Chapter 1: Basic Properties of the Richardson

1.1 The initial value problem for systems of ODEs ................................................................. 3
1.2 Numerical treatment of initial value problems for systems of ODEs .................................. 4
1.3 Introduction of the Richardson extrapolation ...................................................................... 5
1.4 Accuracy of the Richardson extrapolation ......................................................................... 6
1.5 Evaluation of the error ......................................................................................................... 7
1.6 Major drawbacks and advantages of the Richardson extrapolation ................................. 8
1.7 Two implementations of the Richardson extrapolation .................................................... 9

Chapter 2: Using Richardson Extrapolation together with Explicit Runge-Kutta Methods ...

2.1 Stability function of one-step methods for solving systems of ODEs ................................. 10
2.2 Stability polynomials of Explicit Runge-Kutta Methods .................................................. 11
2.3 Using Richardson Extrapolation together with the scalar test-problem ............................ 12
2.4 Impact of Richardson Extrapolation on the absolute stability properties ........................ 13
2.4.1 Stability regions related to the first-order one-stage Explicit Runge-Kutta Method ........ 14
2.4.2 Stability regions related to the second-order two-stage Explicit Runge-Kutta Method ... 15
2.4.3 Stability regions related to the third-order three-stage Explicit Runge-Kutta Methods ... 16
2.4.4 Stability regions related to the fourth-order four-stage Explicit Runge-Kutta Methods ... 17
2.4.5 About the use of complex arithmetic in the program for drawing the plots .................... 18
2.5 Preparation of appropriate numerical examples .................................................................. 19
2.5.1 Numerical example with a large real eigenvalue ............................................................. 20
2.5.2 Numerical example with large complex eigenvalues ..................................................... 21
2.5.3 Non-linear numerical example .................................................................................... 22
2.6 Organization of the computations .................................................................................... 23
2.7 Particular numerical methods used in the experiments .................................................... 24
2.8 Numerical results ............................................................................................................... 25
2.9 Development of methods with enhanced absolute stability properties ................................ 26
2.9.1 Derivation of two classes of numerical methods with good stability properties ........ 27
2.9.2 Selecting particular numerical methods for Case 1: $p=3$ and $m=4$ ............................. 28
2.9.3 Selecting particular numerical methods for Case 2: $p=4$ and $m=6$ ............................. 29
2.9.4 Possibilities for further improvement of the results ....................................................... 30
2.10 Major concluding remarks related to Explicit Runge-Kutta Methods ............................ 31

CONTENTS
### Chapter 3: Richardson Extrapolation for implicit methods

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Description of the class of the $\theta$-methods</td>
<td>72</td>
</tr>
<tr>
<td>3.2 Stability properties of the $\theta$-method</td>
<td>73</td>
</tr>
<tr>
<td>3.3 Combining the $\theta$-method with the Richardson Extrapolation</td>
<td>77</td>
</tr>
<tr>
<td>3.4 Stability of the Richardson Extrapolation combined with $\theta$-methods</td>
<td>79</td>
</tr>
<tr>
<td>3.5 The problem of implicitness</td>
<td>88</td>
</tr>
<tr>
<td>3.5.1 Application of the classical Newton iterative method</td>
<td>89</td>
</tr>
<tr>
<td>3.5.2 Application of the modified Newton iterative method</td>
<td>93</td>
</tr>
<tr>
<td>3.5.3 Achieving more efficiency by keeping an old decomposition of the Jacobian matrix</td>
<td>95</td>
</tr>
<tr>
<td>3.5.4 Selecting stopping criteria</td>
<td>97</td>
</tr>
<tr>
<td>3.5.5 Richardson Extrapolation and the Newton Method</td>
<td>101</td>
</tr>
<tr>
<td>3.6 Numerical experiments</td>
<td>102</td>
</tr>
<tr>
<td>3.6.1 Atmospheric chemical scheme</td>
<td>103</td>
</tr>
<tr>
<td>3.6.2 Organization of the computations</td>
<td>106</td>
</tr>
<tr>
<td>3.6.3 Achieving second order of accuracy</td>
<td>108</td>
</tr>
<tr>
<td>3.6.4 Comparison of the $\theta$-method with $\theta=0.75$ and the Backward Differentiation Formula</td>
<td>108</td>
</tr>
<tr>
<td>3.6.5 Comparing the computing times needed to obtain prescribed accuracy</td>
<td>111</td>
</tr>
<tr>
<td>3.6.6 Using the Trapezoidal Rule in the computations</td>
<td>114</td>
</tr>
<tr>
<td>3.7 Some concluding remarks</td>
<td>115</td>
</tr>
</tbody>
</table>

### References

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>119</td>
</tr>
</tbody>
</table>
Chapter 1

Basic Properties of the Richardson Extrapolation

The basic principles, on which the application of the Richardson Extrapolation in the numerical treatment of systems of ordinary differential equations (ODEs) is based, are discussed in this chapter. This powerful device can also be used in the solution of systems of partial differential equations (PDEs). This is often done after the semi-discretization of the systems of PDEs, by which this system is transformed this system into a system of ODEs. This is a very straight-forward approach, but it is based on an assumption that the selected discretization of the spatial derivatives is sufficiently accurate and, therefore, the errors resulting from this discretization 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 be good. Problems will surely arise when the assumption is not satisfied. Then the discretization errors caused by the treatment of the spatial derivatives must 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 to treat 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 this paper.

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 of high order of accuracy are to be selected. It must also be emphasized here that in this section 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 a system 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 systems. However, references to several books, where such details are presented, 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 and, furthermore, that it is possible to apply both constant and variable time-stepsizes. A very general description of the basic properties of these two approaches for solving approximately systems of ODEs is presented and the advantages and the disadvantages of using 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 numerical method for
solving approximately systems of ODEs is presented. The combination of the Richardson Extrapolation with particular numerical methods is studied in the next chapters.

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 is \( p \) where \( p \) is some positive integer with \( p \geq 1 \) then the application of the Richardson Extrapolation results in a new numerical method which is normally of order \( p+1 \). This means that the order of accuracy 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 (by using additionally the Richardson Extrapolation) is discussed in Section 1.5. It is explained there that the obtained error estimation could be used in the attempt to control the time-stepsize (which is important when variable stepsize numerical methods for solving 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 shown there, with carefully chosen examples arising in air pollution modelling, that the stability of the results is a very important issue and the need of numerical methods with good stability properties is again emphasized.

Two implementations of the Richardson Extrapolation are presented in Section 1.7. Some recommendations are given there in connection with the choice of the better implementation in several different cases.

### 1.1. 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 treated numerically. These problems have been studied in detail in many monographs and text-books in which the numerical solution of system of ODEs is handled; for example, in Burrage (1992), Butcher (2003), Hairer, Nørsett and Wanner (1987), Hairer and Wanner (1991), Hundsdorfer and Verwer (2003), Henrici (1968) and Lambert (1991).

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

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

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 the remaining part of this paper),

(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 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

\[
(1.2) \quad y(a) = \eta
\]

is a given vector with s components.

It is well-known that the following theorem, which is related to 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 \( y(t) \) 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 \( f \) is continuous in the whole domain \( D \) and if, furthermore, there exists a positive constant \( L \) such that the following inequality is satisfied:

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

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

**Definition 1.1:** Every constant \( L \) for which the above inequality is fulfilled is called the Lipschitz constant and it is said that function \( f \) from (1.1) satisfies the Lipschitz condition with regard to the dependent variable \( y \) 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 \( y(t) \) 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. This is beyond the scope of this monograph, but this theorem as well as many other related to the existence and the uniqueness of the solution \( y(t) \) results 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 \( y(t) \) that is imposed in Theorem 1.1 is stronger than the requirement that the right-hand-side function \( f \) is continuous for all points \((x, y)\) from the domain \( D \), because the Lipschitz condition (1.3) must additionally be satisfied. On the other side, this requirement is weaker than the requirement that function \( f \) is continuously differentiable for all points \((x, y)\) from the domain \( D \). This means that in Theorem 1.1 it is assumed that the requirement for the right-hand function \( f \) is a little more than continuity, but a little less than differentiability.

\[ \square \]

**Remark 1.2:** If the right-hand side function \( f \) is continuously differentiable with regard to all values of \( y \) in the whole domain \( D \), then the requirement imposed by (1.3) can be satisfied by the following choice of the Lipschitz constant:

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

\[ \square \]

**Remark 1.3:** The problem defined by (1.1) and (1.2) is 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 it is more convenient to consider autonomous problems. The right-hand-side \( f \) does not depend directly of the time-variable \( t \) when the problem is autonomous. 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.

It should be mentioned here that the results presented in this paper 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, 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 \( p \geq 1 \) 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 order \( p \geq 1 \), and after that applying different rules to transform the truncated series in order to obtain the desired numerical method; see, for example, Henrici (1968). By using (1.1) the derivatives of \( y \) can be expressed by derivatives of \( f \) and, when such procedure is applied, one can easily established that it is necessary to assume that all derivatives of function \( f \) up to order \( p - 1 \) 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 also the derivative for function \( f \), which is of order \( p \) is also continuously differentiable. The necessity of introducing stronger assumptions will be further discussed in many sections in the remaining part of this paper, however this problem is not directly related to the implementation of the Richardson Extrapolation and, therefore, it will not be treated in detail.

**Remark 1.6:** Only initial value problems for systems of ODEs will be studied (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 \( \mathbf{y}(t) \) 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:

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

The calculations are carried out step by step. Denote by \( y_0 \) the initial approximation of the solution, i.e. the approximation at \( t_0 = a \). It is often assumed that \( y_0 = y(a) = \eta \). 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 \( y_0 \approx y(a) \).

After providing some appropriate, exact or approximate, value of the initial condition of the system of ODEs, one calculates (by using some computing formula, which is called “numerical method”) successively a sequence of vectors \( y_1 \approx y(t_1), \ y_2 \approx y(t_2) \) 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, at the end of the computational process a set of vectors \( \{y_0, \ y_1, \ldots, y_N\} \) will be produced. These vectors represent approximately the values of the exact solution \( y(t) \) at the selected by (1.6) set of grid-points \( \{t_0, \ t_1, \ldots, t_N\} \).

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

The quantity \( h \) is called stepsizes (the term time-stepsizes will nearly always be used). When \( y_n \) is calculated, the index \( n \) is giving the number of the currently performed steps (the term time-steps will also be used very often). Finally, the integer \( N \) 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 \( t_n, \ (n = 0, 1, 2, \ldots, N) \) are equidistant. The use of equidistant grids is in many cases very convenient, because it is, for example, possible to express in a simple way an arbitrary grid-point \( t_n \) by using the left-hand point of the time-interval \( (t_n = a + nh) \) when such a choice is made. However, it is not necessary to keep the time-stepsizes constant during the whole computational process. Variable stepsizes can also be used. In such a case the grid-points can be defined as follows:
\[ t_0 = a, \quad t_n = t_{n-1} + h_n \quad (n = 1, 2, ..., N), \quad t_N = b. \]

In principle, the time-stepsize \( h_n > 0 \) that is used at time-step \( n \) could always be different both from the time-stepsize \( h_{n-1} \) that was used at the previous time-step and from the time-stepsize \( h_{n+1} \) that will be used at the next time-step. However, some restrictions on the change of the stepsize are nearly always needed in order

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

(b) to ensure zero-stability of the computational process

and

(c) to increase the efficiency by reducing the amount of calculations 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 (1973), 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).

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) and Zlatev (1978, 1983, 1984, 1989).

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

(a) it is easier to establish and 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 appears in the case where some components of the exact solution are quickly varying in some small part of the time-interval \([a, b]\) and slowly varying in the remaining part of this interval. In such a case, one is forced to use the chosen small constant stepsize during the whole computational process, which could be very time-consuming. If it is allowed to vary the time-stepsize, then small time-stepsizes could be used 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. In this way the number of the needed time-steps will often be reduced considerably, which normally will also lead to a very substantial decrease on the computing time for the solution of the problem.

This means that by allowing variations of the time-stepsize, one is trying to avoid the major disadvantage of the device, in which the time-stepsize is kept constant during the whole
computational process, i.e. one can avoid the necessity to apply a very small time-stepsize on the whole interval \([a,b]\). 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).

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. Most of the conclusions will be valid in both cases.

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 applied, will be valid for any 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 stated in the previous section, by an arbitrary numerical method. Assume also that approximations of the exact solution \(y(t)\) are calculated for the values \(t_n\) \((n = 1, 2, ..., N)\) 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 \(t_i\), \((i = 1, 2, ..., n - 1)\) by using some numerical method the order of accuracy of which is \(p\) and, thus, some approximations of the solution at the grid-points \(t_i\), \((i = 0, 1, 2, ..., n - 1)\) are available, then three actions are to be carried out in order to obtain the next approximation \(y_n\):

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

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

(c) Calculate \(y_n\) by applying the formula:

\[
y_n = \frac{2^p w_n - z_n}{2^p - 1}.
\]
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 numerical method for solving systems of ODEs (it is also applicable when systems of PDEs are to be handled). There are only two requirements:

(i) The same numerical method should be used in the calculation of the two approximations \(z_n\) and \(w_n\).

(ii) The order of the selected numerical method should be \(p\). This second requirement is utilized in the derivation of formula (1.8), in which the positive integer \(p\) is involved; see also the next section.

The main properties of the Richardson Extrapolation will be studied in the next sections.

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).

1.4. Accuracy of the Richardson Extrapolation

Assume that the approximations \(z_n\) and \(w_n\) that have been introduced in the previous section were calculated by some numerical method of order \(p\). If we additionally assume that the exact solution \(y(t)\) of the system of ODEs is sufficiently many times differentiable (actually, we have to assume that this function is \(p + 1\) 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:

\[
(1.9) \quad y(t_n) - z_n = h^p K + O(h^{p+1}) ,
\]

\[
(1.10) \quad y(t_n) - w_n = (0.5 h)^p K + O(h^{p+1}) .
\]

The quantity \(K\) 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 \(z_n\) and \(w_n\) and on the particular problem (1.1) – (1.2) that is handled. However, this quantity does not depend on the time-stepsizw \(h\). 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 \( h \) should be replaced by \( h_n \) in the right-hand-sides of both relations).

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

\[
(1.11) \quad y(t_n) - \frac{2^p w_n - z_n}{2^p - 1} = O(h^{p+1}).
\]

Note that the second term in the left-hand-side of (1.11) is precisely the approximation \( y_n \) that was
obtained by the application of the Richardson Extrapolation (see the end of the previous section).

Therefore, the following relationship can be obtained by applying (1.8):

\[
(1.12) \quad y(t_n) - y_n = O(h^{p+1}).
\]

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 \( h \) the approximation \( y_n \) that is
obtained by applying the Richardson Extrapolation will be more accurate than the accuracy of each
of the two approximations \( z_n \) and \( w_n \) obtained when the selected numerical method is used directly.
Indeed, the order of accuracy of \( y_n \) is \( p + 1 \), while each of \( z_n \) and \( w_n \) is of order of accuracy \( p \).

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 the time-stepsize, which
can hopefully be used successfully at 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 \( K \) is in general not
known. This means that one should eliminate this parameter in order to obtain an expression by which
the error can be estimated.

An expression for \( K \) can easily be obtained by subtracting (1.10) from (1.9). The result of this action is
\[(1.13) \quad K = \frac{2^p (w_n - z_n)}{h^p (2^p - 1)} + O(h^{p+1}).\]

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

\[(1.14) \quad y(t_n) - w_n = \frac{w_n - z_n}{2^p - 1} + O(h^{p+1}).\]

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

\[(1.15) \quad ERROR_n = \left| \frac{w_n - z_n}{2^p - 1} \right|.\]

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

\[(1.16) \quad h_{\text{new}} = \omega \sqrt[p]{\frac{ERROR_n}{\text{TOL}}} h\]

can be used in the attempt to calculate a (hopefully) better time-steps for the next time-step.

The parameter \(\text{TOL}\) 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-steps 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-steps. Some additional 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 again be emphasized that the combination of the selected numerical method for solving systems of ODEs with the Richardson Extrapolation can be considered as a new numerical method. Let us now introduce the following two abbreviations:

(i) **Method A**: the original method for solving systems of ODEs and

(ii) **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 what are the main advantages and the drawbacks of Method B when it is compared with Method A.

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. Thus, the amount of computational work will be increased, roughly speaking, with a factor of three when explicit numerical methods are used. If the underlying method, Method A, is implicit, then the situation is much more complicated. We shall postpone the discussion of this case to Chapter 3, where the application of the Richardson Extrapolation for some implicit numerical methods will be studied.

At the same time Method B has also one clear advantage: it is more accurate, because its order of accuracy is by one higher than the order of accuracy of Method A. This means that the results obtained by using Method B will in general be much more precise than those calculated by Method A when the time-stepsize is sufficiently small.

It is necessary to investigate after these preliminary remarks whether the advantage of Method B is giving us 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 in principle allow us to apply bigger time-stepsizes when this method is used and nevertheless to achieve the same or even better accuracy. Denote by \( h_A \) and \( h_B \) the time-stepsize used when Method A and Method B are used and assume that some particular system of ODEs is to be solved. It is quite clear that if \( h_B > 3 \ h_A \), 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 \( h_B > 3 \ h_A \) should in general be replaced with another inequality \( h_B > m \ h_A \) where \( m > 3 \); see more details in Chapter 3).

Assume now that the combined method, Method B, can be used 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 for the solved problem Method B will perform better than Method...
A 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:

\[
\text{Will Method B be more efficient than Method A when realistic problems (say, problems arising in the treatment of some large-scale mathematical models) are solved and, moreover, will this happen also in the more difficult case when the underlying numerical method, Method A, is implicit?}
\]

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 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 3. 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 \( \tau \) should be kept smaller than \( 10^{-3} \) 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 \( \theta \)-method. It is well-known that the computations with Method A are carried out by using the formula:

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

when the \( \theta \)-method is applied. In this experiment \( \theta = 0.75 \) was selected. From (1.17) it is immediately seen that the \( \theta \)-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 \( \theta \) is equal to zero. The \( \theta \)-method will be reduced to the classical Forward Euler Formula (which will be used in Chapter 2) when this value of parameter \( \theta \) is selected.

For Method B the approximations \( z_n \) and \( w_n \) are first calculated by (1.17) 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 \( b = 129600 \) 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 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-steps 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 $h_{\text{ref}} \approx 6.1307634E^{-05}$ 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 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. 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, is much larger when Method B is used.

No more details about the solution procedure are needed here, but much more information can be found in Chapter 3.

Some numerical results are given 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 the two methods are given in this table.

