Control of Error and Convergence in ODE Solvers

Gustafsson, Kjell

1992

Document Version:
Publisher's PDF, also known as Version of record

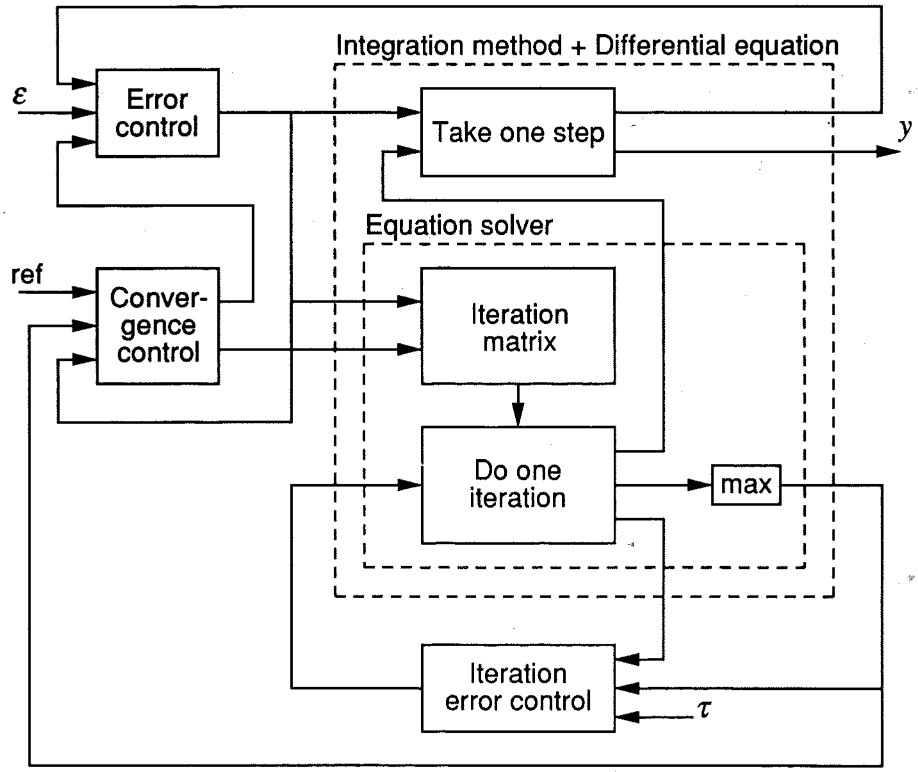
Link to publication

Citation for published version (APA):
Gustafsson, K. (1992). *Control of Error and Convergence in ODE Solvers*. [Doctoral Thesis (monograph), Department of Automatic Control]. Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

Control of Error and Convergence in ODE Solvers

Kjell Gustafsson



Lund 1992

Cover figure: A block diagram demonstrating the feedback control viewpoint of an implicit integration method, cf. Chapter 5.

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

Printed in Sweden
by Studentlitteratur's printing-office
Lund 1992

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
DOCTORAL DISSERTATION

Date of issue
March 1992

Document Number
ISRN LUTFD2/TFRT--1036--SE

Author(s)
Kjell Gustafsson

Supervisor
Karl Johan Åström, Gustaf Söderlind, Per Hagander

Sponsoring organisation
Swedish Research Council for Engineering Sciences (TFR), contract 222/91-405. Nordiska Forskarutbildningsakademien, contract 31068.34.024/91).

Title and subtitle
Control of Error and Convergence in ODE Solvers

Abstract

Feedback is a general principle that can be used in many different contexts. In this thesis it is applied to numerical integration of ordinary differential equations. An advanced integration method includes parameters and variables that should be adjusted during the execution. In addition, the integration method should be able to automatically handle situations such as: initialization, restart after failures, etc. In this thesis we regard the algorithms for parameter adjustment and supervision as a controller. The controller measures different variables that tell the current status of the integration, and based on this information it decides how to continue. The design of the controller is vital in order to accurately and efficiently solve a large class of ordinary differential equations.

The application of feedback control may appear farfetched, but numerical integration methods are in fact dynamical systems. This is often overlooked in traditional numerical analysis. We derive dynamic models that describe the behavior of the integration method as well as the standard control algorithms in use today. Using these models it is possible to analyze properties of current algorithms, and also explain some generally observed misbehaviors. Further, we use the acquired insight to derive new and improved control algorithms, both for explicit and implicit Runge-Kutta methods.

In the explicit case, the new controller gives good overall performance. In particular it overcomes the problem with an oscillating stepsize sequence that is often experienced when the stepsize is restricted by numerical stability. The controller for implicit methods is designed so that it tracks changes in the differential equation better than current algorithms. In addition, it includes a new strategy for the equation solver, which allows the stepsize to vary more freely. This leads to smoother error control without excessive operations on the iteration matrix.

Key words
numerical integration, ordinary differential equations, initial value problems, simulation, Runge-Kutta methods, stepsize selection, error control, convergence control, numerical analysis

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
184

Recipient's notes

Security classification

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

Control of Error and Convergence in ODE Solvers

Kjell Gustafsson

Tekn. Lic., Civ. Ing.

Kristianstads Nation

Akademisk avhandling som för avläggande av teknisk doktorsexamen vid Tekniska fakulteten vid Universitetet i Lund kommer att offentligen försvaras i sal M:A, Maskinhuset, Lunds Tekniska Högskola, fredagen den 22 maj 1992, kl 10.15.

A PhD thesis to be defended in public in room M:A, the M-building, Lund Institute of Technology, on Friday May 22 1992, at 10.15.

Contents

Preface

"It's one small step for man, one giant leap for mankind."
Neil Armstrong

The size of a step should be related to its environment. This is particularly true in numerical integration of ordinary differential equations, where the stepsize is used to control the quality of the produced solution. A large stepsize leads to a large error, while a too small stepsize is inefficient due to the many steps needed to carry out the integration. What is regarded as large or small, depends on the differential equation and the required accuracy.

"The choice of stepsize in an integration method is a problem you control guys should have a look at," said Professor Gustaf Söderlind, when handing out project suggestions at the end of a course on numerical integration methods in 1986/87. Michael Lundh and I got interested in the problem and started working. We soon realized that the standard rule for selecting stepsize could be interpreted as a pure integrating controller. A generalization to PI or PID control is then rather reasonable. We experimented with PI control and achieved good results [Gustafsson *et al.*, 1988].

After having finished the project both Michael and I continued working on other research topics. I was, however, rather annoyed that we did not have a good explanation of why a PI controller made such a difference in performance in the case where numerical stability restricts the stepsize. What was it in the integration method that changed so dramatically? After a couple of months I returned to the problem, strongly encouraged by Professors Karl Johan Åström, Gustaf Söderlind, and Docent Per Hagander. Soon I was hooked. Pursuing the feedback control viewpoint turned out to be very rewarding. Not only was it possible to explain the superiority of the PI controller, but the approach also provided means to ..., well, continue on reading, and you will learn all about it.

Acknowledgements

This will probably be the most read section of this thesis. I have looked forward to writing it, since it gives me an opportunity to express my gratitude to all dear friends. They have been of invaluable assistance during the years I have been working with this material. I have benefited immensely from their feedback on my ideas, from their help to get the

right perspectives, and from their encouragement and support in times of intellectual standstill. Thank you!

There are a few friends I would like to mention in particular:

Working in an interdisciplinary area I have enjoyed the pleasure of several supervisors. I want to express my gratitude to Karl Johan Åström for teaching me about control, and enthusiastically backing me up when I have tried to apply feedback ideas in a somewhat different area. He has taught me to pose the right questions, and I have benefited immensely from his vast research experience. With admirable teaching skill Gustaf Söderlind introduced me to the subject of numerical integration. He has been a tremendous source of information and has teasingly presented one challenging research problem after another. I am also indebted to Per Hagander. He has provided many constructive pieces of advice. I appreciate him for taking the time to penetrate even minute details, and for allowing ample time for all our discussions.

Among my fellow graduate students I would like to mention Michael Lundh, Bo Bernhardsson, and Ola Dahl. The early part of the work was done together with Michael and I am grateful for his contributions. Bo has been a dear and close friend for more than a decade. We have a common fascination for life in general and engineering in particular. Thanks for sharing your insight and ideas with me. Thanks Ola for surviving in the same office with me, patiently overlooking me blocking the computer and spreading my things everywhere.

My colleagues at the Department of Automatic Control in Lund form a friendly and inspiring community. The creative and open-minded atmosphere has influenced me a lot. In addition, the extra-ordinary access to computing power and high quality software, excellently maintained by Leif Andersson and Anders Blomdell, has made work very stimulating.

I have plagued Gustaf Söderlind, Karl Johan Åström, Per Hagander, and Björn Wittenmark with several versions of the manuscript. Their constructive criticism have greatly improved the quality of the final product.

The work has been partly supported by the Swedish Research Council for Engineering Sciences (TFR, under contract 222/91-405). Nordiska Forskarutbildningsakademien has generously funded cooperation with colleagues at the Technical Universities of Norway and Denmark.

Finally, Eva, my beloved wife and best friend, I thank you for your love and support.

K.G.

1

Introduction

Simulation is a powerful alternative to practical experiments. The properties of many physical systems and/or phenomena can be assessed by combining mathematical models with a numerical simulation program. Ordinary differential equations (ODE) are basic building blocks in such models. It is usually impossible to obtain closed form solutions of differential equations. The simulation program therefore has to implement some numerical integration method to obtain the behavior of the system. Such simulation programs are used routinely by e. g. control engineers.

Most simulation programs are constructed so that the user only supplies the model and prescribes a certain tolerance. The program is then expected to produce a solution with the specified accuracy. This is a very difficult task. Different models have quite different properties, and robust implementations of several types of integration methods are required to simulate them accurately and efficiently.

This thesis deals with some of the problems encountered on the road from a discretization formula defined by the integration method to a working simulation. Implementing a numerical method is not only a question of turning a formula into code. Most methods include parameters and variables that need to be adjusted during the execution. An important step towards a successful implementation is the design of algorithms and strategies taking care of these adjustments. The implementation should also include algorithms that check the solution accuracy and administrate special situations, e.g. the initialization, restart after possible failures, etc.

The viewpoint taken in the thesis is to regard the algorithms for parameter adjustment and supervision as a controller. The objective of this controller is to run the numerical integration method so that it solves a large class of differential equations accurately and efficiently.

1.1 Is This Thesis Really Needed?

There are many high quality implementations of different integration methods available today, e.g. DASSL [Brenan *et al.*, 1989], RADAU5 [Hairer and Wanner, 1991], LSODE [Hindmarsh, 1983], STRIDE [Burrage *et al.*, 1980]. This indicates that some people, at least, know how to turn a discretization formula into a successful implementation. Why then devote a thesis to the problem? To illustrate that this thesis is indeed needed we will show two examples:

EXAMPLE 1.1—Control system example
The stepsize is an important variable that needs to be adjusted during the integration. Figure 1.1 shows a signal from a simulation of a small linear control system, cf. Example 4.2. If an explicit integration method is used to solve the problem, stability restricts the size of the integration steps. The standard method of selecting stepsize fails to handle this situation properly, and the result is an oscillatory stepsize sequence. The oscillations introduce an error in the simulation result as is shown in the left plot of Figure 1.1. There are no oscillatory modes in the control system and the solution produced is therefore qualitatively wrong. The artifact can be removed by improving the stepsize selection algorithm using the methods discussed in this thesis. The result is shown in the right plot of Figure 1.1. □

EXAMPLE 1.2—Brusselator
The Brusselator is a test problem often used in numerical integration, cf. Examples 4.1 and 5.4. The result of simulating this problem is shown in Figure 1.2. Just before and during the large state transition at $t \approx 4.8$ many integration steps have to be rejected due to a large error. It is reasonable to expect the error controller to negotiate the change in the differential equation without this amount of wasted computation. In this thesis we will demonstrate how that can be achieved. □

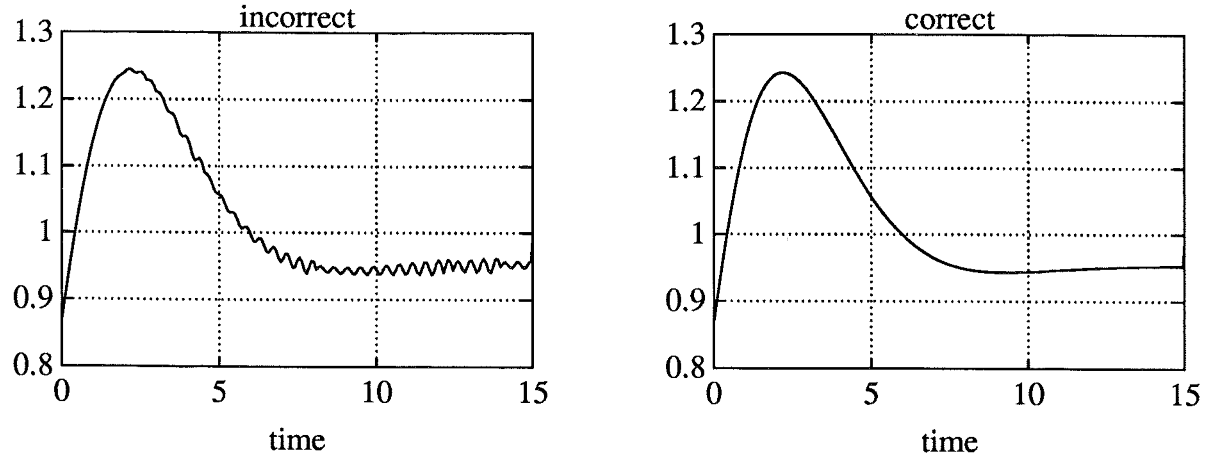


Figure 1.1 A signal drawn from a control system simulation. The oscillatory component in the signal to the left is caused by an irregular stepsize sequence. The correct signal, to the right, is obtained by improving the stepsize selection algorithm.

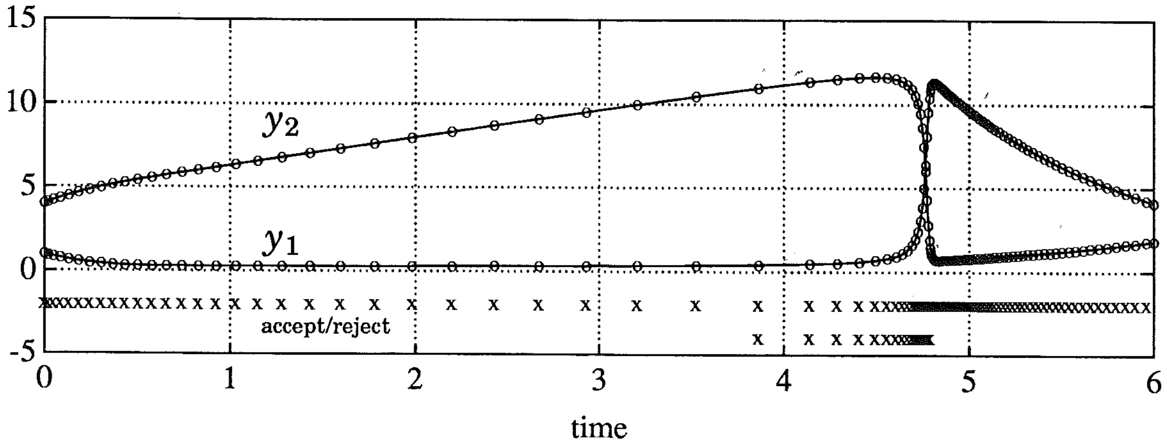


Figure 1.2 The solution to the Brusselator in Example 1.2. The 'o' marks the numerical solution points, and the corresponding 'x' indicates if a specific step was successful (high value) or rejected (low value). During the time interval $t \in [3.0,\ 4.8]$ there are 25 accepted and 20 rejected steps.

The above examples demonstrate two situations where even today's best algorithms fail. Although they normally perform better, the situations shown in the examples are not unique. To improve the robustness of current simulation programs we need to understand the reasons for the misbehavior, and then design remedies.

Classical numerical analysis theory gives, unfortunately, few tools to approach analysis and design of algorithms for supervision and parameter adjustment. Instead it is advantageous to exploit some analogies from feedback control theory. The supervisory algorithms measure different variables (e. g. error estimate, rate of convergence in the equation

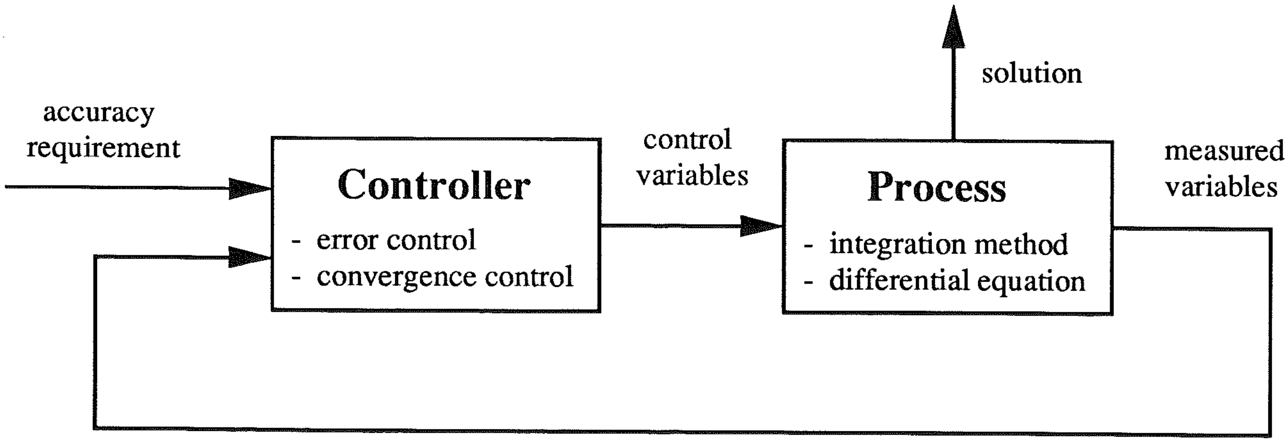


Figure 1.3 Error and convergence control in an integration method. The convergence control is not needed in an explicit method.

solver, stiffness estimate) that tell the current status of the integration. Based on this information they decide how to continue the integration (e. g. accept/reject current step, what stepsize to use, choice of equation solver, change of iteration matrix). This is a feedback control loop, as shown in Figure 1.3. The supervision is the controller and the integration method is the controlled process. Feedback control theory supplies proven methods to analyze the properties of the feedback loop as well as methods to design a good controller.

Separating the controller from the "numerical" part of the integration method is an important idea *per se*. Integration codes are complicated, and the separation makes it easier to obtain a correct implementation. It also makes it easier to introduce global considerations into the controller design.

1.2 Scope of the Thesis

Since the control of numerical integration is of paramount importance for the reliability, efficiency, and accuracy of a simulation, the problem has repeatedly been given considerable attention in numerical analysis. In particular, error control by means of adjusting the stepsize is central to every high quality solver for dynamical systems. The error estimator, to be used for this purpose, is mathematically both intricate and highly sophisticated. In contrast, the stepsize selection strategies implemented are typically based on heuristic principles and asymptotic arguments. They are simplistic from the point of view of control theory, although they perform remarkably well in most cases.

The present thesis, interdisciplinary in character, aims at developing a better process understanding and better use of the error estimate by means of control theory. A few well chosen test problems, that are rich in features, demonstrate shortcomings of the classical approach. Some of these can be remedied by using proven techniques from control theory. Others, however, cannot be treated at all, not because of shortcomings in control, but as a matter of principle. In the latter case our analysis returns the problem to numerical analysis with the conclusion that the method itself and/or its error estimator may be inappropriate for certain classes of problems.

The thesis deals almost solely with Runge-Kutta methods. This is not to say that other types of integration methods are uninteresting. The class of Runge-Kutta methods is, however, sufficiently rich to demonstrate the problems we want to address. The methodology we use, and also many of the results, are applicable to other types of integration methods. Dealing with one class of methods makes comparative studies easier. In addition, care has been taken to provide exactly the same environment in the implementation of the different methods we use.

Controlling an integration method is both a question of accuracy and efficiency. The problem of producing a solution within a specified accuracy will be referred to as *error control*. In an integration method the size of the integration steps relates directly to the error in the solution, and the stepsize is one of the variables the error controller needs to adjust. The stepsize also affects the efficiency, since the smaller the stepsize the more integration steps will be needed to simulate the differential equation. In an implicit integration method a large part of the computation is spent solving a set of nonlinear equations using an iterative equation solver. The convergence rate of this iteration is closely related to the efficiency of the integration method. We will, in loose terms, refer to the supervision and control of the equation solver as *convergence control*.

One way to improve the efficiency of an integration method is to exploit special types of computer hardware, e. g. serial/parallel architectures, or special properties of the differential equation, e. g. sparse/nonsparse or banded Jacobian. These types of considerations will not be addressed in the thesis.

