit* is obtained by taking the part with $t \geq t_0$.
- *The solution curve* of the IVP is the curve

$$\left\{ \begin{bmatrix} t \\ \Psi(t; t_0, \mathbf{x}_0) \end{bmatrix} \mid t \in J \right\}, \tag{2.1.8}$$

  in the time-state space ($\subset \mathbb{R}^{N+1}$). For $n = 1$ it is simply the graph of x as a function of t.

Points $\tilde{\mathbf{x}} \in X$, where

$$\mathbf{u}(t, \tilde{\mathbf{x}}) = 0, \tag{2.1.9}$$

are of special importance and here referred as *stationary points*. A stationary point is a solution of IVP for any $t_0 \in J$ and the orbit is then contracted on one point.

2.2 Existence, uniqueness and stability of ODE solution; Flow problems

To give conditions for the existence and uniqueness of the ODE solution, we introduce some relevant norms. Unless explicitly stated, we shall use *the Euclidean norm*, which is for $\mathbf{x} \in \mathbb{R}^N$ defined by

$$\|\mathbf{x}\| := \sqrt{\mathbf{x}^T \cdot \mathbf{x}} = \left(\sum_{k=1}^{N} x_k^2 \right)^{1/2} . \tag{2.2.1}$$

For $\mathbf{x}$ with components dependent on $t \in J$, the norm of $\mathbf{x}(\cdot)$ on J is defined as

$$\|\mathbf{x}\|_J := \sup_{t \in J} \|\mathbf{x}(t)\| . \tag{2.2.2}$$

In a similar way we may define the norm of a velocity field $\mathbf{u}(t, \mathbf{x})$ with $(t, \mathbf{x}) \in J \times X$, as

$$\|\mathbf{u}\|_J := \sup_{\substack{t \in J \\ x \in X}} \|\mathbf{u}(t, \mathbf{x})\| . \tag{2.2.3}$$

For matrices $\mathbf{A} \in \mathbb{R}^{N \times N}$ we can associate a matrix norm with any norm defined in $\mathbb{R}^N$ as

$$\|\mathbf{A}\| := \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\| . \tag{2.2.4}$$

To prove the local existence of a solution of the IVP (2.1.1)-(2.1.2), it is sufficient to require that the velocity field is continuous on a domain $J \times X$, containing the point $(t_0, \mathbf{x}_0)$. This result is obtained in the Cauchy-Peano theorem (see [23]), which we give without proof.

THEOREM 2.2.1. *If* $J = [t_0, t_1]$ *and* $V = \{\mathbf{x} \in X \,|\, \|\mathbf{x} - \mathbf{x}_0\| < R\}$ *and for all* $\mathbf{x} \in V$, $\mathbf{u}$ *continuous and* $\|\mathbf{u}(\mathbf{x})\| < M$, *where the numbers* R *and* M *satisfy* $M(t_1 - t_0) \leq R$, *then IVP (2.1.1)-(2.1.2) has a solution.*

The uniqueness of the solution is guaranteed if the velocity field satisfies *the Lipschitz condition*, i.e. if it satisfies the following.

DEFINITION 2.2.2. The velocity field $\mathbf{u}(t, \mathbf{x})$ is Lipschitz continuous on $J \times X$ if there is a constant L, known as *the Lipschitz constant*, such that for all $\mathbf{x}, \mathbf{y} \in X$ and all $t \in J$

$$\|\mathbf{u}(t, \mathbf{y}) - \mathbf{u}(t, \mathbf{x})\| \leq L \, \|\mathbf{y} - \mathbf{x}\| . \tag{2.2.5}$$

The Lipschitz continuity property is stronger than continuity, but weaker than differentiability. It expresses that $\mathbf{u}$ can be bounded by a linear function on X for all $t \in J$. The norm present in (2.2.5) can taken as arbitrary, but in case of Euclidean norm, we can also introduce a closely related, but weaker condition.

DEFINITION 2.2.3. The velocity field $\mathbf{u}(t, \mathbf{x})$ on $J \times X$ is said to satisfy *a one-sided Lipschitz condition* if there exists a number ν, known as *the one-sided Lipschitz constant* of $\mathbf{u}$, such that for all $\mathbf{x}, \mathbf{y} \in X$

$$\langle \mathbf{u}(t, \mathbf{y}) - \mathbf{u}(t, \mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \nu \, \|\mathbf{y} - \mathbf{x}\|^2 , \tag{2.2.6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner-product.

Now we can give the uniqueness properties by the following.

THEOREM 2.2.4. *The IVP (2.1.1)-(2.1.2), with* $\mathbf{u}$ *Lipschitz continuous on some domain* $J \times X$ *which contains* $(t_0, \mathbf{x}_0)$ *has at most one solution.*

THEOREM 2.2.5. *The solution* $\mathbf{x}$ *of IVP (2.1.1)-(2.1.2) with* $\mathbf{u}$ *Lipschitz continuous exists globally, i.e. for all* $t \in \mathbb{R}$, *if* $\mathbf{x}(t)$ *exists for some* t, *which implies*

$$\|\mathbf{x}(t) - \mathbf{x}_0\| \leq M, \tag{2.2.7}$$

for some constant $M > 0$.

THEOREM 2.2.6. *The solution* $\mathbf{x}$ *of IVP (2.1.1)-(2.1.2) exists globally if* $\mathbf{u}$ *is bounded, i.e. if* $\|\mathbf{u}\|_{\mathbb{R}\times\mathbb{R}^N} < \infty$

The behaviour of the ODE solution is closely related to the notion of *stability* of the IVP solution. Among many stability definitions we give the most general one only, the definition of *the total stability.* For that we consider the IVP

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{u}(t, \mathbf{x}), \\ \mathbf{x}(t_0) = \mathbf{x}_0. \end{cases} \tag{2.2.8}$$

and its corresponding perturbed problem

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{u}(t, \mathbf{y}) + \mathbf{v}(t, \mathbf{y}), \\ \mathbf{y}(t_1) = \mathbf{x}(t_1) + \mathbf{z}_1, \end{cases} \tag{2.2.9}$$

where $t_1 \geq t_0$. Assuming that the solution $\mathbf{x}$ of (2.2.8) exists for $t_1 \geq t_0$ the definition reads as follows.

DEFINITION 2.2.7. The solution $\mathbf{x}$ of (2.2.8) is *totally stable* if, for every $\epsilon > 0$ and every $t_1 > t_0$, there exists $\delta(\epsilon, t_1)$ such that

$$\{\|\mathbf{z}_1\| < \delta(\epsilon, t_1) \text{ and } R_1 < \delta(\epsilon, t_1)\} \Rightarrow \{\|\mathbf{y}(t) - \mathbf{x}(t)\| < \epsilon, \text{ for all } t \geq t_1\}, \tag{2.2.10}$$

where $\mathbf{y}$ is the solution of (2.2.9) and $R_1 := \sup_{t \geq t_1} \|\mathbf{v}(t, \mathbf{x}(t))\|$.

The stronger form of stability is the situation where δ does not depend on t_1. Then, $\mathbf{x}$ is called *uniformly totally stable.*

In this thesis the main aspect is on the so-called *flow problem*, where the initial value is defined as a set of points (instead of a single point as in IVP). It will be assumed that $\mathbf{u} : U \to \mathbb{R}^N$ is a smooth function, defined on some subset $U \subseteq \mathbb{R}^N$. We say that $\mathbf{u}$ generates *a flow* $\mathbf{I}^t : U \to \mathbb{R}^N$, where $\mathbf{I}^t(\mathbf{x}) = \mathbf{I}(\mathbf{x}, t)$ is a smooth function defined for all $\mathbf{x} \in U$ and $t \in J = [t_0, t_1] \subseteq \mathbb{R}$, and $\mathbf{I}$ satisfies ODE (2.1.1) in the sense that

$$\frac{d}{dt}\, \mathbf{I}(\mathbf{x}, t)|_{t=\tau} = \mathbf{u}(\mathbf{I}(\mathbf{x}, \tau)), \tag{2.2.11}$$

for all $\mathbf{x} \in U$ and $\tau \in J$. Since we are mainly interested in problems where $\mathbf{u}$ is autonomous, for which the corresponding autonomous system is invariant with respect to translations in time, we can assume, without loss of generality, that $t_0 = 0$.

Considering a particular initial value

$$\mathbf{x}(0) = \mathbf{x}^0 \in U., \tag{2.2.12}$$

the corresponding solution $\mathbf{x}(t)$ can be expressed as

$$\mathbf{x}(t) = \mathbf{I}\left(\mathbf{x}^0, t\right), \tag{2.2.13}$$

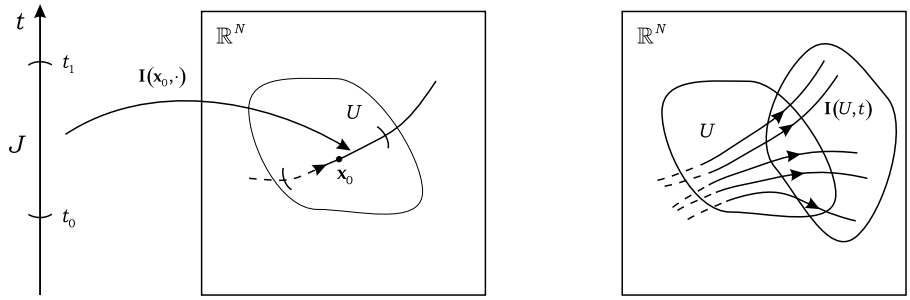


FIGURE 2.2.1. A solution curve and the flow

where

$$\mathbf{I}\left(\mathbf{x}^0, 0\right) = \mathbf{x}^0. \tag{2.2.14}$$

Here $\mathbf{I}\left(\mathbf{x}^0, \cdot\right) : J \to \mathbb{R}^N$ defines *a solution curve* of the differential equation (2.1.1) based at $\mathbf{x}^0$ and it represents a solution of the IVP. The behaviour of the family of such curves in time for all $\mathbf{x} \in U$, determines the time evolution of the flow, see Figure 2.2.1. The set of initial values U can also be defined by the notion of the flow as $\mathbf{I}(\mathbf{x}, 0)$. The following notation will also be used in this thesis

$$\mathbf{I}^0 := \mathbf{I}(\mathbf{x}, 0). \tag{2.2.15}$$

Now, we can define a flow problem as

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{u}(t, \mathbf{x}), \\ \mathbf{x}(0) \in \mathbf{I}^0. \end{cases} \tag{2.2.16}$$

Again if $\mathbf{u}$ does not explicitly depend on t, the flow problem is called *autonomous*. The conditions for the local existence and uniqueness of the solution can be related to those given in Section 2.2, i.e. to the property of Lipschitz continuity. Hence, throughout this thesis, we will consider flow problems where $\mathbf{u}$ is Lipschitz continuous. For such autonomous systems there is an important property which relates the flow solutions to their initial conditions. Note that a similar property also holds for the corresponding IVP.

THEOREM 2.2.8. *Let* $U \subseteq \mathbb{R}^N$ *be open and suppose* $\mathbf{u} : U \to \mathbb{R}^N$ *has a Lipschitz constant* L *and let* $\mathbf{x}(t)$ *and* $\mathbf{y}(t)$ *be solutions of* $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x})$ *on the closed interval* $[t_0, t_1]$. *Then, for all* $t \in [t_0, t_1]$,

$$\|\mathbf{y}(t) - \mathbf{x}(t)\| \leq e^{L(t-t_0)} \|\mathbf{y}_0 - \mathbf{x}_0\|. \tag{2.2.17}$$

A special class of interest in this thesis are flow problems where the velocity field is not known explicitly, i.e. when it is given discretely only. For example, the velocity field $\mathbf{u}$ may be defined by a set of pairs $\{(\mathbf{x}_k, \mathbf{u}_k)\}_{k=0}^n$, where $\mathbf{x}_k$ represents the spatial point in which $\mathbf{u}(\mathbf{x}_k) = \mathbf{u}_k$ value is known. However, if $\mathbf{u}$ is Lipschitz continuous, the aforementioned theory holds. The major difference (or difficulty in a way) is in the implementation of numerical methods for solving the flow problem. This is the main topic of the last section of this chapter and indeed in the rest of this thesis.

2.3 Numerical methods for solving ODEs

In the foregoing sections we assumed that both $\mathbf{x}$ and t are continuous. Applying a numerical method for solving an ODE involves *the time discretisation*, i.e. the use of difference quotients instead of differential ones. This leads to so-called *difference equations*. Here, instead of the continuous variable t we use the index set $J_i = \{0, 1, \ldots, n_i\}$, $n_i \in \mathbb{N}$. Let a sequence $\mathbf{f}^i(\mathbf{x}) : X \to \mathbb{R}^N$ be given, where $i \in J_i$ and $X \in \mathbb{R}^N$. Then a first order difference equation has the form

$$\mathbf{x}^{i+1} = \mathbf{f}^i\left(\mathbf{x}^i\right), \quad \mathbf{x} \in X, \quad i \in J_i. \tag{2.3.1}$$

By defining

$$\mathbf{x}^0 = \mathbf{x}_0, \quad \mathbf{x}_0 \in \mathbb{R}^N, \tag{2.3.2}$$

or

$$\mathbf{x}^0 \in \mathbf{I}^0, \quad \mathbf{I}^0 \subseteq \mathbb{R}^N. \tag{2.3.3}$$

we obtain a discrete analogues of an IVP and the flow problems respectively.

Before we introduce some numerical methods, we first introduce the time discretisation and its relation to difference equations. For simplicity reasons we restrict ourselves to scalar equations. Let $J = \left\{t^i\right\}_{i=0}^{n_i}$, $n_i \in \mathbb{N}$ be a set of time points, which spans a time interval of interest, say $[t_0, t_f]$. This set determines in which points we want to approximate the solution values $x\left(t^i\right)$, $i = 0, \ldots, n_i$. The set J we call *the time-grid* and points t^i time-grid points (or time levels). Because all quantities are calculated at the time-grid points only, we have to replace the differential operator by a differential quotient. One (simple) way of discretising the ODE $\dot{x} = u(t, x)$ reads

$$\frac{x\left(t^i + h^i\right) - x\left(t^i\right)}{h^i} \doteq u\left(t^i, x\left(t^i\right)\right), \quad h^i := t^{i+1} - t^i. \tag{2.3.4}$$

We call h^i *a time step* at the time point t^i. If the set J is equidistant, i.e. if $t^{i+1} - t^i = \mathrm{const} = h$, we have a fixed time step integration and the superscript i will be omitted.

Since we know $x\left(t^0\right)$, we may use (2.3.4) to compute an approximation of the solution. An alternative way is to rewrite the ODE as an integral equation, i.e.

$$x(t) = x\left(t_0\right) + \int_{t_0}^{t} u\left(\tau, x(\tau)\right)\, d\tau. \tag{2.3.5}$$

Applying this on the interval $\left[t^i, t^{i+1}\right]$, we have

$$x\left(t^{i+1}\right) = x\left(t^i\right) + \int_{t^i}^{t^{i+1}} u\left(\tau, x(\tau)\right)\, d\tau. \tag{2.3.6}$$

Now, we can discretise the integral by so-called *quadrature formulae*. The simplest way to approximate the integral on the right-hand side of (2.3.6) is to approximate $u\left(\tau, x(\tau)\right) \doteq u\left(t^i, x\left(t^i\right)\right)$, $\tau \in \left[t^i, t^{i+1}\right]$, which gives

$$x\left(t^{i+1}\right) \doteq x\left(t^i\right) + h^i\, u\left(t^i, x\left(t^i\right)\right), \tag{2.3.7}$$

which is basically the famous *Euler Forward (EF) formula.* We denote x^i to be an approximation (i.e. *the numerical solution*) of $x\left(t^i\right)$ at the time point t^i.

In general, one-step methods are given by

$$x^{i+1} = x^i + h^i\, \Phi\left(t^i, x^i, x^{i+1}, h^i\right), \quad i = 0, \ldots, n_j - 1. \tag{2.3.8}$$

If Φ does not depend on x^{i+1}, we have *an explicit one-step method*. Otherwise, (2.3.8) is called *an implicit one-step method*. If we use $u(\tau, x(\tau)) \doteq u\left(t^{i+1}, x\left(t^{i+1}\right)\right)$, $\tau \in \left[t^i, t^{i+1}\right]$ for approximating (2.3.6), we obtain *the Euler Backward (EB) method*

$$x^{i+1} = x^i + h^i u\left(t^{i+1}, x^{i+1}\right), \quad i = 0, \ldots, n_j - 1, \tag{2.3.9}$$

which will be of special interest in this work. Higher order methods, both one-step and multistep ones, will be introduced is Sections 2.5 and 2.6.

2.4 Stiff problems

Numerical methods of interest in this thesis are implicit, such as e.g. the Euler Backward method introduced in the previous section. Being much more involved than explicit methods, implicit methods are typically used for solving the so-called *stiff problems* only. Historically the notion of stiffness was introduced in [13], which reads: "Stiff equations are equations where certain implicit methods perform better, usually tremendously better, than explicit ones." Here we will address stiffness in the sense of multiple time scales present in the problem. We call a problem stiff if the presence of faster time scales requires much smaller time-steps than needed for accuracy for the relevant slower time scales; in particular for explicit methods.

One way to find out whether a method is acceptable for solving a stiff problem is by introducing *the stability function* ψ. It is defined by considering *the linear test problem*

$$\dot{x} = \lambda x, \quad \lambda \in \mathbb{C}, \tag{2.4.1}$$

and applying a numerical method. For one-step methods this directly leads to

$$x^{i+1} = \psi(h\lambda)\, x^i, \tag{2.4.2}$$

where ψ depends on the method and the term $h\lambda$ turns out to be an invariant quantity (i.e. h and λ always appear jointly). Typically, one substitutes the quantity $h\lambda$ by a single complex variable, say $z := h\lambda$. By analysing the behaviour of $\psi(z)$ for $z \in \mathbb{C}$, one can determine whether or not a method will be acceptable.

The stability function of the already mentioned EF, defined by (2.3.7), reads

$$\psi(z) = 1 + z. \tag{2.4.3}$$

On the other hand, applying the EB method (2.3.9), one can easily find

$$\psi(z) = \frac{1}{1 - z}. \tag{2.4.4}$$

The more general result, namely for the Runge-Kutta methods (introduced later), is given by the following theorem.

THEOREM 2.4.1. *The stability function of the s—stage Runge-Kutta method satisfies*

$$\psi(z) = \frac{\det\left(\mathbf{I}_s - z\,\mathbf{A} + z\,\mathbf{e}\,\mathbf{b}^T\right)}{\det\left(\mathbf{I}_s - z\,\mathbf{A}\right)}. \tag{2.4.5}$$

where $z := h\lambda$, $\mathbf{e} := \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T$, $\mathbf{I}_s$ *represents identity matrix of* s*-th order, and* $\mathbf{A}$ *and* $\mathbf{b}$ *are defined by the Butcher matrix.*

The stability function is used for defining the so-called *stability domain* S, by the following

$$S := \{z \in \mathbb{C} \mid |\psi(z)| \leq 1\}. \tag{2.4.6}$$

Obviously S should be as large as possible. It can be important that S includes the whole negative (complex) half-plane, i.e. to cover the stability region of dynamical systems. This property is called *A-stability* and formally given by the following definition.

DEFINITION 2.4.2. A method whose stability domain satisfies

$$C_- := \{z \mid \Re\{z\} \leq 0\} \subset S, \tag{2.4.7}$$

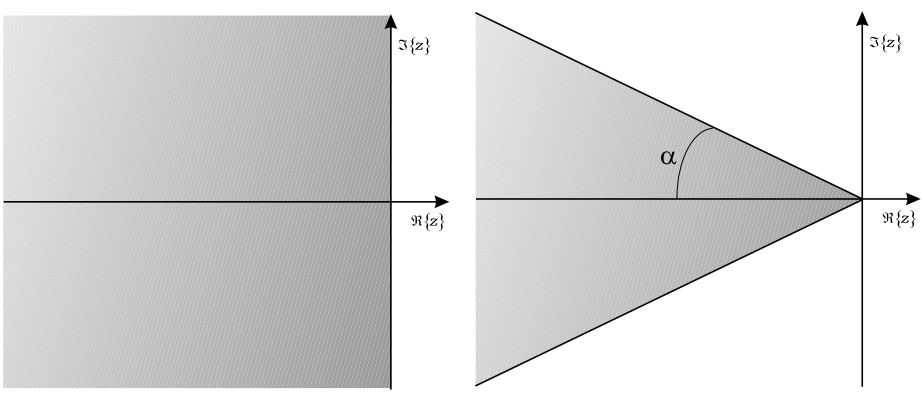
is called *A-stable*.

Unfortunately, all A-stable methods are implicit. Practically all explicit methods have a relatively small stability region, making them inappropriate for solving stiff problems. Moreover, not all implicit methods are A-stable and it can turn out that this property can be too strong in general and discard some methods which are not bad at all. Hence, it is useful to define a little weaker stability property, called $A(\alpha)$-*stability*, by the following.

DEFINITION 2.4.3. A method is said to be $A(\alpha)$-*stable* if

$$C(\alpha) := \{z \mid |\arg(-z)| \leq \alpha,\ \alpha \in [0, \pi/2]\ ,\ z \neq 0\} \subset S. \tag{2.4.8}$$

In Figure 2.4.1 domains C_- and $C(\alpha)$ are shown as shaded areas respectively.



(a) A—stability (b) $A(\alpha)$—stability

FIGURE 2.4.1. Stability domains

THEOREM 2.4.4. *Explicit numerical methods cannot be* $A(\alpha)$*-stable.*

PROOF. We consider Runge-Kutta and linear multistep methods only. For explicit $s-$stage Runge-Kutta methods, the matrix $\mathbf{A}$ is lower-triangular. Hence the denominator of (2.4.5) is

$$\det(\mathbf{I}_s - z\,\mathbf{A}) = 1.$$

This means that the stability function is just a polynomial of the s-th degree. Since polynomials cannot satisfy (2.4.8), it is clear that explicit Runge-Kutta methods cannot be $A(\alpha)$-stable. A more extensive proof can be found in [19]. For the proof concerning explicit linear multistep methods, see [53], where it is shown that these methods cannot be $A(0)$-stable. □

COROLLARY 2.4.5. *For solving stiff problems only implicit methods can efficiently be used (see e.g. [15] and [24]).*

In some applications it may be of importance that the method should have a property that $|\psi(z)|$ is much smaller than 1 for $z \to \infty$. This is basically the concept of another stability property, called L-*stability.*

DEFINITION 2.4.6. A method is called L-*stable* if it is A-stable and if in addition

$$\lim_{z \to -\infty} \psi(z) = 0. \tag{2.4.9}$$

Clearly, the EB method is one example of L-stable methods. In the following sections we will introduce some of the higher order methods which also satisfy some of these stability properties.

2.5 Linear multistep methods

In Section 2.3 we have seen that an ODE can be transformed into an integral equation (2.3.5). Applying this to the interval $\left[t^i, t^{i+1}\right]$, we have obtained the integral equation (2.3.6), from which one can be obtain one-step methods (see the following section). If we increase the integration interval to $\left[t^i - \tilde{t}, t^{i+1}\right]$, the corresponding equation reads

$$x\left(t^{i+1}\right) = x\left(t^i - \tilde{t}\right) + \int_{t^i - \tilde{t}}^{t^{i+1}} u\left(\tau, x(\tau)\right)\, d\tau. \tag{2.5.1}$$

Again, by applying some quadrature rule, one can approximate the integral on the right-hand side of (2.5.1). We can choose $\tilde{t}$ such that the interpolating quadrature involves more than two time-grid points t^j, $j = i+1, i, \ldots$ This leads to multistep formulae, which can be used for solving IVPs and the flow problems. For example, if we use t^{i+1}, t^i and t^{i-1} and the second order equispaced polynomial approximation (i.e. *Simpson's rule*), we obtain the following (implicit) formula

$$x^{i+1} = x^{i-1} + h\left(\frac{1}{3}u\left(t^{i+1}, x^{i+1}\right) + \frac{4}{3}u\left(t^i, x^i\right) + \frac{1}{3}u\left(t^{i-1}, x^{i-1}\right)\right). \tag{2.5.2}$$

One should note that a similar procedure for using t^i, t^{i-1} and t^{i-2} would yield an explicit formula.

A *linear multistep method (LMM)* is generally given by

$$\sum_{l=0}^{m} \alpha_l x^{i-l+1} = h \sum_{l=0}^{m} \beta_l u\left(t^{i-l+1}, x^{i-l+1}\right). \tag{2.5.3}$$

It is basically a m-th order difference equation and for $\beta_0 = 0$ it is *explicit.* Otherwise the LMM method is *implicit.* In this thesis we will consider *implicit linear multistep methods* (*ILMMs*) only. In particular the focus will be on the so-called BDF methods which we address separately.

The implementation of ILMMs leads to the nonlinear system of equations, which needs to be solved at every time level. The existence and uniqueness of the solution is not guaranteed in the general case. A sufficient condition for the general vectorial case, to have an unique solution is given by the following theorem, given in [24].

THEOREM 2.5.1. *Let* $\mathbf{u}$ *be continuously differentiable and satisfy the one-sided Lipschitz condition*

$$\langle \mathbf{u}(\mathbf{y}) - \mathbf{u}(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \nu \, \|\mathbf{y} - \mathbf{x}\|^2. \tag{2.5.4}$$

If

$$h\nu \leq \frac{\alpha_0}{\beta_0}, \tag{2.5.5}$$

the nonlinear system of equations $\alpha_0 \, \mathbf{x}^{i+1} - h \, \beta_0 \, \mathbf{u}\left(\mathbf{x}^{i+1}\right) = \mathbf{s}^i$, *has a unique solution.*

2.5.1 BDF methods

There is another important source for constructing multistep methods. The underlying idea comes from interpolating x and differentiating at one of the time-grid points. Assuming that $x^{i-m+1}, \ldots, x^i$ are known approximations of the IVP solution, one can construct a polynomial, say $q(t)$, which interpolates the points $\left\{\left(t^{i+1-r}, x^{i+1-r}\right)\right\}_{r=0}^{m}$. The unknown value x^{i+1} can now be determined in such a way that the polynomial $q(t)$ satisfies the differential equation at (at least) one time-grid point, i.e.

$$\frac{dq}{dt}\left(t^{i+1-r}\right) = u\left(t^{i+1-r}, x^{i+1-r}\right). \tag{2.5.6}$$

For $r = 1$ the formula is explicit. Then for $m = 1$ and $m = 2$ the corresponding formula is equivalent to the EF method and *the (explicit) midpoint rule*, which reads

$$\frac{1}{h}\left(\frac{1}{2}\, x^{i+1} - \frac{1}{2}\, x^{i-1}\right) = u\left(t^i, x^i\right). \tag{2.5.7}$$

Unfortunately, for $m \geq 3$ the formula is unstable and therefore useless. However, if we choose $r = 0$ in (2.5.6), we find implicit formulae of the following form

$$\sum_{l=0}^{m} \alpha_l \, x^{i-l+1} = h \, u\left(t^{i+1}, x^{i+1}\right). \tag{2.5.8}$$

This formula represents the so-called *BDF (Backward Differences Formula) method*. The BDF methods were introduced in [13] and became frequently used methods (especially after [21] appeared) for solving stiff problems. The reason is, of course, that they have favourable stability properties. For example, BDF1 (which is in fact the EB method) and BDF2 methods are A-stable and BDF3-6 are $A(\alpha)$-stable, see Table 2.5.1. For $m \geq 7$ these methods are unstable.

m	1	2	3	4	5	6
α	90°	90°	86.03°	73.35°	51.84°	17.84°

TABLE 2.5.1. $A(\alpha)$−stability of BDF methods

The coefficient values for the BDF methods are shown in Table 2.5.2.

m	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	1	-1					
2	$\frac{3}{2}$	-2	$\frac{1}{2}$				
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$			
4	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$		
5	$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$	
6	$\frac{147}{60}$	-6	$\frac{15}{2}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$

TABLE 2.5.2. BDF methods

2.5.2 Local error (consistency); stability

One of the basic requirements for every numerical method is that the difference formula is consistent with the differential equation. In other words, *consistency* means that the formula reproduces the ODE if $h \to 0$. This property is closely related to the notion of *the local discretisation error*, say d. The local discretisation error is defined as a difference between the exact value $x\left(t^{i+1}\right)$ and its approximation x^{i+1}, divided by h, and assuming exact input data given, i.e.

$$d_h\left(x\left(t^{i+1}\right)\right) := \frac{1}{h}\left[x\left(t^{i+1}\right) - x^{i+1}\right], \quad x^{i-l+1} = x\left(t^{i-l+1}\right),\ l = 1,\dots,m. \tag{2.5.9}$$

If $d_h(\cdot) = O\left(h^p\right)$, $p \geq 1$, we call the LMM *consistent of order* p. For a sufficiently smooth solution the local discretisation error for the general LMM can be expressed as

$$d_h\left(x\left(t^{i+1}\right)\right) = C\,h^p\,x^{(p+1)}\left(t^{i+1}\right) + O\left(h^{p+1}\right), \tag{2.5.10}$$

where C is called *the error constant*, which can be computed by

$$C = \frac{1}{\sigma(1)\,(p+1)!}\left[\sum_{l=0}^{m}\alpha_l\,l^{p+1} - (p+1)\sum_{l=0}^{m}\beta_l\,l^p\right]. \tag{2.5.11}$$

In previous subsection we mentioned that BDF methods have favourable stability properties. To determine that, one needs to find the stability function which relates numerical solutions of the linear test problem at consecutive time levels, which is straightforward for one-step methods. For $m-$step LMM the homogeneous difference equation generates m basis solutions, each having its own growth behaviour as a function of $h\lambda$. Given $h\lambda$ we can represent them all, once we have the roots $\omega_l(h\lambda)$, $l = 0,\dots,m$ of the characteristic equation

$$\rho(\omega) - h\lambda\,\sigma(\omega) := \sum_{l=0}^{m}\left(\alpha_l - h\lambda\,\beta_l\right)\omega^{m-l} = 0. \tag{2.5.12}$$

The stability condition of the general LMM method is related to the condition of the polynomial $\rho(\omega)$, as given by the following definition.

DEFINITION 2.5.2. An LMM is called root stable if all roots $\rho(\omega)$ are at most 1 in modulus and those which have modulus 1 are simple.

Now, the stability domain of an LMM can be defined by means of roots ω_l, $l = 0, \dots, m$, of the characteristic equation (2.5.12), as

$$S := \left\{ z \in \mathbb{C}, \quad \begin{array}{c} |\omega_l(z)| \le 1, \ j = 0, \dots, m \\ |\omega_l(z)| < 1, \text{ for multiple roots} \end{array} \right\}. \tag{2.5.13}$$

In a similar way as for one-step methods, one can determine whether the LMM satisfies some of the stability properties such as A-stability or $A(\alpha)$-stability (see Definitions 2.4.2 and 2.4.3).

2.6 Runge-Kutta methods

The most important class of one-step methods are *Runge-Kutta (RK) methods*, which were already introduced in Section 2.3. By applying some quadrature formula to approximate the integral in (2.3.6), we can obtain

$$x\left(t^{i+1}\right) \doteq x\left(t^{i}\right) + h \sum_{l=1}^{s} b_l \, u\left(t^{il}, x\left(t^{il}\right)\right), \tag{2.6.1}$$

where h is the time step and $t^{il} := t^i + c_l h$, $0 \le c_l \le 1$, i.e. the nodes on $\left[t^i, t^{i+1}\right]$. To find unknown $x\left(t^{il}\right)$ one can apply another quadrature formula on the interval $\left[t^i, t^{il}\right]$ using a subset of the nodes t^{il}, $l = 1, \dots, s$. Hence we obtain

$$x\left(t^{il}\right) \doteq x\left(t^{i}\right) + h \sum_{q=1}^{s} a_{lq} \, u\left(t^{il}, x\left(t^{il}\right)\right), \quad l = 1, \dots, s, \tag{2.6.2}$$

which together with (2.6.1) defines a Runge-Kutta method with s *stages*. Frequently used compact notation is the so-called *Butcher matrix*

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \\ c_s & a_{s1} & & a_{ss} \\ \hline & b_1 & & b_s \end{array},$$

denoted for short

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array}.$$

The RK method is called *explicit* if $a_{lq} = 0$, $q \ge l$. Otherwise we have an *implicit Runge-Kutta (IRK)* method. A special subclass of implicit methods are *diagonally implicit Runge-Kutta (DIRK)* methods for which $a_{lq} = 0$, $q \ge l + 1$.

2.6.1 Local error (consistency); stability

A one-step method, such as an RK method, can be represented by the following formula

$$x^{i+1} = x^i + h\,\Phi\left(t^i, x^i, x^{i+1}, h\right). \tag{2.6.3}$$

We may use this to define *the local discretisation error*, say d, as a residual divided by h, when we substitute the exact solution in a method, i.e.

$$d_h\left(x\left(t^{i+1}\right)\right) := \left[x\left(t^{i+1}\right) - x\left(t^i\right) - h\,\Phi\left(t^i, x\left(t^i\right), x\left(t^{i+1}\right), h\right)\right]/h. \tag{2.6.4}$$

Like for the LMM methods, *the consistency* of the method means that d_h approaches zero when $h \to 0$. We call p *the consistency order* if $d_h\left(x\left(t^{i+1}\right)\right) = O\left(h^p\right)$, for p as large as possible.

The already mentioned stability properties (A, $A(\alpha)$, L) of the RK methods can be determined by finding the stability function $\psi(h\lambda)$, which was introduced in Section 2.4. Using $\psi(z)$, obtained by (2.4.5), one can analyse whether a particular method satisfies certain stability properties, like A, $A(\alpha)$ or L-stability. As shown in [19], explicit RK methods do not have these properties and they are not of interest in this thesis. Hence, in the following subsection, we will introduce some IRK methods which are at least A-stable.

The aforementioned stability analysis is related to linear systems and not necessarily appropriate for general nonlinear problems. However, there is a generalisation of A-stability to nonlinear problems, related to the contractivity condition of a nonlinear ODE (see [9]), that satisfies the one-sided Lipschitz condition, see (2.2.6). Whenever the one-sided Lipschitz constant ν is non-positive in (2.2.6), the distance between any two solutions of (2.1.1) is a non-increasing function of t. Naturally, the numerical solution should have the same behaviour and this property is called B-*stability* and defined by the following.

DEFINITION 2.6.1. A Runge-Kutta method is called B-*stable* if the *contractivity condition*

$$\langle \mathbf{u}(t, \mathbf{y}) - \mathbf{u}(t, \mathbf{x}), \mathbf{y} - \mathbf{x}\rangle \leq 0, \tag{2.6.5}$$

implies for all $h \geq 0$

$$\left\|\mathbf{y}^{i+1} - \mathbf{x}^{i+1}\right\| \leq \left\|\mathbf{y}^i - \mathbf{x}^i\right\|, \tag{2.6.6}$$

where $\mathbf{y}^{i+1}$ and $\mathbf{x}^{i+1}$ are the numerical solutions after one step starting with initial values $\mathbf{y}^i$ and $\mathbf{x}^i$ respectively. In the following subsection we introduce some of the B-stable IRK methods.

2.6.2 IRK methods; Gauss, Radau, Lobatto

The construction of IRK methods strongly relies on the simplifying assumptions

$$\begin{aligned}
&B(p): && \textstyle\sum_{l=1}^{s} b_l\, c_l^{r-1} = \frac{1}{r}, && r = 1, \ldots, p;\\
&C(\eta): && \textstyle\sum_{q=1}^{s} a_{lq}\, c_q^{r-1} = \frac{c_l^r}{r}, && l = 1, \ldots, s,\ r = 1, \ldots, \eta;\\
&D(\zeta): && \textstyle\sum_{l=1}^{s} b_l\, c_l^{r-1}\, a_{lq} = \frac{b_q}{r}\left(1 - c_q^r\right) && q = 1, \ldots, s,\ r = 1, \ldots, \zeta;
\end{aligned} \tag{2.6.7}$$

Condition $B(p)$ simply means that the quadrature formula (b_l, c_l) is of order p. The other two conditions are important for determining the consistency order of the IRK method, as shown by the following.

THEOREM 2.6.2. *If the coefficients* b_l, c_l *and* a_{lq} *of an RK method satisfy* $B(p)$, $C(\eta)$ *and* $D(\zeta)$, *with* $p \le \eta + \zeta + 1$ *and* $p \le 2q + 2$, *then the method is of order* p.

2.6.2.1 Gauss method

This method (also known as the Kuntzmann-Butcher method) is based on the Gaussian quadrature formula, where $c_1, \dots, c_s$ are the zeros of the shifted Legendre polynomial of degree s, defined by

$$\frac{d^s}{dx^s}\left(x^s\,(x-1)^s\right). \tag{2.6.8}$$

The Gauss method with s stages is of order $2s$ and it has the maximal order of all IRK methods. Also it is A-stable. One and two-stage methods are shown in Table 2.6.1. Higher order methods can be found in [24], which will also hold for other methods presented here.

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}, \qquad \begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

TABLE 2.6.1. Gauss methods of order 2 and 4.

2.6.2.2 Radau IA and Radau IIA methods

These methods are obtained by using the Radau quadrature formula. Instead of (2.6.8), the coefficients c_l, $l = 1, \dots, s$ are zeros of

$$\text{I}: \quad \frac{d^{s-1}}{dx^{s-1}}\left(x^s\,(x-1)^{s-1}\right), \quad \text{(Radau left)}, \tag{2.6.9}$$

$$\text{II}: \quad \frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}\,(x-1)^{s}\right), \quad \text{(Radau right)}. \tag{2.6.10}$$

The weights b_l, $l = 1, \dots, s$ are chosen so that the quadrature formula satisfies $B(2s-1)$. There is a variety of possibilities how to choose the coefficients a_{lq}, $l, q = 1, \dots, s$, but not all of them are equally effective. We mention the most important ones only. The *Radau IA* method is a method of type I, where a_{lq}'s are defined by condition $D(s)$, which is uniquely possible. One and two-stage methods are shown in Table 2.6.2.

$$\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array}, \qquad \begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}.$$

TABLE 2.6.2. Radau IA methods of order 1 and 3.

By employing (2.6.10) for c_l's and imposing condition $C(s)$, we obtain the so-called *Radau IIA* method. The first two members are shown in Table 2.6.3. Both Radau IA and Radau IIA methods are of order $2s-1$ and L-stable.

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}, \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}.$$

TABLE 2.6.3. Radau IIA methods of order 1 and 3.

2.6.2.3 Lobatto IIIA, IIIB and IIIC methods

By choosing the Lobatto quadrature formula, we obtain the so-called type III methods, for which c_l's are zeros of the polynomial

$$\text{III}: \quad \frac{d^{s-2}}{dx^{s-2}}\left(x^{s-1}\,(x-1)^{s-1}\right). \tag{2.6.11}$$

Here the weights b_l's are determined by requiring the condition $B(2s-s)$ to be satisfied. Choosing a_{lq}'s such that $C(s)$ hold, we obtain *the Lobatto IIIA* method. For *the Lobatto IIIB* method we impose $D(s)$. Finally, by letting

$$a_{l1} = b_1, \quad l = 1, \ldots, s, \tag{2.6.12}$$

and determining the remaining a_{lq}'s by $C(s-1)$, we obtain *the Lobatto IIIC* method. The order of all Lobatto methods is $2s-2$. They are all A-stable and, moreover, the Lobatto IIIC method is also L-stable. Two and three-stage members of Lobatto methods are shown in Tables 2.6.4-2.6.6.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

TABLE 2.6.4. Lobatto IIIA methods of order 2 and 4.

$$\begin{array}{c|cc} 0 & \frac{1}{2} & 0 \\ 1 & \frac{1}{2} & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

TABLE 2.6.5. Lobatto IIIB methods of order 2 and 4.

Considering nonlinear stability properties, i.e. B-stability, one can find that not all of the introduced IRK methods are B-stable. To find this, one can use the following (cf. [24]).

$$\begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

TABLE 2.6.6. Lobatto IIIC methods of order 2 and 4.

THEOREM 2.6.3. *If the coefficients of a IRK method satisfy*

(i) $b_l \geq 0, \quad l = 1, \ldots, s,$

(ii) $M = [m_{lq}]_{l,q=1}^{s} = [b_l\, a_{lq} + b_q\, a_{ql} - b_l\, b_q]_{l,q=1}^{s}$ *is non-negative definite,*

then the method is B*-stable.*

Applying this for the introduced methods, we find that the Lobatto IIIA and Lobatto IIIB methods do not satisfy condition *(ii)* and therefore they are not B-stable methods. To summarise properties of all interesting methods, the order and the stability properties are shown in Table 2.6.7.

IRK method	Order	Linear Stab. Property	Nonlinear Stab. Property
Gauss	$2s$	A−stability	B−stability
Radau IA, IIA	$2s-1$	L−stability	B−stability
Lobatto IIIA, IIIB	$2s-2$	A−stability	-
Lobatto IIIC	$2s-2$	L−stability	B−stability

TABLE 2.6.7. Properties of some IRK methods.

2.6.3 DIRK methods

As we shall see in the following section, the implementation of the fully implicit IRK can be very involved and computationally expensive. One way to (partially) overcome this problem is the implementation of DIRK methods. An especially interesting class are the so-called *singly diagonally implicit RK (SDIRK)* methods, defined by the following Butcher matrix

$$\begin{array}{c|cccc} c_1 & \gamma & & & \\ c_2 & a_{21} & \gamma & & \\ \vdots & \vdots & & & \\ c_s & a_{s1} & a_{s2} & \cdots & \gamma \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}.$$

The main reason for employing such a method is due to the easier implementation and yet still having a method which is at least A-stable. However, the A-stability region strongly depends on the value of γ. The value of γ can be chosen to optimise the consistency order or the stability requirement of the method, see Table 2.6.8. For example, if the γ value is chosen such that the order of SDIRK is $p = s + 1$, there are only few low order methods which are still A-stable, see Table 2.6.8. However, by decreasing the order of the method it is possible to find higher order methods which are even L-stable, see Table 2.6.9.

s	A−stability	A−stability and $p = s + 1$
1	$1/2 \leq \gamma < \infty$	$\gamma = 1/2$
2	$1/4 \leq \gamma < \infty$	$\gamma = \left(3 + \sqrt{3}\right)/6$
3	$1/3 \leq \gamma \leq 1.68856$	$\gamma = 1.68856$
4	$0.39434 \leq \gamma \leq 1.28058$	-

TABLE 2.6.8. A−stability for SDIRK with $p \geq s$.

s	L−stability	L−stability and $p = s$
2	$\left(2 - \sqrt{2}\right)/2 \leq \gamma \leq \left(2 + \sqrt{2}\right)/2$	$\gamma = \left(2 \pm \sqrt{2}\right)/2$
3	$0.18043 \leq \gamma \leq 2.1856$	$\gamma = 0.43587$
⋮		
8	$0.15666 \leq \gamma \leq 0.23437$	$\gamma = 0.23437$

TABLE 2.6.9. L−stability for SDIRK with $p \geq s - 1$.

For details about constructing SDIRK methods see [24]. As an example of these methods we introduce one three-stage SDIRK method which is A-stable and of order $s + 1 = 4$, see Table 2.6.10.

$$\begin{array}{c|ccc} \gamma & \gamma & & \\ \frac{1}{2} & \frac{1}{2} - \gamma & \gamma & \\ 1 - \gamma & 2\gamma & 1 - 4\gamma & \gamma \\ \hline & \omega & 1 - 2\omega & \omega \end{array}, \qquad \begin{array}{l} \gamma = \frac{1}{\sqrt{3}} \cos\left(\frac{\pi}{18}\right) + \frac{1}{2}, \\ \omega = \frac{1}{6(2\gamma - 1)^2}. \end{array}$$

TABLE 2.6.10. Three-stage, A−stable SDIRK method ($p = 4$).

2.7 Implementation of the implicit numerical methods

As shown in previous sections, implicit numerical methods possess better stability properties than explicit ones, which is the main reason for their usage in solving stiff ODEs. However, all implicit methods introduce a nonlinear system of equations, which needs to be solved at every time step. For $\mathbf{x} \in \mathbb{R}^N$ the corresponding system of equations is $N \times N$ for LMMs and even $sN \times sN$ for IRK methods, where s is the number of stages. One should note that the solution does not necessarily exist in general. Hence we start with conditions for the existence and uniqueness of the (implicit) numerical solution. We address both IRK and LMM methods.

2.7.1 Existence and uniqueness of the numerical solution

Consider an IRK method, applied to an ODE (2.1.1). Let us rewrite (2.6.2) for the general N-dimensional ODE by replacing the exact (inner-stage) values with their numerical approximations. The system is then

$$\mathbf{x}^{il} = \mathbf{x}^i + h \sum_{q=1}^{s} a_{lq}\, \mathbf{u}\left(t^{il}, \mathbf{x}^{il}\right), \quad l = 1, \dots, s, \tag{2.7.1}$$

where $\mathbf{x}^{il}$ represents the approximation of $\mathbf{x}\left(t^{il}\right)$. This system is clearly $sN \times sN$ and the existence of an unique solution is guaranteed if

$$h < \frac{1}{L \max_l \sum_{q=1}^{s} |a_{lq}|}, \tag{2.7.2}$$

where L is the Lipschitz constant, see [23]. For stiff problems, where L is typically large, this condition imposes severe constraints for the time step size and, thus, does not give a satisfactory result. However, by using results obtained in [24], we can obtain less severe condition.

DEFINITION 2.7.1. Consider the weighted inner product $\langle \mathbf{v}, \mathbf{w} \rangle_D = \mathbf{u}^T \mathbf{D}\, \mathbf{v}$, where $\mathbf{D} = \operatorname{diag}(d_1, \dots, d_s)$ with $d_l > 0$. We then denote by $\alpha_D\left(\mathbf{A}^{-1}\right)$ the largest number α such that

$$\left\langle \mathbf{v}, \mathbf{A}^{-1}\mathbf{v} \right\rangle_D \geq \alpha \left\langle \mathbf{v}, \mathbf{v} \right\rangle_D, \quad \forall \mathbf{v} \in \mathbb{R}^s. \tag{2.7.3}$$

We also set

$$\alpha_0\left(\mathbf{A}^{-1}\right) := \sup_{\mathbf{D}>\mathbf{0}} \alpha_D\left(\mathbf{A}^{-1}\right). \tag{2.7.4}$$

Then an existence and uniqueness condition are given in the following theorem.

THEOREM 2.7.2. *Let* $\mathbf{u}$ *be continuously differentiable and satisfy the one-sided Lipschitz condition* (2.2.6)*. If the RK matrix* $\mathbf{A}$ *is invertible and*

$$h\nu < \alpha_0\left(\mathbf{A}^{-1}\right), \tag{2.7.5}$$

then the nonlinear system (2.7.1) possesses a unique solution.

The corresponding condition for the general LMM (2.5.3) is simpler, since LMMs do not increase the dimensionality of the nonlinear system to be solved. Here the nonlinear system can be expressed by the following

$$\alpha_0\, \mathbf{x}^{i+1} - h\, \beta_0\, \mathbf{u}\left(t^{i+1}, \mathbf{x}^{i+1}\right) - \mathbf{s}^i = 0, \tag{2.7.6}$$

where

$$\mathbf{s}^i := \sum_{l=1}^{m} \left[-\alpha_l \, \mathbf{x}^{i-l+1} + h \, \beta_l \, \mathbf{u}\left(t^{i+1}, \mathbf{x}^{i-l+1}\right) \right], \tag{2.7.7}$$

represents a vector composed of known quantities.

THEOREM 2.7.3. *Let* $\mathbf{u}$ *be continuously differentiable and satisfy the one-sided Lipschitz condition 2.2.6. If*

$$h\nu < \frac{\alpha_0}{\beta_0}, \tag{2.7.8}$$

then the nonlinear system (2.7.6) has a unique solution.

Some other interesting existence and uniqueness results can be found in [17] and [54].

2.7.2 Nonlinear systems of equations

As known from the literature, solving the nonlinear system (2.7.1) or (2.7.6) is not a trivial task and it represents (computationally) the most expensive part of an implicit method. The main reason for this is that typically one needs to apply *the Newton method* or some of its variants (see e.g. [24, 31]). The method gives an approximation of the zero of a nonlinear system of equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{x} \in \mathbb{R}^N. \tag{2.7.9}$$

Note that (2.7.1) and (2.7.6) can be represented in this form. The Newton method is defined by the recursion

$$\mathbf{x}^{[\nu+1]} = \mathbf{x}^{[\nu]} - \left[D\mathbf{f}\left(\mathbf{x}^{[\nu]}\right) \right]^{-1} \mathbf{f}\left(\mathbf{x}^{[\nu]}\right), \quad \nu = 0, 1, \ldots, \tag{2.7.10}$$

where $D\mathbf{f}$ is the Jacobian matrix of $\mathbf{f}$.

