Contents

Preface

The Richardson Extrapolation is a very powerful and popular numerical procedure, which can efficiently be used in the efforts to improve the performance of programs by which large time-dependent scientific and engineering problems are handled on computers. It was introduced in **Richardson (1911, 1927)** and after that has been used numerous times by many scientists and engineers in the treatment of their **advanced large-scale mathematical models**. Different phenomena that arise in science and engineering are described and can successfully be studied by applying such models. In most of the applications, this technique was until now used primarily in the efforts either to improve the accuracy of the model results or to evaluate and check the magnitude of the computational errors (and, thus, to control automatically the time-stepsize in an attempt to achieve easier the required accuracy and to increase the efficiency of the computational process).

Three important items are **always** strongly emphasized when the application of the Richardson Extrapolation is discussed. The first of these important issues is the following:

> **(1)** One must accept a substantial increase of the number of the simple arithmetic operations **per time-step** when this approach is selected.

The increase of the number of simple arithmetic operations per time-step is indeed rather considerable, but the extra computations per time-step can be compensated in most of the cases

> **(2)** by exploiting the possibility to achieve the same degree of accuracy by applying **a considerably larger time-stepsize** during the computations (decreasing in this way the number of time-steps)

and/or

> **(3)** by the fact that the accuracy and, thus, also the time-stepsize, **can be controlled** in a very reliable manner.

In this book, we shall additionally describe **two other very important but in some cases extremely difficult issues.** The efficiency of this device could be considerably improved or, at least, some of the problems related to the implementation of the Richardson Extrapolation can successfully be avoided when these two issues are properly handled:

> **(4)** It is necessary to emphasize strongly the fact that **a fourth very important item**, the numerical stability of the results

that are calculated by **the new numerical method** (the method which is obtained when the Richardson Extrapolation is implemented), is also very important. Therefore, it is necessary to study very carefully the stability properties of the different implementations of the Richardson Extrapolation when these implementations are combined with various numerical methods.

(5) If very large time-dependent mathematical models are to be treated numerically, different splitting procedures have very often to be introduced and used; see, for example, **Faragó, Havasi and Zlatev (2010)**, **Geiser (2008)**, **Marchuk (1968)**, **Strang (1968)**, **Zlatev (1995)**, **Zlatev and Dimov (2006)** and **Zlatev, Faragó and Havasi (2010)**. Therefore, it is worthwhile to study the application of the Richardson Extrapolations and the stability of the resulting new methods also in the case where some kind of splitting is applied.

The last **two items**, the stability properties of the resulting new numerical methods and the introduction of the Richardson Extrapolation in connection with some splitting procedures, will also be discussed in the present book. Especially, the first of the last two important issues will be a major topic of the further presentation.

The mentioned above five fundamental topics that are related to the Richardson Extrapolation (RE) are readily summarized in **Table 1**.

No.	Property of the underlying method	What happens if RE is used?	Does that always happen?
1	Accuracy	It becomes higher	Yes
2	Arithmetic operations per time-step	They are increased	Yes
3	Stepsize control	It is always possible	Yes
4	Preservation of the stability	It is not clear in advance	It should always be studied
5	Combination with splitting	Not very clear	One must be careful

Table 1

Properties of the resulting new numerical methods obtained when Richardson Extrapolation (RE) is used. Some desired properties of the combination with the Richardson Extrapolation (in comparison with the properties of the underlying numerical method) are listed in the second column. The results of using the Richardson Extrapolation (again in comparison with the underlying method) are shown in the third column. The answers to the questions raised in the fourth column emphasize the fact that problems may arise in connection with the last two properties, while the situation is very clear when the first three properties are considered.

Several important conclusions can be drawn by studying carefully the results presented in **Table 1**:

(A) If only **the first three properties** are considered (which is very often, in fact nearly always, the case), then the application of the Richardson Extrapolation is very straight-forward and everything would be extremely simple **when the fourth and the fifth items are not causing troubles**. It is indeed quite true that the number of arithmetic operations is always exceeding if the Richardson Extrapolation is used, but a very good compensation can be achieved because the resulting new method is more accurate and the number of time-steps can be reduced very significantly by increasing the time-stepsize (under the assumption, not always clearly stated, that the stability will not cause any problems). Furthermore, it is possible to control the time-stepsize and, thus, to select optimal time-stepsize at every time-step.

(B) The truth is, however, that, it is necessary to ensure stability of the computational process. This could cause very serious difficulties. For example, the well-known Trapezoidal Rule is a rather reliable numerical procedure (it is A-stable and, therefore, in the most of the cases the computational process will remain stable for large values of the time-stepsize). Unfortunately, the combination of this numerical algorithm with the Richardson Extrapolation results in a new numerical method, which is **not** stable. This fact has been established in **Dahlquist (1963)**. It has been proved in **Zlatev, Faragó and Havasi (2010)** that some other representatives of the class of the $\boldsymbol{\theta}$–methods (the Trapezoidal Rule is belonging to this class and can be obtained for $\boldsymbol{\theta = 0.5}$) will also become unstable when these are combined with the Richardson Extrapolation. These results are very bad, but good results can also be obtained. It was shown in **Zlatev, Georgiev and Dimov (2014)** that the combination of **any** Explicit Runge-Kutta Method with the Richardson Extrapolation results in a new algorithm, which has better absolute stability properties than those of the underlying method when the order of accuracy $\mathbf{p}$ is equal to the number of stages $\mathbf{m}$. These two facts show very clearly that the situation is indeed not clear and it is necessary to investigate very carefully the stability properties of the combination of the Richardson Extrapolation with the selected numerical method.

(C) The situation becomes more complicated when some splitting procedure has additionally to be implemented and used. Results proved in **Zlatev, Faragó and Havasi (2012)** indicate that also in this situation the stability of the

computational process could play an important role and, therefore, it should be studied very carefully.

Taking into account the last two conclusions, we decided to concentrate our efforts on **the difficulties,** which may arise when the Richardson Extrapolation is used. In our opinion, this is worthwhile, because these difficulties are very often (if not always) underestimated when this device is commented and its use is advocated.

The book consists of **eight** chapters and the contents of these chapters can be sketched in the following way:

Chapter 1 contains the definition of the Richardson Extrapolation for an **arbitrary** time-dependent numerical method for solving systems of ordinary differential equations (ODEs). The basic properties of this computational technique are explained there and several introductory remarks are given. Two different implementations of the Richardson Extrapolation as well as the advantages and the disadvantages of the application of any of these two implementations are also discussed in this chapter. Finally, the fact that it is necessary to develop and to use numerical methods, which have **good stability properties** when these are combined with the Richardson Extrapolation, is strongly emphasized.

The application of the Richardson Extrapolation in connection with Explicit Runge-Kutta Methods (ERKMs) for solving systems of ordinary differential equations is discussed in **Chapter 2**. It is shown there that for some classes of Explicit Runge-Kutta Methods the application of the Richardson Extrapolation results **always** in new numerical methods with **better** stability properties. Any of these new methods is the combination of the selected numerical algorithm (in this particular case, the underlying Explicit Runge-Kutta Method) with the Richardson Extrapolation. The most important and very useful for the practical applications fact is that, as stated above, the new numerical methods, which are obtained by applying additionally the Richardson Extrapolation, have **bigger** absolute stability regions than those of the underlying classical Explicit Runge-Kutta Methods. This is very important, because if the absolute stability regions are bigger, then the Richardson Extrapolation can be used with bigger stepsizes also in the cases where the restrictions due to the necessity to preserve the stability are dominating over the requirements for achieving better accuracy. Numerical examples are given in order to show how the improved stability properties can be utilized in order to achieve better efficiency of the computational process.

Linear multistep methods and predictor-corrector methods (including here predictor-corrector methods with several different correctors as those developed in **Zlatev, 1984**) can successfully be used when large mathematical models are handled. The application of such methods in the numerical solution of systems of ordinary differential equations is considered in **Chapter 3**. The basic properties of these methods are described. It is explained why the use of the Richardson Extrapolation in connection with the linear multistep methods is at least difficult. However, another approach, the use of carefully chosen predictor-corrector schemes, can successfully be applied. It is shown that the result achieved by using different predictor-corrector schemes is very similar to the result achieved by using the Richardson Extrapolation. Again, as in the second chapter, we are interested first and foremost in the absolute stability properties of the resulting combined predictor-corrector methods.

The combination of some **implicit** numerical methods for solving stiff systems of ordinary differential equations and the Richardson Extrapolation is the main topic of the discussion in **Chapter 4**. Implicit

numerical methods are as a rule much more expensive with regard to the computational complexity than the explicit methods, because they lead, normally in an inner loop of the well-known Newton iterative procedure (or some modifications of this procedure), to the solution of systems of linear algebraic equations, which can sometimes be very large (containing many millions of equations). Therefore, such numerical methods are mainly used when the system of ordinary differential equations that is to be treated is indeed **stiff** and, because of this, it is very inefficient to apply explicit numerical methods. The absolute stability properties of the numerical methods, which are the main topic of the discussion in the previous two chapters, are as a rule not sufficient in the efforts to achieve good efficiency when stiff problems must be solved. Therefore, it is necessary to use several more restrictive definitions related to the stability of the numerical methods for solving stiff systems of ordinary differential equations. Such definitions (A-stability, strong A-stability and L-stability; see **Burrage, 1995**, **Butcher, 2003**, **Dahlquist, 1963**, **Dahlquist, Björck and Anderson, 1974**, **Hairer, Nørsett and Wanner, 1987**, **Hairer and Wanner, 1991**, **Hundsdorfer and Verwer, 2003**, **Lambert, 1991**) are introduced in the fourth chapter and used consistently in it. After the introduction of the three more restrictive stability definitions, the selection and the application of suitable implicit numerical methods for solving stiff systems of ordinary differential equations possessing good stability properties is discussed. The additional treatment of the Richardson Extrapolation may cause also in this case extra problems (because even if the underlying numerical method has the needed stability properties, then the combined method may become unstable). Therefore, it is necessary to investigate the preservation of the A-stability (as well as the preservation of the other and stronger stability properties) of the chosen implicit numerical method in the essential for our study case where Richardson Extrapolation is additionally used. These important issues are carefully investigated in the fourth chapter. Some theoretical results are proved there and the usefulness of the obtained results is illustrated by performing a series of calculations related to several appropriate numerical examples (arising when badly scaled, extremely ill-conditioned and very stiff atmospheric chemical schemes implemented in many well-known large-scale air pollution models are to be handled).

The performance of the Richardson extrapolation in connection with some splitting procedures is described in **Chapter 5**. It is stressed there that, while splitting procedures are commonly used in the treatment of large-scale and time-consuming simulations that arise in many scientific and engineering fields, the topics of combining these splitting techniques with the Richardson Extrapolation and of investigating the stability properties of the resulting new methods have practically not been carefully studied until now in the literature. An exception is the paper of **Zlatev, Faragó and Havasi (2012)**.

The implementation of the Richardson Extrapolation in connection with some classes of systems of partial differential equations is demonstrated in **Section 6**. It is explained there why in many cases it is much more difficult to use the Richardson Extrapolation for systems of partial differential equations than to apply it in the treatment of systems of ordinary differential equation. In the sixth chapter we discuss in detail a particular application of the Richardson Extrapolation in connection with some linear advection problems, arising, for example, in air pollution modelling, but also in some other areas of fluid dynamics. The development of reliable combined methods (advection equations plus Richardson Extrapolation), which have improved accuracy properties, is also described there. The usefulness of the results obtained in this chapter in the attempts to improve the accuracy of the calculated approximations is demonstrated by appropriate numerical examples.

Time-dependent problems are treated in the first six chapters. However, the Richardson Extrapolation can also be used in the solution of algebraic or transcendental equations or systems of such equations by iterative methods. In the latter case the Richardson Extrapolation can successfully be applied in

an attempt **(a)** to accelerate the speed of the convergence and/or **(b)** to reduce the number of iterations that are needed to achieve a prescribed accuracy. Some results about the application of the Richardson Extrapolation in the treatment of algebraic and transcendental equations or systems of such equations by iterative methods will be presented in **Chapter 7**. The application of the Richardson Extrapolation in numerical integration is also discussed in the seventh chapter.

Some general conclusions and remarks are presented in the last chapter of the book, in **Chapter 8**. The following issues (which are closely related to the presentation given in the previous seven chapters) are summarized and emphasized once again in the last chapter of the monograph:

(a) This book is written primarily for scientists and engineers. It will also be useful for PhD students. Finally, it can be applied in some appropriate courses for students. Therefore, we always try, by using the great experience obtained by us during our long cooperation with physicists and chemists, to make the text **easily understandable** also for non-mathematicians. We shall, of course, give all the proofs that are needed for the presentation of the results, but the truth is that we are much more interested in emphasizing the important fact that the methods discussed in this book can easily and efficiently be applied in **real-life computations**.

(b) In the whole book it is pointed out, many times and in a very strong way, that both the accuracy and the error control are undoubtedly very important issues. However, the necessity to achieve sufficiently good accuracy and/or reliable error estimates is, as was pointed out above, not the only issue, which must be taken into account when the Richardson Extrapolation is used. In many cases, it is also very important to select the right numerical algorithms, such numerical algorithms, which will produce new efficient numerical methods with **good stability properties** when they are combined with the Richardson Extrapolation.

(c) There are two ways of implementing the Richardson Extrapolation: **passive and active**. It is not clear in advance which of the two options performs better in a given particular situation. The performance depends on several factors; the stability properties of the combined numerical method (the underlying numerical method plus the Richardson Extrapolation) being very essential. Therefore, the decision is not easy, at least not always easy, and one must be careful in the choice of one of these two implementations. Some recommendations related to this important decision are given in the last chapter of the book.

It should also be mentioned, once again, that the relatively simple and easily understandable examples describing how to implement efficiently the Richardson Extrapolations can successfully be used also in much more complicated environments in the efforts to improve the calculated results from different advanced and very complicated mathematical models. This is demonstrated in the book by giving

some examples that are related to several modules of large-scale environmental models, which are taken from **Zlatev (1995)** or **Zlatev and Dimov (2006)**.

It is important to emphasize strongly the fact that the Richardson Extrapolation is indeed a very reliable and robust technique. However, at the same time it must also be stressed that **this is true only if and when it is correctly applied** for the particular problems, which are to be handled. The implementation of this procedure seems to be very simple and straight-forward. However, the truth is that achieving high efficiency and good results is by far not always an easy task. The convergence and especially the stability properties of the new numerical method (consisting of the combination of the chosen algorithm and the Richardson Extrapolation) should be carefully studied and well understood. This is absolutely necessary, because the new numerical method will become very efficient and reliable only if and when such an investigation is properly done or if and when the implementation of the Richardson Extrapolation is based on well-established knowledge related to the properties of both the selected underlying numerical algorithm and the new numerical method obtained when one attempts to enhance the efficiency of the underlying method by additionally applying the Richardson Extrapolation.

We hope that this book will help very much the readers to improve the performance of the Richardson Extrapolation in the solution of their particular tasks and we must also emphasize that we do believe that this book will be very useful for many specialists working with large-scale mathematical models arising in different areas of science and engineering.

Moreover, it is worthwhile to mention here that

the first five chapters of the book contain different methods for solving systems of ordinary differential equations (ODEs) and can be used in a short course on numerical treatment of such systems for non-mathematicians (scientists and engineers).

Some readers will be interested in reading only selected chapters of the book. For example, people, who are interested in applying the Richardson Extrapolation together with some splitting techniques, will prefer to start by reading the fifth chapter without studying carefully the previous four chapters. We tried to facilitate the attempts to go directly to the most interesting for a particular reader part of the book by making each chapter relatively independent from the others. In order to achieve this, it was necessary to repeat some issues. We do believe that this approach would be useful in many situations.

Chapter 1

Basic Properties of the Richardson Extrapolation

The basic principles, on which the implementation of the Richardson Extrapolation in the numerical treatment of the important class of initial value problems for systems of ordinary differential equations (ODEs) is based, are discussed in this chapter. It must immediately be stressed that this powerful approach can also be applied in the solution of systems of partial differential equations (PDEs). In the latter case, the device is often applied after the semi-discretization of the systems of PDEs. When the spatial derivatives are discretized, by applying, for example, finite differences or finite elements, the system of PDEs is transformed into a system of ODEs, which is as a rule very large. Then the application of the Richardson Extrapolation is very straight-forward, since it is based on the same rules as those used in the implementation of the Richardson Extrapolation for systems of ODEs, and this simple way of implementing this devise is used very often, but its success is based on an assumption that the selected discretization of the spatial derivatives is sufficiently accurate and, therefore, the errors resulting from the spatial discretization are very small and will not interfere with the errors resulting from the application of the Richardson Extrapolation in the solution of the semi-discretized problem. If this assumption is satisfied, then the results will in general be good, but problems will surely arise when the assumption is not satisfied. Then the discretization errors caused by the treatment of the spatial derivatives must also be taken into account and the strict implementation of Richardson Extrapolation for systems of PDEs will become considerably more complicated than that for systems of ODEs. Therefore, the direct application of the Richardson Extrapolation in the computer treatment of systems of PDEs deserves some special treatment. This is why only the application of the Richardson Extrapolation in the case where systems of ODEs are handled numerically is studied in the first five chapters of this book, while the description of the direct use of the Richardson Extrapolation for systems of PDEs is postponed and will be presented in Chapter 6.

The contents of the first chapter can be outlined as follows:

The initial value problem for systems of ODEs is introduced in **Section 1.1.** It is explained there when the solution of this problem exists and is unique. The assumptions, which are to be made in order to ensure existence and uniqueness of the solution, are in fact not very restrictive, but it is stressed that some additional assumptions must be imposed when accurate numerical results are needed and, therefore, numerical methods for treatment of the initial value problems for systems of ODEs that have **high order of accuracy** are to be selected and used. It must also be emphasized here that in Section 1.1 we are sketching only the main ideas. No details about the assumptions, which are to be made in order to ensure existence and uniqueness of the solution of initial value problems for systems of ODEs, are needed, because this topic is not directly connected to the application of Richardson Extrapolation in conjunction with different numerical methods for solving such problems. However, references to several text books, where such details are presented and discussed, are given.

Some basic concepts that are related to the application of an **arbitrary** numerical method for solving initial value problems for systems of ODEs are briefly described in **Section 1.2.** It is explained there that the computations are as a rule carried out **step by step** at the grid-points of some set of values of

the independent variable (which is the time-variable in many engineering and scientific problems) and, furthermore, that it is possible to apply both constant and variable stepsizes (the name time-stepsizes will often be used). A very general description of the basic properties of these two approaches for solving approximately initial value problems for systems of ODEs is presented and the advantages and the disadvantages of using constant or variable time-stepsizes are discussed.

The Richardson Extrapolation is introduced in **Section 1.3.** The ideas are very general and the application of the Richardson Extrapolation in connection with an **arbitrary** one-step numerical method for solving approximately initial value problems for systems of ODEs is described. The combination of the Richardson Extrapolation with **particular** numerical methods is studied in the next chapters of this book.

The important (for improving the performance and for obtaining greater efficiency) fact that the accuracy of the computed results is increased, when the Richardson Extrapolation is implemented, is explained in **Section 1.4.** More precisely, it is shown there that if the order of accuracy of the selected numerical method for solving initial value problems for systems of ODEs is $\mathbf{p}$, where $\mathbf{p}$ is some positive integer with $\mathbf{p} \geq \mathbf{1}$, then the application of the Richardson Extrapolation results in **a new numerical method**, which is normally of order $\mathbf{p+1}$. This means that the order of accuracy of the new numerical method, the combination of the selected algorithm for solving initial value problems for systems of ODEs and the Richardson Extrapolation, is as a rule increased by one.

The possibility of obtaining an error estimation of the accuracy of the calculated approximations of the exact solution (in the case where the Richardson Extrapolation is additionally used) is discussed in **Section 1.5.** It is explained there that the obtained error estimation could be used in the efforts to control the time-stepsize and to select, in principle at every time-step, time-stepsizes in an optimal way (which is important when variable stepsize numerical methods for solving initial value problems for systems of ODEs are to be applied in the computational process).

The drawbacks and the advantages of the application of Richardson Extrapolation are discussed in **Section 1.6.** It is demonstrated there, with carefully chosen examples arising in an important for the modern society problem (in the application of air pollution modelling to study situations, which can lead to damages for plants, animals and human beings), that the stability of the calculated results is a very important issue and the need of numerical methods with good stability properties is again emphasized in this section.

Two implementations of the Richardson Extrapolation are presented in **Section 1.7**. Some recommendations are given there in connection with the choice, in several different situations, of the better one of these two implementations.

The possibility of achieving even more accurate results is discussed in **Section 1.8**. Assume that the order of the underlying method is $\mathbf{p}$. It is shown in Section 1.8 how to achieve order of accuracy $\mathbf{p+2}$ when the Richardson Extrapolation is additionally applied.

Some general conclusions are drawn and listed in **Section 1.9**.

Research problems are proposed in the last section, **Section 1.10**, of the first chapter.

1.1. Introduction of the initial value problem for systems of ODEs

Initial value problems for systems of ODEs appear very often when different phenomena arising in many areas of science and engineering are to be described mathematically and after that to be treated numerically. These problems have been studied in detail in many monographs and text-books in which the numerical solution of systems of ODEs is handled; as, for example, in **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Hairer and Wanner (1991)**, **Henrici (1968)**, **Hundsdorfer and Verwer (2003)** and **Lambert (1991)**.

The classical initial value problem for systems of ODEs is as a rule defined in the following way:

$$(1.1) \qquad \frac{dy}{dt} = f(t,y), \quad t \in [a,b], \quad a < b, \quad y \in \mathbb{R}^s, \quad s \geq 1, \quad f \in D \subset \mathbb{R} \times \mathbb{R}^s,$$

where

(a) t is the independent variable (in most of the practical problems arising in physics and engineering it is assumed that t is the time variable and, therefore, we shall mainly use this name in our book),

(b) s is the number of equations in the system (1.1),

(c) f is a given function defined in some domain $D \subset \mathbb{R} \times \mathbb{R}^s$ (it will always be assumed in this book that f is a **one-valued** function in the whole domain D)

and

(d) $y = y(t)$ is a vector of dimension s that depends of the time-variable t and represents the unknown function (or, in other words, this vector is the dependent variable and represents the unknown exact solution of the initial value problem for systems of ODEs).

It is furthermore assumed that the initial value

$$(1.2) \qquad y(a) = \eta$$

is a given vector with s components.

It is well-known that the following theorem, which is related to the exact solution of the problem defined by (1.1) and (1.2), can be formulated and proved (see, for example, **Lambert, 1991**).

Theorem 1.1: A continuous and differentiable solution $\mathbf{y}(\mathbf{t})$ of the initial value problem for systems of ODEs that is defined by (1.1) and (1.2) exists and is unique if the right-hand-side function $\mathbf{f}$ is continuous in the whole domain $\mathbf{D}$ and if, furthermore, there exists a positive constant $\mathbf{L}$ such that the following inequality is satisfied:

$$(\mathbf{1.3}) \quad \| \mathbf{f}(\mathbf{t}, \bar{\mathbf{y}}) - \mathbf{f}(\mathbf{t}, \tilde{\mathbf{y}}) \| \leq \mathbf{L} \, \| \bar{\mathbf{y}} - \tilde{\mathbf{y}} \|$$

for any two points $(\mathbf{t}, \bar{\mathbf{y}})$ and $(\mathbf{t}, \tilde{\mathbf{y}})$ from the domain $\mathbf{D}$.

■

Definition 1.1: Every constant $\mathbf{L}$, for which the above inequality is fulfilled, is called the Lipschitz constant and it is said that function $\mathbf{f}$ from (1.1) satisfies the **Lipschitz condition** with regard to the dependent variable $\mathbf{y}$ when (1.3) holds.

■

It can be shown that the assumptions made in Theorem 1.1 provide only sufficient but not necessary conditions for existence and uniqueness of the exact solution $\mathbf{y}(\mathbf{t})$ of (1.1) – (1.2). For our purposes, however, the result stated in the above theorem is quite sufficient. Moreover, there is no need **(a)** to go into details here and **(b)** to prove Theorem 1.1. Any of these two actions is beyond the scope of this monograph, but the above theorem as well as many other results that are related to the existence and the uniqueness of the solution $\mathbf{y}(\mathbf{t})$ of the problem defined by (1.1) and (1.2) are proved in many text-books, in which initial value problems for systems of ODEs are studied. As an example, it should be pointed out that many theorems dealing with the existence and/or the uniqueness of the solution of initial value problems for systems of ODEs are formulated and proved in **Hartmann (1964)**.

It is worthwhile to conclude this section with several remarks.

Remark 1.1: The requirement for existence and uniqueness of a solution $\mathbf{y}(\mathbf{t})$ of the system of ODEs that is imposed in Theorem 1.1 is stronger than the requirement that the right-hand-side function $\mathbf{f}$ is continuous for all points $(\mathbf{x}, \mathbf{y})$ from the domain $\mathbf{D}$, because the Lipschitz condition (1.3) must additionally be satisfied. On the other side, this requirement is weaker than the requirement that function $\mathbf{f}$ is continuously differentiable for all points $(\mathbf{x}, \mathbf{y})$ from the domain $\mathbf{D}$. This means that the requirement made in Theorem 1.1 for the right-hand function $\mathbf{f}$ is a little stronger than continuity, but a little weaker than differentiability.

■

Remark 1.2: If the right-hand-side function $\mathbf{f}$ is continuously differentiable with regard to all values of the independent variable $\mathbf{t}$ and the dependent variable $\mathbf{y}$, i.e. in the whole domain $\mathbf{D}$, then the requirement imposed by (1.3) can be satisfied by the following choice of the Lipschitz constant:

$$(1.4) \quad L = \sup_{(t,y)\in D} \left\| \frac{\partial f(t,y)}{\partial t} \right\| .$$

■

Remark 1.3: The problem defined by (1.1) and (1.2) is as a rule called non-autonomous (the right-hand-side of the non-autonomous problems depends both on the dependent variable y and on the independent variable t). In some cases, especially in many proofs of theorems, it is more convenient to consider **autonomous** problems. The right-hand-side f does not depend **directly** on the time-variable t when the problem is autonomous. This means that an autonomous initial value problem for solving systems of ODEs can be written as:

$$(1.5) \quad \frac{dy}{dt} = f(y), \quad t \in [a,b], \quad a < b, \quad y \in \mathbb{R}^s, \quad s \geq 1, \quad f \in D \subset \mathbb{R} \times \mathbb{R}^s, \quad y(a) = \eta .$$

Any non-autonomous problem can easily be transformed into autonomous by adding a simple extra equation, but it should be noted that if the original problem is scalar (i.e. if it consists of only one equation), then the transformed problem will not be scalar anymore. It will become a system of two equations. This fact might sometimes cause certain difficulties; more details can be found in **Lambert (1991)**.

It should be mentioned here that the results presented in this book are valid both for non-autonomous and autonomous initial value problems for systems of ODEs.

■

Remark 1.4: The problem defined by (1.1) and (1.2) contains only the first-order derivative of the dependent variable y . Initial value problems for systems of ODEs, which contain derivatives of higher order, also appear in many applications. Such problems will not be considered in this book, because these systems can easily be transformed into initial value problems of first-order systems of ODEs; see, for example, **Lambert (1991)**.

■

Remark 1.5: In the practical treatment of initial value problems for systems of ODEs it becomes normally necessary to introduce much more stringent assumptions than the assumptions made in Theorem 1.1 (especially when accurate numerical methods are to be applied in the treatment of these systems). This is due to the fact that numerical methods of order of accuracy $p > 1$ are nearly always used in the treatment of the problem defined by (1.1) and (1.2). Such numerical methods are often derived by expanding the unknown function y in Taylor series, truncating this series after some term,

say the term containing derivatives of some order $\mathbf{p} > \mathbf{1}$, and after that applying different rules to transform the truncated series in order to obtain numerical methods with some desired properties; see, for example, **Henrici (1968)**. By using (1.1), the derivatives of $\mathbf{y}$ can be expressed by derivatives of $\mathbf{f}$ and, when such procedures are applied, one can easily established that it is necessary to assume that all derivatives of function $\mathbf{f}$ up to order $\mathbf{p} - \mathbf{1}$ must be continuously differentiable. It is obvious that this assumption is in general much stronger than the assumption made in Theorem 1.1. In fact, if a requirement to find a reliable error estimation is additionally made, then it will as a rule be necessary to require that the derivative of function $\mathbf{f}$, which is of order $\mathbf{p}$, is also continuously differentiable. The necessity of introducing stronger assumptions will be further discussed to a certain degree in many sections of the remaining part of this book, however this problem is not directly related to the implementation of the Richardson Extrapolation and, therefore, it will not be treated in detail this book.

■

Remark 1.6: Only initial value problems for systems of ODEs will be studied in this book (i.e. no attempt to discuss the properties of boundary value problems for systems of ODEs will be made). Therefore, we shall mainly use the abbreviation "systems of ODEs" instead of "initial value problems for systems of ODEs" in the remaining sections of this chapter and also in the next chapters.

■

1.2. Numerical treatment of initial value problems for systems of ODEs

Normally, the system of ODEs defined by (1.1) and (1.2) could not be solved exactly. Therefore, it is necessary to apply some suitable numerical method in order to calculate sufficiently accurate approximate values of the components of the exact solution vector $\mathbf{y}(\mathbf{t})$ at the grid-points belonging to some discrete set of values of the time-variable. An example for such a set, which is often called computational mesh or grid, is given below:

$$(\mathbf{1.6}) \quad \mathbf{t_0 = a}, \quad \mathbf{t_n = t_{n-1} + h} \quad (\mathbf{n = 1, 2, \dots, N}), \quad \mathbf{t_N = b}, \quad \mathbf{h = \frac{b - a}{N}}.$$

The calculations are carried out **step by step**. Denote by $\mathbf{y_0}$ the initial approximation of the solution, i.e. the approximation at $\mathbf{t_0 = a}$. It is often assumed that $\mathbf{y_0 = y(a) = \eta}$. In fact, the calculations are started with the exact initial value when this assumption is made. However, the calculations can also be started by using some truly approximate initial value $\mathbf{y_0 \approx y(a)}$.

After providing some appropriate, exact or approximate, value of the initial condition of the system of ODEs, one calculates successively (by using some computing formula, which is called "numerical method") a sequence of vectors, $\mathbf{y_1 \approx y(t_1)}$, $\mathbf{y_2 \approx y(t_2)}$ and so on, which are approximations of

the exact solution obtained at the grid-points of (1.6). When the calculations are carried out in this way, i.e. step by step, at the end of the computational process a set of vectors $\{\mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_N}\}$ will be produced. These vectors represent approximately the values of the exact solution $\mathbf{y(t)}$ at the selected by (1.6) set of grid-points $\{\mathbf{t_0}, \mathbf{t_1}, \ldots, \mathbf{t_N}\}$.

It should be mentioned here that it is also possible to obtain, after the calculation of the sequence $\{\mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_N}\}$, approximations of the exact solution at some points of the independent variable $\mathbf{t} \in [\mathbf{a}, \mathbf{b}]$, which do not belong to the set (1.6). This can be done by using some appropriately chosen interpolation formulae.

The quantity $\mathbf{h}$ is called stepsize (the term time-stepsize will be used nearly always in this book). When $\mathbf{y_n}$ is calculated, the index $\mathbf{n}$ is giving the number of the currently performed steps (the term time-steps will also be used very often). Finally, the integer $\mathbf{N}$ is giving the number of time-steps, which have to be performed in order to complete the calculations.

In the above example it is assumed that the grid-points $\mathbf{t_n}$, $(\mathbf{n = 0, 1, 2, \ldots, N})$ are **equidistant**. The use of equidistant grids is in many cases very convenient, because it is, for example, possible to express in a very simple way an arbitrary grid-point $\mathbf{t_n}$ belonging to the set (1.6) by using the left-hand end-point $\mathbf{a}$ of the time-interval $(\mathbf{t_n = a + nh})$ when such a choice is made. However, it is not necessary to keep the time-stepsize constant during the whole computational process. Variable stepsizes can also be used. In such a case the grid-points can be defined as follows:

$$(\mathbf{1.7}) \quad \mathbf{t_0 = a}, \quad \mathbf{t_n = t_{n-1} + h_n} \quad (\mathbf{n = 1, 2, \ldots, N}), \quad \mathbf{t_N = b}.$$

In principle, the time-stepsize $\mathbf{h_n > 0}$ that is used at time-step $\mathbf{n}$ could always be different both from the time-stepsize $\mathbf{h_{n-1}}$ that was used at the previous time-step and from the time-stepsize $\mathbf{h_{n+1}}$ that will be used at the next time-step. However, some restrictions on the change of the stepsize are nearly always needed, see, for example, **Shampine and Gordon (1975)** or **Zlatev (1978, 1983)**, in order

(a) to preserve in a better way the accuracy of the calculated approximations,

(b) to ensure zero-stability during the calculations

and

(c) to increase the efficiency of the computational process by reducing the amount of simple arithmetic operations that are needed to obtain the approximate solution.

Some more details about the use of variable time-stepsize and about the additional assumptions, which are relevant in this case and which have to be imposed when this technique is implemented, can be found, for example, in **Hindmarsh (1971, 1980)**, **Gear (1971)**, **Krogh (1973a,b)**, **Shampine (1984, 1994)**, **Shampine and Gordon (1975)**, **Shampine, Watts and Davenport (1976)**, **Shampine and Zhang (1990)**, **Zlatev (1978, 1983, 1984, 1989)**, and **Zlatev and Thomsen (1979)**.

The use of a variable stepsize may often lead to an improvement of the efficiency of the computational process, because the code used in the numerical solution of the system of ODEs tries at prescribed in some way time-steps to select optimal time-stepsizes. This procedure may result in a substantial reduction of the computer time. However, there are cases, in which the computational process would become unstable. This kind of stability for systems of ODEs is called sometimes **zero-stability**, as, for example, in **Zlatev (1981b)**. Some further information about the zero-stability problems, which may arise when variation of the time-stepsize is allowed, can be found in **Gear and Tu (1974)**, **Gear and Watanabe (1974)**, **Zlatev (1978, 1983, 1984, 1989)**, and **Zlatev, Berkowicz and Prahm (1984a,b)**. Some discussion about the zero-stability of the computational process when it is allowed to vary the stepsize will be given in Chapter 3.

The major advantages of using **constant time-steps** are two:

> **(a)** it is easier to establish in a reliable way and to analyse the basic properties of the numerical method (such as convergence, accuracy and stability)

and

> **(b)** the behaviour of the computational error is more predictable and as a rule very robust.

The major disadvantage of this device (the application of a constant time-stepsize) appears in the case where some components of the exact solution are

> **(A)** quickly varying in some small part of the time-interval $[\mathbf{a}, \mathbf{b}]$

and

> **(B)** slowly varying in the remaining part of this interval.

It is well-known that a small time-stepsize **must** be used in **Case (A)**, while it is possible to apply much larger time-stepsize in **Case (B)**, but if a constant time-stepsize option is selected, then one is forced to use the chosen small constant stepsize during the whole computational process, which could be (or, at least, may be) very time-consuming. If it is allowed to vary the time-stepsize, then small time-stepsizes could be used, in principle at least, only when some of the components of the solution vary very quickly, while large time-stepsizes can be applied in the remaining part of the time-interval. The number of the needed time-steps will often be reduced considerably in this way, which normally will also lead, as mentioned above, to a very substantial decrease of the computing time that is needed to solve numerically the problem.

This means that by allowing some variations of the time-stepsize, one is trying to avoid the major disadvantage of the other option, the option where a constant time-stepsize is used during the whole computational process, i.e. one is trying avoid the necessity to apply a very small time-stepsize on the whole interval $[\mathbf{a}, \mathbf{b}]$. It is nearly obvious that the application of variable time-steps will often be successful, but, as pointed out above, problems may appear and it is necessary to be very careful when this option is selected and used (see also the references given above).

For the purposes of this book, in most of the cases it is not very important which of the two grids, the equidistant grid defined by (1.6) or the non-equidistant grid introduced by (1.7), will be chosen. Many

of the conclusions, which are drawn in the remaining part of this book, will be valid in both cases. We shall always give some explanations when this is not the case.

There is no need to introduce particular numerical methods in this chapter, because the introduction of the Richardson Extrapolation, which will be presented in the next section, and the discussion of some basic properties of the new numerical method, which arises when this computational device is additionally applied, will be valid for **any** one-step numerical method used in the solution of systems of ODEs. However, many special numerical methods will be introduced and studied in the following chapters.

1.3. Introduction of the Richardson Extrapolation

Assume that the system of ODEs is solved, step by step as explained in the previous section, by an **arbitrary** numerical method. Assume also that approximations of the exact solution $\mathbf{y(t)}$ are to be calculated for the values $\mathbf{t_n}$ $(\mathbf{n = 1, 2, \dots, N})$ either of the grid-points of (1.6) or of the grid-points of (1.7). Under these assumptions the simplest form of the Richardson Extrapolation can be introduced as follows.

If the calculations have already been performed for all grid-points $\mathbf{t_i}$, $(\mathbf{i = 1, 2, \dots, n-1})$ by using some numerical method, the order of accuracy of which is $\mathbf{p}$, and, thus, if approximations $\mathbf{y_i \approx y(t_i)}$ of the exact solution are available at the grid-points $\mathbf{t_i}$, $(\mathbf{i = 0, 1, 2, \dots, n-1})$, then **three actions** are to be carried out in order to obtain the next approximation $\mathbf{y_n}$ **:**

(a) Perform one large time-step, with a time-stepsize $\mathbf{h}$ when the grid (1.6) is used or with a time-stepsize $\mathbf{h_n}$ if the grid (1.7) has been selected, in order to calculate an approximation $\mathbf{z_n}$ of $\mathbf{y(t_n)}$.

(b) Perform two small time-steps, with a time-stepsize $\mathbf{0.5\,h}$, when the grid (1.6) is used or with a time-stepsize $\mathbf{0.5\,h_n}$ if the grid (1.7) has been selected, in order to calculate another approximation $\mathbf{w_n}$ of $\mathbf{y(t_n)}$.

(c) calculate an approximation $\mathbf{y_n}$ by applying the formula:

$$\mathbf{y_n = \frac{2^p w_n - z_n}{2^p - 1}}. \qquad (1.8)$$

The algorithm that is defined by the above three actions, the actions **(a)**, **(b)** and **(c)**, is called Richardson Extrapolation. As mentioned before, this algorithm was introduced and discussed by L. F. Richardson in 1911 and 1927, see **Richardson (1911, 1927)**. It should also be mentioned here that L. F. Richardson called this procedure "*deferred approach to the limit*".

Note that the idea is indeed **very general**. The above algorithm is applicable to **any** one-step numerical method for solving systems of ODEs (in Chapter 6 it will be shown that it is also applicable, after introducing some additional requirements, when some systems of PDEs are to be handled). There are **only** two requirements:

(A) The **same** one-step numerical method should be used in the calculation of the two approximations $\mathbf{z_n}$ and $\mathbf{w_n}$.

(B) The **order** of the selected numerical method should be $\mathbf{p}$. This second requirement is utilized in the derivation of formula (1.8), in which the positive integer $\mathbf{p}$ is involved; this will be further discussed in the next section.

The main properties of the Richardson Extrapolation will be studied in the next sections of Chapter 1 and in the next chapters of this book.

It should be noted here that, as already mentioned, the simplest version of the Richardson Extrapolation is described in this section. For our purposes this is quite sufficient, but some other versions of the Richardson Extrapolation can be found in, for example, **Faragó (2008)**. It should also be noted that many of the statements for the simplest version of the Richardson Extrapolation, introduced in this section and discussed in the remaining part of the book, remain also valid for other versions of this device.

1.4. Accuracy of the Richardson Extrapolation

Assume that the approximations $\mathbf{z_n}$ and $\mathbf{w_n}$ that have been introduced in the previous section were calculated by some numerical method, the order of accuracy of which is $\mathbf{p}$. If we additionally assume that the exact solution $\mathbf{y(t)}$ of the system of ODEs is sufficiently many times differentiable (actually, we have to assume that this function is $\mathbf{p+1}$ times continuously differentiable, which makes this assumption much more restrictive than the assumptions made in Theorem 1.1 in order to ensure existence and uniqueness of the solution of the system of ODEs), then the following two relationships can be written when the calculations have been carried out by using the grid-points introduced by (1.6) in Section 1.2:

$$\mathbf{y(t_n) - z_n = h^p K + O(h^{p+1})}\ , \qquad (1.9)$$

$$\mathbf{y(t_n) - w_n = (0.5\,h)^p K + O(h^{p+1})}\ . \qquad (1.10)$$

The quantity $\mathbf{K}$ that participates in the left-hand-side of both (1.9) and (1.10) depends both on the selected numerical method that was applied in the calculation of $\mathbf{z_n}$ and $\mathbf{w_n}$ and on the particular problem (1.1) – (1.2) that is handled. However, this quantity does not depend on the time-stepsize $\mathbf{h}$.

It follows from this observation that if the grid defined by (1.7) is used instead of the grid (1.6), then two new equalities, that are quite similar to (1.9) and (1.10), can immediately be written (only the time-stepsize $\mathbf{h}$ should be replaced by $\mathbf{h_n}$ in the right-hand-sides of both relations).

Let us now eliminate $\mathbf{K}$ from (1.9) and (1.10). After some obvious manipulations the following relationship can be obtained:

$$\mathbf{(1.11)} \qquad \mathbf{y(t_n)} - \frac{\mathbf{2^p w_n - z_n}}{\mathbf{2^p - 1}} = \mathbf{O(h^{p+1})} \, .$$

Note that the second term in the left-hand-side of (1.11) is precisely the approximation $\mathbf{y_n}$ that was obtained by the application of the Richardson Extrapolation, see the relation (1.8) at end of the previous section).

The following relationship can immediately be obtained from (1.8) after the above observation:

$$\mathbf{(1.12)} \qquad \mathbf{y(t_n)} - \mathbf{y_n} = \mathbf{O(h^{p+1})} \, \mathbf{.}$$

Comparing the relationship (1.12) with each of the relationships (1.9) and (1.10), we can immediately conclude that for sufficiently small values of the time-stepsize $\mathbf{h}$ the approximation $\mathbf{y_n}$ that is calculated by applying the Richardson Extrapolation will be more accurate than each of the two approximations $\mathbf{z_n}$ and $\mathbf{w_n}$ obtained when the selected numerical method is used directly. Indeed, the order of accuracy of $\mathbf{y_n}$ is at least $\mathbf{p+1}$, while each of $\mathbf{z_n}$ and $\mathbf{w_n}$ is of order of accuracy $\mathbf{p}$.

This means that **Richardson Extrapolation can be used to increase the accuracy of the calculated numerical solution**.

1.5. Evaluation of the error

The Richardson Extrapolation can also be used, and in fact it is very often used, to evaluate the leading term of the error of the calculated approximations and after that to determine an optimal in some sense time-stepsize, which can hopefully be applied successfully during the next time-step. Note that the relations (1.9) and (1.10) cannot directly be used in the evaluation of the error, because the value of the quantity $\mathbf{K}$ is in general not known. This means that it is necessary to eliminate this parameter in order to obtain an expression by which the error made can be estimated and, after that, an optimal time-stepsize determined and used during the next time-step.

An expression for $\mathbf{K}$ can easily be obtained by subtracting (1.10) from (1.9). It can easily be verified that the result of this action is

$$(1.13) \qquad \mathbf{K} = \frac{2^{p}\,(w_n - z_n)}{h^{p}\,(2^{p}-1)} + O(h^{p+1})\,.$$

Substituting this value of $\mathbf{K}$ in (1.10) leads to the following expression:

$$(1.14) \qquad y(t_n) - w_n = \frac{w_n - z_n}{2^{p}-1} + O(h^{p+1})\,.$$

The relationship (1.14) indicates that the leading term of the global error made in the computation of the approximation $\mathbf{w_n}$ can be estimated by applying the following relationship:

$$(1.15) \qquad \mathbf{ERROR_n} = \left| \frac{w_n - z_n}{2^{p}-1} \right|.$$

If a code for performing the calculations with a variable time-stepsize is developed and used, then (1.15) can be applied in order to decide how to select a good time-stepsize for the next time-step. The expression:

$$(1.16) \qquad h_{new} = \omega \sqrt[p]{\frac{TOL}{ERROR_n}}\; h$$

can be used in the attempt to calculate a time-stepsize for the next time-step, which will (hopefully) be better in some sense than the time-stepsize used at the current time-step. The user should be careful when this formula is used in a computer code because the programme could be terminated by a message that "overflow has taken place" if the calculated value of $\mathbf{ERROR_n}$ becomes very small.

The parameter $\mathbf{TOL}$ that appears in (1.16) is often called the error-tolerance and can freely be prescribed by the user according to the desired by him or by her accuracy.

The parameter $0 < \omega < 1$ is a precaution parameter introduced in an attempt to increase the reliability of the predictions made by using (1.16); $\omega = 0.9$ is used in many codes for automatic variation of the time-stepsize during the computational process, but smaller value of this parameter can also be used and are often advocated; see more details in **Gear (1971)**, **Hindmarsh (1980)**, **Krogh (1973)**, **Shampine and Gordon (1975)**, **Zlatev (1984)** and **Zlatev and Thomsen (1979)**.

It should be mentioned here that (1.16) is normally not sufficient in the determination of the rules for the variation of the time-stepsize. Some additional (and, in most of the cases, heuristic) rules are to be introduced and used. More details about these additional rules can be found in the above references.

1.6. Major drawbacks and advantages of the Richardson Extrapolation

It must once again be emphasized that the combination of the selected numerical method for solving systems of ODEs with the Richardson Extrapolation can be considered (and in fact must be considered) as **a new numerical method**. Let us now introduce the following two abbreviations:

(1) Method A: the original (underlying) method selected for solving systems of ODEs

and

(2) Method B: the combination of the original method, Method A, and the Richardson Extrapolation.

In this section we shall investigate some properties of the two numerical methods, Method A and Method B. More precisely, we shall try to find out **the main advantages and drawbacks of Method B when it is compared with Method A.**

Assume first that the selected Method A is **explicit**. In this case, Method B has one clear **disadvantage**: if this method and Method A are to be used with the **same** time-stepsize **h** , then three times more time-steps will be needed when the computations are carried out with Method B. The number of time-steps will in general be also increased when variations of the time-stepsize are allowed, but the situation is not very clear, because it is not easy to formulate a strategy for variation of the time-stepsize, when Method A is used without adding the Richardson Extrapolation.

If the underlying method, Method A, is **implicit**, then the situation is much more complicated, because systems of algebraic equations have to be handled at each time-step. In the general case, these systems are non-linear. Moreover, very often the non-linear systems of algebraic equations are large, containing millions of equations. This makes the treatment much more complicated. We shall postpone the detailed discussion of this case to Chapter 4, where the application of the Richardson Extrapolation for some implicit numerical methods will be studied.

The need to carry on three time-steps with Method B (instead of one time-step when Method A is used) is indeed a clear disadvantage, but only if the two methods have to be used with the **same** time-stepsize. However, it is not necessary to require the use of the same time-stepsize, because Method B has also one clear **advantage**: it is more accurate, its order of accuracy is at least by one higher than the order of accuracy of Method A. Therefore the results obtained by using Method B will in general be much more precise than those calculated by Method A and, therefore, the same accuracy, as that obtained by Method A, can be achieved when Method B is used with a substantially larger time-stepsize. This explains why it is not necessary to use Method A and Method B with the same time-stepsize, when we are interested in achieving certain prescribed in advance accuracy.

It is necessary to investigate after these preliminary remarks whether the advantage of Method B (i.e. the possibility of achieving better accuracy) is giving a sufficient compensation for its disadvantage.

The people who like the Richardson Extrapolation are claiming that the answer to this question is always a clear "**yes**". Indeed, the fact that Method B is more accurate than Method A will, as mentioned above, allow us, in principle at least, to apply bigger time-stepsizes when this method is used and nevertheless to achieve the same or even better accuracy. Denote by $\mathbf{h_A}$ and $\mathbf{h_B}$ the time-stepsizes used when Method A and Method B are applied and assume that some particular system of ODEs is to be solved. It is quite clear that if $\mathbf{h_B > 3\,h_A}$, then Method B will be computationally more efficient than Method A (but let us repeat here that this is true for the case where Method A is an explicit numerical method; if the Method A is implicit, then the inequality $\mathbf{h_B > 3\,h_A}$ should in general be replaced with another inequality $\mathbf{h_B > m\,h_A}$ where $\mathbf{m > 3}$; see more details in Chapter 4).

Assume now that the combined method, Method B, can indeed be applied with a considerably larger stepsize than that used when the computations are carried out with Method A. If, moreover, the accuracy of the results achieved by using Method B is higher than the corresponding accuracy, which was achieved by using Method A, then Method B will perform better than Method A for the solved problem (both with regard to the computational efficiency and with regard to the accuracy of the calculated approximations).

The big question, which must be answered by the people who like the Richardson Extrapolation can be formulated in the following way:

> **Will Method B be more efficient than Method A when realistic problems (say, problems arising in the treatment of some large-scale mathematical models describing various scientific and engineering problems) are solved and, moreover, will this happen even in the more difficult case when the underlying numerical method, Method A, is implicit?**

The answer to this important question is at least **sometimes** positive and it is worthwhile to demonstrate this fact by an example. The particular example, which was chosen by us for this demonstration is **an atmospheric chemical scheme**, which is described mathematically by a non-linear system of ODEs. We have chosen a scheme that contains $\mathbf{56}$ chemical species. It is one of the three atmospheric chemical schemes used in the Unified Danish Eulerian Model (UNI-DEM), see **Zlatev (1995)** or **Zlatev and Dimov (2006)**. This example will be further discussed and used in Chapter 4. In this chapter, we should like to illustrate **only** the fact that it is possible to achieve great efficiency with regard to the computing time when Method B is used even in the **more difficult case** where Method A is implicit.

The special accuracy requirement, which we imposed in the numerical treatment of the atmospheric chemical scheme, was that the global computational error $\boldsymbol{\tau}$ should be kept smaller than $\mathbf{10^{-3}}$ both in the case when Method A is used and in the case when Method B is applied. The particular numerical method, Method A, which was used in this experiment, was the well-known $\boldsymbol{\theta}$-method. It is well-known that the computations with Method A are carried out by using the formula:

$$(1.17) \quad y_n = y_{n-1} + h[(1-\theta)f(t_{n-1}, y_{n-1}) + \theta f(t_n, y_n)] \qquad \text{for} \qquad n = 1, 2, \ldots, N,$$

when the θ-method is applied. In this experiment $\theta = 0.75$ was selected. From (1.17) it is immediately seen that the θ-method is in general implicit because the unknown quantity y_n appears both in the left-hand side and in the right-hand side of this formula. It is also immediately seen that the method defined by (1.17) will become explicit only in the special case when the parameter θ is equal to zero. The θ-method will be reduced to the classical Forward Euler Formula (which will be used in Chapter 2) when this value of parameter θ is selected.

For Method B the approximations z_n and w_n are first calculated by applying the θ-method with time-stepsizes h and $0.5\,h$ respectively and then (1.8) is used to obtain y_n. These calculations are carried out at every time-step.

The atmospheric chemical scheme mentioned above was treated numerically on a rather long time-interval $[a, b] = [\,43200, 129600\,]$. The value $a = 43200$ corresponds to twelve o'clock at the noon (measured in seconds and starting from the mid-night), while $b = 120600$ corresponds to twelve o'clock on the next day. Thus, the length of the time-interval is 24 hours and it contains important changes from day-time to night-time and from night-time to day-time (when most of the chemical species are very quickly varying and, therefore, causing a lot of problems for any numerical method; this will be further discussed in Section 4). The steep gradients during these special short time-periods (changes from day-time to night-time and from night-time to day-time) are illustrated in **Fig. 1.1**. It is seen that some of the chemical species achieve maximum during day-time, while the maximum is achieved during night-time for other species. Finally, it should be noted that some of the chemical species vary in very wide ranges. More examples will be given in the next chapters.

The exact solution of the non-linear system of ODEs, by which the atmospheric chemical problem is described mathematically, is not known. Therefore a **reference solution** was firstly obtained by solving the problem with a **very small** time-stepsize and a numerical method of high order. Actually, a three-stage fifth-order fully-implicit Runge-Kutta algorithm, see **Butcher (2003)** or **Hairer and Wanner (1991)**, was used with $N = 998244352$ and $h_{ref} \approx 6.1307634E-05$ to calculate the reference solution. The reference solution was used (instead of the exact solution) in order to evaluate the global error. It should be mentioned here that the term "reference solution" in this context was for first time used probably by J. G. Verwer in 1977; see **Verwer (1977)**.

We carried out many runs with both Method A and Method B by using different time-stepsizes. Constant time-stepsizes, defined on the grid (1.6), were actually applied during every run. We started with a rather large time-stepsize and after each run decreased the time-stepsize by a factor of two. It is clear that the decrease of the stepsize by a factor of two leads to an increase of the number of time-steps also by a factor of two. Furthermore, one should expect the error to be decreased by a factor of two every time when the time-stepsize is halved, because the θ-method with $\theta = 0.75$ is a numerical method of order one. This action (decreasing the time-stepsize and increasing the number of time-steps by a factor of two) was repeated as long as the requirement $\tau < 10^{-3}$ was satisfied. Since Method B is more accurate than Method A, the time-stepsize, for which the requirement $\tau < 10^{-3}$ was for first time satisfied, was much larger when Method B is used. No more details about the solution procedure are needed here, but much more information about many different runs with the atmospheric chemical scheme can be found in Chapter 4. Some numerical results are presented in **Table 1.1.** The

computing times and the numbers of time-steps for the runs in which the accuracy requirement is **for first time satisfied** by each of the two methods are given in this table.

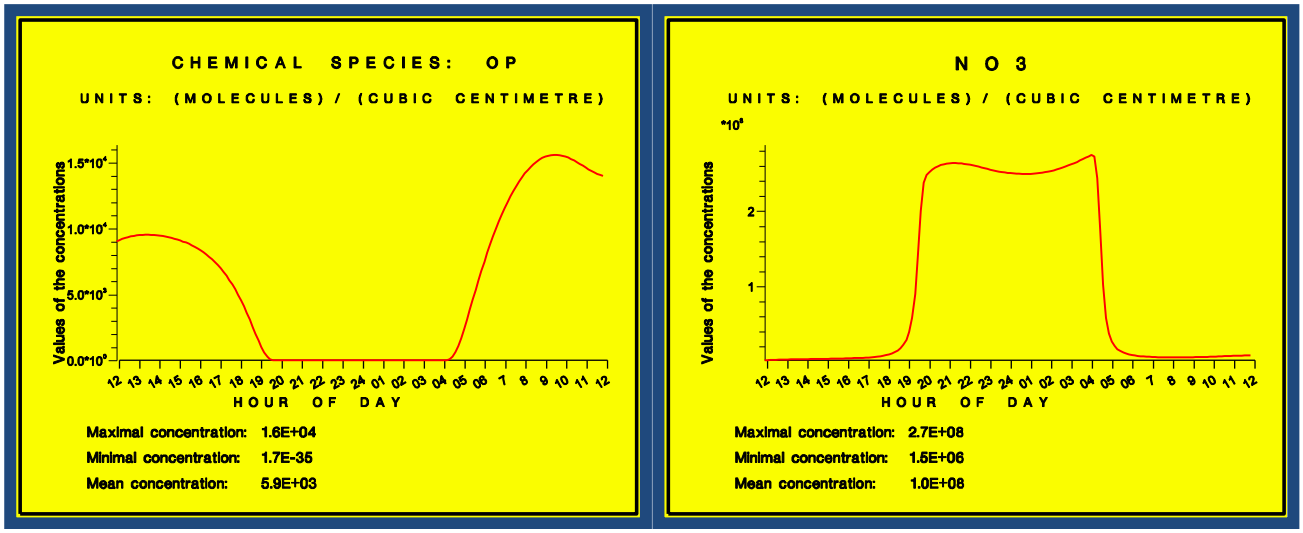

Figure 1.1

Diurnal variations of some of the chemical species involved in one of the atmospheric chemical schemes used in the Unified Danish Eulerian Model (UNI-DEM) for studying air pollution levels in Europe and its surroundings.

Compared characteristics	Method A	Method B	Ratio
Time-steps	344064	2688	128
Computing time	1.1214	0.1192	9.4077

Table 1.1

Numbers of time-steps and computing times (measured in CPU-hours) needed to achieve accuracy $\boldsymbol{\tau < 10^{-3}}$ when Method A (in this experiment the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ was applied in the computations directly) and Method B (the calculations were performed with the new numerical method, which consists of the combination of Method A and the Richardson Extrapolation) are used. In the last column of the table it is shown by how many times the number of time-steps and the computing time are reduced when Method B is used.

The results shown in Table 1.1 indicate that there exist examples, for which Method B is without any doubt much more efficient than Method A. However, it is not entirely satisfactory to establish this fact, because **the people who do not like very much the Richardson Extrapolation** have a very serious objection. They are claiming that it will not be possible **always** to increase the time-stepsize, because the computations may become unstable when large time-stepsizes are used. Moreover, in some cases not only is it not possible to perform the computations with Method B by using a bigger time-stepsize,

but even runs with the same time-stepsize, as that used successfully with Method A, will fail when Method B is applied. In the worst case, it will not be possible to solve the system of ODEs with Method B even if the time-stepsize used is smaller than that used in a successful run with Method A.

This objection is perfectly correct. In order to demonstrate this fact, let us consider again the θ-method defined by (1.17), but this time with $\theta = 0.5$. The particular numerical method obtained with this value of parameter θ is called the **Trapezoidal Rule**. This numerical method has very good stability properties. Actually, it is A-stable, which is very good for the case, in which the atmospheric chemical scheme with 56 species is treated (this fact will be fully explained in Chapter 4, but in the context of this section it is not very important). The big problem arises when the θ-method with $\theta = 0.5$ (i.e. the Trapezoidal Rule) is combined with the Richardson Extrapolation, because the stability properties of the combination of the Trapezoidal Rule with the Richardson Extrapolation are very poor, which was shown in **Dahlquist (1963)** and in **Faragó, Havasi and Zlatev (2010)**. Also this fact will be further clarified in Chapter 4, while now we shall concentrate our attention only on the performance of the two numerical methods (the Trapezoidal Rule and the combination of the Trapezoidal Rule with the Richardson Extrapolation) when the atmospheric chemical scheme with 56 species is to be handled.

Let us again use the names Method A and Method B, this time for the Trapezoidal Rule and for the combination of the Trapezoidal Rule and the Richardson Extrapolation, respectively. The calculations carried out with Method A were stable and the results were good always when the number of time-steps is varied from 168 to 44040192, while Method B produced **unstable results** for **all** of the time-stepsizes, which were used (this will be shown and further explained in Chapter 4).

The last result is very undesirable and, as a matter of fact, this completely catastrophic result indicates that it is necessary to answer the following question:

How can one avoid or at least predict the appearance of similar unpleasant situations?

The answer is, in principle at least, very simple: **the stability properties of Method B must be carefully studied.** If this is properly done, it will be possible to predict when the stability properties of Method B will become poor or even very poor and, thus, to select another (and better) numerical method and to avoid the disaster. This means that it is not sufficient to predict the appearance of bad results. It is, moreover, desirable and perhaps absolutely necessary to develop numerical methods for solving ODEs, for which the corresponding combinations with the Richardson Extrapolations have better stability properties (or, at least, for which the stability properties are not becoming as bad as in the above example where the Trapezoidal Rule was used). These two important tasks:

(a) the development of numerical methods for which the stability properties of the combinations of these methods with Richardson Extrapolation are better than those of the underlying methods when these are used directly (or at least are not becoming worse)

and

(b) the rigorous investigation of the stability properties of the combinations of many particular numerical methods with the Richardson Extrapolation

will be **the major topic** of the discussion in the following chapters of this book.

1.7. Two implementations of the Richardson Extrapolation

Formula (1.8) is in fact only telling us how to calculate the extrapolated approximation of $\mathbf{y_n}$ at every time-step $\mathbf{n}$ where $\mathbf{n = 1, 2, \dots, N}$ under the assumption that the two approximations $\mathbf{z_n}$ and $\mathbf{w_n}$ are available. However, this formula alone is not completely determining the algorithm by which the Richardson Extrapolation is to be used in the whole computational process. This algorithm will be completely described only when it is clearly explained what will happen when the approximations $\mathbf{y_n}$ for $\mathbf{n = 1, 2, \dots, N}$ are obtained. There are at least two possible choices:

(a) the calculated improved approximation $\mathbf{y_n}$ for a given $\mathbf{n}$ will not participate in the further calculations (it can be stored and used later for other purposes)

and

(b) the approximations $\mathbf{y_n}$ for a given $\mathbf{n}$ will directly be used in the computation of the next approximations $\mathbf{y_{n+1}}$.

This leads to two different implementations of the Richardson extrapolation. These implementations are graphically represented in **Fig. 1.2** and **Fig. 1.3**.

The implementation of the Richardson Extrapolation made according to the rule **(a)**, which is shown in **Fig. 1.2**, is called **passive**. It is quite clear why this name has been chosen (the extrapolated values are, as stated above, not participating in the further computations).

The implementation of the Richardson Extrapolation made by utilizing the second rule, rule **(b)**, which is shown in **Fig. 1.3**, is called **active**. It is immediately seen from the plot given in **Fig. 1.3** that in this case every improved value $\mathbf{y_n}$, where $\mathbf{n = 1, 2, \dots, N - 1}$, is actively used in the calculations of the next two approximations $\mathbf{z_{n+1}}$ and $\mathbf{w_{n+1}}$.

In **Botchev and Verwer (2009)**, the terms “global extrapolation” and “local extrapolation” are used instead of passive extrapolation and active extrapolation respectively. We prefer the term “Active Richardson Extrapolation” (to point out immediately that the improvements obtained in the extrapolation are directly applied in the further calculations) as well as the term “Passive Richardson Extrapolation” (to express in a more straightforward way the fact that the values obtained in the extrapolation process at time-step $\mathbf{n}$ will never be used in the consecutive time-steps).

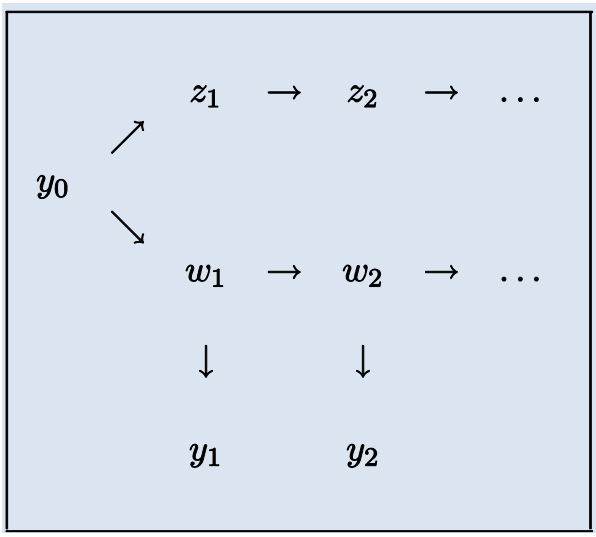

Figure 1.2
Passive implementation of the Richardson Extrapolation.

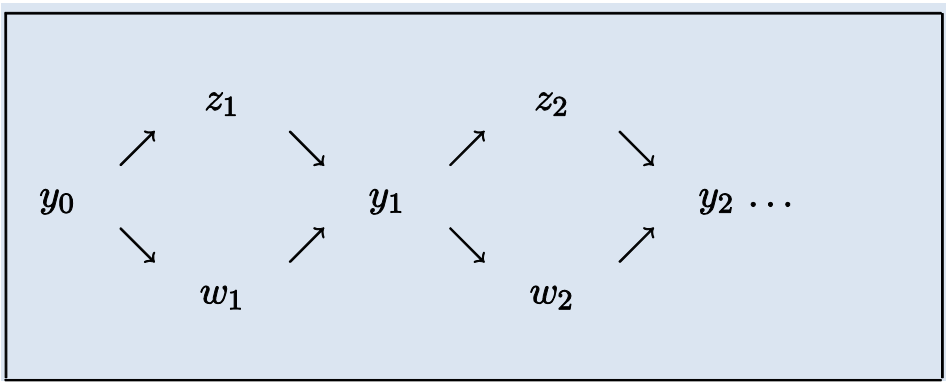

Figure 1.3
Active implementation of the Richardson Extrapolation.

The key question which arises in connection with the two implementations is:

Which of these two rules should be preferred?

There is not a unique answer to this question. Three different situations, the cases **(A)**, **(B)** and **(C)**, listed below, may arise and should be carefully taken into account in order to make the right decision:

(A) The application of both the Passive Richardson Extrapolation and the Active Richardson Extrapolation leads to a new numerical method, Method B, which have the same (or at least very similar) stability properties as those of the underlying numerical method, Method A.

(B) The new numerical method, Method B, which arises when the Passive Richardson Extrapolation is used, has good stability properties, while this is not the case for case

when the Active Richardson Extrapolation is used (as in the example given above with the Trapezoidal Rule). It should be mentioned here that it is nearly obvious that the underlying numerical method, Method A, and the combination of this method with the Passive Richardson Extrapolation, Method B, will always have the same stability properties.

(C) The new numerical method, Method B, which results after the application of the Active Richardson Extrapolation has better stability properties than those of the corresponding Method B, which arises after the application of the Passive Richardson Extrapolation.

In the experiments related to the application of the θ-method with $\theta = 0.75$ (given in the previous section, but much more numerical results will be presented in Chapter 4), **Case (A)** takes place and the results obtained when the two implementations are used are in general quite similar. However, it should be mentioned that **Botchev and Verwer (2009)** reported and explained some cases, in which the Active Richardson Extrapolation gave considerably better results for the special problem, which they were treating.

It is clear that the Passive Richardson Extrapolation should be used in **Case (B)**. Let us reiterate here that the example with the Trapezoidal Rule, which was given in the previous section, confirms in a very strong way this conclusion. Some more details will be given in Chapter 4.

Case (C) is giving some very clear advantages for the Active Richardson Extrapolation. In this situation the Passive Richardson Extrapolation may fail for some large time-stepsizes, for which the Active Richardson Extrapolation produces stable results. Some examples will be given in the next chapter.

The main conclusion from the above analysis is, again as in the previous section, that **it is absolutely necessary to investigate carefully the stability properties of the new numerical method**, the numerical method consisting of the combination of the selected underlying method and the chosen implementation of the Richardson Extrapolation. Only when this is properly done, one will be able to make the right choice and to apply the correct implementation of the Richardson Extrapolation. The application of the Richardson Extrapolation will in general become much more robust and reliable when such an investigation is thoroughly performed.

It must also be mentioned here that the stability properties are not the only factor, which must be taken into account. Some other factors, as, for example, quick oscillations of some components of the solution of (1.1) - (1.2), may also play a very significant role in the decision of which of the two implementations will perform better. However, it must be emphasized that these other factors may play an important role only in the case when the passive and the active implementations have the same (or, at least, very similar) stability properties. Thus, the requirement for investigating the stability properties of the two implementations is more essential. This requirement is necessary, but in some cases it is not sufficient and, therefore, if this is the truth, then some other considerations should be taken into account.

The above conclusions emphasize in a very strong way the fact that it is worthwhile to consider the classical Richardson Extrapolation not only in such a way as it was very often considered in many applications until now, but also **from another point of view**. Indeed, the Richardson Extrapolation defined by (1.8) is not only a simple device for increasing the accuracy of the computations and/or for obtaining an error estimation, although any of these two issues is, of course, very important.

The application of the Richardson Extrapolation results always in a quite new numerical method and this numerical method should be treated as any of the other numerical methods. It is necessary to study carefully all its properties, including here also its stability properties.

Therefore, in the following part of this book, the combination of any of the two implementations of the Richardson Extrapolation with the underlying numerical method will always be treated as **a new numerical method** the properties of which must be investigated in a very careful manner. The importance of the stability properties of this new numerical method will be the major topic in the next chapters.

Our main purpose will be

(A) to explain how new numerical methods, which are based on the Richardson Extrapolation and which have good stability properties, can be obtained

and

(B) to detect cases where the stability properties of the new numerical methods utilizing the Richardson Extrapolation become poor.

1.8. Increasing further the accuracy

Consider again (1.9) and (1.10), change the notation of the calculated approximation and assume that one additional term is kept on the right-hand side:

$$\mathbf{y}(\mathbf{t_n}) - \mathbf{z_n^{[1]}} = \mathbf{h^p K} + \mathbf{h^{p+1} L} + \mathbf{O}(\mathbf{h^{p+2}}) \ , \tag{1.18}$$

$$\mathbf{y}(\mathbf{t_n}) - \mathbf{z_n^{[2]}} = (\mathbf{0.5\ h})^{\mathbf{p}}\mathbf{K} + (\mathbf{0.5\ h})^{\mathbf{p+1}}\mathbf{L} + \mathbf{O}(\mathbf{h^{p+2}}) \ . \tag{1.19}$$

Assume furthermore that a third approximation $\mathbf{z_n^{[3]}}$ is calculated by performing four small time-steps with a time-stepsize $\mathbf{0.25\ h}$:

$$\mathbf{y}(\mathbf{t_n}) - \mathbf{z_n^{[3]}} = (\mathbf{0.25\ h})^{\mathbf{p}}\mathbf{K} + (\mathbf{0.25\ h})^{\mathbf{p+1}}\mathbf{L} + \mathbf{O}(\mathbf{h^{p+2}}) \ . \tag{1.20}$$

We should like to eliminate the constants $\mathbf{K}$ and $\mathbf{L}$ from the three equalities (1.18) –(1.20). First, the constant $\mathbf{K}$ could be eliminated. Multiply (1.19) with $\mathbf{2^p}$ and subtract (1.18) from the modified equality (1.19). Similarly multiply (1.20) with $\mathbf{4^p = 2^{2p}}$ and subtract (1.18) from the modified equality (1.20). The result is:

$$(\mathbf{1.21}) \qquad (\mathbf{2^p - 1})\mathbf{y(t_n)} - \mathbf{2^p z_n^{[2]}} + \mathbf{z_n^{[1]}} = -\mathbf{0.5 h^{p+1} L} + \mathbf{O(h^{p+2})}$$

$$(\mathbf{1.22}) \qquad (\mathbf{2^{2p} - 1})\mathbf{y(t_n)} - \mathbf{2^{2p} z_n^{[3]}} + \mathbf{z_n^{[1]}} = -\mathbf{0.75 h^{p+1} L} + \mathbf{O(h^{p+2})}$$

The constant $\mathbf{L}$ should also be eliminated. Multiply (1.21) with $-\mathbf{0.75}$ and (1.22) with $\mathbf{0.5}$.

$$(\mathbf{1.23}) \qquad -\mathbf{0.75}\,(\mathbf{2^p - 1})\mathbf{y(t_n)} + (\mathbf{0.75})\mathbf{2^p z_n^{[2]}} - \mathbf{0.75 z_n^{[1]}} = \mathbf{0.75(0.5) h^{p+1} L} + \mathbf{O(h^{p+2})}$$

$$(\mathbf{1.24}) \qquad \mathbf{0.5}\,(\mathbf{2^{2p} - 1})\mathbf{y(t_n)} - (\mathbf{0.5})\,\mathbf{2^{2p} z_n^{[3]}} + \mathbf{0.5 z_n^{[1]}} = \mathbf{0.5(-0.75) h^{p+1} L} + \mathbf{O(h^{p+2})}$$

Add the modified (1.21) to the modified (1.22). The result is:

$$(\mathbf{1.25}) \qquad (\mathbf{2^{2p-1} - 3(2^{p-2}) + 0.25})\mathbf{y(t_n)} - \mathbf{2^{2p-1} z_n^{[3]}} + \mathbf{3(2^{p-2}) z_n^{[2]}} - \mathbf{0.25 z_n^{[1]}} = \mathbf{O(h^{p+2})}$$

The last equality can be rewritten as

$$(\mathbf{1.26}) \qquad \mathbf{y(t_n)} = \frac{\mathbf{2^{2p-1} z_n^{[3]} - 3(2^{p-2}) z_n^{[2]} + 0.25 z_n^{[1]}}}{\mathbf{2^{2p-1} - 3(2^{p-2}) + 0.25}} + \mathbf{O(h^{p+2})}$$

Denote

$$(\mathbf{1.27}) \qquad \mathbf{y_n} = \frac{\mathbf{2^{2p-1} z_n^{[3]} - 3(2^{p-2}) z_n^{[2]} + 0.25 z_n^{[1]}}}{\mathbf{2^{2p-1} - 3(2^{p-2}) + 0.25}}.$$

Then (1.26) can be rewritten as

$$(\mathbf{1.28}) \qquad \mathbf{y(t_n) = y_n + O(h^{p+2})}$$

and it is clear that $\mathbf{y_n}$ being of order equal at least to $\mathbf{p+2}$ is more accurate than the three approximations $\mathbf{z_n^{[1]}}$, $\mathbf{z_n^{[2]}}$ and $\mathbf{z_n^{[3]}}$ the order of accuracy of which is $\mathbf{p}$.

It must be pointed out that the computational cost of the improvement of the accuracy achieved in this section is rather high. It is necessary to perform

(a) one time-step with a stepsize $\mathbf{h}$ in order to calculate the first approximation $\mathbf{z_n^{[1]}}$,

(b) two time-steps with a stepsize $\mathbf{0.5h}$ in order to calculate the second approximation $\mathbf{z_n^{[2]}}$

and

(c) four time-steps with a stepsize $\mathbf{0.25h}$ in order to calculate the second approximation $\mathbf{z_n^{[3]}}$.

The device described in this section can be considered as a Repeated Richardson Extrapolation. Some more details about different implementations of Repeated Richardson Extrapolation and about some other extrapolation methods can be found in the fourth chapter of **Hairer, Nørsett and Wanner (1987)**, see also **Deuflhard, Hairer and Zugck (1987)**, **Joyce (1971)** and **Christiansen and Petersen (1989)**. Interesting details about some of the scientists, who initiated the work on different extrapolation methods (including here the Richardson Extrapolations) can be found in the work of **Claude Brezinski** (https://nalag.cs.kuleuven.be/research/projects/WOG/history/talks/brezinski_talk.pdf). The stability of the new method (achieved when repeated Richardson Extrapolation is to be applied) must be studied carefully.

1.9. Major conclusions related to Chapter 1

The advantages and the drawbacks of the Richardson Extrapolation used in combination with systems of ODEs were discussed in this chapter. Two major conclusions were drawn during this discussion:

(a) the use of the Richardson Extrapolation is leading to both more accurate and **new** numerical methods

and

(b) this devise can be used to organize an automatic control of the stepsize according to some prescribed in advance accuracy, which is very important for practical computations.

However, these two conclusions will be true **only if** the new numerical method, the combination of the Richardson Extrapolation with the selected algorithm, is **stable**. This fact shows clearly that the preservation of the stability of the computational process is indeed a key issue. Therefore stability discussions will be the major topic in the next four chapters of this book.

1.10. Topics for further research

The following topics might lead to some very interesting and useful results:

(A) Attempt to verify that it is possible to achieve even higher accuracy than that achieved by using the device presented in Section 1.8. Assume that $\mathbf{p}$ is the order of accuracy of the selected numerical method and that $\mathbf{q}$ is an integer greater than one. Show that it is possible to design a numerical scheme of order $\mathbf{p+q}$ by applying a procedure, which is similar to the procedure used in Section 1.8 for $\mathbf{q=2}$. Note that the Richardson Extrapolation device discussed in Section 1.1 – Section 1.7 will be obtained by this general algorithm by setting $\mathbf{q=1}$.

(B) Try to evaluate the advantages and the drawbacks of the method designed as described above for an arbitrary positive integer $\mathbf{q}$.

(C) The stability properties of the methods obtained with $\mathbf{q>1}$ should also be carefully studied.

Chapter 2

Richardson Extrapolation for Explicit Runge-Kutta Methods

It is convenient to start the investigation of several efficient implementations of the Richardson Extrapolation with the case where this technique is applied together with the Explicit Runge-Kutta Methods (ERKMs), which are very popular among scientists and engineers. It was mentioned in the previous chapter that the implementation of the Richardson Extrapolation with any algorithm for solving systems of ODEs should be considered as a **new numerical method**. Assume now that **Method A** is any numerical algorithm from the class of the ERKMs and that **Method B** is the new numerical method formed by the combination of Method A with the Richardson Extrapolation. If the stability properties of the Method B (which will be discussed in detail in this chapter) are not causing computational problems, then the strategy needed to achieve better performance, when the Richardson Extrapolation is used, is very straight-forward. One can easily achieve excellent efficiency in this case, because Method B can be run successfully with a time-stepsize, which is considerably larger than the time-stepsize used with Method A. It is possible to select large time-stepsizes and to achieve better accuracy results, because Method B is, as was shown in the previous chapter, more accurate than Method A. This means that in this situation, i.e. when the stability is not causing troubles, the application of the Richardson Extrapolation will indeed always lead to a very efficient computational process. However, the situation is, unfortunately, not always so simple. The problem is that the stability requirements are very often putting severe restrictions on the choice of large time-stepsizes when explicit numerical methods are selected. The difficulties are much greater when the systems solved are very large. Therefore, it is necessary to require that at least two extra conditions are satisfied:

(a) Method A should have good stability properties

and, moreover, an additional and rather strong requirement must also be imposed,

(b) Method B should have **better** stability properties than Method A.

The computational process will be efficient even for mildly stiff systems of ODEs when the above two conditions, both condition **(a)** and condition **(b),** are satisfied. It will be shown in this chapter that **it is possible to satisfy simultaneously these two requirements** for some representatives of the class of the ERKMs.

In **Section 2.1** we shall present several definitions, which are related to the important concept of absolute stability of some numerical methods for solving systems of ODEs. These definitions are valid not only for the class of the Explicit Runge-Kutta Methods, but also for the much broader class of one-step methods for solving systems of ODEs.

The class of the Explicit Runge-Kutta Methods is introduced in **Section 2.2** and the stability polynomials, which are induced when these methods are used to handle the classical scalar and linear test-problem that has been introduced by G. Dahlquist in 1963, **Dahlquist (1963)**, are presented in the

case when some numerical method for solving ODEs is **directly** applied in the computational process (i.e. when the selected numerical method is applied without using the Richardson Extrapolation). Many of the assertions made in this section are also valid for the class of one-step methods. The ERKMs form a sub-class of the class of one-step methods.

Stability polynomials for **the new numerical methods**, which are combinations of Explicit Runge-Kutta Methods with the Richardson Extrapolation, are derived in **Section 2.3**. Also in this case the classical scalar and linear test-problem, which was introduced by G. Dahlquist in 1963, is used and the results are valid for the broader class of one-step methods.

In **Section 2.4**, the absolute stability regions of the Explicit Runge-Kutta Methods, when these are applied directly in the solution of systems of ODEs, are compared with the absolute stability regions of the new numerical methods, which appear when the Explicit Runge-Kutta Methods are combined with the Richardson Extrapolation. We assume in this section that the number $\mathbf{m}$ of stages of the selected ERKM is equal to its order of accuracy $\mathbf{p}$. It is verified that the absolute stability regions of the new numerical methods (derived by applying the Richardson Extrapolation) are **always** larger than those of the underlying ERKMs when this assumption, the assumption $\mathbf{m} = \mathbf{p}$, is made.

Three appropriate numerical examples are formulated in **Section 2.5**. By using these examples it will be possible to demonstrate in Section 2.8 the fact that the new numerical methods resulting when Explicit Runge-Kutta Methods are combined with Richardson Extrapolation can be used with larger time-stepsizes than the time-stepsizes used with the original Explicit Runge-Kutta Methods when these are applied directly and, moreover, that this is also true in the situations where the stability restrictions are much stronger than the accuracy requirements.

The organization of the computations, which are related to the three examples introduced in Section 2.5, is explained in **Section 2.6**. The selected by us particular approach during the organization of the computational process allowed us to compare in a better way both the accuracy achieved during the numerical solution and the convergence rates when the time-stepsizes are successively decreased.

The numerical methods, the absolute stability regions of which are shown in Section 2.4, form large classes when the order of accuracy $\mathbf{p}$ is greater than one. All methods within any of these classes have the **same** absolute stability region. The particular Explicit Runge-Kutta Methods from these classes, which are actually applied in the numerical experiments, are presented in **Section 2.7**.

Numerical results, which are obtained with the particular methods selected in Section 2.7 during the solution process organized as explained in Section 2.6, are given and discussed in **Section 2.8**. It is clearly demonstrated that the results are both more accurate and more stable when the ERKMs are combined with the Richardson Extrapolation.

Explicit Runge-Kutta methods (ERKMs) with even more enhanced absolute stability properties are derived and tested in **Section 2.9**. In this section it is assumed that $\mathbf{p} < \mathbf{m}$ and ERKMs obtained by using two particular pairs $(\mathbf{m}, \mathbf{p}) = (\mathbf{4}, \mathbf{3})$ and $(\mathbf{m}, \mathbf{p}) = (\mathbf{6}, \mathbf{4})$ are studied under the requirement to achieve good (and in some sense optimal) stability properties both in the case when these methods are used directly and also in the case when their combinations with the Richardson Extrapolation are to be applied in the solution process. The ERKMs with optimal stability regions form two large classes. Particular methods, which have good accuracy properties, are selected from each of the two classes and their efficiency is demonstrated by numerical experiments.

The discussion in Chapter 2 is finished with several concluding remarks in **Section 2.10**. Some possibilities for further improvements of the efficiency of the Richardson Extrapolation when this technique is used together with Explicit Runge-Kutta Methods are also sketched in the last section of this chapter. A conjecture, which is inspired by the results obtained in connection with the ERKMs and their combinations with the Richardson Extrapolation, is formulated at the end of Chapter 2.

Several problems for future research are proposed in the last section, **Section 2.11**, of this chapter.

2.1. Stability function of one-step methods for solving systems of ODEs

Consider again the classical initial value problem for non-linear systems of ordinary differential equations (ODEs), which was defined by (1.1) and (1.2) in the previous chapter. Assume that approximations $\mathbf{y_n}$ of the values of $\mathbf{y(t_n)}$ are to be calculated at the grid-points given in (1.6), but note that the assumption for an equidistant grid is done in this chapter only in order to facilitate and to shorten the presentation of the results; approximations $\mathbf{y_n}$ that are calculated on the grid (1.7) can also be considered in many of the cases treated in this section and in the following sections.

One of the most important requirements, which has to be imposed during the attempts to select good and reliable numerical methods and which will in principle ensure reliable and robust computer treatment during the numerical solution of the problem defined by (1.1) and (1.2), can be explained in the following way.

Let us assume that the exact solution $\mathbf{y(t)}$ of the initial value problem for systems of ODEs defined by (1.1) and (1.2) is a **bounded function** in the whole integration interval. This assumption is not a serious restriction. On the contrary, it is necessary, because such a requirement appears very often, practically nearly always, and has to be satisfied for many practical problems that arise in different fields of science and engineering. When the assumption for a bounded solution $\mathbf{y(t)}$ of the considered system of ODEs is made, it is very desirable to establish the fact that the following important requirement is also satisfied:

The approximate solution, which is obtained by the selected numerical method for any set of grid-points (1.6), must also be bounded when the exact solution is bounded.

The numerical solution is defined by the sequence $\{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}$, which in this case is computed by some Explicit Runge-Kutta Method on the grid-points of (1.6), but this sequence may also be obtained by using the non-equidistant grid (1.7). It is obvious that the numerical solution is bounded if there exists a constant $\mathbf{L} < \infty$, such that $\|\mathbf{y_n}\| \leq \mathbf{L}$ for the selected norm and for all sets of grid-points (1.6) or (1.7) for all indices $\mathbf{n} \in \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{N}\}$.

It is obvious that such a requirement (the requirement to compute a bounded numerical solution when the exact solution is a bounded function) is quite natural. Moreover, the natural requirement for obtaining a bounded numerical solution, in the case when the exact solution $\mathbf{y(t)}$ is bounded, leads, roughly speaking, to some **stability requirements** that must be imposed in the choice of the numerical methods in an attempt to increase the efficiency of the computational process and to obtain both more accurate and more reliable results. **Dahlquist (1963)** suggested to study the stability properties of the selected numerical method for solving ODEs by applying this method not in the solution of the general system defined by (1.1) and (1.2), but in the solution of one much simpler test-problem. Actually, in that work, **Dahlquist (1963)**, G. Dahlquist suggested to use the following **scalar and linear** test-equation in the stability investigations:

$$(\mathbf{2.1}) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \boldsymbol{\lambda}\,\mathbf{y}, \quad \mathbf{t} \in [\mathbf{0}, \infty], \quad \mathbf{y} \in \mathbb{C}, \quad \boldsymbol{\lambda} = \boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{i} \in \mathbb{C}, \quad \boldsymbol{\alpha} \leq \mathbf{0}, \quad \mathbf{y(0)} = \boldsymbol{\eta} \in \mathbb{C}.$$

It is clear from (2.1) that the constant $\boldsymbol{\lambda}$ is assumed to be a complex number with a non-positive real part and, therefore, in this particular case the dependent variable $\mathbf{y}$ takes values in the complex plane. Note too that the initial value $\boldsymbol{\eta}$ is in general also a complex number.

It is well-known that the exact solution $\mathbf{y(t)}$ of (2.1) is given by

$$(\mathbf{2.2}) \qquad \mathbf{y(t)} = \boldsymbol{\eta}\, e^{\boldsymbol{\lambda} \mathbf{t}}, \quad \mathbf{t} \in [\mathbf{0}, \infty].$$

It can immediately be seen that the exact solution $\mathbf{y(t)}$ given by (2.2) is bounded when the constraint $\boldsymbol{\alpha} \leq \mathbf{0}$ that is introduced in (2.1) is satisfied. Therefore, it is necessary to require that the approximate solution $\{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}$ computed by the selected numerical method is also bounded for any set of grid-points (1.6).

Assume now that (2.1) is treated by using an arbitrary **one-step** numerical method for solving ODEs. One-step methods are discussed in detail, for example, in **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Henrici (1968)**, and **Lambert (1991)**. Roughly speaking, only the approximation $\mathbf{y_{n-1}}$ of the solution at the grid-point $\mathbf{t_{n-1}}$ is used in the calculation of the approximation $\mathbf{y_n}$ at the next grid-point $\mathbf{t_n}$ of (1.6) when one-step methods are selected. A more formal statement can be derived from the definition given on p. 64 in **Henrici (1968)**, however, this is not very important for the further discussion in this chapter, where we shall study Explicit Runge-Kutta Methods, and the above explanation is quite sufficient for our purposes. The important thing is only the fact that the results presented in this section are valid for **any** one-step method for solving systems of ODEs (and, thus, also for the Explicit Runge-Kutta Methods, which form a sub-class of the class of one-step methods).

Let the positive constant (the time-stepsize) $\mathbf{h}$ be given and consider the following set of grid-points, which is very similar to (1.6):

$$(\mathbf{2.3}) \qquad \mathbf{t_0} = \mathbf{0}, \quad \mathbf{t_n} = \mathbf{t_{n-1}} + \mathbf{h} = \mathbf{t_0} + \mathbf{nh} \quad (\mathbf{n} = \mathbf{1}, \mathbf{2}, \ldots).$$

Approximations of the exact solution $\mathbf{y(t)}$ from (2.2) can successively, step by step, be calculated on the grid-points of the set defined in (2.3). Moreover, it is very easy to show, see more details in **Lambert (1991)**, that the application of an arbitrary one-step method in the treatment of (2.1) leads to the following recursive relation:

$$(\mathbf{2.4}) \qquad \mathbf{y_n = R(\nu)\, y_{n-1} = [R(\nu)]^n\, y_0}, \qquad \mathbf{\nu = \lambda\, h}, \qquad \mathbf{n = 1, 2, \ldots}$$

The function $\mathbf{R(\nu)}$ is called **the stability function** (see, for example, **Lambert, 1991**). If the applied one-step method is explicit, then this function is **a polynomial**. It is **a rational function** (some ratio of two polynomials, see Chapter 4) when implicit one-step methods are used.

It can immediately be concluded from (2.4) that if the relation $\mathbf{|R(\nu)| \leq 1}$ is satisfied for some value of $\mathbf{\nu = h\lambda}$, then the selected one-step method will produce a bounded approximate solution of (2.1) for the applied value $\mathbf{h}$ of the time-stepsize. It is said that the selected one-step numerical method is **absolutely stable** for this value of parameter $\mathbf{\nu}$ (see again **Lambert, 1991**).

Consider the set of **all** points $\mathbf{\nu}$ located in the complex plane to the left of the imaginary axis, for which the relationship $\mathbf{|R(\nu)| \leq 1}$ holds. The set of these points is called **absolute stability region** of the one-step numerical method under consideration (**Lambert, 1991, p. 202**).

The absolute stability definitions related to the scalar and linear test-problem (2.1), which were introduced above, can easily be extended for some **linear** systems of ODEs with **constant** coefficients that are written in the form:

$$(\mathbf{2.5}) \qquad \mathbf{\frac{dy}{dt} = A\, y}, \qquad \mathbf{t \in [0, \infty]}, \qquad \mathbf{y \in D \subset \mathbb{C}^s}, \qquad \mathbf{s \geq 1}, \qquad \mathbf{y(0) = \eta}, \qquad \mathbf{\eta \in D}\ .$$

It is assumed here that $\mathbf{A \in \mathbb{C}^{sxs}}$ is a given constant and diagonalizable matrix with complex elements and that $\mathbf{\eta \in \mathbb{C}^s}$ is some given vector with complex components. Under the first of these two assumptions, there exists a non-singular matrix $\mathbf{Q}$ such that $\mathbf{Q^{-1} A\, Q = \Lambda}$ where $\mathbf{\Lambda}$ is a diagonal matrix, whose diagonal elements are the eigenvalues of matrix $\mathbf{A}$ from the right-hand side of (2.5). Substitute now the expression $\mathbf{y = Q^{-1}\, z}$ in (2.5). The result is:

$$(\mathbf{2.6}) \qquad \mathbf{\frac{dz}{dt} = \Lambda\, z}, \qquad \mathbf{t \in [0, \infty]}, \qquad \mathbf{z \in \bar{D} \subset \mathbb{C}^s}, \qquad \mathbf{s \geq 1}, \qquad \mathbf{z(0) = \bar{\eta} = Q\, \eta}\ , \qquad \mathbf{\bar{\eta} \in \bar{D}}\ .$$

Assume that the real parts of **all** eigenvalues of matrix $\mathbf{A}$ are non-positive. It is clear that system (2.6) consists of $\mathbf{s}$ **independent** scalar equations of type (2.1) when this assumption is satisfied. A stability function $\mathbf{R(\nu_i)}, \quad \mathbf{i = 1, 2, \ldots, s}$, where $\mathbf{\nu_i = h\lambda_i}$, can be associated with each of these $\mathbf{s}$ equations when the selected one-step method is implicit. If the applied method is explicit then

$\mathbf{R(\nu_i)}$ is a polynomial. This means that the following definition for absolute stability related to problem (2.6) with some matrix $\mathbf{A}$ can be introduced in both cases. It is said that the one-step method is absolutely stable when it is applied in the numerical solution of (2.6) with some time-stepsize $\mathbf{h}$ if the inequality $\mathbf{|R(\nu_i)| \leq 1}$ is satisfied for **all** eigenvalues $\boldsymbol{\lambda_i}$, $\mathbf{i = 1, 2, \dots, s}$, of matrix $\mathbf{A}$.

The definition can be slightly simplified when all eigenvalues of matrix $\mathbf{A}$ are real (which means that all of them are non-positive). It is necessary to assume in this case that $\boldsymbol{\lambda}$ is an eigenvalue of matrix $\mathbf{A}$ for which the relationship $\mathbf{|\lambda| = max\,(\,|\lambda_1|\,,\ |\lambda_2|\,, \dots,\ |\lambda_s|)}$ holds. Note that if we set $\boldsymbol{\nu = h\lambda}$, then the application of an arbitrary one-step method in the numerical solution of (2.6) will produce a bounded numerical solution when the inequality $\mathbf{|R(\nu)| \leq 1}$ is satisfied. It is clear that this definition relates the concept of absolute stability with only one of the eigenvalues of matrix $\boldsymbol{\Lambda}$, with the eigenvalue, which is largest in absolute value, i.e. with $\mathbf{|\lambda| = max\,(\,|\lambda_1|\,,\ |\lambda_2|\,, \dots,\ |\lambda_s|)}$.

The above analysis shows that the problems connected with the stability of the computations during the numerical solution of the system (2.6) are slightly more complicated than the problems with the stability of the computations of the scalar and linear equation (2.1), because it is necessary to introduce a requirement that the stability polynomials must satisfy the inequality $\mathbf{|R(\nu_i)| \leq 1}$ for **all** eigenvalues $\boldsymbol{\lambda_i}$, $\mathbf{i = 1, 2, \dots, s}$, of matrix $\mathbf{A}$ in the latter case. However, the remarkable thing is that if the absolute stability region for the scalar and linear problem (2.5) is defined in a quite similar way as it was defined for the scalar equation (2.1), then the main ideas remain the same. The single relationship $\mathbf{|R(\nu)| \leq 1}$ for some given value $\boldsymbol{\nu \epsilon \mathbb{C}^-}$, then $\boldsymbol{\nu}$ is a point of the absolute stability region of the one-step numerical method used in the solution of (2.1) should now be replaced by a slightly more complicated requirement. More precisely, as stated above, the computations needed to obtain a numerical solution of the system (2.6) with a one-step method will be stable for a given time-stepsize $\mathbf{h}$, if all points $\boldsymbol{\nu_i = h\lambda_i}$ are inside the absolute stability region of the method. Therefore, it becomes immediately clear that for some linear systems of ODEs with constant coefficients the absolute stability region can be introduced precisely in the same way (or at least in a very similar way) as in the case where the scalar equation (2.1) is considered.

If matrix $\mathbf{A}$ is **not** constant, i.e. if $\mathbf{A = A(t)}$ and, thus, if the elements of this matrix depend on the time-variable $\mathbf{t}$, then the above result is **no more** valid. Nevertheless, under certain assumptions one can still **expect** the computational process to be stable. The main ideas, on which such an expectation is based, can be explained as follows. Assume that $\mathbf{n}$ is an arbitrary positive integer and that a matrix $\mathbf{A(\bar{t}_n)}$ where $\mathbf{\bar{t}_n \in [t_{n-1}, t_n]}$ is involved in the calculation of the approximation $\mathbf{y_n \approx y(\,t_n\,)}$ by the selected one-step numerical method. Assume further that matrix $\mathbf{A(\bar{t}_n)}$ is diagonalizable for all values of $\mathbf{\bar{t}_n}$. Then some diagonal matrix $\mathbf{\Lambda(\bar{t}_n)}$ will appear at time-step $\mathbf{n}$ instead of $\boldsymbol{\Lambda}$ in (2.6). Moreover, the eigenvalues of matrix $\mathbf{A(\bar{t}_n)}$ will be the diagonal elements of $\mathbf{\Lambda(\bar{t}_n)}$. Denote by $\mathbf{\lambda_1(\bar{t}_n)}$, $\mathbf{\lambda_2(\bar{t}_n)}$, ... , $\mathbf{\lambda_s(\bar{t}_n)}$ the eigenvalues of matrix $\mathbf{A(\bar{t}_n)}$. Assume that the real parts of all eigenvalues are non-positive and consider the products of these eigenvalues with the time-stepsize $\mathbf{h}$: $\mathbf{\nu_1(\bar{t}_n) = h\lambda_1(\bar{t}_n)}$, $\mathbf{\nu_2(\bar{t}_n) = h\lambda_2(\bar{t}_n)}$, ... , $\mathbf{\nu_s(\bar{t}_n) = h\lambda_s(\bar{t}_n)}$. A stability polynomial $\mathbf{R\big(\nu_i(\bar{t}_n)\big)}$, $\mathbf{i = 1, 2, \dots, s}$, can be associated with each of these quantities. It is clear that one should not expect the appearance of stability problems during the computations when the inequalities $\mathbf{\big|R\big(\nu_i(\bar{t}_n)\big)\big| \leq 1}$, $\mathbf{i = 1, 2, \dots, s}$, are satisfied. It is clear that it is necessary to require that similar inequalities are satisfied for **all** points $\mathbf{t_n}$ of the grid (2.3). This procedure seems to be rather complicated, but the heart of the matter is the fact that the system (2.6) is decoupled at every point of the grid (2.3) into a set of independent equations; each of them of the type (2.1). After this observation, the expectation for obtaining stable results can be expressed as follows: if the time-stepsize $\mathbf{h}$ is such

that all inequalities $\left|\mathbf{R}\left(\boldsymbol{\nu}_{\mathbf{i}}(\bar{\mathbf{t}}_{\mathbf{n}})\right)\right| \leq \mathbf{1}$ are satisfied at every time-step $\mathbf{n} = \mathbf{1}, \mathbf{2}, ...,$ and for all indices $\mathbf{i} = \mathbf{1}, \mathbf{2}, ... , \mathbf{s}$, then one should **expect** the computational process to remain stable.

Even more important is the fact that if the absolute stability region is formally defined precisely in the same way as it was defined above, i.e. if $|\mathbf{R}(\boldsymbol{\nu})| \leq \mathbf{1}$ for some given value $\boldsymbol{\nu} \in \mathbb{C}^{-}$, then $\boldsymbol{\nu}$ is a point of the absolute stability region of the one-step numerical method used in the solution of (2.6) when matrix $\mathbf{A} = \mathbf{A}(\mathbf{t})$ is time-dependent. Now the requirement of a stable computational process in the solution of (2.6) when $\mathbf{A}$ is a time-dependent matrix can be reformulated in the following way: one should expect the computations in the solution of (2.6) with a one-step method to be stable, when matrix $\mathbf{A} = \mathbf{A}(\mathbf{t})$ is time-dependent, for a given time-stepsize $\mathbf{h}$, if all points $\boldsymbol{\nu}_{\mathbf{i}}(\bar{\mathbf{t}}_{\mathbf{n}}) = \mathbf{h}\boldsymbol{\lambda}_{\mathbf{i}}(\bar{\mathbf{t}}_{\mathbf{n}})$, where $\mathbf{n} = \mathbf{1}, \mathbf{2}, ...,$ and $\mathbf{i} = \mathbf{1}, \mathbf{2}, ... , \mathbf{s}$, are inside the absolute stability region of the method. It is necessary to emphasize once again here that there is no guarantee that the computational process will necessarily be stable, but nearly all practical computations, which were carried out during more than 50 years after the introduction of the absolute stability concept in **Dahlquist (1963)**, indicate that stability is achieved nearly always (or at least very often).

Quite similar, and again heuristic, considerations can also be applied in connection with the **non-linear** system of ODEs described by (1.1) and (1.2). In this case instead of matrix $\mathbf{A}(\mathbf{t})$ one should consider the Jacobian matrix $\mathbf{J} = \partial \mathbf{f} / \partial \mathbf{y}$ of function $\mathbf{f}(\mathbf{t} , \mathbf{y})$ in the right-hand-side of (1.1); see more details for example in **Lambert (1991)**.

The scalar and linear equation (2.1) is very simple, but it is nevertheless very useful in the investigation of the stability of the numerical methods. This fact has been pointed out by many specialists working in the field of numerical solution of systems of ODEs (see, for example, the remark on page 37 of **Hundsdorfer and Verwer, 2003**). The above considerations indicate that it is worthwhile to base the absolute stability theory (at least until some more advanced and more reliable test-problem is found) on the simplest test-problem (2.1) as did G. Dahlquist in 1963; see **Dahlquist (1963)**.

The results presented in this section are valid for an arbitrary (either explicit or implicit) one-step method for solving systems of ODEs. In the next sections of this chapter we shall concentrate our attention on the investigation of the stability properties of the **Explicit Runge-Kutta Methods** (the ERKMs), which form a sub-class of the class of one-step methods. After that we shall show that if some numerical methods from this sub-class are combined with the Richardson Extrapolation, then the resulting new numerical methods will sometimes have increased absolute stability regions. For these new numerical methods it will be possible to apply large time-stepsizes also in the case where the stability requirements are stronger than the accuracy requirements and, thus, if the stability requirements, and not the accuracy requirements, put some restrictions on the choice of the time-stepsize during the numerical treatment of the system of ODEs.

2.2. Stability polynomials of Explicit Runge-Kutta Methods

Numerical methods of Runge-Kutta type for solving systems of ODEs are described and discussed in many text-books and papers; see, for example, **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Henrici (1968)**, and **Lambert (1991)**. Originally, some particular methods of this

type were developed and used (more than a hundred years ago) by **Kutta (1901)** and **Runge (1895)**. It should be mentioned here that the contribution of a third mathematician, Karl Heun (**Heun, 1900**), has been underestimated. He also developed a numerical method belonging to this class, which is still called Heun's method, and it would have been much more correct to use the name Runge-Heun-Kutta methods for the numerical algorithms from this class, but the name Runge-Kutta methods is already firmly established in this field.

The general $\mathbf{m}$-stage Explicit Runge-Kutta Method is a one-step numerical method for solving the systems of ODEs defined by (1.1) and (1.2). This numerical method is defined by the following formula (more details can be found, when necessary, in any of the text-books quoted above):

$$(2.7) \qquad y_n = y_{n-1} + h \sum_{i=1}^{m} c_i k_i^n \ .$$

The coefficients c_i are given constants, while at an arbitrary time-step n the stages k_i^n are defined by

$$(2.8) \qquad k_1^n = f(t_{n-1}, y_{n-1}) \ , \qquad k_i^n = f\left(t_{n-1} + h\, a_i \ , \ y_{n-1} + h \sum_{j=1}^{i-1} b_{ij} k_j^n \right) \ , \quad i = 2, 3, \ldots, m \ ,$$

with

$$(2.9) \qquad a_i = \sum_{j=1}^{i-1} b_{ij} \ , \quad i = 2, 3, \ldots, m \ ,$$

where b_{ij} are also some given constants depending on the particular numerical method.

Assume that the order of accuracy of the chosen Explicit Runge-Kutta Method is p and, additionally, that the choice $p = m$ is made for the numerical method under consideration. It can be shown (see, for example, **Lambert, 1991**) that it is possible to satisfy the requirement $p = m$ only if $m \leq 4$ while we shall necessarily have $p < m$ when m is greater than four. Assume further that the method defined with (2.7), (2.8) and (2.9) is applied in the treatment of the special test-problem (2.1). Then the stability polynomial $R(\nu)$ associated with the selected ERKM is given by (see **Lambert, 1991, p. 202**):

$$(2.10) \qquad R(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \cdots + \frac{\nu^p}{p!}, \qquad p = m, \qquad m = 1, 2, 3, 4 \ .$$

Mainly Explicit Runge-Kutta Methods with $\mathbf{p = m}$ will be considered in this chapter, but in **Section 2.9** some methods with $\mathbf{p < m}$ and with enhanced stability properties will be derived and tested.

2.3. Using Richardson Extrapolation together with the scalar test-problem

Consider an arbitrary (explicit or implicit) one-step method for solving systems of ODEs. Assume that:

(a) the selected one-step numerical method is of order $\mathbf{p}$

and

(b) an approximation $\mathbf{y_n}$ of the exact value $\mathbf{y(t_n)}$ of the solution of (2.1) has to be calculated under the assumption that a sufficiently accurate approximation $\mathbf{y_{n-1}}$ has already been computed.

The classical Richardson Extrapolation, which was introduced in Chapter 1 for the system of ODEs defined in (1.1) and (1.2), can easily be formulated for the case where the scalar and linear test-problem (2.1), which was proposed by **Dahlquist (1963)**, is solved. The algorithm, by which the Richardson Extrapolation is implemented in this way, can be presented as shown below. Note that the relationship (2.4) and, thus, the stability function $\mathbf{R(\nu)}$ (or the stability polynomial when the selected one-step method is explicit) is used in the formulation of this algorithm.

Step 1	Perform **one large** time-step with a time-stepsize $\mathbf{h}$ by using $\mathbf{y_{n-1}}$ as a starting value to calculate: $(\mathbf{2.11})\quad \mathbf{z_n = R(\nu)\, y_{n-1}}$.
Step 2	Perform **two small** time-steps with a time-stepsize $\mathbf{0.5h}$ by using $\mathbf{y_{n-1}}$ as a starting value in the first of the two small time-steps: $(\mathbf{2.12})\quad \mathbf{\bar{w}_n = R\left(\frac{\nu}{2}\right) y_{n-1}}$, $\mathbf{w_n = R\left(\frac{\nu}{2}\right) \bar{w}_n = \left[R\left(\frac{\nu}{2}\right)\right]^2 y_{n-1}}$.
Step 3	Compute (let us repeat here that $\mathbf{p}$ is the order of accuracy of the selected numerical method) an improved solution by applying the basic formula (1.8) by which the Richardson Extrapolation was defined in Chapter 1: $(\mathbf{2.13})\quad \mathbf{y_n = \frac{2^p w_n - z_n}{2^p - 1} = \frac{2^p \left[R\left(\frac{\nu}{2}\right)\right]^2 - R(\nu)}{2^p - 1} y_{n-1}}$.

Note too that in the derivation of the algorithm it is assumed that **the active implementation** of Richardson Extrapolation is used (see Section 1.7).

The last relationship, equality (2.13), in the scheme presented above shows clearly that **the combination of the selected one-step numerical method and the Richardson Extrapolation can also be considered as a <u>one-step</u> numerical method** for solving systems of ODEs when it is used to solve the Dahlquist scalar test-example (2.1).

Furthermore, it can easily be shown (by applying the same technique as that used in Chapter 1) that the approximation $\mathbf{y_n}$ calculated by (2.13) is usually of order $\mathbf{p+1}$ and, therefore, it is **always** more accurate than both $\mathbf{z_n}$ and $\mathbf{w_n}$ when the time-stepsize is sufficiently small. The most important fact is that the stability function (or polynomial, when the underlying numerical method is explicit) of the combined numerical method is expressed by the stability function (the stability polynomial) $\mathbf{R(\nu)}$ of the underlying numerical method and is given by the following expression:

$$(\mathbf{2.14}) \qquad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{2^p}\left[\mathbf{R}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)\right]^{\mathbf{2}} - \mathbf{R}(\boldsymbol{\nu})}{\mathbf{2^p - 1}} \ .$$

The above considerations are very general. As we already stated above, they are valid when the underlying numerical formula is any explicit or implicit one-step numerical method. However, in the following sections (2.4) – (2.8) of this chapter we shall restrict ourselves to the class of **Explicit Runge-Kutta Methods** with $\mathbf{p = m}$.

It is necessary now to emphasize additionally the fact that the stability functions or polynomials of the underlying method and those of its combination with the Richardson Extrapolation, i.e. the functions or the polynomials $\mathbf{R(\nu)}$ and $\bar{\mathbf{R}}(\boldsymbol{\nu})$, are **different**, which implies that the absolute stability regions of the underlying method and its combination with the Richardson Extrapolation **will in general also be different**.

Our purpose will be to study **<u>the impact</u>** of the application of the Richardson Extrapolation on the stability properties of the underlying Explicit Runge-Kutta Methods. In other words, we shall compare the absolute stability region of each of the Explicit Runge-Kutta Methods, for which $\mathbf{p = m}$ is satisfied, with the corresponding absolute stability region, which is obtained when the method under consideration is combined with the Richardson Extrapolation.

2.4. Impact of Richardson Extrapolation on the absolute stability properties

Let us repeat once again that the absolute stability region of a given one-step method (and also of a given numerical method of the class of the Explicit Runge-Kutta Methods) consists of all complex points $\boldsymbol{\nu = h\lambda}$ for which the stability function (if the numerical method is explicit, the stability function is reduced to a polynomial) satisfies the inequality $\boldsymbol{|R(\nu)| \leq 1}$. If the method is combined with the Richardson Extrapolation, the condition $\boldsymbol{|R(\nu)| \leq 1}$ must be replaced with the stronger requirement $\boldsymbol{|\bar{R}(\nu)| \leq 1}$, which was derived in the previous section; see (2.14). The last requirement is indeed stronger, also in the case where the numerical method is explicit, because as mentioned in the end of the previous section the two stability polynomials are different. This can be demonstrated by the following simple example. In the case, where a fourth-order four-stage Explicit Runge-Kutta Method is used, the polynomial $\boldsymbol{R(\nu)}$ will be of degree four, while the degree of the corresponding polynomial $\boldsymbol{\bar{R}(\nu)}$ will be eight when this method is combined with the Richardson Extrapolation. The same rule holds for **all** other Explicit Runge-Kutta Methods: the degree of the polynomial $\boldsymbol{\bar{R}(\nu)}$ is by a factor of two higher than the degree of the corresponding polynomial $\boldsymbol{R(\nu)}$. Therefore, the investigation of the absolute stability regions of the new numerical methods (consisting of the combinations of Explicit Runge-Kutta Methods and the Richardson Extrapolation) will be much more complicated than the investigation of the absolute stability regions of Explicit Runge-Kutta Methods when these are used directly.

The absolute stability regions of the classical Explicit Runge-Kutta Methods (ERKMs) with $\boldsymbol{p = m}$ and $\boldsymbol{m = 1, 2, 3, 4}$ are presented, for example, in **Lambert (1991), p. 202**. In this section these absolute stability regions will be compared with the absolute stability regions obtained when the Richardson Extrapolation is additionally used.

First and foremost, it is necessary to describe the algorithm, which has been used to draw the absolute stability regions. The boundaries of the parts of the absolute stability regions that are located above the negative real axis and to the left of the imaginary axis can been obtained in the following way. Let $\boldsymbol{\nu}$ be equal to $\boldsymbol{\bar{\alpha} + \bar{\beta}i}$ with $\boldsymbol{\bar{\alpha} \leq 0}$ and assume that $\boldsymbol{\varepsilon > 0}$ is some very small increment. We must mention that from (2.1) and (2.4) it follows that $\boldsymbol{\bar{\alpha} = h\alpha}$ and $\boldsymbol{\bar{\beta}i = h\beta i}$. Start with a fixed value $\boldsymbol{\bar{\alpha} = 0}$ of the real part of $\boldsymbol{\nu = \bar{\alpha} + \bar{\beta}i}$ and calculate the values of the stability polynomial $\boldsymbol{R(\nu)}$ for $\boldsymbol{\bar{\alpha} = 0}$ and for $\boldsymbol{\bar{\beta} = 0,\ \varepsilon,\ 2\varepsilon,\ 3\varepsilon,\ \ldots}$. Continue this process as long as the inequality $\boldsymbol{|R(\nu)| \leq 1}$ is satisfied and denote by $\boldsymbol{\bar{\beta}_0}$ the last value for which the requirement $\boldsymbol{|R(\nu)| \leq 1}$ was fulfilled. Set $\boldsymbol{\bar{\alpha} = -\varepsilon}$ and repeat the same computations for the new value of $\boldsymbol{\bar{\alpha}}$ and for $\boldsymbol{\bar{\beta} = 0,\ \varepsilon,\ 2\varepsilon,\ 3\varepsilon,\ \ldots}$. Denote by $\boldsymbol{\bar{\beta}_1}$ the largest value of $\boldsymbol{\bar{\beta}}$ for which the stability requirement $\boldsymbol{|R(\nu)| \leq 1}$ is satisfied. Continuing the computations, it will be possible to calculate the coordinates of a very large set of points $\boldsymbol{\{(0, \bar{\beta}_0),\ (-\varepsilon, \bar{\beta}_1),\ (-2\varepsilon, \bar{\beta}_2), \ldots\}}$ located in the negative part of the complex plane over the real axis. More precisely, all of these points are located close to the boundary of the part of the absolute stability region which is over the negative real axis and to the left of the imaginary axis. Moreover, all these points lie inside the absolute stability region, but if $\boldsymbol{\varepsilon}$ is sufficiently small, then they will be very close to the boundary of the absolute stability region. Therefore, the curve connecting successively all these points will in such a case be a very close approximation of the boundary of the part of the stability region, which is located over the real axis and to the left of the imaginary axis.

It should be mentioned here that $\boldsymbol{\varepsilon = 0.001}$ was actually used in the preparation of all plots that are presented in this section.

It can easily be shown that the absolute stability region is symmetric with regard to the real axis. Therefore, there is no need to repeat the computational process that was described above for negative values of the imaginary part $\bar{\beta}$ of $\nu = h\lambda = \bar{\alpha} + \bar{\beta}i$.

Some people are drawing parts of the stability regions which are located to the right of the imaginary axis (see, for example, **Lambert, 1991**). In our opinion, this is not necessary and in the most of the cases it will not be desirable either. The last statement can be explained as follows. Consider equation (2.1) and let again ν be equal to $\bar{\alpha} + \bar{\beta}i$ but assume now that $\bar{\alpha}$ is a positive number. Then the exact solution (2.2) of (2.1) is **not** bounded and it is clearly not desirable to search for numerical methods, which will produce bounded approximate solutions (the concept of relative stability, see **Lambert, 1991, p. 75**, is more appropriate in this situation, but this topic is beyond the scope of the present book). Therefore, no attempts were made to find the parts of the stability regions which are located to the right of the imaginary axis.

The main advantages of the described in this section procedure for obtaining the absolute stability regions of one-step methods for solving systems of ODEs are two:

(a) it is conceptually very simple

and

(b) it is very easy to prepare computer programs exploiting it; moreover, it is very easy to carry out the computations in parallel.

The same (or at least a very similar) procedure has also been used in **Lambert (1991)**. Other procedures for drawing the absolute stability regions for numerical methods for solving systems of ODEs can be found in many text-books; see, for example, **Hairer, Nørsett and Wanner (1987)**, **Hairer and Wanner (1991)**, **Hundsdorfer and Verwer (2003)** and **Lambert (1991)**.

It should also be stressed here that the procedure for drawing the absolute stability regions of the Explicit Runge-Kutta Methods (ERKMs) with $p = m$, which was described above, is directly applicable for the new numerical methods which arise when any of the ERKMs with $p = m$ is combined with the Richardson extrapolation. It will only be necessary to replace the stability polynomial $R(\nu)$ with $\bar{R}(\nu)$ when these new methods are studied. It should be repeated here that the computations will be much more complicated in the latter case.

2.4.1. Stability regions related to the first-order one-stage Explicit Runge-Kutta Method

The first-order one-stage Explicit Runge-Kutta Method is well-known also as the **Forward Euler Formula** or as the Explicit Euler Method. Its stability polynomial can be obtained from (2.10) by applying $p = m = 1$:

$$R(\nu) = 1 + \nu \,. \qquad (2.15)$$

The application of the Richardson Extrapolation together with the first-order one-stage Explicit Runge-Kutta Method leads according to (2.10) applied with $\mathbf{p = m = 1}$ and (2.14) to a stability polynomial of the form:

$$(\mathbf{2.16}) \qquad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \mathbf{2}\left(\mathbf{1} + \frac{\boldsymbol{\nu}}{\mathbf{2}}\right)^{\mathbf{2}} - (\mathbf{1} + \boldsymbol{\nu}) \ .$$

The absolute stability regions, which are obtained by using (2.15) and (2.16) as well as the procedure discussed in the beginning of this section, are given in **Fig. 2.1**.

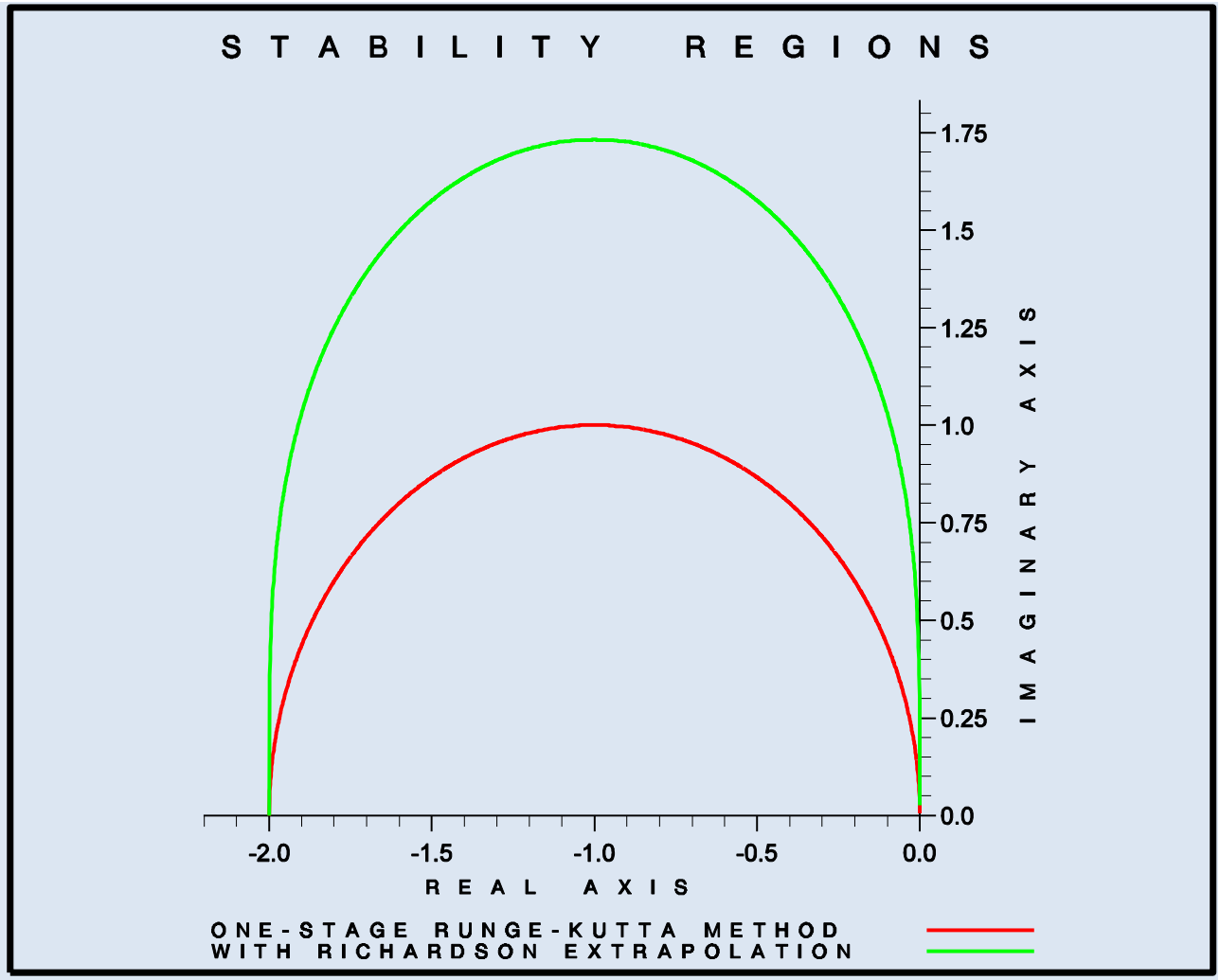

Figure 2.1

Stability regions of the original first-order one-stage Explicit Runge-Kutta Method and the combination of the Richardson Extrapolation with this method.