Before designing a controller it is important to consider the quality of the measured variables that are available. The purpose of the error control is to make the accuracy of the solution comply with the user's requirement. This can only be achieved if the measurements available to the controller truly reflect the real error. If the relation between the measured variable and the global objective is weak, then the result will be poor no matter how the controller is constructed. Our attitude is to make as good control as possible with available information. The problem of constructing methods and relevant error estimators properly, is the responsibility of the numerical analyst.

In the design of the controller we will note that seemingly similar processes differ significantly in how difficult they are to control. Again we will try to do the best with what is available, but new integration methods should be designed also from the control point of view.

1.3 Thesis Outline

This thesis describes a typical application of feedback control. The process is a little different from the standard mechanical, electrical, or chemical examples. The methodology is nevertheless identical to what is developed in standard textbooks on feedback control theory [Franklin *et al.*, 1986; Åström and Wittenmark, 1990; Franklin *et al.*, 1990].

A first step towards successful control is to acquire process knowledge. What is the purpose of the process? What is its normal mode of operation? What are the input signals, and how do they affect the behavior of the process? What are the output signals, and do they truly reflect the internal state of the process? What is the controller expected to achieve? Chapter 2 addresses some of these questions, by presenting an overview of numerical integration in general and Runge-Kutta methods in particular. Not all of the material is essential for the controller design, but most of it is needed to get the overall picture.

Many readers may want to skim Chapter 2. For the numerical analyst most of the material is probably well known, although the notations sometimes differ from what is normally used. The control engineer may find the chapter long and involved. A detailed understanding of the whole material is not needed to appreciate the ideas of later chapters. It would be possible to continue and only consult Chapter 2 as a reference.

Once the properties of the process are understood we proceed to derive models for its behavior. These models will be used in the design of the controller, and they should describe dominating features of the relation between the input and output signals as simply as possible. Important considerations are: is the input-output relation static or dynamic, linear or nonlinear? Does the behavior differ between different operating points?

An integration method has several inputs and outputs. The most important relation is the one between stepsize and error. Chapter 3 is devoted to this. It turns out that different models are needed at different operating points, and, more importantly, for a detailed process description the static models normally used are not sufficient. We derive simple dynamic models that describe the different aspects of the stepsize-error relation well. The models are verified using numerical tests and system identification techniques.

Having obtained process models we continue with the controller design. Chapter 4 is devoted to explicit Runge-Kutta methods. The properties of the algorithms in use today are analyzed and the reasons for some of their shortcomings are explained. Much insight is gained from the emphasis on dynamics provided by the feedback control analysis. A new improved controller is designed and evaluated on different test problems. It results in good overall performance and avoids several of the misbehaviors of current controllers.

Chapter 5 treats the design of a controller for implicit Runge-Kutta methods. An implicit integration method includes an equation solver, and part of the chapter is spent on modeling its input-output behavior. This model is combined with the models of the stepsize-error relation and a new improved controller is derived. The controller achieves better error control and provides a new strategy concerning the administration of the equation solver.

Finally, we conclude in Chapter 6 with a general discussion of the methodology used and the results obtained in the thesis.

2

Runge-Kutta Methods

A robust implementation of an integration method involves the design of strategies and algorithms to supervise and run the integration in an accurate and efficient way. We approach the problem by regarding these algorithms as a controller connected to the integration method. The design of such a controller requires a thorough understanding of the properties of the integration method.

In this chapter we review some basic properties of integration methods in general, and Runge-Kutta methods in particular. The aim is to introduce notations and present the background needed for the control analysis and control design. For more complete presentations we refer to standard textbooks such as [Gear, 1971; Butcher, 1987; Hairer *et al.*, 1987; Hairer and Wanner, 1991].

2.1 Integration Methods in General

An integration method computes the solution to the initial value problem

$$\dot{y} = f(t, y), \qquad y(t_0) = y_0 \in \mathbb{R}^N, \quad t \in [t_0, t_{\text{end}}], \tag{2.1}$$

where f is sufficiently differentiable, by approximating it with the difference equation

$$y_{n+1} = \bar{y}_n + h_n \bar{f}_n, \qquad n = 0, 1, 2, \ldots \tag{2.2}$$

In this equation $\bar{y}_n$ is formed as a combination of past solution points and $\bar{f}_n$ is formed as a combination of function values $f(t, y)$ evaluated at the solution points or in their neighborhood. Using (2.2) we compute $y_0, y_1, y_2, \dots$ as approximations to $y(t_0), y(t_1), y(t_2), \dots$. The stepsize h_n between consecutive solution points is defined as $t_{n+1} = t_n + h_n$.

The discretization defining $\bar{y}_n$ and $\bar{f}_n$ can be constructed in many different ways. The simplest choice is the explicit (forward) Euler method.

EXAMPLE 2.1—Explicit Euler method
The explicit Euler method discretizes the equation (2.1) as

$$y_{n+1} = y_n + h_n f(t_n, y_n)$$

i. e. $\bar{y}_n$ is chosen as the previous numerical solution point y_n and $\bar{f}_n$ is the function value $f(t, y)$ at this point. □

Any integration method aims at producing a numerical solution close to the true solution. Whether this is achieved or not, depends on how the discretization (2.2) is constructed, as well as on the properties of the underlying problem (2.1). Although different methods may choose $\bar{y}_n$ and $\bar{f}_n$ very differently, certain basic properties are shared by all (useful) integration methods.

Consistency, Stability, and Convergence

In a suitable setting, the differential equation (2.1) can be regarded as an operator equation $\Phi = 0$, where Φ maps a normed function space U into another normed function space V. Necessary initial values are assumed to be incorporated in this formulation. An element $y \in U$ is a function $y : t \mapsto y(t)$, $t \in \mathbb{R}^+$, $y(t) \in \mathbb{R}^N$. It is a solution to (2.1) if $\Phi(y) = 0$.

An integration method defines a discrete approximation Φ_h to Φ, which rewrites (2.2) as $\Phi_h(y^h) = 0$. The discrete operator Φ_h maps the normed sequence space U_h into another normed sequence space V_h. The sequence $y^h \in U_h$, is a function $y^h : n \mapsto y_n$, $n \in \mathbb{N}$, $y_n \in \mathbb{R}^N$, and it solves (2.2) if $\Phi_h(y^h) = 0^h$.

The discrete solution y^h is an approximation of the exact solution y on the grid t^h, $t^h : n \mapsto t_n$, $n \in \mathbb{N}$, $t_n \in \mathbb{R}$. To be able to compare y^h with y we need a restriction operator that relates U (V) to U_h (V_h). Introduce $\Gamma : \Gamma t = t^h$ and let Γy denote the restriction of $y(t)$ to the grid points t^h. The relation between the differential equation (2.1) and the

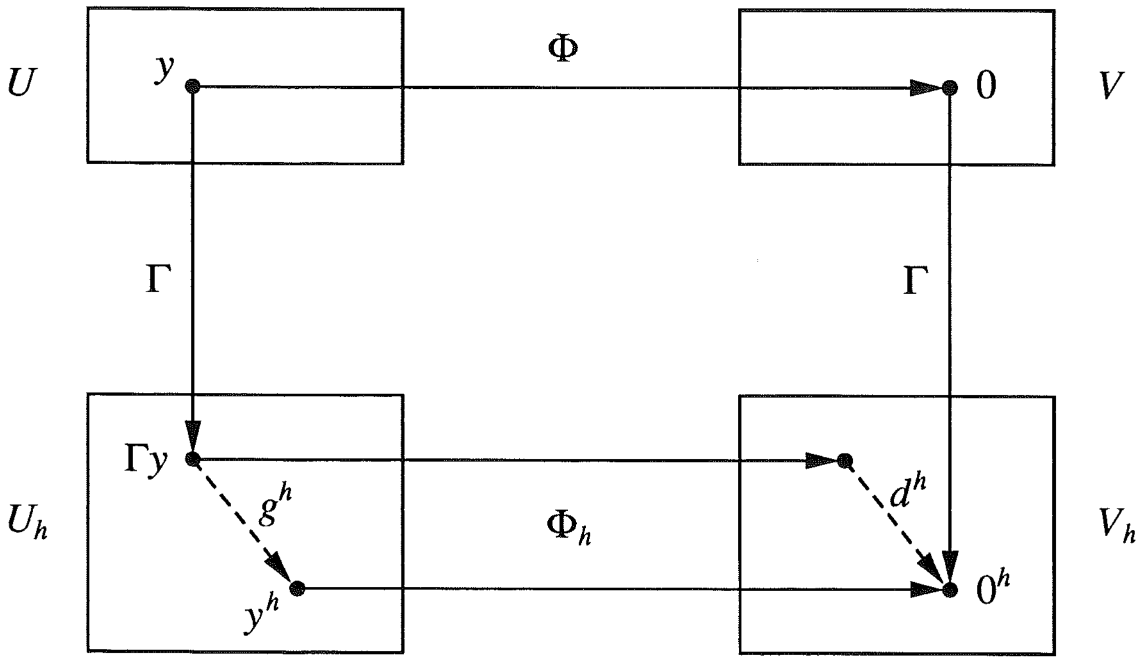


Figure 2.1 The solution y^h generated by the integration method differs from the exact solution Γy, since the discretization Φ_h only solves the underlying differential equation Φ approximately. The discretization error can be studied either through the global error g^h or the defect d^h.

difference equation (2.2) can now be described as in Figure 2.1 [Stetter, 1973, pp. 1–11], [Söderlind, 1987].

The difference between the numerical solution and the exact solution is referred to as the *global error*, i.e. $g^h = y^h - \Gamma y$ or, pointwise, $g_n = y_n - y(t_n)$. The global error is defined in the space U_h. The corresponding error in V_h is called the *defect* and is defined as $d^h = -\Phi_h(\Gamma y)$.

For a (useful) integration method we expect the numerical solution y^h to converge to the exact solution y when $h \to 0$. This property is referred to as *convergence*.

Definition 2.1—Convergence
An integration method is convergent of order p if $||g^h|| = O(h^p)$, or equivalently, if the global error satisfies

$$||(\Phi_h^{-1}\Gamma - \Gamma\Phi^{-1})(0)|| = O(h^p)$$

on V, for all sufficiently smooth problems $\Phi(y) = 0$. □

Convergence is related to two other properties: *consistency* and *stability*. Consistency implies that the difference equation (2.2) converges to the differential equation (2.1) when $h \to 0$. Stability, on the other hand, concerns the operator Φ_h^{-1} and assures that it is well behaved in the limit as $h \to 0$.

DEFINITION 2.2—Consistency
An integration method is consistent of order p if $||d^h|| = O(h^p)$, or equivalently, if the defect satisfies

$$||(\Phi_h\Gamma - \Gamma\Phi)(y)|| = O(h^p)$$

on U, for all sufficiently smooth problems $\Phi(y) = 0$. □

DEFINITION 2.3—Stability
An integration method is stable if Φ_h^{-1} is uniformly Lipschitz on V_h as $h \to 0$. □

This definition is a little more strict than necessary, but it assures that the orders of consistency and convergence agree. The term "order" therefore usually refers to the order of convergence.

The following theorem [Gear, 1971, p. 170], [Dahlquist *et al.*, 1974, p. 377], [Hairer *et al.*, 1987, p. 341], [Butcher, 1987, pp. 352–367], is classical in numerical analysis:

THEOREM 2.1
Convergence, consistency, and stability are related as

$$\text{convergence} \quad \Longleftrightarrow \quad \text{consistency} \quad + \quad \text{stability}$$

□

The definitions above seem to indicate that an integration method should have as high convergence order as possible. The higher the order, the larger we could choose the stepsize and still obtain a numerical solution with sufficient accuracy. Each individual integration step is, however, computationally more expensive in a high order method, and therefore high order does not necessarily lead to less computation for a given accuracy. Which order that is the most efficient depends on the differential equation, on the type of integration method, as well as on the required accuracy. In practice it is advantageous to have methods of different orders available.

Two methods with the same order may behave very differently on the same problem. The reason is that specific discretizations perform better on some classes of problems than other. No discretization is able to handle all types of problems equally well. Consequently, many different integration methods have been suggested over the years, each one with its own advantages and disadvantages. Depending on the type of discretization they use, one can normally separate them into a few different classes, e. g. explicit – implicit, onestep – multistep.

Explicit – Implicit

An integration method is said to be *explicit* if $\bar{f}_n$ in (2.2) is not a function of y_{n+1}. An obvious example is the explicit Euler method described in Example 2.1. Such methods give an explicit formula for the calculation of the next solution point, which makes them straightforward to implement.

The situation is more complicated for *implicit* methods. Then $\bar{f}_n$ depends on y_{n+1}, and a nonlinear equation has to be solved in order to get the next solution point. An example is the implicit (backward) Euler method.

EXAMPLE 2.2—Implicit Euler method
The implicit Euler method discretizes the equation (2.1) as

$$y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1})$$

i. e. $\bar{y}_n$ is chosen as the previous numerical solution point y_n and $\bar{f}_n$ is the function value $f(t, y)$ at the new solution point. To obtain y_{n+1} a nonlinear equation has to be solved. □

The equation that defines y_{n+1} in an implicit method requires the use of a numerical equation solver, leading to increased implementation complexity. This complexity is, however, often well motivated, since for certain problems (e. g. mildly stiff or stiff problems) an implicit method is much more efficient than comparable explicit ones.

Onestep – Multistep

An integration method is a *onestep* method if $\bar{y}_n$ and $\bar{f}_n$ in (2.2) do not depend on solution points prior to y_n. A consistent onestep method is always convergent for ordinary differential equations. The previous two examples, i. e. explicit Euler and implicit Euler, are both onestep methods. More complicated onestep methods normally involve iterated evaluations of f when forming $\bar{f}_n$. Such methods are known as Runge-Kutta methods, and an example is:

EXAMPLE 2.3—Midpoint method
The midpoint method [Hairer *et al.*, 1987, p. 130] makes use of two function evaluations when forming $\bar{f}_n$. The resulting difference equation reads

$$y_{n+1} = y_n + h_n f\left(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} f(t_n, y_n)\right).$$

The method is explicit since y_{n+1} does not enter the right hand side of the formula. □

When computing $\bar{f}_n$ a Runge-Kutta method makes use of intermediate points, called *stages*, in the neighborhood of a local solution of the differential equation. In the previous example $(t_n+h_n/2, y_n+h_n f(t_n, y_n)/2)$ is such a point. Another approach, leading to *multistep* methods, is to construct $\bar{y}$ and/or $\bar{f}$ based on several previous values of the numerical solution instead of using intermediate points.

EXAMPLE 2.4—Second order Adams-Bashforth method, AB2
The Adams-Bashforth methods [Hairer *et al.*, 1987, pp. 304–308] are a family of explicit multistep methods. The second order formula in this family reads

$$y_{n+1} = y_n + h\left(\frac{3}{2} f(t_n, y_n) - \frac{1}{2} f(t_{n-1}, y_{n-1})\right)$$

assuming a constant stepsize h. The method uses the function value at two previous solution points, y_n and y_{n-1} when forming $\bar{f}_n$. □

A multistep method is normally more complicated to implement than a onestep method. Several previous solution points are used when forming $\bar{y}_n$ and/or $\bar{f}_n$, and if the stepsize varies these points come from a nonequidistant grid. The coefficients of the method should be recomputed to compensate for this, making the discretization formula far more complicated than what is indicated in Example 2.4. In addition, due to the dependence on several previous solution points a multistep method needs more initial values than onestep methods. All in all, this adds complexity and makes the implementation quite technical. It may, however, for other reasons still be motivated, e.g. less amount of computation for certain classes of differential equations.

The multistep method in Example 2.4 uses several previous function values when forming $\bar{f}_n$. A multistep method can, naturally, also be based on using several points when computing $\bar{y}_n$, cf. (2.2).

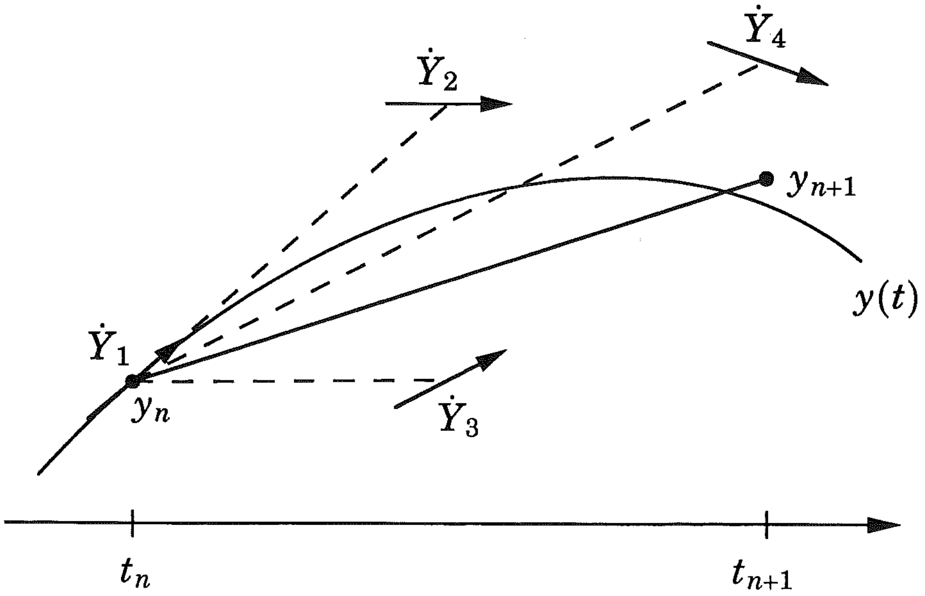


Figure 2.2 The idea behind a Runge-Kutta method. The derivative is sampled at a few points in the neighborhood of the local solution. A step is then taken in a direction based on these derivative values ($\dot{Y}_i$). The figure illustrates the classical method RK4 [Hairer and Wanner, 1991, p. 137].

EXAMPLE 2.5—Second order backward differentiation method, BDF2
The backward differentiation methods [Hairer *et al.*, 1987, pp. 311–312] are a set of implicit multistep methods. The second order member in this set reads

$$y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + \frac{2h}{3}f(t_{n+1}, y_{n+1}).$$

assuming a constant stepsize h. □

A multistep method is called *linear* if $\bar{y}_n$ and $\bar{f}_n$ involves only linear combinations of old solution points and corresponding function values. The Adams-Bashforth method in Example 2.4 and the backward differentiation method in Example 2.5 are both linear multistep methods.

Another important class of multistep methods is the *oneleg* methods. Such methods use only one function evaluation per integration step. The BDF methods belong to this class. Another example is:

EXAMPLE 2.6—Implicit midpoint method
The implicit midpoint method [Hairer *et al.*, 1987, pp. 199-201] reads

$$y_{n+1} = y_n + h_n f\left(\frac{t_{n+1}+t_n}{2}, \frac{y_{n+1}+y_n}{2}\right).$$

□

Some of the definitions for different classes of integration methods overlap. The implicit midpoint method could for instance be regarded both as an implicit oneleg multistep method and an implicit onestep method. Similarly, the backward differentiation methods belong both to the class of linear multistep methods and the class of oneleg multistep methods.

2.2 Runge-Kutta Methods

Runge-Kutta methods belong to the class of onestep integration methods. They may be either explicit or implicit. An s-stage Runge-Kutta method applied to the problem (2.1) takes the form

$$\begin{aligned} \dot{Y}_i &= f(t_n + c_i h_n, y_n + h_n \sum_{j=1}^{s} a_{ij} \dot{Y}_j), \qquad i = 1 \ldots s \\ y_{n+1} &= y_n + h_n \sum_{j=1}^{s} b_j \dot{Y}_j \\ t_{n+1} &= t_n + h_n \end{aligned} \tag{2.3}$$

The intuitive idea behind a Runge-Kutta method is to sample the vector field $f(t, y)$ at a number of points in the neighborhood of the local solution. Based on these values ($\dot{Y}_i$) an “average” derivative on the interval $[t_n, t_{n+1}]$ is constructed, and the solution is updated in that direction (cf. Figure 2.2).

A convenient way to represent a Runge-Kutta method is to use the Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \end{array}$$

or shorter

$$\begin{array}{c|c} c & \mathcal{A} \\ \hline & b^T \end{array} \tag{2.4}$$

The structure of $\mathcal{A}$ gives rise to different types of Runge-Kutta methods. The most common ones are:

ERK Explicit Runge-Kutta, $\mathcal{A}$ is strictly lower triangular.

DIRK Diagonally Implicit Runge-Kutta, $\mathcal{A}$ is lower triangular.

SDIRK Singly Diagonally Implicit Runge-Kutta, $\mathcal{A}$ is lower triangular with all diagonal elements equal.

SIRK Singly Implicit Runge-Kutta, $\mathcal{A}$ is (in general) full, nondiagonalizable with one single s-fold eigenvalue.

FIRK Fully Implicit Runge-Kutta, $\mathcal{A}$ is full with no specific eigenstructure.