The results shown in Table 1.1 indicate that there exist examples for which Method B is without any doubt more efficient than Method A. However, this is not entirely satisfactory, 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 can become unstable. Moreover, in some cases not only it is not possible to perform the computations with Method B by using a bigger time-stepsize, but runs with the same time-stepsize as that used successfully with Method A will fail when Method B is applied.

<table>
<thead>
<tr>
<th>Compared characteristics</th>
<th>Method A</th>
<th>Method B</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-steps</td>
<td>344064</td>
<td>2688</td>
<td>128</td>
</tr>
<tr>
<td>Computing time</td>
<td>1.1214</td>
<td>0.1192</td>
<td>9.4077</td>
</tr>
</tbody>
</table>

Table 1.1
Numbers of time-steps and computing times (measured in CPU-hours) needed to achieve accuracy $\tau < 10^{-3}$ when Method A (in this experiment the $\theta$-method with $\theta = 0.75$ 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 Extrapolations) 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.

This objection is perfectly correct. In order to demonstrate this fact, let us consider again the \( \theta \)-method defined by (1.17), but this time with \( \theta = 0.5 \). The particular method obtained with this value of parameter \( \theta \) 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 3, but in the context of this section it is not very important). The big problem arises when the \( \theta \)-method with \( \theta = 0.5 \) 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 3, 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 an 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 time-stepsizes, which were used (this will be shown and further explained in Chapter 3).

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 avoid the disaster. However, it is by far 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). 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 topics of the discussion in the following chapters.

1.7. Two implementations of the Richardson Extrapolation

Formula (1.8) is in fact only telling us how to calculate the extrapolated approximation of \( y_n \) at every time-step \( n \) where \( n = 1, 2, ..., N \) under the assumption that the two approximations \( z_n \) and \( w_n \) 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. Full determination of this algorithm will be achieved only when it is clearly stated what will happen when the approximations \( y_n \) for \( n = 1, 2, ..., N \) are obtained. There are at least two possible choices:

(a) the calculated improved approximations \( y_n \) will not participate in the further calculations (they can be stored and used later for other purposes)

and

(b) these approximations can be applied in the further computations.

This leads to two different implementations of the Richardson extrapolation. These implementations are graphically represented in Fig. 1.1 and Fig. 1.2.

The implementation of the Richardson Extrapolation made according to the rule (a), which is shown in Fig. 1.1, 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. 1.2, is called active. It is immediately seen from the plot given in Fig. 1.2 that in this case every improved value \( y_n \) where \( n = 1, 2, ..., N - 1 \) is actively used in the calculations of the next two approximations \( z_{n+1} \) and \( w_{n+1} \).

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 \( n \) will never be used in the consecutive time-steps).
The key question which arise in connection with the two implementations is:

Which of these two rules should be preferred?

There is not an unique answer of this question. Three different situations, the cases (A), (B) and (C), listed below, may arise and should be carefully taken into account:

(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 the Active Richardson Extrapolation. 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 the corresponding Method B, which arises after the application of the Passive Richardson Extrapolation.

In our experiments related to these two implementations (some of them will be presented in Chapter 3), the results obtained when the two implementations are used are quite similar when Case (A) takes place. 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). 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 3.

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.

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, consisting of the selected underlying method and the chosen implementation of the Richardson Extrapolation. Only when this is 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 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 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 implementation is more essential. This requirement is necessary, but in some cases it is not sufficient and, therefore, if this is the true, 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 paper, 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

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

and

(ii) to detect cases where the stability properties of the new numerical methods utilizing the Richardson Extrapolation become poor.
Chapter 2

Using Richardson Extrapolation together with Explicit Runge-Kutta Methods

It is convenient to start the investigation of the efficient implementation of the Richardson Extrapolation with the case where this technique is applied together with Explicit Runge-Kutta Methods. It was mentioned in the previous chapter that such an implementation should be considered as a new numerical method. Assume now that Method A is any numerical algorithm from the class of the Explicit Runge-Kutta method and that Method B is the combination of Method A with the Richardson Extrapolation. If the stability properties (which will be discussed in detail in this chapter) are not causing problems, then Method B can be run with a larger time-stepsize than Method A, because it is, as was shown in the previous chapter, more accurate. This means that in this situation the application of the Richardson Extrapolation will often lead to a more efficient computational process. The problem is that the stability requirement very often put a restriction on the choice of the time-stepsize. Therefore, it is necessary to require that (a) Method A has good stability properties and, moreover, an additional requirement is needed: (b) Method B should have better stability properties than Method A. The computational process will be efficient when both (a) and (b) are satisfied. It will be shown in this chapter that it is possible to satisfy the two requirements for some representatives of the class of the Explicit Runge-Kutta Methods.

In Section 2.1 we shall present several definitions, which are related to the important concept of absolute stability of the numerical methods. These definitions are valid not only for 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 appear when these methods are used to handle the classical scalar test-problem that was introduced by G. Dahlquist in 1963) are presented in the case when the Explicit Runge-Kutta Methods are directly used (i.e. when these numerical methods are applied without using the Richardson Extrapolation).

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

In Section 2.4, the absolute stability regions of the Explicit Runge-Kutta Methods when these are applied directly are compared with the absolute stability regions of the new methods which appear when the Explicit Runge-Kutta Methods are combined with the Richardson Extrapolation. We
assume in this section that the number of stages $m$ is equal to the order of accuracy $p$ of the method. It is verified there that the absolute stability regions of these new numerical methods are always bigger than those of the underlying methods when the assumption $m = p$ is made.

Three appropriate numerical examples are given in Section 2.5. By using these examples it will become possible to demonstrate 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 and, moreover, that this is also true when 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 particular Explicit Runge-Kutta Methods, which are actually applied in the numerical experiments are presented in Section 2.7.

Numerical results, which are obtained during the solution process, are given and discussed in Section 2.8.

Explicit Runge-Kutta methods with enhanced absolute stability properties are derived and tested in Section 2.9. In this section it is assumed that $p < m$ and Explicit Runge-Kutta Methods obtained by using two particular pairs $(m, p) = (4, 3)$ and $(m, p) = (6, 4)$ are studied under the requirement to achieve good 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.

The discussion in Chapter 2 is finished with some 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.

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 $y_n$ of the values of $y(t_n)$ are to be calculated at the grid-points given in (1.6), but note that the assumption for an equidistant grid is done only in order to facilitate and to shorten the presentation of the results; approximations $y_n$ calculated on the grid (1.7) can also be considered in many of the cases treated in this chapter.

One of the most important requirements, which has to be imposed in the attempts to select good and reliable numerical methods and which will in principle ensure reliable and robust treatment of (1.1) and (1.2), can be explained in the following way.
Let us assume that the exact solution \( y(t) \) of the initial value problem defined by (1.1) and (1.2) is bounded. This assumption is not a serious restriction, because such a requirement is very often, practically nearly always, satisfied for practical problems that arise in different fields of science and engineering. When the above assumption for a bounded solution \( y(t) \) of the considered system of ODEs is made, it is very desirable to establish that the following requirement is satisfied:

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

The natural requirement for obtaining a bounded numerical solution, in the case when the exact solution 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 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, Dahlquist suggested in his famous paper from 1963 to use the following scalar test-equation in the stability investigations:

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

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

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

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

It is immediately seen that the exact solution \( y(t) \) given by (2.2) is bounded when the constraint \( \alpha \leq 0 \) that is introduced in (2.1) is satisfied. Therefore, it is necessary to require that the approximate solution computed by the selected numerical method is also bounded.

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 (1992), Butcher (2003), Hairer, Norsett and Wanner (1987), Henrici (1968), and Lambert (1991). Roughly speaking, only the approximation \( y_{n-1} \) of the solution at the grid-point \( t_{n-1} \) is used in the calculation of the approximation \( y_n \) at the next grid-point \( t_n \) of (1.6) when one-step methods are used. A more formal definition can be derived from the definition given on p. 64 in Henrici (1968), however this is not very important for the further discussion 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.

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

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

Approximations of the exact solution $y(t)$ of (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:

$$ y_n = R(v) y_{n-1} = [R(v)]^n y_0, \quad v = \lambda h, \quad n = 1, 2, \ldots $$

The function $R(v)$ 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 3) when implicit one-step methods are used.

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

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

The absolute stability definitions for the scalar 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:

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

It is assumed here that $A \in \mathbb{C}^{sx\times s}$ is a given constant and diagonalizable matrix and that $\eta$ is also some given vector. Under these assumptions, there exists a non-singular matrix $Q$ such that $Q^{-1}AQ = \Lambda$ where $\Lambda$ is a diagonal matrix, whose diagonal elements are the eigenvalues of matrix $A$ from (2.5). Substitute now the expression $y = Q^{-1}z$ in (2.5). The result is:
\[
\frac{dz}{dt} = \Lambda z, \quad t \in [0, \infty], \quad z \in \bar{D} \subset \mathbb{C}^s, \quad s \geq 1, \quad z(0) = \bar{\eta} = Q \eta, \quad \bar{\eta} \in \bar{D}.
\]

It is clear that system (2.6) consists of \( s \) independent scalar equations of type (2.1). Assume that the real parts of all eigenvalues of matrix \( \Lambda \) are non-positive. Assume furthermore that \( \lambda \) is an eigenvalue of matrix \( \Lambda \) for which the relationship \( [\lambda] = \max \{ [\lambda_1], [\lambda_2], \ldots, [\lambda_s] \} \) holds. Finally, set \( v = h\lambda \). Then the application of an arbitrary one-step method in the solution of (2.6), and also of (2.5), will produce a bounded numerical solution when the inequality \( |R(v)| \leq 1 \) is satisfied.

Therefore, it is clear that for some linear systems of ODEs with constant coefficients the absolute stability region is defined precisely in the same way as in the case where the scalar equation (2.1) is considered.

If matrix \( \Lambda \) is not constant, i.e. if \( \Lambda = \Lambda(t) \) and, thus, if the elements of this matrix depend on the time-variable \( t \), 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 \( n \) is an arbitrary positive integer and that a matrix \( \mathbf{A}(t_n) \) where \( t_n \in [t_{n-1}, t_n] \) is involved in the calculation of the approximation \( y_n \approx y(t_n) \) by the selected one-step numerical method. Assume further that matrix \( \mathbf{A}(t_n) \) is diagonalizable. Then some diagonal matrix \( \Lambda(t_n) \) will appear instead of \( \Lambda \) in (2.6). Moreover, the eigenvalues of matrix \( \mathbf{A}(t_n) \) will be the diagonal elements of \( \Lambda(t_n) \). Let \( \bar{\lambda}_n \) be an eigenvalue of matrix \( \mathbf{A}(t_n) \) for which the relationship \( [\bar{\lambda}_n] = \max \{ [\bar{\lambda}_1], [\bar{\lambda}_2], \ldots, [\bar{\lambda}_n] \} \) holds. Assume that \( \lambda \) is chosen so that \( [\lambda] = \max \{ [\bar{\lambda}_1], [\bar{\lambda}_2], \ldots, [\bar{\lambda}_n] \} \). Set \( v = \lambda h \). If the condition \( |R(v)| \leq 1 \) is satisfied for any value of \( t_n \) belonging to (2.3), then one could expect the selected one-step method to be stable. However, it must again be noted that the stability is not guaranteed in this case.

Quite similar considerations can also be applied for the non-linear system described by (1.1) and (1.2). In this case instead of matrix \( \mathbf{A}(t) \) one should consider the Jacobian matrix \( \mathbf{J}(t) \) of function \( \mathbf{f}(t, \mathbf{y}) \) in the right-hand-side of (1.1).

The scalar 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 in this field (see, for example, the remark on page 37 of Hundsdorfer and Verwer, 2003). The above considerations indicate that it is nevertheless 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. After that we shall show that if some methods from this class are combined with the Richardson Extrapolation then the resulting new numerical methods will have increased absolute stability regions. For these new numerical methods it will be possible to apply larger time-stepsizes also in the case where the stability requirements are stronger than the accuracy requirements.
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 textbooks and papers; see, for example, Burrage (1992), Butcher (2003), Hairer, Norsett and Wanner (1987), Henrici (1968), and Lambert (1991). Originally, some particular methods of this type were developed and used (more than hundred years ago) by Kutta (1901) and Runge (1895). The general \( m \)-stage Explicit Runge-Kutta Method is a one-step numerical method for solving systems of ODEs. It is defined by the following formula (more details can be found, when necessary, in any of the above quoted textbooks):

\[
(2.7) \quad y_n = y_{n-1} + \sum_{i=1}^{m} c_i k_i^n .
\]

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

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

with

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

where \( b_{ij} \) are some constants depending on the particular numerical method.

Assume that the order of accuracy of the 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 \( R(v) \) is given by (see Lambert, 1991, p. 202):

\[
(2.10) \quad R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \cdots + \frac{v^p}{p!}, \quad p = m, \quad m = 1, 2, 3, 4 .
\]
Mainly Explicit Runge-Kutta Methods with $p = m$ will be considered in this chapter, but in Section 2.9 some methods with $p < m$ 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 $p$

and

(b) an approximation $y_n$ of the exact value $y(t_n)$ of the solution of (2.1) has to be calculated under the assumption that a sufficiently accurate approximation $y_{n-1}$ 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 applied in the case where the test-problem (2.1), which was proposed by Dahlquist (1963), is solved. The algorithm, by which this can be done, is given below. Note that the relationship (2.4) and, thus, the stability function $R(y)$ is used in the formulation of this algorithm.

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 below shows that the combination of the selected one-step numerical method and the Richardson Extrapolation can also be considered as a one-step numerical method for solving systems of ODEs when it is used to solve the Dahlquist scalar test-example (2.1).
**Step 1** Perform a **large** step time-with a time-step \( h \) by using \( y_{n-1} \) as a starting value to calculate:

\[
(2.11) \quad z_n = R(v) y_{n-1} .
\]

**Step 2** Perform **two small** time-steps with a stepsize \( 0.5h \) by using \( y_{n-1} \) as a starting value in the first of the two small time-steps:

\[
(2.12) \quad \bar{w}_n = R\left(\frac{v}{2}\right) y_{n-1} , \quad w_n = R\left(\frac{v}{2}\right) \bar{w}_n = \left[R\left(\frac{v}{2}\right)\right]^2 y_{n-1} .
\]

**Step 3** Compute (let us repeat here that \( p \) is the order 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:

\[
(2.13) \quad y_n = \frac{2^p w_n - z_n}{2^p - 1} = \frac{2^p \left[R\left(\frac{v}{2}\right)\right]^2 - R(v)}{2^p - 1} y_{n-1} .
\]

Furthermore, it can easily be shown (by applying the same technique as that used in Chapter 1) that the approximation \( y_n \) calculated by (2.13) is usually of order \( p + 1 \) and, therefore, it is more accurate than both \( z_n \) and \( w_n \) when the stepsize is sufficiently small. The most important fact is that the stability polynomial of the combined numerical method is given by:

\[
(2.14) \quad \bar{R}(v) = \frac{2^p \left[R\left(\frac{v}{2}\right)\right]^2 - R(v)}{2^p - 1} .
\]

The above considerations are very general. As we already stated above they are valid when the underlying numerical formula is any explicit one-step numerical method for solving systems of ODEs. However, in the following part of this chapter we shall restrict ourselves to the class of Explicit Runge-Kutta Methods with \( p = m \).

It is necessary now to emphasize the fact that the stability polynomial of the underlying method and its combination with the Richardson Extrapolation, i.e. the polynomials \( R(v) \) and \( \bar{R}(v) \), are...
different, which implies 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 the impact 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 \( p = m \) 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 here that the absolute stability region of a given one-step method consists of all points \( v = h \lambda \) for which the stability function (if the numerical method is explicit the stability function is reduced to a polynomial) satisfies the inequality \(| \mathcal{R}(v) | \leq 1\). If the method is combined with the Richardson Extrapolation, the condition \(| \mathcal{R}(v) | \leq 1\) must be replaced with the stronger requirement \(| \bar{\mathcal{R}}(v) | \leq 1\), which was derived in the previous section; see (2.14). This requirement is indeed stronger, because as mentioned in the end of the previous section these polynomials are different. In the case where a fourth-order four-stage Explicit Runge-Kutta Method is used, the polynomial \( \mathcal{R}(v) \) will be of degree four, while the degree of the corresponding polynomial \( \bar{\mathcal{R}}(v) \) will be eight when this method is combined with the Richardson Extrapolation. The same rule holds in for all Explicit Runge-Kutta Methods: the degree of the polynomial \( \bar{\mathcal{R}}(v) \) is by a factor of two higher than the degree of the corresponding polynomial \( \mathcal{R}(v) \). Therefore, the investigation of the absolute stability regions of the new 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 with \( p = m \) and \( m = 1, 2, 3, 4 \) 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 parts of the boundaries of the absolute stability regions, which are located above the negative real axis and to the left of the imaginary axis are obtained in the following way. Let \( v \) be equal to \( \alpha + \beta i \) with \( \alpha \leq 0 \) and assume that \( \epsilon > 0 \) is some very small increment. Start with a fixed value \( \alpha = 0 \) of the real part of \( v = \alpha + \beta i \) and test the values of the stability polynomial \( \mathcal{R}(v) \) for \( \beta = 0, \epsilon, 2\epsilon, 3\epsilon, ... \) \( \). Continue this process as long as \(| \mathcal{R}(v) | \leq 1\) and denote by \( \beta_0 \) the last value for which the inequality \(| \mathcal{R}(v) | \leq 1\) was satisfied. Set \( \alpha = -\epsilon \) and repeat the same computations for this value of \( \alpha \) and for \( \beta = 0, \epsilon, 2\epsilon, 3\epsilon, ... \) \( \). Denote by \( \beta_\epsilon \) the largest value of \( \beta \) for which the stability requirement \(| \mathcal{R}(v) | \leq 1\) is satisfied. Continuing the computations in this way, it will be possible to calculate the coordinates of a very large set of points \( \{(0, \beta_0), (-\epsilon, \beta_1), (-2\epsilon, \beta_2), ... \}\) in the negative part of the complex plane. 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 $\varepsilon$ is sufficiently small they will be very close to the boundary of the absolute stability region. Therefore, the curve connecting 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 $\varepsilon = 0.001$ 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 $\beta$ of $v = h\lambda = \alpha + \beta i$.

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 $v$ be equal to $\alpha + \beta i$ but assume this time that $\alpha$ is positive. 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 paper). 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.

The same (or at least a 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 with $p = m$, which was described above, is directly applicable for the new methods which arise when any of the Explicit Runge-Kutta Methods with $p = m$ is combined with the Richardson extrapolation. It will only be necessary to replace the stability polynomial $R(v)$ with $\bar{R}(v)$. 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 \( p = m = 1 \):

\[
R(v) = 1 + v .
\]

(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 \( p = m = 1 \) and (2.14) to a stability polynomial of the form:

\[
\bar{R}(v) = 2 \left( 1 + \frac{v}{2} \right)^2 - (1 + v) .
\]

(2.16)

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.