Although the Newton method is a popular choice and quite efficient when successful (see e.g. [37]), it may introduce serious difficulties in the implementation. We mention a few. Firstly, the iterative process may not converge if the initial value is not close to the solution, i.e. the convergence is only guaranteed locally. Secondly, for ODE systems with a velocity field $\mathbf{u}$ not being explicitly known (implying that $\mathbf{f}$ is also not known explicitly), it is not easy to implement (2.7.10). Recalling (2.7.1) and (2.7.6), it is clear that one needs to obtain values $\mathbf{u}\left(\mathbf{x}^{[\nu]}\right)$ and $\left[D\mathbf{u}\left(\mathbf{x}^{[\nu]}\right) \right]^{-1}$. This can represent a tedious task, especially considering the inverse Jacobian matrix. Finally, it is not clear how good these approximations should be to preserve the convergence of the Newton method in the continuous case, of course, assuming that it converges at all.

Since in this thesis our main interest is in problems where $\mathbf{u}$ is only given discretely by a set $\{(\mathbf{x}_k, \mathbf{u}_k)\}_{k=0}^{n}$, we will use another approach to solve (2.7.10) and, thus, avoid the Newton method and its aforementioned difficulties. An alternative is inverse interpolation, which is the main topic of the following chapter. However, both approaches may give equivalent (or even identical) results in some situations, as will be pointed out.

CHAPTER 3

Interpolation methods

This chapter is devoted to interpolation being an essential ingredient of methods introduced in the following chapters. We first give some elementary concepts of interpolation. The univariate and multivariate interpolation aspects are given in Sections 3.3 and 3.4 respectively. We introduce some interpolation methods, which are of importance in this thesis. The most important interpolation aspect for the upcoming chapters is the inverse interpolation, which will be employed in numerical methods for solving ODEs. It is given in the final section of this chapter.

3.1 Introduction to approximation and interpolation

The central question in the approximation theory is how to replace a complicated function f, from a large space $\mathcal{F}$, by a simple and yet close-by (or "good" in some sense) function p from a small subset $\mathcal{P} \subset \mathcal{F}$. The literature about approximation is very rich (see e.g. [16, 35, 36]). Usually $\mathcal{F}$ is a Banach space, so that the distance between p and f by means of a norm. Functions from $\mathcal{P}$ are called *approximation functions*.

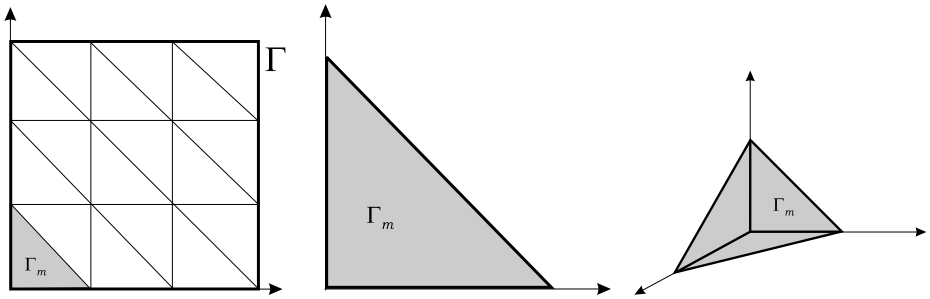
The approximation functions depend on a set of parameters $\{a_i\}_{i=0}^m \in \mathcal{A}$. For example, for $\mathcal{P}$ being finite dimensional, $\mathbf{p} \in \mathcal{P}$ can be represented as

$$\mathbf{p}(\mathbf{x}) = \sum_{i=0}^{m} a_i \mathbf{p}_i(\mathbf{x}), \tag{3.1.1}$$

where $\{\mathbf{p}_i\}_{i=0}^m$ represents a set of *basis functions* in $\mathcal{P}$. There are many ways to choose the space $\mathcal{P}$. The most obvious ones are *algebraic polynomials.* Other important choices of basis functions in theory are trigonometric polynomials, exponential and rational functions, etc. We should remark that the latter two are nonlinear. These types of basis functions are not of special interest in this thesis. However, in the following sections we will consider a well known generalisation of polynomial spaces, i.e. *splines*.

The type of approximation depends on the way how the parameters are obtained. One of the most important is *interpolation*. In the general case for a function $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^M$, we can define a set of pairs

$$\Omega := \left\{ (\mathbf{x}_k, \mathbf{f}_k) \,\middle|\, \mathbf{x}_k \in S \subset \Gamma \subseteq \mathbb{R}^N,\ \mathbf{f}_k := \mathbf{f}(\mathbf{x}_k) \in \mathbb{R}^M,\ k = 0, \ldots, n \right\}, \tag{3.1.2}$$



(a) Grid in 2D (b) Triangular simplex (c) Tetrahedron simplex

FIGURE 3.1.1. Examples of grid and simplices

and the following condition

$$\mathbf{p}(\mathbf{x}_k) := \mathbf{f}_k, \tag{3.1.3}$$

We then say that $\mathbf{p}$ *interpolates* $\mathbf{f}$ at $\mathbf{x}_0, \ldots, \mathbf{x}_n$, i.e. (3.1.2)-(3.1.3) represents *the interpolation problem*. Here S is the set $\{\mathbf{x}_k\}_{k=0}^{n}$ of *interpolation nodes*, i.e. the points where the function values are known, Γ is *the interpolation domain*, and $\hat{\Gamma}$ *the range* (or co-domain) of $\mathbf{f}$. The interpolation problem can also be formulated in another way, viz. as the answer to the following question: How to find a "good" representative of a function that is not known explicitly, but only at some points of the domain of interest. In Section 3.5 we pay special attention to the case where $M = N$. Here we have a mapping between spaces with the same dimensionality, which is an essential requirement for the inverse interpolation.

The interpolation domain Γ and the set of interpolation nodes S play an essential role in the interpolation problem settings. We distinguish between interpolation on *regularly* spaced data, where the distribution of points satisfies some particular condition, and interpolation on *scattered* (irregularly) spaced data, where S is any subset of Γ. Interpolation can be *global* or *local*, depending on the support of the interpolation function. If all nodes are used for determining all the parameters we have a global approach, which means that any parameter or data perturbation will affect the solution throughout the whole interpolation domain. On the other hand, if the same perturbation does not influence the interpolation function values outside some (possibly small) subdomain of Γ, the method is considered to be local. Related to this, it is useful to mention that there is also an interesting class of the interpolation functions where the basis functions have local support only (equal to zero outside the subdomain). In general, these methods are global, but few of them are local. Spline interpolation functions are typical representatives of such functions.

For interpolation by splines, also called *the piece-wise functions*, one typically needs to discretise the domain, i.e. to generate a *grid* which covers Γ. The grid is defined by the set S and the choice of the basic elements, say $\Gamma_m \in \Gamma$, $m = 1, \ldots, n_m$. For example, one can think of a discretisation of Γ where the elements are convex *simplices* like triangles (in 2D space) and tetrahedrons (in 3D space), etc. As an illustration in Figure 3.1.1 an example of the grid and an isolated simplices are shown.

We assume that Γ_m's covers Γ, i.e. that $\bigcup_{m=1}^{n_m} \Gamma_m = \Gamma$ holds. To a simplex Γ_m there is appointed a set of points $S_m \subset S$ such that

$$\Gamma_m := \operatorname{conv} S_m. \tag{3.1.4}$$

The K-dimensional simplex in $\mathbb{R}^N$ is spanned by a set of points $\mathbf{x}_0, \ldots, \mathbf{x}_K$, such that $\mathbf{x}_0, \ldots, \mathbf{x}_K$ do not belong to a subspace that has a dimension smaller than K. Hence the N-dimensional simplex is defined by exactly $N+1$ non-degenerate points. Methods based on such simplices are of most interest in this thesis.

Interpolation methods are well developed to date. The practical applications are numerous and the underpinning theory has been essential for developing whole classes of different methods in numerical analysis. Examples can be found in numerical integration and differentiation, numerical methods for solving ODEs and PDEs, etc. In this thesis our main goal is to employ inverse interpolation techniques in standard implicit numerical methods for solving ODEs and, thus, to avoid Newton iteration. Inverse interpolation has a lot of potential in solving systems of nonlinear equations. It combines the spatial mapping of the interpolation domain and the application of a standard method for direct interpolation. Therefore we will firstly introduce some interesting interpolation methods, both for scalar and vectorial problems.

Before we go into details about interpolation, we introduce some relevant norms that will be used throughout this thesis. More detailed information can be found in e.g. [28]. Like in the previous chapter the notation $\|\cdot\|$ is reserved for the Euclidean norm, defined by (2.2.1). Another important norm is *the Hölder norm* for $\mathbb{R}^N$, defined by

$$\|\mathbf{x}\|_p := \left(\sum_{k=1}^{N} x_k^p \right)^{1/p}, \tag{3.1.5}$$

where $p = 1, 2, \ldots$ For $p \to \infty$, one obtains $\infty-$*norm*

$$\|\mathbf{x}\|_\infty := \sup_k |x_k|. \tag{3.1.6}$$

For a space of all continuous real-valued functions on a interval $J = [a, b]$, there is a corresponding normed space with the so-called L_p-*norm*, which is defined by

$$\|\mathbf{x}\|_{L_p} := \left(\int_a^b |x(t)|\, dt \right)^{1/p}. \tag{3.1.7}$$

Again, for $p \to \infty$, we can define the $L_\infty-$*norm* by the following

$$\|\mathbf{x}\|_{L_\infty} := \sup_{t \in J} |x(t)|. \tag{3.1.8}$$

Other relevant norms in this thesis are introduced later.

3.2 Characteristics of interpolation methods

Evaluating and comparing characteristics of different interpolation methods is somewhat subjective. However, some attempts were made for obtaining the list of the most important characteristics of interpolation methods. In [20] the special case of the two-dimensional (scattered) data interpolation is considered by performing a test over 32 different methods. The results are evaluated by a set of characteristics, given by the following list:

- *Accuracy.* Accuracy is expressed by *the interpolation error*, say r, which is a measure of the difference between the interpolation function and the exact values of the interpolating function $\mathbf{f}$. It can be defined point-wise via the error function

$$r(\mathbf{x}) := |||\mathbf{f}(\mathbf{x}) - \mathbf{p}(\mathbf{x})|||, \tag{3.2.1}$$

  or as a scalar

$$r := |||\mathbf{f} - \mathbf{p}|||, \tag{3.2.2}$$

  where $|||\cdot|||$ is some suitable norm. Usually the error depends on the space $\mathcal{P}$ and the location of the interpolation points. If the interpolation points are the vertices of a grid of simplices, then the *the order of accuracy* is related to the maximum diameter of these simplices. There are also examples where some additional geometrical parameters (as shown later) are used to determine (or evaluate) the accuracy, such as angles between edges, radius of the ball containing the simplex, etc.
- *Visual Aspects.* Of course, the appearance of the interpolation function on Γ is of importance only in low dimensional spaces ($N = 1, 2, 3$). Visual aspects are often in a close relationship with the accuracy, especially at moderate accuracies. An example is shown in Figure 3.2.1, where the function $f(x, y) = x\, e^{-(x^2+y^2)}$ is interpolated by four different methods (which will be introduced later).
- *Sensitivity to Parameters.* It is desirable that a method is stable with respect to perturbations of the parameters and that the solution value is not highly dependent on the sampled function. In principle, as mentioned, local interpolation methods have an advantage, but it does not mean that all global methods will behave badly in general.
- *Computational costs.* Computational efforts depends on a chosen method. Some methods can be extremely expensive in some applications and has to be avoided, even if all other characteristics are good. This also depends on the implementation used.

3.3 Univariate interpolation

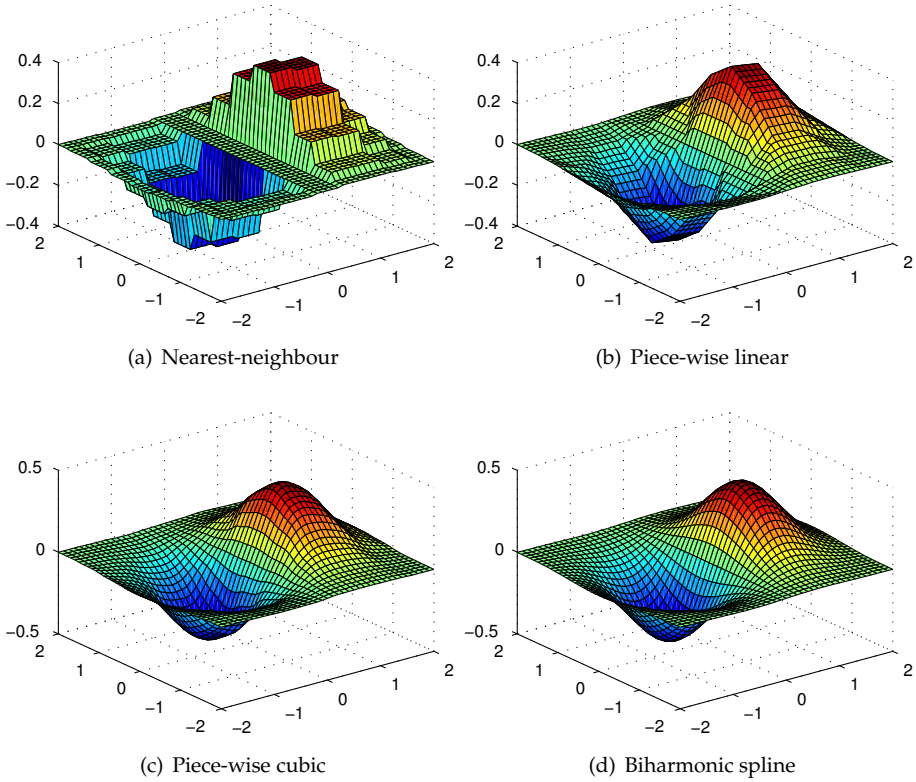
3.3.1 Polynomial interpolation

The theorem of Weierstrass (introduced for the first time in [52]) and its modifications can be considered as one of the foundations of the approximation theory, despite the fact that its original setting concerns only univariate polynomials. In the univariate case the theorem is stated as follows:

THEOREM 3.3.1. *Let* $f \in C[a, b]$. *For every* $\epsilon > 0$, *there exists a polynomial* $p(x)$ *such that*

$$\|f(x) - p(x)\| < \epsilon, \quad \forall x \in [a, b]. \tag{3.3.1}$$

From (3.3.1) it follows that one can always find a polynomial that is arbitrarily close to a given function on some finite interval. This means that the approximation error is bounded and can be reduced by the choice of the adequate polynomial. Unfortunately Theorem 3.3.1 is not a constructive one, i.e. it does not present a way how to obtain such a polynomial. However, many proofs (there are hundreds of them, as



(a) Nearest-neighbour (b) Piece-wise linear

(c) Piece-wise cubic (d) Biharmonic spline

FIGURE 3.2.1. Visual aspects of different interpolation methods

cited in [11]) of this theorem are constructive. One of them, given in [4], is related to *the Bernstein polynomial*.

For a given function $f \in C[0,1]$ the Bernstein polynomial, which is usually denoted by $B_n(f;x) \in \mathcal{P}_n$ (where $\mathcal{P}_n$ is the set of all algebraic polynomials of degree $\leq n$), is defined by

$$B_n(f;x) := \sum_{k=0}^{n} \binom{n}{k} x^k \, (1-x)^{n-k} \, f\left(\frac{k}{n}\right), \quad x \in [0,1]\,. \tag{3.3.2}$$

It can be shown (see e.g. [35]) that

$$\lim_{n\to\infty} \|f - B_n\| = 0. \tag{3.3.3}$$

Moreover, the following error bound holds

$$|f(x) - B_n(f;x)| \leq \frac{9}{4}\,\omega\left(f; \frac{1}{\sqrt{n}}\right), \tag{3.3.4}$$

$$\omega(f;\delta) := \sup_{|x-y|<\delta} |f(x) - f(y)|.$$

If f is Lipschitz continuous, with a Lipschitz constant L, it simplifies to

$$|f(x) - B_n(f;x)| \leq \frac{9L}{4\sqrt{n}}. \tag{3.3.5}$$

Theorem 3.3.1 is often used in conjunction with the following uniqueness theorem.

THEOREM 3.3.2. *Let* $\{x_k\}_{k=0}^n$ *be a set of distinct points, and let* $\{y_k\}_{k=0}^n$ *be an arbitrary set of points in* $\mathbb{R}$*. Then, there exists a unique polynomial of degree* n *which takes given values at* $n+1$ *points.*

Proofs of this theorem are numerous and often constructive, see e.g. [36]. Certainly one of the most famous ones are related to the well-known Lagrange interpolation method. For this method we have the following error bound.

THEOREM 3.3.3. *Suppose* $f \in C^{n+1}[a,b]$*. Let the interpolation nodes satisfy* $a \leq x_0 < x_1 < \cdots < x_n \leq b$*, the polynomial* $p \in \mathcal{P}_n$ *interpolates* f *at* $\{x_k\}_{k=0}^n$ *and* $w(x) := \prod_{k=0}^n (x - x_k)$*. Then there exists* $\xi \in [a,b]$ *such that the error function* $r(x)$ *satisfies*

$$r(x) := f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\, w(x). \tag{3.3.6}$$

Taking $\Gamma = [a,b]$ and using the L_p-norm in (3.3.6), one trivially obtain

$$r := \|f - p\|_{L_p(\Gamma)} \leq \frac{1}{(n+1)!}\, \|f^{(n+1)}\|_{L_p(\Gamma)}\, \|w\|_{L_p(\Gamma)}. \tag{3.3.7}$$

An interesting special case is that of equidistant points where $x_{k+1} = x_k + \Delta x$ for all k. Then we have

$$\|f - p\|_{L_\infty(\Gamma)} \leq \frac{1}{4}\, \frac{\Delta x^{n+1}}{(n+1)}\, \|f^{(n+1)}\|_{L_\infty(\Gamma)}. \tag{3.3.8}$$

3.3.2 Piece-wise interpolation

The main disadvantage of global polynomial interpolation is that the interpolation error is related to higher derivatives of the interpolated function f. For example, the condition $f \in C^{n+1}[a,b]$ met in Theorem 3.3.3 can be too strict. This is one of the most important reasons why often another approach is used. Discretising the interpolation domain and interpolating locally, i.e. on small subsets of $\{x_k\}_{k=0}^n$, the overall accuracy may be significantly improved even if the applied (local) interpolation method is of the low order. Interpolation functions obtained on this principle are piece-wise interpolation functions or splines. Here we just mention some of the most important (most frequently used) piece-wise interpolation methods and illustrate their accuracy by one simple example. The accuracy is expressed in terms of the spatial step size, defined as $\Delta x := \max_k \{x_k, x_{k+1}\}$.

(1) *Nearest-neighbour method.* By far the easiest way to interpolate interpolation pairs $\{(x_k, f_k)\}_{k=0}^n$, by piece-wise constants. The method is $O(\Delta x)$.
(2) *Piece-wise linear interpolation.* An improvement of the previous method, made by constructing a linear function between two consecutive nodes. The accuracy is $O(\Delta x^2)$.
(3) *Piece-wise cubic interpolation.* By increasing the order of the piece-wise polynomial, one can obtain further improvements in the characteristics of the interpolation method. If the polynomial is of the third order, we have piece-wise cubic interpolation. This method is $O(\Delta x^4)$.
(4) *Cubic spline interpolation.* Probably the most popular choice for obtaining the piece-wise interpolation function, which is necessarily differentiable at interpolation nodes. This provides the interpolation function to be smooth

on the entire domain $\Gamma = [a, b]$. The spline theory is well introduced in [2] and [14]. This method is also $O\left(\Delta x^4\right)$.

(5) *Biharmonic spline interpolation.* Starting from the minimum curvature property, which cubic splines satisfy, an additional strategy of constructing a spline interpolation function can be made (see [45]). Considering the biharmonic equation, which has the Green's function as a solution, one can construct an interpolation function. The thus obtained interpolation function is practically identical to the cubic spline.

EXAMPLE 3.3.4. To illustrate these methods we perform a test problem where the function $f(x) = \sin x$ is sampled in 9 equidistant points over $x \in [0, 2\pi]$ with a step size $\Delta x = 0.78$ and then interpolated by various methods. In Figure 3.3.1 errors over the whole domain are shown and results are as expected. □

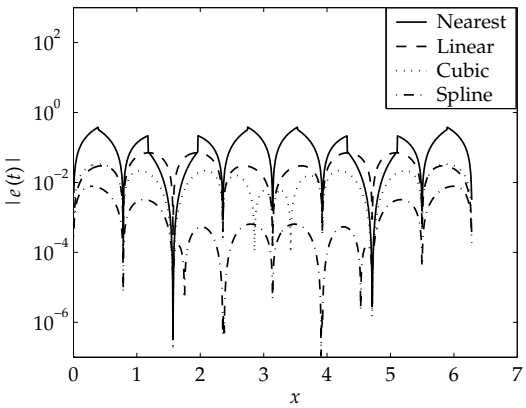


FIGURE 3.3.1. Accuracy of various interpolation methods

3.4 Multivariate interpolation

Multivariate interpolation is analogue to univariate interpolation. Now we have an interpolation domain $\Gamma \subset \mathbb{R}^N$ and an interpolating function as $f : \Gamma \to \hat{\Gamma}$, where $\hat{\Gamma} \subset \mathbb{R}^M$. Throughout this section we assume $M = 1$, i.e. f is a scalar function defined on a multidimensional domain. The interpolation problem is defined by (3.1.2) and (3.1.3), i.e. by the set of pairs $\{(\mathbf{x}_k, f_k)\}_{k=0}^n$. Obtaining an approximation of $f(\mathbf{x})$ means constructing an interpolation function, say $p(\mathbf{x})$, just as we did in the univariate case. However, the problem settings are not so simple and methods are more involved. Since uniqueness is not guaranteed, the interpolation based on algebraic polynomials is not straightforward. The existence and uniqueness of a multivariate polynomial that interpolates the set (S) of interpolation nodes strongly depends on their locations. In fact, such a polynomial does not exist for any given grid. For example, in $\mathbb{R}^2$, by defining $\Pi_N^2 := \left\{x^i y^j \,|\, i + j \leq N\right\}$, it can be shown that use of the bivariate Lagrange formula on a rectangular grid yields to a polynomial which is not in Π_N^2. Actually only special grid configurations provides $p \in \Pi_N^2$. This is the main reason why the piece-wise methods are more popular for practical applications than those based on global algebraic polynomials. Here we introduce only few interpolation methods.

3.4.1 Linear interpolation on a simplex

Following the analogy from the univariate case, one can construct an interpolation function on a simplex by connecting interpolation nodes using multivariate linear functions (examples of such linear functions are a plane in $\mathbb{R}^2$, a tetrahedron in $\mathbb{R}^3$ and a hyperplane in $\mathbb{R}^N$). To do so one needs to discretise the interpolation domain, i.e. to form a grid which covers Γ. The best discretisation technique is to use the simplices Γ_m, $m = 1, \dots, n_m$, where Γ_m is defined by a set, say $S_m \subset S$, of $(N+1)$ points $\{\mathbf{x}_{mk}\}_{k=0}^{N}$. We assume that these points are non-degenerate. On such a simplex one can construct a linear polynomial as a function of $\mathbf{x} = [x_1 \cdots x_N]^T$, i.e.

$$p(\mathbf{x}) := a_0 + a_1 x_1 + \cdots + a_N x_N. \tag{3.4.1}$$

The parameter values a_k can be obtained from the known function values at interpolation nodes. Hence, by substituting $(N+1)$ pairs $\{(\mathbf{x}_{mk}, f_{mk})\}_{k=0}^{N}$ and employing the interpolation condition $p(\mathbf{x}_k) := f_k$, one obtain the parameter set by solving a linear system of equations. The linear polynomial can be defined in an alternative way, i.e. in the Lagrangian form. It can be shown that p on Γ_m can be expressed as

$$p(\mathbf{x}) := \sum_{k=0}^{N} \frac{\Lambda_k(\mathbf{x})}{\Lambda} f_k, \tag{3.4.2}$$

where the double index is omitted and Λ and $\Lambda_k(\mathbf{x})$ are determinants defined by $(N+1)$ points $\mathbf{x}_k = [x_{1,k} \cdots x_{N,k}]$ like

$$\Lambda := \begin{vmatrix} x_{10} & x_{20} & \cdots & x_{N0} & 1 \\ x_{11} & x_{21} & & x_{N1} & 1 \\ \vdots & & & & \\ x_{1N} & x_{2N} & & x_{NN} & 1 \end{vmatrix}, \quad \Lambda_k(\mathbf{y}) = \begin{vmatrix} x_{10} & \cdots & x_{N0} & 1 \\ \vdots & & & \\ x_{1,k-1} & & x_{N,k-1} & 1 \\ y_1 & & y_N & 1 \\ x_{1,k+1} & & x_{N,k+1} & 1 \\ & & & \\ x_{1N} & & x_{NN} & 1 \end{vmatrix}. \tag{3.4.3}$$

Literature about the error analysis of the piece-wise linear interpolation is very extensive. Especially since this method is frequently implemented in some numerical methods for solving PDEs (FEM in particular). A lot of attention is given to a two-dimensional case, where the simplex is the triangle (see e.g. [25], [43]). For the general N-dimensional case two main results are given to date, introduced via L_∞ −error bounds of the interpolation error on the simplex. They are both introduced here since they are different in general and yet related (as shown later). But before we set some notation.

Let Γ_m represent the convex simplex defined by a set of $(N+1)$ non-degenerate points from $S_m = \{\mathbf{x}_k\}_{k=0}^{N}$, i.e.

$$\Gamma_m := \operatorname{conv} S_m, \tag{3.4.4}$$

with a *diameter*, say Δx, defined by

$$\Delta x := \max_{\mathbf{x}_k, \mathbf{x}_j \in S_m} \|\mathbf{x}_k - \mathbf{x}_j\|, \tag{3.4.5}$$

where $\|\cdot\|$ represents the Euclidean norm in $\mathbb{R}^N$. Further let

$$\mathbf{y} = [y_1 \cdots y_N]^T \in \mathbb{R}^N, \quad \mathbf{y} \neq \mathbf{0}, \tag{3.4.6}$$

and let

$$D_{\mathbf{y}} f(\mathbf{x}) := \sum_{k=1}^{N} y_k \frac{\partial f(\mathbf{x})}{\partial x_k}, \tag{3.4.7}$$

be the directional derivative of f in the direction $\mathbf{y}$. Furthermore we define

$$\left\| D^2 f \right\|_{\infty,\Gamma_m} := \sup_{\mathbf{x}\in\Gamma_m} \sup_{\substack{\mathbf{y}\in\mathbb{R}^N \\ \|\mathbf{y}\|=1}} \left| D^2_{\mathbf{y}} f(\mathbf{x}) \right|, \tag{3.4.8}$$

as a measure of the second derivative of f on Γ_m. Finally, recall that p denotes the linear polynomial that interpolates a given function f at points S_m. The first error bound for linear interpolation represented here is given in [48] and relates the interpolation error with a diameter of the simplex.

THEOREM 3.4.1. *(Subbotin) Let $f \in C^2(\Gamma_m)$ on a simplex Γ_m with a diameter Δx. Then the error of the linear interpolation $r := \|f - p\|_{L_\infty(\Gamma_m)}$ satisfies*

$$r \le \frac{1}{4} \frac{N}{N+1} \Delta x^2 \left\| D^2 f \right\|_{\infty,\Gamma_m}. \tag{3.4.9}$$

Clearly, we have second order accuracy with respect to Δx. The second important result (see [51]), gives the error bound in terms of the radius and the position of the center of the circumscribed sphere around Γ_m.

THEOREM 3.4.2. *(Waldron) Let $\mathbf{c}$ be the center and R the radius of the unique circumscribed sphere containing S_m. Then for each $\mathbf{x} \in \Gamma_m$ and $f \in C^2(\Gamma_m)$, there holds the sharp inequality*

$$r(\mathbf{x}) := |f(\mathbf{x}) - p(\mathbf{x})| \le \frac{1}{2} \left(R^2 - \|\mathbf{x} - \mathbf{c}\|^2 \right) \left\| D^2 f \right\|_{\infty,\Gamma_m}, \quad \forall f \in C^2(\Gamma_m). \tag{3.4.10}$$

As a consequence we have the sharp inequality

$$r := \|f - p\|_{L_\infty(\Gamma_m)} \le \frac{1}{2} \left(R^2 - \rho^2 \right) \left\| D^2 f \right\|_{\infty,\Gamma_m}, \tag{3.4.11}$$

where ρ is the distance of $\mathbf{c}$ from Γ_m, i.e.

$$\rho := \min_{\mathbf{x}\in\Gamma_m} \|\mathbf{x} - \mathbf{c}\|. \tag{3.4.12}$$

Proof: Although the proof of this theorem is given in [51], another, more elementary, proof will be introduced here. Let us define a univariate function K as follows

$$K(\theta) := f\left(\mathbf{x} + \theta\ (\mathbf{x}_k - \mathbf{x})\right), \tag{3.4.13}$$

where $\mathbf{x}_k$ is an arbitrary node of Γ_m. Clearly, the first derivative of $K(\theta)$ is equal to the directional derivative of f in the direction $\mathbf{x}_k - \mathbf{x}$, i.e.

$$K'(\theta) = D_{\mathbf{x}_k - \mathbf{x}} f\left(\mathbf{x} + \theta\ (\mathbf{x}_k - \mathbf{x})\right).$$

Partial integration leads to the useful identity

$$K(1) = K(0) + K'(0) + \int_0^1 (1 - \tau)\, K''(\tau)\, d\tau. \tag{3.4.14}$$

Substituting (3.4.13) into (3.4.14), it follows

$$f(\mathbf{x}_k) = f(\mathbf{x}) + D_{\mathbf{x}_k - \mathbf{x}} f(\mathbf{x}) + \int_0^1 (1 - \tau)\, D^2_{\mathbf{x}_k - \mathbf{x}, \mathbf{x}_k - \mathbf{x}} f\left(\mathbf{x} + \tau\ (\mathbf{x}_k - \mathbf{x})\right)\, d\tau. \tag{3.4.15}$$

Let us assume that the linear interpolant $p(\mathbf{x})$ is represented b the Lagrangian form, i.e. by

$$p(\mathbf{x}) := \sum_{k=0}^{N} f(\mathbf{x}_k)\ l_k(\mathbf{x}),$$

where

$$l_k(\mathbf{x}) := \frac{\Lambda_k(\mathbf{x})}{\Lambda},$$

and Λ and $\Lambda_m(\mathbf{x})$ are given by (3.4.3). Since p resolves the linear function f exactly, one has

$$\sum_{k=0}^{N} l_k(\mathbf{x}) = 1,$$
$$\sum_{k=0}^{N} \mathbf{x}_k\, l_k(\mathbf{x}) = \mathbf{x}.$$

Now by multiplying (3.4.15) by $l_k(\mathbf{x})$ and summing over $k = 0, \ldots, N$, we obtain

$$p(\mathbf{x}) = f(\mathbf{x}) + \int_0^1 (1-\tau) \sum_{k=0}^{N} D^2_{\mathbf{x}_k-\mathbf{x},\mathbf{x}_k-\mathbf{x}} f(\mathbf{x} + \tau\ (\mathbf{x}_k - \mathbf{x}))\ l_k(\mathbf{x})\, d\tau. \tag{3.4.16}$$

Here we used

$$\begin{aligned}
\sum_{k=0}^{N} l_k(\mathbf{x})\, f(\mathbf{x}_k) &= p(\mathbf{x}),\\
\sum_{k=0}^{N} l_k(\mathbf{x})\, f(\mathbf{x}) &= f(\mathbf{x}),\\
\sum_{k=0}^{N} l_k(\mathbf{x})\, D_{\mathbf{x}_k-\mathbf{x}} f(\mathbf{x}) &= \sum_{k=0}^{N} l_k(\mathbf{x})\ \langle \nabla f, \mathbf{x}_k - \mathbf{x}\rangle\\
&= \left\langle \nabla f, \sum_{k=0}^{N} l_k(\mathbf{x})\ (\mathbf{x}_k - \mathbf{x}) \right\rangle = 0
\end{aligned}$$

From the definition (3.4.8), it can be shown that

$$\left|D^2_{u,v}\right| \le \left\|D^2 f\right\|_{\infty,\Gamma_m}\ \|u\|\,\|v\|.$$

Hence by substituting this into (3.4.16), we have

$$\begin{aligned}
|f(\mathbf{x}) - p(\mathbf{x})| &\le \left\|D^2 f\right\|_{\infty,\Gamma_m} \left| \sum_{k=0}^{N} l_k(\mathbf{x})\ \|\mathbf{x}_k - \mathbf{x}\|^2 \int_0^1 (1-\tau)\, d\tau \right|\\
&= \frac{1}{2}\ \left\|D^2 f\right\|_{\infty,\Gamma_m} \left| \sum_{k=0}^{N} l_k(\mathbf{x})\ \|\mathbf{x}_k - \mathbf{x}\|^2 \right|. \qquad (3.4.17)
\end{aligned}$$

Rearranging

$$\|\mathbf{x}_k - \mathbf{x}\|^2 = \|\mathbf{x}_k - \mathbf{c} + \mathbf{c} - \mathbf{x}\|^2 = \|\mathbf{x}_k - \mathbf{c}\|^2 + \|\mathbf{c} - \mathbf{x}\|^2 + 2\ \langle \mathbf{x}_k - \mathbf{c}, \mathbf{c} - \mathbf{x}\rangle,$$

and substituting into (3.4.17), we obtain

$$
\begin{aligned}
|f(\mathbf{x}) - p(\mathbf{x})| &\leq \frac{1}{2} \left\|D^2 f\right\|_{\infty,\Gamma_m} \left| \sum_{k=0}^{N} l_k(\mathbf{x}) \left(\|\mathbf{x}_k - \mathbf{c}\|^2 + \|\mathbf{c} - \mathbf{x}\|^2 + 2 \langle \mathbf{x}_k - \mathbf{c}, \mathbf{c} - \mathbf{x} \rangle \right) \right| \\
&= \frac{1}{2} \left\|D^2 f\right\|_{\infty,\Gamma_m} \left| R^2 + \|\mathbf{c} - \mathbf{x}\|^2 + 2 \langle \mathbf{x} - \mathbf{c}, \mathbf{c} - \mathbf{x} \rangle \right| \\
&= \frac{1}{2} \left(R^2 - \|\mathbf{c} - \mathbf{x}\|^2 \right) \left\|D^2 f\right\|_{\infty,\Gamma_m}. \qquad (3.4.18)
\end{aligned}
$$

The sharp inequality (3.4.11) easily follows from (3.4.18) since

$$
\max_{\mathbf{x} \in \Gamma_m} \left\{ R^2 - \|\mathbf{x} - \mathbf{c}\|^2 \right\} = R^2 - \min_{\mathbf{x} \in \Gamma_m} \|\mathbf{x} - \mathbf{c}\|^2, \qquad (3.4.19)
$$

which completes the proof of the theorem. □

The error bounds (3.4.9) and (3.4.11) seems to be rather different at first look. However, they can be completely related as shown by the following theorem, which generalises Theorems 3.4.1 and 3.4.2.

THEOREM 3.4.3. *Let* R *be the diameter of the unique circumscribed sphere of* $\Gamma_m \in \mathbb{R}^N$ *with diameter* Δx *and let* ρ *be the distance between* Γ_m *and the center of the sphere. Then*

(1) For $\mathbf{c} \in \Gamma_m$, which implies $\rho = 0$,

$$
R^2 \leq \frac{1}{2} \frac{N}{N+1} \Delta x^2. \qquad (3.4.20)
$$

(2) For $\mathbf{c} \notin \Gamma_m$

$$
R^2 - \rho^2 \leq \frac{1}{2} \frac{N-1}{N} \Delta x^2. \qquad (3.4.21)
$$

Proof:

(1) Applying the fact that all nodes of the simplex $\mathbf{x}_k$, $k = 0, \ldots, N$ are on the circumscribed sphere, we clearly have

$$
\|\mathbf{x}_k - \mathbf{c}\| = R, \quad k = 0, \ldots, N. \qquad (3.4.22)
$$

Since Γ_m is convex and $\mathbf{c} \in \Gamma_m$, the following holds

$$
\sum_{k=0}^{N} \alpha_k \mathbf{x}_k = \mathbf{c},
$$

where the coefficients α_k satisfy

$$
\sum_{k=0}^{N} \alpha_k = 1, \quad \alpha_k \geq 0, \quad k = 0, \ldots, N. \qquad (3.4.23)
$$

Introducing the usual inner product $\langle \cdot, \cdot \rangle$ in $\mathbb{R}^N$, we have

$$
\begin{aligned}
0 &= \left\langle \sum_{k=0}^{N} \alpha_k \left(\mathbf{x}_k - \mathbf{c} \right), \sum_{k=0}^{N} \alpha_k \left(\mathbf{x}_k - \mathbf{c} \right) \right\rangle \\
&= \sum_{k=0}^{N} \alpha_k^2 \|\mathbf{x}_k - \mathbf{c}\|^2 + 2 \sum_{k \neq j} \alpha_k \alpha_j \langle \mathbf{x}_k - \mathbf{c}, \mathbf{x}_j - \mathbf{c} \rangle. \qquad (3.4.24)
\end{aligned}
$$

Substituting (3.4.22) and

$$\langle \mathbf{x}_k - \mathbf{c}, \mathbf{x}_j - \mathbf{c} \rangle = \frac{1}{2} \left(\|\mathbf{x}_k - \mathbf{c}\|^2 + \|\mathbf{x}_j - \mathbf{c}\|^2 - \|\mathbf{x}_k - \mathbf{x}_j\|^2 \right)$$
$$= R^2 - \frac{1}{2} \|\mathbf{x}_k - \mathbf{x}_j\|^2 ,$$

into (3.4.24), we obtain

$$R^2 \left(\sum_{k=0}^{N} \alpha_k^2 + 2 \sum_{k \neq j} \alpha_k \alpha_j \right) - \sum_{k \neq j} \alpha_k \alpha_j \, \|\mathbf{x}_k - \mathbf{x}_j\|^2 = 0. \tag{3.4.25}$$

Since

$$\sum_{k=0}^{N} \alpha_k^2 + 2 \sum_{k \neq j} \alpha_k \alpha_j = \left(\sum_{k=0}^{N} \alpha_k \right)^2 = 1, \tag{3.4.26}$$

the expression (3.4.25) becomes

$$R^2 = \sum_{k \neq j} \alpha_k \alpha_j \, \|\mathbf{x}_k - \mathbf{x}_j\|^2 .$$

Applying (3.4.5), we obtain

$$R^2 \leq \Delta x^2 \sum_{k \neq j} \alpha_k \alpha_j \leq \Delta x^2 \max_{\alpha_k, \alpha_j} \left\{ \sum_{k \neq j} \alpha_k \alpha_j \right\} .$$

Using (3.4.26) it follows

$$\max_{\alpha_k, \alpha_j} \left\{ \sum_{k \neq j} \alpha_k \alpha_j \right\} = \frac{1}{2} \max_{\alpha_k} \left\{ 1 - \sum_{k=0}^{N} \alpha_k^2 \right\} = \frac{1}{2} \left(1 - \min_{\alpha_k} \left\{ \sum_{k=0}^{N} \alpha_k^2 \right\} \right) . \tag{3.4.27}$$

It can be shown (e.g. by the Lagrange multiplier technique) that, under the constraint (3.4.23), the minimum value of the last expression in (3.4.27) is obtained for $\alpha_0 = \cdots = \alpha_N = 1/(N+1)$. Hence

$$R^2 \leq \frac{1}{2} \Delta x^2 \left(1 - \frac{1}{N+1} \right) = \frac{1}{2} \frac{N}{N+1} \Delta x^2 . \tag{3.4.28}$$

(2) In case $\mathbf{c} \notin \Gamma_m$, we have that $\rho \neq 0$ and it represents the distance between $\mathbf{c}$ and the closest point of the simplex, say $\mathbf{x}^* \in \Gamma_m$. Point $\mathbf{x}^*$ lies in the interior of $\partial\Gamma_m^{\tilde{N}}$, i.e. a boundary facet of dimension $\tilde{N}$ ($\tilde{N} \geq 1$). Moreover, we define $\tilde{N}$ to be the largest dimensionality for which $\mathbf{x}^* \in \partial\Gamma_m^{\tilde{N}}$. For example, $\partial\Gamma_m^{\tilde{N}}$ can be a line, a triangle, up to an $(N-1)$-dimensional facet of the simplex. Let us start with an assumption that $\tilde{N} = N - 1$. First, we show that the longest edge (i.e. Δx) of Γ_m must be at the boundary area for $\rho \neq 0$. Here we have that only one node of Γ_m, say $\mathbf{x}_0$, is on the opposite side of $\partial\Gamma_m^{N-1}$, observing from the side of $\mathbf{c}$. A 3D-illustration is shown in Figure 3.4.1a. Since all nodes lie at the circumscribed sphere, it is clear that every triangle defined by points $\mathbf{x}_0, \mathbf{x}_k, \mathbf{x}_q$, where $\mathbf{x}_k, \mathbf{x}_q \in \partial\Gamma_m$, $k, q = 1, \ldots, N$, is obtuse angled. This means that the following holds

$$\|\mathbf{x}_0 - \mathbf{x}_k\| < \|\mathbf{x}_q - \mathbf{x}_k\| , \quad k, q = 1, \ldots, N.$$

In other words, the diameters of Γ_m and $\partial\Gamma_m^{N-1}$ are identical. The same analogy may be applied for cases where dimensionality of the boundary area, say $\tilde{N}$, is less than $(N-1)$. Again it can be shown that the diameter

of Γ_m lies at $\partial\Gamma_m^{\tilde{N}}$. This follows from the fact that every triangle defined by two points from $\partial\Gamma_m^{\tilde{N}}$ and another point which is not in $\partial\Gamma_m^{\tilde{N}}$ is again obtuse angled. A 3D-example is shown in Figure 3.4.1b, where the boundary is just a line.
The boundary $\partial\Gamma_m^{\tilde{N}}$ is also the convex simplex, defined by $\left(\tilde{N}+1\right)$ points, say $\mathbf{x}_k$, $k = 1, \ldots, \tilde{N}+1$. Moreover we have that $\mathbf{x}^* \in \partial\Gamma_m^{\tilde{N}}$ and $\mathbf{x}^* \notin \partial\Gamma_m^{\tilde{N}-1}$. Let us now prove that $\mathbf{x}^*$ is the center of circumscribed sphere around $\partial\Gamma_m^{\tilde{N}}$. For arbitrary $\mathbf{x}_k \in \partial\Gamma_m^{\tilde{N}}$, we have the following

$$\begin{aligned} \|\mathbf{x}_k - \mathbf{x}^*\|^2 &= \|\mathbf{x}_k - \mathbf{c}\|^2 - \|\mathbf{x}^* - \mathbf{c}\|^2 - 2\left\langle \mathbf{x}_k - \mathbf{x}^*, \mathbf{x}^* - \mathbf{c}\right\rangle \\ &= R^2 - \rho^2 - 2\left\langle \mathbf{x}_k - \mathbf{x}^*, \mathbf{x}^* - \mathbf{c}\right\rangle . \end{aligned} \tag{3.4.29}$$

It can be shown that the inner product on the right-hand side is equal to zero. Define a quadratic polynomial, say $\varphi(\lambda)$, by

$$\varphi(\lambda) := \|\mathbf{x}^* + \lambda\left(\mathbf{x}_k - \mathbf{x}^*\right) - \mathbf{c}\|^2 , \tag{3.4.30}$$

where $\mathbf{x}^* + \lambda\left(\mathbf{x}_k - \mathbf{x}^*\right) \in \partial\Gamma_m^{\tilde{N}}$, for $|\lambda| < \epsilon$ and sufficiently small ϵ. Since $\varphi(\lambda) \geq \varphi(0)$, which follows from the definition of $\mathbf{x}^*$ being the closest point to $\mathbf{c}$, we have

$$\|\mathbf{x}^* + \lambda\left(\mathbf{x}_k - \mathbf{x}^*\right) - \mathbf{c}\|^2 \geq \|\mathbf{x}^* - \mathbf{c}\|^2 = \varphi(0).$$

Clearly we have that $\varphi'(0) = 0$. From (3.4.30) one can obtain the following

$$\varphi'(0) = 2\left\langle \mathbf{x}_k - \mathbf{x}^*, \mathbf{x}^* - \mathbf{c}\right\rangle ,$$

which means that

$$\left\langle \mathbf{x}_k - \mathbf{x}^*, \mathbf{x}^* - \mathbf{c}\right\rangle = 0. \tag{3.4.31}$$

Substituting this into (3.4.29), it follows that the distance of an arbitrary point $\mathbf{x}_k \in \partial\Gamma_m^{\tilde{N}}$ and $\mathbf{x}^*$ is constant, meaning that $\mathbf{x}^*$ is the center of the circumscribed sphere around $\partial\Gamma_m^{\tilde{N}}$, with a radius, say $\tilde{R}$, which can be found by applying (3.4.29) and (3.4.31), as

$$\tilde{R}^2 := \|\mathbf{x}_k - \mathbf{x}^*\|^2 = R^2 - \rho^2.$$

Now by applying the result from the part (1) and the fact that diameters of Γ_m and $\partial\Gamma_m^{\tilde{N}}$ are the same, we obtain

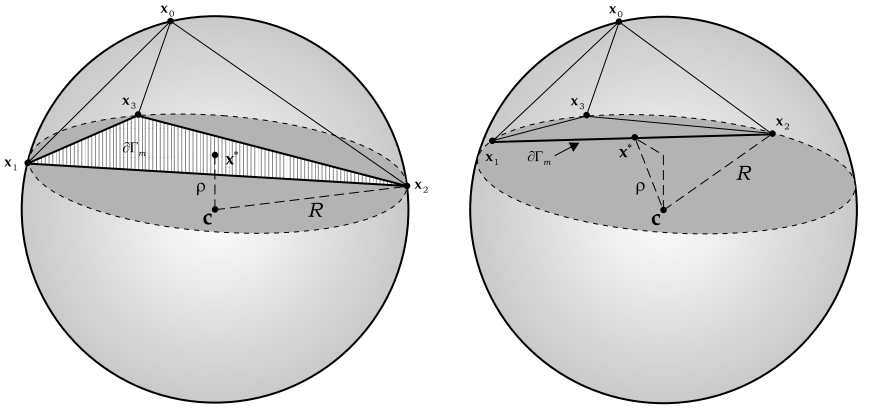
$$R^2 - \rho^2 \leq \frac{1}{2}\,\frac{\tilde{N}}{\tilde{N}+1}\,\Delta x^2.$$

Hence for every fixed $\tilde{N}$ we can find a new bound. However, since the following holds

$$\frac{N-1}{N} > \frac{N-2}{N-1} > \cdots > \frac{3}{4} > \frac{1}{2},$$

it is clear that for every $\tilde{N} \in \{1, \ldots, N-1\}$, we have that (3.4.21) holds. □

Since the principle and the implementation of piece-wise linear interpolation is relatively simple and the second order accuracy is often good enough, the application area of the method is wide (see e.g. [55] and [5], etc.).



(a) $\partial\Gamma_m$ - triangle (b) $\partial\Gamma_m$ - line

FIGURE 3.4.1. A 3D-simplex with the boundary area of different dimensionality

3.4.2 Piece-wise polynomial interpolation of higher order

As mentioned higher order multivariate interpolation is more involved than in the univariate case, since the number of parameters increases dramatically if the space dimension N and the order of the polynomial, say l, are large. In fact, for the space of polynomials of order l, say Π_N^l, the dimension is

$$\dim \Pi_N^l = \binom{N+l}{N}. \tag{3.4.32}$$

The polynomial, say $p(\mathbf{x})$, on a simplex Γ_m, reads

$$p(\mathbf{x}) := \sum_{|\alpha|\le l} a_\alpha x^\alpha, \tag{3.4.33}$$

where $\alpha = [\alpha_1, \ldots, \alpha_N]^T \in \mathbb{Z}_+^N$ is called the *multi-index*. The number of interpolation nodes, say n, that is needed for obtaining the parameter set is equal to $\dim \Pi_N^l$. Having n large (for l large) leads to the question how to choose an appropriate simplex. For example, in $\mathbb{R}^2$ one needs six interpolation nodes for a quadratic polynomial. If the simplex is a triangle, three additional nodes can be placed at the middle of triangle sides. However, this means that the grid must be regular, i.e. the method cannot be applied for general (scattered) data. Another approach is to use three points closest to the triangle, which requires an additional effort in determining them. However, regardless all difficulties, the interpolation function can be significantly improved, especially considering the smoothness.