A nonautonomous problem $\dot{y} = f(t, y)$ can be translated to an autonomous problem $\dot{y} = f(y)$ by adding an extra equation $\dot{y}_{N+1} = 1$. The integration method should give the same result for the nonautonomous problem and its autonomous counterpart. It can be shown [Hairer *et al.*, 1987, pp. 142–143] that this puts the restriction

$$c = \mathcal{A}\mathbb{1}, \quad \mathbb{1} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T \tag{2.5}$$

on the method parameters.

Order Conditions

The coefficients in $\mathcal{A}$ and b (c is given by $\mathcal{A}$ through (2.5)) determine the properties of the Runge-Kutta method. Choosing them is a trade off between a number of properties, such as: method order, stability region, and error coefficients. We will refrain from deriving general formulas but rather demonstrate through an example. For a comprehensive treatment, the reader is referred to [Hairer *et al.*, 1987; Butcher, 1987].

EXAMPLE 2.7—A general explicit 2-stage Runge-Kutta method
A general explicit 2-stage Runge-Kutta method includes three parameters that need to be chosen. It has the Butcher tableau

$$\begin{array}{c|cc} 0 & 0 & 0 \\ a & a & 0 \\ \hline & b_1 & b_2 \end{array}$$

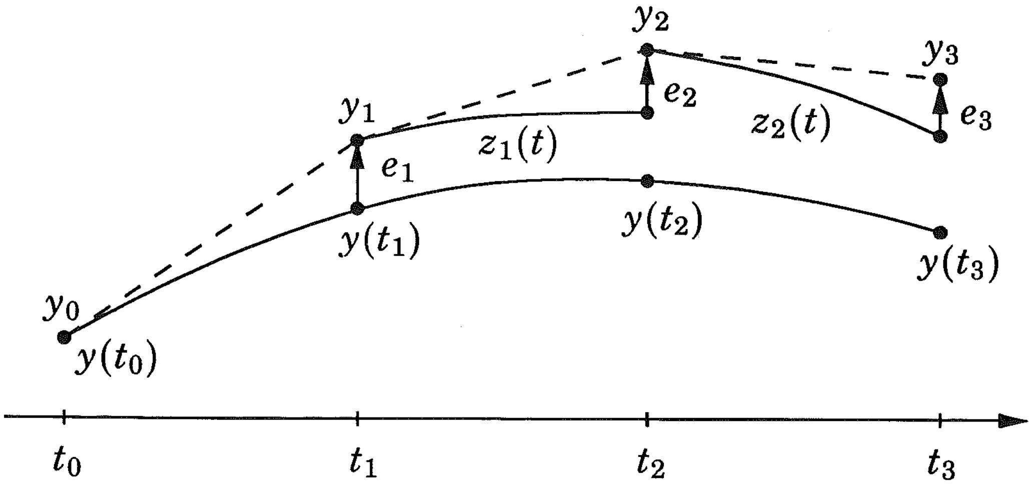


Figure 2.3 The local truncation error e_n in a Runge-Kutta method is the difference between the numerical solution y_n and the solution $z_n(t)$ of the local problem.

which corresponds to the explicit formulas

$$\begin{aligned} \dot{Y}_1 &= f(t_n, y_n) \\ \dot{Y}_2 &= f(t_n + ah_n, y_n + h_n a\dot{Y}_1) \\ y_{n+1} &= y_n + h_n(b_1\dot{Y}_1 + b_2\dot{Y}_2) \end{aligned}$$

The numerical solution thus takes the form

$$y_{n+1} = y_n + h_n\Big(b_1 f(t_n, y_n) + b_2 f\big(t_n + ah_n, y_n + h_n a f(t_n, y_n)\big)\Big).$$

Expanding the right hand side in a Taylor series leads to

$$\begin{aligned} y_{n+1} &= y_n + h_n(b_1 + b_2)f(t_n, y_n) + h_n^2 ab_2(f_t + f_y f)(t_n, y_n) \\ &\quad + \frac{h_n^3 a^2 b_2}{2}(f_{tt} + 2f_{ty}f + f_{yy}ff)(t_n, y_n) + O(h^4). \end{aligned} \tag{2.6}$$

A particular Runge-Kutta step inherits the previous approximation y_n as starting point, and the numerical solution can be thought of as following a local solution $z_n(t)$ (see Figure 2.3 and [Hairer *et al.*, 1987, p. 160]) defined by

$$\dot{z}_n = f(z_n(t)), \qquad z_n(t_n) = y_n. \tag{2.7}$$

The local solution can be expressed as a Taylor expansion about (t_n, y_n), i.e.

$$z_n(t_{n+1}) = z_n(t_n) + h_n \frac{dz_n}{dt}(t_n) + \frac{h_n^2}{2}\frac{d^2 z_n}{dt^2}(t_n) + \frac{h_n^3}{6}\frac{d^3 z_n}{dt^3}(t_n) + O(h_n^4) \tag{2.8}$$

where

$$\begin{aligned}
\frac{dz_n}{dt}(t_n) &= f(t_n, y_n) \\
\frac{d^2z_n}{dt^2}(t_n) &= (f_t + f_y f)(t_n, y_n) \\
\frac{d^3z_n}{dt^3}(t_n) &= (f_{tt} + 2f_{ty}f + f_{yy}ff + f_y f_t + f_y f_y f)(t_n, y_n).
\end{aligned}$$

Trying to match the numerical solution (2.6) with the local solution (2.8) leads to the *order conditions*

$$\begin{aligned}
p \geq 1: \quad b_1 + b_2 &= 1 \\
p \geq 2: \quad b_2 a &= 1/2
\end{aligned} \tag{2.9}$$

Choosing the method parameters according to (2.9) makes the numerical solution (2.6) and the local solution (2.8) identical for all terms up to and including order two. The third order term cannot be matched since the numerical solution (2.6) does not include the elementary differentials $f_y f_t$ and $f_y f_y f$.

The remaining difference between (2.6) and (2.8) is called the *local truncation error*, and belongs to the space U_h, cf. Figure 2.1. It is defined as

$$\begin{aligned}
e_{n+1} = y_{n+1} - z(t_{n+1}) = \frac{h_n^3}{6}\big((3b_2a^2 - 1)(f_{tt} + 2f_{ty}f + f_{yy}ff) \\
- (f_y f_t + f_y f_y f)\big)(t_n, y_n) + O(h_n^4)
\end{aligned} \tag{2.10}$$

Equation (2.9) specifies only two parameters. The remaining free parameter can be used to "minimize" the error, e.g. $3b_2a^2 = 1$, or to "maximize" the stability region. □

The type of derivations in Example 2.7 get very complicated when trying to obtain integration methods of high order. Since all (useful) Runge-Kutta methods obey (2.5) we may work with a problem formulation on autonomous form, i.e. $\dot{y} = f(y)$. This reduces the number of different elementary differentials substantially. Moreover, elementary differentials can be represented as trees [Hairer *et al.*, 1987; Butcher, 1987] each having its unique order condition, leading to a very clean notation.

It is possible to make some rather general observations from Example 2.7. The achievable method order is related to the number of stages.

Above it was only possible to obtain a second order method. Higher order methods require more stages. The problem of determining exactly how many stages that are required to achieve a certain order is only partly solved [Hairer *et al.*, 1987, Section II.6].

The order conditions do not normally give a unique specification of the parameters of the method. The freedom that is left can be used to optimize certain desirable properties. Often one tries, as in Example 2.7, to "maximize" the stability region or to "minimize" certain error coefficients.

As can be seen from (2.10) the local truncation error is problem dependent, in that it depends on many elementary differentials of f evaluated along the solution of the problem. The first nonmatched terms in the Taylor expansion are normally not complete, e. g. in (2.10) part of the third order term is affected by the parameter choice. In general, two methods with the same order need not have the same error terms, and consequently their errors need not be proportional.

Error Estimation

When solving (2.1) numerically the integration error should be kept small. To do this it must be possible to measure the error. The global error $g_n = y_n - y(t_n)$ is the fundamental measure of the integration error. This quantity is not computable (it requires that the true solution is known), and in general, it is even difficult to estimate [Skeel, 1986]. The global error consists of propagated and accumulated local truncation errors (cf. Figure 2.3), and most integration methods resort to estimating and controlling the latter. Once a local truncation error is made, the numerical solution will start tracking a neighboring (possibly divergent) solution. Through the local truncation errors it is possible to calculate bounds on the global error [Dahlquist *et al.*, 1974, p. 336]. These bounds are, however, often quite pessimistic.

The local truncation error can, for an order p method, be written

$$e_{n+1} = \phi(t_n)h_n^{p+1} + O(h_n^{p+2}) \tag{2.11}$$

where $\phi(t)$ is a smooth function (assuming $f \in C^{p+1}$) of t, cf. (2.10). The local truncation error usually depends on several elementary differentials of f of order $p+1$ and higher. It is in general not computable, but can be estimated. *Embedded* Runge-Kutta methods provide an extra set of b-parameters, $\hat{b}$, to advance the solution from t_n to t_{n+1}, i. e.

$$\begin{array}{c|c} c & \mathcal{A} \\ \hline y & b^T \\ \hline \hat{y} & \hat{b}^T \end{array}$$

where

$$y_{n+1} = y_n + h_n \sum_{j=1}^{s} b_j \dot{Y}_j, \qquad \hat{y}_{n+1} = y_n + h_n \sum_{j=1}^{s} \hat{b}_j \dot{Y}_j.$$

The two sets of parameters are chosen so that y_{n+1} and $\hat{y}_{n+1}$ correspond to methods of order p and $p+1$, respectively. Hence, the quantity $\|y_{n+1} - \hat{y}_{n+1}\|$ gives an asymptotically correct estimate of the norm of the local truncation error of y_{n+1}. It is sometimes also of interest to consider the measure $\|y_{n+1} - \hat{y}_{n+1}\|/h_n$. This latter quantity is called error-per-unit-step (EPUS), while the former is called error-per-step (EPS). The heuristic motive to use EPUS is to get the same "accumulated global error" for a fixed integration time regardless of the number of steps used.

Example 2.8—Explicit Euler with embedded error estimator
Consider the explicit 2-stage Runge-Kutta in Example 2.7, and choose the parameters as

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline y & 1 & 0 \\ \hline \hat{y} & 1/2 & 1/2 \end{array}$$

The first stage (read out by b) corresponds to explicit Euler (cf. Example 2.1). Combining both stages ($\hat{b}$) leads to a second order method (modified Euler). The local truncation error for the two methods are

$$e_{n+1} = -\frac{h_n^2}{2} f_y f - \frac{h_n^3}{6}\left(f_{yy} f f + f_y f_y f\right) + O(h_n^4) \quad \text{(expl Euler)} \qquad (2.12a)$$

$$e_{n+1} = -\frac{h_n^3}{6}\left(-\frac{1}{2} f_{yy} f f + f_y f_y f\right) + O(h_n^4) \quad \text{(mod Euler)} \qquad (2.12b)$$

where all function evaluations are done at y_n, and the $O(h_n^4)$-terms are identical. Forming the difference of the two outputs gives the error estimate

$$\hat{e}_{n+1} = y_{n+1} - \hat{y}_{n+1} = -\frac{h_n^2}{2} f_y f - \frac{h_n^3}{4} f_{yy} f f. \qquad (2.13)$$

□

Although the error estimate is asymptotically correct (it recovers the leading term in (2.12a)) it may still over- or underestimate the true error considerably. There may be large error contributions from high order elementary differentials that are not captured by the estimate (cf. (2.12) and (2.13) in Example 2.8). Moreover, some early methods have incomplete error estimators, which are asymptotically correct only for certain classes of problems. The error estimator in the Merson method [Hairer *et al.*, 1987, p. 169] is correct only for linear differential equations with constant coefficients.

There are many ways to measure the size of the vector error estimate (2.13). For robustness, a mixed absolute-relative "norm" is often used. Two common choices are (see [Higham and Hall, 1989] for a list of norms used in different codes)

$$||\hat{e}|| = \max_i \left| \frac{\hat{e}_i}{\tilde{y}_i + \eta_i} \right|, \qquad ||\hat{e}|| = \sqrt{\sum_i \left(\frac{\hat{e}_i}{\tilde{y}_i + \eta_i} \right)^2} \qquad (2.14)$$

where $\tilde{y}_i$ is (a possibly smoothed) absolute value of y_i, and η_i is a scaling factor for that component of y. In the case of EPUS a normalization with h is also used, and we arrive at the scalar error estimate

$$r_{n+1} = ||\hat{e}_{n+1}|| \quad \text{(EPS)}, \qquad r_{n+1} = ||\hat{e}_{n+1}||/h_n \quad \text{(EPUS)} \qquad (2.15)$$

Local Extrapolation

Which one of y and $\hat{y}$ should be used as the new solution point? On the one hand, the estimate of the local truncation error is only asymptotically correct if the low order formula is used, while on the other hand, the high order formula usually gives a better numerical solution. In practice both alternatives are used, and choosing the high order formula is referred to as *local extrapolation*.

There are four possible alternatives for combining error estimate and solution update:

EPS	error-per-step, no local extrapolation
XEPS	error-per-step, local extrapolation
EPUS	error-per-unit-step, no local extrapolation
XEPUS	error-per-unit-step, local extrapolation

All four modes have been used in practice, but often XEPS is regarded as the most efficient [Shampine, 1977].

Since it is not possible to measure the global error, most error control schemes concentrate on keeping an estimate of the local truncation error bounded. One would hope that such a scheme leads to that the global error decreases with the control level for the local truncation error (so-called *tolerance proportionality* [Stetter, 1976; Stetter, 1980]), but this is not always true. It can be shown that to achieve tolerance proportionality, XEPS or EPUS has to be used [Higham, 1990].

Interpolant and Defect

The numerical solution y_n is only given at the grid points t_n. It is often useful to have available a continuous function $q(t)$ which approximates $y(t)$ between the grid points. This allows for dense solution output without having to restrict the distance between consecutive grid points. The stepsize can, consequently, be chosen based on the required accuracy of the numerical solution, instead of being restricted by the desired density of solution points. Many Runge-Kutta methods supply a *continuous extension* $q(t)$ of the form

$$q(t_n + vh_n) = y_n + h_n \sum_{j=1}^{\hat{s}} b_j(v)\dot{Y}_j, \tag{2.16}$$

where each $b_j(v)$ is a polynomial in v [Hairer *et al.*, 1987; Dormand and Prince, 1986; Enright, 1989b; Shampine, 1985a; Shampine, 1986]. If $\hat{s} > s$, extra stages have been added to the original method. The continuous extension is normally chosen to satisfy $q(t_n) = y_n$ and $q(t_{n+1}) = y_{n+1}$, and is therefore an interpolant. The derivatives at the mesh points are often also matched, i. e. $\dot{q}(t_n) = f(t_n, y_n)$ and $\dot{q}(t_{n+1}) = f(t_{n+1}, y_{n+1})$.

The accuracy of an interpolant is related to its order. The extension $q(t)$ approximates a local solution to the differential equation, and the local order of $q(t)$ is l if for any fixed $v \in [0, 1]$

$$q(t_n + vh_n) = z_n(t_n + vh_n) + O(h_n^l)$$

when $h_n \to 0$. Here $z_n(t)$ is the local solution defined by (2.7). The order of the interpolant is, naturally, chosen in relation to the method order, and normally $l = p+1$ or $l = p$. Having $l = p+1$ normally requires $\hat{s} > s$.

It has been suggested to use the interpolant to estimate and control the integration error [Enright, 1989a; Enright, 1989b; Higham, 1989a; Higham, 1989b]. In each step the local defect

$$\delta(t) := \dot{q}(t) - f(t, q(t)),$$

is calculated and a sample $\delta(t_n + v^* h_n)$ of this quantity is controlled. This scheme corresponds to an error control in the space V_h instead of U_h, since the defect is an element in V_h (cf. Figure 2.1). The relation between $\delta(t)$ and the defect d^h is similar to the one between the local truncation error per unit step e_n/h_n and the global error g^h.

It is possible to construct the interpolant $q(t)$ so that one sample at a predefined point v^* gives an asymptotically correct estimate of the maximum defect over the interval $v \in [0,1]$ [Higham, 1989a; Higham, 1989b]. The cost may, however, be rather high, since the formation of $q(t)$ often requires extra stages. To get tolerance proportionality the order of the interpolant must be chosen so that $l = p+1$ [Higham, 1990].

The Test Equation

The discretization (2.2) of (2.1) relates to the concept of system sampling in sampled-data control, e. g. [Åström and Wittenmark, 1990, Chapters 3, and 8.1–8.2], [Franklin *et al.*, 1990, Chapter 4]. We are trying to construct a discrete-time system that mimics the behavior of a continuous-time system. In numerical integration both the differential equation and the difference equation are in general nonlinear, which makes the problem difficult to analyze. The standard analysis is to use a linear test equation with constant coefficients, i. e. $\dot{y} = Ay$. Runge-Kutta methods are diagonalizable (the discretization and matrix diagonalization commute), which simplifies the analysis and it is adequate to study the scalar test equation

$$\dot{y} = \lambda y. \tag{2.17}$$

We derive the difference equation resulting when applying a Runge-Kutta method to (2.17) and then list the result for the general case $\dot{y} = Ay$.

Applying an s-stage Runge-Kutta method (2.3) to the problem (2.17) results in

$$\dot{Y} = \lambda y_n \mathbb{1} + h_n \lambda \mathcal{A} \dot{Y} \quad \Rightarrow \quad h_n \dot{Y} = (I - h_n \lambda \mathcal{A})^{-1} \mathbb{1} h_n \lambda y_n$$

where

$$\dot{Y} = \begin{pmatrix} \dot{Y}_1 & \dot{Y}_2 & \dots & \dot{Y}_s \end{pmatrix}^T .$$

Introduce $z = h_n \lambda \in \mathbb{C}$, and $y_{n+1} = y_n + b^T h_n \dot{Y}$ yields

$$y_{n+1} = \left(1 + z b^T (I - z\mathcal{A})^{-1} \mathbb{1}\right) y_n = P(z) y_n \tag{2.18}$$

where $P(z)$ is a rational function in z, with the highest degree of z being less than or equal to s. In an explicit Runge-Kutta method the matrix $\mathcal{A}$ is nilpotent and consequently $P(z)$ is polynomial with degree less than or equal to s.

Similarly to (2.18),

$$\begin{aligned} \hat{y}_{n+1} &= \left(1 + z\hat{b}^T (I - z\mathcal{A})^{-1} \mathbb{1}\right) y_n = \hat{P}(z) y_n \\ \hat{e}_{n+1} &= z(b - \hat{b})^T (I - z\mathcal{A})^{-1} \mathbb{1} y_n = E(z) y_n \end{aligned} \tag{2.19}$$

where $\hat{P}(z)$ and $E(z)$ are, again, rational functions in z, with the highest degree of z being less than or equal to s.

The way a Runge-Kutta method is constructed leads to special properties for the Taylor expansions of $P(z)$, $\hat{P}(z)$, and $E(z)$. The expansions about $z = 0$ of $P(z)$ and $\hat{P}(z)$ match the Taylor expansion of e^z. All terms are identical up to and including $O(z^p)$ and $O(z^{p+1})$, respectively. The function $E(z)$ is the difference between $P(z)$ and $\hat{P}(z)$, and therefore the $O(z^{p+1})$-term is the first nonzero term in its expansion about $z = 0$.

For the general case $\dot{y} = Ay$, where $y \in \mathbb{R}^N$, the equations corresponding to (2.18) and (2.19) read

$$y_{n+1} = P(h_n A) y_n, \qquad \hat{y}_{n+1} = \hat{P}(h_n A) y_n, \qquad \hat{e}_{n+1} = E(h_n A) y_n$$

with

$$\begin{aligned} P(h_n A) &= y_n + \left(b^T \otimes I_N\right) \left(I_s \otimes I_N - \mathcal{A} \otimes h_n A\right)^{-1} \left(\mathbb{1} \otimes h_n A y_n\right) \\ \hat{P}(h_n A) &= y_n + \left(\hat{b}^T \otimes I_N\right) \left(I_s \otimes I_N - \mathcal{A} \otimes h_n A\right)^{-1} \left(\mathbb{1} \otimes h_n A y_n\right) \\ E(h_n A) &= \left((b - \hat{b})^T \otimes I_N\right) \left(I_s \otimes I_N - \mathcal{A} \otimes h_n A\right)^{-1} \left(\mathbb{1} \otimes h_n A y_n\right) \end{aligned} \tag{2.20}$$

where I_N and I_s are identity matrices of size N and s, respectively, and $\otimes$ is the Kronecker product. For an explicit method the functions $P(h_n A)$, $\hat{P}(h_n A)$, and $E(h_n A)$ are matrix polynomials.

Stability Region

The difference equation (2.2) should mimic the differential equation (2.1). A basic requirement is that for constant h_n it is stable when the differential equation is stable. Even for linear problems this is difficult to achieve for all h_n.

Consider the linear test equation $\dot{y} = Ay$. For which h_n is the difference equation (2.3) stable? The diagonalization property of Runge-Kutta methods simplifies the analysis, and it suffices to consider (2.17) with λ equal to the eigenvalues of A. We arrive at the difference equation (2.18), which is stable if $|P(z)| \leq 1$. The z-region where this is true defines the stability region of the integration method.