2.4.2. Stability regions related to the second order two-stage Explicit Runge-Kutta Methods

The stability polynomial of any second-order two-stage Explicit Runge-Kutta Method (there exists a large class of such methods) can be obtained from (2.10) by applying $\mathbf{p = m = 2}$:

$$\mathbf{R(\nu) = 1 + \nu + \frac{\nu^2}{2!}} \quad (2.17)$$

The application of the Richardson Extrapolation together with any of the second-order two-stage Explicit Runge-Kutta Method leads according to (2.10) applied with $\mathbf{p = m = 2}$ and (2.14) to a stability polynomial of degree four, which can be written in the following form:

$$\mathbf{\bar{R}(\nu) = \frac{4}{3}\left[1 + \frac{\nu}{2} + \frac{1}{2!}\left(\frac{\nu}{2}\right)^2\right]^2 - \frac{1}{3}\left(1 + \nu + \frac{\nu^2}{2!}\right)} \quad (2.18)$$

The stability regions obtained by using (2.17) and (2.18) together with the procedure discussed in the beginning of this section are given in **Fig. 2.2**.

2.4.3. Stability regions related to the third-order three-stage Explicit Runge-Kutta Methods

The stability polynomial of any third-order three-stage Explicit Runge-Kutta Method (there exists a large class of such methods) can be obtained from (2.10) by applying $\mathbf{p = m = 3}$:

$$\mathbf{R(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!}} \quad (2.19)$$

The application of the Richardson Extrapolation together with any of the third-order three-stage Explicit Runge-Kutta Method leads according to (2.10) applied with $\mathbf{p = m = 3}$ and (2.14) to a stability polynomial of degree six, which can be written in the following form:

$$\mathbf{\bar{R}(\nu) = \frac{8}{7}\left[1 + \frac{\nu}{2} + \frac{1}{2!}\left(\frac{\nu}{2}\right)^2 + \frac{1}{3!}\left(\frac{\nu}{2}\right)^3\right]^2 - \frac{1}{7}\left(1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!}\right)} \quad (2.20)$$

The absolute stability regions, which are obtained by using (2.19) and (2.20) as well as the procedure discussed in the beginning of this section, are given in **Fig. 2.3**.

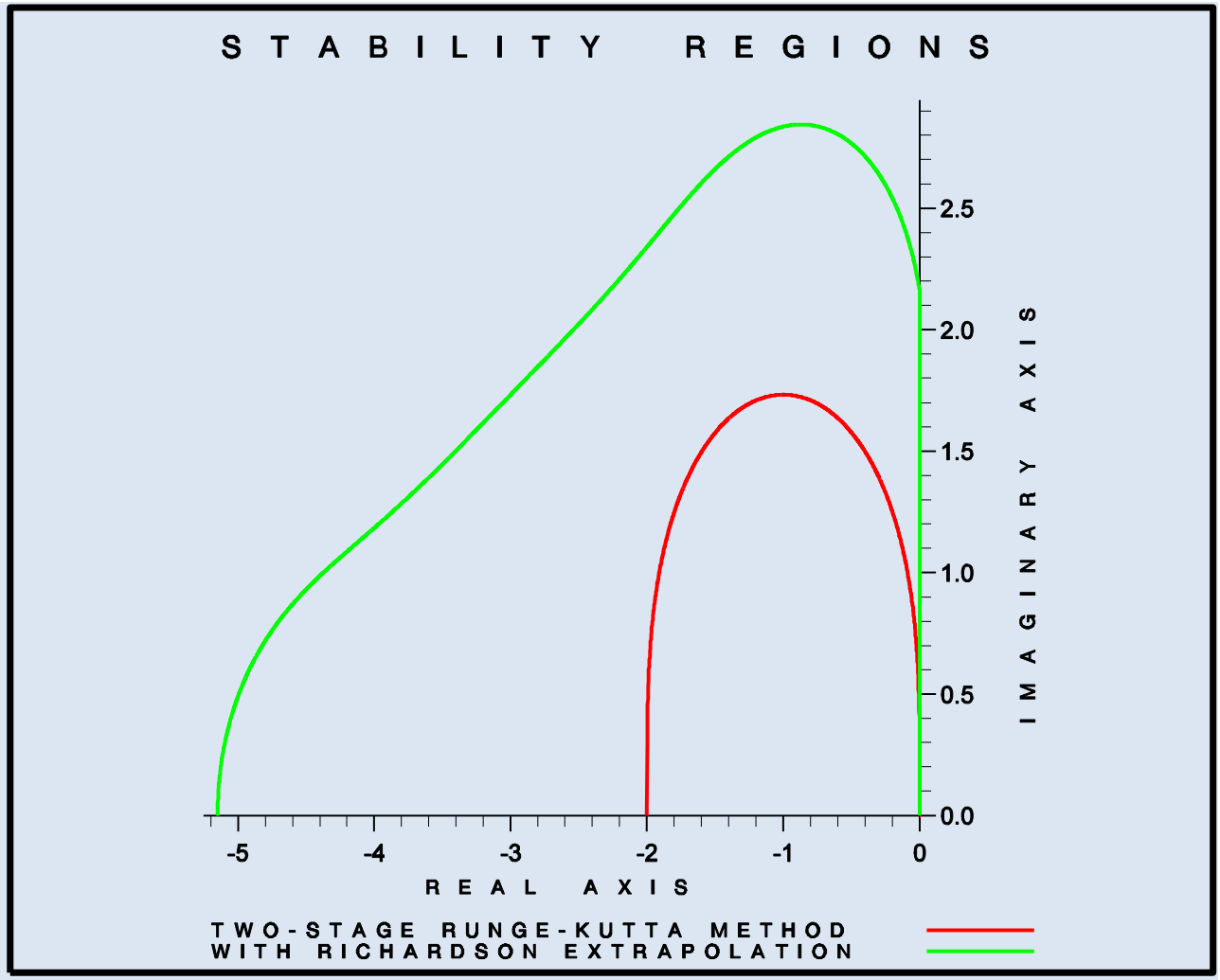

Figure 2.2

Stability regions of **any** representative of the class of the second-order two-stage Explicit Runge-Kutta Methods and the combination of the Richardson Extrapolation with this method.

2.4.4. Stability regions related to the fourth-order four-stage Explicit Runge-Kutta Methods

The stability polynomial of any fourth-order four-stage Explicit Runge-Kutta Method (there exists a large class of such methods) can be obtained from (2.10) by applying **p = m = 4:**

$$(2.21) \qquad \mathbf{R}(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \frac{\nu^4}{4!} \ .$$

The application of the Richardson Extrapolation together with the fourth-order four-stage Explicit Runge-Kutta Method leads according to (2.10) applied with $\mathbf{p = m = 4}$ and (2.14) to a stability polynomial of degree eight, which can be written in the following form:

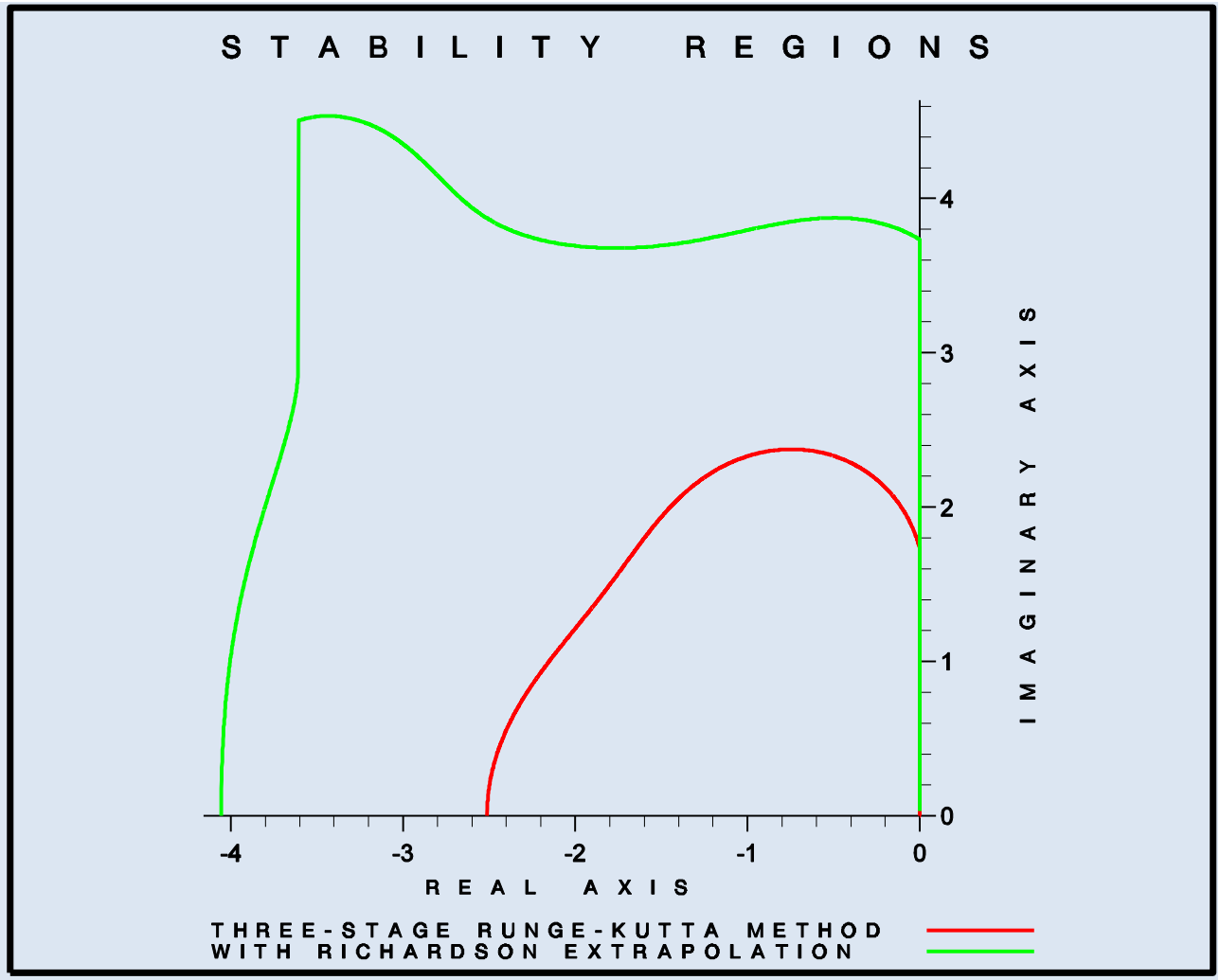

Figure 2.3

Stability regions of **any** representative of the class of the third-order three-stage Explicit Runge-Kutta Methods and the combination of the Richardson Extrapolation with this method.

$$(2.22) \quad \bar{\mathbf{R}}(\nu) = \frac{16}{15}\left[1 + \frac{\nu}{2} + \frac{1}{2!}\left(\frac{\nu}{2}\right)^2 + \frac{1}{3!}\left(\frac{\nu}{2}\right)^3 + \frac{1}{4!}\left(\frac{\nu}{2}\right)^4\right]^2 - \frac{1}{15}\left(1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \frac{\nu^4}{4!}\right).$$

The absolute stability regions, which are obtained by using (2.21) and (2.22) as well as the procedure discussed in the beginning of this section, are given in **Fig. 2.4**.

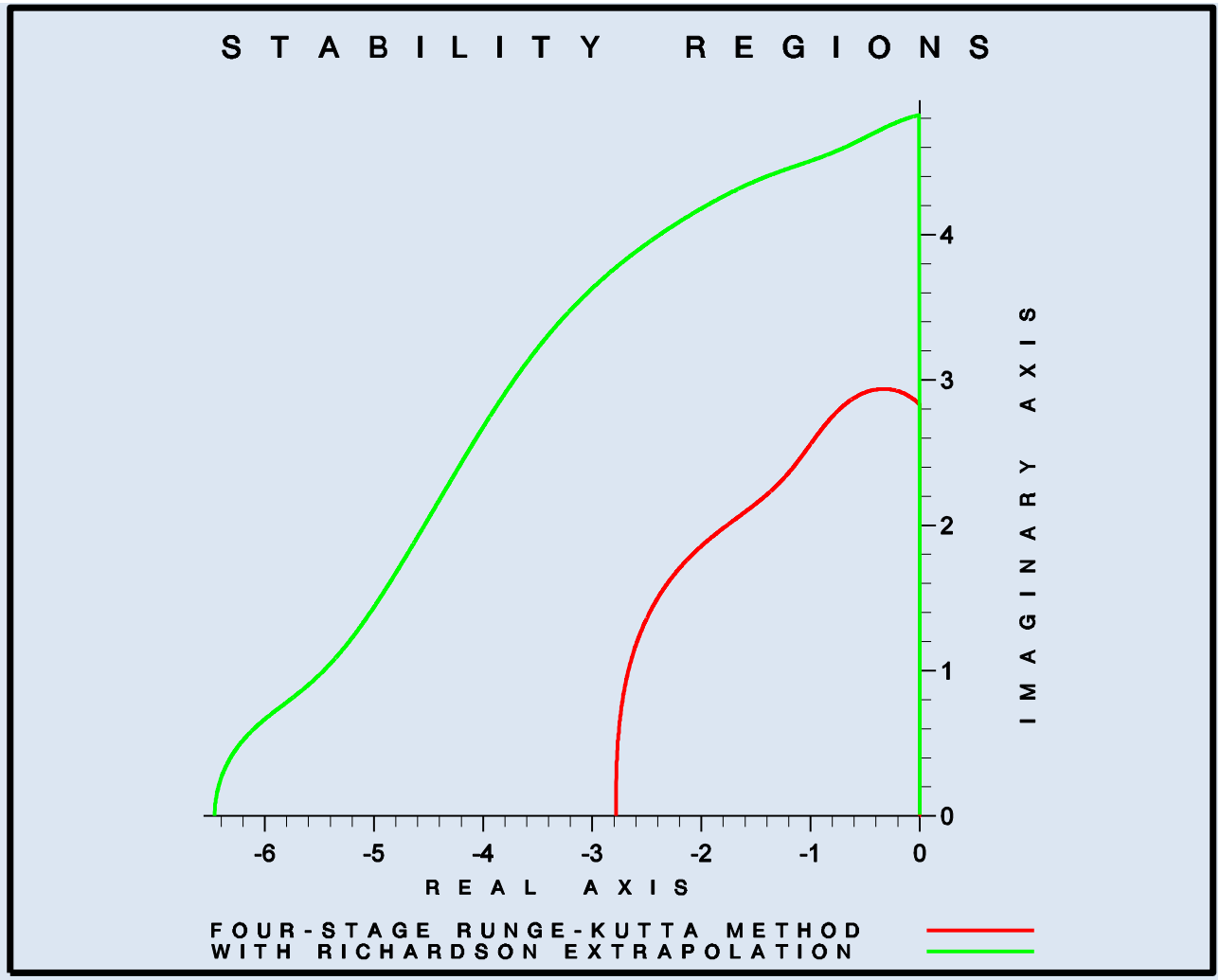

Figure 2.4

Stability regions of any representative of the class of the forth-order four-stage Explicit Runge-Kutta Method and the combination of the Richardson Extrapolation with this method.

2.4.5. About the use of complex arithmetic in the program for drawing the plots.

The variables $\mathbf{R}$ (which is the value of the stability polynomial of the selected method) and $\boldsymbol{\nu}$ were declared as "DOUBLE COMPLEX" in a FORTRAN program implementing the algorithm described in the beginning of this section. After that formulae (2.15) – (2.22) were directly used in the calculations. When the computation of the complex value of $\mathbf{R}$ for a given value of $\boldsymbol{\nu}$ is completed, the real part $\overline{\mathbf{A}}$ and the imaginary part $\overline{\mathbf{B}}$ of $\mathbf{R}$ can easily be extracted. The numerical method under consideration is stable for the current value of $\boldsymbol{\nu}$ if the condition $\sqrt{\overline{\mathbf{A}}^2 + \overline{\mathbf{B}}^2} \leq 1$ is satisfied.

It should be noted that it is also possible to use **only** real arithmetic calculations in the computer program. If such an approach is for some reasons more desirable than the use of complex arithmetic calculations, then long transformations are to be carried out in order first to obtain directly analytic

expressions for $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$. After that the condition $\sqrt{\bar{\mathbf{A}}^2 + \bar{\mathbf{B}}^2} \leq 1$ can again be used to check if the method is stable for the current value of $\boldsymbol{\nu}$. This alternative approach is fully described in **Zlatev, Georgiev and Dimov (2013a)**.

2.5. Preparation of appropriate numerical examples

Three numerical examples will be defined in §2.5.1, §2.5.2 and §2.5.3. These examples will be used to calculate and present numerical results in the following sections. The first and the second examples are linear systems of ODEs with constant coefficients and are created in order to demonstrate the fact that the theoretical results related to the absolute stability are valid also when the Richardson Extrapolation is additionally applied. Each of these two examples contains three equations and its coefficient matrix has both real and complex eigenvalues. In the first example, the real eigenvalue is the dominant one, while the complex eigenvalues put the major constraints on the stability of the computational process in the second example. The third example is a non-linear system of ODEs. It contains two equations and is taken from **Lambert (1991), p. 223**.

The main purpose with the three examples is to demonstrate the fact that the combined methods (Explicit Runge-Kutta methods + Richardson Extrapolation) can be used with large time-stepsizes also when the stability requirements are very restrictive. It will be shown in Section 2.8 that the combined methods will produce good numerical solutions for some large time-stepsizes, for which the original Explicit Runge-Kutta Methods are not stable.

2.5.1. Numerical example with a large real eigenvalue

Consider the linear system of ordinary differential equations (ODEs) with constant coefficients given by

$$(2.23) \quad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{A\,y}, \quad \mathbf{t} \in [\mathbf{0}, \mathbf{13.1072}], \quad \mathbf{y} = (\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3})^{\mathbf{T}}, \quad \mathbf{y(0)} = (\mathbf{1}, \mathbf{0}, \mathbf{2})^{\mathbf{T}}, \quad \mathbf{A} \in \mathbb{R}^{\mathbf{3x3}}.$$

The elements of matrix $\mathbf{A}$ from (2.23) are given below:

$$(2.24) \quad \mathbf{a_{11} = 741.4}, \qquad \mathbf{a_{12} = 749.7}, \qquad \mathbf{a_{13} = -741.7},$$

$$(2.25) \quad \mathbf{a_{21} = -765.7}, \qquad \mathbf{a_{22} = -758}, \qquad \mathbf{a_{23} = 757.7},$$

$$(2.26) \quad \mathbf{a_{31} = 725.7}, \qquad \mathbf{a_{32} = 741.7}, \qquad \mathbf{a_{33} = -734}.$$

The three components of the exact solution of the problem defined by (2.23) – (2.26) are given by

$$(\mathbf{2.27}) \quad \mathbf{y_1(t) = e^{-0.3t}\ \sin 8t + e^{-750t}}\,,$$

$$(\mathbf{2.28}) \quad \mathbf{y_2(t) = e^{-0.3t}\ \cos 8t - e^{-750t}}\,,$$

$$(\mathbf{2.29}) \quad \mathbf{y_3(t) = e^{-0.3t}\ (\sin 8t + \cos 8t) + e^{-750t}}\,.$$

It should be mentioned here that the eigenvalues of matrix $\mathbf{A}$ from (2.23) are given by

$$(\mathbf{2.30}) \quad \boldsymbol{\mu_1 = -750}\,, \quad \boldsymbol{\mu_2 = -0.3 + 8i}\,, \quad \boldsymbol{\mu_3 = -0.3 - 8i}\ .$$

The absolute value of the real eigenvalue $\boldsymbol{\mu_1}$ is much larger than the absolute values of the two complex eigenvalues of matrix $\mathbf{A}$. This means, roughly speaking, that the computations will be stable when $\boldsymbol{|\nu| = h|\mu_1|}$ is smaller than the length of the stability interval on the real axis (from the plots given in **Fig. 2.1 – Fig. 2.4** it is clearly seen that this length is smaller than 3 for all four Explicit Runge-Kutta Methods studied in the previous sections). In fact, one should require that all three points $\boldsymbol{h\mu_1}$, $\boldsymbol{h\mu_2}$ and $\boldsymbol{h\mu_3}$ must lie in the absolute stability region of the used method, but it is clear that the last two points are not very important when the absolute stability is considered (these two points will be inside the absolute stability regions, when this is true for the first one).

The plots of the three components of the solution of the example presented in this sub-section are given in **Fig. 2.5.**

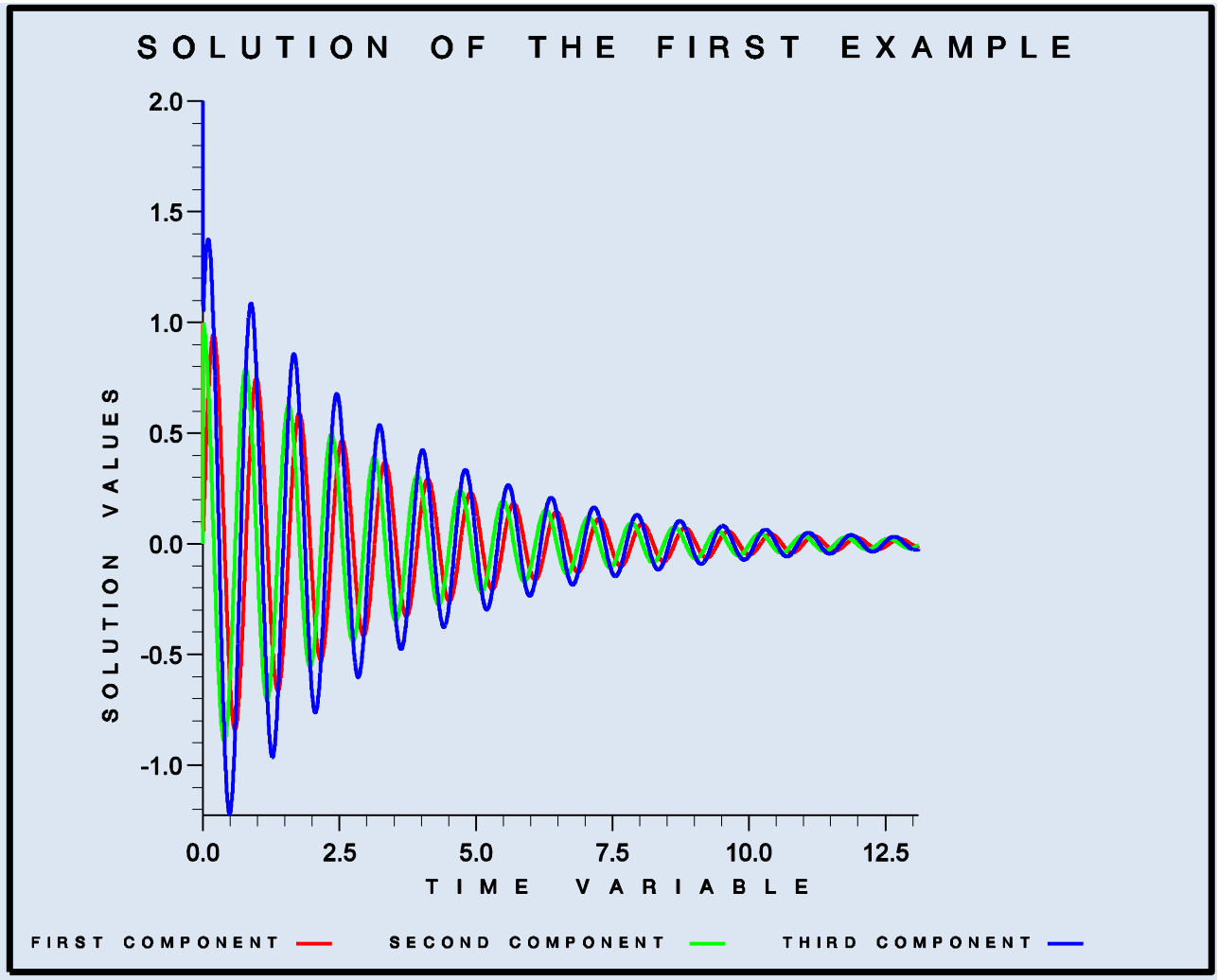

Figure 2.5

Plots of the three components of the solution of the system of ODEs defined by (2.23) – (2.26). The analytical solution is known in this example and is given by the formulae (2.27) – (2.29). The real eigenvalue of matrix $\mathbf{A}$ is much larger, in absolute value, than the two complex eigenvalues; see (2.30). In the program, by which the above plot is produced, the first-order one-stage Explicit Runge-Kutta Method is used with $\mathbf{h = 10^{-5}}$ and the maximal error found during this run was approximately equal to $\mathbf{6.63 * 10^{-4}}$.

2.5.2. Numerical example with large complex eigenvalues

Consider the linear system of ordinary differential equations (ODEs) given by

$$(2.31) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{A\,y + b}, \quad \mathbf{t} \in [\mathbf{0, 13.1072}], \quad \mathbf{y = (y_1, y_2, y_3)^T}, \quad \mathbf{y(0) = (1, 3, 0)^T},$$

$$\mathbf{A} \in \mathbb{R}^{\mathbf{3x3}}, \quad \mathbf{b = (-4\,e^{-0.3t}\,\sin 4t,\ -8\,e^{-0.3t}\,\sin 4t,\ 4\,e^{-0.3t}\,\sin 4t)^T}.$$

The elements of matrix $\mathbf{A}$ from (32) are given below:

$$(\mathbf{2.32}) \quad \mathbf{a_{11} = -937.575}, \qquad \mathbf{a_{12} = 562.425}, \qquad \mathbf{a_{13} = 187.575},$$

$$(\mathbf{2.33}) \quad \mathbf{a_{21} = -187.65}, \qquad \mathbf{a_{22} = -187.65}, \qquad \mathbf{a_{23} = -562.35},$$

$$(\mathbf{2.34}) \quad \mathbf{a_{31} = -1124.925}, \qquad \mathbf{a_{32} = 375.075}, \qquad \mathbf{a_{33} = -375.075}.$$

The three components of the exact solution of the problem defined by (2.31) – (2.34) are given by

$$(\mathbf{2.35}) \quad \mathbf{y_1(t) = e^{-750t}\ \sin 750t + e^{-0.3t} \cos 4t},$$

$$(\mathbf{2.36}) \quad \mathbf{y_2(t) = e^{-750t}\ \cos 750t + 2e^{-0.3t} \cos 4t},$$

$$(\mathbf{2.37}) \quad \mathbf{y_3(t) = e^{-750t}\ (\sin 750t + \cos 750t) - e^{-0.3t} \cos 4t}.$$

It should be mentioned here that the eigenvalues of matrix $\mathbf{A}$ from (32) are given by

$$(\mathbf{2.38}) \quad \boldsymbol{\mu_1 = -750 + 750i}, \quad \boldsymbol{\mu_2 = -750 - 750i}, \quad \boldsymbol{\mu_3 = -0.3}\ .$$

The absolute value of each of the two complex eigenvalues $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ is much larger than the absolute value of the real eigenvalue $\boldsymbol{\mu_3}$. This means that the computations will be stable when $\boldsymbol{\nu = h\mu_1}$ is inside of the absolute stability region of the numerical method under consideration. Note that this value is located above the real axis (not on it, as in the previous example).

The three components of the solution of the example presented in this sub-section are given in **Fig. 2.6**.

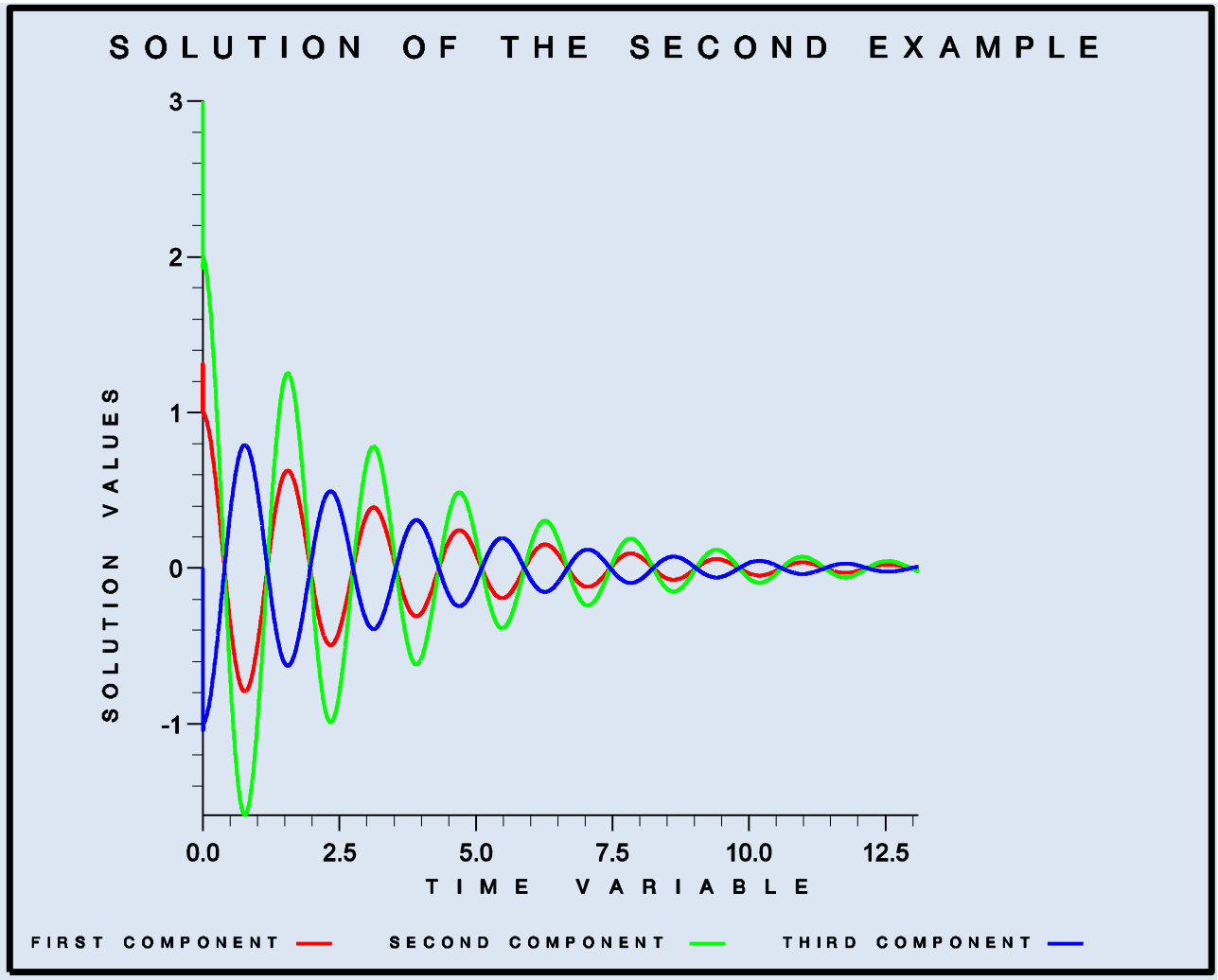

Figure 2.6

Plots of the three components of the solution of the system of ODEs defined by (2.31) – (2.34). The analytical solution is known in this example and is given by the formulae (2.35) – (2.37). The complex eigenvalues of matrix $\mathbf{A}$ are much larger, in absolute value, than the real eigenvalue; see (2.38). In the program, by which the above plot is produced, the first-order one-stage Explicit Runge-Kutta Method is used with $\mathbf{h = 10^{-5}}$ and the maximal error found during this run was approximately equal to $\mathbf{4.03 * 10^{-5}}$.

2.5.3. Non-linear numerical example

Consider the non-linear system of two ordinary differential equations (ODEs) given by

$$(2.39) \qquad \frac{dy_1}{dt} = \frac{1}{y_1} - y_2 \frac{e^{t^2}}{t^2} - t,$$

$$(2.40) \qquad \frac{dy_2}{dt} = \frac{1}{y_2} - e^{t^2} - 2t\, e^{-t^2} .$$

The integration interval is $[0.9, 2.21072]$ and the initial values are

$$(2.41) \qquad y_1(0.9) = \frac{1}{0.9}, \qquad y_2(0.9) = e^{-0.9^2} .$$

The exact solution is given by

$$(2.42) \qquad y_1(t) = \frac{1}{t}, \qquad y_2(0.9) = e^{-t^2} .$$

The eigenvalues of the Jacobian matrix of the function from the right-hand-side of the system of ODEs defined by (2.39) and (2.40) are given by

$$(2.43) \qquad \mu_1 = -\frac{1}{y_1^2}, \qquad \mu_2 = -\frac{1}{y_2^2}.$$

The following expressions can be obtained by inserting the values of the exact solution from (2.42) in (2.43):

$$(2.44) \qquad \mu_1(t) = -t^2, \qquad \mu_2(t) = -e^{2t^2} .$$

It is clear now that in the beginning of the time-interval the problem is non-stiff, but it becomes stiffer and stiffer as the value of the independent variable t grows. At the end of the integration we have $|\mu_2(2.21072)| \approx 17581$ and since the eigenvalues are real, the stability requirement is satisfied if $h|\mu_2| \leq L$ where L is the length of the stability interval on the real axis for the numerical method under consideration.

The two components of the solution of the example presented in this sub-section are given in **Fig. 2.7.**

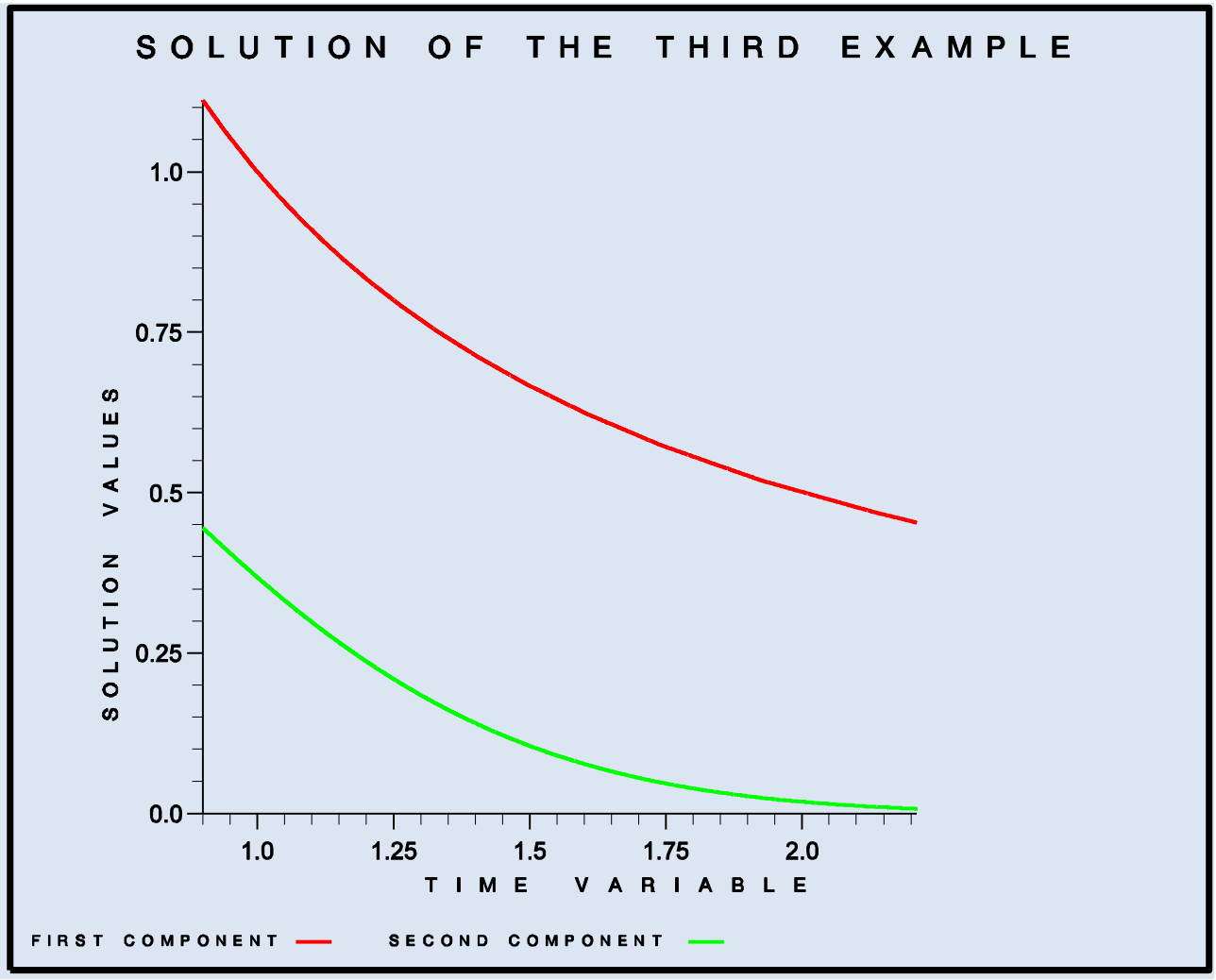

Figure 2.7

Plots of the two components of the solution of the system of ODEs defined by (2.39) – (2.41) with $\mathbf{t} \in [\mathbf{0.9}, \mathbf{2.21072}]$. The exact solution is given in (2.42). The eigenvalues of the Jacobian matrix are real; see (2.43). In the program, by which the above plot is produced, the first-order one-stage Explicit Runge-Kutta method is used with $\mathbf{h} = \mathbf{10^{-6}}$ and the maximal error found during this run was approximately equal to $\mathbf{2.93} * \mathbf{10^{-7}}$.

2.6. Organization of the computations

The integration interval, which is $[\mathbf{0}, \mathbf{13.1072}]$ for the first two examples and $[\mathbf{0.9}, \mathbf{2.21072}]$ for the third one, was divided into $\mathbf{128}$ equal sub-intervals and the accuracy of the results obtained by any of the selected numerical methods was evaluated at the end of each sub-interval. Let $\bar{\mathbf{t}}_\mathbf{j}$, where $\mathbf{j} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{128}$, be the end of any of the $\mathbf{128}$ sub-intervals. Then the following formula is used to evaluate the accuracy achieved by the selected numerical method at this point:

$$(2.45) \quad \mathbf{ERROR_j} = \frac{\sqrt{\sum_{i=1}^{s}\left(y_i(\bar{t}_j) - \bar{y}_{ij}\right)^2}}{\max\left[\sqrt{\sum_{i=1}^{s}\left(y_i(\bar{t}_j)\right)^2}\ , 1.0\right]} \ .$$

The value of parameter $\mathbf{s}$ is $\mathbf{3}$ in the first two examples, while $\mathbf{s = 2}$ is used in the third one. The values $\bar{y}_{ij} \approx y_i(\bar{t}_j)$ are approximations of the exact solution that are calculated by the selected numerical method at time $\bar{t}_j$ (where $\bar{t}_j$ is the end of any of the $\mathbf{128}$ sub-intervals mentioned above).

The global (called sometimes also total) error is computed as

$$(2.46) \quad \mathbf{ERROR} = \max_{j=1,2,\ \ldots,\ 128} \left(\mathbf{ERROR_j}\right) \ .$$

Ten runs were performed with eight numerical methods (four Explicit Runge-Kutta Methods and the combinations of any of the Explicit Runge-Kutta Methods with the Richardson Extrapolation).

The first of the ten runs was carried out by using $\mathbf{h = 0.00512}$ and $\mathbf{h = 0.000512}$ for the first two examples and for the third one respectively. In each of the next nine runs the stepsize is halved (which leads automatically to performing twice more time-steps).

2.7. Particular numerical methods used in the experiments

As already mentioned, there exists only one first-order one-stage Explicit Runge-Kutta Method (called also the **Forward Euler Formula** or the Explicit Euler Method), which is given by

$$(2.47) \quad y_n = y_{n-1} + h\, f(t_{n-1}, y_{n-1}) \ .$$

If $\mathbf{m}$-stage Explicit Runge-Kutta Methods of order $\mathbf{p}$ with $\mathbf{p = m}$ and $\mathbf{p = 2, 3, 4}$ are used, then the situation changes. In this case, for each $\mathbf{p = m = 2, 3, 4}$ there exists a large class of Explicit Runge-Kutta Methods. The class depends on one parameter for $\mathbf{p = 2}$, while classes dependent on two parameters appear for $\mathbf{p = 3}$ and $\mathbf{p = 4}$. All particular methods from such a class have the same stability polynomial and, therefore, the same absolute stability region. This is why it was not necessary until now to specify which particular numerical method was selected in any of these three cases, because in the previous sections of this chapter we were primarily interested in comparing the absolute stability regions of any of the studied by us Explicit Runge-Kutta Methods with the corresponding absolute stability regions that are obtained when the Richardson Extrapolation is additionally used. However, it is necessary to select at least one particular method from each of the three classes when

numerical experiments are to be carried out. The particular numerical methods that were used in the numerical solution of the examples discussed in the previous sections are listed below.

The following method was chosen from the class of the **second-order two-stage** Explicit Runge-Kutta Methods (it is called sometimes the **Improved Euler Method**; see **Lambert, 1991**):

$$\mathbf{k_1} = \mathbf{f}(\mathbf{t_{n-1}}, \mathbf{y_{n-1}}), \tag{2.48}$$

$$\mathbf{k_2} = \mathbf{f}(\mathbf{t_{n-1}} + \mathbf{h}, \mathbf{y_{n-1}} + \mathbf{hk_1}), \tag{2.49}$$

$$\mathbf{y_n} = \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{h}\,(\mathbf{k_1} + \mathbf{k_2})\,. \tag{2.50}$$

The method selected from the class of the **third-order three-stage** Explicit Runge-Kutta Methods is defined as follows (in fact, **this is the numerical method derived by Karl Heun**; **Heun, 1900**):

$$\mathbf{k_1} = \mathbf{f}(\mathbf{t_{n-1}}, \mathbf{y_{n-1}}), \tag{2.51}$$

$$\mathbf{k_2} = \mathbf{f}\left(\mathbf{t_{n-1}} + \frac{\mathbf{1}}{\mathbf{3}}\mathbf{h}, \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{3}}\mathbf{hk_1}\right), \tag{2.52}$$

$$\mathbf{k_3} = \mathbf{f}\left(\mathbf{t_{n-1}} + \frac{\mathbf{2}}{\mathbf{3}}\mathbf{h}, \mathbf{y_{n-1}} + \frac{\mathbf{2}}{\mathbf{3}}\mathbf{hk_2}\right), \tag{2.53}$$

$$\mathbf{y_n} = \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{4}}\,\mathbf{h}\,(\mathbf{k_1} + \mathbf{3k_3})\,. \tag{2.54}$$

One of the most popular methods from the class of the fourth-order four-stage Explicit Runge-Kutta Methods is chosen for our study (this method is so popular that, when one sees a reference to a problem having been solved by *"the Runge-Kutta method"*, it is almost certainly that the method presented below has actually been used; see also **Lambert, 1991**):

$$\mathbf{k_1} = \mathbf{f}(\mathbf{t_{n-1}}, \mathbf{y_{n-1}}), \tag{2.55}$$

$$\mathbf{k_2} = \mathbf{f}\left(\mathbf{t_{n-1}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{h}, \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{hk_1}\right), \tag{2.56}$$

$$(2.57) \quad \mathbf{k_3} = \mathbf{f}\left(\mathbf{t_{n-1}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{h}, \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{hk_2} \right),$$

$$(2.58) \quad \mathbf{k_4} = \mathbf{f}(\mathbf{t_{n-1}} + \mathbf{h}, \mathbf{y_{n-1}} + \mathbf{hk_3}),$$

$$(2.59) \quad \mathbf{y_n} = \mathbf{y_{n-1}} + \frac{\mathbf{1}}{\mathbf{4}}\ \mathbf{h}\ (\mathbf{k_1} + \mathbf{2k_2} + \mathbf{2k_3} + \mathbf{k_4})\ .$$

The numerical results, which will be presented in the next section, were obtained by using both the introduced above three particular Explicit Runge-Kutta Methods as well as the Forward Euler Formula and their combinations with the Richardson Extrapolation. More details about the selected by us particular methods can be found in **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)** and **Lambert (1991)**.

2.8. Numerical results

As mentioned in the previous sections, the three numerical examples that were introduced in Section 2.5 have been run with eight numerical methods: the four particular Explicit Runge-Kutta Methods, which were presented in Section 2.7, and the methods obtained when each of these four Explicit Runge-Kutta Method is combined with the Richardson Extrapolation. The results shown in Table 2.1 –Table 2.6 indicate clearly that

(a) the expected accuracy is nearly always achieved when the stability requirements are satisfied (under the condition that the rounding errors do not interfere with the discretization errors caused by the numerical method which is used; **quadruple precision**, utilizing 32 digits, was applied in all numerical experiments treated in this chapter in order to ensure that this is not happening),

(b) it was verified that the Explicit Runge-Kutta Methods behave (as they should) as methods of **order one,** for the method defined by (2.47), of **order two**, for the method defined by (2.48) – (2.50), of **order three** for the method defined by (2.51) – (2.54), and of **order four**, for the method defined by (2.55) – (2.59),

(c) the combination of each of these four methods with the Richardson Extrapolation behave as a numerical method of increased (by one) order of accuracy

and

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.00512	2560	N.S.	N.S.	N.S.	2.39E-05	N.S.	6.43E-03	N.S.	4.49E-10
2	0.00256	5120	2.01E-01	4.22E-02	4.22E-02	2.99E-06	5.97E-06	7.03E-09	2.46E-08	1.41E-11
3	0.00128	10240	9.21E-02	2.91E-04	2.91E-04	3.73E-07	7.46E-07	4.40E-10	1.54E-09	4.39E-13
4	0.00064	20480	4.41E-02	7.27E-05	7.27E-05	4.67E-08	9.33E-08	2.75E-11	9.62E-11	1.37E-14
5	0.00032	40960	2.16E-02	1.82E-05	1.82E-05	5.83E-09	1.17E-08	1.72E-12	6.01E-12	4.29E-16
6	0.00016	81920	1.07E-02	4.54E-06	4.54E-06	7.29E-10	1.46E-09	1.07E-13	3.76E-13	1.34E-17
7	0.00008	163840	5.32E-03	1.14E-06	1.14E-06	9.11E-11	1.82E-10	6.71E-15	2.35E-14	4.19E-19
8	0.00004	327680	2.65E-03	2.84E-07	2.84E-07	1.14E-11	2.28E-11	4.20E-16	1.47E-15	1.31E-20
9	0.00002	655360	1.33E-03	7.10E-08	7.10E-08	1.42E-12	2.85E-12	2.62E-17	9.18E-17	4.09E-22
10	0.00001	1310720	6.66E-04	1.78E-08	1.78E-08	1.78E-13	3.56E-13	1.64E-18	5.74E-18	1.28E-23

Table 2.1

Accuracy results (error estimations) achieved when **the first example** from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being applied in this experiment). **"N.S."** means that the numerical method is not stable for the stepsize used. "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.00512	2560	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.
2	0.00256	5120	N. A.	N. A.	N. A.	7.99	N. A.	very big	N. A.	31.84
3	0.00128	10240	2.18	145.02	145.02	8.02	8.00	15.98	15.97	32.12
4	0.00064	20480	2.09	4.00	4.00	7.99	8.00	16.00	16.01	32.04
5	0.00032	40960	2.04	3.99	3.99	8.01	7.97	15.99	16.01	31.93
6	0.00016	81920	2.02	4.01	4.01	8.00	8.01	16.07	15.98	32.01
7	0.00008	163840	2.01	3.98	3.98	8.00	8.02	15.95	16.00	31.98
8	0.00004	327680	2.01	4.01	4.01	7.99	7.98	15.97	15.99	31.98
9	0.00002	655360	1.99	4.00	4.00	8.03	8.00	16.03	16.01	32.03
10	0.00001	1310720	2.00	3.99	3.99	7.98	8.01	15.98	15.99	31.95

Table 2.2

Convergent rates (ratios of two consecutive error estimations from Table 2.1) observed when the first example from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being used in this experiment). **"N.A."** means that the convergence rate cannot be calculated (this happens either when the first run is performed or if the computations at the previous runs were not stable). "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.00512	2560	N. S.	N. S.	N. S.	N. S.	N. S.	4.95E-02	N. S.	N. S.
2	0.00256	5120	N. S.	N. S.	N. S.	5.40E-08	N. S.	4.88E-13	N. S.	1.21E-17
3	0.00128	10240	2.37E-02	4.09E-06	6.81E-06	3.22E-11	1.54E-09	3.04E-14	7.34E-13	3.51E-19
4	0.00064	20480	2.58E-03	1.02E-06	1.70E-06	3.99E-12	1.92E-10	1.90E-15	4.59E-14	1.05E-20
5	0.00032	40960	1.29E-03	2.56E-07	4.26E-07	4.97E-13	2.40E-11	1.19E-16	2.87E-15	3.21E-22
6	0.00016	81920	6.45E-04	6.40E-08	1.06E-07	6.21E-14	3.00E-12	7.41E-18	1.79E-16	9.93E-24
7	0.00008	163840	3.23E-04	1.60E-08	2.66E-08	7.75E-15	3.75E-13	4.63E-19	1.12E-17	3.09E-25
8	0.00004	327680	1.61E-04	4.00E-09	6.65E-09	9.68E-16	4.69E-14	2.89E-20	7.00E-19	9.62E-27
9	0.00002	655360	8.06E-05	9.99E-10	1.66E-09	1.21E-16	5.86E-15	1.81E-21	4.38E-20	3.00E-28
10	0.00001	1310720	4.03E-05	2.50E-10	4.16E-10	1.51E-17	7.32E-16	1.13E-22	2.73E-21	9.36E-30

Table 2.3

Accuracy results (error estimations) achieved when the second example from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being applied in this experiment). **"N.S."** means that the numerical method is not stable for the stepsize used. "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.00512	2560	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.
2	0.00256	5120	N. A.	N. A.	N. A.	N. A.	N. A.	1.01E+11	N. A.	N. A.
3	0.00128	10240	N. A.	N. A.	N. A.	167.70	N. A.	16.05	N. A.	34.47
4	0.00064	20480	9.96	4.01	4.01	8.07	8.02	16.00	15.99	33.43
5	0.00032	40960	2.00	3.98	3.99	8.03	8.00	15.97	15.99	32.71
6	0.00016	81920	2.00	4.00	4.02	8.00	8.00	16.06	16.03	32.33
7	0.00008	163840	2.00	4.00	3.98	8.01	8.00	16.00	15.98	32.14
8	0.00004	327680	2.01	4.00	4.00	8.01	8.00	16.02	16.00	32.12
9	0.00002	655360	2.00	4.00	4.01	8.07	8.00	15.97	15.98	32.07
10	0.00001	1310720	2.00	4.00	3.99	8.01	8.01	16.02	16.04	32.05

Table 2.4

Convergent rates (ratios of two consecutive error estimations from Table 2.3) observed when the second example from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being used in this experiment). **"N.A."** means that the convergence rate cannot be calculated (this happens either when the first run is performed or if the computations at the previous runs were not stable). "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.000512	2560	N. S.	N. S.	N. S.	N. S.	N. S.	N. S.	N. S.	N. S.
2	0.000256	5120	N. S.	2.08E-02	N. S.	1.04E-09	N. S.	1.15E-03	N. S.	2.48E-10
3	0.000128	10240	3.76E-05	1.87E-03	8.23E-03	2.08E-10	4.17E-10	4.23E-11	1.03E-09	1.38E-11
4	0.000064	20480	1.88E-05	1.04E-09	1.26E-09	3.26E-11	5.78E-11	1.94E-12	2.68E-11	3.77E-13
5	0.000032	40960	9.39E-06	2.59E-10	3.14E-10	3.93E-12	6.07E-12	1.07E-13	1.29E-12	1.06E-14
6	0.000016	81920	4.70E-06	6.48E-11	7.85E-11	4.68E-13	6.70E-13	6.29E-15	7.08E-14	3.10E-16
7	0.000008	163840	2.35E-06	1.62E-11	1.96E-11	5.68E-14	7.84E-14	3.80E-16	4.13E-15	9.36E-18
8	0.000004	327680	1.17E-06	4.05E-12	4.90E-12	6.98E-15	9.47E-15	2.34E-17	2.50E-16	2.88E-19
9	0.000002	655360	5.87E-07	1.01E-12	1.23E-12	8.65E-16	1.16E-15	1.45E-18	1.53E-17	8.91E-21
10	0.000001	1310720	2.93E-07	2.53E-13	3.06E-13	1.08E-16	1.44E-16	9.00E-20	9.50E-19	2.77E-22

Table 2.5

Accuracy results (error estimations) achieved when the third example from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being applied in this experiment). **"N.S."** means that the numerical method is not stable for the stepsize used. "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

Run	Stepsize	Steps	ERK1	ERK1R	ERK2	ERK2R	ERK3	ERK3R	ERK4	ERK4R
1	0.000512	2560	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.
2	0.000256	5120	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.
3	0.000128	10240	N. A.	N. R.	N. A.	5.00	N. A.	N. R.	N. A.	17.97
4	0.000064	20480	2.00	N. R.	N. R.	6.38	7.28	21.80	38.43	36.60
5	0.000032	40960	2.00	4.02	4.01	8.30	9.52	18.13	20.78	35.57
6	0.000016	81920	2.00	4.00	4.00	8.40	9.06	17.01	18.22	34.19
7	0.000008	163840	2.00	4.00	4.01	8.24	8.55	16.55	17.14	33.12
8	0.000004	327680	2.01	4.00	4.00	8.14	8.28	16.24	16.52	32.50
9	0.000002	655360	1.99	4.01	3.98	8.07	8.16	16.14	16.34	32.32
10	0.000001	1310720	2.00	3.99	4.02	8.09	8.06	16.11	16.11	32.17

Table 2.6

Convergent rates (ratios of two consecutive error estimations from Table 2.5) observed when the third example from Section 2.5 is solved by the eight numerical methods on a SUN computer (quadruple precision being used in this experiment). **"N.A."** means that the convergence rate cannot be calculated (this happens either when the first run is performed or if the computations at the previous runs were not stable). **"NR"** means that the calculated convergence rate will not be reliable (the stepsize is too large). "**ERKp**", $\mathbf{p = 1, 2, 3, 4}$, means Explicit Runge-Kutta Method of order $\mathbf{p}$. **"ERKpR"** refers to the Explicit Runge-Kutta Method of order $\mathbf{p}$ combined with the Richardson Extrapolation.

(d) for some large stepsizes, for which the Explicit Runge-Kutta Methods are unstable when these are used directly, their combinations with the Richardson Extrapolation produced good results.

Some more precise information about the performed runs and about the results is given below. First, it should be emphasized that the accuracy results, which were obtained when the eight numerical methods for the solution of systems of ODEs are used, are given in **Table 2.1** for the first example, in **Table 2.3** for the second one and in **Table 2.5** for the third (non-linear) example. Convergence rates observed for the eight tested numerical methods are shown in **Table 2.2**, **Table 2.4** and **Table 2.6** respectively.

Several additional conclusions can immediately be drawn by investigating carefully the results that are presented in Table 2.1 - Table 2.6:

(e) The non-linear example is not causing problems. As one should expect, the results for the first and the second stepsizes are not stable when the Explicit Runge-Kutta Methods are run, because for large values of $\mathbf{t}$ the inequality $\mathbf{h}|\boldsymbol{\mu}_2(\mathbf{t})| > \mathbf{L}$ holds ($\mathbf{L}$ being the length of the absolute stability interval on the real axis), and, thus, the stability requirement is not satisfied. The condition $\mathbf{h}|\boldsymbol{\mu}_2(\mathbf{t})| \leq \mathbf{L}$ is not satisfied either for all values of $\mathbf{t}$ for the next stepsize, but this happens only in the very end of the integration and the instability had not succeeded to manifest itself. The results become considerably better when the Richardson Extrapolation is used.

(f) The combination of the first-order one-stage Runge-Kutta method and the Richardson Extrapolation gives nearly the same results as the second-order two-stage Runge-Kutta method. It is seen that the stability regions of these two numerical methods are also identical. The results indicate that this property holds not only for the scalar test-example (2.1) proposed by Dahlquist but also for linear systems of ODEs with constant coefficients. Moreover, the explanations given in Section 2.1 indicate that this property perhaps holds also for some more general systems of ODEs.

(g) The results show that the calculated (as ratios of two consecutive error estimations) convergence rates of the Runge-Kutta method of order $\mathbf{p}$ are about $\mathbf{2^p}$ when the stepsize is reduced successively by a factor of two. For the combinations of the Runge-Kutta methods and the Richardson Extrapolation the corresponding convergence rates are approximately equal to $\mathbf{2^{p+1}}$ which means that the order of accuracy is increased by one. This should be expected and, moreover, it is also clearly seen from the tables that the obtained numerical results are nearly perfect. Only when the product of the time-stepsize and the absolute value of the largest eigenvalue is close to the boundary of the absolute stability region there are some deviations from the expected results. For the non-linear example this relationship is not fulfilled for some of the large stepsizes because the condition $\mathbf{h}|\boldsymbol{\mu}_2(\mathbf{t})| \leq \mathbf{L}$ is not satisfied in the very end of the integration interval.

(h) The great power of the Richardson Extrapolation is clearly demonstrated by the results given in Table 2.1. Consider the use of the first-order one-stage Explicit Runge-Kutta method together with the Richardson Extrapolation (denoted as **ERK1R** in the table). The error estimation is $\mathbf{2.91 * 10^{-4}}$ for $\mathbf{h = 0.00128}$ and when $\mathbf{10240}$ time-

steps are performed. Similar accuracy can be achieved by using $\mathbf{131072}$ steps when the first-order one-stage Explicit Runge-Kutta Method, **ERK1,** is used (i.e. the number of time-steps is increased by a factor of $\mathbf{128}$). Of course, for every step performed by the **ERK1** method, the **ERK1R** method performs three steps (one large and two small). Even when this fact is taken into account (by multiplying the number of time-steps for **ERK1R** by three), the **ERK1R** is reducing the number of time-steps performed by **ERK1** by a factor greater than $\mathbf{40}$. The alternative is to use a method of higher order. However, such methods are more expensive and, what is perhaps much more important, a very cheap and rather reliable error estimation can be obtained when the Richardson Extrapolation is used. It is clearly seen (from Table 2.3 and Table 2.5) that the situation is very similar also when the second and the third examples are treated.

(i) In these experiments, it was illustrative to apply **quadruple precision** (working with about 32 digits) in order to demonstrate in a very clear way the ability of the methods to achieve very accurate results when their orders of accuracy are greater than three. However, it should be stressed here that in general it will not be necessary to apply quadruple precision, i.e. the application of the traditionally used double precision will nearly always be quite sufficient.

(j) The so-called **active implementation** (see Section 1.7 and also **Faragó, Havasi and Zlatev, 2010** or **Zlatev, Faragó and Havasi, 2010**) of the Richardson Extrapolation is used in this chapter. In this implementation, at each time-step the improved (by applying the Richardson Extrapolation) value $\mathbf{y_{n-1}}$ of the approximate solution is used in the calculation of $\mathbf{z_n}$ and $\mathbf{w_n}$. One can also apply another approach: the values of the previous approximations of $\mathbf{z_{n-1}}$ and $\mathbf{w_{n-1}}$ can be used in the calculation of $\mathbf{z_n}$ and $\mathbf{w_n}$ respectively and after that one can calculate the Richardson improvement $\mathbf{y_n = (2^p w_n - z_n)/(2^p - 1)}$. As explained in Section 1.7, a **passive implementation** of the Richardson Extrapolation is obtained in this way (in this implementation the improved by the Richardson Extrapolation values of the approximations are calculated at every time-step, but not used in the further computations; they are only stored in order to be used for other purposes). It is obvious that, if the underlying method is absolutely stable for the two stepsizes $\mathbf{h}$ and $\mathbf{0.5h}$, then the passive implementation of the Richardson Extrapolation will also be absolutely stable. However, if it is not absolutely stable (even only for the large time-stepsize), then the results calculated by the passive implementation of the Richardson Extrapolation will be unstable. This is due to the fact that, as stated in Section 1.7, the passive implementation of the Richardson Extrapolation has the same absolute stability properties as those of the underlying method for solving systems of ODEs. Therefore, the results in the first lines of Table 2.1, Table 2.3 and Table 2.5 show very clearly that not only the underlying method but also the passive implementation of the Richardson Extrapolation **may fail** for some large values of the time-stepsize, while the active one is successful. This will happen, because the underlying method is not stable at least for the large stepsize, but the combined method is stable when the active implementation is used (due to the increased stability regions).

2.9. Development of methods with enhanced absolute stability properties

The requirement $\mathbf{p = m}$ was imposed and used in the previous sections of the second chapter. This requirement is very restrictive, because it can be satisfied only for $\mathbf{m \leq 4}$. Therefore it is worthwhile to remove this restriction by considering Explicit Runge-Kutta Methods under the condition $\mathbf{p < m}$ and to try to develop numerical methods with enhanced stability properties. If the condition $\mathbf{p < m}$ is imposed, then the stability polynomial given in (2.10) should be replaced with the following formula:

$$\mathbf{(2.60)} \qquad \mathbf{R(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \cdots + \frac{\nu^p}{p!} + \frac{\nu^{p+1}}{\gamma_{p+1}^{(m,p)}(p+1)!} + \cdots + \frac{\nu^m}{\gamma_{p+1}^{(m,p)}(m)!}}.$$

It is seen that there are $\mathbf{m - p}$ free parameters $\boldsymbol{\gamma_{p+1}^{(m,p)}, \gamma_{p+2}^{(m,p)}, \ldots, \gamma_m^{(m,p)}}$ in (2.60). These parameters will be used to search for methods with **large** absolute stability regions. Two special cases will be studied in this section in order to facilitate the presentation of the results:

Case 1: $\mathbf{p = 3}$ and $\mathbf{m = 4}$

and

Case 2: $\mathbf{p = 4}$ and $\mathbf{m = 6}$.

Three major topics will be explained in this section. We shall show **first** that one can find large classes of numerical methods with enhanced stability properties for each of these two cases. Each representative of any of the two classes have the same absolute stability region as all the other representatives. **After that**, we shall select particular methods in each of the obtained classes (we shall explain carefully that this is a rather complicated procedure) and perform some numerical experiments. **Finally**, some possibilities for improving further the results will be sketched.

2.9.1. Derivation of two classes of numerical methods with good stability properties.

Consider first **Case 1** of the two cases formulated above, i.e. choose $\mathbf{p = 3}$ and $\mathbf{m = 4}$. Then (2.60) is reduced to

$$\mathbf{(2.61)} \qquad \mathbf{R(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \frac{\nu^4}{\gamma_4^{(4,3)}\,4!}}.$$

A systematic search for numerical methods with good absolute stability properties was carried out by comparing the stability regions obtained for $\boldsymbol{\gamma_4^{(4,3)} = 0.00(0.01)5.00}$. It is clear that the number of

tests, $\mathbf{500}$, was very large. Therefore, we reduced the number of the investigated tests by introducing two requirements:

(a) the length of the stability interval on the negative part of the real axis should be greater than $\mathbf{6.00}$

and

(b) the highest point of the absolute stability region should be at a distance not less than $\mathbf{4.00}$ from the real axis.

The number of tests was reduced very considerably by these two restrictions and it was found that the choice $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$ is very good. The absolute stability regions obtained by this value of the free parameter are given in **Fig. 2.8**. The Explicit Runge-Kutta Methods obtained with $\mathbf{p = 3}$, $\mathbf{m = 4}$ and $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$ form a large class of numerical methods. Each representative of this class has the same absolute stability region, the absolute stability region, which is limited by the red curve in **Fig. 2.8**. The corresponding new methods (the combinations of any of the ERKMs with the Richardson Extrapolation are also forming a large class of methods; each representative of the latter class has the same absolute stability region; this region is limited by the green curve in **Fig. 2.8**).

Let us call **Method A** **<u>any</u>** of the Explicit Runge-Kutta Methods from the class determined with $\mathbf{p = 3}$, $\mathbf{m = 4}$ and $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$. The comparison of the absolute stability regions shown in **Fig. 2.8** with those which were presented in **Fig. 2.3** allows us to draw the following three conclusions:

(a) The absolute stability region of **Method A** is considerably smaller than the corresponding absolute stability region of the combination of **Method A** with the Richardson Extrapolation.

(b) The absolute stability region of **Method A** is larger than the corresponding absolute stability region of the Explicit Runge-Kutta Method obtained with $\mathbf{p = m = 3}$.

(c) When **Method A** is combined with the Richardson Extrapolation then its absolute stability region is larger than the corresponding absolute stability region of the combination of the Richardson Extrapolation with the Explicit Runge-Kutta Method obtained $\mathbf{p = m = 3}$.

The stability regions could be further enlarged if the second choice, the choice with $\mathbf{m - p = 2}$, is made, because the number of free parameters was increased from one to two in this case. If $\mathbf{p = 4}$ and $\mathbf{m = 6}$ is applied, then the stability polynomial (2.60) can be written as

$$\mathbf{(2.62)} \qquad \mathbf{R(\nu) = 1 + \nu + \frac{\nu^2}{2!} + \frac{\nu^3}{3!} + \frac{\nu^4}{4!} + \frac{\nu^5}{\gamma_5^{(6,4)}\, 5!} + \frac{\nu^6}{\gamma_6^{(6,4)}\, 6!}}.$$

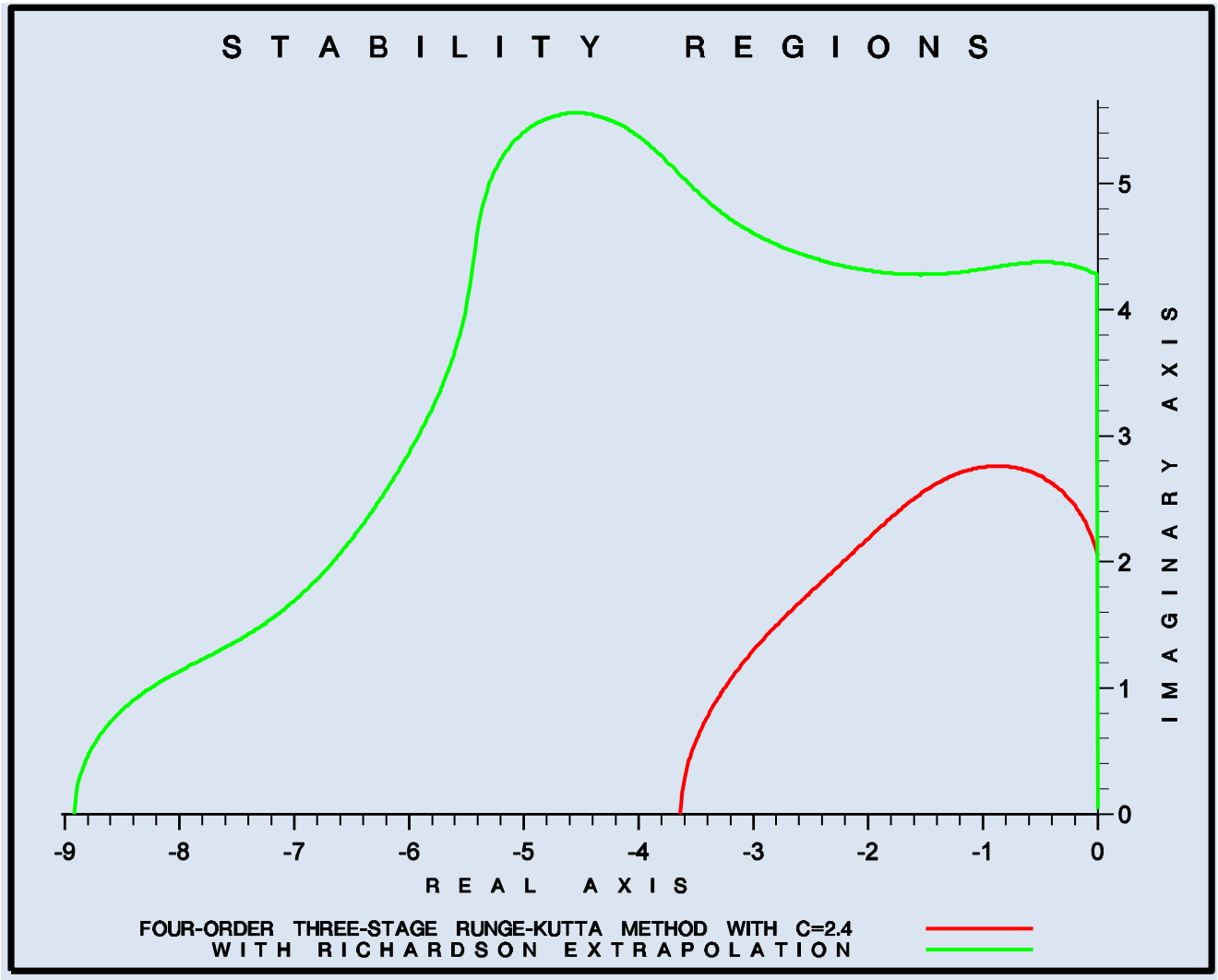

Figure 2.8

Stability regions of any representative of the class of explicit third-order four-stage Runge-Kutta (**ERK43)** methods with $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$ and its combination with the Richardson Extrapolation.

Formula (2.62) shows that there are indeed two free parameters now. A systematic search for numerical methods with good absolute stability regions was performed also in this case. The search was much more complicated and time-consuming. It was carried out by using $\boldsymbol{\gamma_5^{(6,4)} = 0.00(0.01)5.00}$ and $\boldsymbol{\gamma_6^{(6,4)} = 0.00(0.01)5.00}$. The number of tests, $\boldsymbol{250000}$, was much larger than the number of tests in the previous case. Therefore, we reduced again the number of the investigated tests by introducing two extra requirements:

(a) the length of the stability interval on the negative part of the real axis should be greater than $\boldsymbol{12.00}$

and

(b) the highest point of the absolute stability region should be at a distance not less than $\mathbf{5.00}$ from the real axis.

The number of tests was reduced very considerably in this way and it became possible to find out that the choice $\boldsymbol{\gamma_5^{(6,4)} = 1.42}$ and $\boldsymbol{\gamma_6^{(6,4)} = 4.86}$ gives very good results. The absolute stability regions for the class of Explicit Runge-Kutta Methods found with these two values of the free parameters are given in **Fig. 2.9** . Also in this case, the Explicit Runge-Kutta Methods created by using the parameters $\mathbf{p = 4}$, $\mathbf{m = 6,}$ $\boldsymbol{\gamma_5^{(6,4)} = 1.42}$ and $\boldsymbol{\gamma_6^{(6,4)} = 4.86}$ form a large class of numerical methods. Each representative of this class has the same absolute stability region, the absolute stability region limited by the red curve in **Fig. 2.9**. The corresponding new methods (the combinations of any of the ERKMs with the Richardson Extrapolation) are also forming a large class of methods; each representative of the latter class has the same absolute stability region; the region limited by the green curve in **Fig. 2.9**.

Let us call **Method B** **any** representative of the class of the Explicit Runge-Kutta Methods determined by choosing: $\mathbf{p = 4}$, $\mathbf{m = 6,}$ $\boldsymbol{\gamma_5^{(6,4)} = 1.42}$ and $\boldsymbol{\gamma_6^{(6,4)} = 4.86}$ **.** Then the following three statements are true:

(A) The absolute stability region of **Method B** is considerably smaller than the corresponding absolute stability region of the combination of **Method B** with the Richardson Extrapolation.

(B) The absolute stability region of **Method B** is larger than the corresponding absolute stability region of the Explicit Runge-Kutta Method obtained with $\mathbf{p = m = 4}$

and

(C) when **Method B** is applied together with the Richardson Extrapolation, then its absolute stability region is larger than the corresponding absolute stability region of the combination of the Richardson Extrapolation with the Explicit Runge-Kutta Method obtained with $\mathbf{p = m = 4}$.

The lengths of the absolute stability intervals on the negative real axis of **Method A, Method B** and two traditionally used Explicit Runge-Kutta Method (applied also in the previous sections of this chapter) are given in **Table 2.7** together with corresponding absolute stability intervals of their combinations with the Richardson Extrapolation.

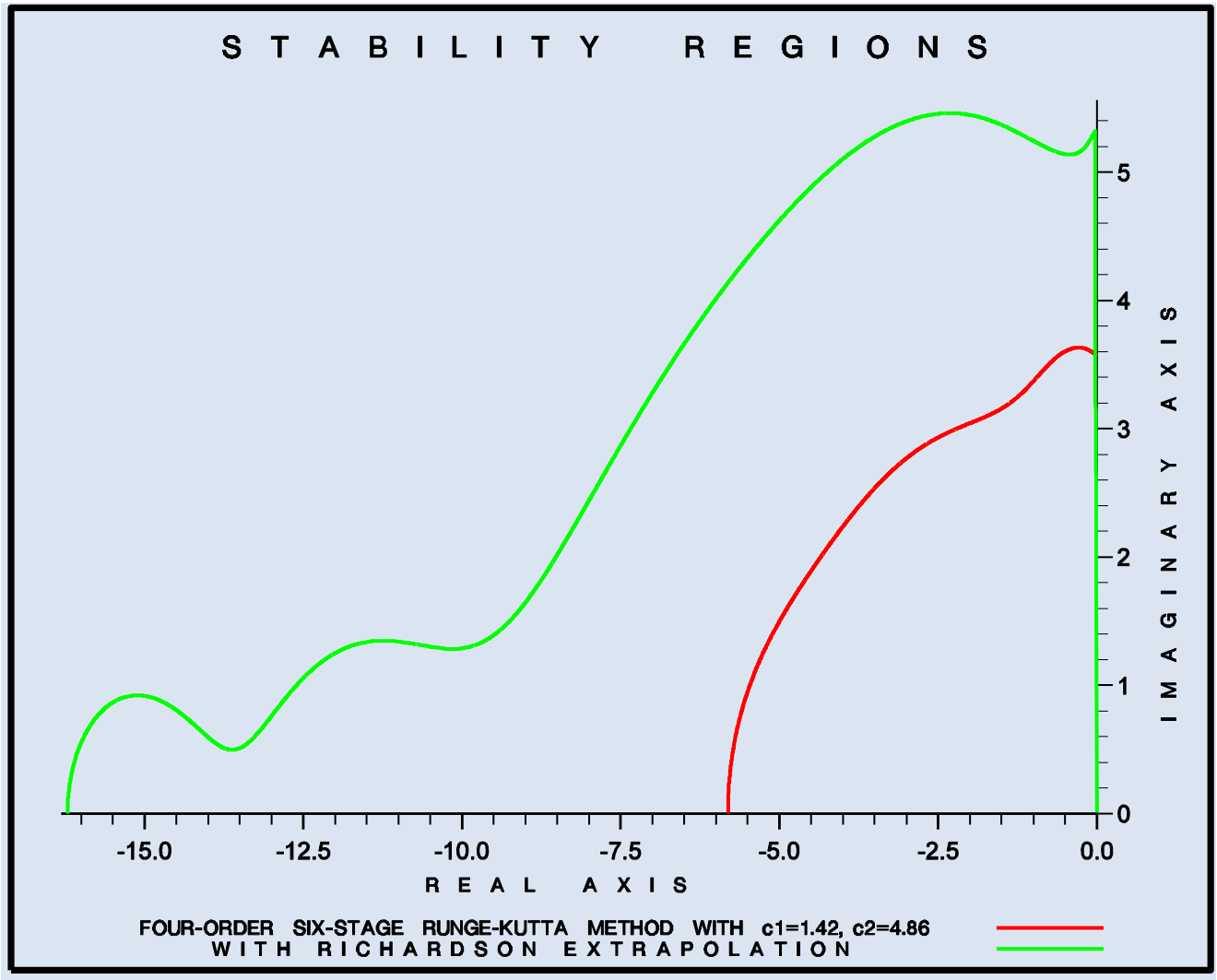

Figure 2.9

Stability regions of any representative of the class of Explicit Runge-Kutta methods determined with $\mathbf{p = 4,\ m = 6,\ } \gamma_5^{(6,4)} = 1.42$ and $\gamma_6^{(6,4)} = 4.86$ together with its combination with the Richardson Extrapolation.

Numerical method	Direct implementation	Combined with Richardson Extrapolation
p = m = 3	2.51	4.02
Method A	**3.65**	**8.93**
p = m = 4	2.70	6.40
Method B	**5.81**	**16.28**

Table 2.7

Lengths of the absolute stability intervals on the negative real axis of four Explicit Runge-Kutta Methods and their combinations with the Richardson Extrapolation.

It is seen from Table 2.7 that

(a) the length of the absolute stability interval on the negative part of the real axis of the new methods, which consist of the combination of any Explicit Runge-Kutta Method obtained with $\mathbf{p = 4}$, $\mathbf{m = 6}$, $\boldsymbol{\gamma_5^{(6,4)} = 1.42}$ and $\boldsymbol{\gamma_6^{(6,4)} = 4.86}$ and the Richardson Extrapolation, is more than six times longer than the length of the of the absolute stability interval of the Explicit Runge-Kutta methods with $\mathbf{p = m = 4}$ when this method is used directly,

(b) it follows from conclusion **(a)** that for mildly stiff problems, in which the real eigenvalues of the Jacobian matrix of function **f** are dominating over the complex eigenvalues, the new numerical method, the combination of a fourth-order six-stage Explicit Runge-Kutta Method with the Richardson Extrapolation, could be run with a time-stepsize, which is by a factor of **six** larger than that for a fourth-order four-stage explicit Runge-Kutta method.

However, this success is not unconditional: two extra stages must be added in order to achieve the improved absolute stability regions, which makes the new numerical method more expensive. It is nevertheless clear that a reduction of the number of time-steps by a factor approximately equal to six will as a rule be a sufficiently good compensation for the use of two additional stages.

The research for developing Explicit Runge-Kutta Methods with $\mathbf{p < m}$, which have good absolute stability properties when they are combined with the Richardson Extrapolation, is by far not finished yet. The results presented in this section only indicate that one should expect good results, but it is necessary

(a) to optimize further the search for methods with good stability properties,

(b) to select particular methods with good accuracy properties among the classes of method with good stability properties obtained after the application of some optimization tool in the search

and

(c) to carry out much more numerical experiments in order to verify the usefulness of the results in some realistic applications.

These additional tasks will be further discussed in the remaining part of Chapter 2.

2.9.2. Selecting particular numerical methods for Case 1: $\mathbf{p = 3}$ and $\mathbf{m = 4}$

It was pointed out above that the numerical methods, the absolute stability region of which were shown in **Fig. 2.8,** form a large class of Explicit Runge-Kutta Methods. It is necessary now to find a good representative of this class. We are mainly interested in finding a method which has good accuracy properties. This is a very difficult task. Three groups of requirements must be satisfied:

(a) four order conditions are to be derived,

(b) one must use the optimal value, $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$ for which the largest absolute stability region is obtained (see the previous sub-section)

and, finally,

(c) several relations between the coefficients of the four-stage ERKMs have to be taken into account.

This leads to the solution of **a non-linear algebraic system of 8 equations with 13 unknowns**. The equations are listed below:

$$\mathbf{(2.63)} \quad \mathbf{c_1 + c_2 + c_3 + c_4 = 1}\,,$$

$$\mathbf{(2.64)} \quad \mathbf{c_2 a_2 + c_3 a_3 + c_4 a_4 = \frac{1}{2}}\,,$$

$$\mathbf{(2.65)} \quad \mathbf{c_2 (a_2)^2 + c_3 (a_3)^2 + c_3 (a_3)^2 = \frac{1}{3}}\,,$$

$$\mathbf{(2.66)} \quad \mathbf{c_3 b_{32} a_2 + c_4 (b_{42} a_2 + b_{43} a_3) = \frac{1}{6}}\,.$$

$$\mathbf{(2.67)} \quad \mathbf{c_4 b_{43} b_{32} a_2 = \frac{1}{\gamma_4^{(4,3)}} \frac{1}{120}}\,.$$

$$\mathbf{(2.68)} \quad \mathbf{b_{21} = a_2}\,,$$

$$\mathbf{(2.69)} \quad \mathbf{b_{31} + b_{32} = a_3}\,.$$

$$\mathbf{(2.70)} \quad \mathbf{b_{41} + b_{42} + b_{43} = a_4}\,.$$

The relationships (2.63)-(2.66) are the order conditions (needed to obtain an Explicit Runge-Kutta Method, the order of accuracy of which is **three**). The equality (2.67) is used in order to obtain good stability properties. The last three equalities, equalities (2.68)-(2.70), are giving some relations between the coefficients of the Runge-Kutta methods.

It can easily be verified that the conditions (2.63) – (2.70) are satisfied if the coefficients are chosen in the following way:

$$(\mathbf{2.71}) \quad \mathbf{c_1 = \frac{1}{6}}, \quad \mathbf{c_2 = \frac{1}{3}}, \quad \mathbf{c_3 = \frac{1}{3}}, \quad \mathbf{c_4 = \frac{1}{6}},$$

$$(\mathbf{2.72}) \quad \mathbf{a_2 = \frac{1}{2}}, \quad \mathbf{a_3 = \frac{1}{2}}, \quad \mathbf{a_4 = 1},$$

$$(\mathbf{2.73}) \quad \mathbf{b_{21} = \frac{1}{2}}, \quad \mathbf{b_{31} = 0}, \quad \mathbf{b_{32} = \frac{1}{2}}, \quad \mathbf{b_{41} = 0}, \quad \mathbf{b_{42} = 1 - \frac{1}{2.4}}, \quad \mathbf{b_{43} = \frac{1}{2.4}}.$$

It should be noted that if the last two coefficients $\mathbf{b_{42}}$ and $\mathbf{b_{43}}$ in (2.73) are replaced with

$$(\mathbf{2.74}) \quad \mathbf{b_{42} = 0}, \quad \mathbf{b_{43} = 1},$$

then the classical fourth-order four stages Explicit Runge-Kutta Method will be obtained; this method is defined by the formulae (2.55)-(2.59) in Section 2.7.

The order of the method determined by the coefficients given in (2.71)-(2.73) is lower than the order of the classical method (three instead of four), but its absolute stability region is (as mentioned above) considerably larger. The absolute stability regions of the derived by us optimal **ERK43** method and its combination with the Richardson Extrapolation are given in **Fig. 2.8.** It will be illustrative to compare these regions with the corresponding absolute stability regions of the classical **ERK33** method (the third-order three stage Explicit Runge-Kutta Method) and with the combination of the **ERK33** method with the Richardson Extrapolation. These plots are shown in **Fig. 2.3**.

The first of the three numerical examples presented in Section 2.5 was used in order to test the efficiency of the **ERK43** method.

The organization of the computations applied to calculate the results given below, in **Table 2.8,** is described in detail in Section 2.6. It is not necessary here to repeat these details, but it should be mentioned that **12 runs** were performed in these tests (not 10 runs as in the previous sections). We are starting with a stepsize $\mathbf{h = 0.02048}$ and reducing the stepsize by a factor of two after the completion of each run. This means that the stepsize in the last run is again $\mathbf{h = 0.00001}$.

Run	Stepsize	ERK33	ERK44	ERK43	ERK43+RE
1	0.02048	N.S.	N.S.	N.S.	N.S.
2	0.01024	N.S	N.S.	N.S.	N.S.
3	0.00512	N.S.	N.S.	8.43E-03	4.86E-08
4	0.00256	5.97E-06	2.46E-08	3.26E-06	3.04E-09 (15.99)
5	0.00128	7.46E-07 (8.00)	1.54E-09 (15.97)	4.07E-07 (8.01)	1.90E-10 (16.00)
6	0.00064	9.33E-08 (8.00)	9.62E-12 (16.00)	5.09E-08 (8.00)	1.19E-11 (15.97)
7	0.00032	1.17E-08 (7.97)	6.01E-12 (16.01)	6.36E-09 (8.00)	7.42E-13 (16.04)
8	0.00016	1.46E-09 (8.01)	3.76E-13 (15.98)	7.95E-10 (8.00)	4.64E-14 (15.99)
9	0.00008	1.82E-10 (8.02)	2.35E-14 (16.00)	9.94E-11 (8.00)	2.90E-15 (16.00)
10	0.00004	2.28E-11 (7.98)	1.47E-15 (15.99)	1.24E-11 (8.02)	1.81E-16 (16.02)
11	0.00002	2.85E-12 (8.00)	9.18E-17 (16.01)	1.55E-12 (8.00)	1.13E-17 (16.02)
12	0.00001	3.56E-13 (8.01)	5.74E-18 (15.99)	1.94E-13 (7.99)	7.08E-19 (15.96)

Table 2.8

Comparison of the third-order four stages explicit Runge-Kutta (**ERK43**) method $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$ and its combination with the Richardson Extrapolation (**ERK43+RE**) with the traditionally used third-order three stages and fourth-order four stages explicit Runge-Kutta methods (**ERK33** and **ERK44**). "**N.S**" means that the method is not stable (the computations are declared as unstable and stopped when the norm of the calculated solution becomes greater than $\boldsymbol{2.4 * 10^7}$). The convergence rates are given in brackets.

The results presented in **Table 2.8** show clearly that following three conclusions are certainly true:

No.	Conclusions
1	The new numerical method (the third-order four-stage Explicit Runge-Kutta Method with $\boldsymbol{\gamma_4^{(4,3)} = 2.4}$; the **ERK43** method) is both more accurate and more stable than the classical third-order three stages explicit Runge-Kutta method (**ERK33**).
2	The classical fourth-order four stages Explicit Runge-Kutta Method, **ERK44**, is more accurate than the new method (which is very natural because its order of accuracy is higher), but the new method behaves in a reasonably stable way for $\boldsymbol{h = 0.00256}$ where the classical method fails.
3	The combination of the new method with the Richardson extrapolation (**ERK43+RE**) is both more accurate and more stable than the two classical methods (**ERK33** and **ERK44**) and the new method (**ERK43**).

2.9.3. Selecting particular numerical methods for Case 2: $\mathbf{p = 4}$ and $\mathbf{m = 6}$

The methods, which have the absolute stability regions shown in **Fig. 2.9**, form (as the methods the stability regions of which were presented in **Fig. 2.8**) a large class of Explicit Runge-Kutta Methods. It is necessary now to find a good representative of this class. We are also in this sub-section interested in finding a method which has not only a large absolute stability region but also good accuracy properties.

A non-linear system of algebraic equations has to be solved in the attempts to find a fourth-order six-stage Explicit Runge-Kutta Method. In our particular case this system contains **15 equations with 26 unknowns**. It should be mentioned that

(a) the first eight equations are the order conditions needed to achieve fourth order of accuracy (the first four of them being the same as the order conditions presented in the previous sub-section; these relationships are given here only for the sake of convenience),

(b) the next two equations will ensure good absolute stability properties

and

(c) the last five conditions are some relations between the coefficients of the Runge-Kutta method.

The **15** equations are listed below:

$$\mathbf{c_1 + c_2 + c_3 + c_4 = 1}\,, \tag{2.75}$$

$$\mathbf{c_2 a_2 + c_3 a_3 + c_4 a_4 = \frac{1}{2}}\,, \tag{2.76}$$

$$\mathbf{c_2 (a_2)^2 + c_3 (a_3)^2 + c_3 (a_3)^2 = \frac{1}{3}}\,, \tag{2.77}$$

$$\mathbf{c_3 b_{32} a_2 + c_4 (b_{42} a_2 + b_{43} a_3) = \frac{1}{6}}\,, \tag{2.78}$$

$$\mathbf{c_2 (a_2)^3 + c_3 (a_3)^3 + c_4 (a_4)^3 + c_5 (a_5)^3 + c_6 (a_6)^3 = \frac{1}{4}}\,, \tag{2.79}$$

$$(2.80)\quad \mathbf{c_3 b_{32}(a_2)^2 + c_4[b_{42}(a_2)^2 + b_{43}(a_3)^2] + c_5[b_{52}(a_2)^2 + b_{53}(a_3)^2 + b_{54}(a_4)^2] + c_6[b_{62}(a_2)^2 + b_{63}(a_3)^2 + b_{64}(a_4)^2 + b_{65}(a_5)^2] = \frac{1}{12}},$$

$$(2.81)\quad \mathbf{c_3 a_3 b_{32} a_2 + c_4 a_4(b_{42} a_2 + b_{43} a_3) + c_5 a_5(b_{52} a_2 + b_{53} a_3 + b_{54} a_4) + c_6 a_6(b_{62} a_2 + b_{63} a_3 + b_{64} a_4 + b_{65} a_5) = \frac{1}{8}},$$

$$(2.82)\quad \mathbf{c_4 b_{43} b_{32} a_2 + c_5[b_{53} b_{32} a_2 + b_{54}(b_{42} a_2 + b_{43} a_3)] + c_6[b_{63} b_{32} a_2 + b_{64}(b_{42} a_2 + b_{43} a_3) + b_{65}(b_{52} a_2 + b_{53} a_3 + b_{54} a_4)] = \frac{1}{24}}$$

$$(2.83)\quad \mathbf{c_6 b_{65} b_{54} b_{43} b_{32} a_2 = \frac{1}{720}\ \frac{1}{4.86}}$$

$$(2.84)\quad \mathbf{c_5 b_{54} b_{43} b_{32} a_2 + c_6\{b_{64} b_{43} b_{32} a_2 + b_{65}[b_{53} b_{32} a_2 + b_{54}(b_{42} a_2 + b_{43} a_3)]\} = \frac{1}{120}\ \frac{1}{1.42}}.$$

$$(2.85)\quad \mathbf{b_{21} = a_2}$$

$$(2.86)\quad \mathbf{b_{31} + b_{32} = a_3}$$

$$(2.87)\quad \mathbf{b_{41} + b_{42} + b_{43} = a_4}$$

$$(2.88)\quad \mathbf{b_{51} + b_{52} + b_{53} + b_{54} = a_5}$$

$$(2.89)\quad \mathbf{b_{61} + b_{62} + b_{63} + b_{64} + b_{65} = a_6}$$

The **26** unknowns in the non-linear system of algebraic equations described by the relationships (2.75)-(2.89) can be seen in the array representing the class of six stages Explicit Runge-Kutta Methods, which is given below:

a_2	b_{21}					
a_3	b_{31}	b_{32}				
a_4	b_{41}	b_{42}	b_{43}			
a_5	b_{51}	b_{52}	b_{53}	b_{54}		
a_6	b_{61}	b_{62}	b_{63}	b_{64}	b_{65}	
	c_1	c_2	c_3	c_4	c_5	c_6

We shall need, for several comparisons, the numerical results obtained with some good and accurate fifth-order six-stage Explicit Runge-Kutta methods. Nine additional relationships must be satisfied in order to achieve such a high accuracy, but the two conditions (2.83) and (2.84), which were imposed for improving the absolute stability properties are now not needed. The extra order conditions that are needed to achieve **fifth order of accuracy** are:

$$(2.90)\quad c_2(a_2)^4 + c_3(a_3)^4 + c_4(a_4)^4 + c_5(a_5)^4 + c_6(a_6)^4 = \frac{1}{5}\,,$$

$$(2.91)\quad c_3 b_{32}(a_2)^3 + c_4[b_{42}(a_2)^3 + b_{43}(a_3)^3] + c_5[b_{52}(a_2)^3 + b_{53}(a_3)^3 + b_{54}(a_4)^3] + c_6[b_{62}(a_2)^3 + b_{63}(a_3)^3 + b_{64}(a_4)^3 + b_{65}(a_5)^3] = \frac{1}{20}\,,$$

$$(2.92)\quad c_3 a_3 b_{32}(a_2)^2 + c_4 a_4[b_{42}(a_2)^2 + b_{43}(a_3)^2] + c_5 a_5[b_{52}(a_2)^2 + b_{53}(a_3)^2 + b_{54}(a_4)^2] + c_6 a_6[b_{62}(a_2)^2 + b_{63}(a_3)^2 + b_{64}(a_4)^2 + b_{65}(a_5)^2] = \frac{1}{15}\,,$$

$$(2.93)\quad c_3(b_{32}a_2)^2 + c_4(b_{42}a_2 + b_{43}a_3)^2 + c_5(b_{52}a_2 + b_{53}a_3 + b_{54}a_4)^2 + c_6(b_{62}a_2 + b_{63}a_3 + b_{64}a_4 + b_{65}a_5)^2 = \frac{1}{20},$$

$$(2.94)\quad c_3(a_3)^2 b_{32}a_2 + c_4(a_4)^2(b_{42}a_2 + b_{43}a_3) + c_5(a_5)^2(b_{52}a_2 + b_{53}a_3 + b_{54}a_4) + c_6(a_6)^2(b_{62}a_2 + b_{63}a_3 + b_{64}a_4 + b_{65}a_5) = \frac{1}{10}\,,$$

$$(2.95)\quad c_4 b_{43} b_{32}(a_2)^2 + c_5\{b_{53} b_{32}(a_2)^2 + b_{54}[b_{42}(a_2)^2 + b_{43}(a_3)^2]\} + c_6\{b_{63} b_{32}(a_2)^2 + b_{64}[b_{42}(a_2)^2 + b_{43}(a_3)^2] + b_{65}[b_{52}(a_2)^2 + b_{53}(a_3)^2 + b_{54}(a_4)^2]\} = \frac{1}{60}\,,$$

$$(2.96)\quad c_5 b_{54} b_{43} b_{32} a_2 + c_6\{b_{64} b_{43} b_{32} a_2 + b_{65}[b_{53} b_{32} a_2 + b_{54}(b_{42} a_2 + b_{43} a_3)]\} = \frac{1}{120},$$

$$(2.97)\quad c_4 a_4 b_{43} b_{32} a_2 + c_5 a_5 [b_{53} b_{32} a_2 + b_{54}(b_{42} a_2 + b_{43} a_3)] + c_6 a_6 [b_{63} b_{32} a_2 + b_{64}(b_{42} a_2 + b_{43} a_3) + b_{65}(b_{52} a_2 + b_{53} a_3 + b_{54} a_4)] = \frac{1}{30},$$

$$(2.98)\quad c_4 b_{43} a_3 b_{32} a_2 + c_5 [b_{53} a_3 b_{32} a_2 + b_{54} a_4 (b_{42} a_2 + b_{43} a_3)] + c_6 [b_{63} a_3 b_{32} a_2 + b_{64} a_4 (b_{42} a_2 + b_{43} a_3) + b_{65} a_5 (b_{52} a_2 + b_{53} a_3 + b_{54} a_4)] = \frac{1}{40},$$

The coefficients of a fifth-order six-stage explicit Runge-Kutta method proposed by John Butcher (**Butcher, 2003**) are shown in the array given below:

$a_2 = \frac{2}{5}$	$b_{21} = \frac{2}{5}$					
$a_3 = \frac{1}{4}$	$b_{31} = \frac{11}{64}$	$b_{32} = \frac{11}{64}$				
$a_4 = \frac{1}{2}$	$b_{41} = 0$	$b_{42} = 0$	$b_{43} = \frac{1}{2}$			
$a_5 = \frac{3}{4}$	$b_{51} = \frac{3}{64}$	$b_{52} = -\frac{15}{64}$	$b_{53} = \frac{3}{8}$	$b_{54} = \frac{9}{16}$		
$a_6 = 1$	$b_{61} = 0$	$b_{62} = \frac{5}{7}$	$b_{63} = \frac{6}{7}$	$b_{64} = -\frac{12}{7}$	$b_{65} = \frac{8}{7}$	
	$c_1 = \frac{7}{90}$	$c_2 = 0$	$c_3 = \frac{32}{90}$	$c_4 = \frac{12}{90}$	$c_5 = \frac{32}{90}$	$c_6 = \frac{7}{90}$

It can easily be verified that all conditions (2.75)-(2.98), except the relationships (2.83)-(2.84), by which the stability properties are improved, are satisfied by the coefficients of the numerical method presented by the above array. **This is, of course, an indirect indication that these conditions were correctly derived.**

Let us consider now the derivation of a particular fourth-order six-stage Explicit Runge-Kutta Method. Assume that **the eleven coefficients** that are listed below

$$(2.99)\quad c_5,\ c_6,\ a_3,\ a_6,\ b_{32},\ b_{41},\ b_{43},\ b_{52},\ b_{54},\ b_{61},\ b_{63}$$

are fixed and have the same values as those given in the above array. Then we have to solve the system of **15** equations with **15** unknowns, which is defined by (2.75)-(2.89). The well-known Newton iterative procedure was used in the numerical solution. The **15** components of the initial values of the components of the solution vector were taken from the Butcher's method and extended precision was used during the iterative process (also in this case quadruple precision, working with 32 digits, was selected). The fact that we are starting with the coefficients of the fifth-order six-stage explicit Runge-Kutta method is giving a reasonable chance to find a fourth-order six-stage Explicit Runge-Kutta which has good accuracy properties.

The numerical solution found at the end of the Newton iterative procedure is given below:

$$\mathbf{c_1 = 0.06636143820913713327361576677234} \quad (2.100)$$

$$\mathbf{c_2 = 0.33466439117348386167956841089170} \quad (2.101)$$

$$\mathbf{c_3 = 0.06029354106292902784346079863927} \quad (2.102)$$

$$\mathbf{c_4 = 0.10534729622111664387002169036336} \quad (2.103)$$

$$\mathbf{a_2 = 0.24412763924409282870819068414842} \quad (2.104)$$

$$\mathbf{a_4 = 0.58389416084413897975810996900256} \quad (2.105)$$

$$\mathbf{a_5 = 0.74232095083880033421170727685848} \quad (2.106)$$

$$\mathbf{b_{21} = 0.24412763924409282870819068414842} \quad (2.107)$$

$$\mathbf{b_{31} = 0.17187500000000000000000000000000} \quad (2.108)$$

$$\mathbf{b_{42} = 0.08389416084413897975810996900256} \quad (2.109)$$

$$\mathbf{b_{51} = -0.00395725816543771434700055768757} \quad (2.110)$$

$$\mathbf{b_{53} = 0.41815320900423804855870783454605} \quad (2.111)$$

$$(2.112) \qquad \mathbf{b_{62}} = \mathbf{0.567921736414093529460202151174 01}$$

$$(2.113) \qquad \mathbf{b_{64}} = \mathbf{-1.110041911712062532318479614250 22}$$

$$(2.114) \qquad \mathbf{b_{65}} = \mathbf{0.684977318155111860001134605933 36}$$

Numerical results obtained when the so derived fourth-order six-stage Explicit Runge-Kutta Method (**ERK64**) and its combination with the Richardson Extrapolation (**ERK64+RE**) are used in the solution of the first example from Section 2.5 are given in **Table 2.9**. The corresponding results, obtained by applying the classical **ERK44** method and the **ERK65B,** the fifth-order six-stage method proposed in Butcher's book (**Butcher, 2003**), are also presented in **Table 2.9**. Additionally, results obtained by using the fifth-order six-stage Explicit Runge-Kutta (**ERK65F**) Method proposed by E. Fehlberg (**Fehlberg, 1966**) are given in **Table 2.9**. It should be mentioned that it was established that also the coefficients of the Fehlberg's method are satisfying all the order conditions (2.75)-(2.98), except the relationships (2.83)-(2.84) by which the stability properties are improved, which verifies once again the correctness of their derivation.

Run	Stepsize	ERK44	ERK65B	ERK65F	ERK64	ERK64+RE
1	0.02048	N.S.	N.S.	N.S.	N.S.	9.00E-08
2	0.01024	N.S.	N.S.	N.S.	N.S.	1.93E-04
3	0.00512	N.S.	1.18E-09	N.S.	1.16E-07	8.82E-11
4	0.00256	2.46E-08	3.69E-11 (31.97)	5.51E-11	7.28E-09 (15.93)	2.76E-12 (31.96)
5	0.00128	1.54E-09 (15.97)	1.15E-12 (32.09)	1.72E-12 (32.03)	4.55E-10 (16.00)	8.62E-14 (32.02)
6	0.00064	9.62E-11 (16.00)	3.61E-14 (31.86)	5.39E-14 (31.91)	2.85E-11 (15.96)	2.69E-15 (32.04)
7	0.00032	6.01E-12 (16.01)	1.13E-15 (31.95)	1.68E-15 (32.08)	1.78E-12 (16.01)	8.42E-17 (31.95)
8	0.00016	3.76E-13 (15.98)	3.52E-17 (32.10)	5.26E-17 (31.94)	1.11E-13 (16.04)	2.63E-18 (32.01)
9	0.00008	2.35E-14 (16.00)	1.10E-18 (32.00)	1.64E-18 (32.07)	6.95E-15 (15.97)	8.22E-20 (32.00)
10	0.00004	1.47E-15 (15.99)	3.44E-20 (31.98)	5.14E-20 (31.91)	4.34E-16 (16.01)	2.57E-21 (31.98)
11	0.00002	9.18E-17 (16.01)	1.07E-21 (32.15)	1.61E-21 (31.93)	2.71E-17 (16.01)	8.03E-23 (32.00)
12	0.00001	5.74E-18 (15.99)	3.36E-23 (31.85)	5.02E-23 (32.07)	1.70E-18 (15.94)	2.51E-24 (31.99)

Table 2.9

Comparison of the first fourth-order six stages explicit Runge-Kutta (**ERK64**) method and its combination with the Richardson Extrapolation (**ERK64+RE**) with the classical fourth-order four stages explicit Runge-Kutta (**ERK44**) method and the fifth-order six stages (**ERK65B** and **ERK65F**) Runge-Kutta methods proposed respectively by Butcher in his book and by Fehlberg in 1968. "**N.S**" means that the method is not stable (the computations are declared as unstable and stopped when the norm of the calculated solution becomes greater than **1.0E+07**). The convergence rates are given in brackets in the table.

Similar conclusions, as those which were valid for the results presented in **Table 2.8**, can also be drawn for the new **ERK64** method and its combination (**ERK64+RE**) with the Richardson Extrapolation. These conclusions are listed below:

No.	Conclusions
1	The new numerical method, the fourth-order six-stage Explicit Runge-Kutta (**ERK64**) Method, is both more accurate and more stable than the classical fourth-order four stages Explicit Runge-Kutta (**ERK44**) Method (which indicates that the choice of the particular **ERK64** method with good accuracy properties was successful).
2	The fifth-order six-stage Explicit Runge-Kutta Method proposed by Butcher, **ERK65B**, is both more accurate and more stable than the fifth-order six-stage Explicit Runge-Kutta Method proposed by Fehlberg, **ERK65F.** This is the reason for using the former method as a starting point in the Newton iterative procedure. We are trying in this way to obtain a method which is in some sense closer to the better one of the two well-known and commonly used fifth-order six stage methods.
3	Both the fifth-order six-stages Explicit Runge-Kutta Method proposed by Butcher, **ERK65B**, and the fifth-order six-stage Explicit Runge-Kutta ethod proposed by Fehlberg, **ERK65F** are more accurate than the new **ERK64** method (which is quite natural, because their order of accuracy is higher), but the new method has better stability properties than the **ERK65F** method and, therefore, behaves in a reasonably stable way in some cases where this method fails.
4	The combination of the new method with the Richardson Extrapolation (**ERK64+RE**) is both more accurate and more stable than the two classical methods (**ERK65B** and **ERK65F**). Note that **ERK43+RE** method is stable for **all** 12 runs.
5	It is not very clear why the numerical error for $\mathbf{h} = \mathbf{0.01024}$ is greater than that for $\mathbf{h} = \mathbf{0.02048}$ when the **ERK64+RE** is used (the opposite should be true), but some conclusions can anyway be drawn by studying the plot presenting the absolute stability region of this method. The border of the absolute stability region around the point $-\mathbf{13.5}$ is rather close to the negative part of the real axis and this fact might have some influence on the results (perhaps due to the fact that two of the eigenvalues have imaginary parts). When the stepsize becomes bigger, the real part of the largest eigenvalue multiplied by $\mathbf{h}$ moves to the left, but there the border of the absolute stability region is not so close to the negative part of the real axis and the numerical results become again more stable.

2.9.4. Possibilities for further improvement of the results

It was mentioned several times that our objective was to derive two numerical methods, which have good stability properties and at the same time are very accurate. In order to achieve this we tried to derive methods, which are in some sense close to a method of a higher order. This strategy was followed in the derivation of both the **ERK43** method and the **ERK64** method.

For the **ERK43** method we used as starting point the classical **ERK44** method determined by the formulae (2.55)-(2.59) in Section 2.7. In order to satisfy the stability condition (2.67) we had only to modify two of the coefficients of the classical method; see (2.74).

The **ERK65B** method (which is clearly better than the **ERK65F** method) was applied in the derivation of an **ERK64** method. Eleven of the coefficients of the **ERK64** method are the same as those in the **ERK65B** method. Moreover, we started the Newton iterative procedure by using as an initial guess the remaining coefficients of the **ERK65B** method. The expectation is that the vector containing the coefficients of the so derived **ERK64** method will be in some sense close to the corresponding vector of the **ERK65B** method.

It is intuitively clear that if the derived numerical method is close in some sense to a method of higher order, then the leading terms of the local truncation error will be small (because for the method of higher order the corresponding terms are equal to zero, which is ensured by the order conditions). The statement that the leading terms of the local truncation error are small is, of course, based on heuristic assumptions. Nevertheless, the results, which are presented in **Table 2.8** and **Table 2.9,** indicate very clearly that the new numerical methods not only have enhanced stability properties, but are in addition very accurate.

The question is:

Is it possible to apply some more strict rules by which to derive some even more accurate methods?

Some ideas, which can be used in the derivation Explicit Runge-Kutta Methods with better accuracy properties, are sketched below.

Let us start with the **ERK64** method. It is reasonable to expect that the results for this methods could be improved if the following procedure is used. Consider the conditions (2.90)-(2.98) needed to achieve fifth order of accuracy. Move the constants from the right-hand-sides of these equalities to the left-hand-sides. Denote by $\mathbf{G_i}$, where $\mathbf{i = 1, 2, \ldots, 9}$, the absolute values of the terms in the left-hand sides that are obtained after these transformations. As an illustration of this process let us point out that

$$(\mathbf{2.115}) \quad \mathbf{G_1} = \left| \mathbf{c_2 (a_2)^4 + c_3 (a_3)^4 + c_4 (a_4)^4 + c_5 (a_5)^4 + c_6 (a_6)^4 - \frac{1}{5}} \right|$$

will be achieved from (2.90) by following the sketched above rules. It is clear how the remaining eight values of the quantities $\mathbf{G_i}$ can be obtained from (2.91)-(2.98).

Now the following constrained non-linear optimization problem can be defined. Find the minimum of the expression:

$$(\mathbf{2.116}) \quad \sum_{\mathbf{i=1}}^{\mathbf{9}} \mathbf{G_i}$$

under the assumption that the equalities (2.75)-(2.89) are also satisfied.

Thus, we have to solve a non-linear optimization problem with 15 constraints in order to obtain a method with (hopefully) better accuracy properties.

It is possible to generalize slightly this idea in the following way. Introduce non-negative weights $\mathbf{w_i}$, assume that the sum of the weights is equal to $\mathbf{9}$ and minimize the sum given below:

$$(\mathbf{2.117}) \quad \sum_{\mathbf{i=1}}^{\mathbf{9}} \mathbf{w_i G_i}$$

again under the assumption that the equalities (2.75)-(2.89) are also satisfied. It is obvious that if all weights are equal to $\mathbf{1}$, then (2.117) reduces to (2.116).

In our opinion the new **ERK43** method is very close to the classical **ERK44** method (only two of the coefficients of the classical methods have to be modified in order to satisfy the relationship arising from the requirement to achieve enhanced absolute stability) and it is hard to believe that some essential improvement can be achieved in this case. Nevertheless, one can try to derive better methods. This can be done in a quite similar way as the procedure used above. Consider the conditions (2.79)-(2.82) needed to obtain fourth-order accuracy. Move the constants in the right-hand-sides of these equalities to the left-hand-side. Denote by $\mathbf{F_i}$, where $\mathbf{i = 1, 2, 3, 4}$, the absolute values of the terms in the left-hand sides of the obtained after these transformations equalities. As an illustration of the outcome from this process let us point out that

$$(\mathbf{2.118}) \quad \mathbf{F_1} = \left| \mathbf{c_2 (a_2)^3 + c_3 (a_3)^3 + c_4 (a_4)^3 + c_5 (a_5)^3 + c_6 (a_6)^3 - \frac{1}{4}} \right|$$

will be achieved from (2.79) by following the rules that were sketched above. It is clear how the remaining three values of the quantities $\mathbf{F_i}$ can be obtained from (2.80)-(2.82). Now the following constrained optimization problem can be defined. Find the minimum of the expression:

$$(\mathbf{2.119}) \quad \sum_{\mathbf{i=1}}^{\mathbf{4}} \mathbf{F_i}$$

under the assumption that the equalities (2.63)-(2.70) are also satisfied.

It is again possible to generalize slightly this idea. Introduce non-negative weights $\mathbf{v_i}$, assume that the sum of the weights is now equal to $\mathbf{4}$ and minimize the sum given below:

$$(\mathbf{2.20}) \quad \sum_{\mathbf{i=1}}^{\mathbf{4}} \mathbf{v_i F_i}$$

under the assumption that the four equalities (2.63)-(2.70) are also satisfied. It is obvious that if all weights $\mathbf{v_i}$ are set equal to $\mathbf{1}$, then (2.20) reduces to (2.19).

It must be emphasized that the set of order conditions (2.75)-(2.82) + (290)-(298) is very general and can be used for developing many kinds of different Explicit Runge-Kutta Methods, whose order is less than or equal to five (in fact, classes of such methods). This set of relationships (actually some sub-sets of this set) has been used in this section to search for good **ERK43** and **ERK64** methods, but it can also be applied, for example, for designing good **ERK63** methods: the absolute stability properties of these methods will probably be considerably better that those of the two classes considered above, because the number of free parameters will be increased by one to become three ($\boldsymbol{\gamma_4^{(6,3)}}, \boldsymbol{\gamma_5^{(6,3)}}, \boldsymbol{\gamma_6^{(6,3)}}$), but the search for particular values of these constants which ensure greater absolute stability regions will be much more complicated.

Explicit Runge-Kutta Methods containing more than six stages can also be used. One can, for example, use relation (2.60) with $\mathbf{m = 11}$ and some $\mathbf{p \leq 8}$ (it can be proved that no EPRM of order greater than $\mathbf{8}$ can be constructed when the number of stages is $\mathbf{11}$; see p.182 in **Lambert, 1991**). The number of free parameters will be greater than or equal to three when such a choice is made and it will be possible to create accurate ERKMs with even larger stability regions. However, the procedure will become much more difficult. It will not be easy to derive the order conditions, but some packages for automatic differentiation and symbolic computations, such as **Maple** and **Mathematica**, can be applied to facilitate the derivation. Even after utilizing these packages, many other problems remain. These problems are related mainly

(a) to the formulation of large underdetermined non-linear systems of algebraic equations (or, even better, large constrained non-linear optimization problems)

and

(b) to finding solutions of these problems that will ensure good accuracy properties of the derived Explicit Runge-Kutta Methods.

It is nevertheless worthwhile to try to derive sufficiently accurate methods with better stability properties, because in this way it may become possible to avoid the application of implicit numerical methods for solving systems of ODEs (which leads to the necessity to handle very large non-linear algebraic systems) and, thus to reduce very considerably the computational work when some classes of large-scale scientific problems are to be treated.

It must also be emphasized that we are not interested so much in finding accurate Explicit Runge-Kutta Methods with good stability properties, but first and foremost in ERKMs, which applied together with the Richardson Extrapolation result in new numerical methods with even better stability properties. This is important, because it is well-known that the application of the Richardson Extrapolation may sometimes result in new numerical methods, which have worse stability properties than those of the

underlying method. The most terrible example is the application of the Richardson Extrapolation together with the well-known Trapezoidal Rule. While the Trapezoidal Rule has excellent stability properties (it is A-stable), its combination with the Richardson Extrapolation leads to an unstable computational process. Some other examples can be found in **Zlatev, Faragó and Havasi (2010)**. It must be strongly emphasized here that **all** Explicit Runge-Kutta Methods which were considered above, also the methods in the previous sections, were designed so that their combinations with the Richardson Extrapolation have bigger absolute stability regions than the underlying methods (see **Fig. 2.8** and **Fig. 2.9**). In this way, **larger time-stepsizes can be used when the Richardson Extrapolation is added not only because the resulting method is more accurate, but also because it is more stable.**

The conclusion made above is indeed very important: we tried until now to design accurate ERKMs with good stability properties and **after that** to show that their combinations with the Richardson Extrapolations have even larger regions of absolute stability (they will in addition be necessarily more accurate). The order should probably be reversed. Indeed, as we pointed out several times, our major objective is to use the Richardson Extrapolation in an attempt to improve the efficiency of the computational process. Therefore, it might be much more profitable to search **directly** for combinations of ERKMs with the Richardson Extrapolation, which have good stability and accuracy properties. This problem is, of course much, more complicated (first and foremost, because the degree of the stability polynomials of the combinations of ERKMs and the Richardson Extrapolation is twice greater than the degree of the underlying ERKMs), but the results of this much more difficult search might be significantly better. Therefore, it is worthwhile to try to resolve this very challenging problem.

2.10. Major concluding remarks related to Explicit Runge-Kutta Methods

Specific conclusions based on numerical results from three examples (introduced in Section 2.5) were drawn in the previous sections. Several additional general conclusions will be drawn below. These conclusions are based not only on numerical results, but also on the established, in Section 2.4 and Section 2.9, facts that the Richardson Extrapolation does lead to a considerable improvement of both the stability properties and the accuracy of the calculated approximations (in comparison with those of the underlying Explicit Runge-Kutta Methods when these are used directly). It was established in Section 2.4 that the stability regions were **always** increased when the Richardson Extrapolation is used and when the number of stages $\mathbf{m}$ is equal to the order of accuracy $\mathbf{p}$. However it was also shown, in the previous section, that even better results can be achieved for some other classes of Explicit Runge-Kutta Methods. We shall try now to summarize these results.

It is well known that the application of the Richardson Extrapolation leads always to an improvement of the accuracy of the underlying numerical method when the stability properties are not restricting the choice of the time-stepsize. This statement holds not only for the Explicit Runge-Kutta Methods, but for **any** numerical method for solving systems of ODEs (see Chapter 1). The remarkable thing for the class of Explicit Runge-Kutta Methods with $\mathbf{p} = \mathbf{m}$, $\mathbf{m} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}$ is, as mentioned above, that the application of the Richardson Extrapolation leads to new numerical methods with considerably **larger** absolute stability regions. In fact, the results shown in Section 2.4 (and more precisely, the results

presented in the plots drawn in Fig. 2.1 – Fig 2.4) could be considered as a graphical proof of the following theorem:

Theorem 2.1: Let us consider **an arbitrary** Explicit Runge-Kutta Method, for which the condition $\mathbf{p = m}$, $\mathbf{m = 1, 2, 3, 4}$ is satisfied. If $\mathbf{p = m = 1}$, then there exists only one such method (the Forward Euler Formula), while large classes of Explicit Runge-Kutta Methods exist for each $\mathbf{p = m}$, when $\mathbf{p}$ is greater than one. Combine the selected method with the Richardson Extrapolation. Then the obtained in this was new numerical method has **always** a larger absolute stability region than that of the underlying Explicit Runge-Kutta Method.

■

In the previous section we have demonstrated that two other results, which are in some sense even stronger, hold:

Theorem 2.2: A large class $\mathcal{F}_{2.4}^{4,3}$ of Explicit Runge-Kutta Methods depending on the three parameters $\mathbf{p = 3}$, $\mathbf{m = 4}$ and $\gamma_4^{(4,3)} = \mathbf{2.4}$ can be derived. Two statements are true for this class:

(a) every representative of class $\mathcal{F}_{2.4}^{4,3}$ (i.e. any Explicit Runge-Kutta Method from class $\mathcal{F}_{2.4}^{4,3}$) has the **same** absolute stability region that is optimal in some sense (which implies, of course, that all combinations of methods from the class $\mathcal{F}_{2.4}^{4,3}$ and the Richardson Extrapolation have the same absolute stability region)

and

(b) the combination of any representative of class $\mathcal{F}_{2.4}^{4,3}$ with the Richardson Extrapolation has **larger** absolute stability region than the absolute stability region of the underlying numerical method.

■

Theorem 2.3: A large class $\mathcal{F}_{1.42,4.86}^{6,4}$ of Explicit Runge-Kutta Methods depending on the three parameters $\mathbf{p = 4}$, $\mathbf{m = 6}$, $\gamma_5^{(6,4)} = \mathbf{1.42}$ and $\gamma_6^{(6,4)} = \mathbf{4.86}$ can be derived. Two statements are true for this class:

(a) every representative of class $\mathcal{F}_{1.42,4.86}^{6,4}$ (i.e. any Explicit Runge-Kutta Method from class $\mathcal{F}_{1.42,4.86}^{6,4}$) has the **same** absolute stability region that is optimal in some sense (which implies, of course, that all combinations of the methods from the class $\mathcal{F}_{1.42,4.86}^{6,4}$ and the Richardson Extrapolation have the same absolute stability region)

and

(b) the combination of any representative of class $\mathcal{F}^{6,4}_{1.42,4.86}$ with the Richardson Extrapolation has a **larger** absolute stability region than the absolute stability region of the underlying numerical method.

■

The validity of the statements in Theorem 2.2 and Theorem 2.3 were verified in Section 2.9.

During the search for ERKMs with large absolute stability regions, we investigated many methods. For all these ERKMs, the new methods obtained when the Richardson Extrapolation is additionally used had bigger absolute stability regions than the underlying method. This fact justifies the formulation of the following conjecture:

Conjecture: **The combination of the Richardson Extrapolation with any Explicit Runge-Kutta Method leads to a new numerical method, which has a larger absolute stability region.**

Finally, at the end of this chapter it should also be emphasized that non-stiff and moderately stiff systems of ODEs, for which the methods studied in this chapter can be very useful, appear after some kind of discretization and/or splitting of mathematical models appearing in different areas of science and engineering.