The accuracy, expressed in terms of the diameter Δx, is related to the order of polynomial. In [47] it is shown that for any $f \in C^{l+1}$ and the higher derivative norm, defined by

$$\left\|D^l f\right\|_{\infty,\Gamma_m} := \sup_{\mathbf{x}\in\Gamma_m} \sup_{\substack{\mathbf{y}\in\mathbb{R}^N \\ \|\mathbf{y}\|=1}} \left|D_{\mathbf{y}}^l f(\mathbf{x})\right|, \quad l \ge 1, \tag{3.4.34}$$

the interpolation error satisfies

$$r := \|f - p\|_{L_\infty(\Gamma_m)} \le C_{N,l}\, \Delta x^{l+1} \left\|D^{l+1} f\right\|_{\infty,\Gamma_m}. \tag{3.4.35}$$

Here the constant $C_{N,l}$ does not depend on f, Γ_m and the chosen grid. For an interesting special case $l = 2$ and $N \geq 2$, it can be shown (see [48]) that $C_{N,2}$ satisfies

$$\frac{\sqrt{2}}{48} \frac{N\sqrt{N}}{(N+1)\sqrt{N+1}} \leq C_{N,2} \leq \frac{5}{16} \frac{N}{N+1}. \tag{3.4.36}$$

On the other hand, for $N = 2$ and $l \geq 2$ we have (see [47])

$$C_{2,l} = \frac{1}{8(l+1)} \left(\frac{3}{l}\right)^{l+1}. \tag{3.4.37}$$

3.4.3 Surface spline interpolation

Being a member of a large class of so-called *radial basis function* methods, the surface spline interpolation is certainly one of the most important global methods. The interpolation function can be found by using a minimization of some suitable seminorm, say $|||\cdot|||$. An extensive theory is given in [18], [34], [7], [56] and [27] for the general case of spline functions in $\mathbb{R}^N$ with a coefficient l being the order of the smoothness of the interpolation function. Although slightly different spaces are used, all seminorms used are a Sobolev-like seminorm, given by

$$|||f|||_{l,L_p(\Gamma)} := \left\{ \sum_{|\alpha| \leq l} \int_\Gamma |D^\alpha f(\mathbf{x})|^2 \, d\mathbf{x} \right\}^{\frac{1}{2}}, \tag{3.4.38}$$

where $\alpha = [\alpha_1, \ldots, \alpha_l]^T$ is a multi-index and the differentiation operator is defined by $D^\alpha := \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \frac{\partial^{\alpha_2}}{\partial x_2^{\alpha_2}} \cdots \frac{\partial^{\alpha_N}}{\partial x_N^{\alpha_N}}$.

For $N = l = 2$ and under some simplifying assumptions, the seminorm (3.4.38) may be physically interpreted as the bending energy of a thin plate of infinite extent. It can be shown (see e.g. [34]) that solving the interpolation problem by minimizing (3.4.38) leads to the interpolation function $p(\mathbf{x})$, of the following form

$$\begin{aligned} p(\mathbf{x}) &:= \sum_{\mathbf{x}_k \in S} a_k \phi_k(\mathbf{x}) + q(\mathbf{x}), \\ \phi_k(\mathbf{x}) &:= \phi(\mathbf{x} - \mathbf{x}_k), \end{aligned} \tag{3.4.39}$$

where S represents the set of interpolation nodes, ϕ_k is the basis function and $q(\mathbf{x}) \in \Pi_{l-1}$ is a polynomial in $\mathbb{R}^N$ of total degree at most $(l-1)$. Of course, the parameters a_k and the polynomial q must be chosen to satisfy the interpolation criteria

$$p(\mathbf{x}_k) = f(\mathbf{x}_k) = f_k, \quad \forall \mathbf{x}_k \in S. \tag{3.4.40}$$

However, the number of parameters exceeds the number of conditions. The standard way of determining the remaining conditions, which are called *natural boundary conditions*, given by

$$\sum_{\mathbf{x}_k \in S} a_k q(\mathbf{x}_k) = 0, \quad \forall q \in \Pi_{l-1}. \tag{3.4.41}$$

Of course, it is essential that S is $\Pi_{l-1}-$unisolvent. Otherwise the polynomial term can be adjusted by any polynomial which is zero on S. To ensure uniqueness of the interpolation function, one possibility is to restrict ϕ to be strictly conditionally positive definite of order l (see [12]). Frequently ϕ is a radial basis function, i.e.

$$\phi(\mathbf{x} - \mathbf{x}_k) := \phi(\|\mathbf{x} - \mathbf{x}_k\|), \tag{3.4.42}$$

where $\|\cdot\|$ is the Euclidean norm. The choice of ϕ may differ and for the surface spline interpolation it is given by

$$\phi(\|\mathbf{x}\|) = \begin{cases} \|\mathbf{x}\|^{2l-N}, & \text{if } N \text{ is odd,} \\ \|\mathbf{x}\|^{2l-N} \log \|\mathbf{x}\|, & \text{if } N \text{ is even.} \end{cases} \tag{3.4.43}$$

For the special case $N = l = 2$, where $\phi(\|\mathbf{x}\|) = \|\mathbf{x}\|^2 \log \|\mathbf{x}\|$, the interpolation method is usually denoted as *the thin-plate spline interpolation* (see e.g. [39]).

The main results concerning accuracy deal with the error bounds of the interpolation error in terms of the measure of density, i.e. the fill-distance

$$\Delta\xi := \sup_{\mathbf{x}\in\Gamma} \min_{\mathbf{x}\in\Gamma} \|\mathbf{x} - \mathbf{x}_k\| . \tag{3.4.44}$$

The L_p-error bound is given by

$$r := \|f - p\|_{L_p(\Gamma)} \leq C\, \Delta\xi^{\gamma_p}\, |||f|||_{l,L_p(\Gamma)}, \tag{3.4.45}$$

where

$$\gamma_p := \min\left\{ l,\ l - \frac{N}{2} + \frac{N}{p} \right\}. \tag{3.4.46}$$

The constant C depends on Γ and l. Clearly, for the L_1 and L_2-norm, the order of accuracy is determined by the smoothness coefficient l. The relatively low order is due to the fact that the interpolation data is scattered. However, it should be remarked that in an ideal situation of equispaced nodal distribution, the L_p-approximation order is $2l$, a value at least twice γ_p (see [27]).

3.4.4 Biharmonic spline interpolation

For a special case $l = 2$ a surface spline can be found by minimizing the curvature of the given function on the interpolation domain . The curvature is, of course, related to the seminorm (3.4.38) and given by

$$C(p) := \left\{ \int_\Gamma \left(\nabla^2 p(\mathbf{x}) \right)^2 d\mathbf{x} \right\}^{\frac{1}{2}}. \tag{3.4.47}$$

As mentioned in the univariate case, p has a minimum curvature if it satisfies the biharmonic equation. For a given set of interpolation data $\{(\mathbf{x}_k, f_k)\}_{k=0}^n$, $\mathbf{x} \in \Gamma$, the multivariate biharmonic equation reads

$$\nabla^4 p(\mathbf{x}) = \sum_{j=0}^{n} a_j \delta(\mathbf{x} - \mathbf{x}_j), \tag{3.4.48}$$

where $\delta(\cdot)$ is a delta function. The solution of (3.4.48) is

$$p(\mathbf{x}) = \sum_{j=0}^{n} a_j \phi(\|\mathbf{x} - \mathbf{x}_j\|). \tag{3.4.49}$$

Functions ϕ are the Green's functions and given in Table 3.4.1 together with their gradients. One should note that ϕ becomes unbounded for $N \geq 4$ at the origin and the gradient for $N \geq 3$. This means that an application is not straightforward and the problem can probably be solved by shifting Green's functions slightly away from

Dimension N	Green's Function $\phi(\|\mathbf{x}\|)$	Gradient of Green's Function $\nabla\phi(\mathbf{x})$
1	$\|\mathbf{x}\|^3$	$\mathbf{x}\|\mathbf{x}\|$
2	$\|\mathbf{x}\|^2(\log\|\mathbf{x}\|-1)$	$\mathbf{x}(2\log\|\mathbf{x}\|-1)$
3	$\|\mathbf{x}\|$	$\mathbf{x}\|\mathbf{x}\|^{-1}$
4	$\log\|\mathbf{x}\|$	$\mathbf{x}\|\mathbf{x}\|^{-2}$
5	$\|\mathbf{x}\|^{-1}$	$-\mathbf{x}\|\mathbf{x}\|^{-3}$
6	$\|\mathbf{x}\|^{-2}$	$-2\mathbf{x}\|\mathbf{x}\|^{-4}$
N	$\|\mathbf{x}\|^{4-N}$	$(4-N)\,\mathbf{x}\|\mathbf{x}\|^{2-m}$

TABLE 3.4.1. Biharmonic Green's Functions

the data points. Parameters of the interpolation function are determined by solving the linear system

$$p(\mathbf{x}_k) = \sum_{j=0}^{n} a_j \phi\left(\|\mathbf{x}_k - \mathbf{x}_j\|\right), \quad k = 0, \ldots, n. \tag{3.4.50}$$

One important advantage of this method is that one can use the slopes (derivative values) at the interpolation nodes to determine the parameter set. This is of importance in applications where the values of slopes are easier to be obtain than the values themselves. Here the system of linear equations reads

$$w_k := (\nabla p \cdot \mathbf{n})_k = \sum_{j=0}^{n} a_j \nabla\phi\left(\mathbf{x}_k - \mathbf{x}_j\right) \cdot \mathbf{n}_k, \quad k = 0, \ldots, n, \tag{3.4.51}$$

where w_k is the slope in the direction $\mathbf{n}_k$.

To illustrate the accuracy of the methods introduced, we consider the following example.

EXAMPLE 3.4.4. Consider the already mentioned (see Section 3.2) function

$$f(x, y) = x\, e^{-(x^2+y^2)}, \tag{3.4.52}$$

on a domain

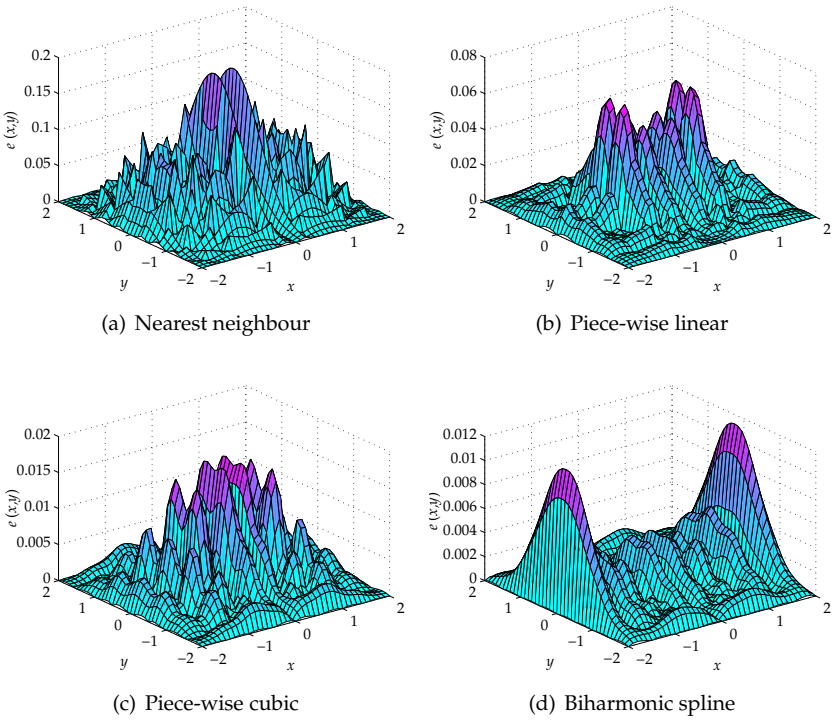
$$\Gamma := \{(x, y) \,|-2 \le x \le 2,\ -2 \le y \le 2\}. \tag{3.4.53}$$

Let us discretise Γ by 9×9 points and then interpolate by applying interpolation methods introduced above. The solutions for various methods are shown in Figure 3.2.1. One should note that their accuracy is different and that higher order methods (as expected) produce better results, see Figure 3.4.2. □

3.5 Inverse interpolation

For purposes to become clear in the subsequent chapters we will need an approximation of the inverse function of $\mathbf{f}$, say $\mathbf{g} = \mathbf{f}^{-1}$, assuming that $\mathbf{f}^{-1}$ exists of course. Hence we require $\mathbf{f}$ to be *injective*, i.e. that for $\mathbf{f} : X \to Y$ and for $\forall \mathbf{x}, \mathbf{y} \in X$, the following holds

$$\mathbf{x} \neq \mathbf{y} \Rightarrow \mathbf{f}(\mathbf{x}) \neq \mathbf{f}(\mathbf{y}). \tag{3.5.1}$$



(a) Nearest neighbour (b) Piece-wise linear

(c) Piece-wise cubic (d) Biharmonic spline

FIGURE 3.4.2. Error of different interpolation methods

Since we encounter derivatives of our function in the interpolation error, we need suitable smoothness. The function $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$ is called a C^n*—diffeomorphism* if $\mathbf{f}$ is injective and if $\mathbf{f}$ and $\mathbf{f}^{-1}$ are continuous maps (i.e. $\mathbf{f}$ is a *homeomorphism*) and both $\mathbf{f}$ and $\mathbf{f}^{-1}$ are C^n, where $n \in \mathbb{N}$. If $\mathbf{f} \in C^\infty$, it is simply called a diffeomorphism. By employing this the existence conditions can be given by *the inverse function theorem* (see e.g. [6] and [3])

THEOREM 3.5.1. *(Inverse Function Theorem) Let* $\Gamma \subset \mathbb{R}^N$ *be an open subset and* $\mathbf{f} : \Gamma \to \hat{\Gamma} \subset \mathbb{R}^N$ *a* C^n *mapping. If* $\mathbf{x}_0 \in \Gamma$ *and the Jacobian matrix* $D\mathbf{f}(\mathbf{x}_0)$ *is non-singular, then there exists an open neighbourhood* U *of* $\mathbf{x}_0$ *in* Γ *such that* $V = \mathbf{f}(U)$ *is an open neighbourhood of* $\mathbf{f}(\mathbf{x}_0)$ *and* $\mathbf{f} : U \to V$ *is a* C^n*—diffeomorphism.*

If $\mathbf{x} \in U$ and $\mathbf{y} = \mathbf{f}(\mathbf{x})$, then we have the following formula for the derivatives of $\mathbf{f}^{-1}$ at $\mathbf{y}$:

$$D\mathbf{f}^{-1}(\mathbf{y}) = (D\mathbf{f}(\mathbf{x}))^{-1}. \tag{3.5.2}$$

The inverse function cannot be obtain explicitly in general. However, it can be approximated by a function, say $\mathbf{p}$, obtained by inverse interpolation. *The inverse interpolation problem* can be introduced similarly to (3.1.2) and (3.1.3), i.e. by defining a set of pairs

$$\hat{\Omega} := \left\{ (\mathbf{f}_k, \mathbf{x}_k) \,\middle|\, \mathbf{f}_k \in \hat{S} \subset \hat{\Gamma} \subseteq \mathbb{R}^N,\ \mathbf{x}_k := \mathbf{g}(\mathbf{f}_k),\ k = 0, \ldots, n \right\}, \tag{3.5.3}$$

where $\mathbf{g} = \mathbf{f}^{-1}$, and the following condition

$$\mathbf{p}(\mathbf{f}_k) := \mathbf{x}_k. \tag{3.5.4}$$

In the inverse interpolation problem we use $\mathbf{p}(\mathbf{x})$ as an approximation of $\mathbf{g} = \mathbf{f}^{-1}$.

Obviously the main difference between the direct and the inverse problem is that the interpolation domain is now the range $\hat{\Gamma}$ of $\mathbf{f}$. This means that the set of "original" data points $S = \{\mathbf{x}_k\}_{k=0}^n$ is mapped by $\mathbf{f}$ into a "new" set $\hat{S} = \{\mathbf{f}_k\}_{k=0}^n$, where clearly $\mathbf{f}_k := \mathbf{f}(\mathbf{x}_k)$. The new set of interpolation data $\hat{S}$ may be used as a set of interpolation nodes that covers $\hat{\Gamma}$. By employing (3.5.4) and some additional conditions, one can construct the interpolation function. Of course, the main problem is the mapping of the grid, since $\mathbf{f}$ is non-linear in general. In other words, $\hat{S}$ is irregular regardless how regular S may be. This means that the interpolation methods to be used here are certainly methods for scattered data. As shown, some of these methods involve the interpolation on a simplex Γ_m. Since the simplex represents an area in general, it has a certain orientation in the original grid. To preserve this orientation it can be shown (see [6]) that $\mathbf{f}$ must satisfy

$$\det[D\mathbf{f}(\mathbf{x})] > 0, \quad \mathbf{x} \in \Gamma_m. \tag{3.5.5}$$

We call the condition (3.5.5) *the topology preservation in space* of the function $\mathbf{f}$. In general, if we have the grid (as a set of simplices) which covers Γ, (3.5.5) should hold on the entire Γ or the grid must be regularized in such a way that (3.5.5) holds on every particular simplex Γ_m, $m = 1, \ldots, n_m$.

There are two main problems involved in the use of inverse interpolation. The first one deals with a situation where $\mathbf{f}$ is known explicitly and the second where $\mathbf{f}$ is given discretely, i.e. by the set of interpolation pairs only. In the first case one needs to discretise Γ by some set $S = \{\mathbf{x}_k\}_{k=0}^n$ and sample $\mathbf{f}$ in S to obtain the function data set $\hat{S} = \{\mathbf{f}_k\}_{k=0}^n$. The set $\hat{S}$ is then used as a set of interpolation nodes. In the second case we can only use points for which the function is known. Although these two cases look rather similar, there is an important difference. By knowing $\mathbf{f}$ explicitly, it allows us to choose S in a way which suits best for the particular application. For example, we can easily add or discard some points. In the other case we cannot change much given S (except discarding some points). Examples where such problems occur, are coming from numerical methods, where $\mathbf{f}$ represents the numerical (thus discrete) solution. Other examples are coming from empirical applications, where $\mathbf{f}$ is the experimental (measured) data.

One typical example of inverse interpolation occurs in solving a nonlinear system of equation

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{3.5.6}$$

Finding a zero of (3.5.6), say $\mathbf{x}^* \in \Gamma$, can be formulated in the following way. Let $\mathbf{g} = \mathbf{f}^{-1}$ be the inverse of $\mathbf{f}$ on Γ. Then

$$\mathbf{x}^* = \mathbf{g}(\mathbf{0}). \tag{3.5.7}$$

We now try to find an approximation of $\mathbf{g}$, say $\mathbf{p}$, to obtain an approximation by means of data $\{(\mathbf{f}_k, \mathbf{x}_k)\}_{k=0}^n$. Then the solution simply follows from

$$\bar{\mathbf{x}}^* = \mathbf{p}(\mathbf{0}). \tag{3.5.8}$$

How close $\bar{\mathbf{x}}^*$ is to $\mathbf{x}^*$ depends on the way how the interpolation domain is chosen (or given), how dense is S in Γ and on the type of the interpolation. This is illustrated by the following two examples.

EXAMPLE 3.5.2. Consider

$$f(x) = x^5 - x^4 + 2x^3 - 2x^2 + 2x - 2 = 0. \tag{3.5.9}$$

It can be verified that f is a diffeomorphism on $\mathbb{R}$, which means that we can choose Γ freely. In this special case we know that the only zero is $x^* = 1$, which allows us to compute the errors of different interpolation methods. Let us discretise $x \in [0, 2]$ by using $n = 20$ equidistant points $x_k = k\Delta x$, $k = 0, \ldots, 19$, where $\Delta x = 0.105$. For all x_k we can obtain corresponding $f_k = f(x_k)$. By using the set of pairs $\{(f_k, x_k)\}_{k=0}^{19}$ we construct different interpolation functions by applying the methods introduced in the previous section. The results are shown in Table 3.5.1. One should note that the accuracy of the various interpolation methods is not necessarily related to the results one would obtain in case of the direct interpolation (as shown later). In particular, the worst result is obtained by Lagrange interpolation, regardless of the fact that a high-order polynomial is constructed.

Interpolation method	Piece-wise linear	Piece-wise cubic	Cubic spline	Lagrange method
$\lvert\bar{x}^* - x^*\rvert$	$4.42\,10^{-3}$	$1.59\,10^{-5}$	$6.3\,10^{-5}$	$4.17\,10^{-2}$

TABLE 3.5.1. The interpolation error of different interpolation methods.

EXAMPLE 3.5.3. Consider the system of nonlinear equations

$$\begin{aligned} f_1(x, y) &= x^3 y - 4 = 0, \\ f_2(x, y) &= (x^2 - 2)(y^2 - 2) = 0, \end{aligned} \tag{3.5.10}$$

assuming $\mathbf{f} = [f_1, f_2]^T$ and $\mathbf{x} = [x, y]^T$. In this special case we see that a (non-unique) solution is given by $\mathbf{x}^* = [x^*, y^*]^T = [\sqrt{2}, \sqrt{2}]^T$. Let us choose

$$\Gamma := \{(x, y) \mid 1 \le x, y \le 1.5\}. \tag{3.5.11}$$

Discretising Γ by 10×10 equispaced points $\mathbf{x}_k = (k\Delta x,\, k\Delta y)$, $k = 0, ..., 9$, we obtain $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k)$. Now, constructing an interpolation function $\mathbf{p}$ on pairs $\{(\mathbf{f}_k, \mathbf{x}_k)\}_{k=0}^{99}$, one can obtain an approximation of the zero of $\mathbf{f}$ as

$$\bar{\mathbf{x}}^* = \mathbf{p}(0, 0). \tag{3.5.12}$$

Employing different interpolation methods, one would expect larger differences in results (as it would be the case if the interpolation is direct). However, in this particular case it is not necessarily true, see Table 3.5.2. This leads to the conclusion that the error of inverse interpolation is not directly proportional to the corresponding error of direct interpolation, as shown hereafter.

Interpolation method	Nearest-neighbour	Piece-wise linear	Piece-wise cubic	Biharmonic spline
$\lVert\bar{\mathbf{x}}^* - \mathbf{x}^*\rVert_2$	$8.7\,10^{-2}$	$2.95\,10^{-2}$	$5.36\,10^{-2}$	$4.98\,10^{-2}$

TABLE 3.5.2. The L_2-interpolation error of different interpolation methods

3.5.1 Accuracy of the inverse interpolation

To estimate the error of the inverse interpolation one can use the error bounds introduced in the previous section for direct interpolation. However, the error bounds depend on derivatives of the interpolating function and the geometrical properties of the data set S. Since the interpolating function is the (generally) unknown inverse function $\mathbf{g}$, the derivatives cannot be assumed to be known and the geometrical properties correspond to the new grid defined by the set $\hat{S}$. Hence, the derivatives of $\mathbf{g}$ need to be related to the derivatives of $\mathbf{f}$ and the geometry of $\hat{S}$ to the geometry of S.

Let us first consider the univariate case. Starting from

$$g\left(f(x)\right) = x, \tag{3.5.13}$$

it easy to show that the following holds

$$g'\left(f(x)\right) = \frac{1}{f'(x)}. \tag{3.5.14}$$

Now the higher derivatives can be obtained by differentiation of (3.5.14) arbitrarily many times with substituting intermediate results. It can be shown that

$$g'' = -\frac{f''}{\left(f'\right)^3}, \quad g''' = -\frac{f''' f' - 3\left(f''\right)^2}{\left(f'\right)^5}, \ldots \tag{3.5.15}$$

Expressions are becoming more complex as the order of differentiation increases, but one should note that in all denominators there is a power of f'. This is important in situations where f' is large, which can lead to a poor accuracy of the direct interpolation. For example, for linear interpolation and a function satisfying $|f'| \leq M$ and $|f''| \leq M$, where $M_1 \leq M \leq M_2$ and M_1 and M_2 are large numbers of the same order, the error of direct interpolation is of order M^2. On the other hand, by applying (3.5.15), the error of the inverse interpolation is of order $1/M^2$. This means that the inverse of the function, which has large derivatives, can be accurately approximated by the inverse interpolation.

In the general vectorial case, the principle remains the same, although obtaining the relation between derivatives is slightly more difficult. From

$$\mathbf{g}\left(\mathbf{f}(\mathbf{x})\right) = \mathbf{x}, \tag{3.5.16}$$

it follows that

$$D\mathbf{g}\, D\mathbf{f} = \mathbf{I}_N, \tag{3.5.17}$$

which is equivalent to (3.5.2), where $\mathbf{I}_N$ is the identity matrix of order N. Further differentiation gives

$$\begin{aligned} D^2\mathbf{g} =& -\left[D\mathbf{f}\right]^{-2} D^2\mathbf{f}\left[D\mathbf{f}\right]^{-1}, \\ D^3\mathbf{g} =& \left[D\mathbf{f}\right]^{-3} D^2\mathbf{f}\left[D\mathbf{f}\right]^{-1} D^2\mathbf{f}\left[D\mathbf{f}\right]^{-1} + 2\left[D\mathbf{f}\right]^{-2} D^2\mathbf{f}\left[D\mathbf{f}\right]^{-2} D^2\mathbf{f}\left[D\mathbf{f}\right]^{-1} - \\ & -\left[D\mathbf{f}\right]^{-3} D^3\mathbf{f}\left[D\mathbf{f}\right]^{-1}. \end{aligned} \tag{3.5.18}$$

Again one should note that the Jacobian matrix $D\mathbf{f}$ is present only via its inverse and similar arguments can be made as in the univariate case, considering the fact that now we have the error bounds related to tensor norms.

Geometrical properties of the new set of the interpolation nodes are highly dependent on the mapping $\mathbf{f}$. For example, in the univariate case any given interval $[x_a, x_b]$

of length $\Delta x := |x_b - x_a|$ is mapped to a new interval, say $[\hat{x}_a, \hat{x}_b]$ with length $\Delta\hat{x} := |\hat{x}_b - \hat{x}_a|$. Since $\hat{x}_a$ and $\hat{x}_b$ are the maps of x_a and x_b, we have

$$\Delta\hat{x} = |\hat{x}_b - \hat{x}_a| = |f(x_b) - f(x_a)|. \tag{3.5.19}$$

By applying the mean value theorem

$$f(x_b) - f(x_a) = (x_b - x_a)\ f'(\tilde{x}), \quad \tilde{x} \in [x_a, x_b], \tag{3.5.20}$$

it follows that

$$\Delta\hat{x} = |(x_b - x_a)\ f'(\tilde{x})| = |x_b - x_a|\ |f'(\tilde{x})| = \Delta x\ |f'(\tilde{x})|, \tag{3.5.21}$$

i.e.

$$\Delta\hat{x} \leq \Delta x \sup_{x \in [x_a, x_b]} |f'(x)|. \tag{3.5.22}$$

Similar results can also be obtained in the general vectorial case. Here the distance between two points should be related to the diameter of the simplex, introduced in piece-wise interpolation methods, or the fill-distance present in the surface spline interpolation. For two points $\mathbf{x}_a, \mathbf{x}_b \in S \subset \Gamma$, with a distance $\Delta x := \|\mathbf{x}_b - \mathbf{x}_a\|$ and corresponding maps $\hat{\mathbf{x}}_a, \hat{\mathbf{x}}_b \in \hat{S} \subset \hat{\Gamma}$, we have that the related distance in $\hat{\Gamma}$ is

$$\Delta\hat{x} := \|\hat{\mathbf{x}}_b - \hat{\mathbf{x}}_a\| = \|\mathbf{f}(\mathbf{x}_b) - \mathbf{f}(\mathbf{x}_a)\|. \tag{3.5.23}$$

Applying the multivariate mean value theorem (see e.g. [3])

$$\mathbf{f}(\mathbf{x}_b) - \mathbf{f}(\mathbf{x}_a) = D\mathbf{f}(\tilde{\mathbf{x}})\ (\mathbf{x}_b - \mathbf{x}_a), \tag{3.5.24}$$

where $\tilde{\mathbf{x}} \in \Gamma$ lies on the line connecting $\mathbf{x}_a$ and $\mathbf{x}_b$, we have

$$\Delta\hat{x} = \|(\mathbf{x}_b - \mathbf{x}_a)\ D\mathbf{f}(\tilde{\mathbf{x}})\| \leq \|\mathbf{x}_b - \mathbf{x}_a\|\ \|D\mathbf{f}(\tilde{\mathbf{x}})\| = \Delta x\ \|D\mathbf{f}(\tilde{\mathbf{x}})\|. \tag{3.5.25}$$

If Δx is the diameter of a necessarily convex simplex Γ_m, we have $\tilde{\mathbf{x}} \in \Gamma_m$ and $\Delta\hat{x}$ satisfies

$$\Delta\hat{x} \leq \Delta x \sup_{\mathbf{x} \in \Gamma_m} \|D\mathbf{f}(\mathbf{x})\|. \tag{3.5.26}$$

Clearly the simplex of the new grid can be significantly larger than the original simplex. This can, of course, decrease the accuracy if $\|D\mathbf{f}(\mathbf{x})\|$ is large.

The error bounds given in previous section are obtained for a scalar function $f : \mathbb{R}^N \to \mathbb{R}$. For inverse interpolation we necessarily have a vectorial map $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$, i.e. to use the error bounds, the involved norms need to be modified. However, since the interpolation function $\mathbf{g} = [g_1, \ldots, g_N]^T$ is defined on the same domain $\hat{\Gamma}$ element-wise, the interpolation error can be defined simply by taking $\infty-$norm over all g_l, $l = 1, \ldots, N$. Here we have

$$r := \|\mathbf{g} - \mathbf{p}\|_\infty = \max_l |||g_l - p_l|||, \tag{3.5.27}$$

where $|||\cdot|||$ represents any given norm introduced previously.

CHAPTER 4

Introduction to the flow method

In this chapter we introduce our method for solving autonomous (stiff) flow problems. The fact that the flow is autonomous, a common situation in many physical applications, is crucial. It implies that the velocity field is known in a certain spatial domain, once it is known at a given time-level. In order to find it at other (spatial) points, we use inverse interpolation. These two aspects (employing autonomy and interpolation) make that we can use implicit methods to discretise the ODE. In this chapter we will concentrate on the Euler Backward method, as it is the simplest one to demonstrate our algorithmic approach and the error analysis is not too complex. Moreover, we will restrict ourselves to scalar problems, which is more accessible to analysis. Most of the ideas can be generalised to higher order methods and to vectorial problems as well. This will be done in the subsequent chapters.

In Section 4.1 we first give an outline of the mathematical problem. In this section we also describe the basic idea behind the method. Since inverse interpolation is an essential ingredient we briefly discuss this in Section 4.2. The numerical solution curves found at the various time points should, of course, not "intersect" in any reasonable setting (as is e.g. implied by Lipschitz continuity). This leads to a condition which we will call *well-posedness*, see Section 4.3. The above-mentioned interpolation introduces a more complicated local error than usual. Therefore an error analysis is given in Section 4.4, where the local error is estimated. The most interesting aspect of our method is its stability. In Section 4.5 it is shown that we obtain stability behaviour similar to that of the Euler Backward method, despite the fact that our method is de facto explicit. Practical aspects of the method are given in Section 4.6. There is a variety of possibilities to generalize the ideas on which this method is based and can be applied in other cases, hence we consider a few. In this section we finally also give a number of numerical examples showing the quality of the method.

4.1 Outline of the Method

Consider the autonomous ODE

$$\dot{x} = u(x), \tag{4.1.1}$$

where $\dot{x}$ represents the time derivative of x. We assume that u is known at a discrete set of points only. Let $S = \{x_k\}_{k=0}^{n}$ be such a set of points for which u values are given by the set $\{u_k\}_{k=0}^{n}$. It is assumed that u is Lipschitz continuous. As in Section 2.2, we

call $I(x,t) \subset \mathbb{R}$ a flow of (4.1.1), if it represents a continuum of solutions x of (4.1.1), such that at time t each $x(t) = I(x(0),t)$. In particular, we assume $I(x,t)$ to be finite; so there are boundary points, say $m(t)$ and $M(t)$, such that

$$m(t) \leq x(t) \leq M(t). \tag{4.1.2}$$

The full system is then formally described by

$$\begin{cases} \dot{x} = u(x), \\ x(0) = x^0 \in [m(0), M(0)]. \end{cases} \tag{4.1.3}$$

To find an approximate solution of (4.1.3) and thus have an approximation of $I(x,t)$, one can use any existing numerical method. In this chapter, as an introduction, we will restrict ourselves to the Euler Backward (EB) method, with a fixed step size h, for ease of argument. However, in the following chapters higher order methods will also be introduced. We denote by x^i the approximation of $x\left(t^i\right)$, at $t^i := i\,h$. Then the EB method reads

$$x^{i+1} = x^i + h\,u\left(x^{i+1}\right). \tag{4.1.4}$$

Let us define

$$f\left(x^{i+1}\right) := x^{i+1} - h\,u\left(x^{i+1}\right) = x^i. \tag{4.1.5}$$

Since the solution at t^{i+1} cannot be found directly, at least when f (i.e. u) is nonlinear, one needs to solve (4.1.4) by iteration. As already mentioned (see Chapter 2) this is typically done by the Newton iterative method (or its modifications). However, due to the fact that u is not explicitly known, the Newton method cannot be directly applied. Instead of solving x^{i+1} by an iterative method, we do the following. At time-level t^{i+1}, define the point

$$\hat{x}^{i+1} := x^i. \tag{4.1.6}$$

Since (4.1.1) is autonomous, we clearly have

$$u\left(\hat{x}^{i+1}\right) = u\left(x^i\right). \tag{4.1.7}$$

Now we can view $\hat{x}^{i+1}$ as the point that would have been obtained by the EB method if we would have started at $\hat{x}^i$, defined by

$$\hat{x}^i := \hat{x}^{i+1} - h\,u\left(\hat{x}^{i+1}\right) = x^i - h\,u\left(x^i\right). \tag{4.1.8}$$

For a general point at time-level t^{i+1} the functional dependence with respect to its corresponding solution value at t^i is unknown analytically, so we have to use approximate values.

The flow represents the time evolution of a continuum of the solution points. Hence, in order to find a numerical solution, one needs to spatially discretise $I\left(x,t^i\right)$, of course only at time discretisation points t^i, $i = 0,1,\ldots$ This discretisation should provide a "good enough" representation of the flow, meaning that the set of the flow points, say $I^i = \left\{x_j^i\right\}_{j=0}^{\mu}$, should be sufficiently dense. In the scalar case this looks unimportant, since all flow points should lie between boundary points. However, in some applications the time evolution of the inner points may be of interest and, moreover, the method can also be applied for higher dimensional problems, where the boundary is not so simple anymore. Hence we assume that the flow at time point t^i is represented by the set I^i. Considering the initial condition, we let $\left\{x_j^0\right\}_{j=0}^{\mu}$ be

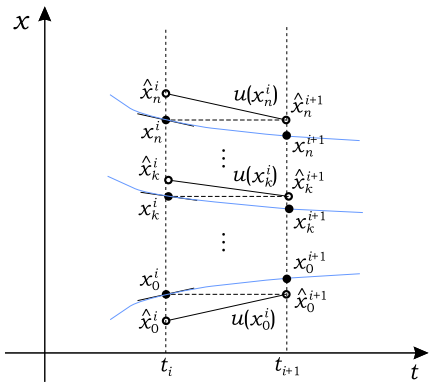


FIGURE 4.1.1. The principle of the flow method.

a set of μ ordered points in $I(x,0)$, such that $x_1^0 = m(0)$ and $x_n^0 = M(0)$. Then we expect $\left\{x_j^i\right\}_{j=0}^{\mu}$ to be a similarly ordered set of points, approximately in I^i, i.e.

$$m\left(t^i\right) \doteq x_1^i < x_2^i < \ldots < x_\mu^i \doteq M\left(t^i\right). \tag{4.1.9}$$

Note that this is a consequence of well-posedness of u (following from Lipschitz continuity) and can be achieved because of consistency of the EB method (i.e. errors small enough for h small enough).

Points $\{x_k\}_{k=0}^n$ for which u values are known may or may not be related to the flow points, depending on the problem under consideration. To keep things simple, in this (introductory) chapter, we will assume that the set S at t^i is equal to I^i, i.e. we can write $x_k^i = x_j^i$, $j,k = 0,\ldots,n$. However, some examples will be shown later, where the set of interpolation points is not directly related to the set of the flow points.

Assuming $S = I^i$ and by applying (4.1.8) to the set $\left\{x_k^i\right\}_{k=0}^n$, we obtain a set of points $\hat{S} := \left\{\hat{x}_k^i\right\}_{k=0}^n$, which can be used to find solutions at time-level t^{i+1}, see Figure 4.1.1. A particular value in $I\left(x,t^{i+1}\right)$, can be found by interpolation. Note that we need to know the velocity field only at this discrete set of points $\left\{x_k^i\right\}_{k=0}^n$, i.e. it is not necessary to know $u(x)$ everywhere in $I\left(x,t^i\right)$.

We will refer to the above-described method as *the flow method*. There is no specific preference for the interpolation method, apart from the requirement that the resulting approximation errors should be commensurate with the discretisation error. Of course, to keep the method effective, it makes sense to apply this interpolation to some sufficiently dense subset of $\left\{\hat{x}_k^i\right\}_{k=0}^n$. The interpolation issue is the main topic of the following section.

4.2 Interpolation

If we assume that $f(x)$, defined by (4.1.5) satisfies the conditions of the inverse-function theorem (see e.g. [3]), i.e. in particular if $f'(x) \neq 0$, then

$$g(x) := f^{-1}(x), \tag{4.2.1}$$

exists and we may write

$$x^{i+1} = g\left(x^i\right). \tag{4.2.2}$$

Of course, the function g is not known in general. Instead of trying to find x^{i+1} by (Newton) iteration on f, we do the following: Since we can at least find some values at time-level t^i, for which the EB values are known, we can use inverse interpolation to obtain an approximation of g, say p, by defining

$$\hat{\Omega} := \left\{ \left(\hat{x}_k^i, x_k^i\right) \middle| \hat{x}_k^i \in \hat{S},\ g\left(\hat{x}_k^i\right) = x_k^i,\ k = 0, \ldots, n \right\},$$

$$p\left(\hat{x}_k^i\right) = g\left(\hat{x}_k^i\right) = x_k^i, \tag{4.2.3}$$

Then the solution of a particular flow point at the next time-level follows from

$$x_k^{i+1} := p\left(x_k^i\right) \doteq g\left(x_k^i\right). \tag{4.2.4}$$

The interpolation should preferably be done locally. For example, for obtaining x_k^{i+1} (k fixed) we can use the point $\hat{x}_k^i$ plus $(m-1)$ additional points ($m \leq n$) closest to $\hat{x}_k^i$, i.e. $\hat{x}_{k+1}^i$, $\hat{x}_{k-1}^i$, $\hat{x}_{k+2}^i$,... as defined by (4.1.8). If the interpolation is linear, this approach (as shown later) gives results identical to one obtained by applying one step of the discrete Newton method with x_k^i and x_{k+1}^i as starting values.

The interpolation error, say $r(x)$, depends, of course, on the type and the order of the interpolation technique involved, as well as on how $I\left(t^i\right)$ is discretised. For example, Lagrangian interpolation of order m gives

$$r(x) := \frac{g^{(m)}(c)}{m!} \prod_{j=0}^{m} (x - \hat{x}_{k-\lceil \frac{m}{2} \rceil + j}^i). \tag{4.2.5}$$

Since interpolation is inverse, the interpolation error depends on the derivatives of g. However, by substituting (4.1.5) into (3.5.14) and (3.5.15), one can obtain higher order derivatives of g in terms of the higher order derivatives of u. For example

$$g' = \frac{1}{(1 - h\,u')}, \quad g'' = h\,\frac{u''}{(1 - h\,u')^3}, \quad g''' = h\,\frac{u'''\,(1 - h\,u') + 3\,h\,(u'')^2}{(1 - h\,u')^5}, \ldots, \tag{4.2.6}$$

where

$$u' := \frac{du}{dx}, \quad u'' := \frac{d^2u}{dx^2}, \ldots \tag{4.2.7}$$

Although expressions for higher derivatives of g are becoming more complex, one should note that powers of $(1 - h\,u')$ are present in all denominators in (4.2.6). Hence we can conclude that if

$$|1 - h\,u'(x)| \gg 1, \tag{4.2.8}$$

and u has bounded higher derivatives for $x(t) \in I(t)$, the interpolation error tends to zero. The condition (4.2.8) is typically fulfilled for stiff problems, meaning that it is reasonable to expect small interpolation errors.

EXAMPLE 4.2.1. To show how this interpolation works out for a simple situation, consider a problem where (4.1.3) is linear, u is known at the flow points and the interpolation is linear. So, let

$$\begin{cases} \dot{x} = \lambda\,x, \\ x(0) \in I(0) = [m(0), M(0)]. \end{cases} \tag{4.2.9}$$

The set $\{\hat{x}_k^i\}_{k=1}^n$ is then defined by

$$\hat{x}_k^i := \hat{x}_k^{i+1} - h\,u\left(\hat{x}_k^{i+1}\right) = (1 - h\lambda)\,x_k^i. \tag{4.2.10}$$

By linear interpolation we have

$$x_k^{i+1} = p(x_k^i) = \frac{x_k^i - \hat{x}_{k+1}^i}{\hat{x}_k^i - \hat{x}_{k+1}^i}\,\hat{x}_k^{i+1} + \frac{x_k^i - \hat{x}_k^i}{\hat{x}_{k+1}^i - \hat{x}_k^i}\,\hat{x}_{k+1}^{i+1}. \tag{4.2.11}$$

After substituting (4.1.6) and (4.2.10) into (4.2.11), we find

$$x_k^{i+1} = \frac{(x_k^i - x_{k+1}^i + h\lambda\,x_{k+1}^i)\,x_k^i - h\lambda\,x_k^i\,x_{k+1}^i}{(1 - h\lambda)\,(x_k^i - x_{k+1}^i)}, \tag{4.2.12}$$

leading to

$$x_k^{i+1} = \frac{x_k^i}{1 - h\lambda}. \tag{4.2.13}$$

This is exactly the same expression as one would have obtained from directly applying the EB method. This result is no surprise, of course, since linear interpolation should be exact for a linear function. □

4.3 Well-posedness of the Flow Method

As stated in Section 4.1 we assume Lipschitz continuity of the function u. Considering the flow problem (4.1.3), this means that integral curves of the exact solutions will not intersect (cf. [26]). We will call this *the well-posedness*. This property should be inherited by the numerical method, i.e. it should be such that the elements in the set $\left\{x_j^{i+1}\right\}_{j=0}^{\mu}$ have the same ordering as in $\{x_j^i\}_{j=0}^{\mu}$. In this section we will seek a condition for well-posedness of the flow method, i.e. for the topology preservation in time of points in the flow. However, before we analyse the well-posedness property of the EB method, we first need to find a condition for the topology preservation in space, which is necessary for correct interpolation. This condition is related to the requirement that the topology of the set $\{\hat{x}_k^i\}_{k=0}^n$ should be preserved from the topology of the set $\{\hat{x}_k^{i+1}\}_{k=0}^n$. So let (4.1.9) hold and again let $x_j^i = x_k^i$ and $\mu = n$, then from (4.1.6) we have

$$\hat{x}_0^{i+1} < \hat{x}_1^{i+1} < \ldots < \hat{x}_n^{i+1}. \tag{4.3.1}$$

Having a proper ordering of the corresponding solution points at t^i, i.e. topology preservation in space, means

$$\hat{x}_k^i < \hat{x}_{k+1}^i, \quad k = 0, \ldots, n-1. \tag{4.3.2}$$

Recalling (4.1.8) we see that inequality (4.3.2) is equivalent to

$$\hat{x}_k^{i+1} - h\,u(\hat{x}_k^{i+1}) < \hat{x}_{k+1}^{i+1} - h\,u(\hat{x}_{k+1}^{i+1}), \tag{4.3.3}$$

i.e.

$$x_k^i - h\,u(x_k^i) < x_{k+1}^i - h\,u(x_{k+1}^i). \tag{4.3.4}$$

Since $x_{k+1}^i - x_k^i > 0$, this reduces to

$$1 - h\,\frac{u(x_{k+1}^i) - u(x_k^i)}{x_{k+1}^i - x_k^i} > 0. \tag{4.3.5}$$

If $u(x)$ is monotone and smooth enough for $x^i \in \left[x_k^i, x_{k+1}^i\right]$, we can approximate the second term on the left of (4.3.5) by the first derivative $h\,u'\left(x_k^i\right)$. For well-posedness we thus require

$$1 - h\,u'\left(x_k^i\right) > 0, \quad x^i \in I^i. \tag{4.3.6}$$

The term $(1 - h\,u')$ in (4.3.6) was also encountered in the denominators of (4.2.6). Clearly, one should not expect inverse interpolation to provide accurate results when this expression is close to zero. This means that, for a divergent flow, the constraint (4.3.6) might be strict. However, this constraint is always fulfilled for a convergent flow, i.e. for all monotonically non-increasing $u(x)$. This is important for stiff problems, i.e. for problems where $h\,u'\left(x^i\right) \ll -1$.

Condition (4.3.6), obtained for the employed EB method, will also ensure that numerically obtained solutions of the flow do not intersect. This can be seen e.g. for linear interpolation, where the flow points are the interpolation points as well. If we subtract two neighbouring solutions, say

$$\begin{aligned} x_{k+1}^{i+1} &= x_{k+1}^i - h\,\frac{u\left(x_{k+1}^i\right)}{1 - h\,\frac{u\left(x_{k+1}^i\right) - u\left(x_k^i\right)}{x_{k+1}^i - x_k^i}}, \\ x_k^{i+1} &= x_k^i - h\,\frac{u\left(x_k^i\right)}{1 - h\,\frac{u\left(x_{k+1}^i\right) - u\left(x_k^i\right)}{x_{k+1}^i - x_k^i}}, \end{aligned} \tag{4.3.7}$$

then

$$\begin{aligned} x_{k+1}^{i+1} - x_k^{i+1} &= x_{k+1}^i - x_k^i - h\,\frac{u\left(x_{k+1}^i\right) - u\left(x_k^i\right)}{1 - h\,\frac{u\left(x_{k+1}^i\right) - u\left(x_k^i\right)}{x_{k+1}^i - x_k^i}}, \\ &= \frac{x_{k+1}^i - x_k^i}{1 - h\,\frac{u\left(x_{k+1}^i\right) - u\left(x_k^i\right)}{x_{k+1}^i - x_k^i}}, \end{aligned} \tag{4.3.8}$$

which is positive as $x_{k+1}^i - x_k^i > 0$ and assuming (4.3.5) is satisfied. This is illustrated by the following example.

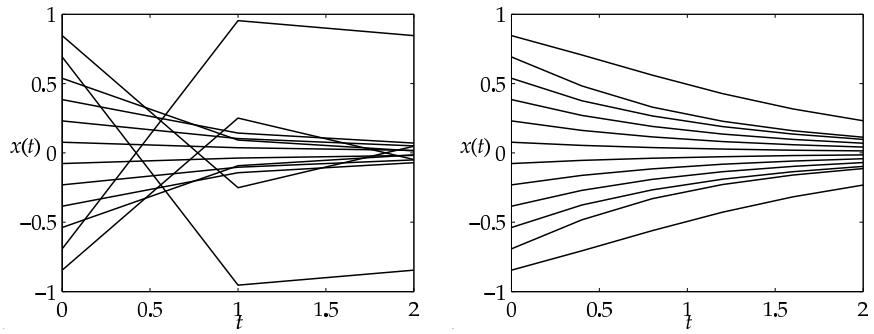
EXAMPLE 4.3.1. Consider

$$\begin{cases} \dot{x} = x\,(x-1)\,(x+1), \\ I^0 = [-1, 1]. \end{cases} \tag{4.3.9}$$

The right-hand side of (4.3.9) is a third order polynomial in x, which changes sign on $[-1, 1]$. Note that u is increasing for $x \in \left(-1, -\frac{1}{\sqrt{3}}\right) \cup \left(\frac{1}{\sqrt{3}}, 1\right)$ and decreasing for $x \in \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$. Since $\max_{x \in I^0} u'(x) = 2$ we find that the condition for the well-posedness of the flow method (4.3.6) requires the time step size h to be smaller then 0.5. For larger h, we may expect the numerical integral curves to intersect for points where $u'(x) > 0$, see Figure 4.3.1a. Note that for points where $u'(x) < 0$, intersections will not occur no matter how large is h. If we use a step size $h < 0.5$, integral curves do not intersect, see Figure 4.3.1b. □

For problems where the set of interpolation points is not directly related to the flow points, the conclusion is similar. First, let x_j^i and x_{j+1}^i be two arbitrary neighbouring flow points that satisfy

$$x_j^i < x_{j+1}^i. \tag{4.3.10}$$



(a) $h = 1.0$ (b) $h = 0.4$

FIGURE 4.3.1. Well-posedness of the flow method

Let the topology preservation in space condition (4.3.5) be satisfied and x_j^i, $x_{j+1}^i \in [\hat{x}_k, \hat{x}_{k+1}]$, i.e. both flow points be in the same subinterval. Then similar to (4.3.8) one can obtain

$$x_{j+1}^{i+1} - x_j^{i+1} = \frac{x_{j+1}^i - x_j^i}{1 - h\frac{u(x_{k+1}) - u(x_k)}{x_{k+1} - x_k}}, \tag{4.3.11}$$

which is again positive since (4.3.10) and (4.3.5) hold. If $x_j^i \in [\hat{x}_k, \hat{x}_{k+1}]$ and $x_{j+1}^i \in [\hat{x}_l, \hat{x}_{l+1}]$, and

$$\hat{x}_k < \hat{x}_{k+1} < \hat{x}_l < \hat{x}_{l+1}, \tag{4.3.12}$$

we have from the topology preservation in space property

$$x_k < x_{k+1} < x_l < x_{l+1}. \tag{4.3.13}$$

Using (4.6.2) we then find

$$x_j^{i+1} < x_{j+1}^{i+1}. \tag{4.3.14}$$

Hence we may conclude that in any case, the flow method is well-posed if the topology in space is preserved.