DEFINITION 2.4—Region of (absolute) stability
When applying a Runge-Kutta method to the linear test equation (2.17) the resulting difference equation will be stable as long as $z \in \mathcal{S}$,

$$\mathcal{S} = \{z \in \mathbb{C} : |1 + zb^T (I - z\mathcal{A})^{-1} \mathbb{1}| \leq 1\}$$

where $z = h_n\lambda$. $\mathcal{S}$ defines the region of absolute stability of the integration method. □

EXAMPLE 2.9—Stability region of the modified Euler method
Consider the explicit 2-stage Runge-Kutta method in Example 2.7. Here

$$P(z) = 1 + z \begin{pmatrix} b_1 & b_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -za & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 1 + (b_1 + b_2)z + b_2 a z^2.$$

From the order conditions (2.9) we obtain

$$P(z) = 1 + z + z^2/2.$$

The stability region $\mathcal{S}$, i.e. the region where $|P(z)| \leq 1$, is depicted in Figure 2.4. □

It is very desirable to have integration methods where the whole left halfplane $\mathbb{C}^-$ belongs to the stability region. Such methods correspond to discretizations that are stable for $\operatorname{Re} \lambda < 0$.

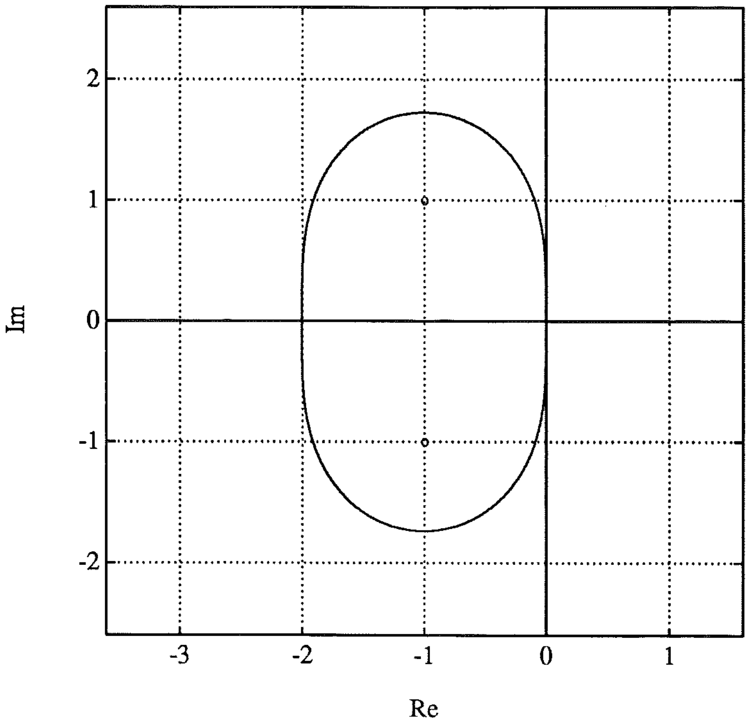


Figure 2.4 The stability region $\mathcal{S}$ of the explicit Runge-Kutta method in Example 2.9. The two rings mark the zeros of $P(z)$.

DEFINITION 2.5—A-stability [Dahlquist, 1963; Hairer and Wanner, 1991]
An integration method where

$$\mathcal{S} \supset \mathbb{C}^{-} = \{z : \mathrm{Re}\, z \leq 0\}$$

is called A-stable. □

Consider again the concept of system sampling in sampled-data control. A-stability guarantees that the discretization retains the stability of the continuous-time system. It is, however, also desirable that the discrete-time system and the continuous-time system have similar dynamical properties, e.g. a fast continuous-time eigenvalue should be mapped to a fast discrete-time eigenvalue. This relates to the stronger property L-stability.

DEFINITION 2.6—L-stability [Ehle, 1969; Hairer and Wanner, 1991]
An integration method is said to be L-stable if it is A-stable, and in addition

$$\lim_{z \to \infty} P(z) = 0$$

□

Discretization Accuracy

The stability region only tells part of the story when trying to determine how well the difference equation (2.2) matches the original differential equation (2.1). Of greater importance is the deviation between $P(z)$ (2.18), or $\hat{P}(z)$ (2.19), from e^z. Consider, again, the linear test equation (2.17). Assuming $y(t_n) = y_n$, we have

$$y(t_n + h_n) = e^{h_n\lambda} y_n, \qquad y_{n+1} = P(h_n\lambda) y_n, \qquad \hat{y}_{n+1} = \hat{P}(h_n\lambda) y_n,$$

with the local truncation error and estimate given by

$$\begin{aligned} e_{n+1} &= y_{n+1} - y(t_n + h_n) = \left(P(h_n\lambda) - e^{h_n\lambda}\right) y_n, \\ \hat{e}_{n+1} &= y_{n+1} - \hat{y}_{n+1} = E(h_n\lambda) y_n. \end{aligned}$$

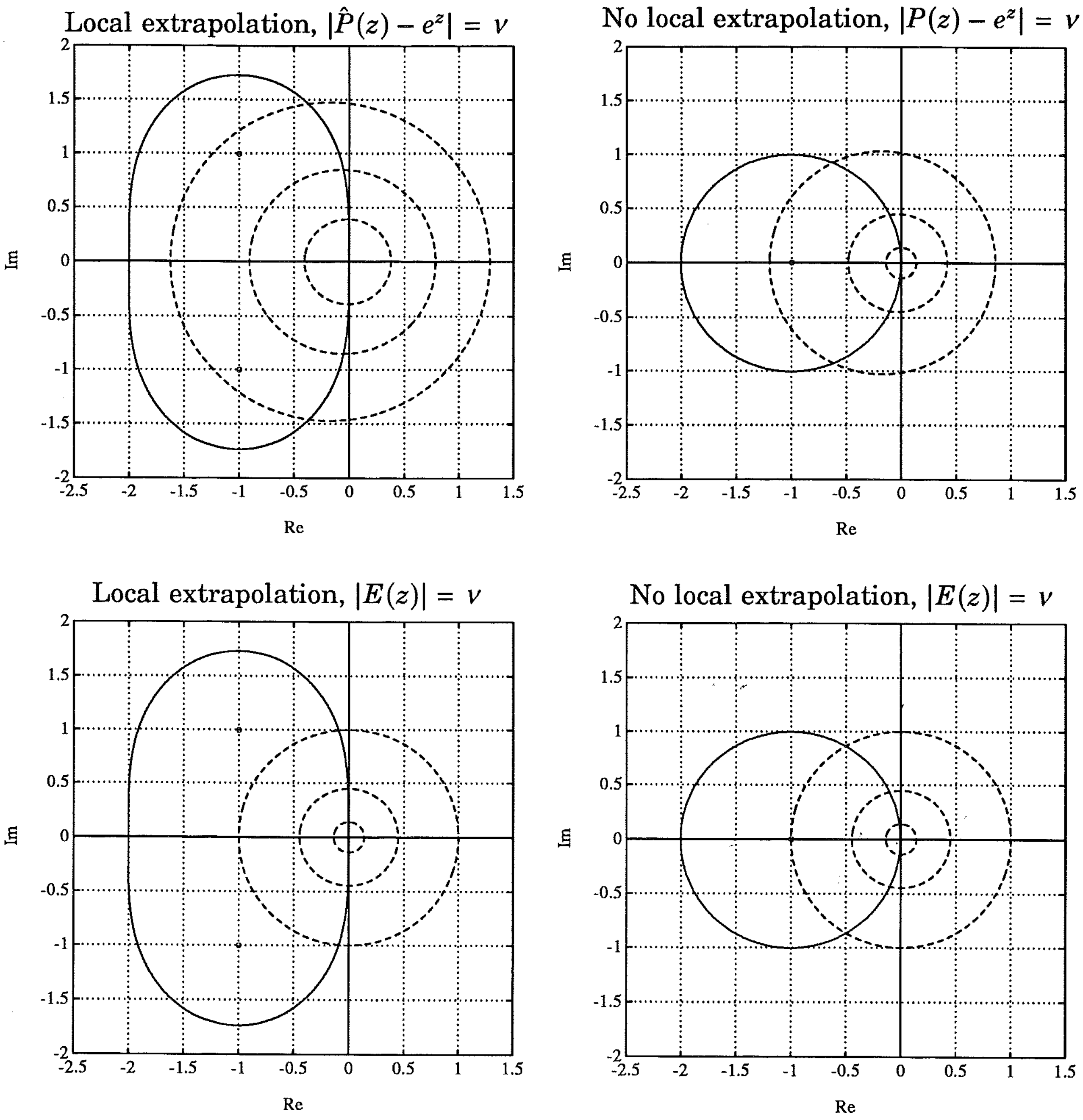
Introduce $z = h_n\lambda \in \mathbb{C}$. When taking one integration step the local truncation error is governed by $P(z) - e^z$, or $\hat{P}(z) - e^z$ in case of local extrapolation. Similarly, the local truncation error estimate is governed by $E(z)$. By restricting the stepsize so that $|E(z)| < \nu$ the local truncation error estimate (and then hopefully also the true local truncation error) will be kept below νy_n.

EXAMPLE 2.10—Discretization accuracy
Consider the embedded Runge-Kutta method in Example 2.8. For this method

$$P(z) = 1 + z, \qquad \hat{P}(z) = 1 + z + z^2/2, \qquad E(z) = -z^2/2.$$

Figure 2.5 depicts the stability regions as well as some level curves for $|P(z) - e^z|$, $|\hat{P}(z) - e^z|$, and $|E(z)|$.

The integration method is of second or first order depending on if local extrapolation is used or not. For the same accuracy, cf. Figure 2.5, the higher order method allows the use of a larger z, and hence a larger stepsize. □



Local extrapolation, $|\hat{P}(z) - e^z| = \nu$

No local extrapolation, $|P(z) - e^z| = \nu$

Local extrapolation, $|E(z)| = \nu$

No local extrapolation, $|E(z)| = \nu$

Figure 2.5 The stability regions (full line) of the embedded Runge-Kutta method in Example 2.10. The level curves (dashed lines) for the functions governing the local truncation error (upper plots) and the local truncation error estimate (lower plots) are depicted for $\nu = 0.01$, 0.1, and 0.5. As can be seen z has to be close to 0 in order to get an accurate numerical solution. In the nonextrapolated case the error estimate would be close to the true error, while it would be overestimating the true error in case of local extrapolation.

2.3 The Standard Stepsize Selection Rule

The integration method is supposed to produce a numerical solution within the error tolerance *tol*. Ideally, the global error should be below *tol*, but the global error is difficult to estimate and most integration methods confine themselves to keeping an estimate of the local truncation error close to ε, where ε is chosen in relation to *tol*. This is achieved by adjusting the stepsize during the integration.

The standard stepsize selection rule [Gear, 1971; Dahlquist *et al.*, 1974; Hairer *et al.*, 1987] is based on the relation (2.11). The error estimate r (2.15) measures the norm of $\hat{e}$ and consequently the asymptotic behavior of r is given by

$$r_n = \varphi_{n-1} h_{n-1}^k, \tag{2.21}$$

where $k = p + 1$ for EPS and $k = p$ for EPUS. The coefficient vector $\varphi_n = \|\phi(t_n) + O(h_n)\|$, cf. (2.11), is assumed constant or slowly varying. In order to achieve $r_{n+1} = \varepsilon$ the next stepsize h_n is chosen as

$$h_n = \left(\frac{\varepsilon}{r_n}\right)^{1/k} h_{n-1}. \tag{2.22}$$

If the error in a step is too large, e.g. $r_n > \rho\, tol$, the step is rejected, and a new try is made with smaller stepsize. By having $\varepsilon < tol$ and $\rho > 1$, typically $0.2\, tol < \varepsilon < 0.8\, tol$ and $1 < \rho < 1.3$, a safety factor is introduced in the choice of stepsize, and the risk for a rejected step is reduced, cf. Figure 2.6.

Occasionally, the error estimator may produce an unusually small (or large) value, thus advocating a very large stepsize change. To improve robustness the controller normally includes some limitation on such changes. In an implicit integration method the choice of stepsize also affects the performance of the equation solver. This has to be considered when choosing stepsize, and most implicit methods augment (2.22) with further restrictions.

2.4 Equation Solvers

In an implicit method it is necessary to solve for $\dot{Y}_i$ in (2.3). In general, $f(t, y)$ is a nonlinear function and the equation (2.3) cannot be solved analytically. Instead an iterative numerical equation solver is normally used.

When considering the equation solver, the Runge-Kutta method (2.3) is often rewritten using $\dot{Y}_i = f(t_n + c_i h_n, Y_i)$ as

$$\begin{aligned} Y_i &= y_n + h_n \sum_{j=1}^{s} a_{ij} f(t_n + c_j h_n, Y_j), \qquad i = 1 \dots s \\ y_{n+1} &= y_n + h_n \sum_{j=1}^{s} b_j f(t_n + c_j h_n, Y_j) \\ t_{n+1} &= t_n + h_n \end{aligned} \tag{2.23}$$

and we solve for Y_i instead of $\dot{Y}_i$. Introduce

$$Y := \begin{pmatrix} Y_1 \\ \vdots \\ Y_s \end{pmatrix}, \qquad F(Y) := \begin{pmatrix} f(t_n + c_1 h_n, Y_1) \\ \vdots \\ f(t_n + c_s h_n, Y_s) \end{pmatrix},$$

and (2.23) takes the form of the implicit equation, cf. (2.20)

$$Y = \mathbb{1} \otimes y_n + (h_n \mathcal{A} \otimes I_N) F(Y) \tag{2.24a}$$

$$y_{n+1} = y_n + h_n \left(b^T \otimes I_N\right) \otimes F(Y). \tag{2.24b}$$

The equation system in (2.24a) is of order $Ns \times Ns$, but it is often possible to make substantial simplifications. One example is in DIRK where $\mathcal{A}$ is lower triangular. It is then possible to successively solve for each stage separately (s systems of order $N \times N$). Other methods may lead to other types of simplifications [Butcher, 1976; Bickart, 1977; Burrage, 1978].

Many different types of equation solvers have been used to solve (2.24a) in practice. Which one to choose depends both on the problem as well as on the integration method itself. The two standard choices are fixed point iteration and modified Newton iteration.

Fixed Point Iteration

The simplest way to solve (2.24a) is to use fixed point iteration, i. e. iterate

$$Y^{m+1} = \mathbb{1} \otimes y_n + (h_n \mathcal{A} \otimes I_N) F(Y^m),$$

until the solution is sufficiently accurate.

The starting value Y^0 for the iteration is often obtained using the interpolant $q(t)$ (2.16) from the previous step. Although $q(t)$ was designed primarily to interpolate within a step, it normally gives reasonably good extrapolations. During successful numerical integration the stepsize h_n is kept on scale, i.e. h_n is chosen so that the solution change is moderate between successive steps, and consequently the extrapolation will normally be good. Should the integration method not supply an interpolant, it is still often possible to use old stage values to construct a crude extrapolant to obtain Y^0.

Under certain conditions there exist a unique solution Y^* to (2.24a) [Kraaijevanger and Schneid, 1991]. The iteration error $E^m = Y^m - Y^*$ can be written

$$E^{m+1} = (h_n \mathcal{A} \otimes I_N) \bar{J} E^m \tag{2.25}$$

where $\bar{J}$ is a mean value Jacobian [Ortega and Rheinboldt, 1970, 3.2.6]

$$\bar{J} = \int_0^1 \frac{\partial F}{\partial Y} (Y^* + \zeta (Y^m - Y^*)) \, d\zeta.$$

The matrix $\bar{J}$ is block diagonal with s blocks. Block number i is formed as $\int_0^1 \frac{\partial f}{\partial y} (t_n + c_i h_n, Y_i^* + \zeta (Y_i^m - Y_i^*)) \, d\zeta$. For the test equation $\dot{y} = Ay$ the mean value Jacobian takes the form $\bar{J} = I_s \otimes A$.

The iteration (2.25) converges if it is a contraction,

$$\left\| (h_n \mathcal{A} \otimes I_N) \bar{J} \right\| < 1. \tag{2.26}$$

We observe that to obtain convergence the stepsize h_n must be limited in correspondence with the magnitude of the eigenvalues of $(\mathcal{A} \otimes I_N)\bar{J}$. If the eigenvalues have approximately the same magnitude this is quite natural; when the solution $y(t)$ changes quickly (large eigenvalues) a small stepsize is necessary. If, on the other hand, there is a large span of eigenvalues the stepsize must be chosen in relation to the largest eigenvalues, while $y(t)$ may be governed by some of the smaller ones. A differential equation with this property is referred to as *stiff* [Shampine and Gear, 1979; Cash, 1985; Shampine, 1985b; Byrne and Hindmarsh, 1987]. The restriction on the stepsize can be very severe, which makes the integration method inefficient. Hence a different type of equation solver should be used.

Modified Newton Iteration

By using Newton iteration it is possible to obtain convergence in the equation solver without the severe stepsize limitations for fixed point iteration. Newton's method applied to (2.24a) needs for each iteration the solution of a linear system with the matrix (compare with the structure of $(h_n\mathcal{A} \otimes I_N)\bar{J})$

$$\begin{pmatrix} I - h_n a_{11}\frac{\partial f}{\partial y}(t_n + c_1 h_n, Y_1^m) & \dots & -h_n a_{1s}\frac{\partial f}{\partial y}(t_n + c_s h_n, Y_s^m) \\ \vdots & \ddots & \vdots \\ -h_n a_{s1}\frac{\partial f}{\partial y}(t_n + c_1 h_n, Y_1^m) & \dots & I - h_n a_{ss}\frac{\partial f}{\partial y}(t_n + c_s h_n, Y_s^m) \end{pmatrix} \tag{2.27}$$

It would be too expensive to evaluate (2.27) at each iteration, and it is common to use the approximation

$$\frac{\partial f}{\partial y}(t_n + c_i h_n, Y_i^m) \approx \frac{\partial f}{\partial y}(t_n, y_n) =: J.$$

This replaces (2.27) with the *iteration matrix* M, defined as

$$M = I_s \otimes I_N - h_n\mathcal{A} \otimes J.$$

Define the residual of (2.24a) as

$$R(Y) := \mathbb{1} \otimes y_n + (h_n\mathcal{A} \otimes I_N)F(Y) - Y, \tag{2.28}$$

and the Newton iteration for (2.24a) takes the form

$$Y^{m+1} = Y^m + M^{-1}R(Y^m). \tag{2.29}$$

In (2.29) an $Ns \times Ns$ linear system has to be solved each iteration. If $\mathcal{A}$ is triangular (DIRK) the problem is reduced to solving s systems of size $N \times N$. Moreover, if the diagonal elements in $\mathcal{A}$ are equal (SDIRK) the same iteration matrix can be used for all the stages. Similar simplifications can be done for $\mathcal{A}$ possessing other types of structure.

The approximation $I_s \otimes J$ differs from the (unknown) mean value Jacobian $\bar{J}$ of $F(Y)$. Similarly to (2.25)

$$E^{m+1} = \left(I_s \otimes I_N - h_n \mathcal{A} \otimes J\right)^{-1} \left((h_n \mathcal{A} \otimes I_N)(\bar{J} - I_s \otimes J)\right) E^m, \qquad (2.30)$$

and for contraction

$$\left\| \left(I_s \otimes I_N - h_n \mathcal{A} \otimes J\right)^{-1} \left((h_n \mathcal{A} \otimes I_N)(\bar{J} - I_s \otimes J)\right) \right\| < 1. \qquad (2.31)$$

As for fixed point iteration the convergence depends on the stepsize h_n. The dependence on the span in the magnitude of the eigenvalues of the Jacobian $\bar{J}$ is, however, reduced due to the premultiplication by $(I_s \otimes I_N - h_n \mathcal{A} \otimes J)^{-1}$. To get convergence we do not need to know $\bar{J}$ exactly. The iteration will converge as long as $\|\bar{J} - I_s \otimes J\|$ is moderate. Newton iteration thus makes it possible to obtain efficient integration methods also for stiff problems.

Newton iteration is computationally expensive. To save computations the same iteration matrix M is used over several integration steps. The approximate matrix M and the fact that it is used during several steps is the reason for the term *modified* Newton iteration. Storing M as an LU decomposition further reduces the operation count. It also facilitates solving for Y^{m+1} in (2.29). The modified Newton iteration will converge as long as the Jacobian or the stepsize do not deviate too much from the values used when forming M (cf. [Alexander, 1991] for more precise conditions). The rules for when to evaluate the Jacobian and/or factorize the iteration matrix are important decisions to assure a correct solution and to make the implementation of the integration method efficient [Shampine, 1980; Hairer and Wanner, 1991, pp. 128–141]. We will return to this problem in Chapter 5.