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 \( p = m = 2 \):

\[
R(v) = 1 + v + \frac{v^2}{2!} .
\]

(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 \( p = m = 2 \) and (2.14) to a stability polynomial of the form:

\[
\bar{R}(v) = \frac{4}{3} \left[ 1 + \frac{v}{2} + \frac{1}{2!} \left( \frac{v}{2} \right)^2 \right]^2 - \frac{1}{3} \left( 1 + v + \frac{v^2}{2!} \right) .
\]

(2.18)

The stability regions obtained by using (2.17) and (2.18) and 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 $p = m = 3$:

\[(2.19) \quad R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} .\]

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 $p = m = 3$ and (2.14) to a stability polynomial of the form:
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 the original second-order two-stage Explicit Runge-Kutta Method 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 $p = m = 4$:

$$R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \frac{v^4}{4!}.$$  \hfill (2.21)

The application of the Richardson Extrapolation together with the fourth-order four-stage Explicit Runge-Kutta Method leads according to (2.10) applied with $p = m = 4$ and (2.14) to a stability polynomial of the form:
\( (2.22) \quad \bar{R}(v) = \frac{16}{15} \left[ 1 + \frac{v}{2} + \frac{1}{2!} \left( \frac{v}{2} \right)^2 + \frac{1}{3!} \left( \frac{v}{2} \right)^3 + \frac{1}{4!} \left( \frac{v}{2} \right)^4 \right]^2 
\quad - \frac{1}{15} \left( 1 + \frac{v^2}{2!} + \frac{v^3}{3!} + \frac{v^4}{4!} \right). \)

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.

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

The variable \( R \) (which is the value of the stability polynomial) and \( v \) 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 \( R \) for a given value of \( v \) is completed, the real part \( \bar{A} \) and the imaginary part \( \bar{B} \) of \( R \) can easily be extracted. The numerical method under consideration is stable for the current value of \( v \) if the condition \( \sqrt{\bar{A}^2 + \bar{B}^2} \leq 1 \) is satisfied.

It should be noted that it is also possible to use only real arithmetic in the computer program. If such an approach is for some reasons more desirable than the use of complex arithmetic, then long transformations are to be carried out in order first to obtain directly analytic expressions for \( \bar{A} \) and \( \bar{B} \). After that the condition \( \sqrt{\bar{A}^2 + \bar{B}^2} \leq 1 \) can again be used to check if the method is stable for the current value of \( v \). 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 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 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 dominant, 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-stepsize, 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

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

The elements of matrix $A$ from (2.23) are given below:

\[
(2.24) \quad a_{11} = 741.4, \quad a_{12} = 749.7, \quad a_{13} = -741.7,
\]

\[
(2.25) \quad a_{21} = -765.7, \quad a_{22} = -758, \quad a_{23} = 757.7,
\]

\[
(2.26) \quad a_{31} = 725.7, \quad a_{32} = 741.7, \quad a_{33} = -734,
\]

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

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

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

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

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

\[
(2.30) \quad \mu_1 = -750, \quad \mu_2 = -0.3 + 8i, \quad \mu_3 = -0.3 - 8i.
\]

The absolute value of the real eigenvalue $\mu_1$ is much larger than the absolute values of the two complex eigenvalues of matrix $A$. This means, roughly speaking, that the computations will be stable when $|\mu| = h|\mu_1|$ 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 this paper). In fact, one must require that all three points $h\mu_1, h\mu_2$ and $h\mu_3$ must lie in the absolute stability region of the used method.

The three components of the solution of the example presented in this sub-section are given in Fig. 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 \( h = 10^{-5} \) and the maximal error found during this run was approximately equal to \( 6.63 \times 10^{-4} \).

### 2.5.2. Numerical example with large complex eigenvalues

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

\[
(2.31) \quad \frac{dy}{dt} = A y + b, \quad t \in [0, 13.1072], \quad y = (y_1, y_2, y_3)^T, \quad y(0) = (1, 3, 0)^T,
\]
\[ A \in \mathbb{R}^{3 \times 3}, \quad b = ( -4 e^{-0.3t} \sin 4t, -8 e^{-0.3t} \sin 4t, 4 e^{-0.3t} \sin 4t )^T. \]

The elements of matrix \( A \) from (32) are given below:

\[(2.32) \quad a_{11} = -937.575, \quad a_{12} = 562.425, \quad a_{13} = 187.575, \]
\[(2.33) \quad a_{21} = -187.65, \quad a_{22} = -187.65, \quad a_{23} = 562.35, \]
\[(2.34) \quad a_{31} = -1124.925, \quad a_{32} = 375.075, \quad a_{33} = -375.075. \]

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

\[(2.35) \quad y_1(t) = e^{-750t} \sin 750t + e^{-0.3t} \cos 4t, \]
\[(2.36) \quad y_2(t) = e^{-750t} \cos 750t + 2e^{-0.3t} \cos 4t, \]
\[(2.37) \quad y_3(t) = e^{-750t} (\sin 750t + \cos 750t) - e^{-0.3t} \cos 4t. \]

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

\[(2.38) \quad \mu_1 = -750 + 750i, \quad \mu_2 = -750 - 750i, \quad \mu_3 = -0.3. \]

The absolute value of each of the two complex eigenvalues \( \mu_1 \) and \( \mu_2 \) is much larger than the absolute value of the real eigenvalue \( \mu_3 \). This means that the computations will be stable when \( v = h\mu_1 \) is inside of the absolute stability region of the numerical method under consideration and 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.
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 $h = 10^{-5}$ and the maximal error found during this run was approximately equal to $4.03 \times 10^{-5}$.

2.5.3. Non-linear numerical example

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

\begin{equation}
\frac{dy_1}{dt} = \frac{1}{y_1} - y_2 \frac{e^{t^2}}{t^2} - t,
\end{equation}
(2.40) \[ \frac{dy_2}{dt} = \frac{1}{y_2} - e^{t^2} - 2t e^{-t^2}. \]

The integration interval is \([0.9, 2.21072]\) and the initial values are

(2.41) \[ y_1(0.9) = \frac{1}{0.9}, \quad y_2(0.9) = e^{-0.9^2}. \]

The exact solution is given by

(2.42) \[ y_1(t) = \frac{1}{t}, \quad y_2(0.9) = e^{-t^2}. \]

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) \[ \mu_1 = -\frac{1}{y_1^2}, \quad \mu_2 = -\frac{1}{y_2^2}. \]

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

(2.44) \[ \mu_1(t) = -t^2, \quad \mu_2 = -e^{2t^2}. \]

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.
Plots of the two components of the solution of the system of ODEs defined by (2.39) – (2.41) with \( t \in [0.9, 2.21072] \). 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 \( h = 10^{-6} \) and the maximal error found during this run was approximately equal to \( 2.93 \times 10^{-7} \).

2.6. Organization of the computations

The integration interval, which is \([0, 13.1072]\) for the first two examples and \([0.9, 2.21072]\) for the third one, was divided into 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 \( t_j \), where \( j = 1, 2, \ldots, 128 \), be the end of any of the 128 sub-intervals. Then the following formula is used to evaluate the accuracy achieved by the selected numerical method at this point:
The value of parameter $s$ is 3 in the first two examples, while $s = 2$ is used in the third one. The values $\tilde{y}_{ij} \approx y_i(\bar{t}_j)$ 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 128 sub-intervals mentioned above).

The total error is computed as

\[ \text{ERROR}_j = \sqrt{\frac{\sum_{i=1}^{s} (y_i(\bar{t}_j) - \tilde{y}_{ij})^2}{\max \left( \sqrt{\sum_{i=1}^{s} (y_i(\bar{t}_j))^2} , 1.0 \right)}}. \]

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

The first of the ten runs was carried out by using $h = 0.00512$ and $h = 0.000512$ 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

\[ y_n = y_{n-1} + hf(t_{n-1}, y_{n-1}). \]

When $m$-stage Explicit Runge-Kutta Methods of order $p$ with $p = m$ and $p = 2, 3, 4$ are used, the situation changes. Then for each $p = m = 2, 3, 4$ there exists a large class of Explicit Runge-Kutta Methods. The class depends on one parameter for $p = 2$, while classes dependent on two parameters appear for $p = 3$ and $p = 4$. All methods from such a class have 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, because we were primarily interested in comparing the absolute stability regions 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 class 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 \textbf{second-order two-stage} Explicit Runge-Kutta Methods:

\begin{align}
(2.48) \quad k_1 &= f(t_{n-1}, y_{n-1}), \\
(2.49) \quad k_2 &= f(t_{n-1} + h, y_{n-1} + h k_1), \\
(2.50) \quad y_n &= y_{n-1} + \frac{1}{2} h (k_1 + k_2).
\end{align}

The method selected from the class of the \textbf{third-order three-stage} Explicit Runge-Kutta Methods is defined as follows:

\begin{align}
(2.51) \quad k_1 &= f(t_{n-1}, y_{n-1}), \\
(2.52) \quad k_2 &= f\left(t_{n-1} + \frac{1}{3} h, y_{n-1} + \frac{1}{3} h k_1\right), \\
(2.53) \quad k_3 &= f\left(t_{n-1} + \frac{2}{3} h, y_{n-1} + \frac{2}{3} h k_2\right), \\
(2.54) \quad y_n &= y_{n-1} + \frac{1}{4} h (k_1 + 3k_3).
\end{align}

One of the most popular methods from the class of the fourth-order four-stage Explicit Runge-Kutta Methods is chosen:

\begin{align}
(2.55) \quad k_1 &= f(t_{n-1}, y_{n-1}), \\
(2.56) \quad k_2 &= f\left(t_{n-1} + \frac{1}{2} h, y_{n-1} + \frac{1}{2} h k_1\right), \\
(2.57) \quad k_3 &= f\left(t_{n-1} + \frac{1}{2} h, y_{n-1} + \frac{1}{2} h k_2\right).
\end{align}
\[ (2.58) \quad k_4 = f(t_{n-1} + h, y_{n-1} + hk_3), \]

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

The numerical results, which will be presented in the next section, were obtained by using the above three particular Explicit Runge-Kutta Methods as well as the Forward Euler Formula. More details about the selected by us 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 show 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) 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

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

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 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.
Methods on a SUN computer (quadruple precision being applied in this experiment). 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.A.” means that the numerical method is not stable for the stepsize used. “ERKp”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.

### Table 2.1

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

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”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.

### Table 2.2

<table>
<thead>
<tr>
<th>Run</th>
<th>Stepsize</th>
<th>Steps</th>
<th>ERK1</th>
<th>ERK1+R</th>
<th>ERK2</th>
<th>ERK2+R</th>
<th>ERK3</th>
<th>ERK3+R</th>
<th>ERK4</th>
<th>ERK4+R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00512</td>
<td>2560</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
<tr>
<td>2</td>
<td>0.00256</td>
<td>5120</td>
<td>2.18</td>
<td>145.02</td>
<td>145.02</td>
<td>8.02</td>
<td>8.00</td>
<td>15.98</td>
<td>15.97</td>
<td>32.12</td>
</tr>
<tr>
<td>3</td>
<td>0.00128</td>
<td>10240</td>
<td>2.09</td>
<td>4.00</td>
<td>4.00</td>
<td>7.99</td>
<td>8.00</td>
<td>16.00</td>
<td>16.01</td>
<td>32.04</td>
</tr>
<tr>
<td>4</td>
<td>0.00064</td>
<td>20480</td>
<td>2.04</td>
<td>3.99</td>
<td>3.99</td>
<td>3.99</td>
<td>8.11</td>
<td>7.97</td>
<td>15.99</td>
<td>16.01</td>
</tr>
<tr>
<td>5</td>
<td>0.00032</td>
<td>40960</td>
<td>2.02</td>
<td>4.01</td>
<td>4.01</td>
<td>8.00</td>
<td>8.01</td>
<td>16.07</td>
<td>15.98</td>
<td>32.01</td>
</tr>
<tr>
<td>6</td>
<td>0.00016</td>
<td>81920</td>
<td>2.01</td>
<td>3.98</td>
<td>3.98</td>
<td>8.00</td>
<td>8.02</td>
<td>15.95</td>
<td>16.00</td>
<td>31.98</td>
</tr>
<tr>
<td>7</td>
<td>0.00008</td>
<td>163840</td>
<td>2.01</td>
<td>3.98</td>
<td>3.98</td>
<td>8.00</td>
<td>8.02</td>
<td>15.95</td>
<td>16.00</td>
<td>31.98</td>
</tr>
<tr>
<td>8</td>
<td>0.00004</td>
<td>327680</td>
<td>2.01</td>
<td>4.01</td>
<td>4.01</td>
<td>7.99</td>
<td>7.98</td>
<td>15.97</td>
<td>15.99</td>
<td>31.98</td>
</tr>
<tr>
<td>9</td>
<td>0.00002</td>
<td>655360</td>
<td>1.99</td>
<td>4.00</td>
<td>4.00</td>
<td>8.03</td>
<td>8.00</td>
<td>16.03</td>
<td>16.01</td>
<td>32.03</td>
</tr>
<tr>
<td>10</td>
<td>0.00001</td>
<td>1310720</td>
<td>2.00</td>
<td>3.99</td>
<td>3.99</td>
<td>7.98</td>
<td>8.01</td>
<td>15.98</td>
<td>15.99</td>
<td>31.95</td>
</tr>
</tbody>
</table>

### 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.A.” means that the numerical method is not stable for the stepsize used. “ERKp”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.
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”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.

### Table 2.4

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

### 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”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.

### 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). “ERKp”, \( p = 1, 2, 3, 4 \), means Explicit Runge-Kutta Method of order \( p \). “ERKp+R” refers to the Explicit Runge-Kutta Method of order \( p \) combined with the Richardson Extrapolation.
Several important conclusions can immediately be drawn by investigating carefully the results that are presented in Table 2.1 - Table 2.6:

(a) The non-linear example is in general not causing problems. As should be expected, 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 $t$ the inequality $h|\mu_2(t)| > L$ holds ($L$ being the length of the absolute stability interval on the real axis), and, thus, the stability requirement is not satisfied. It should be noted that the condition $h|\mu_2(t)| \leq L$ is not satisfied for all values of $t$ also 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.

(b) 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 Dahlquist test-example but also for linear systems of ODEs with constant coefficients. This property perhaps holds also for some more general systems of ODEs.

(c) The results show that the calculated (as ratios of two consecutive error estimations) convergence rates of the Runge-Kutta method of order $p$ are about $2^p$ 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 $2^{p+1}$ 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-step size 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 $h|\mu_2(t)| \leq L$ is not satisfied in the very end of the integration interval.

(d) 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 ERK1+R in the table). The error estimation is $2.91 \times 10^{-4}$ for $h = 0.00128$ and when 10240 time-steps are performed. Similar accuracy can be achieved by using 1310720 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 128). Of course, for every step performed by the ERK1 method, the ERK1+R 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 ERK1-R by three), the ERK1-R is reducing the number of time-steps performed by ERK1 by a factor greater than 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.

(e) In this experiment it was illustrative to apply quadruple precision in order to be able 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.

(g) 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 \( y_n \) of the approximate solution is used in the calculation of \( z_n \) and \( w_n \). One can also apply another approach: the values of the previous approximations of \( z_{n-1} \) and \( w_{n-1} \) can be used in the calculation of \( z_n \) and \( w_n \) respectively and after that to calculate the Richardson improvement \( y_n = (2^p w_n - z_n)/(2^p - 1) \). 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). It is clear that if the underlying method is absolutely stable for the two stepsizes \( h \) and \( 0.5h \) then the passive implementation of the Richardson Extrapolation will also be absolutely stable. However, if it is not stable (even only for the large time-steps), then the results calculated by the passive implementation of the Richardson Extrapolation will be unstable. Thus, 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-steps, 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 \( p = m \) was imposed in the previous sections of the second chapter. This requirement is very restrictive, because it can be satisfied only for \( m \leq 4 \). Therefore it is worthwhile to remove this restriction by considering Explicit Runge-Kutta Methods under the condition \( p < m \) and to try to develop numerical methods with enhanced stability properties. When the condition \( p < m \) is imposed, then the stability polynomial given in (2.10) should be replaced with the following formula:
\( R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \frac{v^p}{p!} + \frac{v^{p+1}}{y_{p+1}^{(m,p)}(p+1)!} + \cdots + \frac{v^m}{y_{p+1}^{(m,p)}(m)!} \),

It is seen that there are \( m - p \) free parameters \( y_{p+1}^{(m,p)}, y_{p+2}^{(m,p)}, \ldots, y_{m}^{(m,p)} \) in (2.60). These parameters will be used to search for methods with big absolute stability regions. More precisely, two special cases will be studied in this section:

**Case 1:** \( p = 3 \) and \( m = 4 \)

and

**Case 2:** \( p = 4 \) and \( m = 6 \).

We shall show first that for each of these two cases one can find classes of method with enhanced stability properties. After that we shall select particular methods in each of the obtained classes 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**, i.e. choose \( p = 3 \) and \( m = 4 \). Then (2.60) is reduced to

\[
R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \frac{v^4}{y_4^{(4,3)} 4!}.
\]

A systematic search for methods with good stability properties was carried out by comparing the stability regions obtained for \( y_{4}^{(4,3)} = 1.00(0.01)5.00 \). It is clear that the number of tests, 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 6.00

and

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

In this way the number of test was reduced considerably and it was found that the choice \( y_4^{(4,3)} = 2.4 \) is very good. The absolute stability regions obtained by this value of the free parameter are given in Fig. 2.8.
Figure 2.8

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

Let us call Method A any of the explicit Runge-Kutta method from the class with $p = 3, m = 4$ and $\gamma_4^{(3,4)} = 2.4$ (there exists infinitely many such methods and all of them have the same absolute stability region). 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 statements:

(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 with $= m = 3$. 

53
(c) When Method A is combined with the Richardson Extrapolation then its absolute stability is larger than the corresponding absolute stability region of the combination of the Richardson Extrapolation with the explicit Runge-Kutta method \( p = m = 3 \).

The stability regions could be further enlarged if the second choice \( m - p = 2 \) is made. If \( p = 4 \) and \( m = 6 \) is applied, then the stability polynomial (2.60) can be written as

\[
R(v) = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \frac{v^4}{4!} + \frac{v^5}{5!} Y_5^{(6,4)} + \frac{v^6}{6!} Y_6^{(6,4)}.\]

There are 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 was carried out by using \( Y_5^{(6,4)} = 1.00(0.01)5.00 \) and \( Y_6^{(6,4)} = 1.00(0.01)5.00 \). The number of tests, \( 250000 \), 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 requirements:

(a) the length of the stability interval on the negative part of the real axis should be greater than 12.00

and

(b) the highest point of the absolute stability region should be at a distance not less than 7.00 from the real axis.