4.4 Error Analysis

Clearly, the local error of the flow method consists of two components. The first one is the local discretisation error arising from the time discretisation. The second one comes from the inverse interpolation. We will assume well-posedness, as defined in the previous section. Recall that the local discretisation error of the EB method applied to (4.1.1) is given by (cf. e.g. [33])

$$d_h\left(x_k\left(t^i\right)\right) := -\frac{h}{2}\ddot{x}_k\left(t^i\right) + O\left(h^2\right). \tag{4.4.1}$$

We restrict ourselves to linear interpolation, for the sake of simplicity.

The local error is found from substituting an exact solution into a numerical scheme, assuming the solution to be known exactly at $t = t^i$. So take

$$\hat{x}_k\left(t^{i+1}\right) := \hat{x}_k^{i+1} = x_k\left(t^i\right) := x_k^i, \quad k = 0, \ldots, n. \tag{4.4.2}$$

Let us first assume that $S = I^i$ and that interpolation is linear ($m = 2$). Then one finds that the interpolation polynomial, as a function of the exact solution $x_k(t)$, is given by

$$
\begin{aligned}
(4.4.3) \qquad p\left(x_k\left(t^i\right)\right) &= \frac{x_k\left(t^i\right)-\hat{x}_k^i}{\hat{x}_{k+1}^i-\hat{x}_k^i}\,\hat{x}_{k+1}^{i+1} + \frac{x_k\left(t^i\right)-\hat{x}_{k+1}^i}{\hat{x}_k^i-\hat{x}_{k+1}^i}\,\hat{x}_k^{i+1} \\
&= \frac{h\,\dot{x}_k\left(t^i\right)x_{k+1}\left(t^i\right)-\left[x_k\left(t^i\right)-x_{k+1}\left(t^i\right)+h\,\dot{x}_{k+1}\left(t^i\right)\right]x_k\left(t^i\right)}{x_{k+1}(t^i)-x_k(t^i)-h\left[\dot{x}_{k+1}(t^i)-\dot{x}_k(t^i)\right]} \\
&= \frac{x_k\left(t^i\right)\left[x_{k+1}\left(t^i\right)-x_k\left(t^i\right)\right]+h\left[\dot{x}_k\left(t^i\right)x_{k+1}\left(t^i\right)-x_k\left(t^i\right)\dot{x}_{k+1}\left(t^i\right)\right]}{x_{k+1}(t^i)-x_k(t^i)-h\left[\dot{x}_{k+1}(t^i)-\dot{x}_k(t^i)\right]}.
\end{aligned}
$$

After adding and subtracting $\dot{x}_k\left(t^i\right)\,x_k\left(t^i\right)$ in the numerator of the last equality of (4.4.3), the second term can be rewritten as

$$
\begin{aligned}
(4.4.4) \qquad & h\left[\dot{x}_k\left(t^i\right)\,x_{k+1}\left(t^i\right)-x_k\left(t^i\right)\,\dot{x}_{k+1}\left(t^i\right)\right] = \\
& h\left[\dot{x}_k\left(t^i\right)\,\left(x_{k+1}\left(t^i\right)-x_k\left(t^i\right)\right)-x_k\left(t^i\right)\,\left(\dot{x}_{k+1}\left(t^i\right)-\dot{x}_k\left(t^i\right)\right)\right].
\end{aligned}
$$

Using (4.4.4) and the well-posedness, i.e. $x_{k+1}\left(t^i\right)-x_k\left(t^i\right)\neq 0$, (4.4.3) becomes

$$
(4.4.5) \qquad p\left(x_k\left(t^i\right)\right) = \frac{x_k\left(t^i\right)+h\left[\dot{x}_k\left(t^i\right)-x_k\left(t^i\right)\,\frac{\dot{x}_{k+1}\left(t^i\right)-\dot{x}_k\left(t^i\right)}{x_{k+1}(t^i)-x_k(t^i)}\right]}{1-h\,\frac{\dot{x}_{k+1}(t^i)-\dot{x}_k(t^i)}{x_{k+1}(t^i)-x_k(t^i)}}.
$$

We introduce the following simpler notation (note that $\dot{x}_k\left(t^i\right)=u\left(x_k\left(t^i\right)\right)$)

$$
(4.4.6) \qquad \frac{\dot{x}_{k+1}\left(t^i\right)-\dot{x}_k\left(t^i\right)}{x_{k+1}\left(t^i\right)-x_k\left(t^i\right)} = \frac{u\left(x_{k+1}\left(t^i\right)\right)-u\left(x_k\left(t^i\right)\right)}{x_{k+1}\left(t^i\right)-x_k\left(t^i\right)} =: \frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}.
$$

This leads to

$$
\begin{aligned}
p\left(x_k\left(t^i\right)\right) &= \frac{x_k\left(t^i\right)+h\left[\dot{x}_k\left(t^i\right)-x_k\left(t^i\right)\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}\right]}{1-h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}} \\
(4.4.7) \qquad &= x_k\left(t^i\right)+h\,\frac{\dot{x}_k\left(t^i\right)}{1-h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}}.
\end{aligned}
$$

One should note that the expression (4.4.7), under assumption (4.4.2) is identical to the expression for obtaining the numerical solution of the EB method by employing one step of the discrete Newton iterative method, with starting values x_k^i and x_{k+1}^i.

Now the local error of the flow method, say $\delta\left(x_k\left(t^{i+1}\right),h\right)$, is found as the residual, i.e. from substituting the exact solution in $x_k^{i+1}=p\left(x_k^i\right)$. We then find

$$
(4.4.8) \qquad \delta\left(x_k\left(t^{i+1}\right),h\right) = \left[x_k\left(t^{i+1}\right)-x_k\left(t^i\right)-h\,\frac{\dot{x}_k\left(t^i\right)}{1-h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}}\right]/h.
$$

Assuming $\left|h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}\right|$ to be small enough for h sufficiently small, we may use the approximation

$$
(4.4.9) \qquad \frac{1}{1-h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}} = 1+h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}+O\left(h^2\right).
$$

This gives

$$\begin{aligned}\delta\left(x_k\left(t^{i+1}\right),h\right) &= \left[x_k\left(t^{i+1}\right)-x_k\left(t^i\right)\right.\\ &- \left. h\,\dot{x}_k\left(t^i\right)\left[1+h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}+O\left(h^2\right)\right]\right]/h. \qquad (4.4.10)\end{aligned}$$

For a sufficiently smooth solution, one can expand the solution at the time-level t^{i+1} as

$$x_k\left(t^{i+1}\right)=x_k\left(t^i\right)+h\,\dot{x}_k\left(t^i\right)+\frac{h^2}{2}\,\ddot{x}_k\left(t^i\right)+O\left(h^3\right). \qquad (4.4.11)$$

If we substitute this in (4.4.10), we have

$$\delta\left(x_k\left(t^{i+1}\right),h\right)=\frac{h}{2}\,\ddot{x}_k\left(t^i\right)-h\,\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}\,\dot{x}_k\left(t^i\right)+O\left(h^2\right). \qquad (4.4.12)$$

Since $u\left(x_{k+1}\left(t^i\right)\right)=u\left(x_k\left(t^i\right)+\Delta x_k\left(t^i\right)\right)$, expansion around $x_k\left(t^i\right)$ leads to

$$\frac{\Delta u_k\left(t^i\right)}{\Delta x_k\left(t^i\right)}=u'\left(x_k\left(t^i\right)\right)+\frac{\Delta x_k\left(t^i\right)}{2}\,u''\left(x_k\left(t^i\right)\right)+O\left(\Delta x_k^2\left(t^i\right)\right). \qquad (4.4.13)$$

Now the local error is found to be

$$\begin{aligned}\delta\left(x_k\left(t^{i+1}\right),h\right) &= \frac{h}{2}\,\ddot{x}_k\left(t^i\right)-h\,u'\left(x_k\left(t^i\right)\right)\,\dot{x}_k\left(t^i\right)\\ &- \frac{h}{2}\,\Delta x_k\left(t^i\right)\,u''\left(x_k\left(t^i\right)\right)\,\dot{x}_k\left(t^i\right)\\ &+ O(h)\,O\left(\Delta x_k^2\left(t^i\right)\right)+O\left(h^2\right), \qquad (4.4.14)\end{aligned}$$

i.e.

$$\begin{aligned}\delta\left(x_k\left(t^{i+1}\right),h\right) &= -\frac{h}{2}\,\ddot{x}_k\left(t^i\right)-\frac{h}{2}\,\Delta x_k\left(t^i\right)\,u''\left(x_k\left(t^i\right)\right)\,\dot{x}_k\left(t^i\right)\\ &+ O(h)\,O\left(\Delta x_k^2\left(t^i\right)\right)+O\left(h^2\right). \qquad (4.4.15)\end{aligned}$$

Comparing right-hand sides of (4.4.15) and (4.4.1), we see that the expression of the local error of the flow method, contains two additional terms, coming from the interpolation. However, it is clear that the consistency order 1 is preserved. The second term on the right-hand side of (4.4.15), contains the second derivative of the function $u(x)$ with respect to the variable x. This is a consequence of the linear inverse Lagrangian interpolation, as shown in Section 4.2. Clearly, if the order of the interpolation is higher, higher derivatives $u^{(m)}(x)$, $m=3,4,\ldots$ are coming into play and their influence in the local error can become dominant. The expression (4.4.15) can be seen as a sum of the local discretisation error of the EB method and the interpolation error, viz.

$$\delta\left(x_k\left(t^{i+1}\right),h\right)=d_h\left(x_k\left(t^i\right)\right)+r_h\left(x_k\left(t^i\right)\right). \qquad (4.4.16)$$

By neglecting higher order terms and recalling (4.4.2), the interpolation error can be estimated as

$$r_{h,k}^i=-\frac{h}{2}\,\Delta x_k^i\,u''\left(x_k^i\right)\,u\left(x_k^i\right), \qquad (4.4.17)$$

which means that we take $r_{h,k}^i \doteq r_h(x_k^i)$, which is defined by (4.2.5) for $m=2$. We can conclude from (4.4.15) that the local error is $O(h)\,\left(1+O\left(\Delta x_k\left(t^i\right)\right)\right)$. From this it appears that interpolation error is not extremely important, as long as it is not too large as compared to 1 rather than to h.

The accuracy strongly depends on the interpolation method used. For example if we apply piece-wise linear interpolation we can obtain a somewhat different, yet useful error estimate. Let us also assume that the set $S = \{x_k\}_{k=0}^{n}$, is time invariant, i.e. used at every time-level. Repeating the similar procedure as (4.4.3)-(4.4.7), we now have

$$p\left(x_j\left(t^i\right)\right) = \frac{x_j\left(t^i\right) + h\left[u\left(x_k\right) - x_k\frac{\Delta u_k}{\Delta x_k}\right]}{1 - h\frac{\Delta u_k}{\Delta x_k}}. \tag{4.4.18}$$

Here $x_j\left(t^i\right)$ represents the flow point and x_k and x_{k+1} are the points determined so that the numerical solution satisfies

$$\begin{aligned} \hat{x}_k \leq \quad & x_j^i \quad \leq \hat{x}_{k+1}, \\ x_k \leq \quad & x_j^{i+1} \quad \leq x_{k+1}. \end{aligned} \tag{4.4.19}$$

Hence the local error is now

$$\begin{aligned} \delta\left(x_j\left(t^{i+1}\right), h\right) &= \left[x_j\left(t^{i+1}\right) - \frac{x_j\left(t^i\right) + h\left[u\left(x_k\right) - x_k\frac{\Delta u_k}{\Delta x_k}\right]}{1 - h\frac{\Delta u_k}{\Delta x_k}}\right]/h \\ &= \left[\frac{x_j\left(t^{i+1}\right) - x_j\left(t^i\right) - h\,u\left(x_k\right) + h\left(x_j\left(t^{i+1}\right) - x_k\right)\frac{\Delta u_k}{\Delta x_k}}{1 - h\frac{\Delta u_k}{\Delta x_k}}\right]/h. \end{aligned} \tag{4.4.20}$$

Using

$$x_j\left(t^{i+1}\right) = x_j^{i+1} + h\,\delta(\cdot), \tag{4.4.21}$$

we have

$$x_j\left(t^{i+1}\right) - x_k = x_j^{i+1} - x_k + h\,\delta(\cdot) \leq \Delta x_k + h\,\delta(\cdot). \tag{4.4.22}$$

Assuming that h is small enough, i.e. that $x_j\left(t^{i+1}\right)$ belongs to the same subinterval as x_j^{i+1}, we can can use the approximation

$$x_j\left(t^{i+1}\right) - x_k \leq \Delta x_k. \tag{4.4.23}$$

On the other hand, if we expand $u\left(x_k\right)$ around $u\left(x_j\left(t^{i+1}\right)\right)$, we find

$$\begin{aligned} u\left(x_k\right) = {} & u\left(x_j\left(t^{i+1}\right)\right) + \left(x_j\left(t^{i+1}\right) - x_k\right) u'\left(x_j\left(t^{i+1}\right)\right) \\ & + \frac{\left(x_j\left(t^{i+1}\right) - x_k\right)^2}{2} u''\left(x_j\left(t^{i+1}\right)\right) + O\left(\left(x_j\left(t^{i+1}\right) - x_k\right)^3\right). \end{aligned} \tag{4.4.24}$$

Assuming that (4.4.23) holds and that Δx_k is sufficiently small, we have

$$u\left(x_k\right) \doteq u\left(x_j\left(t^{i+1}\right)\right) + \Delta x_k\, u'\left(x_j\left(t^{i+1}\right)\right) + \frac{\Delta x_k^2}{2} u''\left(x_j\left(t^{i+1}\right)\right) + O\left(\Delta x_k^3\right). \tag{4.4.25}$$

Substituting (4.4.25) into (4.4.20) and performing some reordering, the local error becomes

$$\delta\left(x_j\left(t^{i+1}\right), h\right) = -\frac{h}{2}\frac{\ddot{x}_j\left(t^{i+1}\right)}{1 - h\frac{\Delta u_k}{\Delta x_k}} - \frac{\Delta x_k^2}{2}\frac{u''\left(x_j\left(t^{i+1}\right)\right)}{1 - h\frac{\Delta u_k}{\Delta x_k}} + O\left(h^2\right) + O\left(\Delta x_k^3\right). \tag{4.4.26}$$

Again the second term on the right-hand side is due to the interpolation and closely related to the error estimate which one would obtain by applying the inverse piece-wise linear interpolation. Also note that for sufficiently small h and Δx_k the local error is now $O(h) + O\left(\Delta x_k^2\right)$. This means that there is a constraint for Δx_k so that the interpolation error is commensurate with the local discretisation error of the EB

method. Of course, for $\Delta x_k = O\left(h^{1/2}\right)$, both errors are of the same order and the consistency order of the EB method is preserved.

4.5 Stability

In Example 3.1 it was shown that the flow method, applied to a (linear) test problem, produces the same results as the EB method if the interpolation error is equal to zero. This means that the stability domain, defined as the subset of the complex plane where the stability function (see Section 2.4)

$$\psi(h\lambda) := \frac{1}{1 - h\lambda}, \tag{4.5.1}$$

is in modulus bounded by 1, is identical to that of the EB method. Of course, in the general (nonlinear) case we cannot say that the stability properties of the flow method are the same as those of the EB method. Hence, we will investigate its stability now.

We can view our method as finding the solution via

$$x_k^{i+1} = p_{i,k}\left(x_k^i\right), \quad k \text{ fixed}, \quad i = 0, 1, \ldots \tag{4.5.2}$$

Here we used the indices i and k to denote its dependence on the time level t^i and the interpolation point x_k^i. If the interpolation linear polynomial is obtained as in (4.4.7), we have

$$p_{i,k}(x) = x + h\,\frac{u(x)}{1 - h\,\frac{u\left(x_{k+1}^i\right) - u(x)}{x_{k+1}^i - x}}. \tag{4.5.3}$$

In order to find out about the stability of the recursion (4.5.2), we look for the first variation. Let $\left\{z_k^i\right\}_{k\,\text{fixed}}$ denote a small perturbation of the solution $\left\{x_k^i\right\}_{k\,\text{fixed}}$ of (4.5.2), such that $\left\{x_k^i + z_k^i\right\}_{k\,\text{fixed}}$ also satisfies (4.5.2) to first order. Then we have

$$x_k^{i+1} + z_k^{i+1} = p_{i,k}\left(x_k^i + z_k^i\right) \doteq p_{i,k}\left(x_k^i\right) + p'_{i,k}\left(x_k^i\right)\, z_k^i. \tag{4.5.4}$$

By neglecting second and higher order terms we have

$$z_k^{i+1} = p'_{i,k}\left(x_k^i\right)\, z_k^i. \tag{4.5.5}$$

For stability in a nonlinear situation it is sufficient to prove contractivity of the discrete equation (4.5.5), i.e.

$$\left|p'_{i,k}\left(x_k^i\right)\right| < 1. \tag{4.5.6}$$

Carrying out the differentiation of $p_{i,k}(x)$ explicitly gives

$$p'_{i,k}(x) = 1 + h\,\frac{u'(x)\left(1 - h\frac{u\left(x_{k+1}^i\right) - u(x)}{x_{k+1}^i - x}\right) + hu(x)\frac{u\left(x_{k+1}^i\right) - u(x) - u'(x)\left(x_{k+1}^i - x\right)}{\left(x_{k+1}^i - x\right)^2}}{\left(1 - h\,\frac{u\left(x_{k+1}^i\right) - u(x)}{x_{k+1}^i - x}\right)^2}. \tag{4.5.7}$$

After substituting x_k^i and some reordering, (4.5.7) becomes

$$p'_{i,k}\left(x_k^i\right) = \frac{1 - h\left(\frac{\Delta u_k^i}{\Delta x_k^i} - u'\left(x_k^i\right)\right)}{1 - h\,\frac{\Delta u_k^i}{\Delta x_k^i}} + h^2\, f(x_k^i)\,\frac{\frac{\frac{\Delta u_k^i}{\Delta x_k^i} - u'\left(x_k^i\right)}{\Delta x_k^i}}{\left(1 - h\,\frac{\Delta u_k^i}{\Delta x_k^i}\right)^2}. \tag{4.5.8}$$

For further analysis of (4.5.8), we first make an expansion

$$\frac{\Delta u_k^i}{\Delta x_k^i} = u'\left(x_k^i\right) + \frac{\Delta x_k^i}{2} u''\left(x_k^i\right) + \frac{\left(\Delta x_k^i\right)^2}{6} u'''\left(x_k^i\right) + O\left(\left(\Delta x_k^i\right)^3\right). \tag{4.5.9}$$

Substituting (4.5.9) in (4.5.7), we find

$$\begin{aligned} \left|p'_{i,k}\left(x_k^i\right)\right| &= \left| \frac{1 - h\frac{\Delta x_k^i}{2} u''\left(x_k^i\right) + O\left(\left(\Delta x_k^i\right)^2\right)}{1 - h u'\left(x_k^i\right) + O\left(\Delta x_k^i\right)} + \right. \\ &+ \left. h^2 u\left(x_k^i\right) \frac{\frac{1}{2}u''\left(x_k^i\right) + \frac{\Delta x_k^i}{6} u'''\left(x_k^i\right) + O\left(\left(\Delta x_k^i\right)^2\right)}{\left(1 - h u'\left(x_k^i\right) + O\left(\Delta x_k^i\right)\right)^2} \right|. \end{aligned} \tag{4.5.10}$$

The expression (4.5.10) can be assessed as follows. It is clear that interpolation influences the stability. If I^i is "well-sampled", i.e. if Δx_k^i is sufficiently small so as to make the last terms negligible, the condition (4.5.6) can be simplified to give

$$\left| \frac{1}{1 - h u'\left(x_k^i\right)} + \frac{h^2}{2} \frac{u\left(x_k^i\right) u''\left(x_k^i\right)}{\left(1 - h u'\left(x_k^i\right)\right)^2} \right| < 1. \tag{4.5.11}$$

Note that a similar condition for the EB method would read

$$\left| \frac{1}{1 - h u'\left(x_k^{i+1}\right)} \right| < 1. \tag{4.5.12}$$

The additional term in (4.5.11) does not necessarily tend to zero for stiff problems, where $\left|h u'\left(x_k^i\right)\right| \gg 1$. Indeed, in this latter case (4.5.11) becomes approximately

$$\left| \frac{1}{2} \frac{u\left(x_k^i\right) u''\left(x_k^i\right)}{\left(u'\left(x_k^i\right)\right)^2} \right| < 1. \tag{4.5.13}$$

However, the latter constraint is not very severe, certainly compared to those found for explicit methods. We illustrate this in Section 4.6.

If the interpolation linear polynomial is obtain by the piece-wise linear interpolation, i.e. by (4.4.18), we have

$$p_{i,k}(x) = \frac{x + h\left[u(x_k) - x_k \frac{\Delta u_k}{\Delta x_k}\right]}{1 - h\frac{\Delta u_k}{\Delta x_k}}, \tag{4.5.14}$$

which, together with (4.5.6), simply gives the contractivity condition

$$\left| \frac{1}{1 - h\frac{\Delta u_k}{\Delta x_k}} \right| < 1. \tag{4.5.15}$$

Comparing (4.5.12) and (4.5.15), we clearly have that the stability condition of the flow method based on piece-wise linear interpolation is very close to the one of the EB method and better than the one obtained by the previous interpolation approach. The reason for this is, of course, that interpolation points are now not related to the solution points.

4.6 Practical Aspects

In the previous sections we showed the basic idea and gave some properties and error estimates of the flow method. We restricted ourselves basically to the simplest implementation, namely linear interpolation with an equidistantly sampled initial interval I^0. Another approach is to interpolate points $u\left(\hat{x}_k^{i+1}\right)$ instead of $\hat{x}_k^{i+1}$, using the corresponding $\hat{x}_k^i$. As before, this results in an approximation obtained by the interpolation polynomial, say $\tilde{p}\left(x_k^i\right)$, which represents the approximation of $u\left(x_k^{i+1}\right)$, i.e. we have

$$\tilde{p}\left(x_k^i\right) \doteq u\left(x_k^{i+1}\right). \tag{4.6.1}$$

It can easily be shown that linear interpolation by (4.6.1) instead of (4.2.4), yields an identical result.

As mentioned, interpolation should preferably be done locally. In principal there are two approaches that one can use to ensure this. We introduce both by the following:

(1) To obtain x_k^{i+1} (k fixed) we can use the point $\hat{x}_k^i$ plus $(m-1)$ additional points ($m \leq n$) closest to $\hat{x}_k^i$, i.e. $\hat{x}_{k+1}^i$, $\hat{x}_{k-1}^i$, $\hat{x}_{k+2}^i$,... as defined by (4.1.8). This approach gives an algorithm that is faster than the following one, since we do not search in which subinterval $\left[\hat{x}_k^i, \hat{x}_{k+1}^i\right]$, the point x_k^i is. However, for stiff problems the point x_k^i may be relatively far from the subinterval $\left[\hat{x}_k^i, \hat{x}_{k+1}^i\right]$, which can introduce large interpolation error due to the extrapolation.

(2) A more accurate (but also computationally more expensive) approach is to use all the points for interpolation. For example, one can apply piece-wise linear interpolation for constructing p that satisfies (4.2.3). To distinguish this approach from the previous one, we will attach it to problems where $S \neq I^i$, i.e. we omit the superscript i for the interpolation points. The main difference with the first approach is that we now first look for a subinterval, in which a particular flow point, say x_j^i is located, i.e. for which $x_j^i \in [\hat{x}_k, \hat{x}_{k+1}]$. Now by using points $\hat{x}_k$ and $\hat{x}_{k+1}$, one can construct a linear function and obtain x_j^{i+1} which satisfies

$$x_k \leq x_j^{i+1} \leq x_{k+1}. \tag{4.6.2}$$

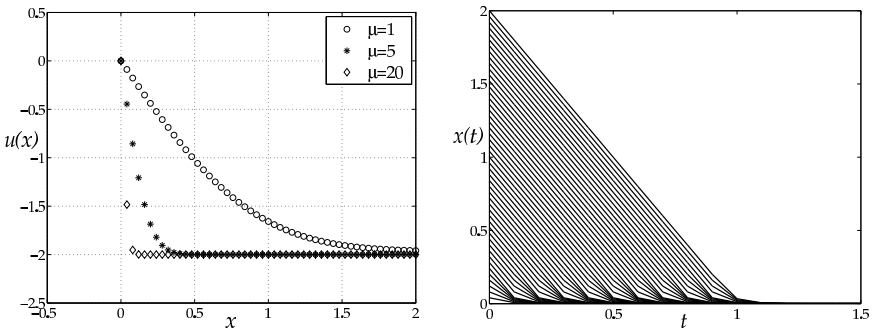
As pointed out before, one of the most important aspects of the flow method is that it can be used for stiff problems where the velocity field is not known explicitly, i.e. it is known at some discrete values $\{(x_k, u_k)\}_{k=0}^n$ only. In the following example we will show how we can solve such a problem rather easily and yet being efficient.

EXAMPLE 4.6.1. Consider the flow problem (4.1.3) where the velocity field $u(x)$ is to be found from the two-point boundary value problem (BVP)

$$\begin{cases} u''(x) + 2\mu x u'(x) = 0, & \mu > 0, \\ m(0) \leq x \leq M(0), \\ u\left(m(0)\right) = 0, \\ u\left(M(0)\right) + u'\left(M(0)\right) = c, & c \in \mathbb{R}. \end{cases} \tag{4.6.3}$$

To solve the BVP (4.6.3) numerically, we need to discretise $[m(0), M(0)]$. Of course, the number of grid points, say n, depends on the required accuracy. The numerical solution is a discrete set of points $\{(x_k, u_k)\}_{k=0}^n$. If we would choose the (obviously hassle free) Euler Forward method for the time integration we only would have to

employ interpolation to obtain u at intermediate points. However, if the problem is stiff, this approach requires a rather small time-step. In particular one should note that when μ is large, $u(x)$ is changing more rapidly around $x = 0$, see Figure 4.6.1a. Then the Euler Forward method clearly requires a small step-size, if only to ensure the solution to remain positive. Let us take $m(0) = 0$, $M(0) = 2$, $n = 51$, $c = -2$ and $\mu = 100$. Euler Forward then requires $h \leq 0.0225$. In Figure 4.6.1b, the results obtained by the flow method with $h = 0.1$ are shown. Clearly, the behaviour of the solution indicates that the method does not have problems with the stiffness, in a similar way as one would have for EB method. □



(a) Discrete velocity field (b) Flow in time

FIGURE 4.6.1. The solution of the two-point BVP ($n = 51$, $\mu = 100$, $h = 0.1$).

The error analysis in Section 4.4 showed that the interpolation error depends not only on the time step size h, but also on the spatial step size Δx_k^i. The latter should control the interpolation error appropriately.

EXAMPLE 4.6.2. Consider the ODE

$$\begin{cases} \dot{x} = -\arctan(10\,x), \\ x(0) \in [-1, 1]. \end{cases} \tag{4.6.4}$$

As can be seen the second derivative of the velocity field is (absolutely) large for x close to zero. This means that we can expect larger interpolation errors there. Let us take $h = 0.01$. In the first experiment we take $\Delta x_k^0 = 1.0\,(n = 3)$. The global errors of the upper boundary solution, obtained by the flow method and EB are shown in Table 4.6.1, columns 2 and 4 respectively. Note that the accuracy of the flow method is slightly less than the one obtained by EB, showing that the interpolation error is dominant here. This also follows from an estimate of the interpolation error, see column 3, derived from (4.4.17). If we take $\Delta x_k^0 = 0.01\,(n = 201)$, the interpolation error becomes of order smaller then the local discretisation error (see column 6), making the accuracy of the flow method almost the same as one obtained by EB, see columns 4 and 5. □

In a practical setting, if possible, one would like to make the computation more effective, by adapting the number of points in the flow at every time-level. Let us denote this number of points at t^i by n^i. Given an interpolation tolerance, say INTTOL, we require the interpolation error of the boundary points to be approximately equal to

Time	Flow method		Euler Backward	Flow method	
	Global error	Interpolation error		Global error	Interpolation error
	$\Delta x_k^0 = 1.0$ $(n = 3)$			$\Delta x_k^0 = 0.01$ $(n = 201)$	
t^i	$\lvert e_n^i \rvert$	$\lvert r_n^i \rvert$	$\lvert e_n^i \rvert$	$\lvert e_n^i \rvert$	$\lvert r_n^i \rvert$
0.1	2.1651e-03	1.9325e-03	8.4558e-05	8.3723e-05	2.2478e-05
0.2	4.5909e-03	2.7015e-03	1.9997e-04	1.9781e-04	3.7453e-05
0.3	7.3125e-03	3.9829e-03	3.6517e-04	3.6073e-04	6.8000e-05
0.4	1.0324e-02	6.2680e-03	6.1608e-04	6.0736e-04	1.3768e-04
0.5	1.3440e-02	1.0551e-02	1.0230e-03	1.0056e-03	3.1492e-04
0.6	1.5813e-02	1.7761e-02	1.6985e-03	1.6608e-03	7.4473e-04
0.7	1.4959e-02	2.0338e-02	2.5251e-03	2.4609e-03	1.0332e-03
0.8	9.6363e-03	6.5075e-03	2.5258e-03	2.4642e-03	2.9339e-04
0.9	4.6815e-03	6.0262e-04	1.6611e-03	1.6330e-03	2.4763e-05
1.0	2.0696e-03	3.7538e-05	8.8604e-04	8.7792e-04	1.5129e-06

TABLE 4.6.1. The global errors for $h = 0.01$.

INTTOL. Let Δx^i denote the spatial step such that $\max(\left|r_1^i\right|, \left|r_n^i\right|) \approx$ INTTOL. Then n^i is given by

$$n^i := \left\lfloor \frac{|x_n^i - x_0^i|}{\Delta x^i} \right\rfloor + 1. \tag{4.6.5}$$

EXAMPLE 4.6.3. Consider the problem in Example 4.6.2 again and apply regridding as described above. From Table 4.6.1 one should note that the absolute discretisation error (the error of the EB method) for $h = 0.01$ is about 10^{-3}. Hence we choose INTTOL $= 10^{-3}$ which should ensure that the interpolation error be commensurate with the discretisation error. The results are shown in Table 4.6.2. Note that the number of points n^i is varying in time, ensuring the interpolation error to be bellow INTTOL. □

The flow method can be also used for non-autonomous problems of the following type

$$\dot{x} = u(x) + w(t). \tag{4.6.6}$$

The principle remains the same, i.e. we first determine a set of points

$$\hat{x}_k^i = \hat{x}_k^{i+1} - h\,u\left(\hat{x}_k^{i+1}\right) - h\,w\left(t^{i+1}\right) = x_k^i - h\,u\left(x_k^i\right) - h\,w\left(t^{i+1}\right), \tag{4.6.7}$$

and then interpolate points $\hat{x}_k^{i+1}$ with corresponding $\hat{x}_k^i$, in order to obtain the interpolation polynomial $p\left(x^i\right)$. Since w does not depend on the spatial variable x, the additional source terms are cancelling in all denominators of $p\left(x^i\right)$ (for arbitrary m), making w appearing only as an additional term of the interpolation polynomial. To illustrate this, we can reformulate Example 4.2.1 by adding a source term $w(t)$ on the right-hand side of (4.2.9). The analogue of (4.2.12) then reads

$$x_k^{i+1} = \frac{\left(x_k^i - x_{k+1}^i + h\,\left(\lambda\,x_{k+1}^i + w\left(t^{i+1}\right)\right)\right)\,x_k^i - h\,\left(\lambda\,x_k^i + w\left(t^{i+1}\right)\right)\,x_{k+1}^i}{(1 - h\lambda)\,\left(x_k^i - x_{k+1}^i\right)}. \tag{4.6.8}$$

	Flow method			
Time	Flow points	Interpolation error	Global error	Euler Backward
t^i	n^i	$\lvert r_n^i \rvert$	$\lvert e_n^i \rvert$	$\lvert e_n^i \rvert$
0.1	5	9.6143e-04	2.6942e-04	8.4558e-05
0.2	7	9.8869e-04	5.2216e-04	1.9997e-04
0.3	10	7.6281e-04	7.9990e-04	3.6517e-04
0.4	14	7.8502e-04	1.1326e-03	6.1608e-04
0.5	24	8.0818e-04	1.5783e-03	1.0230e-03
0.6	39	8.3235e-04	2.2013e-03	1.6985e-03
0.7	39	8.5756e-04	2.8430e-03	2.5251e-03
0.8	11	8.8390e-04	2.6857e-03	2.5258e-03
0.9	2	9.1142e-04	1.7548e-03	1.6611e-03
1.0	2	9.4019e-04	9.2635e-04	8.8604e-04

TABLE 4.6.2. The global errors for $h = 0.01$ and INTTOL $= 10^{-3}$.

This leads to

$$x_k^{i+1} = \frac{x_k^i}{1 - h\lambda} + \frac{h\, w\left(t^{i+1}\right)}{1 - h\lambda}, \tag{4.6.9}$$

as would have been obtained from applying EB directly. In the following example we will consider one non-autonomous problem, where moreover, u is not known explicitly.

EXAMPLE 4.6.4. Consider an electrical circuit with nonlinear elements, such as nonlinear resistors (varistors), diodes, transistors, etc. (see e.g. [22]). Here the transfer function (voltage-current dependence) is nonlinear and can only be obtained experimentally, i.e. it is defined by a discrete set of points $\{(i_k, v_k)\}_{k=0}^n$. A particular such a system is a DC motor with a nonlinear resistor as a protection element, see Figure 4.6.2a. The nonlinear resistor ensures the current not to increase dramatically, even for very large voltages on the motor stator side. The most dangerous situation is when the rotor is blocked (i.e. the motor is not spinning); when this happens the current is maximal. The equivalent electrical scheme of the motor is then defined as a serial connection of a resistor (R) and an inductance (L), see Figure 4.6.2b.

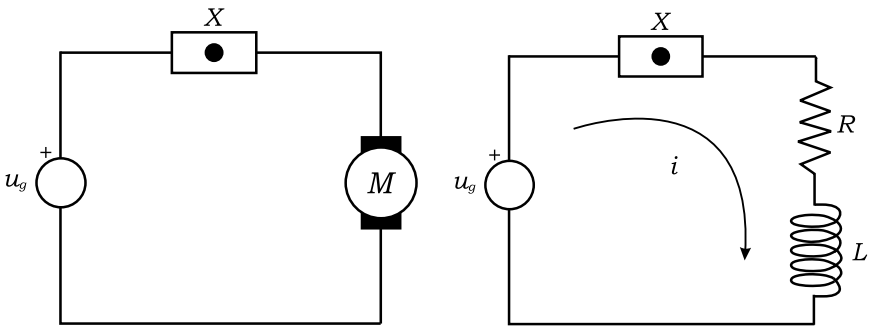
The motor is driven by a generator v_g. For a time varying generator voltage, i.e. $v_g = v_g(t)$, we have a non-autonomous problem defined by (4.6.6). Here we choose $v_g(t) = V \cos \omega t$.

The mathematical model of this system follows from Kirchhoff's second law, i.e.

$$L\frac{di}{dt} = -v(i) - Ri + v_g(t), \tag{4.6.10}$$

where $i = i(t)$ is the current in the loop and $v(i)$ is the voltage on the nonlinear resistor defined only at some discrete points $\{i_k\}_{k=0}^n$. This can be rewritten as

$$\begin{aligned} x(t) &= i(t), \\ u(x) &= -\frac{1}{L}\, v(x) - \frac{R}{L}\, x, \\ w(t) &= \frac{V}{L} \cos \omega t. \end{aligned} \tag{4.6.11}$$



(a) Stator side (b) Equivalent electrical scheme

FIGURE 4.6.2. DC motor circuit

Since we would like to observe the behaviour of the system for a range of initial conditions (the inductance current value at $t = 0$, we take $x(0) = i(0) \in [I_{min}, I_{max}]$. We discretise the interval using as many points as required to monitor the evolution of the flow. However, for this application the number of points of the flow does not influence the accuracy of the method (in particular the interpolation part). This is related only to a number of experimentally obtained points $\{(i_k, \nu_k)\}_{k=0}^{n}$ to be used for interpolation. We take 201 points, spanning the interval $[-10, 10]$, see Table 4.6.3 and Figure 4.6.4a. Of course, this interval should be large enough to cover entire flow in time to avoid large errors coming from extrapolation. Typical parameter values are $R = 2\Omega$, $L = 0.01H$, $V = 220V$ and $\omega = 1rad/s$.

i	-10.0	-9.9	...	-0.1	0.0	0.1	...	10.0
$\nu(i)$	1.0e+9	9.3e+8	...	20.0	0.0	-20.0	...	-1.0e+9

TABLE 4.6.3. The discrete voltage-current characteristics.

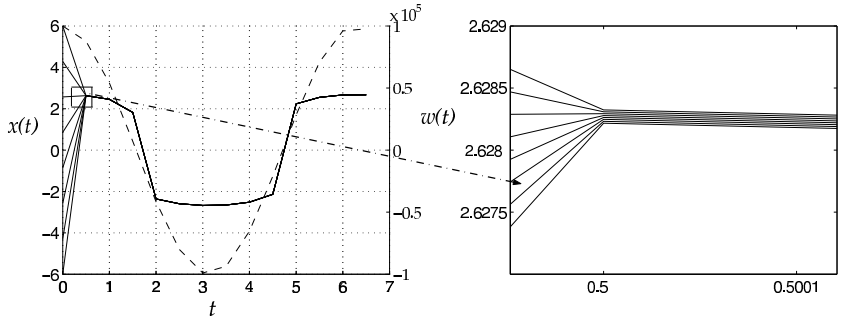
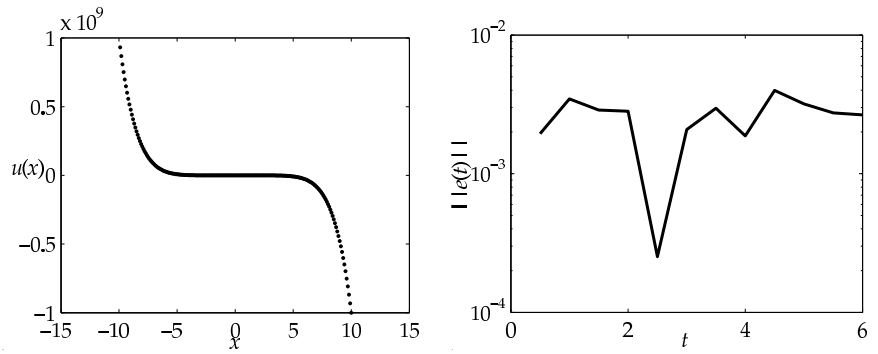


FIGURE 4.6.3. The flow (current) in time compared with the source term $w(t)$.

The problem is stiff since the transfer function of the nonlinear resistor is very steep, making the flow highly convergent, see Figure 4.6.3. However, the flow method shows proper accuracy (see Figure 4.6.4b) and stability properties, despite the fact that problem is non-autonomous. □



(a) Discrete voltage-current characteristics ($n = 201$)

(b) The global error ($h = 0.5$)

FIGURE 4.6.4. Numerical results

CHAPTER 5

The flow method based on linear multistep methods

In this chapter we extend our analysis of the flow method by employing LMMs. Following the same problem settings as in the previous chapter, we first discuss the implementation of the method based on LMMs. In Section 5.2 we give a condition for the flow method to be well-posed. Since we also use interpolation in this implementation, the error involved is given in Section 5.3, where we obtain the local error estimate. We conclude this chapter with stability properties of the method, see Section 5.4.

5.1 Method implementation

Consider again the autonomous (scalar) flow problem

$$\begin{cases} \dot{x} = u(x), \\ x(0) \in I(x, 0), \end{cases} \tag{5.1.1}$$

which was already introduced in the previous chapter. Again we also assume that we have a problem of a finite flow, denoted by $I(x, t)$ and that the velocity field $u(x)$ is Lipschitz continuous and unknown explicitly. We let $S = \{x_k\}_{k=0}^{n}$ to be a set of points for which u values are given by the set $\{u_k\}_{k=0}^{n}$. We now investigate how to solve (5.1.1) by the flow method which employs some LMM. Again let x^i represent the approximation of $x\left(t^i\right)$ at $t^i = i\,h$, with h fixed, then the solution point satisfies

$$\sum_{l=0}^{m} \alpha_l \, x^{i-l+1} = h \sum_{l=0}^{m} \beta_l u\left(x^{i-l+1}\right). \tag{5.1.2}$$

Since we are interested in implicit methods only, we assume that $\beta_0 \neq 0$. A particularly interesting class of methods are BDF methods of various order ($m \leq 6$), which were introduced in 2.5.1. They have good stability properties, which makes them especially suitable for stiff problems. Moreover, as shown later, they have a significant advantage over other LMMs, when employed in the flow method. Their general form is given by

$$\sum_{l=0}^{m} \alpha_l \, x^{i-l+1} = h\, u\left(x^{i+1}\right). \tag{5.1.3}$$

The general LMM (5.1.2) can be reformulated as

$$\alpha_0 x^{i+1} - h\beta_0 u\left(x^{i+1}\right) = \sum_{l=1}^{m} \left[-\alpha_l x^{i-l+1} + h\beta_l u\left(x^{i-l+1}\right)\right]. \tag{5.1.4}$$

By defining

$$s^i := \sum_{l=1}^{m} \left[-\alpha_l x^{i-l+1} + h\beta_l u\left(x^{i-l+1}\right)\right], \tag{5.1.5}$$

we can write (5.1.4) as

$$\alpha_0 x^{i+1} - h\beta_0 u\left(x^{i+1}\right) = s^i, \tag{5.1.6}$$

where s^i is assumed to be known. The expression (5.1.6) represents a nonlinear equation (a system in the general vectorial case). For the existence and uniqueness condition of the LMM solution see Theorem 2.5.1. In the reminder of this chapter we simply assume

$$u'(x) \leq \nu, \quad x \in I(x,t), \tag{5.1.7}$$

where $u'(x)$ represents the first derivative of u w.r.t. x and ν is the scalar analogue of the one-sided Lipschitz constant in Theorem 2.5.1.

To avoid confusion with the notation we define

$$f\left(x^{i+1}\right) := \alpha_0 x^{i+1} - h\beta_0 u\left(x^{i+1}\right). \tag{5.1.8}$$

Assuming that (5.1.7) holds, the solution of (5.1.6) can formally be obtained by finding the inverse function of f, say g $\left(g = f^{-1}\right)$. The exact solution of (5.1.6), which we denote by $\tilde{x}^{i+1}$, is then

$$\tilde{x}^{i+1} = g\left(s^i\right). \tag{5.1.9}$$

Since obtaining g explicitly is impossible in general, one often computes the approximative solution by employing some iterative method. In particular, for stiff problems, the iteration needs to be done by the Newton method (see Section 2.7), which is not straightforward due to the discrete nature of u. Another way is to apply the flow method, i.e. the inverse interpolation technique similarly as already shown for the EB method. Again let $I^i = \left\{x_j^i\right\}_{j=0}^{\mu}$ be a sufficiently dense set, obtained by spatially discretising the flow $I\left(x, t^i\right)$ at t^i. We let the set $S = \{x_k\}_{k=0}^{n}$ to be time-invariant.

Construction of the flow method based on LMMs is similar to the one introduced for the Euler Backward method. Hence, let us define

$$\hat{x}_k^{i+1} := x_k^i = x_k. \tag{5.1.10}$$

Since u in (5.1.1) is autonomous, we clearly have

$$u\left(\hat{x}_k^{i+1}\right) = u\left(x_k\right) = u_k. \tag{5.1.11}$$

Substituting (5.1.10) into (5.1.8) and using (5.1.11), we have

$$f_k := f\left(\hat{x}_k^{i+1}\right) = \alpha_0 \hat{x}_k^{i+1} - h\beta_0 u\left(\hat{x}_k^{i+1}\right) = \alpha_0 x_k - h\beta_0 u_k. \tag{5.1.12}$$

Since the u_k values are known, the set $\hat{S} = \{f_k\}_{k=0}^{n}$ can be obtained. This can be used for defining the inverse interpolation problem, which can be solved by constructing the interpolation function, say p. The interpolation function is then the approximation of the inverse function $g = f^{-1}$. Hence, by defining

$$\hat{\Omega} := \left\{(f_k, x_k) \mid f_k \in \hat{S},\, g(f_k) = x_k,\, k = 0, \ldots, n\right\},$$

$$p(f_k) = g(f_k) = x_k, \tag{5.1.13}$$

and constructing p by some interpolation method, one may obtain the approximation of the LMM solution $\tilde{x}^{i+1}$. For an arbitrary flow point, the solution at the next time level follows from

$$x_j^{i+1} = p\left(s_j^i\right), \tag{5.1.14}$$

where s_j^i is given by (5.1.5) for $x^i = x_j^i$. For general LMMs computing s_j^i can represent a problem if u is not given at flow points at previous time levels. This problem can be overcome by additional (in this case direct) interpolation of points $\{(x_k, u_k)\}_{k=0}^n$. Thus obtained approximations of $u\left(x_j^{i-l+1}\right)$, $l = 1, \ldots, m$, can be used to determine s_j^i. Note that the same argument holds for standard LMMs as well. However, in case of BDF methods, which are of special interest here, there is no need for additional interpolation. Indeed, now we have

$$s_j^i = -\sum_{l=1}^{m} \alpha_l \, x_j^{i-l+1}, \tag{5.1.15}$$

and since x_j^{i-l+1} values are known, the application of the method is straightforward.

An illustration of the method principle is shown in Figure 5.1.1, where the interpolation function is based on piece-wise linear interpolation. Here the solution at the next time level reads

$$x_j^{i+1} = p\left(s_j^i\right) = \frac{s_j^i - f_k}{f_{k+1} - f_k} \, x_{k+1} + \frac{s_j^i - f_{k+1}}{f_k - f_{k+1}} \, x_k. \tag{5.1.16}$$

We choose points f_k so that $f_k \le s_j^i \le f_{k+1}$. Assuming that integral curves do not intersect between time levels t^i and t^{i+1}, this yields

$$x_k \le x_j^{i+1} \le x_{k+1}. \tag{5.1.17}$$

After substituting (5.1.12) and some reordering, the expression (5.1.16) becomes

$$x_j^{i+1} = x_k - \frac{\alpha_0 \, x_k - s_j^i - h \, \beta_0 \, u_k}{\alpha_0 - h \, \beta_0 \, \frac{u_{k+1} - u_k}{x_{k+1} - x_k}}, \tag{5.1.18}$$

which is exactly one step of the discrete Newton method with initial values x_k and x_{k+1}. According to (5.1.17), these values are already close to the solution, meaning that a good accuracy result could be expected. This will be discussed later.