The Iteration Error in the Equation Solver

In each step of the integration there are two types of error contributions: the discretization error and the error from the inexact solution of (2.24a). The first term is measured either as an estimate of the local truncation error or the defect, and its contribution can be controlled through the stepsize. The second term is measured either as the displacement $\Delta_m = Y^{m+1} - Y^m$ or as the residual $R(Y^m)$ (2.28), and its size is affected by the number of iterations in the equation solver. In principal, the displacement is an element in the space U_h, while the residual belongs to V_h, cf. Figure 2.1. For a consistent error control one either combines the use of the local truncation error with the displacement (error control in U_h), or combines the defect with the residual (error control in V_h). The most common choice is the former [Houbak *et al.*, 1985].

The stepsize control algorithm keeps the local truncation error below some value ε, and the equation solver continues to iterate until the displacement is below τ. The value ε is, normally, chosen based on the user-specified tolerance *tol*, and the remaining question is how τ should be related to ε. The iteration error should, naturally, not be too large. On the other hand, the smaller τ is chosen, the more it costs to compute $\hat{Y}$. Experiments show that the solution of the differential equation is not improved by making $\tau \ll \varepsilon$ [Shampine, 1980; Nørsett and Thomsen, 1986a; Hairer and Wanner, 1991, p. 131].

The local truncation error e is related to the local truncation error estimate $\hat{e}$ and the displacement Δ as [Houbak *et al.*, 1985; Nørsett and Thomsen, 1986a]

$$\hat{e}_n \approx e_n + \big((b - \hat{b})^T \mathcal{A}^{-1} \otimes I_N\big) M^{-1} \big((h_n \mathcal{A} \otimes I_N)(\bar{J} - I_s \otimes J)\big)\Delta_n. \quad (2.32)$$

The quantity $\|M^{-1}\left((h_n\mathcal{A} \otimes I_N)(\bar{J} - I_s \otimes J)\right)\|$ is an estimate of the rate of convergence (cf. (2.30)), which must be less than 1. The contribution of the second term in (2.32) is therefore bounded by $\|(b - \hat{b})^T \mathcal{A}^{-1} \otimes I_N\|\tau$. By setting $J = 0$ it is easy to see that this holds also for fixed point iteration.

If the iteration error contribution to $\hat{e}$ is to be kept below a fraction κ of ε then

$$\tau = \left\|(b - \hat{b})^T \mathcal{A}^{-1} \otimes I_N\right\|^{-1} \kappa\varepsilon. \quad (2.33)$$

Few implementations calculate τ based on κ and the method coefficients (2.33). Instead τ is set at a fixed fraction of ε. A value τ/ε around 10^{-1} or 10^{-2} is common [Hairer and Wanner, 1991, p. 131].

Let $\hat{Y}$ denote the numerical solution to (2.24a). The structure of (2.24b) suggests that, after having obtained $\hat{Y}$, one should evaluate $F(\hat{Y})$ in order to calculate the new solution point. The value $\hat{Y}$ is, however, inaccurate, and the actual evaluation of $f(t_n + c_i h_n, Y_i)$ would, especially for stiff problems, amplify these errors [Shampine, 1980]. Instead, we solve for $F(\hat{Y})$ in (2.24a),

$$F(\hat{Y}) := \big((h_n\mathcal{A})^{-1} \otimes I_N\big)\big(\hat{Y} - \mathbb{1} \otimes y_n\big), \quad (2.34)$$

and use this value in (2.24b). Through (2.34) we also save one function evaluation per stage.

Rate of Convergence and Stiffness

Let L be the Lipschitz constant of the iteration map in (2.25) or (2.30). If $L < 1$ the iteration is contractive. The convergence in the equation solver can be monitored by estimating the rate of convergence by

$$\alpha = \max_m \alpha_m = \max_m \frac{||\Delta_m||}{||\Delta_{m-1}||}, \tag{2.35}$$

and often α or α_m is used instead of L, although both underestimate L.

To get efficiency in the equation solver one would in practice not allow α to get close to 1, i.e. (say) $\alpha < 0.5$. Slow convergence when using fixed point iteration is an indication that the problem may turn stiff, and it is reasonable to switch to modified Newton iteration. In the case modified Newton iteration is already in use one would instead consider a refactorization of the iteration matrix and/or a reevaluation of the Jacobian (cf. Chapter 5).

Switching from modified Newton iteration back to fixed point iteration requires (2.26) to be satisfied. The quantity $||(h_n\mathcal{A} \otimes I_N)\bar{J}||$ can be estimated using variables available in the modified Newton iteration. Define the "stiffness measure" β as [Nørsett and Thomsen, 1986a]

$$\beta = \max_m \frac{||R(Y^m)||}{||\Delta_m||} = \max_m \frac{||M\Delta_m||}{||\Delta_m||}. \tag{2.36}$$

We have

$$\beta \leq ||M|| = ||I_s \otimes I_N - h_n\mathcal{A} \otimes J|| \leq 1 + ||h_n\mathcal{A} \otimes J||. \tag{2.37}$$

Assume that $I_s \otimes J$ is a good approximation to $\bar{J}$. When $||(h_n\mathcal{A} \otimes I_N)\bar{J}|| < 1$, and hence $\beta < 2$, we know that fixed point iteration converges. Some codes [Nørsett and Thomsen, 1987] therefore use $\beta < 2$ as an indication to switch from modified Newton iteration to fixed point iteration. The estimate (2.36) may, however, underestimate the norm of M, and $\beta < 2$ is necessary but not sufficient for $||(h_n\mathcal{A} \otimes I_N)\bar{J}|| < 1$. Some codes are more elaborate, and calculate $||J||$ in order to decide when to switch back to fixed point iteration.

2.5 Standard Error and Convergence Control

To guarantee efficiency and accuracy all numerical integration codes include algorithms for error control and, in case of an implicit method, convergence control. The stepsize is the main variable used to control the error (cf. Section 2.3), although the error also depends on the accuracy of the equation solver iteration, cf. (2.32). The convergence in the equation solver is assured by choosing the correct type of solver and updating the iteration matrix appropriately. Most control actions affect, however, both the error and the convergence of the iteration, and this cross-coupling has to be considered in the choice of control algorithms.

Choosing Stepsize

The standard rule to select the stepsize is (2.22). As (2.25) and (2.30) illustrate the stepsize also affects the convergence rate, and most control algorithms restrict stepsize changes so that the convergence is not jeopardized. The structure of (2.25) and (2.30) suggests a linear relation between h and α, and if $\alpha_{\max}$ is the worst acceptable convergence rate, the new stepsize must obey

$$h_n < \frac{\alpha_{\max}}{\alpha} h_{n-1}.$$

For efficiency the iteration matrix M in the equation solver is stored in LU-factorized form. When the stepsize is changed the factorization "should" be updated accordingly. The modified Newton iteration may, naturally, still converge; it all depends on how much the stepsize changes and/or if the Jacobian itself also has changed. Doing a factorization at every step would be very inefficient, and instead most control algorithms incorporate some logic that prevents the stepsize from changing too frequently. All in all the stepsize selection rule becomes rather involved.

Example 2.11—Stepsize selection in SIMPLE(2)3
SIMPLE(2)3 is a third order implicit Runge-Kutta implementation due to Nørsett and Thomsen, [Nørsett and Thomsen, 1984; Nørsett and Thomsen, 1987; Nørsett and Thomsen, 1990]. At each step SIMPLE(2)3 calculates

$$\mu_1 = \left(\frac{tol}{r_n}\right)^{1/k}, \qquad \mu_2 = \frac{\alpha_{\max}}{\alpha},$$

where $k = 3$ and $\alpha_{max} = 0.8$. The two values give an estimate of how much the stepsize could change without getting $r_{n+1} > tol$ or $\alpha > \alpha_{\max}$,

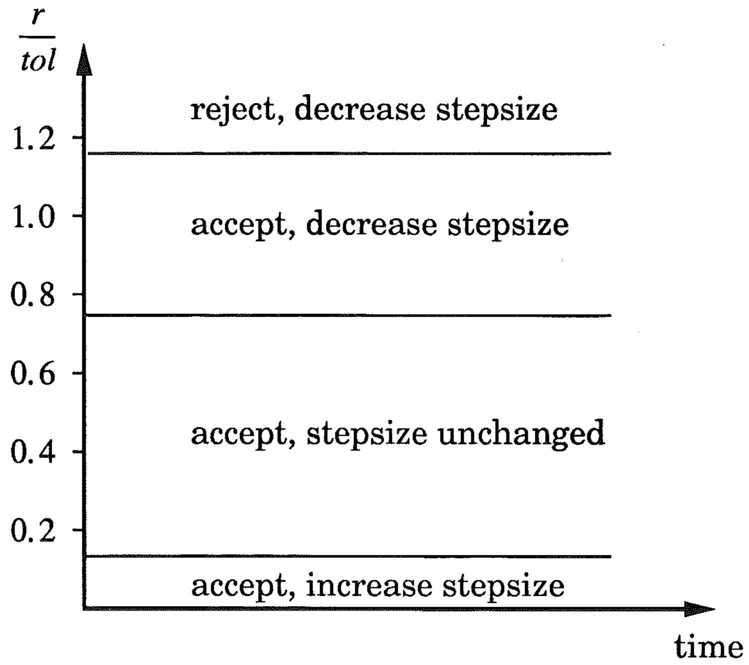

$\frac{r}{tol}$
1.2
1.0
0.8
0.6
0.4
0.2
reject, decrease stepsize
accept, decrease stepsize
accept, stepsize unchanged
accept, increase stepsize
time

Figure 2.6 Depending on the value of r/tol SIMPLE(2)3 uses different rules for choosing the next stepsize.

respectively. The choice of stepsize depends on μ_1 and μ_2 as

$$h_n = \begin{cases} 0.8h_{n-1}\min\left(\mu_1, \max\left(0.1, \mu_2\right)\right), & 0 \leq \mu_1 < 0.95 \quad \text{(reject)} \\ 0.9h_{n-1}, & 0.95 \leq \mu_1 < 1.1 \quad \text{(accept)} \\ h_{n-1}, & 1.1 \leq \mu_1 < 2.0 \quad \text{(accept)} \\ 0.9h_{n-1}\min\left(\mu_1, \mu_2\right), & 2.0 \leq \mu_1 < \infty \quad \text{(accept)} \end{cases}$$

In terms of r/tol these rules can be viewed as in Figure 2.6. The algorithm does not use fixed values on the safety factors ρ and ε/tol discussed in Section 2.3. The rejection in case 1 corresponds to $\rho = (1/0.95)^3 \approx 1.2$, and the factors 0.8 and 0.9 in case 1 and case 4 corresponds to having $\varepsilon = 0.8^3\,tol \approx 0.5\,tol$ and $\varepsilon = 0.9^3\,tol \approx 0.7\,tol$, respectively. The deadband (case 3) reduces the number of stepsize changes in order to get efficiency in the equation solver.

In addition, the following logic is also used:

- the stepsize is never chosen larger than the simulation interval,
- the stepsize is never increased by more than a factor of 4 in one step, and
- no stepsize increase is allowed during 3 steps following a rejected step. □

Supervising the Equation Solver

In the equation solver one normally tries to use fixed point iteration as often as possible. Should the iteration fail to converge there is no new solution point and it is not possible to calculate the estimate of the local truncation error. In this case one could either switch to modified Newton iteration or recompute the step with a smaller stepsize and hope that fixed point iteration will converge. Since modified Newton iteration is computationally more expensive it seems natural to reduce the stepsize before changing iteration method. For efficiency the stepsize should, however, not be too small. What choice to make is a nontrivial decision, and different codes use different strategies [Nørsett and Thomsen, 1986b; Shampine, 1981; Hairer and Wanner, 1991].

The switch back from modified Newton to fixed point is done as fast as possible. Normally, the norm of the iteration matrix or the stiffness measure β (2.36) is calculated, and a switch is done when this measure signals that fixed point iteration would converge.

There are two natural reasons for the iteration in the equation solver to terminate: a sufficiently accurate solution has been found or the iteration does not converge. The equation solver must also include logic to handle a third case: poor convergence so that too many iterations will be needed to find a solution.

Most control algorithms handle the factorization of the iteration matrix and the evaluation of the Jacobian in a similar way [Nørsett and Thomsen, 1986b; Shampine, 1981; Hairer and Wanner, 1991]. Typically, the iteration matrix is factorized when the stepsize is changed, and the Jacobian is evaluated if the convergence is bad. The iteration matrix is, naturally, refactorized in case the Jacobian was evaluated.

EXAMPLE 2.12—Equation solver logic in SIMPLE(2)3

The logic handling the equation solver in SIMPLE(2)3 is rather straightforward [Nørsett and Thomsen, 1986a; Nørsett and Thomsen, 1986b; Nørsett and Thomsen, 1990]. At each iteration step the equation solver calculates an estimate α of the rate of convergence (2.35). Depending on this value and the last displacement Δ the following decisions are made:

1. $\Delta < \tau$, the solution is accepted and the iteration is terminated.
2. $\Delta > \tau$, the iteration error is too large, and if
   a. $\alpha < \alpha_{\max}$, the convergence is good and the iteration is continued,

b. $\alpha_{\max} < \alpha < 10$, the convergence is poor but the iteration is continued until either an acceptable solution is found (case 1) or another iteration with poor convergence is experienced, and

c. $\alpha > 10$, the iteration is terminated due to poor convergence.

A switch from fixed point iteration to modified Newton iteration is done as soon as the fixed point iteration fails to converge. A stiffness estimate β (2.36) is calculated to determine when it is safe to switch back to fixed point iteration. The stiffness measure is smoothed, using geometric averaging, before used, i. e. $\beta_{\text{smooth}} := (\beta + \beta_{\text{smooth}})/2$. After having switched to modified Newton iteration a change back is prevented during 10 steps. In addition, the following strategy is implemented for the iteration matrix and the Jacobian:

- The iteration matrix is factorized every time the Jacobian is evaluated or the stepsize is changed.
- The Jacobian is evaluated if the stepsize change was restricted by the convergence rate ($\mu_2 < \mu_1$, see Example 2.11), or if a step was rejected due to a convergence failure in the equation solver. □

2.6 Control Issues in Runge-Kutta Methods

A Runge-Kutta method is a complicated nonlinear system with several inputs and outputs (Figure 2.7). By choosing the inputs to this system we want to control its behavior. This requires a thorough understanding of the properties of the method.

A fundamental problem is whether the outputs of the integration method reflect the true situation, i. e. does the error estimate give an accurate picture of the true error? Part of the problem with poor measurements can be solved in the controller design; a noisy signal could for instance be averaged before being used. The overall system, i. e. the controller and the process, will, however, fail to accomplish its task if the output signals relate weakly to the global objectives. It is important to consider this problem when designing an integration method. Sophisticated controller design will never save the performance of an integration method supplying a poor error estimator. On the other hand, poor control can destroy the performance of any integration method.

The standard controller is based on asymptotic models and heuristics. These assumptions are sometimes not valid and as a result the

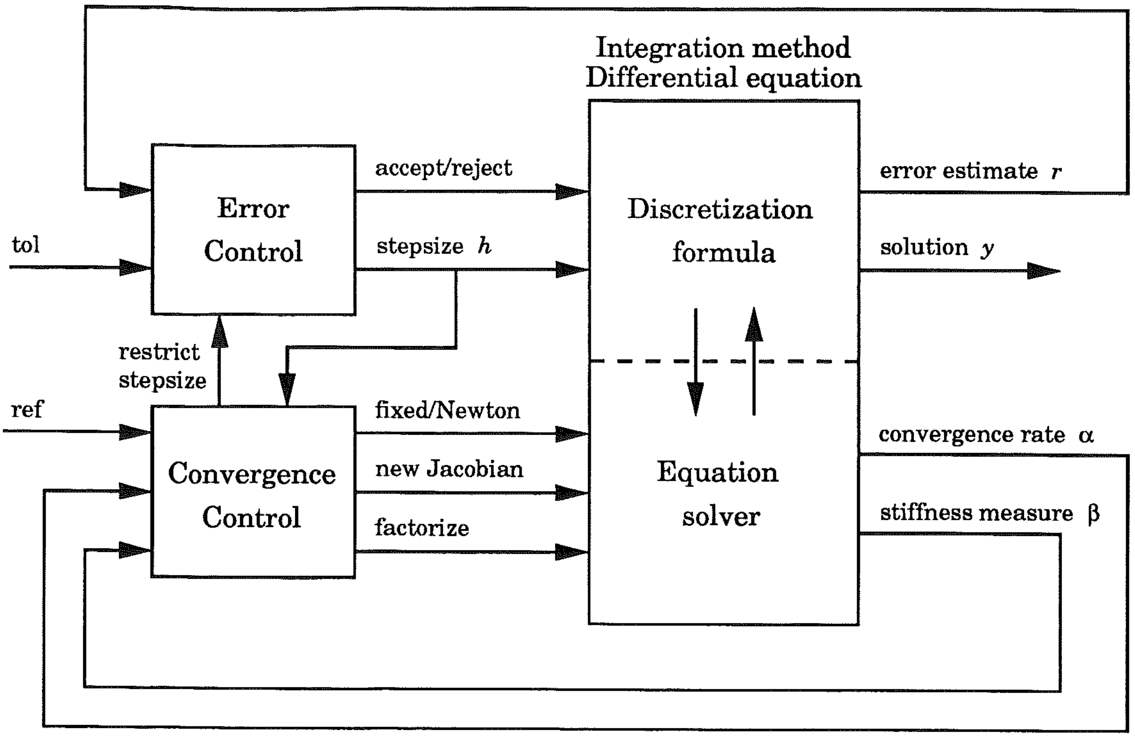


Figure 2.7 The integration method can be regarded as a multi-input-multi-output system. There are quite strong cross couplings and the dependence between the inputs and the outputs is complicated. The error controller keeps the error estimate r in correspondence with the prescribed accuracy, and the convergence controller supervise the equation solver so that it operates efficiently. The equation solver and the convergence control is only present in an implicit method.

closed loop system behaves poorly. In order to improve the control it is vital to understand the relationship between the different inputs and outputs. When and why do the standard asymptotic models fail? A large part of this thesis (Chapter 3 and part of Chapter 5) is devoted to answering that question. A part of the answer is to derive new models that better describe the behavior of the process. Such models are critical in the design of an improved controller, cf. Chapters 4 and 5.

The main objective of the controller is to keep the error at, or below, a user prescribed level. The error control is achieved by a feedback loop around the upper part of the process in Figure 2.7. The error estimate is kept at a tolerable level, by adjusting the stepsize and deciding whether a specific integration step should be accepted or rejected. The error is of course also affected by the equation solver in the lower part of the process, cf. Figure 2.7, but by keeping the iteration error sufficiently small the cross-coupling can be neglected. The iteration error is not kept down just to avoid cross-coupling. Its contribution to the error estimate is nonsmooth, and if allowed to be large it would seriously impair the error control.

The integration method should produce an acceptable solution as efficiently as possible. For an explicit Runge-Kutta method this relates closely to the error control. The error control should not try to obtain a solution of higher accuracy than asked for, since that would involve unnecessarily small integration steps. The same is of course also true for an implicit method, but the efficiency is here also related to the properties of the equation solver. By carefully choosing the inputs related to the equation solver the numerical solution can be produced with a minimum of computation. This is the mission of the convergence control.

Although the convergence control relates primarily to the inputs and outputs of the equation solver (cf. the lower part of the process in Figure 2.7), it also depends on the error control. The stepsize affects the convergence, and the error and convergence control algorithms can therefore not be designed separately.

3

The Stepsize-Error Relation

A Runge-Kutta method is a complicated nonlinear system with several inputs and outputs (cf. Figure 2.7). Successful control of the integration requires a good understanding of how the different inputs affect the outputs. There are quite strong cross couplings inside the integration method and each output variable is affected by, more or less, all the input variables. The strongest, and most important, dependence is the one between the stepsize h and the error estimate r.

The traditional asymptotic model (2.21) sometimes fails to correctly describe the stepsize-error relation, and consequently the error control degrades. In this chapter the asymptotic model is revisited and we investigate when it is insufficient. New models that better describe the stepsize-error relation are derived and verified. These models are used later (Chapters 4 and 5) to improve the error control algorithm.

3.1 The Limitations of the Asymptotic Model

Motivated by the asymptotic model (2.21), the stepsize should be calculated from

$$\varepsilon = \varphi_n h_n^k \tag{3.1}$$

to achieve $r_{n+1} = \varepsilon$, cf. Section 2.3. The quantity φ_n is unknown at the time t_n, and the controller cannot use (3.1) as is. Instead φ is assumed slowly varying and the previous value φ_{n-1}, which is computable from the most recent solution point ($\varphi_{n-1} = r_n/h_{n-1}^k$), is used to predict φ_n. Hence, the standard stepsize controller (2.22) is based on the stepsize-error model

$$r_{n+1} = \hat{\varphi}_n h_n^k, \qquad \hat{\varphi}_n = \varphi_{n-1}. \tag{3.2}$$

The key assumptions in this simple model are that higher order h-terms are negligible in the asymptotic relation (2.21), and that φ varies slowly. Both these assumptions are in some cases questionable, and as a consequence the model (3.2) is insufficient for successful error control. Some important reasons for the shortcomings are:

- The stepsize is nonzero during the integration, and it is not necessarily the $p+1$ term that dominates in (2.11). Consequently the error may behave as if k is larger than expected in (2.21).
- Some implicit methods lose convergence order when applied to stiff problems (order reduction [Prothero and Robinson, 1974]), causing k in (2.21) to be smaller than expected.
- φ may change considerably between two solution points, making φ_{n-1} a poor substitute for φ_n. The model (3.2) then gives an inaccurate description of the stepsize-error relation.
- In an integration method with bounded stability region the stepsize may grow large enough to move eigenvalues outside or close to the border of the stability region, cf. Definition 2.4. In this case the dynamics of the integration method changes significantly, and the resulting stepsize-error relation is very different from (2.21).
- In an implicit integration method the iteration error from the equation solver contributes to r, cf. (2.32). This contribution varies nonsmoothly from step to step, and φ will seem to vary irregularly.