In this way the number of tests was reduced very considerably and it was found that the choice \( Y_5^{(6,4)} = 1.42 \) and \( Y_6^{(6,4)} = 4.86 \) gives very good results. The absolute stability regions for the class found with these two values of the free parameters are given in Fig. 2.9.

Let us call Method B any representative of the class of the explicit Runge-Kutta methods determined by choosing: \( p = 4, m = 6, Y_5^{(6,4)} = 1.42 \) and \( Y_6^{(6,4)} = 4.86 \). 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 with \( p = m = 4 \)

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 $p = m = 4$.

**Figure 2.9**

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

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 are given in Table 2.7 together with corresponding absolute stability intervals of their combinations with the Richardson Extrapolation.

It is seen from Table 2.7 that

(a) the length of the absolute stability interval of the new methods, consisting of the combination of any explicit Runge-Kutta method obtained with
\( p = 4, \ m = 6, \ c_5 = 1.42 \) and \( c_6 = 4.86 \) and the Richardson Extrapolation, is more than six times longer than the length of the absolute stability interval of the explicit Runge-Kutta methods with \( p = m = 4 \) when this method is used directly,

(b) it follows from conclusion (a) that for mildly stiff problems (1), 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 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 had to 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 of six will as a rule be a sufficiently good compensation for the use of two more stages.

<table>
<thead>
<tr>
<th>Numerical method</th>
<th>Direct implementation</th>
<th>Combined with Richardson Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p = m = 3 )</td>
<td>2.51</td>
<td>4.02</td>
</tr>
<tr>
<td>( p = 3 ) and ( m = 4 )</td>
<td>3.65</td>
<td>8.93</td>
</tr>
<tr>
<td>( p = m = 4 )</td>
<td>2.70</td>
<td>6.40</td>
</tr>
<tr>
<td>( p = 4 ) and ( m = 6 )</td>
<td>5.81</td>
<td>16.28</td>
</tr>
</tbody>
</table>

**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.

The research for developing Explicit Runge-Kutta Methods with \( p < m \), which have good absolute stability properties when they are combined by 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) one has 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) appropriate numerical experiments have to be carried out in order to verify the usefulness of the results in some realistic applications.
These additional tasks will be further discussed in the next part of this section and in the last section of Chapter 2.

2.9.2. Selecting particular numerical methods for Case 1: \( p = 3 \) and \( m = 4 \)

It was pointed out above that the methods, the absolute stability regions of which 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 here interested to find a method which has good accuracy properties.

The determination of such a particular method among the Explicit Runge-Kutta Methods arising in Case 1 leads to the solution of a non-linear algebraic system of 8 equations with 13 unknowns. These equations are listed below:

\[
\begin{align*}
(2.63) \quad & c_1 + c_2 + c_3 + c_4 = 1, \\
(2.64) \quad & c_2 a_2 + c_3 a_3 + c_4 a_4 = \frac{1}{2}, \\
(2.65) \quad & c_2 (a_2)^2 + c_3 (a_3)^2 + c_3 (a_3)^2 = \frac{1}{3}, \\
(2.66) \quad & c_3 b_{32} a_2 + c_4 (b_{42} a_2 + b_{43} a_3) = \frac{1}{6}, \\
(2.67) \quad & c_4 b_{43} b_{32} a_2 = \frac{1}{\gamma_4^{(4,3)}} \frac{1}{120}, \\
(2.68) \quad & b_{21} = a_2, \\
(2.69) \quad & b_{31} + b_{32} = a_3, \\
(2.70) \quad & b_{41} + b_{42} + b_{43} = a_4.
\end{align*}
\]

The relationships (2.63)-(2.66) are 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 the 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:

\[
\begin{align*}
(2.71) \quad & c_1 = \frac{1}{6}, \quad c_2 = \frac{1}{3}, \quad c_3 = \frac{1}{3}, \quad c_4 = \frac{1}{6}.
\end{align*}
\]
\begin{align*}
(2.72) \quad a_2 &= \frac{1}{2}, \quad a_3 = \frac{1}{2}, \quad a_4 = 1, \\
(2.73) \quad b_{21} &= \frac{1}{2}, \quad b_{31} = 0, \quad b_{32} = \frac{1}{2}, \quad b_{41} = 0, \quad b_{42} = 1 - \frac{1}{2.4}, \quad b_{43} = \frac{1}{2.4}.
\end{align*}

It should be noted that if the last two coefficients $b_{42}$ and $b_{43}$ in (2.73) are replaced with
\begin{equation}
(2.74) \quad b_{42} = 0, \quad b_{43} = 1,
\end{equation}
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)-(2.72) is lower than the order of the classical method (three instead of four), but its absolute stability region is considerably larger. The absolute stability regions of the derived by us method 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 and its combination with the Richardson Extrapolation, which are given in Fig. 2.3).

The ERK43 method was tested by using the first of the three problems presented in Section 2.5. 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 (not 10 as in the previous sections). We are starting with a stepsize $h = 0.02048$ 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 $h = 0.00001$.

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

**Table 2.8**
Comparison of the third-order four stages explicit Runge-Kutta (ERK43) method 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 1.0E+07).
The results presented in Table 2.8 show clearly that following three statements are true:

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The new numerical method (the third-order four stages explicit Runge-Kutta method; <strong>ERK43</strong>) is both more accurate and more stable than the classical third-order three stages explicit Runge-Kutta method (<strong>ERK33</strong>).</td>
</tr>
<tr>
<td>2</td>
<td>The classical fourth-order order four stages explicit Runge-Kutta method, <strong>ERK44</strong>, is more accurate than the new method (which is natural because its order of accuracy is higher), but the new method behaves in a reasonably stable way for $h = 0.00256$ where the classical methods fails.</td>
</tr>
<tr>
<td>3</td>
<td>The combination of the new method with the Richardson extrapolation (<strong>ERK43+RE</strong>) is both more accurate and more stable than the two classical methods (<strong>ERK33</strong> and <strong>ERK44</strong>) and the new method (<strong>ERK43</strong>).</td>
</tr>
</tbody>
</table>

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

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 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**. The equations are listed below. 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:

\[(2.75) \quad c_1 + c_2 + c_3 + c_4 = 1,\]

\[(2.76) \quad c_2 a_2 + c_3 a_3 + c_4 a_4 = \frac{1}{2},\]
\[ (2.77) \quad c_2(a_2)^2 + c_3(a_3)^2 + c_3(a_3)^2 = \frac{1}{3}, \]

\[ (2.78) \quad c_3 b_{32} a_2 + c_4 (b_{42} a_2 + b_{43} a_3) = \frac{1}{6}. \]

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

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

\[ (2.81) \quad c_3 a_3 b_{32} a_2 + c_4 a_4 (b_{42} a_2 + b_{43} a_3) + c_5 a_5 (b_{52} a_2 + b_{53} a_3 + b_{54} a_4) + c_6 a_6 (b_{62} a_2 + b_{63} a_3 + b_{64} a_4 + b_{65} a_5) = \frac{1}{6}, \]

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

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

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

\[ (2.85) \quad b_{21} = a_2 \]

\[ (2.86) \quad b_{31} + b_{32} = a_3 \]

\[ (2.87) \quad b_{41} + b_{42} + b_{43} = a_4 \]

\[ (2.88) \quad b_{51} + b_{52} + b_{53} + b_{54} = a_5 \]

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

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:

60
We shall need for the comparisons of the numerical results a fifth-order Explicit Runge-Kutta method. 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. These conditions are:

\begin{equation}
(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},
\end{equation}

\begin{equation}
(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},
\end{equation}

\begin{equation}
(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},
\end{equation}

\begin{equation}
(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},
\end{equation}

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

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

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

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

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:

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

\[(2.99) \quad c_5, \ c_6, \ a_3, \ a_3, \ b_{32}, \ b_{41}, \ b_{42}, \ b_{52}, \ b_{52}, \ b_{41}, \ b_{42}\]

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 value of the solution vector were taken from the Butcher’s method and extended precision was used (working with 32 digits) during the iterative process. 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:

\[(2.100) \quad c_1 = 0.06636143820913713327361576677234\]
\[(2.101) \quad c_2 = 0.33466439117348386167956841089170\]
\[(2.102) \quad c_3 = 0.06029354106292902784346079863927\]
\[(2.103) \quad c_4 = 0.10534729622111664387002169036336\]
\[(2.104) \quad a_2 = 0.24412763924409282870819068414842\]
\[(2.105) \quad a_4 = 0.58389416084413897975810996900256\]
\[(2.106) \quad a_5 = 0.74232095083880033421170727685848\]
\[(2.107) \quad b_{21} = 0.24412763924409282870819068414842\]
\[(2.108) \quad b_{31} = 0.171875000000000000000000000000\]
\[(2.109) \quad b_{42} = 0.08389416084413897975810996900256\]
\[(2.110) \quad b_{51} = -0.00395725816543771434700055768757\]
\[(2.111) \quad b_{53} = 0.41815320900423804855870783454605\]
(2.112) \( b_{62} = 0.567921736441409352946020215117401 \)

(2.113) \( b_{64} = -1.11004191171206253231847961425022 \)

(2.114) \( b_{65} = 0.68497731815511186000113460593336 \)

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 given in Table 2.9. The corresponding results, obtained by applying the classical ERK44 method and the ERK65B method proposed in Butcher’s book are also presented in Table 2.9. Additionally, results obtained by using the fifth-order six stages explicit Runge-Kutta (ERK65F) method proposed by E. Fehlberg (Fehlberg 1966) are also given in Table 2.9. It should be mentioned that it was establish 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.

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

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).

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:
<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>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 fifth-order methods.</td>
</tr>
<tr>
<td>3</td>
<td>Both the fifth-order six-stages explicit Runge-Kutta method proposed by Butcher, ERK65B, and the fifth-order six-stage explicit Runge-Kutta method 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.</td>
</tr>
<tr>
<td>4</td>
<td>The combination of the new method with the Richardson extrapolation (ERK43+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.</td>
</tr>
<tr>
<td>5</td>
<td>It is not very clear why the numerical error for $h = 0.01024$ is greater than that for $h = 0.02048$ 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 $-13.5$ is very close to the negative part of the real axis and this fact has some influence on the results (because two of the eigenvalues have imaginary parts). When the stepsize becomes bigger, the real part of the largest eigenvalue multiplied by $h$ 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.</td>
</tr>
</tbody>
</table>

### 2.9.4. Possibilities for further improvement of the results

We have already mentioned several times that we decided to derive methods which are close in some sense to a method of a higher order.

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 with the 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). This statement is, of course, based on heuristic assumptions. Nevertheless, the results, which are presented in Table 2.8 and Table 2.9, indicate clearly that the new numerical methods not only have enhanced stability properties, but will also be very accurate.

The question is: is it possible to apply more strict rules in the choice of good explicit Runge-Kutta methods by which to achieve even more accurate results?

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 nine order conditions (2.90)-(2.98). Move the constants from the right-hand-sides of these equalities to the left-hand-sides. Denote by $G_i$, where $i = 1, 2, \ldots, 9$, 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

\[(2.15) \quad G_1 = \left| c_2(a_2)^4 + c_3(a_3)^4 + c_4(a_4)^4 + c_5(a_5)^4 + c_6(a_6)^4 - \frac{1}{5} \right| \]

will be achieved from (2.90) by following the sketched above rules. It is clear how the remaining eight values of the quantities $G_i$ can be obtained from (2.91)-(2.98). Now the following constrained optimization problem can be defined. Find the minimum of the expression:

\[(2.16) \quad \sum_{i=1}^{9} G_i \]

under the assumption that the 15 equalities (2.75)-(2.89) are also satisfied.

It is possible to generalize slightly this idea in the following way. Introduce non-negative weights $w_i$ and minimize

\[(2.17) \quad \sum_{i=1}^{9} w_i G_i \]

again under the assumption that the 15 equalities (2.75)-(2.89) are also satisfied. It is obvious that if all weights are equal to 1, then (2.17) reduces to (2.16).

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 used 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 four order conditions (2.79)-(2.82). Move the constants in the right-hand-sides of these equalities to the left-hand-side. Denote by $F_i$, where $i = 1, 2, 3, 4$, the absolute values of the terms in the left-hand sides of the
obtained after these transformations. As an illustration of the outcome from this process let us point out that

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

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 \( F_i \) can be obtained from (2.80)-(2.82). Now the following constrained optimization problem can be defined. Find the minimum of the expression:

\[(2.119) \quad \sum_{i=1}^{4} F_i \]

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

It is again possible to generalize slightly this idea. Introduce non-negative weights \( v_i \) and minimize

\[(2.120) \quad \sum_{i=1}^{4} v_i F_i \]

under the assumption that the eight equalities (2.63)-(2.79) are also satisfied. It is obvious that if all weights \( v_i \) are set equal to 1, then (2.20) reduces to (2.19).

It must be emphasized that the set of order conditions (2.75)-(2.98) 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 (or some sub-sets of it) has been used in this section to search for good \( \text{ERK}43 \) and \( \text{ERK}64 \) methods, but it can be applied, for example, for designing good \( \text{ERK}63 \) 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 \((Y_4^{(6,3)}, Y_5^{(6,3)}, Y_6^{(6,3)})\), but the search for particular values of these constants which ensure greater absolute stability regions will be much more complicated.

It must also be emphasized that we are not interested so much in finding explicit Runge-Kutta methods with good stability properties, but first and foremost in methods 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, were designed so that their combinations with the Richardson Extrapolation have bigger absolute stability regions then 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.

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 section. Some more general conclusions, 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 the stability properties of the Explicit Runge-Kutta Methods will be drawn below. It was shown in Section 2.4 that the stability regions were increased in the case where the number of stages $m$ is equal to the order of accuracy $p$, however it was also shown in the previous section that this is true for some other classes of Explicit Runge-Kutta Methods.

It is well known that the application of the Richardson Extrapolation leads to an improvement of the accuracy of the underlying numerical method. This statement holds for any numerical method for solving systems of ODEs. The remarkable thing for the class of Explicit Runge-Kutta Methods with $p = m, \ m = 1, 2, 3, 4$ is, as mentioned above, that the application of the Richardson Extrapolation leads to new numerical methods with bigger absolute stability regions. In fact, the results shown in Section 2.4 (and more precisely, 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 $p = m, \ m = 1, 2, 3, 4$ is satisfied (if $p = m = 1$, then there exists only one such method, while large classes of Explicit Runge-Kutta Methods exist for each $p = m, \ when \ p$ is greater than one). Combine the selected method with the Richardson Extrapolation. Then the combined method has always a bigger absolute stability region than that of the underlying Explicit Runge-Kutta Method.

In the previous section we have demonstrated that another, in some sense even stronger, result holds:

**Theorem 2.2:** Let us consider an arbitrary Explicit Runge-Kutta Method, for which the condition $p < m$ and consider the two pairs $(m, p) = (4, 3)$ and $(m, p) = (6, 4)$. Then it is possible to develop Explicit Runge-Kutta Methods with enhanced absolute stability properties. Moreover, the absolute stability properties of their combinations with the Richardson Extrapolation have even bigger absolute stability regions.

The validity of the statement of Theorem 2.2 was verified in Section 2.9.
Finally, in the end of this chapter it should also be emphasized here that non-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 et al. (1987, 2004), Zlatev (1995) and Zlatev and Dimov (2004). 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, Georgiev and Dimov (2013b) and Zlatev, Havasi and Faragó (2011). The advection terms of the air pollution models can be treated with explicit methods; see again Alexandrov et al. (1987, 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.
Chapter 3

Richardson Extrapolation for implicit methods

The application of the Richardson Extrapolation in connection with some implicit methods is discussed in this chapter. Actually, representatives of the well-known \( \theta \)-methods, which were already mentioned in the first chapter, are studied. We are mainly interested in the stability properties of these methods when they are combined with the Richardson Extrapolation. All details that are needed in order to implement efficiently the Richardson Extrapolation for the \( \theta \)-methods are fully explained and it will be easy to apply the same technique in connection with many other implicit numerical methods for solving systems of ODEs.

The \( \theta \)-methods are introduced in Section 3.1. It is explained there that the name “\( \theta \)-method” is often used, but it causes some confusion, because in fact this is not a single numerical scheme, but a class of methods that depend on the parameter \( \theta \), which can freely be varied.

The stability properties of different numerical schemes from the class of the \( \theta \)-methods, which are often used in many applications, are discussed in Section 3.2.

The implementation of the Richardson Extrapolation in combination with the class of the \( \theta \)-methods is described in Section 3.3. The presentation is 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 \( \theta \)-methods are taken into account.

The stability properties of the resulting \textbf{new numerical methods} (the combinations of the numerical schemes from the class of the \( \theta \)-methods with the Richardson Extrapolation) are studied in Section 3.4. It is shown there that the stability properties of the underlying \( \theta \)-methods are not always preserved when these are combined with the Richardson Extrapolation. Some recommendations about the choice of robust and reliable combinations of the Richardson Extrapolation with numerical schemes from the class of the \( \theta \)-methods are given.

The computational difficulties, which arise when numerical schemes belonging to the class of the \( \theta \)-methods are used in the solution of stiff systems of ODEs, are discussed in the next section, Section 3.5. The schemes selected for solving stiff systems of ODEs have necessarily to be implicit and because of this fact, some difficulties must be resolved when they are handled on computers. The problems arising because of the need to apply implicit schemes are fully described and it is explained how to resolve them.

Numerical results are presented in Section 3.6. An atmospheric chemical schemes, which is used in large-scale environmental models, is introduced and used in the numerical experiments. The numerical experiments demonstrate clearly two important facts: (a) the ability of the numerical methods based on the application of the Richardson Extrapolation to preserve the stability of the
computational process (according to the results proven in Section 3.3) and (b) the possibility to achieve higher accuracy when these methods are used.

Several conclusions are given in Section 3.7. Some possibilities for further improvements of the results are also discussed in this section.

3.1. Description of the class of $\theta$-methods

The computations are again 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 is 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 carried out.

In this section and in the whole Chapter 3, the following formula is always used (with some particular value of the parameter $\theta$) in the computational process:

$$y_n = y_{n-1} + h[(1 - \theta)f(t_{n-1}, y_{n-1}) + \theta f(t_n, y_n)] \quad \text{for} \quad n = 1, 2, \ldots, N.$$  \hspace{1cm} (3.1)

The algorithm defined by the above formula is nearly always called the $\theta$-method. However, formula (3.1) shows very clearly that the $\theta$-method is in fact a large class of numerical methods, which depend on the particular parameter $\theta$. We shall sometimes use the traditionally quoted in the literature name “$\theta$-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 name “$\theta$-method” is used.