EXAMPLE 5.1.1. To illustrate the efficiency of the flow method, we apply it to the linear test problem, as we did in the previous chapter. Consider

$$\begin{cases} \dot{x} = \lambda \, x, \\ x(0) \in I(x, 0), \end{cases} \tag{5.1.19}$$

Since the problem is linear, linear piece-wise interpolation, as defined by (5.1.16), will yield the full accuracy of the latter. After substituting values for u_k into (5.1.18) and (5.1.5) we have

$$x_j^{i+1} = \frac{\sum_{l=1}^{m} \left(-\alpha_l + h \, \lambda \, \beta_l\right) \, x_j^{i-l+1}}{\alpha_0 - h \, \lambda \, \beta_0}, \tag{5.1.20}$$

which is exactly the same result as one would have obtained by applying standard LMM. Again the result is as expected since linear interpolation is exact for linear functions. □

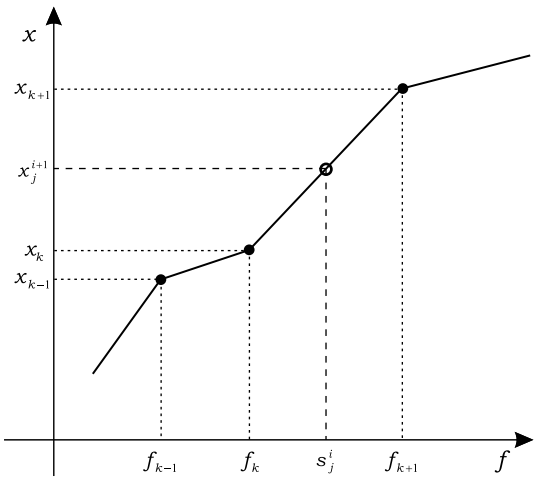


FIGURE 5.1.1. Inverse interpolation principle

As for the one-step case, the flow method can also be efficiently applied for a class of non-autonomous problems of the following type

$$\dot{x} = u(x) + w(t). \tag{5.1.21}$$

The method principle remains the same and the solution at the next time point follows from

$$x_j^{i+1} = p\left(s_j^i + h\, w\left(t^{i+1}\right)\right). \tag{5.1.22}$$

5.2 Well-posedness

As already mentioned, exact solution curves will not intersect if the velocity field u is Lipschitz continuous. This property we call well-posedness (cf. 4.3.6). The situation for LMMs is more complex than for the EB method, since the solution depends also on values at previous time levels. Hence, starting from any two flow points, say x_j and x_{j+1}, and assuming that the following holds

$$x_{j+1}^{i-l+1} - x_j^{i-l+1} > 0, \quad l = 1, \ldots, m, \tag{5.2.1}$$

i.e. that intersections did not occur before time level t^i, we seek the condition under which we have

$$x_{j+1}^{i+1} - x_j^{i+1} > 0. \tag{5.2.2}$$

Applying (5.1.6) for these two flow points, we have

$$x_{j+1}^{i+1} - x_j^{i+1} = \left(\alpha_0 - h\,\beta_0\, \frac{u\left(x_{j+1}^{i+1}\right) - u\left(x_j^{i+1}\right)}{x_{j+1}^{i+1} - x_j^{i+1}}\right)^{-1} \left(s_{j+1}^i - s_j^i\right), \tag{5.2.3}$$

where s_j^i and s_{j+1}^i are defined by (5.1.5). By using the mean value theorem, we find

$$\frac{u\left(x_{j+1}^{i+1}\right) - u\left(x_j^{i+1}\right)}{x_{j+1}^{i+1} - x_j^{i+1}} = u'(\eta), \quad \eta \in \left[x_{j+1}^{i+1}, x_j^{i+1}\right], \tag{5.2.4}$$

and (5.2.3) becomes

$$x_{j+1}^{i+1} - x_j^{i+1} = (\alpha_0 - h\,\beta_0\,u'(\eta))^{-1}\,\left(s_{j+1}^i - s_j^i\right). \tag{5.2.5}$$

Under the Lipschitz condition and assuming that the existence and uniqueness condition (5.1.7) is satisfied, it is clear that

$$\alpha_0 - h\,\beta_0\,u'(\eta) > 0. \tag{5.2.6}$$

This means that the condition of the LMM to be well-posed is

$$s_{j+1}^i - s_j^i > 0. \tag{5.2.7}$$

Substituting (5.1.5) into (5.2.7), this means

$$\sum_{l=1}^{m}\left[-\alpha_l\left(x_{j+1}^{i-l+1} - x_j^{i-l+1}\right) + h\,\beta_l\,\left(u\left(x_{j+1}^{i-l+1}\right) - u\left(x_j^{i-l+1}\right)\right)\right] > 0. \tag{5.2.8}$$

The latter expression can be simplified by using the mean value theorem and by defining the differences

$$\Delta x_j^{i-l+1} := x_{j+1}^{i-l+1} - x_j^{i-l+1}, \quad l = 1, \ldots, m, \tag{5.2.9}$$

which are all positive under the assumption (5.2.1). Hence (5.2.8) becomes

$$\sum_{l=1}^{m}\left(-\alpha_l + h\,\beta_l\,u'\left(\xi_l\right)\right)\Delta x_j^{i-l+1} > 0, \tag{5.2.10}$$

where $\xi_l \in \left[x_j^{i-l+1}, x_{j+1}^{i-l+1}\right]$, $l = 1, \ldots, m$.

The well-posedness condition may imply constraints on the time step size h (as shown in the Example below). Since coefficients α_l's and β_l's are having alternating signs in standard LMMs, it is not easy to judge which methods are more suitable for solving flow problems. Moreover, in this (general) case the behaviour of $u'(x)$ over the flow domain is also playing an important role. The special case of interest, i.e. BDF methods do not have this additional problem. Here the well-posedness condition is simply

$$\sum_{l=1}^{m}\alpha_l \Delta x_j^{i-l+1} < 0. \tag{5.2.11}$$

Since for all BDF methods we have

$$\begin{cases} \alpha_l < 0, & \text{for } l \text{ odd}, \\ \alpha_l > 0 & \text{for } l \text{ even}, \end{cases} \tag{5.2.12}$$

and since $\Delta x_j^{i-l+1} > 0$, $l = 1, \ldots, m$, it is clear that BDF methods of odd order (m odd) have an advantage over BDF methods of even order. This will be illustrated by the following example.

EXAMPLE 5.2.1. Consider a simple test problem

$$\begin{cases} \dot{x} = \lambda\,x, & \lambda < 0, \\ x(0) \in I^0 = [-0.5, 0.5], & I^0 = \left\{x_j^0\right\}_{j=0}^{\mu}. \end{cases} \tag{5.2.13}$$

Since $\lambda < 0$, the existence and uniqueness conditions are satisfied for $\forall x \in \mathbb{R}$ and for every BDF method. If the method of choice is the BDF2 method, (5.2.11) reads

$$-2\,\Delta x^i + \frac{1}{2}\,\Delta x_j^{i-1} < 0, \tag{5.2.14}$$

i.e.

$$\Delta x_j^i > \frac{1}{4}\,\Delta x_j^{i-1}. \tag{5.2.15}$$

For stiff problems the convergent flow may rapidly collapse over a single time step. This means that the actual factor relating Δx_j^i and Δx_j^{i-1} can be much smaller than $\frac{1}{4}$ and the application of BDF2 with larger time steps will introduce intersections of the integral curves. Let us now solve (5.2.13) by using BDF2 over a single time step only. This method requires initial conditions for first two time levels. In this particular case we know the exact solution, which reads

$$x_j(t) = x_j^0\, e^{\lambda t}, \quad j = 0, \ldots, \mu. \tag{5.2.16}$$

Thus, we will take $x_j^1 = x_j^0\, e^{\lambda h}$. Now, from (5.2.15) we have

$$e^{\lambda h} > \frac{1}{4}, \tag{5.2.17}$$

i.e.

$$h < \frac{\log 4}{|\lambda|}. \tag{5.2.18}$$

One should note that (5.2.18) is rather restrictive for $|\lambda|$ large, which is typically the case for stiff problems. Hence, if h does not satisfy this condition, the intersections will occur already after one time step. Surprisingly, the distance of the flow points does not influence the condition (5.2.18). In other words, no matter how far x_j^0 and x_{j+1}^0 are from each other, the intersections will occur if h is not sufficiently small. On the other hand, by applying BDF3 and exact initial conditions, the condition equivalent to (5.2.17) after one time step, reads

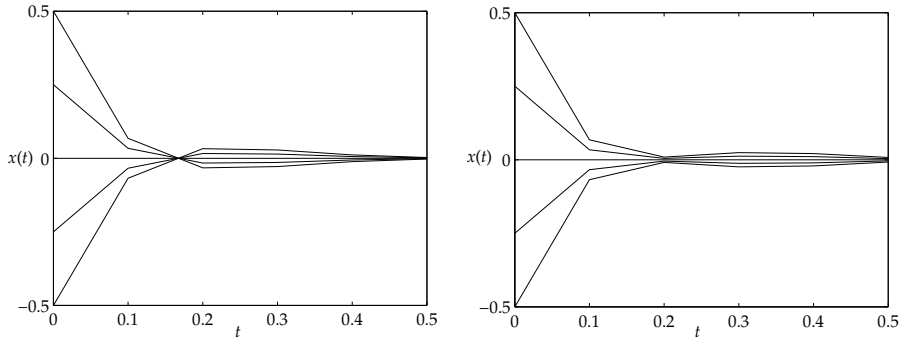
$$3\, e^{2\lambda h} - \frac{3}{2}\, e^{\lambda h} + \frac{1}{4} > 0. \tag{5.2.19}$$

It is not difficult to check that (5.2.19) holds for every $h\lambda$, implying that intersections after a single step will not occur no matter how large is h. This, of course, does not guarantee that intersection will be avoided after several time steps, but clearly shows why BDF methods of odd order have the advantage over ones of even order. In Table 5.2.1 the time step constraints for a single computational step is shown for the whole BDF family. Time step constraints for higher order BDF methods are obtained numerically.

BDF1	BDF2	BDF3	BDF4	BDF5	BDF6
-	$h < \frac{1.3863}{\lvert\lambda\rvert}$	-	$h < \frac{1.1421}{\lvert\lambda\rvert}$	-	$h < \frac{1.0314}{\lvert\lambda\rvert}$

TABLE 5.2.1. The well-posedness time step constraint of BDF methods over a single computational step

The numerical solutions obtained BDF2 and BDF3 for $\lambda = -20$ and $h = 0.1$ are shown in Figure 5.2.1. Since for the BDF2 method the well-posedness condition is not satisfied (h is not sufficiently small), intersections occur already after one step. On the other hand, this is not the case for the BDF3 method. □



(a) BDF2 (b) BDF3

FIGURE 5.2.1. Flow obtained by BDF methods

The flow method based on LMMs should follow the same pattern as described above. However, the influence of the interpolation involved may be significant, especially for stiff problems. To illustrate this, a special case of the flow method based on the piece-wise linear interpolation will be analysed. Starting from (5.1.18) and introducing the notation

$$\begin{aligned} \Delta x_k :=& x_{k+1} - x_k, \\ \Delta u_k :=& u_{k+1} - u_k, \end{aligned} \tag{5.2.20}$$

we find that the solution at the next time point satisfies

$$\left(\alpha_0 - h\,\beta_0\,\frac{\Delta u_k}{\Delta x_k}\right) x_j^{i+1} = s_j^i + h\,\beta_0\,\left(u_k - x_k\,\frac{\Delta u_k}{\Delta x_k}\right). \tag{5.2.21}$$

Clearly (5.1.17) also holds (from the interpolation principle). Now, let the following flow point be between different two points, i.e. let $x_{j+1}^{i+1} \in [x_q, x_{q+1}]$. Then x_{j+1}^{i+1} satisfies

$$\left(\alpha_0 - h\,\beta_0\,\frac{\Delta u_q}{\Delta x_q}\right) x_{j+1}^{i+1} = s_{j+1}^i + h\,\beta_0\,\left(u_q - x_q\,\frac{\Delta u_q}{\Delta x_q}\right). \tag{5.2.22}$$

Subtracting the left-hand sides of (5.2.21) and (5.2.22) yields

$$\begin{aligned} &\left(\alpha_0 - h\,\beta_0\,\frac{\Delta u_q}{\Delta x_q}\right) x_{j+1}^{i+1} - \left(\alpha_0 - h\,\beta_0\,\frac{\Delta u_k}{\Delta x_k}\right) x_j^{i+1} \\ &\quad = \alpha_0\left(x_{j+1}^{i+1} - x_j^{i+1}\right) - h\,\beta_0\,\left(x_{j+1}^{i+1}\,\frac{\Delta u_q}{\Delta x_q} - x_j^{i+1}\,\frac{\Delta u_k}{\Delta x_k}\right). \end{aligned} \tag{5.2.23}$$

We may simplify (5.2.23) by assuming that the set S is sufficiently dense, i.e. that Δx_k and Δx_q are relatively small, such that

$$\begin{aligned} \frac{\Delta u_k}{\Delta x_k} \doteq& u'\left(x_j^{i+1}\right), \\ \frac{\Delta u_q}{\Delta x_q} \doteq& u'\left(x_{j+1}^{i+1}\right). \end{aligned} \tag{5.2.24}$$

Although this approximation seems a little bit crude, it does not influence the analysis about the well-posedness problem. Especially since (5.1.17) holds. Substituting

(5.2.24) into (5.2.23), we have

$$\begin{aligned}&\alpha_0\left(x_{j+1}^{i+1}-x_j^{i+1}\right)-h\,\beta_0\,\left(x_{j+1}^{i+1}\,u'\left(x_{j+1}^{i+1}\right)-x_j^{i+1}\,u'\left(x_j^{i+1}\right)\right)\\&=\left(x_{j+1}^{i+1}-x_j^{i+1}\right)\left(\alpha_0-h\,\beta_0\frac{G\left(x_{j+1}^{i+1}\right)-G\left(x_j^{i+1}\right)}{x_{j+1}^{i+1}-x_j^{i+1}}\right),\end{aligned}\tag{5.2.25}$$

where the function $G(x)$ is defined by

$$G(x):=x\,u'(x),\tag{5.2.26}$$

with the first derivative

$$G'(x)=u'(x)+xu''(x).\tag{5.2.27}$$

From (5.2.25), (5.2.26) and (5.2.27) and by applying the mean value theorem, we have

$$\begin{aligned}&\left(x_{j+1}^{i+1}-x_j^{i+1}\right)\left(\alpha_0-h\,\beta_0\,G'(\eta)\right)\\&=\left(x_{j+1}^{i+1}-x_j^{i+1}\right)\left(\alpha_0-h\,\beta_0\,u'(\eta)-h\,\beta_0\,\eta\,u''(\eta)\right),\end{aligned}\tag{5.2.28}$$

where $\eta\in\left[x_j^{i+1},x_{j+1}^{i+1}\right]$. Now from (5.2.21), (5.2.22) and (5.2.28) it follows

$$\begin{aligned}x_{j+1}^{i+1}-x_j^{i+1}=&\left(\alpha_0-h\,\beta_0\,u'(\eta)-h\,\beta_0\,\eta\,u''(\eta)\right)^{-1}\left[s_{j+1}^{i}-s_j^{i}\right.\\&\left.+h\,\beta_0\,\left(u_q-u_k-\left(x_q\frac{\Delta u_q}{\Delta x_q}-x_k\frac{\Delta u_k}{\Delta x_k}\right)\right)\right].\end{aligned}\tag{5.2.29}$$

Defining the function $H(x)$ as

$$H(x):=u(x)-x\,u'(x),\tag{5.2.30}$$

which has the first derivative

$$H'(x)=-x\,u''(x),\tag{5.2.31}$$

we may write

$$H\left(x_q\right)-H\left(x_k\right)=u\left(x_q\right)-u\left(x_k\right)-\left(x_q\,u'\left(x_q\right)-x_k\,u\left(x_k\right)\right).\tag{5.2.32}$$

Again we assume that Δx_k and Δx_q are relatively small, such that we may use the approximation

$$\begin{aligned}\frac{\Delta u_k}{\Delta x_k}&\doteq u'\left(x_k\right),\\\frac{\Delta u_q}{\Delta x_q}&\doteq u'\left(x_q\right).\end{aligned}\tag{5.2.33}$$

Substituting (5.2.31), (5.2.32) and (5.2.33) into (5.2.29), the well-posedness condition of the flow method reads

$$\begin{aligned}x_{j+1}^{i+1}-x_j^{i+1}=&\left(\alpha_0-h\,\beta_0\,u'(\eta)-h\,\beta_0\,\eta\,u''(\eta)\right)^{-1}\left[s_{j+1}^{i}-s_j^{i}\right]\\&-h\,\beta_0\,\left(x_q-x_k\right)\left(\alpha_0-h\,\beta_0\,u'(\eta)-h\,\beta_0\,\eta\,u''(\eta)\right)^{-1}\xi\,u''(\xi),\end{aligned}\tag{5.2.34}$$

where

$$x_k\le x_j^{i+1}\le\xi,\eta\le x_{j+1}^{i+1}\le x_{q+1}.\tag{5.2.35}$$

Comparing (5.2.34) and (5.2.5), it is clear that the well-posedness condition of the flow method is close to one of the related LMM. Basically the flow method will most likely behave like the standard LMM and, thus, similar arguments hold. However, in (5.2.34) some additional terms are present, which are related to the derivatives of

u. This means that for stiff problems, where these derivatives are typically large, the additional terms may come into play and, thus significantly change the well-posedness condition. From (5.2.6) and assuming

$$\alpha_0 - h\,\beta_0\,u'(\eta) - h\,\beta_0\,\eta\,u''(\eta) > 0, \tag{5.2.36}$$

the first part on the right-hand side of (5.2.34) is actually equivalent to (5.2.5). Hence, observing the second part, assuming (5.2.36) and $\beta_0 > 0$, one should note that the well-posedness condition of the flow method is less severe than the related one of LMM if

$$x\,u''(x) < 0, \quad x \in [x_k, x_{q+1}]\,. \tag{5.2.37}$$

On the other hand, for $x\,u''(x) > 0$, the right-hand side of (5.2.34) becomes smaller than (5.2.5), i.e. (5.2.34) is more severe than (5.2.5).

5.3 Error analysis

Naturally the accuracy of the flow method depends on the accuracy of standard LMMs. However, since (5.1.6) is basically solved by the interpolation technique, there is an additional error coming from interpolation, as shown in the previous chapter. Again to show this we analyse the local error, say δ. The local error of a particular flow point x_j is defined as

$$\delta\left(x_j\left(t^{i+1}\right), h\right) := \frac{1}{h}\left|x_j\left(t^{i+1}\right) - x_j^{i+1}\right|. \tag{5.3.1}$$

Here $x_j\left(t^{i+1}\right)$ represents the value of the exact solution curve and x_j^{i+1} is the numerical solution obtained by the flow method assuming $x_j\left(t^{i-l+1}\right) = x_j^{i-l+1}$, $l = 1, \ldots, m$. The numerical solution of the standard LMM is assumed to be the exact solution of the nonlinear equation (5.1.6), which we denote by $\tilde{x}_j^{i+1}$. Hence by adding and subtracting $\tilde{x}_j^{i+1}$ in (5.3.1), we have

$$\delta(\cdot, h) \le \frac{1}{h}\left|x_j\left(t^{i+1}\right) - \tilde{x}_j^{i+1}\right| + \frac{1}{h}\left|\tilde{x}_j^{i+1} - x_j^{i+1}\right| =: d_h\left(x_j\left(t^{i+1}\right)\right) + r_h\left(x_j\left(t^{i+1}\right)\right). \tag{5.3.2}$$

Here $d_h(\cdot)$ represents the local discretisation error of the standard LMM and $r(\cdot)$ is the interpolation error. As shown in Section 2.5, we have

$$d_h\left(x_j\left(t^{i+1}\right)\right) = C\,h^p\,x^{(p+1)}\left(t^{i+1}\right) + O\left(h^{p+1}\right), \tag{5.3.3}$$

where C is the error constant that depends on the method and p is the consistency order of the LMM. For example, for BDF methods $p = m$, i.e. we have $d_h = O\left(h^m\right)$. The more interesting part of the local error is the interpolation error. It depends on the available data, the nature of the velocity field u and on the interpolation method involved. To illustrate this, let us find the local error of the flow method based on piece-wise linear interpolation.

Starting from (5.3.1) and substituting (5.1.18), with notation (5.2.20), we have

$$\begin{aligned} h\,\delta(\cdot, h) &= \left|x_j\left(t^{i+1}\right) - \frac{s_j^i + h\,\beta_0\left(u_k - x_k\frac{\Delta u_k}{\Delta x_k}\right)}{\alpha_0 - h\,\beta_0\frac{\Delta u_k}{\Delta x_k}}\right| \\ &= \left|\frac{\alpha_0\,x_j\left(t^{i+1}\right) - s_j^i - h\,\beta_0\,u_k - h\,\beta_0\,\left(x_j\left(t^{i+1}\right) - x_k\right)\frac{\Delta u_k}{\Delta x_k}}{\alpha_0 - h\,\beta_0\frac{\Delta u_k}{\Delta x_k}}\right|. \end{aligned} \tag{5.3.4}$$

Adding and subtracting the term $h\,\beta_0\,u\left(x_j\left(t^{i+1}\right)\right)$ in the numerator, the local error satisfies

$$\begin{aligned}\delta(\cdot,h) \le \frac{1}{h}&\left|\frac{\alpha_0\,x_j\left(t^{i+1}\right)-h\,\beta_0\,u\left(x_j\left(t^{i+1}\right)\right)-s_j^i}{\alpha_0-h\,\beta_0\,\frac{\Delta u_k}{\Delta x_k}}\right| \\ &+\left|\frac{\beta_0\left[u\left(x_j\left(t^{i+1}\right)\right)-u_k-\left(x_j\left(t^{i+1}\right)-x_k\right)\frac{\Delta u_k}{\Delta x_k}\right]}{\alpha_0-h\,\beta_0\,\frac{\Delta u_k}{\Delta x_k}}\right|.\end{aligned} \tag{5.3.5}$$

Recalling that s_j^i is defined by (5.1.5) and that $x_j\left(t^{i-l+1}\right)=x_j^{i-l+1}$, $l=1,\dots,m$, the first expression on the right-hand side of (5.3.5) is actually very close to $d_h(\cdot)$, i.e. the local discretisation error of LMM. The only (small) difference is in the denominator. However, since

$$x_k \le x_j^{i+1} \le x_{k+1}, \tag{5.3.6}$$

and by applying the mean value theorem, we have

$$\alpha_0-h\,\beta_0\,\frac{\Delta u_k}{\Delta x_k}=\alpha_0-h\,\beta_0\,u'(\eta),\quad x_k\le\eta\le x_{k+1}. \tag{5.3.7}$$

Hence by assuming that Δx_k is relatively small, we may adopt that the first expression on the right-hand side of (5.3.5) is $O\left(h^p\right)$. The second expression is, of course, the interpolation error for which we have

$$r_h(\cdot)=\left|\beta_0\left(\alpha_0-h\,\beta_0\,u'(\eta)\right)^{-1}\right|\left|u\left(x_j\left(t^{i+1}\right)\right)-u_k-\left(x_j\left(t^{i+1}\right)-x_k\right)\frac{\Delta u_k}{\Delta x_k}\right|. \tag{5.3.8}$$

Let us denote the second term on the right-hand side of (5.3.8) by $\tilde{r}(\cdot)$, i.e.

$$\tilde{r}(\cdot):=u\left(x_j\left(t^{i+1}\right)\right)-u_k-\left(x_j\left(t^{i+1}\right)-x_k\right)\frac{\Delta u_k}{\Delta x_k}. \tag{5.3.9}$$

Taylor expansions of $\frac{\Delta u_k}{\Delta x_k}$ and $u\left(x_j\left(t^{i+1}\right)\right)$ around x_k read

$$\frac{\Delta u_k}{\Delta x_k}=u'(x_k)+\frac{\Delta x_k}{2}\,u''(x_k)+O\left(\Delta x_k^2\right), \tag{5.3.10}$$

and

$$\begin{aligned}u\left(x_j\left(t^{i+1}\right)\right)=&u(x_k)+\left(x_j\left(t^{i+1}\right)-x_k\right)u'(x_k)\\ &+\frac{1}{2}\left(x_j\left(t^{i+1}\right)-x_k\right)^2u''(x_k)+O\left(\left(x_j\left(t^{i+1}\right)-x_k\right)^3\right).\end{aligned} \tag{5.3.11}$$

The order of $\left(x_j\left(t^{i+1}\right)-x_k\right)$ can be related to Δx_k by the following

$$x_j\left(t^{i+1}\right)-x_k=x_j^{i+1}-x_k+h\,\delta(\cdot,h)\le\Delta x_k+h\,\delta(\cdot,h). \tag{5.3.12}$$

Assuming that $\Delta x_k > h\,\delta(\cdot,h)$, which can always be achieved for h small enough, we have

$$O\left(\left(x_j\left(t^{i+1}\right)-x_k\right)\right)=O\left(\Delta x_k\right). \tag{5.3.13}$$

Hence, by substituting (5.3.10) and (5.3.11) into (5.3.9), and neglecting higher order terms, we obtain

$$\tilde{r}(\cdot)=\frac{1}{2}\left(x_j\left(t^{i+1}\right)-x_k\right)\left(x_j\left(t^{i+1}\right)-x_{k+1}\right)u''(x_k). \tag{5.3.14}$$

Substituting this into (5.3.8), the interpolation error reads

(5.3.15)
$$r_h(\cdot) = \left| \frac{\beta_0}{2} \left(\alpha_0 - h\,\beta_0\,u'(\eta)\right)^{-1} \right| \left| \left(x_j\left(t^{i+1}\right) - x_k\right)\left(x_j\left(t^{i+1}\right) - x_{k+1}\right) u''(x_k) \right|.$$

If $x_j\left(t^{i+1}\right) \in [x_k, x_{k+1}]$, we find the error bound of inverse linear interpolation

(5.3.16)
$$r_h(\cdot) \leq \frac{\Delta x_k^2}{8} \left| \beta_0 \left(\alpha_0 - h\,\beta_0\,u'(\eta)\right)^{-1} u''(x_k) \right|.$$

According to (5.3.2), (5.3.3) and (5.3.16), the local error of the flow method is $O\left(h^p\right) + O\left(\Delta x_k^2\right)$. If both errors are of the same order, the flow method has the same accuracy as standard LMM. To achieve this, the spatial step size Δx_k should be $O\left(h^{\frac{p}{2}}\right)$. Hence for higher order LMMs, i.e. p large, and h relatively small, we need larger numbers of data points. Of course, higher order interpolation would do a better job and, thus, overcome this problem. To illustrate this let us solve the following example.

EXAMPLE 5.3.1. Consider the flow problem

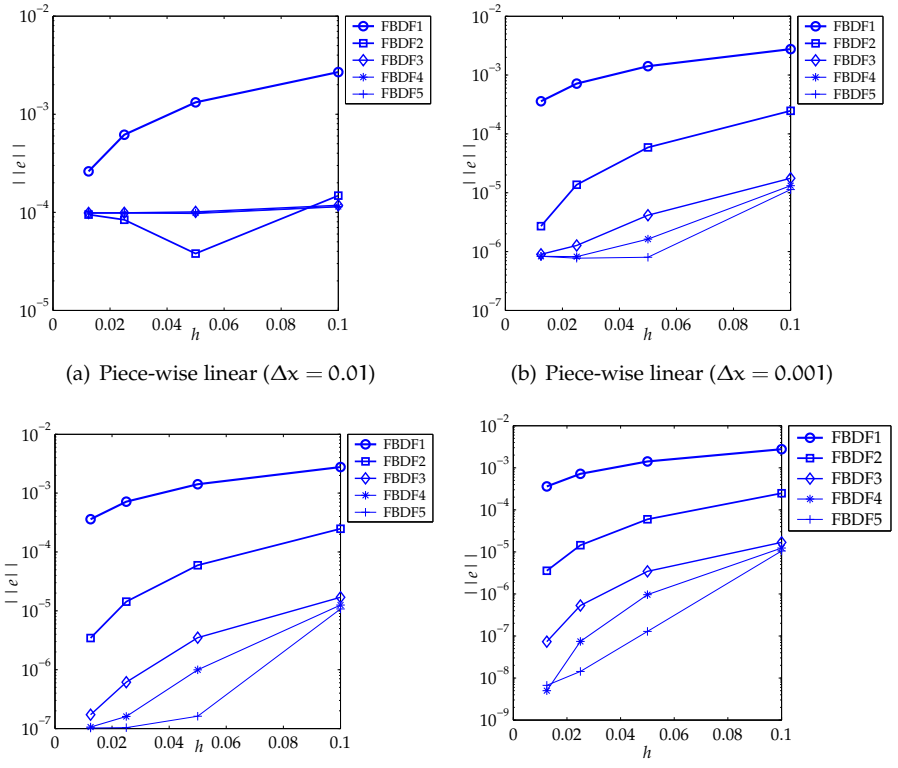
(5.3.17)
$$\begin{cases} \dot{x} = -\arctan(10\,x) + \cos t, \\ x(0) \in I^0 = [-1, 1]. \end{cases}$$

Let us solve this problem by applying the flow method, based on BDF methods of various order $m = 1, \ldots, 5$ and various types of interpolation. We denote these methods by FBDFm. Instead of using (in this particular case) the known autonomous part u of the velocity field, we first discretise I^0 by n equidistant points and obtain a set of pairs $\{(x_k, u_k)\}_{k=0}^n$, which then we use in our implementation. In the first experiment we choose piece-wise linear interpolation. For $n = 200$, i.e. $\Delta x = 0.01$, the results are shown in Figure 5.3.1a. Since Δx is not sufficiently small (for higher order BDF methods), the interpolation error is dominant and the accuracy cannot be increased by employing a BDF method of the higher order. The situation is somewhat better if we decrease the spatial step, say by a factor of 10, see Figure 5.3.1b. However, for h relatively small and $m \geq 4$, we encounter the same accuracy problem. Clearly, the interpolation error represents a sort of accuracy threshold. Only if we take a sufficiently small spatial step, e.g. $\Delta x = 0.0001$, we can obtain the full accuracy of all BDF methods, see Figure 5.3.1c. Rather than decreasing Δx, we clearly achieve better accuracy by employing higher order interpolation. For example, if we choose piece-wise cubic interpolation, errors are almost identical to ones of the standard BDF methods even for relatively large spatial steps, see Figure 5.3.1d. □

5.4 Stability

The stability of the general LMM is usually addressed by computing the stability domain of the method. Once obtained (see 2.5.2), it can be used to determine does the method satisfy some of the stability properties, such as A, $A(\alpha)$ or L-stability. In the Example 5.1.1 it was shown that the flow method (based on piece-wise linear interpolation) gives exactly the same result as the standard LMM. This leads to a conclusion that the flow method has the same stability domain as the standard LMM on which the flow method is based. Hence, the flow method is A-stable if the related LMM is A−stable. The same analogy can be made for $A(\alpha)$-stable LMMs.

As mentioned before, the most popular choice of LMMs for solving stiff problems are BDF methods. Hence, it is natural to believe that the flow method based on BDF



(a) Piece-wise linear ($\Delta x = 0.01$)

(b) Piece-wise linear ($\Delta x = 0.001$)

(c) Piece-wise linear ($\Delta x = 0.0001$)

(d) Piece-wise cubic ($\Delta x = 0.01$)

FIGURE 5.3.1. The error of the flow method

methods will have similar stability properties, also in the general nonlinear case. Let us first analyse how the numerical solution of the flow method behaves for a stiff problem. From (5.1.18) and (5.1.15) we have

$$x_j^{i+1} = \frac{-\sum_{l=1}^{m} \alpha_l x_j^{i-l+1} + h\left(u_k - x_k \frac{\Delta u_k}{\Delta x_k}\right)}{\alpha_0 - h \frac{\Delta u_k}{\Delta x_k}}. \tag{5.4.1}$$

Supposing that $|h\,u'(x)|$ is very large for $x \in I(x,t)$, which is typically true for stiff problems, the expression (5.4.1) becomes

$$x_j^{i+1} = x_k - \frac{u_k}{\frac{\Delta u_k}{\Delta x_k}}. \tag{5.4.2}$$

This is actually one step of the discrete Newton method applied to the equation

$$u(x) = 0, \tag{5.4.3}$$

with initial values x_k and x_{k+1}. Hence, assuming the Newton method to be convergent, all solution points rapidly move toward stationary points of the ODE (5.1.1), as they should. In particular, for a convergent flow, where $u'(x) < 0$, $x \in I(x,t)$, the flow of numerical solutions converges quickly and the numerical solutions are stable. One should note that this argument does not hold for the general LMM. Here one should replace (5.1.15) by (5.1.5), introducing u-values in the numerator of (5.4.1). Hence (5.4.1) is not necessarily equivalent to (5.4.2) for a stiff problem. This

means that the solution of the flow method based on an LMM that is not (at least) $A(\alpha)$-stable (e.g. Adams method) would most likely be unstable.

The global error of the particular flow point is defined by

$$e^i := \left| x_j\left(t^i\right) - x_j^i \right|. \tag{5.4.4}$$

For problems where u satisfies a Lipschitz condition, it is known (see e.g. [24]) that the global error of the numerical solution can be related to the local error. In the previous section it was shown that the local error of the flow method consists of two components: the local discretisation error and the interpolation error. Hence we can obtain the global error bound as

$$e^i \leq K_1 \left(\max_{0\leq\nu<m} \left| x_j\left(t^\nu\right) - x_j^\nu \right| + h \max_{0\leq\nu<m} \left| \dot{x}_j\left(t^\nu\right) - u\left(x_j^\nu\right) \right| \right) + K_2\, h^p + K_3\, \Delta x_k^q, \tag{5.4.5}$$

where p represents the consistency order of the LMM involved and q the order of accuracy of the applied interpolation method. The constant K_1 depends on the type of LMM involved, K_2 on bounds for $x_j^{(p+1)}(t)$ and K_3 on bounds for $u^{(q)}(x)$. Clearly by controlling the interpolation error, the flow method convergence behaviour is similar to that of the related LMM.

CHAPTER 6

The flow method based on Runge-Kutta methods

This chapter is devoted to the implementation of implicit Runge-Kutta (IRK) schemes in the flow method. Like in the previous two chapters we consider the same problem settings. Since IRK schemes are more involved than LMMs, we first discuss how to implement IRK methods in the flow method. In Section 6.2 we give conditions for the method to be well-posed. We conclude this chapter by an error analysis and discuss stability properties of the method in Section 6.3 and Section 6.4 respectively.

6.1 Method implementation

Again we consider the flow problem

$$\begin{cases} \dot{x} = u(x), \\ x(0) \in I(x, 0), \end{cases} \tag{6.1.1}$$

with a discrete velocity field $u(x)$, given by a set of pairs $\{(x_k, u_k)\}_{k=0}^{n}$. For simplicity, we assume points x_k, $k = 0, \dots, n$ to be equidistant, i.e. for some given step-size Δx we have

$$x_k = x_0 + k\,\Delta x, \quad k = 0, \dots, n. \tag{6.1.2}$$

Now we discuss how can we employ IRK (Implicit Runge-Kutta) methods in the flow method. To this end consider the Butcher matrix

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \\ c_s & a_{s1} & & a_{ss} \\ \hline & b_1 & & b_s \end{array},$$

which corresponds to the system of equations

$$x^{i+1} = x^i + h \sum_{l=1}^{s} b_l\, u\left(y^{il}\right), \tag{6.1.3}$$

$$y^{il} = x^i + h \sum_{q=1}^{s} a_{lq}\, u\left(y^{il}\right). \tag{6.1.4}$$

Here y^{il}, $l = 1, \ldots, s$, denote inner-stages of the IRK method. Basically they represent numerical approximations of the solution at time points $t^i + c_l\, h$, $l = 1, \ldots, s$. Rewriting (6.1.4) in vector form, we have the $s-$dimensional system

$$\begin{bmatrix} y^{i1} \\ \vdots \\ y^{is} \end{bmatrix} - h \begin{bmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & & \\ a_{s1} & & a_{ss} \end{bmatrix} \begin{bmatrix} u\left(y^{i1}\right) \\ \vdots \\ u\left(y^{is}\right) \end{bmatrix} = x^i \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{6.1.5}$$

By defining $\mathbf{y}^i := \begin{bmatrix} y^{i1} & \cdots & y^{is} \end{bmatrix}^T$, $\mathbf{A} := \{a_{ij}\}_{s\times s}$, $\mathbf{e} := \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T$ and

$$\mathbf{u}\left(\mathbf{y}^i\right) := \begin{bmatrix} u\left(y^{i1}\right) & \cdots & u\left(y^{is}\right) \end{bmatrix}^T, \tag{6.1.6}$$

the system (6.1.5) can be written as

$$\mathbf{y}^i - h\,\mathbf{A}\,\mathbf{u}\left(\mathbf{y}^i\right) = x^i\,\mathbf{e}. \tag{6.1.7}$$

This system needs to be solved at every time level and for all flow points. Defining a vectorial function $\mathbf{f}$ by

$$\mathbf{f}\left(\mathbf{y}^i\right) := \mathbf{y}^i - h\,\mathbf{A}\,\mathbf{u}\left(\mathbf{y}^i\right), \tag{6.1.8}$$

it is clear that the value of $\mathbf{y}^i$ follows from the inverse function of $\mathbf{f}$, say $\mathbf{g}$, at point $x^i\,\mathbf{e}$, i.e.

$$\mathbf{y}^i = \mathbf{g}\left(x^i\,\mathbf{e}\right) := \mathbf{f}^{-1}\left(x^i\,\mathbf{e}\right). \tag{6.1.9}$$

Since $\mathbf{g}$ cannot be found explicitly in general and since u (and thus consequently $\mathbf{u}$) is only known discretely, the implementation of IRK methods is not straightforward. However, we can employ the main idea of the flow method, i.e. inverse interpolation, in order to find an approximation of the inverse function. The main difference with preceding chapters is that the interpolation function, say $\mathbf{p}$, is now a vectorial and multivariate function, i.e. the solution of inner-stage values is obtained from

$$\mathbf{y}^i = \mathbf{p}\left(x^i\,\mathbf{e}\right). \tag{6.1.10}$$

This vectorial function problem can be simplified if the employed IRK method has $c_s = 1$, i.e. if the following holds

$$x^{i+1} = y^{is}. \tag{6.1.11}$$

In this case one needs to obtain only a scalar multivariate function $p_s(\mathbf{x})$ and the solution at the next time level follows from

$$x^{i+1} = p_s\left(x^i\,\mathbf{e}\right). \tag{6.1.12}$$

There is another important advantage of methods which have $c_s = 1$. If $c_s \neq 1$, one needs to compute x^{i+1} by using (6.1.3), for which the values of $u\left(y^{il}\right)$, $l = 1, \ldots, s$ are required. Since $u(x)$ is not known explicitly, this implies that these values need to be approximated as well, which will introduce additional errors and computational costs.

There is also an alternative approach, i.e. to reformulate the method by increasing the number of stages and setting $c_{s+1} = 1$. The reformulated method is then

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} & 0 \\ \vdots & \vdots & & & \\ c_s & a_{s1} & & a_{ss} & 0 \\ 1 & b_1 & & b_s & 0 \\ \hline & b_1 & & b_s & 0 \end{array}$$

Although the implementation problem is overcome this way, one should note that the matrix $\mathbf{A}$ is now significantly changed. Fortunately there exists a number of methods which have $c_s = 1$ such as Radau IIA, Lobatto IIIA, Lobatto IIIB and Lobatto IIIC (see Section 2.6). In this chapter we pay special attention to them, since they are frequently used for solving stiff problems (see e.g. [24], [10], etc.).

The flow method based on the standard IRK method can be constructed as follows. Since the nonlinear system to be solved is $s-$dimensional, we first need to create an $s-$dimensional grid, which has function values known at all nodes. By letting y^{il}, $l = 1, \ldots, s$ to be independent variables in $\mathbb{R}^s$, we can define a point, say $\mathbf{y}$, of such a space by the vector

$$\mathbf{y} := \begin{bmatrix} y^1 & \cdots & y^s \end{bmatrix}^T. \tag{6.1.13}$$

According to (6.1.8) we need to find a set of points, say $\{\mathbf{y}_K\}_K$, for which $\mathbf{f}(\mathbf{y}_K)$ can be computed. Hence we employ the known scalar data set, i.e. the set $\{(x_k, u_k)\}_{k=0}^n$ and create a set of $s-$dimensional points $\{\mathbf{y}_K\}_{K=1}^{s(n+1)}$, as follows

$$\begin{aligned} \mathbf{y}_1 &= \begin{bmatrix} x_0 & x_0 & \cdots & x_0 \end{bmatrix}^T, \\ \mathbf{y}_2 &= \begin{bmatrix} x_0 & x_0 & \cdots & x_1 \end{bmatrix}^T, \\ &\vdots \\ \mathbf{y}_{s(n+1)} &= \begin{bmatrix} x_n & x_n & \cdots & x_n \end{bmatrix}^T. \end{aligned} \tag{6.1.14}$$

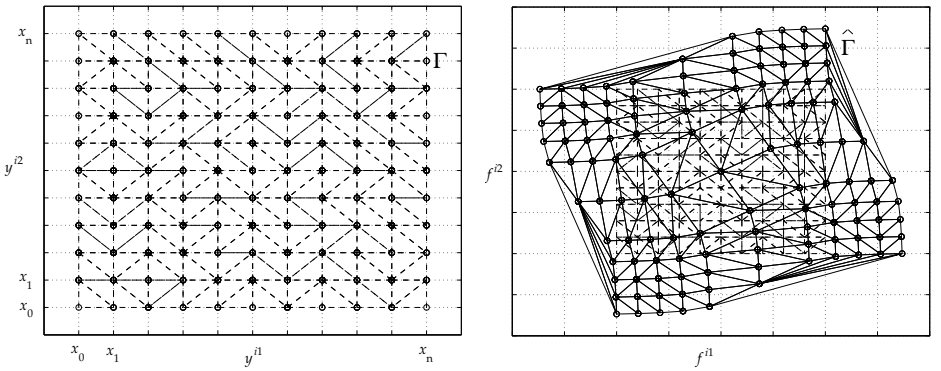
Thus obtained points can be used for creating a grid (e.g. by triangulation). The convex hull of such a grid is denoted by Γ and we naturally have $\mathbf{y}_K \in \Gamma \subset \mathbb{R}^s$. A two-dimensional example is illustrated in Figure 6.1.1a.

Clearly from (6.1.5) and (6.1.8) it follows that we can compute $\mathbf{f}(\mathbf{y}_K)$ for every $\mathbf{y}_K$ defined by (6.1.14). This means that we can map the original grid onto a new grid, defined by points

$$\mathbf{f}_K := \mathbf{y}_K - h\,\mathbf{A}\,\mathbf{u}(\mathbf{y}_K). \tag{6.1.15}$$

Of course, we now have a new convex hull, say $\hat{\Gamma}$, defined by points $\mathbf{f}_K$. In Figure 6.1.1b a two-dimensional example is shown, where the original grid (shown in Figure 6.1.1a) is mapped by (6.1.15), where $\mathbf{A}$ is determined by the chosen IRK method.

The approximation function $\mathbf{p}$ can be found by interpolating points $\{(\mathbf{f}_K, \mathbf{y}_K)\}_{k=1}^{s(n+1)}$ and requiring $\mathbf{p}(\mathbf{f}_K) = \mathbf{y}_K$, $K = 1, \ldots, s(n+1)$, i.e. $p_s(\mathbf{f}_K) = y_K^s$ for methods where $c_s = 1$. Obviously the interpolation problem which arises here for the scalar problem (6.1.1) is closely related to general vectorial problems, which we analyse in the following chapter. However, there is one important difference. The vectorial function $\mathbf{u}$, defined by (6.1.6), has univariate functions as elements instead of multivariate ones as in general vectorial problems. This is of importance (as shown later) for the analysis of the method properties.



(a) Original grid (b) Mapped grid

FIGURE 6.1.1. Mapping of the grid for two-stage IRK method

An important class of IRK methods are so-called DIRK (Diagonally Implicit Runge-Kutta) methods, which were introduced in Chapter 2 and defined by

$$\begin{array}{c|cccc} c_1 & a_{11} & 0 & \cdots & 0 \\ c_2 & a_{21} & a_{22} & & 0 \\ \vdots & \vdots & & & \\ c_s & a_{s1} & a_{s2} & & a_{ss} \\ \hline & b_1 & b_2 & & b_s \end{array}$$

There are two possible ways for solving (6.1.7). The first one, of course, is to apply the algorithm for the general IRK method, introduced above. An alternative way is to apply the univariate interpolation s−times, i.e. to find approximations of inverse scalar functions for each inner-stage. Rewriting the equation for the first inner-stage, we have

$$y^{i1} - h\, a_{11}\, u\left(y^{i1}\right) = x^i. \tag{6.1.16}$$

One should note that this is almost identical (with a modified time-step $\tilde{h} = a_{11}\, h$) to the Euler Backward method and it can be employed as shown in Chapter 4. Hence we can obtain $p_1(x)$ and compute y^{i1} by

$$y^{i1} = p_1\left(x^i\right). \tag{6.1.17}$$

The second inner-stage value then follows from

$$y^{i2} - h\, a_{22}\, u\left(y^{i2}\right) = x^i + h\, a_{21}\, u\left(y^{i1}\right), \tag{6.1.18}$$

and the solution is obtained by

$$y^{i2} = p_2\left(x^i + h\, a_{21}\, u\left(y^{i1}\right)\right). \tag{6.1.19}$$

Obviously an additional problem arises here. Since we do not know u explicitly, we cannot compute the value of $u\left(y^{i1}\right)$ directly. Hence it needs to be approximated (e.g. by direct interpolation). The same problem is present in computing values of all following stages, where we have one extra evaluation to perform per each inner-stage. Finally, for the last stage we have

$$y^{is} - h\, a_{ss}\, u\left(y^{is}\right) = x^i + h \sum_{q=1}^{s-1} a_{sq}\, u\left(y^{iq}\right). \tag{6.1.20}$$

Clearly to obtain the solution one needs $s-$times additional direct interpolation evaluations (in case $c_s \neq 1$, otherwise $s-1$), plus $s-$times univariate inverse interpolation evaluation.

As seen in previous chapters, the flow method can also be efficiently applied to a class of non-autonomous ODEs of the following type

$$\dot{x} = u(x) + w(t). \tag{6.1.21}$$

The same holds for the flow method that employs IRK methods. Here the interpolation principle remains the same, but since t is explicitly present on the right-hand side of (6.1.21), coefficients c_l, $l = 1, \ldots, s$ are coming into play. Now one needs to modify the nonlinear system (6.1.7) to

$$\mathbf{y}^i - h\,\mathbf{A}\,\mathbf{u}\left(\mathbf{y}^i\right) = x^i\,\mathbf{e} + h\,\mathbf{A}\,\mathbf{w}\left(t^i\right), \tag{6.1.22}$$

where

$$\mathbf{w}\left(t^i\right) := \left[w\left(t^i + c_1\,h\right) \ldots w\left(t^i + c_s\,h\right)\right]. \tag{6.1.23}$$

The solution at the next time level then follows from

$$x^{i+1} = p_s\left(x^i\,\mathbf{e} + h\,\mathbf{A}\,\mathbf{w}\left(t^i\right)\right). \tag{6.1.24}$$

6.2 Well-posedness

As shown in previous chapters, the well-posedness is an important issue in solving flow problems. Some methods, in particular BDF methods of even order, posed severe constraints for the time-step in order to avoid intersections of integral curves. In this section we analyse the behaviour of the flow method employing IRK methods. Since the approach, introduced in the previous section, uses interpolation in higher dimensional space, we distinguish two aspects, namely:

(1) Topology preservation in space, i.e. conditions for the preservation of the grid orientation.
(2) Topology preservation in time, i.e. the absence of the intersections of the integral curves. We keep the name well-posedness associated to this aspect like in previous chapters.

Both aspects are analysed in following two subsections.

6.2.1 Topology preservation in space

The property is closely related to the nonlinear mapping of the original (generated) grid, defined by (6.1.14), into the new grid. Starting from the linear test problem

$$\dot{x} = \lambda\,x, \quad \lambda < 0, \tag{6.2.1}$$

and recalling (6.1.15), the new grid points are defined by

$$\mathbf{f}_K := \mathbf{f}\left(\mathbf{x}_K\right) = \left(\mathbf{I}_s - h\,\lambda\,\mathbf{A}\right)\,\mathbf{x}_K, \tag{6.2.2}$$

where $\mathbf{I}_s$ is the identity matrix of order s. As shown in Chapter 6, the grid orientation is preserved if

$$\det D\mathbf{f} > 0. \tag{6.2.3}$$

Here, we have

$$\det D\mathbf{f} = \det\left(\mathbf{I}_s - h\,\lambda\,\mathbf{A}\right). \tag{6.2.4}$$

The determinant of $D\mathbf{f}$ is equal to the product of its eigenvalues, say ξ_l, $l = 1, \ldots, s$. Assuming that all ξ_l are real, the determinant $\det D\mathbf{f}$ is necessarily positive if the real eigenvalues of $\mathbf{A}$, say η_l, $l = 1, \ldots, s$, are positive. Indeed, starting from

$$\mathbf{A}\,\mathbf{x} = \eta_l\,\mathbf{x}, \tag{6.2.5}$$

it is easy to show that

$$(\mathbf{I}_s - h\lambda\,\mathbf{A})\,\mathbf{x} = \mathbf{x} - h\lambda\,\mathbf{A}\,\mathbf{x} = (1 - h\lambda\eta_l)\,\mathbf{x}. \tag{6.2.6}$$

This means that the eigenvalues of $D\mathbf{f}$ are defined by $\xi_l = 1 - h\lambda\eta_l$, $l = 1, \ldots, s$. Obviously, since $\lambda < 0$ and $\eta_l > 0$, all real eigenvalues of $D\mathbf{f}$ are positive, i.e. we have $1 - h\lambda\eta_l > 0$ and consequently $\det D\mathbf{f} > 0$. The same conclusion also holds if some eigenvalues are complex. Since $\mathbf{A}$ is a real matrix, clearly all complex eigenvalues η_l are present as complex conjugate pairs, which means that their product is always positive. For example, for arbitrary $\eta_l \in \mathbb{C}$ we have another eigenvalue (conjugate pair), say $\eta_{l+1} = \eta_l^*$ and their product is then

$$\eta_l\,\eta_{l+1} = |\eta_l|^2 > 0. \tag{6.2.7}$$

Of course, the same also holds for the corresponding pair ξ_l, ξ_{l+1}. Hence one should note that complex eigenvalues do not change sign of $\det D\mathbf{f}$.