As we will see in the sequel it is possible to devise new models that capture part of these properties. Before turning to the modeling, we will demonstrate the behavior through three examples.

EXAMPLE 3.1—Nonasymptotic behavior in the stepsize-error relation
The nonlinear problem [Hairer *et al.*, 1987, pp. 107, 236]

$$\begin{aligned} \dot{y}_1 &= y_2, & y_1(0) &= 2 \\ \dot{y}_2 &= \sigma\left(1 - y_1^2\right) y_2 - y_1, & y_2(0) &= 0 \end{aligned} \tag{3.3}$$

is referred to as the van der Pol oscillator and is a common test problem for numerical integration methods. The problem has a periodic solution (upper diagram of Figure 3.1) with quite different properties along the trajectory. The parameter σ is used to vary the properties of the problem. The choice $\sigma = 5$ makes the problem mildly stiff during the smooth part of the solution.

The problem (3.3) was solved using the explicit RKF(1)2 method (cf. Example 2.8 and Appendix A). The method was run as XEPS, with the error measured using the mixed absolute-relative 2-norm in (2.14). The tolerance was set to $\varepsilon = 0.01$ and the norm used $\eta = 0.01$ and $\tilde{y} = |y|$.

RKF(1)2 is a simple integration method, and it is fairly straightforward to calculate the error estimate explicitly (cf. (2.13) in Example 2.8)

$$\hat{e}_{n+1} = y_{n+1} - \hat{y}_{n+1} = -\frac{h_n^2}{2} f_y f - \frac{h_n^3}{4} f_{yy} f f. \tag{3.4}$$

With the help of the symbolic manipulation tool Maple [Char *et al.*, 1988] the differentials $f_y f$ and $f_{yy} f f$ were calculated for (3.3). At each step n we evaluate the differentials and use these values to calculate the stepsize h_n required to make $r_{n+1} = \varepsilon$. The procedure involves a lot of computation since r_{n+1} depends nonlinearly on h_n and y_n. The result is a computationally very expensive error controller that calculates a stepsize sequence that achieves "perfect" control, i. e. $r \equiv \varepsilon$. There is, however, no guarantee that this also makes the global error behave well. The numerical solution points resulting from the integration procedure are plotted as crosses in the upper diagram of Figure 3.1. The middle diagram depicts how the stepsize changes during the simulation interval.

The error estimate (3.4) consists of one h^2 and one h^3 component. The lower diagram in Figure 3.1 shows the norm of each of the two components. The h^2 term normally dominates, but the h^3 term cannot be neglected at the turning point and in the middle of the smooth phase. In 20 % of the integration steps the h^3 term is larger than one tenth of the h^2 term. The full line in the lower diagram of Figure 3.1 shows the norm of the local truncation error, i. e. $\|e\|$. Since RKF(1)2 uses local extrapolation the error estimator normally overestimates the local truncation error.

It is interesting to note that at times both the h^2 and the h^3 error terms are much larger than ε. In these cases the two error vectors have different directions and partly cancel. The norm of their combined

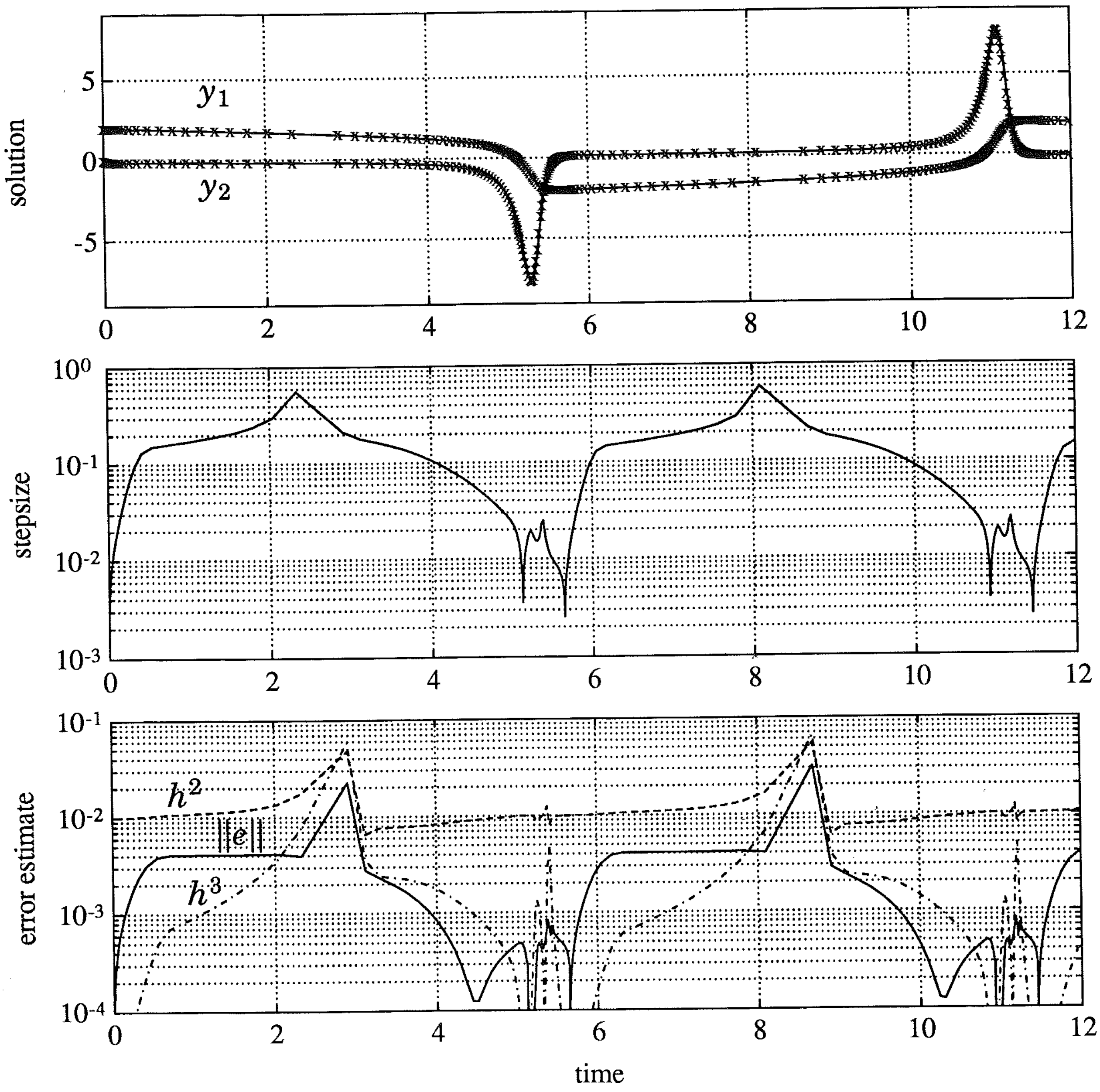


Figure 3.1 The upper diagram shows the solution to the van der Pol oscillator ($\sigma = 5$) in Example 3.1. The numerical solution points are shown as crosses on the solution curve. The stepsize, depicted in the middle diagram, is chosen so that the error estimate r is equal to the tolerance $\varepsilon = 10^{-2}$ at each numerical solution point. The error estimate includes one h^2 and one h^3 term. The lower diagram shows the norm of each of these components. The h^2 term normally dominates but the h^3 component is often large enough to cause a stepsize-error relation different from (2.21). The lower diagram also depicts the norm of the local truncation error $\|e\|$ (calculated with a different method of high accuracy). RKF(1)2 uses local extrapolation and as a consequence the embedded error estimator often overestimates the local truncation error.

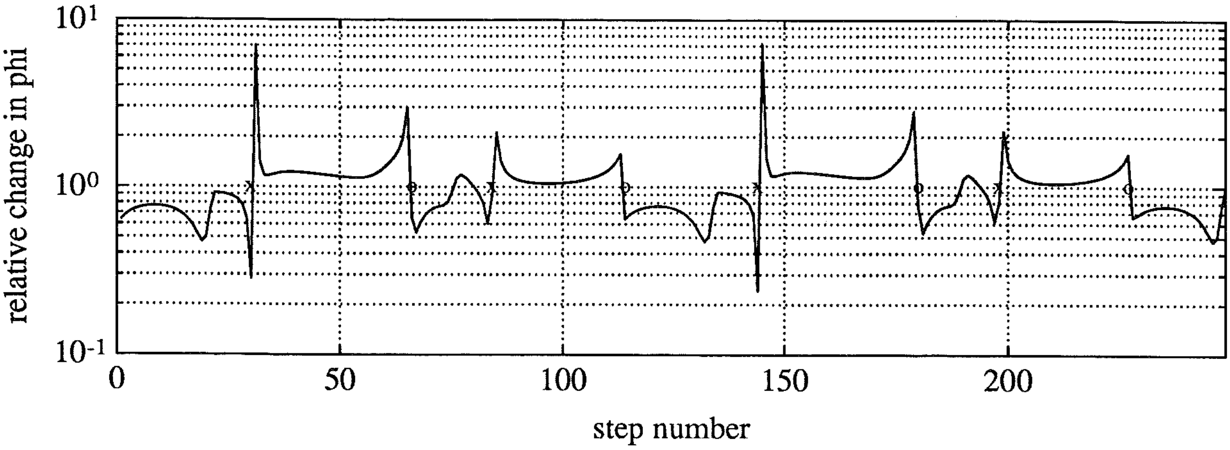


Figure 3.2 The diagram depicts the ratio φ_{n+1}/φ_n during the integration of the van der Pol oscillator in Example 3.1. As can be seen the variations in φ may be rather large. Changes labeled with 'x' corresponds to rapid changes in the elementary differentials, while changes labeled 'o' are provoked by the absolute-relative norm.

contribution is still exactly equal to ε.

In this example the error estimate is kept exactly equal to the tolerance, i. e. $\varepsilon = \varphi_n h_n^k$, and consequently $\varphi_{n+1} h_{n+1}^k = \varphi_n h_n^k$. From this expression it is straightforward to calculate the variation in φ, i. e.

$$\frac{\varphi_{n+1}}{\varphi_n} = \left(\frac{h_n}{h_{n+1}}\right)^2.$$

This quantity is plotted in Figure 3.2. The variations are quite large and any stepsize selection algorithm based on the assumption that φ is approximately constant will at times behave poorly. An algorithm that has $\varepsilon \ll tol$ would in principle manage large φ variations without rejected steps, but the integration would be very inefficient.

Large changes in φ stem from two different sources: rapid changes in the elementary differentials and "discontinuities" in the norm. The jumps at $t \approx 2.0$, 5.4, 7.8 and 11.2 in φ (marked by 'x' in Figure 3.2) belong to the former category and are caused by rapid direction changes in $f_y f$. The other jumps can be attributed to the change from relative to absolute error norm that takes place when a solution component gets close to 0. This change introduces a discontinuity that affects φ (marked by 'o' in Figure 3.2). □

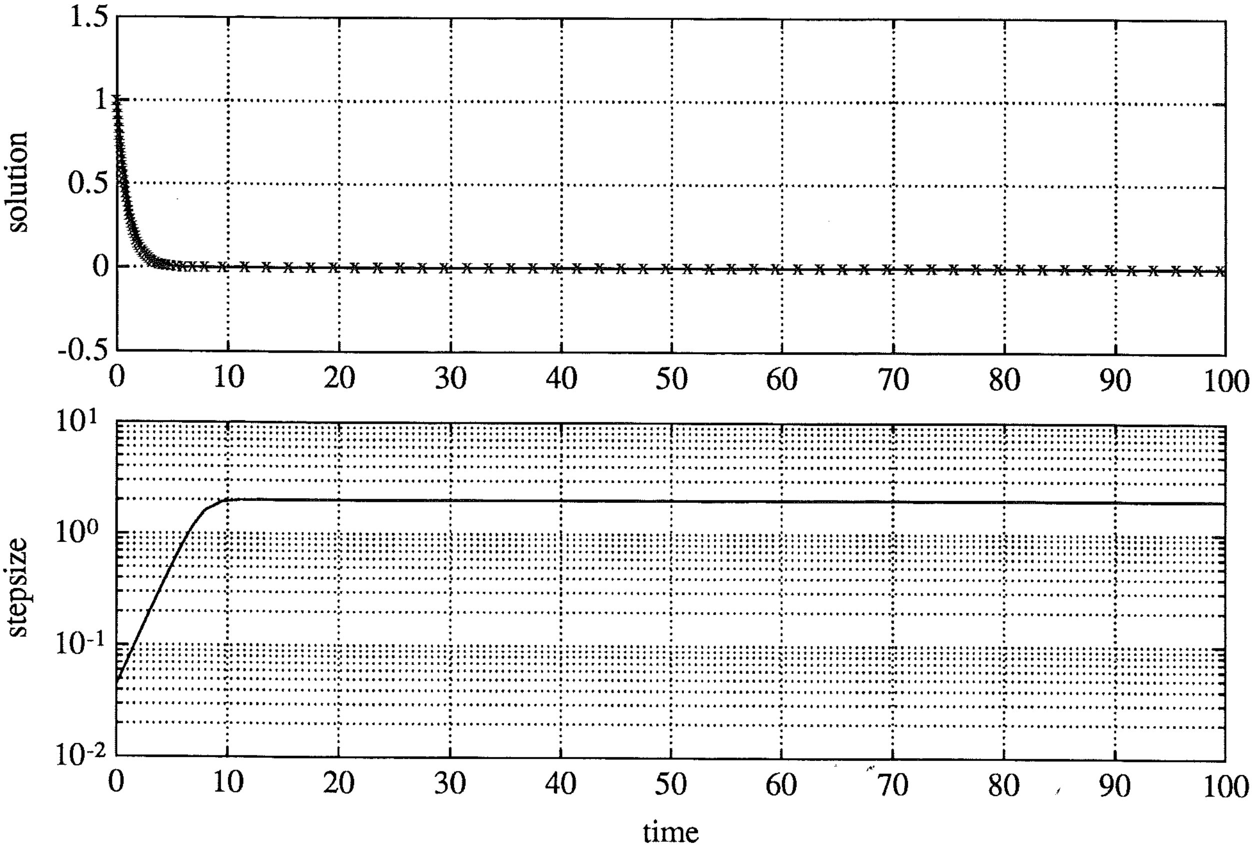


Figure 3.3 The upper part of the figure depicts the solution to the problem in Example 3.2. Each numerical solution point is marked with a cross. The lower part of the figure shows the stepsize sequence used when solving the problem. The stepsize sequence was chosen so that $r_n = \varepsilon$ at all solution points.

EXAMPLE 3.2—Stepsize restricted by numerical stability

The problem

$$\dot{y} = \lambda y, \qquad \lambda = -1, \quad y(0) = 1 \tag{3.5}$$

was solved with the explicit method RKF(1)2. The stepsize sequence was calculated so that the error estimate is exactly equal to $\varepsilon = 10^{-3}$ at each solution point. A pure absolute 2-norm, i.e. $r = \sqrt{\sum_i \hat{e}_i^2} = |\hat{e}|$, was used to measure the error.

As the solution decays towards its stationary value the stepsize increases to keep $r_n = \varepsilon$. The discretization becomes unstable if $h_n\lambda < -2$ (cf. Example 2.9), and as seen in Figure 3.3 the stepsize levels out at $h_n = 2$.

To investigate the stepsize-error dynamics we use the calculated stepsize sequence to perform a step response test. The nominal stepsize was increased by 10 % during two intervals: one during the transient phase of the solution and one when the stepsize has reached the stability limit. When increasing the stepsize by a factor of θ one would expect

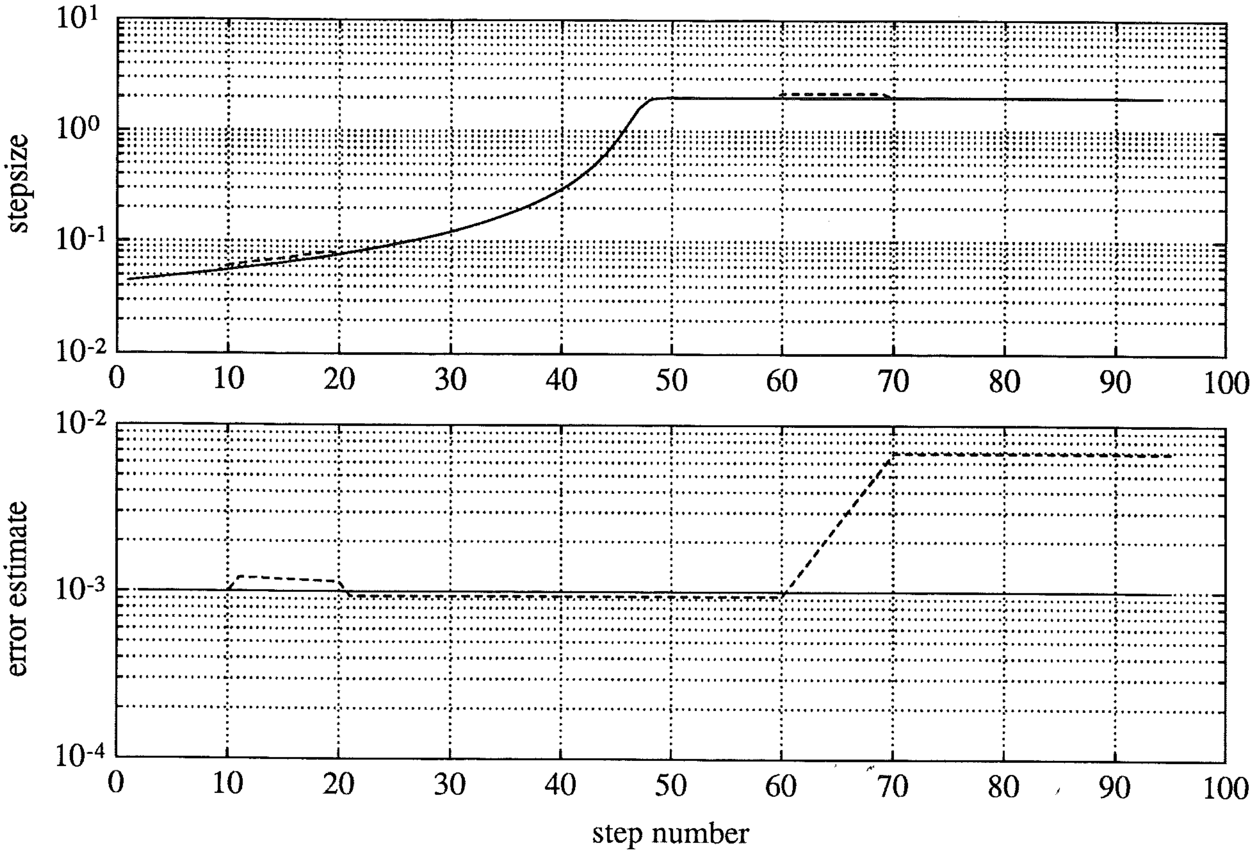


Figure 3.4 The problem in Example 3.2 was solved with two stepsize sequences. The first one (upper plot, full line) makes $r_n = \varepsilon$ at each step. The second sequence (upper plot, dashed line) is a perturbed version of the first sequence, with the stepsize increased by 10 % in the intervals [10, 19] and [60, 69]. As can be seen in the lower diagram the response to the perturbation is quite different in the two intervals. The behavior in the first interval is described by (2.11) while the second case is a consequence of the integration method operating on its stability boundary. Note that, in contrast to Figure 3.3, all variables are plotted as a function of integration step number.

the error estimate to grow like θ^k. This is indeed the case for the first interval (see Figure 3.4). In the second interval the behavior is quite different. The error estimate seems to accumulate previous values. The reason is that the integration method operates on the edge of instability, resulting in quite different error dynamics.