The class of the $\theta$-methods is normally used with $\theta \in [0, 1]$. The numerical methods, which are obtained for $\theta = 0$, $\theta = 0.5$ and $\theta = 1$, are very popular among scientists and engineers and are very often used in practical computations. The method obtained when $\theta = 0.75$ is specified will also be used in this chapter.

The Forward Euler Formula (which is also well-known as the Explicit Euler Method) is obtained for $\theta = 0$:
(3.2) \[ y_n = y_{n-1} + h f(t_{n-1}, y_{n-1}) \quad \text{for} \quad n = 1, 2, \ldots, N. \]

This numerical scheme is a first-order one-stage Explicit Runge-Kutta Method and it was used in the discussion in Chapter 2. It will not be further discussed in this chapter.

The well-known Trapezoidal Rule is obtained for \( \theta = 0.5 \):

(3.3) \[ y_n = y_{n-1} + 0.5 h [ f(t_{n-1}, y_{n-1}) + f(t_n, y_n) ] \quad \text{for} \quad n = 1, 2, \ldots, N. \]

This rule was mentioned in one numerical example which was presented in Chapter 1. Its order of accuracy is two.

The Backward Differentiation Formula (known also as the Implicit Euler Method) is obtained from (3.1) for \( \theta = 1 \):

(3.4) \[ y_n = y_{n-1} + h f(t_n, y_n) \quad \text{for} \quad n = 1, 2, \ldots, N. \]

The order of accuracy of the Backward Differentiation Formula is only one, but it has very good stability properties.

The Forward Euler Method is explicit, while both the Trapezoidal Rule and the Backward Differentiation Formula are implicit numerical schemes, because the unknown vector \( y_n \) participates both in the left-hand-side and in the right-hand-side of (3.3) and (3.4). In fact, as was mentioned in the beginning of this chapter, the only explicit numerical scheme from the class of the \( \theta \)-methods defined by (3.1) is the Forward Euler Method.

### 3.2. Stability properties of the \( \theta \)-method

It is both relatively easy and very convenient to study, as in the previous chapters, the stability properties of the \( \theta \)-method by the use of the scalar test-problem proposed in Dahlquist (1963):

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

the exact solution of which is given by
\( (3.6) \quad y(t) = \eta e^{\lambda t}, \quad t \in [0, \infty]. \)

It should be mentioned here that, as in the first and in the second chapters, the exact solution \( y(t) \) of equation \( (3.5) \) is a bounded function, because the assumption \( \bar{\alpha} \leq 0 \) is made there.

The application of the numerical algorithms that are defined by \((3.1)\), the numerical algorithms from the class of the \(\theta\)-methods, in the solution of the special scalar test-problem \((3.5)\) leads to a relationship, which is of the same form as that derived in Chapter 2:

\( (3.7) \quad y_n = R(v) y_{n-1} = [R(v)]^n y_0, \quad v = h \lambda, \quad n = 1, 2, \ldots \)

However, the stability function \( R(v) \), is in general not a polynomial as in the previous chapters, but a ratio of two first-degree polynomials, which is given by the following formula:

\( (3.8) \quad R(v) = \frac{1 + (1 - \theta)v}{1 - \theta v}. \)

It is immediately seen, however, that if \( \theta = 0 \), i.e. when the Forward Euler Method is used, then the stability function is reduced to a first-degree polynomial \( R(v) = 1 + v \) 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 \( \theta \neq 0 \). \( R(v) \) is always a rational function for this choice of parameter \( \theta \). In fact, numerical methods, which have good stability properties are obtained when \( \theta \in [0.5, 1.0] \) and it will be assumed in the remaining part of this chapter that \( \theta \) is in this interval.

As in Chapter 2, we can conclude that the numerical solution of \((3.6)\), which is calculated (with given value of the time-stepsize \( h \) and for some particular coefficient \( \lambda \)) by using some numerical scheme from the class of the \(\theta\)-methods, will be bounded when the condition \( R(v) \leq 1 \) is satisfied.

In Chapter 2, we were interested in solving the problem \((3.5)\) in the case where the parameter \( \lambda \) was not very large in absolute value. When this assumption is made, then the problem can be treated numerically with a reasonably large time-stepsize although the absolute stability region of the selected numerical scheme is finite (as were all absolute stability regions presented in Fig. 2.1 – Fig. 2.4, Fig. 2.8 and Fig. 2.9).

Now we shall be interested in the case where \( |\lambda| \) is very large (in which case the problem will normally become stiff). If \( |\lambda| \) is really very large, then it is highly desirable, in fact it is nearly always absolutely necessary, to be able to use a large time-stepsize in the numerical solution of the systems of ODEs, especially when these systems are very large. The requirement of using a large time-stepsize is very strong when at the same time parameter \( |\lambda| \) is very large. This is why it is not sufficient in this situation to search (as in Chapter 2) for finite absolute stability regions that contain all points of
\[ v = \alpha + \beta i \] with \( \alpha \leq 0 \) for which \( R(v) \leq 1 \). Instead of this it much more is reasonable to require that

\[ R(v) \leq 1 \quad \text{for} \quad \forall v = \alpha + \beta i \quad \text{with} \quad \alpha \leq 0. \]  

(3.9)

In other words, we shall now demand that the crucial inequality \( R(v) \leq 1 \) 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 (3.9) 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 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 condition (3.9) and to the conclusion that it is necessary to use implicit method for solving systems of ODEs, the following definition, proposed by G. Dahlquist, can be given.

**Definition 3.1:** It is said that the numerical method for solving systems of ODEs is **A-stable** when the relationship \( R(v) \leq 1 \) is fulfilled for \( \forall v = \alpha + \beta i \) with \( \alpha \leq 0 \) in the case where the numerical method is applied in the solution of the Dahlquist scalar test-example (3.5).

Because of the second Dahlquist barrier, it is clear that **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 will be discussed in Section 3.5).

It can be proved that the \( \theta \)-method is A-stable when \( \theta \in [0.5, 1.0] \), 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 \( \theta \)-methods with \( \theta \) varying in this interval.

We defined the concept of A-stability in connection with the simple scalar equation (3.5). 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 strong. Unfortunately, in some situations even this requirement is not sufficient in the efforts to achieve an efficient computational process. This can be explain as follows. Consider the Trapezoidal rule (3.3). By using (3.7) and (3.8) with \( \theta = 0.5 \) the following relationship can be obtained:

\[ y_n = \frac{1 + 0.5v}{1 - 0.5v} y_{n-1} = \left( \frac{1 + 0.5v}{1 - 0.5v} \right)^n y_0. \]  

(3.10)
Assume further that

(a) $\lambda$ is very large in absolute value negative number,

(b) $h$ is again some fixed positive increment ($h\lambda = v$ being satisfied)

and

(c) $y_0 = 1$ is the initial value of the scalar test-problem (3.5).

Then the exact solution $y(t)$ of (3.5) will very quickly tends to zero. However, if the assumptions (a), (b) and (c) are satisfied, then the last term in (3.10) will tend quickly to zero only when the time-stepsize $h$ is very small, which is clearly not desirable when large-scale scientific 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 (3.10) 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 (a), (b) and (c) hold and if $h$ is fixed, but $|\lambda| \to \infty$, then $|(1 + 0.5v)/(1 - 0.5v)| \to 1$.

This example shows that in some cases the use of the Trapezoidal Rule will not lead to an efficient computational process in spite of the fact that this numerical method is A-stable.

The situation changes completely when the Backward Differentiation Formula is used. Indeed, for $\theta = 1$ formula (3.8) could be rewritten as

$$
(3.11) \quad y_n = \frac{1}{1 - 0.5v} y_{n-1} = \left(\frac{1}{1 - 0.5v}\right)^n y_0
$$

and it is clear now that $|y_n|$ will quickly tend to zero when $n \to \infty$ even for rather large values of the time-stepsize $h$ and, furthermore, also in the case where the above conditions (a) – (c) are satisfied. It is also clear that, assuming once again that the above three assumptions are satisfied, if $h$ is arbitrary large but fixed and if $|\lambda| \to \infty$, then $|1/(1 - 0.5v)| \to 0$ which in most of the cases will be quite satisfactory.

The two examples that are presented by applying the two formulae (3.10) and (3.11) justify the introduction of a new and more restrictive stability definition, the definition for L-stability.

**Definition 3.2:** 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 (3.5), it leads to the relationship (3.7) with $|R(v)| \to 0$ as $\text{Re}(v) \to -\infty$.

The real part of the complex number $v$ is denoted in Definition 3.2 as usual by $\text{Re}(v)$ and it is perhaps worthwhile to reiterate here that $v = \alpha + \beta i$ with $\alpha \leq 0$, see (3.9). This means that $\text{Re}(v) = \alpha$ 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 3.3:** 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 to the Dahlquist scalar test problem (3.5), it leads to the relationship (3.7) with $|R(v)| \to c < 1$ as $\text{Re}(v) \to -\infty$.

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 3.3 with Definition 3.1 and Definition 3.2). It will be shown in the end of this chapter that for some systems of ODEs strongly A-stable methods may even perform 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 (3.1) is strongly A-stable. The Backward Differentiation Formula ($\theta = 1.0$) is L-stable (see more details, for example, in Lambert, 1991).

We are ready now firstly to introduce the Richardson Extrapolation for the class of the $\theta$-methods and after that to give an answer to the important question: are the stability properties of all new methods (the combinations of the $\theta$-methods with the Richardson Extrapolation) preserved?

### 3.3. Combining the $\theta$-method with the Richardson Extrapolation

The Richardson Extrapolation for the class of the $\theta$-methods can be introduced by following closely the rules explained in Section 1.3 (see also Section 2.3). We shall explain the application of the Richardson Extrapolation directly for the case where the Dahlquist scalar test problem (3.5) 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 $\theta$-methods).

Assume that $t_{n-1}$ and $t_n$ are grid-points of the set (1.6) and that $y_{n-1}$ has already been calculated. Three computational steps should successively be carried out by using (3.7) and (3.8) in order to calculate an improved by the Richardson Extrapolation vector $y_n$.

**Step 1:** Perform one large time-step with a time-stepsize $h$ to calculate an approximation $z_n$ of the exact solution $y(t_n)$:

\[
(3.12) \quad z_n = \frac{1 + (1 - \theta)v}{1 - \theta v} y_{n-1}.
\]
Step 2: Perform two small time-steps with a time-stepsize $0.5 \, h$ to calculate another approximation $w_n$ of the exact solution $y(t_n)$:

$$w_n = \left[ \frac{1 + (1 - \theta)(0.5 \, v)}{1 - \theta(0.5 \, v)} \right]^2 y_{n-1}. \tag{3.13}$$

Step 3: Use $z_n$ and $w_n$ to calculate an improved approximation $y_n$ of the exact solution $y(t_n)$ according to the following two rules:

$$y_n = 2\, w_n - z_n \quad \text{when} \quad \theta \neq 0.5 \tag{3.14}$$

and

$$y_n = \frac{4w_n - z_n}{3} \quad \text{when} \quad \theta = 0.5. \tag{3.15}$$

Note that the fact that the $\theta$-method is of first-order of accuracy when $\theta \neq 0.5$ is used in the derivation of (3.14), while the fact that the Trapezoidal Rule, which is obtained when $\theta = 0.5$, is a second-order numerical method, is exploited when (3.15) is obtained. Therefore, it is clearly seen that formulae (3.14) and (3.15) are obtained by using (1.8) with $p = 1$ and $p = 2$ 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. The derivation of the passive implementation of the Richardson Extrapolation in connection with the $\theta$-methods is quite similar: it will only be necessary to use $z_{n-1}$ in (3.12) and $w_{n-1}$ in (3.13) instead of $y_{n-1}$.

The following two relationships can be obtained by inserting the expressions for $z_n$ and $w_n$ from (3.12) and (3.13) in (3.14) and (3.15) respectively:

$$y_n = \left\{ 2 \left[ \frac{1 + (1 - \theta)(0.5 \, v)}{1 - \theta(0.5 \, v)} \right]^2 - \frac{1 + (1 - \theta)v}{1 - \theta v} \right\} y_{n-1} \quad \text{when} \quad \theta \neq 0.5 \tag{3.16}$$

and

$$y_n = \frac{4 \left[ \frac{1 + (1 - \theta)(0.5 \, v)}{1 - \theta(0.5 \, v)} \right]^2 - \frac{1 + (1 - \theta)v}{1 - \theta v}}{3} y_{n-1} \quad \text{when} \quad \theta = 0.5. \tag{3.17}$$
It is immediately seen from (3.16) and (3.17) that the combinations of the Richardson Extrapolation with \( \theta \)-methods are one-step methods (i.e. only the approximation \( y_{n-1} \) is used in the calculation of the improved value \( y_n \)), the stability functions of which are given by the following two expressions:

\[
(3.18) \quad \bar{R}(v) = 2 \left( \frac{1 + (1 - \theta)(0.5 v)}{1 - \theta(0.5 v)} \right)^2 - \frac{1 + (1 - \theta)v}{1 - \theta v} \quad \text{when} \quad \theta \neq 0.5
\]

and

\[
(3.19) \quad \bar{R}(v) = \frac{4 \left( \frac{1 + 0.25 v}{1 - 0.25 v} \right)^2 - \frac{1 + 0.5v}{1 - 0.5v}}{3} \quad \text{when} \quad \theta = 0.5
\]

The stability properties of the new numerical methods that are combinations of the Richardson Extrapolation with \( \theta \)-methods will be studied in the next section.

### 3.4. Stability of the Richardson Extrapolation combined with \( \theta \)-methods

It is necessary to investigate when the application of the Richardson Extrapolation together with different \( \theta \)-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 problems may sometimes arise. More precisely, the following theorem holds; see also Zlatev, Faragó and Havasi (2010):

**Theorem 3.1:** 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 \( \theta \)-methods is strongly A-stable when \( \theta \in [\theta_0, 1] \) with \( \theta_0 = 2/3 \).

**Proof:** According to Definition 3.3 that was given in Section 3.2, 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 \( |R(i\beta)| \leq 1 \) holds for all real values of \( \beta \))

and

(b) \( R(v) \) 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. that, according to Definition 3.3, the following relationship $|R(v)| \to c < 1$ as $\text{Re}(v) \to -\infty$ should be additionally satisfied.

The above analysis indicates that Theorem 3.1 can be proved in three steps:

**Step A:** Prove that the combination of the Richardson Extrapolation with the $\theta$-methods is stable on the imaginary axis.

**Step B:** Show that the stability function $R(v)$ is analytic.

**Step C:** Prove that $|R(v)| \to c < 1$ as $\text{Re}(v) \to -\infty$.

We shall start with Step A.

**Step A – Stability on the imaginary axis**

It is immediately seen that the stability function $R(v)$ from (3.18) can be written in the following form:

$$(3.20) \quad R(v) = \frac{P(v)}{Q(v)},$$

where $P(v)$ is the following polynomial:

$$(3.21) \quad P(v) = 2 \left[ 1 + (1 - \theta)(0.5v) \right]^2 \left( 1 - \theta v \right) - \left[ 1 + (1 - \theta)v \right] \left[ 1 - \theta(0.5v) \right]^2.$$

After some rather long but straight-forward transformations, (3.21) can be rewritten as a third-degree (in $v$) polynomial, which coefficients depend on the particular choice of parameter $\theta$:

$$(3.22) \quad P(v) = (-0.25\theta^3 + 0.75\theta^2 - 0.5\theta)v^3 + (1.25\theta^2 - 2\theta + 0.5)v^2 + (-2\theta + 1)v + 1.$$

The polynomial $Q(v)$ from (3.20) is given by

$$(3.23) \quad Q(v) = [1 - \theta(0.5v)]^2 (1 - \theta v).$$
Also this polynomial can be rewritten as a third-degree (in \( \nu \)) polynomial, which coefficients depend on parameter \( \theta \), however it will be more convenient to use directly (3.23) 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 \( \beta \) from \( \nu = \alpha + i\beta \) the inequality

\[
(3.24) \quad E(\beta) \geq 0
\]

holds.