In the general nonlinear case, where $D\mathbf{u}$ cannot be expressed as a constant times the identity matrix, the positivity of $\det D\mathbf{f}$ is not guaranteed, not even by employing certain properties of the matrix $\mathbf{A}$, coming from the principles of creating standard IRK methods. The eigenvalues of $D\mathbf{f}$ are now related to the eigenvalues, say μ_l, $l = 1, \ldots, s$, of the matrix, say

$$\mathbf{M} := -\mathbf{A}\,D\mathbf{u}. \tag{6.2.8}$$

The corresponding Jacobian matrix of $\mathbf{f}$ is then

$$D\mathbf{f} = \mathbf{I}_s + h\,\mathbf{M}, \tag{6.2.9}$$

and we can draw similar conclusions concerning relations between ξ_l and corresponding μ_l, as we did in the linear case. Hence, the topology is necessarily preserved if all real eigenvalues μ_l are non-negative. To show the relation between this condition and the properties of the matrix $\mathbf{A}$, we will now analyse the implementation of IRK methods with a relatively small number of stages ($s = 2, 3, 4$) and assume that

$$u'(x) < 0, \tag{6.2.10}$$

holds on the entire domain of interest.

Let us start with two-stage IRK methods, for which the matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}, \tag{6.2.11}$$

where d_1 and d_2 are the eigenvalues of the matrix $[-D\mathbf{u}]$. Since we assume that (6.2.10) holds, they are both positive, i.e. $d_1, d_2 > 0$.

THEOREM 6.2.1. *Let the IRK method have 2 stages and let $d_1, d_2 > 0$. If*

$$\begin{aligned} &a_{ll} \geq 0, \quad l = 1, 2, \\ &\det \mathbf{A} \geq 0, \end{aligned} \tag{6.2.12}$$

the real eigenvalues of the matrix $\mathbf{M}$ are non-negative.

PROOF. By obtaining the characteristic polynomial of $\mathbf{M}$, say $P(\mu)$, by

$$P(\mu) := \det(\mu \mathbf{I}_s - \mathbf{M}), \tag{6.2.13}$$

and comparing with

$$P(\mu) = (\mu - \mu_1)(\mu - \mu_2), \tag{6.2.14}$$

one can obtain the following system of equation

$$\begin{aligned} \mu_1 + \mu_2 &= d_1 a_{11} + d_2 a_{22}, \\ \mu_1 \mu_2 &= d_1 d_2 \det \mathbf{A}. \end{aligned} \tag{6.2.15}$$

Applying the assumption (6.2.12), we have that the eigenvalues μ_1 and μ_2, defined by the system (6.2.15), satisfy

$$\begin{aligned} \mu_1 + \mu_2 &\geq 0, \\ \mu_1 \mu_2 &\geq 0. \end{aligned} \tag{6.2.16}$$

Since we assumed that μ_1 and μ_2 are real, it is obvious that they are non-negative. □

In order to obtain conditions for the eigenvalues μ_l to be non-negative for the three-stage IRK method, we may use the same approach.

THEOREM 6.2.2. *Let the IRK method have 3 stages and let the eigenvalues of* $D\mathbf{u}$ *satisfy* $d_1, d_2, d_3 > 0$. *If*

$$\begin{aligned} a_{ll} &\geq 0, \\ A_{ll} &\geq 0, \quad l = 1,2,3, \\ \det \mathbf{A} &\geq 0, \end{aligned} \tag{6.2.17}$$

where A_{ll}, $l = 1,2,3$, *are two-by-two minors of* $\mathbf{A}$ *related to diagonal elements* a_{ll}, $l = 1,2,3$, *respectively, the real eigenvalues of the matrix* $\mathbf{M}$ *are non-negative.*

PROOF. Repeating the procedure from the previous proof, one can obtain the system which corresponds to (6.2.15) as

$$\begin{aligned} \mu_1 + \mu_2 + \mu_3 &= d_1 a_{11} + d_2 a_{22} + d_3 a_{33}, \\ \mu_1 \mu_2 + \mu_1 \mu_3 + \mu_2 \mu_3 &= d_1 d_2 A_{33} + d_1 d_3 A_{22} + d_2 d_3 A_{11}, \\ \mu_1 \mu_2 \mu_3 &= d_1 d_2 d_3 \det \mathbf{A}, \end{aligned} \tag{6.2.18}$$

Assuming (6.2.17) the left-hand side of the system (6.2.18) satisfy

$$\mu_1 + \mu_2 + \mu_3 \geq 0, \tag{6.2.19}$$

$$\mu_1 \mu_2 + \mu_1 \mu_3 + \mu_2 \mu_3 \geq 0, \tag{6.2.20}$$

$$\mu_1 \mu_2 \mu_3 \geq 0. \tag{6.2.21}$$

Since all eigenvalues are assumed to be real, it can be shown that (6.2.19-6.2.21) holds if and only if $\mu_1, \mu_2, \mu_3 \geq 0$. Let us suppose the opposite, i.e. assume that some eigenvalues are negative. Clearly, from (6.2.21) it follows that at least one eigenvalue, say μ_1, needs to be non-negative. One should note that a situation where one eigenvalue is equal to zero, reduces the system to the form (6.2.16), for which we saw that holds only if both remaining eigenvalues are non-negative. Hence we take the other two to be strictly negative. From (6.2.19) we have

$$\mu_1 \geq -\mu_2 - \mu_3. \tag{6.2.22}$$

Rewriting (6.2.20) and employing (6.2.22) gives

$$\mu_2\,\mu_3 \geq \mu_1\,(-\mu_2-\mu_3) \geq (-\mu_2-\mu_3)^2\,. \tag{6.2.23}$$

This finally leads to a condition

$$\mu_2^2+\mu_2\,\mu_3+\mu_3^2 \leq 0, \tag{6.2.24}$$

which holds only if $\mu_2 = \mu_3 = 0$ and it is a contradiction to the assumption that μ_2 and μ_3 are strictly negative. Hence the system (6.2.18) has a solution only if all eigenvalues are non-negative. □

Following the same approach, it can shown that the similar result can be found for the four-stage IRK methods, as shown by the following theorem which we give without proof.

THEOREM 6.2.3. *Let the IRK method have* 4 *stages and let the eigenvalues of* $D\mathbf{u}$ *satisfy* $d_1, d_2, d_3, d_4 > 0$. *If*

$$\begin{aligned} a_{ll} &\geq 0, \\ A_{ll} &\geq 0, \quad l = 1,2,3,4 \\ \alpha_{lm} &\geq 0, \quad l = 1,2,3, \quad m = l+1,\ldots,4, \\ \det \mathbf{A} &\geq 0, \end{aligned} \tag{6.2.25}$$

where A_{ll}, $l = 1,2,3,4$, *are 3-by-3 minors of* $\mathbf{A}$ *related to diagonal elements* a_{ll}, $l = 1,2,3,4$, *respectively, and* α_{lm} *are 2-by-2 minors of* $\mathbf{A}$ *defined by*

$$\alpha_{lm} = \begin{vmatrix} a_{ll} & a_{lm} \\ a_{ml} & a_{mm} \end{vmatrix}, \tag{6.2.26}$$

then the real eigenvalues of the matrix $\mathbf{M}$ *are non-negative.*

The condition for the topology preservation in space, given by (6.2.3) is equivalent to the condition that the real eigenvalues of $\mathbf{M}$ are non-negative, as shown by the following.

COROLLARY 6.2.4. *If the real eigenvalues of the matrix* $\mathbf{M}$ *are non-negative, then the Jacobian of the matrix* $D\mathbf{f}$ *is strictly positive.*

PROOF. From (6.2.9) we clearly have

$$\xi_l = 1 + h\,\mu_l, \tag{6.2.27}$$

and since the real μ_l is non-negative, the corresponding ξ_l is strictly positive. Of course, since all real ξ_l are positive, it follows that $D\mathbf{f} > 0$. □

Conditions (6.2.12), (6.2.17) and (6.2.25) hold for a number of important IRK methods, such as Gauss and the whole Radau and Lobatto families, making all of them suitable candidates concerning the topology preserving property. To illustrate this, we consider these three-stage IRK methods and show the values of determinants and relevant minors present in (6.2.17), see Table 6.2.1. One should note that all determinants and relevant minors are non-negative.

Method	Gauss	Rad. IA	Rad. IIA	Lob. IIIA	Lob. IIIB	Lob. IIIC
$\det \mathbf{A}$	0.0083	0.0167	0.0167	0	0	0.0417
A_{11}	0.0417	0.0833	0.0537	0.0833	0	0.1250
A_{22}	0.0167	0.0129	0.0129	0	0	0
A_{33}	0.0417	0.0537	0.0833	0	0.0833	0.1250

TABLE 6.2.1. Determinants and minors of three-stage IRK methods

If the method of choice is a DIRK method, the condition for topology preservation is simpler for multivariate interpolation. Here we have that the eigenvalues ξ_l, $l = 1, \ldots, s$, that correspond to the matrix $D\mathbf{f}$ are determined by

$$\xi_l = 1 + h\, a_{ll}\, d_l, \quad l = 1, \ldots, s. \tag{6.2.28}$$

This follows from the fact that the eigenvalues of $\mathbf{M}$ are $\mu_l = a_{ll}\, d_l$, which is a consequence of $\mathbf{A}$ being triangular matrix. Assuming (6.2.10) again, it is clear that $\xi_l > 0$ if

$$a_{ll} > 0, \quad l =, \ldots, s. \tag{6.2.29}$$

Hence the condition (6.2.29) represents the topology preservation condition for DIRK methods. Typically this holds for all DIRK methods of interest, implying that one should not expect problems in their implementation.

6.2.2 Topology preservation in time

As before the well-posedness property of the method can be formulated as the condition that guarantees the absence of intersections between integral curves of flow points during the integration time whenever u is Lipschitz continuous. It can be simplified by observing the behaviour of the solution points over a single time-step and then generalised by requiring that it holds for the whole integration interval, i.e. for each time-step. As shown below, the well-posedness condition is related to properties of the employed IRK method. Moreover the choice of the interpolation method may also be of importance. However, we will restrict ourselves to piece-wise linear interpolation and analyse results for different IRK methods of interest.

Let us start by assuming that there are no intersections between flow points solutions up to the time-level t^i. This can be expressed by relating arbitrary flow points, say x_j^i and x_{j+1}^i, by

$$x_{j+1}^i - x_j^i > 0, \tag{6.2.30}$$

and, of course, assuming that the same holds at every previous time level. The well-posedness property is then a requirement that the same holds at the next time level t^{i+1}. Since employing an IRK method introduces inner-stage solutions, this property may also be generalised to include similar conditions for every inner-stage. However, we are mainly interested in the condition for the last inner-stage, which is closest to solutions at t^{i+1} or even identical for methods where $c_s = 1$. Starting from (6.1.10) and employing linear interpolation, we have

$$\mathbf{y}_j^i = x_j^i\, \mathbf{B}_j\, \mathbf{e} + \mathbf{d}_j, \tag{6.2.31}$$

where $\mathbf{y}_j^i := \left[y_j^{i1} \cdots y_j^{is}\right]^T$ is the vector containing inner-stage values which correspond to the flow point x_j. The matrix $\mathbf{B}_j$ and the vector $\mathbf{d}_j$ are defined by inverse piece-wise linear interpolation, i.e. by the set $\{(\mathbf{f}_K, \mathbf{y}_K)\}_{K=M}^{M+s}$. Here the index M is related to the M−th element, say $\hat{\Gamma}_M$, of the new grid which contains the point $\mathbf{y} = x_j^i\,\mathbf{e}$. Similarly the inner-stage vector of the point x_{j+1} is

$$\mathbf{y}_{j+1}^i = x_{j+1}^i{}_{\cdot}\,\mathbf{B}_{j+1}\,\mathbf{e} + \mathbf{d}_{j+1}. \tag{6.2.32}$$

Subtracting (6.2.31) from (6.2.32), we have

$$\mathbf{y}_{j+1}^i - \mathbf{y}_j^i = \left[x_{j+1}^i{}_{\cdot}\,\mathbf{B}_{j+1} - x_j^i\,\mathbf{B}_j\right]\,\mathbf{e} + \mathbf{d}_{j+1} - \mathbf{d}_j. \tag{6.2.33}$$

For the well-posedness requirement the vector $\mathbf{y}_{j+1}^i - \mathbf{y}_j^i$ should clearly be positive element-wise. Considering the last stage only, the condition reads

$$y_{j+1}^{is} - y_j^{is} = x_{j+1}^i{}_{\cdot}\,\mathbf{b}_{\Sigma,j+1} - x_j^i\,\mathbf{b}_{\Sigma,j} + d_{s,j+1} - d_{s,j}, \tag{6.2.34}$$

where $\mathbf{b}_{\Sigma,j} := \left[\sum_{l+1}^{s} b_{1l}, \ldots, \sum_{l+1}^{s} b_{sl}\right]^T = \left[b_{\Sigma,j}^1, \ldots, b_{\Sigma,j}^s\right]^T$.

The expressions (6.2.33) and (6.2.34) are not easily assessable for the analysis due to the complexity of the inverse interpolation used for determining $\mathbf{B}_j$ and $\mathbf{d}_j$. However, if we assume that points $x_j^i\,\mathbf{e}$ and $x_{j+1}^i\,\mathbf{e}$ are in the same element $\hat{\Gamma}_M$ of the new grid, the expression (6.2.33) is significantly simplified, i.e.

$$\mathbf{y}_{j+1}^i - \mathbf{y}_j^i = \left(x_{j+1}^i{}_{\cdot} - x_j^i\right)\,\mathbf{B}_j\,\mathbf{e} = \left(x_{j+1}^i{}_{\cdot} - x_j^i\right)\,\left[b_{\Sigma,j}^1, \ldots, b_{\Sigma,j}^s\right]^T. \tag{6.2.35}$$

Taking only the last stage into account and recalling (6.2.30), we obtain the well-posedness condition as

$$\sum_{l=1}^{s} b_{sl} > 0. \tag{6.2.36}$$

However, the condition is still related to the coefficients of the unknown interpolation matrix $\mathbf{B}_j$. Obtaining b_{sl}, $l = 1, \ldots, s$, present in (6.2.36), from the interpolation method we have the following result.

THEOREM 6.2.5. *Let $x_j^i\,\mathbf{e}$, $x_{j+1}^i\,\mathbf{e} \in \hat{\Gamma}_M$ and let the employed IRK method satisfies the property of the topology preservation in space (i.e. $D\mathbf{f} > 0$). Further let the matrix $\mathbf{F}_M$ be defined element-wise by*

$$f_{lK} := \Delta x\,\left[\delta_{l,K} - h\,a_{lK}\,\frac{\Delta u\left(y_M^K\right)}{\Delta x}\right],\quad l, k = 1, \ldots, s, \tag{6.2.37}$$

where δ represents the Kronecker delta and $\Delta u\left(y_M^K\right) := u\left(y_M^K + \Delta x\right) - u\left(y_M^K\right)$. Then the flow method is well-posed if

$$\sum_{l=1}^{s} F_{M,l,s} > 0. \tag{6.2.38}$$

where $F_{M,l,s}$ is the $(s-1)$-dimensional minor which corresponds to the element f_{ls} of the matrix $\mathbf{F}_M$.

PROOF. The elements of $\mathbf{B}_j$ are determined by the interpolation, i.e. by requiring $\mathbf{y}_K = \mathbf{p}\left(\mathbf{f}_K\right)$, $K = M, \ldots, M+s$ and the fact that $x_j^i\,\mathbf{e} \in \hat{\Gamma}_M$. Hence we can obtain $\mathbf{B}_j$ from the linear system

$$\mathbf{B}_j\,\mathbf{f}_K + \mathbf{d}_j = \mathbf{y}_K,\quad K = M, \ldots, M+s, \tag{6.2.39}$$

which can be reformulated as

$$\mathbf{B}_j\ (\mathbf{f}_K - \mathbf{f}_M) = \mathbf{y}_K - \mathbf{y}_M, \quad K = M, \ldots, M+s. \tag{6.2.40}$$

By writing the complete matrix form of (6.2.40), we have

$$\mathbf{B}_j\, \mathbf{F}_M = \mathbf{Y}_K, \tag{6.2.41}$$

where the matrices $\mathbf{F}_M$ and $\mathbf{Y}_M$ are defined as

$$\begin{aligned} \mathbf{F}_M &:= [\mathbf{f}_{M+1} - \mathbf{f}_M, \ldots, \mathbf{f}_{M+s} - \mathbf{f}_M]\,, \\ \mathbf{Y}_M &:= [\mathbf{y}_{M+1} - \mathbf{y}_M, \ldots, \mathbf{y}_{M+s} - \mathbf{y}_M]\,. \end{aligned} \tag{6.2.42}$$

The matrix $\mathbf{B}_j$ now follows from (6.2.41), i.e.

$$\mathbf{B}_j = \mathbf{Y}_K\, \mathbf{F}_M^{-1}. \tag{6.2.43}$$

By defining vectors, say $\mathbf{b}_j^s$ and $\mathbf{y}_M^s$, which contain elements of the $s-$th row of $\mathbf{B}_j$ and $\mathbf{Y}_M$ respectively, i.e.

$$\begin{aligned} \mathbf{b}_j^s &:= [b_{s1}, \ldots, b_{ss}]^T\,, \\ \mathbf{y}_M^s &:= \left[y_{M+1}^s - y_M^s, \ldots, y_{M+s}^s - y_M^s\right], \end{aligned} \tag{6.2.44}$$

we obtain

$$\mathbf{b}_j^s = \mathbf{F}_M^{-T}\, \mathbf{y}_M^s. \tag{6.2.45}$$

Obviously the right-hand side of the condition (6.2.36) is actually the sum of elements of $\mathbf{b}_j^s$. Since $\mathbf{y}_M, \ldots, \mathbf{y}_{M+s} \in \{\mathbf{y}_K\}_{K=0}^{s(n+1)}$, defined by (6.1.14), and assuming, without loss of generality, the following ordering of points

$$\begin{aligned} \mathbf{y}_{M+K} &:= \mathbf{y}_M + \Delta x\, \mathbf{e}_K, \quad K = 1, \ldots, s, \\ \mathbf{e}_K &:= [0, \ldots, 0, 1, 0, \ldots 0]^T\,, \end{aligned} \tag{6.2.46}$$

we have

$$\mathbf{y}_M^s = [0, \ldots, 0, \Delta x]^T\,. \tag{6.2.47}$$

Defining

$$\mathbf{C} = \mathbf{F}_M^{-T}, \tag{6.2.48}$$

and employing (6.2.47), we obtain

$$\mathbf{b}_j^s = \Delta x\ [c_{1s}, \ldots, c_{ss}]^T\,. \tag{6.2.49}$$

Hence the condition (6.2.36) is equivalent to

$$\sum_{l=1}^{s} c_{ls} > 0, \tag{6.2.50}$$

which is actually

$$\frac{1}{\det \mathbf{F}_M} \sum_{l=1}^{s} F_{M,l,s} > 0. \tag{6.2.51}$$

According to (6.2.42), (6.2.46) and (6.1.15), we may express the element of $\mathbf{F}_M$ by the following

$$\begin{aligned} f_{lK} \coloneqq f^l_{M+K} - f^l_M &= y^l_{M+K} - y^l_M - h \sum_{q=1}^{s} a_{lq} \left[u\left(y^q_{M+K}\right) - u\left(y^q_M\right) \right] \\ &= \Delta x \left[\delta_{K,l} - h\, a_{lK} \frac{u\left(y^K_M + \Delta x\right) - u\left(y^K_M\right)}{\Delta x} \right] \\ &= \Delta x \left[\delta_{K,l} - h\, a_{lK} \frac{\Delta u\left(y^K_M\right)}{\Delta x} \right]. \end{aligned} \tag{6.2.52}$$

Clearly the matrix $\mathbf{F}_M$ represents the first order approximation of $D\mathbf{f}$, i.e. we may write

$$\det \mathbf{F}_M \doteq \Delta x^s \ \det D\mathbf{f}. \tag{6.2.53}$$

Recalling results from the previous subsection, where it was shown that for IRK methods of interest $\det D\mathbf{f} > 0$ holds, i.e. that the topology in space is preserved, we find $\det \mathbf{F}_M$ to be positive as well. This means that the well-posedness condition follows from (6.2.38). □

Due to the complexity of the matrix $\mathbf{F}_M$, in the analysis that follows we restrict ourselves to two- and three-stage IRK methods.

COROLLARY 6.2.6. *For the flow method that employs a two-stage IRK method, which satisfy the property of the topology preserving in space, the well-posedness condition reads*

$$1 + h\,(a_{21} - a_{11})\, \frac{\Delta u\left(y^1_M\right)}{\Delta x} > 0. \tag{6.2.54}$$

Additionally if $u'(x) < 0$ *on the whole domain of interest, the well-posedness condition reads*

$$a_{21} - a_{11} \leq 0. \tag{6.2.55}$$

Unfortunately (6.2.55) does not hold for IRK methods in general. For the linear test problem (6.2.1), the well-posedness conditions for several IRK methods are shown in Table 6.2.2. One should note that some methods introduce severe constraints for the time step (assuming of course that $|\lambda|$ is large). On the other hand, some methods do not have any constraints, which makes them appropriate candidates for implementation into the flow method.

Method	Gauss	Rad. IA	Rad. IIA	Lob. IIIA	Lob. IIIB	Lob. IIIC
$h\,\lvert\lambda\rvert$	$h\,\lvert\lambda\rvert < 3.46$	-	$h\,\lvert\lambda\rvert < 3$	$h\,\lvert\lambda\rvert < 2$	-	-

TABLE 6.2.2. Time-step constraints for different two-stage IRK methods

If one repeats the same procedure the first inner-stage, then the condition similar to (6.2.55) reads

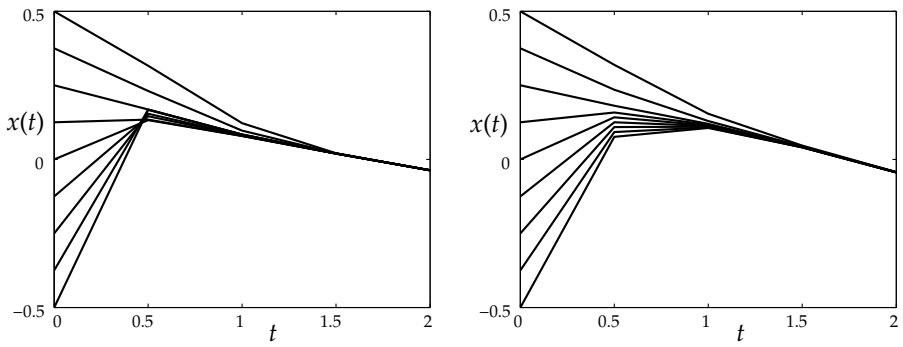
$$a_{12} - a_{22} \leq 0, \tag{6.2.56}$$

which holds for all methods introduced in Table 6.2.2.

EXAMPLE 6.2.7. Consider

$$\begin{cases} \dot{x} = -\arctan(10\,x) + \cos t, \\ x_0 \in I^0 = [-0.5, 0.5]. \end{cases} \tag{6.2.57}$$

The solution of this problem can be obtained by discretising I^0 and applying the flow method that employs different two-stage IRK methods. In Figure 6.2.1 results obtained by employing Radau IIA and Lobatto IIIC methods are shown. One should note that intersections occur if the IRK method is Radau IIA and the time-step is relatively large. This is, of course, due to the well-posedness constraint of Radau IIA method, as shown in Table 6.2.2. On the other hand, a similar problem is not present if the method of choice is Lobatto IIIC, regardless of the size of the time-step. □



(a) Radau IIA (b) Lobatto IIIC

FIGURE 6.2.1. Flow in time ($h = 0.5$)

COROLLARY 6.2.8. *The well-posedness condition of the flow method which employs a three-stage IRK method reads*

$$\begin{aligned} 1 + h &\left[(a_{13} + a_{23}) \frac{\Delta u\left(y_M^3\right)}{\Delta x} - a_{11} \frac{\Delta u\left(y_M^1\right)}{\Delta x} - a_{22} \frac{\Delta u\left(y_M^2\right)}{\Delta x} \right] \\ &+ h^2 \left[(a_{11}a_{22} - a_{12}a_{21}) \frac{\Delta u\left(y_M^1\right)}{\Delta x} \frac{\Delta u\left(y_M^2\right)}{\Delta x} \right. \\ &+ (a_{21}a_{13} - a_{11}a_{23}) \frac{\Delta u\left(y_M^1\right)}{\Delta x} \frac{\Delta u\left(y_M^3\right)}{\Delta x} \\ &\left. + (a_{12}a_{23} - a_{22}a_{13}) \frac{\Delta u\left(y_M^2\right)}{\Delta x} \frac{\Delta u\left(y_M^3\right)}{\Delta x} \right] > 0. \end{aligned} \tag{6.2.58}$$

Consider again the linear test problem. We can define a quadratic polynomial, say $q(\mu)$, by

$$\begin{aligned} q(\mu) := &(a_{11}a_{22} + a_{21}a_{13} + a_{12}a_{23} - a_{12}a_{21} - a_{11}a_{23} - a_{22}a_{13})\,\mu^2 \\ &+ (a_{13} + a_{23} - a_{11} - a_{22})\,\mu + 1, \end{aligned} \tag{6.2.59}$$

where $\mu := h\lambda$. Now we can analyse the sign of q for various IRK methods. Clearly if

$$q(\mu) > 0, \quad \mu \in \mathbb{R}, \tag{6.2.60}$$

the well-posedness property does not introduce constraints on time-step. Computing coefficients of q for various IRK methods, one should note that all IRK methods introduced in Table 6.2.2 satisfy (6.2.60) except Lobatto IIIA. For this method the polynomial reads

$$q(\mu) = -\frac{3}{8}\,\mu + 1, \tag{6.2.61}$$

and we have a time-step constraint

$$h\,|\lambda| < \frac{8}{3}. \tag{6.2.62}$$

This leads to the conclusion that most of the three-stage IRK methods of interest are appropriate candidates (concerning the well-posedness property) to be employed into the flow method.

The analysis of the well-posedness property is given under the assumption that the points $x_j^i\,\mathbf{e}$ and $x_{j+1}^i\,\mathbf{e}$ belong to the same simplex of the new grid. In a more general case, we should consider the linear test problem, for which $\mathbf{B}_j$ and $\mathbf{d}_j$ (respectively $\mathbf{B}_{j+1}$ and $\mathbf{d}_{j+1}$) are constant matrices for all simplices of the new grid, i.e.

$$\begin{aligned} \mathbf{B}_j &= \mathbf{B}_{j+1} = \mathbf{B}, \\ \mathbf{d}_j &= \mathbf{d}_{j+1} = \mathbf{d}, \end{aligned} \tag{6.2.63}$$

where $\mathbf{B}$ and $\mathbf{d}$ does not depend on $\hat{\Gamma}_M$. This means that the analysis is similar and the conditions (6.2.55) and (6.2.58) are the same.

6.3 Error analysis

In the previous two chapters, it was shown that the error of the flow method consists of two components: the local discretisation error and the interpolation error. The local discretisation error is related to the method employed, here the IRK method, and the interpolation error depends on the choice of the interpolation method involved. The easiest way to show this is by studying the behaviour of the local error, say δ, for the scalar case. Assuming $x_j\left(t^i\right) = x_j^i$, the local error of the flow method at time-level t^{i+1} is defined by

$$\delta\left(x_j\left(t^{i+1}\right), h\right) := \frac{1}{h}\left|x_j\left(t^{i+1}\right) - x_j^{i+1}\right|, \tag{6.3.1}$$

where $x_j\left(t^{i+1}\right)$ is the exact solution of the flow point $x_j(t)$ at t^{i+1} and x_j^{i+1} is the numerical solution obtained by the flow method. Denoting $\tilde{x}_j^{i+1}$ to be the IRK numerical solution obtained exactly (i.e. without any additional error), it is clear that the local error satisfies

$$\begin{aligned} \delta(\cdot, h) &= \frac{1}{h}\left|x_j\left(t^{i+1}\right) - \tilde{x}_j^{i+1} + \tilde{x}_j^{i+1} - x_j^{i+1}\right| \\ &\le \frac{1}{h}\left|x_j\left(t^{i+1}\right) - \tilde{x}_j^{i+1}\right| + \frac{1}{h}\left|\tilde{x}_j^{i+1} - x_j^{i+1}\right| \\ &=: d_h(\cdot) + r_h(\cdot). \end{aligned} \tag{6.3.2}$$

Clearly $d_h(\cdot)$ represents the local discretisation error and $r_h(\cdot)$ the interpolation error. As known from the literature (see e.g. [33]), the accuracy of the IRK method is related to the consistency order, say p, for which the following holds

$$d_h\left(x_j\left(t^{i+1}\right)\right) = O\left(h^p\right), \tag{6.3.3}$$

for p as large as possible. The value of p depends on the IRK method employed. For example, IRK methods of interest satisfy

$$p = \begin{cases} 2s, & \text{Gauss,} \\ 2s-1, & \text{Radau,} \\ 2s-2, & \text{Lobatto.} \end{cases} \tag{6.3.4}$$

The more interesting part is the interpolation error. Since the problem here is vectorial and multivariate, we need to use error bounds for vectorial functions obtained in 3.4. For simplicity we will restrict ourselves to piece-wise linear interpolation. From Theorem 3.4.1, (6.1.9) and (6.1.10), we have the following

$$\|r_h(\cdot)\|_{L_\infty} \leq \frac{1}{2h}\,\frac{s}{s+1}\,\Delta f_M^2\,\left\|D^2\mathbf{g}\right\|_{\infty,\hat{\Gamma}_M}, \tag{6.3.5}$$

where Δf_M is the diameter of the new grid simplex $\hat{\Gamma}_M$, which is defined by the set $\hat{S}_M = \{\mathbf{f}_K\}_{K=M}^{M+s}$. The norm of $D^2\mathbf{g}$, present on the right-hand side, is defined by

$$\left\|D^2\mathbf{g}\right\|_{\infty,\hat{\Gamma}_M} := \max_l \sup_{\mathbf{f}\in\hat{\Gamma}_M} \sup_{\substack{\mathbf{z}\in\mathbb{R}^s \\ \|\mathbf{z}\|=1}} \left|D_{\mathbf{z}}^2 g_l(\mathbf{f})\right|, \tag{6.3.6}$$

where $D_{\mathbf{z}}g$ is the directional derivative of g in the direction $\mathbf{z}$. Let us now relate derivatives of $\mathbf{g}$ to derivatives of $\mathbf{f}$, i.e. $\mathbf{u}$, as well as the diameter Δf_M to the corresponding original grid diameter, say Δy_M. From

$$\Delta f_M = \max_{K,J} \|\mathbf{f}_{M+K} - \mathbf{f}_{M+J}\| := \|\mathbf{f}_{M1} - \mathbf{f}_{M2}\|, \tag{6.3.7}$$

where $\mathbf{f}_{M1}, \mathbf{f}_{M2} \in \hat{S}_M$, using (6.1.15) and the mean value theorem, we can find

$$\begin{aligned} \Delta f_M &= \|\mathbf{y}_{M1} - \mathbf{y}_{M2} - h\,\mathbf{A}\,(\mathbf{u}(\mathbf{y}_{M1}) - \mathbf{u}(\mathbf{y}_{M2}))\| \\ &= \|[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}(\tilde{\mathbf{y}}_M)]\,[\mathbf{y}_{M1} - \mathbf{y}_{M2}]\| \\ &\leq \|\mathbf{y}_{M1} - \mathbf{y}_{M2}\|\,\|\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}(\tilde{\mathbf{y}}_M)\| \\ &= \Delta y_M\,\|\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}(\tilde{\mathbf{y}}_M)\|. \end{aligned} \tag{6.3.8}$$

The point $\tilde{\mathbf{y}}_M$ is on the line between points $\mathbf{y}_{M1}$ and $\mathbf{y}_{M2}$, which means that $\tilde{\mathbf{y}}_M \in \Gamma_M$. Assuming that the set of points $\{x_k\}_{k=0}^n$ (for which $u-$values are known) is equidistant, it is clear that Δy_M (for arbitrary s) is

$$\Delta y_M = \sqrt{2}\Delta x, \tag{6.3.9}$$

where Δx is the spatial-step size present in (6.1.2). Substituting (6.3.8) and (6.3.9) into (6.3.5), we see that the interpolation error satisfies

$$\|r_h(\cdot)\|_{L_\infty} \leq \frac{1}{h}\,\frac{s}{s+1}\,\Delta x^2\,\|\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}(\tilde{\mathbf{y}}_M)\|^2\,\left\|D^2\mathbf{g}\right\|_{\infty,\hat{\Gamma}_M}. \tag{6.3.10}$$

As it turns out in Chapter 3 the norm of the second derivative of $\mathbf{g}$, defined by (6.3.6), can be related to the corresponding norm that introduces the derivatives of $\mathbf{u}$ by the

following. Since $\mathbf{g} = \mathbf{f}^{-1}$, we clearly have

$$D\mathbf{g} = [D\mathbf{f}]^{-1},$$

$$D^2\mathbf{g} = -[D\mathbf{f}]^{-1}\, D^2\mathbf{f}\, [D\mathbf{f}]^{-2}, \tag{6.3.11}$$

and since $D\mathbf{f} = \mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}$ and $D^2\mathbf{f} = -h\,\mathbf{A}\,D^2\mathbf{u}$, we find

$$D^2\mathbf{g} = h\,[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}]^{-1}\,\mathbf{A}\,D^2\mathbf{u}\,[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}]^{-2}. \tag{6.3.12}$$

Substituting this into (6.3.10), we finally obtain the interpolation bound

(6.3.13)

$$\|r_h(\cdot)\|_{L_\infty} \le \frac{s\Delta x^2}{s+1}\,\|\mathbf{I}_s - h\mathbf{A}D\mathbf{u}\,(\tilde{\mathbf{y}}_M)\|^2 \left\|[\mathbf{I}_s - h\mathbf{A}D\mathbf{u}]^{-1}\mathbf{A}D^2\mathbf{u}\,[\mathbf{I}_s - h\mathbf{A}D\mathbf{u}]^{-2}\right\|_{\infty,\Gamma_M}.$$

Note that the matrix $[\mathbf{I}_s - h\mathbf{A}D\mathbf{u}]$ is present only by its inverse, implying that the $\infty-$norm on the right hand side of (6.3.13) is not necessarily large even if the norm $\|\mathbf{I}_s - h\mathbf{A}D\mathbf{u}\|_{\infty,\Gamma_M}$ is very large. For methods which satisfy $c_s = 1$, the norm defined by (6.3.6) for the vectorial function should be replaced by

$$\left\|D^2 g_s\right\|_{\infty,\hat{\Gamma}_M} := \sup_{\mathbf{f}\in\hat{\Gamma}_M}\ \sup_{\substack{\mathbf{z}\in\mathbb{R}^s\\ \|\mathbf{z}\|=1}} \left|D^2_{\mathbf{z}} g_s(\mathbf{f})\right|, \tag{6.3.14}$$

where $D^2 g_s$ follows from (6.3.12).

From (6.3.2), (6.3.3) and (6.3.13) one should note that the local error of the flow method is $O(h^p) + O(\Delta x^2)$. The interpolation error can become dominant if the IRK method has many stages, especially since the order p is twice the number of stages, see (6.3.4). To avoid this, the spatial step-size Δx needs to be chosen very small, e.g. $O(h^s)$ for the Gauss method or $O(h^{s-1})$ for Lobatto methods. The other option, of course, is to use different interpolation methods which are more accurate, but also computationally more expensive.

Although obtaining the numerical solution by employing methods for which $c_s \neq 1$ is much more expensive, these methods can give more accurate results. This follows from the fact that the solution at the next time-level is now obtained by applying (6.1.3), i.e. its modification where unknown $u\left(y_j^{il}\right)$ values (related to the flow point x_j) are approximated by $v\left(y_j^{il}\right)$ say, obtained by interpolating points $\{(x_k, u_k)\}_{k=0}^n$. Denoting

$$y_j^{il} := \tilde{y}_j^{il} + r_j^l, \quad l = 1, \ldots, s, \tag{6.3.15}$$

where $\tilde{y}_j^{il}$, $l = 1, \ldots, s$ represent exact solutions of the nonlinear system (6.1.7) and r_j^l the interpolation error related to lth inner-stage. Substituting this into (6.1.3), the solution at the next time-level follows from

$$x_j^{i+1} = x_j^i + h\sum_{l=1}^{s} b_l\, v\left(y_j^{il}\right). \tag{6.3.16}$$

Since $v\left(y_j^{il}\right)$ is the approximation of $u\left(y_j^{il}\right)$, we may write

$$v\left(y_j^{il}\right) = u\left(y_j^{il}\right) + \rho_j^l, \tag{6.3.17}$$

where ρ_j^l represents the error of the univariate direct interpolation. Substituting (6.3.17) into (6.3.16) we have

$$x_j^{i+1} = x_j^i + h\sum_{l=1}^{s} b_l\, u\left(y_j^{il}\right) + h\sum_{l=1}^{s} b_l\,\rho_j^l. \tag{6.3.18}$$

Assuming that interpolation errors r_j^l are sufficiently small, we can use the approximation

$$u\left(y_j^{il}\right) = u\left(\tilde{y}_j^{il} + r_j^l\right) \doteq u\left(\tilde{y}_j^{il}\right) + r_j^l\, u'\left(\tilde{y}_j^{il}\right). \tag{6.3.19}$$

Applying this into (6.3.18), we have

$$x_j^{i+1} = x_j^i + h \sum_{l=1}^{s} b_l\, u\left(\tilde{y}_j^{il}\right) + h \sum_{l=1}^{s} b_l\, r_j^l\, u\left(\tilde{y}\right) + h \sum_{l=1}^{s} b_l\, \rho_j^l. \tag{6.3.20}$$

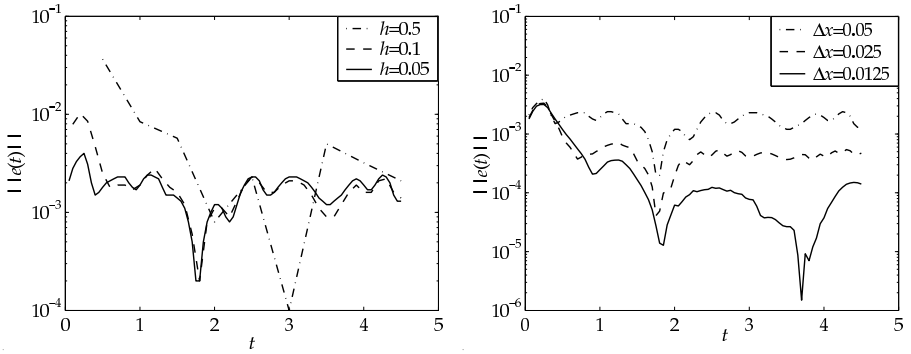
Obviously the local error, which can be obtained from (6.3.20), consists of three parts. Of course, the first one is the local discretisation error of the employed IRK method, the second one is related to the interpolation error of the inverse (multivariate) interpolation, and the third one is related to the interpolation errors of the direct (univariate) interpolation. Now, let us assume that the inverse interpolation is linear and that ρ_j^l is $O\left(\Delta x^q\right)$, where q represents the order of the direct interpolation involved. Then from (6.3.1), (6.3.3) and (6.3.13) and assuming that the solution is sufficiently smooth, we have

$$\delta(\cdot, h) \leq O\left(h^p\right) + O\left(h\right)\, O\left(\Delta x^2\right) + O\left(\Delta x^q\right). \tag{6.3.21}$$

Hence, if direct interpolation does not represent the dominant error source, it is clear that the inverse interpolation error is reduced by the order of h, which can improve the overall accuracy of the flow method. This is illustrated by the following example.

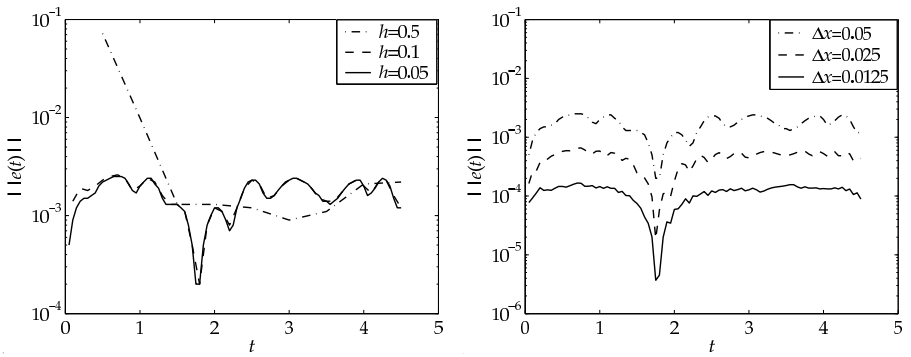
EXAMPLE 6.3.1. Consider Example 6.2.7 once again. Let us discretise the initial flow $I^0 = [0.5, 0.5]$ by $\mu = 9$ points and compute numerical solutions for the time interval $t = [0, 4.5]$. Since in this particular example we explicitly know $u(x)$ we can choose the spatial step-size freely and assess its influence (coming via interpolation error) on the accuracy. Of course, the accuracy is also related to the time step-size and, thus, we perform series of tests for various values of h. Finally, since the choice of the employed IRK method is also of importance, either because of the fact that they are of different orders or the implementation is different (see above), we employ three different methods, namely Lobatto IIIC, Radau IIA and the Gauss method. To complete the flow method settings, we choose piece-wise linear inverse interpolation. For the first set of experiments we fix the number of discretisation points $\{x_k\}_{k=0}^n$, for which we compute $\{u_k\}_{k=0}^n$, to be $n = 21$ (which gives $\Delta x = 0.05$). Then we vary the value of the time-step, say $h = 0.5, 0.1, 0.05$, for all aforementioned methods and compute the ∞-norm of the global error over all flow points solutions in time, see Figures 6.3.1a, 6.3.2a and 6.3.3a. For the Lobatto IIIC and Radau IIA method (which satisfy $c_s = 1$) one should note that decreasing the time step does not improve the accuracy much since the interpolation error is dominant. On the other hand, the implementation of the Gauss method introduces the error which satisfies (6.3.21), i.e. the error is decreasing by the factor of h, see Figure 6.3.3a. To avoid the influence of the error coming from direct interpolation evaluations, i.e. the last part of the right-hand side of (6.3.21), we use univariate spline interpolation, which is $O\left(\Delta x^4\right)$.

In the second experiment we fix the time step-size, say $h = 0.05$, and vary the spatial step-size, say $\Delta x = 0.05, 0.025, 0.0125$ and again compute the ∞-norm of the global error over all flow points solutions in time. The results for different (employed) IRK methods are shown in Figures 6.3.1b, 6.3.2b and 6.3.3b. One should note that the ratio between errors is approximately $\frac{1}{4}$, which is clearly coming from the fact that the interpolation error is $O\left(\Delta x^2\right)$.



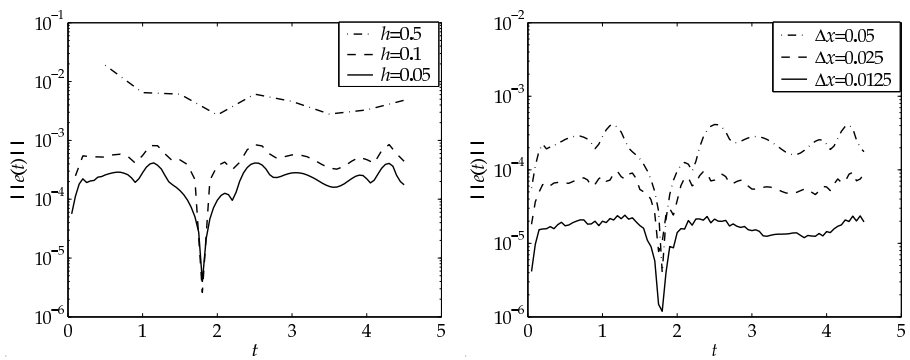
(a) $\Delta x = 0.05$ (b) $h = 0.05$

FIGURE 6.3.1. Global error of the flow method which employs the Lobatto IIIC method.



(a) $\Delta x = 0.05$ (b) $h = 0.05$

FIGURE 6.3.2. Global error of the flow method which employs the Radau IIA method.



(a) $\Delta x = 0.05$ (b) $h = 0.05$

FIGURE 6.3.3. Global error of the flow method which employs the Gauss method.

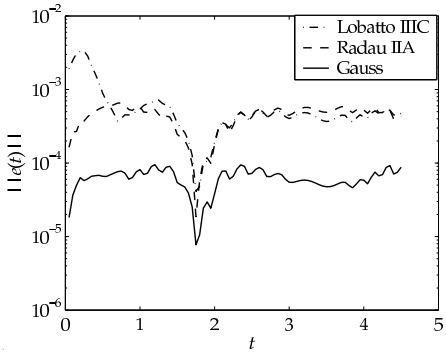


FIGURE 6.3.4. Global error of different (employed) IRK methods with $h = 0.05$ and $\Delta x = 0.025$.

Finally, we can compare results obtained by different methods by using the same time and spatial step-sizes, say $h = 0.05$ and $\Delta x = 0.025$, see Figure 6.3.4. Since it was already shown that the interpolation error is dominant, it is clear that the errors of the employed Lobatto IIIC and Radau IIA are of the same order and determined by the value of the interpolation error. Hence, by recalling (6.3.4), one should note that the larger accuracy order of the Radau IIA method is lost. Of course, employing the IRK method of higher order, especially in choosing the number of stages, does not improve the overall accuracy if the interpolation error is not sufficiently small. On the other hand, the accuracy may be improved (by the factor of h) by employing (more expensive) methods, e.g. the Gauss method, which require additional direct interpolation, which should be, of course, accurate enough. In Figure 6.3.4 it is clearly shown that the error of the employed Gauss method is h times smaller than the error of the other two methods.

6.4 Stability

As already mentioned, the main reason for using IRK methods is based on their favourable stability properties. The flow method should then, of course, inherit these properties, such as A, $A(\alpha)$, L-stability. To check this, we use the stability function (see Section 2.4). However, since linear interpolation is exact for a linear problem, it is clear that the flow method has the same properties as the employed IRK method. For example, the flow method is A-stable if and only if the employed IRK method is A-stable. Of course, the same also holds for $A(\alpha)$ and L-stable methods.