This example demonstrates something that will happen whenever an explicit integration method is used to solve a differential equation where a fast mode decays towards a constant or slowly varying solution. The choice of norm is not critical and we would see the same effect for a nonzero stationary value and a relative norm. □

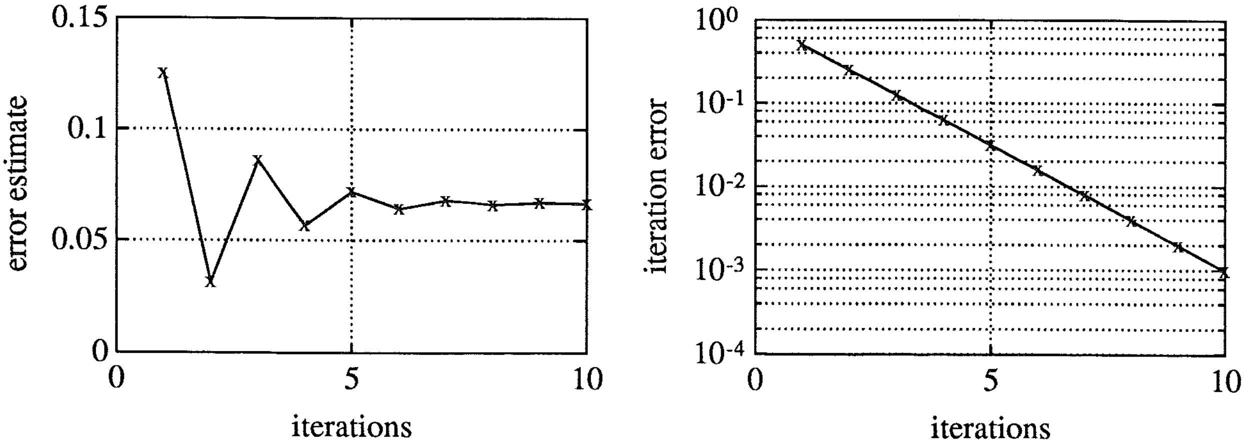


Figure 3.5 In Example 3.3 the initial integration step is calculated using different number of iterations in the equation solver. The figure depicts the iteration error and its effect on the error estimate as function of the number of iterations.

Example 3.3—The iteration error effect on the stepsize-error relation
Consider again the linear test equation (3.5) in Example 3.2. We use the implicit midpoint rule for the integration and estimate the error as the difference between the implicit midpoint step and an implicit Euler step (cf. Appendix A). Fixed point iteration was used to solve for the stage values Y, cf. (2.24a).

The first integration step ($h = 0.5$) was taken using a different number of iterations in the equation solver. Figure 3.5 depicts how the error contribution from the iteration affects the error estimate. As the number of iterations grow the iteration error gets smaller, and the error estimate levels out at approximately 0.067.

When the iteration error is allowed to be large, i. e. τ large and therefore few iterations, the error estimate is strongly affected when the number of iterations change. This adds a nonsmooth error component, depending weakly on the stepsize, to the error estimate. □

It is difficult to capture all the effects demonstrated in Examples 3.1–3.3 in one model. Most of what is seen in Example 3.1 can be handled by extending the standard model (2.21) with a model for the variations in φ. The case where the stepsize gets restricted by stability (Example 3.2) is, however, quite different, and a separate model has to be used for that case.

The contribution from the iteration error (Example 3.3) depends on decisions made in the equation solver, which may vary considerably between successive steps. The coupling between the iteration error and the stepsize is quite weak, and we will refrain from trying to model the dependence. Instead we will consider the iteration error effect on r as a disturbance that can be kept small by controlling the equation solver appropriately.

3.2 Different Operating Regions

Based on the properties of the stepsize-error relation an integration method can be regarded as operating in one of three different regions:

- the asymptotic region,
- the nonasymptotic region, and
- the stepsize restricted by numerical stability.

The dependence between stepsize and error is quite different in these regions, and one single model is insufficient.

Again we return to the linear test equation (2.17). Figure 3.6 depicts the functions $P(z)$ and $|P(z) - e^z|$ on the real axis for two integration methods: RKF(1)2 and implicit Euler (cf. Example 2.10). When $z = h\lambda$ is close to 0 the function $P(z)$ matches the exponential function very well, and the numerical solution will be close to the true solution. In this case the integration method operates in its asymptotic region, and the error is governed by the h^k asymptotics, cf. (2.21).

For $\lambda < 0$ the solution to the test equation decays towards zero, and eventually it will approach values that are below the required accuracy of the numerical integration (For simplicity consider an absolute error norm. The same arguments will be true also for a relative norm with $\eta \neq 0$ or a relative norm around a nonzero stationary value). As the solution tends to zero the error drops and the error controller increases the stepsize in response. The relative accuracy of the discretization gets worse as z decreases along the negative real axis (Figure 3.6), but due to the small value of the solution the local truncation error will still be within the accuracy requirements. As the stepsize increases the integration method moves to the nonasymptotic operating region, and the error no longer obeys the h^k asymptotics.

If the stepsize continues to increase explicit methods become unstable, i. e. $|P(z)| > 1$ (cf. RKF(1)2 in Figure 3.6). The instability will cause the solution to grow and soon it will be large enough to cause nonnegligible local truncation errors and the stepsize has to be decreased. In this case the stepsize is said to be limited by numerical stability. The situation is different for a well-designed implicit method. Here the stepsize

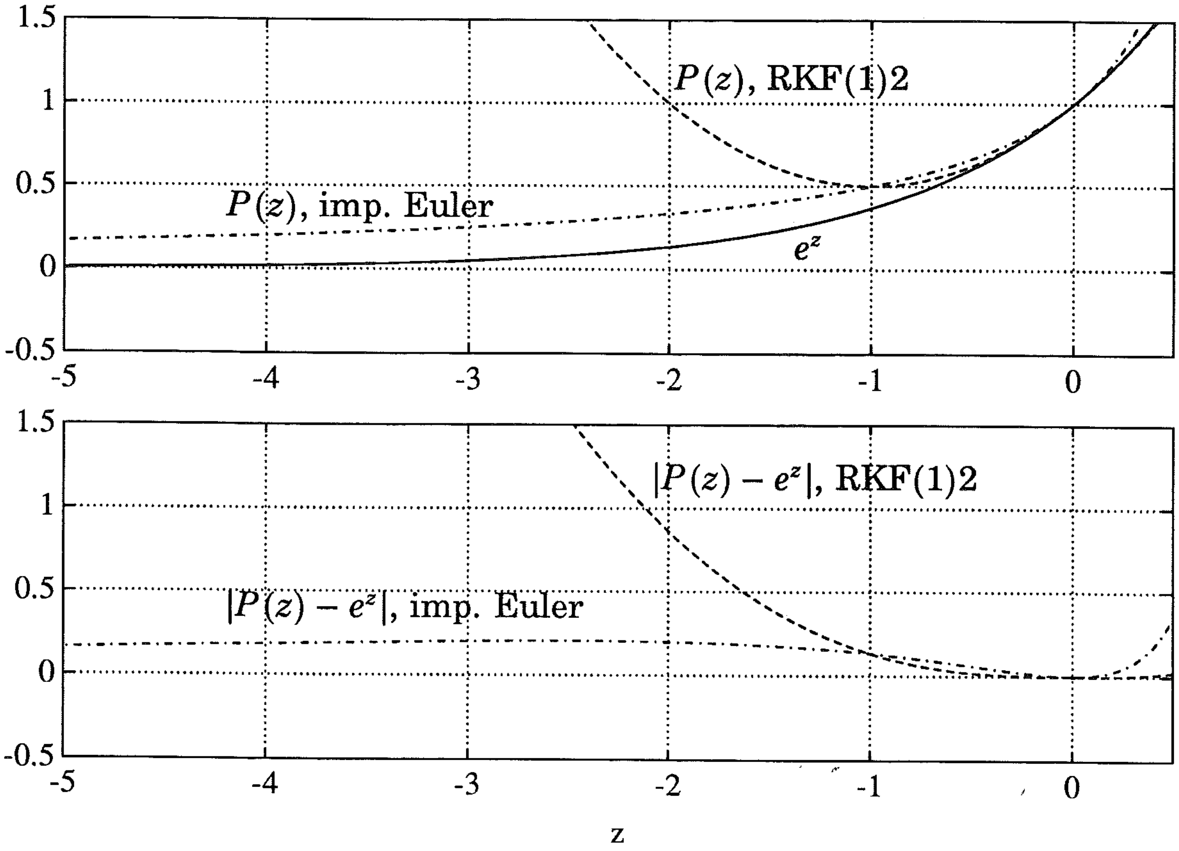


Figure 3.6 The functions $P(z)$ and $|P(z) - e^z|$ for two different integration methods. The functions are evaluated along the real axis. When $z \approx 0$ the exponential function is closely matched by $P(z)$. As $z \to -\infty$ the matching deteriorates and at $z \approx -2.1$ the explicit method RKF(1)2 gets unstable.

may be increased without $|P(z)|$ getting larger than 1 (cf. implicit Euler in Figure 3.6). Thus, as the solution decays the error controller increases the stepsize, and the integration method moves farther and farther out in the nonasymptotic region, but without becoming unstable.

Although Figure 3.6 depicts the situation for the linear test equation, it is still of interest when trying to solve nonlinear differential equations where the Jacobian has a spectrum with large differences in eigenvalue magnitude. At startup the error controller has to use a small stepsize to correctly resolve the transients corresponding to the eigenvalues with large negative real part. As these modes decay their contribution to the local truncation error decreases, and it is no longer necessary to keep them in the asymptotic region. The stepsize is, consequently, increased and it will be chosen in relation to the slower modes. The situation is identical to the one for the linear test equation, although we now have fast transients converging towards a slowly varying stationary solution instead of a single component approaching zero.

Control Authority

In the asymptotic region the error controller can easily affect the local truncation error by changing the stepsize. For an explicit method the situation is similar in the nonasymptotic region. The stepsize-error relation differs from h^k, but the error can still be affected with moderate stepsize changes. Implicit methods are, however, a completely different case. For large negative values on z the value of $|P(z) - e^z|$ is almost constant, and very large stepsize changes are needed to affect the error. This is normally not called for since, in general, the error contribution from the nonasymptotic modes is well below the accuracy requirement.

When a mode has decayed and been moved out to the nonasymptotic region the error controller has lost control authority over it. Its error contribution cannot be controlled without very large changes in the control signal. The behavior of $P(z)$ as $z \to -\infty$ is therefore of great importance. Although, $|P(z)| < 1$ is enough for stability this requirement is not sufficient for good performance. The value of $P(z)$ also governs how much of a specific mode is propagated to the next step, and $P(z)$ should, as the exponential, tend to zero as $z \to -\infty$ (a property shared by all L-stability methods, Definition 2.6). If this is not the case, errors in the fast modes will propagate from step to step. In particular, errors due to an inaccurate iterative solution may excite these modes. If $|P(z)|$ gets close to 1 the errors may accumulate and eventually become large enough to cause a step rejection [Arévalo, 1992]. It then takes a drastic stepsize decrease in order to continue the integration. The fast mode must be returned to the asymptotic region, i. e. where $E(z)$ is small enough and control authority is regained. This behavior has been observed in practical codes, cf. Figures 5.4–5.6 and [Hairer and Wanner, 1991, p. 122].

Having $P(z) \to 0$ as $z \to -\infty$ is important since it prevents from accumulating an error that is difficult to control. It is, however, equally important for $E(z) \to 0$ as $z \to -\infty$. Otherwise the error estimator may observe fictitious errors in a region where the error controller has almost no control authority. The error estimator should observe only the “real” errors, so that the control effort can be concentrated on the error terms that matter.

3.3 An Improved Asymptotic Model

As was demonstrated in Example 3.1 both the φ model

$$\hat{\varphi}_n = \varphi_{n-1} \tag{3.6}$$

as well as the fixed exponent k are questionable in the model (3.2). During the integration we have, however, available only recent errors r and recent stepsizes h. From these measurements it is impossible to separate variations in φ from variations in k. If stepsizes that are far outside the asymptotic error region are disregarded (we will treat them in Section 3.4), variations in k are less likely than variations in φ. We will therefore continue to regard k as fixed. Should k still vary, this can be anticipated as changes in φ provided the variation is moderate.

Comparing Different Models for φ

There is a lot of structure in the variations of φ along the solution of the differential equation (cf. Figure 3.2). If some of this structure could be captured in a model it would be possible to improve the prediction $\hat{\varphi}_n$. The model should preferably be simple. Sometimes there are very abrupt changes in φ, and at such situations it would be difficult to reinitialize the states in a complicated model.

Using a linear approximation for φ we obtain

$$\hat{\varphi}_n = \varphi_{n-1} + \dot{\varphi}_{n-1} h_n. \tag{3.7}$$

An inherent problem of this model is $\dot{\varphi}$. This quantity is difficult to calculate or estimate. To obtain an asymptotically correct estimate requires extra stages in the integration method [Söderlind, 1991].

A simplification of (3.7) would be to have

$$\hat{\varphi}_n = \varphi_{n-1} + \nabla \varphi_{n-1} \tag{3.8}$$

where ∇ is the backward difference operator, i.e. $\nabla \varphi_{n-1} = \varphi_{n-1} - \varphi_{n-2}$, and $\nabla \varphi_{n-1}$ is a measure of how much φ changed in the last step. If the stepsize changes considerably between consecutive steps it seems natural to normalize $\nabla \varphi$, i.e.

$$\hat{\varphi}_n = \varphi_{n-1} + \frac{h_{n-1}}{h_{n-2}} \nabla \varphi_{n-1}. \tag{3.9}$$

The factor φ (and hence also h) may vary several orders of magnitude, and it might be beneficial to use logarithmic variables, i.e. to consider multiplicative changes instead of additive. One obtains

$$\log \hat{\varphi}_n = \log \varphi_{n-1} + \nabla \log \varphi_{n-1}, \tag{3.10}$$

where $\nabla \log \varphi_{n-1} = \log \varphi_{n-1} - \log \varphi_{n-2}$ is a measure of how much $\log \varphi$ changed in the last step. To normalize $\nabla \log \varphi$ with the stepsize, one could introduce

$$\log \hat{\varphi}_n = \log \varphi_{n-1} + \frac{h_{n-1}}{h_{n-2}} \nabla \log \varphi_{n-1}. \tag{3.11}$$

We compare the five models (3.6), (3.8)–(3.11) using a simple example.

Example 3.4—Different models for φ
The five models (3.6), (3.8)–(3.11) can be tested on the φ-sequence calculated for the van der Pol simulation in Example 3.1. Histograms over the onestep prediction errors are depicted in Figure 3.7. The width of a bin corresponds to a 2 % error in $\log_{10} \hat{\varphi}_n$, and the height of a bin shows the number of steps belonging to the bin. In total there were 250 integration steps.

A prediction error in $\hat{\varphi}_n$ will result in an error r_{n+1} that is different from ε. To keep r_{n+1} within ±20 % from ε corresponds to a prediction error $|\log_{10} \varphi_n - \log_{10} \hat{\varphi}_n|$ less than 0.1.

From Figure 3.7 it is clear that the logarithmic models perform best, the linear ones slightly worse, while the standard model behaves quite poorly. The models that normalize with the stepsize are slightly less successful than the versions that do not use normalization. In particular, they tend to underestimate φ. The stepsize varies in relation to φ, and these variations act as a time scaling of the φ-changes. A model which normalizes with the stepsize removes some of the positive effects of this scaling.

Occasionally, all models result in very large prediction errors. This corresponds to the situations where there are abrupt changes in φ (cf. Figure 3.2). The models presented here are based on the smoothness of φ and abrupt changes cannot be captured. When φ_n is underestimated the resulting integration error will be large, and the step has to be rejected. An overestimation, on the other hand, results in an integration step that is unnecessarily short. □

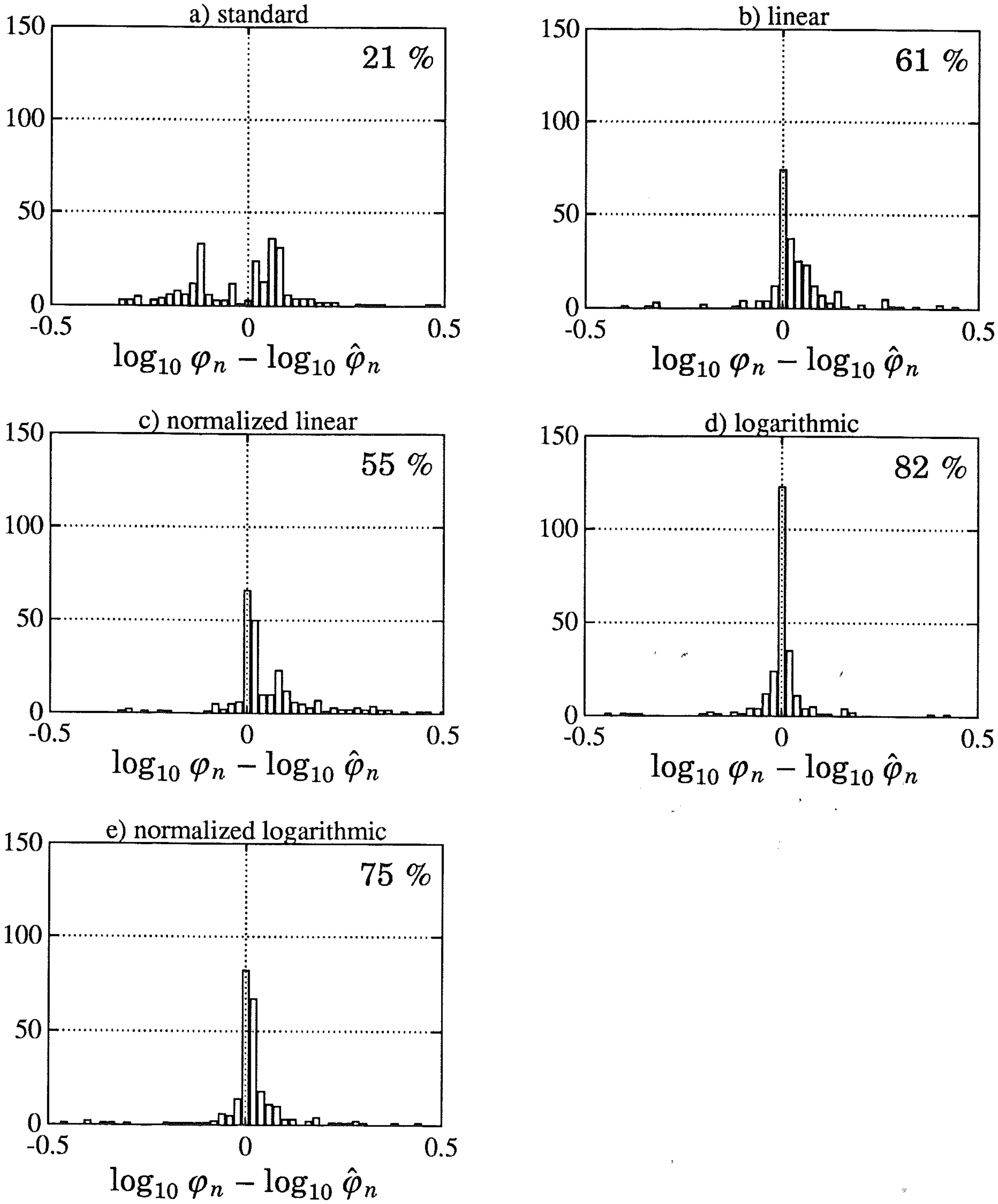


Figure 3.7 Histograms for the prediction error of $\log \varphi$ for five different models for φ. The models were used on the φ sequence from the simulation of the van der Pol oscillator in Example 3.1. The height of a bin depicts how many steps that resulted in a specific prediction error. The five models are **a)** standard model (3.6), **b)** linear model (3.8), **c)** linear model with stepsize normalization (3.9), **d)** logarithmic model (3.10), and **e)** logarithmic model with stepsize normalization (3.11). The figure in the upper right corner of each plot tells how many percent of the predictions that were within ±0.05 of the correct value. This corresponds approximately to a 10 % error in r_{n+1}. As can be seen the logarithmic model (3.10) is the most successful in predicting φ_n.

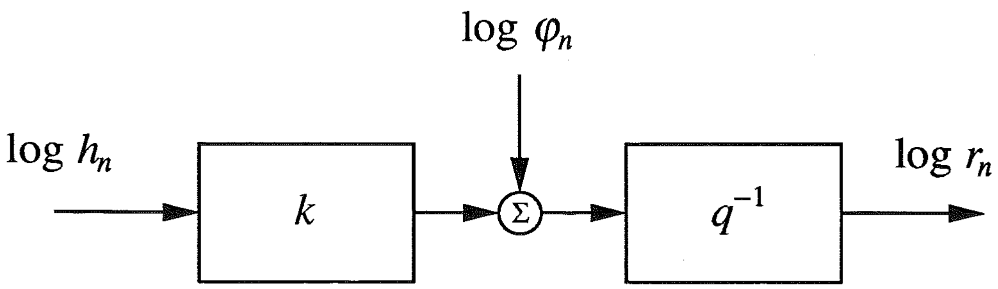


Figure 3.8 For small stepsizes the integration method can be viewed as a linear process with $\log h_n$ as input and $\log r_n$ as output. The external disturbance $\log \varphi_n$ depends on both the differential equation and the integration method.