\( E(\beta) \) is a polynomial, which is defined by

\[
(3.25) \quad E(\beta) = Q(i\beta) Q(-i\beta) - P(i\beta) P(-i\beta).
\]

Consider the first term in the right-hand-side of (3.25). By performing the following successive transformations it can be shown that this term is a sixth-degree polynomial containing only even degrees of \( \beta \):

\[
(3.26) \quad Q(i\beta) Q(-i\beta) = [1 - \theta(0.5i\beta)]^2 (1 - \theta i\beta)[1 + \theta(0.5i\beta)]^2 (1 + \theta i\beta) \\
= [(1 - 0.50i\beta)(1 + 0.50i\beta)]^2 (1 - \theta i\beta)(1 + \theta i\beta) \\
= (1 + 0.25\theta^2\beta^2)^2 (1 + \theta^2\beta^2) \\
= (0.0625\theta^4\beta^4 + 0.50^2\beta^2 + 1)(1 + \theta^2\beta^2) \\
= 0.0625\theta^6\beta^6 + 0.5625\theta^4\beta^4 + 1.50^2\beta^2 \\
= \frac{1}{2^4} (\theta^6\beta^6 + 9\beta^4 + 24\theta^2\beta^2 + 16).
\]

Similar transformations are to be carried out in order to represent also the second term in (3.25), the term \( P(i\beta) P(-i\beta) \), as a sixth-degree polynomial containing only even degrees of \( \beta \). Introduce first the following three constants:

\[
(3.27) \quad A = -0.25 \theta^3 + 0.75 \theta^2 - 0.5 \theta, \quad B = 1.25 \theta^2 - 2 \theta + 0.5, \quad C = -2 \theta + 1.
\]
Now the second term in the right-hand-side of (3.25) can be rewritten in the following form:

\[
(3.28) \quad \mathbf{P}(i\beta) \mathbf{P}(-i\beta) = [\mathbf{A}(i\beta)^3 + \mathbf{B}(i\beta)^2 + \mathbf{C}(i\beta) + 1][\mathbf{A}(-i\beta)^3 + \mathbf{B}(-i\beta)^2 + \mathbf{C}(-i\beta) + 1] \\
= (-Ai\beta^3 - B\beta^2 + C\beta + 1) (Ai\beta^3 - B\beta^2 - C\beta + 1) \\
= A^2\beta^6 + ABi\beta^5 - AC\beta^4 - Ai\beta^3 \\
- ABi\beta^5 + B^2\beta^4 + BCi\beta^3 - B\beta^2 \\
- AC\beta^4 - BCi\beta^3 + C^2\beta^2 + Ci\beta \\
+ Ai\beta^3 - B\beta^2 - C\beta + 1 \\
= A^2\beta^6 - 2AC\beta^4 + B^2\beta^4 - 2B\beta^2 + C^2\beta^2 + 1 \\
= A^2\beta^6 + (B^2 - 2AC)\beta^4 + (C^2 - 2B)\beta^2 + 1.
\]

By using the expressions for \( \mathbf{A} \), \( \mathbf{B} \) and \( \mathbf{C} \) from (3.27) the last equality can be rewritten in the following way:

\[
(3.29) \quad \mathbf{P}(i\beta) \mathbf{P}(-i\beta) = \quad (-0.25\theta^3 + 0.75\theta^2 - 0.5\theta)^2 \beta^6 \\
+ \left[ (1.25\theta^2 - 2\theta + 0.5)^2 - 2 (-0.25\theta^3 + 0.75\theta^2 - 0.5\theta)(-2\theta + 1) \right] \beta^4 \\
+ \left[ (-2\theta + 1)^2 - 2 (1.25\theta^2 - 2\theta + 0.5) \right] \beta^2 \\
+ 1 \\
= \frac{1}{2^4} \left( \theta^6 + 9\theta^4 + 4\theta^2 - 6\theta^5 + 4\theta^4 - 12\theta^3 \right) \beta^6 \\
+ \left[ \frac{1}{2^4} (25\theta^4 - 800\theta^3 + 840\theta^2 - 32\theta + 4) - \theta^4 + 3.5\theta^3 - 3.5\theta^2 + \theta \right] \beta^4 \\
+ (4\theta^2 - 4\theta + 1 - 2.5\theta^2 + 4\theta - 1) \beta^2 \\
+ 1 \\
= \frac{1}{2^4} \left( \theta^6 - 6\theta^5 + 13\theta^4 - 12\theta^3 + 4\theta^2 \right) \beta^6 \\
+ \frac{1}{2^4} (9\theta^4 - 24\theta^3 + 28\theta^2 - 16\theta + 4) \beta^4 
\]
\[ + 1.5 \theta^2 \beta^2 \]
\[ + 1. \]

Everything is prepared now for the determination of the sign of the polynomial \( E(\beta) \) from (3.25). It is necessary to substitute the last terms in the right-hand-sides of (3.26) and (3.29) in (3.25). The result is

\[
(3.30) \quad E(\beta) = \frac{1}{24} (\theta^6 \beta^6 + 9 \theta^4 + 24 \theta^2 \beta^2 + 16) \\
- \frac{1}{24} (\theta^6 - 6 \theta^5 + 13 \theta^4 - 12 \theta^3 + 4 \theta^2) \beta^6 \\
- \frac{1}{24} (9 \theta^4 - 24 \theta^3 + 28 \theta^2 - 16 \theta + 4) \beta^4 \\
- \frac{1}{24} 24 \theta^2 \beta^2 \\
- \frac{1}{24} 16 \\
= \frac{1}{24} (6 \theta^5 - 13 \theta^4 + 12 \theta^3 - 4 \theta^2) \beta^6 + \frac{1}{24} (6 \theta^3 - 7 \theta^2 + 4 \theta - 1) \beta^4. 
\]

It is easily seen that

\[
(3.31) \quad E(\beta) \geq 0 \quad \Rightarrow \quad (6 \theta^5 - 13 \theta^4 + 12 \theta^3 - 4 \theta^2) \beta^2 + 4(6 \theta^3 - 7 \theta^2 + 4 \theta - 1) \geq 0. 
\]

Let us introduce the following two polynomials:

\[
(3.32) \quad H_1(\theta) = 6 \theta^3 - 13 \theta^2 + 12 \theta - 4 \quad \text{and} \quad H_2(\theta) = 6 \theta^3 - 7 \theta^2 + 4 \theta - 1. 
\]

It follows from (3.30) and (3.31) that \( E(\beta) \) will be non-negative for all values of \( \beta \) and for a given value of \( \theta \) if and only if both polynomials from (3.32) are non-negative for the selected value of \( \theta \). It can easily be shown that the inequalities

\[
(3.33) \quad \frac{dH_1}{d\theta} > 0 \quad \text{and} \quad \frac{dH_2}{d\theta} > 0 
\]
hold when $\theta \in [0.5, 1.0]$, which implies that the two polynomials $H_1(\theta)$ and $H_2(\theta)$ are increasing in this interval. Since $H_1(2/3) = 0$ and $H_2(2/3) > 0$, the two polynomials are clearly non-negative for $\theta \in [2/3, 1.0]$ and, therefore, $E(\theta)$ will certainly be non-negative for all values of $\theta$ in the interval $[\theta_0, 1.0]$, where $\theta_0 = 2/3$ is the unique zero of the polynomial $H_1(\theta)$ in the interval $[0.5, 1.0]$.

This completes the proof of the first step of Theorem 3.1, because we have shown that the combinations of the Richardson Extrapolation with numerical schemes from the class of the $\theta$-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 $H_1(\theta)$ and $H_2(\theta)$ are non-negative for $\theta \in [2/3, 1.0]$ is demonstrated graphically on Fig. 3.1.

**Figure 3.1**

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 B – A-stability**

After the proof that the combination of the Richardson Extrapolation with the \( \theta \)-method is stable on the imaginary axis when \( \theta \in [2/3, 1.0] \), it should also be proved that the stability function \( R(v) \) is analytic in \( \mathbb{C}^- \) for these values of \( \theta \). The stability function is, according to equality (3.20), a ratio of the two polynomials \( P(v) \) and \( Q(v) \). It is well-known that polynomials are analytic functions and that a ratio of two polynomials is analytic function in \( \mathbb{C}^- \) if the denominator has no roots in \( \mathbb{C}^- \). In our case, the roots of the denominator \( Q(v) \) of the stability function \( R(v) \) are \( v_1 = 1/\theta \) (a single root) and \( v_{2,3} = 2/\theta \) (a double root). This means that the stability function \( R(v) \) is analytic in \( \mathbb{C}^- \) (because these roots are positive), which completes the proof of Step B.

**Step C: Strong A-stability**

It remains to establish for which values of \( \theta \) in the interval \([2/3, 1.0]\) the relationship \( |R(v)| \to c < 1 \) as \( \text{Re}(v) \to -\infty \) holds. Since \( v = \alpha + \beta i \) with \( \alpha \leq 0 \), it is clear that \( \text{Re}(v) = \alpha \). This fact will be exploited in the proof.

Rewrite first (3.18) as

\[
(3.34) \quad \overline{R}(v) = 2 \left[ \frac{1 + (1 - \theta)(0.5v)}{1 - \theta(0.5v)} \right]^2 - \frac{1 + (1 - \theta)v}{1 - \theta v}
\]

\[
= 2 \left[ \frac{\frac{1}{v} - 0.5 + 0.5\theta}{\frac{1}{v} + 0.5\theta} \right]^2 - \frac{\frac{1}{v} - 1 + \theta}{\frac{1}{v} + \theta}
\]

\[
= 2 \left[ \frac{\frac{1}{-\alpha - \beta i} - 0.5 + 0.5\theta}{\frac{1}{-\alpha - \beta i} + 0.5\theta} \right]^2 - \frac{\frac{1}{-\alpha - \beta i} - 1 + \theta}{\frac{1}{-\alpha - \beta i} + \theta}
\]

Assume now that \( \beta \) is fixed and let \( \alpha = \text{Re}(v) \to -\infty \). The result is:

\[
(3.35) \quad \lim_{\text{Re}(v) \to -\infty} \overline{R}(v) = 2 \left[ \frac{\theta - 1}{\theta} \right]^2 - \frac{\theta - 1}{\theta}
\]

\[
= \frac{\theta^2 - 3\theta + 2}{\theta^2}
\]
Since the terms in the right-hand-side of (3.35) are real, the requirement \(|R(\nu)| \to c < 1\) as \(\text{Re}(\nu) \to -\infty\) reduces to \(|(\theta^2 - 3\theta + 2)/\theta^2| \leq 1\). This inequality implies that the following relationships are satisfied:

\[
(3.36) \quad \frac{\theta^2 - 3\theta + 2}{\theta^2} < 1 \quad \Rightarrow \quad \theta^2 - 3\theta + 2 < \theta^2 \quad \Rightarrow \quad \theta > \frac{2}{3}
\]

and

\[
(3.37) \quad -1 < \frac{\theta^2 - 3\theta + 2}{\theta^2} \quad \Rightarrow \quad 2\theta^2 - 3\theta + 2 > 0.
\]

This completes the proof of the theorem, because the second inequality in (3.37) holds for all real values of \(\theta\) (the minimal value of the polynomial \(2\theta^2 - 3\theta + 2\) is \(7/8\), which is achieved for \(\theta = 3/4\)).

Corollary 3.1: 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 (3.35) is equal to zero when \(\theta = 1.0\) and, thus, the method is L-stable.

Remark 3.1: It is much easier to prove Theorem 3.1 directly for the Backward Differentiation Formula. Indeed, the stability function (3.18) becomes much simpler with \(\theta = 1.0\) and the expressions for \(Q(i\beta) Q(-i\beta)\) and \(P(i\beta) P(-i\beta)\) from (3.26) and (3.29) become also much simpler in this case:

\[
(3.38) \quad Q(i\beta) Q(-i\beta) = 0.0625\beta^6 + 0.5625\beta^4 + 1.5\beta^2 + 1
\]

and

\[
(3.39) \quad P(i\beta) P(-i\beta) = 0.0625\beta^4 + 1.5\beta^2 + 1.
\]
Theorem 3.1 was proved directly for the Backward differentiation Formula in Faragó, Havasi and Zlatev (2010).

**Remark 3.2:** Corollary 3.1 and Remark 3.1 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 3.1, which was proved above.

**Remark 3.3:** Equality (3.35) shows that the constant \( c \) depends on the selected value of parameter \( \theta \). For every value of this parameter, the corresponding value of \( c \) can be calculated by using (3.35). Theorem 3.1 shows that \( c \) is less than one or equal to one for all \( \theta \geq 2/3 \). For example, if \( \theta = 0.75 \), then \( c = 5/9 \).

**Remark 3.4:** Theorem 3.1 cannot be applied directly for the Trapezoidal Rule. The problem is that the stability function from (3.18), which is valid for the case \( \theta \neq 0.5 \) was used in the proof of this theorem. It is necessary to apply the stability function from (3.17), because the Trapezoidal Rule, which is obtained for \( \theta = 0.5 \) from (3.1), is a second-order numerical method. This is done in Theorem 3.2, which is proved below.

**Theorem 3.2:** The combination of the active implementation of the Richardson Extrapolation with the Trapezoidal Rule (i.e. with the \( \theta \)-method with \( \theta = 0.5 \)) is not an A-stable numerical method.

**Proof:** Consider (3.19) and perform the following transformations:

\[
R(v) = \frac{4}{3} \frac{\left(1 + 0.25v\right)^2}{1 - 0.25v} - \frac{1 + 0.5v}{1 - 0.5v}
\]

\[
4 \left(\frac{1 + 0.25}{1 - 0.25}\right)^2 - \frac{1 + 0.5}{1 - 0.5}
\]

\[
= \frac{4}{3} \frac{\left(\frac{1}{v} + 0.25\right)^2}{\frac{1}{v} - 0.25} - \frac{1 + 0.5}{\frac{1}{v} - 0.5}
\]

It is obvious that
\[(3.41) \quad \lim_{v \to \infty} |\tilde{R}(v)| = \frac{5}{3},\]

which means that $|\tilde{R}(v)| > 1$ when $|v|$ 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 perhaps useful to present additionally the following two remarks here:

**Remark 3.5:** It is necessary to explain what is the meaning of $v \to \infty$ when $v$ is a complex number. It is convenient to apply the following definition in this case. If $v \in \mathbb{C}$, then $v \to \infty$ will always mean that $|v|$ grows beyond any assigned positive real number.

**Remark 3.6:** The numerical schemes from the class of the $\theta$-methods have good stability properties when $\theta \in [0.5, 2/3)$. The Trapezoidal Rule, obtained with $\theta = 0.5$, is A-stable, while the numerical methods found when $\theta \in (0.5, 2/3)$ are even strongly A-stable. Unfortunately the good stability properties are sometimes lost when some of these methods are combined with the active implementation of the Richardson Extrapolation; for $\theta \in [0.5, 2/3)$. 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 $\theta$-methods should not be used with $\theta \in [0.5, 2/3)$. 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).

### 3.5. 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 forced to use A-stable, strongly A-stable or L-stable methods in the numerical solution. As stated in the previous sections of this chapter, these methods are necessarily implicit (because of the second Dahlquist barrier). The implicitness of the numerical schemes is very often causing difficulties. This is especially true when combinations of numerical schemes from the class of the $\theta$-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 related to the efficient treatment of the computational process when Richardson Extrapolation is used will be given. Three 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 \( \theta \)-methods will be described. After that the implications which arise when these schemes are combined with the Richardson Extrapolation will be explained.

3.5.1. Application of the classical Newton iterative method

Assume that some numerical scheme from the class of the \( \theta \)-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 has to be solved at every time-step:

\[
(3.42) \quad y_n - h \theta f(t_n, y_n) - g_{n-1} = 0 \quad \text{for} \quad n = 1, 2, \ldots, N.
\]

The solution of (3.42), which in general must be found by solving a large non-linear system of equations, is \( y_n \), while

\[
(3.43) \quad g_{n-1} = -y_{n-1} - h (1 - \theta)f(t_{n-1}, y_{n-1})
\]

is a known vector.

It is clear that (3.42) and (3.43) can easily be obtained by using (3.1).

It is convenient now to introduce the following notation:

\[
(3.44) \quad F(y_n) = y_n - h \theta f(t_n, y_n) - g_{n-1} \quad \text{for} \quad n = 1, 2, \ldots, N,
\]

\[
(3.45) \quad J = \frac{\partial f(t, y)}{\partial y} \quad \text{and} \quad J_n = \frac{\partial f(t_n, y_n)}{\partial y_n} \quad \text{for} \quad n = 1, 2, \ldots, N,
\]

as well as

\[
(3.46) \quad \frac{\partial F(y_n)}{\partial y_n} = I - h \theta J_n \quad \text{for} \quad n = 1, 2, \ldots, N,
\]
where \( I \) is the identity matrix in \( \mathbb{R}^{s \times s} \).

Assume that the classical Newton iterative method is used to solve (approximately, according to some prescribed accuracy) the non-linear system of equations:

\[
F(y_n) = 0,
\]

or, in other words, the Newton iterative method is used to solve the non-linear system of equations

\[y_n - h \theta f(t_n, y_n) - g_{n-1} = 0,\]

which appears when an arbitrary implicit numerical scheme from the class of the \( \theta \)-methods is used with \( \theta \in [0.5, 1.0] \).

The major formulae that are needed at the \( k^{\text{th}} \) 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):

\[
(I - h \theta J_n^{[k-1]}) \Delta y_n^{[k]} = -y_n^{[k-1]} + h \theta f(t_n, y_n^{[k-1]}) + g_{n-1} \quad \text{for} \quad k = 1, 2, ...
\]

\[
y_n^{[k]} = y_n^{[k-1]} + \Delta y_n^{[k]} \quad \text{for} \quad k = 1, 2, ...
\]

Some initial approximation \( y_n^{[0]} \) is needed in order to start the iterative process defined by (3.48) and (3.49). The following two choices are often used in practice:

\[
y_n^{[0]} = y_{n-1}
\]

and

\[
y_n^{[0]} = y_{n-1} + \frac{h_n}{h_{n-1}} (y_{n-1} - y_{n-2}),
\]

where it is assumed that \( h_n \) and \( h_{n-1} \) 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 (3.51) is reduced to

\[
y_n^{[0]} = 2y_{n-1} - y_{n-2},
\]
when \( h_n = h_{n-1} \).

It should be mentioned that (3.51) and (3.52) are used in the experiments, results of which will be reported in the next section.

Consider an arbitrary iteration step \( k \) (\( k = 1, 2, \ldots, \text{end} \)) of the classical Newton iterative process applied in the solution of (3.47). It is also assumed that \( \text{end} \) 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 §3.5.4). When the iterative process is successfully stopped, \( y_n^{[\text{end}]} \) is accepted as a sufficiently good approximation of the exact value \( y(t_n) \) of the solution of (1.1) – (1.2) and \( y_n \) is set equal to \( y_n^{[\text{end}]} \).

The iteration step \( k \) of the Newton iterative process consists of six parts, which must consecutively be performed. The computational 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 \( s \) components of the right-hand-side vector \( f\left(t_n, y_n^{[k-1]}\right) \) of (1.1).

**Part 2 – Jacobian evaluation.** Calculate the elements of the Jacobian matrix \( J_n^{[k-1]} \).

**Part 3 – Factorize the shifted Jacobian matrix \( I - h \theta J_n^{[k-1]} \).** Calculate the elements of the shifted Jacobian matrix and the triangular matrices \( L_n^{[k-1]} \) and \( U_n^{[k-1]} \) such that \( L_n^{[k-1]} U_n^{[k-1]} \approx I - h \theta J_n^{[k-1]} \) 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 \( I - h \theta J_n^{[k-1]} \) when the calculations are carried out on computer. However, \( L_n^{[k-1]} U_n^{[k-1]} \) will normally be a very close approximation of \( I - h \theta J_n^{[k-1]} \). Nevertheless, one should not discard totally the effect of the rounding errors. We shall assume that some care has been taken to reduce or even eliminate the effect of the rounding errors (for example by applying quadruple precision as we did in Chapter 2) and, because of this, shall use \( L_n^{[k-1]} U_n^{[k-1]} = I - h \theta J_n^{[k-1]} \) 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), in order to obtain the solution $\Delta \mathbf{y}^{[k]}_n$ of the system of linear algebraic equations $\mathbf{L}_n^{-1} \mathbf{U}_n^{-1} \Delta \mathbf{y}^{[k]}_n = -\mathbf{y}^{[k-1]}_n + \mathbf{h} \theta \mathbf{f} (\mathbf{t}_n, \mathbf{y}^{[k-1]}_n) + \mathbf{g}_{n-1}$. Also here because of the rounding errors some approximation of the correction vector $\Delta \mathbf{y}^{[k]}_n$ will be obtained but as a rule the calculated vector will be a very close approximation of the exact $\Delta \mathbf{y}^{[k]}_n$. 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 quadruple precision as in Chapter 2).

Part 5 – Update the solution. Use formula (3.49) to calculate the components of vector $\mathbf{y}^{[k]}_n$.

Part 6 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation $\mathbf{y}^{[k]}_n$ is acceptable or not.

Three actions 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 $k$ as $k_{\text{end}}$,

(b) set $\mathbf{y}_n$ is set equal to $\mathbf{y}_{n}^{[k_{\text{end}}]}$ 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 $k := k + 1$

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 either divergent or very slowly convergent, then

(a) set \( k := 1 \),

(b) reduce the time-stepsize \( h \)

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 modifications will be discussed in the following two sections.