The stability analysis of the flow method of the general nonlinear problem, as shown below, leads to similar conclusions, i.e. that the B-stability property is closely related to one of the employed IRK method. This can be shown by studying first variations. Although we are mainly interested in the behaviour of the solution at time grid points t^i, $i = 1, 2, \ldots$, it will be shown that the behaviour of the inner-stage solutions, obtained by the flow method, is also determined by the behaviour of the inner-stage solutions of the standard IRK method. Let us start from the inner-stage solutions of IRK for a particular flow point, say x_j, which follows from (6.1.9), i.e.

$$\mathbf{y}_j^i = \mathbf{g}\left(x_j^i\, \mathbf{e}\right). \tag{6.4.1}$$

Let $\mathbf{z}_j^i\big|_{j\,fixed} := z_j^i\,\mathbf{e}$ denote a small perturbation of $\mathbf{x}_j^i := x_j^i\,\mathbf{e}$, such that $\mathbf{x}_j^i + \mathbf{z}_j^i\big|_{j\,fixed}$ also satisfies (6.4.1) to first order. Then we have

$$\mathbf{y}_j^i + \Delta\mathbf{y}_j^i = \mathbf{g}\left(\left(x_j^i + z_j^i\right)\,\mathbf{e}\right) \doteq \mathbf{g}\left(x_j^i\,\mathbf{e}\right) + z_j^i\,D\mathbf{g}\left(x_j^i\,\mathbf{e}\right)\,\mathbf{e}, \tag{6.4.2}$$

where $\Delta\mathbf{y}_j^i$ represents the perturbation of the inner-stage solutions. Hence from (6.4.1) and (6.4.2) it follows

$$\Delta\mathbf{y}_j^i = z_j^i\,D\mathbf{g}\left(x_j^i\,\mathbf{e}\right)\,\mathbf{e}. \tag{6.4.3}$$

Assuming $c_s = 1$ and observing the last stage only, we have

$$z_j^{i+1} = \Delta y_j^{is} = \left\langle \nabla g_s\left(x_j^i\,\mathbf{e}\right), \mathbf{e}\right\rangle z_j^i = \left(\frac{\partial g_s}{\partial y^1}\left(x_j^i\,\mathbf{e}\right) + \cdots + \frac{\partial g_s}{\partial y^s}\left(x_j^i\,\mathbf{e}\right)\right) z_j^i. \tag{6.4.4}$$

Now, by employing the contractivity condition, which assures the stability of the method, we obtain the following

$$\left|\left\langle \nabla g_s\left(x_j^i\,\mathbf{e}\right), \mathbf{e}\right\rangle\right| < 1, \tag{6.4.5}$$

which we call the stability condition of the IRK method. This condition is related with derivatives of the unknown inverse function and, thus, needs to be expressed in terms of $\mathbf{f}$, i.e. $\mathbf{u}$. Since

$$D\mathbf{g}\left(x_j^i\,\mathbf{e}\right) = D\mathbf{f}^{-1}\left(\mathbf{y}_j^i\right), \tag{6.4.6}$$

and recalling (6.1.8), we have

$$D\mathbf{g}\left(x_j^i\,\mathbf{e}\right) = \left[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}\left(\mathbf{y}_j^i\right)\right]^{-1}, \tag{6.4.7}$$

from which we can obtain the condition (6.4.5) related to the derivative of u and the coefficients of the matrix $\mathbf{A}$.

The inner-stage solutions of the flow method with piece-wise linear interpolation follows from (6.2.31), and clearly the following holds

$$D\mathbf{p}\left(x_j^i\,\mathbf{e}\right) = \mathbf{B}_j, \tag{6.4.8}$$

where $\mathbf{B}_j$ is the interpolation matrix, which was introduced in the Subsection 6.2.2. Now, we can show that the matrix $D\mathbf{p}$ is closely related to $D\mathbf{g}$, i.e. that the stability condition of the flow method is approximately the same as (6.4.5). Recalling (6.2.43), we clearly have

$$\mathbf{B}_j^{-1} = \mathbf{F}_M\,\mathbf{Y}_K^{-1}. \tag{6.4.9}$$

However, according to (6.2.42) and (6.2.46), we have

$$\mathbf{Y}_K = \Delta x\,\mathbf{I}_s, \tag{6.4.10}$$

which leads to

$$\mathbf{B}_j^{-1} = \frac{1}{\Delta x}\,\mathbf{F}_M. \tag{6.4.11}$$

The elements of $\mathbf{F}_M$ are defined by (6.2.52). Assuming that Δx is sufficiently small, so that

$$\frac{u\left(y_M^K + \Delta x\right) - u\left(y_M^K\right)}{\Delta x} \doteq u'\left(y_M^K\right), \tag{6.4.12}$$

we may write

$$\mathbf{F}_M = \Delta x\,\left[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}\left(\mathbf{y}_M\right)\right]. \tag{6.4.13}$$

Substituting this into (6.4.11) and recalling (6.4.8), we have

$$D\mathbf{p}\left(x_j^i\,\mathbf{e}\right) = \left[\mathbf{I}_s - h\,\mathbf{A}\,D\mathbf{u}\left(\mathbf{y}_M\right)\right]^{-1}. \tag{6.4.14}$$

Comparing (6.4.7) and (6.4.14) one should note that the expressions are approximately the same and that the only difference is coming from the value of $D\mathbf{u}$. However, since points $\mathbf{y}_j^i$ and $\mathbf{y}_M$ belong to the same simplex Γ_M of the original grid, this difference is small. Hence we may conclude that the behaviour of the inner-stage solutions of the flow method are closely related to ones of the standard IRK methods.

Following the same approach as above, one can obtain the contractivity condition for the flow method as

$$\left|\left\langle \nabla p_s\left(x_j^i\,\mathbf{e}\right),\mathbf{e}\right\rangle\right| < 1. \tag{6.4.15}$$

From (6.2.31), we have

$$\left\langle \nabla p_s\left(x_j^i\,\mathbf{e}\right),\mathbf{e}\right\rangle = \sum_{l=1}^{s} b_{sl}. \tag{6.4.16}$$

Since

$$\sum_{l=1}^{s} b_{sl} = \frac{\Delta x}{\det \mathbf{F}_M}\sum_{l=1}^{s} F_{M,l,s}, \tag{6.4.17}$$

where $F_{M,l,s}$ are, as before, the minors which correspond to elements f_{ls} of the matrix $\mathbf{F}_M$. Hence the contractivity condition of the flow method is given by

$$\left|\frac{\Delta x}{\det \mathbf{F}_M}\sum_{l=1}^{s} F_{ls}\right| < 1. \tag{6.4.18}$$

For the flow method which employs a two-stage method, the condition (6.4.18) reads

$$\left|\frac{1+h\left(a_{21}-a_{11}\right)\frac{\Delta u\left(y_M^1\right)}{\Delta x}}{1-h\left(a_{11}\frac{\Delta u\left(y_M^1\right)}{\Delta x}+a_{22}\frac{\Delta u\left(y_M^2\right)}{\Delta x}\right)+h^2\left(a_{11}a_{22}-a_{12}a_{21}\right)\frac{\Delta u\left(y_M^1\right)}{\Delta x}\frac{\Delta u\left(y_M^2\right)}{\Delta x}}\right| < 1. \tag{6.4.19}$$

Clearly the expression on the left-hand side tends to zero if the problem is stiff, i.e. if $h\,u' \ll -1$. This means that there are no time-step constraints concerning stability. For some IRK methods, e.g. Radau IA, Lobatto IIIB and Lobatto IIIC, we even have $a_{21} = a_{11}$, which gives

$$\left|\frac{1}{1-h\left(a_{11}\frac{\Delta u\left(y_M^1\right)}{\Delta x}+a_{22}\frac{\Delta u\left(y_M^2\right)}{\Delta x}\right)+h^2\left(a_{11}a_{22}-a_{12}a_{21}\right)\frac{\Delta u\left(y_M^1\right)}{\Delta x}\frac{\Delta u\left(y_M^2\right)}{\Delta x}}\right| < 1, \tag{6.4.20}$$

and holds even stronger for the stiff case.

CHAPTER 7

The flow method for solving vectorial ODEs

In previous chapters we introduced the flow method based on various implicit numerical methods. However, the analysis was based on scalar problems. In this chapter we consider (general) vectorial problems. Even though the method principle remains the same, a further analysis is needed since multivariate inverse interpolation is more involved. Again we analyse the flow method that employs the Euler Backward method, just to keep the analysis simple. However, we also extend the analysis of our method to *the implicit midpoint rule (IMR)* method. Since this method is closely related to EB (as we will show later), the implementation of IMR into the flow method is straightforward.

In the following section we consider the method implementation, as well as interpolation issues of the method. The local error analysis is given in Section 7.2. The stability properties of the flow method are still closely related to the standard (employed) method, as shown in Section 7.3. Finally, we conclude this chapter with some numerical examples, which illustrate the quality of the method.

7.1 Method implementation

Here we consider the general flow problem, as defined in Section 2.2, i.e. the following

$$\begin{cases} \frac{dx}{dt} = \mathbf{u}(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^N, \\ \mathbf{x}(0) \in \mathbf{I}(\mathbf{x}, 0), \end{cases} \tag{7.1.1}$$

which is the vectorial analogue of the scalar flow problem from previous chapters. Here $\mathbf{x}$ is a point of the flow and $\mathbf{u}(\mathbf{x})$ is not given explicitly, although we assume that $\mathbf{u}(\mathbf{x})$ is Lipschitz continuous. To track the flow $\mathbf{I}(\mathbf{x}, t)$ numerically in time, one needs to properly discretise $\mathbf{I}(\mathbf{x}, t)$ in space. Hence, we denote by $\{\mathbf{x}_j(t)\}_{j=0}^{\mu} \subset \mathbf{I}(\mathbf{x}, t)$ a set of points which gives a numerical spatial representation of the flow. Let us now assume that at a particular time point, $\mathbf{u}(\mathbf{x})$ is obtained discretely (e.g. numerically or experimentally) at some spatial points, say $\{\mathbf{x}_k\}_{k=0}^{n} \in \Gamma$, i.e. it is given by the set of values $\{\mathbf{u}_k\}_{k=0}^{n}$. Here Γ represents a convex hull of the set $\{\mathbf{x}_k\}_{k=0}^{n}$. We distinguish

$\{\mathbf{x}_k\}_{k=0}^{n}$ from $\{\mathbf{x}_j\}_{j=0}^{\mu}$, since $\mathbf{u}(\mathbf{x})$ can also be known at some additional points. Of course, $\mathbf{I}(\mathbf{x}, t) \subseteq \Gamma$ should hold for all t to avoid lack of information about the velocity.

Again for solving (7.1.1), one needs a proper time discretisation. For stiff problems we need to use an implicit method. We illustrate our method employing the Euler Backward (EB) method (see Section 2.3). Even though the method principle remains the same, like e.g. we discussed in Chapter 4, a further analysis is needed since multivariate inverse interpolation is more involved. We also extend the analysis of our method to the IMR (Implicit Midpoint Rule) method. Since this method is closely related to EB (as we will show later), the implementation of IMR into the flow method is straightforward. Since IMR is a symplectic integrator, this allows us to employ our method in the applications where the volume preservation is an essential issue. The generalization of the flow method which employs other numerical methods can be done similarly as already shown for the scalar case.

Time discretisation of (7.1.1) by the EB method gives

$$\mathbf{x}_j^{i+1} = \mathbf{x}_j^i + h\,\mathbf{u}\left(\mathbf{x}_j^{i+1}\right). \tag{7.1.2}$$

If we apply IMR then we have

$$\mathbf{x}_j^{i+1} = \mathbf{x}_j^i + h\,\mathbf{u}\left(\frac{\mathbf{x}_j^{i+1} + \mathbf{x}_j^i}{2}\right), \tag{7.1.3}$$

which can be transformed into

$$\mathbf{x}_j^{i+\frac{1}{2}} = \mathbf{x}_j^i + \frac{h}{2}\,\mathbf{u}\left(\mathbf{x}_j^{i+\frac{1}{2}}\right), \tag{7.1.4}$$

$$\mathbf{x}_j^{i+1} = 2\,\mathbf{x}_j^{i+\frac{1}{2}} - \mathbf{x}_j^i. \tag{7.1.5}$$

Both (7.1.2) and (7.1.4) are systems of nonlinear equations which, in general, cannot be solved directly. One possible way to solve them is to employ some Newton type iterative method. However, this introduces some additional problems, as mentioned in Section 2.7. Hence we apply the flow method, as we did in the scalar case before.

Let us define

$$\hat{\mathbf{x}}_k^{i+1} := \mathbf{x}_k, \tag{7.1.6}$$

and since (7.1.1) is autonomous, we clearly have

$$\mathbf{u}\left(\hat{\mathbf{x}}_k^{i+1}\right) = \mathbf{u}\left(\mathbf{x}_k\right) = \mathbf{u}_k. \tag{7.1.7}$$

Taking $\hat{\mathbf{x}}_k^{i+1}$ now as the result of an EB step then this should correspond to a value $\hat{\mathbf{x}}_k^i$ defined by

$$\mathbf{f}_k = \hat{\mathbf{x}}_k^i := \hat{\mathbf{x}}_k^{i+1} - h\,\mathbf{u}\left(\hat{\mathbf{x}}_k^{i+1}\right) = \mathbf{x}_k - h\,\mathbf{u}_k. \tag{7.1.8}$$

In Figure 7.1.1 a 2-D example is shown where the points $\{\mathbf{x}_k\}_{k=0}^{n}$ are the vertices of a triangular grid. Clearly there is a functional dependence of points at two consecutive time-levels (which is, of course, analytically unknown). Nevertheless we can employ this fact for finding approximate values for all points in Γ and thus for the solution points. For the general point $\mathbf{x}^i \in \Gamma$ we can rewrite (7.1.8) as

$$\mathbf{x}^i = \mathbf{x}^{i+1} - h\,\mathbf{u}\left(\mathbf{x}^{i+1}\right) =: \mathbf{f}\left(\mathbf{x}^{i+1}\right). \tag{7.1.9}$$

If $\mathbf{f}(\mathbf{x})$ satisfies the conditions of the inverse function theorem (see Theorem 3.5.1), then there exists

$$\mathbf{g}(\mathbf{x}) := \mathbf{f}^{-1}(\mathbf{x}), \tag{7.1.10}$$

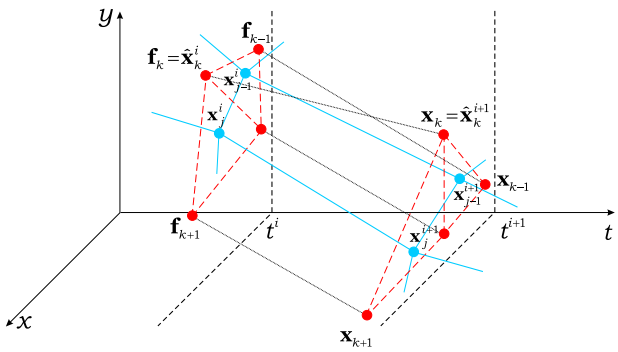


FIGURE 7.1.1. Flow method principle for 2-D (original) triangular grid.

and we may write

$$\mathbf{x}^{i+1} = \mathbf{g}\left(\mathbf{x}^i\right). \tag{7.1.11}$$

Since $\mathbf{g}$ is unknown in general, we can try to find an approximation, say $\mathbf{p}$, by requiring $\mathbf{p}\left(\hat{\mathbf{x}}_k^i\right) = \mathbf{g}\left(\hat{\mathbf{x}}_k^i\right)$. An obvious choice is to interpolate points $\{(\mathbf{f}_k, \mathbf{x}_k)\}_{k=0}^n$ in 2N-dimensional space. Now the solution at the next time-level follows from

$$\mathbf{x}_j^{i+1} = \mathbf{p}\left(\mathbf{x}_j^i\right). \tag{7.1.12}$$

The choice of the interpolation method involved can be seen as arbitrary, but there are some preferences which can help answering the question which method should be used. Firstly, the interpolation should preferably be local to avoid big costs in computations. Secondly, the additional (interpolation) error should be commensurate with the local discretisation error to preserve the accuracy of EB. Finally, the interpolation method should be applicable for irregular grids, since the new grid is obtained from the original (possibly regular) grid by the nonlinear mapping $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$, defined by (7.1.9).

If we use IMR, the method goes essentially similar. Indeed, it can be seen, from (7.1.4) and (7.1.5), that IMR is just an EB step on a half-interval followed by an (explicit) algebraic evaluation. This means that we can apply the flow method with a step-size $\frac{h}{2}$ and obtain the solution at the half-interval by

$$\mathbf{x}_j^{i+\frac{1}{2}} = \mathbf{p}(\mathbf{x}_j^i). \tag{7.1.13}$$

Now by substituting (7.1.13) in (7.1.5) we obtain the desired solution at the next time level.

The nonlinear system (7.1.9) is equivalent to a well-studied problem

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \tag{7.1.14}$$

which we already discussed in Section 3.5. The literature about this problem is rich (see e.g. [37]) and in the last decade a number of papers addressed the particular case arising from the use of implicit numerical methods for solving ODEs. Conditions for existence and uniqueness of the solution of (7.1.14) as well as a time step constraints for which these conditions are guaranteed are given in [54] and [17]. Hence we will assume that the solution exists and that it is unique. Our main interest here is to analyse the influence of the nonlinear mapping $\mathbf{f}$ (under the influence of $\mathbf{u}$) on our interpolation technique. Also we assume that $\mathbf{f}$ is a diffeomorphism.

Let us assume that Γ is discretised in space, i.e. the grid which covers Γ is defined by points $\{\mathbf{x}_k\}_{k=0}^n$ and adequate elements: For example, one can think of a triangular grid in 2-D. For our algorithm, the grid needs to be mapped onto a new grid defined by points $\{\mathbf{f}_k\}_{k=0}^n$ and elements of the same type. The diffeomorphism $\mathbf{f}$ can cause a change in the orientation of the grid elements, causing an overlapping of elements in the new grid. This makes interpolation difficult to handle or even highly ill-posed, which one should avoid of course. In Section 3.5, where we discussed the inverse interpolation principle, it was shown that the essential property of this mapping for accurate interpolation, is the topology preservation in space, given by (3.5.5). Typically for stiff systems we have multiple time scales which are related to different eigenvalues of the Jacobian matrix of $\mathbf{u}$, say $\lambda_l(t),\ l = 1, \ldots, N$ that are widely spread (but with a significant gap between them) in the left half of the complex plane. In other words all eigenvalues have negative real part. From (7.1.9) we see that the Jacobian matrix of $\mathbf{f}$ reads

$$D\mathbf{f} = \mathbf{I}_N - h\, D\mathbf{u}, \tag{7.1.15}$$

where $\mathbf{I}_N$ is the identity matrix of order N. If we now express the Jacobian $\det[D\mathbf{f}]$ via the aforementioned eigenvalues of $D\mathbf{u}$ we have

$$\det[D\mathbf{f}] = \prod_{l=1}^{N} [1 - h\lambda_l(t)]\,. \tag{7.1.16}$$

From the characteristics of the stiff problems mentioned above, i.e. $Re\{\lambda_l(t)\} < 0$ and (7.1.16), it is clear that $\det[D\mathbf{f}] > 0$ holds on the entire $\mathbb{R}^N$.

Again, we point out that the flow method can be used for the class of the non-autonomous problems given by

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}) + \mathbf{w}(t), \tag{7.1.17}$$

where $\mathbf{w}$ is an explicitly given time dependent function. The interpolation is then done only on the autonomous part of the velocity field and the solution at the next time-level reads

$$\mathbf{x}_j^{i+1} = \mathbf{p}\left(\mathbf{x}_j^i + h\,\mathbf{w}\left(t^{i+1}\right)\right). \tag{7.1.18}$$

It can easily be shown (as in Chapter 4) that, for a case where the autonomous part is linear in $\mathbf{x}$ and by applying (7.1.18) with $\mathbf{p}$ as the linear interpolation function, the result is *identical* to one obtained by EB.

7.2 Error analysis

In Section 4.4 it was shown that for the univariate case the local error of the flow method consists of two components: the local discretisation error of EB and the interpolation error. Of course, the same also holds for the multivariate case. Assuming that $\mathbf{u}$ is Lipschitz continuous and smooth enough, the local error of a particular point in the flow can be expressed as

$$\delta\left(\mathbf{x}_j\left(t^{i+1}\right), h\right) = d_h\left(\mathbf{x}_j\left(t^{i+1}\right)\right) + r_h\left(\mathbf{x}_j\left(t^{i+1}\right)\right), \tag{7.2.1}$$

where d_h is the local discretisation error and r_h is the interpolation error. The EB method has the consistency order 1 and IMR order 2, i.e. d_h is $O(h)$ and $O\left(h^2\right)$ respectively. The goal is thus to have the interpolation error commensurate with the discretisation error. In the previous section we pointed out that the interpolation method can be of arbitrary type if it satisfies all of three preferences noted there.

However, we will restrict ourselves to piecewise linear interpolation, i.e. the interpolation by linear polynomials to function values at $(N+1)$ points in $\mathbb{R}^N$, just for ease of argument. Also, this interpolation is local and it can be applied on irregular grids, which means that the only requirement left is that of sufficient accuracy.

Piece-wise linear interpolation was introduced in Section 3.4, where the method principle and accuracy aspects were given. Here we apply these results with the same notation. For the error bounds of piece-wise linear interpolation, we use Theorem 3.4.1 and Theorem 3.4.2, together with Theorem 3.4.3, which relates these two results. Again we will assume $\hat{\Gamma}$ to be the interpolation domain, which is actually a range of the flow domain Γ. We denote all symbols related to $\hat{\Gamma}$ by providing them with a hat symbol above.

Assume that $\hat{\Gamma}$ is covered by a set of (nondegenerate) simplices, say $\hat{\Gamma}_m$, $m = 1, \ldots, M$. The simplex $\hat{\Gamma}_m$ is defined by a set, say $\hat{S}_m$, of affinely independent $N+1$ points in $\mathbb{R}^N$. These points, which we denote by $\hat{\mathbf{x}}_{m1}, \hat{\mathbf{x}}_{m2}, \ldots, \hat{\mathbf{x}}_{m,N+1}$, are actually the vertices of the new (mapped) grid simplex. Now, $\hat{\Gamma}_m$ can be seen as a convex hull of $\hat{S}_m$, i.e.

$$\hat{\Gamma}_m := \operatorname{conv} \hat{S}_m. \tag{7.2.2}$$

Similarly to (3.4.5), we can define a diameter, say $\Delta\hat{x}$, of the new grid, like

$$\Delta\hat{x} := \operatorname{diam} \hat{S}_m = \max_{\hat{\mathbf{x}}_{mr}, \hat{\mathbf{x}}_{ms} \in \hat{S}_m} \|\hat{\mathbf{x}}_{mr} - \hat{\mathbf{x}}_{ms}\|. \tag{7.2.3}$$

The aforementioned error bounds are given for a scalar function $g : \mathbb{R}^N \to \mathbb{R}$, while our problem concerns a vectorial mapping $\mathbf{g} : \mathbb{R}^N \to \mathbb{R}^N$. However, all elements of the vector $\mathbf{g}$, say g_l, $l = 1, \ldots, N$, are defined on the same simplex $\hat{\Gamma}_m$, which allows us to do an element-wise analysis first. For any g_l we can define a linear interpolation map p_l, $l = 1, \ldots, N$, defined on $\hat{\Gamma}_m$. From (3.4.10) it follows that

$$|g_l(\hat{\mathbf{x}}) - p_l(\hat{\mathbf{x}})| \leq \frac{1}{2} \left(\hat{R}^2 - \|\hat{\mathbf{x}} - \hat{\mathbf{c}}\|^2 \right) \left\| D^2 g_l \right\|_{\infty, \hat{\Gamma}_m}, \tag{7.2.4}$$

where $\hat{R}$ and $\hat{\mathbf{c}}$ are the radius and the center of the (unique) sphere containing $\hat{S}$. The norm $\|\cdot\|$ is again the Euclidean norm and $\left\| D^2 g_l \right\|_{\infty, \hat{\Gamma}_m}$ represents the $\infty-$norm of the second derivative $D^2 g_l$ on $\hat{\Gamma}_m$, see (3.4.8).

Since all g_l (and p_l) are defined on the same simplex we have, by taking the $\infty-$norm over $|g_l(\hat{\mathbf{x}}) - p_l(\hat{\mathbf{x}})|$ and $\left\| D^2 g_l \right\|_{\infty, \hat{\Gamma}_m}$, $l = 1, \ldots, N$

$$\begin{aligned} \|\mathbf{g}(\mathbf{x}) - \mathbf{p}(\mathbf{x})\|_{\infty, \hat{\Gamma}_m} := \max_l |g_l(\hat{\mathbf{x}}) - p_l(\hat{\mathbf{x}})| &\leq \frac{1}{2} \left(\hat{R}^2 - \|\hat{\mathbf{x}} - \hat{\mathbf{c}}\|^2 \right) \max_l \left\| D^2 g_l \right\|_{\infty, \hat{\Gamma}_m} \\ &= \frac{1}{2} \left(\hat{R}^2 - \|\hat{\mathbf{x}} - \hat{\mathbf{c}}\|^2 \right) \left\| D^2 \mathbf{g} \right\|_{\infty, \hat{\Gamma}_m}. \end{aligned} \tag{7.2.5}$$

The expression $\hat{R}^2 - \|\hat{\mathbf{x}} - \hat{\mathbf{c}}\|^2$ in $\hat{\Gamma}_m$ has a maximum for the point $\hat{\mathbf{x}}^* \in \hat{\Gamma}_m$, closest to $\hat{\mathbf{c}}$. Defining the distance, say $\hat{\rho}$ between $\hat{\mathbf{c}}$ and $\hat{\Gamma}_m$, by using (3.4.12), we then find

$$r_h := \|\mathbf{g} - \mathbf{p}\|_{L_\infty(\hat{\Gamma}_m)} \leq \frac{1}{2} (\hat{R}^2 - \hat{\rho}^2) \left\| D^2 \mathbf{g} \right\|_{\infty, \hat{\Gamma}_m}. \tag{7.2.6}$$

Now, we apply (7.2.6) to our analysis. This error bound is given, of course, for direct interpolation. In our case it does not give explicit information since it still requires knowledge of a second derivative of the unknown inverse function $\mathbf{g}$ and depends

on the geometry of the new grid. However, since $D\mathbf{f} = \mathbf{I}_N - h\,D\mathbf{u}$ and $D^2\mathbf{f} = -h\,D^2\mathbf{u}$, from (3.5.18) we find

$$D^2\mathbf{g} = h\,[\mathbf{I}_N - h\,D\mathbf{u}]^{-1}\,D^2\mathbf{u}\,[\mathbf{I}_N - h\,D\mathbf{u}]^{-2}. \tag{7.2.7}$$

Substituting (7.2.7) into (7.2.6), we can eliminate $\mathbf{g}$, i.e the interpolation error bound reads

$$r_h \leq \frac{h}{2}\,(\hat{R}^2 - \hat{\rho}^2)\,\left\|[\mathbf{I}_N - h\,D\mathbf{u}]^{-1}\,D^2\mathbf{u}\,[\mathbf{I}_N - h\,D\mathbf{u}]^{-2}\right\|_{\infty,\Gamma_m}, \tag{7.2.8}$$

where Γ_m is a simplex of the original grid, i.e the original of $\hat{\Gamma}_m$. Clearly, if $\|D\mathbf{u}\|_{\infty,\Gamma_m}$ is large, which typically occurs in stiff problems (our problems of interest), then $\left\|D^2\mathbf{g}\right\|_{\infty,\hat{\Gamma}_m}$ is not necessarily large due to the inversion of the matrix $[\mathbf{I}_N - h\,D\mathbf{u}]$.

The expression $\hat{R}^2 - \hat{\rho}^2$, present in the error bound, is strongly depending on the diameter of the simplex $\hat{\Gamma}_m$ and the geometry of the new grid. Hence, let us first relate $\Delta\hat{x}$ to the diameter of the original grid Δx. By applying (7.1.8), (7.2.3) and recalling (3.5.26) we obtain an estimate of $\Delta\hat{x}$

$$\Delta\hat{x} \leq \Delta x\,\|\mathbf{I}_N - h\,D\mathbf{u}(\tilde{\mathbf{x}})\|. \tag{7.2.9}$$

Note that $\|\mathbf{x}_{mq} - \mathbf{x}_{ms}\| \leq \Delta x$, since the diameter of the original grid is not necessarily attached to the corresponding points with the same indexes of the new grid.

Now by applying Theorem 3.4.3 and (7.2.9), we find

$$\hat{R}^2 - \hat{\rho}^2 \leq \frac{1}{2}\,\frac{N}{N+1}\,\Delta x^2\,\|\mathbf{I}_N - h\,D\mathbf{u}(\tilde{\mathbf{x}})\|^2, \tag{7.2.10}$$

which holds in general case. Substituting (7.2.10) into (7.2.8), we finally obtain

$$r_h \leq \frac{h}{4}\,\frac{N}{N+1}\,\Delta x^2\,\|\mathbf{I}_N - h\,D\mathbf{u}(\tilde{\mathbf{x}})\|^2\,\left\|[\mathbf{I}_N - h\,D\mathbf{u}]^{-1}\,D^2\mathbf{u}\,[\mathbf{I}_N - h\,D\mathbf{u}]^{-2}\right\|_{\infty,\Gamma_m}. \tag{7.2.11}$$

From (7.2.11) it follows that the interpolation error is $O\left(\Delta x^2\right)$, meaning that the local error of the flow method is order $O\left(h^p\right) + O\left(\Delta x^2\right)$, where $p = 1, 2$ for EB and IMR respectively. Clearly this means that the original grid simplex size (defined by Δx) should be as such that both local error components are of the same order. If Δx is larger, then the interpolation error can become a dominant source of error, i.e. the accuracy of the original implicit method may be lost. On the other hand by doing the interpolation more accurately than needed, the error will not decrease below the error of the original implicit method.

7.3 Stability

Numerical stability properties of the flow method should be similar to those of the implicit method involved. In particular we will analyse EB for stiff problems, by studying first variations. Let $\mathbf{z}_j^i|_{j\,\mathrm{fixed}}$ denote a small perturbation of the solution $\mathbf{x}_j^i|_{j\,\mathrm{fixed}}$ of (7.1.12), such that $\mathbf{x}_j^i + \mathbf{z}_j^i|_{j\,\mathrm{fixed}}$ also satisfies (7.1.12) to first order. Now we have

$$\mathbf{x}_j^{i+1} + \mathbf{z}_j^{i+1} = \mathbf{p}\left(\mathbf{x}_j^i + \mathbf{z}_j^i\right) \doteq \mathbf{p}\left(\mathbf{x}_j^i\right) + D\mathbf{p}\left(\mathbf{x}_j^i\right)\,\mathbf{z}_j^i. \tag{7.3.1}$$

By neglecting higher order terms we have

$$\mathbf{z}_j^{i+1} = D\mathbf{p}\left(\mathbf{x}_j^i\right)\,\mathbf{z}_j^i. \tag{7.3.2}$$

For stability in a nonlinear situation it is sufficient to prove that the contractivity condition of the discrete equation (7.3.2) is satisfied, i.e.

$$\left\|\mathrm{D}\mathbf{p}\left(\mathbf{x}_j^i\right)\right\| < 1. \tag{7.3.3}$$

Before continuing we would like to relate (7.3.3) to the equivalent condition for EB, which reads

$$\left\|[\mathrm{D}\mathbf{f}]^{-1}\right\| = \left\|[\mathbf{I}_N - h\,\mathrm{D}\mathbf{u}]^{-1}\right\| < 1. \tag{7.3.4}$$

Since $\mathrm{D}\mathbf{p} \doteq \mathrm{D}\mathbf{g} = [\mathrm{D}\mathbf{f}]^{-1}$ one should expect that (7.3.3) and (7.3.4) are equivalent in a way. Indeed, this can be shown as follows. Again for the ease of argument, we will stick to the case where the interpolation (vectorial) polynomial is the piecewise linear function w.r.t. $\mathbf{x}$, i.e. $\mathbf{p}(\mathbf{x})$ on a certain simplex reads

$$\mathbf{p}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}. \tag{7.3.5}$$

Clearly $\mathbf{A} = \mathrm{D}\mathbf{p} \doteq \mathrm{D}\mathbf{g}$, which means that the matrix $\mathbf{A}^{-1}$ should be an approximation of $\mathrm{D}\mathbf{f}$. To find $\mathbf{A}$ let us choose the simplex Γ_m from the original grid with vertices $\mathbf{x}_k = [x_{1k}, x_{2k}, \ldots, x_{N,k}]^T$, $k = 1, \ldots, N+1$, and corresponding $\hat{\Gamma}_m$ from the new grid with vertices $\mathbf{f}_k = [f_{1k}, f_{2k}, \ldots, f_{N,k}]^T$, $k = 1, \ldots, N+1$, defined by (7.1.8). By applying this to (7.3.5), we have

$$\mathbf{A} = \mathbf{B}\,\mathbf{C}^{-1}, \tag{7.3.6}$$

where

$$\mathbf{B} = \begin{bmatrix} x_{12} - x_{11} & x_{13} - x_{11} & \ldots & x_{1,N+1} - x_{11} \\ x_{22} - x_{21} & x_{23} - x_{21} & & x_{2,N+1} - x_{21} \\ \vdots & & & \\ x_{N,2} - x_{N,1} & x_{N,3} - x_{N,1} & & x_{N,N+1} - x_{N,1} \end{bmatrix}, \tag{7.3.7}$$

$$\mathbf{C} = \begin{bmatrix} f_{12} - f_{11} & f_{13} - f_{11} & \ldots & f_{1,N+1} - f_{11} \\ f_{22} - f_{21} & f_{23} - f_{21} & & f_{2,N+1} - f_{21} \\ \vdots & & & \\ f_{N,2} - f_{N,1} & f_{N,3} - f_{N,1} & & f_{N,N+1} - f_{N,1} \end{bmatrix}. \tag{7.3.8}$$

For ease of argument we will choose vertices of Γ_m with coordinates

$$x_{lk} = x_{l1} + \delta_{l+1,k}\,\Delta x_l, \quad l = 1, \ldots, N,\ k = 1, \ldots, N+1, \tag{7.3.9}$$

where $\delta_{l+1,k}$ is the Kronecker delta. This gives the (original) simplex where all vertices are at the axes of the orthogonal coordinate system with the origin at $\mathbf{x}_1$ and Δx_l, $l = 1, \ldots, N$ are the lengths of edges connecting vertices $\mathbf{x}_2, \ldots, \mathbf{x}_{N+1}$ with the origin $\mathbf{x}_1$. As a 3-D illustration one should think of a tetrahedron, which has three edges parallel to x, y and z axes respectively, see Figure 7.3.1. By substituting (7.3.9) into (7.3.7) we have

$$\mathbf{B} = \mathrm{diag}\left[\Delta x_1, \Delta x_2, \ldots, \Delta x_N\right]. \tag{7.3.10}$$

On the other hand, for elements in $\mathbf{C}$, we have

$$f_{l,k+1} - f_{l1} = \Delta x_k \left(\delta_{l,k} - h\,\frac{\partial u_l\left(x_{11}, x_{21}, \ldots, c_k, \ldots, x_{N,1}\right)}{\partial x_k}\right),\ l, k = 1, \ldots, N, \tag{7.3.11}$$

where $x_{k1} \le c_k \le x_{k1} + \Delta x_k$, i.e. $c_k \in \Gamma_m$.

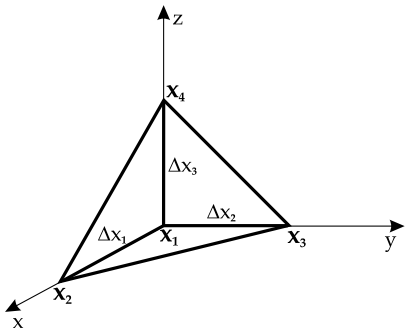


FIGURE 7.3.1. 3-D example of the original grid simplex.

We can now use (7.3.10) and (7.3.11) to obtain $\mathbf{A}^{-1}$. From (7.3.6) we have

$$\mathbf{A}^{-1} = \mathbf{C}\,\mathbf{B}^{-1} = \left\{ \delta_{l,k} - h\,\frac{\partial u_l\,(x_{11}, x_{21}, \ldots, c_k, \ldots, x_{N,1})}{\partial x_l} \right\}_{k,j=1}^{N}. \tag{7.3.12}$$

Clearly we have that $\mathbf{A}^{-1}$ is close to $D\mathbf{f}(\mathbf{x}_j^{i+1}) = \mathbf{I}_N - D\mathbf{u}(\mathbf{x}_j^{i+1})$ since $\mathbf{x}_j^{i+1} \in \Gamma_m$. This means that the contractivity condition $\|\mathbf{A}\| < 1$ of the flow method is equivalent to one of EB, given by (7.3.4). Of course, this holds only if $\det[D\mathbf{f}] > 0$, i.e. if the topology in space is preserved; otherwise the interpolation can become ill-posed. In particular obtaining $\mathbf{A}$ can represent a problem if $\mathbf{C}$ is ill-conditioned. However, as mentioned before, that is typically not the case for stiff problems of interest.

7.4 Practical Aspects

In the previous sections we introduced our method in higher dimensional problems and gave an error and stability analysis. In this section we will apply the flow method to several examples, where typically the velocity field is not known explicitly. The first one concerns the situation where the velocity is obtained experimentally. In particular we will solve a stiff problem coming from electrical networks with nonlinear elements. Hence we apply the flow method based on EB, showing that it has the desirable accuracy and stability properties.

The second example concerns a problem where the velocity is a numerical solution of a divergence free velocity field, coming from a boundary problem, that is part of a PDE. The resulting ODE can be related to a Hamiltonian form, which means that the volume, defined by the flow, should be preserved during the time integration. A typical symplectic numerical method, which has this property, is IMR (see [46]). Hence we will apply the flow method based on IMR and show that a sufficient accuracy can be achieved even for relatively long time integration intervals.

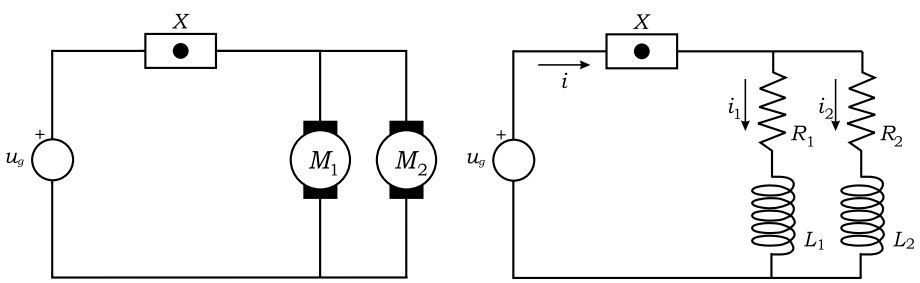
The third example is a synchronisation (control) problem of a harmonic oscillatory system. The control law is defined by the sliding mode regime, i.e. by a variable structure controller. It employs switching mechanism, which is usually modelled by the signum function. However, in practical realisation settings, one needs to use real electrical switches, which have nonlinear and explicitly not known transfer function. Hence, we use the transfer function that is represented by the tabular values only, which represent experimentally obtained data. The mathematical model of an

oscillator is an ODE in a Hamiltonian form and, thus, one needs to compute the numerical solution by applying a symplectic method. Hence we apply the flow method based on IMR to show that computing the solution can be achieved with a sufficient accuracy. It is also shown that the amplitude of the solution is approximately constant even for a long time integration interval.

The last example in this section is related to the motion of a viscous axisymmetric body driven by the surface tension. Here the velocity field is described by the Stokes equations that need to be solved by some numerical method (e.g. the finite element method) at every time level. Here the physical flow evolution is such that the body volume is preserved in time. Hence for the time discretisation, a symplectic method needs to be used. We again apply the flow method based on the IMR and obtain sufficiently accurate results, even for large time scales. The example is of a special interest since the flow problem is not quite autonomous and it shows that the flow method can be successfully applied even for some non-autonomous problems.

7.4.1 An Electrical Network

Consider an electrical network with two DC motors, power-supplied by the same source and current protected by a nonlinear resistor, see Figure 7.4.1a. The most severe situation is when both motor shafts (i.e. rotors) are blocked. Then currents are largest and the equivalent electrical circuit is shown in Figure 7.4.1b. Here both motors are modeled as serial connections of a resistor (R_k, $k = 1, 2$) and an inductor (L_k) and since both R_k and L_k are relatively small all currents tend to increase under the influence of the relatively large input voltage.



(a) Principle scheme (b) Electrical scheme with rotors blocked

FIGURE 7.4.1. DC motors supplied from the same power source

The task of the nonlinear resistor is to prevent currents not to increase (motors current protection) under the high voltage at the input, here $u_g = u_g(t)$. According to this the transfer function (U-I characteristic) of the nonlinear resistor X is very steep, making the problem stiff. Moreover, this characteristic is not known in closed form and usually given by some tabular values. The mathematical model of the system follows from the second Kirchhoff's law, i.e

$$\begin{aligned} L_1 \frac{di_1}{dt} &= -u(i_1 + i_2) - R_1 i_1 + u_g(t), \\ L_2 \frac{di_2}{dt} &= -u(i_1 + i_2) - R_2 i_2 + u_g(t), \end{aligned} \tag{7.4.1}$$

where $i_k = i_k(t)$ are the currents in the motors loops and $u(i)$ is the voltage on the nonlinear resistor. The system (7.4.1) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}) + \mathbf{w}(t), \tag{7.4.2}$$

where

$$\mathbf{x} = \begin{bmatrix} i_1, & i_2 \end{bmatrix}^T, \quad \mathbf{w}(t) = u_g(t) \begin{bmatrix} \frac{1}{L_1}, & \frac{1}{L_2} \end{bmatrix}^T,$$

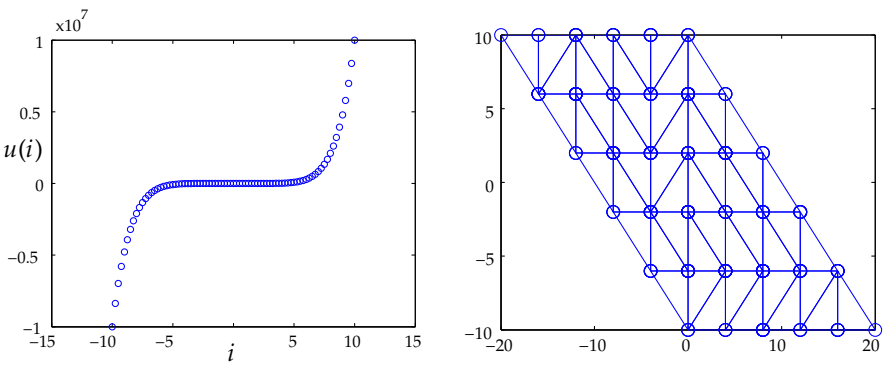
$$\mathbf{v}(\mathbf{x}) = \begin{bmatrix} -\frac{1}{L_1}\left(u(x_1 + x_2) + R_1 x_1\right), & -\frac{1}{L_2}\left(u(x_1 + x_2) + R_2 x_2\right) \end{bmatrix}^T. \tag{7.4.3}$$

Here we take $u_g(t) = V \cos \omega t$ and as typical parameters values $R_1 = 2\Omega$, $R_2 = 1\Omega$, $L_1 = 10^{-2}$H, $L_2 = 10^{-4}$H, $V = 220$V and $\omega = 1\,rad/s$. The U-I transfer function $u(x)$ is obtained experimentally (at n points $\{y_k\}_{k=1}^n$), see Table 7.4.1 and Figure 7.4.2a.

y	-10.0	-9.75	...	-0.25	0.0	0.25	...	10.0
$u(y)$	-1.0e+7	-8.376e+6	...	-6.1e-5	0.0	6.1e-5	...	1.0e+7

TABLE 7.4.1. Discrete U-I transfer function of the nonlinear transistor

Of course, this set can be used both partially and totally. Hence we will assume that $n \leq 81$. Since we know u only for the given set $\{y_k\}_{k=1}^n$ we can create a 2-D triangular grid with points $\mathbf{x}_{k,l} = (x_{1,k,l}, x_{2,k,l})$, $l = 1, \ldots, M$, where the coordinates of all grid points satisfy $y_k = x_{1,k,l} + x_{2,k,l}$. Of course, the number of points M is arbitrary. But since we would like to keep our original grid as good as possible, we keep the symmetry of the grid, meaning that the distances between neighbouring points in both directions, say Δy_1 and Δy_2, are equal to Δy, which is determined by the experimentally obtained data. Hence we take $M = n$. Now we have a set of n^2 points which can be triangularized, obtaining a 2-D grid (see Figure 7.4.2b where $n = 6$), for which $\mathbf{v}(\mathbf{x})$ is known in all vertices.



(a) Discrete U-I transfer function (b) Triangular 2-D grid

FIGURE 7.4.2. Discrete velocity field and the constructed mesh

To show the influence of the interpolation error we do the following numerical experiment. For fixed $h = 0.01$ and the flow points initial values

$$\mathbf{I}^0 = \left\{ [1.0, 0.0]^T, [0.5, 0.5]^T \right\},$$

we compute the solutions up to the final time of computation $T_f = 3.5$. The computation is performed by using different numbers of the given tabular points, in particular $n = 11, 21, 41, 81$, which correspond to the spatial step sizes $\Delta y = 2.0, 1.0, 0.5, 0.25$ respectively. Of course for a triangularisation as shown in Figure 7.4.2, the diameter of all simplices reads $\Delta x = \sqrt{2}\Delta y$. To assess the accuracy of the flow method we compare results with the EB solutions with a much smaller time step $h^* = 0.0001$. By taking the $\infty-$norm of the global error at T_f over all flow points (and their both coordinates) we obtain the dependence between the error and the diameter of the original grid simplex. According to (7.2.11) this dependence should be quadratic (up to the constant), which can be seen in Figure 7.4.3. Here the solid line represents the error of the flow method and the dashed line a quadratic function $q(a) = Ca^2$. Of course, such a behaviour of the global error is due to the fact that the time step is much smaller than the simplex diameter, making the interpolation error the dominant error source.

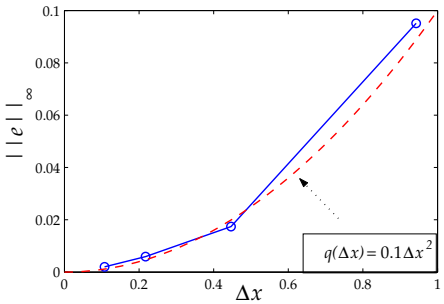


FIGURE 7.4.3. Global error of the flow method as a function of the original grid diameter

As the final remark we note that the flow method is numerically stable, i.e. there are no time step constraints even for the highly stiff problems as this one.

7.4.2 A Boundary Problem

Consider the following problem

$$\nabla^2 \cdot \mathbf{u} = \mathbf{F}(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{7.4.4}$$

defined on some domain Γ with the boundary $\partial\Gamma$. Here $\mathbf{x}$ represents the point in Γ with the velocity $\mathbf{u} = \mathbf{u}(\mathbf{x})$. By describing Dirichlet boundary conditions (BC) for $\mathbf{u}$, we have a boundary value problem. To obtain the exact solution of such problem is in general impossible. However, there is a variety of different numerical schemes, such as e.g. finite elements or finite differences which give an approximation of the solution (up to a certain accuracy) on a discrete (finite dimensional) subdomain, say $\Gamma^* \subset \Gamma$. Actually Γ^* represents the set of nodes of the grid which covers Γ. To solve the flow problem (7.1.1) defined by such velocity and the initial condition $\mathbf{I}^0 \subseteq \Gamma$, one needs to solve an autonomous ODE with discretely given $\mathbf{u}$.

Let us consider the problem (7.4.4), where $\mathbf{x} = [x, y]^T$ is the point in Cartesian coordinates, $\mathbf{u} = [u_x(x, y), u_y(x, y)]^T$ and

$$\mathbf{F}(x, y) = -\cos y - x \sin y. \tag{7.4.5}$$

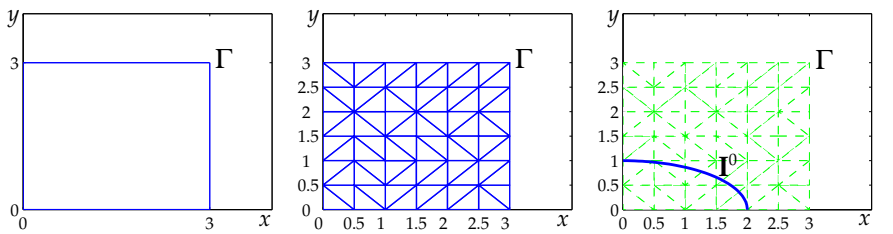
The domain is a rectangle (see Figure 7.4.4a) defined by

$$\Gamma := \{(x, y) \,|\, 0 \le x \le 3,\ 0 \le y \le 3\}. \tag{7.4.6}$$

Let the boundary conditions be given by

$$\mathbf{u}(\mathbf{x})|_{\partial\Gamma} = \begin{cases} [0,\ 0]^T, & x = 0, \\ \left[-\frac{1}{2}x^2,\ 0\right]^T, & y = 0, \\ \left[-\frac{9}{2}\cos y,\ 3\sin y\right]^T, & x = 3, \\ \left[-\frac{\cos 3}{2}x^2,\ x\sin 3\right]^T, & y = 3. \end{cases} \tag{7.4.7}$$

To obtain the numerical solution of this problem one needs to discretise (7.4.4). To avoid a discussion on error contamination due to this discretisation we will use the exact solution of the boundary value problem, which we happen to know in this case. Indeed, we find a solution of (7.4.4), (7.4.5), (7.4.6) and (7.4.7)



(a) The domain Γ (b) The triangular grid (c) The flow initial condition

FIGURE 7.4.4. The boundary problem domain

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}x^2\cos y \\ x\sin y \end{bmatrix}. \tag{7.4.8}$$

We will assume this to be known at a set of grid points $\{\mathbf{x}_k\}_{k=1}^n \in \Gamma^*$ only. Now, let us define the flow problem

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \tag{7.4.9}$$

with initial condition $\mathbf{x}(0) \in \mathbf{I}^0$, where $\mathbf{I}^0$ is a quarter of an ellipse defined by

$$\mathbf{I}^0 = \left\{ (x, y) \,\middle|\, \frac{x^2}{4} + y^2 \le 1,\ x, y \ge 0 \right\}. \tag{7.4.10}$$

Clearly $\mathbf{I}^0 \subset \Gamma$ as shown in Figure 7.4.4c and it is discretised (as pointed out in Section 7.1) by a set of points $\{\mathbf{x}_j\}_{j=1}^\mu$. Assuming well-posedness these points can be placed at the boundary of the flow only. Now we apply the flow method, defined by (7.1.5) and (7.1.13), and compute the flow evolution in time, see Figure 7.4.5. Here we take $h = 0.01$, $n_j = 10$ and $T_f = 2.0$. To compare the results we also compute the solution of the "standard" IMR method with the same time step size.