As an example, large-scale **air pollution models** should be mentioned; see **Alexandrov, Sameh, Siddique and Zlatev (1997)**, **Alexandrov, Owczarz, Thomsen and Zlatev (2004)**, **Zlatev (1995)** and **Zlatev and Dimov (2006)**. Large-scale air pollution models can be used in many important environmental studies. The most important of the different studies is perhaps the investigation of the impact of climate changes on the high air pollution levels. Such investigations were carried out by using the Unified Danish Eulerian Model (UNI-DEM) in **Zlatev (2010)**, **Zlatev and Christensen (1989)**, **Zlatev, Georgiev and Dimov (2013b)** and **Zlatev, Havasi and Faragó (2011)**. The advection terms (the terms containing first-order spatial derivatives in the systems of partial differential equations by which large-scale air pollution models are described) can be treated with explicit methods for solving systems of ODEs after applying some splitting procedure and discretization; see again **Alexandrov, Sameh, Siddique and Zlatev (1997)**, **Alexandrov, Owczarz, Thomsen and Zlatev (2004)**, **Zlatev (1995)** and **Zlatev and Dimov (2006)**.

An attempt to implement combinations of the Explicit Runge-Kutta Methods discussed in this chapter with the Richardson Extrapolation in the non-stiff sub-models of UNI-DEM will be carried out in the near future.

2.11. Topics for further research

The following topics might lead to some very interesting and useful results:

(A) Explicit Runge-Kutta Methods of order up to four were studied in this chapter. It will be worthwhile to try to extend the results for methods of order five, six and seven and to find some classes of methods with enlarged absolute stability regions. There will be some free parameters when such methods are considered. Will it be possible to find values of these parameters, which are in some sense optimal?

(B) It will not be a very easy task to obtaine some particular methods within the classes of methods mentioned in **(A)**. The determination of these methods will lead to the solution of non-linear algebraic equations in which the number of equations is smaller than the number of unknowns. Will it be possible to find particular methods which have optimal properties (or at least some good properties)?

(C) It is commonly accepted that the ERKM's with many stages are very expensive. However, if the numerical problems solved are moderately stiff, then such methods can be useful if they have good stability properties (they will be cheaper than numerical methods for stiff systems of ODE's, because there is no need to solve large systems of linear algebraic equations during the Newton iterative procedure; see Chapter 4). Therefore, it is worthwhile to construct ERKM's with many stages and good stability properties.

Chapter 3

Linear multistep and predictor-corrector methods

The linear multistep methods are very popular numerical tools, which are often used in the numerical solution of systems of ODEs, especially in the case where the solved problems are non-stiff. However, some difficulties appear when one attempts to apply the Richardson Extrapolation in conjunction with these numerical methods. These difficulties will be shortly discussed in this chapter, but the major objective will be the presentation of another useful computational approach, the development and the usage of easily applicable, efficient and reliable **predictor-corrector methods with improved stability properties**. It will be demonstrated that these devices can successfully play the same role as the role of the Richardson Extrapolation when it is used in the calculation of approximate solutions of the non-stiff systems of ODEs by Explicit Runge-Kutta Methods. By applying predictor-corrector schemes, one will be able both to improve the accuracy of the calculated results and to control in a sufficiently robust way the time-stepsize selection during the whole integration process. This means that the application of these alternative computational schemes, of the predictor-corrector methods, will have precisely the same computational effect as the effect achieved when the Richardson Extrapolation is applied in the numerical solution of systems of ODEs together with the Explicit Runge-Kutta Methods.

The class of the linear multistep methods will be introduced in **Section 3.1**. Some basic information about this large class of numerical algorithms, including some discussion of the important topics of **consistency, zero-stability and convergence**, will also be presented there. The advantages and the drawbacks of the linear multistep methods will be outlined. The most frequently used methods from this class will be introduced at the end of the first section.

Variation of the time-stepsize is not always improving the efficiency of the computational process, but may sometimes lead to a very substantial reduction of the computing time. If the variation of the time-stepsize will result in an improvement of the efficiency of the computational process and, thus, if it is desirable to introduce such a device in connection with the selected linear multistep methods, then at least one extra requirement appears. This additional requirement is the necessity to preserve the zero-stability of the selected linear multistep methods during the computations. The loss of zero-stability when the time-stepsize is varied **may** cause serious computational problems. This important topic will be treated in **Section 3.2**.

The absolute stability properties of the linear multistep methods also deserve some special treatment. The definition of absolute stability will be given in **Section 3.3** and the problems related to this concept will be shortly discussed there.

The difficulties connected to the application of the Richardson Extrapolation together with the linear multistep methods and the necessity to apply some alternative approach in the computer treatment of systems of ODEs will be shortly discussed in **Section 3.4**.

Predictor-corrector methods will be presented in **Section 3.5**. After a brief discussion of these schemes, several special predictor-corrector methods with improved absolute stability properties will also be presented. It will be emphasized that the predictor-correctors schemes are **explicit** numerical algorithms.

One of the most important and most useful properties of the predictor-corrector schemes is the fact that an estimation of the local truncation error can easily be calculated when these computational tools are used. An efficient device for estimating the local truncation error will be introduced and discussed in **Section 3.6**. Other ways to control the accuracy of the computed approximations and to select an optimal step-size, or at least a good one, for the next time-step will also be presented there.

The absolute stability properties of the predictor-corrector methods will be studied in **Section 3.7**. The necessity of improving the absolute stability properties of some commonly used classes of predictor-corrector schemes will be justified in this section and some predictor-corrector schemes with improved absolute stability properties will be presented.

Predictor-corrector schemes with several **different** correctors can sometimes be successfully used in the efforts to develop algorithms with improved absolute stability properties for some special situations, which appear when large-scale scientific and engineering models are to be handled on computers. This topic will be discussed in **Section 3.8**.

A stronger stability requirement, the requirement to achieve A-stability, will be briefly discussed in **Section 3.9**. In fact, only implicit numerical methods can be A-stable and, the predictor-corrector schemes are not A-stable, because as mentioned above, these schemes are explicit. The A-stability concept is applicable for a few **implicit** linear **k**-step methods (in fact, only linear **k**-step methods with $\mathbf{k} \leq \mathbf{2}$ can be A-stable). Weaker (but very useful for some applications) stability concepts are applicable to some numerical methods belonging to the class of **Backward Differentiation Formulae**. Some results will be presented in Section 3.9, but these concepts, especially the A-stability, will be discussed in detail in the next chapter, in Chapter 4.

For some readers, who wish to apply linear multistep methods, it will be useful to have access to a list of the coefficients of the most popular linear multistep methods. Such a list is given in **Section 3.10**. It is not easy to find such a list in the commonly used text-books, where as a rule only methods of lower orders are given.

Some general conclusions related to the linear multistep methods are listed in the last section, **Section 3.11**, of the third chapter.

Some proposals for a future research in this field are given in **Section 3.12**.

3.1. Linear multistep methods for solving systems of ODEs

Consider again the system of ODEs, which was defined by equalities (1.1) and (1.2) in Chapter 1. A general linear **k**-step method, where $\mathbf{k} \geq \mathbf{1}$ is a given integer, can be defined by the following formula:

$$(3.1) \qquad y_n + \sum_{i=1}^{k} \alpha_i \, y_{n-i} = h \sum_{i=0}^{k} \beta_i \, f_{n-i} \, , \qquad |\alpha_k| + |\beta_k| > 0 \, , \qquad n = k, \; k+1, \ldots , \; N,$$

where α_i and β_i are real constants and N is, as in Chapter 1, the number of grid-points in (1.6). Furthermore, for every value of the integer $n \geq 0$ the abbreviation f_n is introduced by the following formula:

$$(3.2) \qquad f_n = f(t_n \, , y_n), \qquad n = 0 \, , \; 1, \; 2 \, , \; \ldots \; , N \; .$$

If $\beta_0 = 0$, then the computational formula defined by (3.1) is an explicit numerical method, because the right-hand-side does not depend on the unknown vector y_n, while this formula is representing an implicit numerical method for $\beta_0 \neq 0$.

If $k = 1$, then the calculations with formula (3.1) can be started directly. It will be sufficient in this case to use the initial value $y_0 = y(a) = \eta$ from (1.2) at the first time-step and after that to carry on successively, step by step, the further computations. In principle, some approximation $y_0 \approx \eta$ can also be applied during the first time-step. It must be mentioned here that **one-step formulae** are actually obtained when the choice $k = 1$ is made in (3.1) and that some of these formulae, as the Forward Euler Method, were studied in the previous chapter.

If $k > 1$, then it is not possible to start directly the computations with formula (3.1). In the beginning of the computations it will be necessary to calculate sufficiently accurate approximations of the first $k - 1$ values of the exact solution at the equidistant grid defined by (1.6) by using some other numerical methods of an appropriate order (as, for example, with some Runge-Kutta methods). More precisely, it will indeed be necessary to calculate **sufficiently accurate** approximations of the following vectors $y_1 \approx y(t_1)$, $y_2 \approx y(t_2)$, ... , $y_{k-1} \approx y(t_{k-1})$. These approximations are often called **starting values**. This can be done, as mentioned above, by using some one-step numerical method of an appropriate order of accuracy in the beginning of the computational process (the order of accuracy of the auxiliary numerical method used to obtain the starting values must at least be equal to the order of accuracy of the selected linear multistep method that will be used in the further calculations). The need to prepare sufficiently accurate starting values is one of the serious drawbacks of the linear multistep methods (see also §3.1.4). More details about the calculation of the starting values, when linear multistep methods with $k > 1$ are to be used in the solution of systems of ODEs, can be found, for example, in **Henrici (1968)** and **Lambert (1991)**.

Two polynomials, which are often called **characteristic polynomials** (see **Lambert, 1991**), are usually associated with the coefficients of the linear multistep method defined by (3.1):

$$(3.3) \qquad \rho(z) = 1 + \sum_{i=1}^{k} \alpha_i \, z^i \, , \qquad \sigma(z) = \sum_{i=0}^{k} \beta_i \, z^i \, , \qquad z \in \mathbb{C} \, .$$

Many important properties of the linear multistep methods, such as order conditions, attainable order of accuracy, consistency, zero-stability and convergence, can efficiently be studied and will be studied in the remaining part of this chapter by using these polynomials.

3.1.1. Order conditions

If the numerical method defined by (3.1) is of order of accuracy $\mathbf{p \geq 1}$, then the coefficients $\boldsymbol{\alpha_i}$ and $\boldsymbol{\beta_i}$ must satisfy the following relationships (see **Lambert, 1991**):

$$\mathbf{(3.4)} \qquad \mathbf{1 + \sum_{i=1}^{k} \alpha_i = 0} \ ,$$

$$\mathbf{(3.5)} \qquad \mathbf{\sum_{i=1}^{k} i\alpha_i = \sum_{i=0}^{k} \beta_i} \ ,$$

$$\mathbf{(3.6)} \qquad \mathbf{\frac{1}{q!} \sum_{i=1}^{k} i^q \alpha_i = \frac{1}{(q-1)!} \sum_{i=0}^{k} i^{q-1} \beta_i} \ , \qquad \mathbf{q = 2, 3, \ldots, p} \ .$$

Note that, by applying the characteristic polynomials given in (3.3), the equalities (3.4) and (3.5) can be rewritten as

$$\mathbf{(3.7)} \qquad \boldsymbol{\rho(1) = 0}$$

and

$$\mathbf{(3.8)} \qquad \mathbf{\frac{d\rho(1)}{dz} = \sigma(1)}$$

respectively. Note too that the linear multistep method is of order one when the relationships (3.7) and (3.8) or, which is the same, when (3.4) and (3.5), are satisfied.

More details about the order conditions and their application in the efforts to design efficient linear multistep methods and different predictor-corrector methods can be found in **Butcher (2003)**, **Hairer,**

Nørsett and Wanner (1987), **Hundsdorfer and Verwer (2003)**, **Lambert (1991)** and **Shampine (1994)**.

3.1.2. Basic definitions

The following definitions are related to three fundamental properties of the linear multistep methods: to the concepts of **consistency, zero-stability and convergence**. These concepts are important not only for the linear multistep methods, but also for many other numerical methods for solving systems of ODEs (as, for example, for the Explicit Runge-Kutta Methods that were presented and discussed in the previous chapter).

Definition 3.1: The linear multistep method (3.1) is said to be **consistent** if (3.3) and (3.4) hold or, in other words, if (3.7) and (3.8) hold and, thus, if the method (3.1) is at least of order one.

■

Definition 3.2: The linear multistep method (3.1) is said to be **zero-stable** if no root of the first characteristic polynomial $\boldsymbol{\rho(z)}$ is greater than one in absolute value and if every root that is equal to one (in absolute value) is a simple root (i.e. there are no multiple roots equal to one in absolute value).

■

Definition 3.3: The linear multistep method introduced by (3.1) is said to be **convergent** if for all initial value problems for systems of ODEs defined by (1.1) and (1.2), for which the exact solution $\mathbf{y(t)}$ exists and is unique on the interval $[\mathbf{a}, \mathbf{b}]$ (which means that the conditions of Theorem 1.1 in Chapter 1 are satisfied), the relationship

$$(\mathbf{3.9}) \qquad \lim_{\substack{\mathbf{h\to 0} \\ \wedge\ \mathbf{nh=t_n-a}}} \mathbf{y_n = y(t_n)}$$

holds for all $\mathbf{t_n \in (a, b]}$ and for numerical solutions $\{\mathbf{y_n}\}$, $\mathbf{n = k}$, $\mathbf{k+1}$, ... , $\mathbf{N}$ of the difference equation (3.1) satisfying starting conditions $\mathbf{y_i = \eta_i(h)}$, $\mathbf{i = 1}$, $\mathbf{2}$, ... , $\mathbf{k-1}$, for which

$$(\mathbf{3.10}) \qquad \lim_{\mathbf{h\to 0}} \boldsymbol{\eta_i(h) = \eta}\ .$$

■

Now the following theorem holds (see, for example, **Lambert, 1991**).

Theorem 3.1: The linear multistep method defined by (3.1) is convergent if and only if it is both consistent and zero-stable.

■

The statement of Theorem 3.1 is often abbreviated as

consistency + zero-stability ⇔ convergence

This means that both consistency and zero-stability are needed in the efforts to design convergent linear multistep methods. Note too that in the definitions related to these important concepts it was implicitly assumed that a **constant** time-stepsize is used on the grid (1.6) defined in Chapter 1. The variation of the time-stepsize causes some problems when the linear multistep methods defined by (3.1) are used; this is a rather serious drawback of these methods.

It follows from Theorem 3.1 that the concept of zero-stability is very important, because combined with consistency it guarantees convergence. It turns out, however, that problems with the preservation of the zero-stability of the linear multistep methods may arise when variations of the time-stepsize are allowed. Some classes of linear multistep methods for which the zero-stability is preserved also in the case when the time-stepsize is varied will be introduced in Section 3.2.

3.1.3. Attainable order of linear multistep methods

It is natural to ask the following question:

What is the maximal order of a convergent linear k-step method?

The order conditions, which are given by (3.4) – (3.6), depend on the coefficients α_i and β_i of the linear multistep method. It is immediately seen that the number of these coefficients is $2k$ when the

method (3.1) is explicit (i.e. when $\boldsymbol{\beta_0 = 0}$) and $\mathbf{2k+1}$ when the method (3.1) is implicit (i.e. when $\boldsymbol{\beta_0 \neq 0}$). This means that in principle either $\mathbf{2k}$ or $\mathbf{2k+1}$ order conditions of type (3.4) – (3.6) arise and are to be satisfied for the explicit and implicit linear **k**-step methods respectively. It is furthermore clear that these relationships (the order conditions, which have to be satisfied) form a system of **linear algebraic equations**. Therefore, one should expect that linear **k**-step methods of orders $\mathbf{p = 2k}$ and $\mathbf{p = 2k + 1}$, which are both reliable and efficient, can be constructed when explicit and implicit linear multistep methods are selected. Unfortunately, such a conclusion is not true when the requirement for constructing **convergent** numerical methods is additionally imposed. This requirement, the requirement to ensure convergence, must necessarily be imposed and it leads to the famous **first Dahlquist barrier**. In fact, the restriction of the attainable order for convergent linear **k**-step methods that is caused by the first Dahlquist barrier is a consequence of the following theorem; see p. 55 in **Lambert (1991)**:

Theorem 3.2: No zero-stable **k**-step linear method can have order of accuracy exceeding $\mathbf{k+1}$ when **k** is odd and $\mathbf{k+2}$ when **k** is even.

■

Since a convergent linear **k**-step method must be zero-stable (according to Theorem 3.1) and a method of order greater than or equal to one is consistent, it is clear that "zero-stable" can be replaced with "convergent" in the statement of Theorem 3.2.

A statement similar to the assertion of Theorem 3.2 has been proved by G. Dahlquist in 1956 (see **Dahlquist, 1956, 1959**).

3.1.4. Drawbacks and advantages of the linear multistep methods

Some of the major drawbacks of the linear multistep methods were already mentioned in the previous sub-sections. We shall list the major drawbacks below.

(a) Need to calculate starting values. If $\mathbf{k > 1}$, then it is necessary to calculate $\mathbf{k - 1}$ starting values. Several different approaches can be used to resolve this problem. It is simplest, in principle at least, to apply some one-step method in the calculation of the needed starting values. However, one must remember that it is necessary to calculate **sufficiently accurate** starting values and this requirement can cause additional difficulties. This means that the order of accuracy of the one-step method selected for the computation of the starting values should be at least equal to the order of accuracy of the linear multistep method, which will be used in the remaining part of the calculations.

(b) Difficulties with the variations of the time-stepsize. The use of variable time-stepsizes might be very useful in some situations. This fact was explained in Chapter 1. The variation of the time-stepsize will certainly be very efficient when some

components of the solution vector are quickly varying in a short part of the time-interval $[\mathbf{a}, \mathbf{b}]$, while all its components are slowly varying in the remaining part of this interval. In such a case, it will be efficient to use small time-steps as well as more accurate formulae when some components of the solution vector have steep gradients and large time-stepsizes when all its components are smoothly and slowly varying. The straight-forward development and implementation of a suitable technique for an automatic variation of the time-stepsize and/or of the integration formula during the solution process results in a method with variable coefficients depending on the time-stepsizes that were used before the current time-step (more details can be found, for example, in **Zlatev, 1978, 1981b, 1983, 1984**). The calculation of the coefficients of the linear multistep formula can cause some problems when variable time-stepsizes are used and especially if in addition a variation of the formulae used is also allowed. This important topic will be further discussed in Section 3.2.

(c) The absolute stability regions of the linear multistep methods are small in comparison with those of the Explicit Runge-Kutta Methods. Moreover, the absolute stability regions are in general becoming smaller when the order of accuracy is increased (while the opposite is true for the Explicit Runge-Kutta Methods). The decrease of the size of the absolute stability regions is especially true for the explicit linear multistep methods. The fact that there are problems (or at least that problems may appear) in connection with the stability of the computational process is another serious drawback of the linear multistep methods. This is especially true in the case when mildly-stiff systems of ODEs are to be handled. The problems related to the absolute stability regions of the linear multistep methods will be treated in some more detail in Section 3.4 for linear multistep methods as well as in Section 3.7 and Section 3.8 for some special classes of predictor-corrector schemes.

(d) Only very few of the linear multistep methods are A-stable. A-stability is important when **stiff** systems of ODEs are to be treated. A-stable linear multistep method can be developed only for $\mathbf{k} \leq \mathbf{2}$. The A-stability and some related stability concepts will be described in Section 3.9. It should be mentioned here that the Backward Differentiation Formulae, see the end of the next sub-section, §3.1.5, have reasonably good stability properties and are very useful in the treatment of some special systems of ODEs.

The linear multistep methods have also some very important advantages:

(A) The formulae, by which these methods are defined, are very simple and as a rule it is easy to implement them in different parts of large-scale scientific and engineering models.

(B) The leading terms of the truncation error can be evaluated in a relatively straight-forward and very efficient way.

(C) These methods can be implemented in a very straight-forward manner as predictor-corrector schemes, which are very efficient when non-stiff or even mildly-stiff systems of ODEs are to be solved numerically.

The predictor-corrector schemes will be discussed in the next sections.

3.1.5. Frequently used linear multistep methods

The most frequently used linear multistep methods are the **Adams formulae**, which are defined in the following way:

$$(3.11) \qquad y_n = y_{n-1} + h \sum_{i=0}^{k} \beta_i f_{n-i}, \qquad |\beta_k| > 0, \qquad n = k, \; k+1, \ldots, \; N.$$

This means that

$$(3.12) \qquad \alpha_1 = -1, \qquad \alpha_i = 0, \qquad i = 2, \; 3, \; \ldots, \; k,$$

when the Adams formulae are used.

If $\beta_0 = 0$, i.e. if the linear multistep methods defined by using (3.11) and (3.12) are explicit, then they are called **Adams-Bashforth Formulae** (**Adams, 1883**, **Bashforth, 1883**).

If $\beta_0 \neq 0$, i.e. if the methods defined by applying (3.11) and (3.12) are implicit, then they are called **Adams-Moulton Formulae** (**Adams, 1883**, **Moulton 1926**).

Some people claim that both the Adams-Bashforth Formulae and the Adams-Moulton Formulae were invented by J. C. Adams in 1855.

The linear multistep methods from two other classes, the Nyström methods and the generalized Milne-Simpson methods, are not so popular as the Adams methods. However, some interesting properties of these methods when they are combined with the Adams methods can be derived in the important case when variation of both the time-stepsize and the formulae are allowed.

The Nyström and the generalized Milne-Simpson methods are defined in the following way:

$$(3.13) \qquad y_n = y_{n-2} + h \sum_{i=0}^{k} \beta_i f_{n-i}, \qquad |\beta_k| > 0, \qquad n = k, \; k+1, \ldots, \; N.$$

This means that

$$(3.14) \quad \alpha_1 = 0\,, \quad \alpha_2 = -1\,, \quad \alpha_i = 0\,, \qquad i = 3,\ 4\,,\ \ldots\,,\ k,$$

when the Nyström methods and/or the generalized Milne-Simpson methods are used.

If $\beta_0 = 0$, i.e. if the methods defined by using (3.13) and (3.14) are explicit, then they are called **Nyström methods** (**Nyström, 1925**).

If $\beta_0 \neq 0$, i.e. if the methods defined by using (3.13) and (3,14) are implicit, then they are called **generalized Milne-Simpson methods** (**Milne, 1926, 1953**).

If the problem solved, i.e. the system of ODEs defined by (1.1) and (1.2), is stiff, then some methods from the class of the **Backward Differentiation Formulae** can often be successfully used. The Backward Differentiation Formulae form a sub-class of the class of the linear multistep methods, which is defined by:

$$(3.17) \quad y_n + \sum_{i=1}^{k} \alpha_i\, y_{n-i} = h\,\beta_0 f_n\,, \qquad |\alpha_k| > 0\,, \qquad |\beta_0| > 0\,, \qquad n = k,\ k+1, \ldots,\ N\,.$$

The methods of this sub-class have relatively good stability properties (at least in the case when the order of accuracy is not greater than six), but they are implicit and, thus, in general large non-linear systems of algebraic equations are to be solved at every time-step when these methods are used.

The well-known and very popular first-order Backward Differentiation Formula, which is called also the Implicit Euler Method, can be obtained from (3.17) by setting there $k = 1$. This method is also an implicit Runge-Kutta method of order one. It will be further discussed in Chapter 4.

More details about the Backward Differentiation Formulae can be found, for example, in **Gear (1971)**, **Hairer and Wanner (1980)**, **Hundsdorfer and Verwer (2003)** and **Lambert (1991)**.

3.2. Variation of the time-stepsize for linear multistep methods

It was pointed out in the previous section (and also in the previous chapters) that it is worthwhile in some situations to be able to vary the time-stepsize during the computational process. Indeed, it is intuitively clear that if some of components of the exact solution vector $y(t)$ are rapidly varying in some parts of the integration interval $[a, b]$, while all components of $y(t)$ are slowly varying in the remaining parts of the integration interval $[a, b]$, then it will be more profitable to use small time-stepsizes in the parts where there are quickly varying components and large time-stepsizes in the remaining part of the time-interval. Furthermore, it will be efficient to use more accurate (and more expensive) formulae in the first case and less accurate, but considerably cheaper, formulae in the second case. These considerations lead to the idea of developing **linear multistep variable stepsize variable**

formula methods (LM VSVFMs). While the potential advantage of these methods is obvious, it is clear that two major difficulties arise and must be resolved during the development of such methods:

(a) the coefficients of the linear multistep methods will not remain constant anymore (they will be dependent on several of the last used time-stepsizes when this approach is applied)

and

(b) the zero-stability properties of the linear multistep method might be affected when it is allowed to vary the time-stepsize and also the formulae.

It will be shown, in the remaining part of this section, how the above two difficulties can be overcome.

3.2.1. Calculation of the coefficients of an LM VSVFM

Consider a set $\mathcal{F} = \{\mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_m}\}$ of $\mathbf{m}$ linear multistep methods with constant coefficients. Formula $\mathbf{F_j}$ of this set can be represented by the following expression, where index $\mathbf{j}$ is expressing the fact that precisely the $\mathbf{j^{th}}$ formula from set $\mathcal{F}$ is used:

$$\mathbf{y_n} + \sum_{i=1}^{k_j} \alpha_{ji}\, \mathbf{y_{n-i}} = \mathbf{h} \sum_{i=0}^{k_j} \beta_{ji}\, \mathbf{f_{n-i}}, \qquad \mathbf{j = 1, 2, \ldots, m}, \qquad \mathbf{n > k_j}. \tag{3.18}$$

Assume that methods of type (3.18) from the set $\mathcal{F}$ are to be used in the development of a variable stepsize variable formula procedure and that precisely the formula $\mathbf{F_j}$ is used at time-step $\mathbf{n}$. Denote by $\mathbf{F_{start}} \in \mathcal{F}$ the first linear multistep formula, which has been used after the calculation of the starting values, and consider the time-stepsize

$$\mathbf{h_n = x_n - x_{n-1}}, \qquad \mathbf{n = k_{start}, k_{start} + 1, \ldots, N}, \tag{3.19}$$

that is to be used at time-step $\mathbf{n}$, with $\mathbf{n \geq k_{start}}$. Set $\mathbf{h_0 = h_1}$, assume that some formula $\mathbf{F_j} \in \mathcal{F}$ is to be applied and consider the vector:

$$\bar{\mathbf{h}}_{\mathbf{nj}} = \left\{ \frac{\mathbf{h_{n-1}}}{\mathbf{h_n}}, \frac{\mathbf{h_{n-2}}}{\mathbf{h_n}}, \ldots, \frac{\mathbf{h_{n-k_j+1}}}{\mathbf{h_n}} \right\}, \qquad \mathbf{n > k_j}. \tag{3.20}$$

Under these assumptions, formula $\mathbf{F_j}$, the coefficients of which are not constants but depend on the time-stepsizes used during the last $\mathbf{k_j}$ time-steps, can be written in the following form:

$$(\mathbf{3.21}) \qquad \mathbf{y_n} + \sum_{\mathbf{i=1}}^{\mathbf{k_j}} \boldsymbol{\alpha_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}})\ \mathbf{y_{n-i}} = \sum_{\mathbf{i=0}}^{\mathbf{k_j}} \mathbf{h_{n-i}}\boldsymbol{\beta_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}})\,\mathbf{f_{n-i}}\ .$$

Assume again that $\mathbf{n} > \mathbf{k_j}$ and introduce the quantities:

$$(\mathbf{3.22}) \qquad \mathbf{h^*_{n-i}} = \sum_{\mathbf{q}}^{\mathbf{i-1}} \mathbf{h_{n-q}}\ , \qquad \mathbf{i = 1,\ 2,\ \ldots,\ k_j}\ , \qquad \mathbf{n > k_j}\ , \qquad \mathbf{h^*_n} \stackrel{\text{def}}{=} \mathbf{0}\ ,$$

and

$$(\mathbf{3.23}) \qquad \bar{\mathbf{h}}^*_{\mathbf{n-i}} = \frac{\mathbf{h^*_{n-i}}}{\mathbf{h_n}} = \mathbf{1} + \frac{\mathbf{h_{n-1}}}{\mathbf{h_n}} + \frac{\mathbf{h_{n-2}}}{\mathbf{h_n}} + \cdots + \frac{\mathbf{h_{n-i+1}}}{\mathbf{h_n}}\ , \qquad \mathbf{i = 1,\ 2,\ \ldots,\ k_j}\ , \qquad \bar{\mathbf{h}}^*_{\mathbf{n}} \stackrel{\text{def}}{=} \mathbf{0}\ .$$

The basic question now is: how to calculate the coefficients of formula (3.21)? It can be proved (see, for example, **Zlatev, 1983**) that if formula (3.18) is of order $\mathbf{p_j}$, then formula (3.21) is of the same order of accuracy and its coefficients can be obtained by the solving the system of linear algebraic equations formed by the equalities (3.24)-(3.26), which are listed below. It is immediately seen that these equalities are very similar to the equalities (3.4)-(3.6).

$$(\mathbf{3.24}) \qquad \mathbf{1} + \sum_{\mathbf{i=1}}^{\mathbf{k_j}} \boldsymbol{\alpha_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}}) = \mathbf{0}\ ,$$

$$(\mathbf{3.25}) \qquad \sum_{\mathbf{i=1}}^{\mathbf{k_j}} \bar{\mathbf{h}}^*_{\mathbf{n-i}}\boldsymbol{\alpha_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}}) = \sum_{\mathbf{i=0}}^{\mathbf{k_j}} \frac{\mathbf{h_{n-i}}}{\mathbf{h_n}}\boldsymbol{\beta_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}})\ ,$$

$$(\mathbf{3.26}) \qquad \frac{\mathbf{1}}{\mathbf{q!}}\sum_{\mathbf{i=1}}^{\mathbf{k_j}} \left(\bar{\mathbf{h}}^*_{\mathbf{n-i}}\right)^{\mathbf{q}} \boldsymbol{\alpha_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}}) = \frac{\mathbf{1}}{\mathbf{(q-1)!}}\sum_{\mathbf{i=0}}^{\mathbf{k_j}} \frac{\mathbf{h_{n-i}}}{\mathbf{h_n}}\left(\bar{\mathbf{h}}^*_{\mathbf{n-i}}\right)^{\mathbf{q-1}} \boldsymbol{\beta_{ji}}(\bar{\mathbf{h}}_{\mathbf{nj}})\ , \qquad \mathbf{q = 2, 3, \ldots, p_j}\ .$$

Moreover, it can also be shown that if the same time-stepsize has been used during the last $\mathbf{k_j}$ time-steps or, in other words, if $\mathbf{h_n = h_{n-1} = \cdots = h_{n-k_j} = h}$, then equalities (3.24)-(3.26) will be reduced to equalities (3.4)-(3.6) and formula (3.21) will be reduced to formula (3.18).

No proof has been given in this paragraph, but all proofs of the above statements can be found in **Zlatev (1983)**.

3.2.2. Zero-stability properties of an LM VSVFM

We have shown in the previous paragraph how to calculate the coefficients of the formulae (these coefficients depend on a sequence of several of the last time-stepsizes) that are to be used in an LM VSVFM. However, this does not solve all problems related to the development of reliable and efficient LM VSVFMs. The biggest of the remaining problems is caused by the fact that the variation of the stepsize and/or of the formula may affect the zero-stability of the computational process. This phenomenon was first pointed out by Nordsieck in 1962, see **Nordsieck (1962)**. After that many investigations were carried out, for example by **Piotrowski (1969)**, **Gear and Tu (1974)** and **Gear and Watanabe (1974)**. Two important results were established during these investigations:

(a) it is necessary to impose some restrictions in the variation of the time-stepsize and/or the formulae

and

(b) zero-stability can be guaranteed (under an assumption that several restrictions on the variation of the time-stepsizes and/or the formulae were properly introduced) when the Adams methods are used.

The last result, established in **Gear and Tu (1974)** and **Gear and Watanabe (1974)** for the Adams type formulae, was extended in **Zlatev (1978, 1981b, 1983)** for some more general methods.

Rather general restrictions on the time-stepsize can be introduced by the following relationships, where a constant $\mathbf{k_{start}}$ is defined in (3.19) and in the text before this formula:

(A) Assume that $\mathbf{h = \max_{n=k_{start}, k_{start}+1, \ldots, N} h_n}$. Then the inequality $\mathbf{hN \leq c}$ must be satisfied for some positive constant $\mathbf{c}$.

(B) There exist two constants $\bar{\alpha}$ and $\bar{\beta}$ such that $\mathbf{0 < \bar{\alpha} \leq h_n/h_{n+1} \leq \bar{\beta} < \infty}$ for all pairs of time-stepsizes, i.e. for $\mathbf{n = k_{start}, k_{start}+1, \ldots, N-1}$.

(C) If $\mathbf{N \to \infty}$, then $\mathbf{\lim(Nh) = c}$.

Similar restrictions were introduced first by **Piotrowski (1962)** and after that used in different proofs by **Gear and Tu (1974)**, **Gear and Watanabe (1974)** and **Zlatev (1978, 1981b, 1983)**. In the first two of these papers, it was proved that if the restrictions **(A)-(C)**, or some similar restrictions, are satisfied,

then the Adams formulae are zero-stable. The major result proved in the papers of Zlatev was related to LM VSVFMs defined by the following more general formulae:

$$(3.27) \quad y_n + \alpha_{j1}(\bar{h}_{nj})y_{n-1} + [1 - \alpha_{j1}(\bar{h}_{nj})]y_{n-2} = \sum_{i=0}^{k_j} h_{n-i}\beta_{ji}(\bar{h}_{nj})\, f_{n-i}\,, \qquad n > k_j\,.$$

The LM VSVFMs based on formulae of type (3.27) are zero-stable when the restrictions **(A)-(C)** are satisfied.

It is clear that if $\alpha_{j1}(\bar{h}_{nj}) = 1$ and $\beta_{j0}(\bar{h}_{nj}) = 0$ for all values of $\bar{h}_{nj}$, then formula (3.27) will be reduced to an Adams-Bashforth method. This means that the results proved by **Gear and Tu (1974)** and **Gear and Watanabe (1974)** are a special case of the results proved in **Zlatev (1978, 1981b, 1983)**. Furthermore, it is immediately seen that if $\alpha_{j1}(\bar{h}_{nj}) = 0$ and $\beta_{j0}(\bar{h}_{nj}) = 0$, then the formula (3.27) is reduced to a Nyström method. Thus, formula (3.27) can be considered as a linear combination of Nyström and Adams-Bashforth methods in this case. Similarly, if $\alpha_{j1}(\bar{h}_{nj}) = 0$ and $\beta_{j0}(\bar{h}_{nj}) \neq 0$, then the formula (3.27) is reduced to a generalized Milne-Simpson method. It follows that formula (3.27) can be considered as a linear combination of generalized Milne-Simpson and Adams-Moulton methods in this case.

More results as well as the proofs of the results given above can be found in **Zlatev (1983)**.

3.3. Absolute stability of the linear multistep methods

Convergence, consistency and zero-stability are fundamental requirements of the numerical methods for solving systems of ODEs defined by (1.1) and (1.2), which are unconditionally needed. If these requirements are satisfied, then the numerical solution y_n at a given grid-point t_n will be close to the corresponding value $y(t_n)$ of the exact solution when time-stepsize h is sufficiently small. However, in many situations, especially when large-scale scientific models are to be handled, it is much more important and useful to achieve sufficiently accurate results also when relatively **large** time-stepsizes are used during the solution of the system of ODEs defined by (1.1) and (1.2) with **explicit** numerical methods. Good **absolute stability properties** of the selected numerical method should additionally be required in the efforts to be able to perform successfully the computations and to reach the end-point t_N of the time-interval by using sufficiently large time-stepsizes.

Consider the linear multistep method defined by (3.1) and assume (as in Chapter 2) that it is applied in the numerical solution of the scalar and linear Dahlquist test-equation:

$$(3.28) \quad \frac{dy}{dt} = \lambda y, \quad t \in [0, \infty]\,, \quad y \in \mathbb{C}\,, \quad \lambda = \bar{\alpha} + \bar{\beta}i \in \mathbb{C}^-\,, \quad \bar{\alpha} \leq 0, \quad y(0) = \eta \in \mathbb{C}\,.$$

The following stability polynomial, which is formulated by the use of

(a) the two characteristic polynomials $\boldsymbol{\rho(z)}$ and $\boldsymbol{\sigma(z)}$ that were defined in equalities (3.3)

and

(b) the complex quantity $\boldsymbol{\nu = h\lambda}$ **,**

can be associated with the Dahlquist test-equation (3.28):

$$(\mathbf{3.29}) \qquad \boldsymbol{\pi(z,\nu) = \rho(z) - \nu\,\sigma(z)}\,, \qquad \mathbf{z} \in \mathbb{C}\,, \qquad \boldsymbol{\nu = \alpha + \beta i} \in \mathbb{C}^{-}\,, \qquad \boldsymbol{\alpha \leq 0}\quad .$$

Now several absolute stability definitions are very useful in connection with the treatment of systems of ODEs by linear multistep methods.

Definition 3.4: The linear multistep method is said to be absolutely stable for a given $\boldsymbol{\nu = h\lambda}$ if for that value of $\boldsymbol{\nu}$ the inequality $\boldsymbol{|z_i| < 1}$ holds for all roots $\boldsymbol{z_i}$, $\boldsymbol{i = 1}$, $\boldsymbol{2}$, ... , $\boldsymbol{k}$, of the stability polynomial (3.29).

■

Definition 3.5: The set of all values of $\boldsymbol{\nu \in \mathbb{C}^{-}}$, for which the linear multistep method is absolutely stable, forms the absolute stability region of this method.

■

Note that the absolute stability concept is defined in different ways for linear multistep methods and for Explicit Runge-Kutta Methods. In the first case a linear multistep formula is called absolutely stable for a given $\boldsymbol{\nu = \alpha + \beta i} \in \mathbb{C}^{-}$ if all the roots of its stability polynomial are within the unit disc. In the second case, an Explicit Runge-Kutta Method is absolutely stable for a given $\boldsymbol{\nu = \alpha + \beta i} \in \mathbb{C}^{-}$ if the value of the absolute stability polynomial is less than or equal to one. It is intuitively clear that the first requirement is more restrictive than the second one and this explains the fact that the absolute stability regions of the linear multistep methods are in general smaller than those of the Explicit Runge-Kutta Methods of the same order of accuracy. However, it should be emphasized here that the reason for introducing this concept is exactly **the same** in both cases: one wishes to ensure that if the exact solution is bounded, then the numerical solution is bounded too (or at least one has some good reasons to expect that the numerical solution remains bounded).

The absolute stability regions of different linear multistep methods (both explicit and implicit) are given in many text-books treating the numerical solution of linear multistep method. These regions are not very impressive, especially when the linear multistep methods are explicit. They are, as mentioned

in the previous paragraph, in general considerably smaller than the absolute stability regions of the corresponding Explicit Runge-Kutta Methods, which were studied in the previous chapter. More precisely, they are smaller than for the Explicit Runge-Kutta Methods of the same order of accuracy. Furthermore, the absolute stability regions are becoming smaller when the order of accuracy of the linear multistep methods is increasing. For six explicit linear multistep methods, more precisely for the Adams-Bashforth methods of order less than or equal to six, this fact is demonstrated in **Fig. 3.1**.

The absolute stability regions presented in Fig. 3.1 should be compared with the corresponding absolute stability regions given in Fig. 2.1 – Fig. 2.4 (the regions limited by the red curves) in the second chapter. Two major conclusions can be drawn when such a comparison is made:

(a) The absolute stability regions of the Adams-Bashforth methods are smaller than the corresponding absolute stability regions of the Explicit Runge-Kutta Methods. The only exception is the case $\mathbf{k = 1}$ resulting in the Explicit Euler Method (the Forward Euler Formula) both when the class of the linear multistep method is considered and when the class of Explicit Runge-Kutta Methods are studied. It should be mentioned here that the computational work needed to perform a time-step by the ERKMs is increased considerably when the order of accuracy becomes greater, because more function evaluations are needed, while the number of function evaluations remains the same (one only) when explicit linear multistep methods are used. This gives some compensation for the necessity to use smaller step-sizes, because of the stability restrictions, when explicit linear multistep methods are used.

(b) If $\mathbf{k > 1}$, then the absolute stability regions of the Adams-Bashforth methods are becoming smaller when the order of accuracy is increased, while the opposite effect is observed when the Explicit Runge-Kutta Methods are studied.

The length of the absolute stability intervals on the negative part of the real axis are also quickly decreasing when the order is increased. This fact is illustrated in **Table 3.1.** The results in this table should be compared with the corresponding results for the Explicit Runge-Kutta Methods presented in the previous chapter.

It should be noted here that the results related to the absolute stability restrictions, which were derived under the assumption that the Dahlquist scalar and linear test-problem is treated by the selected numerical method, can be generalized, precisely as in Chapter 2, for some linear systems of ODEs with constant coefficients. One should furthermore expect that, under some assumptions, the results will remain stable also in the case when linear systems of ODEs with variable coefficients and even for non-linear systems of ODEs (more details about the argumentation of these statements can be found in Chapter 2).

Some further details related to the absolute stability will be presented in Section 3.7, where this concept will be discussed in relation to the very important from a practical point of view predictor-corrector schemes.

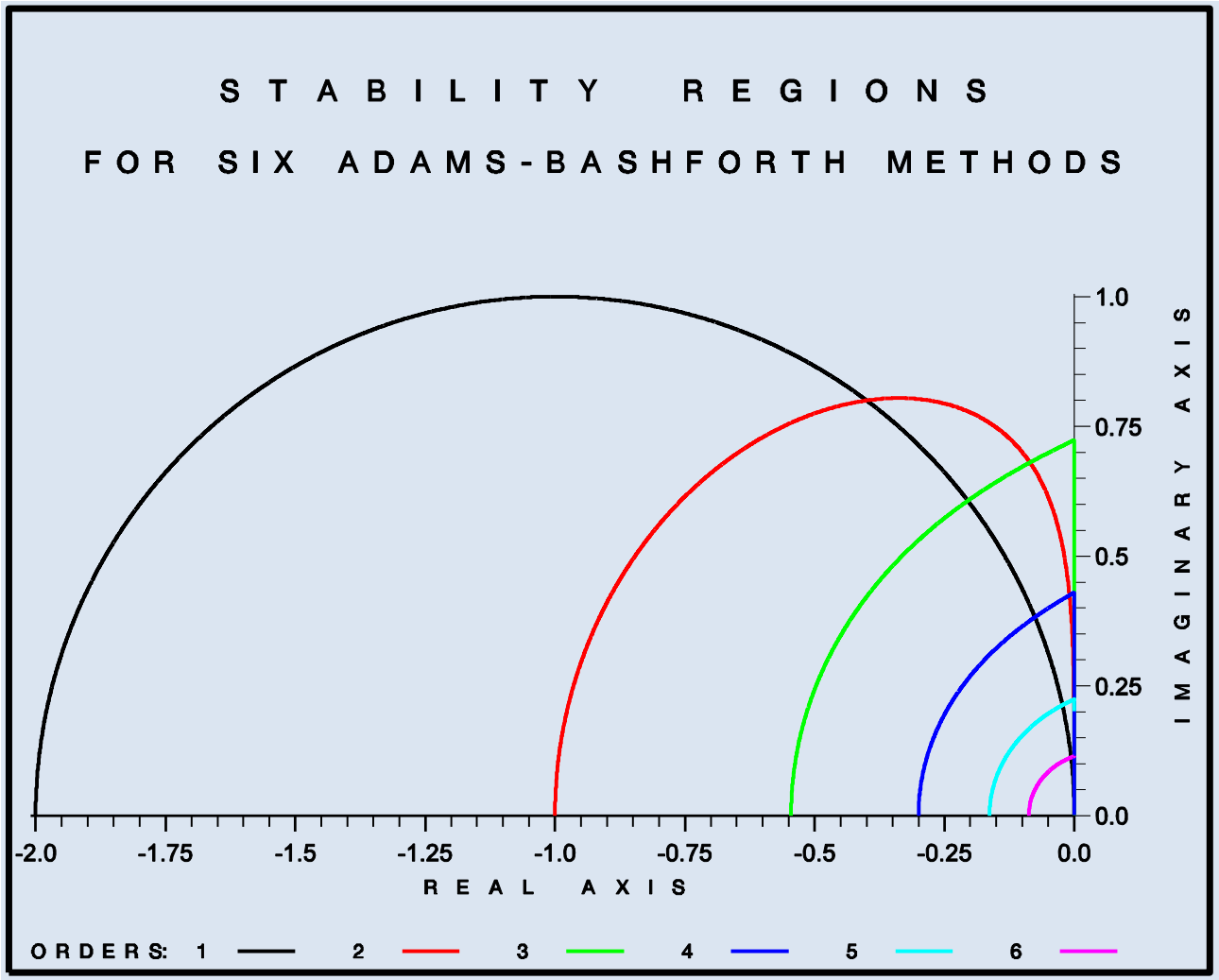

Figure 3.1
The absolute stability regions of the first six Adams-Bashforth formulae.

Order	Length of the absolute stability interval
1	2.000
2	1.000
3	0.545
4	0.300
5	0.163
6	0.086

Table 3.1
Lengths of the absolute stability intervals on the negative parts of the real axis for six Adams-Bashforth methods.

The absolute stability regions of the implicit Adams-Moulton methods are larger than those of the corresponding explicit Adams-Bashforth methods. The Backward Differentiation Formulae have pretty good stability properties. We shall not discussed these two classes of linear multistep methods

here, because our major objective in this chapter will be the description of the most important properties of the predictor-corrector schemes, which are in some sense similar to the Richardson Extrapolation. However, more details about the Backward Differentiation Formulae can be found in many text-books on the numerical solution of systems of ODEs as, for example, in **Butcher (2003)**, **Gear (1971)**, **Hairer, Nørsett and Wanner (1987)**, **Lambert (1991)** and **Shampine (1994)**.

3.4. Difficulties related to the implementation of Richardson Extrapolation

Let us assume that an attempt to introduce the Richardson Extrapolation together with a given linear multistep method is to be carried out in a similar way as described in Chapter 1. More precisely, if the calculations have already been performed for all grid-points $\mathbf{t_i}$, $(\mathbf{i = 1, 2, \ldots, n-1})$ by using an arbitrary linear $\mathbf{k}$-step method of order $\mathbf{p}$, and, thus, if approximations $\mathbf{y_i \approx y(t_i)}$ of the exact solution are available at the grid-points $\mathbf{t_i}$, $(\mathbf{i = 1, 2, \ldots, n-1})$, then the following **three actions** are to be carried out in order to obtain the next approximation $\mathbf{y_n}$:

(a) Perform one large time-step, with a time-stepsize $\mathbf{h}$ when the grid (1.6) is used or with a time-stepsize $\mathbf{h_n}$ if the grid (1.7) has been selected, in order to calculate an approximation $\mathbf{z_n}$ of $\mathbf{y(t_n)}$.

(b) Perform two small time-steps, with a time-stepsize $\mathbf{0.5\,h}$, when the grid (1.6) is used or with a time-stepsize $\mathbf{0.5\,h_n}$ if the grid (1.7) has been selected, in order to calculate another approximation $\mathbf{w_n}$ of $\mathbf{y(t_n)}$.

(c) calculate an approximation $\mathbf{y_n}$ by applying the formula:

$$\mathbf{y_n = \frac{2^p w_n - z_n}{2^p - 1}}. \qquad (3.30)$$

No problems appear during the first action if the grid (1.6) is used (at least, if a constant time-stepsize $\mathbf{h}$ is selected), but the second action is causing some technical problems, because some values of the approximations at several extra points, which do not belong to the grid (1.6), will be needed. The additional points are: $\mathbf{t_{n-0.5} = t_n - 0.5h}$, $\mathbf{t_{n-1.5} = t_n - 1.5h}$, …, $\mathbf{t_{n-0.5k} = t_n - k(0.5h)}$. The need of approximations at these points means that even if it is possible to carry on the procedure that was described above, its implementation will increase very considerably the storage requirements.

The solution of the other problem, the problem related to the fact that a procedure as that described by the above three actions can be applied only when an assumption that the grid used is equidistant is made, is also very problematic. The attempt to allow the usage of a variable time-stepsize strategy will be extremely difficult.

Because of these difficulties, we shall not try to implement the Richardson Extrapolation together with linear multistep methods. This seems not to be necessary either, because there exists an alternative approach which is

(A) rather efficient

and

(B) easily applicable.

Moreover, the major advantages of the Richardson Extrapolation are preserved when the alternative approach, the application of predictor-corrector schemes, is carefully implemented: **(a)** the accuracy can be improved, **(b)** the local truncation error can be estimated and **(c)** the time-stepsize can be controlled. It should be stressed also that the computational cost of a predictor-corrector scheme is considerably cheaper than the computational cost of the Richardson Extrapolation, which is an additional advantage of the predictor-corrector schemes.

In the following part of this chapter we shall show that predictor-corrector schemes can successfully be used instead of the Richardson Extrapolation. The predictor-corrector methods will be introduced in the next section of this chapter.

3.5. Introduction of some predictor-corrector schemes

The linear multistep methods are mainly used in the practical computations as predictor-corrector schemes. Consider a pair of an explicit and an implicit linear **k**-step methods written in the following form:

$$(3.31) \qquad y_n^{[0]} + \sum_{i=1}^{k} \alpha_i^{[0]} y_{n-i} = h \sum_{i=1}^{k} \beta_i^{[0]} f_{n-i}\,, \qquad \left|\alpha_k^{[0]}\right| + \left|\beta_k^{[0]}\right| > 0$$

and

$$(3.32) \qquad y_n^{[1]} + \sum_{i=1}^{k} \alpha_i^{[1]} y_{n-i} = h\beta_0^{[1]} f_n^{[0]} + h \sum_{i=1}^{k} \beta_i^{[1]} f_{n-i}\,, \qquad \left|\alpha_k^{[1]}\right| + \left|\beta_k^{[1]}\right| > 0 \ \wedge \ \beta_0^{[1]} \neq 0\,,$$

where the quantities f_{n-i} and $f_n^{[0]}$ are abbreviations of $f(t_{n-i}, y_{n-i})$ and $f\left(t_n, y_n^{[0]}\right)$ respectively for $i = 1, 2, \ldots, k$.

In fact the implicitness of the second formula, formula (3.32), disappears when it is combined with equality (3.31) in the manner shown above. Indeed, it is immediately seen that both (3.31) and (3.32) are explicit methods, because vector $\mathbf{f}_n^{[0]} = \mathbf{f}\left(\mathbf{t}_n\,,\ \mathbf{y}_n^{[0]}\right)$, which appears in the right-hand-side of (3.32), can be computed by using the vector $\mathbf{y}_n^{[0]}$ that was calculated by (3.31). The fact that (3.32) is an explicit formula is more clearly expressed in the very popular scheme consisting of four steps, which is shown below.

Step	Process	Formula used
Step 1	**Prediction**	$(3.33)\quad \mathbf{y}_n^{[0]} + \sum_{i=1}^{k} \alpha_i^{[0]}\, \mathbf{y}_{n-i} = \mathbf{h} \sum_{i=1}^{k} \beta_i^{[0]} \mathbf{f}_{n-1}$
Step 2	**Evaluation**	$(3.34)\quad \mathbf{f}_n^{[0]} = \mathbf{f}\left(\mathbf{t}_n\,,\ \mathbf{y}_n^{[0]}\right)$
Step 3	**Correction**	$(3.35)\quad \mathbf{y}_n^{[1]} + \sum_{i=1}^{k} \alpha_i^{[1]} \mathbf{y}_{n-i} = \mathbf{h}\beta_k^{[1]} \mathbf{f}_n^{[0]} + \mathbf{h} \sum_{i=1}^{k} \beta_i^{[1]}\, \mathbf{f}_{n-1}$
Step 4	**Evaluation**	$(3.36)\quad \mathbf{f}_n^{[1]} = \mathbf{f}\left(\mathbf{t}_n\,,\ \mathbf{y}_n^{[1]}\right)$

One should set $\mathbf{y_n} = \mathbf{y}_n^{[1]}$ and $\mathbf{f_n} = \mathbf{f}_n^{[1]}$, when the fourth step of the above computational scheme is completed and, after this action, everything is prepared for the calculations at the next time-step, i.e. for the calculation (by using the same scheme) of $\mathbf{y_{n+1}}$. The scheme given above is very often called **PECE** scheme, where **PECE** is an abbreviation of "prediction - evaluation (of the right-hand-side function **f**) – correction-evaluation (again of the right-hand-side function **f**)".

Sometimes the second evaluation of the right-hand-side function **f** is not carried out and the calculations at the time-step **n** are finished after the performance of the third step. The calculations for the next time-step **n+1** are prepared by setting $\mathbf{y_n} = \mathbf{y}_n^{[1]}$ and $\mathbf{f_n} = \mathbf{f}_n^{[0]}$. The resulting scheme is called **PEC** (which is an abbreviation of prediction-evaluation-correction). The amount of the computational work is reduced in this way: only one function evaluation is used in the **PEC** scheme instead of the two function evaluations which have to be performed when the **PECE** is selected. However, one should take into account the fact that some price has often to be paid for the reduction of the computational work, because the absolute stability regions of the **PEC** schemes are often considerably smaller than the corresponding absolute stability regions of the **PECE** schemes (see, for example, **Lambert, 1991**). This may sometimes require considerable reductions of the time-stepsize when the **PEC** scheme is selected.

The corrector formula (3.35) can be used not only once, but several, say **r**, times, where **r** is some integer greater than one. In this case the schemes $\mathbf{P(EC)^rE}$ and $\mathbf{P(EC)^r}$ can be used instead of **PECE** and **PEC** respectively. However, these schemes are obviously more expensive and one must

furthermore be careful with the stability of the computational process also in this choice, because the absolute stability properties of the $\mathbf{P(EC)^rE}$ and $\mathbf{P(EC)^r}$ schemes with $\mathbf{r > 1}$ are in some cases poorer than those of the **PECE** and **PEC** schemes (but these may also be better, especially in the case where $\mathbf{P(EC)^rE}$ schemes are used) .

It will be more informative sometimes to show the orders of accuracy of the involved in the above abbreviations of the different predictor-corrector schemes. Assume, for example, that the orders of accuracy of the predictor and the corrector are $\mathbf{p}$ and $\mathbf{p+1}$ respectively. Then the abbreviations can be rewritten as follows: $\mathbf{P_pEC_{p+1}}$, $\mathbf{P_pEC_{p+1}E}$, $\mathbf{P_p(EC_{p+1})^r}$ and $\mathbf{P_p(EC_{p+1})^rE}$.

One can also apply predictor-corrector schemes with several **different** correctors in an attempt to improve the absolute stability properties of the resulting predictor-corrector schemes. This approach will be discussed in the Section 3.8.

3.6. Local Error Estimation

The possibility of estimating in a sufficiently accurate way and in an easy manner the local truncation error at every time-step is one of the most important advantages of the predictor-corrector schemes based on linear multistep methods. Some examples, which demonstrate how the local error can be estimated, are given in this section. Much more examples and a more general presentation of the results can be found in **Lambert (1991)**.

Consider a $\mathbf{P_pEC_pE}$ (which means that it is assumed that the predictor formula and the corrector formula are of the **same order of accuracy**, of order $\mathbf{p}$). Then the following relationships can be derived, see again **Lambert (1991)**:

$$\mathbf{(3.37)} \qquad \mathbf{y(t_n) - y_n^{[0]} = C_{p+1}^{[0]} h^{p+1} y^{(p+1)}(t_n) + O(h^{p+2})}$$

and

$$\mathbf{(3.38)} \qquad \mathbf{y(t_n) - y_n^{[1]} = C_{p+1}^{[1]} h^{p+1} y^{(p+1)}(t_n) + O(h^{p+2})} \ ,$$

where the constants $\mathbf{C_{p+1}^{[0]}}$ and $\mathbf{C_{p+1}^{[1]}}$ do not depend on the time-stepsize $\mathbf{h}$ and can be calculated by using the equalities:

$$\mathbf{(3.39)} \qquad \mathbf{C_{p+1}^{[0]} = \frac{1}{(p+1)!} \sum_{i=1}^{k} i^{p+1} \alpha_i^{[0]} - \frac{1}{p!} \sum_{i=0}^{k} i^{p} \beta_i^{[0]}}$$

and

$$(\mathbf{3.40}) \quad \mathbf{C_{p+1}^{[1]}} = \frac{\mathbf{1}}{\mathbf{(p+1)!}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{i^{p+1}}\, \boldsymbol{\alpha}_{\mathbf{i}}^{\mathbf{[1]}} - \frac{\mathbf{1}}{\mathbf{p!}} \sum_{\mathbf{i=0}}^{\mathbf{k}} \mathbf{i^{p}}\, \boldsymbol{\beta}_{\mathbf{i}}^{\mathbf{[1]}} .$$

Subtracting (3.38) from (3.37) we obtain:

$$(\mathbf{3.41}) \quad \left(\mathbf{C_{p+1}^{[0]}} - \mathbf{C_{p+1}^{[1]}}\right) \mathbf{h^{p+1}} \mathbf{y^{(p+1)}}(\mathbf{t_n}) = \mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}} + \mathbf{O}(\mathbf{h^{p+2}}) .$$

Multiply both sides of (3.41) with $\mathbf{C_{p+1}^{[1]}}$, divide by $\mathbf{C_{p+1}^{[0]}} - \mathbf{C_{p+1}^{[1]}}$ and neglect the term $\mathbf{O}(\mathbf{h^{p+2}})$. The result is:

$$(\mathbf{3.41}) \quad \mathbf{C_{p+1}^{[1]}}\, \mathbf{h^{p+1}}\, \mathbf{y^{(p+1)}}(\mathbf{t_n}) = \frac{\mathbf{C_{p+1}^{[1]}}}{\mathbf{C_{p+1}^{[0]}} - \mathbf{C_{p+1}^{[1]}}} \left(\mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}}\right).$$

Substitute this value of $\mathbf{C_{p+1}^{[1]}}\, \mathbf{h^{p+1}}\, \mathbf{y^{(p+1)}}(\mathbf{t_n})$ in (3.38) to obtain the following expression for the evaluation of an approximate value of the local truncation error:

$$(\mathbf{3.42}) \quad \mathbf{y}(\mathbf{t_n}) - \mathbf{y_n^{[1]}} \approx \frac{\mathbf{C_{p+1}^{[1]}}}{\mathbf{C_{p+1}^{[0]}} - \mathbf{C_{p+1}^{[1]}}} \left(\mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}}\right).$$

It is seen that the expression on the right-hand side of (3.42) can be computed and, therefore, the above procedure is very similar to the Richardson Extrapolation. Moreover, we can introduce:

$$(\mathbf{3.43}) \quad \bar{\mathbf{y}}_{\mathbf{n}}^{\mathbf{[1]}} = \mathbf{y_n^{[1]}} + \frac{\mathbf{C_{p+1}^{[1]}}}{\mathbf{C_{p+1}^{[0]}} - \mathbf{C_{p+1}^{[1]}}} \left(\mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}}\right)$$

and it follows, from (3.42) and (3.43), that the new approximation $\bar{\mathbf{y}}_{\mathbf{n}}^{\mathbf{[1]}}$ has of order of accuracy $\mathbf{p+1}$, which is by one higher than the order of accuracy of both $\bar{\mathbf{y}}_{\mathbf{n}}^{\mathbf{[1]}}$ and $\bar{\mathbf{y}}_{\mathbf{n}}^{\mathbf{[0]}}$. Note too that if the Richardson Extrapolation is to be carried out, then one has to perform three time-steps (one large time-step with a time-stepsize $\mathbf{h}$ and two small time-steps with a time stepsize $\mathbf{0.5h}$) in order to

achieve the same effect. The corresponding computational work with the predictor-corrector scheme is equivalent to the performance of two time-steps (one with the predictor formula and the other with the corrector formula). This means that not only is the use of a predictor-corrector scheme very similar to the application of the Richardson Extrapolation, but it will often require a smaller amount of computations.

The device for calculating an approximation of the local truncation error by (3.42) and for computing a more accurate numerical solution by using (3.43) was derived by **Milne (1953)** and was the major motivation for the further development of predictor-corrector schemes. Note that it was derived in this section for schemes of the type $\mathbf{P_pEC_pE}$, but it can easily be extended also for predictor-corrector schemes of the type $\mathbf{P_pEC_p}$, $\mathbf{P_p(EC_p)^rE}$ and $\mathbf{P_p(EC_p)^r}$, see, for example, **Lambert (1991)**. The requirement is that the order of accuracy of the predictor formula must be equal to the order of the corrector formula. However, this is not a serious restriction, because

(a) there is obviously no meaning to use predictor formulae the order of accuracy of which is higher than that of the corrector formulae,

and

(b) if the order of the corrector formulae is higher than that of the predictor formulae it is very easy to obtain a reliable approximation of the local truncation error (for example, the difference $\mathbf{y_n^{[1]} - y_n^{[0]}}$ is giving a sufficiently good approximation of the local truncation error in the case where predictor-corrector schemes of the $\mathbf{P_pEC_{p+1}E}$ are used).

The above discussion demonstrates the fact that by using appropriate predictor-corrector schemes we can indeed achieve in an efficient way the same two effects, which were derived during the presentation of the Richardson Extrapolation in the previous sections:

(a) it is possible to calculate an estimation of the local truncation error, by using the term in the right-hand-side of (3.42)

and

(b) it is possible to calculate a new approximation $\mathbf{\bar{y}_n^{[1]}}$, which is more accurate than both $\mathbf{y_n^{[1]}}$ and $\mathbf{y_n^{[0]}}$.

However, one important question is still remaining and it has to be answered:

Will it be possible to apply the results presented above in the calculation of a good guess for the time-stepsize, which has to be applied during the next time-step or, in other words, is it possible to implement efficiently a variable stepsize variable formula predictor corrector schemes based on linear multistep methods?

It is not very easy to answer this question. The major problem is related to the fact that the quantities $\mathbf{C_{p+1}^{[0]}}$ and $\mathbf{C_{p+1}^{[1]}}$ from (3.39) and (3.40) will not be constants anymore when it is allowed to vary the time-stepsize (and also to vary the formulae if this will additionally increase the efficiency of the computational process). The solution of this difficult problem is presented below.

Consider, as in Section 3.2, a set $\mathcal{F} = \{\mathbf{F_1}, \mathbf{F_2}, ..., \mathbf{F_m}\}$, which now consists of $\mathbf{m}$ predictor-corrector schemes with constant coefficients and assume that the predictor-corrector scheme $\mathbf{F_j}$ is of the type a $\mathbf{P_pEC_pE}$. By using the same notation as that introduced in Section 3.2 and the relationship (3.26), the equalities (3.39) and (3.40) can be replaced by the following two relationships:

$$\mathbf{(3.44)} \quad \mathbf{C_{p+1}^{[0]}}(\bar{\mathbf{h}}_{\mathbf{nj}}) = \frac{\mathbf{1}}{\mathbf{(p+1)!}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{i^{p+1}}\, \boldsymbol{\alpha}_{\mathbf{ji}}^{[\mathbf{0}]}(\bar{\mathbf{h}}_{\mathbf{nj}}) - \frac{\mathbf{1}}{\mathbf{p!}} \sum_{\mathbf{i=0}}^{\mathbf{k}} \mathbf{i^{p}}\, \boldsymbol{\beta}_{\mathbf{ji}}^{[\mathbf{0}]}(\bar{\mathbf{h}}_{\mathbf{nj}})$$

and

$$\mathbf{(3.45)} \quad \mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}}) = \frac{\mathbf{1}}{\mathbf{(p+1)!}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{i^{p+1}}\, \boldsymbol{\alpha}_{\mathbf{ji}}^{[\mathbf{1}]}(\bar{\mathbf{h}}_{\mathbf{nj}}) - \frac{\mathbf{1}}{\mathbf{p!}} \sum_{\mathbf{i=0}}^{\mathbf{k}} \mathbf{i^{p}}\, \boldsymbol{\beta}_{\mathbf{ji}}^{[\mathbf{1}]}(\bar{\mathbf{h}}_{\mathbf{nj}}) .$$

It is seen that the quantities $\mathbf{C_{p+1}^{[0]}}(\bar{\mathbf{h}}_{\mathbf{nj}})$ and $\mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})$ depend on the time-stepsizes that were used during the last $\mathbf{j}$ time-steps. However, in the present special situation this is not very important. The important thing is that the coefficients in the terms participating in the right-hand-sides of (3.44) and (3.45) can be calculated (as was shown in Section 3.2). Therefore, the total quantities on the left-hand-side of the last two equalities can also be calculated and used in (3.42) and (3.43). This means that the following two relationships can be obtained when the stepsize can be varied:

$$\mathbf{(3.46)} \quad \mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})(\mathbf{h_n})^{\mathbf{p+1}}\, \mathbf{y}^{(\mathbf{p+1})}(\mathbf{t_n}) = \frac{\mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})}{\mathbf{C_{p+1}^{[0]}}(\bar{\mathbf{h}}_{\mathbf{nj}}) - \mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})} \left(\mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}}\right)$$

and

$$\mathbf{(3.47)} \quad \mathbf{y}(\mathbf{t_n}) - \mathbf{y_n^{[1]}} = \frac{\mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})}{\mathbf{C_{p+1}^{[0]}}(\bar{\mathbf{h}}_{\mathbf{nj}}) - \mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})} \left(\mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}}\right) + \mathbf{O}[(\mathbf{h_n})^{\mathbf{p+1}}] .$$

It is clear now that an approximation of the norm $\mathbf{E_n}$ of the local truncation error at time-step $\mathbf{n}$ is given by

$$(\mathbf{3.48}) \qquad \mathbf{E_n} = \left| \frac{\mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})}{\mathbf{C_{p+1}^{[0]}}(\bar{\mathbf{h}}_{\mathbf{nj}}) - \mathbf{C_{p+1}^{[1]}}(\bar{\mathbf{h}}_{\mathbf{nj}})} \right| \left\| \mathbf{y_n^{[1]}} - \mathbf{y_n^{[0]}} \right\| .$$

Assume now that some error tolerance $\mathbf{TOL}$ has been prescribed. Then three possibilities arise and must be analysed carefully:

(a) $\mathbf{E_n} > \mathbf{TOL}$,

(b) $\mathbf{E_n} = \mathbf{TOL}$

and

(c) $\mathbf{E_n} < \mathbf{TOL}$.

If **(a)** is satisfied, then the accuracy requirement introduced by the error tolerance has not been satisfied at time-step $\mathbf{n}$. The conclusion is that the time-step $\mathbf{n}$ was not successful and therefore it must be rejected, because the time-stepsize $\mathbf{h_n}$ was too big and should be reduced in order to calculate a more accurate approximation. In many codes based on the use of a variable stepsize, the time-stepsize is reduced by some constant factor. Very often it is halved and the time-step is repeated. It is not very recommendable to try to use $\mathbf{E_n}$ in an attempt to calculate more precisely the reduction factor, because the fact that $\mathbf{E_n} > \mathbf{TOL}$ indicates that the results from the calculations at the current time-step are not very reliable.

In the case where **(b)** is satisfied, everything is fine and it is best to perform the next time-size with time-stepsize $\mathbf{h_{n+1}} = \mathbf{h_n}$. Of course, in the practical computations one should require that the quantity $|\mathbf{E_n} - \mathbf{TOL}|$ is in some sense small instead of the stringent requirement $\mathbf{E_n} = \mathbf{TOL}$, which will practically never satisfied in computer arithmetic. The introduction of $|\mathbf{E_n} - \mathbf{TOL}|$ leads to some obvious modifications of **(a)** and **(c)**.

Case **(c)** is the most interesting one. The relationship $\mathbf{E_n} < \mathbf{TOL}$ indicates that it is possible to perform the next time-step with a larger time-stepsize. An attempt to find a good guess for the time-stepsize, which can successfully be used in the next time-step, can be carried out as follows. Assume that the following two equalities are satisfied:

$$(\mathbf{3.49}) \qquad \mathbf{E_n} = \boldsymbol{\gamma}\,\mathbf{TOL} \qquad \textbf{and} \qquad \mathbf{E_n} = \boldsymbol{\delta}(\mathbf{h_n})^{\mathbf{p+1}}$$

with some constant $\boldsymbol{\gamma} < \mathbf{1}$ and some other constant $\boldsymbol{\delta}$. It is clear that the constants $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ can easily be computed. $\mathbf{E_n}$ can be calculated by using (3.48), $\mathbf{TOL}$ is prescribed by the user and $\mathbf{h_n}$ is the last-time-stepsize. Therefore, $\boldsymbol{\gamma} = \mathbf{E_n}/\mathbf{TOL}$ and $\boldsymbol{\delta} = \boldsymbol{\gamma}\,\mathbf{TOL}/(\mathbf{h_n})^{\mathbf{p+1}}$, where the quantities participating in the right-hand-sides of the two equalities are known. It is clear now that the largest

guess for a time-stepsize $\mathbf{h_{n+1}}$, which is suitable for the next time-step, time-step $\mathbf{n+1}$, can be calculated by imposing a requirement to achieve

$$(\mathbf{3.50}) \qquad \mathbf{E_{n+1} = TOL = \delta(h_{n+1})^{p+1}}$$

when the next time-step is completed.

By using the relations $\mathbf{E_n = \delta(h_n)^{p+1}}$ and $\mathbf{TOL = \delta(h_{n+1})^{p+1}}$ one can find out that the following number is a good guess for an optimal time-stepsize, which can be used at the next time-step:

$$(\mathbf{3.51}) \qquad \mathbf{h_{n+1} = h_n \left(\frac{TOL}{E_n}\right)^{\frac{1}{p+1}}} \ .$$

There is no guarantee, of course, that the time-step $\mathbf{n+1}$ will be successful when the time-stepsize $\mathbf{h_{n+1}}$ calculated by (3.51) is applied. Therefore, the right-hand-side of (3.51) is multiplied by some precaution positive factor less than one in many practical codes for solving systems of ODEs with predictor-corrector schemes with a variable time-stepsize.

The approach described above can be used successfully when only one predictor-corrector scheme based on linear multistep methods, a predictor-corrector scheme of the type $\mathbf{P_pEC_pE}$, is used in the solution of the system of ODEs defined by (1.1) and (1.2). In many situations, it is worthwhile to vary also the predictor-corrector scheme. Assume that two other predictor-corrector schemes from the set $\mathcal{F} = \{\mathbf{F_1}, \mathbf{F_2}, ..., \mathbf{F_m}\}$ are appropriate candidates for the performance of the next time-step. In many practical codes, one selects as candidates for the next time-step predictor-corrector schemes of the types $\mathbf{P_{p-1}EC_{p-1}E}$ and $\mathbf{P_{p+1}EC_{p+1}E}$. Replace $\mathbf{h_{n+1}}$ by $\mathbf{h_{n+1}^{[p]}}$ in (3.51). Use

$$(\mathbf{3.52}) \qquad \mathbf{E_{n+1}^{[p]} = TOL} \qquad \textbf{and} \qquad \mathbf{E_{n+1}^{[p]} = \delta\left(h_{n+1}^{[p-1]}\right)^p}$$

instead of (3.50). Apply now equalities (3.52) and (3.49) to calculate a good guess for the time-stepsize $\mathbf{h_{n+1}^{[p-1]}}$ when the next time-step $\mathbf{n+1}$ will be carried out with the predictor-corrector scheme $\mathbf{P_{p-1}EC_{p-1}E}$. The result is:

$$(\mathbf{3.53}) \qquad \mathbf{h_{n+1}^{[p-1]} = h_n \left(h_n \frac{TOL}{E_n}\right)^{\frac{1}{p}}} \ .$$

Use

$$(3.54) \qquad E_{n+1}^{[p+2]} = TOL \qquad \text{and} \qquad E_{n+1}^{[p+2]} = \delta \left(h_{n+1}^{[p+1]} \right)^{p+2}$$

instead of (3.50). Use now equalities (3.54) and (3.49) to calculate a good guess for the time-stepsize $h_{n+1}^{[p+1]}$ when the next time-step $n+1$ will be carried out with the predictor-corrector scheme $P_{p+1}EC_{p+1}E$. The result is:

$$(3.55) \qquad h_{n+1}^{[p+1]} = h_n \left(\frac{1}{h_n} \frac{TOL}{E_n} \right)^{\frac{1}{p+2}} .$$

A good guess for the time-stepsize h_{n+1} that can be used to perform the next time-step $n+1$ can be obtained when the quantities on the right-hand-sides of (3.51), (3.53) and (3.55) are calculated. One can choose:

$$(3.56) \qquad h_{n+1} = \max \left(h_{n+1}^{[p-1]}, \; h_{n+1}^{[p]}, \; h_{n+1}^{[p+1]} \right)$$

and use this time-stepsize during the next-time step together with the respective predictor-corrector scheme.

It is possible to involve more than three predictor-corrector schemes in the search for the scheme, which will produce the largest possible time-stepsize for the next time-step.

To simplify the presentation of the results, we used above a set $\mathcal{F} = \{F_1, \; F_2, \; \ldots, \; F_m\}$ of predictor-corrector schemes of the type P_pEC_pE where parameter p is varied in the different schemes in this set. Other sets of predictor-corrector scheme can also be applied in a quite straight-forward way; see more details in **Lambert (1991)**.

The difficult problem related to the preservation of the zero-stability arises also when predictor-corrector schemes are used in a variable stepsize variable formula manner. It can be proved that in this case the method will be zero-stable if the last corrector formula in every scheme $F_j \in \mathcal{F}$, $j = 1, 2, \ldots, m$, is zero-stable (**Zlatev, 1985a, 1987, 1988, 1989**). This implies that the last corrector formula should be of the type:

$$(3.57) \qquad y_n + \alpha_{j1}(\bar{h}_{nj}) y_{n-1} + \left[1 - \alpha_{j1}(\bar{h}_{nj}) \right] y_{n-2} = \sum_{i=0}^{k_j} h_{n-i} \beta_{ji}(\bar{h}_{nj}) \, f_{n-i} ,$$

with $\beta_{j0}(\bar{h}_{nj}) \neq 0$ and, furthermore, the same restrictions on the time-stepsize as those proposed in Section 3.2 must be imposed. Zero-stable methods based on sets of predictor-corrector schemes, where both the time-stepsize and the scheme can be varied during the computational process, were proposed in **Thomsen and Zlatev (1979)** and **Zlatev and Thomsen (1979).** In the development of these methods an attempt to improve the absolute stability properties of the predictor-corrector schemes has been carried out. The absolute stability of predictor-corrector schemes will be studied in the next section.

It should be pointed out here that variable stepsize variable order schemes based on the use of Adams methods (Adams-Bashforth methods for the predictors and Adams-Moulton methods for the correctors) are normally used in efficient codes for solving non-stiff systems of ODEs defined by (1.1) and (1.2); see, for example, **Enright, Bedet, Farkas and Hull (1974)**, **Gear (1971)**, **Hindmarsh (1971)**, **Krogh (1973a, 1973b)**, **Shampine and Gordon (1975)**, **Shampine, Watts and Davenport (1976)**.

3.7. Absolute stability of the predictor-corrector schemes

We shall start the discussion in this section in the same way as in the beginning of Section 3.2, i.e. by pointing out that convergence, consistency and zero-stability are fundamental requirements, also when predictor-corrector schemes are to be used, related to the numerical solution of systems of ODEs defined by (1.1) and (1.2). These three properties are unconditionally needed, because only if such requirements are satisfied, then the numerical solution $\mathbf{y_n}$ at a given grid-point $\mathbf{t_n}$ will be close to the corresponding value $\mathbf{y(t_n)}$ of the exact solution when time-stepsize $\mathbf{h}$ is sufficiently small. However, in many situations, especially when large-scale scientific and engineering models are to be handled numerically on computers, it will be much more important and useful to achieve sufficiently accurate results even when relatively **large** time-stepsizes are used during the treatment of the systems of ODEs defined by (1.1) and (1.2) with explicit numerical methods. Therefore, good **absolute stability properties** of the selected predictor-corrector schemes should additionally be required in the efforts to be able to perform successfully the computations and to reach the end-point $\mathbf{t_N}$ of the time-interval by using larger time-stepsizes.

Consider a predictor-corrector scheme of type $\mathbf{P_pEC_pE}$ and assume (as was assumed in Chapter 2 and in Section 3.2) that this numerical device is applied in the numerical solution of the scalar and linear Dahlquist test-equation:

$$(\mathbf{3.58}) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \boldsymbol{\lambda}\,\mathbf{y}, \quad \mathbf{t} \in [\mathbf{0}, \infty], \quad \mathbf{y} \in \mathbb{C}, \quad \boldsymbol{\lambda} = \bar{\boldsymbol{\alpha}} + \bar{\boldsymbol{\beta}}\mathbf{i} \in \mathbb{C}^{-}, \quad \bar{\boldsymbol{\alpha}} \leq \mathbf{0}, \quad \mathbf{y(0)} = \boldsymbol{\eta} \in \mathbb{C}.$$

Denote by $\boldsymbol{\rho}^*(\mathbf{z})$ and $\boldsymbol{\sigma}^*(\mathbf{z})$ the two characteristic polynomials of the predictor formula and use the notation $\bar{\boldsymbol{\rho}}(\mathbf{z})$ and $\bar{\boldsymbol{\sigma}}(\mathbf{z})$ for the characteristic polynomials of the corrector formula. Use again as in Section 3.2 the notation $\boldsymbol{\nu} = \mathbf{h}\boldsymbol{\lambda}$. Assume that both the predictor and the corrector are linear

k-step methods given by (3.33) and (3.35) respectively. Then the following stability polynomial can be associated with the selected $\mathbf{P_pEC_pE}$ scheme:

$$(\mathbf{3.59}) \quad \boldsymbol{\pi_{PECE}(z,\nu) = \bar{\rho}(z) - \nu\,\bar{\sigma}(z) + \nu\beta_0^{[1]}[\rho^*(z) - \nu\,\sigma^*(z)]} \ .$$

Now several absolute stability definitions that are very similar to those introduced in Section 3.2 are very useful in connection with the treatment of systems of ODEs by predictor-corrector schemes.

Definition 3.6: The predictor-corrector scheme **PECE** consisting of the two linear **k**-step formulae given by (3.33) and (3.35) is said to be absolutely stable for a given $\nu = h\lambda$ if for that value of ν the inequality $|z_i| < 1$ holds for all roots z_i, $i = 1, 2, \ldots, k$, of its stability polynomial (3.59).

■

Definition 3.7: The set of all values of $\nu \in \mathbb{C}^-$, for which the predictor-corrector scheme **PECE** consisting of the two linear **k**-step formulae given by (3.33) and (3.35) is absolutely stable, forms the absolute stability region of this method.

■

It should be noted here that the orders of both the predictor formula and the corrector formula are not very important when absolute stability properties are studied. Therefore, **p** was omitted in formula (3.59) and this parameter will not be used in the remaining part of this section. Another parameter, the parameter **k**, is much more important in this situation, because this parameter determines the degrees of the four polynomials $\rho^*(z)$, $\sigma^*(z)$, $\bar{\rho}(z)$ and $\bar{\sigma}(z)$, which are involved in the absolute stability polynomial $\pi_{PECE}(z,\nu)$. All these five polynomials are of degree **k** and the important issue is the requirement the absolute values of all zeros of the absolute stability polynomial $\pi_{PECE}(z,\nu)$ to be less than or equal to one.

The absolute stability regions of different predictor-corrector schemes consisting of two linear **k**-step formulae are given in several text-books treating the numerical solution of systems of ODEs; see for example, **Lambert (1991)**. The above presentation is related to predictor-corrector schemes of the type **PECE**. Absolute stability polynomials for other types of predictor-corrector methods can be introduced in the following way. Define the two quantities:

$$(\mathbf{3.60}) \quad \boldsymbol{\xi \stackrel{def}{=} \nu\beta_0^{[1]}}$$

and

$$(3.61) \quad \omega_i(\xi) \stackrel{\text{def}}{=} \frac{\xi^i(1-\xi)}{1-\xi^i}, \qquad i = 1, 2, \ldots, r.$$

It is worthwhile to note here that $\omega_1(\xi) = \xi$.

In this notation, the absolute stability polynomials for predictor-corrector schemes of types $P_k(EC_{k+1})^r$ and $P_k(EC_{k+1})^rE$ are given by

$$(3.62) \quad \pi_{P(EC)^r}(z,\nu) = \beta_0^{[1]} z^k [\bar{\rho}(z) - \nu\,\bar{\sigma}(z)] + \omega_r(\xi)[\rho^*(z)\,\bar{\sigma}(z) - \bar{\rho}(z)\sigma^*(z)]$$

and

$$(3.63) \quad \pi_{P(EC)^rE}(z,\nu) = \bar{\rho}(z) - \nu\,\bar{\sigma}(z) + \omega_r(\xi)[\rho^*(z) - \nu\sigma^*(z)]$$

respectively.

It is clear that (3.63) will be reduced to (3.59) by setting $r = 1$.

Much more details can be found in **Lambert (1991)**. Two important facts should be stressed here instead of giving a full description of the derivation of the above absolute stability polynomials:

(A) The absolute stability regions for the predictor-corrector schemes are not very impressive, especially when predictor-corrector schemes of type $P(EC)^r$ are used. Even if $P(EC)^rE$ schemes are selected, all regions are in general considerably smaller than the absolute stability regions of the corresponding Explicit Runge-Kutta Methods (the Explicit Runge-Kutta Method of the same order of accuracy), which were studied in the previous chapter. Furthermore, the absolute stability regions are normally becoming smaller when the order of accuracy of the predictor-corrector schemes is increasing. Some results, which demonstrate the fact that the absolute stability regions of the predictor-corrector schemes are as a rule becoming smaller when the order of accuracy in increased, are shown in **Fig. 3.2**. The orders of the predictors is varied from one to six in **Fig. 3.2**. The corresponding orders of the correctors are varied from two to seven. This means that $P_kEC_{k+1}E$ schemes were used with $k = 1, 2, \ldots, 6$.

(B) The conclusion made in **(A)** indicates that it is worthwhile to try to improve the absolute stability properties of the predictor-corrector schemes. It must be emphasized here that one should try not only to improve the absolute stability properties but also to preserve the zero-stability properties of these numerical methods when these are to be used as

variable stepsize variable formula methods. The results of the efforts made in this directions will be described in the following part of this section.

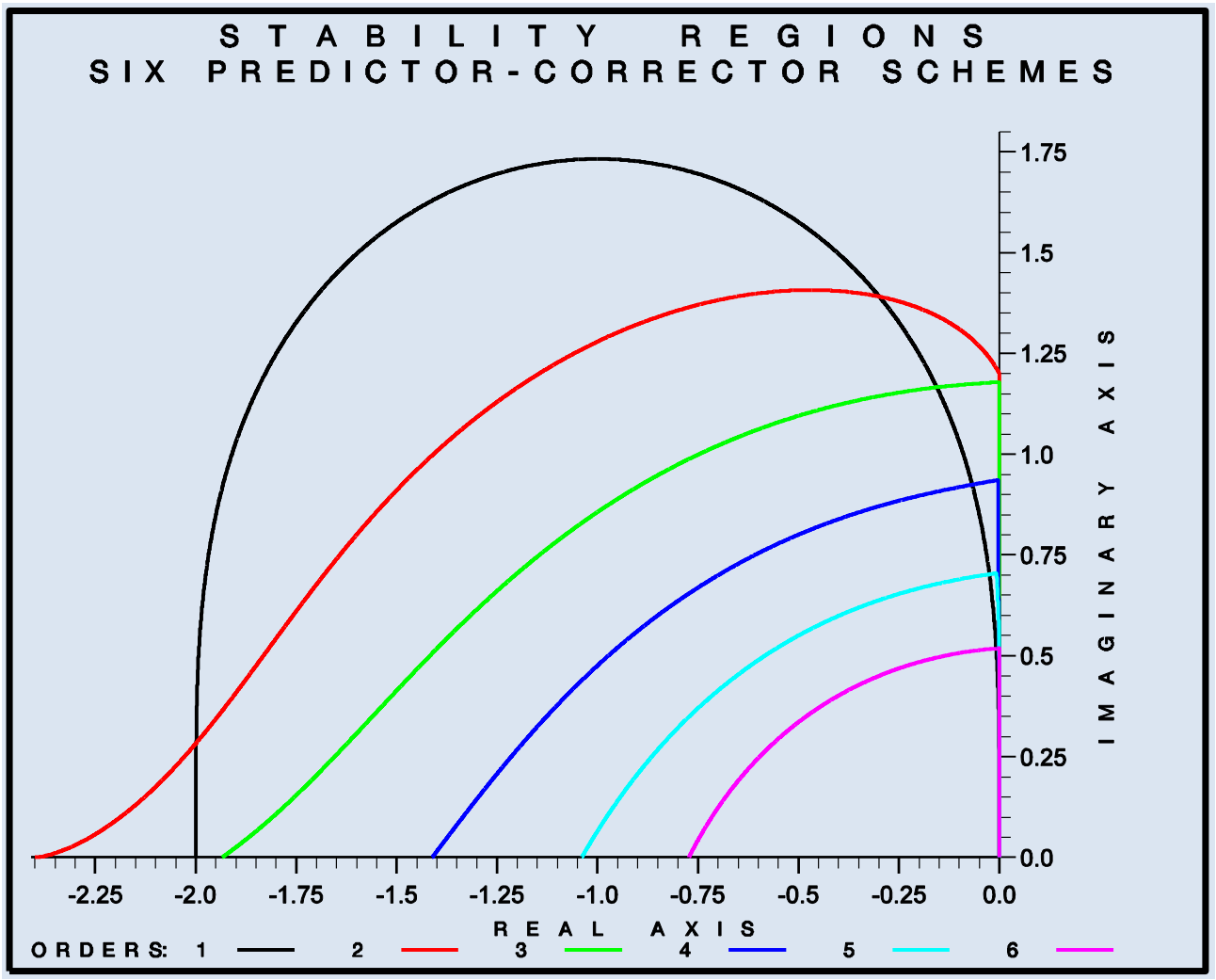

Figure 3.2
Absolute stability regions of six Adams $\mathbf{P_kEC_{k+1}E}$ schemes.

Consider the following four formulae:

$$(3.64) \qquad y_n = y_{n-1} + h \sum_{i=1}^{k} \overline{\beta}_i^{[0]} f_{n-i}, \qquad k = 2, 3, \ldots, 12,$$

$$(3.65) \qquad y_n = y_{n-2} + h \sum_{i=1}^{k} \widehat{\beta}_i^{[0]} f_{n-i}, \qquad k = 2, 3, \ldots, 12,$$

$$(3.66) \qquad y_n = y_{n-1} + h \sum_{i=0}^{k} \bar{\beta}_i^{[1]} f_{n-i}, \qquad k = 2, 3, \ldots, 12,$$

$$(3.67) \qquad y_n = y_{n-2} + h \sum_{i=0}^{k} \hat{\beta}_i^{[1]} f_{n-i}, \qquad k = 2, 3, \ldots, 12.$$

Adams-Bashforth formulae, Nyström formulae, Adams-Moulton formulae and generalized Milne-Simpson formulae are represented by (3.64), (3.65), (3.66) and(3.67) respectively. We shall additionally assume that the formulae (3.64) and (3.65) are of order of accuracy $p = k$, while the order of accuracy of the other two formulae is $p = k + 1$.

Multiply (3.64) by $\bar{\alpha}$ and (3.65) with $1 - \bar{\alpha}$. Add the two equalities obtained after performing these two actions. The result is:

$$(3.68) \qquad y_n + \bar{\alpha} y_{n-1} + (1 - \bar{\alpha}) y_{n-2} = h \sum_{i=1}^{k} \left[\bar{\alpha} \bar{\beta}_i^{[0]} + (1 - \bar{\alpha}) \hat{\beta}_i^{[0]} \right] f_{n-i}, \qquad k = 2, 3, \ldots, 12,$$

Multiply (3.66) by α and (3.67) with $1 - \alpha$. Add the two equalities obtained after performing these two actions. The result is:

$$(3.69) \qquad y_n + \alpha y_{n-1} + (1 - \alpha) y_{n-2} = h \sum_{i=0}^{k} \left[\alpha \bar{\beta}_i^{[1]} + (1 - \alpha) \hat{\beta}_i^{[1]} \right] f_{n-i}, \qquad k = 2, 3, \ldots, 12.$$

It is clear that the two formulae (3.68) and (3.69) form a predictor-corrector scheme of $P_k EC_{k+1} E$ type for $k = 2, 3, \ldots, 12$. Moreover, it is also clear that each of these predictor-corrector schemes depend on the two free parameters $\bar{\alpha}$ and α. These parameters were used in **Thomsen and Zlatev (1979)** and in **Zlatev and Thomsen (1979)** to design predictor-corrector schemes with improved absolute stability properties and to implement the so obtained schemes in a code for automatic integration of systems of ODEs. Some information about the properties of the improved predictor-corrector schemes are given in **Table 3.2**. Plots of the of six $P_k EC_{k+1} E$ schemes with $k = 3, \ldots, 8$ are given in **Fig. 3.3**.

Much more details about the predictor-corrector schemes with enhanced absolute stability properties can be found in the above two references. Plots, which show that the absolute stability regions of the $P_k EC_{k+1} E$ that are combinations of Adams-Bashforth predictors and Adams-Moulton correctors with Nyström predictors and Generalized Milne-Simpson correctors respectively are larger than those of predictor-corrector schemes based on Adams formulae, are also given there.

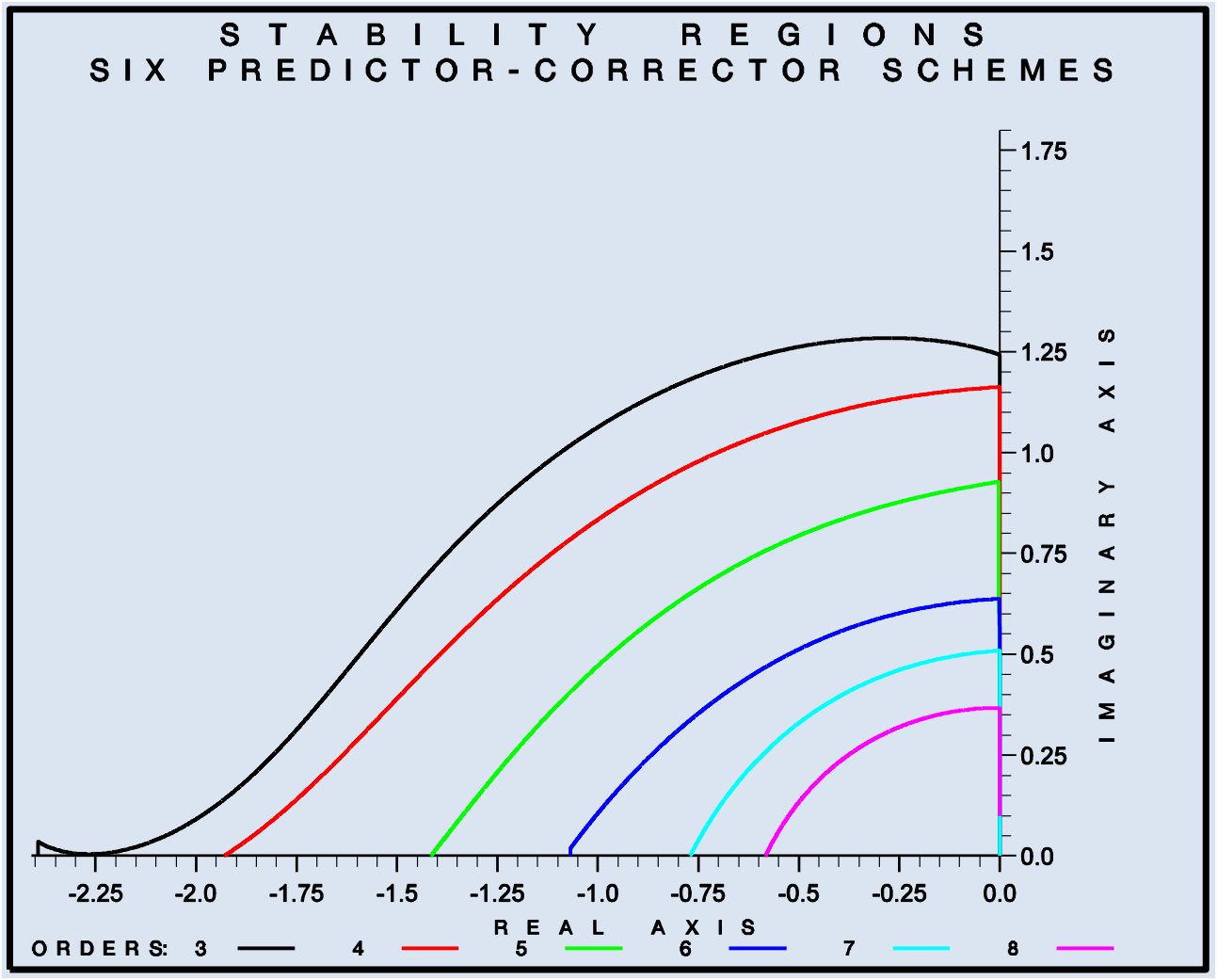

Figure 3.3

Absolute stability regions of six $\mathbf{P_kEC_{k+1}E}$ schemes, which are combinations of Adams-Bashforth predictors and Adams-Moulton correctors with Nyström predictors and Generalized Milne-Simpson correctors respectively.

3.8. Application of several different correctors

The results in Table 3.2 show clearly that the lengths of the absolute stability intervals on the imaginary axis are much smaller than the intervals of the absolute stability on the real axis when predictor-corrector schemes, which are based on linear multistep formulae, are used. Therefore, it is desirable sometimes to have longer absolute stability interval on the imaginary axis and near this axis. This is the case for many applications from different fields of science and engineering. This will be demonstrated in this section by taking an example arising in the field of environmental modelling. After that we shall show how to design special schemes, which have longer intervals of absolute stability along the imaginary axis.

K	α	$\bar{\alpha}$	Interval on the real axis	Interval on the imaginary axis
3	1.40	1.36	2.39 (1.92)	1.24 (1.18)
4	1.90	1.87	1.92 (1.41)	1.16 (0.93)
5	1.95	1.90	1.41 (1.04)	0.92 (0.64)
6	1.50	0.55	1.06 (0.76)	0.63 (0.52)
7	1.90	1.70	0.77 (0.58)	0.51 (0.37)
8	1.90	1.50	0.58 (0.44)	0.37 (0.26)
9	1.80	0.50	0.45 (0.34)	0.25 (0.18)
10	1.50	-2.50	0.36 (0.26)	0.16 (0.12)
11	1.80	-0.60	0.27 (0.21)	0.11 (0.07)
12	1.80	-1.00	0.21 (0.17)	0.07 (0.04)

Table 3.2

Lengths of the absolute stability intervals on the real and imaginary axes of ten predictor-corrector schemes of the type $\mathbf{P_kEC_{k+1}E}$ with improved absolute stability properties. The values of the two free parameters $\bar{\alpha}$ and α, for which the improved predictor-corrector schemes were obtained, are listed in the second and the third columns respectively. The corresponding lengths of the traditionally used Adams-Bashforth-Moulton predictor-corrector $\mathbf{P_kEC_{k+1}E}$ schemes are given for comparison in brackets.

3.8.1. An example from the field of environmental modelling

Consider the following systems of partial differential equations (PDEs), which can be used to study long-range transport of air pollutants, mainly sulphur and nitrogen pollutants) in the atmosphere:

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial x} - v\frac{\partial c}{\partial y} - w\frac{\partial c}{\partial z} + Q, \quad x \in [a_1, b_1], \ y \in [a_2, b_2], \ z \in [a_3, b_3], \ t \in [a, b]. \tag{3.70}$$

The quantities involved in (3.70) can be defined as follows:

(a) $c = c(x, y, z, t)$ is the unknown function, a vector containing the concentrations of the studied by the model pollutants; it will be assumed here that the chemical part of the model is very simple, and more precisely that either SO_2 and SO_4 or NO_2 and NO_3 are studied by the above model, i.e. the model can be used to study transport of either sulphur pollutants (sulphur di-oxide and sulphate) or nitrogen pollutants (nitrogen di-oxide and nitrate), which means that the model consists of two equations,

(b) $u = u(x, y, z, t)$, $v = v(x, y, z, t)$ and $w = w(x, y, z, t)$ are given functions representing the components of the wind velocity vector along the coordinate axes

and

(c) the chemical reactions are described by the function $\mathbf{Q} = \mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{t}, \mathbf{c})$, in fact it is assumed in connection with (3.70) that the chemical reactions are linear and that there are only two chemical species (this assumption is made only in Chapter 3).

The model described mathematically by the system of PDEs (3.70) must be considered together with some initial and boundary conditions. It is assumed that periodic boundary conditions are specified and that the values of function $\mathbf{c}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{t})$ are given for $\mathbf{t} = \mathbf{a}$ and for all values of the spatial variable in the beginning of the time-interval.

It is also assumed (this is typical for the large-scale air pollution models) that the spatial domain is a parallelepiped and an equidistant grid is introduced by $\mathbf{G_{LM}} = \mathbf{X_{M_x}} \times \mathbf{Y_{M_y}} \times \mathbf{Z_{M_z}}$ where $\mathbf{X_{M_x}}$, $\mathbf{Y_{M_y}}$ and $\mathbf{Z_{M_z}}$ are defined by

$$(\mathbf{3.71}) \qquad \mathbf{X_{M_x}} \stackrel{\text{def}}{=} \left\{ \mathbf{x_1} = \mathbf{a_1},\ \mathbf{x_i} = \mathbf{x_{i-1}} + \Delta\mathbf{x}\,,\ \mathbf{i} = \mathbf{1}, \mathbf{2}, \dots, \mathbf{2M_x},\ \Delta\mathbf{x} = \frac{(\mathbf{b_1} - \mathbf{a_1})}{\mathbf{2M_x}},\ \mathbf{x_{2M_x}} = \mathbf{b_1} \right\},$$

$$(\mathbf{3.72}) \qquad \mathbf{Y_{M_x}} \stackrel{\text{def}}{=} \left\{ \mathbf{y_1} = \mathbf{a_2},\ \mathbf{y_i} = \mathbf{y_{i-1}} + \Delta\mathbf{y},\ \mathbf{i} = \mathbf{1}, \mathbf{2}, \dots, \mathbf{2M_y},\ \Delta\mathbf{y} = \frac{(\mathbf{b_2} - \mathbf{a_2})}{\mathbf{2M_y}},\ \mathbf{x_{2M_y}} = \mathbf{b_2} \right\},$$

and

$$(\mathbf{3.73}) \qquad \mathbf{Z_{M_z}} \stackrel{\text{def}}{=} \left\{ \mathbf{z_1} = \mathbf{a_3},\ \mathbf{z_i} = \mathbf{z_{i-1}} + \Delta\mathbf{z},\ \mathbf{i} = \mathbf{1}, \mathbf{2}\,, \dots, \mathbf{2M_z},\ \Delta\mathbf{z} = \frac{(\mathbf{b_3} - \mathbf{a_3})}{\mathbf{2M_z}},\ \mathbf{z_{2M_z}} = \mathbf{b_3} \right\}.$$

Assume that the pseudospectral method is used to discretize the spatial derivatives in (3.70) and to transform the system of PDEs into a large system of ODEs of the type:

$$(\mathbf{3.74}) \qquad \frac{\mathbf{dg}}{\mathbf{dt}} = \mathbf{f}(\mathbf{t}, \mathbf{g}) \ .$$

The number of equations in this system is $\mathbf{M} = \mathbf{2M_x} \times \mathbf{2M_y} \times \mathbf{2M_z}$. This number can indeed be very large. If we assume that $\mathbf{2M_x} = \mathbf{96}$, $\mathbf{2M_y} = \mathbf{96}$ and $\mathbf{2M_z} = \mathbf{10}$, then $\mathbf{M} = \mathbf{92160}$. This number is already very large, but if the spatial domain has to cover the whole of Europe together with its surroundings, i.e. if long-range transport of air pollutants in different European countries is to be studied, then the surface cells will be $\mathbf{50\ km} \times \mathbf{50\ km}$, which is rather crude. It is clear that much smaller surface cells are needed in order to improve the reliability of the model results. Therefore, it is much more reasonable to take $\mathbf{2M_x} = \mathbf{480}$, $\mathbf{2M_y} = \mathbf{480}$ and keep $\mathbf{2M_z} = \mathbf{10}$ and this second discretization will results in a system of $\mathbf{4608000}$ ODEs when the three grids (3.71), (3.72) and

(3.73) are applied in connection with (3.70). The time-integration interval is very long (covering normally a year) and many thousands of time-steps have to be carried out. Moreover, many different scenarios have to be performed in order to study the sensitivity of the model results to the variation of different key parameters, such as emissions, temperature variations, boundary conditions, etc. It should be mentioned here that such grids and **much more chemical species** were used in connection with several studies related to the impacts of future climatic changes on high air pollution levels, see for example, **Csomós et al. (2006)**, **Zlatev (2010)**, **Zlatev, Georgiev and Dimov (2013b)**, **Zlatev, Havasi and Faragó (2011)**, **Zlatev and Moseholm (2008)**.

The above discussion shows clearly that it is necessary to try to solve the system of ODEs appearing in some environmental problems by using as large as possible time-stepsizes. In the remaining part of this section we will show that this is not an easy task and that the development of some special methods is needed in order to resolve it. The major problem is that the Jacobian matrix of (3.74) has eigenvalues close to the imaginary axis.

3.8.2. Some absolute stability considerations related to environmental modelling

Consider again (3.70) and assume $\mathbf{u}$ is a constant. Assume furthermore that the following conditions are additionally satisfied:

$$(\mathbf{3.75}) \qquad \mathbf{v} \equiv \mathbf{0}, \qquad \mathbf{w} \equiv \mathbf{0}, \qquad \mathbf{Q} \equiv \mathbf{0}\,.$$

Finally, assume that the pseudospectral method, see, for example, **Fornberg (1975, 1996)** or **Zlatev, Berkowicz and Prahm (1983a,b,c)**, is applied in the discretization of the so simplified equation (3.70). The result (see **Zlatev, Berkowitcz and Prahm, 1984a,b**) is:

$$(\mathbf{3.76}) \qquad \frac{\mathbf{dg}}{\mathbf{dt}} = -\mathbf{uSg}\,,$$

where $\mathbf{S}$ is a $\mathbf{2M_x \times 2M_x}$ skew-symmetric matrix and thus all its eigenvalues lie on the imaginary axis. In fact, the eigenvalues are given by the set $\mathbf{\Lambda} = \{-\mathbf{M} + \mathbf{j})\mathbf{i},\ \mathbf{j} = \mathbf{0}, \mathbf{1}, \ldots,\ \mathbf{2M}, \mathbf{i^2} = -\mathbf{1}\}$ and it can be proved, **Zlatev, Berkowicz and Prahm (1984c)**, that the computations will be stable when (3.76) is solved by the pseudospectral method if the time-stepsize satisfies the following inequality:

$$(\mathbf{3.77}) \qquad \mathbf{h} < \left[\frac{\mathbf{h_{imag}}}{|\mathbf{u}|\boldsymbol{\pi}}\ \frac{\mathbf{M}}{\mathbf{M}-\mathbf{1}}\right]\Delta\mathbf{x}\,,$$

where $\mathbf{h_{imag}}$ is the length of the absolute stability interval on the imaginary axis of the selected numerical method.

If other discretization methods, as, for example, methods based on the application of finite elements or finite differences, are used instead of the pseudospectral method, then (3.76) has to be modified, but the major parameters, Δx, h_{imag} and some quantities depending on the way in which the spatial derivatives are discretized and on the particular problem solved, would be represented also in the modified formula.

The inequality (3.77) shows that the stability of the computational process when (3.76) is solved numerically depends on

(a) the problem solved (because of the presence of u in the inequality),

(b) the spatial discretization used (mainly because of the presence of Δx)

and

(c) on the time-integration algorithm (because of the presence of the length h_{imag} of the stability interval on the imaginary axis).

When the problem is given and the spatial discretization is chosen, it will be important to select a numerical scheme with as large as possible length of the stability interval on the imaginary axis. It turns out, however, that this is not an easy task.

Assume that linear multistep methods, either single formulae or predictor-corrector schemes combining several formulae, are to be used. Then the following theorem, which can be deduced from a more general result proved in **Jeltsch and Nevanlinna (1981)**, see also **Jeltsch and Nevanlinna (1982)** and **Zlatev (1984a, 1985b)**, is establishing a barrier for the size of h_{imag} for the linear multistep methods.

Theorem 3.3: Consider a numerical method consisting either of a single linear multistep formula or of a predictor-corrector scheme with r correctors, where both the predictor and the correctors are linear multistep formulae. The length of the absolute stability interval of this methods satisfies the inequality:

$$(3.78) \quad 0 \leq h_{imag} \leq r + 1 ,$$

assuming here that $r = 0$ when the numerical method consists of a single explicit linear multistep formula.

■

The statement of Theorem 3.3 is rather restrictive. It is telling us that we must use several linear multistep formulae if we wish to increase the length of the absolute stability interval of the predictor-corrector scheme over a certain level. The requirement to ensure zero-stability of the selected predictor-corrector scheme is imposing another restriction: it is necessary to use the special predictor-corrector schemes derived in the previous section. Let us reiterate here that these predictor-corrector schemes depend on two parameters, $\bar{\alpha}$ for the predictor and α for the corrector. This means that the number

of free parameters that can be used in the efforts to increase the length of the absolute stability interval on the imaginary axis is **only two** for all values of $\mathbf{r}$ both when zero-stable predictor-corrector schemes of type $\mathbf{P_p(EC_{p+1})^r}$ and of type $\mathbf{P_p(EC_{p+1})^rE}$ are used. This number is increased from $\mathbf{2}$ to $\mathbf{r+1}$, when it is allowed to use different correctors in the predictor-corrector schemes. The increased number of free parameters will give us more freedom in the search for better predictor-corrector schemes.

It is worthwhile now to change a little the notation in this section. We shall replace $\bar{\boldsymbol{\alpha}}$ with $\boldsymbol{\alpha}^{[0]}$ when we refer to the predictor formula and $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}^{[i]}$, where $\mathbf{i=1,2,\ldots,r}$, when we refer to the $\mathbf{i^{th}}$ corrector. The $\mathbf{i^{th}}$ corrector will be denoted by $\mathbf{C_p^{[i]}}$, where the lower index is showing the order of accuracy, while the upper index is indicating the position of the formula in the predictor-corrector scheme.

An optimization process was organized by using the subroutine **VA10AD** from the Harwell Library of Fortran programs. This subroutine is discussed in **Fletcher (1972)**. We found out during the search that the predictor-corrector schemes with best $\mathbf{h_{imag}}$ have normally poor accuracy properties. A similar problem is also discussed in **Jeltsch and Nevanlinna (1982)**. Therefore, we had to make a compromise in order to obtain predictor-corrector schemes, which have both good absolute stability properties on the imaginary axis (but not the best possible) and which are sufficiently accurate.

Some results which were obtained during the search are listed in **Table 3.3**.

No.	Type of the predictor-corrector scheme	$\mathbf{h_{imag}}$
F1	$(-0.0041, 1.9884) - \mathbf{P_2EC_3^{[1]}E}$	1.995
F2	$(-0.8440, 1.8841) - \mathbf{P_3EC_4^{[1]}E}$	1.733
F3	$(-1.5909, 0.3748, 0.7710) - \mathbf{P_2EC_2^{[1]}EC_2^{[2]}E}$	2.330
F4	$(-1.7631, 0.5577, 0.0044) - \mathbf{P_2EC_2^{[1]}EC_3^{[2]}E}$	2.462
F5	$(-0.0026, 1.9919, 0.0050) - \mathbf{P_2EC_3^{[1]}EC_4^{[2]}E}$	2.842
F6	$(0.2885, 0.9980, 0.3446, 1.0233) - \mathbf{P_2EC_2^{[1]}EC_2^{[2]}EC_2^{[3]}E}$	3.345
F7	$(0.0375, 0.3510, 0.7124, 0.0336) - \mathbf{P_2EC_2^{[1]}EC_2^{[2]}EC_3^{[3]}E}$	3.296
F8	$(-2.1687, 0.2109, 0.3695, 0.2191) - \mathbf{P_2EC_2^{[1]}EC_3^{[2]}EC_4^{[3]}E}$	3.147

Table 3.3

Eight predictor-corrector schemes with long absolute stability intervals $\mathbf{h_{imag}}$ on the imaginary axis. The values of the free parameters, for which these schemes were found, are given in brackets.

The predictor-corrector schemes with long absolute stability intervals on the imaginary axis were derived under the assumption that the analogue (3.76) of the famous Dahlquist scalar and linear test-equation (3.28) is solved. In fact, we are interested in the solution of the simplified air pollution model (3.70). Absolute stability cannot be guaranteed when the system of PDEs (3.70) has to be treated

numerically. However, by applying arguments similar to those used in Section 2.1 and in Section 3.3, one can conclude that stable computational process should be expected when all eigenvalues of the Jacobian matrix are close to the imaginary axis. To enhance this expectation care was taken (during the derivation of the predictor-corrector schemes listed in Table 3.3) to ensure that absolute stability regions of these schemes contain also some parts of the complex half-plane $\mathbb{C}^-$ that are located close to the imaginary axis. When this additional task was accomplished, we can expect the computational process to be stable, assuming additionally for this special case that the pseudospectral method is used to discretize the spatial derivatives in (3.70) on the grids (3.71)-(3.73), if the following condition is satisfied:

$$(3.79) \qquad \mathbf{h} < \frac{\mathbf{h_{imag}}}{\pi}\left[\frac{\mathbf{u}^*(\mathbf{M_x}-\mathbf{1})}{\mathbf{M_x}\Delta\mathbf{x}} + \frac{\mathbf{v}^*(\mathbf{M_y}-\mathbf{1})}{\mathbf{M_y}\Delta\mathbf{y}} + \frac{\mathbf{w}^*(\mathbf{M_z}-\mathbf{1})}{\mathbf{M_z}\Delta\mathbf{z}}\right]^{-1},$$

with

$$(3.80) \qquad \mathbf{u}^* \stackrel{\text{def}}{=} \max_{\mathbf{x}\in[\mathbf{a_1},\mathbf{b_1}],\ \mathbf{y}\in[\mathbf{a_2},\mathbf{b_2}],\ \mathbf{z}\in[\mathbf{a_3},\mathbf{b_3}],\ \mathbf{t}\in[\mathbf{t}^*,\mathbf{t}^{**}]} \{\,|\mathbf{u}(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{t})|\,\},$$

$$(3.81) \qquad \mathbf{v}^* \stackrel{\text{def}}{=} \max_{\mathbf{x}\in[\mathbf{a_1},\mathbf{b_1}],\ \mathbf{y}\in[\mathbf{a_2},\mathbf{b_2}],\ \mathbf{z}\in[\mathbf{a_3},\mathbf{b_3}],\ \mathbf{t}\in[\mathbf{t}^*,\mathbf{t}^{**}]} \{\,|\mathbf{v}(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{t})|\,\},$$

$$(3.82) \qquad \mathbf{w}^* \stackrel{\text{def}}{=} \max_{\mathbf{x}\in[\mathbf{a_1},\mathbf{b_1}],\ \mathbf{y}\in[\mathbf{a_2},\mathbf{b_2}],\ \mathbf{z}\in[\mathbf{a_3},\mathbf{b_3}],\ \mathbf{t}\in[\mathbf{t}^*,\mathbf{t}^{**}]} \{\,|\mathbf{w}(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{t})|\,\},$$

where the interval $[\mathbf{t}^*, \mathbf{t}^{**}]$ contains the current integration point (typically denoted by $\mathbf{t_n}$).

It was assumed until now that a constant stepsize is used. This requirement can be removed if some restrictions on the variation of the stepsize, as those imposed in § 3.2.2, are imposed. If this has been done, then $\mathbf{h}$ should be replaced by $\mathbf{h_n}$, where $\mathbf{n}$ is the time-step that has to be performed.

3.8.3. Numerical Experiments

Some of the predictor-corrector schemes presented in Table 3.3 were applied to develop variable stepsize variable formula methods in **Zlatev (1984)** and **Zlatev, Berkowicz and Prahm (1984a,b)**. It is not necessary to describe in detail the experiments, but it is worthwhile to emphasize two important facts:

(A) It was necessary to introduce some diffusion terms in (3.70) in order to obtain more reliable results. This is not causing additional stability problems because the advection terms are dominant and restricting the choice of the time-stepsizes.

(B) The calculated results were compared with measurements collected in many European countries within the project EMEP (European Monitoring and Evaluation Program), which is still functioning; see the **EMEP Home Web-page** in the reference list, where detailed information about this program, including numerous reports, can be found. It should be mentioned too that also input data (meteorological data and emission data) for the model were obtained from EMEP.

The usefulness of selecting predictor-corrector schemes with several different correctors is shown by comparing these methods with methods in which the same correctors are used in the predictor-corrector schemes. Let us call the method obtained when the first approach is used as a variable stepsize variable formula method **Algorithm 1**, while the name **Algorithm 2** is used for the second approach, where the correctors remain the same.

Three predictor-corrector schemes are used in **Algorithm 2**. These formulae are given in Table 3.4. The predictor-corrector schemes used in **Algorithm 1** are given in Table 3.5. These predictor-corrector schemes have not the best possible stability intervals, but they are more accurate than those with optimal lengths of the absolute stability intervals along the imaginary axis, which were listed in Table 3.3.

No.	Type of the predictor-corrector scheme	h_{imag}
G1	$(-0.05, 1.85) - P_2EC_3^{[1]}E$	1.95
G2	$(-0.85, 1.80) - P_3EC_4^{[1]}E$	1.70
G3	$(1.00, 1.00) - P_3EC_4^{[1]}E$	1.17

Table 3.4

The three more traditional predictor-corrector schemes, which are used in Algorithm 2 and their absolute stability intervals h_{imag} on the imaginary axis. The values of the free parameters, for which these schemes were found, are given in brackets. The third scheme is based on the commonly used Adams- Bashforth predictor and Adams-Moulton corrector.

No.	Type of the predictor-corrector scheme	h_{imag}
H1	$(-0.3412, 0.3705, 0.5766, 0.4548) - P_2EC_2^{[1]}EC_2^{[2]}EC_3^{[3]}E$	3.26
H2	$(0.6500, 1.500, 1.0000) - P_2EC_3^{[1]}EC_4^{[2]}E$	2.51
H3	$(-0.0900, 1.600) - P_3EC_4^{[1]}E$	1.62

Table 3.5

The three predictor-corrector schemes used in Algorithm 1, in which an attempt to balance the requirement for long absolute stability intervals h_{imag} on the imaginary axis with a requirement for achieving better accuracy is made. The values of the free parameters, for which these schemes were found, are given in brackets.

A careful comparison of the predictor-corrector schemes presented in Table 3.4 with those presented in Table 3.5 shows that schemes in Table 3.4, which is based on the use of only two formulae, are computationally cheaper than the first two schemes in Table 3.5 involving three and four formulae respectively. On the other hand, all predictor-corrector schemes used in Algorithm 1 have better stability properties than the stability properties of the corresponding schemes of Algorithm 2; compare the values of $\mathbf{h_{imag}}$ in the two tables.

Both a stability control and an accuracy control was carried out at every time-step in the runs of the model with Algorithm 1 and Algorithm 2. The stability control is based on the condition (3.79), while the accuracy control is based on the results presented in Section 3.6.

Some numerical results obtained in a long run, covering meteorological data for one year, are presented in Table 3.6. It is seen that the algorithm based on several different correctors performs better, especially in the three-dimensional case.

Version of the model	**Two-dimensional version**		**Three-dimensional version**	
	Time-steps	**CPU-times**	**Time-steps**	**CPU-times**
Algorithm 1	636 (60.3%)	341.5 (92.6%)	638 (60.3%)	2574.9 (78.0%)
Algorithm 2	1055	368.9	1058	3299.3

Table 3.6

Numbers of time-steps and CPU-times (measured in seconds) when Algorithm 1 and Algorithm 2 are run over a long time-interval with meteorological and emission data for one year. The reductions (measures in percent) when Algorithm 1 is used are given in brackets.

3.9. A-stability of the linear multistep methods

One must require more than absolute stability of the numerical method that should be applied when the solved system of ODEs is stiff or very stiff. A-stability is more appropriate in this case.

Definition 3.8: A linear multistep method is said to be A-stable if its absolute stability region contains **all** points in the negative part of the complex plane (i.e. if all roots of the stability polynomial $\boldsymbol{\pi(z,\nu) = \rho(z) - \nu\,\sigma(z)}$ satisfy the inequality $\boldsymbol{|z_i| < 1}$ for $\boldsymbol{i = 1, 2, \ldots, k}$ always when $\boldsymbol{\nu \in \mathbb{C}^-}$).

■

Unfortunately, only very few linear multistep methods are A-stable. The result showing that this is true has been established by G. Dahlquist in 1963, **Dahlquist (1963).** It became well-known in the literature

about numerical solution of systems of ODEs as the **second Dahlquist barrier**; see, for example, **Lambert (1991)**.

Theorem 3.4: The following three statements are true:

(A) an explicit linear multistep method cannot be A-stable,

(B) the order of accuracy of an A-stable linear multistep method cannot exceed two

and

(C) the second order linear multistep method with a smallest error constant is the Trapezoidal Rule.

■

The Trapezoidal Rule and the linear multistep method with $\mathbf{k = 1}$ belong also to the class of the $\boldsymbol{\theta}$-methods and will be studied in the next chapter. Some other stability concepts will also be introduced and discussed in Chapter 4.

Theorem 3.3 indicates that the use of linear multistep methods in the solution of stiff systems of ODEs is connected with many problems, but as mentioned in the previous sections of this chapter the Backward Differentiation Formulae can be applied in the solution of certain classes of stiff problems and there are several well written packages of computer programs based on these methods; see, for example, **Hindmarsh (1980)**.

3.10. Coefficients of some popular linear multistep methods

Some of the readers might wish to apply suitable linear multistep methods either in their research or in the treatment of their applications. The coefficients of the most popular linear multistep methods, which are listed below, will be very useful for such readers. It must be emphasized here that it is not very easy to find such a list in the commonly used text-books, where as a rule only methods of lower orders are given.