### 3.5.2. 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 \( k = 1 \), is carried by Algorithm 1, while the algorithm given below is used in the next iteration steps, i.e. in the iteration steps with \( k > 1 \).

**Algorithm 2: Performing an arbitrary iteration of the modified Newton Method.**

**Part 1 – Function evaluation.** Calculate the \( s \) components of the right-hand-side vector \( f \left( t_n, y_n^{[k-1]} \right) \) 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 \( \Delta y_n^{[k]} \) of the system of linear algebraic equations \( L_n^{[1]} u_n^{[1]} \Delta y_n^{[k]} = -y_n^{[k-1]} + h \theta f \left( t_n, y_n^{[k-1]} \right) + g_{n-1} \).

**Part 3 – Update the solution.** Use formula (3.49) to calculate the components of vector \( y_n^{[k]} \).
Part 4 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation $y_n^{[k]}$ 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 $k$ as $k_{\text{end}}$,

(b) set $y_n$ is set equal to $y_n^{[k_{\text{end}}]}$

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 $k := k + 1$

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 $k > 1$ and if the iterative process is either divergent or very slowly convergent, then

(a) set $k := 1$

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 \( k = 1 \) and if the iterative process is either divergent or very slowly convergent, then

(a) reduce the time-stepsize \( h \)

and

(b) restart the Newton iteration.

The advantages of this algorithm are two: the expensive (in terms of 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). 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 because of the reductions of the numbers of Jacobian evaluations and matrix factorizations are normally a very good compensation for the increase of the numbers of iterations when the solved problems are large.

### 3.5.3. Achieving more 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 Jacobian matrix as long as possible. Let \( j < n \) and \( i \geq 1 \) are 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 \( L_j^{[i]} \) and \( U_j^{[i]} \) of the Jacobian matrix \( I - h \theta J_j^{[i]} \) also when time-step \( n \) 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 \( n \) 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. As mentioned in the previous sub-section, this is especially true when the solved problems are large. Some discussion about the convergence of the Newton iterative process in this case is given in Zlatev (1981a).

The fact that this approach gives often good results explains why it is implemented in many well-known codes for solving large systems of ODEs; see, for example, 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 \( s \) components of the right-hand-side vector \( f(t_n, y_{n[k-1]}^n) \) 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 \( \Delta y_n^{[k]} \) of the system of linear algebraic equations
\[
L^{[i]} U^{[i]} \Delta y_n^{[k]} = -y_n^{[k-1]} + h \theta f(t_n, y_{n[k-1]}^n) + g_{n-1}
\]
where \( i \leq n \) and \( j \geq 1 \).

Part 3 – Update the solution. Use formula (3.49) to calculate the components of vector \( y_n^{[k]} \).

Part 4 – Perform stopping checks. Apply some stopping criteria in order to decide whether the calculated approximation \( y_n^{[k]} \) 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 \( k \) as \( k_{\text{end}} \),

(b) set \( y_n \) is set equal to \( y_n^{[k_{\text{end}}]} \)

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 \( k := k + 1 \)

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 $j < n$ or $j = n$ but $i > 1$, and if the iterative process is either divergent or very slowly convergent, then

(a) set $j: = n$ as well as $i := 1$,

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 stopping criteria, which are not satisfied, if $j = n$ and $i = 1$ and if the iterative process is either divergent or very slowly convergent, then

(a) reduce the time-stepsize $h$

and

(b) restart the Newton iteration.

### 3.5.3. Selecting stopping criteria

By using different stopping criteria in the three algorithms describes in §3.5.1, §3.5.2 and §3.5.3 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 \( y_n^{[k]} \) calculated at iteration \( k \) of the Newton iterative process.

Assume that the accuracy requirement is prescribed by some error tolerance \( TOL \), which is provided by the user (for example, if it is required that the numerical errors are kept less than \( 10^{-3} \) then \( TOL = 10^{-3} \) should be specified). By using the error tolerance \( TOL \) one can try to control, at every iteration step, the accuracy checking whether either

\[
(3.53) \quad \| \Delta y_n^{[k]} \| < TOL \quad \text{for} \quad k = 1, 2, \ldots
\]

or

\[
(3.54) \quad \frac{\| \Delta y_n^{[k]} \|}{\| y_n^{[k]} \|} < TOL \quad \text{for} \quad k = 1, 2, \ldots .
\]

The choice of norm in our opinion is not very important (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 one of these two checks. The absolute check can give problems when \( \| y_n^{[k]} \| \) is large. In such a case the relative stopping check is more preferable. However, the relative check can cause problems when \( \| y_n^{[k]} \| \to 0 \). In such a case the absolute check should be used.

One can try to combine the two check and force the code to select automatically the better check by requiring:

\[
(3.55) \quad \frac{\| \Delta y_n^{[k]} \|}{\max \left( \| y_n^{[k]} \| , 1 \right)} < TOL \quad \text{for} \quad k = 1, 2, \ldots .
\]

It is clear that the check introduced by (3.55) will work as an absolute stopping criterion when \( \| y_n^{[k]} \| < 1 \) and as a relative one otherwise. The check (3.55) is often called mixed stopping criterion. Some positive constant (say, \( c \) ) can be used instead of \( 1 \) in (3.55).
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 $y^{[k]}_n$ 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. An example, the atmospheric chemical scheme used in the Unified Danish Eulerian Model (UNI-DEM, see Zlatev, 1995, or Dimov and Zlatev, 2006), was mentioned in Chapter 1 and will be discussed in detail in the next section. 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 $y^{[k]}_n$ and $\Delta y^{[k]}_n$ are denoted by $y^{[k]}_{nq}$ and $\Delta y^{[k]}_{nq}$ where $q = 1, 2, \ldots, s$. By using this notation, three component-wise stopping criteria, corresponding to the stopping criteria defined by (3.53), (3.54) and (3.55) are given below:

\[(3.56) \quad \max_{q=1, 2, \ldots, s} \left( \frac{\Delta y^{[k]}_{nq}}{|y^{[k]}_{nq}|} \right) < TOL \quad \text{for} \quad k = 1, 2, \ldots,\]

\[(3.57) \quad \max_{q=1, 2, \ldots, s} \left( \frac{|\Delta y^{[k]}_{nq}|}{|y^{[k]}_{nq}|} \right) < TOL \quad \text{for} \quad k = 1, 2, \ldots,\]

\[(3.58) \quad \max_{q=1, 2, \ldots, s} \left( \frac{\Delta y^{[k]}_{nq}}{\max \left( |y^{[k]}_{nq}|, 1 \right)} \right) < TOL \quad \text{for} \quad k = 1, 2, \ldots.\]

Also here some positive constant (say, $c$) can be used instead of $1$.

It should be mentioned here that the check (3.56) 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 (3.56) is a particular norm of this vector.

It should also be mentioned here that the component-wise stopping criteria (3.58) 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 $k_{\text{max}}$ only may be quite inefficient. Assume, for example, that $k_{\text{max}} = 50$ or $k_{\text{max}} = 100$. 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 and to stop the iterations if there is a danger that this will not be the case. Very often this is done by requiring that

$$
\|\Delta y_n^{[k]}\| < \gamma \|\Delta y_n^{[k-1]}\| \quad \text{for} \quad k = 2, 3, \ldots
$$

and stopping the iterative process if this condition is not satisfied at some iteration $k$. Parameter $\gamma$ 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 (3.59) 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 (3.59) 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 (3.59) is often used in this decision, i.e. if this check fails and if an 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 $\theta$-methods with $\theta \in [0, 5, 1, 0]$ 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.
3.5.5. 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 $\theta$-methods with $\theta \in [0.5, 1.0]$ are to be used in the solution of stiff systems of ODEs. However, the difficulties become in general considerably bigger when the $\theta$-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 $\theta$-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, 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 §3.5.1, §3.5.2 and §3.5.3, are used.

Assume first that the classical Newton iterative procedure from §3.5.1 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 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 small stepsize is used), will be rather small (because not the time for performing the iterations but the factorization time is 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 §3.5.2 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 appear when Algorithm 3 from §3.5.3 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 stepsize used in the large time-step and the stepsize used in the two small time-steps, are different). This means that it is not possible to use Algorithm 3 together with Method B.
Therefore, it is time now to point out again that it is not necessary to run the selected 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 \( h_A \) and \( h_B \) 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 by Method B will be comparable to the computing time spent by using Method A if \( h_A \approx 3h_B \) 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. 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 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 \( h_A \approx mh_B \), where \( m > 3 \). Moreover, the factor \( m \) could sometimes be considerably larger than 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).

### 3.6. Numerical experiments

Also in this section we shall use the abbreviations **Method A** for the underlying numerical method (now it will be the selected numerical scheme from the class of the \( \theta \)-methods with some particular value of parameter \( \theta \in [0.5, 1.0] \)) and **Method B** for the new numerical method, obtained when Method A is combined with the Richardson Extrapolation.

It is necessary to demonstrate (by using appropriate numerical experiments) that Method B has the following properties:

- (a) It behaves as a **second-order** numerical method when the stability properties of the underlying numerical scheme, i.e. of Method A, are preserved (this is the case where, according to the results proved in the previous section, when the following relationship \( \theta \in [2/3, 1.0] \) holds),
(b) For some values of $\theta < 1$ the results produced by Method B are more accurate than the results produced by the combination consisting of the Richardson Extrapolation and the Backward Euler Formula (which is obtained when $\theta = 1$),

(c) Method B is often much more efficient than Method A (in terms of the computing time needed to obtain the results) when a prescribed (not too low) accuracy is required,

and

(d) if the conditions of Theorem 3.1 are not satisfied, i.e. if $\theta \in [0.5, 2/3)$, then Method B produces 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. This chemical scheme is further discussed in the following sub-section and, after that, it is used in the calculations, results of which will be presented in this chapter.

3.6.1. Atmospheric chemical scheme

An atmospheric chemical scheme, in which $s = 56$ chemical species are involved, was 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, several radicals and many hydrocarbons). The 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 Memesheimer, Ebel and Roemer, 1997) and in the model system developed and used in Bulgaria (see Syrakov et al., 2011). In all these models the chemical species are 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 because

(a) it is non-linear,

(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 chemical scheme is described is non-linear and stiff implies, as was pointed out in the previous sections, firstly, the use of implicit numerical methods for solving systems of ODEs and secondly, the application of the Newton iterative procedure in the treatment of the arising non-linear system of algebraic equations.

The greatest problem is caused by the fact that the shifted Jacobian matrix, which has to be used in the Newton iterative procedure, is both very ill-conditioned and extremely badly scaled.

The bad scaling and the ill-conditioning of the Jacobian matrix \( J = \frac{df}{dx} \) 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 caused 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 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 demonstrated in Table 3.1. It is seen, for example, that while the concentrations of \( CO \) is about \( 10^{14} \) molecules per cubic centimetre, the corresponding concentrations of \( O^{(1)}D \) remain less than \( 1.3 \times 10^{3} \) molecules per cubic centimetre (i.e. more than eleven 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 \( \text{COND} \) can be used for the condition number calculated at any time-step. The relationship \( \text{COND} \in [4.56 \times 10^{6}, 9.27 \times 10^{12}] \) was established, which shows very clearly that the condition number of the Jacobian matrix \( J = \frac{df}{dt} \) 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 which appear in the periods of changes from day-time to night-time and from night-time to day-time are given in Fig. 3.2 and Fig. 3.3. Also the fact that some of the concentration are decreased during the night, while others are increased in this period is demonstrated in this two figures. Moreover, the changes of the concentrations are very quick and create steep gradients.
The orders of magnitude and the variations of the concentrations of some chemical species during a period of 24 hours (from twelve o’clock at the noon in some given day to the twelve o’clock at the noon in the next day) are shown in this table. The units are (numbers of molecules) / (cubic centimetre).

<table>
<thead>
<tr>
<th>Chemical species</th>
<th>Starting concentration</th>
<th>Minimal concentration</th>
<th>Maximal concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>$10^{10}$</td>
<td>$3.6 \times 10^{3}$</td>
<td>$1.8 \times 10^{10}$</td>
</tr>
<tr>
<td>NO$_2$</td>
<td>$10^{11}$</td>
<td>$8.8 \times 10^{8}$</td>
<td>$10^{11}$</td>
</tr>
<tr>
<td>CO</td>
<td>$3.8 \times 10^{10}$</td>
<td>$3.7 \times 10^{14}$</td>
<td>$3.8 \times 10^{14}$</td>
</tr>
<tr>
<td>O$^{(1)}$D</td>
<td>$10^{3}$</td>
<td>$2.1 \times 10^{-43}$</td>
<td>$1.3 \times 10^{3}$</td>
</tr>
</tbody>
</table>

**Table 3.1**

**Figure 3.2**

Diurnal variation of the concentrations of the chemical species OP.
3.6.2. 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 briefly discussed in Chapter 1 and in Chapter 2. However, because of the implicitness of the applied in this chapter numerical methods, a more detailed description is needed here. Such description will be given in this section.

The atmospheric chemical scheme, which was discussed in the previous section was treated numerically on the time-interval \([a, b] = [43200, 129600]\). The value \(a = 43200\) corresponds to twelve o’clock at the noon (measured in seconds and starting from mid-night, while \(b = 129600\) 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 is 24 hours and it contains important changes from day-time to the night-time and from the 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).

![Figure 3.3](image)

Diurnal variation of the concentrations of the chemical species \(\text{NO}_3\).
Several experiments were run and some of the results will be presented in this chapter. In each experiment the first run is performed by using \( N = 168 \) time-steps, which means that the time-steps is \( h \approx 514.285 \) seconds. After each run the time-steps is halved (which means that the number of time-steps is doubled). This action is repeated eighteen times. The behaviour of the error made in this rather long sequence of nineteen runs is studied. The error made at time \( \tilde{t}_j \) in any of the nineteen runs is measured in the following way. Denote:

\[
(3.60) \quad \text{ERROR}_j = \max_{i=1,2, \ldots, 56} \left( \frac{|y_{ji} - y_{ji}^{\text{ref}}|}{\max(|y_{ji}^{\text{ref}}|, 1.0)} \right),
\]

where \( y_{ji} \) and \( y_{ji}^{\text{ref}} \) are the calculated value and the reference solution (the meaning of the term “reference solution” was explained in Chapter 1) of the \( i^{\text{th}} \) chemical species at time \( \tilde{t}_j = t_0 + jh_0 \) (where \( j = 1, 2, \ldots, 168 \) and \( h_0 \approx 514.285 \) is the time-steps that has been used in the first run). As was mentioned in the first chapter, the reference solution was calculated by using a three-stage fifth-order L-stable fully implicit Runge-Kutta algorithm (see Butcher, 2003 or Hairer and Wanner, 1991) with \( N = 998244352 \) time-steps and a time-steps \( h_{\text{ref}} \approx 6.1307634 \times 10^{-5} \).

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 more important, however, that errors of the calculated approximations was estimated at the same 168 grid points in all nineteen runs.

The global error made during the computations over the whole time-interval is estimated by using the following formula:

\[
(3.61) \quad \text{ERROR} = \max_{j=1,2, \ldots, 168} \left( \text{ERROR}_j \right).
\]

It is highly desirable to eliminate the influence of the rounding errors when the quantities involved in (3.42) and (3.43) 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 56 chemical species on the time-interval $[a, b] = [43200, 129600]$ vary in the range $[4.56 \times 10^8, 9.27 \times 10^{12}]$. 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 $\theta$-methods are developed and used and, furthermore, to show that the rounding errors do not affect the accuracy of the results.

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 demonstrate the advantages of the application of Richardson Extrapolation.

### 3.6.3. Achieving second order of accuracy

Numerical results, which are obtained by using numerical schemes belonging to the class of the $\theta$-methods in combination with the Richardson Extrapolation are given in Table 3.2. The value $\theta = 0.75$ is selected, which means that the relationship $|R(v)| \to c < 1$ as $\text{Re}(v) \to -\infty$ holds with $c = 5/9$, see (3.35). The results in Table 3.2 show clearly that the $\theta$-method with $\theta = 0.75$ 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 the time-stepsize by a factor of two leads to an increase of the accuracy by a factor of two when the $\theta$-method with $\theta = 0.75$ 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.
3.6.4. Comparison of the $\theta$-method with $\theta = 0.75$ and the Backward Differentiation Formula

It can theoretically be justified that the $\theta$-method with $\theta = 0.75$ should very often give more accurate results than the Backward Differentiation Formula. More precisely, the following theorem holds:

**Theorem 3.3:** The principal part of the local truncation error of the $\theta$-method with $\theta = 0.75$ is twice smaller than that of the Backward Euler Formula.

**Proof:** Consider two approximations $y_n^{\text{backward}}$ 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 Differentiation Formula and the $\theta$-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 $y_n^{\text{backward}}$ and $y_n^{\theta=0.75}$, can be written in the following form:

$$(3.63) \quad y_n^{\text{backward}} - y_{n-1} - h f(t_n, y_n^{\text{backward}}) = 0,$$

and

$$(3.64) \quad y_n^{\theta=0.75} - y_{n-1} - 0.25 h f(t_{n-1}, y_{n-1}) - 0.75 h f(t_n, y_n^{\theta=0.75}) = 0.$$

Replace:

(a) $y_n^{\text{backward}}$ 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 (3.63) and (3.64).