By taking the $\infty-$norm of the vector of differences between results for both coordinates we have the measure of the additional error (coming from interpolation) introduced by the flow method. We perform the numerical experiment by taking $n = 6 \times 6$, 11×11, 21×21, 41×41, 81×81, for which we have the corresponding spatial step (for both coordinates) $\Delta y = 0.6, 0.3, 0.15, 0.075, 0.0375$ and $\Delta x = \sqrt{2}\Delta y$.

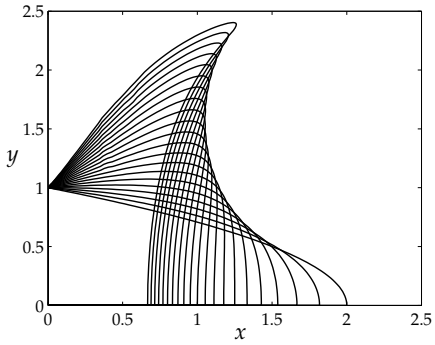
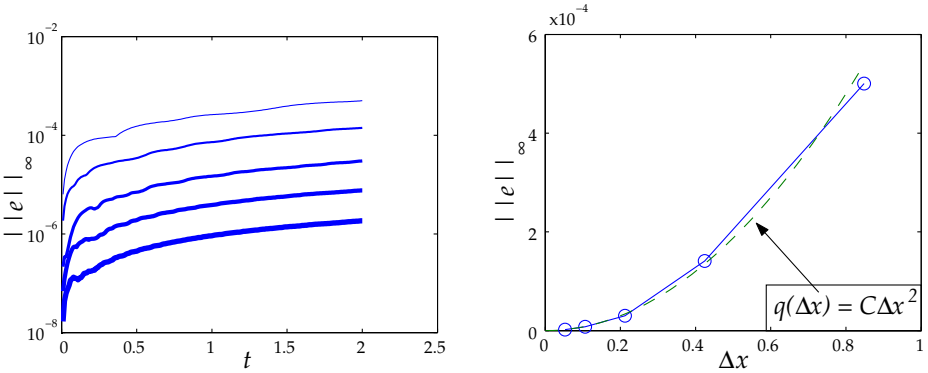


FIGURE 7.4.5. Quarter of ellipse time evolution

Now we can observe how the interpolation error behaves in time, depending on the size of the spatial step used, see Figure 7.4.6a. Here, the thicker the line the smaller the grid step size. Note that even for a relatively large spatial step size ($\Delta y \leq 0.3$), the order of the error is smaller then $O\left(h^2\right)$, which is the order of IMR, even for the relatively long time scale introduced here. This means that we can compute the flow evolution relatively cheaply (without using extreme number of grid points) and yet keep the interpolation error negligible comparing to the local discretisation error for a large number of time steps. The interpolation error is again quadratically dependent on the diameter of the simplex, which can be shown by computing the error at the end of the computational time for different values of a. Again we obtain the relation, which is close to quadratic function $q(a) = Ca^2$, see Figure 7.4.6b (where $C = 0.00075$). For the volume preservation results one should see [32], where this example was introduced.



(a) Error in time (thicker line - smaller Δx)

(b) Error-diameter dependence

FIGURE 7.4.6. Error in time and as a function of the diameter

7.4.3 Harmonic oscillator synchronisation

Consider a synchronisation control problem of the harmonic oscillatory system introduced in [50] and shown in Figure 7.4.7. Although the original problem concerns the control of both the amplitude and the phase of the system, we restrict ourselves to the

amplitude synchronisation only, since in these settings the problem is autonomous. Moreover, we restrict ourselves to the unperturbed, two-phase oscillator.

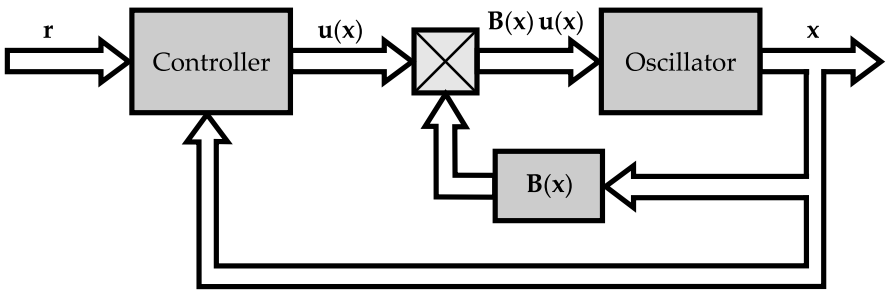


FIGURE 7.4.7. Variable structure control of the oscillatory harmonic system

A mathematical model of such a system is given by the following ODE system

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}(\mathbf{x})\,\mathbf{v}(\mathbf{x}) := \mathbf{u}(\mathbf{x}), \tag{7.4.11}$$

where $\mathbf{x} = [x_1,\, x_2]^T$ and $\mathbf{v}(\mathbf{x}) = [v_1\,(x_1, x_2)\,,\, v_2\,(x_1, x_2)]^T$. Matrices $\mathbf{A}$ and $\mathbf{B}(\mathbf{x})$ are defined by

$$\mathbf{A} := \begin{bmatrix} 0 & \omega_0 \\ -\omega_0 & 0 \end{bmatrix}, \quad \mathbf{B}(\mathbf{x}) := \begin{bmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{bmatrix}. \tag{7.4.12}$$

The value of ω_0 represents the natural radial frequency of the oscillatory system and the matrix $\mathbf{B}(\mathbf{x})$ is chosen in such a (symmetrical) form to allow decoupling of the amplitude and the phase control.

As mentioned, we are only interested in the synchronisation of amplitude, say $x_A(t)$, with the amplitude of the reference input signal, say r_r. Then the control requirement can be expressed as

$$x_A(t) = r_r, \tag{7.4.13}$$

and we assume that amplitudes of the both components of $\mathbf{x}$ are the same. The reference input signal is given by

$$\mathbf{r} := \begin{bmatrix} r_r\, \sin\left(\omega_r t + \theta_{r0}\right) \\ r_r\, \cos\left(\omega_r t + \theta_{r0}\right) \end{bmatrix}. \tag{7.4.14}$$

Since the phase control is omitted, the role of the referent frequency ω_r and initial angle θ_{r0} is insignificant.

As shown in detail in [50], the control law is constructed by ensuring a sliding mode regime of the system on an appropriate sliding surface, say s, defined by

$$s\,(x_1, x_2) := r_r^2 - x_1^2 - x_2^2. \tag{7.4.15}$$

Obviously, if the sliding regime, on the nonlinear sliding surface $s = 0$, is possible, requirement (7.4.13) is fulfilled. Hence by choosing

$$\mathbf{v}(\mathbf{x}) := \begin{bmatrix} \alpha\,\psi\left(r_r^2 - x_1^2 - x_2^2\right) \\ 0 \end{bmatrix}, \tag{7.4.16}$$

where ψ is the switching function, the solution will reach the sliding surface in finite time. Hence we obtain the control goal, since after this time (7.4.13) holds. We remark that the second component of $\mathbf{v}$ is set to zero since we are not interested in controlling the phase of the system.

In the theory of the variable structure control systems the switching function is often modelled by the signum function, i.e. $\psi(x) := \operatorname{sgn}(x)$, where

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0, \\ 0, & x = 0, \\ 1, & x > 0. \end{cases} \tag{7.4.17}$$

However, in practical realisations electrical switches do not have such (ideal) transfer functions. Often their (real) transfer functions are functions that are approximately close to (7.4.17). For example, as shown in [1], the voltage controlled switch is approximated by a transfer function defined by

$$\psi(x) = \begin{cases} -1, & x < 0, \\ \phi(x), & x = 0, \\ 1, & x > 0, \end{cases} \tag{7.4.18}$$

where

$$\phi(x) = \begin{cases} X = \frac{x - X_0}{2X_0}, & \\ Y = 2X^2, & X < 0.5, \\ Y = 1 - 2(X-1)^2, & X \geq 0.5, \\ \phi(x) = \Re\left\{(-1)^{1-Y}\right\}. & \end{cases} \tag{7.4.19}$$

To obtain the exact transfer function, one would need to perform an experiment. The result would then be a discrete set of tabular values. Here, for simplicity we just sample the approximate switching function ψ at n points, with $X_0 = 0.2$, as shown in Figure 7.4.8.

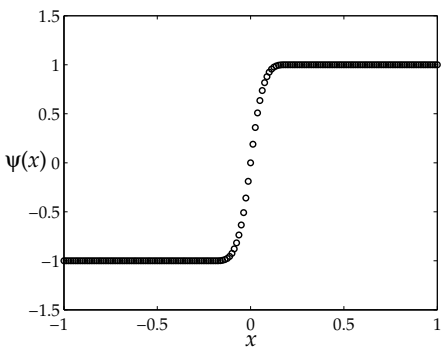


FIGURE 7.4.8. Transfer function of the voltage controlled switch

By taking $\omega_0 = 6$, $r_r = 1$ and $\alpha = 3$ and by substituting the discrete switching function ψ into (7.4.16) and (7.4.11) we obtain the velocity field $\mathbf{u}$. On a rectangle

$$\Gamma := \{(x_1, x_2) \,|\, -2 \leq x \leq 2,\ -2 \leq y \leq 2\}, \tag{7.4.20}$$

the first component of the velocity field u_1 is shown in Figure 7.4.9. The other one is linear. To obtain the velocity field, we triangularize Γ by $n \times n$, where

$$\Delta x = \frac{4\sqrt{2}}{n-1}, \tag{7.4.21}$$

represents the diameter of the original grid.

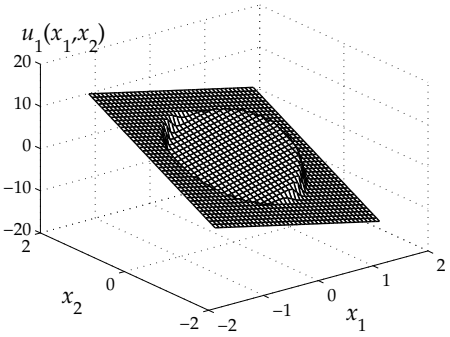


FIGURE 7.4.9. The first component of the velocity field

Since the solution approach time to the sliding surface depends on initial values, we take a set

$$\mathbf{I}(\mathbf{x},0) := \{(x_1,x_2)\,|\,-1.5 \le x \le 1.5,\ -1.5 \le y \le 1.5\}, \tag{7.4.22}$$

to be the initial flow. Of course, to obtain the flow, we first need to discretise the set $\mathbf{I}(\mathbf{x},0)$. Let us take $n = 16$ points equispaced points (four per direction) on a rectangle $\mathbf{I}(\mathbf{x},0)$. Now, we have the initial flow as $\mathbf{I}^0 = \left\{\mathbf{x}_j^0\right\}_{j=0}^{15}$.

To compute the solution, we need an appropriate numerical method. Since the system, once the sliding mode is ensured, is in the Hamiltonian form, we choose (like in the previous example) a symplectic method. Hence we apply the flow method based on IMR. By taking $h = \pi/30 \doteq 0.1$, $\Delta x = 0.1$ and the time of computation $T_f = \pi/2$, we obtain the solution as shown in Figure 7.4.10 from two different aspect angles. Clearly, after some finite time interval the flow solutions are close to the unit circle, which is represented as a cylinder in time.

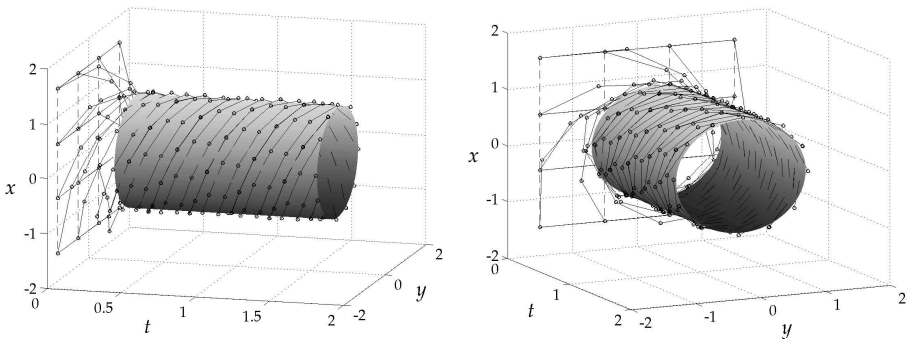
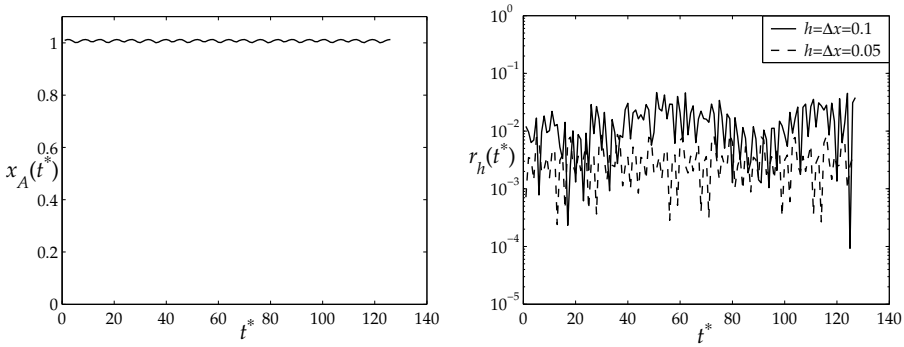


FIGURE 7.4.10. Flow in time

Numerically the most interesting aspect of this problem is the preservation of the amplitude for larger time intervals. As known, some numerical methods generate an increasing global error, causing the increase or decrease of the amplitude in time. Of course, the IMR does not have this problem, since it is a symplectic method and the flow method should preserve this property. Hence let us solve the same problem,

with the same values of coefficients, but on the interval $T_f = 45\,\pi$. The result is given in Figure 7.4.11a, where we show the amplitude of the solution as function of the number of periods, say t^*. The amplitude is not constant, which is caused by the imperfections of the switching function, the sliding mode algorithm and, most of all, the relatively large time step. However, one should note that the value of the amplitude stays in the neighbourhood of the input amplitude. In other words, there is effectively no increase (or decrease), even for a relatively large time scale.

In this particular case we know the exact switching function, whence we can compute the solution by applying the standard IMR. The difference between two results is actually the interpolation error, which is shown in Figure 7.4.11b. Since interpolation is linear, one should note that the error is commensurate with the latter.



(a) Amplitude (b) Interpolation error

FIGURE 7.4.11. Numerical results

7.4.4 Stokes problem

Consider the motion of a viscous axisymmetric body driven by the surface tension introduced in [32]. At a fixed time point and on a certain spatial domain Γ, the problem is described by the Stokes equations (see [41])

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, \\ \nabla \cdot \sigma &= 0, \end{aligned} \tag{7.4.23}$$

where $\sigma = -p\,\mathbf{I} + \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)$ is the stress tensor. Here $\mathbf{u}$ is the velocity of the fluid and p is the pressure. Let us assume the body to be axisymmetric, so we then have (in cylindrical coordinates) $\mathbf{u} = [u_r, u_z]^T$. The evolution of the body is given by the flow problem

$$\begin{cases} \frac{dx}{dt} = \mathbf{u}, & \mathbf{x} = [r,\, z]^T, \\ \mathbf{x}(0) \in \mathbf{I}(\mathbf{x}, 0). \end{cases} \tag{7.4.24}$$

In particular we are interested in the flow of the boundary of the computational domain Γ. Of course, the initial flow is defined by the boundary of the initial computational domain, say Γ^0. Let us take Γ^0 to be an ellipsoid with principle axes in r and z direction equal to 1 and 0.5 respectively. Employing the symmetry of the body, it is sufficient to let Γ^0 be a quarter of an ellipse. In Figure 7.4.12 the time evolution of the ellipsoid is shown.

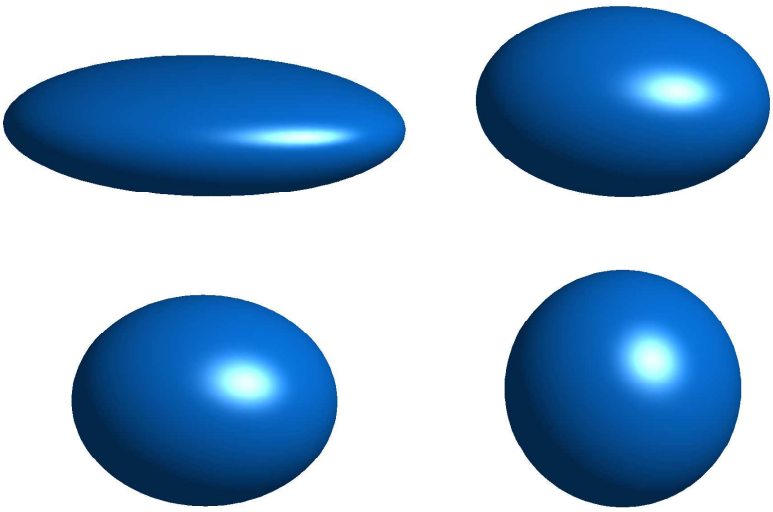


FIGURE 7.4.12. Time evolution of the ellipsoid

To solve (7.4.23) we need boundary conditions. At a particular time point t^i, the boundary is defined by the flow $\mathbf{I}^i$ that consists of three parts $\mathbf{I}^i_{Or}$, $\mathbf{I}^i_{Oz}$ and $\mathbf{I}^i_\rho$, where $\mathbf{I}^i_{Or}$ and $\mathbf{I}^i_{Oz}$ are parts of r and z axes and $\mathbf{I}^i_\rho$ is defined by the free boundary, see Figure 7.4.13 for the initial flow.

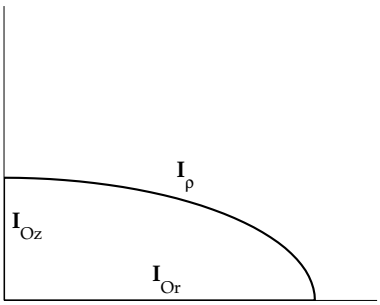


FIGURE 7.4.13. Initial flow

For $\mathbf{x} \in \mathbf{I}^i_{Or} \cup \mathbf{I}^i_{Oz}$ we have symmetric boundary conditions

$$\mathbf{u} \cdot \mathbf{n} = 0,$$
$$\sigma \mathbf{n} \cdot \mathbf{t} = 0, \tag{7.4.25}$$

where $\mathbf{n}$ is the outward normal and $\mathbf{t}$ is the tangent to the boundary. For $\mathbf{x} \in \mathbf{I}^i_\rho$ we take surface tension boundary conditions (see [41]), which read

$$\sigma \mathbf{n} = -\kappa(\rho)\, \mathbf{n}, \tag{7.4.26}$$

where $\kappa(\rho)$ is the curvature of the boundary.

The solution of (7.4.23) can e.g. be obtained by using a finite element method as shown in [32]. A thus obtained velocity can be used for solving the flow problem (7.4.24). Rewriting the system (7.4.24) in Hamiltonian form we can also numerically

preserve the body volume if we use a symplectic numerical method. Applying the flow method based on the implicit midpoint rule, one can solve the time evolution of the flow as already shown in previous examples. However, in this example the velocity field is not quite autonomous, since it depends on the time dependent free boundary. Hence (7.1.7) does not hold. On the other hand, from

$$\mathbf{u}\left(t^{i+1/2},\mathbf{x}\right) \doteq \mathbf{u}\left(t^{i},\mathbf{x}\right) + \frac{h}{2}\frac{\partial \mathbf{u}\left(t^{i},\mathbf{x}\right)}{\partial t}, \tag{7.4.27}$$

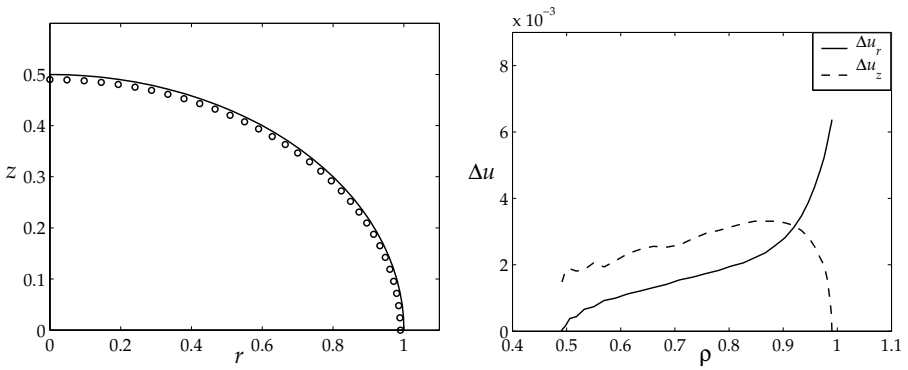
it follows that the explicit time dependence of $\mathbf{u}$ can be neglected if the following holds

$$\frac{h}{2}\left\|\frac{\partial \mathbf{u}\left(t,\mathbf{x}\right)}{\partial t}\right\|_{t\in\left[t^{i},t^{i+1/2}\right]} \ll \left\|\mathbf{u}\left(t,\mathbf{x}\right)\right\|_{t\in\left[t^{i},t^{i+1/2}\right]}, \quad \mathbf{x}\in\Gamma. \tag{7.4.28}$$

In this particular example if

$$\frac{h}{2}\left\|\frac{\partial \mathbf{u}\left(t,\mathbf{x}\right)}{\partial t}\right\|_{t\in\left[t^{i},t^{i+1/2}\right]} = O\left(\Delta x^{2}\right), \tag{7.4.29}$$

where Δx is the largest diameter of the grid elements, the overall accuracy is the same, if the finite element method has the second order accuracy. To verify this, we fix a number of control points near the free boundary where the velocity field changes the most, see Figure 7.4.14a. We compute velocity differences Δu_r and Δu_z at these points at two consecutive time-levels, see Figure 7.4.14b. Here ρ denotes the distance from the origin to a control point. The number of boundary nodes is $n_b = 2^6$, which gives $\Delta x \approx 0.08$. Since $\Delta u_r, \Delta u_z = O(\Delta x^2)$, it is clear that (7.4.29) holds, which allows us to implement the flow method directly.



(a) Control points (b) Velocity differences

FIGURE 7.4.14. Velocity differences computed over a single time step ($h = 0.01$, $\Delta x = 0.03$)

Numerical results obtained in [32] are shown in Figure 7.4.15. One should note that the accuracy of the computed area is much higher than the discretisation error.

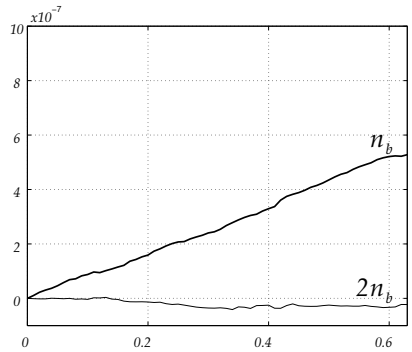


FIGURE 7.4.15. The evolution of the volume error ($h = 0.01$, $n_b = 2^7$)

CHAPTER 8

Conclusions and recommendations

Standard approach for solving ODE flows by applying implicit numerical methods involve Newton iteration for which the iterative function needs to be known explicitly as well as the Jacobian of the velocity. Moreover, iteration is a recursive process and the convergence is not guaranteed in general. In this thesis we developed a new method for solving autonomous flow problems, which has some advantages over the standard methods that are typically used. The method is based on inverse interpolation and it needs known values of the velocity field at some spatial points only, i.e. it does not require the velocity to be explicitly expressed. It does not introduce recursion and it is effectively an explicit method. The accuracy is similar to the accuracy of the employed implicit method, assuming that the interpolation accuracy is sufficient. The interpolation error can be controlled independently by choosing an appropriate grid and an appropriate interpolation method.

The interpolation method used by the flow method, needs to satisfy several prerequisites. Firstly, the method should be appropriate for interpolation of scattered data, since interpolation is performed on an irregular grid in general. Secondly, the interpolation accuracy should be sufficient so that the interpolation error be commensurate with the local discretisation error of the standard implicit method employed. Finally, interpolation computational costs should not be large, especially for higher dimensional problems. Interpolation should preferably be done locally, which should decrease computational costs and relax the implementation problems. An example of a method that has these characteristics is piece-wise linear interpolation. This method showed satisfactory results, both with respect to accuracy and stability.

In this thesis we considered some of the most important implicit numerical methods, in particular BDF methods, and some implicit RK methods such as Gauss, Radau, Lobatto and DIRK methods. Beside the stability and the accuracy analysis, we also investigated the well-posedness of the solution, i.e. conditions for flow integral curves not to intersect. It turns out that not all of the methods mentioned generate satisfactory results. The flow method mimics this property, which is important for choosing the method to be used.

Practical aspects of the flow method considered here are problems from fluid dynamics, electrical networks and control systems. We have solved a number of stiff and Hamiltonian problems with a discretely given velocity field. Although the method is constructed for autonomous problems, it can successfully be applied to a class of

non-autonomous problems where explicit time dependence of the velocity field is expressed via the forcing term only. Moreover, the method can produce satisfactory results even for general non-autonomous problem if the velocity is relatively slowly changing in time, i.e. over a single time-step of integration.

If the problem that needs to be solved is stiff and the stiffness is not related to the explicit time dependence of the velocity field, one can simply extrapolate velocity values over a single time interval and yet obtain satisfactory results. For solving Hamiltonian problems, in particular the evolution of a material blob with a slowly (in time) varying velocity field (defined by Stokes equations), it was shown that results obtained by the flow method based on implicit symplectic method are much more accurate than ones obtained by some explicit (non-symplectic) method, such as e.g. Euler Forward. For problems where the velocity field is not slowly varying in time, the flow method can be used in combination with an additional numerical method, which should be used for computing several starting values only in order to obtain the information about the explicit time dependence of the velocity field. Then by increasing the dimensionality of the system, i.e. by including time as the additional unknown, the problem is transformed into the autonomous form.

Implicit numerical methods are also used for solving other, more complex problems described by DAEs (Differential Algebraic Equations) and PDEs (Partial Differential Equations). The principle of the flow method shows that the method implementation can similarly be done for these problems. Moreover, solving some one-dimensional semidiscretised PDEs showed that the solution can be obtained by applying univariate interpolation only, although the actual ODE system is of much higher dimensionality (coming from spatial discretisation). This can decrease computational costs tremendously, since there is no need to compute the Jacobian of the higher dimensional system. However, since this is not the case in general, it was not included in this thesis.

Bibliography

[1] AGILENT. Circuit Components System Models. Tech. rep., 2002.

[2] AHLBERG, J. H., NILSON, E. N., AND WALSH, J. L. *The Theory of Splines and Their Applications*. Academic Press, 1967.

[3] AUBIN, T. *A Course in Differential Geometry*. American Mathematical Society, 2001.

[4] BERNSTEIN, S. N. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Comm. Soc. Math. Kharkov 13* (1912), 1–2.

[5] BLU, T., THÉVENAZ, P., AND UNSER, M. How a simple shift can significantly improve the performance of linear interpolation. In *Proceedings of the 2002 IEEE International Conference on Image Processing (ICIP'02)* (Rochester NY, USA, September 22-25, 2002), vol. III, pp. 377–380.

[6] BOOTHBY, W. M. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, Inc., 1975.

[7] BROWNLEE, R., AND LIGHT, W. Approximation orders for interpolation by surface splines to rough functions. *IMA J. Numer. Anal.* (to appear).

[8] BUTCHER, J. C. Implicit Runge-Kutta processes. *Math. Comp. 18* (1964), 50–64.

[9] BUTCHER, J. C. A stability property of implicit Runge-Kutta methods. *BIT 15* (1975), 358–361.

[10] BUTCHER, J. C. *The Numerical Analysis of Ordinary Differential Equations*. John Wiley & Sons, 1987.

[11] CAROTHERS, N. L. A short course on approximation theory. Tech. Rep. Math 682, Bowling Green State University, 1998.

[12] CHENEY, E., AND LIGHT, W. *A Course in Approximation Theory*. Brooks Cole, Pacific Grove, 2000.

[13] CURTISS, C. F., AND HIRSCHFELDER, J. O. Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America 38* (1952), 235–243.

[14] DE BOOR, C. *A Practical Guide to Splines*. Springer, 1978.

[15] DEKKER, K., AND VERWER, J. G. *Stability of Runge-Kutta methods for stiff nonlinear differential equations*. North-Holland, 1984.

[16] DEVORE, R. A., AND LORENTZ, G. G. *Constructive Approximation*. Springer-Verlag, 1993.

[17] DORSSELAER, J. L. M., AND SPIJKER, M. N. The error committed by stopping the Newton iteration in the numerical solution of stiff initial value problem. *IMA J. of Num. An. 14* (1994), 183–209.

[18] DUCHON, J. Sur l'erreur d' interpolation des fonctions de plusieurs variables par les D^m-splines. *RAIRO Analyse numerique 12* (1978), 325–334.

[19] EHLE, B. L. High order A-stable methods for the numerical solution of systems of differential equations. *BIT 8* (1968), 276–278.

[20] FRANKE, R. Scattered data interpolation: Tests of some methods. *Mathematics of Computation 38* (1982), 181–200.

[21] GEAR, C. W. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, 1971.

[22] GRAY, P. R., HURST, P. J., LEWIS, S. H., AND MEYER, R. G. *Analysis and design of analog integrated circuits*, 4th ed. John Wiley & Sons, 2001.

[23] HAIRER, E., NORSET, S. P., AND WANNER, G. *Solving ordinary differential equations I - Nonstiff problems*. Springer-Verlag, 1987.

[24] HAIRER, E., AND WANNER, G. *Solving ordinary differential equations II - Stiff and differential-algebraic problems*. Springer-Verlag, 1991.

[25] HANDSCOMB, D. C. Errors of linear interpolation on a triangle. Tech. Rep. Research Report NA-95/09, Oxford University, 1995.

[26] HARTMAN, P. *Ordinary Differential Equations*. John Wiley & Sons, 1964.

[27] JOHNSON, M. J. The L_2-approximation order of surface spline interpolation. *Mathematics of Computations 70* (2000), 719–737.

[28] KREYSZIG, E. *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1989.

[29] KUNTZMANN, J. Neure entwicklungen der methoden methoden von runge und kutta. *Z. Angew. Math. Mech. 41* (1961), 28–31.

[30] KUNZ, K. S. *Numerical Solutions of Ordinary Differential Equations: Methods of Starting the Solution*. McGraw-Hill, 1957.

[31] LAMBERT, J. D. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons Ltd., 1991.

[32] MATTHEIJ, R. M. M., AND LAEVSKY, K. Numerical volume preservation of a divergence free fluid under symmetry. Tech. Rep. RANA 01-11, Eindhoven University Of Technology, 2001.

[33] MATTHEIJ, R. M. M., AND MOLENAAR, J. *Ordinary Differential Equations in Theory and Practice*. John Wiley & Sons, 1996.

[34] MEINGUET, J. Surface spline interpolation: Basic theory and computational aspects. In *Approximation Theory and Spline Functions* (1984), S. P. Singh, Ed., D. Reidel Publ. Company, pp. 127–142.

[35] MHASKAR, H. N., AND PAI, D. V. *Fundamentals of Approximation Theory*. Narosa Publishing House, 2000.

[36] MILOVANOVIĆ, G. *Numerička Analiza - II deo*. Naučna Knjiga, 1991.

[37] ORTEGA, J. M., AND RHEINBOLDT, W. C. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, Inc., 1970.

[38] OSHER, S., AND SETHIAN, J. A. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 1 (1988), 12–49.

[39] POWELL, M. J. D. The uniform convergence of thin plate spline interpolation in two dimensions. *Numer. Math. 68* (1994), 107–128.

[40] RAMSDEN, D., AND HOLLOWAY, G. Timestepping Langrangian particles in two dimensional Eulerian flow fields. *Journal of Computational Physics 95* (1991), 101–116.

[41] RICHARDSON, S. Two-dimensional Stokes flows with time-dependent free boundaries driven by surface tension. *European Journal of Applied Mathematics 8* (1997), 311–330.

[42] RIENSTRA, S. W., AND CHANDRA, T. D. Analytical approximations to the viscous glass flow problem in the mould-plunger pressing process, including an investigation of boundary conditions. *Journal of Engineering Mathematics 39* (2001), 241–259.

[43] RIPPA, S. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal. 29* (1992), 257–270.

[44] ROSSO, R., SONNET, A. M., AND VIRGA, E. G. Evolution of vesicles subject to adhesion. *R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci. 456* (2000), 1523–1545.

[45] SANDWELL, D. T. Biharmonic spline interpolation of geos-3 and seasat altimeter data. *Geophysical Research Letters 14* (1987), 139–142.

[46] SANZ-SERNA, J. M., AND CALVO, M. P. *Numerical Hamiltonian Problems*. Chapman & Hall, 1994.

[47] SUBBOTIN, Y. N. Dependence of estimates of a multidimensional piecewise polynomial approximation on the geometric characteristics of the triangulation. *Proc. Stek. Inst. Math. 189* (1990), 135–159.

[48] SUBBOTIN, Y. N. Error of the approximation by interpolation polynomials of small degrees on n-simplices. *Math. Notes 48* (1990), 1030–1037.

[49] VAN DE VORST, G. A. L. Numerical simulation of axisymmetric viscous sintering. *Engineering Analysis with Boundary Elements 14* (1995), 193–207.

[50] VESELIĆ, B., AND Č. MILOSAVLJEVIĆ. Sliding mode based harmonic oscillator synchronization. *International Journal of Electronics 90* (2003), 553–570.

[51] WALDRON, S. The error in linear interpolation at the vertices of a simplex. *SIAM Journal on Numerical Analysis 35* (1998), 1191–1200.

[52] WEIERSTRASS, K. Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen. *Sitzungsberichte der Akademie zu Berlin* (1885), 633–639 and 789–805.

[53] WIDLUND, O. B. A note on unconditionally stable linear multistep methods. *BIT 7* (1967), 65–70.

[54] WILLIAMS, J. Existence and uniqueness of solutions of the algebraic equations in the BDF methods. Tech. Rep. Numerical Analysis Report No. 272, University Of Manchester, 1995.

[55] YAMAMURA, K. An efficient algorithm for finding all solutions of piecewise-linear resistive circuits. *IEEE Transactions on Circuits and Systems 39* (1992), 213–221.

[56] YOON, J. L_p-error estimates for "shifted" surface spline interpolation on Sobolev space. *Mathematics of Computations 72* (2002), 1349–1367.

Index

Summary

Ordinary differential equations (ODEs) are present in almost every technical discipline. Electrical circuits, mechanical systems, control problems, physical and chemical processes, etc. are typically modelled by systems of ODEs. Many of these are autonomous. For example, in glass industry the common ODE problem is a deforming material blob, where one needs to obtain the boundary evolution of the molten glass. To solve such a problem, one needs to obtain the time evolution of a continuum of solutions, which is called a flow. Another example is coming from electrical networks where one is interested in the circuit behaviour under the influence of various initial values or variations of components parameters. The solution is again a flow, which needs to be obtained on a certain time interval.

Although the theory is extensive, solving a flow problem cannot be seen as a trivial task in general. Since not all ODEs can be solved analytically, the solution has to be obtained numerically. To do so one needs to perform a proper time discretisation. This discretisation is done by a numerical method for solving ODEs. Numerical methods are numerous and their application area depends on the nature of the problem. How to choose an appropriate numerical method for a particular problem depends on dimensionality, requested accuracy, stiffness, preservation of volume, energy, etc. Often one needs to use an implicit method, like for solving so-called stiff problems or problems where the preservation of the volume is an essential issue. Implicit methods introduce an additional problem in solving (non)linear system of equations at every time-level. To solve such a system, one needs to use Newton's iterative method. This is not a straightforward matter as the iteration function is not known explicitly.

In this thesis a new method is constructed based on existing implicit methods. To preserve favourable properties of these methods and avoid iteration, we apply inverse interpolation. Inverse interpolation turns out to be a powerful tool for solving nonlinear systems coming from discretised ODEs. Moreover, the interpolation theory is rich and a variety of interpolation methods can be used in these settings. The natural choice is for a method which is not computationally expensive, like piecewise linear interpolation. This method is easy to implement and does not require regular grids.

The method discussed in this thesis is referred to as the flow method. It can be based on any existing implicit integration method. The most interesting ones are linear multistep methods and Runge-Kutta methods that are used for solving stiff problems; in particular, BDF and methods based on Gaussian quadrature, such as Gauss, Radau and Lobatto, which are $A(\alpha)$-stable. The flow method inherits this property and gives similar results. The method can also be constructed based on

so-called symplectic methods, which are used for solving Hamiltonian problems. Some of these methods, like the midpoint rule, are implicit and introduce similar implementation difficulties. In problems, where the volume needs to be preserved during the computational time, the implementation of the flow method shows that this can be done with high accuracy even for large time scales.

The accuracy of the flow method depends on two sources of errors. The first one arises from discretisation and the second one from interpolation. The overall error should be controlled so that both errors are the same order. This can be achieved by choosing a proper time discretisation and a proper interpolation method.

A number of numerical examples illustrates the potential of the method. It is also shown that the flow method can be applied successfully for problems where the explicit time dependency is present via a forcing term only.

Samenvatting

Gewone differentiaal vergelijkingen komen in bijna elke discipline voor. Elektrische circuits, mechanische systemen, sturingsproblemen, scheikundige en natuurkundige processen enz. worden typisch gemodelleerd door stelsels van differentiaalvergelijkingen. Veel van deze systemen zijn autonoom. Een probleem, dat bijvoorbeeld voorkomt in de glasindustrie, is het berekenen van de voortgang van de positie van de rand van een vervormende glazen bel. Om zo'n probleem op te lossen, moet de tijdsevolutie van een geheel aan oplossingen worden verkregen, wat een stroming wordt genoemd. Een ander, vergelijkbaar voorbeeld kan worden gevonden bij de bestudering van elektrische netwerken, waarbij men geïnteresseerd is in het gedrag van het circuit onder invloed van verschillende beginvoorwaarden of variaties in de parameters van de componenten. De oplossing is weer de stroming die op een bepaald tijdsinterval bepaald dient te worden. Deze problemen worden dan ook wel stromingsproblemen genoemd.

Hoewel de theorie uitgebreid is, is het oplossen van een stromingsprobleem doorgaans verre van triviaal. Omdat niet alle differentiaalvergelijkingen analytisch kunnen worden opgelost, wordt de oplossing doorgaans numeriek verkregen. Hiertoe dient men vooraleerst een deugdelijke discretisatie van tijd en ruimte uit te voeren. De tijdsdiscretisatie wordt hierbij gedaan door een numerieke methode voor het oplossen van gewone differentiaalvergelijkingen. Er bestaan veel van deze numerieke methoden; hun toepassing hangt af van de aard van het probleem. De keuze van een bepaalde methode hangt af van de dimensionaliteit, de benodigde nauwkeurigheid, de stijfheid, het behoud van volume of energie, enz. Ze komen in zowel expliciete als impliciete vorm voor. Als de stijfheid van het probleem of juist het behoud van volume een grote zorg zijn, moet doorgaans een impliciete methode gebruikt worden. Deze impliciete methoden introduceren helaas een aanvullend probleem, omdat ze vereisen voor elke tijdstap een stelsel niet-lineaire vergelijkingen op te lossen. Hiervoor wordt Newton's methode voor gebruikt, waarvan de convergentie helaas niet gegarandeerd kan worden. De situatie wordt verder bemoeilijkt als de functie, waarover geïtereerd wordt, niet expliciet bekend is.

In dit proefschrift wordt een nieuwe methode geconstrueerd gebaseerd op bestaande impliciete methoden. Om de gunstige eigenschappen van deze methoden te behouden maar iteratie te vermijden, passen we inverse interpolatie toe. Inverse interpolatie blijkt een krachtig gereedschap voor het oplossen van de niet-lineaire stelsels, die uit gediscretiseerde differentiaalvergelijkingen volgen. Bovendien is de theorie over interpolatie rijk, en kunnen talloze interpolatie technieken voor dit doel gebruikt worden. Voor de hand liggende keuzes zijn rekenkundig goedkope methoden, zoals bijvoorbeeld stuksgewijs lineaire interpolatie. Deze laatste is gemakkelijk

te implementeren en heeft geen regulier rooster nodig. Om echter nauwkeurigere resultaten te behalen, kunnen ook hogere-orde methoden toegepast worden. Voorbeelden van zulke methoden zijn stuksgewijs derdegraads interpolatie, oppervlaktespline interpolatie, en biharmonische spline interpolatie.

De stromingsmethode kan worden gebaseerd op elke willekeurige bestaande impliciete numerieke methode. De interessantste zijn lineaire meerstapsmethoden en Runge-Kutta-methoden voor stijve problemen; in het bijzonder BDF en methoden, die zijn afgeleid van Gausse kwadratuur, zoals Gauss, Radau, en Lobatto. Deze methoden zijn $A(\alpha)$-stabiel en worden vaak gebruikt om stijve problemen op te lossen. De stromingsmethode erft deze eigenschap en geeft vergelijkbare resultaten. De stromingsmethode kan ook worden afgeleid waarbij deze gebaseerd wordt op zogenaamde symplektische methoden, die gebruikt worden om Hamiltonse problemen op te lossen. Enkele van deze methoden, zoals de impliciete midpuntsregel, zijn impliciet en introduceren vergelijkbare moeilijkheden bij implementatie.

Voor problemen waarbij het volume gedurende de berekening behouden moet blijven, toont onze implementatie van de stromingsmethode aan, dat zelfs voor heel grote tijdsschalen hoge nauwkeurigheid behaald kan worden. De nauwkeurigheid van de stromingsmethode hangt van twee factoren af: ten eerste van de discretisatiefout in de eerdergenoemde onderliggende methoden; ten tweede van de interpolatiefout. De gezamenlijke fout dient dusdanig gecontroleerd te worden dat beide samenstellende fouten van dezelfde orde zijn. Dit gebeurt door de juiste discretisatie-methode bij de juiste interpolatie-methode te kiezen.

Numerieke voorbeelden laten zien dat de hier afgeleide methode veel potentieel biedt voor het oplossen van autonome stromingsproblemen. Bovendien is ze even succesvol voor de klasse van niet-autonome problemen, waarbij de tijdsafhankelijkheid van het snelheidsveld wordt weergegeven via de krachtsterm. Het belangrijkste voordeel van de stromingsmethode is dat het gebruikt kan worden voor problemen waar het snelheidsveld alleen in discrete punten bekend is.

CVRRICVLVM VITÆ

The author was born in Niš, Serbia and Montenegro, on January 26th 1971. After completing pre-university schooling at the Gymnasium "Svetozar Marković" in Niš in 1990 and a year of a regular military service, in 1991 he started M.Sc. studies in Automatic Control in Electrical Engineering at Faculty of Electronic Engineering, University of Niš, Serbia and Montenegro. After finishing his graduation project *Application of Nonlinear Techniques in Control of Induction Motor Coordinates* in 1998, he became a research assistant and a postgraduate student at the Department of Automatic Control, Faculty of Electronic Engineering, University of Niš. His specialization subject was *Control of processes and electromechanical systems*. In 2000 he moved to Eindhoven, The Netherlands, where he started the PhD research discussed in this dissertation at the Scientific Computing Group of the Eindhoven University of Technology. Since October 2004, he is employed by MICHAEL BAILEY ASSOCIATES LTD. as a Mathematic Software Engineer and works on a project of ED&T Group, which is part of the company PHILIPS.

The Road Goes Ever On

The Road goes ever on and on
Out from the door where it began.
Now far ahead the Road has gone,
Let others follow it who can!
Let them a journey new begin,
But I at last with weary feet
Will turn towards the lighted inn,
My evening-rest and sleep to meet.

by J. R. R. Tolkien

Stellingen

behorende bij het proefschrift

Numerical Methods for Solving ODE Flow

van

Bratislav Tasić

I

A stiff ODE problem can successfully be solved without use of Newton's iterative method. This can be achieved by means of inverse interpolation instead.

II

The flow method can be applied to a variety of flow problems in which the velocity field is not known explicitly. Problems as such can be found in glass production, electrical networks and mechanical systems.

III

In solving flow problems it is important that numerical integral curves do not intersect. In this respect the BDF methods of odd order behave better than the BDF methods of even order. The same holds for the Runge-Kutta methods of Radau IA, Lobatto IIIB and Lobatto IIIC type as compared to Gauss, Radau IIA and Lobatto IIIA methods.

IV

The error constant C, present in the error bound of piece-wise linear interpolation on a simplex Γ, reads

$$C = \begin{cases} \frac{1}{4}\,\frac{N}{N+1}\,\Delta x^2, & \mathbf{c} \in \Gamma, \\ \frac{1}{4}\,\frac{N-1}{N}\,\Delta x^2, & \mathbf{c} \notin \Gamma, \end{cases}$$

where $\mathbf{c}$ is the centre of the circumscribed sphere around the simplex, Δx is the diameter and N is the dimensionality of the simplex.

V

The increasing number of allergy problems within the human population (currently around 20% in developed countries) is related to the once highly spread helminth (parasitic worms) infections. Long time ago people were more affected by these infections and, according to the law of evolution, persons with a stronger immune mechanism lived longer and had more descendants. Today these diseases are rare and the mechanism is counterproductive since it reacts to various substances in human surroundings.

Hypothesis of L. LICHTENSTEIN ET AL, *University John Hopkins, Baltimore, USA.*

VI

Soon there will be "a new kind of webmail, built on the idea that you should never have to delete mail and you should always be able to find the message you want" (announced by the Company Google). Although an interesting approach, it may cause difficulties in information readability after a long time period due to an extensive amount of unnecessary data.

VII

Electronic paper display modules may completely substitute all present paper media. It may represent a solution for saving forests destroyed for paper production, but it will also shrink our home libraries to just a few e-Book readers.

First-Generation Electronic Paper Display from Philips, Sony and E-Ink to Be Used in New Electronic Reading Device, Philips Research Press Release, March 2004.

VIII

At first sight nature finds strange equilibria sometimes. A life cycle of Plasmodium species (cause of malaria) starts in a mosquito, continues in a human, where it must reach the liver within 30 minutes after a bite and then go to the blood destroying blood-cells. Finally, the reproduction cycle must end again in a mosquito. According to the World Health Organisation over 20 million people are infected and over a million die every year from malaria. If the Plasmodium life cycle was just slightly less complex, the whole species would cease to exist since there would be no more humans to be infected.

W.J. BECK AND E.J. DAVIES, *Medical Parasitology,* CV Mosby Company, 1981.

IX

The phrase "what works should not be changed", based on economical reasons, will cause that a lot of old, yet still actively used, software will remain unchanged in time. This reminds of the evolution principle in nature, where many primitive organisms are still present in their original form, although they were created millions of years ago.

X

All science and art disciplines are based on certain rules and therefore they can be expressed in some mathematical form.

"No human investigation can be called real science if it cannot be demonstrated mathematically" - LEONARDO DA VINCI.

XI

No matter how necessary it may seem sometimes, the use of global parameters should be avoided; especially in programming and politics.

XII

It is better to have even a wrong opinion than not to have one at all. Opinions are products of the mind's activity, while the lack of them are signs of laziness. For defending someone else's opinion the opposite holds.

XIII

Human emotions are directly proportional to perturbations of dynamical steady states of a human mind. Frequent appearances and often rapid behaviour of these perturbations are the main reason why it is so difficult to model and predict the thinking process. However, it is almost certain that the effect of emotions on the state of mind has a gradient nature.

XIV

Since all over the world millions of people have been involved in scientific work, it is not easy to obtain a new and original idea. Fortunately, nature still has to offer an indefinite number of mysteries to be explained.

"I would rather discover one scientific fact than become King of Persia" - DEMOCRITUS.

XV

The only three ways to reach a state of the complete happiness are stupidity, ignorance and love.