It is worthwhile to present also in this section the formulae for the Adams-Bashforth Methods, the Nyström Methods, the Adams-Moulton Methods and the Generalized Milne-Simpson Methods:

$$(\mathbf{3.83}) \qquad \mathbf{y_n = y_{n-1} + h \sum_{i=1}^{k} \bar{\beta}_i^{[0]} f_{n-i}},$$

$$(3.84) \qquad y_n = y_{n-2} + h \sum_{i=1}^{k} \widehat{\beta}_i^{[0]} f_{n-i} ,$$

$$(3.85) \qquad y_n = y_{n-1} + h \sum_{i=0}^{k} \overline{\beta}_i^{[1]} f_{n-i} ,$$

$$(3.86) \qquad y_n = y_{n-2} + h \sum_{i=0}^{k} \widehat{\beta}_i^{[1]} f_{n-i} .$$

The coefficients of the Adams-Bashforth, Nyström, Adams-Moulton and the Generalized Milne-Simpson methods are also listed for $1 \leq k \leq 10$ in Table 3.7, Table 3.8, Table 3.9 and Table 3.10 respectively.

k	$\xi\overline{\beta}_1^{[0]}$	$\xi\overline{\beta}_2^{[0]}$	$\xi\overline{\beta}_3^{[0]}$	$\xi\overline{\beta}_4^{[0]}$	$\xi\overline{\beta}_5^{[0]}$	$\xi\overline{\beta}_6^{[0]}$	$\xi\overline{\beta}_7^{[0]}$	$\xi\overline{\beta}_8^{[0]}$	$\xi\overline{\beta}_9^{[0]}$	$\xi\overline{\beta}_{10}^{[0]}$	ξ
1	1										1
2	3	-1									2
3	23	-16	5								12
4	55	-59	37	-9							24
5	1901	-3774	2616	-1274	251						720
6	4277	-7923	9982	-7298	2877	-475					1440
7	198721	-447288	705549	-688256	407139	-134472	19087				60480
8	434241	-1152169	2183877	-26664477	21002243	-1041723	295767	-36799			120960
9	140097247	-43125206	95476786	-139855262	137968480	-91172642	38833486	-9664106	1070017		3628800
10	30277247	-105995189	265932680	-454661776	538363838	-444772162	252618224	-94307320	20884811	-2082753	7257600

Table 3.7

The products of the coefficients of the Adams-Bashforth k-step methods, where $1 \leq k \leq 10$, with the multipliers ξ, which are given in the last column. This means that the coefficients are the ratios of the numbers given in columns 2-11 and the corresponding multipliers ξ (the multipliers ξ located in the same rows). The order p of each formula is equal to k. The first two of the coefficients α are equal to 1 and -1 respectively, while the remaining coefficients α are equal to 0.

The Backward Differentiation Formulae are given by

$$(3.87) \qquad y_n + \sum_{i=1}^{k} \alpha_i y_{n-i} = h \beta_0 f_n , \qquad |\alpha_k| > 0 , \qquad |\beta_0| > 0 , \qquad n = k, \; k+1, \ldots , \; N .$$

These formulae are zero-stable up to sixth order of accuracy, while for $k > 7$ all Backward Differentiation Formulae are zero-unstable. This fact was established in **Cryer (1972)**. Therefore, in

Table 3.11, which is given below, only the coefficients of the Backward Differentiation Formulae with $\mathbf{1 \leq k \leq 6}$ are listed.

k	$\xi\hat{\beta}_1^{[0]}$	$\xi\hat{\beta}_2^{[0]}$	$\xi\hat{\beta}_3^{[0]}$	$\xi\hat{\beta}_4^{[0]}$	$\xi\hat{\beta}_5^{[0]}$	$\xi\hat{\beta}_6^{[0]}$	$\xi\hat{\beta}_7^{[0]}$	$\xi\hat{\beta}_8^{[0]}$	$\xi\hat{\beta}_9^{[0]}$	$\xi\hat{\beta}_{10}^{[0]}$	ξ
2	2	0									1
3	7	-2	1								3
4	8	-5	4	-1							3
5	269	-266	294	-146	29						90
6	297	-406	574	-426	169	-28					90
7	13613	-23886	41193	-40672	24183	-8010	1139				3780
8	14720	-31635	64440	-79417	62928	-31257	8888	-1107			3780
9	439777	-1208066	2839756	-4195622	4155230	-2750822	1173196	-292226	32377		113400
10	505625	-1492898	3979084	-6854054	8141878	-6738470	3831628	-1431554	317209	-31648	113400

Table 3.8

The products of the coefficients of the Nyström **k**-step methods, where $\mathbf{2 \leq k \leq 10}$, with the multipliers $\boldsymbol{\xi}$, which are given in the last column. This means that the coefficients are the ratios of the numbers given in columns 2-11 and the corresponding multipliers $\boldsymbol{\xi}$ (the multipliers $\boldsymbol{\xi}$ located in the same rows). The order $\mathbf{p}$ of each formula is equal to $\mathbf{k}$. The first and the third coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{1}$ and $\mathbf{-1}$ respectively, while the remaining coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{0}$.

k	$\xi\bar{\beta}_0^{[1]}$	$\xi\bar{\beta}_1^{[1]}$	$\xi\bar{\beta}_2^{[1]}$	$\xi\bar{\beta}_3^{[1]}$	$\xi\bar{\beta}_4^{[1]}$	$\xi\bar{\beta}_5^{[1]}$	$\xi\bar{\beta}_6^{[1]}$	$\xi\bar{\beta}_7^{[1]}$	$\xi\bar{\beta}_8^{[1]}$	$\xi\bar{\beta}_9^{[1]}$	$\xi\bar{\beta}_{10}^{[1]}$
1	1	1									
2	5	8	-1								
3	9	19	-5	1							
4	251	646	-264	106	-19						
5	475	1427	-798	582	-173	27					
6	19087	65112	-46461	37504	-20211	6312	-863				
7	36799	139849	-121797	123133	-88547	41499	-11351	1375			
8	1070017	4467094	-4604594	5595358	-5033120	3146338	-1291214	312874	-33953		
9	2082753	9449717	-11271304	16002320	-17283646	13510082	-7394032	2687864	-583435	57281	
10	134211265	656185652	-890175549	1446205080	-1823311566	1710774528	-1170597042	567450984	-184776195	36284876	-3250433

Table 3.9

The products of the coefficients of the Adams-Moulton **k**-step methods, where $\mathbf{1 \leq k \leq 10}$, with the multipliers $\boldsymbol{\xi}$, which are given in the last column. This means that the coefficients are the ratios of the numbers given in columns 2-11 and the multipliers $\boldsymbol{\xi}$ (the multipliers $\boldsymbol{\xi}$ corresponding to the ten rows of the tables are: $\boldsymbol{\xi_1 = 2}$, $\boldsymbol{\xi_2 = 12}$, $\boldsymbol{\xi_3 = 24}$, $\boldsymbol{\xi_4 = 720}$, $\boldsymbol{\xi_5 = 1440}$, $\boldsymbol{\xi_6 = 60480}$, $\boldsymbol{\xi_7 = 120960}$, $\boldsymbol{\xi_8 = 3628800}$, $\boldsymbol{\xi_9 = 7257600}$, $\boldsymbol{\xi_{10} = 479001600}$). The order $\mathbf{p}$ of each formula is equal to $\mathbf{k+1}$. The first two of the coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{1}$ and $\mathbf{-1}$ respectively, while the remaining coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{0}$.

k	$\xi\hat{\beta}_0^{[1]}$	$\xi\hat{\beta}_1^{[1]}$	$\xi\hat{\beta}_2^{[1]}$	$\xi\hat{\beta}_3^{[1]}$	$\xi\hat{\beta}_4^{[1]}$	$\xi\hat{\beta}_5^{[1]}$	$\xi\hat{\beta}_6^{[1]}$	$\xi\hat{\beta}_7^{[1]}$	$\xi\hat{\beta}_8^{[1]}$	$\xi\hat{\beta}_9^{[1]}$	$\xi\hat{\beta}_{10}^{[1]}$	ξ
2	1	4	1									3
3	1	4	1	0								3
4	29	124	24	4	-1							90
5	28	129	14	14	-6	1						90
6	1139	5640	33	1328	-87	264	-37					3780
7	1107	5864	-639	2448	-1927	936	-261	32				3780
8	32377	182584	-42494	120088	-116120	74728	-31154	7624	-833			113400
9	31648	189145	68738	181324	-207974	166582	-92390	33868	-7394	729		113400
10	2046263	12908620	-6449433	17067984	-22652334	21705672	-15023790	7335888	-2400729	473164	-42505	7484400

Table 3.10

The products of the coefficients of the generalized Milne-Simpson **k**-step methods, where $\mathbf{2 \leq k \leq 10}$, with the multipliers $\boldsymbol{\xi}$, which are given in the last column. This means that the coefficients are the ratios of the numbers given in columns 2-11 and the corresponding multipliers $\boldsymbol{\xi}$ (the multipliers $\boldsymbol{\xi}$ located in the same rows). The order $\mathbf{p}$ of each formula is equal to $\mathbf{k+1}$. The first and the third coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{1}$ and $\mathbf{-1}$ respectively, while the remaining coefficients $\boldsymbol{\alpha}$ are equal to $\mathbf{0}$.

k	α_1	α_2	α_3	α_4	α_5	α_6	β_0
1	-1						1
2	-4/3	1/3					2/3
3	-18/11	9/11	-2/11				6/11
4	-48/25	36/25	-16/25	3/25			12/25
5	-300/137	300/137	-200/137	75/137	-12/137		60/137
6	-360/147	450/147	-400/147	225/147	-72/147	10/147	60/147

Table 3.11

The coefficients of the Backward Differentiation **k**-step methods with $\mathbf{2 \leq k \leq 6}$. The order $\mathbf{p}$ of each formula is equal to $\mathbf{k}$.

3.11. Some general conclusions related to the third chapter

It was shown in this chapter that predictor-corrector schemes can successfully be used instead of the Richardson Extrapolation in conjunction with linear multistep methods. The computational work per time-step is reduced when the predictor-corrector schemes are used, but the stability requirements could cause problems, because the absolute stability regions of the traditionally used predictor-corrector schemes are smaller than the corresponding absolute stability regions for the Explicit Runge-Kutta Methods. Moreover, it was verified that the absolute stability regions are as a rule becoming smaller when the order of accuracy of the predictor-corrector schemes is increased. Therefore, it is necessary to search for special predictor-corrector schemes with increased absolute stability regions. Some results obtained in such a search were presented, but an important question remains:

Will it be possible to improve further the results and to obtain predictor-corrector schemes which have even bigger absolute stability regions?

Such an aim could probably be achieved, but something has to be paid for it. Two approaches can be used.

The first possibility is to use more general predictor-corrector schemes of the type:

$$(3.88) \qquad y_n^{[0]} + \sum_{i=1}^{\bar{k}} \alpha_i^{[0]} y_{n-i} = h \sum_{i=1}^{k} \beta_i^{[0]} f_{n-i} ,$$

$$(3.89) \qquad y_n^{[1]} + \sum_{i=1}^{\bar{k}} \alpha_i^{[1]} y_{n-i} = h\beta_0^{[1]} f_n^{[0]} + h \sum_{i=1}^{k} \beta_i^{[1]} f_{n-i} ,$$

where $2 < \bar{k} \leq k$. The price that has to be paid is the need of extra storage, because the vectors y_{n-i}, $i = 3, 4, \ldots, \bar{k}$, must be stored and used in the computation of the approximations $y_n^{[0]}$ and $y_n^{[1]}$.

In principle, one can obtain something for the extra storage used. Assume that the coefficients $\beta_i^{[j]}$, $j = 0, 1,$ were used to achieve order of accuracy k for the predictor formula and $k+1$ for the corrector formula. Then the extra free parameters ($\alpha_i^{[j]}$, $i = 3, , 4, \ldots, \bar{k}$, $j = 0, 1$) cannot be used to increase the order of accuracy of the formulae (3.88) and (3.89), because of the first Dahlquist barrier (see § 3.1.3). They could be applied in an attempt to improve the absolute stability properties of the scheme defined by these two formulae. Such an attempt might be successful, but one should take care for the preservation of the zero-stability in the case when these formulae are used with variable time-stepsize, which has been established only for $\bar{k} \leq 2$. Therefore, it will additionally be necessary to prove that the formulae of the predictor-corrector scheme are zero-stable when these are used in a variable time-stepsize mode (at least the corrector formula must be zero-stable). This may be a very difficult task.

The second approach seems to be more promising. One can use the formulae (3.88) and (3.89) with $\bar{k} \leq 2$, but drop the requirement for achieving the maximal orders of accuracy: k for the predictor formula and $k+1$ for the corrector formula. Assume that the required order of accuracy is j for the predictor formula and $j+1$ for the corrector formula. Then there will be $k-j$ free parameters in the predictor formula and $k-j$ free parameters in the corrector formula, which can be used in the efforts to improve the stability of the computational process. Thus, the accuracy of the results will be lower, but the stability might be increased.

There will be no problems with the zero-stability in this case.

Both the first approach and the second approach rely on some compromise: it may be possible to achieve some positive result (improved stability), but for some price (one accepts to use more storage in the first case, while the accuracy requirements are relaxed in the second case). The need of finding a good compromise during the selection of a numerical algorithms is not surprising. The fact that there arise many difficulties related to the choice of efficient numerical methods was well understood from the very beginning of the development of the numerical analysis and the scientific computing. Moreover, it was also well-known that it is very difficult to find the most efficient numerical method and to justify fully its application in the solution of large-scale scientific and/or engineering problems. Finding the best possible algorithm is practically impossible in many of the complicated situations that have often to be handled in practice and this fact was also well-known from the very beginning of the development of the numerical analysis. This is why R. W. Hamming wrote in one of the first books on numerical methods, **Hamming (1962)**, that the choice of a good numerical method is in nearly all cases a question of finding a good compromise. The resolution of the stability problems in relation to predictor-corrector schemes is one of the areas where a compromise is needed in the efforts to improve the stability properties of the studied algorithms.

3.12. Topics for further research

The following topics might lead to some very interesting and useful results:

(A) Consider the second approach from the previous section. Will it be possible to design some linear multistep methods, whose order of accuracy is $\mathbf{j}$ with $\mathbf{j} < \mathbf{k}$, with increased absolute stability regions? Could one select some appropriate pairs $(\mathbf{j}, \mathbf{k})$ with good absolute stability properties?

(B) Will the approach sketched above in **(A)** be applicable also in the case where predictor-corrector schemes are used?

Chapter 4

Richardson Extrapolation for some implicit methods

The implementation of the Richardson Extrapolation in connection with several selected for our book, but also often used in many applications, implicit numerical methods for solving systems of ODEs is discussed in this chapter. Actually, representatives of the well-known θ-methods, which were already mentioned in the first chapter, and some Implicit Runge-Kutta Methods are studied. The main topic of the discussion will again be **the investigation of the stability properties** of all these methods in the case when they are combined with the Richardson Extrapolation. All details that are needed in order to implement and to use efficiently the Richardson Extrapolation for the θ-methods are fully explained in the first part of this chapter. Then, when the basic rules are well explained, the same technique is applied in connection with some other implicit numerical methods for solving systems of ODEs, more precisely in connection with Fully Implicit Runge-Kutta (FIRK) Methods and with Diagonally Implicit Runge-Kutta (DIRK) Methods.

The θ-methods will be introduced in **Section 4.1**. It will be explained there that the name "θ-method" is often used by many researchers, but it causes confusion in some situations, because in fact the "θ-method" is not a single numerical scheme, but a considerably large class of methods depending on the parameter θ, which can freely be varied.

The stability properties of most popular numerical schemes from the class of the θ-methods, which are often used by scientists and engineers, will be presented and discussed in **Section 4.2**.

The implementation of the Richardson Extrapolation in combination with the class of the θ-methods will be described in **Section 4.3.** The presentation will be very similar to that given in Section 1.3 and Section 2.3, but in this section the specific properties of the numerical schemes from the class of the θ-methods will be taken into account.

The stability properties of the resulting **new numerical methods** (which are combinations of numerical schemes from the class of the θ-methods with the Richardson Extrapolation) will be studied in **Section 4.4.** It will be shown in this section that the stability properties of the underlying θ-methods are **not** always preserved when these are combined with the Richardson Extrapolation. Some recommendations related to the choice of robust and reliable combinations of the Richardson Extrapolation with numerical schemes from the class of the θ-methods will be given.

The computational difficulties, which arise when numerical schemes belonging to the class of the θ-methods are used in the solution of stiff systems of ODEs, will be discussed in the next section, **Section 4.5**. It will be explained there that the schemes selected for solving **stiff** systems of ODEs have

necessarily to be implicit in order to ensure or, at least, to try to ensure stability of the computational process. The implicitness of the numerical methods is causing additional difficulties and some problems must be resolved when the numerical methods are handled on computers (both directly and in a combination with the Richardson Extrapolation). The problems, which arise because of the need to apply implicit numerical schemes, will be described and it will be explained how to resolve them.

Numerical results will be presented in **Section 4.6.** An atmospheric chemical scheme, which is implemented in several well-known large-scale environmental models, will be introduced and systematically used in the numerical experiments. The presented results will demonstrate clearly two very important facts:

(a) the ability of the combined numerical methods, based on the application of the Richardson Extrapolation, to preserve very well the stability of the computational process (according to the results that will be proved in Section 4.3)

and

(b) the possibility to achieve higher accuracy when the new numerical methods (consisting of combinations of selected θ-methods with the Richardson Extrapolation) are used.

The ideas used in Section 4.2 – Section 4.6, where the θ-methods are treated, will be generalized in **Section 4.7** and used there in relation to both Fully Implicit Runge-Kutta (FIRK) Methods and Diagonally Implicit Runge-Kutta (DIRK) Methods.

Several major conclusions will be given in **Section 4.8.** Some possibilities for further improvements of the results are also discussed in this section.

Some topics for future research will be sketched in the last section of this chapter, in **Section 4.9**.

4.1. Description of the class of θ-methods

The computations based on the use of the θ-methods will be carried out step by step as explained in Chapter 1. Approximations of the exact solution of the initial value problem for the systems of ODEs described by (1.1) and (1.2) are calculated at the grid-points $\{t_0, t_1, \ldots, t_{n-1}, t_n, \ldots, t_N\}$ of (1.6). Two relationships hold for all indices n from the set $\{1, 2, \ldots, N\}$:

(a) $t_n = t_{n-1} + h$ (where the time-stepsize h is some fixed positive number)

and

(b) $y_n \approx y(t_n)$.

This means that an equidistant grid will be mainly used in this chapter, but this is done only in order to facilitate both the presentation and the understanding of the results. Most of the conclusions will remain

valid also when variations of the time-stepsize are allowed and used during the numerical solution of the system of ODEs. In the previous chapter it was verified that the variation of the time-stepsize is causing some technical difficulties when linear multistep methods are used and, therefore, it was necessary to develop and implement, at every time-step, some special devices at every time-step in order to be able to calculate the coefficients of the chosen method and to select the optimal time-stepsize. It should be emphasized here that the variation of the time-stepsize is in general not causing additional technical difficulties when either the θ-methods or other one-step methods are used.

In this section and in several of the next sections of Chapter 4, the following formula is used (with some particular value of the parameter $\boldsymbol{\theta}$) in the computational process:

$$(\mathbf{4.1}) \quad \mathbf{y_n = y_{n-1} + h[(1-\theta)f(t_{n-1}, y_{n-1}) + \theta\, f(t_n, y_n)]} \qquad \text{for} \qquad \mathbf{n = 1, 2, \dots, N}.$$

As mentioned above, the algorithm defined by the above formula is nearly always or at least very often called **the θ-method**. However, formula (4.1) shows very clearly that the θ-method is in fact a large class of numerical methods, which depend on the particular parameter $\boldsymbol{\theta}$. We shall sometimes use the traditionally quoted in the literature name "θ-method" both when we are describing some special numerical schemes from this class and when we are discussing properties, which are valid for the whole class. From the context it will be quite clear in what sense the term "θ-method" is used.

The class of the θ-methods is normally used with $\boldsymbol{\theta \in [0, 1]}$. The numerical methods, which are obtained for $\boldsymbol{\theta = 0}$, $\boldsymbol{\theta = 0.5}$ and $\boldsymbol{\theta = 1}$, are very popular among scientists and engineers and are very often used by them in practical computations. It will be shown that also the method, which is obtained when $\boldsymbol{\theta = 0.75}$ is specified, has some nice properties. Therefore, this representative of the class of θ-methods will also be used in this chapter.

The Forward Euler Formula (which is also well-known as the Explicit Euler Method) is obtained for $\boldsymbol{\theta = 0}$:

$$(\mathbf{4.2}) \quad \mathbf{y_n = y_{n-1} + h\, f(t_{n-1}, y_{n-1})} \qquad \mathbf{for} \qquad \mathbf{n = 1, 2, \dots, N}.$$

This numerical scheme is a first-order one-stage Explicit Runge-Kutta Method. It can also be considered as a representative of the linear $\mathbf{k}$-step methods studied in the previous chapter, which is obtained by the special and in fact the simplest choice $\mathbf{k = 1}$. The Forward Euler Formula (4.2) had already been used in the discussion both in Chapter 2 and in Chapter 3. It will not be further discussed in this chapter.

The well-known Trapezoidal Rule is obtained for $\boldsymbol{\theta = 0.5}$:

$$(\mathbf{4.3}) \quad \mathbf{y_n = y_{n-1} + 0.5\, h\, [f(t_{n-1}, y_{n-1}) + f(t_n, y_n)]} \qquad \text{for} \qquad \mathbf{n = 1, 2, \dots, N}.$$

This numerical algorithm was mentioned in one numerical example, which was presented in Chapter 1, but it will be discussed further and in some more detail in the next sections of this chapter. The order of accuracy of the Trapezoidal Rule is two.

The Backward Euler Formula (known also as the Implicit Euler Method) is obtained from (4.1) for $\boldsymbol{\theta = 1}$:

$$(\mathbf{4.4}) \quad \mathbf{y_n = y_{n-1} + h\, f(t_n, y_n)} \quad \text{for} \quad \mathbf{n = 1, 2, \ldots, N}.$$

The order of accuracy of the Backward Euler Formula is only one, but it has very good stability properties.

As we pointed out above, the Forward Euler Method is explicit, while both the Trapezoidal Rule and the Backward Euler Formula are implicit numerical schemes, because the unknown vector $\mathbf{y_n}$ participates both in the left-hand-side and in the right-hand-side of (4.3) and (4.4). In fact, as was mentioned in the beginning of this chapter, the only explicit numerical scheme from the class of the $\boldsymbol{\theta}$-methods defined by (4.1) is the Forward Euler Method.

4.2. Stability properties of the θ-methods

It is both relatively easy and very convenient to study, as was done in the previous chapters, the stability properties of the $\boldsymbol{\theta}$-method by the application of the famous **scalar and linear** test-problem proposed in **Dahlquist (1963)**:

$$(\mathbf{4.5}) \quad \frac{\mathbf{dy}}{\mathbf{dt}} = \boldsymbol{\lambda}\, \mathbf{y}, \quad \mathbf{t} \in [\mathbf{0}, \infty], \quad \mathbf{y} \in \mathbb{C}, \quad \boldsymbol{\lambda} = \bar{\boldsymbol{\alpha}} + \bar{\boldsymbol{\beta}}\mathbf{i} \in \mathbb{C}, \quad \bar{\boldsymbol{\alpha}} \leq \mathbf{0}, \quad \mathbf{y(0) = \eta}.$$

The exact solution of equation (4.5) is given by

$$(\mathbf{4.6}) \quad \mathbf{y(t) = \eta\, e^{\lambda t}}, \quad \mathbf{t} \in [\mathbf{0}, \infty].$$

It should be mentioned also here that the exact solution $\mathbf{y(t)}$ of the Dahlquist test-equation (4.5) is a bounded function, because the assumption $\bar{\boldsymbol{\alpha}} \leq \mathbf{0}$ was made in (4.5).

The application of the numerical algorithms that are defined by (4.1) in the solution of the special scalar test-problem (4.5) leads to a relationship, which is of the same form as that derived in Chapter 2:

$$(\mathbf{4.7}) \quad \mathbf{y_n = R(\nu)\, y_{n-1} = [R(\nu)]^n\, y_0}, \quad \boldsymbol{\nu} = \mathbf{h\, \lambda}, \quad \mathbf{n = 1, 2, \ldots}$$

However, if $\boldsymbol{\theta \neq 0}$, then the stability function $\mathbf{R(\nu)}$, is not a polynomial as in Chapter 2, but a ratio of two first-degree polynomials. This ratio can be represented by the following formula:

$$(\mathbf{4.8}) \quad \mathbf{R(\nu)} = \frac{\mathbf{1 + (1 - \theta)\nu}}{\mathbf{1 - \theta\nu}}.$$

It is immediately seen that if $\boldsymbol{\theta = 0}$, i.e. when the Forward Euler Method is used, then the stability function is indeed reduced to the first-degree polynomial $\mathbf{R(\nu) = 1 + \nu}$ and, as mentioned above, this case was studied in Chapter 2, see (2.15) in §2.4.1.

In this chapter, we shall be interested only in the case $\boldsymbol{\theta \neq 0}$. $\mathbf{R(\nu)}$ is always a rational function and more precisely a ratio of two polynomials for such a choice of parameter $\boldsymbol{\theta}$. In fact, numerical methods, which have good stability properties are obtained when $\boldsymbol{\theta \in [0.5, 1.0]}$ and it will be assumed in the remaining part of this chapter that $\boldsymbol{\theta}$ is in this interval.

As in Chapter 2, we can conclude that the numerical solution of (4.6), which is calculated by using some numerical scheme from the class of the $\boldsymbol{\theta}$-methods with a given value of the time-stepsize $\mathbf{h}$ and for some particular coefficient $\boldsymbol{\lambda}$, will be bounded when the stability requirement $\mathbf{R(\nu) \leq 1}$ is satisfied.

In Chapter 2, we were interested in solving the problem (4.5) in the case where the parameter $\boldsymbol{\lambda}$ was **not** very large in absolute value. When this assumption is made, i.e. when $\boldsymbol{|\lambda|}$ is not very large, then the problem defined by (4.5) is non-stiff or, at least, only moderately stiff and it can be treated numerically with a **reasonably large** time-stepsize in spite of the fact that the absolute stability region of the selected numerical scheme is **finite** (as were all absolute stability regions that were presented in Chapter 2; see **Fig. 2.1 – Fig. 2.4, Fig. 2.8** and **Fig. 2.9**).

Now we shall be interested in the case where $\boldsymbol{|\lambda|}$ is **very large** (in which case the problem will normally become stiff). If this is the case, i.e. if $\boldsymbol{|\lambda|}$ is really very large, then it is highly desirable to be able to use a sufficiently large time-stepsize in the numerical solution of the (4.5). This is even more desirable and in fact it is nearly always necessary when the problem defined by (1.1) and (1.2) is very large and when the Jacobian matrix of function $\mathbf{f}$ has eigenvalues with non-positive real parts, which are very large in absolute value.

The requirement of using a large time-stepsize in the solution of (4.5) is indeed very restrictive, when at the same time parameter $\boldsymbol{|\lambda|}$ is very large. This is why **it is often not sufficient** in this situation to search (as we did in Chapter 2) for large but **finite** absolute stability regions that contain all points of $\boldsymbol{\nu = \alpha + \beta i}$ with $\boldsymbol{\alpha \leq 0}$ for which $\mathbf{R(\nu) \leq 1}$. Instead of this it much more is reasonable to require that

$$(\mathbf{4.9}) \quad \mathbf{R(\nu) \leq 1} \quad \text{for} \quad \forall\, \boldsymbol{\nu = \alpha + \beta i} \quad \text{with} \quad \boldsymbol{\alpha \leq 0}.$$

In other words, we shall now demand that the crucial inequality $\mathbf{R(\nu) \leq 1}$ is satisfied everywhere in the negative part of the complex plane and that the absolute stability regions of the numerical methods are **infinite** (containing the whole negative part of the complex plane). This is a very strong requirement. It can be proved that the assumption made in (4.9) can be satisfied **only** when a requirement for applying some **implicit** numerical method is additionally imposed. This extra requirement, the requirement to use some implicit numerical method for solving systems of ODEs, is, as mentioned in the previous chapters, a part of a theorem proved in **Dahlquist (1963)**, which is often called **the second Dahlquist barrier (**see, for example, pp. 243-244 in **Lambert, 1991**).

By applying the sketched above discussion, which led us to the necessity to impose the strong requirement (4.9) and to the conclusion that implicit methods for solving systems of ODEs must be used, the following definition, proposed originally by G. Dahlquist, can be given.

Definition 4.1: It is said that the numerical method for solving systems of ODEs is **A-stable** when the relationship $\mathbf{R(\nu) \leq 1}$ is fulfilled for $\mathbf{\forall \nu = \alpha + \beta i}$ with $\mathbf{\alpha \leq 0}$ in the case where the numerical method is applied in the solution of the Dahlquist scalar and linear test-example (4.5).

■

Because of the second Dahlquist barrier, it is clear that **every A-stable numerical method is necessarily implicit.** The numerical treatment of systems of ODEs by implicit numerical methods is much more difficult than the numerical treatment of such systems by explicit numerical methods (this topic will be further discussed in Section 4.5).

It can be proved that the θ-method is A-stable when $\mathbf{\theta \in [0.5, 1.0]}$, see, for example **Hairer and Wanner (1991)**. Because of this fact, in this chapter we shall, as stated above, consider numerical schemes from the class of the θ-methods with θ varying in this interval.

We defined the concept of A-stability in connection with the simple scalar and linear equation (4.5). However, the results can be generalized for some linear systems of ODEs with constant matrices. Moreover, there are some reasons to expect that the results will hold also for some more general, linear and non-linear, systems of ODEs. These issues have been presented and discussed in Chapter 2 (see Section 2.1) and there is no need to repeat the explanations here.

The requirement for A-stability is, as we pointed out above, very restrictive. Unfortunately, in some situations even this requirement is not sufficient in the efforts to achieve an efficient computational process. This can be explained as follows. Consider the Trapezoidal Rule (4.3). By using (4.7) and (4.8) with $\mathbf{\theta = 0.5}$ the following relationship can be obtained:

$$\mathbf{(4.10)} \quad y_n = \frac{1 + 0.5\nu}{1 - 0.5\nu} y_{n-1} = \left(\frac{1 + 0.5\nu}{1 - 0.5\nu}\right)^n y_0 .$$

Assume further that

(a) $\boldsymbol{\lambda}$ is a very large in absolute value **negative** number,

(b) $\mathbf{h}$ is again some fixed positive increment ($\mathbf{h\lambda = \nu}$ being satisfied)

and

(c) $\mathbf{y_0 = 1}$ is the initial value of the scalar test-problem (4.5).

Then the exact solution $\mathbf{y(t)}$ of (4.5) will very quickly tend to zero. However, if the assumptions **(a)**, **(b)** and **(c)** are satisfied, then the last term in (4.10) will tend **quickly** to zero **only when the time-stepsize $\mathbf{h}$ is very small**, which is clearly not desirable when large-scale scientific and engineering models are to be handled numerically (because in such a case many time-steps are to be performed and, therefore, the computational process will become very expensive). If the assumption for a very small time-stepsize is not satisfied, then the term in the parenthesis in (4.10) will be still smaller than one, but very close to one. Therefore, it is obvious that the convergence of the numerical solution to zero will be very slow. Moreover, note that if three conditions **(a)**, **(b)** and **(c)** hold and if $\mathbf{h}$ is fixed, but $|\boldsymbol{\lambda}| \rightarrow \infty$, then $|(\mathbf{1 + 0.5\nu})/(\mathbf{1 - 0.5\nu})| \rightarrow \mathbf{1}$.

This example shows very clearly that in some cases the use of the Trapezoidal Rule will not lead to an efficient and accurate computational process under the assumptions made above in spite of the fact that this numerical method is A-stable.

The situation changes completely when the Backward Euler Formula is used. Indeed, formula (4.8) could be rewritten for $\boldsymbol{\theta = 1}$ as

$$(\mathbf{4.11}) \quad \mathbf{y_n} = \frac{\mathbf{1}}{\mathbf{1 - 0.5\nu}}\,\mathbf{y_{n-1}} = \left(\frac{\mathbf{1}}{\mathbf{1 - 0.5\nu}}\right)^{\mathbf{n}}\mathbf{y_0}$$

It is clear now, i.e. when $\boldsymbol{\theta = 1,}$ that $|\mathbf{y_n}|$ will quickly tend to zero when $\mathbf{n} \rightarrow \infty$ even for rather large values of the time-stepsize $\mathbf{h}$ and, furthermore, also in the case where the above three conditions **(a)** – **(c)** are satisfied. It is also clear, assuming once again that the assumptions **(a)** – **(c)** are satisfied, that if $\mathbf{h}$ is arbitrary large but fixed and if $|\boldsymbol{\lambda}| \rightarrow \infty$, then $|\mathbf{1}/(\mathbf{1 - 0.5\nu})| \rightarrow \mathbf{0}$, which in most of the cases will be quite satisfactory.

The two examples that are presented above, by deriving and applying the two formulae (4.10) and (4.11), justify the necessity to introduce a new and more restrictive stability definition, the definition for **L-stability**.

Definition 4.2: A numerical method for solving systems of ODEs is said to be **L-stable** when it is A-stable and, in addition, when it is applied in the solution to the scalar test-problem (4.5), it leads to the relationship (4.7) with $|\mathbf{R(\nu)}| \rightarrow \mathbf{0}$ as $\mathbf{Re(\nu)} \rightarrow -\infty$.

■

The real part of the complex number ν is denoted in Definition 4.2 as usual by $\mathbf{Re}(\nu)$ and it is perhaps worthwhile to reiterate here that $\nu = \alpha + \beta i$ with $\alpha \leq 0$, see (4.9). This means that $\mathbf{Re}(\nu) = \alpha$ is a non-positive number.

Sometimes it is very useful to relax a little the requirement for L-stability, by introducing the concept of **strong A-stability**.

Definition 4.3: A numerical method for solving systems of ODEs is said to be **strongly A-stable** when it is A-stable and, in addition, if it is applied to the Dahlquist scalar test-problem (4.5), then it leads to the relationship (4.7) with $|\mathbf{R}(\nu)| \rightarrow c < 1$ as $\mathbf{Re}(\nu) \rightarrow -\infty$.

■

It is obvious that the definition for strong A-stability is a compromise between the weaker definition for A-stability and the stronger definition for L-stability (compare Definition 4.3 with Definition 4.1 and Definition 4.2). It will be shown at the end of this chapter that for **some non-linear systems of ODEs strongly A-stable methods may perform even better than L-stable methods.**

As stated above, the Trapezoidal Rule ($\theta = 0.5$) is only A-stable. If $\theta \in (0.5, 1.0)$, then the numerical method (4.1) is strongly A-stable. The Backward Euler Formula ($\theta = 1.0$) is L-stable. See more details related to the different concepts of stability in, for example, **Hairer and Wanner, 1991**, **Hundsdorfer and Verwer, 2003** or **Lambert, 1991**).

We are ready now first to introduce the Richardson Extrapolation for the class of the θ-methods and after that to give an answer to the important question:

Are the stability properties of all new methods (the combinations of the θ-methods with the Richardson Extrapolation) preserved?

4.3. Combining the θ-method with the Richardson Extrapolation

The Richardson Extrapolation can easily be introduced for the class of the θ-methods by following closely the rules, which were applied and explained in Section 1.3 (see also Section 2.3). Since we are very interested in the preservation of the stability properties for the obtained in this way new numerical methods, we shall explain the application of the Richardson Extrapolation directly for the case where the Dahlquist scalar and linear test-problem (4.5) is solved (because precisely these formulae will be needed in the study of the stability properties of the resulting **new numerical methods**; the combinations of the Richardson Extrapolation with representatives of the class of the θ-methods).

Assume that $\mathbf{t_{n-1}}$ and $\mathbf{t_n}$ are any two grid-points of the set (1.6) and that a sufficiently accurate approximation $\mathbf{y_{n-1}}$ has already been calculated. Three computational steps should successively be carried out by using (4.7) and (4.8) in order to calculate an improved by the Richardson Extrapolation vector $\mathbf{y_n}$.

Step 1: Perform one large time-step with a time-stepsize $\mathbf{h}$ to calculate an approximation $\mathbf{z_n}$ of the exact solution $\mathbf{y(t_n)}$:

$$(\mathbf{4.12}) \qquad \mathbf{z_n} = \frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})\boldsymbol{\nu}}{\mathbf{1} - \boldsymbol{\theta}\boldsymbol{\nu}}\, \mathbf{y_{n-1}}\,.$$

Step 2: Perform two small time-steps with a time-stepsize $\mathbf{0.5\,h}$ to calculate another approximation $\mathbf{w_n}$ of the exact solution $\mathbf{y(t_n)}$:

$$(\mathbf{4.13}) \qquad \mathbf{w_n} = \left[\frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})(\mathbf{0.5}\,\boldsymbol{\nu})}{\mathbf{1} - \boldsymbol{\theta}(\mathbf{0.5}\,\boldsymbol{\nu})}\right]^{\mathbf{2}} \mathbf{y_{n-1}}\,.$$

Step 3: Use $\mathbf{z_n}$ and $\mathbf{w_n}$ to calculate an improved approximation $\mathbf{y_n}$ of the exact solution $\mathbf{y(t_n)}$ according to the following two rules:

$$(\mathbf{4.14}) \qquad \mathbf{y_n} = \mathbf{2}\,\mathbf{w_n} - \mathbf{z_n} \qquad \text{when} \qquad \boldsymbol{\theta} \neq \mathbf{0.5}$$

and

$$(\mathbf{4.15}) \qquad \mathbf{y_n} = \frac{\mathbf{4w_n} - \mathbf{z_n}}{\mathbf{3}} \qquad \text{when} \qquad \boldsymbol{\theta} = \mathbf{0.5}\,.$$

Note that the fact that the $\boldsymbol{\theta}$-method is of first-order of accuracy when $\boldsymbol{\theta} \neq \mathbf{0.5}$ is used in the derivation of (4.14), while the fact that the Trapezoidal Rule, which is obtained when $\boldsymbol{\theta} = \mathbf{0.5}$, is a second-order numerical method, is exploited when (4.15) is obtained. Therefore, it is clearly seen that formulae (4.14) and (4.15) are obtained by using (1.8) with $\mathbf{p} = \mathbf{1}$ and $\mathbf{p} = \mathbf{2}$ respectively.

Note too that it is assumed that **the active implementation** of the Richardson Extrapolation (see Section 1.7) is used in the formulation of the above algorithm. However, this is not a restriction, because the derivation of the passive implementation of the Richardson Extrapolation in connection

with the θ-methods is quite similar: it will only be necessary in such a case to use $\mathbf{z_{n-1}}$ in (4.12) and $\mathbf{w_{n-1}}$ in (4.13) instead of $\mathbf{y_{n-1}}$.

The following two relationships can be obtained by inserting the expressions for $\mathbf{z_n}$ and $\mathbf{w_n}$ from (4.12) and (4.13) in (4.14) and (4.15) respectively:

$$(\mathbf{4.16}) \quad \mathbf{y_n} = \left\{ \mathbf{2} \left[\frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})(\mathbf{0.5}\,\boldsymbol{\nu})}{\mathbf{1} - \boldsymbol{\theta}(\mathbf{0.5}\,\boldsymbol{\nu})} \right]^{\mathbf{2}} - \frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})\boldsymbol{\nu}}{\mathbf{1} - \boldsymbol{\theta}\boldsymbol{\nu}} \right\} \mathbf{y_{n-1}} \qquad \text{when} \qquad \boldsymbol{\theta} \neq \mathbf{0.5}$$

and

$$(\mathbf{4.17}) \quad \mathbf{y_n} = \frac{\mathbf{4} \left[\frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})(\mathbf{0.5}\,\boldsymbol{\nu})}{\mathbf{1} - \boldsymbol{\theta}(\mathbf{0.5}\,\boldsymbol{\nu})} \right]^{\mathbf{2}} - \frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})\boldsymbol{\nu}}{\mathbf{1} - \boldsymbol{\theta}\boldsymbol{\nu}}}{\mathbf{3}} \mathbf{y_{n-1}} \qquad \text{when} \qquad \boldsymbol{\theta} = \mathbf{0.5}\,.$$

It is immediately seen from (4.16) and (4.17) that the combinations of the Richardson Extrapolation with θ-methods are one-step methods (i.e. only the approximation $\mathbf{y_{n-1}}$ is used in the calculation of the improved value $\mathbf{y_n}$), the stability functions of which are given by the following two expressions:

$$(\mathbf{4.18}) \quad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \mathbf{2} \left[\frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})(\mathbf{0.5}\,\boldsymbol{\nu})}{\mathbf{1} - \boldsymbol{\theta}(\mathbf{0.5}\,\boldsymbol{\nu})} \right]^{\mathbf{2}} - \frac{\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})\boldsymbol{\nu}}{\mathbf{1} - \boldsymbol{\theta}\boldsymbol{\nu}} \qquad \text{when} \qquad \boldsymbol{\theta} \neq \mathbf{0.5}$$

and

$$(\mathbf{4.19}) \quad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{4} \left[\frac{\mathbf{1} + \mathbf{0.25}\,\boldsymbol{\nu}}{\mathbf{1} - \mathbf{0.25}\,\boldsymbol{\nu}} \right]^{\mathbf{2}} - \frac{\mathbf{1} + \mathbf{0.5}\boldsymbol{\nu}}{\mathbf{1} - \mathbf{0.5}\boldsymbol{\nu}}}{\mathbf{3}} \qquad \text{when} \qquad \boldsymbol{\theta} = \mathbf{0.5}\,.$$

The stability properties of **the new numerical methods** that are combinations of the Richardson Extrapolation with θ-methods will be studied in the next section.

4.4. Stability of the Richardson Extrapolation combined with θ-methods

It is necessary to investigate when the application of the Richardson Extrapolation together with different θ-methods preserves the stability properties of the underlying methods and when this is not the case. We shall show in this section that one should be careful, because stability problems may

sometimes arise. More precisely, the following theorem holds; see also **Zlatev, Faragó and Havasi (2010)**:

Theorem 4.1: The new numerical method consisting of a combination of the active implementation of the Richardson Extrapolation with any numerical scheme belonging to the class of the θ-methods is **strongly A-stable** when $\theta \in (\theta_0, 1]$ with $\theta_0 = 2/3$.

Proof: According to Definition 4.3 that was given in Section 4.2, a strongly A-stable numerical method **must** in addition be A-stable (see also, for example, **Hundsdorfer and Verwer, 2003**). In **Hairer and Wanner (1991)** it is shown that a numerical method for solving systems of ODEs is A-stable if and only if

(a) it is stable on the imaginary axis (i.e. when $|\bar{R}(i\beta)| \leq 1$ holds for **all** real values of β)

and

(b) $\bar{R}(\nu)$ is analytic in $\mathbb{C}^-$.

If we show that the two requirements **(a)** and **(b)** hold (i.e. if we show that the considered numerical method is A-stable), then it will be necessary to show additionally that the new numerical method is also strongly A-stable, i.e. we must prove that, according to Definition 4.3, the following basic relationship $|\bar{R}(\nu)| \to c < 1$ as $Re(\nu) \to -\infty$ should be additionally satisfied.

The above analysis indicates that Theorem 4.1 can be proved in three consecutively performed steps:

Step A: Prove that the combination of the Richardson Extrapolation with the θ-methods is stable on the imaginary axis.

Step B: Show that the stability function $\bar{R}(\nu)$ is analytic.

Step C: Prove that $|\bar{R}(\nu)| \to c < 1$ as $Re(\nu) \to -\infty$.

We shall start with Step A.

Step A – Stability on the imaginary axis

It is immediately seen that the stability function $\bar{R}(\nu)$ from (4.18) can be written in the following form:

$$(4.20) \qquad \bar{R}(\nu) = \frac{P(\nu)}{Q(\nu)} ,$$

where $P(\nu)$ is the following polynomial:

$$(4.21) \quad P(\nu) = 2\,[1 + (1-\theta)(0.5\nu)]^2\,(1-\theta\nu) - [1 + (1-\theta)\nu]\,[1-\theta(0.5\nu)]^2 .$$

After some rather long but straight-forward transformations, (4.21) can be rewritten as a third-degree (in ν) polynomial, whose coefficients depend on the particular choice of parameter θ:

$$(4.22) \quad P(\nu) = (-0.25\theta^3 + 0.75\theta^2 - 0.5\theta)\nu^3 + (1.25\theta^2 - 2\theta + 0.5)\nu^2 + (-2\theta + 1)\nu + 1.$$

The polynomial $Q(\nu)$ from (4.20) is represented by the following expression:

$$(4.23) \quad Q(\nu) = [1-\theta(0.5\nu)]^2\,(1-\theta\nu) .$$

Also this polynomial can be rewritten as a third-degree (in ν) polynomial, whose coefficients depend on parameter θ, however it will be much more convenient to use directly (4.23) in the further computations.

Now we shall use a result, proved in **Hairer and Wanner (1991)**, stating that the stability of a numerical method on the imaginary axis is ensured if for **all** (real) values of β from the relationship $\nu = \alpha + i\beta$ the inequality

$$(4.24) \quad E(\beta) \geq 0$$

holds.

It is easy to verify that $E(\beta)$ is a polynomial, which is defined by

$$(4.25) \quad E(\beta) = Q(i\beta)\,Q(-i\beta) - P(i\beta)\,P(-i\beta) .$$

Consider the first term in the right-hand-side of (4.25). By performing the following successive transformations it can be shown that this term is a sixth-degree polynomial containing only even degrees of β:

$$(4.26) \quad Q(i\beta)\,Q(-i\beta) = [1-\theta(0.5i\beta)]^2\,(1-\theta i\beta)[1+\theta(0.5i\beta)]^2\,(1+\theta i\beta)$$
$$= [(1-0.5\theta i\beta)(1+0.5\theta i\beta)]^2(1-\theta i\beta)(1+\theta i\beta)$$

$$= (\mathbf{1} + \mathbf{0.25\theta^2\beta^2})^2\,(\mathbf{1} + \mathbf{\theta^2\beta^2})$$

$$= (\mathbf{0.0625\theta^4\beta^4} + \mathbf{0.5\theta^2\beta^2} + \mathbf{1})(\mathbf{1} + \mathbf{\theta^2\beta^2})$$

$$= \mathbf{0.0625\theta^6\beta^6} + \mathbf{0.5625\theta^4\beta^4} + \mathbf{1.5\theta^2\beta^2} + \mathbf{1}$$

$$= \frac{\mathbf{1}}{\mathbf{2^4}}\,(\mathbf{\theta^6\beta^6} + \mathbf{9\beta^4} + \mathbf{24\theta^2\beta^2} + \mathbf{16})\,.$$

Similar transformations are to be carried out in order to represent also the second term in (4.25), the term $\mathbf{P(i\beta)\,P(-i\beta)}$, as a sixth-degree polynomial containing only even degrees of $\boldsymbol{\beta}$. Introduce first the following three constants:

$$(\mathbf{4.27}) \qquad \mathbf{A} = -\mathbf{0.25\,\theta^3} + \mathbf{0.75\,\theta^2} - \mathbf{0.5\,\theta}\,, \qquad \mathbf{B} = \mathbf{1.25\,\theta^2} - \mathbf{2\,\theta} + \mathbf{0.5}\,, \qquad \mathbf{C} = -\mathbf{2\,\theta} + \mathbf{1}\,.$$

Now the second term in the right-hand-side of (4.25) can be rewritten, by several successively performed transformations, in the following form:

$$(\mathbf{4.28}) \qquad \mathbf{P(i\beta)\,P(-i\beta)} = [\mathbf{A(i\beta)^3} + \mathbf{B(i\beta)^2} + \mathbf{C(i\beta)} + \mathbf{1}][\mathbf{A(-i\beta)^3} + \mathbf{B(-i\beta)^2} + \mathbf{C(-i\beta)} + \mathbf{1}]$$

$$= (-\mathbf{Ai\beta^3} - \mathbf{B\beta^2} + \mathbf{Ci\beta} + \mathbf{1})\,(\mathbf{Ai\beta^3} - \mathbf{B\beta^2} - \mathbf{Ci\beta} + \mathbf{1})$$

$$= \mathbf{A^2\beta^6} + \mathbf{ABi\beta^5} - \mathbf{AC\beta^4} - \mathbf{Ai\beta^3}$$

$$- \mathbf{ABi\beta^5} + \mathbf{B^2\beta^4} + \mathbf{BCi\beta^3} - \mathbf{B\beta^2}$$

$$- \mathbf{AC\,\beta^4} - \mathbf{BCi\beta^3} + \mathbf{C^2\beta^2} + \mathbf{Ci\beta}$$

$$+ \mathbf{Ai\beta^3} - \mathbf{B\beta^2} - \mathbf{Ci\beta} + \mathbf{1}$$

$$= \mathbf{A^2\beta^6} - \mathbf{2AC\beta^4} + \mathbf{B^2\beta^4} - \mathbf{2B\beta^2} + \mathbf{C^2\beta^2} + \mathbf{1}$$

$$= \mathbf{A^2\beta^6} + (\mathbf{B^2} - \mathbf{2AC})\mathbf{\beta^4} + (\mathbf{C^2} - \mathbf{2B})\mathbf{\beta^2} + \mathbf{1}\,.$$

By using the expressions for $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ from (4.27), the last equality can be rewritten in the following way:

$$(\mathbf{4.29}) \qquad \mathbf{P(i\beta)\,P(-i\beta)} = \quad (-\mathbf{0.25\theta^3} + \mathbf{0.75\theta^2} - \mathbf{0.5\theta}\,)^2\,\mathbf{\beta^6}$$

$$+ [\,(\mathbf{1.25\theta^2 - 2\theta + 0.5})^2 \; - \mathbf{2}\,(\mathbf{-0.25\theta^3 + 0.75\theta^2 - 0.5\theta})(\mathbf{-2\theta + 1}\,)\,]\boldsymbol{\beta^4}$$

$$+[(\mathbf{-2\theta + 1}\,)^2 - \mathbf{2}\,(\mathbf{1.25\theta^2 - 2\theta + 0.5})]\boldsymbol{\beta^2}$$

$$+ \mathbf{1}$$

$$= \quad \frac{\mathbf{1}}{\mathbf{2^4}}\left(\boldsymbol{\theta^6 + 9\theta^4 + 4\theta^2 - 6\theta^5 + 4\theta^4 - 12\theta^3}\right)\boldsymbol{\beta^6}$$

$$+ \left[\frac{\mathbf{1}}{\mathbf{2^4}}\,(\mathbf{25\theta^4 - 80\theta^3 + 84\theta^2 - 32\theta + 4}) - \boldsymbol{\theta^4} + \mathbf{3.5\theta^3 - 3.5\theta^2 + \theta}\right]\boldsymbol{\beta^4}$$

$$+ \;(\mathbf{4\theta^2 - 4\theta + 1 - 2.5\theta^2 + 4\theta - 1})\;\boldsymbol{\beta^2}$$

$$+\; \mathbf{1}$$

$$= \quad \frac{\mathbf{1}}{\mathbf{2^4}}\left(\boldsymbol{\theta^6 - 6\theta^5 + 13\theta^4 - 12\theta^3 + 4\theta^2}\right)\boldsymbol{\beta^6}$$

$$+\frac{\mathbf{1}}{\mathbf{2^4}}\,(\mathbf{9\theta^4 - 24\theta^3 + 28\theta^2 - 16\theta + 4})\,\boldsymbol{\beta^4}$$

$$+\; \mathbf{1,5\theta^2}\,\boldsymbol{\beta^2}$$

$$+\; \mathbf{1}\;.$$

Everything is prepared now for the determination of the sign of the polynomial $\mathbf{E}(\boldsymbol{\beta})$ from (4.25). It is necessary to substitute the last terms in the right-hand-sides of (4.26) and (4.29) in (4.25). The result is

$$(\mathbf{4.30}) \qquad \mathbf{E}(\boldsymbol{\beta}) = \quad \frac{\mathbf{1}}{\mathbf{2^4}}\,(\boldsymbol{\theta^6\beta^6} + \mathbf{9}\boldsymbol{\beta^4} + \mathbf{24}\boldsymbol{\theta^2\beta^2} + \mathbf{16})$$

$$-\frac{\mathbf{1}}{\mathbf{2^4}}\left(\boldsymbol{\theta^6 - 6\theta^5 + 13\theta^4 - 12\theta^3 + 4\theta^2}\right)\boldsymbol{\beta^6}$$

$$-\frac{\mathbf{1}}{\mathbf{2^4}}\,(\mathbf{9\theta^4 - 24\theta^3 + 28\theta^2 - 16\theta + 4})\,\boldsymbol{\beta^4}$$

$$-\frac{\mathbf{1}}{\mathbf{2^4}}\;\mathbf{24}\,\boldsymbol{\theta^2}\,\boldsymbol{\beta^2}$$

$$-\frac{\mathbf{1}}{\mathbf{2^4}}\;\mathbf{16}$$

$$= \quad \frac{1}{2^4}\left(6\theta^5 - 13\theta^4 + 12\theta^3 - 4\theta^2\right)\beta^6 + \frac{1}{2^2}\left(6\theta^3 - 7\theta^2 + 4\theta - 1\right)\beta^4 .$$

It is easily seen that

$$(4.31) \quad \mathbf{E}(\beta) \geq 0 \quad \Longrightarrow \quad \left(6\theta^5 - 13\theta^4 + 12\theta^3 - 4\theta^2\right)\beta^2 + 4(6\theta^3 - 7\theta^2 + 4\theta - 1) \geq 0 .$$

Let us introduce the following two polynomials:

$$(4.32) \quad \mathbf{H_1}(\theta) = 6\theta^3 - 13\theta^2 + 12\theta - 4 \quad \text{and} \quad \mathbf{H_2}(\theta) = 6\theta^3 - 7\theta^2 + 4\theta - 1 .$$

It follows from (4.30) and (4.31) that $\mathbf{E}(\beta)$ will be non-negative for **all** values of β and for a given value of θ if and only if both polynomials from (4.32) are non-negative for the selected value of θ. It can easily be shown that the inequalities

$$(4.33) \quad \frac{d\mathbf{H_1}}{d\theta} > 0 \quad \text{and} \quad \frac{d\mathbf{H_2}}{d\theta} > 0$$

hold when $\theta \in [0.5, 1.0]$, which implies that the two polynomials $\mathbf{H_1}(\theta)$ and $\mathbf{H_2}(\theta)$ are increasing in this interval. Since $\mathbf{H_1}(2/3) = 0$ and $\mathbf{H_2}(2/3) > 0$, the two polynomials are clearly non-negative for $\theta \in [2/3, 1.0]$ and, therefore, $\mathbf{E}(\beta)$ will certainly be non-negative for all values of θ in the interval $[\theta_0, 1.0]$, where $\theta_0 = 2/3$ is the unique zero of the polynomial $\mathbf{H_1}(\theta)$ in the interval $[0.5, 1.0]$.

This completes the proof of the first step of Theorem 4.1, because we have shown that the combinations of the Richardson Extrapolation with numerical schemes from the class of the θ-methods are stable on the imaginary axis when $\theta \in [2/3, 1.0]$.

Before starting the proof of the second step of the theorem, it is worthwhile to point out that the fact that the two polynomials $\mathbf{H_1}(\theta)$ and $\mathbf{H_2}(\theta)$ are non-negative for $\theta \in [2/3, 1.0]$ is demonstrated graphically in **Fig. 4.1**.

Step B – A-stability

After the proof that the combination of the Richardson Extrapolation with the θ-method is stable on the imaginary axis when $\theta \in [2/3, 1.0]$, it should also be proved that the stability function $\mathbf{R}(\nu)$ is analytic in $\mathbb{C}^-$ for these values of θ. The stability function is, according to equality (4.20), a ratio of the two polynomials $\mathbf{P}(\nu)$ and $\mathbf{Q}(\nu)$. It is well-known that polynomials are analytic functions and that a ratio of two polynomials is an analytic function in $\mathbb{C}^-$ if the denominator has no roots in $\mathbb{C}^-$. In our case, the roots of the denominator $\mathbf{Q}(\nu)$ of the stability function $\mathbf{R}(\nu)$ are

$\nu_1 = 1/\theta$ (a single root) and $\nu_{2,3} = 2/\theta$ (a double root). This means that the stability function $R(\nu)$ is analytic in $\mathbb{C}^-$ (because these roots are positive), which completes the proof of Step B.

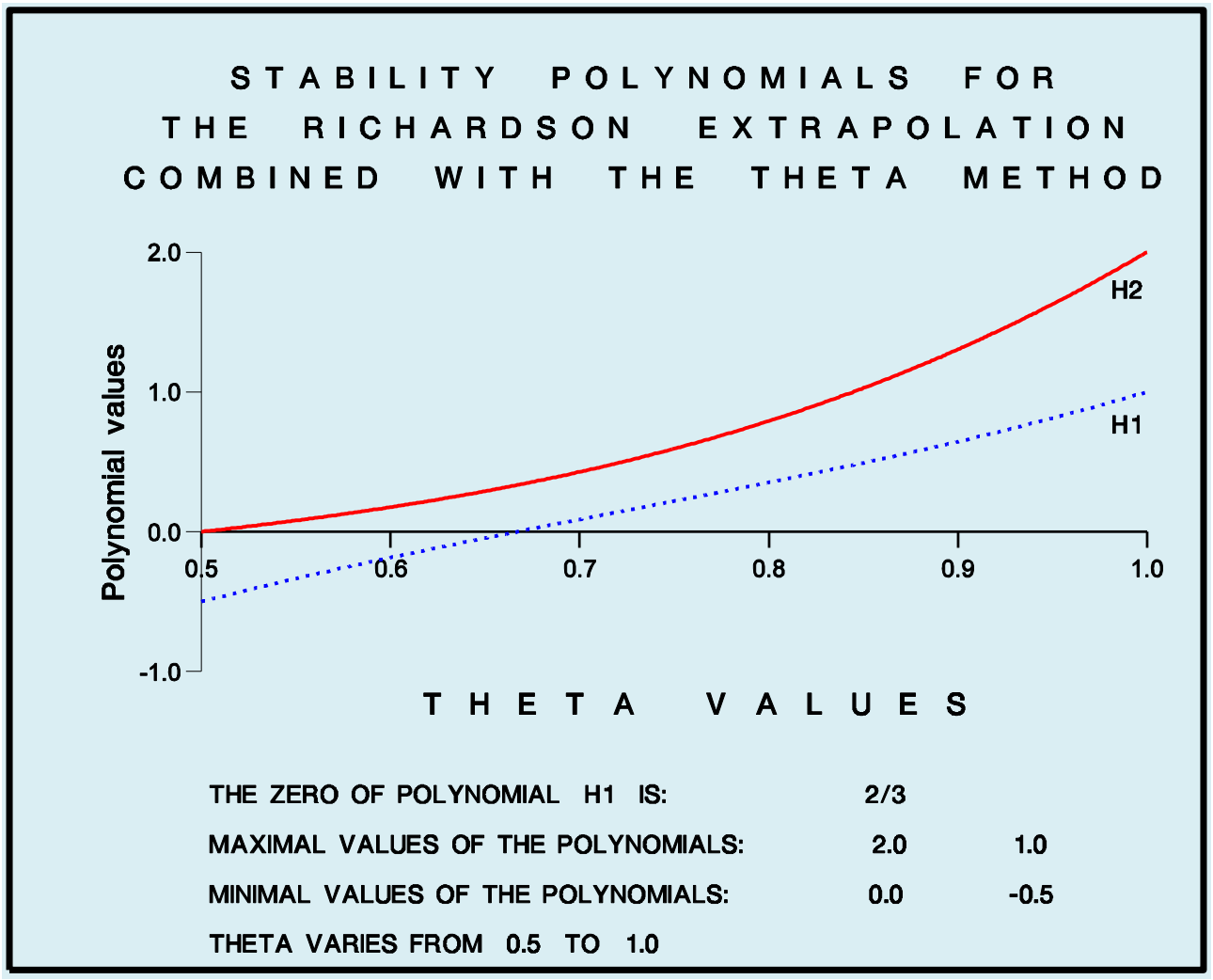

Figure 4.1

Variations of the two polynomials $H_1(\theta)$ and $H_2(\theta)$ for $\theta \in [2/3, 1.0]$. The dotted curve represents the polynomial H_1, while the continuous curve represents the polynomial H_2. It is clearly seen that the two polynomials are non-negative in the interval $[2/3, 1.0]$.

Step C: Strong A-stability

It remains to establish for which values of θ in the interval $[2/3, 1.0]$ the required relationship $|R(\nu)| \to c < 1$ holds as $Re(\nu) \to -\infty$. Since $\nu = \alpha + \beta i$ with $\alpha \leq 0$, it is clear that $Re(\nu) = \alpha$. This fact will be exploited in the proof.

Rewrite first (4.18) as

$$(4.34) \quad \bar{R}(\nu) = 2\left[\frac{1+(1-\theta)(0.5\,\nu)}{1-\theta(0.5\,\nu)}\right]^2 - \frac{1+(1-\theta)\nu}{1-\theta\nu}$$

$$= 2\left[\frac{\frac{1}{-\nu}-0.5+0.5\theta}{\frac{1}{-\nu}+0.5\theta}\right]^2 - \frac{\frac{1}{-\nu}-1+\theta}{\frac{1}{-\nu}+\theta}$$

$$= 2\left[\frac{\frac{1}{-\alpha-\beta i}-0.5+0.5\theta}{\frac{1}{-\alpha-\beta i}+0.5\theta}\right]^2 - \frac{\frac{1}{-\alpha-\beta i}-1+\theta}{\frac{1}{-\alpha-\beta i}+\theta} \,.$$

Assume now that β is fixed and let $\alpha = \mathrm{Re}(\nu) \to -\infty$. The result is:

$$(4.35) \quad \lim_{\mathrm{Re}(\nu)\to-\infty} \bar{R}(\nu) = 2\left[\frac{\theta-1}{\theta}\right]^2 - \frac{\theta-1}{\theta}$$

$$= \frac{\theta^2-3\theta+2}{\theta^2} \,.$$

Since the terms in the right-hand-side of (4.35) are real, the requirement $|R(\nu)| \to c < 1$ as $\mathrm{Re}(\nu) \to -\infty$ reduces to $\left|(\theta^2-3\theta+2)/\theta^2\right| \le 1$. This inequality implies that the following two relationships are simultaneously satisfied:

$$(4.36) \quad \frac{\theta^2-3\theta+2}{\theta^2} < 1 \quad \Rightarrow \quad \theta^2-3\theta+2 < \theta^2 \quad \Rightarrow \quad \theta > \frac{2}{3} \,,$$

$$(4.37) \quad -1 < \frac{\theta^2-3\theta+2}{\theta^2} \quad \Rightarrow \quad 2\theta^2-3\theta+2 > 0 \,.$$

This completes the proof, because the second inequality in (4.37) holds for all real values of θ (the minimal value of the polynomial $2\theta^2-3\theta+2$ is $7/8$, which is achieved for $\theta = 3/4$).

■

Corollary 4.1: If $\theta = 1.0$ (i.e. if the Backward Euler Formula is used) then the combined method (the Backward Euler Formula + the Richardson Extrapolation) is L-stable.

Proof: It is immediately seen that the right-hand-side of (4.35) is equal to zero when $\boldsymbol{\theta = 1.0}$ and, thus, the method is L-stable.

■

Remark 4.1: It is much easier to prove Theorem 4.1 directly for the Backward Euler Formula. Indeed, the stability function (4.18) becomes much simpler with $\boldsymbol{\theta = 1.0}$ and the expressions for $\boldsymbol{Q(i\beta)\, Q(-i\beta)}$ and $\boldsymbol{P(i\beta)\, P(-i\beta)}$ from (4.26) and (4.29) become also much simpler in this case:

$$\boldsymbol{Q(i\beta)\, Q(-i\beta) = 0.0625\beta^6 + 0.5625\beta^4 + 1.5\beta^2 + 1} \qquad (\boldsymbol{4.38})$$

and

$$\boldsymbol{P(i\beta)\, P(-i\beta) = 0.0625\beta^4 + 1.5\beta^2 + 1}\,. \qquad (\boldsymbol{4.39})$$

Theorem 4.1 was proved directly for the Backward Euler Formula in **Faragó, Havasi and Zlatev (2010)**.

■

Remark 4.2: Corollary 4.1 and Remark 4.1 show that the main result in **Faragó, Havasi and Zlatev (2010)**, the assertion that the Backward Euler Formula is L-stable, is just a special cases of Theorem 4.1, which was proved above.

■

Remark 4.3: Equality (4.35) shows that the constant $\boldsymbol{c}$ depends on the selected value of parameter $\boldsymbol{\theta}$. For every value of this parameter, the corresponding value of $\boldsymbol{c}$ can be calculated by using (4.35). Theorem 4.1 shows that $\boldsymbol{c}$ is less than one or equal to one for all $\boldsymbol{\theta \geq 2/3}$. For example, if $\boldsymbol{\theta = 0.75}$, then $\boldsymbol{c = 5/9}$.

■

Remark 4.4: Theorem 4.1 cannot be applied directly for the Trapezoidal Rule. The problem is that the expression for the stability function from (4.18), which is valid for the case $\boldsymbol{\theta \neq 0.5}$ was used in the proof of this theorem. It is necessary to apply the stability function from (4.17), because the Trapezoidal Rule, which is obtained for $\boldsymbol{\theta = 0.5}$ from (4.1), is a second-order numerical method. This is done in Theorem 4.2, which is proved below.

■

Theorem 4.2: The combination of the active implementation of the Richardson Extrapolation with the Trapezoidal Rule (i.e. with the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.5}$) is **not** an A-stable numerical method.

Proof: Consider (4.19) and perform the following transformations:

$$(\mathbf{4.40}) \qquad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{4}\left[\frac{\mathbf{1+0.25\,\nu}}{\mathbf{1-0.25\,\nu}}\right]^{\mathbf{2}} - \frac{\mathbf{1+0.5\nu}}{\mathbf{1-0.5\nu}}}{\mathbf{3}}$$

$$= \frac{\mathbf{4}\left[\frac{\frac{\mathbf{1}}{\boldsymbol{\nu}}+\mathbf{0.25}}{\frac{\mathbf{1}}{\boldsymbol{\nu}}-\mathbf{0.25}}\right]^{\mathbf{2}} - \frac{\frac{\mathbf{1}}{\boldsymbol{\nu}}+\mathbf{0.5}}{\frac{\mathbf{1}}{\boldsymbol{\nu}}-\mathbf{0.5}}}{\mathbf{3}}.$$

It is obvious that

$$(\mathbf{4.41}) \qquad \lim_{\boldsymbol{\nu}\to\infty} |\bar{\mathbf{R}}(\boldsymbol{\nu})| = \frac{\mathbf{5}}{\mathbf{3}},$$

which means that $|\bar{\mathbf{R}}(\boldsymbol{\nu})| > \mathbf{1}$ when $|\boldsymbol{\nu}|$ is sufficiently large and, thus, the combination of the active implementation of the Richardson Extrapolation with the Trapezoidal Rule is **not** an A-stable numerical method.

■

It is worthwhile to present additionally the following two remarks here:

Remark 4.5: It is perhaps necessary to explain what is the meaning of $\boldsymbol{\nu} \to \infty$ when $\boldsymbol{\nu}$ is a complex number. It is convenient to apply the following definition in this case. If $\boldsymbol{\nu} \in \mathbb{C}$, then $\boldsymbol{\nu} \to \infty$ will always mean that $|\boldsymbol{\nu}|$ grows beyond any assigned positive real number.

■

Remark 4.6: The numerical schemes from the class of the $\boldsymbol{\theta}$-methods have good stability properties when $\boldsymbol{\theta} \in [\mathbf{0.5},\ \mathbf{2/3})$. The Trapezoidal Rule, obtained with $\boldsymbol{\theta = 0.5}$, is A-stable, while the numerical methods found when $\boldsymbol{\theta} \in (\mathbf{0.5},\ \mathbf{2/3})$ are even strongly A-stable. Unfortunately the good stability properties are lost when these methods are combined with the active implementation of the Richardson Extrapolation for $\boldsymbol{\theta} \in [\mathbf{0.5},\ \mathbf{2/3})$. This means that **the new methods** obtained when the active implementation of the Richardson Extrapolation is combined with the numerical schemes from the class of the $\boldsymbol{\theta}$-methods should not be used with $\boldsymbol{\theta} \in [\mathbf{0.5},\ \mathbf{2/3})$. However, the new methods obtained with **the passive implementation** of the Richardson Extrapolation will very

often give good results also for $\theta \in [0.5,\ 2/3)$ (because the combination of the passive implementation of the Richardson Extrapolation with any numerical method has the same stability properties as those of the underlying method).

■

4.5. The problem with the implicitness

If the problem solved, the initial values problem for systems of ODEs defined by (1.1) and (1.2), is stiff, then one is as a rule forced to use A-stable, strongly A-stable or L-stable methods during the numerical solution. However, as stated in the previous sections of this chapter, all these methods are necessarily **implicit** (because of the second Dahlquist barrier). The implicitness of the numerical schemes is very often causing great difficulties when the problem solved is large. This is especially true when combinations of numerical schemes from the class of the θ-methods with the Richardson Extrapolation are applied in the solution of (1.1) – (1.2).

The problem of implicitness arising when stiff systems of ODEs are solved will be discussed in this section and some recommendations and conclusions that are related to the efficient treatment of the computational process in the case where the Richardson Extrapolation is used will be given. Three important applications of the well-known Newton iterative method, see, for example, **Kantorovich and Akilov (1964)**, in connection with the numerical treatment of stiff systems of ODEs by implicit numerical schemes from the class of the θ-methods will be described in this section. After that the implications, which arise when these schemes are combined with the Richardson Extrapolation, will be explained.

4.5.1. Application of the classical Newton iterative method

Assume that some A-stable, strongly A-stable or L-stable numerical scheme from the class of the θ-methods with $\theta \in [0.5, 1.0]$ is to be used. When such a scheme, which is implicit, is used in the solution of the system of ODEs defined with (1.1) and (1.2), the following non-linear system of algebraic equations or system of transcendental equations has to be solved at every time-step:

$$y_n - h\,\theta\,f(t_n, y_n) - g_{n-1} = 0 \quad \text{for} \quad n = 1, 2, \ldots, N. \tag{4.42}$$

The solution of (4.42), which in general must be found by solving a large (or even very large, see, for example, **Zlatev and Dimov, 2006**) non-linear system of algebraic equations and/or system of transcendental equations, is y_n, while

$$g_{n-1} = -y_{n-1} - h\,(1-\theta) f(t_{n-1}, y_{n-1}) \tag{4.43}$$

is a known vector.

It is clear that (4.42) and (4.43) can easily be obtained by using (4.1).

It is convenient now to introduce the following notation:

$$(\mathbf{4.44}) \quad \mathbf{F(y_n) = y_n - h\,\theta\,f(t_n, y_n) - g_{n-1}} \qquad \text{for} \quad \mathbf{n = 1, 2, \ldots, N},$$

$$(\mathbf{4.45}) \quad \mathbf{J = \frac{\partial f(t, y)}{\partial y}} \quad \text{and} \quad \mathbf{J_n = \frac{\partial f(t_n, y_n)}{\partial y_n}} \qquad \text{for} \quad \mathbf{n = 1, 2, \ldots, N}$$

as well as

$$(\mathbf{4.46}) \quad \mathbf{\frac{\partial F(y_n)}{\partial y_n} = I - h\,\theta\,J_n} \qquad \text{for} \quad \mathbf{n = 1, 2, \ldots, N},$$

where $\mathbf{I}$ is the identity matrix in $\mathbb{R}^{\mathbf{s\times s}}$.

Assume that the classical Newton iterative method is used to solve (approximately, according to some prescribed in advance accuracy) the non-linear system of algebraic and/or transcendental equations:

$$(\mathbf{4.47}) \quad \mathbf{F(y_n) = 0},$$

or, in other words, assume that the classical Newton iterative method is used to solve the non-linear system of algebraic and/or transcendental equations $\mathbf{y_n - h\,\theta\,f(t_n, y_n) - g_{n-1} = 0}$, which appears when an arbitrary implicit numerical algorithm from the class of the $\boldsymbol{\theta}$-methods is used with some $\boldsymbol{\theta} \in [\mathbf{0.5, 1.0}]$.

The major formulae that are needed at the $\mathbf{k^{th}}$ iteration of the **classical Newton iterative method** can be written in the following form (assuming that the iteration numbers are given as superscripts in square brackets):

$$(\mathbf{4.48}) \quad \left(\mathbf{I - h\,\theta\,J_n^{[k-1]}}\right) \mathbf{\Delta y_n^{[k]} = -y_n^{[k-1]} + h\,\theta\,f\left(t_n, y_n^{[k-1]}\right) + g_{n-1}} \qquad \text{for} \quad \mathbf{k = 1, 2, \ldots}$$

$$(\mathbf{4.49}) \quad \mathbf{y_n^{[k]} = y_n^{[k-1]} + \Delta y_n^{[k]}} \qquad \text{for} \quad \mathbf{k = 1, 2, \ldots}.$$

Some initial approximation $\mathbf{y_n^{[0]}}$ is needed in order to start the iterative process defined by (4.48) and (4.49). The following two choices are often used in practice:

$$\mathbf{(4.50)} \quad \mathbf{y_n^{[0]} = y_{n-1}}$$

and

$$\mathbf{(4.51)} \quad \mathbf{y_n^{[0]} = y_{n-1} + \frac{h_n}{h_{n-1}} (y_{n-1} - y_{n-2})},$$

where it is assumed that $\mathbf{h_n}$ and $\mathbf{h_{n-1}}$ are the last two time-stepsizes that were used in the computational process. This means that it is furthermore assumed here that variations of the time-stepsize are allowed. It is obvious that (4.51) is reduced to

$$\mathbf{(4.52)} \quad \mathbf{y_n^{[0]} = 2\, y_{n-1} - y_{n-2}},$$

when $\mathbf{h_n = h_{n-1}}$.

It should be mentioned that (4.51) and (4.52) are used in the experiments, results of which will be reported in the next section.

Consider an arbitrary iteration step $\mathbf{k}$ ($\mathbf{k = 1, 2, ..., k^{end}}$) of the classical Newton iterative process applied in the solution of (4.47). It is also assumed that $\mathbf{k^{end}}$ is the last iteration step, i.e. the iteration step at which the iterative process will be stopped by using some appropriate stopping criteria (the choice of stopping criteria will be discussed in §4.5.4). When the iterative process is successfully stopped, $\mathbf{y_n^{[k^{end}]}}$ is accepted as a sufficiently good approximation of the exact value $\mathbf{y(t_n)}$ of the solution of (1.1) – (1.2) and $\mathbf{y_n}$ is set equal to $\mathbf{y_n^{[k^{end}]}}$.

The computational work during iteration step $\mathbf{k}$ of the Newton iterative process consists of six parts, which must consecutively be performed. The numerical algorithm given below is defined by using these six parts:

Algorithm 1: Performing an arbitrary iteration of the classical Newton Method.

Part 1 – Function evaluation. Calculate the $\mathbf{s}$ components of the right-hand-side vector $\mathbf{f\left(t_n, y_n^{[k-1]}\right)}$ of (1.1).

Part 2 – Jacobian evaluation. Calculate the elements of the Jacobian matrix $\mathbf{J_n^{[k-1]}}$.

Part 3 – Factorize the shifted Jacobian matrix $\mathbf{I - h\,\theta\,J_n^{[k-1]}}$**.** Calculate the elements of the shifted Jacobian matrix and the triangular matrices $\mathbf{L_n^{[k-1]}}$ and $\mathbf{U_n^{[k-1]}}$ such that $\mathbf{L_n^{[k-1]}U_n^{[k-1]} \approx I - h\,\theta\,J_n^{[k-1]}}$ by using some version of the well-known Gaussian Elimination. The symbol $\approx$ is used here only in order to emphasize the fact that, because of the rounding errors, in practice it is impossible to obtain an exact factorization of matrix $\mathbf{I - h\,\theta\,J_n^{[k-1]}}$ when the calculations are carried out on computers. However, $\mathbf{L_n^{[k-1]}U_n^{[k-1]}}$ will normally be a very close approximation of $\mathbf{I - h\,\theta\,J_n^{[k-1]}}$. Nevertheless, one should not discard totally the effect of the rounding errors especially when the shifted Jacobian matrix is very ill-conditioned. We shall assume that some care has been taken to reduce or even eliminate to a certain degree the effect of the rounding errors (for example by applying extended, quadruple, precision as we already did in Chapter 2) and, because of this, we shall use the notation $\mathbf{L_n^{[k-1]}U_n^{[k-1]} = I - h\,\theta\,J_n^{[k-1]}}$ in the remaining part of this chapter.

Part 4 – Solve the system of linear algebraic equations. Use the computational process, which is very often called "back substitution" (see, for example, **Golub and Van Loan, 1983**, **Jennings, 1977** or **Wilkinson, 1963, 1965**), in order to obtain the solution $\mathbf{\Delta y_n^{[k]}}$ of the system of linear algebraic equations $\mathbf{L_n^{[k-1]}U_n^{[k-1]}\,\Delta y_n^{[k]} = -y_n^{[k-1]} + h\,\theta\,f\left(t_n, y_n^{[k-1]}\right) + g_{n-1}}$. Also here because of the rounding errors some approximation of the correction vector $\mathbf{\Delta y_n^{[k]}}$ will be obtained, but as a rule the calculated vector will be a very close approximation of the exact $\mathbf{\Delta y_n^{[k]}}$. As in Part 3, we shall assume that some care has been taken in order to reduce the effect of the rounding errors (for example by applying again, as in Chapter 2, quadruple precision arithmetic in the computational process).

Part 5 – Update the solution. Use formula (4.49) to calculate the components of vector $\mathbf{y_n^{[k]}}$.

Part 6 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation $\mathbf{y_n^{[k]}}$ is acceptable or not.

One of the three actions listed below are to be taken in Part 6 after the check of the stopping criteria:

Action 1: If **all** stopping criteria are satisfied, then

(a) declare $\mathbf{k}$ as $\mathbf{k^{end}}$,

(b) set $\mathbf{y_n}$ equal to $\mathbf{y_n^{[k^{end}]}}$

and

(c) stop the iterative process.

Action 2: If some of the stopping criteria are **not** satisfied, but the code judges that the convergence rate is sufficiently fast, then

(a) set $\mathbf{k := k + 1}$

and

(b) go to Part 1 of the above algorithm in order to start the next iteration.

Action 3: If there are stopping criteria, which are not satisfied and if the iterative process is judged to be either divergent or very slowly convergent, then

(a) set $\mathbf{k := 1}$,

(b) reduce the time-stepsize $\mathbf{h}$

and

(c) restart the Newton iteration.

The most time-consuming parts when large or very large systems of ODEs are solved are Part 2, Part 3 and Part 4 of the above algorithm for performing an arbitrary step of the Newton iterative process. Very often Part 1 is also time-consuming.

Different modifications of the algorithm are to be introduced in order to achieve a more efficient computational process. Some of the modifications will be discussed in the following two sub-sections.

4.5.2. Application of the modified Newton iterative method

The first attempt to improve the efficiency of the computational process is made by calculating the Jacobian matrix and factorizing it only during the first iteration step of the Newton iterative process. In other words, the first iteration step, when $\mathbf{k = 1}$, is carried out by Algorithm 1, while the algorithm given below is used in the next iteration steps, i.e. in the iteration steps with $\mathbf{k > 1}$.

Algorithm 2: Performing an arbitrary iteration of the modified Newton Method.

Part 1 – Function evaluation. Calculate the $\mathbf{s}$ components of the right-hand-side vector $\mathbf{f}\left(\mathbf{t_n}, \mathbf{y_n^{[k-1]}}\right)$ of (1.1).

Part 2 – Solve the system of linear algebraic equations. Use the computational process, which is called, as mentioned in the previous sub-section, "back substitution", in order to obtain the solution $\Delta\mathbf{y_n^{[k]}}$ of the system of linear algebraic equations $\mathbf{L_n^{[1]}U_n^{[1]}}\,\Delta\mathbf{y_n^{[k]}} = -\mathbf{y_n^{[k-1]}} + \mathbf{h}\,\boldsymbol{\theta}\,\mathbf{f}\left(\mathbf{t_n}, \mathbf{y_n^{[k-1]}}\right) + \mathbf{g_{n-1}}$.

Part 3 – Update the solution. Use formula (4.49) to calculate the components of vector $\mathbf{y_n^{[k]}}$.

Part 4 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation $\mathbf{y_n^{[k]}}$ is acceptable or not.

Some modifications of the actions used in the stopping criteria are also needed. The modified actions, which are to be taken in Part 4 of Algorithm 2 (after the check of the stopping criteria) are listed below:

Action 1: If all stopping criteria are satisfied, then

(a) declare $\mathbf{k}$ as $\mathbf{k^{end}}$,

(b) set $\mathbf{y_n}$ equal to $\mathbf{y_n^{[k^{end}]}}$

and

(c) stop the iterative process.

Action 2: If some of the stopping criteria are not satisfied, but the code judges that the convergence rate is sufficiently fast, then

(a) set $\mathbf{k := k + 1}$

and

(b) go to Part 1 of the above algorithm in order to start the next iteration.

Action 3: If there are stopping criteria, which are not satisfied, if $\mathbf{k > 1}$ and if the iterative process is either divergent or very slowly convergent, then

(a) set $\mathbf{k := 1}$,

and

(b) restart the Newton iteration (i.e. perform one iteration step by using Algorithm 1 and continue after that with Algorithm 2).

Action 4: If there are stopping criteria, which are not satisfied, if $\mathbf{k = 1}$ and if the iterative process is either divergent or very slowly convergent, then

(a) reduce the time-stepsize $\mathbf{h}$

and

(b) restart the Newton iteration.

The advantages of this algorithm are two: the expensive (in terms of the performed arithmetic operations) Part 2 and Part 3 of Algorithms 1 are carried out as a rule only during the first iteration step (and omitted at the next iteration steps as long as the process is converging and the convergence rate is sufficiently fast). However, one has to pay something for the reduction of the number of arithmetic operations. The problem is that, while the classical Newton iterative process is of second order of accuracy, the modified one is of first order only (see more details in Chapter XVIII of **Kantorovich and Akilov, 1964**). This will often lead to an increase of the number of iterations. Nevertheless, the gains are achieved because of the reductions of the numbers of Jacobian evaluations and matrix factorizations is normally greater than the increase of the number of iterations.

4.5.3. Achieving better efficiency by keeping an old decomposition of the Jacobian matrix

The efficiency of the computational process could in many cases be further improved by trying to keep the factorized, at some previous time-step, Jacobian matrix as long as possible. Let $\mathbf{j < n}$ and $\mathbf{i \geq 1}$ be the time-step and the iteration number at which the last evaluation of the Jacobian matrix and the last factorization of this matrix were performed. One can attempt to apply the triangular factors $\mathbf{L_j^{[i]}}$ and $\mathbf{U_j^{[i]}}$ of the Jacobian matrix $\mathbf{I - h\,\theta\,J_j^{[i]}}$ also when time-step $\mathbf{n}$ is carried out.

The advantage of using this approach is due to the fact that very often there will be no need to calculate the elements of the Jacobian matrix at step $\mathbf{n}$ and no need to factorize it. The disadvantage is the same as that mentioned in the previous sub-section: the convergence rate may become slow. However, as in the case with the modified Newton iterative process, the experimental results indicate that often this algorithm works rather well in practice and this approach is implemented in many standard codes for solving stiff systems of ODEs.

It should be emphasized that, as mentioned in the previous sub-section, the computational scheme described below is normally very effective when the solved problems are large. Some discussion about the convergence of the Newton iterative process in this case is given in **Zlatev (1981a)**.

More details about the implementation of this algorithm as well as many numerical results obtained in the treatment of many problems can be found for example in **Hindmarsh (1980)**, **Krogh (1973)**, **Shampine (1984, 1994)**, **Shampine and Gordon (1976)** or **Zlatev and Thomsen (1979)**.

Algorithm 3: Further improvement of the performance of the Newton Method.

Part 1 – Function evaluation. Calculate the $\mathbf{s}$ components of the right-hand-side vector $\mathbf{f}\left(\mathbf{t_n}, \mathbf{y_n^{[k-1]}}\right)$ of (1.1).

Part 2 – Solve the system of linear algebraic equations. Use the computational process, which is normally called "back substitution", in order to obtain the solution $\Delta\mathbf{y_n^{[k]}}$ of the system of linear algebraic and/or transcendental equations $\mathbf{L_j^{[i]} U_j^{[i]}\,\Delta y_n^{[k]} = -y_n^{[k-1]} + h\,\theta\,f\left(t_n, y_n^{[k-1]}\right) + g_{n-1}}$ where $\mathbf{j \leq n}$ and $\mathbf{j \geq 1}$.

Part 3 – Update the solution. Use formula (4.49) to calculate the components of vector $\mathbf{y_n^{[k]}}$.

Part 4 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation $\mathbf{y_n^{[k]}}$ is acceptable or not.

Also in this case some modifications of the actions used in the stopping criteria are needed. The modified actions, which are carried out in Part 4 of Algorithm 3 are listed below:

Action 1: If all stopping criteria are satisfied, then

(a) declare $\mathbf{k}$ as $\mathbf{k^{end}}$,

(b) set $\mathbf{y_n}$ equal to $\mathbf{y_n^{[k^{end}]}}$

and

(c) stop the iterative process.

Action 2: If some of the stopping criteria are not satisfied, but the code judges that the convergence rate is sufficiently fast, then

(a) set $\mathbf{k := k + 1}$

and

(b) go to Part 1 of the above algorithm in order to start the next iteration.

Action 3: If there are stopping criteria, which are not satisfied, if $\mathbf{j < n}$ or $\mathbf{j = n}$ but $\mathbf{k > 1}$, and if the iterative process is either divergent or very slowly convergent, then

(a) set $\mathbf{j := n}$ as well as $\mathbf{k := 1}$,

and

(b) restart the Newton iteration (i.e. perform one iteration step by using Algorithm 1 and continue after that with Algorithm 3).

Action 4: If there are some stopping criteria, which are not satisfied, if $\mathbf{j = n}$ and $\mathbf{k = 1}$ and if the iterative process is either divergent or very slowly convergent, then

(a) reduce the time-stepsize $\mathbf{h}$

and

(b) restart the Newton iteration.

4.5.4. Selecting stopping criteria

By using different stopping criteria in the three algorithms describes in §4.5.1, §4.5.2 and §4.5.3 one is mainly trying:

(A) to achieve sufficiently good accuracy,

(B) to avoid the use of too many iterations,

(C) to decide whether it is worthwhile to continue the iterative process

and

(D) to find out whether it is necessary to update the Jacobian matrix and its factorization when Algorithm 2 and Algorithm 3 are used.

These four categories of stopping criteria are discussed in the following part of this sub-section.

(A) Efforts to ensure sufficiently accurate approximations. One is first and foremost interested in achieving sufficiently accurate approximations. Therefore, the first group of the stopping checks is related to the evaluation of the accuracy of the approximation $\mathbf{y_n^{[k]}}$ calculated at iteration $\mathbf{k}$ of the Newton iterative process.

Assume that the accuracy requirement is prescribed by some error tolerance $\mathbf{TOL}$, which is provided by the user (for example, if it is required that the numerical errors are kept less than $\mathbf{10^{-3}}$ then $\mathbf{TOL = 10^{-3}}$ should be specified). By using the error tolerance $\mathbf{TOL}$ one can try to control, at every iteration step, the accuracy checking whether either

$$\left\|\Delta \mathbf{y_n^{[k]}}\right\| < \mathbf{TOL} \qquad \text{for} \quad \mathbf{k = 1, 2, \ldots} \tag{4.53}$$

or

$$\frac{\left\|\Delta \mathbf{y_n^{[k]}}\right\|}{\left\|\mathbf{y_n^{[k]}}\right\|} < \mathbf{TOL} \qquad \text{for} \quad \mathbf{k = 1, 2, \ldots} \ . \tag{4.54}$$

The choice of a particular norm in our opinion is in many cases not very critical (because all norms in finite spaces are in some sense equivalent).

The first check is absolute, the second one is relative. One should be careful with the choice of any of these two checks. The absolute check can give problems when $\left\|\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}\right\|$ is large, because it will lead to the performance of too many iterations. Therefore, the relative stopping check is more preferable in such a case. However, the relative check can cause problems when $\left\|\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}\right\| \rightarrow \mathbf{0}$ and the absolute check should be used in this situation.

One can try to combine the two check and force the code to select automatically the better check by requiring:

$$\mathbf{(4.55)} \qquad \frac{\left\|\Delta \mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}\right\|}{\max\left(\left\|\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}\right\|,\ \mathbf{1}\right)} < \mathbf{TOL} \qquad \text{for} \quad \mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots .$$

It is clear that the check introduced by (4.55) will work as an absolute stopping criterion when $\left\|\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}\right\| < \mathbf{1}$ and as a relative one otherwise. The check (4.55) is often called mixed stopping criterion. Some positive constant (say, $\mathbf{c}$) can be used instead of $\mathbf{1}$ in (4.55).

It should be pointed out here that in all three stopping criteria, which were introduced above, it is implicitly assumed that all components of vector $\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}$ are of the same order of magnitude. Unfortunately, this requirement is not always satisfied when different problems arising in science and engineering are to be treated numerically. One such example is the atmospheric chemical scheme used in the Unified Danish Eulerian Model (UNI-DEM, see **Zlatev, 1995**, or **Zlatev and Dimov, 2006**), which was mentioned in Chapter 1 and will be discussed in detail in the next section. The concentrations of the chemical species involved in this scheme differ by many orders of magnitude. Therefore, it is necessary to introduce and to use component-wise stopping criteria (instead of stopping criteria based on norms) when such problems are to be handled.

Assume that the components of vectors $\mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}$ and $\Delta \mathbf{y}_{\mathbf{n}}^{[\mathbf{k}]}$ are denoted by $\mathbf{y}_{\mathbf{nq}}^{[\mathbf{k}]}$ and $\Delta \mathbf{y}_{\mathbf{nq}}^{[\mathbf{k}]}$ where $\mathbf{q} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{s}$. By using this notation, three component-wise stopping criteria, corresponding to the stopping criteria defined by (4.53), (4.54) and (4.55) are given below:

$$\mathbf{(4.56)} \qquad \max_{\mathbf{q}=\mathbf{1}, \mathbf{2}, \ldots, \mathbf{s}} \left(\left|\Delta \mathbf{y}_{\mathbf{nq}}^{[\mathbf{k}]}\right|\right) < \mathbf{TOL} \qquad \text{for} \quad \mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots ,$$

$$\mathbf{(4.57)} \qquad \max_{\mathbf{q}=\mathbf{1}, \mathbf{2}, \ldots, \mathbf{s}} \left(\frac{\left|\Delta \mathbf{y}_{\mathbf{nq}}^{[\mathbf{k}]}\right|}{\left|\mathbf{y}_{\mathbf{nq}}^{[\mathbf{k}]}\right|}\right) < \mathbf{TOL} \qquad \text{for} \quad \mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots ,$$

$$(4.58) \qquad \max_{q=1,\ 2,\ \ldots,\ s} \left(\frac{\left|\Delta y_{nq}^{[k]}\right|}{\max\left(\left|y_{nq}^{[k]}\right|,\ 1\right)} \right) < TOL \qquad \text{for} \quad k = 1, 2, \ldots .$$

Also here some positive constant (say, c) can be used instead of 1.

It should be mentioned that the check (4.56) is not very different from the checks based on the norm of the calculated solution vector. In fact the quantity in the right-hand-side of (4.56) is a particular norm of this vector.

It is clear that the stopping criteria based on (4.56) will cause difficulties when the absolute values of all components of the solution vector are large numbers, while problems will appear when (4.57) is used and all components of the solution are very small in absolute value. Therefore, stopping criteria based on the use of (4.58) with some positive constant c seems to be most reliable when the components of the involved vectors vary in a wide range. It should be mentioned that some of the problems arising when the stopping criteria (4.56) and (4.57) are used will disappear if appropriate scaling could be performed.

It should also be mentioned here that the component-wise stopping criteria (4.58) is used in the numerical experiments, which will be described in the next section.

(B) Preventing performance of too many iterations. If the convergence is too slow or if the computational process is divergent, the computations should be stopped. A special parameter k^{max} should be used and the iterative process should be carried out as long as the iteration number k is less than k^{max}.

(C) Efforts to discover whether the computational process will be convergent. The use of parameter k^{max} only may be quite inefficient. Assume, for example, that $k^{max} = 50$ or $k^{max} = 100$. It will not be very efficient to perform 50 or 100 iterations and only after that to find out that the required accuracy could not be achieved (because the Newton method converges too slowly). It is much more desirable to control, from the very beginning, whether the convergence of the iterative process is sufficiently fast and to stop the iterations if there is a danger that this is not the case. Very often this is done by requiring that

$$(4.59) \qquad \left\|\Delta y_n^{[k]}\right\| < \gamma \left\|\Delta y_n^{[k-1]}\right\| \qquad \text{for} \quad k = 2, 3, \ldots$$

and stopping the iterative process if this condition is not satisfied at some iteration k. Parameter γ with $0 < \gamma \leq 1$ is some appropriately chosen factor, by which one attempts to measure the convergence rate.

This stopping criterion in some situations is rather stringent, because the errors sometimes may fluctuate also when the iterative process is convergent (the fluctuations becoming smaller and smaller). Therefore, it is relaxed sometimes by requiring that (4.59) is not satisfied several consecutive times (say, two or three times) before stopping the iterations.

If either Algorithm 2 or Algorithm 3 is used, then (4.59) is also used to decide whether the Jacobian matrix has to be updated and factorized (see below).

(D) Updating the Jacobian matrix and factorizing it. One has to decide when to update the Jacobian matrix and to re-factorize it when Algorithm 2 and Algorithm 3 are used. As mentioned above the check introduced by (4.59) is often used in this decision, i.e. if this check fails and if and old Jacobian matrix is used, then the stepsize is not automatically reduced, but first a new Jacobian matrix is calculated and factorized. In this way some reductions of the stepsize can be avoided.

Sometimes a much simpler check, based on the accuracy tests, is selected. If an old Jacobian matrix is used and if the required accuracy is not achieved after some prescribed number of iterations (often this number is set to three), then a new Jacobian matrix is calculated and factorized.

It is assumed in this subsection that the system of ODEs is non-linear. Then it is necessary to apply some version of the Newton iterative method (or some other iterative procedure). If the systems of ODEs is linear, then the situation is not very clear. The application of any representative of the $\boldsymbol{\theta}$-methods with $\boldsymbol{\theta} \in [\mathbf{0.5}, \mathbf{1.0}]$ leads in this situation to the solution of systems of linear algebraic equations. In principle, one must try to exploit the linearity by solving the system of linear algebraic equation directly. However, if the system of ODEs is very large, then the resulting system of linear algebraic equations is very large too. Therefore, it may be worthwhile to keep, as long as possible, an old Jacobian matrix (calculated and factorized at some previous step) and to use again an iterative method.

It was assumed in this sub-section that the shifted Jacobian matrix $\mathbf{I} - \mathbf{h}\,\boldsymbol{\theta}\,\mathbf{J_n}$ is a general matrix, which has no special properties. However, shifted Jacobian matrices with special properties do appear in many scientific and engineering problems. The shifted Jacobian matrix can, for example, be

(a) positive definite,

(b) diagonally dominant,

(c) banded

and

(d) general sparse.

The shifted Jacobian matrix could possess simultaneously even several of these properties. It is worthwhile to try to exploit these properties. It is not necessary to discuss this topic here, but good explanation of different techniques for exploiting these properties can be found in many text books; see, for example, **Demmel (1997)**, **Duff, Erisman and Reid (1986)**, **George and**

Liu (1981), **Golub and Van Loan (1983)**, **Parlett (1980)**, **Sewell (2004)**, **Trefethen and Bau (1997)** and **Zlatev (1991)**. Standard well-optimized programs for the solution of systems of linear algebraic equations in different situations can be found in **Anderson et al. (1992)** or in **Barker et al. (2001)**.

4.5.5. Richardson Extrapolation and the Newton Method

It was explained in the previous sub-section that the problem of implicitness is causing great difficulties when numerical schemes from the class of the θ-methods with $\theta \in [0.5, 1.0]$ are to be used directly in the solution of stiff systems of ODEs. However, the difficulties become in general considerably bigger when the θ-methods are combined with the Richardson Extrapolation. In this sub-section we shall discuss these difficulties.

Let us assume that the underlying numerical method, i.e. the selected numerical scheme from the class of the θ-methods with some particular value of parameter $\theta \in [0.5, 1.0]$, is called (as in Section 1.6) **Method A**, while the new numerical method, which is obtained when Method A is combined with the Richardson Extrapolation, is called **Method B**. In this sub-section we shall be interested in the comparison of the performance of Method A and Method B, when the three versions of the Newton iterative procedure, which were discussed in §4.5.1, §4.5.2 and §4.5.3, are used.

Assume first that the **classical Newton iterative procedure** from §4.5.1 is to be applied. Assume further that Method A and Method B are used with the same time-stepsize. Then Method B will be approximately three times more expensive with regard to the computing time needed than Method A. Indeed for every time-step performed with Method A, three time-steps (one large and two small) have to be carried out with Method B. In fact, the computing time needed when Method B is used will often be less than three times the computing time needed when Method A is used in the numerical solution of the solved systems of ODEs. The reduction is due to the fact that the number of iterations needed when the two small time-stepsizes are carried out will often be less than the corresponding number, which is needed in the case where the large time-stepsize is used. Nevertheless, this reduction, if it takes place (i.e. if the number of iterations is really reduced when the halved time-stepsize is used), will be rather small (because not the time for performing the iterations but the factorization time is as a rule dominant) and the situation in this case is similar to the situation which occurs when explicit numerical methods are used. As in that case, i.e. when explicit methods are used, the amount of the computational work is increased by a factor approximately equal to three when Method B is used instead of Method A and when additionally both methods are used with the **same** time-stepsize.

Assume now that the **modified Newton iterative process** from §4.5.2 is to be applied. Assume again that Method A and Method B are used with the same time-stepsize. Then the situation remains very similar to the situation, which occurs when the **classical Newton iterative process** is used. Also in this case Method B will be approximately three times more expensive with regard to the computing time needed than Method A.

The real difficulties related to the use of the Richardson Extrapolation appear when Algorithm 3 from §4.5.3 is used. If Method A is used, then an old Jacobian matrix (in fact, its factorization to two triangular matrices) can be kept and used during several consecutive time-steps (as long as the time-stepsize remains constant and the convergence rate is sufficiently fast). This will, unfortunately, not be

possible when Method B is used (because the two time-stepsizes, the time-stepsize used in the large time-step and the halved time-stepsize used in the two small time-steps, are different and, thus, the corresponding shifted Jacobian matrices are also different). This means that it is not very easy to implement efficiently Algorithm 3 together with Method B.

Therefore, it is time now to point out again that **it is not necessary to run the selected numerical scheme and its combination with the Richardson Extrapolation with the same stepsize** (the latter numerical method could be run with a larger stepsize, because it is more accurate). This means that it is much more worthwhile to try to find out by how much the stepsize should be increased in order to make the combination of the selected method with the Richardson Extrapolation at least competitive with the case where the selected method is used directly. We shall try to answer this question in the remaining part of this sub-section.

Denote, as in Chapter 1, by $\mathbf{h_A}$ and $\mathbf{h_B}$ the maximal time-stepsizes by which the prescribed accuracy will be achieved when respectively Method A and Method B are used. It is clear that the computing time spent when Method B is applied will be comparable to the computing time spent by using Method A if $\mathbf{h_B \approx 3h_A}$ when Algorithm 1 or Algorithm 2 is used in the treatment of the Newton iterative method.

As stated above, Algorithm 3 cannot be used together with Method B (the attempt to use Algorithm 3 separately for the large and the small stepsizes will at least lead to a large increase of the storage requirements). It will be more efficient to apply Algorithm 2 than Algorithm 1 with this method. It is clear that Algorithm 2 is the best choice for Method B, while Algorithm 3 is normally the best choice for Method A. Assume now that Algorithm 2 is used with Method B and Algorithm 3 with Method A in the treatment of the Newton iterative method.

Then the computing time spent by Method B will be comparable to the computing time spent by using Method A if $\mathbf{h_B \approx 3mh_A}$, where $\mathbf{m > 3}$. Moreover, the factor $\mathbf{m}$ could sometimes be considerably larger than $\mathbf{3}$. Therefore, the big question now is:

Will it be nevertheless possible to obtain better results with regard to the computing time when Method B is used?

It will be demonstrated in the next section, by applying appropriate numerical examples, that the answer to this question is positive (this was also demonstrated in Table 1.1 of **Chapter 1** but only as a fact, with no explanation of the reasons for achieving the good results).

4.6. Numerical experiments

Also in this section we shall use the abbreviations **Method A** for the underlying numerical method (in the next sub-sections the underlying method will be the selected numerical scheme from the class of the θ-methods with some particular value of parameter $\boldsymbol{\theta} \in [\mathbf{0.5}, \mathbf{1.0}]$) and **Method B** for the new numerical method, obtained when Method A is combined with the Richardson Extrapolation.

We shall demonstrate (by using appropriate numerical experiments) that the two methods, Method A and Method B, have the following useful properties:

(a) Method B behaves as a **second-order** numerical method when the stability properties of the underlying numerical scheme, i.e. the stability properties of Method A, are preserved (this is the case, according to the results proved in the previous section, when the relationship $\boldsymbol{\theta \in [2/3, 1.0]}$ holds).

(b) For some values of $\boldsymbol{\theta < 1}$ the results produced by both Method A and Method B are **more accurate** than the corresponding results produced when the Backward Euler Formula (which is obtained when $\boldsymbol{\theta = 1}$) is used either directly or in a combination with the Richardson Extrapolation.

(c) Method B is often much more efficient than Method A in terms of the computing time needed to obtain the desired results when some prescribed but not too low accuracy is required.

(d) If the conditions of Theorem 4.1 are not satisfied, i.e. if $\boldsymbol{[\theta \in 0.5, 2/3)}$, then Method B produces, as should be expected, unstable results; the well-known Trapezoidal Rule will be used in order to demonstrate this fact.

Several numerical experiments were carried out in order to illustrate the fact that statements **(a)** – **(d)** hold. A representative **atmospheric chemical scheme** was briefly introduced and used in Chapter 1. More details about this chemical scheme will be discussed in the following sub-section and, after that, it will be used in the calculations, the results of which will be presented in this chapter.

4.6.1. Atmospheric chemical scheme

An atmospheric chemical scheme, in which $\boldsymbol{s = 56}$ chemical species are involved, is applied in all experiments, results of which will be presented in the next subsections. This scheme contains all important air pollutants, which can be potentially dangerous when their levels are high (ozone, sulphur pollutants, nitrogen pollutants, ammonium-ammonia pollutants, several radicals and many hydro-carbons). This atmospheric chemical scheme is used, together with two other chemical schemes, in the Unified Danish Eulerian Model (UNI-DEM), see, **Alexandrov et al. (1997, 2004)**, **Zlatev (1995)** and **Zlatev and Dimov (2006)**. Similar atmospheric chemical schemes are used in several other well-known large-scale environmental models as, for example, in the EMEP models (see **Simpson et al., 2003**), in the EURAD model (see **Ebel et al., 2008** and **Memmesheimer, Ebel and Roemer, 1997**) and in the model system carefully adjusted for application in different studies of air pollution levels in Bulgaria and in its surrounding countries (see **Syrakov et al., 2011**). In all these models the chemical species are mainly concentrations of pollutants, which are transported in the atmosphere and transformed under the transportation.

The atmospheric chemistry scheme is described mathematically by a **non-linear** system of ODEs of type (1.1) and (1.2). The numerical treatment of this system is extremely difficult not only because

(a) it is non-linear,

but also because

(b) it is very badly scaled

and

(c) some chemical species vary very quickly during the periods of changes from day-time to night-time and from night-time to day-time when some quick chemical reactions (called photo-chemical) are activated or deactivated.

The fact that the system of ODEs (by which the atmospheric chemical scheme is described mathematically) is non-linear, badly scaled and stiff implies, as was pointed out in the previous sections,

(A) the use of implicit numerical methods for solving systems of ODEs

and

(B) the application of the Newton iterative procedure in the treatment of the arising at every time-step non-linear system of algebraic equations.

The greatest problems are related to the shifted Jacobian matrix, which has to be used **in the Newton iterative procedure**. The shifted Jacobian matrix that appear and has to be treated during the performance of this procedure is both very ill-conditioned and extremely badly scaled.

The bad scaling and the ill-conditioning of the Jacobian matrix $\mathbf{J = df/dx}$ is causing difficulties also in the treatment of **the systems of linear algebraic equations**, which have to be solved at each iteration of the Newton method.

The bad scaling is generated mainly by the fact that the concentrations of some of the chemical species vary in quite different and very wide ranges.

The quick diurnal variation of some of the concentrations is due to the fact that the involved species participate in the so-called photo-chemical reactions which are activated in the morning at the sun-rise and deactivated in the evening after the sun-set. This means that the periods of changes from day-time to night-time and from night-time to day-time are very critical for some of the chemical species.

Both the bad scaling of the chemical species and the steep gradients in the periods of changes from day-time to night-time and from night-time to day-time are clearly demonstrated in in **Table 4.1** in connection with four chemical species. It is seen, for example, that while the maximal concentration of ozone, $\mathbf{O_3}$, is about $\mathbf{10^{12}}$ molecules per cubic centimetre, the minimal concentration of $\mathbf{OP}$ is about $\mathbf{10^{-35}}$ molecules per cubic centimetre (i.e. the difference is about $\mathbf{47}$ orders of magnitude!).

Also the condition numbers of the Jacobian matrices appearing in the same period of 24 hours were calculated at every time-step (by calling standard LAPACK subroutines, see **Anderson et al., 1992**, or **Barker et al., 2001**). The abbreviation $\mathbf{COND}$ is used for the condition number calculated at any time-step during the numerical integration. It was established that the condition numbers are varied in

the interval $\mathbf{COND} \in [\,\mathbf{4.56 \times 10^{8}}\,,\ \mathbf{9.27 \times 10^{12}}\,]$, which shows very clearly that the condition number of the Jacobian matrix $\mathbf{J = \partial f / \partial t}$ can really be very large (this topic will be further discussed in the next sub-section).

Plots, which illustrate the diurnal variation of two chemical species as well as the sharp gradients that appear in the periods of changes from day-time to night-time and from night-time to day-time are given in **Fig. 4.2** and **Fig. 4.3**. Also the fact that some of the concentrations are decreased during the night, while others are increased in this period is demonstrated in these two figures. Moreover, the changes of the concentrations are very quick and create steep gradients. Other examples will be given in Chapter 5.

Chemical species	Maximal concentration	Minimal concentration	Mean concentration
$\mathbf{O_3}$	$\mathbf{1.8 \times 10^{12}}$	$\mathbf{1.4 \times 10^{12}}$	$\mathbf{1.5 \times 10^{12}}$
PAN	$\mathbf{1.3 \times 10^{10}}$	$\mathbf{9.4 \times 10^{5}}$	$\mathbf{2.3 \times 10^{9}}$
ISOPRENE	$\mathbf{3.7 \times 10^{9}}$	$\mathbf{1.1 \times 10^{6}}$	$\mathbf{1.5 \times 10^{9}}$
OP	$\mathbf{1.6 \times 10^{4}}$	$\mathbf{1.7 \times 10^{-35}}$	$\mathbf{5.9 \times 10^{3}}$

Table 4.1

The orders of magnitude and the variations of the concentrations of some chemical species during a period of $\mathbf{24}$ hours (from twelve o'clock at the noon on a given day to twelve o'clock at the noon on the next day). The units are **(numbers of molecules) / (cubic centimetre**).

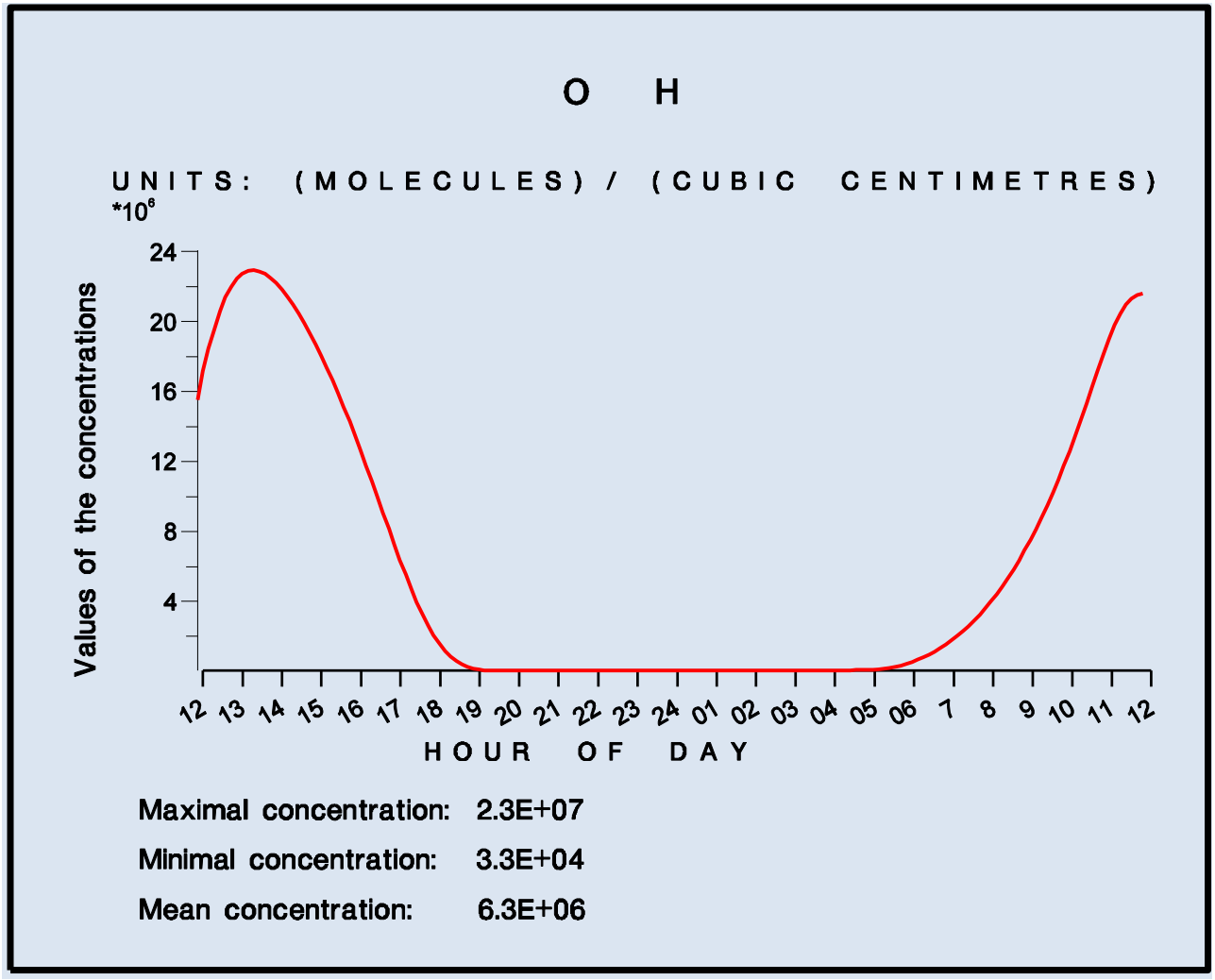

Figure 4.2
Diurnal variation of the concentrations of the chemical species **OH** .

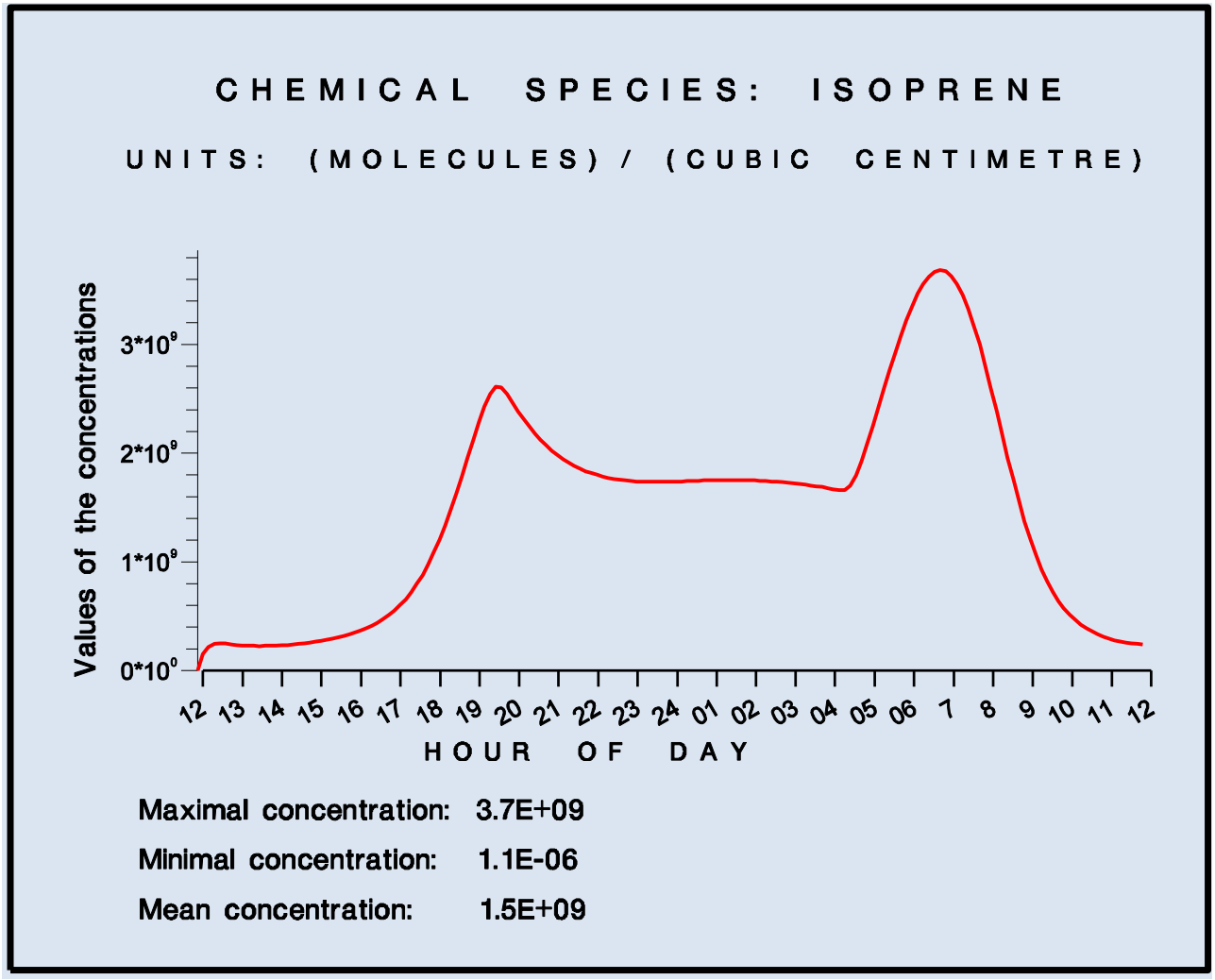

Figure 4.3
Diurnal variation of the concentrations of the chemical species $\mathbf{N_2O_5}$.

4.6.2. Organization of the computations

The organization of the computations, which were carried out in connection with the atmospheric chemical scheme, is very similar to that, which was discussed in the previous chapters, in Chapter 1, Chapter 2 and Chapter 3. However, a more detailed description is needed here, because of the implicitness of the applied in this chapter numerical methods. Such description will be given in this sub-section.

The atmospheric chemical scheme, which was discussed in the previous sub-section, was treated numerically on the time-interval $[\mathbf{a}, \mathbf{b}] = [\mathbf{43200}, \mathbf{129600}]$. The value $\mathbf{a} = \mathbf{43200}$ corresponds to twelve o'clock at the noon (measured in seconds and starting from mid-night), while $\mathbf{b} = \mathbf{129600}$ corresponds to twelve o'clock at the next day (measured also in seconds from the same starting point). Thus, the length of the time-interval used in the numerical experiments in this chapter is $\mathbf{24}$ hours and it contains the important changes from day-time to night-time and from night-time to day-time (when most of the chemical species, as stated in the previous sub-section, are very quickly

varying, because the photo-chemical reactions are deactivated and activated when these changes take place).

Several long sequences of runs were carried out and some of the obtained results will be presented below. The first run in any of these sequences was performed by using $\mathbf{N = 168}$ time-steps, which means that the time-stepsize was $\mathbf{h \approx 514.285}$ seconds. Several runs were successively carried out after the first one. As in the previous two chapters, the time-stepsize $\mathbf{h}$ was halved after each successful run (which means that the number of time-steps was doubled). The behaviour of the errors made during all the runs was studied. The error made at time $\mathbf{\bar{t}_j}$ in any of the runs was measured in the following way. Assume that $\mathbf{\bar{k}}$ runs are to be carried out. The errors calculated during step $\mathbf{j}$ of run $\mathbf{k}$, $\mathbf{k = 1, 2, \ldots, \bar{k}}$ are estimated by using the following formula:

$$\mathbf{(4.60)} \quad \mathbf{ERROR_j^{(k)} = \max_{i=1,2,\ldots 56} \left(\frac{|y_{i,j} - y_{i,j}^{ref})|}{\max(|y_{i,j}^{ref}|, 1.0)} \right), \quad j = 2^{k-1}, 2 \times 2^{k-1}, \ \ldots, 168 \times 2^{k-1},}$$

where $\mathbf{y_{i,j}}$ and $\mathbf{y_{i,j}^{ref}}$ are the calculated values and the values of the reference solution of the $\mathbf{i^{th}}$ chemical species at time $\mathbf{\bar{t}_j = t_0 + jh_0}$ (where $\mathbf{j = 1, 2, \ldots, 168}$ and $\mathbf{h_0 \approx 514.285}$ was the time-stepsize that has been used in the first run). The values of the reference solution were calculated by using the three-stage fifth-order L-stable Fully Implicit Runge-Kutta (FIRK) Method (this method will be further discussed in next section) with $\mathbf{N = 998244352}$ time-steps and a time-stepsize $\mathbf{h_{ref} \approx 6.1307634 \times 10^{-5}}$. Also the errors made for selected chemical species were calculated for some important pollutants (by fixing the index $\mathbf{i}$). In this section we shall use (4.60), but in the next section we shall fix index $\mathbf{i}$ and report results related to the important pollutant **ozone**.