Use the relationship $\frac{dy(t)}{dt} = f(t, y(t))$ and introduce, as on p. 48 in Lambert (1991), **linear difference operators** to express the fact that the right-hand-sides of the expressions obtained from (3.63) and (3.64) will not be equal to zero after the above substitutions are made. The following two relationships can be obtained when these actions are performed:
<table>
<thead>
<tr>
<th>Job Number</th>
<th>Number of time-steps</th>
<th>Direct use of the θ-method</th>
<th>Richardson Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accuracy</td>
<td>Rate</td>
</tr>
<tr>
<td>1</td>
<td>168</td>
<td>1.439E-00</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>336</td>
<td>6.701E-01</td>
<td>2.147</td>
</tr>
<tr>
<td>3</td>
<td>672</td>
<td>3.194E-01</td>
<td>2.098</td>
</tr>
<tr>
<td>4</td>
<td>1344</td>
<td>1.550E-01</td>
<td>2.060</td>
</tr>
<tr>
<td>5</td>
<td>2688</td>
<td>7.625E-02</td>
<td>2.033</td>
</tr>
<tr>
<td>6</td>
<td>5376</td>
<td>3.779E-02</td>
<td>2.018</td>
</tr>
<tr>
<td>7</td>
<td>10752</td>
<td>1.881E-02</td>
<td>2.009</td>
</tr>
<tr>
<td>8</td>
<td>21504</td>
<td>9.385E-02</td>
<td>2.005</td>
</tr>
<tr>
<td>9</td>
<td>43008</td>
<td>4.687E-03</td>
<td>2.002</td>
</tr>
<tr>
<td>10</td>
<td>86016</td>
<td>2.342E-03</td>
<td>2.001</td>
</tr>
<tr>
<td>11</td>
<td>172032</td>
<td>1.171E-03</td>
<td>2.001</td>
</tr>
<tr>
<td>12</td>
<td>344064</td>
<td>5.853E-04</td>
<td>2.000</td>
</tr>
<tr>
<td>13</td>
<td>688128</td>
<td>2.926E-04</td>
<td>2.000</td>
</tr>
<tr>
<td>14</td>
<td>1376256</td>
<td>1.463E-04</td>
<td>2.000</td>
</tr>
<tr>
<td>15</td>
<td>2752512</td>
<td>7.315E-05</td>
<td>2.000</td>
</tr>
<tr>
<td>16</td>
<td>5505024</td>
<td>3.658E-05</td>
<td>2.000</td>
</tr>
<tr>
<td>17</td>
<td>11010048</td>
<td>1.829E-05</td>
<td>2.000</td>
</tr>
<tr>
<td>18</td>
<td>22020096</td>
<td>9.144E-06</td>
<td>2.000</td>
</tr>
<tr>
<td>19</td>
<td>44040192</td>
<td>4.572E-06</td>
<td>2.000</td>
</tr>
</tbody>
</table>

**Table 3.2**

Numerical results that are obtained (a) in nineteen runs, in which the direct implementation of the θ-method with θ = 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 θ = 0.75 is applied. The errors obtained by using formula (3.61) are given in the columns under “Accuracy”. The ratios of two successive errors (the convergence rates) are given in the columns under “Rate”.

\[
L^{\text{backward}}[y(t_n); h] = y(t_n) - y(t_{n-1}) - h \frac{dy(t_n)}{dt} \quad (3.65) 
\]

and

\[
L^{\theta=0.75}[y(t_n); h] = y(t_n) - y(t_{n-1}) - 0.25 h \frac{dy(t_{n-1})}{dt} - 0.75 h \frac{dy(t_n)}{dt} \quad (3.66) 
\]

Expanding \( y(t_n) \) and \( \frac{dy(t_n)}{dt} \) in Taylor series about \( t_{n-1} \) and keeping the terms containing \( h^2 \) one can rewrite (3.65) and (3.66) in the following way:
The terms in the right-hand-sides of (3.67) and (3.68) 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 \( \theta \)-method applied with \( \theta = 0.75 \) is twice smaller than that of the Backward Euler Formula. This completes the proof of the theorem.

\[ L^{\text{backward}}[y(t_n); h] = -\frac{h^2}{2} \frac{d^2y(t_{n-1})}{dt^2} + O(h^3) \]

and

\[ L^{\theta=0.75}[y(t_n); h] = -\frac{h^2}{4} \frac{d^2y(t_{n-1})}{dt^2} + O(h^3). \]

Theorem 3.3 demonstrates very clearly the fact that one should expect, as stated above, the \( \theta \)-method with \( \theta = 0.75 \) to be more accurate than the Backward Differentiation Formula.

Several experiments were carried out to confirm this expectation. Some of the obtained results are shown in Table 3.3. It is seen that accuracy of the results obtained by using the \( \theta \)-method with \( \theta = 0.75 \) 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 3.3).

It is remarkable that the accuracy is improved precisely by a factor of two when the time-stepsize becomes sufficiently small.

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. with the Backward Differentiation Formula and with the \( \theta \)-method with \( \theta = 0.75 \)). Probably the same approach (or at least a similar approach) as that which was used in Theorem 3.3 can be applied to compare the leading terms of the local truncation error also in this case.

The results presented in Table 3.3 show that the accuracy of the calculated approximations is in general improved by a factor, which is greater than two, when the \( \theta \)-method with \( \theta = 0.75 \) is used as an underlying method instead of the Backward Differentiation Formula.

### 3.6.5. 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 3.2 and Table 3.3). 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 $\theta$-method with $\theta = 0.75$ is applied directly

and

(b) when the same numerical method is combined with the Richardson Extrapolation.

<table>
<thead>
<tr>
<th>Job Number</th>
<th>Number of time-steps</th>
<th>Backward Euler Formula</th>
<th>The $\theta$-method with $\theta = 0.75$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Direct</td>
<td>Richardson</td>
</tr>
<tr>
<td>1</td>
<td>168</td>
<td>2.564E-00</td>
<td>3.337E-01</td>
</tr>
<tr>
<td>2</td>
<td>336</td>
<td>1.271E-00</td>
<td>1.719E-01</td>
</tr>
<tr>
<td>3</td>
<td>672</td>
<td>6.227E-01</td>
<td>5.473E-02</td>
</tr>
<tr>
<td>4</td>
<td>1344</td>
<td>3.063E-01</td>
<td>7.708E-03</td>
</tr>
<tr>
<td>5</td>
<td>2688</td>
<td>1.516E-01</td>
<td>1.960E-03</td>
</tr>
<tr>
<td>6</td>
<td>5376</td>
<td>7.536E-02</td>
<td>5.453E-04</td>
</tr>
<tr>
<td>7</td>
<td>10752</td>
<td>3.757E-02</td>
<td>1.455E-04</td>
</tr>
<tr>
<td>8</td>
<td>21504</td>
<td>1.876E-02</td>
<td>3.765E-05</td>
</tr>
<tr>
<td>9</td>
<td>43008</td>
<td>9.371E-03</td>
<td>9.583E-06</td>
</tr>
<tr>
<td>10</td>
<td>86016</td>
<td>4.684E-03</td>
<td>2.418E-06</td>
</tr>
<tr>
<td>11</td>
<td>172032</td>
<td>2.341E-03</td>
<td>6.072E-07</td>
</tr>
<tr>
<td>12</td>
<td>344064</td>
<td>1.171E-03</td>
<td>1.522E-07</td>
</tr>
<tr>
<td>13</td>
<td>688128</td>
<td>5.853E-04</td>
<td>3.809E-08</td>
</tr>
<tr>
<td>14</td>
<td>1376256</td>
<td>2.926E-04</td>
<td>9.527E-09</td>
</tr>
<tr>
<td>15</td>
<td>2752512</td>
<td>1.463E-04</td>
<td>2.382E-09</td>
</tr>
<tr>
<td>16</td>
<td>5505024</td>
<td>7.315E-05</td>
<td>5.957E-10</td>
</tr>
<tr>
<td>17</td>
<td>11010048</td>
<td>3.658E-05</td>
<td>1.489E-10</td>
</tr>
<tr>
<td>18</td>
<td>22020096</td>
<td>1.829E-05</td>
<td>3.720E-11</td>
</tr>
<tr>
<td>19</td>
<td>44040192</td>
<td>9.144E-06</td>
<td>9.273E-12</td>
</tr>
</tbody>
</table>

Table 3.3
Comparison of the accuracy achieved when the Backward Differentiation Formula (obtained by using $\theta=1.0$) and the $\theta$-method with $\theta = 0.75$ are run with 19 different time-steps. The errors obtained by (3.61) are given in the last four columns in this table. The ratios (the errors obtained when the $\theta$-method with $\theta = 0.75$ is used divided by the corresponding errors obtained when the Backward Differentiation Formula is used) are given in brackets.
The computing times needed in the efforts to achieve prescribed accuracy are given in Table 3.3. If the desired accuracy is \(10^{-k}\) \((k = -1, -2, \ldots, -11)\), then the computing times achieved in the first run in which the quantity \(\text{ERROR}\) from (3.43) becomes less than \(10^{-k}\) are given in Table 3.4. This means that the actual error, found in this way, is in the interval \([10^{-(k+1)}, 10^{-k}]\) when accuracy of order \(10^{-k}\) is required.

Four important conclusions can immediately be drawn by studying the numerical results that are shown in Table 3.4:

- The direct use of the \(\theta\)-method with \(\theta = 0.75\) 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 \(\text{ERROR}\) from (3.61) should be in the interval \([10^{-2}, 10^{-1}]\), compare the CPU times in the first row of Table 3.4.

<table>
<thead>
<tr>
<th>Desired Accuracy of the calculated approximations</th>
<th>Application of the (\theta)-method with (\theta=0.75)</th>
<th>Combination with the Richardson Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>([1.0E-02, 1.0E-01])</td>
<td>CPU time (in hours) 0.0506</td>
<td>Number of the time-steps 2688</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU time (in hours) 0.0614</td>
</tr>
<tr>
<td>([1.0E-03, 1.0E-02])</td>
<td>CPU time (in hours) 0.1469</td>
<td>Number of the time-steps 21504</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU time (in hours) 0.0897</td>
</tr>
<tr>
<td>([1.0E-04, 1.0E-03])</td>
<td>CPU time (in hours) 1.1242</td>
<td>Number of the time-steps 344032</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU time (in hours) 0.1192</td>
</tr>
<tr>
<td>([1.0E-05, 1.0E-04])</td>
<td>CPU time (in hours) 6.6747</td>
<td>Number of the time-steps 275212</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU time (in hours) 0.2458</td>
</tr>
<tr>
<td>([1.0E-06, 1.0E-05])</td>
<td>CPU time (in hours) 43.0650</td>
<td>Number of the time-steps 2202096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU time (in hours) 0.6058</td>
</tr>
<tr>
<td>([1.0E-07, 1.0E-06])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 1.0197</td>
<td>Number of the time-steps 86016</td>
</tr>
<tr>
<td>([1.0E-08, 1.0E-07])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 3.1219</td>
<td>Number of the time-steps 344064</td>
</tr>
<tr>
<td>([1.0E-09, 1.0E-08])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 10.3705</td>
<td>Number of the time-steps 1376256</td>
</tr>
<tr>
<td>([1.0E-10, 1.0E-09])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 35.3331</td>
<td>Number of the time-steps 5505024</td>
</tr>
<tr>
<td>([1.0E-11, 1.0E-10])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 66.1322</td>
<td>Number of the time-steps 11010048</td>
</tr>
<tr>
<td>([1.0E-12, 1.0E-11])</td>
<td>Required accuracy was not achieved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU time (in hours) 230.2309</td>
<td>Number of the time-steps 44040192</td>
</tr>
</tbody>
</table>

Table 3.4
Comparison of the computational costs (measured by the CPU hours) needed to achieve prescribed accuracy in the cases where (a) the \(\theta\)-method with \(\theta = 0.75\) is implemented directly and (b) the Richardson Extrapolation is used in combination with the same underlying numerical scheme.

- The implementation of the Richardson Extrapolation becomes much more efficient than the direct \(\theta\)-method with \(\theta = 0.75\) when the accuracy requirement is increased (see the second, the third, the fourth and the fifth lines of Table 3.4). If it desirable to achieve accuracy, which is better than \(10^{-5}\), and more precisely if it is required to have \(\text{ERROR}\) from (3.61) should be in the interval \([10^{-6}, 10^{-5}]\), then the computing time spent with the Richardson Extrapolation is more than 70
times smaller than the corresponding computing time for the $\theta$-method with $\theta = 0.75$ when it used directly (compare the CPU times on the fifth line of Table 3.4).

- Accuracy better than $10^{-5}$ has not been achieved in the 19 runs with the $\theta$-method with $\theta = 0.75$ when it is used directly (see Table 3.4), while even accuracy better than $10^{-11}$ is achievable when the Richardson extrapolation is used (see the last line of Table 3.4 and Table 3.2).

- 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).

3.6.6. Using the Trapezoidal Rule in the computations

Consider the Trapezoidal Rule (which is a special numerical scheme belonging to the class of the $\theta$-methods and found from this class by setting $\theta = 0.5$). It has been shown (see Theorem 3.2) that, while the Trapezoidal Rule itself is a second-order $A$-stable numerical method, its combination with the active implementation of the Richardson 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 the 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 3.5. 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):

(a) The order of the Trapezoidal Rule is two. Therefore, it should be expected that doubling the number $N$ of time-steps, which leads to a decrease of the time-step size $h = (129600 - 43200)/N = 86400/N$ by a factor of two, will in general lead to an improvement of the accuracy by a factor of 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 3.2. 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 \(10^{10}\) 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 \(n\), then the stepsize \(h\) is halved. This can happen several times at the time-step \(n\). If the reduced time-stepsizes becomes less than \(10^{-5}h\), then the computational process is stopped and declared to be unstable. If the time-stepsizes is reduced at time-step \(n\), then the remaining calculations from \(t_{n-1}\) to \(t_n\) are performed with the reduced time-stepsizes (with the reduced time-stepsizes, if the time-stepsizes has been reduced several times), however an attempt is carried out to perform the next time-step \(n+1\) (i.e. to proceed from \(t_{n-1}\) to \(t_n\)) with the time-stepsizes \(h = (129600 - 43200)/N = 86400/N\) that is used in the current run \(j\) where \(j = 1, 2, \ldots, 19\).

(c) The order of the Passive Richardson Extrapolation with the Trapezoidal Rule should be three. Therefore, it should be expected that doubling the number \(N\) of time-steps, which leads to a decrease of the time-stepsizes \(h = (129600 - 43200)/N = 86400/N\) 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 3.2 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).

### 3.7. Some concluding remarks

The implementation of the of the Richardson Extrapolation in connections of numerical schemes from the class of the \(\theta\)-methods was studied in detail in this chapter. It was shown that for some values of the parameter \(\theta\) the application of the Richardson Extrapolation together with the corresponding method will lead to unstable results. On the other hand it was proved that for many other values of \(\theta\)
the stability properties of the underlying methods are preserved when these are combined with the Richardson Extrapolation.

<table>
<thead>
<tr>
<th>Job Number</th>
<th>Number of steps</th>
<th>Direct Implementation</th>
<th>Richardson Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accuracy Rate</td>
<td>Active Accuracy Rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Passive Accuracy Rate</td>
</tr>
<tr>
<td>1</td>
<td>168</td>
<td>3.605E-01 -</td>
<td>Unstable n.a. 4.028E-02</td>
</tr>
<tr>
<td>2</td>
<td>336</td>
<td>7.785E-02 4.631</td>
<td>Unstable n.a. 3.246E-03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.407</td>
</tr>
<tr>
<td>3</td>
<td>672</td>
<td>1.963E-02 3.961</td>
<td>Unstable n.a. 1.329E-03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.443</td>
</tr>
<tr>
<td>4</td>
<td>1344</td>
<td>4.915E-03 3.998</td>
<td>Unstable n.a. 1.462E-04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9.091</td>
</tr>
<tr>
<td>5</td>
<td>2688</td>
<td>1.228E-03 4.001</td>
<td>Unstable n.a. 5.823E-05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.510</td>
</tr>
<tr>
<td>6</td>
<td>5376</td>
<td>3.071E-04 4.000</td>
<td>Unstable n.a. 3.765E-05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.547</td>
</tr>
<tr>
<td>7</td>
<td>10752</td>
<td>7.677E-05 4.000</td>
<td>Unstable n.a. 2.229E-05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.689</td>
</tr>
<tr>
<td>8</td>
<td>21504</td>
<td>2.811E-05 2.731</td>
<td>Unstable n.a. 1.216E-05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.833</td>
</tr>
<tr>
<td>9</td>
<td>43008</td>
<td>1.615E-05 1.741</td>
<td>Unstable n.a. 6.300E-06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.930</td>
</tr>
<tr>
<td>10</td>
<td>86016</td>
<td>8.761E-06 1.843</td>
<td>Unstable n.a. 3.188E-06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.976</td>
</tr>
<tr>
<td>11</td>
<td>172032</td>
<td>4.581E-06 1.912</td>
<td>Unstable n.a. 1.600E-06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.993</td>
</tr>
<tr>
<td>12</td>
<td>344064</td>
<td>2.345E-06 1.954</td>
<td>Unstable n.a. 8.007E-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.998</td>
</tr>
<tr>
<td>13</td>
<td>688128</td>
<td>1.187E-06 1.976</td>
<td>Unstable n.a. 4.005E-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.999</td>
</tr>
<tr>
<td>14</td>
<td>1376256</td>
<td>5.970E-07 1.988</td>
<td>Unstable n.a. 2.002E-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>15</td>
<td>2752512</td>
<td>2.994E-07 1.994</td>
<td>Unstable n.a. 1.001E-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>16</td>
<td>5505024</td>
<td>1.499E-07 1.997</td>
<td>Unstable n.a. 5.005E-08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>17</td>
<td>11010048</td>
<td>7.503E-08 1.998</td>
<td>Unstable n.a. 2.503E-08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>18</td>
<td>22020092</td>
<td>3.753E-08 1.999</td>
<td>Unstable n.a. 1.252E-08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>19</td>
<td>44040192</td>
<td>1.877E-08 2.000</td>
<td>Unstable n.a. 6.257E-09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.000</td>
</tr>
</tbody>
</table>

**Table 3.5**

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 (3.61) 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 very difficult example representing an atmospheric chemistry scheme with 56 important chemical species was used in the experiments. This scheme is badly scaled, very stiff and some of its components have very steep gradients. Therefore many numerical methods fail in the computer treatment of this problem. The tests performed by us with some of the selected methods (which are representatives of the class of the θ-methods) gave in general quite good results (the Trapezoidal Rule being an exception).

The behaviour of the Richardson Extrapolation was studied in detail when this device was applied to three well-known representatives of the class of the θ-methods (the Backward Differentiation Formula, the Trapezoidal Rule and the θ-method obtained by using $θ = 0.75$).
For the Backward Differentiation Formula and for the $\theta$-method obtained by using $\theta = 0.75$, which are L-stable and strongly A-stable respectively, the numerical results confirm the proved theoretical results.

For the Trapezoidal Rule, which is only A-stable, some problems with the accuracy of the results were detected. This numerical method failed completely when it is actively combined with the Richardson Extrapolation. It is not a surprise, because it was proved that the new numerical method obtained by this combination is unstable. More surprising is the fact that the underlying method, which is A-stable, has some difficulties to obtain always the expected accuracy during the computations. Also the passive implementation of the Richardson Extrapolation (which has the same stability properties, A-stable, as the underlying method) is not giving the expected second order of accuracy. This fact indicates that strongly A-stable and L-stable numerical schemes are indeed performing better when the solved problems are very difficult.

The detailed description of the implementation of the Richardson Extrapolation in connection with the class of the $\theta$-methods could be used to achieve similar results also for other numerical method (as, for example, for some fully implicit Runge-Kutta methods and for some of the somewhat simpler diagonally implicit Runge-Kutta methods). It will be interesting to check the performance of the Richardson Extrapolation when some high-order numerical methods of Runge-Kutta type are used in the computations,
References


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.