This means that we estimate the error at the same set of grid-points in each of the $\mathbf{\bar{k}}$ runs when (4.60) is used. More precisely, the error is estimated at every time-step during the first run, at every second time-step during the second run, at every fourth time-step during the third run and we continue in the same manner after the third run. Thus, the number of grid-points, at which the error is estimated, is $\mathbf{168}$ for any of the $\mathbf{\bar{k}}$ runs. It should be pointed out that $\mathbf{\bar{k} = 19}$ is used in this section.

It is clear from the above discussion that only the values of the reference solution at the grid-points of the coarse grid (which is used in the first run) have been stored and applied in the evaluation of the error (it is, of course, also possible to store all values of the reference solution, but such an action will increase tremendously the storage requirements). It is much more important and must be emphasized here that errors of the calculated approximations were always, in all nineteen runs, computed at the **same** $\mathbf{168}$ grid points.

The global error made at run $\mathbf{k}$, $\mathbf{k = 1, 2, \ldots, \bar{k}}$ is estimated by:

$$\mathbf{(4.61)} \quad \mathbf{ERROR^{(k)} = \max_{j=2^{k-1}, 2\times 2^{k-1}, \ \ldots, 168\times 2^{k-1}} \left(ERROR_j^{(k)} \right).}$$

It is highly desirable to eliminate the influence of the rounding errors when the quantities involved in (4.42) and (4.43) are calculated. This is not very easy in this situation. Normally, this task can successfully be accomplished when double precision arithmetic is used during the computations. Unfortunately, this is not always true when the atmospheric chemical scheme is handled. The difficulty can be explained as follows. If the problem is stiff, and the atmospheric chemical scheme is as mentioned above a very stiff non-linear system of ODEs, then implicit numerical methods are to be used. The application of such numerical methods leads to the solution of systems of non-linear algebraic equations, which are treated, as described in the previous sub-section, at each time-step by the Newton Iterative Method (see also, for example, **Hairer and Wanner, 1991**). This means that long sequences of systems of linear algebraic equations are to be handled during the iterative process. As a rule, this does not cause great problems. However, the atmospheric chemical scheme is, as mentioned in the previous sub-section, very badly scaled and the condition numbers of the involved in the solution of the systems of linear algebraic equations matrices are very large. It was found, as mentioned above, by applying a LAPACK subroutine for calculating eigenvalues and condition numbers (**Anderson et al., 1992** and **Barker et al., 2001**), that the condition numbers of the matrices involved in the Newton Iterative Process during the numerical integration of the atmospheric chemical scheme with $\mathbf{56}$ chemical species on the time-interval $[\mathbf{a},\mathbf{b}] = [\mathbf{43200}, \mathbf{129600}]$ vary in the range $[\mathbf{4.56 \times 10^{8}}, \mathbf{9.27 \times 10^{12}}]$. Simple application of some error analysis arguments from **Stewart (1973)** and **Wilkinson (1963, 1965)** indicates that there is a danger that the rounding errors could affect the accuracy up to twelve of the sixteen significant digits of the approximate solution on most of the existing computers when double precision arithmetic (based on the use of **REAL*8** declarations of the real numbers and leading to the use of about 16-digit arithmetic on many computers) is applied. Therefore, all computations reported in the next sub-sections were performed by selecting **quadruple-precision** (i.e. by using **REAL*16** declarations for the real numbers and, thus, about 32-digit arithmetic) in order to eliminate completely the influence of the rounding errors in the first 16 significant digits of the computed approximate solutions. This is done in order to demonstrate the possibility of achieving very accurate results under the assumption that stable implementations of the Richardson Extrapolation for the class of the θ-methods are developed and used and, furthermore, to show that the rounding errors do not affect the accuracy of the results in our runs.

After the explanation of the organization of the computations, we are now ready to present some of the results from the numerical experiments, which were carried out in order to demonstrate the advantages of the application of Richardson Extrapolation.

4.6.3. Achieving second order of accuracy

Numerical results, which are obtained by using first-order numerical schemes belonging to the class of the θ-methods in combination with the Richardson Extrapolation are given in **Table 4.2**. The value $\boldsymbol{\theta = 0.75}$ is selected, which means that the relationship $|\mathbf{R}(\nu)| \rightarrow \mathbf{c} < \mathbf{1}$ as $\mathbf{Re}(\nu) \rightarrow -\infty$ holds with $\mathbf{c = 5/9}$, see (4.35). The results in **Table 4.2** show clearly that the θ-method with $\boldsymbol{\theta = 0.75}$ performs:

(a) as a first-order method (as it should) when it is applied directly

and

(b) as **a stable second-order method** when it is used as an underlying method in the Richardson Extrapolation.

Indeed, the decrease of the time-stepsize by a factor of two leads to an increase of the accuracy by a factor of **two** when the θ-method with $\boldsymbol{\theta = 0.75}$ is used directly and by a factor of **four** when this method is combined with the Richardson Extrapolation. Moreover, it is also seen that these two relations (increases of the achieved accuracy by factors of two and four respectively) are fulfilled in a nearly perfect way.

Job Number	Number of time-steps	Direct use of the θ-method		Richardson Extrapolation	
		Accuracy	Rate	Accuracy	Rate
1	168	1.439E-00	-	3.988E-01	-
2	336	6.701E-01	2.147	5.252E-02	7.593
3	672	3.194E-01	2.098	1.503E-03	3.495
4	1344	1.550E-01	2.060	3.787E-03	3.968
5	2688	7.625E-02	2.033	9.502E-04	3.985
6	5376	3.779E-02	2.018	2.384E-04	3.986
7	10752	1.881E-02	2.009	5.980E-05	3.986
8	21504	9.385E-03	2.005	1.499E-05	3.989
9	43008	4.687E-03	2.002	3.754E-06	3.993
10	86016	2.342E-03	2.001	9.394E-07	3.996
11	172032	1.171E-03	2.001	2.353E-07	3.993
12	344064	5.853E-04	2.000	6.264E-08	3.756
13	688128	2.926E-04	2.000	1.618E-08	3.873
14	1376256	1.463E-04	2.000	4.111E-09	3.935
15	2752512	7.315E-05	2.000	1.036E-09	3.967
16	5505024	3.658E-05	2.000	2.601E-10	3.984
17	11010048	1.829E-05	2.000	6.514E-11	3.993
18	22020096	9.144E-06	2.000	1.628E-11	4.001
19	44040192	4.572E-06	2.000	4.051E-12	4.019

Table 4.2

Numerical results that are obtained **(a)** in nineteen runs, in which the direct implementation of the θ-method with $\boldsymbol{\theta = 0.75}$ is used, and **(b)** in the corresponding nineteen runs in which the combination consisting of the Richardson Extrapolation and the θ-method with $\boldsymbol{\theta = 0.75}$ is applied. The errors obtained by using formula (4.61) are given in the columns under "Accuracy". The ratios of two successive errors (the convergence rates) are given in the columns under "Rate".

4.6.4. Comparison of the θ-method with $\theta = 0.75$ and the Backward Euler Formula

It can theoretically be justified that the θ-method with $\boldsymbol{\theta = 0.75}$ should normally give more accurate results than the Backward Euler Formula. More precisely, the following theorem holds:

Theorem 4.3: The principal part of the local truncation error of the θ-method with $\theta = 0.75$ is twice smaller than that of the Backward Euler Formula.

Proof: Consider two approximations $y_n^{backward}$ and $y_n^{\theta=0.75}$ of the exact solution $y(t_n)$ of the problem defined by (1.1) and (1.2), which are obtained at time-step n by applying respectively the Backward Euler Formula and the θ-method with $\theta = 0.75$ assuming that the same initial value $y_n \approx y(t_n)$ is applied. The equations, which are used in the calculation of the approximations $y_n^{backward}$ and $y_n^{\theta=0.75}$, can be written in the following form:

$$(4.63) \quad y_n^{backward} - y_{n-1} - h\, f\left(t_n, y_n^{backward}\right) = 0,$$

and

$$(4.64) \quad y_n^{\theta=0.75} - y_{n-1} - 0.25\, h\, f(t_{n-1}, y_{n-1}) - 0.75\, h\, f\left(t_n, y_n^{\theta=0.75}\right) = 0.$$

Replace:

(a) $y_n^{backward}$ and $y_n^{\theta=0.75}$ with $y(t_n)$

and

(b) y_{n-1} with $y(t_{n-1})$

in the left-hand-side of (4.63) and (4.64).

Use the relationship $dy(t)/dt = f\left(t, y(t)\right)$ and introduce, as on p. 48 in **Lambert (1991)**, two **linear difference operators** in order to express the fact that the right-hand-sides of the expressions obtained from (4.63) and (4.64) will not be equal to zero when the above substitutions are made. The following two relationships can be obtained when these actions are performed:

$$(4.65) \quad L^{backward}[y(t_n); h] = y(t_n) - y(t_{n-1}) - h\,\frac{dy(t_n)}{dt}$$

and

$$(4.66) \quad L^{\theta=0.75}[y(t_n); h] = y(t_n) - y(t_{n-1}) - 0.25\, h\,\frac{dy(t_{n-1})}{dt} - 0.75\, h\,\frac{dy(t_n)}{dt}.$$

Expanding $\mathbf{y(t_n)}$ and $\mathbf{dy(t_n)/dt}$ in Taylor series about $\mathbf{t_{n-1}}$ and keeping the terms containing $\mathbf{h^2}$ one can rewrite (4.65) and (4.66) in the following way:

$$\mathbf{(4.67)} \quad \mathbf{L^{backward}[y(t_n); h] = -\frac{h^2}{2}\frac{d^2y(t_{n-1})}{dt^2} + O(h^3)}$$

and

$$\mathbf{(4.68)} \quad \mathbf{L^{\theta=0.75}[y(t_n); h] = -\frac{h^2}{4}\frac{d^2y(t_{n-1})}{dt^2} + O(h^3)} .$$

The terms in the right-hand-sides of (4.67) and (4.68) are called **local truncation errors** (see p. 56 in Lambert, 1991). It is seen that the principal part of the local truncation error of the θ-method applied with $\boldsymbol{\theta = 0.75}$ is twice smaller than that of the Backward Euler Formula. This completes the proof of the theorem.

■

Theorem 4.3 demonstrates very clearly the fact that one should expect, as stated above, the θ-method with $\boldsymbol{\theta = 0.75}$ to be more accurate than the Backward Euler Formula.

Several experiments were carried out to confirm this expectation. Some of the obtained results are shown in **Table 4.3**. It is seen that the accuracy of the numerical results obtained by using the θ-method with $\boldsymbol{\theta = 0.75}$ is indeed considerably better than that obtained by the Backward Euler Formula (see the figures given in the third and the fifth columns of Table 4.3).

It is remarkable that the accuracy is improved precisely by a factor of two when the time-stepsize becomes sufficiently small and, which is very important, when the influence of the rounding errors in the first sixteen digits is eliminated.

It is not clear how to derive corresponding expressions for the principal parts of the local truncation error when the Richardson Extrapolation is used together with these two numerical methods for solving systems of ODEs (i.e. together with the Backward Euler Formula and with the θ-method with $\boldsymbol{\theta = 0.75}$). Probably the same approach (or at least a similar approach) as that which was used in Theorem 4.3 can be applied to compare the leading terms of the local truncation error also in this case.

The results presented in Table 4.3 show that the accuracy of the calculated approximations is in general improved by a factor, which is **greater than two**, when the θ-method with $\boldsymbol{\theta = 0.75}$ is used as an underlying method instead of the Backward Euler Differentiation Formula.

Job Number	Number of time-steps	Backward Euler Formula		The θ-method with θ = 0.75	
		Direct	Richardson	Direct	Richardson
1	168	2.564E-00	3.337E-01	1.439E-00 (0.561)	3.988E-01 (1.195)
2	336	1.271E-00	1.719E-01	6.701E-01 (0.527)	5.252E-02 (0.306)
3	672	6.227E-01	5.473E-02	3.194E-01 (0.513)	1.503E-03 (0.027)
4	1344	3.063E-01	7.708E-03	1.550E-01 (0.506)	3.787E-03 (0.491)
5	2688	1.516E-01	1.960E-03	7.625E-02 (0.503)	9.502E-04 (0.484)
6	5376	7.536E-02	5.453E-04	3.779E-02 (0.501)	2.384E-04 (0.437)
7	10752	3.757E-02	1.455E-04	1.881E-02 (0.501)	5.980E-05 (0.411)
8	21504	1.876E-02	3.765E-05	9.385E-03 (0.500)	1.499E-05 (0.398)
9	43008	9371E-03	9583E-06	4.687E-03 (0.500)	3.754E-06 (0.392)
10	86016	4.684E-03	2.418E-06	2.342E-03 (0.500)	9.394E-07 (0.389)
11	172032	2.341E-03	6.072E-07	1.171E-03 (0.500)	2.353E-07 (0.388)
12	344064	1.171E-03	1.522E-07	5.853E-04 (0.500)	6.264E-08 (0.411)
13	688128	5.853E-04	3.809E-08	2.926E-04 (0.500)	1.618E-08 (0.425)
14	1376256	2.926E-04	9.527E-09	1.463E-04 (0.500)	4.111E-09 (0.432)
15	2752512	1.463E-04	2.382E-09	7.315E-05 (0.500)	1.036E-09 (0.435)
16	5505024	7.315E-05	5.957E-10	3.658E-05 (0.500)	2.601E-10 (0.437)
17	11010048	3.658E-05	1.489E-10	1.829E-05 (0.500)	6.514E-11 (0.437)
18	22020096	1.829E-05	3.720E-11	9.144E-06 (0.500)	1.628E-11 (0.438)
19	44040192	9.144E-6	9.273E-12	4.572E-06 (0.500)	4.051E-12 (0.437)

Table 4.3

Comparison of the accuracy achieved when the Backward Euler Formula (obtained by using $\boldsymbol{\theta = 1.0}$) and the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ are run with 19 different time-stepsizes. The errors obtained by (4.61) are given in the last four columns in this table. The ratios (the errors obtained when the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ is used divided by the corresponding errors obtained when the Backward Euler Formula is used) are given in brackets.

4.6.5. Comparing the computing times needed to obtain prescribed accuracy

Three time-steps (one large and two small) with the underlying numerical method are necessary when one time-step of the Richardson Extrapolation is performed. This means that if the Richardson Extrapolation and the underlying numerical method are used with the same time-stepsize, then the computational cost of the Richardson Extrapolation will be more than three times greater than that of the underlying numerical method (see the analysis performed in the previous section).

However, the use of the Richardson Extrapolation leads also to an improved accuracy of the calculated approximations (see Table 4.2 and Table 4.3). Therefore, it is not relevant (and not fair either) to compare the Richardson Extrapolation with the underlying method under the assumption that both devices are run with equal number of time-steps. It is much more relevant to investigate how much computational work will be needed in order to achieve the same accuracy in the cases where

(a) the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ is applied directly

and

(b) when the same numerical method is combined with the Richardson Extrapolation.

The computing times needed in the efforts to achieve prescribed accuracy are given in **Table 4.4**. If the desired accuracy is $\mathbf{10^{-k}}$ $(\mathbf{k = 1, 2, \ldots, 11})$, then the computing times achieved in the first run in which the quantity **ERROR** from (4.43) becomes less than $\mathbf{10^{-k}}$ are given in **Table 4.4**. This means that the actual error, found in this way, is in the interval $[\mathbf{10^{-(k+1)}}, \mathbf{10^{-k}})$ when accuracy of order $\mathbf{10^{-k}}$ is required.

Desired Accuracy of the calculated approximations	Application of the θ-method with θ=0.75		Combination with the Richardson Extrapolation	
	CPU time (in hours)	Number of the time-steps	CPU time (in hours)	Number of the time-steps
[1.0E-02, 1.0E-01)	0.0506	2688	0.0614	336
[1.0E-03, 1.0E-02)	0.1469	21504	0.0897	1344
[1.0E-04, 1.0E-03)	1.1242	344032	0.1192	2688
[1.0E-05, 1.0E-04)	6.6747	2752512	0.2458	10752
[1.0E-06, 1.0E-05)	43.0650	22020096	0.6058	43008
[1.0E-07, 1.0E-06)	Required accuracy was not achieved		1.0197	86016
[1.0E-08, 1.0E-07)	Required accuracy was not achieved		3.1219	344064
[1.0E-09, 1.0E-08)	Required accuracy was not achieved		10.3705	1376256
[1.0E-10, 1.0E-09)	Required accuracy was not achieved		35.3331	5505024
[1.0E-11, 1.0E-10)	Required accuracy was not achieved		66.1322	11010048
[1.0E-12, 1.0E-11)	Required accuracy was not achieved		230.2309	44040192

Table 4.4

Comparison of the computational costs (measured by the CPU hours) needed to achieve prescribed accuracy in the cases where **(a)** the θ-method with $\boldsymbol{\theta = 0.75}$ is implemented directly and **(b)** the Richardson Extrapolation is used in combination with the same underlying numerical scheme.

Four important conclusions can immediately be drawn by studying the numerical results that are shown in **Table 4.4**:

- The direct use of the θ-method with $\boldsymbol{\theta = 0.75}$ is slightly more efficient with regard to the computing time than the implementation of the Richardson Extrapolation when the desired accuracy is very low, for example when it is required that **ERROR** from (4.61) should be in the interval $[\mathbf{10^{-2}}, \mathbf{10^{-1}})$; compare the CPU times in the first row of Table 4.4.

- The implementation of the Richardson Extrapolation becomes much more efficient than the direct θ-method with $\boldsymbol{\theta = 0.75}$ when the accuracy requirement is increased (see the second, the third, the fourth and the fifth lines of Table 4.4). If it is desirable to achieve accuracy, which is better than $\mathbf{10^{-5}}$, and more precisely if it is required to have that the **ERROR** from (4.61) should be in the interval $[\mathbf{10^{-6}}, \mathbf{10^{-5}})$, then the computing time spent with the Richardson Extrapolation is more than

$\mathbf{70}$ times smaller than the corresponding computing time for the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ when it is used directly (compare the CPU times in the fifth line of Table 4.4).

- Accuracy better than $\mathbf{10^{-5}}$ has not been achieved in the $\mathbf{19}$ runs with the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.75}$ when it is used directly (see Table 4.4), while even accuracy better than $\mathbf{10^{-11}}$ is achievable when the Richardson extrapolation is used (see the last lines of Table 4.4 and Table 4.2).

- The major conclusion is that not only is the Richardson Extrapolation a powerful tool for improving the accuracy of the underlying numerical method, but it is also extremely efficient with regard to the computational cost (this being especially true when the accuracy requirement is not very low).

4.6.6. Using the Trapezoidal Rule in the computations

Consider the Trapezoidal Rule (which is a special numerical scheme belonging to the class of the $\boldsymbol{\theta}$-methods and can be found from this class by setting $\boldsymbol{\theta = 0.5}$). It has been shown (see Theorem 4.2) that, while the Trapezoidal Rule itself is a second-order **A-stable** numerical method, its combination with the **active implementation** of the Richardson Extrapolation is **not** an A-stable numerical method. However the passive implementation of the Richardson Extrapolation together with the Trapezoidal Rule is remaining A-stable. Now we shall use the atmospheric chemical scheme to confirm experimentally these facts. More precisely,

(a) we shall investigate whether the Trapezoidal Rule behaves as a second-order numerical method when it is directly applied in the solution of the atmospheric chemical scheme,

(b) we shall show that the results are unstable when this numerical method is combined with the active implementation of the Richardson Extrapolation

and

(c) we shall verify the fact that the results remain stable when the Trapezoidal Rule is combined with the passive implementation of the Richardson Extrapolation.

Numerical results are presented in **Table 4.5** . Several important conclusions can be drawn from the results shown in this table (it should be mentioned here that many other runs were also performed and the conclusions were similar):

Job Number	Number of steps	Direct Implementation		Richardson Extrapolation			
				Active		Passive	
		Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
1	168	3.605E-01	-	Unstable	n.a.	4.028E-02	-
2	336	7.785E-02	4.631	Unstable	n.a.	3.246E-03	12.407
3	672	1.965E-02	3.961	Unstable	n.a.	1.329E-03	2.443
4	1344	4.915E-03	3.998	Unstable	n.a.	1.462E-04	9.091
5	2688	1.228E-03	4.001	Unstable	n.a.	5.823E-05	2.510
6	5376	3.071E-04	4.000	Unstable	n.a.	3.765E-05	1.547
7	10752	7.677E-05	4.000	Unstable	n.a.	2.229E-05	1.689
8	21504	2.811E-05	2.731	Unstable	n.a.	1.216E-05	1.833
9	43008	1.615E-05	1.741	Unstable	n.a.	6.300E-06	1.930
10	86016	8.761E-06	1.843	Unstable	n.a.	3.188E-06	1.976
11	172032	4.581E-06	1.912	Unstable	n.a.	1.600E-06	1.993
12	344064	2.345E-06	1.954	Unstable	n.a.	8.007E-07	1.998
13	688128	1.187E-06	1.976	Unstable	n.a.	4.005E-07	1.999
14	1376256	5.970E-07	1.988	Unstable	n.a.	2.002E-07	2.000
15	2752512	2.994E-07	1.994	Unstable	n.a.	1.001E-07	2.000
16	5505024	1.499E-07	1.997	Unstable	n.a.	5.005E-08	2.000
17	11010048	7.503E-08	1.998	Unstable	n.a.	2.503E-08	2.000
18	22020092	3.753E-08	1.999	Unstable	n.a.	1.252E-08	2.000
19	44040192	1.877E-08	2.000	Unstable	n.a.	6.257E-09	2.000

Table 4.5

Numerical results obtained in 19 runs of **(i)** the direct implementation of the Trapezoidal Rule, **(ii)** the Active Richardson Extrapolation with the Trapezoidal Rule and **(iii)** the Passive Richardson Extrapolation with the Trapezoidal Rule are given. The errors obtained by (4.61) are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate". "Unstable" means that the code detected that the computations are not stable, while "n.a." stands for not applicable.

(a) The order of the Trapezoidal Rule is two. Therefore, it should be expected that doubling the number $\mathbf{N}$ of time-steps, which leads to a decrease of the time-stepsize $\mathbf{h} = (\mathbf{129600} - \mathbf{43200})/\mathbf{N} = \mathbf{86400}/\mathbf{N}$ by a factor of two, will in general lead to an improvement of the accuracy by a factor of **four**. It is seen that in the beginning this is the case. However, after the seventh run the convergence rates are quickly shifting from four to two. It is not clear why the rate of convergence is deteriorated and the method behaves as a first-order numerical scheme for small time-stepsizes.

(b) The application of the Active Richardson Extrapolation with the Trapezoidal Rule leads to unstable computations. As mentioned above this is a consequence of Theorem 4.2. It is only necessary to explain here how the instability is detected. Two stability checks are carried out. The first check is based on monitoring the norm of the calculated approximate solutions: if this norm becomes $\mathbf{10^{10}}$ times greater than the norm of the initial vector, then the computations are stopped and the computational process is declared to be unstable. The second check is based on the convergence of the Newton Iterative Process. If this process is not convergent or very slowly convergent at some time-step $\mathbf{n}$, then

the stepsize $\mathbf{h}$ is halved. This can happen several times at the time-step $\mathbf{n}$. If the reduced time-stepsize becomes less than $\mathbf{10^{-5}h}$, then the computational process is stopped and declared to be unstable. If the time-stepsize is reduced at time-step $\mathbf{n}$, then the remaining calculations from $\mathbf{t_{n-1}}$ to $\mathbf{t_n}$ are performed with the reduced time-stepsize (with the reduced time-stepsizes, if the time-stepsize has been reduced several times), however an attempt is carried out to perform the next time-step $\mathbf{n+1}$ (i.e. to proceed from $\mathbf{t_{n-1}}$ to $\mathbf{t_n}$) with the time-stepsize $\mathbf{h = (129600 - 43200)/N = 86400/N}$ that is used in the current run $\mathbf{j}$ where $\mathbf{j = 1, 2, \ldots, 19}$.

(c) The order of the Passive Richardson Extrapolation with the Trapezoidal Rule should be three. Therefore, it should be expected that doubling the number $\mathbf{N}$ of time-steps, which leads to a decrease of the time-stepsize $\mathbf{h = (129600 - 43200)/N = 86400/N}$ by a factor of two, will in general lead to an improvement of the accuracy by a factor of eight. It is seen from Table 4.2 that this is not the case, the convergence rates are increased by a factor of two only and, therefore, the Trapezoidal Rule combined with the passive implementation of the Richardson Extrapolation behaves as a first-order numerical scheme (excepting perhaps, to some degree, the first three runs). However, it is also seen that the Passive Richardson Extrapolation combined with the Trapezoidal Rule is a stable method and gives consistently more accurate results than those obtained when the Trapezoidal Rule is applied directly. It should be mentioned here that the combination of the Backward Differentiation Formula with the Richardson Extrapolation behaves (as it should) as a second-order numerical scheme (see, **Faragó, Havasi and Zlatev, 2010)**.

4.7. Using Implicit Runge-Kutta Methods

The use of the Richardson Extrapolation together with numerical schemes from the class of the $\boldsymbol{\theta}$-methods, which are very often used by scientists and engineers, was studied in detail in the previous sections of this chapter. Some results about the application of Implicit Runge-Kutta Methods together with the Richardson Extrapolation will be presented in this section. The emphasis will again be on the stability properties of the combined methods.

4.7.1. Fully Implicit Runge Kutta Methods

Implicit Runge-Kutta Methods can be introduced by the following formulae:

$$\mathbf{y_n = y_{n-1} + h \sum_{i=1}^{m} c_i k_i^n} \quad . \tag{4.69}$$

The coefficients $\mathbf{c_i}$ are given constants (the requirement to achieve at least first-order of accuracy implies that the sum of the coefficients $\mathbf{c_i}$ should be equal to one) , while at an arbitrary time-step $\mathbf{n}$ the stages $\mathbf{k_i^n}$ are defined by

$$(4.70) \qquad \mathbf{k_i^n = f\left(t_{n-1} + h\, a_i\,,\ y_{n-1} + h \sum_{j=1}^{m} b_{ij}\, k_j^n \right)\,, \qquad i = 1, 2, 3, \ldots, m\,,}$$

with

$$(4.71) \qquad \mathbf{a_i = \sum_{j=1}^{m} b_{ij}\ ,} \qquad \mathbf{i = 1, 2, 3, \ldots, m\ ,}$$

where $\mathbf{b_{ij}}$ are also some given constants depending on the particular numerical method.

Many alternative, but in some cases equivalent, formulations can be found in **Butcher (2003)**, in **Hairer and Wanner (1991)** or in **Hundsdorfer and Verwer (2003)**.

The vectors $\mathbf{k_i^n}$ participating in the right-hand-side of (4.69) and defined in (4.70) are called stages as in Chapter 2. Each of these vectors consists of $\mathbf{s}$ components where $\mathbf{s}$ is the number of equations in the system of ODEs, defined by (1.1) and (1.2). The numerical method defined by the equalities (4.69)-(4.71) is an implicit **m**–stage Runge-Kutta numerical scheme (the term "fully implicit" is often used and it is also adopted here). The implicitness arises in (4.70), because the stage vectors $\mathbf{k_i^n}$ appear in both sides of these $\mathbf{m}$ relationships. This means that at every time-step we have to solve a system of $\mathbf{ms}$ equations, which is in general non-linear.

The major advantages of the Fully Implicit Runge-Kutta (FIRK) Methods are two:

(a) these methods are very accurate

and

(b) numerical schemes from this class, which have very good stability properties, can be derived.

The major drawback of the Fully Implicit Runge-Kutta Methods is caused by the necessity to handle systems consisting of $\mathbf{ms}$ algebraic equations at every time-step, which are in many cases non-linear. These systems can be enormously big when $\mathbf{s}$ is a large integer. Serious computational problems arise when this is true. Therefore, an attempt to design simplified numerical methods, which are based on the same basic idea, but are not so expensive, is worthwhile. Such methods were developed and will be discussed in the next sub-section.

4.7.2. Diagonally Implicit Runge-Kutta Methods

It was emphasized in the previous sub-section that if the system of ODEs defined by (1.1) and (1.2) is large, i.e. if $\mathbf{s}$ is large, then the solution of the system (4.70) becomes very time-consuming even when high speed modern computers are available and efficiently used. Therefore, it is necessary to find some way of reducing the computational complexity of the Fully Implicit Runge-Kutta Methods. This can be done by applying the approach proposed by **Alexander (1977)**. The truth is that there are several other works as, for example, an institutional report written by **Nørsett (1974)** and a conference proceedings paper written by **Cruziex (1976)**, where this approach or at least a very similar approach is also introduced. The methods, which were developed by Nørsett, Cruziex and Alexander as well as by some other scientists, form the class of **Diagonally Implicit Runge-Kutta Methods (DIRK Methods)**. The name "diagonally implicit" has been introduced first by R. Alexander, but there is some confusion related to these methods, because some authors use occasionally the terms "semi-implicit methods" or "semi-explicit methods" instead of diagonally implicit methods. It should be pointed out here that these two classes of methods are in some sense similar but not the same as the class of the Diagonally Implicit Runge-Kutta Methods.

The DIRK Methods are based on the following formulae:

$$\mathbf{y_n = y_{n-1} + h \sum_{i=1}^{m} c_i k_i^n} \quad . \tag{4.72}$$

The coefficients $\mathbf{c_i}$ are again, as in the previous sub-section, given constants (the requirement to achieve at least first-order of accuracy implies that the sum of the coefficients $\mathbf{c_i}$ should be equal to one) , while the stages $\mathbf{k_i^n}$ are defined at an arbitrary time-step $\mathbf{n}$ by

$$\mathbf{k_i^n = f\left(t_{n-1} + h\, a_i\,,\ y_{n-1} + h \sum_{j=1}^{i-1} b_{ij}\, k_j^n + \gamma k_i^n \right)}\,, \qquad \mathbf{i = 1, 2, 3, \ldots, m}\,, \tag{4.73}$$

with

$$\mathbf{a_i = \sum_{j=1}^{i-1} b_{ij} + \gamma} \quad , \qquad \mathbf{i = 1, 2, 3, \ldots, m} \quad , \tag{4.74}$$

where $\mathbf{b_{ij}}$ and $\boldsymbol{\gamma}$ are also some given constants depending on the particular numerical method.

The sums in (4.73) and (4.74) are by definition assumed to be equal to zero when the upper index is less than the lower one.

It is immediately seen that (4.72) is the same as (4.69), but the equalities presented in (4.73) is different from those given in (4.70) and the difference is indeed very essential. While the relations (4.70) have to be handled as a large and in the general case non-linear system of $\mathbf{ms}$ algebraic equations that has to be solved at every time-step, (4.73) leads to the solution of $\mathbf{m}$ systems, each of them consisting of $\mathbf{s}$ equations. Indeed, when $\mathbf{i = 1}$, the first of the $\mathbf{m}$ formula (4.73) contains only one unknown vector, vector $\mathbf{k_1^n}$. If the first vector $\mathbf{k_1^n}$ is found by solving (4.73) for $\mathbf{i = 1}$, then the second of the $\mathbf{m}$ formulae (4.73), which is obtained for $\mathbf{i = 2}$, will contain only one unknown vector, vector $\mathbf{k_2^n}$. If this vector is also found, then for $\mathbf{i = 3}$ the third of the $\mathbf{m}$ formula (4.73) will contain only one unknown vector, vector $\mathbf{k_3^n}$. It is clear that continuing the computations in this way, we shall be able to obtain all stage vectors $\mathbf{k_i^n}$, $\mathbf{i = 1, 2, 3, \ldots, m}$, by treating successively a sequence of $\mathbf{m}$ smaller systems, each of them containing $\mathbf{s}$ equations.

A reasonable question can be asked here:

How significant is the reduction of the computational work made in the transition from Fully Implicit Runge-Kutta Methods to Diagonally Implicit Runge-Kutta Methods?

We shall try to answer this question in the next sub-section.

4.7.3. Evaluating the reduction of the computational cost when DIRK Methods are used

The following example indicates that the reduction obtained when a Diagonally Implicit Runge-Kutta (DIRK) Method is used instead of a corresponding Fully Implicit Runge-Kutta (FIRK) Method could indeed be rather large. Here "corresponding" means that the number of stage vectors used in the DIRK Method and the FIRK Method is the same.

Consider a system of ODEs defined by (1.1) and (1.2), which is **linear** and for which the following relationships hold:

$$(\mathbf{4.75}) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{A(t)\,y}, \qquad \mathbf{t} \in [\mathbf{a}, \mathbf{b}], \qquad \mathbf{y} \in \mathbf{D} \subset \mathbb{R}^{\mathbf{s}}, \qquad \mathbf{s} \geq \mathbf{1}, \qquad \mathbf{y(a) = \eta}, \qquad \boldsymbol{\eta} \in \mathbf{D}\ .$$

It is additionally assumed that $\mathbf{A(t)} \in \mathbb{R}^{\mathbf{sxs}}$ is a given real matrix, which depends on the independent variable $\mathbf{t}$ and that $\boldsymbol{\eta} \subset \mathbb{R}^{\mathbf{s}}$ is some given real vector with $\mathbf{s}$ components. Then linear systems of algebraic equations have to be solved at every time-step when either a Fully Implicit Runge-Kutta Method or a Diagonally Implicit Runge-Kutta Method is applied.

If a Fully Implicit Runge-Kutta Method is used in the solution of (4.75), then the coefficient matrix $\overline{\mathbf{B}}$ of the linear system, which is represented by formula (4.70), is dependent on matrix $\mathbf{A(t)}$ and it can be partitioned into $\mathbf{m \times m}$ blocks. Each of these blocks is a $\mathbf{s \times s}$ square matrix. It can be

established that any of the off-diagonal blocks of matrix $\bar{\mathbf{B}}$ can be expressed by the formulae $\bar{\mathbf{B}}_{\mathbf{ij}} = -\mathbf{hb_{ij}A}(\mathbf{t_{n-1}} + \mathbf{ha_i})$, $\mathbf{i = 1,\ 2, \ldots,\ m}$, $\mathbf{j = 1,\ 2,\ \ldots,\ m}$, $\mathbf{i \neq j}$, while the diagonal blocks are given by $\bar{\mathbf{B}}_{\mathbf{ii}} = \mathbf{I} - \mathbf{hb_{ii}A}(\mathbf{t_{n-1}} + \mathbf{ha_i})$, $\mathbf{j = 1,\ 2,\ \ldots,\ m}$, where $\mathbf{I}$ is the identity matrix in $\mathbb{R}^{\mathbf{sxs}}$.

Assume furthermore that $\mathbf{s}$ is large and that the matrix $\mathbf{A(t)}$ is **dense**. It is clear that matrix $\bar{\mathbf{B}}$ will also be dense and the number of arithmetic operations that are to be performed during the most time-consuming process in the solution of the system (4.70), i.e. during the factorization of matrix $\bar{\mathbf{B}}$, will be $\mathbf{O(m^3 s^3)}$.

If a Diagonally Implicit Runge-Kutta (DIRK) Method is to be used in the solution of (4.75), then the coefficient matrix $\hat{\mathbf{B}}$ of the linear system that is represented by (4.73) is also dependent on matrix $\mathbf{A(t)}$ and can again be partitioned into $\mathbf{m \times m}$ blocks, each of the diagonal blocks being a $\mathbf{s \times s}$ square matrix. However, all elements in the blocks, which are over the main diagonal of the partitioned matrix $\hat{\mathbf{B}}$ are now equal to zero. This means that the coefficient matrix of the systems defined by (4.73) is a lower block-diagonal matrix. The sub-diagonal blocks of matrix $\hat{\mathbf{B}}$ are given by the formulae $\hat{\mathbf{B}}_{\mathbf{ij}} = -\mathbf{hb_{ij}A}(\mathbf{t_{n-1}} + \mathbf{ha_i})$, $\mathbf{i = 2,\ 3, \ldots,\ m}$, $\mathbf{j = 1,\ 2,\ \ldots,\ m}$, $\mathbf{i > j}$, while the diagonal blocks are $\hat{\mathbf{B}}_{\mathbf{ii}} = \mathbf{I} - \mathbf{h\gamma A}(\mathbf{t_{n-1}} + \mathbf{ha_i})$, $\mathbf{i = 1,\ 2, \ldots,\ m}$), where $\mathbf{I}$ is again the identity matrix in $\mathbb{R}^{\mathbf{sxs}}$. The above statements show clearly that now it is necessary to treat a sequence of $\mathbf{m}$ systems of dimension $\mathbf{s}$ instead of one large system of dimension $\mathbf{ms}$ as was the case when Fully Implicit Runge-Kutta Methods are used and (4.70) is to be handled. Therefore, the computational cost of the factorization of the coefficient matrix in (4.73) will be $\mathbf{O(ms^3)}$, i.e. a reduction of order $\mathbf{O(m^2)}$ is achieved when the coefficient matrix of the linear system (4.73) is factorized instead of the coefficient matrix of the linear system (4.70). It is quite clear that this reduction will be rather substantial when $\mathbf{s}$ is a large integer.

It should be noted that a very special example has been discussed above (it was assumed that the coefficient matrix is dense). However, the situation will not change too much when other examples are to be treated. Assume, for instance, that matrix **A(t)** from (4.75) has some special property and this property is to be exploited in the treatment of both (4.70) and (4.73). In order to be more specific, let us assume that **A(t)** is a banded matrix. Then the situation will not be improved because the bandwidth of the diagonal blocks $\hat{\mathbf{B}}_{\mathbf{ii}}$ in (4.73) remain the same as that of **A(t)**, while the bandwidth of matrix $\bar{\mathbf{B}}$ emerging from (4.70) will become much wider. This will lead to a very significant increase of the number of arithmetic operations not only because this matrix is much bigger than the diagonal blocks $\hat{\mathbf{B}}_{\mathbf{ii}}$ of (4.73), which have to be treated when the Diagonally Implicit Runge-Kutta Method is used, but also because much more non-zero elements will be created during the factorization of matrix $\bar{\mathbf{B}}$, as a consequence of the fact that its bandwidth is much broader. Thus, the reduction of the computational work obtained when some DIRK Method is used instead of a corresponding Runge-Kutta Method will in most of the cases become even greater than the reduction achieved when matrix **A(t)** is dense.

It is necessary to explain why a requirement that all coefficients $\mathbf{a_{ii}}$, $\mathbf{i = 1,\ 2, \ldots,\ m}$, should be equal to $\boldsymbol{\gamma}$ is imposed for the Diagonally Implicit Runge-Kutta Methods. Consider now the general non-linear case and assume that some version of the Newton Iterative Method is to be used in the successive solution of the $\mathbf{m}$ systems that that appear in (4.73). Each of these systems contain $\mathbf{s}$ non-linear algebraic equations and one have to use the **LU** factorizations of the shifted Jacobian matrices $\mathbf{J_i = I - h\gamma(\partial f/\partial k_i^n)}$. The expectation is that if $\mathbf{a_{ii} = \gamma}$, $\mathbf{i = 1,\ 2, \ldots,\ m}$, then it will be

sufficient to factorize only the first shifted Jacobian matrix and then to use the same **LU** factorization during the solution of all remaining systems. If the attempt to apply the same **LU** factorization in the treatment of all **m** systems in (4.73) is successful, then the reduction will indeed be very significant, but one will be able to use the same **LU** factorization for all systems, after the first one, only when the shifted Jacobian matrix is slowly varying in **t** . However, it should be noted that a rather substantial reduction will be achieved when a Diagonally Implicit Runge-Kutta Method is used instead of the corresponding Fully Implicit Runge-Kutta Method even if the requirement for slow variation of the shifted Jacobian matrix in **t** is not satisfied. Finally, let us mentioned here that the methods are called semi-implicit Runge-Kutta methods if it is not assumed that $\mathbf{a_{ii} = \gamma}, \quad \mathbf{i = 1, 2, \dots, m}$.

4.7.4. Applying Richardson Extrapolation for Fully Implicit Runge-Kutta Methods

The Richardson Extrapolation can be implemented in relation to Fully Implicit Runge-Kutta methods in the same way as it was implemented for general methods for solving systems of ODEs in Chapter 1 and for Explicit Runge-Kutta methods in Chapter 2. Consider any Fully Implicit Runge-Kutta Method defined by the formulae (4.69)-(4.71). Assume that the calculations have already been performed for all grid-points $\mathbf{t_i}$, $(\mathbf{i = 1, 2, \dots, n - 1})$ by using some numerical method, the order of accuracy of which is **p .** If approximations $\mathbf{y_i \approx y(t_i)}$ of the exact solution are available (i.e. these approximations have already been calculated at the grid-points $\mathbf{t_i}$, $(\mathbf{i = 0, 1, 2, \dots, n - 1})$, then **three actions** are to be carried out successively in order to obtain the next approximation $\mathbf{y_n}$ **:**

(a) Perform one large time-step, with a time-stepsize **h** when the grid (1.6) is used or with a time-stepsize $\mathbf{h_n}$ if the grid (1.7) has been selected, in order to calculate an approximation $\mathbf{z_n}$ of $\mathbf{y(t_n)}$.

(b) Perform two small time-steps, with a time-stepsize $\mathbf{0.5\,h}$, when the grid (1.6) is used or with a time-stepsize $\mathbf{0.5\,h_n}$ if the grid (1.7) has been selected, in order to calculate another approximation $\mathbf{w_n}$ of $\mathbf{y(t_n)}$.

(c) calculate an improved approximation $\mathbf{y_n}$ by applying the formula:

$$\mathbf{y_n = \frac{2^p w_n - z_n}{2^p - 1}}. \qquad (4.76)$$

As was pointed out in Chapter 1 and Chapter 2, the above algorithm is applicable to **any** numerical method for solving systems of ODEs (in Chapter 6 it will be shown that it is also applicable, after introducing some additional requirements, when some systems of PDEs are to be handled). There are **only** two requirements:

(A) The **same** numerical method should be used in the calculation of the two approximations $\mathbf{z_n}$ and $\mathbf{w_n}$ **.**

(B) The **order** of the selected numerical method should be $\mathbf{p}$. This second requirement is utilized in the derivation of formula (1.8), in which the positive integer $\mathbf{p}$ is involved; this will be done in the next section.

It is clear, see Chapter 1 and Chapter 2 as well as **Zlatev, Faragó and Havasi (2010, 2012)**, that that order of accuracy of the improved approximation $\mathbf{y_n}$ will be at least $\mathbf{p+1}$.

If the stability properties of the Richardson Extrapolation are to be studied, then it will be worthwhile, as we did in Chapter 2, to apply both the Fully Implicit Runge-Kutta Method formulated with (4.69)-(4.71) and the Richardson Extrapolation associated with this computational scheme in the solution of the Dahlquist scalar and linear test-equation:

$$(4.77) \qquad \frac{dy}{dt} = \lambda y, \quad t \in [0,\infty], \quad y \in \mathbb{C}, \quad \lambda = \bar{\alpha} + \bar{\beta} i \in \mathbb{C}, \quad \bar{\alpha} \le 0, \quad y(0) = \eta .$$

It is appropriate now to select some particular Fully Implicit Runge-Kutta Method in order to facilitate the explanation of the results. The method, which was originally proposed in **Ehle (1968)**, see also **Hairer, Nørsett and Wanner (1987)** or **Hairer and Wanner (1991)**, and which is based on the formulae listed below, is used:

$$(4.78) \qquad k_1^n = f\left(t_{n-1} + \frac{4-\sqrt{6}}{10}h,\ y_{n-1} + \frac{88-7\sqrt{6}}{360}hk_1^n + \frac{296-169\sqrt{6}}{1800}hk_2^n + \frac{-2+3\sqrt{6}}{225}hk_3^n\right),$$

$$(4.79) \qquad k_2^n = f\left(t_{n-1} + \frac{4+\sqrt{6}}{10}h,\ y_{n-1} + \frac{296+169\sqrt{6}}{1800}hk_1^n + \frac{88+7\sqrt{6}}{360}hk_2^n + \frac{-2-3\sqrt{6}}{225}hk_3^n\right),$$

$$(4.80) \qquad k_3^n = f\left(t_{n-1} + h,\ y_{n-1} + \frac{16-\sqrt{6}}{36}hk_1^n + \frac{16+\sqrt{6}}{36}hk_2^n + \frac{1}{9}hk_3^n\right),$$

$$(4.81) \qquad y_n = \frac{16-\sqrt{6}}{36}hk_1^n + \frac{16+\sqrt{6}}{36}hk_2^n + \frac{1}{9}hk_3^n .$$

This method is a very well-known three-stage five-order L-stable Fully Implicit Runge-Kutta method (see again the above references) and can be found in many text books on numerical methods for systems of ODEs.

If this algorithm is applied in the solution of the Dahlquist test-problem (4.77), then the following relationship (see **Hairer and Wanner (1991)**, p. 42) can be obtained after the elimination of the quantities $\mathbf{k_1^n}$, $\mathbf{k_2^n}$ and $\mathbf{k_3^n}$:

$$(\mathbf{4.82}) \qquad \mathbf{y_n} = \left(\frac{\frac{\mathbf{1}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{2}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}}{-\frac{\mathbf{1}}{\mathbf{60}}\boldsymbol{\nu}^3 + \frac{\mathbf{3}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{3}}{\mathbf{5}}\boldsymbol{\mu} + \mathbf{1}} \right) \mathbf{y_{n-1}} = \left(\frac{\frac{\mathbf{1}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{2}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}}{-\frac{\mathbf{1}}{\mathbf{60}}\boldsymbol{\nu}^3 + \frac{\mathbf{3}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{3}}{\mathbf{5}}\boldsymbol{\mu} + \mathbf{1}} \right)^{\mathbf{n}} \mathbf{y_0} \ .$$

The parameter $\boldsymbol{\nu}$ is equal as usual to the product $\mathbf{h}\,\boldsymbol{\lambda}$.

From (4.82) it follows that the stability function of the Fully Implicit Runge-Kutta Method defined by the equalities (4.78)-(4.81) is determined by the following expression:

$$(\mathbf{4.83}) \qquad \mathbf{R}(\boldsymbol{\nu}) = \frac{\frac{\mathbf{1}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{2}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}}{-\frac{\mathbf{1}}{\mathbf{60}}\boldsymbol{\nu}^3 + \frac{\mathbf{3}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{3}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}} \ .$$

Let us turn back to the three actions mentioned in the beginning of this section. It is clear that, if the Dahlquist test-example (4.77) is solved, then we can write:

$$(\mathbf{4.84}) \qquad \mathbf{z_n} = \left(\frac{\frac{\mathbf{1}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{2}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}}{-\frac{\mathbf{1}}{\mathbf{60}}\boldsymbol{\nu}^3 + \frac{\mathbf{3}}{\mathbf{20}}\boldsymbol{\nu}^2 - \frac{\mathbf{3}}{\mathbf{5}}\boldsymbol{\nu} + \mathbf{1}} \right) \mathbf{y_{n-1}} = \mathbf{R}(\boldsymbol{\nu})\, \mathbf{y_{n-1}} \ .$$

and

$$(\mathbf{4.85}) \qquad \mathbf{w_n} = \left(\frac{\frac{\mathbf{1}}{\mathbf{20}}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)^2 - \frac{\mathbf{2}}{\mathbf{5}}\frac{\boldsymbol{\nu}}{\mathbf{2}} + \mathbf{1}}{-\frac{\mathbf{1}}{\mathbf{60}}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)^3 + \frac{\mathbf{3}}{\mathbf{20}}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)^2 - \frac{\mathbf{3}}{\mathbf{5}}\frac{\boldsymbol{\nu}}{\mathbf{2}} + \mathbf{1}} \right)^{\mathbf{2}} \mathbf{y_{n-1}} = \left[\mathbf{R}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)\right]^{\mathbf{2}} \mathbf{y_{n-1}} \ .$$

Substitute now the last terms of the above equalities in (4.76) and use the fact that $\mathbf{p = 5}$ for the numerical method defined by (4.78)-(4.81). The results is:

$$(4.86) \qquad y_n = \frac{32\left[R\left(\frac{\nu}{2}\right)\right]^2 - R(\nu)}{31} y_{n-1} .$$

The last equality shows that the new method, consisting of the combination of the Fully Explicit Runge-Kutta Method defined by (4.78)-(4.81), is a one-step method and its stability function $\bar{R}(\nu)$ can be expressed by the stability function $R(\nu)$ of the underlying method by the following formula:

$$(4.87) \qquad \bar{R}(\nu) = \frac{32\left[R\left(\frac{\nu}{2}\right)\right]^2 - R(\nu)}{31} .$$

It should be mentioned here that similar results that are related to the stability functions for numerical methods, which are combinations of the Richardson Extrapolation with schemes for solving systems of ODEs, were obtained in Chapter 2 in connection with Explicit Runge-Kutta Methods and in the previous sections of this chapter in connection with the class of the θ-methods.

4.7.5. Applying Richardson Extrapolation for Diagonally Implicit Runge-Kutta Methods

The same rules, as those discussed in the previous section, can be used in the implementation of the Richardson Extrapolation for DIRK methods, i.e. after the calculation of the approximations z_n and w_n, the improved by applying the Richardson extrapolation value y_n can again be obtained from (6). It is again appropriate to select a particular method. We have chosen one of the two methods listed on p. 196 in **Lambert (1991)**:

$$(4.88) \qquad k_1^n = f\left(t_{n-1} + \frac{3 \pm \sqrt{3}}{6} h,\ y_{n-1} + \frac{3 \pm \sqrt{3}}{6} h k_1^n\right),$$

$$(4.89) \qquad k_2^n = f\left(t_{n-1} + \frac{3 \mp \sqrt{3}}{6} h,\ y_{n-1} + \frac{\mp\sqrt{3}}{3} h k_1^n + \frac{3 \pm \sqrt{3}}{6} h k_2^n\right),$$

$$(4.90) \qquad y_n = \frac{1}{2} h k_1^n + \frac{1}{2} h k_2^n .$$

The two methods, any of them can be obtained from the above formulae by selecting one of the two alternative signs, are two-stage third-order Diagonally Implicit Runge-Kutta Methods. The stability functions of these methods are given by

$$(4.91) \qquad \mathbf{R}(\boldsymbol{\nu}) = \frac{-\frac{\mathbf{1} \pm \sqrt{\mathbf{3}}}{\mathbf{6}}\boldsymbol{\nu}^{\mathbf{2}} \mp \frac{\sqrt{\mathbf{3}}}{\mathbf{3}}\boldsymbol{\nu} + \mathbf{1}}{\frac{\mathbf{2} \pm \sqrt{\mathbf{3}}}{\mathbf{6}}\boldsymbol{\nu}^{\mathbf{2}} - \frac{\mathbf{3} \pm \sqrt{\mathbf{3}}}{\mathbf{3}}\boldsymbol{\nu} + \mathbf{1}} .$$

The stability function of the combined numerical method, the method obtained by using (18)-(20) together with the Richardson Extrapolation, is given by

$$(4.92) \qquad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{8}\left[\mathbf{R}\left(\frac{\boldsymbol{\nu}}{\mathbf{2}}\right)\right]^{\mathbf{2}} - \mathbf{R}(\boldsymbol{\nu})}{\mathbf{7}} .$$

This function will be used to investigate the stability properties of the numerical method based on the combination of the **active** Richardson Extrapolation and the Diagonally Implicit Runge-Kutta Method described by (4.88)-(4.98).

4.7.6. Stability results related to the active Richardson Extrapolation

The three-stage fifth-order Fully Implicit Runge-Kutta Method defined by (4.78)-(4.81) is L-stable. The two-stage third-order Diagonally Implicit Runge-Kutta Method described by the formulae (4.88)-(4-90) is A-stable when the upper of the alternative signs is selected (and it will be assumed in the remaining part of this chapter that precisely this choice is made).

The above statements are telling us that the computational process will remain stable when any of these two methods, the FIRK Method and the DIRK Method, is used **directly** in the solution of the Dahlquist test-equation (7). The same is true for the **passive** Richardson Extrapolation.

We did not succeed to establish that the combinations of the **active** Richardson Extrapolation with the two selected above numerical methods have the same stability properties (L-stability and A-stability respectively) as the underlying methods. This means that there is no guarantee that the two stability functions from (17) and (22) satisfy the inequality $|\bar{\mathbf{R}}(\boldsymbol{\nu})| \leq \mathbf{1}$ for all values of $\boldsymbol{\nu} = \bar{\boldsymbol{\alpha}} + \bar{\boldsymbol{\beta}}\mathbf{i}$ when $\bar{\boldsymbol{\alpha}} \leq \mathbf{0}$. However, the requirement for A-stability (as well as the requirement for the stronger concept of L-stability) is **only a sufficient condition** for the preservation of the stability of the computational process. It is not necessary and could successfully be replaced by a requirement that the absolute stability region of the selected method is in some sense **very large** (absolute stability regions are discussed in detail, for example, in **Lambert (1991)**; see also the previous chapters).

We succeeded to establish that the important inequality $|\bar{\mathbf{R}}(\boldsymbol{\nu})| \leq \mathbf{1}$ is satisfied, for both of the two selected numerical methods, in a **huge square domain** with vertices: $(\mathbf{0.0}, \mathbf{0.0})$, $(\mathbf{0.0}, \mathbf{10^5\,i})$, $(\mathbf{-10^5\,i}, \mathbf{10^5\,i})$ and $(\mathbf{-10^5}, \mathbf{0.0})$. The parts of the domains, in which the active Richardson Extrapolation is **absolutely stable** when it is combined with the three-stage fifth-order Fully Implicit Runge-Kutta Method and the two-stage third-order Diagonally Implicit Runge-Kutta Method, are

given in **Fig. 4.4** and **Fig. 4.5**. The procedure used to obtain these large parts of the absolute stability regions of the two selected methods was nearly the same as that applied in the previous chapters, see also **Zlatev, Georgiev and Dimov (2014)**, and can be described in the following way. Assume that ν is equal to $\bar{\alpha} + \bar{\beta}\mathbf{i}$ with $\bar{\alpha} \leq 0$ and select some small positive increment ε (in our runs the value $\varepsilon = 0.01$ was chosen). Start the computational procedure by setting the real part $\bar{\alpha}$ of the complex number $\nu = \bar{\alpha} + \bar{\beta}\mathbf{i}$ equal to **zero** and calculate successively the values of the stability functions (4.87) and (4.92) for $\bar{\alpha} = 0$ and for $\bar{\beta} = 0,\ \varepsilon,\ 2\varepsilon,\ 3\varepsilon,\ \ldots$. Continue this process as long as $\bar{\beta}$ becomes equal to 10^5 under the condition that $|\bar{R}(\nu)|$ stays less than one during all these computations. Repeat successively the same procedure for a sequence of new values of $\bar{\alpha}$ that are equal to $-\varepsilon,\ -2\varepsilon, -3\varepsilon,\ \ldots$ Continue to decrease the parameter $\bar{\alpha}$ until it becomes equal to -10^5 (again under the requirement that $|\bar{R}(\nu)|$ stays always less than one). It is clear that one should expect that all points within the squares plotted in **Fig. 4.4** and **Fig. 4.5**, obtained by applying the above algorithm, belong to the absolute stability regions of the studied numerical methods.

It is obvious that the applied approach is very robust, but, on the other hand, it is computationally very expensive. One must calculate the values of each of the stability functions from (4.87) and (4.92) at 10^{14} points (i.e. a lot of computations in **complex arithmetic** have to be performed!). However, this task is not extremely difficult if modern high-speed computers are available. Moreover, the process can be parallelized efficiently and in a very easy way, because the computations along the vertical lines for each value of parameter $\bar{\alpha}$ are independent of the computations related to the other values of this parameter.

Only the parts of the stability regions located above the real axis are drawn in **Fig. 4.4** and **Fig. 4.5**. This is quite sufficient, because the stability regions of the numerical methods for solving systems of ODEs are symmetric with regard to the real axis.

We must emphasize here that we do not claim that the two methods are unstable outside the domains shown in the two figures. We succeeded to establish that the inequality $|\bar{R}(\nu)| \leq 1$ is satisfied in these domains, but it might be satisfied, and it is most probably satisfied, also for some points outside these domains (even for the whole part of the complex plane located to the left of the imaginary axis).

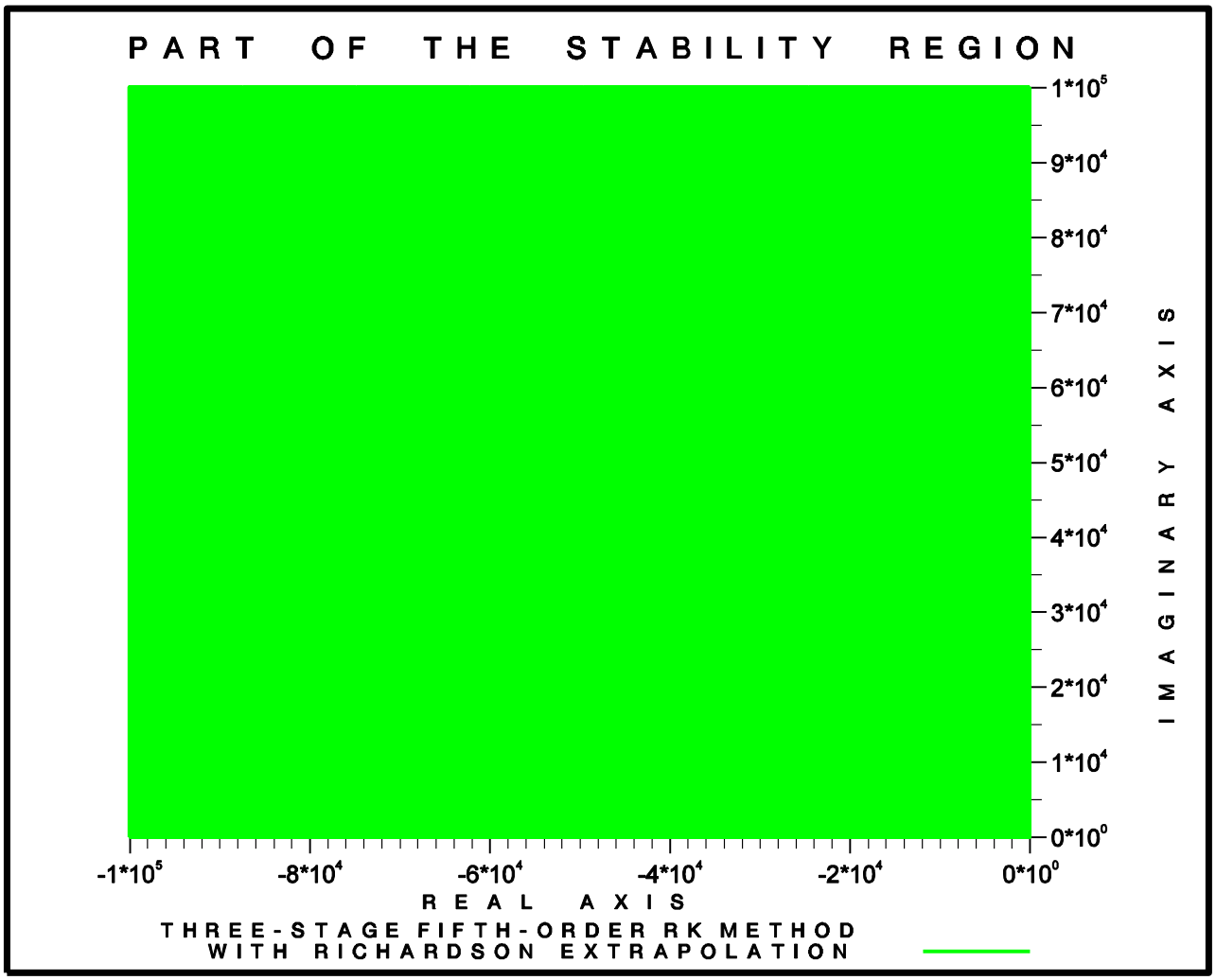

Figure 4.4

A part of the absolute stability region of the three-stage fifth-order Fully Implicit Runge-Kutta (FIRK) Method defined by the equalities (4.78)-(4.81) is given. This method is stable in a very large square whose side is $\mathbf{1.0 * 10^5}$.

The solution depictured in **Fig. 4.4** and **Fig. 4.5** and based on finding very large absolute stability regions is a compromise. We did not succeed to prove that the active Richardson Extrapolation combined with the two selected algorithms results in new numerical methods, which are stable for all values of $\boldsymbol{\nu = \bar{\alpha} + \bar{\beta} i}$ when $\boldsymbol{\bar{\alpha} \leq 0}$, i.e. in the whole $\mathbb{C}^-$. On the other hand, we did succeeded to show that the stability of each of the selected two numerical methods is preserved in a huge domain. **R. W. Hamming** claimed in one of the first monographs on numerical analysis, **[Hamming, 1962]**, that the choice of a numerical method is a question of finding some working compromise. Our solution is a good compromise. Let us emphasize, however, that it will, of course, be good to prove in a strict manner that the active Richardson Extrapolation applied either with the three-stage fifth-order Fully Implicit Runge-Kutta Method or with the two-stage third-order Diagonally Implicit Runge-Kutta Method results at least in an A-stable numerical algorithm. Nevertheless, from a practical point of view, the solution sketched above is as good as a proof. It gives the user a guarantee that the computational process will remain stable if the Dahlquist test-example is solved by applying any of the two particular

methods discussed above **with reasonably large time-stepsizes**. Moreover, one should expect that the computations remain stable also when other systems of ODEs are handled. The last statement will be verified in the next section, where an atmospheric chemical scheme from a well-known large-scale environmental model, fully described and tested in **Alexandrov et al. (2004), Bastrup-Birk et al. (1997), Zlatev (1995)** and **Zlatev and Dimov (2006)**, will be used in some numerical experiments.

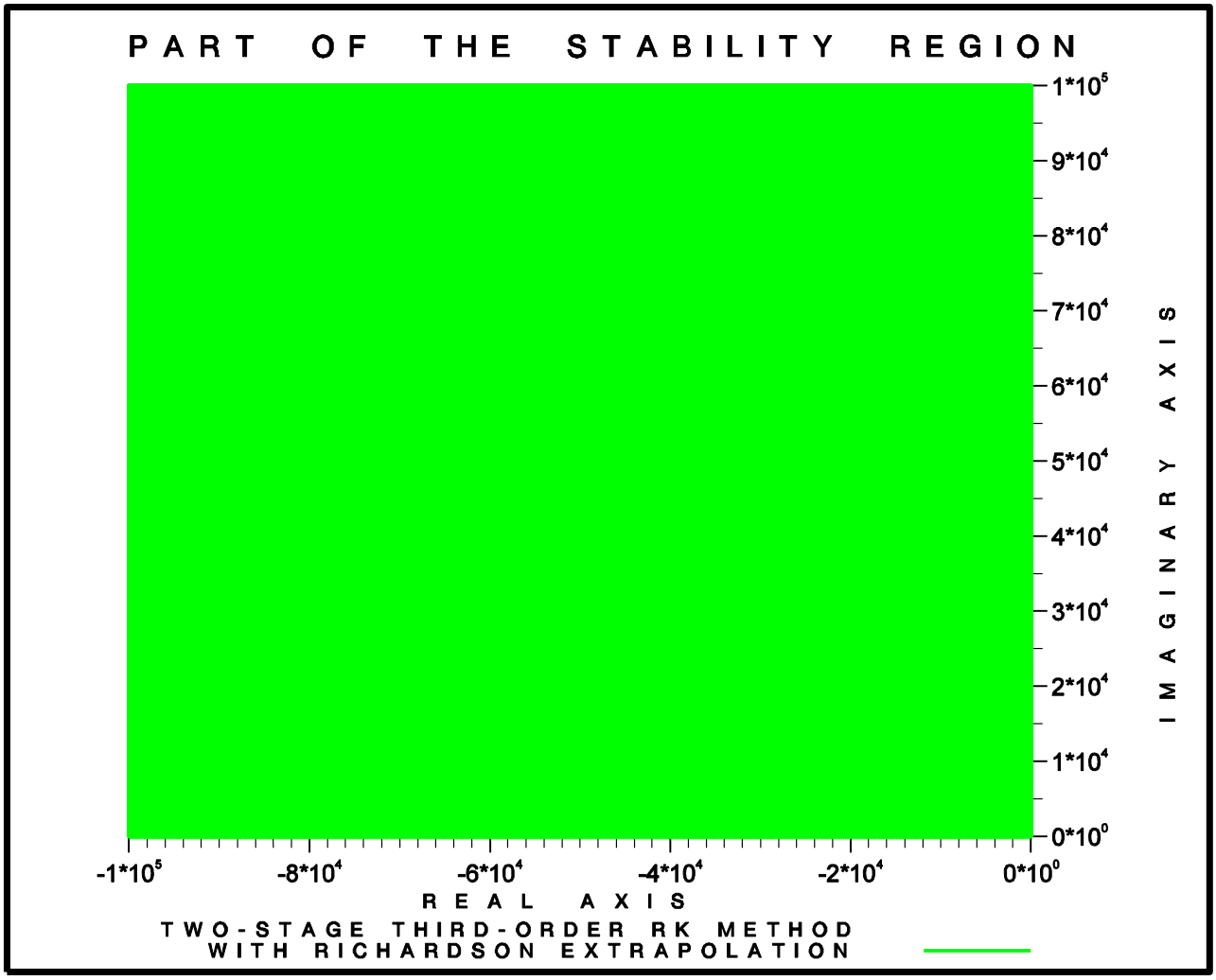

Figure 4.5

A part of the absolute stability region of the two-stage third-order Diagonally Implicit Runge-Kutta (DIRK) Method defined by the equalities (4.88)-(4.90) is given. This method is also stable within a very large square domain, whose side is $\mathbf{1.0 * 10^5}$.

4.7.7. Numerical experiments

Three numerical examples will be presented in this section. The first two of the selected examples are linear systems of ODEs with constant coefficients. This means that the Dahlquist theory presented in **Dahlquist (1963)** is valid for these two problems and the calculations **must** remain stable when parameter $\boldsymbol{\nu}$ is inside the stability regions shown in **Fig. 4.4** and **Fig. 4.5.** The third example is the

extremely badly scaled and very stiff non-linear system of ODEs arising in atmospheric chemistry, which was discussed in the previous sections of this chapter. The stability of the computational process is not guaranteed for the third example when the active implementation of the Richardson Extrapolation together with the two selected implicit Runge-Kutta methods is used (and the stability of the calculations will not be guaranteed even if the active implementation of the Richardson Extrapolation is A-stable or L-stable). The results presented in this section will show clearly that the computations remain nevertheless stable even when the time-stepsizes are very large.

Numerical example 1: Jacobian matrix with a large negative eigenvalue

This example was introduced in § 2.5.1 and it is a system of three equations with constant coefficients. The eigenvalues of its Jacobian matrix are $\boldsymbol{\mu_1 = -750}$, $\boldsymbol{\mu_2 = -0.3 + 8i}$, $\boldsymbol{\mu_3 = -0.3 - 8i}$ and it is clearly seen that the real eigenvalue is the dominant one. The system was integrated over the interval $\boldsymbol{t \in [0, 13.1072]}$ in § 2.5.1 and the largest time-stepsize used there was $\boldsymbol{h = 0.00512}$. We shall treat the same system over a much larger time-interval, $\boldsymbol{t \in [0, 2684.3545]}$, and will show that the computational process remains stable even if the time-stepsize becomes very large (time-stepsizes up to $\boldsymbol{h = 20.97152}$ will be used in this section).

The computations were organized as follow. A sequence of $\boldsymbol{22}$ runs was handled. The time-stepsize used during the first run was $\boldsymbol{h = 20.97152}$. The time-stepsize was reduced by a factor of two after every run (which means the number of steps was increased by a factor of two). The last stepsize was $\boldsymbol{h = 0.00001}$. The error made during step $\boldsymbol{j}$ of run $\boldsymbol{k}$, $\boldsymbol{k = 1, 2, 3, \ldots, 22}$, is estimated by

$$\boldsymbol{(4.94)} \quad \mathbf{ERROR}_j^{(k)} = \max_{i=1,2,3} \left(\frac{|y_{i,j} - y_i(t_j)|}{\max\left(|y_i(t_j)|, 1.0 \right)} \right), \quad j = 2^{k-1}, 2 \times 2^{k-1}, \ldots, 128 \times 2^{k-1},$$

where $\boldsymbol{y_{i,j}}$ and $\boldsymbol{y_i(t_j)}$ are components of the calculated approximation and the exact solution respectively. This means that we estimate the error at the same set of grid-points in each of the $\boldsymbol{22}$ runs. More precisely, the error is estimated at every time-step during the first run, at every second time-step during the second run, at every fourth time-step during the third run and we continue in the same manner after the third run. Thus, the number of grid-points, at which the error is estimated, is $\boldsymbol{128}$ for any of the $\boldsymbol{22}$ runs.

The global error made at run $\boldsymbol{k}$, $\boldsymbol{k = 1, 2, 3, \ldots, 22}$ is estimated by:

$$\boldsymbol{(4.95)} \quad \mathbf{ERROR}^{(k)} = \max_{j=2^{k-1}, 2\times 2^{k-1}, \ldots, 128\times 2^{k-1}} \left(\mathbf{ERROR}_j^{(k)} \right).$$

The obtained results are given in **Table 4.6** for the three-stage fifth-order Fully Implicit Runge-Kutta Method and its active combination with the Richardson Extrapolation.

Job number	Number of steps	Stepsize used	Direct FIRK		Richardson + FIRK	
			Accuracy	Rate	Accuracy	Rate
1	128	20.97152	3.005E-02		2.216E-03	
2	256	10.48576	3.004E-03	1.00	2.945E-03	0.72
3	512	5.24288	2.930E-03	1.03	2.959E-03	1.00
4	1024	2.62144	2.969E-03	0.99	2.959E-03	1.00
5	2048	1.31072	2.969E-03	1.00	2.959E-03	1.00
6	4096	0.65536	2.969E-03	1.00	2.216E-03	1.34
7	8192	0.32768	2.637E-03	1.13	8.043E-05	27.55
8	16384	0.16384	1.957E-04	13.47	1.125E-06	71.49
9	32768	0.08192	7.107E-06	27.54	1.544E-08	72.86
10	65536	0.04096	2.325E-07	30.57	2.233E-10	69.14
11	131072	0.02048	3.398E-09	68.42	3.349E-12	66.68
12	262144	0.01024	2.330E-10	14.58	5.125E-14	65.34
13	524288	0.00512	7.310E-12	31.87	7.926E-16	64.66
14	1048576	0.00256	2.288E-13	31.95	1.232E-17	64.33
15	2097152	0.00128	7.158E-15	31.96	1.920E-19	64.17
16	4194304	0.00064	2.238E-16	31.98	2.996E-21	64.09
17	8388608	0.00032	6.995E-18	31.99	4.678E-23	64.04
18	16777216	0.00016	2.186E-19	32.00	7.308E-25	64.01
19	33554432	0.00008	6.832E-21	32.00	1.142E-26	64.00
20	67108864	0.00004	2.135E-22	32.00	1.786E-28	63.94
21	134217728	0.00002	6.672E-24	32.00	2.787E-30	64.08
22	268439456	0.00001	2.085E-25	32.00	4.376E-32	62.32

Table 4.6

Numerical results obtained in **22** runs when the example with a large negative eigenvalue of the Jacobian matrix is treated with: **(i)** the direct implementation and **(ii)** the Active Richardson Extrapolation of the L-stable three-stage fifth-order Fully Implicit Runge-Kutta (FIRK) Method. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate" (the perfect rates being **32** for the direct method and **64** for the Richardson Extrapolation).

The corresponding results for the two-stage third-order Runge-Kutta Method are presented in **Table 4.7.** It should be mentioned here that **quadruple precision** was used in the computations. The following conclusions can be drawn from the results given in **Table 4.6** and **Table 4.7:**

Job number	Number of steps	Stepsize used	Direct FIRK		Richardson + FIRK	
			Accuracy	Rate	Accuracy	Rate
1	128	20.97152	1.611E-00		1.524E-00	
2	256	10.48576	1.132E-00	1.42	9.076E-01	1.68
3	512	5.24288	4.909E-01	2.31	2.112E-01	4.30
4	1024	2.62144	7.218E-02	6.80	1.957E-01	1.08
5	2048	1.31072	2.133E-02	3.38	5.357E-02	3.65
6	4096	0.65536	2.971E-03	7.18	2.987E-02	1.79
7	8192	0.32768	2.970E-03	1.00	6.710E-03	4.45
8	16384	0.16384	2.969E-03	1.00	2.399E-03	2.80
9	32768	0.08192	3.092E-03	0.96	2.520E-04	9.52
10	65536	0.04096	9.985E-04	3.10	1.969E-05	12.80
11	131072	0.02048	1.637E-04	6.10	2.038E-06	9.66
12	262144	0.01024	2.218E-05	7.38	1.106E-07	13.80
13	524288	0.00512	2.852E-06	7.78	8.017E-09	14.92
14	1048576	0.00256	3.606E-07	7.91	5.048E-10	15.88
15	2097152	0.00128	4.533E-08	7.95	3.193E-11	15.81
16	4194304	0.00064	5.682E-09	7.98	2.007E-12	16.03
17	8388608	0.00032	7.111E-10	7.99	1.252E-13	16.03
18	16777216	0.00016	8.895E-11	7.99	7.815E-15	16.02
19	33554432	0.00008	1.112E-11	8.00	4.881E-16	16.01
20	67108864	0.00004	1.390E-12	8.00	3.050E-17	16.00
21	134217728	0.00002	1.738E-13	8.00	1.906E-18	16.00
22	268439456	0.00001	2.173E-14	8.00	1.191E-19	16.00

Table 4.7

Numerical results obtained in **22** runs when the example with large complex eigenvalues is treated with: **(i)** the direct implementation and **(ii)** the Active Richardson Extrapolation of the A-stable version of the two-stage third-order Diagonally Implicit Runge-Kutta (DIRK) Method. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate" (the perfect rates being **8** for the direct method and **16** for the Richardson Extrapolation).

(a) The rates of convergence are close to the expected theoretical rates (assuming here that the stepsize becomes sufficiently small) see the results given in **Table 4.6** and **Table 4.7**.

(b) The rates of convergence of the three-stage fifth-order Fully Implicit Method when it is applied directly are slightly closer to the expected rates than the rates of convergence for

the combination of this method with the Richardson Extrapolation, but in both cases the results are very good (see **Table 4.6**).

(c) The rates of convergence of the two-stage third-order Diagonally Implicit Runge-Kutta method and its combination with the Richardson Extrapolation are closer to the expected values than those obtained with the three-stage fifth-order Fully Implicit Method, but in the latter case the results are much more accurate (compare the results given in **Table 4.6** with those given in **Table 4.7**).

(d) The results obtained by the Richardson Extrapolation are nearly always more accurate, and very often **much more** accurate, than those obtained by the underlying methods (compare again the two tables).

Numerical example 2: Jacobian matrix with large complex eigenvalues

This example was introduced in § 2.5.2 and it is also a system of three equations with constant coefficients. The eigenvalues of the Jacobian matrix of the linear system of ODEs from § 2.5.2 are $\mu_1 = -750 + 750i$, $\mu_2 = -750 - 750i$, $\mu_3 = -0.3$ and it is clearly seen that the two complex eigenvalue are much larger than the real eigenvalue (in absolute value). This system was also integrated in § 2.5.2 over the time-interval $t \in [0, 13.1072]$ and the largest time-stepsize used there was $h = 0.00512$. We shall treat the same system over a much larger time-interval, $t \in [0, 2684.3545]$, and will show that the computational process remains stable even if the time-stepsize becomes very large (time-stepsizes up to $h = 20.97152$ will be used in this section).

The computations were organized precisely in the same way as in the previous example. A sequence of 22 runs was handled also in this case. The time-stepsize that has been used during the first run was $h = 20.97152$. The time-stepsize was reduced by a factor of two after every run (which means the number of steps was increased by a factor of two). The last stepsize was $h = 0.00001$. The error made during step j of run k, $k = 1, 2, 3, \ldots, 22$, is estimated by using formulae (4.94) and (4.95).

Numerical results are shown in **Table 4.8** for the three-stage fifth-order Fully Implicit Runge-Kutta Method and in **Table 4.9** for the two-stage third-order Diagonally Implicit Runge-Kutta Method. Similar conclusions (as those drawn in the previous sub-section) can be drawn from the results given in **Table 4.8** and **Table 4.9**. It should additionally be noted that the accuracy obtained when the second example was run is greater than that obtained in the runs with the first example (compare the results shown in **Table 4.6** and **Table 4.7** with the results presented in this sub-section). This is not very surprising, because the oscillations in the solution of the second example are considerably smaller than those in the first one, see the formulae giving the exact solution of the two examples in § 2.5.1 and §2.5.2.

Job number	Number of steps	Stepsize used	Direct FIRK		Richardson + FIRK	
			Accuracy	Rate	Accuracy	Rate
1	128	20.97152	7.062E-01		7.863E-02	
2	256	10.48576	5.410E-02	13.05	3.871E-02	2.06
3	512	5.24288	3.887E-02	1.39	1.082E-02	3.58
4	1024	2.62144	9.268E-03	4.19	1.032E-03	10.48
5	2048	1.31072	7.098E-04	13.06	1.511E-05	68.30
6	4096	0.65536	7.544E-06	94.09	3.287E-08	459.69
7	8192	0.32768	2.039E-07	37.00	1.559E-10	21.08
8	16384	0.16384	6.221E-09	32.78	2.664E-13	58.52
9	32768	0.08192	1.942E-10	32.03	1.205E-14	22.11
10	65536	0.04096	6.079E-12	31.95	3.146E-16	38.30
11	131072	0.02048	1.903E-13	31.94	5.906E-18	53.27
12	262144	0.01024	5.952E-15	31.97	1.000E-19	59.06
13	524288	0.00512	1.861E-16	31.98	1.624E-21	61.58
14	1048576	0.00256	5.817E-18	31.99	2.585E-23	62.82
15	2097152	0.00128	1.818E-19	31.99	4.075E-25	63.44
16	4194304	0.00064	5.682E-21	32.00	6.397E-27	63.70
17	8388608	0.00032	1.776E-22	31.99	1.002E-28	63.84
18	16777216	0.00016	5.549E-24	32.01	1.567E-30	63.94
19	33554432	0.00008	1.734E-25	32.01	2.449E-32	63.99
20	67108864	0.00004	5.419E-27	32.00	1.851E-34	132.31
21	134217728	0.00002	1.693E-28	32.01	4.893E-34	0.37
22	268439456	0.00001	5.292E-30	31.99	3.824E-35	12.80

Table 4.8

Numerical results obtained in **22** runs when the example with large complex eigenvalues of the Jacobian matrix is treated with: **(i)** the direct implementation and **(ii)** the Active Richardson Extrapolation of the L-stable three-stage fifth-order Fully Implicit Runge-Kutta (FIRK) Method. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate" (the perfect rates being **32** for the direct method and **64** for the Richardson Extrapolation).

Numerical Example 3: Atmospheric Chemical Scheme

An important module, taken from the Unified Danish Eulerian Model (**UNI-DEM**), see **Alexandrov et al. (2004), Bastrup-Birk et al. (1997), Zlatev (1995)** and **Zlatev and Dimov (2006)**, was applied, as mentioned above, in several tests with the two selected numerical methods, the three-stage fifth-order Fully Implicit Runge-Kutta (FIRK) Method defined by (4.78)-(4.81) and the two-stage third-order Diagonally Implicit Runge-Kutta Method (DIRK) defined by (4-88)-(4.90) as well as with their

combinations of the Richardson Extrapolation. We shall use in this sub-section both the active implementation and the passive implementation of the Richardson Extrapolation.

Job number	Number of steps	Stepsize used	Direct FIRK		Richardson + FIRK	
			Accuracy	Rate	Accuracy	Rate
1	128	20.97152	3.089E-00		1.181E-00	
2	256	10.48576	1.199E-00	2.58	8.026E-01	1.47
3	512	5.24288	4.342E-01	2.76	3.575E-01	2.25
4	1024	2.62144	1.104E-01	3.93	9.895E-02	3.61
5	2048	1.31072	1.208E-02	9.14	5.669E-03	17.45
6	4096	0.65536	1.510E-04	80.00	2.632E-05	215.39
7	8192	0.32768	8.025E-06	18.82	3.891E-07	67.64
8	16384	0.16384	6.627E-07	12.11	2.191E-08	17.76
9	32768	0.08192	6.367E-08	10.41	1.921E-09	11.42
10	65536	0.04096	6.802E-09	9.36	8.145E-11	23.59
11	131072	0.02048	7.790E-10	8.73	5.061E-12	16.09
12	262144	0.01024	9.295E-11	8.38	3.961E-13	12.78
13	524288	0.00512	1.134E-11	8.20	2.772E-14	14.29
14	1048576	0.00256	1.401E-12	8.09	1.596E-15	17.37
15	2097152	0.00128	1.740E-13	8.05	8.185E-17	19.50
16	4194304	0.00064	2.168E-14	8.03	4.803E-18	17.04
17	8388608	0.00032	2.706E-15	8.01	3.001E-19	16.00
18	16777216	0.00016	3.380E-16	8.01	1.876E-20	16.00
19	33554432	0.00008	4.222E-17	8.01	1.172E-21	16.01
20	67108864	0.00004	5.278E-18	8.00	7.327E-23	16.00
21	134217728	0.00002	6.597E-19	8.00	4.579E-24	16.00
22	268439456	0.00001	8.246E-20	8.00	2.862E-25	16.00

Table 4.9

Numerical results obtained in **22** runs when the example with a large complex eigenvalues is treated with: **(i)** the direct implementation and **(ii)** the Active Richardson Extrapolation of the A-stable version of the two-stage third-order Diagonally Implicit Runge-Kutta (DIRK) Method. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate" (the perfect rates being **8** for the direct method and **16** for the Richardson Extrapolation).

The atmospheric chemical scheme was discussed in the previous sections. It is worthwhile to add here only the following facts. The air pollution model, **UNI-DEM**, in which this scheme is implemented, has been successfully applied in many comprehensive scientific investigations related to potentially harmful pollution levels in

(a) the Balkan Peninsula (**Zlatev, Georgiev and Dimov, 2013b**),

(b) Bulgaria (**Zlatev, 1995, Zlatev and Dimov, 2006, Zlatev, Dimov and Georgiev (2016), Zlatev and Syrakov, 2004a, 2004b**),

(c) Denmark (**Zlatev, 1995, Zlatev and Dimov, 2006, Zlatev and Moseholm, 2008**),

(d) England (**Abdalmogith, Harrison and Zlatev, 2004**),

(e) Europe (**Ambelas Skøth et al., 2000, Bastrup-Birk et al., 1997, Csomós et al., 2006, Geernaert and Zlatev, 2004, Zlatev, 1995, 2010, Zlatev and Dimov, 2006**),

(f) Hungary (**Havasi and Zlatev, 2002, Zlatev, Havasi and Faragó, 2011]**)

and

(g) the North Sea (**Harrison, Zlatev and Ottley, 1994**).

Most of the above studies, and especially **Zlatev, (2010)**, are related to the important topic of the influence of **the future climatic changes** on the high pollution levels in different parts of Europe. The model was furthermore used in a long sequence of comprehensive inter-comparisons of several well-known European large-scale models (**Hass et al., 2004, Roemer et al., 2004**).

The computations were organized as in Section 4.6, where more details about the atmospheric chemical scheme are given. Assume that $\bar{\mathbf{k}}$ runs are to be carried out. The errors calculated during step $\mathbf{j}$ of run $\mathbf{k}$, $\mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots, \bar{\mathbf{k}}$ are estimated by using the formula (4.60). The number of grid-points, at which the error is estimated, is $\mathbf{168}$ for any of the $\bar{\mathbf{k}}$ runs. Only the values of the reference solution at the grid-points of the coarse grid (which is used in the first run) have been stored and applied in the evaluation of the error (it is, of course, also possible to store all values of the reference solution, but such an action will increase tremendously the storage requirements). It is much more important and must be emphasized here that errors of the calculated approximations were always, in all nineteen runs, computed at the **same** $\mathbf{168}$ grid points. The global error made at run $\mathbf{k}$, $\mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots, \bar{\mathbf{k}}$ is estimated by using formula (4.61). All computations in this section were performed as in the previous sections by selecting **quadruple precision** (i.e. by using **REAL*16** declarations for the real numbers and, thus, about 32-digit arithmetic) in an attempt to eliminate completely the influence of the rounding errors in the first 16 significant digits of the computed approximate solutions. The calculations were carried out until the rounding errors start to interfere with the truncation errors. For the three-stage five-order FIRK method defined by (4.78)-(4.81) this happened for $\bar{\mathbf{k}} = \mathbf{14}$, while for the DIRK method the calculations could be performed without some very visible effect of the rounding errors until $\bar{\mathbf{k}}$ becomes $\mathbf{17}$. It should be pointed out, however, that the convergence rate is deteriorated when the number of correct digits becomes about sixteen. Now, after the explanation of the organization of the computations, we are ready to present some of the results from the numerical experiments, which were carried out in order to demonstrate the advantages of the application of Richardson Extrapolation.

Results obtained by using the three-stage fifth-order Fully Implicit Runge-Kutta Method are presented in **Table 4.10**. Several conclusions can be drawn from the results presented in **Table 4.10**:

(a) the direct method behaves in the beginning as a method of order five, but the rate of convergence becomes very slow after the run in which the accuracy becomes equal to $\mathbf{2.5 \times 10^{-16}}$,

(b) the two implementations of the Richardson Extrapolation are behaving as methods of order six in the beginning, but also here the rate of convergence is very slow when the accuracy becomes high, approximately equal to $\mathbf{1.0 \times 10^{-15}}$,

(c) during the last runs, the interference of the rounding errors becomes clear (the accuracy is no more increasing when the time-stepsize is reduced)

Job Number	Number of steps	Direct Implementation		Richardson Extrapolation			
				Active		Passive	
		Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
1	168	1.041E-06		3.986E-08		3.985E-08	
2	336	7.116E-08	14.64	5.989E-10	66.55	1.939E-09	20.56
3	672	3.451E-09	20.62	2.086E-11	28.71	3.091E-10	62.83
4	1344	9.673E-11	35.68	4.639E-12	44.97	4.649E-12	66.38
5	2688	7.527E-12	12.85	9.112E-14	50.91	9.117E-14	50.99
6	5376	2.804E-13	26.84	1.357E-15	67.15	1.357E-15	67.18
7	10752	8.515E-15	32.93	2.884E-16	4.71	2.889E-16	4.70
8	21504	2.508E-16	33.95	3.821E-17	7.55	3.871E-17	7.64
9	43008	5.413E-16	3.48	1.783E-17	2.14	1.796E-17	2,11
10	86016	1.114E-17	4.86	6.682E-18	2.69	6.682E-18	2.69
11	172032	6.599E-18	1.69	3.970E-18	1.68	3.970E-18	1.68
12	344064	2.382E-18	3.11	9.359E-19	4.24	9.359E-19	4.24
13	688128	1.179E-18	2.02	4.958E-19	1.89	4.958E-19	1.89
14	1376256	4.435E-19	2.66	1.596E-19	3.11	1.596E-19	3.11

Table 4.10

Numerical results obtained for ozone in $\bar{\mathbf{k}} = \mathbf{14}$ runs with three algorithms: **(i)** the direct implementation, **(ii)** the Active Richardson Extrapolation and **(iii)** the Passive Richardson Extrapolation of the L-stable three-stage fifth-order Fully Implicit Runge-Kutta (FIRK) Method defined by (8)-(11). The atmospheric chemical scheme with $\mathbf{56}$ species is handled. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under "Rate" (the perfect rates being $\mathbf{32}$ for the direct method and $\mathbf{64}$ for the Richardson Extrapolation).

and

(a) the accuracy is rather high from the very beginning and the situation where the rounding errors start to interfere with the truncation errors occurs quicker than for the two-stage third-order DIRK Method.

Results, obtained when the simpler two-stage third-order Diagonally Implicit Runge-Kutta Method is used, are given in **Table 4.11**.

Job Number	Number of steps	Direct Implementation		Richardson Extrapolation			
				Active		Passive	
		Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
1	168	1.581E-05		8.047E-06		8.047E-06	
2	336	5.555E-06	2.85	1.390E-06	5.79	3.597E-07	22.37
3	672	7.276E-07	7.63	5.173E-08	26.87	7.674E-08	4.69
4	1344	1.528E-07	4.76	9.852E-09	5.25	1.126E-08	6.82
5	2688	2.895E-08	5.28	1.353E-09	7.28	1.423E-09	7.91
6	5376	4.864E-09	5.95	1.419E-10	9.53	1.441E-10	9.87
7	10752	7.341E-10	6.63	1.214E-11	11.69	1.217E-11	11.85
8	21504	1.024E-10	7.17	9.007E-13	13.48	8.977E-13	13.55
9	43008	1.359E-11	7.54	6.070E-14	14.84	6.035E-14	14.87
10	86016	1.751E-12	7.76	3.847E-15	15.78	3.812E-15	15.83
11	172032	2.222E-13	7.88	2.268E-16	16.96	2.263E-16	16.85
12	344064	2.798E-14	7.94	1.478E-17	15.34	1.473E-17	15.37
13	688128	3.510E-15	7.97	6.133E-19	24.11	6.116E-19	24.08
14	1376256	4.393E-16	7.99	6.048E-20	10.14	6.051E-20	10.11
15	2752512	5.493E-17	8.00	5.652E-20	1.01	5.652E-20	1.01
16	5505024	6.844E-18	8.03	5.618E-20	1.01	5.618E-20	1.01
17	11010048	8.321E-19	8.23	5.618E-20	1.00	5.618E-20	1.00

Table 4.11

Numerical results obtained for ozone in $\bar{\mathbf{k}} = \mathbf{17}$ runs with three algorithms: **(i)** the direct implementation, **(ii)** the Active Richardson Extrapolation and **(iii)** the Passive Richardson Extrapolation of the A-stable version of the Diagonally Implicit Runge-Kutta (DIRK) Method defined by (18)-(20). The atmospheric chemical scheme with **56** species is handled. The errors obtained in the different runs are given in the columns under "Accuracy". The ratios of two successive errors are given in the columns under

"Rate" (the perfect rates being **8** for the direct method and **16** for the Richardson Extrapolation).

It is clearly seen that

(a) for sufficiently small values of the time-stepsize the direct implementation of the algorithm (18)-(20) behaves as a third-order numerical method (i.e. reducing the stepsize by a factor of three leads to an improvement of the accuracy by a factor approximately equal to eight),

(b) the two implementations of the Richardson Extrapolation are giving approximately the same degree of accuracy, which is nearly always higher that the corresponding accuracy of the direct method,

(c) if the time-stepsize becomes rather small, then the two implementations of the Richardson Extrapolation are gradually beginning to behave as methods of order four (i.e. reduction of the time-stepsize by a factor of two leads to an increase of the accuracy by a factor greater than eight, but in general less than sixteen)

and

(d) during the last runs with the Richardson Extrapolation, the interference of the rounding errors becomes very clear (the accuracy is no more increasing when the time-stepsize is reduced).

4.8. Some General Conclusions related to Chapter 4

Implicit Runge Kutta Methods were discussed in this chapter. We started with full description of the use of Richardson Extrapolation together with numerical schemes from the relatively simple class of the θ-methods. There are several reasons to study this case in detail:

(a) these methods are very popular and are implemented in many models treating different scientific and engineering problems,

(b) theoretical results can easily be achieved,

(c) it is clear, in principle at least, how these results can be extended for more complicated numerical methods,

(d) numerical experiments, which confirm the conclusions derived theoretically, can easily be organized and run.

It has been shown that the use of some methods can be worthwhile even if important properties of the numerical methods, such as A-stability and L- stability, cannot be established. These properties are only sufficient conditions for achieving stable computations. Stability of the computational process can also be achieved if the absolute stability properties of the numerical method under consideration are sufficiently large.

Also in this chapter we were mainly interested in the application of the Richardson Extrapolation, this time in relation to implicit numerical methods. The preservation of the stability of the computational process was our major aim as in the previous chapters. Several results obtained in this direction were presented and discussed. It is necessary to emphasize here the fact that all results related to the preservation of the stability in the numerical solution of ODEs are formulated and proved only for very special problems (as, for example, the scalar and linear test-problem proposed by **Dahlquist, 1963**, and some obvious generalization of this very special problem). One expects that if the numerical method under consideration preserves the stability during the computations related to these very special test-problems, then the calculation of the solution of much more complex systems of ODEs will also remain stable. It is, of course, worthwhile to verify the fact that such an expectation is fulfilled. A rather difficult from a computational point of view problem arising in atmospheric chemistry was chosen in order to illustrate the ability of the numerical method to produce stable numerical results also when non-linear, badly scaled and extremely ill-conditioned stiff systems of ODEs are handled.

4.9. Topics for further research

Several questions arise when the results presented in this chapter are carefully studied. It will be worthwhile to answer these questions, which are formulated as topics for future research below.

(A) It was shown that the two particular numerical methods (the combinations of the Richardson Extrapolation with the three-stage fifth-order Fully Implicit Runge-Kutta Method and with the two-stage third-order Diagonally Implicit Runge-Kutta Method), which were introduced in Section 4.7, have very large absolute stability regions and, therefore, could be successfully used in the solution of stiff systems of ODEs. Will it be possible to prove that these methods are at least A-stable (probably, by using the approach applied when the stability of the combinations of the Richardson Extrapolation with numerical algorithms from the class of the θ-methods was studied in Section 4.4)?

(B) Will it be possible to establish that also some other Implicit Runge-Kutta Methods (preferably methods of higher orders of accuracy) have also very large absolute stability regions when these are combined with the Richardson Extrapolation?

Chapter 5

Richardson Extrapolation for splitting techniques

The application of splitting procedures in the treatment of many large scientific and engineering problems is an excellent tool (and, very often, the only tool) by which huge computational tasks can be made tractable on the available computers by dividing them to a sequence of tasks, which are both smaller and simpler (see, for example, **Zlatev and Dimov, 2006**). The use of the Richardson Extrapolation in connection with two well-known and commonly used splitting procedures will be discussed in this chapter. We shall again be mainly interested in **the preservation of the stability properties** when the splitting procedures are combined with the Richardson Extrapolation.

Only rather simple examples will be used in Chapter 5 in order to make the understanding of the main ideas easier. We do believe that when these ideas are well understood, then it will be possible to apply them in different advanced and complex scientific models. The $\boldsymbol{\theta}$-methods with $\mathbf{0.5} \leq \boldsymbol{\theta} \leq \mathbf{1.0}$ will be used as underlying numerical methods when the simple sequential splitting procedure is used. Runge-Kutta methods will be used together with the Marchuk-Strang splitting procedure.

The simplest splitting technique, the so-called sequential splitting, will be introduced in **the first section**.

An expression for the stability function of the sequential splitting procedure will be derived in **the second section**.

Several results related to the stability properties will be proved in **the third section**.

Numerical results will be presented in **the fourth section**.

The Marchuk-Strang splitting procedure will be studied in **the fifth section.** Stability problems and numerical results will also be presented there.

Several concluding remarks are given in **Section 5.6**.

Some suggestions for research investigations will be formulated in the last section, **Section 5.7.**

5.1. Richardson Extrapolation for sequential splitting

Rewrite the initial value problem given in (1.1) in the following form:

$$(5.1) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{f_1(t,y)} + \mathbf{f_2(t,y)}, \ \mathbf{t} \in [\mathbf{a},\mathbf{b}], \ \mathbf{a} < \mathbf{b}, \ \mathbf{y} \in \mathbb{R}^{\mathbf{s}}, \ \mathbf{f_1} \in \mathbb{R}^{\mathbf{s}}, \ \mathbf{f_2} \in \mathbb{R}^{\mathbf{s}}, \ \mathbf{s} \geq \mathbf{1},$$

where $\mathbf{f_1(t,y)} + \mathbf{f_2(t,y)} = \mathbf{f(t,y)}$ and $\mathbf{y(a)} = \boldsymbol{\eta}$ being a given initial value. It is also assumed, only in an attempt to make the presentation of the results simpler, that the numerical solution of (5.1) is to be calculated on an equidistant grid (non-equidistant grids can also be introduced and used):

$$(5.2) \qquad \mathbf{t_0} = \mathbf{a}, \quad \mathbf{t_n} = \mathbf{t_{n-1}} + \mathbf{h} = \mathbf{t_0} + \mathbf{nh}, \quad \mathbf{n} = \mathbf{1,2,\ldots,N}, \quad \mathbf{t_N} = \mathbf{b}.$$

The simplest splitting procedure, the sequential splitting, can be introduced in the following way. Consider two systems of ODEs defined by

$$(5.3) \qquad \frac{\mathbf{dy^{[1]}}}{\mathbf{dt}} = \mathbf{f_1}\left(\mathbf{t}, \mathbf{y^{[1]}}\right), \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y^{[1]}} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_1} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1},$$

$$(5.4) \qquad \frac{\mathbf{dy^{[2]}}}{\mathbf{dt}} = \mathbf{f_2}\left(\mathbf{t}, \mathbf{y^{[2]}}\right), \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y^{[2]}} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_2} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1},$$

It is normally assumed here that it is easier (or even much easier) to solve numerically any of the two systems (5.3) and (5.4) than to solve the original system (5.1).

Assume that approximations of the exact solution $\mathbf{y(t)}$ of (5.1) are calculated, step by step, at the grid-points of the equidistant grid (5.2). Assume also that some approximation $\mathbf{y_{n-1}} \approx \mathbf{y(t_{n-1})}$ is available and that it is necessary to calculate the next approximation $\mathbf{y_n} \approx \mathbf{y(t_n)}$. This can be done by carrying out three successive steps:

(a) Find an approximate solution $\mathbf{y_n^{[1]}}$ of system (5.2) by using the selected numerical method and $\mathbf{y_{n-1}}$ as initial value.

(b) Find an approximate solution $\mathbf{y_n^{[2]}}$ of system (5.3) by using the selected numerical method and $\mathbf{y_n^{[1]}}$ as initial value.

(c) Set $\mathbf{y_n} = \mathbf{y_n^{[2]}}$ and consider it as an acceptable approximation to the solution $\mathbf{y(t_n)}$ of (5.1).

The algorithm described by the three consecutive steps **(a)**, **(b)** and **(c)** defines fully the calculations at an arbitrary step. It is only necessary to explain how to start the computations, i.e. how to calculate the first approximation $\mathbf{y_1} \approx \mathbf{y(t_1)}$, but this is not causing problems, because the initial value $\mathbf{y(t_0)} = \mathbf{y(a)} = \boldsymbol{\eta} = \mathbf{y_0}$ is available.

We are interested in the first part of this chapter to study the combination of the Richardson Extrapolation with the sequential splitting procedure. It will be necessary, as in the previous chapters, to calculate some intermediate results, when the Richardson Extrapolation is used together with the sequential splitting procedure. More precisely, we shall need the vectors $\mathbf{z_n^{(1)}}$, $\mathbf{z_n^{(2)}}$, $\mathbf{w_n^{(1)}}$ and $\mathbf{w_n^{(2)}}$. For the sake of simplicity we shall assume, as in the first part of Chapter 4, that the calculations are carried out by using the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta} \in [\mathbf{0.5}, \mathbf{1.0}]$. It is appropriate to consider additionally two approximations $\mathbf{w_{n-0.5}^{(1)}}$ and $\mathbf{w_{n-0.5}^{(2)}}$, which are calculated at the point $\mathbf{t_{n-0.5}} = \mathbf{t_n} - \mathbf{0.5h}$, where $\mathbf{h}$ is, as always in this book, the time-stepsize.

Under these assumptions, the calculations at time-step $\mathbf{n}$ that are related to the Richardson Extrapolation when it is combined with the sequential splitting procedure and with the $\boldsymbol{\theta}$-method can be carried out in the following three consecutive steps.

Step 1: Use a large time-stepsize $\mathbf{h}$ to calculate an approximation $\mathbf{z_n}$ of the exact value $\mathbf{y(t_n)}$ of the solution of (5.1) by selecting some of the $\boldsymbol{\theta}$-methods and by starting with the approximation $\mathbf{y_{n-1}}$ of $\mathbf{y(t_{n-1})}$ obtained at the previous time-step:

$$\mathbf{z_n^{(1)} = y_{n-1} + h\left[(1-\theta)f_1(t_{n-1}, y_{n-1}) + \theta f_1\left(t_n, z_n^{(1)}\right)\right]}, \tag{5.5}$$

$$\mathbf{z_n^{(2)} = z_n^{(1)} + h\left[(1-\theta)f_2\left(t_{n-1}, z_n^{(1)}\right) + \theta f_2\left(t_n, z_n^{(2)}\right)\right]}, \tag{5.6}$$

$$\mathbf{z_n \stackrel{def}{=} z_n^{(2)}}. \tag{5.7}$$

■

Step 2: Perform two small time-steps by using the same $\boldsymbol{\theta}$-method with a time-stepsize $\mathbf{0.5h}$ and by starting with the approximation $\mathbf{y_{n-1}}$ of $\mathbf{y(t_{n-1})}$ obtained at the previous time-step to calculate a second approximation $\mathbf{w_n}$ of the exact value $\mathbf{y(t_n)}$ of the solution of (5.1):

$$\mathbf{w_{n-0.5}^{(1)} = y_{n-1} + 0.5h\left[(1-\theta)f_1(t_{n-1}, y_{n-1}) + \theta f_1\left(t_{n-0.5}, w_{n-0.5}^{(1)}\right)\right]}, \tag{5.8}$$

$$\mathbf{w_{n-0.5}^{(2)} = w_{n-0.5}^{(1)} + 0.5h\left[(1-\theta)f_2\left(t_{n-1}, w_{n-0.5}^{(1)}\right) + \theta f_2\left(t_{n-0.5}, w_{n-0.5}^{(2)}\right)\right]}, \tag{5.9}$$

$$\mathbf{w_n^{(1)} = w_{n-0.5}^{(2)} + 0.5h\left[(1-\theta)f_1\left(t_{n-0.5}, w_{n-0.5}^{(2)}\right) + \theta f_1\left(t_n, w_n^{(1)}\right)\right]}, \tag{5.10}$$

$$(5.11) \quad \mathbf{w_n^{(2)} = w_n^{(1)} + 0.5h\left[(1-\theta)f_2\left(t_{n-0.5}, w_n^{(1)}\right) + \theta f_2\left(t_n, w_n^{(2)}\right)\right],}$$

$$(5.12) \quad \mathbf{w_n \stackrel{def}{=} w_n^{(2)}}\ .$$

■

Step 3: Assume that $\mathbf{0.5 < \theta \leq 1.0}$ and apply the formula $\mathbf{y_n = (2^p w_n - z_n)/(2^p - 1)}$ for computing the Richardson Extrapolation with $\mathbf{p = 1}$ to obtain an improved approximation $\mathbf{y_n}$ of $\mathbf{y(t_n)}$:

$$(5.13) \quad \mathbf{y_n = 2w_n - z_n}\ .$$

■

Note that if $\mathbf{0.5 < \theta \leq 1.0}$ then the combination consisting of the $\boldsymbol{\theta}$-method and the sequential splitting procedure is a **first-order** numerical method and, as stated in Chapter 1, the formula $\mathbf{y_n = (2^p w_n - z_n)/(2^p - 1)}$ for computing the Richardson Extrapolation should be used with $\mathbf{p = 1}$. The order of the Trapezoidal Rule, the $\boldsymbol{\theta}$-method with $\boldsymbol{\theta = 0.5}$, is **two**, but the combination of the Trapezoidal Rule and the sequential splitting is **again** a first-order numerical method. It is not very clear what to do in this situation, but the decision is not very important because the new method will anyway be unstable in this situation. One can formally apply the Richardson Extrapolation scheme with $\mathbf{p = 2}$ when the Trapezoidal Rule is directly used.

The combination consisting of the Richardson Extrapolation, the sequential splitting procedure and the $\boldsymbol{\theta}$-method will be a **second-order numerical method** and, therefore, it should be expected that the accuracy will be improved when the stability is preserved and the time-stepsize is sufficiently small.

5.2. Derivation of the stability function for the sequential splitting procedure

It is again (as in the previous chapters) appropriate to consider the simple scalar and linear test-problem proposed by **Dahlquist (1963)**, instead of the non-linear systems of ODEs (5.3) and (5.4), when the stability properties of the combined numerical method (consisting of the Richardson Extrapolation, the Sequential Splitting Procedure and the $\boldsymbol{\theta}$-method) is studied:

$$(5.14) \quad \frac{\mathbf{dy^{[1]}}}{\mathbf{dt}} = \boldsymbol{\lambda}\, \mathbf{y^{[1]}}, \qquad \frac{\mathbf{dy^{[2]}}}{\mathbf{dt}} = \boldsymbol{\lambda}\, \mathbf{y^{[2]}},$$

$$\mathbf{t} \in [\mathbf{0}, \infty], \quad \mathbf{y}^{[1]} \in \mathbb{C}, \quad \mathbf{y}^{[2]} \in \mathbb{C}, \quad \boldsymbol{\lambda} = \bar{\boldsymbol{\alpha}} + \bar{\boldsymbol{\beta}}\mathbf{i} \in \mathbb{C}, \quad \bar{\boldsymbol{\alpha}} \leq \mathbf{0}, \quad \mathbf{y}(\mathbf{0}) = \boldsymbol{\eta}.$$

The exact solution of each of the two systems in (2.14) is given by

$$(\mathbf{5.15}) \quad \mathbf{y}^{[1]}(\mathbf{t}) = \mathbf{y}^{[2]}(\mathbf{t}) = \boldsymbol{\eta}\, e^{\boldsymbol{\lambda}\mathbf{t}}, \quad \mathbf{t} \in [\mathbf{0}, \infty].$$

The three steps defined in the previous section can be rewritten in the following way under the assumption that the equalities (5.5) - (5.7) and (5.8) – (5.12) are used to calculate the approximations $\mathbf{z_n}$ and $\mathbf{w_n}$ of the exact values $\mathbf{y^{[1]}(t)}$ and $\mathbf{y^{[2]}(t)}$ from (5.15).

Step 1A: Use a large time-stepsize $\mathbf{h}$ to calculate an approximation $\mathbf{z_n}$ of the exact value $\mathbf{y(t_n)}$ of the solution of (5.14) by using the $\boldsymbol{\theta}$-method $\mathbf{0.5 < \theta \leq 1.0}$ and by starting with the approximation $\mathbf{y_{n-1}}$ of $\mathbf{y(t_{n-1})}$ obtained at the previous time-step:

$$(\mathbf{5.16}) \quad \mathbf{z_n^{(1)} = y_{n-1} + h(1-\theta)\lambda y_{n-1} + h\theta\lambda z_n^{(1)}}$$

$$\Rightarrow \quad \mathbf{z_n^{(1)} = \frac{1 + h(1-\theta)\lambda}{1 - h\theta\lambda} y_{n-1}},$$

$$(\mathbf{5.17}) \quad \mathbf{z_n^{(2)} = z_n^{(1)} + h(1-\theta)\lambda z_n^{(1)} + h\theta\lambda z_n^{(2)}}$$

$$\Rightarrow \quad \mathbf{z_n^{(2)} = \frac{1 + h(1-\theta)\lambda}{1 - h\theta\lambda}\, z_n^{(1)}}$$

$$\Rightarrow \quad \mathbf{z_n^{(2)} = \left[\frac{1 + h(1-\theta)\lambda}{1 - h\theta\lambda}\right]^2 y_{n-1}},$$

$$(\mathbf{5.18}) \quad \mathbf{z_n \overset{\text{def}}{=} z_n^{(2)} = \left[\frac{1 + h(1-\theta)\lambda}{1 - h\theta\lambda}\right]^2 y_{n-1}}.$$

■

Step 2A: Perform two small time-steps by using the same $\boldsymbol{\theta}$-method with a time-stepsize $\mathbf{0.5h}$ and by starting with the approximation $\mathbf{y_{n-1}}$ of $\mathbf{y(t_{n-1})}$ obtained at the previous time-step to calculate a second approximation $\mathbf{w_n}$ of the exact value $\mathbf{y(t_n)}$ of the solution of (5.14):

$$(5.19) \quad w_{n-0.5}^{(1)} = y_{n-1} + 0.5h(1-\theta)\lambda y_{n-1} + 0.5h\theta\lambda w_{n-0.5}^{(1)}$$

$$\Rightarrow \quad w_{n-0.5}^{(1)} = \frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda} y_{n-1},$$

$$(5.20) \quad w_{n-0.5}^{(2)} = w_{n-0.5}^{(1)} + 0.5h(1-\theta)\lambda w_{n-0.5}^{(1)} + 0.5h\theta\lambda w_{n-0.5}^{(2)}$$

$$\Rightarrow \quad w_{n-0.5}^{(2)} = \frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda} w_{n-0.5}^{(1)}$$

$$\Rightarrow \quad w_{n-0.5}^{(2)} = \left[\frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda}\right]^2 y_{n-1},$$

$$(5.21) \quad w_{n}^{(1)} = w_{n-0.5}^{(2)} + 0.5h(1-\theta)\lambda w_{n-0.5}^{(2)} + 0.5h\theta\lambda w_{n}^{(1)}$$

$$\Rightarrow \quad w_{n}^{(1)} = \frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda} w_{n-0.5}^{(2)}$$

$$\Rightarrow \quad w_{n}^{(1)} = \left[\frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda}\right]^3 y_{n-1},$$

$$(5.22) \quad w_{n}^{(2)} = w_{n}^{(1)} + 0.5h(1-\theta)\lambda w_{n}^{(1)} + 0.5h\theta\lambda w_{n}^{(2)}$$

$$\Rightarrow \quad w_{n}^{(2)} = \frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda} w_{n}^{(1)}$$

$$\Rightarrow \quad w_{n}^{(2)} = \left[\frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda}\right]^4 y_{n-1},$$

$$(5.23) \quad w_n \stackrel{\text{def}}{=} w_n^{(2)} = \left[\frac{1+0.5h(1-\theta)\lambda}{1-0.5h\theta\lambda}\right]^4 y_{n-1}$$

■

Step 3A: Apply, as in in the previous section, the formula $y_n = (2^p w_n - z_n)/(2^p - 1)$ for computing the Richardson Extrapolation with $p = 1$ to obtain an improved approximation y_n of $y(t_n)$:

$$(5.24) \quad y_n = 2w_n - z_n = \widetilde{R}(\nu)\, y_{n-1}$$

where $\nu = h\lambda$ and

$$(5.25) \quad \widetilde{R}(\nu) = 2\left[\frac{1+(1-\theta)(0.5\nu)}{1-\theta(0.5\nu)}\right]^4 - \left[\frac{1+(1-\theta)\nu}{1-\theta\nu}\right]^2$$

■

The last two formulae show clearly that the application of the Richardson Extrapolation combined with the sequential splitting procedure and the θ-method is resulting in a one-step numerical method for solving ODEs with a stability function $\widetilde{R}(\nu)$ when the test-problem (5.11) is solved under the assumptions made above. This means that all definitions used in the previous chapters are also valid for the new numerical methods which are combinations of

(a) a sequential splitting procedure,

(b) a scheme from the class of the θ-methods with $0.5 < \theta \leq 1.0$

and

(c) the Richardson Extrapolation.

It should also be noted that the **active** implementation of the Richardson Extrapolation is used in the algorithm described above. The passive implementation can also be applied. This implementation does not cause any stability problems, but may in some situations be not very accurate.

Note that if the underlying θ-method is applied directly (i.e. without combining it with the splitting procedure and the Richardson Extrapolation), then the stability function, see (4.8) in Chapter 4, is given by

$$(5.26) \quad R(\nu) = \frac{1+(1-\theta)\nu}{1-\theta\nu}.$$

If the underlying θ-method is combined only with the Richardson Extrapolation (but not with the sequential splitting procedure), then the stability function, see (4.18) in Chapter 4, is given by

$$(5.27) \quad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \mathbf{2}\left[\frac{\mathbf{1}+(\mathbf{1}-\boldsymbol{\theta})(\mathbf{0.5}\,\boldsymbol{\nu})}{\mathbf{1}-\boldsymbol{\theta}(\mathbf{0.5}\,\boldsymbol{\nu})}\right]^{\mathbf{2}} - \frac{\mathbf{1}+(\mathbf{1}-\boldsymbol{\theta})\boldsymbol{\nu}}{\mathbf{1}-\boldsymbol{\theta}\boldsymbol{\nu}} \qquad \text{when} \quad \boldsymbol{\theta} \neq \mathbf{0.5}$$

and

$$(5.28) \quad \bar{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{4}\left[\frac{\mathbf{1}+\mathbf{0.25}\,\boldsymbol{\nu}}{\mathbf{1}-\mathbf{0.25}\,\boldsymbol{\nu}}\right]^{\mathbf{2}} - \frac{\mathbf{1}+\mathbf{0.5}\boldsymbol{\nu}}{\mathbf{1}-\mathbf{0.5}\boldsymbol{\nu}}}{\mathbf{3}} \qquad \text{when} \quad \boldsymbol{\theta} = \mathbf{0.5}\,.$$

Comparing (5.25) with (5.26) or with (5.27) – (5.28) it is seen that the stability function becomes much more complicated when the Richardson Extrapolation is combined both with the scheme from the class of the $\boldsymbol{\theta}$-methods and with the sequential splitting procedure. Therefore, one should expect that the requirement $\left|\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right| \leq \mathbf{1}$, which is imposed in the different stability definitions, will be much more difficult than the corresponding requirements $|\bar{\mathbf{R}}(\boldsymbol{\nu})| \leq \mathbf{1}$ and $|\mathbf{R}(\boldsymbol{\nu})| \leq \mathbf{1}$ used in the previous chapter. This fact explains why it is necessary to study the stability properties of the new numerical methods, which are combinations of a sequential splitting procedure, a scheme from the class of the $\boldsymbol{\theta}$-methods and the Richardson Extrapolation. Stability properties will be studied in the next section.

5.3. Stability properties of the sequential splitting procedure

We shall mainly be interested in the application of the splitting procedure for stiff systems of ODEs. Therefore, it is necessary to require that the new numerical methods, the combinations of a sequential splitting procedure, a scheme from the class of the $\boldsymbol{\theta}$-methods and the Richardson Extrapolation, are at least A-stable. Sometimes more restrictive definitions, the definitions for strongly A-stable and L-stable numerical methods for solving systems of ODEs, will be needed. These three definitions are given in Chapter 4, but it is convenient to repeat them also here.

When the system of ODEs is stiff, it is reasonable to require that

$$(5.29) \quad \widetilde{\mathbf{R}}(\boldsymbol{\nu}) \leq \mathbf{1} \quad \text{for} \quad \forall\, \boldsymbol{\nu} = \boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{i} \quad \text{with} \quad \boldsymbol{\alpha} \leq \mathbf{0}\,,$$

where $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is the stability function from (5.25).

In other words, we shall again, as in Section 4, require that the crucial inequality $\widetilde{\mathbf{R}}(\boldsymbol{\nu}) \leq \mathbf{1}$ is satisfied **everywhere in the negative part of the complex plane** and that the absolute stability regions of the numerical methods are **infinitely large** (containing the whole negative part of the complex plane). More precisely, the definition for A-stability is formulated in the following way by using the concepts defined in this chapter.

Definition 5.1: It is said that the numerical method for solving systems of ODEs is **A-stable** when the relationship $\widetilde{\mathbf{R}}(\boldsymbol{\nu}) \leq \mathbf{1}$ is fulfilled for $\forall\, \boldsymbol{\nu} = \boldsymbol{\alpha} + \boldsymbol{\beta i}$ with $\boldsymbol{\alpha} \leq \mathbf{0}$ in the case where the selected numerical method is applied in the solution of the two Dahlquist scalar and linear test-examples (5.14).

■

It is worthwhile to emphasize again the fact that, because of the second Dahlquist barrier, which was introduced in Chapter 4, **every A-stable numerical method is necessarily implicit.** The numerical treatment of the implicit numerical methods is much more difficult than the numerical treatment of explicit numerical methods (this topic has been discussed in detail in Section 4.5).

As mentioned in Chapter 4, the $\boldsymbol{\theta}$-method is **A-stable** when $\boldsymbol{\theta} \in [\mathbf{0.5}, \mathbf{1.0}]$**.** Because of this fact, also in this chapter we shall, as stated above, consider numerical schemes of the class of the $\boldsymbol{\theta}$-methods with $\boldsymbol{\theta}$ varying in this interval.

We defined the concept of A-stability in connection with the simple scalar and linear equations (5.14). However, the results can be generalized for some linear systems of ODEs with constant matrices. Moreover, there are some reasons to expect that the results will hold also for some more general, linear and non-linear, systems of ODEs. These issues have been presented and discussed in Chapter 2 (see Section 2.1) and there is no need to repeat these explanations here.

As was pointed out in Section 4, the **A-stability** is sometimes not sufficient in the efforts to achieve an efficient computational process (an example was also given there to justify this statement). **L-stability** is necessary in the solution of some more difficult problems. This stronger concept is defined below.

Definition 5.2: A numerical method for solving systems of ODEs is said to be **L-stable** when it is A-stable and, in addition, when it is applied in the numerical solution to the two scalar and linear test-problems (5.14) proposed by Dahlquist, it leads to the relationship $\mathbf{y_n} = \widetilde{\mathbf{R}}(\boldsymbol{\nu})\, \mathbf{y_{n-1}} = \left[\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right]^{\mathbf{n}} \mathbf{y_0}$ with $\left|\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right| \rightarrow \mathbf{0}$ as $\mathbf{Re}(\boldsymbol{\nu}) \rightarrow -\infty$.

■

Sometimes it is very useful to relax a little the requirement for L-stability, by introducing the concept of **strong A-stability**.

Definition 5.3: A numerical method for solving systems of ODEs is said to be **strongly A-stable** when it is A-stable and, in addition, when it is applied in the numerical solution of the two scalar and linear test-problems (5.14) proposed by Dahlquist, it leads to the relationship $\mathbf{y_n} = \widetilde{\mathbf{R}}(\boldsymbol{\nu})\, \mathbf{y_{n-1}} = \left[\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right]^{\mathbf{n}} \mathbf{y_0}$ with $\left|\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right| \rightarrow \mathbf{c} < \mathbf{1}$ as $\mathbf{Re}(\boldsymbol{\nu}) \rightarrow -\infty$.

■

It is obvious that the definition for strong A-stability is a compromise between the weaker definition for A-stability and the stronger definition for L-stability (compare Definition 5.3 with Definition 5.1 and Definition 5.2). It is follows from the above statement that the class of the L-stable methods is a sub-class of the class of the strongly A-stable methods. The strongly A-stable methods form in their turn a class which is a sub-class of the class of A-stable methods.

It will be shown at the end of this chapter that for **some systems of ODEs strongly A-stable methods may perform better than L-stable methods.**

The stability properties of the combination of the Richardson Extrapolation applied together with the sequential splitting procedure and the $\boldsymbol{\theta}$-method when $\boldsymbol{\theta} \in [\mathbf{0.5}, \mathbf{1.0}]$ will be studied in the remaining part of this chapter. More precisely, the following theorem will be proved:

Theorem 5.1: The numerical method consisting of a combination of the Richardson Extrapolation applied together with the sequential splitting procedure and the $\boldsymbol{\theta}$-method is **strongly A-stable** if $\boldsymbol{\theta} \in [\boldsymbol{\theta_0}, \mathbf{1.0}]$ with $\boldsymbol{\theta_0} \approx \mathbf{0.638}$.

Proof: The same main principles, as those used in the proof of Theorem 4.1, can also be applied in the proof of Theorem 5.1. According to Definition 5.3 that was given above, a strongly A-stable numerical method **must** also be A-stable (see also, for example, **Hundsdorfer and Verwer, 2003**). In **Hairer and Wanner (1991)** it is shown that a numerical method for solving systems of ODEs is A-stable if and only if

(a) it is stable on the imaginary axis (i.e. when $\left|\widetilde{\mathbf{R}}(\mathbf{i\beta})\right| \leq \mathbf{1}$ holds for **all** real values of $\boldsymbol{\beta}$)

and

(b) $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is analytic in $\mathbb{C}^{-}$.

If we succeed to prove that **(a)** and **(b)** hold (i.e. if we show that the considered numerical method is A-stable), then it will be necessary to show additionally that the new numerical method is also strongly A-stable, i.e. that, according to Definition 5.3, the relationship $\left|\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right| \rightarrow \mathbf{c} < \mathbf{1}$ holds as $\mathbf{Re}(\boldsymbol{\nu}) \rightarrow -\infty$.

The above analysis indicates that Theorem 5.1 can be proved in three steps:

Step A: Show that the combination of the Richardson Extrapolation with the $\boldsymbol{\theta}$-methods and the sequential splitting procedure is stable on the imaginary axis.

Step B: Verify that the stability function $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is analytic.

Step C: Prove that $\left|\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right| \rightarrow \mathbf{c} < \mathbf{1}$ as $\mathbf{Re}(\boldsymbol{\nu}) \rightarrow -\infty$.

We shall start with Step A.

Step A – Stability on the imaginary axis

The stability function $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ from (5.25) can be rewritten as a ratio of two polynomials:

$$(\mathbf{5.30}) \quad \widetilde{\mathbf{R}}(\boldsymbol{\nu}) = \frac{\mathbf{P}(\boldsymbol{\nu})}{\mathbf{Q}(\boldsymbol{\nu})} .$$

The polynomial $\mathbf{P}(\boldsymbol{\nu})$ can be represented by the following expression:

$$(\mathbf{5.31}) \quad \mathbf{P}(\boldsymbol{\nu}) = \mathbf{2}[\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})(\mathbf{0.5}\boldsymbol{\nu})]^{\mathbf{4}}(\mathbf{1} - \boldsymbol{\theta}\boldsymbol{\nu})^{\mathbf{2}} - [\mathbf{1} + (\mathbf{1} - \boldsymbol{\theta})\boldsymbol{\nu}]^{\mathbf{2}}[\mathbf{1} - \boldsymbol{\theta}(\mathbf{0.5}\boldsymbol{\nu})]^{\mathbf{4}} ,$$

which is of order six with respect to $\boldsymbol{\nu}$, depends on the parameter $\boldsymbol{\theta}$ and can be rewritten in the following form:

$$(\mathbf{5.32}) \quad \mathbf{P}(\boldsymbol{\nu}) = \mathbf{A}\boldsymbol{\nu}^{\mathbf{6}} + \mathbf{B}\boldsymbol{\nu}^{\mathbf{5}} + \mathbf{C}\boldsymbol{\nu}^{\mathbf{4}} + \mathbf{D}\boldsymbol{\nu}^{\mathbf{3}} + \mathbf{E}\boldsymbol{\nu}^{\mathbf{2}} + \mathbf{F}\boldsymbol{\nu} + \mathbf{1} ,$$

where the coefficients are given by

$$(\mathbf{5.33}) \quad \mathbf{A} = \frac{\boldsymbol{\theta}^{\mathbf{2}}\,(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{2}}(\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{4}\boldsymbol{\theta} + \mathbf{2})}{\mathbf{2}^{\mathbf{4}}},$$

$$(\mathbf{5.34}) \quad \mathbf{B} = \frac{\boldsymbol{\theta}(\mathbf{1} - \boldsymbol{\theta})[-\mathbf{2}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{3}} + \mathbf{8}\boldsymbol{\theta}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{2}} + \mathbf{4}\boldsymbol{\theta}^{\mathbf{2}}\,(\mathbf{1} - \boldsymbol{\theta}) - \boldsymbol{\theta}^{\mathbf{3}}\,]}{\mathbf{2}^{\mathbf{3}}},$$

$$(\mathbf{5.35}) \quad \mathbf{C} = \frac{\mathbf{2}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{4}} - \mathbf{32}\boldsymbol{\theta}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{3}} + \mathbf{24}\boldsymbol{\theta}^{\mathbf{2}}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{2}} + \mathbf{16}\boldsymbol{\theta}^{\mathbf{3}}(\mathbf{1} - \boldsymbol{\theta}) - \boldsymbol{\theta}^{\mathbf{4}}}{\mathbf{2}^{\mathbf{4}}},$$

$$(\mathbf{5.36}) \quad \mathbf{D} = \frac{\mathbf{2}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{3}} - \mathbf{8}\boldsymbol{\theta}(\mathbf{1} - \boldsymbol{\theta})^{\mathbf{2}} + \mathbf{2}\boldsymbol{\theta}^{\mathbf{2}}\,(\mathbf{1} - \boldsymbol{\theta}) + \boldsymbol{\theta}^{\mathbf{3}}}{\mathbf{2}},$$

$$(\mathbf{5.37}) \quad \mathbf{E} = \frac{\mathbf{13}\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{16}\boldsymbol{\theta} + \mathbf{4}}{\mathbf{2}},$$

$$(5.38) \quad \mathbf{F = 2 - 4\theta}\,.$$

The denominator of (5.30) can be written as

$$(5.39) \quad \mathbf{Q(\nu) = [1 - \theta(0.5\nu)]^4(1 - \theta\nu)^2}\,.$$

Assume that the complex variable $\boldsymbol{\nu}$ is represented as $\boldsymbol{\nu = \alpha + i\beta}$. It is shown in **Hairer and Wanner (1991)** that the numerical method with a stability function $\boldsymbol{\tilde{R}(\nu)}$ is stable on the imaginary axis if

$$(5.40) \quad \mathbf{E(\beta) \geq 0}$$

for all real values of $\boldsymbol{\beta}$, where $\mathbf{E(\beta)}$ is defined by

$$(5.41) \quad \mathbf{E(\beta) = Q(i\beta)Q(-i\beta) - P(i\beta)P(-i\beta)}\,.$$

After long but straight-forward transformations the following two relationships can be derived:

$$(5.42) \quad \mathbf{Q(i\beta)Q(-i\beta) = \frac{\theta^{12}}{2^8}\beta^{12} + \frac{90\theta^{10}}{2^7}\beta^{10} + \frac{129\theta^8}{2^8}\beta^8 + \frac{29\theta^6}{2^4}\beta^6 + \frac{27\theta^4}{2^3}\beta^4 + 3\theta^2\beta^2 + 1}\,.$$

$$(5.43) \quad \mathbf{P(i\beta)P(-i\beta) = A^2\beta^{12} + (B^2 - 2AC)\beta^{10} + (C^2 - 2BD + 2AE)\beta^8} \\ \mathbf{+(D^2 - 2CE + 2BF - 2A)\beta^6 + (E^2 - 2DF + 2C)\beta^4 + (F^2 - 2E)\beta^2 + 1}\,.$$

Substitute the expressions on the right-hand-sides of (5.42) and (5.43) in (5.41):

$$(5.44) \quad \mathbf{E(\beta) = \frac{\theta^{12} - 2^8A^2}{2^8}\beta^{12} + \frac{9\theta^{10} - 2^7(B^2 - 2AC)}{2^7}\beta^{10}} \\ \mathbf{+ \frac{129\theta^8 - 2^8(C^2 - 2BD + 2AE)}{2^8}\beta^8 + \frac{29\theta^6 - 2^4(D^2 - 2CE + 2BF - 2A)}{2^4}\beta^6} \\ \mathbf{+ \frac{27\theta^4 - 2^4(E^2 - 2DF + 2C)}{2^4}\beta^4}\,.$$

Introduce the following notations:

$$(5.45) \quad \mathbf{H_1(\theta)} = \frac{\boldsymbol{\theta^{12} - 2^8 A^2}}{\boldsymbol{2^8}} ,$$

$$(5.46) \quad \mathbf{H_2(\theta)} = \frac{\boldsymbol{9\theta^{10} - 2^7 (B^2 - 2AC)}}{\boldsymbol{2^7}} ,$$

$$(5.47) \quad \mathbf{H_3(\theta)} = \frac{\boldsymbol{129\theta^8 - 2^8 (C^2 - 2BD + 2AE)}}{\boldsymbol{2^8}} ,$$

$$(5.48) \quad \mathbf{H_4(\theta)} = \frac{\boldsymbol{29\theta^6 - 2^4 (D^2 - 2CE + 2BF - 2A)}}{\boldsymbol{2^4}} ,$$

$$(5.49) \quad \mathbf{H_5(\theta)} = \frac{\boldsymbol{27\theta^4 - 2^4 (E^2 - 2DF + 2C)}}{\boldsymbol{2^4}} .$$

It is clear that $\mathbf{E(\beta)}$ will be non-negative for all values of $\boldsymbol{\beta}$ and for a given value of $\boldsymbol{\theta}$ if all the five polynomials (5.45) – (5.49) are non-negative for the selected values of parameter $\boldsymbol{\theta}$. The curves representing these polynomials for $\boldsymbol{\theta \in [\, 0.5 \,,\, 1.0 \,]}$ are drawn in **Fig. 5.1 – Fig. 5.5**. The results show that the combination of the Richardson Extrapolation, the sequential splitting and the $\boldsymbol{\theta}$-method for all values of $\boldsymbol{\theta}$ is stable on the imaginary axis in the interval $\boldsymbol{[\, \theta_0 \,,\, 1.0 \,]}$ with $\boldsymbol{\theta_0 \approx 0.638}$. Thus the first part of Theorem 5.1 is proved.

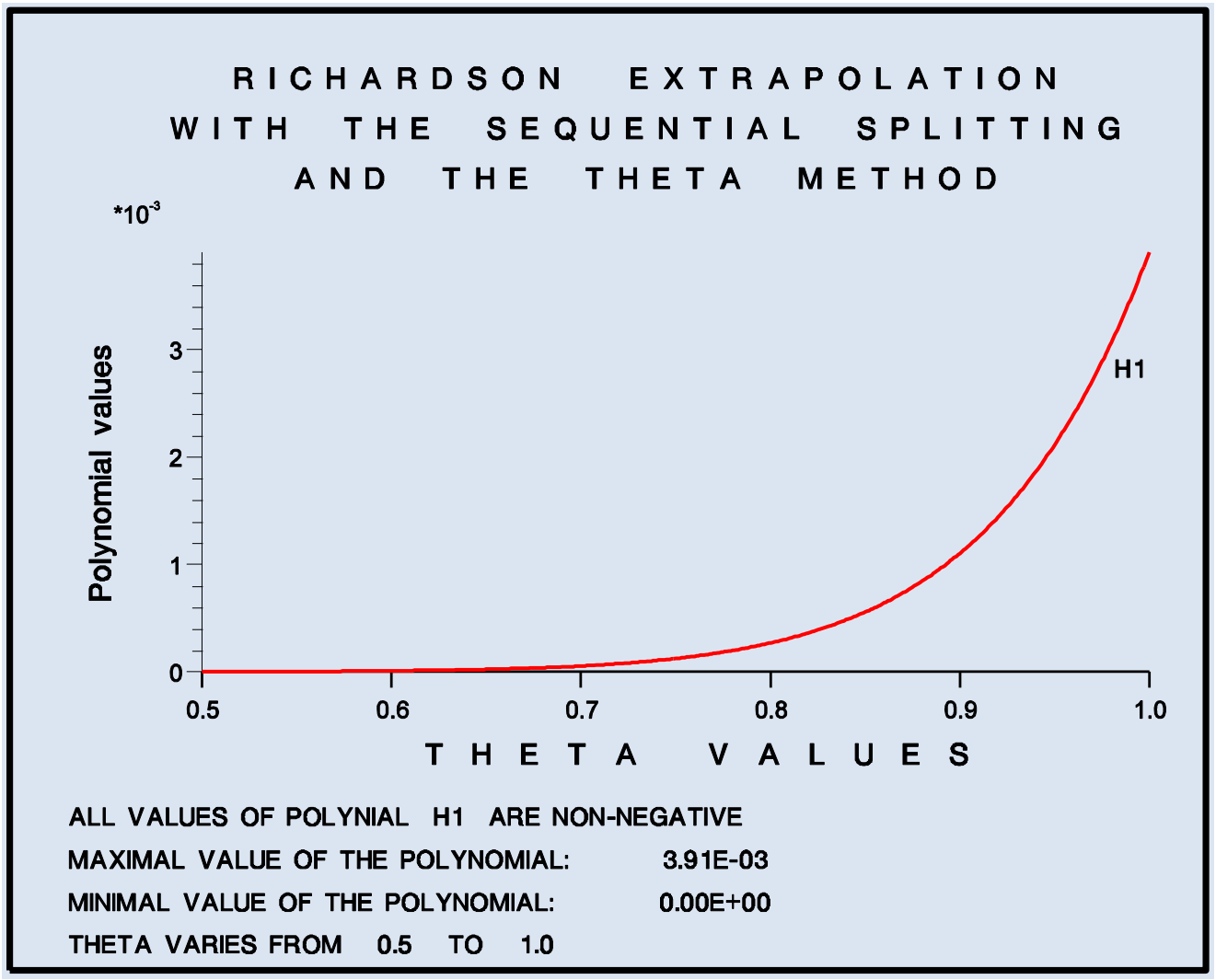

Figure 5.1
Variation of the polynomial $\mathbf{H_1(\theta)}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$.

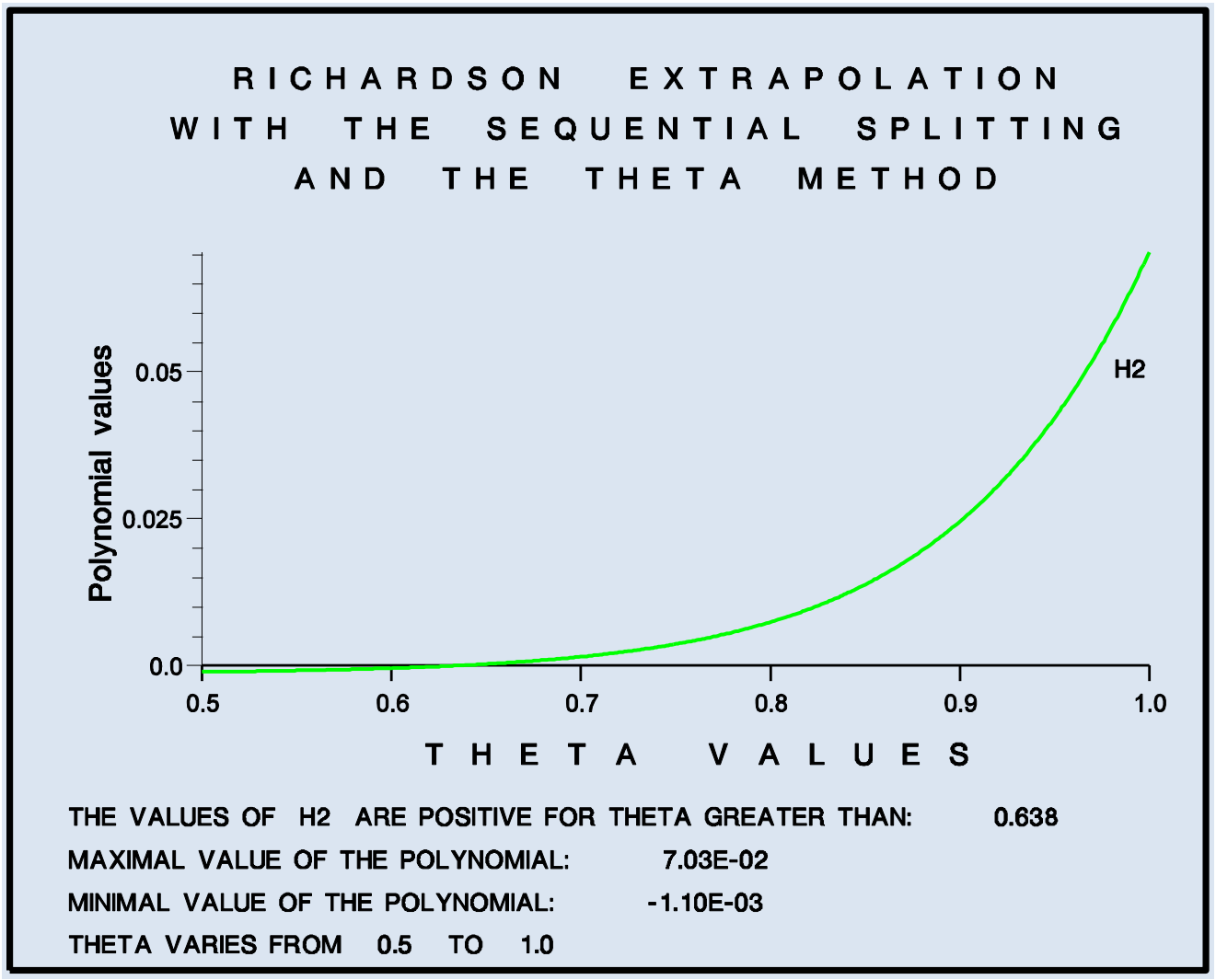

Figure 5.2

Variation of the polynomial $\mathbf{H_2(\theta)}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$.

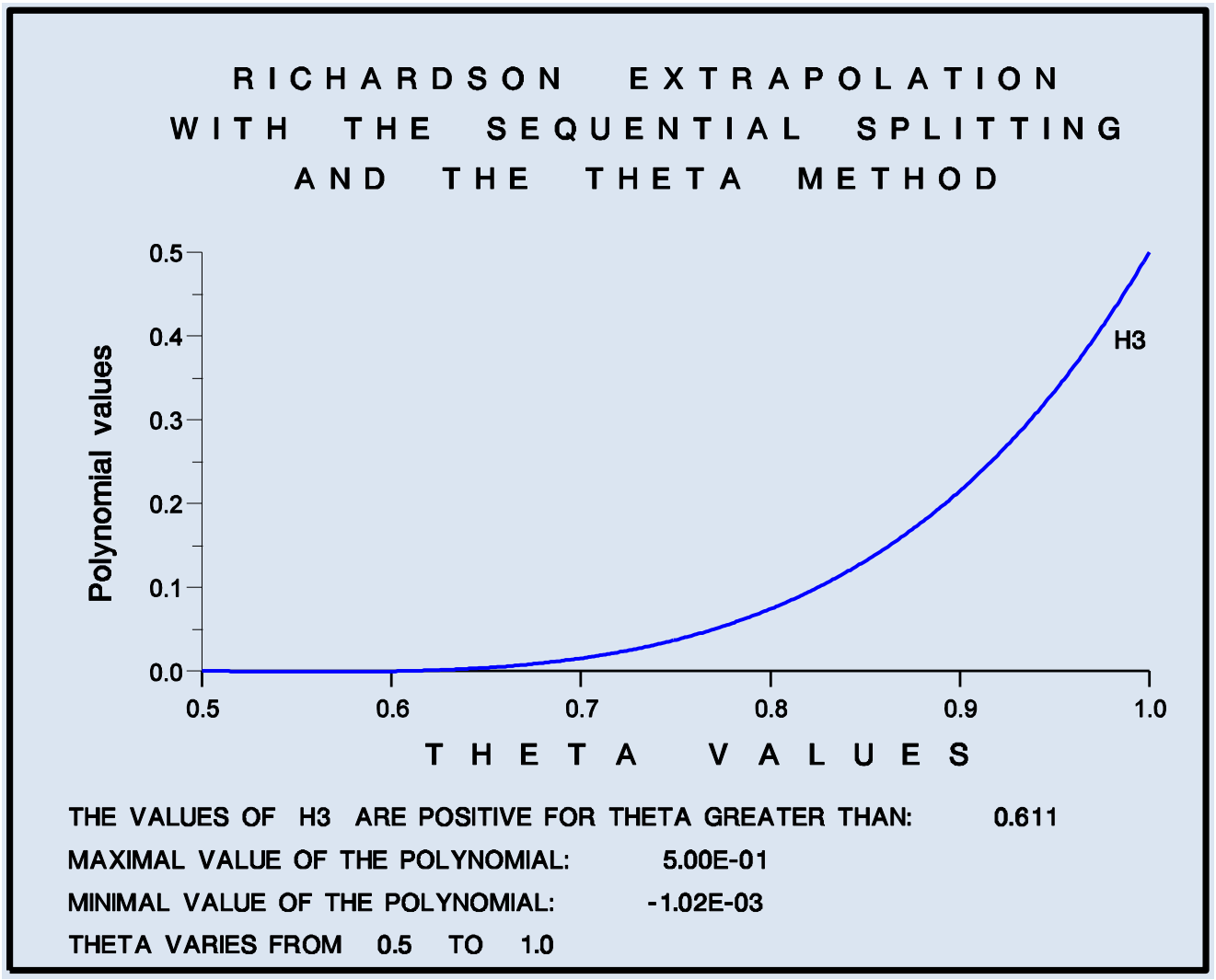

Figure 5.3:
Variation of the polynomial $\mathbf{H_3(\theta)}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$.

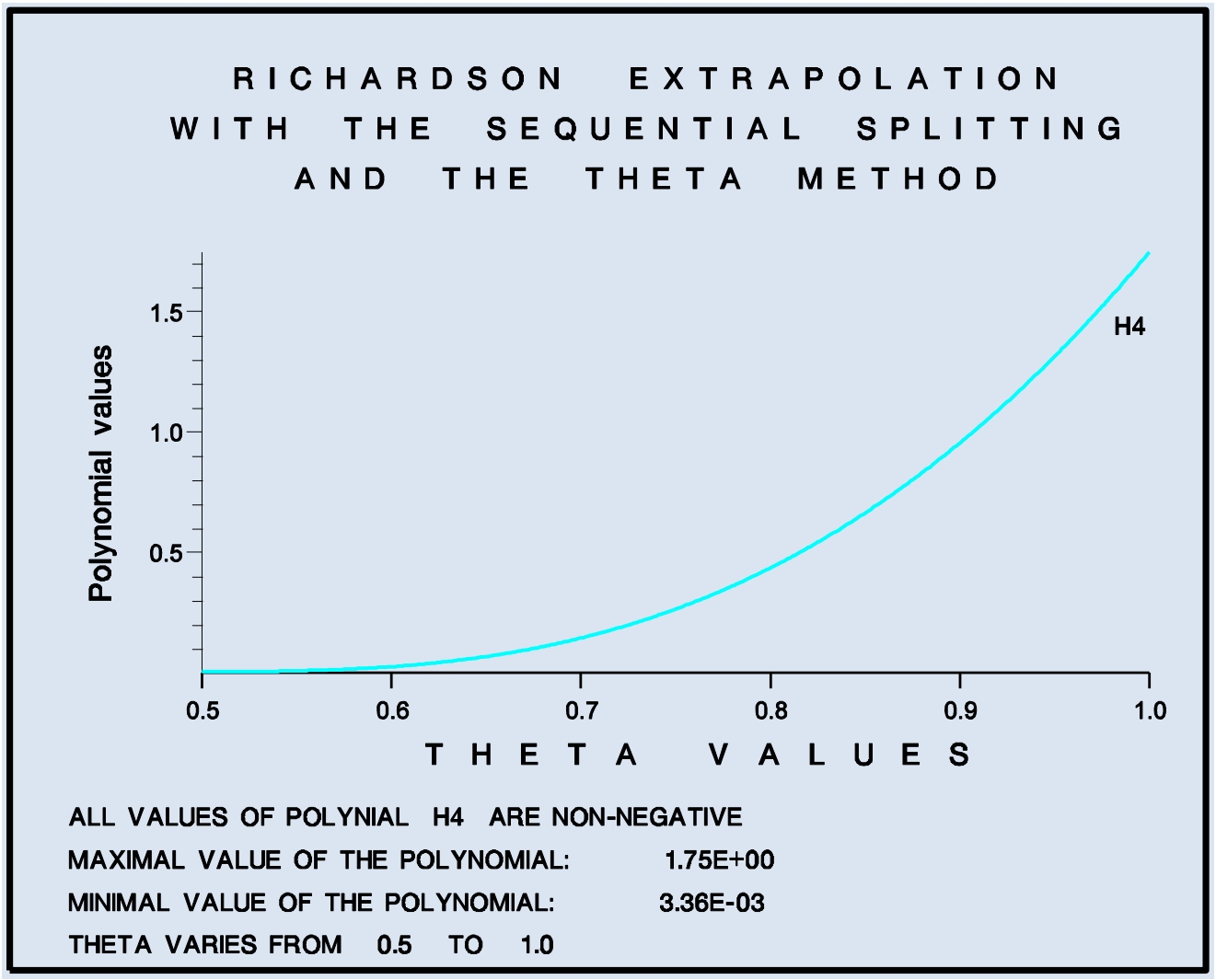

Figure 5.4

Variation of the polynomial $\mathbf{H_4(\theta)}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$.

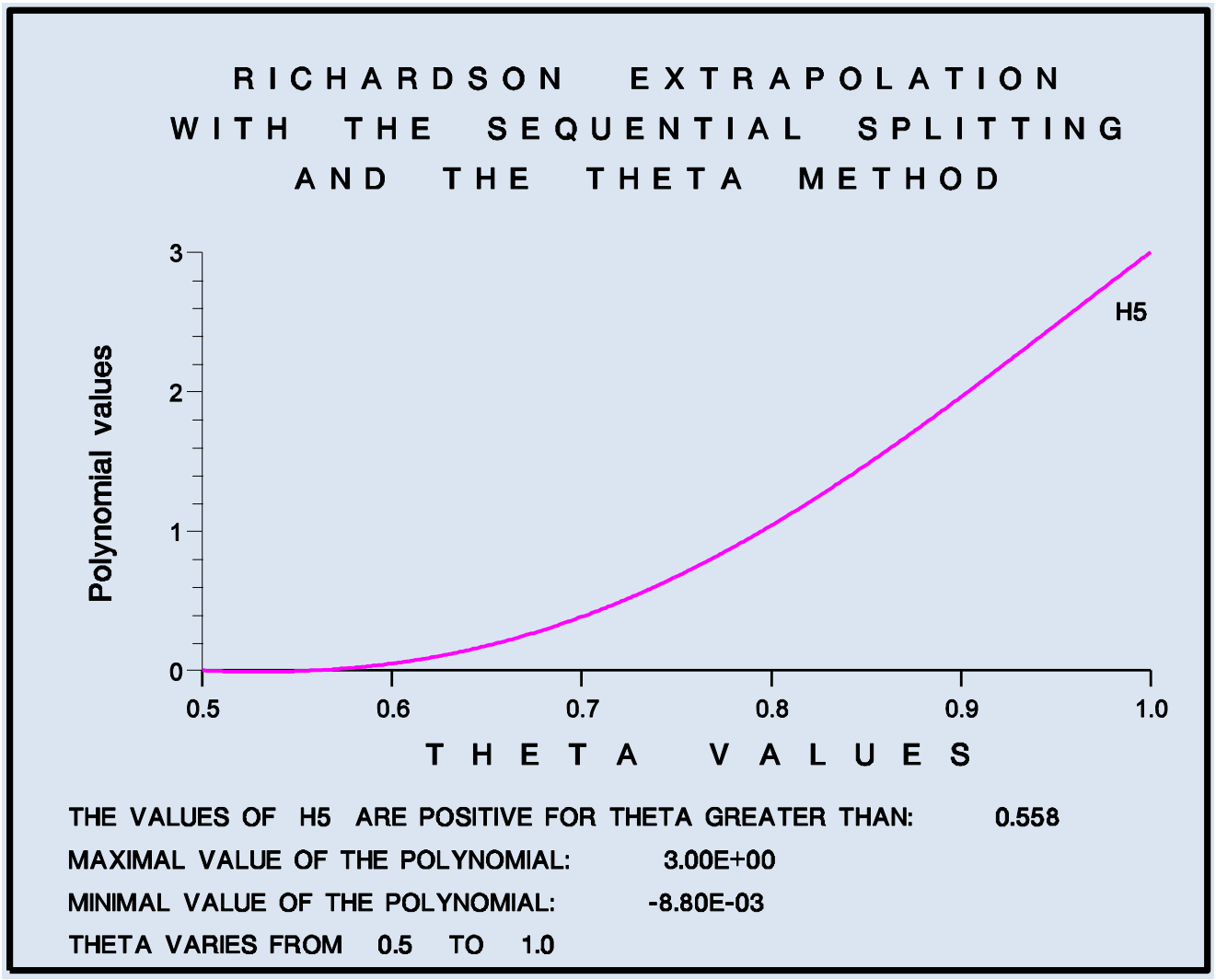

Figure 5.5
Variation of the polynomial $\mathbf{H_5(\theta)}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$.

Step B – A-stability

After the proof that the combination of the Richardson Extrapolation, the sequential splitting and the $\boldsymbol{\theta}$-method is stable on the imaginary for all values of $\boldsymbol{\theta}$ in the interval $[\,\boldsymbol{\theta_0}\,,\ \mathbf{1.0}\,]$ with $\boldsymbol{\theta_0} \approx \mathbf{0.638}$ it is necessary to prove that the stability function $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is analytic in $\mathbb{C}^-$, which will ensure that the combined numerical method is A-stable. The stability function $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is a ratio of two polynomials, $\mathbf{P}(\boldsymbol{\nu})$ and $\mathbf{Q}(\boldsymbol{\nu})$; see (5.27). It is well-known that polynomials are analytic functions and a ratio of two polynomials is an analytic function in $\mathbb{C}^-$ if the denominator $\mathbf{Q}(\boldsymbol{\nu})$ of $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ has no zero in $\mathbb{C}^-$. All zeros of the denominator $\boldsymbol{\nu_1} = \boldsymbol{\nu_2} = \mathbf{1}/\boldsymbol{\theta} > \mathbf{0}$ and $\boldsymbol{\nu_3} = \boldsymbol{\nu_4} = \boldsymbol{\nu_5} = \boldsymbol{\nu_6} = \mathbf{2}/\boldsymbol{\theta} > \mathbf{0}$ are positive, which means that $\widetilde{\mathbf{R}}(\boldsymbol{\nu})$ is analytic in $\mathbb{C}^-$ and, thus, the method is A-stable. This proves the second part of Theorem 5.1.

Step C – Strong A-stability

Rewrite (5.25) as

$$(\mathbf{5.50}) \qquad \widetilde{\mathbf{R}}(\boldsymbol{\nu}) = \mathbf{2}\left[\frac{\frac{\mathbf{1}}{\boldsymbol{\nu}} + \mathbf{0.5}(\mathbf{1}-\boldsymbol{\theta})}{\frac{\mathbf{1}}{\boldsymbol{\nu}} - \mathbf{0.5}\boldsymbol{\theta}}\right]^{\mathbf{4}} - \left[\frac{\frac{\mathbf{1}}{\boldsymbol{\nu}} + (\mathbf{1}-\boldsymbol{\theta})}{\frac{\mathbf{1}}{\boldsymbol{\nu}} - \boldsymbol{\theta}}\right]^{\mathbf{2}}$$

Assume that the imaginary part of $\boldsymbol{\nu}$ is fixed. Then the expression $\lim\limits_{\mathbf{Re}(\boldsymbol{\nu})\to\infty}\left[\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right]$ can be evaluated in the following way for any fixed value of the imaginary part of $\boldsymbol{\nu}$:

$$(\mathbf{5.51}) \qquad \lim_{\mathbf{Re}(\boldsymbol{\nu})\to\infty}\left[\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right] = \mathbf{2}\left[\frac{\mathbf{0.5}(\mathbf{1}-\boldsymbol{\theta})}{-\mathbf{0.5}\boldsymbol{\theta}}\right]^{\mathbf{4}} - \left[\frac{(\mathbf{1}-\boldsymbol{\theta})}{-\boldsymbol{\theta}}\right]^{\mathbf{2}} = \frac{\mathbf{2}(\mathbf{1}-\boldsymbol{\theta})^{\mathbf{4}}}{\boldsymbol{\theta}^{\mathbf{4}}} - \frac{(\mathbf{1}-\boldsymbol{\theta})^{\mathbf{2}}}{\boldsymbol{\theta}^{\mathbf{2}}}$$

$$= \frac{\mathbf{2} - \mathbf{8}\boldsymbol{\theta} + \mathbf{11}\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{6}\boldsymbol{\theta}^{\mathbf{3}} + \boldsymbol{\theta}^{\mathbf{4}}}{\boldsymbol{\theta}^{\mathbf{4}}}$$

Since the last expression in (5.51) is real, the requirement $\lim\limits_{\mathbf{Re}(\boldsymbol{\nu})\to\infty}\left[\widetilde{\mathbf{R}}(\boldsymbol{\nu})\right] \leq \boldsymbol{\xi} < \mathbf{1}$ is satisfied when the following two relationships hold:

$$(\mathbf{5.52}) \qquad \frac{\mathbf{2} - \mathbf{8}\boldsymbol{\theta} + \mathbf{11}\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{6}\boldsymbol{\theta}^{\mathbf{3}} + \boldsymbol{\theta}^{\mathbf{4}}}{\boldsymbol{\theta}^{\mathbf{4}}} < \mathbf{1} \qquad \Longrightarrow \qquad -\mathbf{2} + \mathbf{8}\boldsymbol{\theta} - \mathbf{11}\boldsymbol{\theta}^{\mathbf{2}} + \mathbf{6}\boldsymbol{\theta}^{\mathbf{3}} > \mathbf{0}$$

and

$$(\mathbf{5.53}) \qquad -\mathbf{1} < \frac{\mathbf{2} - \mathbf{8}\boldsymbol{\theta} + \mathbf{11}\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{6}\boldsymbol{\theta}^{\mathbf{3}} + \boldsymbol{\theta}^{\mathbf{4}}}{\boldsymbol{\theta}^{\mathbf{4}}} \qquad \Longrightarrow \qquad \mathbf{2} - \mathbf{8}\boldsymbol{\theta} + \mathbf{11}\boldsymbol{\theta}^{\mathbf{2}} - \mathbf{6}\boldsymbol{\theta}^{\mathbf{3}} + \boldsymbol{\theta}^{\mathbf{4}} > \mathbf{0}\ .$$

It can easily be established that (5.52) and (5.53) are satisfied for all values of $\boldsymbol{\theta}$ in the interval $[\,\mathbf{0.5}\,,\ \mathbf{1.0}\,]$, but since the method should be A-stable, we have to take the interval $[\,\boldsymbol{\theta}_{\mathbf{0}}\,,\ \mathbf{1.0}\,]$ with $\boldsymbol{\theta}_{\mathbf{0}} \approx \mathbf{0.638}$. This completes the third part of Theorem 5.1.

■

Corollary 5.1 If $\boldsymbol{\theta = 1}$, i.e. if the Backward Euler Formula is used, then the new numerical method (the combination of the underlying method, the sequential splitting procedure and the Richardson Extrapolation) is L-stable.

Proof: It is immediately seen that the right-hand-side of (5.51) is zero and, thus, the combined method is L-stable when the Backward Euler Formula is the underlying method.

■

5.4. Some numerical experiments

We shall use again, as in the previous chapter, the atmospheric chemical scheme in order to demonstrate the usefulness of the results described in the previous sections of this chapter. More precisely, the following actions are carried out in this section:

(a) The atmospheric chemical scheme is split into two parts.

(b) The organization of the computations is carried out as in the previous chapter.

(c) Numerical results obtained by using the sequential splitting scheme and the Backward Euler Formula are given.

(d) Some results, which are obtained by using the sequential splitting with θ-method with $\boldsymbol{\theta = 0.75}$ are shown.

(e) Results from runs of the sequential splitting with the Trapezoidal Rule are also given.

(f) Some conclusions related to the numerical results are drawn at the end of Section 5.4.

5.4.1. Splitting the atmospheric chemical scheme

The atmospheric chemical scheme with $\boldsymbol{56}$ species from the Unified Danish Eulerian Model (UNI-DEM), which was considered in the previous chapter will be considered also here. In this chapter, the atmospheric chemical scheme is split into two parts. The first part, $\mathbf{f_1(t, y)}$, contains mainly the chemical reactions in which ozone participates. The second part, $\mathbf{f_2(t, y)}$, contains all the other chemical reactions.

5.4.2. Organization of the computations

The same approach as in the previous chapters will be used also in this section. This means that the computations were organized as in Section 4.6, where more details about the atmospheric chemical scheme are given. Assume that $\bar{\mathbf{k}} = \mathbf{11}$ runs are to be carried out. The errors calculated during an arbitrary step $\mathbf{j}$ of run $\mathbf{k}$, $\mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots, \bar{\mathbf{k}}$ are estimated by using the formula (4.60). The number of grid-points, at which the error is estimated, is $\mathbf{168}$ for any of the $\bar{\mathbf{k}} = \mathbf{11}$ runs. Only the values of the reference solution at the grid-points of the coarse grid (which is used in the first run) have been stored and applied in the evaluation of the error (it is, of course, also possible to store all values of the reference solution, but such an action will increase tremendously the storage requirements). It is much more important and must be emphasized also in this chapter that errors of the calculated approximations were always, in all the nineteen runs, computed at the **same** $\mathbf{168}$ grid points. The global error made at run $\mathbf{k}$, $\mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots, \bar{\mathbf{k}} = \mathbf{11}$ is estimated by using formula (4.61). All computations in this section were performed as in the previous sections by selecting **quadruple precision** (i.e. by using **REAL*16** declarations for the real numbers and, thus, about 32-digit arithmetic) in an attempt to eliminate completely the influence of the rounding errors in the first 16 significant digits of the computed approximate solutions.

5.4.3. Results obtained when the Backward Euler Formula is used

The Backward Euler Formula (obtained when $\boldsymbol{\theta} = \mathbf{1}$ is used) was run in combination

(a) only with the sequential splitting procedure

and

(b) with both the sequential splitting procedure and with the Richardson Extrapolation.

Results are given in **Table 5.1**.

It is clearly seen that the application of the Richardson Extrapolation together with the Backward Euler Formula and the sequential splitting procedure results in a second-order numerical method and, thus, the results are significantly more accurate.

5.4.4. Results obtained when the θ-method with $\boldsymbol{\theta} = \mathbf{0.75}$ is used

The θ-method with $\boldsymbol{\theta} = \mathbf{0.75}$ was run as in the previous sub-section in combination

(a) only with the sequential splitting procedure

and

(b) with both the sequential splitting procedure and with the Richardson Extrapolation.

Results are given in **Table 5.2**. It is again clearly seen that the application of the Richardson Extrapolation together with the θ-method with $\boldsymbol{\theta} = \mathbf{0.75}$ and the sequential splitting procedure results in a second-order numerical method and, thus, in significantly more accurate results. Moreover, comparing the results in **Table 5.2** with those in **Table 5.1**, shows clearly that the results presented in

this paragraph are more accurate than those presented in the previous one (in Chapter 4 it was explained that the θ-method with $\theta = 0.75$ is more accurate than the Backward Euler Formula).

Job Number	Number of time-steps	Only sequential splitting		Richardson Extrapolation	
		Accuracy	Rate	Accuracy	Rate
1	168	5.965E-00	-	7.521E-01	-
2	336	1.214E-00	4.147	1.261E-01	5.965
3	672	4.294E-01	2.827	4.542E-02	2.776
4	1344	2.154E-01	1.993	1.799E-02	2.525
5	2688	1.093E-01	1.970	5.862E-03	3.068
6	5376	5.509E-02	1.985	1.698E-03	3.453
7	10752	2.764E-02	1.993	4.598E-04	3.692
8	21504	1.384E-03	1.997	1.199E-04	3.835
9	43008	6.926E-03	1.998	3.062E-05	3.915
10	86016	3.464E-03	1.999	7.740E-06	3.956
11	172032	1.733E-03	2.000	1.946E-06	3.978

Table 5.1

Numerical results that are obtained in 11 runs when **the Backward Euler Formula** is used **(a)** only with the sequential splitting procedure and **(b)** with both the sequential splitting procedure and the Richardson Extrapolation. The errors are given in the columns under "Accuracy". The ratios of two successive errors (calculated in an attempt to measure the rate of convergence) are given in the columns under "Rate".

Job Number	Number of time-steps	Only sequential splitting		Richardson Extrapolation	
		Accuracy	Rate	Accuracy	Rate
1	168	4.081E-00	-	5.843E-02	-
2	336	5.965E-01	6.842	3.681E-02	1.587
3	672	2.087E-01	2.859	1.863E-02	1.976
4	1344	1.052E-01	1.984	6.797E-03	2.741
5	2688	5.386E-02	1.952	2.1.05E-03	3.229
6	5376	2.731E-02	1.972	5.921E-04	3.555
7	10752	1.376E-02	1.985	1.576E-04	3.756
8	21504	6.904E-03	1.993	4.073E-05	3.869
9	43008	3.459E-03	1.996	1.037E-05	3.928
10	86016	1.731E-03	1.999	2.620E-06	3.957
11	172032	8.659E-04	1.999	6.597E-07	3.972

Table 5.2

The same as **Table 5.1** but for the case when the θ-method with $\theta = 0.75$ is used instead of the Backward Euler Formula.

5.4.4. Results obtained when the Trapezoidal Rule is used

The Trapezoidal Rule obtained with $\boldsymbol{\theta = 0.5}$ was run as in the previous two sub-sections in combination

(a) only with the sequential splitting procedure

and

(b) with both the sequential splitting procedure and with the Richardson Extrapolation.

Results are given in **Table 5.3**. It is seen that the application of the second-order Trapezoidal Rule with the sequential splitting results in a stable numerical scheme, but its order of accuracy is one. The application of the Trapezoidal Rule with the sequential splitting and with the Richardson Extrapolation results in an unstable numerical algorithm. This should be expected (see Theorem 5.4).

Job Number	Number of time-steps	Only sequential splitting		Richardson Extrapolation	
		Accuracy	Rate	Accuracy	Rate
1	168	2.419E-00	-	not stable	-
2	336	1.881E-01	12.859	not stable	n.a
3	672	2.807E-01	6.704	not stable	n.a
4	1344	7.317E-01	3.836	not stable	n.a
5	2688	2.670E-01	2.741	not stable	n.a
6	5376	1.335E-02	1.999	not stable	n.a
7	10752	6.678E-02	2.000	not stable	n.a
8	21504	3.339E-03	2.000	not stable	n.a
9	43008	1.670E-03	2.000	not stable	n.a
10	86016	8.349E-03	2.000	not stable	n.a
11	172032	4.174E-03	2.000	not stable	n.a

Table 5.3

The same as Table 1 but for the case when **the Trapezoidal Rule** is used instead of the Backward Euler Formula; "not stable" means that the method is not stable, "n.a" means not applicable.

5.4.6. Some conclusions from the numerical experiments

Several conclusions can be drawn from the results of the numerical experiments that were presented in **Table 5.1 – Table 5.3**:

(A) The sequential procedure behaves, when it is stable, always as a numerical method of order one. This is also true when this procedure is used together with the Trapezoidal Rule, the order of accuracy of which is two (although in the first runs the order of accuracy in this case seems to be greater than one).

(B) The θ-method with $\boldsymbol{\theta = 0.75}$ produces more accurate results than the Backward Euler formula. The errors are reduced approximately by a factor of two, (the reason for this was explained in the previous chapter).

(C) The accuracy achieved by using the combination of the sequential procedure with the Trapezoidal Rule, which is a second order numerical method, are less accurate than those obtained when the other two methods, which are first-order numerical algorithms, are used; compare the results in Table 5. 3 with the results in the previous two tables. It is not very clear what is the reason for this behavior.

(D) The combination of the sequential procedure with the two first-order numerical methods and the Richardson Extrapolation leads, in accordance with Theorem 5.1, to a stable computational process, its order of accuracy being two.

(E) As should be expected from Theorem 5.1, the combination of the sequential procedure, the Trapezoidal Rule and the Richardson Extrapolation leads to an unstable computational process.

5.5. Marchuk-Strang splitting procedure

Only θ-methods were used in the previous sections of this chapter. This is justified, because the combination of any numerical method for solving systems of ordinary differential equations with the sequential splitting procedure results in a new combined numerical method the order of accuracy of which is one (the additional application of the Richardson Extrapolation leading to a new combined method of order two). In this section some results related to the application of the Marchuk-Strang splitting procedure will be used. Its order of accuracy is two. Therefore, it is necessary to apply numerical methods of higher order in the treatment of the systems of ODEs. Some Runge-Kutta methods will be used in the remaining part of this section.

5.5.1. Some introductory remarks

Consider again, as in the first four chapters and as in the previous sections of this chapter, the system of ordinary differential equations (ODEs) defined by

$$(\mathbf{5.54}) \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{f(t,y)}, \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1}, \quad \mathbf{y(a)} = \boldsymbol{\eta} \ .$$

Assume that a Runge-Kutta method, explicit or implicit, see, for example, **Burrage (1995)**, **Butcher(2003)**, **Hairer, Nørsett and Wanner (1987)**, **Hundsdorfer and Verwer (2003)**, **Lambert(1991)**, is used in the numerical solution of (5.54). Assume furthermore that the time-interval is discretized by using an equidistant grid (but it should be emphasized that this assumption is made only in order to simplify the presentation of the results; most of the results in the following part of this chapter are also valid when non-equidistant grids are used):

$$(5.55) \quad t_0 = a\,, \quad t_n = t_{n-1} + h = t_0 + nh, \quad n = 1, 2, \ldots, N, \quad t_N = b, \quad h > \frac{b-a}{N}.$$

Under these assumptions, the Runge-Kutta methods can be based on the following formulae:

$$(5.56) \quad y_n = y_{n-1} + h \sum_{i=1}^{m} c_i k_i(f, t_{n-1}, t_n; h)\,,$$

where for $i = 1, 2, 3, \ldots, m$:

$$(5.57) \quad k_i(f, t_{n-1}, t_n; h) = f\left(t_{n-1} + h\, a_i\,, y_{n-1} + h \sum_{j=1}^{m} b_{ij}\, k_j(f, t_{n-1}, t_n; h) \right), \quad a_i = \sum_{j=1}^{m} b_{ij}\,,$$

The coefficients c_i and b_{ij}, $i, j = 1, 2, 3, \ldots, m$, are constants dependent on the selected particular Runge-Kutta method. If all $b_{ij} = 0$ when $j \geq i$, then the selected Runge-Kutta method is explicit, while the method is implicit when at least one $b_{ij} \neq 0$ when $j \geq i$. The vectors $k_i(f, t_{n-1}, t_n; h)$ are called stages. Simple notation, k_i instead of $k_i(f, t_{n-1}, t_n; h)$, is often used (and was also applied in the previous chapters). The above notation will simplify the introduction of the combination of the Marchuk-Strang splitting procedure with Runge-Kutta methods in Sub-section 5.5.2.

We are interested in the case where the method based on (5.55)-(5.57) is used in a combination with both the Marchuk-Strang splitting procedure, see **Marchuk (1968, 1980, 1982, 1986, 1988)** and **Strang (1968)** and the Richardson Extrapolation, see **Faragó, Havasi and Zlatev (2010)** and **Richardson (1911, 1927)**. This combination is a new numerical method and we shall study some stability properties of the combined numerical method both in the general case when the Runge-Kutta methods is defined by (5.56)-(5.57) and in some particular cases. The accuracy and the stability of the obtained results will be validated by applying an atmospheric chemical scheme, which is described mathematically by a non-linear system of 56 ordinary differential equations. It is convenient to re-iterate here that this chemical scheme has successfully been used in the Unified Danish Eulerian Model, **UNI-DEM,** which is a large-scale mathematical model for studying:

(a) transport of air pollution over Europe (**Alexandrov et al., 2004**, **Ambelas Skjøth et al., 2000**, **Bastrup-Birk et al., 1997**, **Hass et al., 2004**, **Roemer et al., 2004**, **Zlatev, 2010**, **Zlatev and Dimov, 2006**,

(b) pollution distribution in different countries (**Abdalmogith, Harrison and Zlatev, 2004**, **Harrison, Zlatev and Ottley, 1994**, **Havasi and Zlatev, 2002**, **Zlatev, Georgiev and Dimov, 2013b**, **Zlatev and Syrakov, 2004a, 2004b**)

and

(c) impact of climate changes on some critical air pollution levels (**Csomós et al., 2006**, **Zlatev, 2010**, **Zlatev, Faragó and Havasi, 2010, Zlatev, Dimov and Georgiev, 2016, Zlatev, Georgiev and Dimov, 2013b**, **Zlatev, Havasi and Faragó, 2011**, **Zlatev and Moseholm, 2008**).

The contents of this section of Chapter 5 can be outlined as follows. The application of the Marchuk-Strang splitting procedure together with the Richardson Extrapolation is discussed in **Sub-section 5.5.2**. An expression for the stability function of the combination of the Runge-Kutta methods with the Marchuk-Strang splitting procedure and the Richardson Extrapolation is derived in **Sub-section 5.5.3**. Some special Runge-Kutta methods, the two-stage Diagonally Implicit Runge-Kutta (DIRK) methods and the Implicit Mid-point Rule, will be introduced in **Sub-section 5.5.4**. Absolute stability regions of the selected methods will be presented in **Sub-section 5.5.5.** Numerical results, based on the application of the atmospheric chemical scheme with $\mathbf{56}$ species, will be given in **Sub-section 5.5.6**. Several conclusions will be drawn in **Sub-section 5.5.7**.

5.5.2. The Marchuk-Strang splitting procedure and the Richardson Extrapolation

Rewrite, as in the previous sections, the initial value problem given in (5.54) in the following form:

$$\mathbf{(5.58)} \qquad \frac{\mathbf{dy}}{\mathbf{dt}} = \mathbf{f_1(t,y)} + \mathbf{f_2(t,y)}, \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_1} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_2} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1},$$

where $\mathbf{f_1(t,y)} + \mathbf{f_2(t,y)} = \mathbf{f(t,y)}$ and $\mathbf{y(a)} = \boldsymbol{\eta}$ being again a given initial value. Consider two systems of ODEs defined by

$$\mathbf{(5.59)} \qquad \frac{\mathbf{dy^{[1]}}}{\mathbf{dt}} = \mathbf{f_1}\left(\mathbf{t},\mathbf{y^{[1]}}\right), \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y^{[1]}} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_1} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1},$$

$$\mathbf{(5.60)} \qquad \frac{\mathbf{dy^{[2]}}}{\mathbf{dt}} = \mathbf{f_2}\left(\mathbf{t},\mathbf{y^{[2]}}\right), \quad \mathbf{t} \in [\mathbf{a},\mathbf{b}], \quad \mathbf{a} < \mathbf{b}, \quad \mathbf{y^{[2]}} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{f_2} \in \mathbb{R}^{\mathbf{s}}, \quad \mathbf{s} \geq \mathbf{1},$$

It is normally assumed that it is easier (or even much easier) to solve numerically any of the two systems (5.59) and (5.60) than to solve the system of ODEs (5.58). The combined use of the Marchuk-Strang splitting procedure and the Richardson Extrapolation consists of the following three steps:

Step 1

Apply any algorithm from the class of the Runge-Kutta methods defined by formulae (5.56)-(5.57) and the Marchuk-Strang splitting procedure to calculate an approximation $\mathbf{z_n \approx y(t_n)}$ by using the following formulae:

$$\mathbf{(5.61)} \quad \mathbf{z_{n-0.5}^{[1]} = y_{n-1} + 0.5h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-1}, t_{n-0.5}; 0.5h)}$$

$$\mathbf{(5.62)} \quad \mathbf{z_n^{[2]} = z_{n-0.5}^{[1]} + h \sum_{i=1}^{m} c_i k_i(f_2, t_{n-1}, t_n; h)}$$

$$\mathbf{(5.63)} \quad \mathbf{z_n^{[3]} = z_n^{[2]} + 0.5h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-0.5}, t_n; 0.5h)}$$

$$\mathbf{(5.64)} \quad \mathbf{z_n = z_n^{[3]}}$$

■

Step 2

Apply twice the same algorithm from the class of Runge-Kutta methods defined by formulae (5.56)-(5.57) and the Marchuk-Strang splitting procedure to calculate another approximation $\mathbf{w_n \approx y(t_n)}$ by using the following formulae:

$$\mathbf{(5.65)} \quad \mathbf{w_{n-0.75}^{[1]} = y_{n-1} + 0.25h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-1}, t_{n-0.75}; 0.25h)}$$

$$\mathbf{(5.66)} \quad \mathbf{w_{n-0.5}^{[2]} = w_{n-0.75}^{[1]} + 0.5h \sum_{i=1}^{m} c_i k_i(f_2, t_{n-1}, t_{n-0.5}; 0.5h)}$$

$$\mathbf{(5.67)} \quad \mathbf{w_{n-0.5}^{[3]} = w_{n-0.5}^{[2]} + 0.25h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-0.75}, t_{n-0.5}; 0.25h)}$$

$$(5.68)\quad \mathbf{w_{n-0.25}^{[1]} = w_{n-0.5}^{[3]} + 0.25h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-0.5}, t_{n-0.25}; 0.25h)}$$

$$(5.69)\quad \mathbf{w_{n}^{[2]} = w_{n-0.25}^{[1]} + 0.5h \sum_{i=1}^{m} c_i k_i(f_2, t_{n-0.5}, t_{n}; 0.5h)}$$

$$(5.70)\quad \mathbf{w_{n}^{[3]} = w_{n}^{[2]} + 0.25h \sum_{i=1}^{m} c_i k_i(f_1, t_{n-0.25}, t_{n}; 0.25h)}$$

$$(5.71)\quad \mathbf{w_n = w_n^{[3]}}$$

■

Step 3

Form the Richardson Extrapolation:

$$(5.72)\quad \mathbf{y_n = \frac{4w_n - z_n}{3}}\ .$$

■

The Richardson Extrapolation formula $\mathbf{y_n = (2^p w_n - z_n)/(2^p - 1)}$ is used with $\mathbf{p = 2}$ in (5.72). This means that the order of accuracy of the selected underlying Runge-Kutta method should be at least **two**.

Note that although the two auxiliary and simpler problems (5.59) and (5.60) are used in the computations involved in the three steps described above, the calculated by (5.72) vector $\mathbf{y_n}$ is an approximation of the original problem (5.54).

5.5.3. Stability function of the combined numerical method

The stability properties of the combined numerical method, the combination of the selected Runge-Kutta method with the Marchuk-Strang splitting procedure and the Richardson Extrapolation, will be studied in this section. We shall start with the stability properties of the underlying Runge-Kutta method, which are often studied by using the famous test-equation introduced in **Dahlquist (1963)**, see also **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Hairer and Wanner (1991)**, **Hundsdorfer and Verwer (2003)**, **Lambert (1991)**:

$$(5.73) \qquad \frac{dy}{dt} = \lambda y,$$

$$t \in [0, \infty], \qquad y \in \mathbb{C}, \qquad \lambda = \bar{\alpha} + \bar{\beta} i \in \mathbb{C}, \qquad \bar{\alpha} \leq 0, \qquad y(0) = \eta \in \mathbb{C}.$$

Denote $h\lambda = \mu$. Then the stability of the computations with any Runge-Kutta method is related to a stability function $R(\mu)$. If the particular Runge-Kutta method is explicit, then this function is a polynomial, while it is a rational function (a ratio of two polynomials) when the method is implicit. The stability function of a special class of Runge-Kutta methods will be derived in the next section. For our purposes here, it is important to emphasize two facts:

(a) if for a given value of $\mu \in \mathbb{C}^-$ we have $R(\mu) \leq 1$, then the computations with the particular method will remain stable for the stepsize h when (5.73) is solved (and one should expect that the computational process will remain stable also when other problems are to be handled)

and

(b) the stability function of the combined numerical method (the particular Runge-Kutta method + the Marchuk-Strang splitting procedure + the Richardson Extrapolation) can be expressed by the stability function $R(\mu)$ of the underlying Runge-Kutta method.

The second fact is very important. It is telling us that if we know the stability function of the underlying Runge-Kutta method then we shall be able to calculate easily the stability function of the combined method. We can formulate this statement in a strict manner as follows:

Theorem 5.2: Consider an arbitrary Runge-Kutta method, explicit or implicit, with a stability function $R(\mu)$ and combine it with the Marchuk-Strang splitting procedure and with the Richardson Extrapolation. Then the absolute stability function $\bar{R}(\mu)$ of the combined method is given by

$$(5.74) \qquad \bar{R}(\mu) = \frac{\left[R\left(\frac{\mu}{2}\right)\right]^2 \left\{4\left[R\left(\frac{\mu}{4}\right)\right]^4 - R(\mu)\right\}}{3}.$$

Proof: Consider **first** two different special problems that are quite similar to (5.73):

$$(5.75) \qquad \frac{dy^{[1]}}{dt} = \lambda_1 y^{[1]}, \quad t \in [0, \infty], \quad y^{[1]} \in \mathbb{C}, \quad \lambda_1 = \alpha_1 + \beta_1 i, \quad \alpha_1 \leq 0, \quad y(0) = \eta,$$

$$(5.76) \qquad \frac{dy^{[2]}}{dt} = \lambda_2 y^{[2]}, \quad t \in [0,\infty], \quad y^{[2]} \in \mathbb{C}, \quad \lambda_2 = \alpha_2 + \beta_2 i, \quad \alpha_2 \leq 0, \quad y(0) = \eta ,$$

Denote $h\lambda_1 = \mu_1$ and $h\lambda_2 = \mu_2$. The assumption that the selected Runge-Kutta method has a stability function $R(\mu)$ means that the relationship $y_n = R(\mu)y_{n-1}$ is satisfied when a time-step with a step-size h is carried out to obtain an approximation y_n of the solution of (5.73) by using y_{n-1} as a starting approximation, see, for example, **Lambert (1991)**. It is clear that a similar relationship is satisfied, then the problems (5.75) and (5.76) are handled. By using this fact in connection to (5.61), (5.62), (5.63) and (5.64), we can successively obtain:

$$(5.77) \qquad z_{n-0.5}^{[1]} = R\left(\frac{\mu_1}{2}\right) y_{n-1} ,$$

$$(5.78) \qquad z_{n-0.5}^{[2]} = R(\mu_2) z_{n-0.5}^{[1]} = R(\mu_2) R\left(\frac{\mu_1}{2}\right) y_{n-1} ,$$

$$(5.79) \qquad z_n = z_n^{[3]} = R\left(\frac{\mu_1}{2}\right) z_{n-0.5}^{[2]} = R\left(\frac{\mu_1}{2}\right) R(\mu_2) R\left(\frac{\mu_1}{2}\right) y_{n-1} = \left[R\left(\frac{\mu_1}{2}\right)\right]^2 R(\mu_2)\, y_{n-1} .$$

In the same way, by using this time the equalities (5.65)-(5.71), it will be possible to obtain:

$$(5.80) \qquad w_n = \left[R\left(\frac{\mu_1}{4}\right)\right]^4 \left[R\left(\frac{\mu_2}{2}\right)\right]^2 y_{n-1}$$

Use now (5.72) to derive the following expression:

$$(5.81) \qquad y_n = \frac{4\left[R\left(\frac{\mu_1}{4}\right)\right]^4 \left[R\left(\frac{\mu_2}{2}\right)\right]^2 - \left[R\left(\frac{\mu_1}{2}\right)\right]^2 R(\mu_2)}{3} y_{n-1} .$$

Assume that $\lambda_1 = \lambda_2 = \lambda$, which leads to $\mu_1 = \mu_2 = \mu$. Then we have:

$$(5.82) \qquad y_n = \frac{4\left[R\left(\frac{\mu}{4}\right)\right]^4 \left[R\left(\frac{\mu}{2}\right)\right]^2 - \left[R\left(\frac{\mu}{2}\right)\right]^2 R(\mu)}{3} y_{n-1} .$$

It follows from (5.82) that the stability function of the combined numerical algorithm is

$$(5.83)\quad \bar{R}(\mu) = \frac{4\left[R\left(\frac{\mu}{4}\right)\right]^4\left[R\left(\frac{\mu}{2}\right)\right]^2 - \left[R\left(\frac{\mu}{2}\right)\right]^2 R(\mu)}{3} = \frac{\left[R\left(\frac{\mu}{2}\right)\right]^2\left\{4\left[R\left(\frac{\mu}{4}\right)\right]^4 - R(\mu)\right\}}{3},$$

which completes the proof of Theorem 5.2.

■

Remark 5.1: Equality (5.83) shows that the stability polynomial $\bar{R}(\mu)$ of the combined method depends only on the stability polynomial $R(\mu)$ of the underlying method, but this does not mean that if the underlying method is stable when the Dahlquist test-problem (5.73) is solved, then the combined method is also stable when this equation is handled. Indeed, $R(\mu) \leq 1$ does not necessarily imply $\bar{R}(\mu) \leq 1$. Therefore, the stability properties of the combined method have to be investigated very carefully also in the case where the underlying method is stable.

■

5.5.4. Selection of an appropriate class of Runge-Kutta methods

Good candidates of underlying methods when the combination a Runge-Kutta method + the Marchuk-Strang splitting procedure + the Richardson Extrapolation is to be used are the two-stage Diagonally Implicit Runge-Kutta (DIRK) methods, see Chapter 4 and **Alexander (1977)**, **Cruziex (1976)**, **Nørsett (1976)**, which can be introduced by the following formulae:

$$(5.84)\quad y_n = y_{n-1} + h\left[c_1 k_1(f, t_{n-1}, t_n; h) + c_2 k_2(f, t_{n-1}, t_n; h)\right], \qquad c_1 + c_2 = 1,$$

where

$$(5.85)\quad k_1(f, t_{n-1}, t_n; h) = f\left(t_{n-1} + h\gamma,\ y_{n-1} + h\gamma k_1(f, t_{n-1}, t_n; h)\right),$$

$$k_2(f, t_{n-1}, t_n; h) = f\left(t_{n-1} + ha_2, y_{n-1} + hb_{21}k_1(f, t_{n-1}, t_n; h) + h\gamma k_2(f, t_{n-1}, t_n; h)\right).$$

The advantages of using the method introduced by (5.84) and (5.85) are three:

(a) The order of accuracy of the Marchuk-Strang splitting procedure is two and, therefore, it is desirable to use an underlying method, which is at least of order two also. That is easily achievable when the class of DIRK methods defined by (5.84) and (5.85) is selected.

(b) The implementation of any method defined by (5.84) and (5.85) leads to the solution of **two** small systems of $\mathbf{s}$ algebraic equations while a general two-stage Runge-Kutta method will lead to the solution of **one** large system of $\mathbf{2s}$ algebraic equations.

(c) One has to treat $\mathbf{s \times s}$ matrices during every time-step, while $\mathbf{2s \times 2s}$ have to be treated if a general two-stage Runge-Kutta method is used.

More details about the advantages of **the DIRK methods** can be found in **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Hairer and Wanner(1991)**, **Hundsdorfer and Verwer (2003)**, **Lambert (1991)**.

The stability function of any numerical algorithm belonging to the class of the two-stage DIRK methods defined by (5.84) and (5.85) can be derived as follows. Assume again that the Dahlquist test-equation (5.73) is used. The following formulae can be derived by using the relation $\mathbf{f(t,y) = \lambda y}$, which follows from the comparison of (5.54) with (5.73) and from the notation $\mathbf{h\lambda = \mu}$:

$$\mathbf{k_1(f, t_{n-1}, t_n; h) = \lambda\left(y_{n-1} + h\gamma\, k_1(f, t_{n-1}, t_n; h)\right) = \lambda y_{n-1} + \gamma\mu\, k_1(f, t_{n-1}, t_n; h)}, \quad \mathbf{(5.86)}$$

$$\mathbf{k_1(f, t_{n-1}, t_n; h) - \gamma\mu\, k_1(f, t_{n-1}, t_n; h) = \lambda y_{n-1}}, \quad \mathbf{(5.87)}$$

$$\mathbf{k_1(f, t_{n-1}, t_n; h) = \frac{1}{1-\gamma\mu}\lambda y_{n-1}}\ . \quad \mathbf{(5.88)}$$

Similar, but slightly more complicated calculations lead to

$$\mathbf{k_2(f, t_{n-1}, t_n; h) = \frac{1-\gamma\mu+b_{21}\mu}{(1-\gamma\mu)^2}\,\lambda y_{n-1}}\ . \quad \mathbf{(5.89)}$$

Insert the right-hand-sides of (5.88) and (5.89) in (5.84) to obtain the following result after some not very complicated computations:

$$\mathbf{y_n = \frac{(1-\gamma\mu+\mu)(1-\gamma\mu)+c_{21}b_{21}\mu^2}{(1-\gamma\mu)^2}\,y_{n-1}}\ , \quad \mathbf{(5.90)}$$

The last formula shows that **the stability function** of the class of the two-stage Diagonally Implicit Runge-Kutta (DIRK) methods is the expression given below:

$$(5.91) \qquad \mathbf{R}^{[\mathbf{DIRK}]}_{[\mathbf{2-stage}]}(\boldsymbol{\mu}) = \frac{(\mathbf{1}-\boldsymbol{\gamma\mu}+\boldsymbol{\mu})(\mathbf{1}-\boldsymbol{\gamma\mu})+\mathbf{c_2 b_{21}}\boldsymbol{\mu}^{\mathbf{2}}}{(\mathbf{1}-\boldsymbol{\gamma\mu})^{\mathbf{2}}} \ .$$

According to Theorem 1, the stability function of the combined method (the two-stage DIRK method + the Marchuk-Strang splitting + the Richardson Extrapolation) can be obtained by using the formula:

$$(5.92) \qquad \overline{\mathbf{R}}^{[\mathbf{DIRK}]}_{[\mathbf{2-stage}]}(\boldsymbol{\mu}) = \frac{\left[\mathbf{R}^{[\mathbf{DIRK}]}_{[\mathbf{2-stage}]}\left(\frac{\boldsymbol{\mu}}{\mathbf{2}}\right)\right]^{\mathbf{2}}\left\{\mathbf{4}\left[\mathbf{R}^{[\mathbf{DIRK}]}_{[\mathbf{2-stage}]}\left(\frac{\boldsymbol{\mu}}{\mathbf{4}}\right)\right]^{\mathbf{4}} - \mathbf{R}^{[\mathbf{DIRK}]}_{[\mathbf{2-stage}]}(\boldsymbol{\mu})\right\}}{\mathbf{3}} .$$

It is necessary now to select a particular method from the class of the two-stage DIRK methods. This can be done by the following choice of the coefficients:

$$(5.93) \qquad \boldsymbol{\gamma} = \mathbf{1} - \frac{\sqrt{\mathbf{2}}}{\mathbf{2}}, \qquad \mathbf{a_2} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}}, \qquad \mathbf{b_{21}} = \sqrt{\mathbf{2}} - \mathbf{1}, \qquad \mathbf{c_1} = \mathbf{c_2} = \mathbf{0.5} \ .$$

The resulting method is A-stable. The concept of A-stability was defined above, see more details in **Burrage (1995)**, **Butcher (2003)**, **Hairer, Nørsett and Wanner (1987)**, **Hairer and Wanner(1991)**, **Hundsdorfer and Verwer (2003)**, **Lambert (1991)**. More about the particular method obtained by the choice made in (5.93) can be found in **Zlatev (1981)**.

We shall also use in the experiments the one-stage second-order Runge-Kutta method, which is much better known under the name Implicit Mid-point Rule:

$$(5.94) \qquad \mathbf{y_n} = \mathbf{y_{n-1}} + \mathbf{h}\ \mathbf{k_1}(\mathbf{f}, \mathbf{t_{n-1}}, \mathbf{t_n}; \mathbf{h}) = \mathbf{y_{n-1}} + \mathbf{hf}(\mathbf{t_{n-0.5}}, \mathbf{0.5}(\mathbf{y_n} + \mathbf{y_{n-1}})) \ .$$

This method belongs also to the class of the so-called **one-leg** methods, it is A-stable and it is identical with the Trapezoidal Rule when it is used to handle systems of **linear** ODEs, see more details in **Lambert (1991)**. The stability function of the Mid-point Rule, which is the same as the stability function of the Trapezoidal Rule, is

$$(5.95) \qquad \mathbf{R}^{[\mathbf{Mid-point}]}_{[\mathbf{1-stage}]}(\boldsymbol{\mu}) = \frac{\mathbf{1}+\mathbf{0.5}\boldsymbol{\mu}}{\mathbf{1}-\mathbf{0,5}\boldsymbol{\mu}} \ .$$

The application of the assertion of Theorem 1 to the Implicit Mid-Point Rule combined with the Marchuk-Strang splitting procedure and the Richardson Extrapolation leads to the following expression for the stability function:

$$(5.96)\quad \bar{R}^{[\text{Mid-point}]}_{[1-\text{stage}]}(\mu) = \frac{\left[R^{[\text{Mid-point}]}_{[1-\text{stage}]}\left(\frac{\mu}{2}\right)\right]^2 \left\{4\left[R^{[\text{Mid-point}]}_{[1-\text{stage}]}\left(\frac{\mu}{4}\right)\right]^4 - R^{[\text{Mid-point}]}_{[1-\text{stage}]}(\mu)\right\}}{3}.$$

The stability properties of the particular numerical algorithm from the class of the two-stage DIRK methods, the coefficients of which are given in (5.93), and the Implicit Mid-point Rule, when these are combined with the Marchuk-Strang splitting procedure and the Richardson Extrapolation, will be studied in the next section.

5.5.5. Absolute stability regions of the combined numerical methods

It is difficult to investigate the stability properties of the combinations of the two selected methods with Marchuk-Strang splitting procedure and the Richardson Extrapolation. Indeed, the stability functions of the underlying methods are ratios of complex polynomials of degree two, while the corresponding degrees of the complex polynomials in the combined methods are eight. This is why the same approach as that successfully used in the first previous chapters as well as in **Zlatev et al. (2016)** was also applied in this case.

It was established that the important inequality $|\bar{R}(\nu)| \leq 1$ holds in a big square domain with vertices: $(0.0\,,\ 0.0)$, $(0.0\,,\ 10^5\,i)$, $(-10^5\,i\,,\ 10^5\,i)$ and $(-10^5\,,\ 0.0)$ for the two-stage DIRK method combined with the Marchuk-Strang splitting procedure and the Richardson Extrapolation. A part of the domain, in which the combined method is **absolutely stable** is given in **Fig. 5.6**. The same approach as that used in **Zlatev, Georgiev and Dimov (2014)** was applied. It can be described as follows. Let μ be equal to $\bar{\alpha} + \bar{\beta}i$ with $\bar{\alpha} \leq 0$ and select some increment $\varepsilon > 0$ (we have chosen $\varepsilon = 0.01$). Start the calculations with $\bar{\alpha} = 0$ and compute successively the values $|\bar{R}(\nu)|$ for $\bar{\alpha} = 0$ and for $\bar{\beta} = 0,\ \varepsilon,\ 2\varepsilon,\ 3\varepsilon,\ \ldots$. Continue the calculations until $\bar{\beta}$ becomes equal to 10^5 under the condition that $|\bar{R}(\nu)|$ stays less than one during all these computations. Repeat this procedure for values of $\bar{\alpha}$ equal to $-\varepsilon,\ -2\varepsilon, -3\varepsilon,\ \ldots$ Continue to decrease $\bar{\alpha}$ until it becomes equal to -10^5 (requiring again $|\bar{R}(\nu)|$ stays always less than one). It is clear that one should expect that all points within the squares plotted in **Fig. 5.6** that are obtained by applying the above algorithm, belong to the absolute stability regions of the studied numerical methods.

The same approach applied to the combination of the Implicit Mid-point Rule with the Marchuk-Strang splitting procedure and the Richardson Extrapolation shows that the absolute stability region of this method is **finite** but quite large, see **Fig. 5.7**.

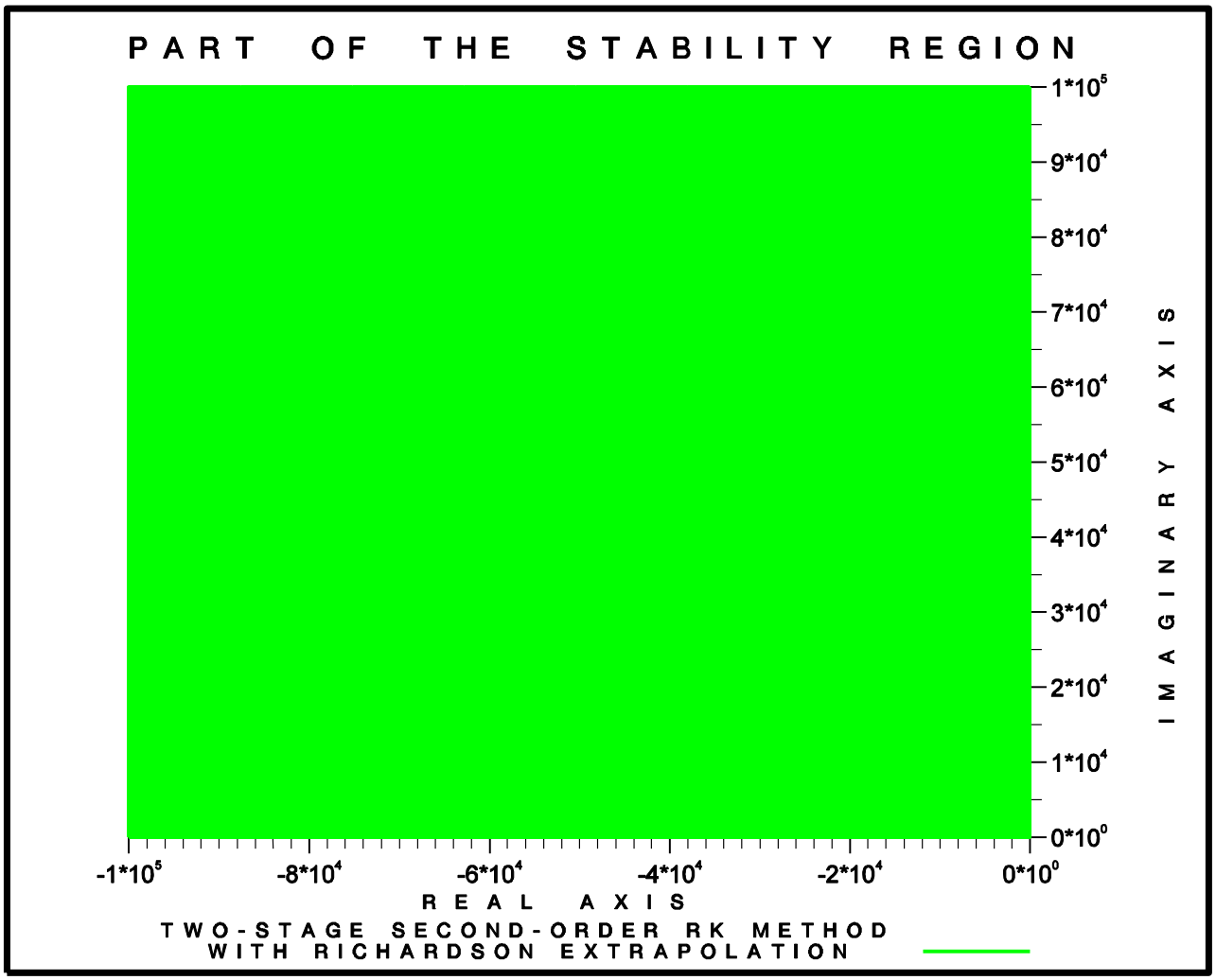

Figure 5.6

Part of the absolute stability region of the two-stage second-order diagonally implicit Runge-Kutta method defined with $\boldsymbol{\gamma = 1 - \sqrt{2}/2}$, $\mathbf{a_2 = \sqrt{2}/2}$, $\mathbf{b_{21} = \sqrt{2} - 1}$, $\mathbf{c_1 = c_2 = 0.5}$ and combined with the Marchuk-Strang splitting procedure and the Richardson Extrapolation.

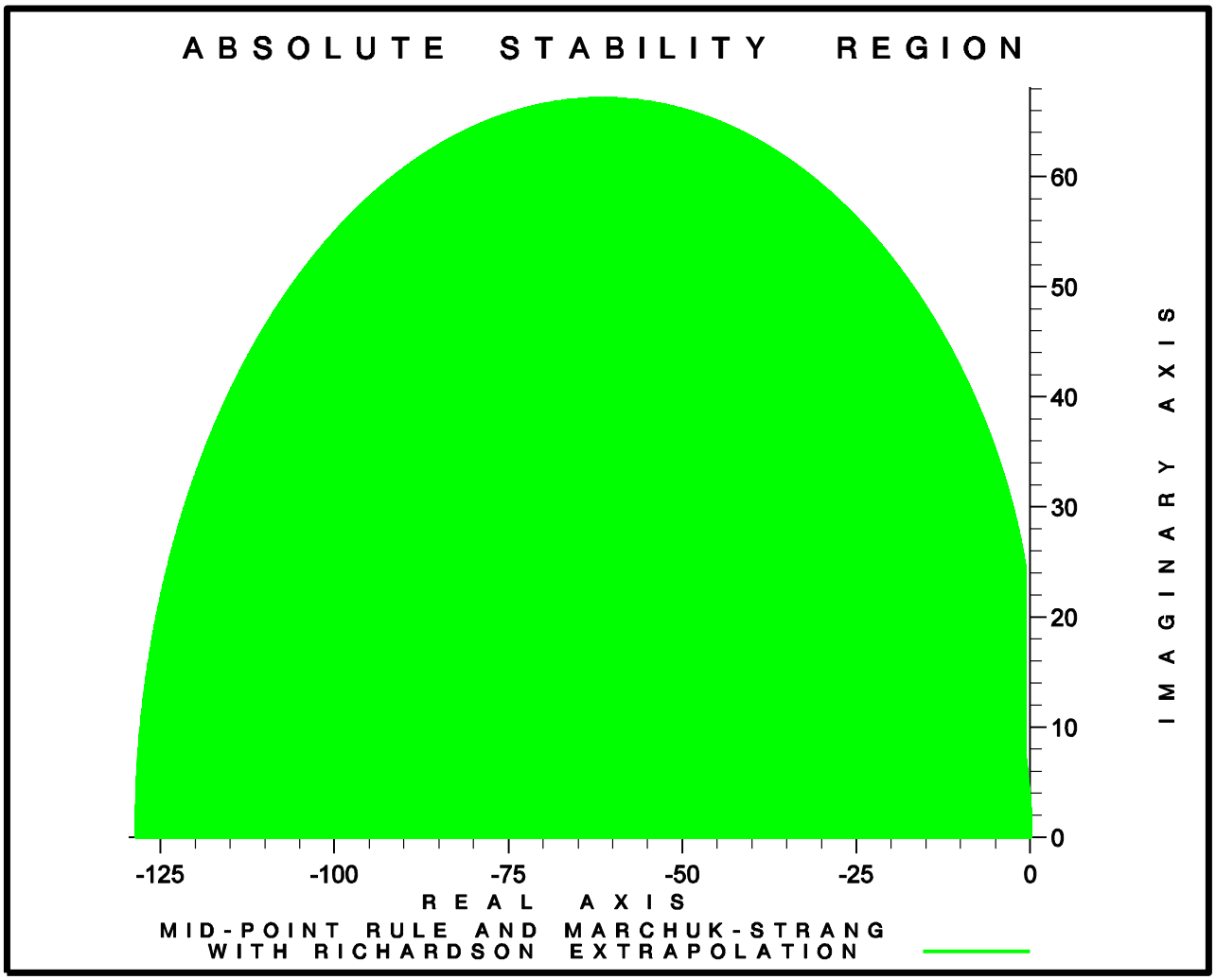

Figure 5.7
The absolute stability region of the Implicit Mid-point Rule combined with the Marchuk-Strang splitting procedure and the Richardson Extrapolation.

5.5.6. Some numerical results

As mentioned in the beginning of this section, an important module, taken from the Unified Danish Eulerian Model (**UNI-DEM**), see **Abdalmogith, Harrison and Zlatev (2004)**, **Alexandrov et al. (2004)**, **Ambelas Skøth et al. (2000)**, **Bastrup Birk et al. (1997)**, **Geenaert and Zlatev (2004)**, **Harrison, Zlatev and Ottley (1994)**, **Hass et al. (2004)**, **Havasi and Zlatev (2002)**, **Roemer et al. (2004]** , **Zlatev (1995)**, **Zlatev and Dimov (2006)**, will be used in several numerical experiments. This module is an **atmospheric chemical scheme** with $\mathbf{56}$ species, which is represented by a non-linear, badly scaled and stiff system of ODEs. The badly scaling of the components in the system of ODE's is demonstrated in **Table 5.4**. The maximal concentration of $\mathbf{CH_3CHO}$ is $\mathbf{6.9 \times 10^{10}}$, while the minimal concentration of $\mathbf{OD}$ is $\mathbf{6.5 \times 10^{-40}}$ i.e., the difference is about 50 orders of magnitude. The diurnal variations of many concentrations are both rapid and very large, which is shown in **Fig. 5.8** and **Fig. 5.9**. It is also seen that some of the concentrations are decreasing during the

night, while others are increasing in this period. Furthermore, the variations of the concentrations in these two rather short but very critical time-periods are extremely quick and create steep gradients.

Chemical species	Maximal concentrations	Minimal concentrations	Mean concentrations
CH_3CHO	6.9×10^{10}	6.4×10^{5}	6.8×10^{9}
N_2O_5	1.8×10^{9}	5.3×10^{4}	4.3×10^{8}
OH	2.3×10^{7}	3.3×10^{4}	6.3×10^{6}
OD	4.4×10^{-2}	2.5×10^{-40}	1.1×10^{-2}

Table 5.4

Orders of magnitude of the concentrations of four chemical species in a period of 24 hours; from 12 o'clock at the noon on a given day to 12 o'clock at the noon on the next day. Units: **(numbers of molecules) / (cubic centimetre**).

The condition numbers of the Jacobian matrices $\mathbf{J} = \partial \mathbf{f}/\partial \mathbf{t}$ appearing in the period of 24 hours were calculated at every time-step (by using LAPACK software from **Anderson et al. (1992)** and **Barker et al. (2001)**. Introduce the abbreviation $\mathbf{COND}$ for the condition number computed at any time-step. We found out that $\mathbf{COND} \in [\, 4.56 \times 10^{8}\,,\ 9.27 \times 10^{12}\,]$ during the time-period of 24 hours. This shows that difficulties might appear not only because the selected numerical scheme is not sufficiently accurate, but also because the rounding errors may interfere with the truncation errors, see, for example, **Hamming (1962)**, **Stewart (1973)**, **Wilkinson (1963, 1965)**.

It is necessary for the computer programs to define the time-interval in seconds. Then the chemical atmospheric scheme was handled on the interval $[\, \mathbf{a}, \mathbf{b}\,] = [\, 43200\,,\ 129600\,]$. The starting value $\mathbf{a} = 43200$ corresponds to 12 o'clock at the noon (measured in seconds and starting from mid-night), while $\mathbf{b} = 129600$ corresponds to 12 o'clock at the next day (measured also in seconds from the starting point).

Sequences of 19 runs were carried out and selected results are presented below. The first run was always performed by using $\mathbf{N} = 168$ time-steps, the time-stepsize being $\mathbf{h} \approx 514.285$ seconds. The time-stepsize $\mathbf{h}$ was halved after each run. This implies that the number of time-steps was doubled. The local error made at $\bar{\mathbf{t}}_{\mathbf{j}}$ in any of the runs was measured for $\mathbf{k} = 1\,,\ 2, \ldots\ ,\ 19$ by using the following formula:

$$(5.97) \qquad \mathbf{ERROR}_{\mathbf{j}}^{(\mathbf{k})} = \max_{\mathbf{i}=1,2,\ldots 56} \left(\frac{\left| \mathbf{y}_{\mathbf{i},\mathbf{j}} - \mathbf{y}_{\mathbf{i},\mathbf{j}}^{\mathbf{ref}} \right)\right|}{\max\left(\left| \mathbf{y}_{\mathbf{i},\mathbf{j}}^{\mathbf{ref}} \right|, 1.0 \right)} \right)\ ,\ \mathbf{j} = 2^{\mathbf{k}-1}, 2 \times 2^{\mathbf{k}-1}\ ,\ \ldots\ , 168 \times 2^{\mathbf{k}-1}\ ,$$

where $\mathbf{y}_{\mathbf{i},\mathbf{j}}$ and $\mathbf{y}_{\mathbf{i},\mathbf{j}}^{\mathbf{ref}}$ are respectively the calculated values and the values of the reference solution of the $\mathbf{i}^{\mathbf{th}}$ chemical species at $\bar{\mathbf{t}}_{\mathbf{j}} = \mathbf{t}_{\mathbf{0}} + \mathbf{j}\mathbf{h}_{\mathbf{0}}$ (where $\mathbf{j} = 1\,,\ 2,\ \ldots\ ,\ 168$ and $\mathbf{h}_{\mathbf{0}} \approx 514.285$ was the time-stepsize that has been used in the first run). The values of the reference solution were calculated, as in Chapter 4, by using a very accurate method (three-stage fifth-order L-stable Fully Implicit Runge-Kutta (FIRK) Method), **Ehle (1968)**, see also **Burrage (1995)**, **Butcher (2003)**,

Hairer, Nørsett and Wanner (1987), **Hairer and Wanner (1991)**, **Hundsdorfer and Verwer (2003)**, **Lambert (1991)**, with $\mathbf{N = 998244352}$ time-steps and a time-stepsize $\mathbf{h_{ref} \approx 6.1307634 \times 10^{-5}}$. This means that we estimate the local errors at the same set of grid-points in each of the $\mathbf{19}$ runs. Moreover, the number of grid-points, at which the error is estimated, is $\mathbf{168}$ for any of the $\mathbf{19}$ runs. It should also be mentioned that the values of the reference solution at the grid-points of the coarse grid used in the first run have been preserved and applied in the evaluation of the local errors. It is possible to store all values of the reference solution, but such an action will increase too much the storage requirements. It is more important that the local errors of the calculated approximations were computed at the **same** $\mathbf{168}$ grid points.

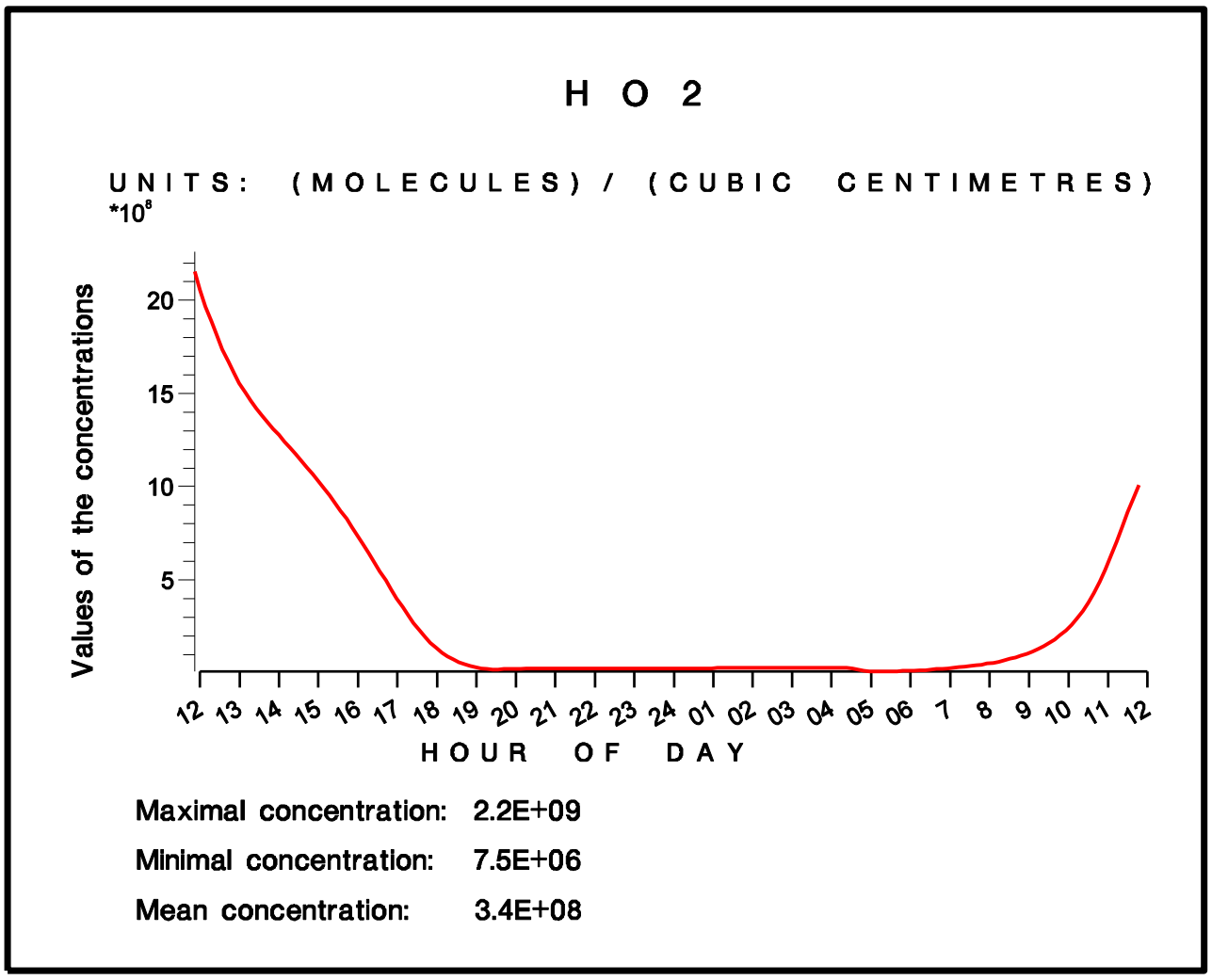

Figure 5.8
Diurnal variation during the time-interval of **24** hours.

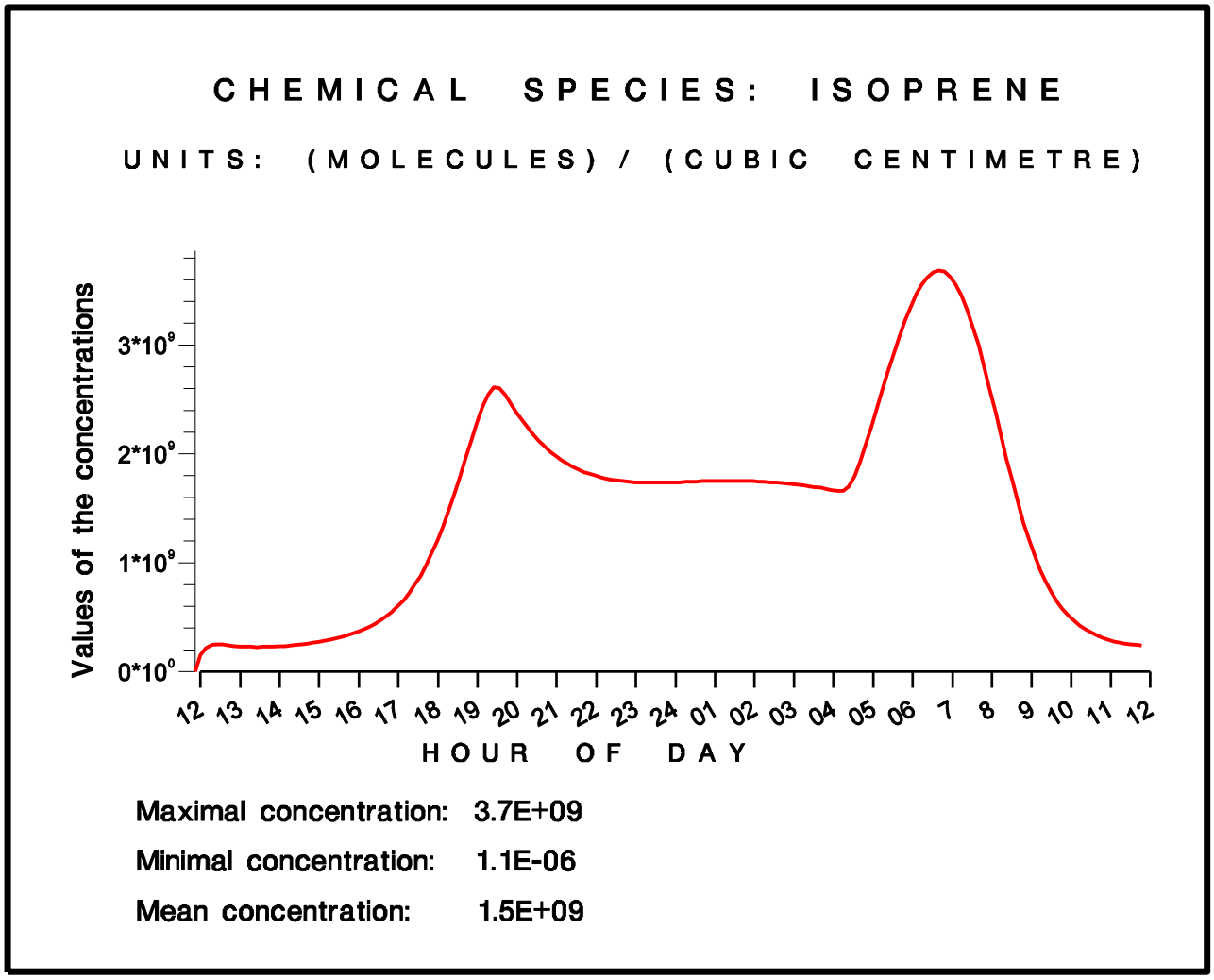

Figure 5.9
Diurnal variation during the time-interval of **24** hours.

The global error made at run $\mathbf{k}$, $\mathbf{k = 1}$, $\mathbf{2}, \ldots$, $\mathbf{19}$ is estimated by:

$$(\mathbf{5.98}) \quad \mathbf{ERROR}^{(\mathbf{k})} = \max_{\mathbf{j}=2^{\mathbf{k}-1}, 2\times 2^{\mathbf{k}-1}, \ldots, 168\times 2^{\mathbf{k}-1}} \left(\mathbf{ERROR}_{\mathbf{j}}^{(\mathbf{k})} \right) .$$

An attempt was made to eliminate the influence of the rounding errors when the quantities involved in (5.97) and (5.98) are calculated. This is indeed needed, because the Jacobian matrices involved in the treatment of the atmospheric chemical scheme are badly scaled and extremely ill-conditioned. Accurate results, without large rounding errors, can as a rule be obtained by using double precision arithmetic in the calculations, but that is not always sufficient when the chemical scheme is handled. The reason for this can be explained as follows. The atmospheric chemical scheme is a stiff non-linear system of ODEs. Therefore, implicit numerical methods must be used, which leads to the solution of systems of non-linear equations at each time-step by the Newton Iterative Method, see **Hairer and Wanner (1991)**, **Hamming (1962)**, **Hundsdorfer and Verwer (2003)**, **Stewart (1973)**, and, thus, to treatment

of long sequences of systems of linear algebraic equations during the iterative process. Let us reiterate here that it was found, by calculating eigenvalues and condition numbers with subroutines from **Anderson et al. (1992)** and **Barker et al. (2001)**, that the condition numbers of the matrices involved in the Newton Iterative Process on the time-interval $[\mathbf{a}, \mathbf{b}] = [\mathbf{43200}, \mathbf{129600}]$ vary in a very wide range $[\mathbf{4.56} \times \mathbf{10^8}, \mathbf{9.27} \times \mathbf{10^{12}}]$. If $\mathbf{REAL} * \mathbf{8}$ declarations for the real numbers are used in the computer programs, then computations involving about $\mathbf{16}$ significant digits are carried out. Simple application of error analysis arguments from **Stewart (1973)**, **Wilkinson (1963, 1965)** show clearly that there is a danger that the rounding errors could appear in the last twelve significant digits of the calculated numbers. Therefore, the computations presented in this section were carried out by selecting **quadruple precision** leading to the use of $\mathbf{REAL} * \mathbf{16}$ declarations for the real numbers and, thus, working with about $\mathbf{32}$ significant digits during all computations). We eliminated in this way the influence of **the rounding errors** at least in the first twenty significant digits of the approximate solutions. We did this in order to demonstrate the possibility of achieving very accurate results under the assumption that stable implementations of the Richardson Extrapolation were developed and used.

The results obtained by using the rules discussed above are given in **Table 5.5** for the two-stage second-order Diagonally Implicit Runge-Kutta (DIRK) methods used **(a)** directly, **(b)** together with the Marchuk-Strang splitting procedure and **(c)** together with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation. The corresponding results obtained when the Implicit Mid-Point Rule are given in **Table 5.6.**

Several major conclusions can be drawn by studying the numerical results presented in **Table 2** and **Table 3**:

(A) The results obtained by the direct implementation of a second-order method are as a rule more accurate than the corresponding results obtained when the Marchuk-Strang splitting procedure is additionally used. The hope is that the splitting procedure will lead to a better computational efficiency, because each of the two problems (5.59) and (5.60) is easier or even much easier than the original problem (5.54).

(B) The application of the two-stage second-order DIRK method together with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation leads to a third-order combined numerical method for sufficiently small stepsizes.

(C) The application of the second order Implicit Mid-point Rule together with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation is not stable for large time-stepsizes. This fact shows clearly that one should study carefully the stability properties of the combined method when the Richardson Extrapolation is to be used.

(D) The application of the second-order Implicit Mid-point Rule together with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation with small stepsizes leads to a combined numerical method which behaves as a fourth-order numerical scheme and gives very accurate results.

Number of jobs	Number of steps	Length of the stepsize	Direct DIRK method		Marchuk-Strang Splitting		Richardson Extrapolation	
			Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
1	168	514.28571	4.16E-01	-	4.15E-01	-	1.60E-01	-
2	336	257.14285	8.29E-02	5.02	2.01E-01	2.06	7.03E-02	2.28
3	672	128.57143	1.89E-02	4.39	9.52E-02	2.11	2.65E-02	2.66
4	1344	64.28571	4.55E-03	4.15	3.94E-02	2.42	8.61E-03	3.08
5	2688	32.14286	1.12E-03	4.07	1.47E-02	2.69	2.42E-03	3.56
6	5376	16.07143	2.77E-04	4.03	4.92E-03	2.95	5.83E-04	4.15
7	10752	8.03571	6.91E-05	4.02	1.55E-03	3.20	1.19E-04	4.89
8	21504	4.01786	1.72E-05	4.01	4.49E-04	3.46	1.99E-05	5.99
9	43008	2.00893	4.30E-06	4.00	1.22E-04	3.68	2.73E-06	7.29
10	86016	1.00446	1.07E-06	4.00	3.19E-05	3.82	3.30E-07	8.29
11	172032	0.50223	2.69E-07	4.00	8.17E-06	3.91	3.83E-08	8.62
12	344064	0.25112	6.72E-08	4.00	2.07E-06	3.95	4.56E-09	8.39
13	688128	0.12556	1.68E-08	4.00	5.19E-07	3.98	5.64E-10	8.09
14	1376256	0.06278	4.20E-09	4.00	1.30E-07	3.99	7.09E-11	7.96
15	2752512	0.03139	1.05E-09	4.00	3.26E-08	3.94	8.95E-12	7.93
16	5505024	0.01569	2.62E-10	4.00	8.16E-09	4.00	1.13E-12	7.94
17	11010048	0.00785	6.56E-11	4.00	2.04E-09	4.00	1.41E-13	7.97
18	22020096	0.00392	1.64E-11	4.00	5.10E-10	4.00	1.77E-14	7.98
19	44040192	0.00196	4.10E-12	4.00	1.27E-10	4.00	2.21E-15	7.99

Table 5.5

Running the atmospheric chemical scheme with **56** species by using the two-stage DIRK method (directly, together with the Marchuk-Strang splitting procedure and in combination with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation). Nineteen values of the stepsize were successively applied.

5.5.7. Concluding remarks

The implementation of some methods of Runge-Kutta type together with the Marchuk-Strang splitting procedure and the Richardson Extrapolation was studied in the previous sections. It was shown that the stability function of the combined numerical method can be expressed by a formula, which contains **only** the stability function of the underlying Runge-Kutta method. Moreover, the accuracy order of the resulting combined numerical methods is in general three, but may also be four in some cases. The stability properties of the combined numerical methods should be carefully investigated, because stability of the underlying methods cannot guarantee stability of the combined method when the Richardson Extrapolation is also used.

Number of jobs	Number of steps	Length of the stepsize	Implicit Mid-point Rule		Marchuk-Strang Splitting		Richardson Extrapolation	
			Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
1	168	514.28571	1.30E-00	-	3.37E-00	-	N.S.	N.A.
2	336	257.14285	1.58E-01	8.23	1.77E-01	19.04	N.S.	N.A.
3	672	128.57143	3.70E-02	4.27	6.54E-02	2.71	N.S.	N.A.
4	1344	64.28571	9.12E-03	4.06	1.39E-02	4.72	N.S.	N.A.
5	2688	32.14286	2.27E-03	4.01	3.72E-03	3.72	N.S.	N.A.
6	5376	16.07143	5.67E-04	4.00	9.47E-04	3.93	2.44E-04	-
7	10752	8.03571	1.42E-04	4.00	2.38E-04	3.98	4.13E-05	5.91
8	21504	4.01786	3.54E-05	4.00	5.95E-05	4.00	8.61E-06	4.80
9	43008	2.00893	8.86E-06	4.00	1.49E-05	4.00	1.53E-06	5.63
10	86016	1.00446	2.22E-06	4.00	3.72E-06	4.00	2.10E-07	7.27
11	172032	0.50223	5.54E-07	4.00	9.31E-07	4.00	2.23E-08	9.42
12	344064	0.25112	1.38E-07	4.00	2.33E-07	4.00	1.94E-09	11.53
13	688128	0.12556	3.46E-08	4.00	5.82E-08	4.00	1.46E-10	13.26
14	1376256	0.06278	8.65E-09	4.00	1.45E-08	4.00	1.01E-11	14.46
15	2752512	0.03139	2.16E-09	4.00	3.64E-09	4.00	6.65E-13	15.18
16	5505024	0.01569	5.41E-10	4.00	9.09E-10	4.00	4.27E-14	15.57
17	11010048	0.00785	1.35E-10	4.00	2.27E-10	4.00	2.71E-15	15.78
18	22020096	0.00392	3.38E-11	4.00	5.68E-11	4.00	2.60E-18	10.18
19	44040192	0.00196	8.45E-12	4.00	1.42E-11	4.00	3.32E-17	7.83

Table 5.6

Running the atmospheric chemical scheme with $\mathbf{56}$ species by using the Implicit Mid-point Rule (directly, together with the Marchuk-Strang splitting procedure and in combination with both the Marchuk-Strang splitting procedure and the Richardson Extrapolation). Nineteen values of the stepsize were successively applied. **"N.S."** and **"N.A"** mean "not stable" and "not applicable" respectively.

5.6. General conclusions related to Chapter 5

Only θ-methods were used in the first four sections of this chapter. This is justified, because the combination of any numerical method with the sequential procedure results in new numerical methods, the order of which is one. Therefore, it will not be very useful to apply numerical methods of higher order. Moreover, it was proved that for a larger sub-class of the class of the θ-methods, the computational process remains stable also when stiff systems of ODEs are treated by combinations of the sequential splitting procedure, the θ-methods for all values of θ in the interval $[\theta_0, 1.0]$ with $\theta_0 \approx 0.638$ and the Richardson Extrapolation; see **Theorem 5.1**.

Implicit methods of higher order may become relevant if splitting procedures of higher order are used in combination with the Richardson Extrapolation. This will certainly be true if the Marchuk-Strang splitting procedure is used. This procedure, which was introduced in **Marchuk (1968)**, see also **Marchuk (1980, 1982, 1986, 1988)**, and **Strang (1968)**, is of order two. Therefore, it will be optimal to use this procedure together with second-order numerical methods. Then the combination of the Marchuk-Strang splitting procedure with second-order numerical methods and the Richardson

Extrapolation will be a numerical method of third-order of accuracy. It should be noted however, that the Marchuk-Strang splitting procedure is more expensive computationally than the sequential splitting.

It has been shown in Theorem 5.1 that the combination of the sequential splitting procedure, the θ-methods for all values of θ in the interval $[\theta_0, 1.0]$ with $\theta_0 \approx 0.638$ and the Richardson Extrapolation, see **Theorem 5.1**, is a new numerical method, which is strongly A-stable. However, it should be emphasized also in this chapter that the requirement of strong A-stability is only a sufficient condition for achieving stable computations. Numerical methods, which do not possess this property should not automatically be discarded. Such methods could be very useful if one can show that their absolute stability regions are very large, as was done in the end of the previous chapter.

It was assumed that the right-hand-side of the system of ODEs is a sum of two operators only. The results can be generalized for the case where more operators are presented.

5.7. Topics for further research

The following topics might lead to some very interesting and useful results:

(A) It was mentioned in the previous section that the Marchuk-Strang splitting procedure is of order two and, therefore, it might be a good candidate for combined methods, also when the Richardson Extrapolation is used. Some stability results for the combinations (the Marchuk-Strang splitting procedure, some classes of second order implicit numerical methods and the Richardson Extrapolation) will be very useful.

(B) If the Marchuk-Strang splitting procedure is used, then the use of the numerical schemes from the class of the θ-methods will not be a good choice, because these numerical schemes are of first order of accuracy when $\theta \neq 0.5$, while the Marchuk-Strang splitting procedure is of second order of accuracy. Therefore, **the use of numerical schemes, the order of accuracy of which is two, will be optimal in this case**. It should be noted, however, that the Trapezoidal Rule, which belongs to the class of the θ-methods and can be found by setting $\theta = 0.5$, is not a good choice, because it will lead to unstable computations. Some second-order Implicit Runge-Kutta method should be selected.

(C) It is also possible to apply **first order** numerical scheme to each of the problems (5.3) and (5.4) obtained after applying splitting in relation to problem (5.1). Some numerical schemes from the class of the θ-methods can be selected (different numerical schemes can be used). The Richardson Extrapolation can be used for each of the two sub-problems. Thus, second order of accuracy will be obtained during the computations with each of the two sub-problems and it will be appropriate to apply the Marchuk-Strang splitting procedure.

(D) Some other splitting procedures can also be applied together with the Richardson Extrapolation in a similar manner as the sequential splitting was treated in this chapter.

Chapter 6

Richardson Extrapolation for advection problems

The application of the Richardson Extrapolation in connection with initial value problems for systems of ODEs was studied in the previous five chapters. Some applications of this approach in the case where partial differential equations (PDEs) are to be handled numerically on computers will be presented and discussed in this chapter. There are two ways to apply the Richardson Extrapolation in the treatment of PDEs:

(A) One may first semi-discretize the partial differential equation (or the system of partial differential equations). This is often performed by applying either finite elements of finite differences in relation to the spatial derivatives. Then the Richardson Extrapolation can be applied to the resulting (normally very large) system of ordinary differential equations.

(B) One can apply the Richardson Extrapolation directly to the partial differential equation (or to the system of partial differential equations).

The implementation of the Richardson Extrapolation in Case **(A)** is in principle the same as the implementation of this device, which was studied in the previous chapters. It is only necessary first to discretize the spatial derivatives: the result of this semi-discretization will be a system of ODEs of the same type as the systems of ODEs studied in Chapter 1 – Chapter 4. Therefore, it is not necessary to consider again this case here. Only two important facts must be stressed:

(a) the system of ODEs that is obtained after the semi-discretization of the system of PDEs is normally very large, which causes or at least may very often cause serious technical difficulties

and

(b) the accuracy of the results will be improved when the Richardson Extrapolation is implemented only if the errors due to the spatial discretization are considerably smaller than the error due to the use of the selected time-integration numerical method.

The short discussion in the previous paragraphs shows clearly that it is necessary to study here only Case **(B)**. This will be done in connection with some special partial differential equations, the advection equations, which are a very substantial part of large-scale air pollution models and in these applications describe mathematically the transport of air pollutants in the atmosphere (see **Alexandrov et al., 2004**, **Zlatev, 1995** or **Zlatev and Dimov, 2006**), but such equations appear in many other problems arising in science and engineering.

One-dimensional advection equations will be introduced in **Section 6.1**. The discretization of these equations by using the Crank-Nicolson numerical scheme will be briefly discussed there. The Crank-

Nicolson numerical scheme is of second order of accuracy in regard both to the time variable and to the space variable (see, for example, **Strikwerda, 2004**).

In **Section 6.2** it will be shown how the Richardson Extrapolation can be combined with a rather general numerical method for solving PDEs.

Four different implementations of the Richardson Extrapolation in connection with the Crank-Nicolson scheme will be introduced and discussed in **Section 6.3**.

In **Section 6.4** it will be proved that the new method (the combination of the Richardson Extrapolation with the Crank-Nicolson scheme) is of order four.

Three numerical examples will be given in **Section 6.5** in order to illustrate the fact that under certain assumptions the accuracy is indeed increased by two orders when the Richardson Extrapolation is used.

In the next section, in **Section 6.6**, the result proved in the fourth section for the one-dimensional advection will be generalized for the multi-dimensional case. Several special cases, one-dimensional advection, two-dimensional advection and three-dimensional advection will also be presented and discussed in Section 6.6.

Some conclusions and remarks will be presented in **Section 6.7**.

Some topics for further research in this area will be suggested in **Section 6.8**.

6.1. The one-dimensional advection problem

In the first five sections of this chapter, we shall mainly consider a very simple one-dimensional advection equation, which appears (very often after performing some kind of splitting) in many advanced mathematical models describing different processes arising in fluid dynamics; see, for example **Zlatev, 1995**, **Zlatev and Dimov, 2006**), but it should immediately be emphasized that this simplification is made only in order to facilitate the understanding of the main ideas. Most of the results can easily be extended for other and more complicated cases. One particular example for such an extension will be presented in **Section 6.6**.

We are interested in applying the Richardson Extrapolation for the following scalar partial differential equation:

$$(\mathbf{6.1}) \qquad \frac{\partial \mathbf{c}}{\partial \mathbf{t}} = -\mathbf{u}\,\frac{\partial \mathbf{c}}{\partial \mathbf{x}}\,, \qquad \mathbf{x} \in [\mathbf{a_1}, \mathbf{b_1}] \subset (-\infty, \infty)\,, \qquad \mathbf{t} \in [\mathbf{a}, \mathbf{b}] \subset (-\infty, \infty)\,.$$

It will be assumed that $\mathbf{u} = \mathbf{u}(\mathbf{x}, \mathbf{t})$ is a given function, which varies both in time and in space. The physical meaning of this function depends on the particular area in which equation (6.1) arises. For example, in many fluid dynamics applications it is interpreted as a velocity field. More specifically, in atmospheric modelling of long-range transport of air pollutants, $\mathbf{u}(\mathbf{x}, \mathbf{t})$ represents the wind velocity

field in the considered spatial domain (see, for example, **Alexandrov, 2004**, **Zlatev, 1995**, **Zlatev and Dimov, 2006**). It should be mentioned here that in this special case the unknown function $\mathbf{c}(\mathbf{x},\mathbf{t})$ is the concentration of some air pollutant, which could be dangerous for humans, animals and plants.

The well-known Crank-Nicolson numerical scheme (see, for example, **Strikwerda, 2004**, page 63) can be used in the treatment of (6.1) on computers. The calculations are carried out by applying the following formula:

$$(6.2)\quad \sigma_{i,n+0.5}\, c_{i+1,n+1} + c_{i,n+1} - \sigma_{i,n+0.5}\, c_{i-1,n+1} + \sigma_{i,n+0.5}\, c_{i+1,n} - c_{i,n} - \sigma_{i,n+0.5}\, c_{i-1,n} = 0\,,$$

when the Crank-Nicolson numerical scheme is used (more details about this particular numerical scheme as well as for other numerical schemes can also be found in **Crank and Nicolson, 1947**, **Lapidus and Pinder, 1982**, **Morton, 1996**, **Smith, 1978**).

Equation (6.1) must always be considered together with some appropriate initial and boundary conditions. In principle, it is only necessary to provide a boundary condition at the left-hand-side end-point of the spatial interval $[a_1, b_1]$ when $u(a_1, t)$ is positive. However, the use of the Crank-Nicolson numerical scheme (6.2) requires also a boundary condition on the right-hand-side end-point of this interval. Therefore, it will always be assumed that Dirichlet boundary conditions are available at the two end-points of the interval $[a_1, b_1]$.

Particular initial and boundary conditions will be discussed in **Section 6.4**, where some numerical examples will be introduced and handled.

The quantity $\sigma_{i,n+0.5}$ is defined by

$$(6.3)\quad \sigma_{i,n+0.5} = \frac{k}{4h}\, u(x_i, t_{n+0.5})\,,$$

where $t_{n+0.5} = t_n + 0.5k$ and the increments h and k of the spatial and time variables respectively are introduced by using two equidistant grids:

$$(6.4)\quad G_x = \left\{ x_i \mid x_0 = a_1,\ \ x_i = x_{i-1} + h,\ \ i = 1, 2,\ \ \ldots,\ \ N_x\,,\ \ h = \frac{b_1 - a_1}{N_x}\,,\ \ x_{N_x} = b_1 \right\}$$

and

$$(6.5)\quad G_t = \left\{ t_n \mid t_0 = a,\ \ t_n = t_{n-1} + k,\ \ n = 1, 2,\ \ \ldots,\ \ N_t\,,\ \ k = \frac{b - a}{N_t}\,,\ \ t_{N_t} = b \right\}.$$

It should be possible to vary the increments $\mathbf{h}$ and $\mathbf{k}$ (for example, the relationships $\mathbf{h} \to \mathbf{0}$ and $\mathbf{k} \to \mathbf{0}$ must be allowed when the convergence of the numerical method and the convergent rates of the calculated approximations are to be studied). However, it will be assumed that the ratio $\mathbf{h/k}$ remains always constant when the increments are varied. This implies a requirement that if, for example, $\mathbf{h}$ is halved, then $\mathbf{k}$ is also halved. More precisely, it will be assumed that if an arbitrary pair of increments $(\mathbf{h_1}, \mathbf{k_1})$ is replaced with another pair $(\mathbf{h_2}, \mathbf{k_2})$, then the following relationship must hold:

$$(\mathbf{6.6}) \qquad \frac{\mathbf{h_1}}{\mathbf{k_1}} = \frac{\mathbf{h_2}}{\mathbf{k_2}} = \boldsymbol{\gamma} .$$

where $\boldsymbol{\gamma}$ is a constant which does not depend on the increments. The requirement to keep $\mathbf{h/k}$ constant is not a serious restriction.

Assume that $\mathbf{c}(\mathbf{x_i}, \mathbf{t_n})$ is the exact solution of the advection equation (6.1) at an arbitrary grid-point $(\mathbf{x_i}, \mathbf{t_n})$ belonging to the two sets of points defined by the equidistant grids (6.4) and (6.5). Then the values $\mathbf{c_{i,n}}$ ($\mathbf{i = 0}$, $\mathbf{1}$, ..., $\mathbf{N_x}$ and $\mathbf{n = 1}$, $\mathbf{2}$, ..., $\mathbf{N_t}$) calculated by (6.2) will be approximations of the exact solution at the grid-points $(\mathbf{x_i}, \mathbf{t_n})$, i.e. the relationships $\mathbf{c_{i,n}} \approx \mathbf{c}(\mathbf{x_i}, \mathbf{t_n})$ will hold for all grid-points. Our major task in the following part of this chapter will be to show how the accuracy of the calculated approximations $\mathbf{c_{i,n}}$ can be improved essentially by using additionally the Richardson Extrapolation.

The application of the Richardson Extrapolation, when an arbitrary one-dimensional partial differential equation (not only the particular equation which was introduced in the beginning of this section) is treated by **any** numerical method, will be described in **Section 6.2**.

Four different implementations of the Richardson Extrapolation will be presented in **Section 6.3**.

The order of accuracy of **the new numerical method** consisting of the combination of the Crank-Nicolson scheme and the Richardson Extrapolation will be established in **Section 6.4**. The error constants in the leading terms of the numerical error for the Crank-Nicolson scheme will also be calculated there.

Numerical results will be presented in **Section 6.5** in order to demonstrate the applicability of the theorems proved in Section 6.4 and the fact that the application of the Richardson Extrapolation leads always to more accurate results. Several concluding remarks will also be given there.

6.2. Combining the advection problem with the Richardson Extrapolation

Assume that a one-dimensional time-dependent partial differential equation is treated by an arbitrary numerical method, which is of order $\mathbf{p \geq 1}$ with regard to the two independent variables $\mathbf{x}$ and $\mathbf{t}$.

Let $\{z_{i,n}\}_{i=0}^{N_x}$ be the set of approximations of the solution of (6.1) calculated for $t = t_n \in G_t$ at all grid-points $x_i \in G_x$, where $i = 0, 1, \ldots, N_x$. Assume furthermore that the set $\{z_{i,n}\}_{i=0}^{N_x}$ is calculated by using the numerical method chosen and consider the corresponding approximations $\{z_{i,n-1}\}_{i=0}^{N_x}$ calculated at the previous time-step, i.e. for $t = t_{n-1} \in G_t$. Introduce the following three vectors $\bar{c}(t_n)$, $\bar{z}_{i,n-1}$ and $\bar{z}_{i,n}$ the components of which are $\{c(x_i, t_n)\}_{i=0}^{N_x}$, $\{z_{i,n-1}\}_{i=0}^{N_x}$ and $\{z_{i,n}\}_{i=0}^{N_x}$ respectively. Since the order of the numerical method is assumed to be p with regard both to x and t, we can write:

$$(6.7) \quad \bar{c}(t_n) = \bar{z}_{i,n} + h^p K_1 + k^p K_2 + O(k^{p+1}),$$

where K_1 and K_2 are some quantities, which do not depend on the increments h and k. In fact, the last term in (6.7) will in general depend both on h^{p+1} and k^{p+1}. However, by using the assumption (6.6), we can write $h^{p+1} = \gamma^{p+1} k^{p+1}$ and since γ is a constant which does not depend on the increments h and k, it is clear that in our case, i.e. when the assumption (6.6) is made, the last term in (6.7) depends essentially only on k.

It is convenient to rewrite, by using once again (6.6), the last equality in the following form:

$$(6.8) \quad \bar{c}(t_n) = \bar{z}_n + k^p K + O(k^{p+1}).$$

where

$$(6.9) \quad K = \left(\frac{h}{k}\right)^p K_1 + K_2 .$$

If the increments h and k are sufficiently small, then the sum $h^p K_1 + k^p K_2$, which occurs in (6.7) will be a good approximation of the truncation errors in the calculated values of the numerical solution $\{z_{i,n}\}_{i=0}^{N_x}$. Of course, if the relationship (6.6) is satisfied and if again h and k are sufficiently small, then $k^p K$ from (6.8) will also be a good approximation of the error of the numerical solution $\bar{z}_{i,n}$. This means that if we succeed to eliminate the term $k^p K$ in (6.8), then we shall obtain approximations of higher order, of order at least equal to $p + 1$. The Richardson Extrapolation can be applied in the attempt to achieve such an improvement of the accuracy. In order to apply the Richardson Extrapolation it is necessary to introduce an additional grid:

$$(6.10) \quad G_x^2 = \left\{ x_i \mid x_0 = a_1, \quad x_i = x_{i-1} + \frac{h}{2}, \quad i = 1, 2, \ldots 2N_x, \quad h = \frac{b_1 - a_1}{N_x}, \quad x_{2N_x} = b_1 \right\}.$$

Assume that approximations $\{w_{i,n-1}\}_{i=0}^{2N_x}$ (calculated at the grid-points of G_x^2 for $t = t_{n-1} \in G_t$) are available and perform two small time-steps with a stepsize $k/2$ to compute the approximations $\{w_{i,n}\}_{i=0}^{2N_x}$. Use only the components with even indices i, $i = 0$, 2, ..., $2N_x$, to form the vector $\tilde{w}_n$. It is easily seen that the following equality holds for this vector:

$$(6.11) \quad \bar{c}(t_n) = \tilde{w}_n + \left(\frac{k}{2}\right)^p K + O(k^{p+1}) .$$

where the quantity K is defined as in (6.9).

Now, it is possible to eliminate the quantity K from (6.8) and (6.11). This can successfully be done in the following way:

(a) remove the last terms in (6.8) and (6.11),

(b) multiply (6.11) by 2^p

and

(c) subtract (6.8) from the modified equality (6.11).

The result is:

$$(6.12) \quad \bar{c}(t_n) = \frac{2^p \tilde{w}_n - \bar{z}_n}{2^p - 1} + O(k^{p+1}) .$$

Denote:

$$(6.13) \quad \bar{c}_n = \frac{2^p \tilde{w}_n - \bar{z}_n}{2^p - 1} .$$

It is clear that the approximation $\bar{c}_n$, being of order at least equal to $p + 1$, will in general be more accurate than both $\bar{z}_n$ and $\tilde{w}_n$ when the increments h and k are sufficiently small. The device used to construct the approximation $\bar{c}_n$ is obviously the Richardson Extrapolation, which is applied this time in order to improve the accuracy of a numerical method of order p for solving partial differential equations, not necessarily the particular equation (6.1). Indeed, if we assume that the partial derivatives of the unknown function $c(x, t)$ up to order $p + 1$ exist and are continuous, then it should be expected that the approximation calculated by using (6.13) will produce more accurate results than the two approximations $\bar{z}_n$ and $\tilde{w}_n$ that are obtained by applying directly the underlying numerical method.

Remark 6.1: No specific assumptions about the particular partial differential equation or about the numerical method used to solve it were made in this section excepting only the requirement that the order of the applied numerical method is $\mathbf{p}$ with regard to the two independent variables $\mathbf{x}$ and $\mathbf{t}$. This approach was used in order to demonstrate how general is the idea, on which the Richardson Extrapolation is based. It must be emphasized, however, that in the following part of this chapter we shall consider the case where

(a) equation (6.1) is solved under the assumptions made in the previous section

and

(b) the underlying numerical algorithm applied to handle it numerically is always the second-order Crank-Nicolson scheme.

■

6.3. Implementation of the Richardson Extrapolation

When the advection equation (6.1) is solved by using the Crank-Nicolson scheme, Richardson Extrapolation can be implemented in **four** different manners depending on the way in which the computations at the next time-step, step $\mathbf{n+1}$, will be carried out.

(1) Simplified Active Richardson Extrapolation: Use $\bar{\mathbf{c}}_{\mathbf{n}}$ as initial value to compute $\bar{\mathbf{z}}_{\mathbf{n+1}}$. Use the set of values $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n}}\right\}_{\mathbf{i=0}}^{\mathbf{2N_x}}$ as initial values to compute $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n+1}}\right\}_{\mathbf{i=1}}^{\mathbf{2N_{x-1}}}$ and after that to form $\widetilde{\mathbf{w}}_{\mathbf{n+1}}$. Since we assumed that Dirichlet boundary conditions are available on both end-points of the spatial interval, the values of $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n+1}}\right\}_{\mathbf{i=0}}^{\mathbf{2N_x}}$ for $\mathbf{i=0}$ and $\mathbf{i=2N_x}$ are also available.

(2) Passive Richardson Extrapolation: Use $\bar{\mathbf{z}}_{\mathbf{n}}$ as initial value to compute the approximation $\bar{\mathbf{z}}_{\mathbf{n+1}}$. Use, in the same way as in the Simplified Active Richardson Extrapolation, the set of values $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n}}\right\}_{\mathbf{i=0}}^{\mathbf{2N_x}}$ as initial values to compute $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n+1}}\right\}_{\mathbf{i=1}}^{\mathbf{2N_{x-1}}}$ and after that to form $\widetilde{\mathbf{w}}_{\mathbf{n+1}}$. The values of $\left\{\mathbf{w}_{\mathbf{i},\mathbf{n+1}}\right\}_{\mathbf{i=0}}^{\mathbf{2N_x}}$ for $\mathbf{i=0}$ and $\mathbf{i=2N_x}$ are again known, because it is assumed that Dirichlet boundary conditions are available on both end-points of the spatial interval.

(3) Active Richardson Extrapolation with linear interpolation on the finer spatial grid (9): Use $\bar{\mathbf{c}}_{\mathbf{n}}$ as initial values to compute $\bar{\mathbf{z}}_{\mathbf{n+1}}$. Set $\mathbf{w_{2i,n}} = \mathbf{c_{i,n}}$ for $\mathbf{i=0}, \mathbf{1}, \ldots, \mathbf{N_x}$. Use linear interpolation to obtain approximations of the values of $\mathbf{w_{2i,n}}$ for $\mathbf{i=1}, \mathbf{3}, \ldots, \mathbf{2N_x-1}$. Use the updated in this

way set of values $\{w_{i,n}\}_{i=0}^{2N_x}$ and the boundary conditions as initial values to compute $\{w_{i,n+1}\}_{i=0}^{2N_x}$ and to form $\tilde{w}_{n+1}$.

(4) Active Richardson Extrapolation with third-order interpolation on the finer spatial grid (6.9): Apply again $\bar{c}_n$ as initial values to compute $\bar{z}_{n+1}$. Set $w_{2i,n} = c_{i,n}$ for $i = 0, 1, \ldots, N_x$. Use third-order Lagrangian interpolation polynomials to obtain approximations of the values of $w_{2i,n}$ for $i = 0, 1, \ldots, 2N_x - 1$. Use the updated in this way set of values $\{w_{i,n}\}_{i=0}^{2N_x}$ and the boundary conditions as initial values to compute $\{w_{i,n+1}\}_{i=0}^{2N_x}$ and to form $\tilde{w}_{n+1}$.

The improvements obtained by applying (6.11) are not used in the further computations when the Passive Richardson Extrapolation is selected. These improvements are partly used in the calculations related to the large step (only to compute $\bar{z}_{n+1}$) when the Simplified Active Richardson Extrapolation is used. An attempt to produce and exploit more accurate values also in the calculation of $\tilde{w}_{n+1}$ is made in the last two implementations.

Information about the actual application of the third-order Lagrangian interpolation is given below. Assume that $w_{2i,n} = c_{i,n}$ for $i = 0, 1, \ldots, N_x$. This means that the improved (by the Richardson Extrapolation) solution on the coarser grid (6.44) is projected at the grid-points with even indices $0, 2, \ldots, 2N_x$ of the finer grid (6.9). The interpolation rule used to get better approximations at the grid-points of (6.9) which have odd indices can be described by the following formula:

$$(6.14) \quad w_{i,n} = -\frac{3}{48} w_{i-3,n} + \frac{9}{16} w_{i-1,n} + \frac{9}{16} w_{i+1,n} - \frac{3}{48} w_{i+3,n}, \quad i = 3, 5, \ldots, 2N_x - 3.$$

Formula (6.14) is obtained by using a third-order Lagrangian interpolation for the case where the grid-points are **equidistan**t and when an approximation at the mid-point x_i of the spatial interval $[x_{i-3}, x_{i+3}]$ is to be found. Note that only improved values are involved in the right-hand-side of (6.14).

Formula (6.14) cannot be used to improve the values at the points x_1 and x_{2N_x-1} of the finer grid (6.9). It is possible to use second-order interpolation at these two points:

$$(6.15) \quad w_{1,n} = -\frac{3}{8} w_{0,n} + \frac{3}{4} w_{2,n} - \frac{1}{8} w_{4,n},$$

$$w_{2N_x-1,n} = -\frac{3}{8} w_{2N_x,n} + \frac{3}{4} w_{2N_x-2,n} - \frac{1}{8} w_{2N_x-4,n}.$$

Some other formulae can be used instead of (6.14) and (6.15) in an attempt to achieve better accuracy at the grid-points $w_{i,n}$ of set (6.10) with odd indices.

6.4. Order of accuracy of the combined numerical method

Before starting the investigation of the accuracy of the combination of the Crank-Nicolson scheme with the Richardson Extrapolation, it is necessary to derive a formula showing the leading terms of the errors made when the Crank-Nicolson scheme is used directly.

Consider formula (6.2). Following **Lambert (1991)**, we shall replace the approximations contained in this formula with the corresponding exact values of the solution of (6.1). The result is:

$$
\begin{aligned}
(\mathbf{6.16}) \quad \mathbf{L}[\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}}; \mathbf{h}, \mathbf{k})] = \; & \boldsymbol{\sigma}_{\mathbf{i,n+0.5}}\, \mathbf{c}(\mathbf{x_{i+1}}, \mathbf{t_{n+1}}) + \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+1}}) - \mathbf{c}(\mathbf{x_{i-1}}, \mathbf{t_{n+1}}) \\
& + \boldsymbol{\sigma}_{\mathbf{i,n+0.5}}\, \mathbf{c}(\mathbf{x_{i+1}}, \mathbf{t_n}) - \mathbf{c}(\mathbf{x_i}, \mathbf{t_n}) - \mathbf{c}(\mathbf{x_{i-1}}, \mathbf{t_n}) \\
= \; & \boldsymbol{\sigma}_{\mathbf{i,n+0.5}}\, [\mathbf{c}(\mathbf{x_{i+1}}, \mathbf{t_{n+1}}) - \mathbf{c}(\mathbf{x_{i-1}}, \mathbf{t_{n+1}})] \\
& + [\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+1}}) - \mathbf{c}(\mathbf{x_i}, \mathbf{t_n})] + \boldsymbol{\sigma}_{\mathbf{i,n+0.5}}\, [\mathbf{c}(\mathbf{x_{i+1}}, \mathbf{t_n}) - \mathbf{c}(\mathbf{x_{i-1}}, \mathbf{t_n})] \, .
\end{aligned}
$$

The term $\mathbf{L}[\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}}; \mathbf{h}, \mathbf{k})]$ on the left hand side of (6.16) appears because the approximations participating in equality (6.2) are replaced by the corresponding exact values of the solution of (6.1).

The following theorem can be proved by using this notation:

Theorem 6.1: The quantity $\mathbf{L}[\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}}; \mathbf{h}, \mathbf{k})]$ can be written (assuming that all involved derivatives exist and are continuous) as:

$$
\begin{aligned}
(\mathbf{6.17}) \quad \mathbf{L}[\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}}; \mathbf{h}, \mathbf{k})] = \; & \frac{\boldsymbol{\gamma}\mathbf{h^3}}{\mathbf{6}}\, \mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.5}}) \frac{\boldsymbol{\partial^3}\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}})}{\boldsymbol{\partial x^3}} + \frac{\mathbf{k^3}}{\mathbf{24}} \frac{\boldsymbol{\partial^3}\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}})}{\boldsymbol{\partial t^3}} \\
& + \frac{\mathbf{k^3}}{\mathbf{8}}\, \mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.5}}) \frac{\boldsymbol{\partial^3}\mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}})}{\boldsymbol{\partial x \partial t^3}} + \mathbf{O}(\mathbf{k^5}) \, .
\end{aligned}
$$

Proof: Use Taylor expansions, of the functions in two variables involved in (6.16) around the point $(\mathbf{x_i}, \mathbf{t_{n+0.5}})$, where, as in Section 6.1, we have $\mathbf{t_{n+0.5}} = \mathbf{t_n} + \mathbf{0.5\, k}$ and keep the terms containing $\mathbf{k^r}$, where $\mathbf{r = 0, 1, 2, 3, 4.}$ After some rather long but quite straight-forward transformations (6.17) will be obtained.

■

Theorem 6.1 ensures that the Crank-Nicolson scheme is a second-order numerical method, which is, of course, well-known. It is much more important for our study that

(A) it provides **the leading terms** of the error of this method (which are needed in the proof of Theorem 6.2)

and

(B) it shows that there are **no fourth-order terms** in the expression for the numerical error.

After presenting the above preliminary results connected to

(a) the problem solved,

(b) the Crank-Nicolson scheme

and

(c) the Richardson Extrapolation,

everything is now prepared for the proof of a theorem showing that the use of the combination of the Crank-Nicolson scheme and the Richardson Extrapolation leads to **a fourth-order numerical method** when the problem (6.1) is solved. More precisely, the following theorem holds:

Theorem 6.2: The combination of the Crank-Nicolson scheme and the Richardson Extrapolation behaves as a fourth-order numerical method when (6.1) is solved and all derivatives of the unknown function **c(x,t)** up to order four exist and are continuous.

Proof: Assume that all approximations $\{c_{i,n}\}_{i=0}^{N_x}$ at time-step **n** have already been found. Then the Richardson Extrapolation is carried out by using the Crank-Nicolson scheme to perform one large time-step with stepsize **k** and two small time-steps with stepsize $0.5k$. The major part of the computations during **the large time-step** is carried out by using the formula:

$$(6.18)\quad \sigma_{i,n+0.5}\, z_{i+1,n+1} + z_{i,n+1} - \sigma_{i,n+0.5}\, z_{i-1,n+1} + \sigma_{i,n+0.5}\, c_{i+1,n} - c_{i,n} - \sigma_{i,n+0.5}\, c_{i-1,n} = 0 .$$

The major part of the computations during **the two small time-steps** is based on the use of the following two formulas:

$$(6.19)\quad \sigma_{i,n+0.25}\, w_{i+0.5,n+0.5} + w_{i,n+0.5} - \sigma_{i,n+0.25}\, w_{i-0.5,n+0.5}$$

$$+\sigma_{i,n+025}\, c_{i+0.5,n} - c_{i,n} - \sigma_{i,n+0.25}\, c_{i-0.5,n} = 0$$

and

$$(6.20)\quad \sigma_{i,n+0.75}\, w_{i+0.5,n+1} + w_{i,n+1} - \sigma_{i,n+0.75}\, w_{i-0.5,n+1}$$
$$+\sigma_{i,n+075}\, w_{i+0.5,n+0.5} - w_{i,n+0.5} - \sigma_{i,n+0.75}\, w_{i-0.5,n+0.5} = 0\,.$$

Let us start with (6.18). Equality (6.17) will obviously be obtained when all approximate values in (6.18) are replaced with the corresponding values of the exact solution. This means that the assertion of Theorem 6.1, equality (6.17), holds for the large time-step.

The treatment of the two small time-steps is more complicated. Combining (6.19) and (6.20) leads to the formula:

$$(6.21)\quad \sigma_{i,n+0.75}\, w_{i+0.5,n+1} + w_{i,n+1} - \sigma_{i,n+0.75}\, w_{i-0.5,n+1}$$
$$+\sigma_{i,n+075}\, w_{i+0.5,n+0.5} - w_{i,n+0.5} - \sigma_{i,n+0.75}\, w_{i-0.5,n+0.5}$$
$$+\sigma_{i,n+0.25}\, w_{i+0.5,n+0.5} + w_{i,n+0.5} - \sigma_{i,n+0.25}\, w_{i-0.5,n+0.5}$$
$$+\sigma_{i,n+025}\, c_{i+0.5,n} - c_{i,n} - \sigma_{i,n+0.25}\, c_{i-0.5,n} = 0$$

Replace all approximate values participating in (6.21) with the corresponding exact values of the solution of (6.1) to obtain an expression for the local approximation error $\hat{L}$ in the form:

$$(6.22)\quad \hat{L} = \hat{L}_1 + \hat{L}_2\,,$$

where

$$(6.23)\quad \hat{L}_1 = \sigma_{i,n+075}\,[c(x_{i+0.5}, t_{n+1}) - c(x_{i-0.5}, t_{n+1}) + c(x_{i+0.5}, t_{n+0.5}) - c(x_{i-0.5}, t_{n+0.5})]$$
$$+ c(x_i, t_{n+1}) - c(x_i, t_{n+0.5})$$

and

$$(6.24)\quad \hat{L}_2 = \sigma_{i,n+025}\,[c(x_{i+0.5}, t_{n+0.5}) - c(x_{i-0.5}, t_{n+0.5}) + c(x_{i+0.5}, t_n) - c(x_{i-0.5}, t_n)]$$
$$+ c(x_i, t_{n+0.5}) - c(x_i, t_n)\,.$$

Our aim is to derive an expression for $\mathbf{L}$.

First, we consider the terms participating in $\hat{\mathbf{L}}_1$. We use Taylor expansions of the involved functions around the point $(\mathbf{x_i}, \mathbf{t_{n+0.75}})$ and apply a similar transformation as in the proof of Theorem 6.1. In this way we can obtain the relation:

$$(\mathbf{6.25}) \quad \hat{\mathbf{L}}_1 = \frac{\boldsymbol{\gamma}\mathbf{h}^3}{\mathbf{48}}\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.75}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{x}^3} + \frac{\mathbf{k}^3}{\mathbf{192}}\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{t}^3} + \frac{\mathbf{k}^3}{\mathbf{64}}\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.75}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{x}\partial \mathbf{t}^3} + \mathbf{O}(\mathbf{k}^5).$$

We repeat the same kind of transformations also when $\hat{\mathbf{L}}_2$ is considered. Now we apply Taylor expansions around the point $(\mathbf{x_i}, \mathbf{t_{n+0.25}})$ of the involved functions. Then we obtain:

$$(\mathbf{6.26}) \quad \hat{\mathbf{L}}_2 = \frac{\boldsymbol{\gamma}\mathbf{h}^3}{\mathbf{48}}\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.25}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{x}^3} + \frac{\mathbf{k}^3}{\mathbf{192}}\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{t}^3} + \frac{\mathbf{k}^3}{\mathbf{64}}\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.25}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{x}\partial \mathbf{t}^3} + \mathbf{O}(\mathbf{k}^5).$$

The following result can be found by combining (6.25) and (6.26):

$$\begin{aligned}(\mathbf{6.27}) \quad \hat{\mathbf{L}} = {} & \frac{\boldsymbol{\gamma}\mathbf{h}^3}{\mathbf{48}}\left[\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.75}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{x}^3} + \mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.25}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{x}^3}\right] \\ & + \frac{\mathbf{k}^3}{\mathbf{192}}\left[\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{t}^3} + \frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{t}^3}\right] \\ & + \frac{\mathbf{k}^3}{\mathbf{64}}\left[\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.75}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.75}})}{\partial \mathbf{x}\partial \mathbf{t}^3} + \mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.25}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.25}})}{\partial \mathbf{x}\partial \mathbf{t}^3}\right] + \mathbf{O}(\mathbf{k}^5).\end{aligned}$$

Now, by expanding all terms in the right-hand-side of (6.27) around the point $(\mathbf{x_i}, \mathbf{t_{n+0.5}})$ and after some very long but straight-forward transformations, we obtain

$$(\mathbf{6.28}) \quad \hat{\mathbf{L}} = \frac{\boldsymbol{\gamma}\mathbf{h}^3}{\mathbf{24}}\mathbf{u}(\mathbf{x_i}, \mathbf{t_{n+0.5}})\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}})}{\partial \mathbf{x}^3} + \frac{\mathbf{k}^3}{\mathbf{96}}\frac{\partial^3 \mathbf{c}(\mathbf{x_i}, \mathbf{t_{n+0.5}})}{\partial \mathbf{t}^3}$$

$$+\frac{\mathbf{k^3}}{\mathbf{32}}\,\mathbf{u(x_i,t_{n+0.5})}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial x \partial t^3}}+\mathbf{O(k^5)}\,.$$

It should be noted here that a detailed derivation of the important relationship (6.28) can be found in **Zlatev et al. (2011b)**.

Since the order of the Crank-Nicolson scheme is $\mathbf{p=2}$, it is clear that the improved by the Richardson Extrapolation approximate solution at time-step $\mathbf{n+1}$ is obtained by

(a) multiplying the result obtained in (6.28), i.e. at the end of the second small time-step, by $\mathbf{4/3}$,

(b) multiplying the result obtained in (6.17), in fact the result found at the end of the large time-step, by $\mathbf{1/3}$

and

(c) subtracting the two results obtained in (a) and (b).

Performing operations **(a)** –**(c)** will give:

$$\begin{aligned}(\mathbf{6.29})\quad \frac{\mathbf{4}}{\mathbf{3}}\mathbf{\hat{L}}-\frac{\mathbf{1}}{\mathbf{3}}\mathbf{L[c(x_i,t_{n+0.5};h,k)]}&=\frac{\mathbf{\gamma h^3}}{\mathbf{18}}\mathbf{u(x_i,t_{n+0.5})}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial x^3}}+\frac{\mathbf{k^3}}{\mathbf{72}}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial t^3}}\\&+\frac{\mathbf{k^3}}{\mathbf{24}}\,\mathbf{u(x_i,t_{n+0.5})}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial x \partial t^3}}\\&-\frac{\mathbf{\gamma h^3}}{\mathbf{18}}\mathbf{u(x_i,t_{n+0.5})}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial x^3}}-\frac{\mathbf{k^3}}{\mathbf{72}}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial t^3}}\\&-\frac{\mathbf{k^3}}{\mathbf{24}}\,\mathbf{u(x_i,t_{n+0.5})}\frac{\mathbf{\partial^3 c(x_i,t_{n+0.5})}}{\mathbf{\partial x \partial t^3}}+\mathbf{O(k^5)}\,.\end{aligned}$$

It is immediately seen that the first six terms in the right-hand-side of (6.27) vanish. Therefore, the order of accuracy of the combined numerical method (the Crank-Nicolson scheme + the Richardson Extrapolation) is four, which completes the proof of the theorem.

■

It should once again be emphasized that a full proof of Theorem 6.2, containing all needed details, can be found in **Zlatev et al. (2011c)**, see also **Zlatev at al. (2011b)**.

6.5. Three numerical examples

In this section it will be verified numerically that the following two statements are true:

(a) if the solution is continuously differentiable up to order two, then the direct application of the Crank-Nicolson scheme gives second-order accuracy

and

(b) if the solution is continuously differentiable up to order four, then the combined method consisting of the Crank-Nicolson scheme and the Richardson Extrapolation behaves normally as a fourth-order numerical algorithm.

Furthermore, we shall also demonstrate the fact that if the above requirements are not satisfied, then neither the direct use of the Crank-Nicolson scheme leads to second-order accuracy, nor the new numerical method based on the combination of the Crank-Nicolson scheme with the Richardson Extrapolation behaves as a fourth-order numerical algorithm.

6.5.1. Organization of the computations

It is convenient for the purposes in this chapter, but not necessary, to divide the time-interval $[\mathbf{a}, \mathbf{b}]$ into $\mathbf{24}$ equal sub-intervals and to call each of these sub-intervals "hour". By this convention, the length of the time-interval becomes $\mathbf{24}$ hours in all three examples given in this section and we shall study the size of the numerical errors at the end of every hour. It should also be added that this convention is very useful in the air pollution model **UNI-DEM** where the advection scheme is considered together with the atmospheric chemical scheme considered in the previous chapter and the calculations have to be synchronized (more details can be found in **Zlatev, 1995** or in **Zlatev and Dimov, 2006**).

In each experiment **the first run** is performed by using $\mathbf{N_t = 168}$ and $\mathbf{N_x = 160}$. Ten additional runs are performed after the first one. When a run is finished, both $\mathbf{h}$ and $\mathbf{k}$ are halved (this means that both $\mathbf{N_t}$ and $\mathbf{N_x}$ are doubled) and a new run is started. Thus, in the last run, in the eleventh run, we have $\mathbf{N_t = 172032}$ and $\mathbf{N_x = 163840}$, which means that $\mathbf{172032}$ systems of linear algebraic equations, each of them containing $\mathbf{163840}$ equations, are to be solved. This short description is giving some ideas about the size of the problems that have to be handled in numerical examples, which were selected by us. It is also becoming quite clear that the problems related to the treatment of partial differential equations are in general much bigger than the problems related to the solution of systems of ODEs, which were treated in the previous chapters.

It is worthwhile to reiterate here that the ratio $\mathbf{h/k}$ is kept constant, i.e. the requirement, which was introduced by (6.6) in Section 6.1, is satisfied, when the two increments $\mathbf{k}$ and $\mathbf{h}$ are varied in the manner described above.

We are mainly interested in the behavior of the numerical errors. As mentioned above, these errors must be evaluated at the end of every hour (i.e. $\mathbf{24}$ times in each run). Moreover, the errors are evaluated at the grid-points of the coarsest spatial grid.

The evaluation of the errors, which is based on these two assumptions, is described below.

Assume that run number $\mathbf{r}$, where $\mathbf{r = 1}$, $\mathbf{2}$, ... , $\mathbf{11}$, is to be carried out and let $\mathbf{R = 2^{r-1}}$. Then the error made at the end of hour $\mathbf{m}$ is calculated by using the following formula:

$$(\mathbf{6.30}) \quad \mathbf{ERROR_m} = \max_{\mathbf{j=0,1,\dots,160}} \left(\frac{\left| \mathbf{c_{\tilde{\imath},\tilde{n}}} - \mathbf{c^{exact}_{\tilde{\imath},\tilde{n}}} \right|}{\max\left(\left| \mathbf{c^{exact}_{\tilde{\imath},\tilde{n}}} \right|, \ \mathbf{1.0} \right)} \right),$$

$$\mathbf{m = 1, 2, \dots, 24}, \quad \mathbf{\tilde{\imath} = jR}, \quad \mathbf{\tilde{n} = 7mR},$$

where $\mathbf{c_{\tilde{\imath},\tilde{n}}}$ and $\mathbf{c^{exact}_{\tilde{\imath},\tilde{n}}}$ are the calculated approximation and the exact solution of the solved problem at the end of hour $\mathbf{m}$ and at the grid-points of the coarsest spatial grid (i.e., the spatial grid with $\mathbf{N_x = 160}$). It should be mentioned here that in the three experiments, which will be presented in this section, the exact solution **is known**.

The global error made during the computations is estimated by using the following formula:

$$(\mathbf{6.31}) \quad \mathbf{ERROR} = \max_{\mathbf{m=0,1,\dots,24}} (\mathbf{ERROR_m}).$$

It is necessary to point out that the numerical values of the unknown function, which are improved by the Richardson Extrapolation, i.e. by applying (6.13) with $\mathbf{p = 2}$, are available only on the coarser spatial grid (6.4). It is necessary to get appropriate approximations for all values on the finer spatial grid (6.10). Several devices for obtaining such approximations that have been suggested and tested in **Zlatev et al. (2011a)** were described in Section 6.3.

It was emphasized in **Zlatev et al. (2011a)** that the application of third-order interpolation polynomials gives best results. This device has been used also in the next three paragraphs, but it will also be compared with the other options in §6.5.5.

6.5.2. Construction of a test-problem with steep gradients of the unknown function

Assume that the spatial and the time intervals are given by

$$(\mathbf{6.32}) \quad \mathbf{x} \in [\mathbf{0}, \mathbf{50000000}], \quad \mathbf{t} \in [\mathbf{43200}, \mathbf{129600}].$$

and consider a function $\mathbf{u(x,t)}$ defined by

$$(6.33) \quad u(x,t) = 320 .$$

Let the initial condition be given by

$$(6.34) \quad f(x) = \xi \left[1 + 99.0\, e^{-\omega(x-10000000)^2}\right], \quad \xi = 1.4679 \times 10^{12}, \quad \omega = 10^{-12} .$$

The exact solution of the test-problem, which is defined as above, is:

$$(6.35) \quad c(x,t) = f(x - 320(t - 43200)) .$$

It is not very important in the treatment of the numerical example defined by (6.1), (6.32), (6.33) and (6.34), but it should nevertheless be pointed out that both this example and the next two examples were used to test the reliability of the results obtained by using several modules of the Unified Danish Eulerian Model (UNI-DEM), see **Alexandrov et al. (2004)**, **Zlatev (1995)** and **Zlatev and Dimov (2006)**. This is why the same units, as those used in UNI-DEM, are also used here and, more precisely, the distances are measured in centimetres, while the time is measured in seconds.

The test-problem introduced in this sub-section was run both by using the Crank-Nicolson scheme directly and by applying the combination of this scheme and the Richardson Extrapolation (actually, as mentioned above, the fourth implementation of the Richardson Extrapolation, the Active Richardson Extrapolation with third-order interpolation on the finer spatial grid was used in this sub-section).

Numerical results are presented in **Table 6.1**. The following conclusions can be drawn by studying the results presented in **Table 6.1**:

- The direct application of the Crank-Nicolson scheme leads to quadratic convergence of the accuracy of the numerical results (i.e. halving the increments k and h leads to a decrease of the error by a factor of four). This behaviour should be expected according to Theorem 6.1.

- The combination of the Crank-Nicolson scheme and the Richardson Extrapolation behaves in general as a numerical method of order four (or, in other words, halving the increments k and h leads to a decrease of the error by a factor of sixteen). This behaviour should also be expected (according to Theorem 6.2).

- At the end of the computations with the combined numerical method (the Crank-Nicolson scheme + the Richardson Extrapolation) the convergence rate deteriorates. Two facts are very important when this happens:

(A) the computed solution is already very accurate

and

(B) the rounding errors start to affect the calculated results.

The use of quadruple precision (as in the previous chapters) will eliminate the effect of the rounding errors. However, this action will be too expensive in this case.

No.	NT	NX	Direct Solution		Richardson Extrapolation	
			Error	Ratio	Error	Ratio
1	168	160	7.373E-01	-	1.454E-01	-
2	336	320	4.003E-01	1.842	1.741E-02	8.350
3	672	640	1.254E-01	3.142	1.224E-03	14.220
4	1344	1280	3.080E-02	4.135	7.730E-05	15.837
5	2688	2560	7.765E-03	3.967	4.841E-06	15.970
6	5376	5120	1.954E-03	3.974	3.026E-07	15.999
7	10752	10240	4.892E-04	3.994	1.891E-08	16.004
8	21504	20480	1.224E-04	3.999	1.181E-09	16.011
9	43008	40960	3.059E-05	4.000	7.609E-11	15.519
10	86016	81920	7.648E-06	4.000	9.848E-12	7.726
11	172032	163840	1.912E-06	4.000	4.966E-12	1.983

Table 6.1

Results obtained when the test-problem defined by (6.32)-(6.34) is handled directly by the Crank-Nicolson scheme and by using the combination of the Crank-Nicolson scheme and the fourth implementation of the Richardson Extrapolation. The numerical errors calculated by (6.30) and (6.31) are given in the columns under "Error". In row $\mathbf{i}$, where $\mathbf{i = 2, 3, \ldots, 11}$, the ratios of the errors in this row and in the previous row are given in the columns under "Ratio".

Three plots are presented in **Fig. 6.1 – Fig. 6.3**. These plots show:

(a) the initial values,

(b) the solution in the middle of the time interval (i.e. after 12 hours)

and

(c) the solution at the end of the time interval

for the test-problem defined by (6.1) and (6.32) - (6.34).

Remark 6.2: A similar test-example was used in **Zlatev, Berkowicz and Prahm (1983)**. It should also be noted that a very similar advection module is a part of the large-scale air pollution model **UNI-DEM** (**Alexandrov et al. 2004**, **Zlatev, 1995**, and **Zlatev and Dimov, 2006**) and the quantities used in (6.32) - (6.34) are either the same or very similar to the corresponding quantities in this model. Note too that the values of the unknown function are of the same order of magnitude as the ozone concentrations in the atmosphere when these are measured in (number of molecules) / (cubic centimetre).

■

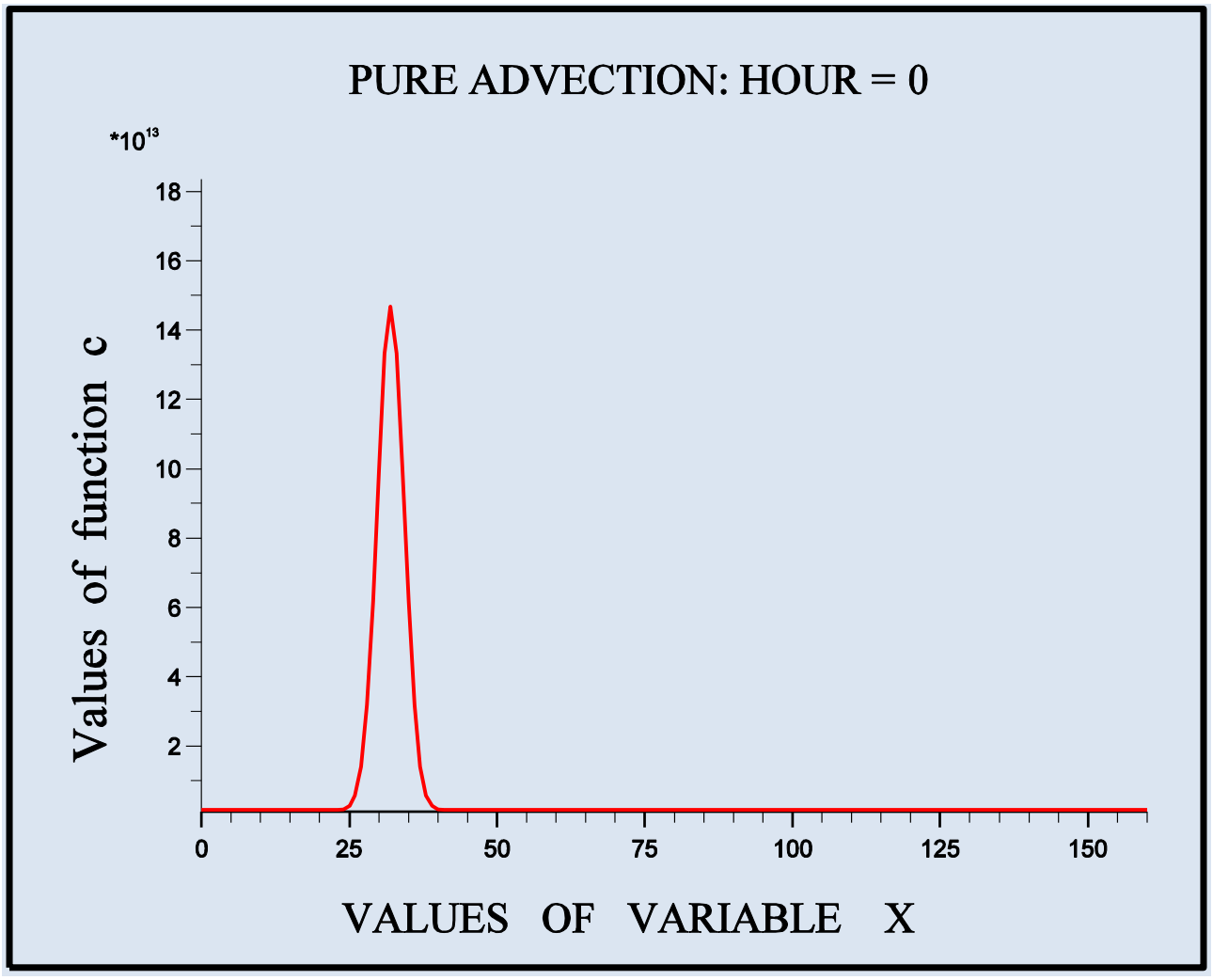

Figure 6.1

The initial value of the solution of the one-dimensional advection equation, which was defined in §6.5.2 and the solution of which has steep gradients.

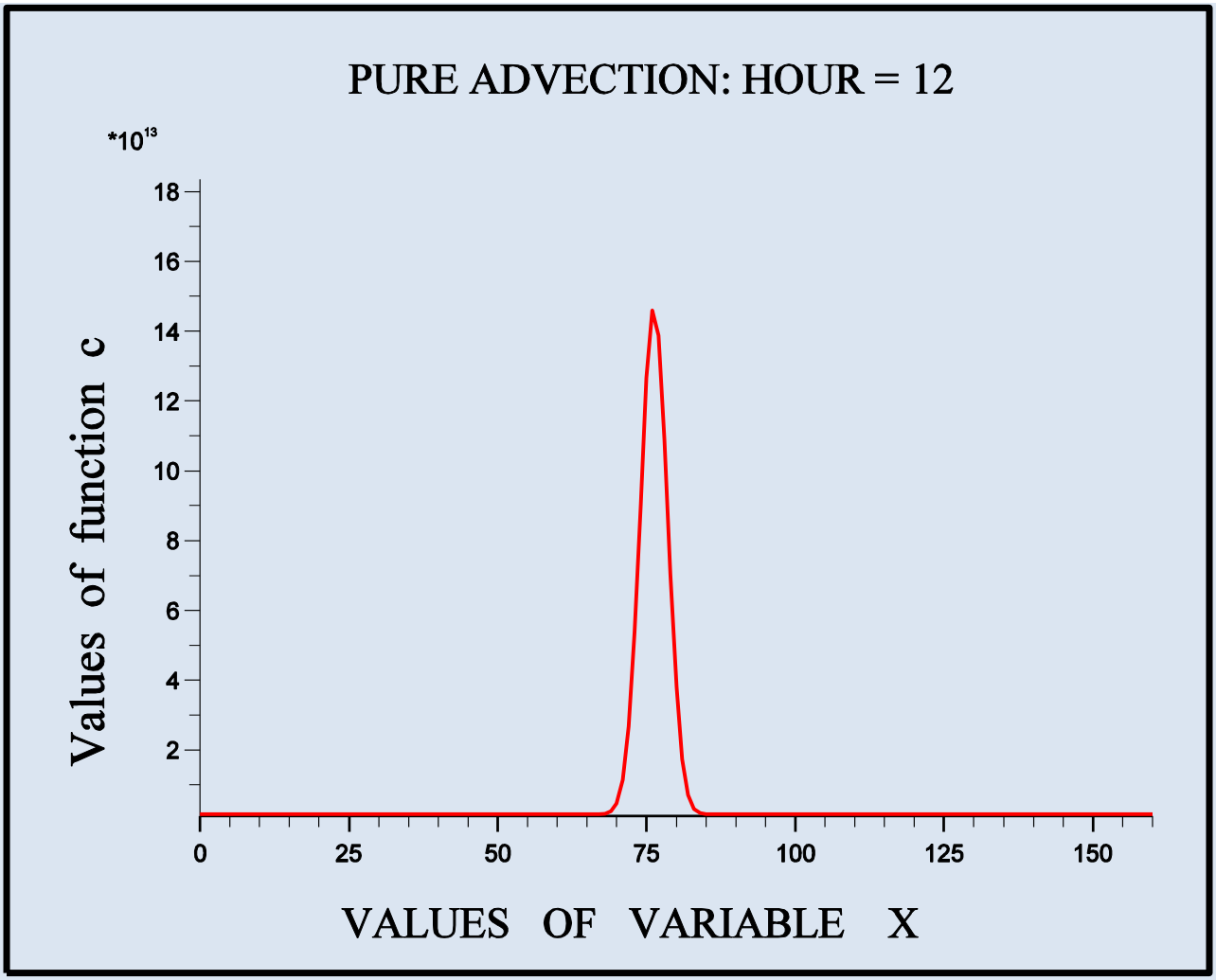

Figure 6.2

The calculated solution at the end of the twelfth hour of the one-dimensional advection equation, which was defined in §6.5.2 and the solution of which has steep gradients.

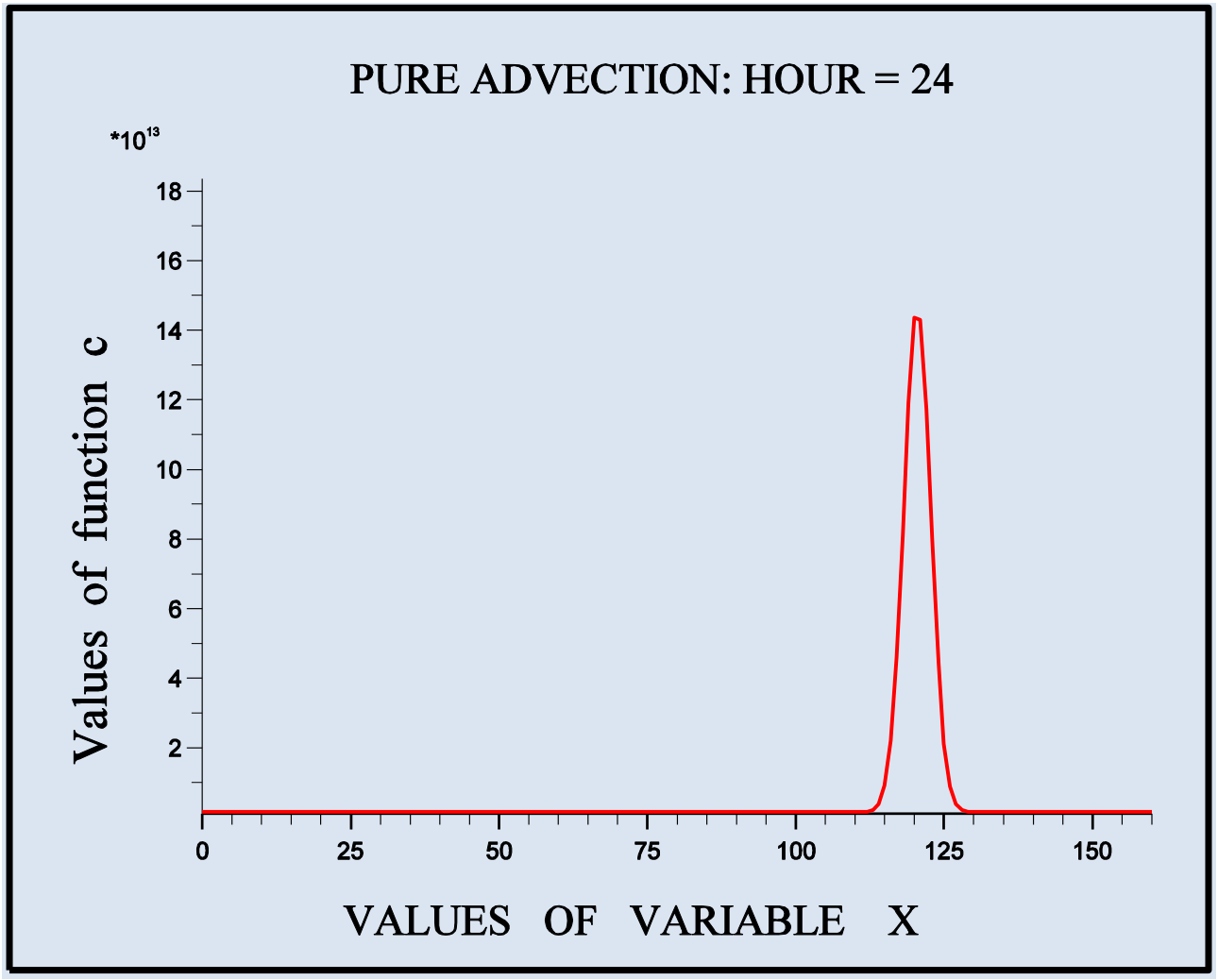

Figure 6.3

The calculated solution at the end of the twenty fourth hour of the one-dimensional advection equation, which was defined in §6.5.2 and the solution of which has steep gradients.

6.5.3. Construction of an oscillatory test-problem

Define the spatial and time intervals of the advection problem (6.1) by

$$(6.36) \quad a = a_1 = 0\,, \qquad b = b_1 = 2\pi$$

and consider a function $u(x,t)$ defined by

$$(6.37) \quad u(x,t) = 0.5\,.$$

Let the initial values be defined by

$$(6.38) \quad f(x) = \xi\,[100 + 99\,\sin(10x)]\,, \quad \xi = 1.4679 \times 10^{12}\,.$$

The exact solution of the test-problem, which is defined as above, is:

$$(6.39) \quad c(x,t) = f(x - 0.5t)\,.$$

As in §6.5.2, the test-problem introduced above was run both by using the Crank-Nicolson scheme directly and by applying the combination of this scheme and the fourth implementation of the Richardson Extrapolation.

Numerical results are presented in Table 6.2.

			Direct Solution		Richardson Extrapolation	
No.	NT	NX	Error	Ratio	Error	Ratio
1	168	160	7.851E-01	-	1.560E-02	-
2	336	320	2.160E-01	3.635	1.227E-03	12.713
3	672	640	5.317E-02	4.062	1.072E-04	11.432
4	1344	1280	1.327E-02	4.007	1.150E-05	9.333
5	2688	2560	3.319E-03	3.997	1.193E-06	9.641
6	5376	5120	8.299E-04	4.000	1.478E-07	8.071
7	10752	10240	2.075E-04	4.000	1.618E-08	9.136
8	21504	20480	5.187E-05	4.000	1.965E-09	8.233
9	43008	40960	1.297E-05	4.000	2.387E-10	8.233
10	86016	81920	3.242E-06	4.000	3.241E-11	7.365
11	172032	163840	8.104E-07	4.000	1.267E-11	2.557

Table 6.2

Results obtained when the oscillatory test-problem defined by (6.36) - (6.38) is handled directly by the Crank-Nicolson scheme and by using the combination of the Crank-Nicolson scheme and the fourth implementation of the Richardson Extrapolation. The numerical errors calculated by (6.30) and (6.31) are given in the columns under "Error". In row $\mathbf{i}$, where $\mathbf{i = 2}$, $\mathbf{3}$, ... , $\mathbf{11}$, the ratios of the errors in this row and in the previous row are given in the columns under "Ratio".

The conclusions, which can be drawn from the results presented in Table 6.2, are quite similar to those given in §6.5.2. However, for the oscillatory test-problem the actual convergence rate achieved in the eleven runs is less than four (greater than three in the beginning and after that equal to or less than three). It is not very clear what the reason for this behaviour is. Perhaps, the second-order interpolation rule, see (6.15) in Section 6.3 and **Zlatev et al. (2011a)**, which is used to improve the precision of the values of the solution at grid-points of the finer spatial grid that are close to the boundaries is not

sufficiently accurate for this example, because the solution varies very quickly also there. Nevertheless, it is clearly seen that the achieved accuracy is nearly the same as the accuracy achieved in the solution of the previous test-problem (compare Table 6.1 with Table 6.2).

Plots, which show

(a) the initial values,

(b) the solution in the middle of the time interval (i.e. after 12 hours)

and

(c) the solution at the end of the time interval

for the oscillatory test-problem, are given in **Fig. 6.4 – Fig. 6.6**, respectively.

6.5.4. Construction of a test-problem with discontinuous derivatives of the unknown function

Assume that the spatial interval, the time-interval and function $\mathbf{u}(\mathbf{x},\mathbf{t})$ are defined as in §6.5.2,, i.e. by (6.32) and (6.33), and introduce initial values by using the following formulae:

$$(\mathbf{6.40}) \quad \mathbf{f}(\mathbf{x}) = \boldsymbol{\xi}\ , \quad \boldsymbol{\xi} = \mathbf{1.4679} \times \mathbf{10^{12}}\ , \quad \text{when} \quad \mathbf{x} \leq \mathbf{5000000} \quad \text{or} \quad \mathbf{x} \geq \mathbf{15000000}\ ,$$

$$(\mathbf{6.41}) \quad \mathbf{f}(\mathbf{x}) = \boldsymbol{\xi}\left[\mathbf{1} + \mathbf{99.0} \times \frac{\mathbf{x} - \mathbf{5000000}}{\mathbf{5000000}}\right], \quad \text{when} \quad \mathbf{5000000} < \mathbf{x} < \mathbf{10000000}\ ,$$

$$(\mathbf{6.42}) \quad \mathbf{f}(\mathbf{x}) = \boldsymbol{\xi}\left[\mathbf{1} + \mathbf{99.0} \times \frac{\mathbf{15000000} - \mathbf{x}}{\mathbf{5000000}}\right], \quad \text{when} \quad \mathbf{10000000} < \mathbf{x} < \mathbf{15000000}\ ,$$

The exact solution of the test-problem, which is defined as above, is given by (6.35).

As in the previous two sub-sections, the test-problem introduced above was run both by using the Crank-Nicolson scheme directly and by applying the combination of this scheme and the Richardson Extrapolation.

Numerical results are presented in Table 6.3.

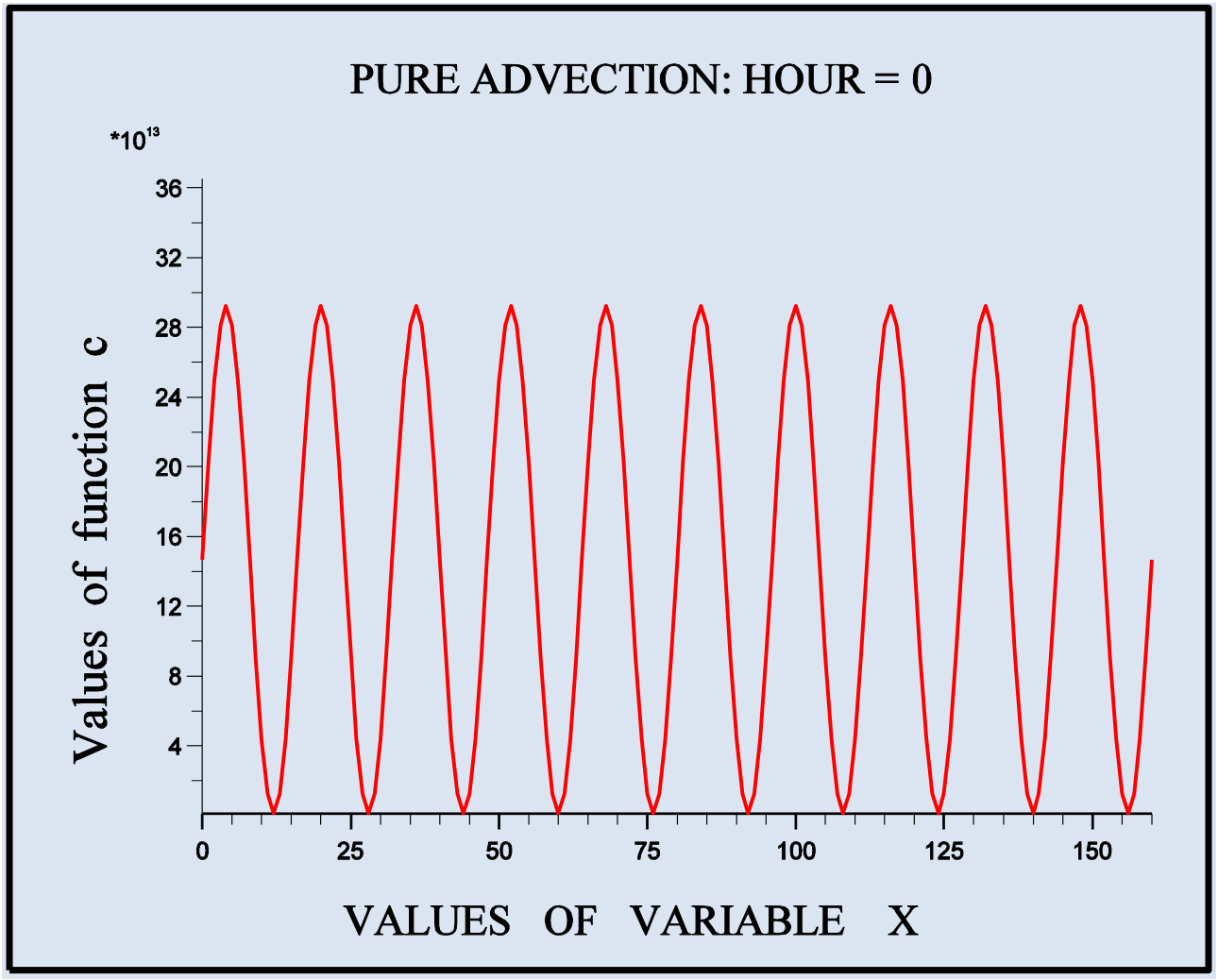

Figure 6.4

The initial value of the solution of the oscillatory one-dimensional advection equation, which was defined in §6.5.3.

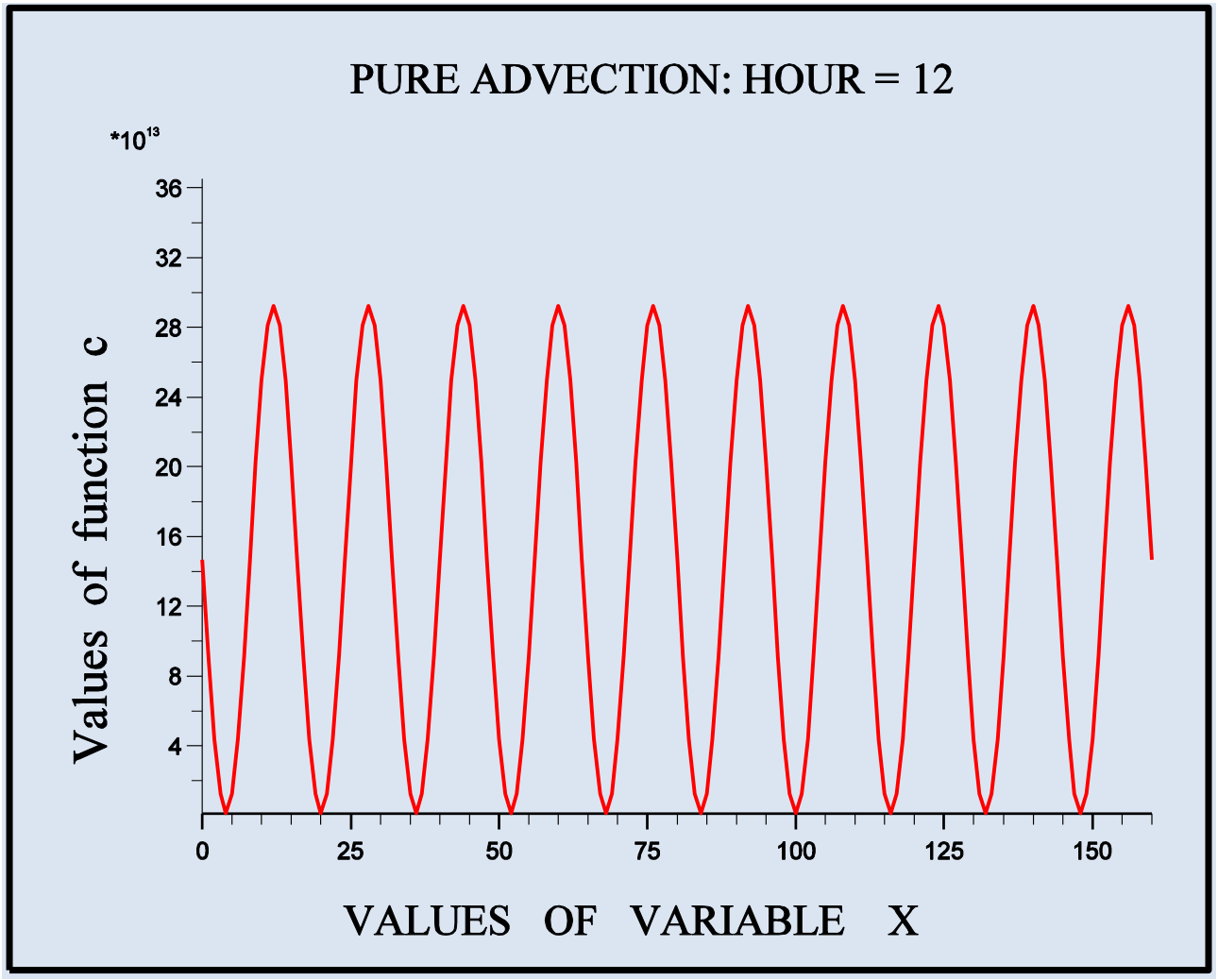

Figure 6.5

The calculated solution at the end of the twelfth hour of the oscillatory one-dimensional advection equation, which was defined in §6.5.3.

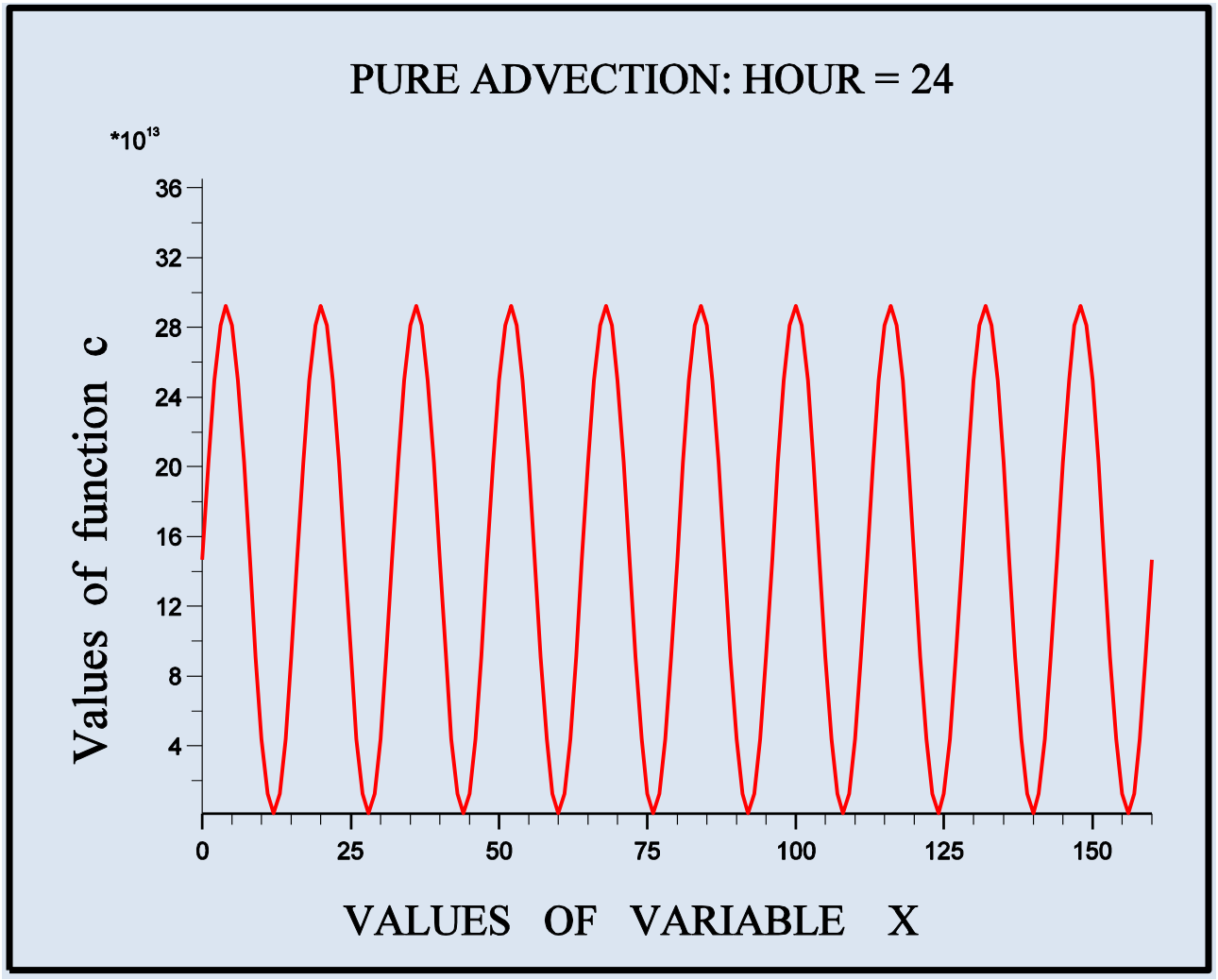

Figure 6.6

The calculated solution at the end of the twenty fourth hour of the oscillatory one-dimensional advection equation, which was defined in §6.5.3.

Two major conclusions can be drawn from the results presented in Table 6.3:

(a) neither the direct Crank-Nicolson scheme, nor the combination of the Crank-Nicolson scheme with the Richardson Extrapolation gives the prescribed by the theory accuracy (orders two and four, respectively)

and

(b) also in this case, i.e. in the presence of discontinuities, the combination of the Crank-Nicolson scheme and the Richardson Extrapolation is considerably more accurate than the direct application of the Crank-Nicolson scheme.

No.	NT	NX	Direct Solution		Richardson Extrapolation	
			Error	Ratio	Error	Ratio
1	168	160	1.353E-01	-	4.978E-02	-
2	336	320	7.687E-02	1.760	2.761E-02	1.803
3	672	640	4.424E-02	1.737	1.551E-02	1.780
4	1344	1280	2.555E-02	1.732	8.570E-03	1.810
5	2688	2560	1.636E-02	1.561	4.590E-03	1.867
6	5376	5120	1.051E-02	1.552	2.318E-03	1.980
7	10752	10240	5.551E-03	1.899	1.188E-03	1.951
8	21504	20480	2.921E-03	1.900	6.575E-04	1.807
9	43008	40960	2.644E-03	1.105	2.379E-04	2.746
10	86016	81920	1.619E-03	1.633	1.501E-04	1.585
11	172032	163840	1.145E-03	1.414	2.787E-05	4.941

Table 6.3

Results obtained when the test-problem defined by (6.32), (6.33) and (6.40)-(6.42), i.e. the test with discontinuous derivatives of the unknown function $\mathbf{c}(\mathbf{x}, \mathbf{t})$ is handled directly by the Crank-Nicolson scheme and by using the combination of the Crank-Nicolson scheme and the fourth implementation of the Richardson Extrapolation. The numerical errors calculated by (6.30) and (6.31) are given in the columns under "Error". In row $\mathbf{i}$, where $\mathbf{i = 2, 3, \ldots, 11}$, the ratios of the errors in this row and in the previous row are given in the columns under "Ratio".

Plots, which show

(a) the initial values,

(b) the solution in the middle of the time interval (i.e. after 12 hours)

and

(c) the solution at the end of the time interval

for the oscillatory test-problem, are given in Fig. 6.7 – Fig. 6.9, respectively.

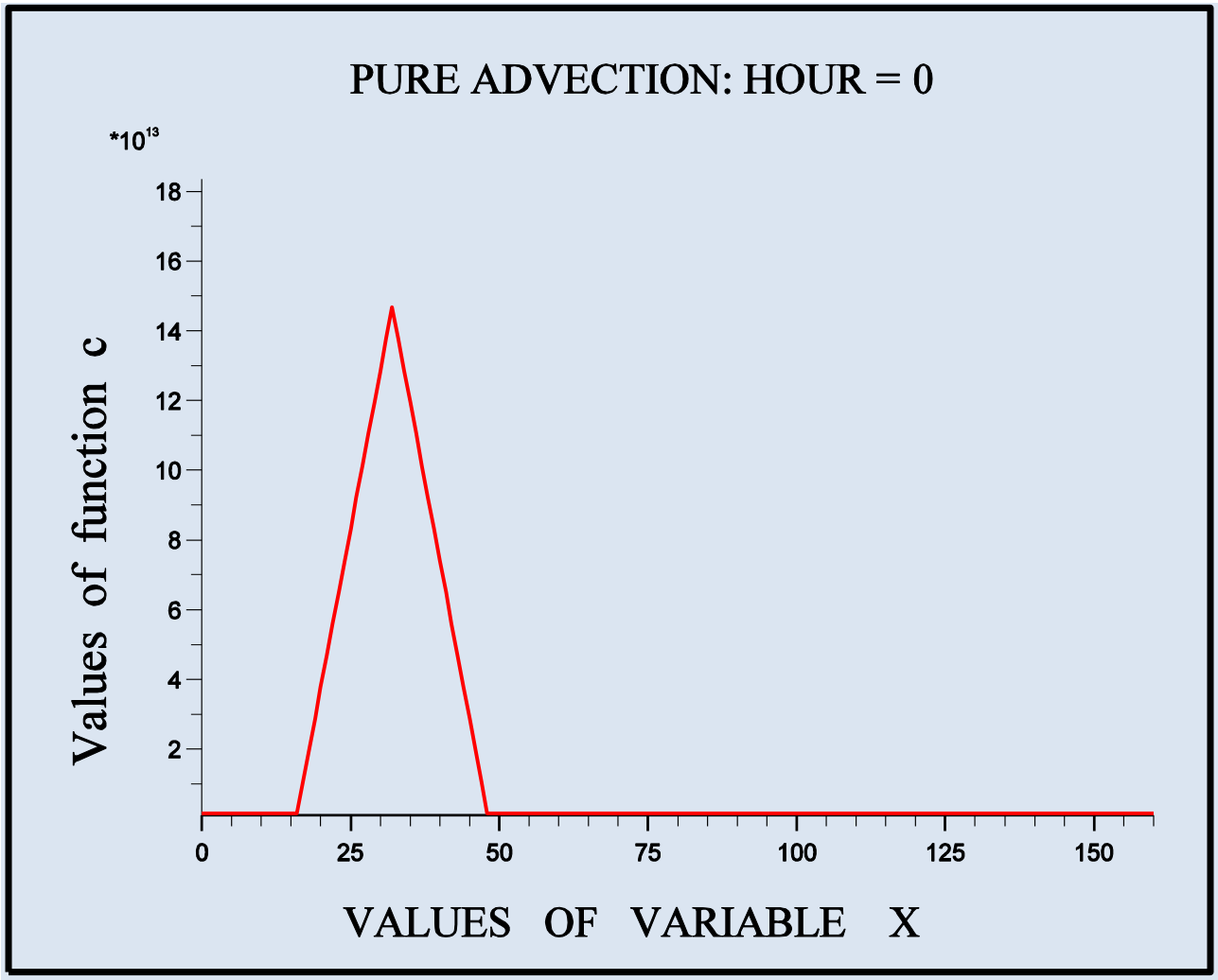

Figure 6.7

The initial value of the solution of the one-dimensional advection equation, which was defined in §6.5.4, i.e. the example with discontinuities in the derivatives of the unknown function .

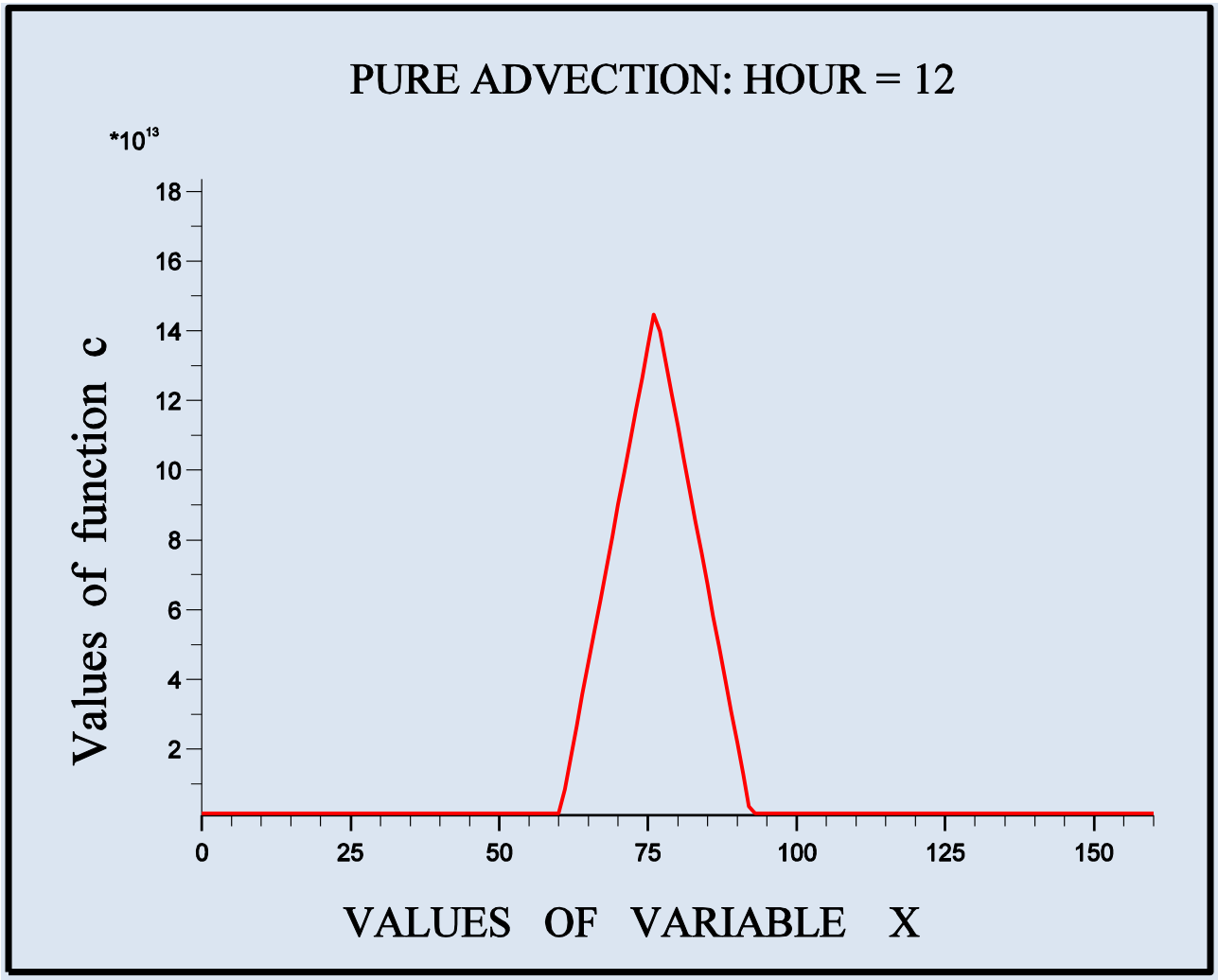

Figure 6.8

The calculated at the end of the twelfth hour solution of the one-dimensional advection equation, which was defined in §6.5.4, i.e. the example with discontinuities in the derivatives of the unknown function.

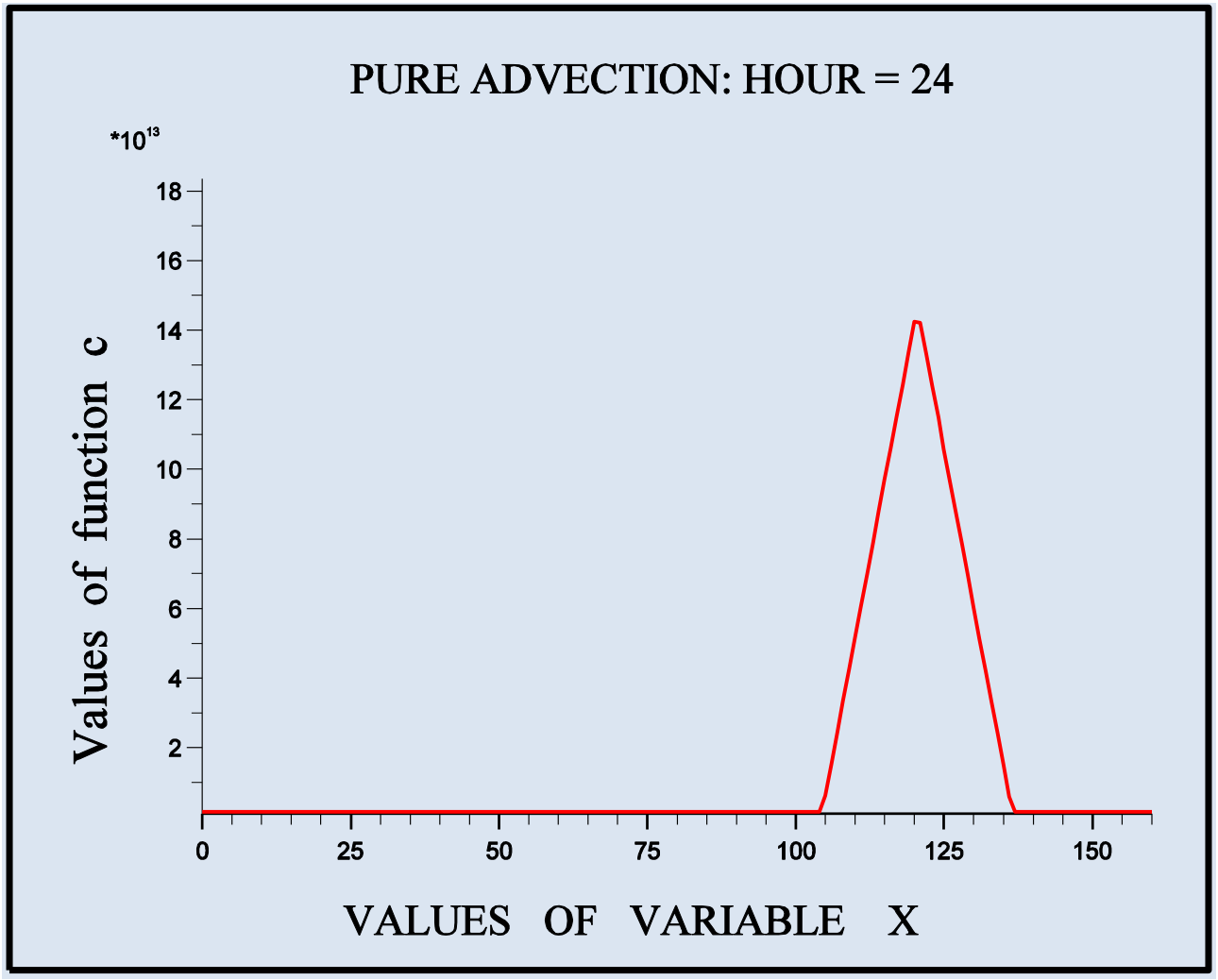

Figure 6.9

The calculated at the end of the twenty fourth hour solution of the one-dimensional advection equation, which was defined in §6.5.4, i.e. the example with discontinuities in the derivatives of the unknown function.

6.5.5. Comparison of the four implementations of the Richardson Extrapolation

Only the fourth implementation of the Richardson Extrapolation was used in §6.5.2, §6.5.3 and §6.5.4. Now we shall compare this implementation with the other three. Some results, which are obtained by applying the oscillatory test-problem from §6.5.3, are given in Table 6.4.

The following conclusions can be drawn from the results presented in Table 6.4:

- The fourth implementation of the Richardson Extrapolation, which is based on the use of the third-order interpolation rule described in Section 6.3, is performing much better than the other three (giving accuracy which is by several orders of magnitude better than that obtained by the first three implementations).

- The first three implementations give accuracy which is slightly better, but anyway of the same order as that obtained when the Crank-Nicolson scheme is used directly (compare the results given in columns four, five and six of Table 6.4 with the results in the fourth column of Table 6.2).

- The results show very clearly that one must be very careful when implements the Richardson Extrapolation.

More numerical results can be found in **Zlatev et al. (2011a)**.

No.	NT	NX	Richardson Extrapolation			
			Active	Passive	Lin. Interp.	Third-order Interp.
1	168	160	2.044E-01	2.789E-01	3.829E-01	1.560E-02
2	336	320	4.948E-02	7.135E-02	1.185E-01	1.227E-03
3	672	640	1.254E-02	1.760E-02	2.466E-02	1.073E-04
4	1344	1280	3.145E-03	4.334E-03	6.250E-03	1.150E-05
5	2688	2560	7.871E-04	1.074E-03	1.567E-03	1.193E-06
6	5376	5120	1.968E-04	2.671E-04	3.921E-04	1.478E-07
7	10752	10240	4.922E-05	6.659E-05	9.806E-05	1.618E-08
8	21504	20480	1.230E-05	1.663E-05	2.452E-05	1.965E-09
9	43008	40960	3.076E-06	4.154E-06	6.129E-06	2.387E-10
10	86016	81920	7.960E-07	1.038E-06	1.532E-06	3.241E-11
11	172032	163840	1.923E-07	2.595E-07	3.831E-07	1.267E-11

Table 6.4

Running the oscillatory advection test-example from §6.5.3 by using the Crank-Nicolson scheme directly and in combination with four implementations of the Richardson Extrapolation.

6.6. Multi-dimensional advection problem

The results obtained for the one-dimensional advection problem can be extended for the multi-dimensional case. This will be done in this section.

6.6.1. Introduction of the multi-dimensional advection equation

The multi-dimensional advection equation (**Zlatev et al., 2014**) can be represented by the following formula:

$$(6.43) \quad \frac{\partial c}{\partial t} = -\sum_{q=1}^{Q} u_q \frac{\partial c}{\partial x_q} \quad , \quad x_q \in [a_q, b_q] , \quad q = 1 , 2 , \ldots , Q , \quad t \in [a, b] .$$

It is assumed that the coefficients $u_q = u_q(t , x_1 , x_2 , \ldots , x_Q) , \quad q = 1 , 2 , \ldots , Q ,$ before the spatial partial derivatives in the right-hand-side of (6.43) are some given functions.

Let D be the domain in which the independent variables involved in (6.43) vary and assume that:

$$(6.44) \quad (t , x_1 , x_2 , \ldots , x_Q) \in D \quad \Longrightarrow \quad t \in [a, b] \ \wedge \ x_q \in [a_q, b_q] , \quad q = 1 , 2 , \ldots , Q \ .$$

By applying the definition proposed in (6.44), it is assumed here that the domain D is rather special, but this is done only for the sake of simplicity. In fact, many of the results will also be valid for some more complicated domains.

It will always be assumed that the unknown function $c = c(t , x_1 , x_2 , \ldots , x_Q)$ is continuously differentiable up to some order $2p$ with $p \geq 1$ in all points of the domain D and for all independent variables. Here p is the order of the numerical method which will be used in order to obtain some approximations of the unknown function at some mesh defined somehow in the domain (6.44). For some of the proofs it will also be necessary to assume that continuous derivatives up to order two of all functions u_q exist with respect of all independent variables.

The multi-dimensional advection equation (6.43) must always be considered, as the one-dimensional advection equation (6.1), together with appropriate initial and boundary conditions.

The following abbreviations will be useful in the efforts to facilitate the presentation of the results in this section. Note that some given positive increments h_q appears in the equalities (6.46) – (6.49).

$$(6.45) \quad \bar{x} = (x_1 , x_2 , \ldots , x_Q) ,$$

$$(6.46) \quad \bar{x}^{(+q)} = (x_1 , x_2 , \ldots , x_{q-1} , x_q + h_q , x_{q+1} , \ldots , x_Q) , \quad q = 1 , 2 , \ldots , Q ,$$

$$(6.47) \quad \bar{x}^{(-q)} = (x_1 , x_2 , \ldots , x_{q-1} , x_q - h_q , x_{q+1} , \ldots , x_Q) , \quad q = 1 , 2 , \ldots , Q ,$$

$$(6.48) \quad \bar{x}^{(+0.5q)} = (x_1 , x_2 , \ldots , x_{q-1} , x_q + 0.5 h_q , x_{q+1} , \ldots , x_Q) , \quad q = 1 , 2 , \ldots , Q ,$$

$$(6.49) \quad \bar{x}^{(-0.5q)} = (x_1 , x_2 , \ldots , x_{q-1} , x_q - 0.5 h_q , x_{q+1} , \ldots , x_Q) , \quad q = 1 , 2 , \ldots , Q ,$$

6.6.2. Expanding the unknown function in Taylor series

By using appropriate expansions of the unknown function $\mathbf{c}(\mathbf{t}, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_Q}) = \mathbf{c}(\mathbf{t}, \bar{\mathbf{x}})$ in Taylor series it is possible to find an expression that contains only even degrees of the increments $\mathbf{k}$ and $\mathbf{h_q}$. More precisely, the following theorem holds.

Theorem 6.3: Consider the multi-dimensional advection equation (6.43), assume that $(\mathbf{t}, \bar{\mathbf{x}}) \in \mathbf{D}$ is an arbitrary but fixed point and introduce the positive increments $\mathbf{k} > \mathbf{0}$ and $\mathbf{h_q} > \mathbf{0}$ such that $\mathbf{t} + \mathbf{k} \in [\mathbf{a}, \mathbf{b}]$, $\mathbf{x_q} - \mathbf{h_q} \in [\mathbf{a_q}, \mathbf{b_q}]$ and $\mathbf{x_q} + \mathbf{h_q} \in [\mathbf{a_q}, \mathbf{b_q}]$ for all $\mathbf{q} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{Q}$. Assume furthermore that the unknown function $\mathbf{c}(\mathbf{t}, \bar{\mathbf{x}})$ is continuously differentiable up to some order $\mathbf{2p}$ with regard to **all** independent variables. Then there exists an expansion in Taylor series of the unknown function $\mathbf{c}(\mathbf{t}, \bar{\mathbf{x}})$ around the point $(\mathbf{t} + \mathbf{0.5k}, \bar{\mathbf{x}})$ which contains terms involving only even degrees of the increments $\mathbf{k}$ and $\mathbf{h_q}$ $(\mathbf{q} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{Q})$.

Proof: It is clear that the following two formulae hold when the assumptions of Theorem 6.3 are satisfied:

$$(\mathbf{6.50}) \qquad \mathbf{c}(\mathbf{t}+\mathbf{k}, \bar{\mathbf{x}}) = \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}}) + \frac{\mathbf{k}}{\mathbf{2}} \frac{\partial \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})}{\partial \mathbf{t}} + \sum_{\mathbf{s}=\mathbf{2}}^{\mathbf{2p-1}} \frac{\mathbf{k^s}}{\mathbf{2^s\, s!}} \frac{\partial^{\mathbf{s}} \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})}{\partial \mathbf{t^s}} + \mathbf{O}(\mathbf{k^{2p}}),$$

$$(\mathbf{6.51}) \qquad \mathbf{c}(\mathbf{t}, \bar{\mathbf{x}}) = \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}}) - \frac{\mathbf{k}}{\mathbf{2}} \frac{\partial \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})}{\partial \mathbf{t}} + \sum_{\mathbf{s}=\mathbf{2}}^{\mathbf{2p-1}} (-\mathbf{1})^{\mathbf{s}} \frac{\mathbf{k^s}}{\mathbf{2^s\, s!}} \frac{\partial^{\mathbf{s}} \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})}{\partial \mathbf{t^s}} + \mathbf{O}(\mathbf{k^{2p}}).$$

Eliminate the quantity $\mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})$ from (6.50) and ((6.51), which can be achieved by subtracting (6.51) from (6.50). The result is:

$$(\mathbf{6.52}) \qquad \mathbf{c}(\mathbf{t}+\mathbf{k}, \bar{\mathbf{x}}) - \mathbf{c}(\mathbf{t}, \bar{\mathbf{x}}) = \mathbf{k} \frac{\partial \mathbf{c}(\mathbf{t}+\mathbf{0.5k}, \bar{\mathbf{x}})}{\partial \mathbf{t}}$$

$$+ 2 \sum_{s=1}^{p-1} \frac{k^{2s+1}}{2^{s+1}\,(2s+1)!} \frac{\partial^{2s+1} c(t+0.5k\,,\bar{x})}{\partial t^{2s+1}} + O(k^{2p})\,.$$

The last equality can be rewritten as

$$(6.53) \qquad \frac{\partial c(t+0.5k\,,\bar{x})}{\partial t} = \frac{c(t+k\,,\bar{x}) - c(t\,,\bar{x})}{k} + \sum_{s=1}^{p-1} \frac{k^{2s}}{2^{s}\,(2s+1)!} \frac{\partial^{2s+1} c(t+0.5k\,,\bar{x})}{\partial t^{2s+1}} + O(k^{2p-1})\,.$$

Consider the following two relationships:

$$(6.54) \qquad \frac{\partial c(t+k\,,\bar{x})}{\partial x_q} = \frac{\partial c(t+0.5k\,,\bar{x})}{\partial x_q} + \sum_{s=1}^{2p-1} \frac{k^{s}}{2^{s}\,s!} \frac{\partial^{s+1} c(t+0.5k\,,\bar{x})}{\partial t^{s} \partial x_q} + O(k^{2p})\,,$$

$$(6.55) \qquad \frac{\partial c(t\,,\bar{x})}{\partial x_q} = \frac{\partial c(t+0.5k\,,\bar{x})}{\partial x_q} + \sum_{s=1}^{2p-1} (-1)^{s} \frac{k^{s}}{2^{s}\,s!} \frac{\partial^{s+1} c(t+0.5k\,,\bar{x})}{\partial t^{s} \partial x_q} + O(k^{2p})\,.$$

Add (6.54) to (6.55). The result is:

$$(6.56) \qquad \frac{\partial c(t+k\,,\bar{x})}{\partial x_q} + \frac{\partial c(t\,,\bar{x})}{\partial x_q} = 2\,\frac{\partial c(t+0.5k\,,\bar{x})}{\partial x_q} + 2 \sum_{s=1}^{p-1} \frac{k^{2s}}{2^{2s}\,(2s)!} \frac{\partial^{2s+1} c(t+0.5k\,,\bar{x})}{\partial t^{2s} \partial x_q} + O(k^{2p})\,.$$

The last equality can be rewritten as:

$$(6.57) \qquad \frac{\partial c(t+0.5k\,,\bar{x})}{\partial x_q} = \frac{1}{2}\left[\frac{\partial c(t+k\,,\bar{x})}{\partial x_q} + \frac{\partial c(t\,,\bar{x})}{\partial x_q}\right]$$

$$- \sum_{s=1}^{p-1} \frac{k^{2s}}{2^{2s}\,(2s)!} \frac{\partial^{2s+1} c(t+0.5k, \bar{x})}{\partial t^{2s} \partial x_q} + O(k^{2p}) .$$

Now the following four relationships can be written:

$$(6.58) \qquad c\left(t+k, \bar{x}^{(+q)}\right) = c(t+k, \bar{x}) + h_q \frac{\partial c(t+k, \bar{x})}{\partial x_q} + \sum_{s=2}^{2p-1} \frac{h_q^s}{s!} \frac{\partial^s c(t+k, \bar{x})}{\partial x_q^s} + O\left(h_q^{2p}\right),$$

$$(6.59) \qquad c\left(t+k, \bar{x}^{(-q)}\right) = c(t+k, \bar{x}) - h_q \frac{\partial c(t+k, \bar{x})}{\partial x_q} + \sum_{s=2}^{2p-1} (-1)^s \frac{h_q^s}{s!} \frac{\partial^s c(t+k, \bar{x})}{\partial x_q^s} + O\left(h_q^{2p}\right),$$

$$(6.60) \qquad c\left(t, \bar{x}^{(+q)}\right) = c(t, \bar{x}) + h_q \frac{\partial c(t, \bar{x})}{\partial x_q} + \sum_{s=2}^{2p-1} \frac{h_q^s}{s!} \frac{\partial^s c(t, \bar{x})}{\partial x_q^s} + O\left(h_q^{2p}\right),$$

$$(6.61) \qquad c\left(t, \bar{x}^{(-q)}\right) = c(t, \bar{x}) - h_q \frac{\partial c(t, \bar{x})}{\partial x_q} + \sum_{s=2}^{2p-1} (-1)^s \frac{h_q^s}{s!} \frac{\partial^s c(t, \bar{x})}{\partial x_q^s} + O\left(h_q^{2p}\right),$$

Subtract (6.59) from (6.58) to obtain:

$$(6.62) \qquad \frac{\partial c(t+k, \bar{x})}{\partial x_q} = \frac{c\left(t+k, \bar{x}^{(+q)}\right) - c\left(t+k, \bar{x}^{(-q)}\right)}{2h_q}$$

$$-\sum_{s=2}^{p-1}\frac{h_q^{2s}}{(2s+1)!}\frac{\partial^{2s+1}c(t+k,\bar{x})}{\partial x_q^{2s+1}}+O\left(h_q^{2p-1}\right).$$

Similarly, the following relationship can be obtained by subtracting (6.61) from (6.60):

$$(6.63)\qquad \frac{\partial c(t,\bar{x})}{\partial x_q}=\frac{c\left(t,\bar{x}^{(+q)}\right)-c\left(t,\bar{x}^{(-q)}\right)}{2h_q}$$

$$-\sum_{s=2}^{p-1}\frac{h_q^{2s}}{(2s+1)!}\frac{\partial^{2s+1}c(t,\bar{x})}{\partial x_q^{2s+1}}+O\left(h_q^{2p-1}\right).$$

Assume that $(t+0.5,k\,\bar{x})=\left(t+0.5k,\ x_1,\ x_1,\ \ldots,x_Q\right)$ is some arbitrary but fixed point in the domain D. Then use the abbreviation $u_q(t+0.5,k\,\bar{x})=u_q\left(t+0.5k,\ x_1,\ x_1,\ \ldots,x_Q\right)$ in order to obtain, from (6.43), the following formula:

$$(6.64)\qquad \frac{\partial c(t+0.5,k\,\bar{x})}{\partial t}=-\sum_{q=1}^{Q}u_q(t+0.5,k\,\bar{x})\ \frac{\partial c(t+0.5,k\,\bar{x})}{\partial x_q}\ .$$

Use (6.53) and (6.57) in (6.64) to obtain:

$$(6.65)\qquad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k}=-\sum_{q=1}^{Q}u_q(t+0.5,k\,\bar{x})\left\{\frac{1}{2}\left[\frac{\partial c(t+0.5k,\bar{x})}{\partial x_q}+\frac{\partial c(t,\bar{x})}{\partial x_q}\right]\right\}$$

$$+\sum_{q=1}^{Q}u_q(t+0.5,k\,\bar{x})\left\{\sum_{s=1}^{p-1}\frac{k^{2s}}{2^{2s}\,(2s+1)!}\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s}\partial x_q}\right\}$$

$$+\sum_{s=1}^{p-1}\frac{k^{2s}}{2^{2s}\,(2s+1)!}\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s+1}}+O(k^{2p-1})\ .$$

The last equality can be rewritten as

$$(6.66)\quad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k} = -\sum_{q=1}^{Q} u_q(t+0.5,k\,\bar{x})\left\{\frac{1}{2}\left[\frac{\partial c(t+0.5k,\bar{x})}{\partial x_q}+\frac{\partial c(t,\bar{x})}{\partial x_q}\right]\right\}$$
$$+\sum_{s=1}^{p-1}\frac{k^{2s}}{2^{2s}\,(2s)!}\left\{\frac{1}{(2s+1)}\,\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s+1}}\right.$$
$$\left.+\sum_{q=1}^{Q} u_q(t+0.5,k\,\bar{x})\,\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s}\partial x_q}\right\}+O(k^{2p-1})\,.$$

Denote:

$$(6.67)\quad K_t^{(2s)} = \frac{1}{2^{2s}(2s)!}\left[\frac{1}{(2s+1)}\,\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s+1}}\right.$$
$$\left.+\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\,\frac{\partial^{2s+1}c(t+0.5k,\bar{x})}{\partial t^{2s}\partial x_q}\right].$$

Then (6.65) can be rewritten as:

$$(6.68)\quad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k} = -\sum_{q=1}^{Q} u_q(t+0.5,k\,\bar{x})\left\{\frac{1}{2}\left[\frac{\partial c(t+0.5k,\bar{x})}{\partial x_q}+\frac{\partial c(t,\bar{x})}{\partial x_q}\right]\right\}$$
$$+\sum_{s=1}^{p-1} k^{2s}\,K_t^{(2s)}+O(k^{2p-1})\quad.$$

Use (6.63) and (6.64) in the expression in the square bracket of (6.67) to obtain

$$(6.69)\quad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k} = -\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\,\frac{c\left(t+k,\bar{x}^{(+q)}\right)-c\left(t+k,\bar{x}^{-q)}\right)}{4h_q}$$

$$-\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\,\frac{c\left(t,\bar{x}^{(+q)}\right)-c\left(t,\bar{x}^{-q)}\right)}{4h_q}$$

$$+\frac{1}{2}\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\left\{\sum_{s=1}^{p-1}\frac{h_q^{2s}}{(2s+1)!}\left[\frac{\partial^{2s+1}c(t+k,\bar{x})}{\partial x_q^{2s+1}}+\frac{\partial^{2s+1}c(t,\bar{x})}{\partial x_q^{2s+1}}\right]\right\}$$

$$+\sum_{s=1}^{p-1} k^{2s}\,K_t^{(2s)}+O(k^{2p-1})+O\left(h_q^{2p+1}\right).$$

The last equality can be rewritten in the following form:

$$(6.70)\qquad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k}=-\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\,\frac{c\left(t+k,\bar{x}^{(+q)}\right)-c\left(t+k,\bar{x}^{-q)}\right)}{4h_q}$$

$$-\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\,\frac{c\left(t,\bar{x}^{(+q)}\right)-c\left(t,\bar{x}^{-q)}\right)}{4h_q}$$

$$+\sum_{q=1}^{Q}\frac{1}{2}\,\frac{h_q^{2s}}{(2s+1)!}\sum_{s=1}^{p-1} u_q(t+0.5k,\bar{x})\left[\frac{\partial^{2s+1}c(t+k,\bar{x})}{\partial x_q^{2s+1}}+\frac{\partial^{2s+1}c(t,\bar{x})}{\partial x_q^{2s+1}}\right]$$

$$+\sum_{s=1}^{p-1} k^{2s}\,K_t^{(2s)}+O(k^{2p-1})+O\left(h_q^{2p+1}\right).$$

Denote:

$$(6.71)\qquad K_q^{(2s)}=\frac{1}{2}\frac{1}{(2s+1)!}\,u_q(t+0.5k,\bar{x})\left[\frac{\partial^{2s+1}c(t+k,\bar{x})}{\partial x_q^{2s+1}}+\frac{\partial^{2s+1}c(t,\bar{x})}{\partial x_q^{2s+1}}\right].$$

Substitute this value of $K_q^{(2s)}$ in (6.70). The result is:

$$(6.72)\qquad \frac{c(t+k,\bar{x})-c(t,\bar{x})}{k}$$

$$= -\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\frac{c\left(t+k,\bar{x}^{(+q)}\right)-c\left(t+k,\bar{x}^{(-q)}\right)+c\left(t,\bar{x}^{(+q)}\right)-c\left(t,\bar{x}^{(-q)}\right)}{4h_q}$$

$$+\sum_{s=1}^{p-1}\left[k^{2s}\,K_t^{(2s)}+\sum_{q=1}^{Q}h_q^{2s}\,K_q^{(2s)}\right]+O(k^{2p-1})+O\left(h_q^{2p-1}\right)\ .$$

The last equality can also be rewritten as:

$$(6.73)\qquad \frac{c(t+k\,,\bar{x})-c(t\,,\bar{x})}{k}=$$

$$= -\sum_{q=1}^{Q} u_q(t+0.5k,\bar{x})\frac{c\left(t+k,\bar{x}^{(+q)}\right)-c\left(t+k,\bar{x}^{(-q)}\right)+c\left(t,\bar{x}^{(+q)}\right)-c\left(t,\bar{x}^{(-q)}\right)}{4h_q}$$

$$+\sum_{s=1}^{p-1}k^{2s}K^{(2s)}+O(k^{2p-1})\,,$$

where

$$(6.74)\qquad K^{(2s)}=K_t^{(2s)}+\sum_{q=1}^{Q}\frac{h_q^{2s}}{k^{2s}}\,K_q^{(2s)}\ .$$

Assume that all ratios h_q/k, $q=1$, 2, ..., Q, remain constants when $k\to 0$ (which can easily be achieved, for example, by reducing all h_q by a factor of two when k is reduced by a factor of two). Then, the last two equalities, equalities (39) and (40), show that the assertion of Theorem 6 3 holds.

■

6.6.3. Three special cases

Multi-dimensional advection equations arise when many physical processes are described mathematically (for example, in some of the sub-models of a complex large-scale environmental model; see **Alexandrov et al. (2004)**, **Faragó, Havasi and Zlatev (2010)**, **Zlatev (1995)** and **Zlatev and Dimov (2006)**. In most of these cases we have $Q=1$, 2, 3. Therefore, the following three corollaries of **Theorem 6.2** are important for many applications in science and engineering.

Corollary 6.1 (the one-dimensional case): If $\mathbf{Q = 1}$ then (6.43)) can be written as

$$(\mathbf{6.75}) \quad \frac{\partial \mathbf{c}}{\partial \mathbf{t}} = -\mathbf{u_1}\frac{\partial \mathbf{c}}{\partial \mathbf{x_1}} \quad , \quad \mathbf{x_1} \in [\mathbf{a_1}, \mathbf{b_1}] \, , \quad \mathbf{t} \in [\mathbf{a}, \mathbf{b}] \, .$$

and (6.73) reduces to the following equality:

$$(\mathbf{6.76}) \quad \frac{\mathbf{c(t + k\,, x_1) - c(t\,, x_1)}}{\mathbf{k}} = -\mathbf{u_1(t + 0.5k, x_1)}\frac{\mathbf{c(t + k, x_1 + h_1) - c(t + k, x_1 - h_1)}}{\mathbf{4h_1}}$$

$$-\mathbf{u_1(t + 0.5k, \bar{x})}\,\frac{\mathbf{c(t, x_1 + h_1) - c(t, x_1 - h_1)}}{\mathbf{4h_1}}$$

$$+\mathbf{k^2K^{(2)} + k^4K^{(4)} + \cdots + k^{2p-2}K^{(2p-2)} + O(k^{2p-1})}$$

where the values of $\mathbf{K^{(2s)}}$, $\mathbf{s = 1, 2, \ldots, 2p - 2}$ for the one-dimensional case can be obtained in an obvious way from (6.66), (6.71) and (6.74).

■

Corollary 6.2 (the two-dimensional case): If $\mathbf{Q = 2}$ then (6.43) can be written as

$$(\mathbf{6.77}) \quad \frac{\partial \mathbf{c}}{\partial \mathbf{t}} = -\mathbf{u_1}\,\frac{\partial \mathbf{c}}{\partial \mathbf{x_1}} - \mathbf{u_2}\,\frac{\partial \mathbf{c}}{\partial \mathbf{x_2}} \quad , \quad \mathbf{x_1} \in [\mathbf{a_1}, \mathbf{b_1}] \, , \quad \mathbf{x_2} \in [\mathbf{a_2}, \mathbf{b_2}] \, , \quad \mathbf{t} \in [\mathbf{a}, \mathbf{b}] \, .$$

and (6.73) reduces to the following equality:

$$(\mathbf{6.78}) \quad \frac{\mathbf{c(t + k\,, x_1, x_2) - c(t\,, x_1, x_2)}}{\mathbf{k}}$$

$$= -\mathbf{u_1(t + 0.5k, x_1, x_2)}\frac{\mathbf{c(t + k, x_1 + h_1, x_2) - c(t + k, x_1 - h_1, x_2)}}{\mathbf{4h_1}}$$

$$-\mathbf{u_1(t + 0.5k, x_1, x_2)}\,\frac{\mathbf{c(t, x_1 + h_1, x_2) - c(t, x_1 - h_1, x_2)}}{\mathbf{4h_1}}$$

$$-\mathbf{u_2(t + 0.5k, x_1, x_2)}\frac{\mathbf{c(t + k, x_1, x_2 + h_2) - c(t + k, x_1, x_2 - h_2)}}{\mathbf{4h_2}}$$

$$-\mathbf{u_2}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2})\,\frac{\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2}+\mathbf{h_2})-\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2}-\mathbf{h_2})}{\mathbf{4h_2}}$$

$$+\mathbf{k^2K^{(2)}}+\mathbf{k^4K^{(4)}}+\cdots+\mathbf{k^{2p-2}K^{(2p-2)}}+\mathbf{O(k^{2p-1})}\,,$$

where the values of $\mathbf{K^{(2s)}}$, $\mathbf{s=1,2,\ldots,2p-2}$ for the two-dimensional case can be obtained in an obvious way from (6.66), (6.71) and (6.74).

■

Corollary 6.3 (the three-dimensional case): If $\mathbf{Q=3}$ then (6.43) can be written as

$$(\mathbf{6.79})\qquad \frac{\partial \mathbf{c}}{\partial \mathbf{t}} = -\mathbf{u_1}\frac{\partial \mathbf{c}}{\partial \mathbf{x_1}} - \mathbf{u_2}\frac{\partial \mathbf{c}}{\partial \mathbf{x_2}} - \mathbf{u_3}\frac{\partial \mathbf{c}}{\partial \mathbf{x_3}},$$

$$\mathbf{x_1}\in[\mathbf{a_1},\mathbf{b_1}]\,,\quad \mathbf{x_2}\in[\mathbf{a_2},\mathbf{b_2}]\,,\quad \mathbf{x_3}\in[\mathbf{a_3},\mathbf{b_3}]\,,\quad \mathbf{t}\in[\mathbf{a},\mathbf{b}]\,.$$

and (6.73) reduces to the following equality:

$$(\mathbf{6.80})\qquad \frac{\mathbf{c}(\mathbf{t}+\mathbf{k}\,,\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})-\mathbf{c}(\mathbf{t}\,,\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})}{\mathbf{k}}$$

$$=-\mathbf{u_1}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1}+\mathbf{h_1},\mathbf{x_2},\mathbf{x_3})-\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1}-\mathbf{h_1},\mathbf{x_2},\mathbf{x_3})}{\mathbf{4h_1}}$$

$$-\mathbf{u_1}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t},\mathbf{x_1}+\mathbf{h_1},\mathbf{x_2},\mathbf{x_3})-\mathbf{c}(\mathbf{t},\mathbf{x_1}-\mathbf{h_1},\mathbf{x_2},\mathbf{x_3})}{\mathbf{4h_1}}$$

$$-\mathbf{u_2}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1},\mathbf{x_2}+\mathbf{h_2},\mathbf{x_3})-\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1},\mathbf{x_2}-\mathbf{h_2},\mathbf{x_3})}{\mathbf{4h_2}}$$

$$-\mathbf{u_2}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2}+\mathbf{h_2},\mathbf{x_3})-\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2}-\mathbf{h_2},\mathbf{x_3})}{\mathbf{4h_2}}$$

$$-\mathbf{u_3}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3}+\mathbf{h_3})-\mathbf{c}(\mathbf{t}+\mathbf{k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3}-\mathbf{h_3})}{\mathbf{4h_3}}$$

$$-\mathbf{u_3}(\mathbf{t}+\mathbf{0.5k},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3})\,\frac{\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3}+\mathbf{h_3})-\mathbf{c}(\mathbf{t},\mathbf{x_1},\mathbf{x_2},\mathbf{x_3}-\mathbf{h_3})}{\mathbf{4h_3}}$$

$$+\mathbf{k^2K^{(2)}} + \mathbf{k^4K^{(4)}} + \cdots + \mathbf{k^{2p-2}K^{(2p-2)}} + \mathbf{O(k^{2p-1})}$$

where the values of $\mathbf{K^{(2s)}}$, $\mathbf{s = 1, 2, \ldots, 2p - 2}$ for the three-dimensional case can be obtained in an obvious way from (6.66), (6.71) and (6.74).

■

6.6.4. Designing a second-order numerical method for multi-dimensional advection

Consider the grids:

$$(\mathbf{6.81}) \quad \mathbf{G_t} = \left\{ \mathbf{t_n} \mid \mathbf{t_0 = a},\ \ \mathbf{t_n = t_{n-1} + k},\ \ \mathbf{n = 1, 2},\ \ \ldots,\ \ \mathbf{N_t},\ \ \mathbf{k = \frac{b - a}{N_t}},\ \ \mathbf{t_{N_t} = b} \right\}$$

and (for $\mathbf{q = 1}$, $\mathbf{2}$, $\ldots$, $\mathbf{Q}$ and $\mathbf{h_q = (b_q - a_q)/N_q}$)

$$(\mathbf{6.82}) \quad \mathbf{G_x^{(q)}} = \left\{ \mathbf{x_q^{i_q}}, \mathbf{i_q = 0, 1, \ldots, N_q} \mid \mathbf{x_q^0 = a_q},\ \mathbf{x_q^{i_q} = x_q^{i_q - 1} + h_q},\ \mathbf{i_q = 1, 2, \ldots, N_q},\ \mathbf{x_q^{N_q} = b_q} \right\}.$$

Consider (assuming that $\mathbf{j_q} \in \{\mathbf{0, 1, \ldots, N_q}\}$):

$$(\mathbf{6.83}) \quad \tilde{\mathbf{x}} = \left(\mathbf{x_1^{j_1}},\ \mathbf{x_2^{j_2}},\ \ldots,\ \mathbf{x_Q^{j_Q}} \right), \qquad \mathbf{q = 1},\ \mathbf{2},\ \ldots,\ \mathbf{Q},$$

$$(\mathbf{6.84}) \quad \tilde{\mathbf{x}}^{(+\mathbf{q})} = \left(\mathbf{x_1^{j_1}}, \mathbf{x_2^{j_2}}, \ldots, \mathbf{x_q^{j_q - 1}},\ \mathbf{x_q^{j_q} + h_q},\ \mathbf{x_q^{j_q + 1}}, \ldots,\ \mathbf{x_Q^{j_Q}} \right), \quad \mathbf{q = 1},\ \mathbf{2},\ \ldots,\ \mathbf{Q},$$

$$(\mathbf{6,85}) \quad \tilde{\mathbf{x}}^{(-\mathbf{q})} = \left(\mathbf{x_1^{j_1}}, \mathbf{x_2^{j_2}}, \ldots, \mathbf{x_q^{j_q - 1}},\ \mathbf{x_q^{j_q} - h_q},\ \mathbf{x_q^{j_q + 1}}, \ldots,\ \mathbf{x_Q^{j_Q}} \right), \quad \mathbf{q = 1},\ \mathbf{2},\ \ldots,\ \mathbf{Q},$$

where $\mathbf{x_q^{j_q}} \in \mathbf{G_x^{(q)}}$ for $\mathbf{q = 1}$, $\mathbf{2}$, $\ldots$, $\mathbf{Q}$.

In this notation the following numerical method can be defined:

$$(\mathbf{6.86}) \quad \frac{\tilde{\mathbf{c}}(\mathbf{t_{n+1}}, \tilde{\mathbf{x}}) - \tilde{\mathbf{c}}(\mathbf{t_n}, \tilde{\mathbf{x}})}{\mathbf{k}}$$

$$= -\sum_{q=1}^{Q} u_q(t_n + 0.5k, \bar{x}) \frac{\tilde{c}\left(t_{n+1}, \tilde{x}^{(+q)}\right) - \tilde{c}\left(t_{n+1}, \tilde{x}^{(-q)}\right) + \tilde{c}\left(t_n, \tilde{x}^{(+q)}\right) - \tilde{c}\left(t_n, \tilde{x}^{(-q)}\right)}{4h_q},$$

where $\tilde{c}(t_{n+1}, \tilde{x})$, $\tilde{c}(t_n, \tilde{x})$, $\tilde{c}\left(t_{n+1}, \tilde{x}^{(+q)}\right)$, $\tilde{c}\left(t_{n+1}, \tilde{x}^{(-q)}\right)$, $\tilde{c}\left(t_n, \tilde{x}^{(+q)}\right)$ and $\tilde{c}\left(t_n, \tilde{x}^{(-q)}\right)$ are some approximations of the exact values of the solution of multi-dimensional equation (6.43 calculated at appropriate grid-points of (6.81) and (6.82). It is clear that (6,86) can be obtained from (6.73) by neglecting the terms in the last line and by considering only values of the independent variables, which belong to (6.81) and (6.82). It is clear (see the assertion of **Theorem 6.3**) that the method defined by (6.52) is of order two with respect to all independent variables.

Assume that the values of $\tilde{c}(t_n, \tilde{x})$ have been calculated at some time $t = t_n \in G_t$ for all grid-points of (6.82). Then the values $\tilde{c}(t_{n+1}, \tilde{x})$ of the unknown function at the next time-point $t = t_{n+1} = t_n + k \in G_t$ can be obtained by solving a huge system of linear algebraic equations of dimension $\tilde{N}$, where $\tilde{N}$ is defined by

$$(6.87) \quad \tilde{N} = \prod_{q=1}^{Q} (N_q - 1).$$

It should be mentioned that the numerical method defined by (6.86) is called the Crank-Nicolson scheme when $Q = 1$, see, for example **Strikwerda (2004).**

6.6.5. Application of Richardson Extrapolation

Consider (6.86) with $\tilde{c}$ replaced by z when $t = t_{n+1}$:

$$(6.88) \quad \frac{z(t_{n+1}, \tilde{x}) - \tilde{c}(t_n, \tilde{x})}{k}$$

$$= -\sum_{q=1}^{Q} u_q(t_n + 0.5k, \tilde{x}) \frac{z\left(t_{n+1}, \tilde{x}^{(+q)}\right) - z\left(t_{n+1}, \tilde{x}^{(-q)}\right) + \tilde{c}\left(t_n, \tilde{x}^{(+q)}\right) - \tilde{c}\left(t_n, \tilde{x}^{(-q)}\right)}{4h_q}.$$

Suppose that $0.5k$ and $0.5h_q$ are considered instead of k and h_q ($q = 1, 2, \ldots, Q$) respectively. Consider the formulae (6.47) and (6.49), but assume that the independent variables are restricted to the grid-points of the grids (6.81) and (6.62). Then, by using the notation introduced in (6.84) and (6.85) for $q = 1, 2, \ldots, Q$, the following two formulae can be derived:

$$(6.89)\qquad \frac{\mathbf{w}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})-\tilde{\mathbf{c}}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{0.5k}}$$

$$=-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\tilde{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4(0.5h_q)}}$$

$$-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\bar{\mathbf{x}})\frac{\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4(0.5h_q)}},$$

$$(6.90)\qquad \frac{\mathbf{w}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\mathbf{w}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\mathbf{0.5k}}$$

$$=-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4(0.5h_q)}}$$

$$-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\bar{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4(0.5h_q)}}.$$

Add (6.89) to (6.90) and multiply by $\mathbf{0.5}$ the obtained equation. The result is:

$$(6.91)\qquad \frac{\mathbf{w}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\tilde{\mathbf{c}}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{k}}$$

$$=-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4h_q}}$$

$$-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4h_q}}$$

$$-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\tilde{\mathbf{x}})\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4h_q}}$$

$$-\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\bar{\mathbf{x}})\frac{\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+0.5q)}\right)-\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-0.5q)}\right)}{\mathbf{4h_q}}.$$

Multiply (6.88) by $\mathbf{1/3}$ and (6.91) by $\mathbf{4/3}$. Subtract the modified equality (6.88) from the modified equality (6.91). Then the following equality will be obtained:

$$(\mathbf{6.92}) \quad \frac{\mathbf{4}}{\mathbf{3}}\,\frac{\mathbf{w}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\tilde{\mathbf{c}}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{k}}-\frac{\mathbf{1}}{\mathbf{3}}\,\frac{\mathbf{z}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\tilde{\mathbf{c}}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{k}}$$

$$=-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\,\frac{\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\mathbf{w}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\,\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\tilde{\mathbf{x}})\,\frac{\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\mathbf{w}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.25k},\bar{\mathbf{x}})\,\frac{\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$+\frac{\mathbf{1}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})\,\frac{\mathbf{z}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(+\mathbf{q})}\right)-\mathbf{z}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(-\mathbf{q})}\right)}{\mathbf{4h_q}}$$

$$+\frac{\mathbf{1}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.5k},\bar{\mathbf{x}})\,\frac{\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+\mathbf{q})}\right)-\tilde{\mathbf{c}}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-\mathbf{q})}\right)}{\mathbf{4h_q}}\,.$$

Replace the approximate values $\mathbf{z}$, $\mathbf{w}$ and $\tilde{\mathbf{c}}$ in (6.92) with the corresponding exact values of the unknown function to derive the following formula:

$$(\mathbf{6.93}) \quad \frac{\mathbf{4}}{\mathbf{3}}\,\frac{\mathbf{c}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{k}}-\frac{\mathbf{1}}{\mathbf{3}}\,\frac{\mathbf{c}(\mathbf{t_{n+1}},\tilde{\mathbf{x}})-\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\mathbf{k}}$$

$$=-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\,\frac{\mathbf{c}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\mathbf{c}\left(\mathbf{t_{n+1}},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$-\frac{\mathbf{4}}{\mathbf{3}}\sum_{\mathbf{q=1}}^{\mathbf{Q}}\mathbf{u_q}(\mathbf{t_n}+\mathbf{0.75k},\tilde{\mathbf{x}})\,\frac{\mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+\mathbf{0.5q})}\right)-\mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-\mathbf{0.5q})}\right)}{\mathbf{4h_q}}$$

$$-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n+0.25k,\tilde{x})\frac{c\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-c\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n+0.25k,\bar{x})\frac{c\left(t_n,\tilde{x}^{(+0.5q)}\right)-c\left(t_n,\tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$+\frac{1}{3}\sum_{q=1}^{Q} u_q(t_n+0.5k,\tilde{x})\frac{c\left(t_{n+1},\tilde{x}^{(+q)}\right)-c\left(t_{n+1},\tilde{x}^{(-q)}\right)}{4h_q}$$

$$+\frac{1}{3}\sum_{q=1}^{Q} u_q(t_n+0.5k,\bar{x})\frac{c\left(t_n,\tilde{x}^{(+q)}\right)-c\left(t_n,\tilde{x}^{(-q)}\right)}{4h_q}$$

$$+\frac{4}{3}k^2\left(0.5\bar{K}^{(2)}+0.5\tilde{K}^{(2)}\right)-\frac{1}{3}k^2K^{(2)}+O(k^4)\,.$$

Equation (6.73) with $p=2$ can be used, together with transformations similar to those made in §6.6.2 (some of the needed transformations will in fact be performed in the remaining part of this section), in order to obtain the last terms in (6.93). Note too, that the assumed in §6.6.2 after equation (6.74), relationship $h_q/k=\text{const}$ is used to get the last term in (5.93).

Subtract (6.92) from (6.93) and use the notation ε for the differences between the corresponding exact and approximate values of the unknown function. The result is:

$$(6.94)\qquad \frac{4}{3}\frac{\varepsilon(t_{n+1},\tilde{x})-\varepsilon(t_n,\tilde{x})}{k}-\frac{1}{3}\frac{\varepsilon(t_{n+1},\tilde{x})-\varepsilon(t_n,\tilde{x})}{k}$$

$$=-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n+0.75k,\tilde{x})\frac{\varepsilon\left(t_n+k,\tilde{x}^{(+0.5q)}\right)-\varepsilon\left(t_n+k,\tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n+0.75k,\tilde{x})\frac{\varepsilon\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-\varepsilon\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n+0.25k,\tilde{x})\frac{\varepsilon\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-\varepsilon\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$-\frac{4}{3}\sum_{q=1}^{Q} u_q(t_n + 0.25k, \bar{x})\frac{\varepsilon\left(t_n, \tilde{x}^{(+0.5q)}\right) - \varepsilon\left(t_n, \tilde{x}^{(-0.5q)}\right)}{4h_q}$$

$$+\frac{1}{3}\sum_{q=1}^{Q} u_q(t_n + 0.5k, \tilde{x})\frac{\varepsilon\left(t_{n+1}, \tilde{x}^{(+q)}\right) - \varepsilon\left(t_{n+1}, \tilde{x}^{(-q)}\right)}{4h_q}$$

$$+\frac{1}{3}\sum_{q=1}^{Q} u_q(t_n + 0.5k, \bar{x})\frac{\varepsilon\left(t_n, \tilde{x}^{(+q)}\right) - \varepsilon\left(t_n, \tilde{x}^{(-q)}\right)}{4h_q}$$

$$+ k^2\left[\frac{2}{3}\left(\bar{K}^{(2)} + \tilde{K}^{(2)}\right) - \frac{1}{3}K^{(2)}\right] + O(k^4)\,.$$

The interesting term is the expression in the square brackets in the last line of (6.94):

$$(6.95)\qquad \hat{K} = \frac{2}{3}\left(\bar{K}^{(2)} + \tilde{K}^{(2)}\right) - \frac{1}{3}K^{(2)}$$

$$= \frac{2}{3}\left\{\left[\bar{K}_t^{(2)} + \sum_{q=1}^{Q}\frac{h_q^2}{k^2}\bar{K}_q^{(2)}\right] + \left[\tilde{K}_t^{(2)} + \sum_{q=1}^{Q}\frac{h_q^2}{k^2}\tilde{K}_q^{(2)}\right]\right\} - \frac{1}{3}\left[K_t^{(2)} + \sum_{q=1}^{Q}\frac{h_q^2}{k^2}K_q^{(2)}\right]$$

$$= \frac{1}{3}\left[2\bar{K}_t^{(2)} + 2\tilde{K}_t^{(2)} - K_t^{(2)}\right] + \frac{1}{3}\sum_{q=1}^{Q}\frac{h_q^2}{k^2}\left[2\bar{K}_q^{(2)} + 2\tilde{K}_q^{(2)} - K_q^{(2)}\right].$$

In the derivation of (6.95) it is assumed that the quantities $\bar{K}^{(2s)}$ and $\tilde{K}^{(2s)}$ have the same structure as the expression for $K^{(2s)}$ in (6.74). It is immediately clear that this assumption is true (some additional information will be given in the remaining part of this section). Furthermore $s = 1$ is used to obtain all three quantities involved in the last line of (6.95).

It is quite clear that if $\hat{K} = O(k^2)$ then the Richardson Extrapolation will be a numerical method of order four. Consider now the last row of (6.95). It obvious that it will be quite sufficient to prove that both the terms of the expression in the first square brackets and the terms in the second square brackets of (61) are of order $O(k^2)$, in the latter case for all values of q.

The following equality can easily be obtained from (6.66) by setting $s = 1$ and replacing t with t_n and $\bar{x}$ with $\tilde{x}$:

$$(6.96) \qquad \mathbf{K_t^{(2)}} = \frac{1}{4}\,\frac{1}{2!}\left[\frac{1}{3}\,\frac{\partial^3 c(t_n + 0.5k\,,\tilde{x})}{\partial t^3} + \sum_{q=1}^{Q} u_q(t_n + 0.5k, \tilde{x})\,\frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^2 \partial x_q}\right].$$

Similar expressions for the other quantities occurring in the last line of (6.95); i.e. for the quantities $\bar{K}_t^{(2)}$, $\tilde{K}_t^{(2)}$, $\bar{K}_q^{(2)}$, $\tilde{K}_q^{(2)}$ and $K_q^{(2)}$, are derived below by following very closely the procedure applied in §6.6.2. In fact, $K_q^{(2)}$ can be obtained directly from (37) by setting $s = 2$ and replacing t with t_n, $t + k$ with t_{n+1} and $\bar{x}$ with $\tilde{x}$.

Consider the two points $(t_n + 0.5k, \widetilde{x})$ and $(t_n + 0.25k, \widetilde{x})$. The following two relationships can be written in connection with these points:

$$(6,97) \qquad c(t_n + 0.5k, \tilde{x}) = c(t_n + 0.25k, \tilde{x}) + \frac{k}{4}\,\frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial t} + \sum_{s=2}^{2p-1} \frac{k^s}{4^s\, s!}\,\frac{\partial^s c(t + 0.25k, \tilde{x})}{\partial t^s} + O(k^{2p}) \ ,$$

$$(6.98) \qquad c(t_n\,, \tilde{x}) = c(t_n + 0.25k, \tilde{x}) - \frac{k}{4}\,\frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial t} + \sum_{s=2}^{2p-1} (-1)^s\,\frac{k^s}{4^s\, s!}\,\frac{\partial^s c(t_n + 0.25k, \tilde{x})}{\partial t^s} + O(k^{2p})\,.$$

Eliminate the quantity $c(t_n + 0.25k\,, \tilde{x})$ from (6.97) and (6.98), which can be achieved by subtracting (6.98) from (6.97). The result is:

$$(6.99) \qquad c(t_n + 0.5k, \tilde{x}) - c(t_n, \tilde{x}) = \frac{k}{2}\,\frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial t} + 2\sum_{s=1}^{p-1} \frac{k^{2s+1}}{4^{2s+1}(2s+1)!}\,\frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s+1}} + O(k^{2p}) \ .$$

The last equality can be rewritten as

$$(6.100) \qquad \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial t} = \frac{c(t_n + 0.5k, \tilde{x}) - c(t_n, \tilde{x})}{0.5k}$$

$$- \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s+1)!} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s+1}} + O(k^{2p-1}) \quad .$$

It can easily be seen that (6.100) can be obtained from (6.50) by replacing $t + 0.5k$ with $t_n + 0.25k$, $t + k$ with $t_n + 0.5k$, t with t_n, $\bar{x}$ with $\tilde{x}$ and 2^{2s} with 4^{2s}, which is quite understandable.

Consider the following two relationships:

$$(6.101) \qquad \frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} = \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial x_q} + \sum_{s=1}^{2p-1} \frac{k^s}{4^s s!} \frac{\partial^{s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^s \partial x_q} + O(k^{2p}) ,$$

$$(6.102) \qquad \frac{\partial c(t_n, \tilde{x})}{\partial x_q} = \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial x_q} + \sum_{s=1}^{2p-1} (-1)^s \frac{k^s}{4^s s!} \frac{\partial^{s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^s \partial x_q} + O(k^{2p}) \quad .$$

Add (6.101) to (6.102). The result is:

$$(6.103) \qquad \frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} + \frac{\partial c(t_n, \tilde{x})}{\partial x_q} = 2 \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial x_q} + 2 \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s)!} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s} \partial x_q} + O(k^{2p}) .$$

The last equality can be rewritten as:

$$(6.104) \qquad \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial x_q} = \frac{1}{2} \left[\frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} + \frac{\partial c(t_n, \tilde{x})}{\partial x_q} \right] - \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s)!} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s} \partial x_q} + O(k^{2p}) .$$

Note that (6.104) can be obtained from (6.57) in a similar way as (6.100) can be obtained from (6.53), i.e., by replacing $t + k$ with $t_n + 0.5k$, $t + 0.5k$ with $t_n + 0.25k$, $\bar{x}$ with $\tilde{x}$ and 2^{2s} with 4^{2s}, which is again quite understandable.

Now the following four relationships can be written:

$$(6.105)\quad \mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5\mathbf{q})}\right)=\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})+\frac{\mathbf{h_q}}{\mathbf{2}}\frac{\partial\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}+\sum_{\mathbf{s=2}}^{\mathbf{2p-1}}\frac{\mathbf{h_q^s}}{\mathbf{2^s\,s!}}\frac{\partial^{\mathbf{s}}\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q^s}}+\mathbf{O}\left(\mathbf{h_q^{2p}}\right),$$

$$(6.106)\quad \mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5\mathbf{q})}\right)=\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})-\frac{\mathbf{h_q}}{\mathbf{2}}\frac{\partial\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}+\sum_{\mathbf{s=2}}^{\mathbf{2p-1}}(-\mathbf{1})^{\mathbf{s}}\frac{\mathbf{h_q^s}}{\mathbf{2^s\,s!}}\frac{\partial^{\mathbf{s}}\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q^s}}+\mathbf{O}\left(\mathbf{h_q^{2p}}\right),$$

$$(6.107)\quad \mathbf{c}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+0.5\mathbf{q})}\right)=\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})+\frac{\mathbf{h_q}}{\mathbf{2}}\frac{\partial\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}+\sum_{\mathbf{s=2}}^{\mathbf{2p-1}}\frac{\mathbf{h_q^s}}{\mathbf{2^s\,s!}}\frac{\partial^{\mathbf{s}}\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\partial\mathbf{x_q^s}}+\mathbf{O}\left(\mathbf{h_q^{2p}}\right),$$

$$(6.108)\quad \mathbf{c}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-0.5\mathbf{q})}\right)=\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})-\frac{\mathbf{h_q}}{\mathbf{2}}\frac{\partial\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}+\sum_{\mathbf{s=2}}^{\mathbf{2p-1}}(-\mathbf{1})^{\mathbf{s}}\frac{\mathbf{h_q^s}}{\mathbf{2^s\,s!}}\frac{\partial^{\mathbf{s}}\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\partial\mathbf{x_q^s}}+\mathbf{O}\left(\mathbf{h_q^{2p}}\right).$$

Subtract (6.106) from (6.105) to obtain:

$$(6.109)\quad \frac{\partial\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}=\frac{\mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(+0.5\mathbf{q})}\right)-\mathbf{c}\left(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}}^{(-0.5\mathbf{q})}\right)}{\mathbf{2(0.5h_q)}}-\sum_{\mathbf{s=1}}^{\mathbf{p-1}}\frac{\mathbf{h_q^{2s}}}{\mathbf{2^{2s}(2s+1)!}}\frac{\partial^{\mathbf{2s+1}}\mathbf{c}(\mathbf{t_n}+\mathbf{0.5k},\tilde{\mathbf{x}})}{\partial\mathbf{x_q^{2s+1}}}+\mathbf{O}\left(\mathbf{h_q^{2p-1}}\right).$$

Similarly, the following relationship can be obtained by subtracting (6.108) from (6.107):

$$(6.110)\quad \frac{\partial\mathbf{c}(\mathbf{t_n},\tilde{\mathbf{x}})}{\partial\mathbf{x_q}}=\frac{\mathbf{c}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(+0.5\mathbf{q})}\right)-\mathbf{c}\left(\mathbf{t_n},\tilde{\mathbf{x}}^{(-0.5\mathbf{q})}\right)}{\mathbf{2(0.5h_q)}}$$

$$- \sum_{s=1}^{p-1} \frac{h_q^{2s}}{2^{2s}(2s+1)!} \frac{\partial^{2s+1} c(t_n, \tilde{x})}{\partial x_q^{2s+1}} + O\left(h_q^{2p-1}\right).$$

It can easily be seen how (6.109) and (6.110) can be obtained from (6.62) and (6.63) respectively: it is necessary to replace $t + k$ with $t_n + 0.5k$, t with t_n, $\bar{x}$ with $\tilde{x}$, $(+q)$ with $(+0.5q)$, $(-q)$ with $(-0.5q)$, h_q with $0.5h_q$ and, finally, 1 with 2^{2s} in the denominators of the sums).

Consider formula (6.43). Replace t with $t_n + 0.25k$ and $\bar{x}$ with $\tilde{x}$. The result is:

$$(6.111) \qquad \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial t} = - \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \frac{\partial c(t_n + 0.25k, \tilde{x})}{\partial x_q}.$$

Use (6.100) and (6.104) in (6.11) to obtain:

$$(6.112) \qquad \frac{c(t_n + 0.5k, \tilde{x}) - c(t_n, \tilde{x})}{0.5k}$$

$$= - \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \left\{ \frac{1}{2} \left[\frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} + \frac{\partial c(t_n, \tilde{x})}{\partial x_q} \right] \right\}$$

$$+ \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s+1)!} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s+1}}$$

$$+ \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s)!} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s} \partial x_q} + O(k^{2p-1}).$$

The last equality can be rewritten as

$$(6.113) \qquad \frac{c(t_n + 0.5k, \tilde{x}) - c(t_n, \tilde{x})}{0.5k}$$

$$= -\sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \left\{ \frac{1}{2} \left[\frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} + \frac{\partial c(t_n, \tilde{x})}{\partial x_q} \right] \right\}$$

$$+ \sum_{s=1}^{p-1} \frac{k^{2s}}{4^{2s}(2s)!} \left\{ \frac{1}{2s+1} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s+1}} \right.$$

$$\left. + \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s} \partial x_q} \right\} + O(k^{2p-1}) .$$

Denote:

$$(6.114) \quad \bar{K}_t^{(2s)}$$

$$= \frac{1}{4^{2s}(2s)!} \left\{ \frac{1}{2s+1} \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s+1}} \right.$$

$$\left. + \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \frac{\partial^{2s+1} c(t_n + 0.25k, \tilde{x})}{\partial t^{2s} \partial x_q} \right\}.$$

Note that (6.114) can be obtained from (6.66) by performing similar replacements as those made in connection with formulae (6.100) and (6.104).

Then (6.113) can be rewritten as:

$$(6.115) \quad \frac{c(t_n + 0.5k, \tilde{x}) - c(t_n, \tilde{x})}{0.5k}$$

$$= -\sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \left\{ \frac{1}{2} \left[\frac{\partial c(t_n + 0.5k, \tilde{x})}{\partial x_q} + \frac{\partial c(t_n, \tilde{x})}{\partial x_q} \right] \right\}$$

$$+ \sum_{s=1}^{p-1} k^{2s} \bar{K}_t^{(2s)} + O(k^{2p-1}) .$$

Use (6.109) and (6.110) in the expression in the square bracket from (6.115) to obtain

$$(6.116)\quad \frac{c(t_n+0.5k,\tilde{x})-c(t_n,\tilde{x})}{0.5k}=-\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\frac{c\left(t_n,\tilde{x}^{(+0.5q)}\right)-c\left(t_n,\tilde{x}^{(-0.5q)}\right)}{4\left(0.5h_q\right)}$$

$$-\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\frac{c\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-c\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4(0.5h_q)}$$

$$+\frac{1}{2}\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\sum_{s=1}^{p-1}\frac{h_q^{2s}}{2^{2s}(2s+1)!}\frac{\partial^{2s+1}c(t_n+0.5k,\tilde{x})}{\partial x_q^{2s+1}}$$

$$+\frac{1}{2}\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\sum_{s=1}^{p-1}\frac{h_q^{2s}}{2^{2s}(2s+1)!}\frac{\partial^{2s+1}c(t_n,\tilde{x})}{\partial x_q^{2s+1}}$$

$$+\sum_{s=1}^{p-1}k^{2s}\bar{K}_t^{(2s)}+O(k^{2p-1}).$$

The last equality can be rewritten in the following form:

$$(6.117)\quad \frac{c(t_n+0.5k,\tilde{x})-c(t_n,\tilde{x})}{0.5k}=-\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\frac{c\left(t_n,\tilde{x}^{(+0.5q)}\right)-c\left(t_n,\tilde{x}^{(-0.5q)}\right)}{4(0.5h_q)}$$

$$-\sum_{q=1}^{Q}u_q(t_n+0.25k,\tilde{x})\frac{c\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-c\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4(0.5h_q)}$$

$$+\sum_{s=1}^{p-1}\left\{\sum_{q=1}^{Q}\frac{h_q^{2s}}{2^{2s+1}(2s+1)!}u_q(t_n+0.25k,\tilde{x})\left[\frac{\partial^{2s+1}c(t_n+0.5k,\tilde{x})}{\partial x_q^{2s+1}}+\frac{\partial^{2s+1}c(t_n,\tilde{x})}{\partial x_q^{2s+1}}\right]\right\}$$

$$+\sum_{s=1}^{p-1}k^{2s}\bar{K}_t^{(2s)}+O(k^{2p-1}).$$

Denote:

$$(6.118)\quad \bar{K}_q^{(2s)} = \sum_{q=1}^{Q} \frac{1}{2^{2s+1}(2s+1)!} u_q(t+0.25k,\tilde{x}) \left[\frac{\partial^{2s+1} c(t_n+0.5k,\tilde{x})}{\partial x_q^{2s+1}} + \frac{\partial^{2s+1} c(t_n,\tilde{x})}{\partial x_q^{2s+1}} \right].$$

Substitute this value of $\bar{K}_q^{(2s)}$ in (6.117). The result is:

$$(6.119)\quad \frac{c(t_n+0.5k,\tilde{x})-c(t_n,\tilde{x})}{0.5k} = -\sum_{q=1}^{Q} u_q(t_n+0.25k,\tilde{x}) \frac{c\left(t_n,\tilde{x}^{(+0.5q)}\right)-c\left(t_n,\tilde{x}^{(-0.5q)}\right)}{4(0.5h_q)}$$

$$-\sum_{q=1}^{Q} u_q(t_n+0.25k,\tilde{x}) \frac{c\left(t_n+0.5k,\tilde{x}^{(+0.5q)}\right)-c\left(t_n+0.5k,\tilde{x}^{(-0.5q)}\right)}{4(0.5h_q)}$$

$$+\sum_{s=1}^{p-1} \left(k^{2s} \bar{K}_t^{(2s)} + \sum_{q=1}^{Q} h_q^{2s} \bar{K}_q^{(2s)} \right) + O(k^{2p-1}).$$

Performing similar transformations around the point $t_n+0.75k$ the following two equalities can be obtained:

$$(6.120)\quad \tilde{K}_t^{(2s)} = \frac{1}{4^{2s}(2s)!} \left\{ \frac{1}{s+1} \frac{\partial^{2s+1} c(t_n+0.75k,\tilde{x})}{\partial t^{2s+1}} + \sum_{q=1}^{Q} u_q(t+0.75k,\tilde{x}) \frac{\partial^{2s+1} c(t_n+0.75k,\tilde{x})}{\partial t^{2s} \partial x_q} \right\}$$

$$(6.121)\quad \tilde{K}_q^{(2s)} = \frac{1}{2^{2s+1}(2s+1)!} u_q(t+0.75k,\tilde{x}) \left[\frac{\partial^{2s+1} c(t_n+0.5k,\tilde{x})}{\partial x_q^{2s+1}} + \frac{\partial^{2s+1} c(t_{n+1},\tilde{x})}{\partial x_q^{2s+1}} \right].$$

Consider now the expression $2\bar{K}_t^{(2)} + 2\tilde{K}_t^{(2)} - K_t^{(2)}$ in (6.95).

$$(6.122)\quad 2\bar{K}_t^{(2)} + 2\tilde{K}_t^{(2)} - K_t^{(2)} = \frac{1}{24} \left[0.5 \frac{\partial^3 c(t_n+0.25k,\tilde{x})}{\partial t^3} + 0.5 \frac{\partial^3 c(t_n+0.75k,\tilde{x})}{\partial t^3} - \frac{\partial^3 c(t_n+0.5k,\tilde{x})}{\partial t^3} \right]$$

$$+\frac{1}{16}\sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x})\frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial t^2 \partial x_q}$$
$$+\frac{1}{16}\sum_{q=1}^{Q} u_q(t_n + 0.75k, \tilde{x})\frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial t^2 \partial x_q}$$
$$-\frac{1}{8}\sum_{q=1}^{Q} u_q(t_n + 0.5k, \tilde{x})\frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^2 \partial x_q}.$$

Since

$$(6.123)\quad \frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial t^3} + \frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial t^3} = 2\frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^3} + O(k^2),$$

it is clear that the expression in the square brackets in (6.122) is of order $O(k^2)$.

Consider now the following relations:

$$(6.124)\quad u_q(t_n + 0.75k, \tilde{x}) = u_q(t_n + 0.5k, \tilde{x}) + 0.25k\frac{\partial u_q(t_n + 0.5k, \tilde{x})}{\partial t} + O(k^2),$$

$$(6.125)\quad u_q(t_n + 0.25k, \tilde{x}) = u_q(t_n + 0.5k, \tilde{x}) - 0.25k\frac{\partial u_q(t_n + 0.5k, \tilde{x})}{\partial t} + O(k^2),$$

$$(6.126)\quad \frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial t^2 \partial x_q} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^2 \partial x_q} + 0.25k\frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t^3 \partial x_q} + O(k^2),$$

$$(6.127)\quad \frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial t^2 \partial x_q} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^2 \partial x_q} - 0.25k\frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t^3 \partial x_q} + O(k^2).$$

Denote

$$(6.128)\quad A = u_q(t_n + 0.5k, \tilde{x}), \qquad B = 0.25\frac{\partial u_q(t_n + 0.5k, \tilde{x})}{\partial t},$$

$$(6.129) \quad \mathbf{C} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial t^2 \partial x_q}, \quad \mathbf{D} = 0.25 \frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t^3 \partial x_q}.$$

Then for an arbitrary value of $\mathbf{q}$ it is possible to obtain from the last three terms of (1.22) the following relationship by omitting always the terms containing the multiplier $\mathbf{k^2}$:

$$(6.130) \quad \frac{1}{16}(A - kB)(C - kD) + \frac{1}{16}(A + kB)(C + kD) - \frac{1}{8}AC$$

$$= \frac{1}{16}(AC - kAD - kBC) + \frac{1}{16}(AC + kAD + kBC) - \frac{1}{8}AC = 0.$$

This shows that the sum of the last three terms of (6.122) is also of order $\mathbf{k^2}$.

Consider now the expression $2\bar{K}_q^{(2)} + 2\tilde{K}_q^{(2)} - K_q^{(2)}$ in the last line of (6.95). The following three equalities can be obtained by using (6.71), (6.118) and (6.123) with $s = 1$:

$$(6.131) \quad K_q^{(2)} = \frac{1}{12} u_q(t_n + 0.5k, \tilde{x}) \left[\frac{\partial^3 c(t_n + k, \tilde{x})}{\partial x_q^3} + \frac{\partial^3 c(t_n, \tilde{x})}{\partial x_q^3} \right]$$

$$= \frac{1}{6} u_q(t_n + 0.5k, \tilde{x}) \left[\frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} + O(k^2) \right],$$

$$(6.132) \quad \bar{K}_q^{(2)} = \frac{1}{48} u_q(t_n + 0.25k, \tilde{x}) \left[\frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} + \frac{\partial^3 c(t_n, \tilde{x})}{\partial x_q^3} \right]$$

$$= \frac{1}{24} u_q(t_n + 0.25k, \tilde{x}) \left[\frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial x_q^3} + O(k^2) \right]$$

$$(6.133) \quad \tilde{K}_q^{(2)} = \frac{1}{48} u_q(t_n + 0.75k, \tilde{x}) \left[\frac{\partial^3 c(t_n + k, \tilde{x})}{\partial x_q^3} + \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} \right]$$

$$= \frac{1}{24} u_q(t_n + 0.75k, \tilde{x}) \left[\frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial x_q^3} + O(k^2) \right].$$

The following two relationships are used in the derivation of (6.131):

$$\frac{\partial^3 c(t_n + k, \tilde{x})}{\partial x_q^3} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} + 0.5k \frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t \partial x_q^3} + O(k^2) , \quad (6.134)$$

$$\frac{\partial^3 c(t_n, \tilde{x})}{\partial x_q^3} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} - 0.5k \frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t \partial x_q^3} + O(k^2) . \quad (6.135)$$

Now (6.134) and (6.135) should be added to obtain:

$$\frac{\partial^3 c(t_n + k, \tilde{x})}{\partial x_q^3} + \frac{\partial^3 c(t_n, \tilde{x})}{\partial x_q^3} = 2 \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} + O(k^2) . \quad (6.136)$$

Equality (6.136) shows how (1.31) can be obtained. Similar operations are to be used to obtain (6.132) and (6.133).

By using (6.131), (6.132) and (6.133) and by omitting terms containing $O(k^2)$ it is possible to transform the expression $2\bar{K}_q^{(2)} + 2\tilde{K}_q^{(2)} - K_q^{(2)}$ in the following way:

$$\begin{aligned} 2\bar{K}_q^{(2)} + 2\tilde{K}_q^{(2)} - K_q^{(2)} = & \frac{1}{12} \sum_{q=1}^{Q} u_q(t_n + 0.25k, \tilde{x}) \frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial x_q^3} \\ & + \frac{1}{12} \sum_{q=1}^{Q} u_q(t_n + 0.75k, \tilde{x}) \frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial x_q^3} \\ & - \frac{1}{6} \sum_{q=1}^{Q} u_q(t_n + 0.5k, \tilde{x}) \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} . \end{aligned} \quad (6.137)$$

It is now necessary to apply (6.124) and (6.125) together with the following two equalities:

$$\frac{\partial^3 c(t_n + 0.75k, \tilde{x})}{\partial x_q^3} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} + 0.25k \frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t \partial x_q^3} + O(k^2) . \quad (6.138)$$

$$\frac{\partial^3 c(t_n + 0.25k, \tilde{x})}{\partial x_q^3} = \frac{\partial^3 c(t_n + 0.5k, \tilde{x})}{\partial x_q^3} - 0.25k \frac{\partial^4 c(t_n + 0.5k, \tilde{x})}{\partial t \partial x_q^3} + O(k^2) . \quad (6.139)$$

Denote:

$$(6.140) \qquad \mathbf{E} = \frac{\partial^3 \mathbf{c}(\mathbf{t_n} + \mathbf{0.5k}, \tilde{\mathbf{x}})}{\partial \mathbf{x_q^3}}, \qquad \mathbf{F} = \mathbf{0.25}\ \frac{\partial^4 \mathbf{c}(\mathbf{t_n} + \mathbf{0.5k}, \tilde{\mathbf{x}})}{\partial \mathbf{t} \partial \mathbf{x_q^3}}.$$

Then for an arbitrary value of $\mathbf{q}$ it is possible to obtain the following relationship from (6.137) by omitting always the terms containing multiplier $\mathbf{k^2}$:

$$(6.141) \quad \frac{\mathbf{1}}{\mathbf{12}}(\mathbf{A} - \mathbf{kB})(\mathbf{E} - \mathbf{kF}) + \frac{\mathbf{1}}{\mathbf{12}}(\mathbf{A} + \mathbf{kB})(\mathbf{E} + \mathbf{kF}) - \frac{\mathbf{1}}{\mathbf{6}}\mathbf{AE}$$

$$= \frac{\mathbf{1}}{\mathbf{12}}(\mathbf{AE} - \mathbf{kAF} - \mathbf{kBE}) + \frac{\mathbf{1}}{\mathbf{12}}(\mathbf{AE} + \mathbf{kAF} + \mathbf{kBE}) - \frac{\mathbf{1}}{\mathbf{6}}\mathbf{AE} = \mathbf{0}.$$

Therefore, the expression in (6.137) is of order $\mathbf{k^2}$. This means that the expression $\widehat{\mathbf{K}}$ in (6.95) is of order $\mathbf{O(k^2)}$ and it can be concluded that if the multi-dimensional advection is treated by the second-order numerical method defined by (6.86) and combined with the Richardson Extrapolation, then it results in a numerical method of order four (not of order three as should be expected). In this way the following theorem has been proved:

Theorem 6.4: Consider the multi-dimensional advection equation (6.43). Assume that the coefficients $\mathbf{u_q}$ before the spatial derivatives in (6.43) are continuously differentiable up to order two with respect to all independent variables and continuous derivatives of the unknown function **c(x,t)** up to order four exist, again with respect to all variables. Then the combination of the numerical method (6.86) and the Richardson Extrapolation is of order four.

■

Remark 6.3: It is clear that Theorem 6.2 is a special case of Theorem 6.4, which can be obtained by setting $\mathbf{q = 1}$ in (6.43).

■

6.7. General conclusions related to the sixth chapter

The simple one-dimensional advection problem was considered in the first five sections of this chapter. This case was described in detail in an attempt to make the main ideas very clear. The hope is that if

this is done, then it will be easy to generalize the results. As an illustration, the generalization of the results, obtained for the one-dimensional advection problem, for the multi-dimensional advection problem is presented in Section 6.6. It is worthwhile to generalize further the results for other classes of partial differential equations.

It is proved that for the advection problem treated by the second-order Crank-Nicolson scheme the application of the Richardson Extrapolation leads to a rather accurate new numerical method, the order of accuracy is increased from two to four. It should be emphasized, however, that normally the order of accuracy is increased by one, i.e. if the underlying method is of order $\mathbf{p}$, then its combination with the Richardson Extrapolation is of order $\mathbf{p+1}$.

As mentioned in the beginning of this chapter, the most straight-forward way of implementing the Richardson Extrapolation for partial differential equations or for systems of partial differential equations is based on a preliminary application of some kind of discretization of the spatial derivatives. In this way the partial differential equations (or the system of partial differential equations) is transformed to a system (normally very large) of ordinary differential equations. Then the Richardson Extrapolation can be applied to the system of ODEs as in the first four chapters of this book. However, one must be aware of the fact that the errors arising during the discretization of the spatial derivatives must somehow be taken into account, especially in the case where the Richardson Extrapolation will be used to check the error made during the computations.

Stability problems were not discussed in Chapter 6. However, it must be emphasized here that the stability of the computational process is a very important issue also in the case where the Richardson Extrapolation is used together with partial differential equations (or systems of partial differential equations).

6.8. Topics for further research

The following topics might lead to some very interesting and useful results:

(A) It was mentioned in the previous section that the stability of the computational process should also be studied. It is worthwhile to try to establish some stability result in connection with the use of the Richardson Extrapolation for the advection problems discussed in the first six sections of this chapter. One can start perhaps with the simple one-dimensional advection.

(B) The accuracy of the Richardson Extrapolation applied together with the simple problem (6.43) was studied in Section 4.6. The same results can probably be proved for the slightly more complex and more realistic problem:

$$\frac{\partial c}{\partial t} = -\sum_{q=1}^{Q} \frac{\partial (u_q c)}{\partial x_q}.$$

Even results for the case of the one-dimensional advection problem, obtained for $\mathbf{q = 1}$, might be useful.

(C) The application of the Richardson Extrapolation in conjunction with other problems (e.g. parabolic partial differential equations or some systems of parabolic partial differential equations) may lead to some useful results.

(D) Some comprehensive studies of the application of the Richardson Extrapolation in conjunction with some important particular problems that occur often in practice (as, for example the famous rotation test, see, for example, **Chock, 1985, 1991**, **Chock and Dunker, 1983**, **Crowley, 1968**, **LeVeque, 1992**, **Molenkampf, 1968**, **Zlatev, Berkowicz and Prahm, 1983**) are also desirable.

Chapter 7

Richardson Extrapolation for some other problems

Systems of ordinary differential equations as well as some special partial differential equations were handled in the previous chapters by applying the Richardson Extrapolation in order both to improve the accuracy of the numerical solution and to increase the efficiency of the computational process (or at least to achieve one of these two important aims). However, the Richardson Extrapolation can also be used in the solution of many other problems as, for example, in the solution of systems of algebraic and transcendental equations as well as in numerical integration.

In this chapter we shall first assume that a system of algebraic and transcendental equations is written in the form:

$$\mathbf{f(u) = 0}, \qquad \mathbf{u \in D \subset \mathbb{R}^s}, \qquad \mathbf{s \geq 1}\,. \tag{7.1}$$

We shall furthermore assume that (7.1) has at least one solution $\mathbf{u}^*$ and that some iterative method is used to calculate a sufficiently accurate approximation of this solution. The application of the selected iterative method leads, when it is convergent, to the calculation of a sequence of approximations to the solution $\mathbf{u}^*$ given by

$$\mathbf{u_1},\ \mathbf{u_2},\ \ldots,\ \mathbf{u_n},\ \ldots \quad \text{where} \quad \mathbf{n \rightarrow \infty} \quad \text{and} \quad \lim_{\mathbf{n \rightarrow \infty}}\{\mathbf{u_n}\} = \mathbf{u}^*\,. \tag{7.2}$$

If the iterative process is convergent, then the following relationship will clearly hold:

$$\lim_{\mathbf{n \rightarrow \infty}}\{\|\mathbf{u_n} - \mathbf{u}^*\|\} = \mathbf{0}\,, \tag{7.3}$$

where $\|*\|$ is some appropriately chosen vector norm in $\mathbb{R}^s$.

In all practical situations, one is primarily interested in stopping the iterative process when certain prescribed in advance accuracy, say ε, where ε is some small positive number, is achieved. This means that we should like to stop the iterative process after n iterations if, for example, the following criterion is satisfied:

$$(7.4) \qquad \|u_n - u^*\| \leq \varepsilon .$$

The great problem is that (7.4) cannot be directly applied, because the exact solution u^* is unknown. Therefore different stopping criteria must be carefully designed and consistently used. These criteria will, of course, depend on the particular method selected for the solution of (7.1). Some stopping criteria will be introduced for several particular methods in the remaining part of this chapter. We are primarily interested in the answer to the following two general questions:

If an arbitrary numerical method is selected for the computer treatment of (7.1), then will it be possible to design some general device, similar to the Richardson Extrapolation, the application of which will lead to an acceleration of the speed of convergence?

Moreover, will it be possible to apply precisely the Richardson Extrapolation?

The above problems were studied in many papers and books; see, for example, **Brezinski (1985, 2000)**, **Brezinski and Matos (1996)**, **Burden and Faires (2001)**, **Dutka (1984)**, **Joyce (1971)**, **Osada (1993)** or **Waltz (1996)**. The solutions of some of the problems were **discovered and re-discovered** several times (see the above references again).

Consider the sequence (7.2) with $s = 1$ and assume that $\lim_{n \to \infty} \{u_n\} = u^*$. Suppose that another sequence $\{\tilde{u}_n\}_{n=1}^{\infty}$ is created by using some expressions containing elements of the first sequence $\{u_n\}_{n=1}^{\infty}$. It is said that the second sequence is **accelerating** the convergence of the first one **if and only if** the following relationship holds: $\lim_{n \to \infty} \{(\tilde{u}_n - u^*)/(u_n - u^*)\} = 0$; see **Brezinski (2000)**. Several transformations leading to an acceleration of the rate of convergence of sequences will be considered in this chapter. It will be assumed, as we have already done above, that $s = 1$, i.e. we shall consider **sequences of real numbers**, but most of the ideas discussed in this chapter are also applicable for more general cases.

It must be noted here that equations of the type (7.1) appear when systems of ordinary differential equations are handled by using implicit numerical methods (see Chapter 4) as well as when the spatial

derivatives of a system of partial differential equations are discretized and the resulting system of ordinary differential equations is handled by implicit methods. That means that efficient methods for the numerical treatment of equations of type (7.1) might be useful also in connection with the material presented in the previous chapters.

The applications of iterative methods in the solution of (7.1) leads to sequences of real vectors (7.2), but we shall often assume that the treated problems are scalar and will consider the task of accelerating the rate of convergence of the resulting sequences of real numbers.

The contents of the seventh chapter can be sketched as follows:

Some very simple examples, describing how the speed of convergence can be accelerated, will be given in the first section, **Section 7.1**.

Some information about the use of Romberg methods in the numerical integration will be presented, together with some numerical results, in **Section 7.2**.

Several major conclusions will be drawn in **Section 7.3**, while several topics for further research will be presented in **Section 7.4**.

7.1. Acceleration of the speed of convergence for sequences of real numbers

In this section we shall mainly consider sequences of real numbers, i.e. we shall again use (7.1) under the assumption that $\mathbf{s = 1}$ has been selected.

We shall start, in **Sub-section 7.1.1**, with a simple example related to the calculation of approximations of the famous number $\boldsymbol{\pi}$. The algorithm sketched in this sub-section was used (several hundred years ago) by the Japanese mathematician **Takakazu Seki (1642-1708)**, see **Hirayama, Shimodaira and Hirose (1974)**.

Some definitions that are related to the rate of convergence of the selected iterative method in the case where $\mathbf{s = 1}$ will be introduced and discussed, together with some simple examples, in **Sub-section 7.1.2**.

Then we shall proceed, in **Sub-section 7.1.3**, with the introduction of the algorithm proposed by A. Aitken and an improvement of this algorithm suggested by J. F Steffensen; see **Aitken (1926)** and **Steffensen (1950)**.

Some short remarks about the use of Richardson Extrapolation in connection with sequences of real numbers will be given in **Sub-section 7.1.4**.

7.1.1. Improving the accuracy in calculations related to the number $\boldsymbol{\pi}$

Takakazu Seki, see again **Hirayama, Shimodaira and Hirose (1974)**, defined $\mathbf{u_n}$ as the perimeter of the regular polygon with $\mathbf{2^n}$ sides that is inscribed in a circle with a diameter $\mathbf{d = 1}$ and calculated three consecutive approximations of the number $\boldsymbol{\pi}$:

$$(7.5) \qquad \mathbf{u_{15}} = \mathbf{3.141592}6487769856708, \quad \mathbf{u_{16}} = \mathbf{3.141592}6523865913571,$$
$$\mathbf{u_{17}} = \mathbf{3.141592}6532889927750.$$

Then he applied the formula:

$$(7.6) \qquad \tilde{\mathbf{u}}_{\mathbf{15}} = \mathbf{u_{16}} + \frac{(\mathbf{u_{16}} - \mathbf{u_{15}})(\mathbf{u_{17}} - \mathbf{u_{16}})}{(\mathbf{u_{16}} - \mathbf{u_{15}}) - (\mathbf{u_{17}} - \mathbf{u_{16}})},$$

to calculate an improved approximation:

$$(7.7) \qquad \tilde{\mathbf{u}}_{\mathbf{15}} = \mathbf{3.14159265358}97932476.$$

Twelve digits in the approximation $\tilde{\mathbf{u}}_{\mathbf{15}}$ are correct, while the number of the correct digits in $\mathbf{u_{15}}$, $\mathbf{u_{16}}$ and $\mathbf{u_{17}}$ are **seven**, **eight** and **nine** respectively; the correct digits in (7.5) and (7.7) are marked in green. This illustrates very clearly the fact that the computational devise introduced by formula (7.6) leads to a very considerable increase of the accuracy.

7.1.2. Definitions related to the convergence rate of some iterative processes

Consider (7.4) with $\mathbf{s = 1}$ and introduce the abbreviation $\mathbf{e_n} = |\mathbf{u_n} - \mathbf{u}^*|$. It is clear that $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_n}\} = \mathbf{0}$ when the iterative process is convergent. Then, the following definitions related to the concept of convergence for sequences of real numbers can be introduced:

Definition 7.1: We say that the convergence of (7.4) with $\mathbf{s = 1}$ is **linear** if there exists some real number $\mathbf{K} \in (\mathbf{0}, \mathbf{1})$ for which the equality $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_{n+1}}/\mathbf{e_n}\} = \mathbf{K}$ holds. If $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_{n+1}}/\mathbf{e_n}\} = \mathbf{1}$ or $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_{n+1}}/\mathbf{e_n}\} = \mathbf{0}$ is satisfied, then the convergence is **sub-linear** or **super-linear**, respectively. If both $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_{n+1}}/\mathbf{e_n}\} = \mathbf{1}$ and $\lim_{\mathbf{n}\to\infty}\{|\mathbf{u_{n+2}} - \mathbf{u_{n+1}}|/|\mathbf{u_{n+1}} - \mathbf{u_n}|\} = \mathbf{1}$ hold, then it is said that the sequence (7.4) with $\mathbf{s = 1}$ is converging **logarithmically** to $\mathbf{u}^*$. The rate of convergence will be of **order p**, where $\mathbf{p \geq 1}$ is an integer, when the relationship $\lim_{\mathbf{n}\to\infty}\{\mathbf{e_{n+1}}/(\mathbf{e_n})^{\mathbf{p}}\} = \mathbf{K}$ is satisfied.

■

In the case of linear convergence the error $\mathbf{e_{n+1}}$ will become approximately $\mathbf{K}$ times smaller than the error $\mathbf{e_n}$ when $\mathbf{n}$ is sufficiently large. The sequence (7.2) with $\mathbf{s = 1}$ is similar to a geometric sequence with a common ratio $\mathbf{K}$ when it converges linearly. It is clear that the convergence becomes slow when $\mathbf{K}$ is approaching $\mathbf{1}$. The sub-linear convergence may be very slow.

Several examples are given below.

Example 7.1: Consider a smooth function $\mathbf{f}$: $[\mathbf{a}, \mathbf{b}] \rightarrow [\mathbf{a}, \mathbf{b}]$ with $\mathbf{M} \stackrel{\text{def}}{=} \max\limits_{\mathbf{u} \in [\mathbf{a},\mathbf{b}]} \{|\mathbf{f}'|\} < \mathbf{1}$. Then according to Banach's fixed point theorem, see **Banach (1922)**, this function has a unique fixed point $\mathbf{u}^*$, which can be found by applying a simple iterative process $\mathbf{u_{n+1}} = \mathbf{f}(\mathbf{u_n})$ for $\mathbf{n} = \mathbf{1}, \mathbf{2}, \ldots$, starting with an arbitrary real number $\mathbf{u_0} \in [\mathbf{a}, \mathbf{b}]$. It can be shown that the convergence rate of this iterative process is linear.

The following relationship, based on the application of a truncated Taylor expansion, obviously holds for some given $\boldsymbol{\zeta} = \boldsymbol{\zeta}(\mathbf{u}) \in [\mathbf{a}, \mathbf{u}]$**:**

$$(\mathbf{7.8}) \qquad \mathbf{f}(\mathbf{u}) = \mathbf{f}(\mathbf{u}^*) + \mathbf{f}'(\mathbf{u}^*)(\mathbf{u} - \mathbf{u}^*) + \frac{\mathbf{f}''(\boldsymbol{\zeta})}{\mathbf{2}!}(\mathbf{u} - \mathbf{u}^*)^{\mathbf{2}} \quad .$$

By substituting $\mathbf{u}$ with $\mathbf{u_n}$ we can obtain:

$$(\mathbf{7.9}) \qquad \mathbf{f}(\mathbf{u_n}) = \mathbf{f}(\mathbf{u}^*) + \mathbf{f}'(\mathbf{u}^*)(\mathbf{u_n} - \mathbf{u}^*) + \frac{\mathbf{f}''(\boldsymbol{\zeta})}{\mathbf{2}!}(\mathbf{u_n} - \mathbf{u}^*)^{\mathbf{2}} \quad .$$

Apply here the relations $\mathbf{u_{n+1}} = \mathbf{f}(\mathbf{u_n})$ and $\mathbf{u}^* = \mathbf{f}(\mathbf{u}^*)$ to obtain:

$$(\mathbf{7.10}) \qquad \frac{\mathbf{u_{n+1}} - \mathbf{u}^*}{\mathbf{u_n} - \mathbf{u}^*} = \mathbf{f}'(\mathbf{u}^*) + \frac{\mathbf{f}''(\boldsymbol{\zeta})}{\mathbf{2}!}(\mathbf{u_n} - \mathbf{u}^*) \quad .$$

It is clear that

$$(\mathbf{7.11}) \qquad \lim_{\mathbf{n}\rightarrow\infty}\left\{\frac{\mathbf{u_{n+1}} - \mathbf{u}^*}{\mathbf{u_n} - \mathbf{u}^*}\right\} = \lim_{\mathbf{n}\rightarrow\infty}\left\{\mathbf{f}'(\mathbf{u}^*) + \frac{\mathbf{f}''(\boldsymbol{\zeta})}{\mathbf{2}!}(\mathbf{u_n} - \mathbf{u}^*)\right\} \quad .$$

The second term on the right-hand-side of (7.11) tends to zero, while the first one is constant. Therefore, we have

$$(7.12) \qquad \lim_{n\to\infty}\left\{\frac{u_{n+1}-u^*}{u_n-u^*}\right\} = f'(u^*) \quad .$$

It is clear now that the following relationship can be obtained from (7.12):

$$(7.13) \qquad \lim_{n\to\infty}\left\{\left|\frac{u_{n+1}-u^*}{u_n-u^*}\right|\right\} = |f'(u^*)| \quad .$$

Therefore the convergence is **<u>linear</u>** because $|f'(u^*)| \leq M < 1$.

■

<u>Example 7.2:</u> Consider the sequence defined by $u_n = 1/n$ for $n = 1, 2, \ldots, .$ It is clear that $\lim_{n\to\infty}\{u_n\} = 0$ and $\lim_{n\to\infty}\{u_{n+1}/u_n\} = 1$. This means that the convergence is only sub-linear. Furthermore, it is easy to show that $\lim_{n\to\infty}\{|u_{n+2}-u_{n+1}|/|u_{n+1}-u_n|\} = 1$, which shows that the sequence $u_n = 1/n$ converges **<u>logarithmically</u>**.

■

<u>Example 7.3:</u> Consider Example 7.1 with the additional assumption that $f'(u^*) = 0$. Use the equality:

$$(7.14) \qquad f(u) = f(u^*) + f'(u^*)(u-u^*) + \frac{f''(u^*)}{2!}(u-u^*)^2 + \frac{f'''(\zeta)}{3!}(u-u^*)^3 \quad .$$

Substitute u with u_n , $f(u_n)$ with u_{n+1} and $f(u^*)$ with u^* to obtain:

$$(7.15) \qquad u_{n+1} = u^* + \frac{f''(u^*)}{2!}(u_n-u^*)^2 + \frac{f'''(\zeta)}{3!}(u_n-u^*)^3 \quad .$$

Simple transformations lead to the relations:

$$(7.16) \qquad \lim_{n\to\infty}\left\{\left|\frac{u_{n+1}-u^*}{(u_n-u^*)^2}\right|\right\} = \frac{|f''(u^*)|}{2} \leq \frac{M}{2} \quad ,$$

which show that we have a second-order (quadratic) convergence in this example.

■

7.1.3. The Aitken scheme and the improvement made by Steffensen

Assume again that (7.1) is a scalar equation, i.e. $\mathbf{s = 1}$, and consider the sequence (7.2). Assume furthermore that the convergence is linear, which means, according to Definition 7.1, that the relationship $\lim_{\mathbf{n}\to\infty}\{(\mathbf{u_{n+1}} - \mathbf{u}^*)/(\mathbf{u_n} - \mathbf{u}^*)\} = \mathbf{K}$ holds for some value of $\mathbf{K}$ with $|\mathbf{K}| \in (\mathbf{0}, \mathbf{1})$. This means that we can expect that the two relationships

$$(\mathbf{7.17}) \qquad \frac{\mathbf{u_{n+1}} - \mathbf{u}^*}{\mathbf{u_n} - \mathbf{u}^*} \approx \mathbf{K} \qquad \text{and} \qquad \frac{\mathbf{u_{n+2}} - \mathbf{u}^*}{\mathbf{u_{n+1}} - \mathbf{u}^*} \approx \mathbf{K}$$

hold for sufficiently large values of $\mathbf{n}$, which implies that

$$(\mathbf{7.18}) \qquad \frac{\mathbf{u_{n+1}} - \mathbf{u}^*}{\mathbf{u_n} - \mathbf{u}^*} \approx \frac{\mathbf{u_{n+2}} - \mathbf{u}^*}{\mathbf{u_{n+1}} - \mathbf{u}^*} \qquad \Rightarrow \qquad \mathbf{u}^* \approx \frac{\mathbf{u_n u_{n+2}} - (\mathbf{u_{n+1}})^2}{\mathbf{u_{n+2}} - \mathbf{2u_{n+1}} + \mathbf{u_n}} .$$

Then a new approximation $\tilde{\mathbf{u}}_\mathbf{n}$ can be defined by setting:

$$(\mathbf{7.19}) \qquad \tilde{\mathbf{u}}_\mathbf{n} \stackrel{\text{def}}{=} \frac{\mathbf{u_{n+2}u_n} - (\mathbf{u_{n+1}})^2}{\mathbf{u_{n+2}} - \mathbf{2u_{n+1}} + \mathbf{u_n}} .$$

Let us introduce the operators:

$$(\mathbf{7.20}) \qquad \Delta\mathbf{u_n} = \mathbf{u_{n+1}} - \mathbf{u_n} \qquad \text{and} \qquad \Delta^2\mathbf{u_n} = \Delta(\Delta\mathbf{u_n}) = \mathbf{u_{n+2}} - \mathbf{2u_{n+1}} + \mathbf{u_n} .$$

Then simple transformations lead to the following expression for $\tilde{\mathbf{u}}_\mathbf{n}$:

$$(\mathbf{7.21}) \qquad \tilde{\mathbf{u}}_\mathbf{n} = \mathbf{u_{n+1}} - \frac{(\Delta\mathbf{u_{n+1}})(\Delta\mathbf{u_n})}{\Delta^2\mathbf{u_n}} .$$

The two equalities (7.19) and (7.21) will produce the same results when the calculations are carried out in exact arithmetic, but on computers (7.21) will often produce more accurate and more reliable results, because it is less sensitive to rounding errors.

Two alternative versions of (7.21) can also be introduced and used:

$$(7.22) \quad \tilde{u}_n = u_n - \frac{(\Delta u_n)^2}{\Delta^2 u_n} \qquad \text{and} \qquad \tilde{u}_n = u_{n-1} - \frac{(\Delta u_n)(\Delta u_{n-1})}{\Delta^2 u_n} \ .$$

The algorithm for calculating improved approximations that is based on any of the formulae in (7.21) and (7.22) was proposed by A. Aitken in 1926, see **Aitken (1926)**. The fact that this algorithm is converging faster is clear from the following theorem proved in **Aitken (1926)**:

Theorem 7.1: Assume that the convergence of the sequence (7.2) is linear, i.e. assume that $\lim_{n \to \infty} \{|u^* - u_{n+1}|/|u^* - u_n|\} = K$ with some $K \in (0, 1)$. Then the sequence $\{\tilde{u}_n\}_{n=0}^{\infty}$ defined in (7.22) converges to the same limit u^* and the rate of convergence is super-linear, which means that $\lim_{n \to \infty} \{|u^* - \tilde{u}_{n+1}|/|u^* - \tilde{u}_n|\} = 0$.

■

Remark 7.1: The scheme, proposed by **Takakazu Seki** in the seventeenth century and discussed shortly in the beginning of this chapter, is obviously an Aitken process; compare (7.6) with (7.19).

■

The Aitken scheme can be applied in conjunction with the fixed point methods discussed in Example 7.2. Consider again the equation $u = f(u)$ with $u \in [a, b]$ and with $M \stackrel{\text{def}}{=} \max_{u \in [a,b]} \{f'\} < 1$. Assume that the solution of this equation is u^*. Consider also the sequence obtained by using the recurrent relationship $u_{n+1} = f(u_n)$ for $n = 1, 2, \ldots$, starting with an arbitrary $u_0 \in [a, b]$. Since the sequence $\{u_n\}_{n=0}^{\infty}$ is linearly convergent to the solution u^*, the Aitken process defined by any of the relationships in (7.19), (7.21) and (7.22) will in general accelerate the convergence. However, this will not always be the case. Example 7.3 shows that the convergence of the sequence $\{u_n\}_{n=0}^{\infty}$ to u^* will be quadratic in the case $f(u^*) = 0$. Therefore, the Aitken process will not accelerate the convergence in this case.

Steffensen's method (**Steffensen, 1950**) is a combination of the fixed-point iteration and the Aitken method. The algorithm can be introduced in the following way. Start with an arbitrary real number u_0 and calculate $u_1 = f(u_0)$ and $u_2 = f(u_1)$ by using the Aitken scheme. Then a new and better approximation $\tilde{u}_0$ can be calculated by using the three approximations u_0, u_1 and u_2 and the formula $\tilde{u}_0 = u_0 - (u_1 - u_0)^2/(u_2 - 2u_1 + u_0)$. This new value can then be applied to restart

the Aitken scheme by using the three approximations. More precisely perform the following three actions: **(a)** denote $\mathbf{v_0 = \tilde{u}_0}$ and calculate $\mathbf{v_1 = f(v_0)}$ and $\mathbf{v_2 = f(v_1)}$, **(b)** compute a new Aitken approximation by using: $\mathbf{\tilde{v}_0 = v_0 - (v_1 - v_0)^2/(v_2 - 2v_1 + v_0)}$ and **(c)** set $\mathbf{\tilde{u}_1 = \tilde{v}_0}$. The same process can be continued in a quite similar manner: for $\mathbf{i = 1}$, $\mathbf{2}$, ... perform the three actions: **(A)** denote $\mathbf{v_0 = \tilde{u}_i}$ and calculate $\mathbf{v_1 = f(v_0)}$ and $\mathbf{v_2 = f(v_1)}$, **(B)** compute a new Aitken approximation by using: $\mathbf{\tilde{v}_0 = v_0 - (v_1 - v_0)^2/(v_2 - 2v_1 + v_0)}$ and **(C)** set $\mathbf{\tilde{u}_{i+1} = \tilde{v}_0}$.

The convergence of the numerical algorithm, obtained as sketched above, is in general quadratic, but for some special problems, see the example given below, could be even higher.

It is worthwhile to compare qualitatively the Steffensen method that has been described above with the well-known Newton method (see Chapter 4). Both algorithms lead to a quadratic convergence. Function $\mathbf{f}$ must be three times continuously differentiable in order to achieve the quadratic convergence when the Steffensen method is used, while the existence of a continuous second derivative of this function is sufficient when the Newton method is used. One function evaluation and one derivative calculation per iteration are needed when the Newton method is used, while two functions evaluations and some more complex algebraic computations are used at every iteration when the Steffensen method is used. The fact that no derivative evaluation is needed when the Steffensen method is used can be considered as an advantage of this method.

A simple numerical example is given below.

Example 7.5: Consider the problem of finding the unique fixed point of the trigonometric function $\mathbf{f(u) = \cos(u)}$ in the interval $[\mathbf{0}, \mathbf{1}]$. The problem has been solved by

(a) using directly the fixed-point iteration,

(b) applying the Aitken scheme

and

(c) using the Steffensen method.

The starting approximation was $\mathbf{u_0 = 0.5}$. A reference solution $\mathbf{u^{ref}}$ was calculated by using fixed-point iterations until the difference between two successive approximations became $\mathbf{|u_{i+1} - u_i| \leq 10^{-30}}$. This requirement was satisfied for $\mathbf{i = 174}$. It must be mentioned here that extended (quadruple) precision was used in the calculation on the computer used.

The reference solution found in this way was $\mathbf{u^{ref} = 0.739085133215160641655312087 7.}$

After that the three methods were run until the difference between the current approximation and the reference solution became less (in absolute value) than $\mathbf{10^{-20}}$ or, in other words, until the inequality $\mathbf{|u^{ref} - u_i| < 10^{-20}}$ was satisfied, which happened after $\mathbf{115}$, $\mathbf{52}$ and $\mathbf{4}$ iterations for the fixed-point algorithm, the Aitken scheme and the Steffensen method respectively.

Numerical results are given in **Table 7.1**.

No.	Fixed-point iteration			Aitken algorithm			Steffensen method		
	Calc. values	Accuracy	Rate	Calc. values	Accuracy	Rate	Calc. values	Accuracy	Rate
0	0.500000000000000	2.4E-01	-	0.731385186382581	7.7E-03	-	0.7313851863825817	7.7E-03	-
1	0.877582561890372	3.7E-01	0.633	0.736086691713016	3.0E-03	2.568	0.7390763403695222	8.8E-06	8.8E+02
2	0.639012494165259	2.3E-01	1.583	0.737652871396399	1.4E-03	2.094	0.7390851332036611	1.1E-11	7.6E+05
3	0.802685100682334	1.6E-01	1.458	0.738469220876263	6.2E-04	2.325	0.7390851332151606	2.2E-23	5.8E+11
4	0.694778026788006	1.0E-01	1.517	0.738798065173590	2.9E-04	2.145			
5	0.768195831282016	7.3E-02	1.470	0.738957710941417	1.3E-04	2.253			
6	0.719165445942419	4.9E-02	1.497	0.739026560427970	5.9E-05	2.175			
7	0.752355759421527	3.3E-03	1.477	0.773905880831365	2.6E-05	2.225			
8	0.730081063137823	2.2E-02	1.490	0.739073115644591	1.2E-05	2.191			
9	0.735006309014843	1.5E-02	1.481	0.739079703326153	5.4E-06	2.213			
10	0.735006309014843	1.0E-02	1.483	0.739082662515195	2.5E-06	2.198			
11	0.741826522643245	6.8E-03	1.483	0.739084014267376	1.1E-06	2.208			
12	0.737235725442231	4.5E-03	1.486	0.739084624843225	5.1E-07	2.201			
13	0.740329651878263	3.0E-03	1.484	0.739084902738891	2.3E-07	2.206			
14	0.738246238332233	2.0E-03	1.485	0.773908502857531	1.0E-07	2.203			
15	0.739649962769661	1.4E-03	1.484	0.73908508575306	4.7E-08	2.205			
16	0.738704539356983	9.4E-04	1.485	0.73908511167342	2.2E-08	2.203			
17	0.739341452281210	6.3E-04	1.484	0.73908512344226	9.8E-09	2.204			
18	0.738912449332103	4.2E-04	1.485	0.73908512878014	4.4E-09	2.204			
⋮	⋮	⋮	⋮	⋮	⋮	⋮			
52	0.739085132962136	6.3E-10	1.485	0.739085133215160	9.5E-21	2.203			
⋮	⋮	⋮	⋮						
115	0.739085133215160	9.7E-21	1.485						

Table 7.1

Calculations performed with three numerical algorithms. Extended (quadruple) precision is used (this means that the actual computations are carried out by using $\mathbf{32}$ significant digits, but the calculated approximations are given in the table with $\mathbf{15}$ significant digits). The absolute values of the difference between the calculated approximations and the reference solution are given in the columns under **"Accuracy"**. The ratios between two successive approximations are given in the columns under **"Rate"**. The calculations were carried out until the absolute value of the calculated approximation and the reference solution becomes less than $\mathbf{10^{-20}}$. It was not possible to list all results for the first two methods in the above table, but the last approximations are given .

7.1.4. Richardson Extrapolation for sequences of real numbers – an example

The results shown in Table 7.1 are telling us that both the Aitken scheme and the Steffensen method are accelerating the rate of convergence of the fixed-point iteration. However, it is not easy to predict by how much the rate of convergence will be increased. The application of the Richardson Extrapolation for accelerating the convergence of sequences of real numbers is giving us a means for the evaluation of the rate of convergence of the resulting new numerical procedures.

Consider, as in Example 7.1, a sufficiently smooth function $\mathbf{f}(\mathbf{u})$ (i.e. we are assuming that all its derivatives up to some order $\mathbf{p} \geq \mathbf{1}$ exist and are continuous), where $\mathbf{f}: [\mathbf{a}, \mathbf{b}] \rightarrow \mathbb{R}$ and $\mathbf{M} \stackrel{\text{def}}{=} \max_{\mathbf{u} \in [\mathbf{a}, \mathbf{b}]} \{\mathbf{f}'\} < \mathbf{1}$. Assume furthermore that $\mathbf{u_0} \in (\mathbf{a}, \mathbf{b})$ and that $\mathbf{n_0}$ is some sufficiently large integer, such that $\mathbf{u_0} \pm \mathbf{1/n} \in (\mathbf{a}, \mathbf{b})$ when $\mathbf{n} \geq \mathbf{n_0}$. Form two sequences the $\mathbf{n}$'th terms of which are defined by the following expressions:

$$(\mathbf{7.23}) \quad \frac{\boldsymbol{f\left(u_0 + \frac{1}{n}\right) - f(u_0)}}{\boldsymbol{\frac{1}{n}}} \ , \qquad \boldsymbol{n = n_0,\ n_0 + 1\ , \ldots\ ,}$$

$$(\mathbf{7.24}) \quad \frac{\boldsymbol{f(u_0) - f\left(u_0 - \frac{1}{n}\right)}}{\boldsymbol{\frac{1}{n}}} \ , \qquad \boldsymbol{n = n_0, n_0 + 1\ , \ldots\ .}$$

It is clear that

$$(\mathbf{7.25}) \quad \boldsymbol{f\left(u_0 + \frac{1}{n}\right) = f(u_0) + \frac{1}{n} f'(u_0) + \frac{1}{2n^2} f''(u_0) + \frac{1}{6n^3} f'''(\zeta)}$$

and hence we have

$$(\mathbf{7.26}) \quad \frac{\boldsymbol{f\left(u_0 + \frac{1}{n}\right) - f(u_0)}}{\boldsymbol{\frac{1}{n}}} - \boldsymbol{f'(u_0) = \frac{1}{2n} f''(u_0) + O\left(\frac{1}{n^2}\right)} \ ,$$

which means that the following two relationships hold:

$$(\mathbf{7.27}) \quad \boldsymbol{\lim_{n \to \infty} \left\{ \frac{f\left(u_0 + \frac{1}{n}\right) - f(u_0)}{\frac{1}{n}} - f'(u_0) \right\} = 0\ , \quad \lim_{n \to \infty} \left\{ \frac{f\left(u_0 + \frac{1}{n}\right) - f(u_0)}{\frac{1}{n}} \right\} = f'(u_0)\ .}$$

The following definition is useful for the presentation of the results in the remaining part of this section:

Definition 7.2: The convergence of the sequence of real numbers $\{\mathbf{f}(\mathbf{u_n})\}_{\mathbf{n=0}}^{\infty}$ to the limit $\mathbf{f}(\mathbf{u}^*)$ is of order $\mathbf{p \geq 1}$ if the inequality $|\mathbf{f}(\mathbf{u_n}) - \mathbf{f}(\mathbf{u}^*)| \leq \mathbf{M_2}\,/\mathbf{n^p}$, where $\mathbf{M_2}$ is a constant, holds for $\forall\ \mathbf{n}$.

■

We can obtain from (7.26) the following inequality (with some constant $\mathbf{M_2}$ that does not depend on $\mathbf{n}$), which shows that the convergence of the sequence defined by (7.23) to $\mathbf{f'(u_0)}$ is linear:

$$(\mathbf{7.28}) \qquad \left| \frac{f\left(u_0 + \frac{1}{n}\right) - f(u_0)}{\frac{1}{n}} - f'(u_0) \right| \leq \frac{M_2}{n}\ .$$

Similar arguments can be used to show that the sequence defined by (7.24) converges linearly to $\mathbf{f'(u_0)}$. More precisely the following inequality holds too with some constant $\mathbf{M_2}$, which does not depend on $\mathbf{n}$ and is not necessarily equal to that in (7.28):

$$(\mathbf{7.29}) \qquad \left| \frac{f(u_0) - f\left(u_0 - \frac{1}{n}\right)}{\frac{1}{n}} - f'(u_0) \right| \leq \frac{M_2}{n}\,.$$

The above analysis shows that we have two sequences of real numbers, (7.23) and (7.24), both of them converging linearly to $\mathbf{f'(u_0)}$. Let us create now a third sequence, each term of which is the sum of the corresponding terms of the sequences (7.23) and (7.24):

$$(\mathbf{7.30}) \qquad \frac{f\left(u_0 + \frac{1}{n}\right) - f\left(u_0 - \frac{1}{n}\right)}{\frac{2}{n}}\,, \qquad u_0 \pm \frac{1}{n} \in [a, b]\,, \qquad n = n_0,\ n_0 + 1\,, \ldots\ .$$

Transformations, as those performed above, applied in connection with the sequence (7.23), will give:

$$(\mathbf{7.31}) \qquad f\left(u_0 + \frac{1}{n}\right) = f\left(u_0 - \frac{1}{n}\right) + \frac{2}{n} f'\left(u_0 + \frac{1}{n}\right) + \frac{1}{3n^3} f'''(u_0) + \frac{1}{120n^5}\left[f^v(\zeta_1) + f^v(\zeta_2)\right],$$

and

$$(7.32) \quad \frac{f\left(u_0+\frac{1}{n}\right)-f\left(u_0+\frac{1}{n}\right)}{\frac{2}{n}}-f'(u_0)=\frac{1}{6n^2}f'''(u_0)+O\left(\frac{1}{n^4}\right) .$$

If we assume that function $\mathbf{f}$ is sufficiently smooth (i.e. in this particular case if it is continuously differentiable up to order five), then there exists a constant $\mathbf{M_3}$ such that the following inequality holds for sufficiently large values of $\mathbf{n}$:

$$(7.33) \quad \left|\frac{f\left(u_0+\frac{1}{n}\right)-f\left(u_0+\frac{1}{n}\right)}{\frac{2}{n}}-f'(u_0)\right| \leq \frac{M_3}{n^2} ,$$

which means that it is possible to obtain a sequence, the rate of convergence of which is **two**, by combining in an appropriate manner two sequences, which are only linearly convergent.

Higher rates of convergence can also be achieved. For example, it can be shown (by using the same approach as that used above) that the rate of convergence of the sequence given below is four:

$$(7.34) \quad f(\bar{\bar{u}}_n)=\frac{-f\left(u_0+\frac{2}{n}\right)+8f\left(u_0+\frac{1}{n}\right)-8f\left(u_0-\frac{1}{n}\right)+f\left(u_0+\frac{2}{n}\right)}{\frac{12}{n}} ,$$

$$u_0 \pm \frac{1}{n} \in [a,b] \quad \wedge \quad u_0 \pm \frac{2}{n} \in [a,b] , \qquad n=n_0,\, n_0+1 , \ldots .$$

Richardson Extrapolation can be applied in the efforts to predict better the behaviour of the improvement of the convergence rate. It is convenient here to introduce the same kind of notation as that used in the previous chapters of this book.

We shall first introduce the following two definitions:

$$(7.35) \quad w_n \stackrel{\text{def}}{=} \frac{f\left(u_0+\frac{1}{n}\right)-f(u_0)}{\frac{1}{n}} , \qquad z_n \stackrel{\text{def}}{=} \frac{f\left(u_0+\frac{2}{n}\right)-f(u_0)}{\frac{2}{n}} .$$

It should be noted that the first sequence in (7.35) is in fact the sequence defined in (7.23), the rate of convergence of which is one. It can easily be shown (by using after the second term Taylor expansions of function $\mathbf{f}$) that the rate of convergence of the second sequence in (7.35) is also one.

Form now, as in the previous chapters, the Richardson Extrapolation for the case where the order of convergence $\mathbf{p}$ is equal to one:

$$(7.36) \quad y_n \stackrel{\text{def}}{=} 2w_n - z_n = 2\frac{f\left(u_0+\frac{1}{n}\right)-f(u_0)}{\frac{1}{n}} - \frac{f\left(u_0+\frac{2}{n}\right)-f(u_0)}{\frac{2}{n}},$$

$$(7.37) \quad y_n = \frac{-f\left(u_0+\frac{2}{n}\right)+4f\left(u_0+\frac{1}{n}\right)-3f(u_0)}{\frac{2}{n}}.$$

It can be shown, by applying truncated (after the fifth term) expansions in Taylor series, that the Richardson Extrapolation is giving a sequence the order of convergence of which is two. Thus, the convergence is accelerated (by order at least equal to one) when the Richardson Extrapolation is used.

Consider now two other sequences of real numbers:

$$(7.38) \quad w_n = \frac{f\left(u_0+\frac{1}{n}\right)-f\left(u_0-\frac{1}{n}\right)}{\frac{2}{n}} \quad \wedge \quad z_n = \frac{f\left(u_0+\frac{2}{n}\right)-f\left(u_0-\frac{2}{n}\right)}{\frac{4}{n}}.$$

Since the order of convergence for these two sequence is $\mathbf{p}=\mathbf{2}$, it is clear that the order of convergence of the sequence obtained by using the Richardson extrapolation:

$$(7.39) \quad y_n = \frac{4w_n - z_n}{3}$$

will be $\mathbf{p}>\mathbf{2}$. It can easily be verified that $\mathbf{y_n}$ is actually equal to $\mathbf{f}(\overline{\overline{\mathbf{u}}}_\mathbf{n})$ and, thus, the order of convergence in this special case is not $\mathbf{p}=\mathbf{3}$ as usual, but $\mathbf{p}=\mathbf{4}$ due to the cancellation of the terms containing the third derivatives of $\mathbf{f}$ in the Taylor expansion given in (7.31).

7.2. Application of the Richardson Extrapolation to numerical integration

In this section, we shall discuss the use of the Romberg method, **Romberg (1955)**, for calculating approximations of the integral:

$$(7.40) \quad I(f) = \int_a^b f(x)dx\ , \quad a \in \mathbb{R}\ , \quad b \in \mathbb{R}\ , \quad a < b\ , \quad x \in [a,b]\ , \quad f:[a,b] \rightarrow \mathbb{R}\ .$$

The Romberg method is essentially an application of the Richardson Extrapolation to the Composite Trapezoidal Rule (see, for example, **Burden and Faires, 2001**).

Assume that $\mathbf{h = (b-a)/n}$, $\mathbf{x_j = a + jh}$, $\mathbf{j = 1}$, $\mathbf{2}, \ldots$, $\mathbf{n-1}$, and $\mathbf{f}$ is at least twice continuously differentiable in the interval $\mathbf{[a,b]}$, i.e. that $\mathbf{f \in C^2\,[a,b]}$. Then the **Composite Trapezoidal Rule** can be represented for some $\boldsymbol{\xi} \in \mathbf{(a,b)}$ by the following formula:

$$(7.41) \quad I_n(f) = \int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] + \frac{(b-a)f''(\xi)}{12}\,h^2\ .$$

The last term on the right-hand-side of (7.41) is the error of the integration method, but this term cannot be used directly for the evaluation of the error, because in the general case $\boldsymbol{\xi}$ is not known. The formula (7.41) is nevertheless important, because it shows that the Composite Trapezoidal Rule has **second order of accuracy**.

If we assume additionally that $\mathbf{f \in C^{\infty}\,[a,b]}$, then the Composite Trapezoidal Rule can be re-written as

$$(7.42) \quad I_n(f) = \int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] + \sum_{i=1}^{\infty} K_i h^{2i}\ .$$

The last formulation will be used in the derivation of the Romberg method as a successive implementation (in several consecutive steps) of the (repeated) Richardson Extrapolation in order to eliminate one term in the last sum in the right-hand-side at each step. Let us emphasize here that the coefficients $\mathbf{K_i}$ in the sum of (7.42) depend on the derivatives of function $\mathbf{f}$, but not on the increment $\mathbf{h}$.

Formula (7.42) is giving us the exact value of the integral, but unfortunately it cannot be directly used in practice either (due to the presence of the infinite sum in the last term on the right-hand-side). Therefore, it is necessary to truncate somehow this sum when actual computations are to be carried out. It is reasonable to start with removing the whole infinite sum and then gradually to improve the

accuracy of the approximations by adding successively terms containing $\mathbf{K_1}$, $\mathbf{K_2}$, $\mathbf{K_3}$, …, $\mathbf{K_r}$ and eliminating (one at a time) these constants. Moreover, the resulting truncated formulae can be used with different value of the parameter $\mathbf{n}$.

Let us start by removing the whole infinite sum from (7.42) and by applying the resulting formula for $\mathbf{n = 2^{k-1}}$, $\mathbf{k = 1, 2, 3, 4, 5,\ \ldots, q}$, It is convenient (and it is also very often used in the literature) to introduce the following notation for the resulting approximations: $\mathbf{T_{01}}$, $\mathbf{T_{02}}$, $\mathbf{T_{03}}$, …, $\mathbf{T_{0q}}$.

Consider two successive approximations $\mathbf{T_{0k}}$ and $\mathbf{T_{0k+1}}$. Then the true value of the integral can be expressed either by $\mathbf{T_{0k} + K_1\left(h/2^k\right)^2 + O(h^4)}$ or by $\mathbf{T_{0k+1} + K_1\left(h/2^{k+1}\right)^2 + O(h^4)}$. From these expressions for the value of the integral, it is clear that the order of accuracy of the following two approximations $\mathbf{T_{0k} + K_1\left(h/2^k\right)^2}$ or $\mathbf{T_{0k+1} + K_1\left(h/2^{k+1}\right)^2}$ is four. The problem is that the value of $\mathbf{K_1}$ is unknown. Therefore, Richardson Extrapolation should be used to eliminate this constant. We shall neglect the rest term $\mathbf{O(h^4)}$ in the above two expressions and then form:

$$(7.43)\quad T_{1k+1} = \frac{4\left[T_{0k+1} + K_1\left(h/2^{k+1}\right)^2\right] - \left[T_{0k} + K_1\left(h/2^k\right)^2\right]}{3} = \frac{4T_{0k+1} - T_{0k}}{3}.$$

It is seen that the constant $\mathbf{K_1}$ is indeed eliminated and the new approximations $\mathbf{T_{1k}}$, where $\mathbf{k = 1, 2, \ldots, q}$, are of order **four**.

Similar procedure can be used to obtain the next approximations. Consider now the following two successive approximations $\mathbf{T_{1k}}$ and $\mathbf{T_{1k+1}}$. Then the true value of the integral can be expressed either by $\mathbf{T_{1k} + K_2\left(h/2^k\right)^4 + O(h^6)}$ or by $\mathbf{T_{1k+1} + K_2\left(h/2^{k+1}\right)^4 + O(h^6)}$. From these expressions for the value of the integral, it is clear that the order of accuracy of the following two approximations $\mathbf{T_{0k} + K_2\left(h/2^k\right)^4}$ or $\mathbf{T_{0k+1} + K_2\left(h/2^{k+1}\right)^4}$ is four. The problem is, again, that the value of the constant $\mathbf{K_2}$ is unknown. Therefore, Richardson Extrapolation should be used again to eliminate this constant. We shall neglect the rest term $\mathbf{O(h^6)}$ in the above two expressions and then form:

$$(7.44)\quad T_{2k+1} = \frac{16\left[T_{1k+1} + K_2\left(h/2^{k+1}\right)^4\right] - \left[T_{1k} + K_2\left(h/2^k\right)^4\right]}{15} = \frac{16T_{1k+1} - T_{1k}}{15}.$$

It is seen that the constant $\mathbf{K_2}$ is indeed eliminated and the new approximations $\mathbf{T_{2k}}$, where $\mathbf{k = 2, 3, \ldots, q}$, are of order six.

In the same way, approximations $\mathbf{T_{3k}}$ of order **eight** can be obtained.

It is clear that the process can easily be continued if more accurate results are needed. Assume that the approximations $\mathbf{T_{j-1k}}$ have been calculated for $\mathbf{k = j - 1}$, $\mathbf{j}$, … , $\mathbf{q}$. Then the next column of

approximations, the approximations $\mathbf{T_{jk}}$ for $\mathbf{k = j, j + 1, \ldots, q}$ can be obtained by applying the following formulae:

$$(7.45) \quad T_{jk+1} = \frac{4^{j-1}T_{j-1k+1} - T_{j-1k}}{4^{j-1} - 1}.$$

The Romberg table (see below) can be used to represent conveniently the approximations:

T_{00}					
T_{01}	T_{11}				
T_{02}	T_{12}	T_{22}			
T_{03}	T_{13}	T_{23}	T_{33}		
T_{04}	T_{14}	T_{24}	T_{34}	T_{44}	
T_{05}	T_{15}	T_{25}	T_{35}	T_{45}	T_{55}
T_{06}	T_{16}	T_{26}	T_{36}	T_{46}	T_{56}
T_{07}	T_{17}	T_{27}	T_{37}	T_{47}	T_{57}
T_{08}	T_{18}	T_{28}	T_{38}	T_{48}	T_{58}

Figure 7.1

The Romberg table achieved when the Richardson Extrapolation is repeatedly applied in the calculation of approximations of the value of the integral given in formula (7.42).

Example 7.6: Consider the problem of finding the value of the integral:

$$(7.46) \quad I(f) = \int_0^1 e^{-x^2} dx$$

by using the Romberg method.

■

This problem was solved in three steps.

Step 1: Calculations with the **Composite Trapezoidal Rule** were performed during the first step for $\mathbf{41}$ different values of $\mathbf{n = 2^k}$ with $\mathbf{k = 0}$, $\mathbf{1}$, ... , $\mathbf{40}$ in order to calculate approximations $\mathbf{T_{0k}}$. The accuracy of the last of these $\mathbf{41}$ approximations was approximately equal to $\mathbf{10^{-25}}$. We proceeded by calculating, applying the Richardson Extrapolation, the quantities $\mathbf{T_{1k}}$ and continuing the calculations until the difference between two successive approximations became less (in absolute value) than $\mathbf{10^{-30}}$. This requirement was satisfied for $\mathbf{k = 21}$. The last of these $\mathbf{21}$ approximations, $\mathbf{T_{1,21}}$ was declared as a **reference solution**.

Step 2: The reference solution was used during the second step to evaluate the accuracy of the already calculated approximations and the rates of convergence.

Step 3: During the third step, Richardson Extrapolation was used to calculate approximations $\mathbf{T_{ik}}$ for $\mathbf{i = 2}$, $\mathbf{3}$, $\mathbf{4}$, $\mathbf{5}$. The calculations were continued, for each value of $\mathbf{i}$, until the difference $\mathbf{|T_{ik} - T_{ik-1}|}$ remained greater than $\mathbf{10^{-30}}$.

The value of the reference solution is: $\mathbf{0.746824132812427025399467436 1317}$.

Results are shown in Table 7.2 –Table 7.5.

k	T_{0k}	T_{1k}	T_{2k}	T_{3k}	T_{4k}	T_{5k}
0	0.683939720585721	-	-	-	-	-
1	0.731370251828563	0.747180428909510	-	-	-	-
2	0.742984097800381	0.746855379790987	0.746833709849752	-	-	-
3	0.745865614845695	0.746826120527466	0.746824169909898	0.746824018482281	-	-
4	0.746584596788221	0.746824257435730	0.746824133229614	0.746824132647387	0.746824133095094	-
5	0.746764254652296	0.746824140606985	0.746824132818402	0.746824132811874	0.746824132812519	0.746824132812243
6	0.746809163637827	0.746824133299672	0.746824132812518	0.746824132812424	0.746824132812427	0.746824132812427
7	0.746820390541617	0.746824132842881	0.746824132812428	0.746824132812427	0.746824132812427	0.746824132812427
8	0.746823197246152	0.746824132814330	0.746824132812427	0.746824132812427	0.746824132812427	
9	0.746823898920947	0.746824132812545	0.746824132812427	0.746824132812427		
10	0.746824074339562	0.746824132812434	0.746824132812427			
11	0.746824118194211	0.746824132812427				
12	0.746824129157873	0.746824132812427				
13	0.746824131898788	0.746824132812427				
14	0.746824132584017	0.746824132812427				
15	0.746824132755324	0.746824132812427				
16	0.746824132798151					
17	0.746824132808858					
18	0.746824132811534					
19	0.746824132812203					
20	0.746824132812371					
21	0.746824132812413					
22	0.746824132812423					
23	0.746824132812426					
24	0.746824132812426					
25	0.746824132812426					
26	0.746824132812427					
27	0.746824132812427					
28	0.746824132812427					
29	0.746824132812427					
30	0.746824132812427					
31	0.746824132812427					
32	0.746824132812427					

Table 7.2

Values of the integral (7.46) that are calculated by using the Romberg method. Extended (quadruple) precision is used (this means that the actual computations are carried out by using 32 significant digits, but the calculated approximations are given in the table with 15 significant digits). The calculations (excepting these used to calculate the results in the first column of the table) were carried out until the absolute value of the difference of the calculated approximation and the reference solution becomes less than 10^{-30}, but not all results are given in this table, because the rounded, to 15 significant digits, values are becoming quickly the same.

k	T_{0k}	T_{1k}	T_{2k}	T_{3k}	T_{4k}	T_{5k}
0	6.2884E-02	-	-	-	-	-
1	1.5454E-02	3.5630E-04	-	-	-	-
2	3.8400E-03	3.1247E-05	9.5770E-06	-	-	-
3	9.5852E-04	1.9877E-06	3.7097E-08	1.1433E-07	-	-
4	2.3954E-04	1.2462E-07	4.1719E-10	1.6504E-10	2.8267E-10	-
5	5.9878E-05	7.7946E-09	5.9751E-12	5.5212E-13	9.2926E-14	1.8329E-13
6	1.4969E-05	4.8725E-10	9.1314E-14	2.0786E-15	7.8405E-17	1.2355E-17
7	3.7423E-06	3.0454E-11	1.4189E-15	8.0460E-18	7.3962E-20	2.6086E-21
8	9.3557E-07	1.9034E-12	2.2139E-17	3.1358E-20	7.1614E-23	6.1516E-25
9	2.3389E-07	1.1896E-13	3.4580E-19	1.0944E-22	6.9787E-26	1.4884E-28
10	5.8473E-08	7.4352E-15	5.4026E-21	4.7815E-25	6.8226E-29	7.4726E-32
11	1.4618E-08	4.6470E-16	8.4415E-23	1.8682E-27	4.4970E-31	
12	3.6546E-09	2.9044E-17	1.3190E-24	7.3875E-30		
13	9.1364E-10	1.8152E-18	2.0610E-26	8.5906E-31		
14	2.2841E-10	1.1345E-19	3.2166E-28			
15	5.7102E-11	7.0908E-21	5.2778E-30			
16	1.4276E-11	4.4317E-22	1.0015E-31			
17	3.5689E-12	2.7698E-23				
18	8.9223E-13	1.7311E-24				
19	2.2306E-13	1.0820E-25				
20	5.5764E-14	6.7620E-27				
21	1.3941E-14	4.2266E-28				
22	3.4853E-15	2.6967E-29				
23	8.7131E-16	2.2999E-30				
24	2.1783E-16	6.3325E-31				
25	5.4457E-17					
26	1.3614E-17					
27	3.4036E-18					
28	8.5088E-19					
29	2.1271E-19					
30	5.3168E-20					
31	1.3282E-20					
32	3.3108E-21					

Table 7.3

Accuracy achieved in computations of the integral (7.46) carried out by using the Romberg method. Extended (quadruple) precision is used (this means that the actual computations are carried out by using 32 significant digits, but the calculated approximations are given in the table with 5 significant digits which is quite sufficient for this case). The calculations for T_{ik}, where $i = 1, 2, 3, 4, 5$, were carried out until the absolute value of the calculated approximation and the reference solution becomes less than 10^{-30}, which means that no calculations were carried out for the white cells in the table.

k	T_{0k}	T_{1k}	T_{2k}	T_{3k}	T_{4k}	T_{5k}
1	4.0692	-	-	-	-	-
2	4.0244	11.4026	-	-	-	-
3	4.0062	15.7200	258.1588	-	-	-
4	4.0016	15.9498	88.9228	692.7455	-	-
5	4.0004	15.9885	69.8215	298.9184	3041.8470	-
6	4.0001	15.9972	65.4341	265.6186	1185.1999	14835.9408
7	4.0000	15.9993	64.3573	258.3441	1060.0809	4736.1650
8	4.0000	15.9998	64.0892	256.5824	1032.7875	4240.5409
9	4.0000	16.0000	64.0223	256.1454	1026.1818	4133.1341
10	4.0000	16.0000	64.0056	256.0364	1022.8802	1991.7539
11	4.0000	16.0000	64.0011	255.9386	151.7120	
12	4.0000	16.0000	64.0004	252.8891		
13	4.0000	16.0000	63.9974	8.5995		
14	4.0000	16.0000	64.0740			
15	4.0000	16.0000	60.9448			
16	4.0000	16.0000	52.7000			
17	4.0000	16.0000				
18	4.0000	16.0000				
19	4.0000	15.9999				
20	4.0000	16.0009				
21	4.0000	15.9996				
22	4.0000	15.6724				
23	4.0000	11.7249				
24	4.0000	3.6320				
25	4.0000					
26	4.0000					
27	4.0000					
28	4.0000					
29	4.0000					
30	4.0000					
31	4.0000					
32	4.0000					

Table 7.4

Rates of convergence achieved in the calculations of values of the integral from (7.46) by using the Romberg method. Extended (quadruple) precision is used (this means that the actual computations are carried out by using $\mathbf{32}$ significant digits, but the calculated approximations are given in the table with $\mathbf{5}$ significant digits which is quite sufficient for this case). The calculations were carried out until the absolute value of the calculated approximation and the reference solution becomes less than $\mathbf{10^{-30}}$, which means that no calculations were carried out for the white cells in the table. The perfect values of the rates are: $\mathbf{4}$, $\mathbf{16}$, $\mathbf{64}$, $\mathbf{256}$, $\mathbf{1024}$ and $\mathbf{4096}$ respectively.

T_{ik}	Calculated Result	Accuracy	Rate
T_{01}	0.74658459678822154917677658816 9	2.3954E-04	-
T_{12}	0.74682414060698510199109020578 1	7.7946E-09	3.0731E+04
T_{23}	0.74682413281251833958804978174 2	9.1314E-14	8.5360E+04
T_{34}	0.74682413281242701735351577762 2	8.0460E-18	1.1349E+04
T_{45}	0.74682413281242702539953904987 2	7.1614E-23	1.1235E+05
T_{56}	0.74682413281242702539946743598 3	1.4884E-28	4.8116E+05

Table 7.5

Values of the integral calculated in the second diagonal of the Romberg method. Extended (quadruple) precision is used (this means that the actual computations are carried out by using $\mathbf{32}$ significant digits) and all significant digits are given in the column under "Calculated Results"; the use of less significant digits was quite enough when accuracy results and rates of convergence were listed. It is demonstrated in this table that **the convergence along the diagonals of the Romberg table is extremely fast**.

7.3. General conclusions related to the seventh chapter

It has been shown in this chapter that the Richardson Extrapolation and some alternative approaches (the Aitken scheme and the Steffensen method) can successfully be applied to accelerate the convergence of sequences of real numbers. Such sequences can appear when iterative methods are used in the solution of algebraic and transcendental equations. To facilitate the explanations, the results were derived for the scalar case, but most of the results can be generalized for the case where systems of algebraic and/or transcendental equations are to be treated.

It was also demonstrated that repeated Richardson Extrapolation is a very powerful approach in the attempts to increase the accuracy of the approximations to the values of integrals in the case where the Composite Trapezoidal Rule is the underlying method.

7.4. Some research topics

The following topics might lead to some very interesting and useful results:

(A) Try to generalize some of the results presented in Section 7.1 for the case where $\mathbf{s > 1}$.

(B) The results presented in Section 7.2 might be used with other methods for numerical integrations (different from the Composite Trapezoidal Rule).

Chapter 8

General conclusions

It was shown in the first four chapters of this book that the Richardson Extrapolation is a very general and powerful tool for improving the performance of time-dependent problems arising in many different areas of science and engineering. It can successfully be used both as a tool for improving the accuracy of the numerical results and for error control during the computational process. In this book we emphasized the fact that one should be very careful because the Richardson Extrapolation may lead to unstable computations. Precisely this happens when the well-know and often used by scientists and engineers Trapezoidal Rule is combined with the Richardson Extrapolation. Therefore it is necessary to study carefully the stability properties of the combined method (the method for solving systems of ordinary differential equations when it is used together with the Richardson Extrapolation). This is normally very difficult. Different cases in which the stability properties are preserved are studied in the first four chapters. It is shown, in Chapter 2, that sometimes the application of the Richardson Extrapolation may result in new numerical methods with improved stability properties.

The important for scientists and engineers problem of using splitting procedures is shortly discussed in Chapter 5. Some stability results for the sequential splitting procedure are treated there.

There are two approaches of introducing the Richardson Extrapolation for partial differential equations (or systems of partial differential equations), which are studied in Chapter 6:

(A) **The first approach** is based on the use of semi-discretization (discretization of the spatial derivatives) by applying either finite elements or finite differences. This leads to a transformation of the partial differential equation or the systems of partial differential equations into a system of ordinary differential equations. The methods from the first four chapters can after that be used in the application of the Richardson Extrapolation. This approach is very straight-forward, but it can successfully be used only when the truncation errors made when the spatial derivatives are discretized are much smaller than the errors due to the use of the selected numerical method for solving ordinary differential equations.

(B) **The second approach** is based on a direct implementation of the Richardson Extrapolation to the partial differential equation or the system of partial differential equations. The solution of the problem of applying the Richardson Extrapolation in this case is much more difficult than the solution of the problem in the first approach. However, the second approach is much more robust when it is correctly implemented.

The above analysis indicates that each of the approaches has advantages and drawbacks. The correct choice will in general depend very much on the particular problem that must be solved.

Some results related to the applications of the Richardson Extrapolation and some other methods for accelerating the rate of convergence of sequences are treated in the first section of the seventh chapter.

The Romberg method for calculations of vales of integrals also is treated in the seventh chapter (in the second sections).

At the end of each chapter, some research problems are listed. The hope is that the reader will be able to solve some of these problems if that is necessary for the solution of the particular problem which she or he has to handle.

References

Abdalmogith, S., Harrison, R.M. and Zlatev, Z. (2004): *"Intercomparison of inorganic aerosol concentrations in the UK with predictions of the Danish Eulerian Model"*, Journal of Atmospheric Chemistry, **Vol. 54**, pp. 43-66.

Adams, J. C. (1883): *"Appendix"*, published in **Bashforth, F. (1883):** *"An attempt to test the theories of capillary action by comparing theoretical and measured forms of drops of fluids. With an explanation of the methods of integration employed in constructing the tables which give the theoretical form of such drops"*, Cambridge University Press, Cambridge.

Aitken, A. (1926): *"On Bernoulli's numerical solution of algebraic equations"*, Proceedings of the Royal Society of Edinburgh, **Vol. 46**, pp. 289-305.

Alexander, R. (1977): *"Diagonally implicit Runge-Kutta methods for stiff ODE's"*, SIAM Journal of Numerical Analysis, **Vol. 14**, pp. 1006-1021.

Alexandrov, V., Owczarz, W., Thomsen, P. G. and Zlatev, Z. (2004): *"Parallel runs of large air pollution models on a grid of SUN computers"*, Mathematics and Computers in Simulations, **Vol. 65**, pp. 557-577.

Alexandrov, V., Sameh, A., Siddique, Y. and Zlatev, Z. (1997): *"Numerical integration of chemical ODE problems arising in air pollution models"*, Environmental Modelling and Assessment, **Vol. 2**, pp. 365-377.

Ambelas Skjøth, C. Bastrup-Birk, A, Brandt, J. and Zlatev, Z. (2000): *"Studying variations of pollution levels in a given region of Europe during a long time-period"*, Systems Analysis Modelling Simulation, **Vol. 37**, pp. 297-311.

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. (1992): *"LAPACK: Users' Guide"***.** SIAM, Philadelphia.

Banach, S. (1922): *"Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales"* Fundamenta Mathematicae, **Vol. 3**, pp. 133–181.

Barker, V. A., Blackford, L. S., Dongarra, J., Du Croz, J., Hammarling, S., Marinova, M., Wasnewski, J., and Yalamov, P. (2001): *"LAPACK95: Users' Guide"***.** SIAM, Philadelphia.

Bastrup-Birk, A., Brandt, J., Uria, I. and Zlatev, Z. (1997): *"Studying cumulative ozone exposures in Europe during a 7-year period"*, Journal of Geophysical Research, **Vol. 102** , pp. 23917-23035.

Bashforth, F. (1883): *"An attempt to test the theories of capillary action by comparing theoretical and measured forms of drops of fluids. With an explanation of the methods of integration employed in constructing the tables which give the theoretical form of such drops"*, Cambridge University Press, Cambridge.

Botchev, M. A. and Verwer, J. G. (2009): *"Numerical Integration of Damped Maxwell Equations"***,** SIAM Journal on Scientific Computing, **Vol. 31,** pp. 1322-1346.

Brezinski, C.: *"Some pioneers of extrapolation methods"***,** Université des Sciences and Technologies de Lille, Lille, France http://nalag.cs.kuleuven.be/research/projects/WOG/history/talks/brezinski_talk.pdf

Brezinski, C. (1985): *"Convergence acceleration methods: The past decade"*, Journal of Computational and Applied Mathematics, **Vol. 12-13,** pp. 19-36.

Brezinski, C. (2000): *"Convergence acceleration during the twentieth century"*, Journal of Computational and Applied Mathematics, **Vol. 122**, pp. 1-21.

Brezinski, C. and Matos, A. C. (1996): *"Derivation of extrapolation algorithms based on error estimates"*, Journal of Computational and Applied Mathematics, **Vol. 66**, pp. 5-26.

Burden, R. L. and Faires, J. D. (2001): *"Numerical Analysis", Ninth edition, Brooks/Cool, United States.*

Burrage, K. (1995): *"Parallel and Sequential Methods for Ordinary Differential Equations"*, Oxford University Press, Oxford, New York.

Butcher, J. C. (2003): *"Numerical Methods for Ordinary Differential Equations"*, Second edition, Wiley, New York.

Chock, D. P. (1985): *A comparison of numerical methods for solving advection equations III.* Atmospheric Environment, **Vol. 19,** pp. 571-586

Chock, D. P. (1991): *A comparison of numerical methods for solving advection equations II.* Atmospheric Environment,. **Vol. 25A,** pp. 853-871.

Chock, D. P. and Dunker, A. M. (1983): *A comparison of numerical methods for solving advection equations I.* Atmospheric Environment, **Vol. 17,** pp. 11-24.

Christiansen, E. and Petersen, H. G. (1989): *"Estimation of convergence orders in repeated Richardson extrapolation"*, BIT, **Vol. 29,** pp. 48-59.

Crank, J. and Nicolson, P. (1947): *"A practical method for numerical integration of partial differential equations of heat-conduction type"*, Proceedings of the Cambridge Philosophical Society, **Vol. 43,** pp. 50-67.

Crowley, W. P. (1968): *"Numerical advection experiments"*, Monthly Weather Review, **Vol. 96,** pp. 1-11.

Cruziex, M. (1976): *"Sur les méthodes de Runge-Kutta pour l'approximation des problemes d'évolution"*, in "Computing Methods in Applied Science and Engineering", Second International Symposium, 1975, (R. Glowiski and J. L. Lions, eds.), pp. 206-233, Lecture Notes in Economical and Mathematical Systems, No. 134, Springer-Verlag, Berlin.

Cryer, C. W. (1972): *"On the instability of high-order backward-difference multistep methods"*, BIT, **Vol. 12**, pp. 17-25.

Csomós, P., Cuciureanu, R., Dimitriu, G., Dimov, I., Doroshenko, A., Faragó, I., Georgiev, K., Havasi, Á., Horváth, R., Margenov, S., Ostromsky, Tz., Prusov, V., Syrakov, D. and Zlatev, Z., (2006): "*Impact of Climate Changes on Pollution Levels in Europe"*, Final report for a NATO Linkage Project (Grant 980505), available at http://www.cs.elte.hu/~faragois/NATO.pdf, http://www2.dmu.dk/atmosphericenvironment/Climate%20and%20Pollution, http://www.umfiasi.ro/NATO.pdf, http://www.personal.rdg.ac.uk/~sis04itd/MyPapers/climatic_scenarios_NATO.pdf, http://www.softasap.net/ips/climatic_scenarios_NATO.pdf, http://www.meteo.bg/bulair/NATO.pdf .

Dahlquist, G. (1956): *"Convergence and stability in the numerical integration of ordinary differential equations"*, Mathematica Scandinavica, **Vol. 4,** pp. 33-53.

Dahlquist, G. (1959): *"Stability and error bounds in the numerical integration of ordinary differential equations"*, Doctoral thesis in Transactions of the Royal Institute of Technology, **No. 130,** Stockholm.

Dahlquist, G. (1963): *"A special stability problem for linear multistep methods"*, BIT, **Vol. 3,** pp. 27-43.

Dahlquist, G., Björck. Å. and Anderson, N. (1974): *"Numerical Methods"*, Prentice Hall, Englewoods Cliffs, NJ.

Demmel, J. W. (1997): *"Applied Numerical Linear Algebra"*, SIAM (Society for Industrial and Applied Mathematics), Philadelphia.

Deuflhard, P., Hairer, E. and Zugck, J. (1987): *"One-step and extrapolation methods for differential-algebraic systems"*, Numerische Mathematik, **Vol. 51,** pp. 501-516.

Duff, I. S., Erisman, A. M. and Reid, J. K. (1986): *"Direct Methods for Sparse Matrices"*, Oxford University Press, Oxford-London.

Dutka, J.: (1984): *"Richardson-extrapolation and Romberg-integration"* Historia Mathematica, **Vol. 11**, pp. 3-21.

Ebel, A., Feldmann, H., Jakobs, H. J., Memmesheimer, M. Offermann, D., Kuell, V. and Schäller, B. (2008): *"Simulation of transport and composition changes during a blocking episode over the East Atlantic and North Europe"*, Ecological Modelling, **Vol. 217**, pp. 240-254.

Ehle, B. L. (1968): *"High order A-stable methods for numerical solution of systems of DEs"*, BIT, Vol. 8, pp. 276-278.

EMEP Home Web-page: http://www.emep.int/index_data.html.

Enright, W. H., Bedet, R. Farkas, I. and Hull, T. E. (1974): *"Test results on initial value methods for non-stiff ordinary differential equations"*, Technical Report No. 68, Department of Computer Science, University of Toronto, Toronto, Canada.

Faragó, I. (2008): *"Numerical treatment of linear parabolic problems"*, Doctoral Dissertation, Eötvös Loránd University, Budapest.

Faragó, I., Havasi, Á. and Zlatev, Z. (2010): *"Efficient implementation of stable Richardson Extrapolation algorithms",* Computers and Mathematics with Applications, **Vol. 60,** pp. 2309-2325.

Fehlberg, E. (1966): *"New high-order Runge-Kutta formulas with an arbitrarily small truncation error"*, Z. Angew. Math. Mech., **Vol. 46**, pp. 1-15.

Fletcher, R. (1972): *"FORTRAN subroutines for minimization by quasi Newton methods"*, Report No. R7125, A.E.R.E., Harwell, England.

Fornberg, E. (1975): *"On a Fourier method for the integration of hyperbolic equations"*, SIAM Journal on Numerical Analysis, **Vol. 12**, pp. 509-528.

Fornberg, E. (1996): *"A practical guide to pseudospectral methods"*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge.

Gear, C. W. (1971): *"Numerical Initial Value Problem for Ordinary Differential Equations",* Prentice-Hall, Englewood Cliffs, New Jersey, USA.

Gear, C. W. and Tu, K. W. (1974): *"The effect of variable mesh on the stability of multistep methods",* SIAM Journal on Numerical Analysis, **Vol. 11**, pp. 1024-1043.

Gear, C. W. and Watanabe, D. S. (1974): *"Stability and convergence of variable multistep methods",* SIAM Journal on Numerical Analysis, **Vol. 11**, pp. 1044-1053.

Geernaert, G. and Zlatev, Z (2004): *"Studying the influence of the biogenic emissions on the AOT40 levels in Europe", International Journal of Environment and Pollution,* **Vol. 23, No. 12,** pp. 29-41.

Geiser, J. (2008): *"A numerical investigation for a model of the solid-gas phase of a crystal growth apparatus"*, Communications in Computational Physics. **Vol. 3**, pp. 913-934.

George, J. A. and Liu, J. W. (1981): *"Computer Solution of Large Sparse Positive Definite Matrices"*, Prentice-Hall, Englewood Cliffs, N. J.

Golub, G. H. Van Loan, C. F. (1983): *"Matrix Computations"*, The Johns Hopkins University Press, Baltimore, Maryland.

Hairer, E., Nørsett, S. P. and Wanner, G. (1987): *"Solving Ordinary Differential Equations: I Nonstiff"*, Springer-Verlag, Berlin.

Hairer, E. and Wanner, G. (1991): *"Solving Ordinary Differential Equations: II Stiff and Differential-Algebraic Problems"*, Springer-Verlag, Berlin.

Hamming, R. W. (1962): *"Numerical Methods for Scientists and Engineers"*, McGraw Hill Book Company, New York-San Francisco-Toronto-London. There exists a Russian translation of this book: Хемминг, Р. В. (1968): *"Численные Методы для Научных Работников и Инжихеров"*, Издательство "Наука", Москва.

Harrison, R. M., Zlatev, Z. and Ottley, C. J. (1994): *"A comparison of predictions of an Eulerian atmospheric transport-chemistry model with experimental measurements over the North Sea"*, Atmospheric Environment, **Vol. 28, No. 2**, pp. 497-516.

Hartmann, Ph. (1964): *"Ordinary Differential Equations"*, Wiley, New York. There exists a Russian translation of this book: **Хартман, Ф. (1970):** *"Обыкновеные Дифференциалные Уравнения"*, Издательство "Мир", Москва.

Hass, H., van Loon, M., Kessler, C., Stern, R., Mathijsen, J., Sauter. F., Zlatev, Z., Langner, J., Foltescu, V. and Schaap, M. (2004): *"Aerosol modelling: Results and intercomparison from European regional-scale modelling systems"*. GSF–National Research Center for Environment and Health, International Scientific Secretariat (ISS), EUROTRAC-2, Münich, see also http://www.trumf.fu-berlin.de/veranstaltungen/events/gloream, http://www.rivm.nl/bibliotheek/digitaaldepot/GLOREAM_PMmodel-comparison.pdf.

Havasi, Á. and Zlatev, Z. (2002): *"Trends of Hungarian air pollution levels on a long time-scale"*, Atmospheric Environment, **Vol. 36**, pp. 4145-4156.

Henrici, P. (1968): *"Discrete Variable Methods in Ordinary Differential Equations"*, Wiley, New York.

Heun, K. (1900): *"Neue Methode zur approximativen Interation der Differentialgleichungen einer unabhängigen Veränderlichen"*, Zeitschrift für Mathematik und Physik, **Vol. 45**, pp. 23-38.

Hindmarsh, A. C. (1971): *"GEAR: ordinary differential equation solver"*, Report No. UCLR-51186, Lawrence Livermore Laboratory, Livermore, California, USA.

Hindmarsh, A. C. (1980): *"LSODE and LSODI, two new solvers of initial value ordinary differential equations"*, ACM SIGNUM Newsletter, **Vol. 15**, pp. 10-11.

Hirayama, A., Shimodaira, K. and Hirose, H., eds., (1974): *"Takakazu Seki's collected works"*, Kyoiku Tosho, Osaka, Japan.

Hundsdorfer, W. and Verwer, J. G. (2003): *"Numerical Solution of Time-Dependent Advection–Diffusion–Reaction Equations"*, Springer-Verlag, Berlin.

Jeltsch, R. and Nevanlinna, O. (1981): *"Stability of explicit time discretizations for solving initial value problems"*, Numerische Mathematik, **Vol. 37**, pp 245-261.

Jeltsch, R. and Nevanlinna, O. (1982): *"Stability and accuracy of time-discretizations for initial value problems"*, Numerische Mathematik, **Vol. 40**, pp. 61-91.

Jennings, A. (1977): *"Matrix Computations for Engineers and Scientists"*, Wiley, Chichester - New York – Brisbane, Toronto.

Joyce, D. C. (1971): *"Survey of extrapolation techniques in numerical analysis",* SIAM Review, **Vol. 13,** pp. 435-490.

Kantorovich, L. V. and Akilov, G. P. (1964): *"Function Analysis in Normed Spaces"*, Pergamon Press, Oxford-London-Edinburgh-New York-Paris-Frankfurt (this is the English translation of the original Russian book: **Канторович, Л. В. и Акилов, Г. П. (1959):** *"Функциональный Анализ в Нормированных Пространствах"*, Физматлит, Москва).

Krogh, F. T. (1973a): *"Algorithms for changing the stepsize",* SIAM Journal on Numerical Analysis, **Vol. 10,** pp. 949–965.

Krogh, F. T. (1973b): *"On testing a subroutine for numerical integrations of ordinary differential equations",* Journal of the Association of Computing Machinery, **Vol. 20,** pp. 545–562.

Kutta, W. (1901): *"Beitrag zur näherungsweisen Integration totaler Differentialgleichungen",* Zeitschrift für Mathematik und Physik, **Vol. 46,** pp. 435–453.

Lambert, J. D. (1991): *"Numerical Methods for Ordinary Differential Equations"*, Wiley, New York.

Lapidus, L. and Pinder, G. P. (1982): *"Numerical Solution of Partial Differential Equations in Science and Engineering"*, Wiley, New York.

LeVeque, R. J. (1992): *"Finite Volume Methods for Hyperbolic Problems"*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge.

Marchuk, G. I. (1968): *"Some applications of splitting-up methods to the solution of mathematical physics problems"*, Applikace Matematiky (Applications of Mathematics), **Vol. 13, No. 2,** pp. 103-132.

Marchuk, G. I. (1980): *"Methods of Computational Mathematics"*, Nauka, Moscow.

Marchuk, G. I. (1982): *"Mathematical Modelling for the Problem of Environment"*. Nauka, Moscow.

Marchuk, G. I. (1986): *"Mathematical Modelling for the Problem of Environment"*, Studies in Mathematics and Applications, No. 16, North-Holland, Amsterdam.

Marchuk, G. I. (1988): *"Methods of Splitting"*, Nauka, Moscow.

Memmesheimer, M., Ebel, A. and Roemer, F. (1997): *"Budget calculations for ozone and its precursors: Seasonal and episodic features based on model simulations"*, Journal of Atmospheric Chemistry, **Vol. 28,** pp.283-317.

Milne, W. E. (1926): *"Numerical integration of ordinary differential equations"*, American Mathematical Monthly, **Vol. 33,** pp.455-460.

Milne, W. E. (1953): *"Numerical Solution of Differential equations"*, American Mathematical Monthly, Wiley, New York (there exists a second edition of this book published in 1970 by Dover Publications, New York).

Molenkampf, C. R. (1968): *"Accuracy of finite-difference methods applied to advection equations"*, Journal of Applied Meteorology, **Vol. 7,** pp.160-167.

Morton, K. W. (1996): *"Numerical Solution of Convection-Diffusion Problems"*, Chapman and Hall, London.

Moulton, F. R. (1926): *"New Methods in Exterior Ballistics"*, University of Chicago, Illinois.

Nordsieck, A. (1962): *"On the numerical integration of ordinary differential equations"*, Mathematics of Computation, **Vol. 16,** pp. 22-49.

Nørsett, S. P. (1976): *"Semi-explicit Runge-Kutta methods"*, Report Mathematics and Computations, No. 6/74, Department of Mathematics, University of Trondheim.

Osada, N, (1993): *"Acceleration methods for slowly convergent sequences and their applications"*, Thesis, http://www.cis.twcu.ac.jp/~osada/thesis_osada.pdf.

Parlett, B. N. (1980): *"The Symmetric Eigenvalue Problem"*, Prentice Hall, Englewood Cliffs, N. J.

Piotrowski, P. (1969): "Stability, consistency and convergence of K-step methods for numerical integration of large systems of ordinary differential equations", in **"Conference for Numerical Solution of Differential Equations"** (J. Ll. Morris, ed.), pp. 221-227, Springer, Berlin-Heidelberg-New York.

Richardson, L. F. (1911): *"The Approximate Arithmetical Solution by Finite Differences of Physical Problems Including Differential Equations, with an Application to the Stresses in a masonry dam",* Philosophical Transactions of the Royal Society of London, **Series A, Vol. 210,** pp. 307–357.

Richardson, L. F. (1927): *"The Deferred Approach to the Limit, I—Single Lattice",* Philosophical Transactions of the Royal Society of London, **Series A, Vol. 226,** pp. 299–349.

Roemer, M., Beekman, M., Bergsröm, R., Boersen, G., Feldmann. H., Flatøy. F., Honore. C., Langner, J., Jonson. J. E., Matthijsen. J., Memmesheimer, M., Simpson, D., Smeets, P., Solberg. S., Stevenson, D., Zandveld, P. and Zlatev. Z. (2004): *"Ozone trends according to ten dispersion models".* GSF – National Research Center for Environment and Health, International Scientific Secretariat (ISS), EUROTRAC-2, Münich, **2004** (available also at: http://www.mep.tno.nl/eurotrac/EUROTRAC-trends.pdf).

Romberg, W. (1955): *"Vereinfachte numerische Integration"*, Det Kongelige Norske Videnskabers Selskab Forhandlinger (Trondheim), **Vol. 28**, pp. 30-36.

Runge, C. (1895): *"Über die numerische Auflösung von Differentialgleichungen",* Mathematische Annallen, **Vol. 46,** pp. 167–178.

Sewell, G. (2014): *"Computational Methods of Linear Algebra"*, Third Edition, World Scientific Publishing, Singapore.

Shampine, L. F. (1984): *"Stiffness and automatic selection of ODE codes",* Journal of Computational Physics, **Vol. 54,** pp. 74-86.

Shampine, L. F. (1994): *"Numerical Solution of Ordinary Differential Equations",* Chapman and Hall, New York - London.

Shampine, L. F. and Gordon, M. K. (1975): *"Computer Solution of Ordinary Differential Equations: The Initial Value Problem",* Freeman, San Francisco, California.

Shampine, L. F., Watts, H. A. and Davenport, S. M. (1976): *"Solving non-stiff ordinary differential equations: The state of the art",* SIAM Reviews, **Vol. 18**, pp. 376-411.

Shampine, L. F. and Zhang, W. (1990): *"Rate of convergence of multistep codes started with variation of order and stepsize",* SIAM Journal on Numerical Analysis, **Vol. 27**, pp. 1506-1518.

Simpson, D., Fagerli, H., Jonson, J. E., Tsyro, S. G., Wind, P. and J.-P. Tuovinen, J.-P. (2003): *"Transboundary Acidification, Eutrophication and Ground Level Ozone in Europe, Part I. Unified EMEP Model Description"*, EMEP/MSC-W Status Report 1/2003, Norwegian Meteorological Institute, Oslo, Norway**.**

Smith, G. D. (1978): *"Numerical Solution of Partial Differential Equations: Finite Difference Methods"*, Oxford Applied Mathematics and Computing Science Series, Second Edition (First Edition published in 1965), Clarendon Press, Oxford.

Steffensen, J. F. (1950): *"Interpolation"*, Chelsea, New York.

Stewart, G. W. (1973): *"Introduction to Matrix Computations"*, Academic Press, New York – San Francisco, London.

Strang, G. (1968): *"On the construction and comparison of difference schemes"*, SIAM Journal of Numerical Analysis, **Vol. 5**, pp. 505-517.

Strikwerda, J. C. (2004): *"Finite Difference Schemes and Partial Differential Equations"*, Second edition, SIAM, Philadelphia.

Syrakov, D., Spiridonov, V., Prodanova, M., Bogatchev, A., Miloshev, N., Ganev, K., Katragkou, E., Melas, D., Poupkou, A., Markakis, K., San José, R. and Pérez, J. I. (2011): *"A system for assessment of climatic air pollution levels in Bulgaria: description and first steps towards validation"*, International Journal of Environment and Pollution, **Vol. 46**, pp. 18-42.

Thomsen, P. G. and Zlatev, Z. (1979): *"Two-parameter families of predictor-corrector methods for the solution of ordinary differential equations"*, BIT, **Vol. 19**, pp. 503-517.

Trefethen, Ll. N. and Bau, D. (1997): *"Numerical Linear Algebra"*, SIAM (Society for Industrial and Applied Mathematics), Philadelphia.

Verwer, J. G. (1977): *"A class of stabilized Runge-Kutta methods for the numerical integration of parabolic equations",* Journal of Computational and Applied Mathematics, **Vol. 3**, pp. 155-166.

Waltz, G. (1996): *"The history of extrapolation methods in numerical analysis"*, Akademie, Berlin, https://ub-madoc.bib.uni-mannheim.de/1674/1/130_1991.pdf

Wilkinson, J. H. (1963): *"Rounding Errors in Algebraic Processes"*, Notes in Applied Sciences, No, 32, HMSO, London.

Wilkinson, J. H. (1965): *"The algebraic eigenvalue problem"*, Oxford University Press, Oxford-London.

Zlatev, Z. (1978): *"Stability properties of variable stepsize variable formula methods",* Numerische Mathematik, **Vol. 31**, pp. 175-182.

Zlatev, Z. (1981a): *"Modified diagonally implicit Runge-Kutta methods",* SIAM Journal of Scientific and Statistical Computing, **Vol. 2**, pp. 321-334.

Zlatev, Z. (1981b): *"Zero-stability properties of the three-ordinate variable stepsize variable formula methods",* Numerische Mathematik, **Vol. 37**, pp. 157-166.

Zlatev, Z. (1983): *"Consistency and convergence of general multistep variable stepsize variable formula methods",* Computing, **Vol. 31**, pp. 47-67.

Zlatev, Z. (1984): *"Application of predictor-corrector schemes with several correctors in solving air pollution problems",* BIT, **Vol. 24**, pp. 700-715.

Zlatev, Z. (1985a): *"Variable stepsize variable formula methods based on predictor-corrector schemes",* Applied Numerical Mathematics, **Vol. 1**, pp. 223-233.

Zlatev, Z. (1985b): *"Mathematical model for studying the sulphur pollution over Europe",* Journal of Computational and Applied Mathematics, **Vol. 12**, pp. 651-661.

Zlatev, Z. (1987): *"Transition to variable stepsize variable formula methods for solving ordinary differential equations",* In "Computational Mathematics II" (S. O. Fatunla, ed.), pp. 97-114, Boole Press, Dublin.

Zlatev, Z. (1988): *"Treatment of some mathematical models describing long-range transport of air pollutants on vector processors",* Parallel Computing, **Vol. 6**, pp. 88-98.

Zlatev, Z. (1989): *"Advances in the theory of variable stepsize variable formula methods for ordinary differential equations",* Applied Mathematics and Computations, **Vol. 31**, pp. 209-249.

Zlatev, Z. (1991): *"Computational Methods for General Sparse Matrices"*, Kluwer Academic Publishers, Dordrecht-Boston-London (now distributed by Springer-Verlag, Berlin).

Zlatev, Z. (1995): *"Computer Treatment of Large Air Pollution Models"*, Kluwer Academic Publishers, Dordrecht- Boston-London (now distributed by Springer-Verlag, Berlin).

Zlatev, Z. (2010): *"Impact of future climate changes on high ozone levels in European suburban areas"*. Climatic Change, **Vol. 101**, pp. 447-483.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1982): *"Choice of time-integration scheme in pseudospectral algorithm for advection equations"*, In **"Numerical Methods for Fluid Dynamics"** (K. W. Morton and M. J. Baines, eds.), pp. 303-321, Academic Press, London-New York.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1983a): *"Testing Subroutines Solving Advection-Diffusion Equations in Atmospheric Environments"*, Computers and Fluids, **Vol. 11**, pp. 13-38.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1983b): *"Three–dimensional advection-diffusion modelling for regional scale"*, Atmospheric Environment, **Vol. 17**, pp. 491-499.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1983c): *"Stability restriction of time-stepsize for numerical methods for first-order partial differential equations"*, Journal of Computational Physics, **Vol. 51**, pp. 1-27.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1984a): *"Implementation of a variable stepsize variable formula method in the time-integration part of a code for treatment of long-range transport air pollution"*, Journal of Computational Physics, **Vol. 55**, pp. 278-301.

Zlatev, Z., Berkowicz, R. and Prahm, L. P. (1984b): *"Package ADM for studying long-range transport of pollutants in the atmosphere"*, In **"Software: Modules, Interfaces and Systems"** (B. Engquist and T. Smedsaas, eds.), pp. 153-169, North-Holland, Amsterdam.

Zlatev, Z. and Christensen, J. (1989): *"Studying the sulphur and nitrogen pollution over Europe"*, In **"Pollution Modeling and Its Application"** (H. van Dop, ed.), pp. 351-360, North-Holland, Amsterdam

Zlatev, Z. and Dimov, I. (2006): *"Computational and Numerical Challenges in Environmental Modelling"*, Elsevier, Amsterdam, Boston, Heidelberg, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sidney, Tokyo.

Zlatev, Z., Dimov, I. and Georgiev, K. (2016): *"Relations between climatic changes and high pollution levels in Bulgaria"*, Open Journal of Applied Sciences, **Vol. 6**, pp. 386-401.

Zlatev, Z., Dimov, I., Faragó, I., Georgiev, K. and Havasi, Á. (2016): *"Stability properties of Runge-Kutta methods combined with Marchuk-Strang splitting and Richardson Extrapolation"*, unpublished manuscript.

Zlatev, **Z., Dimov, I. Faragó, I., Georgiev, K. and Havasi, Á. (2017):** *"Stability of the Richardson Extrapolation combined with some implicit Runge-Kutta methods"*, Journal of Computational and Applied Mathematics, **Vol. 310**, pp. 224-240.

Zlatev, Z., Dimov, I., Faragó, I., Georgiev, K., Havasi, Á. and Ostromsky, Tz. (2011a): *"Implementation of Richardson Extrapolation in the treatment of one-dimensional advection equations"*, In: "Numerical Methods and Applications" (I. Dimov, S. Dimova and N. Kolkovska, eds.), Lecture Notes in Computer Science, Vol. 6046, pp. 198-206, Springer, Berlin.

Zlatev, Z., Dimov, I., Faragó, I., Georgiev, K., Havasi, Á and Ostromsky, Tz. (2011b): *"Solving advection equations by applying the Crank-Nicolson scheme combined with the Richardson Extrapolation"*, International Journal of Differential Equations, Article ID 520840, 16 pages, **http://dx.doi.org/10.1155/2011/520840**.

Zlatev, Z., Dimov, I., Faragó, I., Georgiev, K., Havasi, Á and Ostromsky, Tz. (2011c): *"Solving advection equations by applying the Crank-Nicolson scheme combined with the Richardson Extrapolation (extended version)"*, available online at http://nimbus.elte.hu/~hagi/IJDE/ .

Zlatev, Z., Dimov, I., Faragó, I., Georgiev, K., Havasi, Á and Ostromsky, Tz. (2014): *"Application of Richardson Extrapolation for multi-dimensional advection equations"*, Computers and Mathematics with Applications, **Vol. 67, No. 12**, pp. 2279-2293.

Zlatev, Z., Faragó, I. and Havasi Á. (2010): *"Stability of the Richardson Extrapolation together with the θ-method"*, Journal of Computational and Applied Mathematics, **Vol. 235,** pp. 507-520.

Zlatev, Z., Faragó, I. and Havasi Á. (2012): *"Richardson Extrapolation combined with the sequential splitting procedure and the θ-method"*, Central European Journal of Mathematics, **Vol. 10,** pp. 159-172.

Zlatev, Z., Georgiev, K. and Dimov, I. (2013a): *"Studying absolute stability properties of the Richardson Extrapolation combined with explicit Runge-Kutta methods-extended version"*. Available at the web-site: http://parallel.bas.bg/dpa/BG/dimov/index.html, http://parallel.bas.bg/dpa/EN/publications_2012.htm.

Zlatev, Z., Georgiev, K. and Dimov, I. (2013b): *""Influence of climatic changes on air pollution levels in the Balkan Peninsula"*. Computers and Mathematics with Applications, **Vol. 65, No. 3,** pp. 544-562.

Zlatev, Z., Georgiev, K. and Dimov, I. (2014): *"Studying absolute stability properties of the Richardson Extrapolation combined with explicit Runge-Kutta methods"*, Computers and Mathematics with Applications, **Vol 67, No. 12,** pp.2294-2307.

Zlatev, Z., Havasi, Á. and Faragó, I. (2011): *"Influence of climatic changes on pollution levels in Hungary and its surrounding countries"*. Atmosphere, **Vol. 2,** pp. 201-221.

Zlatev, Z. and Moseholm, L. (2008): *"Impact of climate changes on pollution levels in Denmark"*, Ecological Modelling, **Vol. 217**, pp. 305-319.

Zlatev, Z. and Syrakov, D. (2004a): *"A fine resolution modelling study of pollution levels in Bulgaria. Part 1:* **SO_x** *and* **NO_x** *pollution"*, International Journal of Environment and Pollution, **Vol. 22, No. 1-2,** pp. 186-202.

Zlatev, Z. and Syrakov, D. (2004b): *"A fine resolution modelling study of pollution levels in Bulgaria. Part 2: High ozone levels"*, International Journal of Environment and Pollution, **Vol. 22, No. 1-2**, pp. 203-222.

Zlatev, Z. and Thomsen, P. G. (1979): *"Automatic solution of differential equations based on the use of linear multistep methods",* ACM Transactions of Mathematical Software, **Vol. 5**, pp. 401-414.

List of abbreviations

DIRK Methods	Diagonally Implicit Runge-Kutta Methods
ERKMp	Explicit Runge-Kutta Method of order **p**
ERKMp+R	Explicit Runge-Kutta Method of order **p** combined with the Richardson Extrapolation
FIRK Methods	Fully Implicit Runge-Kutta Methods
LM VSVFM	Linear Multistep Variable Stepsize Variable formula Method
ODEs	Ordinary Differential Equations
PDEs	Partial Differential Equations
PEC Methods	Prediction-Evaluation-Correction Methods (used in relation to the linear multistep methods; indices denoting the order of the number of steps are also used in some cases)
PECE Methods	Prediction-Evaluation-Correction-Evaluation Methods (used in relation to the linear multistep methods; indices denoting the order or the number of steps are also used in some case)
RE	Richardson Extrapolation
UNI-DEM	Unified Danish Eulerian Model (a large-scale air pollution model)
VSVFM	Variable Stepsize Variable Formula Method (used in relation to the linear multistep methods)

Author Index

Subject Index