The Asymptotic Model

We conclude the section on the asymptotic model by rewriting it in a slightly different way. By regarding $\log h_n$ as the input signal and $\log r_n$ as the output of the process (see Figure 3.8), the asymptotic model (2.21) is turned into an affine relation. Using the forward shift operator q we obtain

$$\log r_n = G_{p1}(q) \log h_n + q^{-1} \log \varphi_n, \qquad G_{p1}(q) = kq^{-1}. \tag{3.12}$$

Thus, the process is just a constant gain k, depending on the order of the error estimator, and a disturbance $\log \varphi_n$ depending on the properties of the differential equation and its solution, cf. (2.11). The delay q^{-1} is a consequence of the indexing conventions, i.e. the stepsize h_n is used to advance y_n to y_{n+1} giving r_{n+1} as output.

In the asymptotic region the model (3.12) describes the stepsize error relation very well. The disturbance φ_n should, however, be predicted using model (3.10) rather than (3.2). The change in the disturbance model may seem insignificant, but, as will be seen in Chapters 4 and 5, it implies a different type of error controller.

3.4 Outside the Asymptotic Region

If a mode in the nonasymptotic region dominates the error estimate we may experience a stepsize-error relation with a very different k than expected. Examples of when this may happen are:

- The character of the differential equation may change so that a fast mode at stationarity gets excited and induces a new transient.

- It may be difficult to choose the initial integration step putting all fast modes within the asymptotic region, without the stepsize being unnecessarily small.
- Some implicit integration methods lose convergence order when being applied to a stiff differential equation [Prothero and Robinson, 1974]. The phenomenon is referred to as order reduction, and it results in a k value in (2.21) that is lower than expected.

From measurements of r and h it is impossible to distinguish variations in k from variations in φ. This is a serious problem. A varying φ is just an external signal that the controller will be able to handle, but a varying k changes the dynamics of the feedback loop, cf. Figure 3.8, which may call for a redesign of the controller so that sufficient robustness is obtained.

Due to the difficulty with a varying k the process should be designed making this situation less likely. It is possible to design *stiffly accurate* integration methods where the effect from order reduction is minimized [Cash, 1979; Kværnø, 1988; Hairer *et al.*, 1989; Hairer and Wanner, 1991], and from the control point of view it is recommended to use these methods whenever trying to solve stiff differential or differential-algebraic equations.

Even when using stiffly accurate integration methods there may be error contributions from modes in the nonasymptotic region. This may be experienced as a value of k different from the one expected [Hairer and Wanner, 1991, p. 434]. If the experienced variation is moderate it may be modeled as changes in φ. An error controller that is based on a stepsize-error model where φ is not assumed constant, e. g. (3.10) and (3.12), will be able to handle also this situation.

Initial Stepsize

To demonstrate some of the properties of the stepsize-error relation outside the asymptotic region, we will consider the choice of initial stepsize. When starting the integration of a new problem it can be quite difficult to choose an appropriate stepsize. Some codes leave this choice to the user, while other implement an automatic strategy [Watts, 1982; Hairer *et al.*, 1987, pp. 182–183]. Independently of how it was obtained, there is always a risk that the initial stepsize is outside the asymptotic region of the stepsize-error relation. In such a case k will differ from its expected value. The behavior differs between explicit and implicit methods. In an

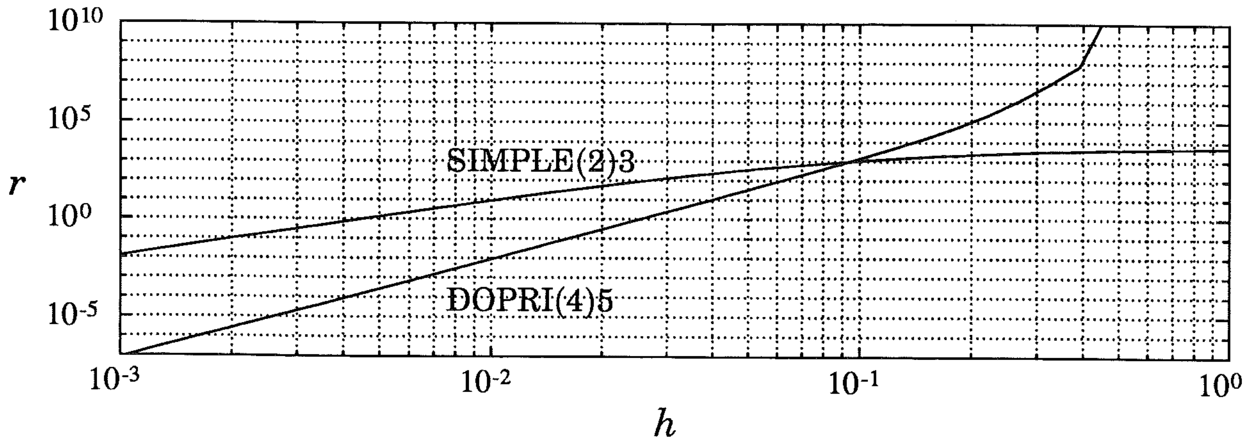


Figure 3.9 The error estimate in the initial step as function of stepsize. For the van der Pol equation with $\sigma = 5$ the two methods, DOPRI(4)5 and SIMPLE(2)3, have quite different behavior outside the asymptotic region $h < 0.1$.

explicit method the error estimate depends almost polynomially on the stepsize. When the stepsize increases, higher order terms will dominate and the error grows faster than h^k. In implicit methods, on the other hand, the error estimate is a rational function of the stepsize. The order of the numerator polynomial is less than or equal to the order of the denominator polynomial, and for large stepsizes the error estimate will be almost constant or tend to zero.

EXAMPLE 3.5—Error dependence on stepsize at initial step

Consider DOPRI(4)5, $k = 5$, and SIMPLE(2)3, $k = 3$, applied to the van der Pol oscillator in Example 3.1. Figure 3.9 shows how the error estimate in the initial step depends on the stepsize. A mixed absolute-relative 2-norm (2.14) with $\eta = 10^{-6}$ and $\tilde{y} = |y|$ was used. The asymptotic behavior is clearly visible for stepsizes $h < 0.1$. For large stepsizes the error behaves quite differently. The error in the implicit method stays fairly constant while the error in the explicit method grows strongly with increasing stepsize. □

A Simplistic Model

As was pointed out above it is not possible to separate changes in k and φ during normal integration. One exception is at a rejected step, where we get two measurements from the same point. If rejected steps are repeated, it therefore seems reasonable to use the information to estimate the current k-value, i.e.

$$\hat{k} = \frac{\log r_{n+1} - \log r_n}{\log h_n - \log h_{n-1}} \tag{3.13}$$

where (r_{n+1}, h_n) and (r_n, h_{n-1}) are errors and stepsizes from two consecutive rejected steps. The obtained value can be used when calculating a new stepsize. In practice $\hat{k}$ should be compared with the expected k, and too large deviations should not be trusted.

In some situations the benefit from using an estimated k value may be very large. Order reduction may sometimes lead to a long sequence of rejected steps [Hairer and Wanner, 1991, p. 122]. In Chapter 5 we will use a controller that estimates k, and through that greatly reduces the number of rejections in the described situation.

3.5 Stepsize Limited by Stability

In Example 3.2 the decay of the transient made it possible to increase the stepsize and still fulfill the accuracy requirement. This is a common situation, i.e. it is in general possible to increase the stepsize when initial transient have died out and the forcing function in the differential equation varies slowly compared to the equation dynamics. An integration method with bounded stability region (cf. Example 2.9) cannot increase the stepsize above a critical value h_s. This value puts some eigenvalue λ of the (linearized) problem on $\partial\mathcal{S}$, the boundary of the stability region $\mathcal{S} = \{h\lambda : |P(h\lambda)| \leq 1\}$ of the integration method. Increasing the stepsize beyond h_s makes the nonlinear difference equation system (2.3) unstable. This will cause the error estimate r to grow and the error controller reduces the stepsize to h_s. The stepsize is said to be limited by numerical stability.

When h_n approaches h_s the dynamics of the stepsize-error relation changes considerably. An important question is whether the error control loop is stable in spite of this change in process dynamics. There are two types of stability discussed here: the possibly unstable integration method which causes a change in process dynamics, and the stability of the error control loop. The error control loop may very well be stable although the integration method approaches instability.

The standard stepsize selection rule (2.22) is often unable to handle the change in process dynamics. The instability in the error control loop leads to an irregular stepsize sequence, where the stepsize oscillation around h_s causes many rejected steps. Hall and Higham have analyzed the stability of the equilibrium state h_s assuming the standard stepsize selection rule (2.22) [Hall, 1985; Hall, 1986; Hall and Higham, 1988; Higham and Hall, 1989]. Their analysis leads to a stability test for the error control loop based on the coefficients ($\mathcal{A}$, b, and c) of the integration method. Using this stability test they then proceed to design methods where the equilibrium h_s is stable [Higham and Hall, 1987].

To be able to improve the properties of the error control loop it is essential to model the behavior of the process when $h_n\lambda$ approaches $\partial\mathcal{S}$. We will derive such a model, using an analysis similar to that of Hall and Higham. Our approach is, however, different, in that we want to model the stepsize-error relation separately *without* making assumptions on a specific error control strategy. Having obtained that model we will later (cf. Chapter 4) design a controller that gives the complete system desired properties. This approach gives, in general, a better damped error control loop than what is achieved with the methodology of Higham and Hall.

A Linearized Test Problem

Consider the problem

$$\dot{x} = A\,(x - w(t)) + \dot{w}(t) \qquad x(0) = x_0 \in \mathbb{R}^N, \quad t \geq 0, \tag{3.14}$$

which is a linearization of the general problem (2.1) about a slowly varying solution $x(t) = w(t)$. The eigenvalues of A have negative real parts, and describe how $x(t)$ approaches $w(t)$ from $x(0) \neq w(0)$.

Introduce $y_n := x_n - w(t_n)$. A Runge-Kutta method applied to (3.14) may then be written (a generalization of (2.7) in [Prothero and Robinson, 1974])

$$y_{n+1} = P(h_nA)y_n + \omega_n, \qquad \hat{e}_{n+1} = E(h_nA)y_n + \varsigma_n, \tag{3.15}$$

where $P(h_nA)$ and $E(h_nA)$ are defined in (2.20) and

$$\begin{aligned}\omega_n &= w(t_n) - w(t_{n+1}) + \left(\hat{b}^T \otimes I_N\right)\left(I_s \otimes I_N - \mathcal{A} \otimes h_nA\right)^{-1} W_n\\ \varsigma_n &= \left((b - \hat{b})^T \otimes I_N\right)\left(I_s \otimes I_N - \mathcal{A} \otimes h_nA\right)^{-1} W_n\end{aligned}$$

and

$$W_n = \begin{pmatrix} h_n A\big(w(t_n) - w(t_n + c_1 h_n)\big) + \dot{w}(t_n + c_1 h_n) \\ \vdots \\ h_n A\big(w(t_n) - w(t_n + c_s h_n)\big) + \dot{w}(t_n + c_s h_n) \end{pmatrix}.$$

The forcing functions ω_n and ς_n in (3.15) depend on $h_n A$ and the change in $w(t)$ during the integration step. The structure of ς_n is similar to the one of $E(h_n A)$, and the $O(h^{p+1})$ term is the first nonzero term in a Taylor expansion about $h_n = 0$. This is to be expected in view of (2.11).

From (3.15) it is clear that φ_n in (2.21) depends both on y_n and ς_n. When y_n is large the error controller will use small stepsizes to resolve the transient. Small integration steps experience small changes in $w(t)$ and the two quantities ω_n and ς_n will therefore also be small. As the transient decays the disturbance φ_n gets smaller due to its dependence on y_n. The error controller will increase the stepsize and eventually φ_n approaches a value determined by $w(t)$ (through ω_n and ς_n). If the properties of $w(t)$ stay relatively constant with respect to time, the error control loop reaches a kind of stationary point where the stepsize h_n and the error r_n are almost constant (cf. the simulation of the Robertson problem in Section 3.6). The stationary point is a complicated function of $w(t)$, the integration method coefficients, and the error set-point ε.

On the Stability Boundary

The scenario above assumes that after the transient has decayed the error estimate is dominated by the effect from ω_n and ς_n, cf. (3.15). In an integration method with bounded stability region, e. g. explicit methods, the situation may be different. If the increasing stepsize reaches h_s we have $|P(h_s\lambda)| = 1$ for some dominating eigenvalue λ of A. We use the term *dominating* to indicate that λ is the eigenvalue where $|P(h_s\lambda)| = 1$ is the most restrictive on h_s. When $|P(h_n\lambda)|$ is close to one, ω_n will accumulate in y_n, cf. (3.15), and the error estimate may be large although ω_n and ς_n are small. By adjusting the stepsize about h_s the error controller reaches a kind of stationary point where the error estimate is kept almost constant. The dynamics of the stepsize-error relation at this point is very different from (3.12).

To derive a model for the stepsize-error relation about the stationary point h_s we consider the case with constant $w(t)$. Repeating the arguments of [Shampine, 1975; Higham and Hall, 1989] we assume that the

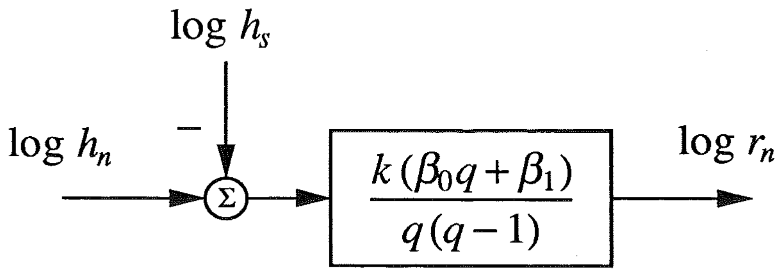


Figure 3.10 For a stepsize h_s which places $h_s\lambda$ on the stability boundary of the integration method, the integration method can be described with the dynamic relation (3.22) between $\log h_n$ and $\log r_n$. The external disturbance $\log h_s$ depends on both the differential equation and the integration method.

with E' and P' being the derivatives of the functions E and P, respectively. Then

$$|\hat{e}_{n+1}| = |P(h_s\lambda)| \left|\frac{h_n}{h_s}\right|^{C_E} \left|\frac{h_{n-1}}{h_s}\right|^{-C_E+C_P} |\hat{e}_n| \tag{3.19}$$

Using EPS, and noting that $|P(h_s\lambda)| = 1$, (3.19) can be written in terms of transfer functions as

$$\log r_n = G_p(q)\left(\log h_n - \log h_s\right), \qquad G_p(q) = \frac{C_E q + C_P - C_E}{q(q-1)}, \tag{3.20}$$

which is quite different from (3.12). EPUS introduces a normalization of r_{n+1} with h_n, which changes the expression for $G_p(q)$ in (3.20) to

$$G_p(q) = \frac{(C_E-1)q + C_P - C_E + 1}{q(q-1)}. \tag{3.21}$$

For future calculations (Chapter 4) it is beneficial to rewrite (3.20) and (3.21) as

$$\log r_n = G_{p2}(q)\left(\log h_n - \log h_s\right), \qquad G_{p2}(q) = \frac{k(\beta_0 q + \beta_1)}{q(q-1)} \tag{3.22}$$

with

$$\beta_0 = \begin{cases} C_E/k, & \text{EPS} \\ (C_E-1)/k, & \text{EPUS} \end{cases} \qquad \beta_1 = \begin{cases} (C_P - C_E)/k, & \text{EPS} \\ (C_P - C_E + 1)/k, & \text{EPUS} \end{cases} \tag{3.23}$$

The dynamical behavior of the process is governed by $G_{p2}(q)$ and the only influence of the differential equation is the external perturbation $\log h_s$ (cf. Figure 3.10). The result (3.22) can be generalized to all linear systems dominated by a real eigenvalue [Hall, 1985].

A Second Order Problem with Complex Eigenvalues. Assume that A has the following structure

$$A = \begin{pmatrix} \beta & -\gamma \\ \gamma & \beta \end{pmatrix}, \quad \beta < 0, \quad \gamma \neq 0, \tag{3.24}$$

with the eigenvalues $\lambda = \beta + i\gamma$ and $\bar{\lambda} = \beta - i\gamma$.

Finding a stationary solution similar to the one for the scalar case, requires the use of an error measure based on an absolute 2-norm [Hall, 1986; Hall and Higham, 1988]. The matrix A is normal, and $A^TA = \lambda\bar{\lambda}I$, $A^T + A = (\lambda + \bar{\lambda})I$. Hence

$$\begin{aligned} \|\hat{e}_{n+1}\|_2^2 &= y_n^T E(h_nA)^T E(h_nA) y_n = E(h_n\lambda)E(h_n\bar{\lambda})y_n^T y_n \\ &= |E(h_n\lambda)|^2 \|y_n\|_2, \\ \|y_{n+1}\|_2^2 &= y_n^T P(h_nA)^T P(h_nA) y_n = P(h_n\lambda)P(h_n\bar{\lambda})y_n^T y_n \\ &= |P(h_n\lambda)|^2 \|y_n\|_2. \end{aligned}$$

There exists a stepsize h_s so that $|P(h_s\lambda)| = 1$, and consequently the 2-norm of y is constant. Consider small perturbations, i. e. $h_n = h_s(1+\kappa_n)$ [Hall, 1986; Hall and Higham, 1988], and use the same type of derivation as in (3.18). Then, from (3.17)

$$\|\hat{e}_{n+1}\|_2^2 \approx |1 + \kappa_n C_E|^2 \frac{|1 + \kappa_{n-1}C_P|^2}{|1 + \kappa_{n-1}C_E|^2} \|\hat{e}_n\|_n^2,$$

and using $|1 + \kappa C|^2 \approx 1 + 2\kappa \operatorname{Re} C$, we may write

$$\|\hat{e}_{n+1}\|_2^2 \approx \left(\frac{h_n}{h_s}\right)^{2\operatorname{Re} C_E} \left(\frac{h_{n-1}}{h_s}\right)^{2\operatorname{Re}(-C_E+C_P)} \|\hat{e}_n\|_2^2. \tag{3.25}$$

Hence, if the definitions for C_E and C_P are changed to

$$C_E(h_s\lambda) = \operatorname{Re}\left(h_s\lambda \frac{E'(h_s\lambda)}{E(h_s\lambda)}\right), \quad C_P(h_s\lambda) = \operatorname{Re}\left(h_s\lambda \frac{P'(h_s\lambda)}{P(h_s\lambda)}\right) \tag{3.26}$$

the result (3.22) holds also for problems with the special structure (3.24). The result can be generalized to all linear systems dominated by a complex conjugate pair of eigenvalues having a negative real part [Hall, 1985; Hall and Higham, 1988].

The derivation of (3.25) hinges on the structure of A and the specific norm that is used. If the problem is transformed with the nonsingular matrix T, resulting in the system matrix $T^{-1}AT$, the norm $\|\cdot\|_T = \|T\cdot\|_2$ has to be used to establish (3.22).

In practice T is unknown, and consequently the norm will not be aligned with the differential equation, i. e. the Jacobian is not normal. The constant value h_s is then no longer an equilibrium. The model (3.22) is, however, still of interest. It is possible to find a stepsize sequence varying around h_s keeping the norm of the solution y_n constant. The larger the misalignment between the differential equation and the norm, the larger the amplitude of the variations. If we consider deviations about this varying stepsize sequence (and the corresponding error sequence) it is reasonable to assume a behavior similar to (3.22).

Other Error Norms

The existence of a constant equilibrium state h_s is a result of using a uniform absolute error measure aligned with the differential equation. The situation is more complicated when a mixed absolute-relative norm is used, [Higham and Hall, 1989; Gustafsson, 1991].

For an integration method with $P(h_s\lambda) = -1$, e. g. RKF4(5), the numerical solution x_n of (3.14) oscillates around $w(t)$. A relative error norm then results in an oscillating error estimate r_n even if $\hat{e}_n$ has constant size. The variation in r_n makes the error controller adjust the stepsize, and as a result r_n and h_n oscillate around ε and h_s, respectively. The oscillations may sometimes be reduced by using averaged solution values when forming $\tilde{y}$ in the error norm (2.14), but in general, a mixed absolute-relative norm will cause oscillations [Higham and Hall, 1989]. The model (3.22) is, however, still of interest since it reveals the behavior of the underlying dynamic stepsize-error relation.

Implicit Integration Methods

The result (3.22) is valid also for implicit Runge-Kutta methods. Normally this is of little interest since many implicit methods are designed to be L-stable. Methods only possessing the weaker property A-stability may have ∞ on the boundary of $\mathcal{S}$. Thus, as the stepsize is increased the stepsize-error relation will eventually approach the behavior predicted by (3.22).

stepsize is locally constant, $h_n = h_s$. Then

$$\hat{e}_{n+1} = E(h_sA)y_n = E(h_sA)P(h_sA)y_{n-1} = P(h_sA)\hat{e}_n, \tag{3.16}$$

which indicates that on the stability boundary the estimate of the local truncation error stays approximately constant. During normal integration the stepsize will vary, but we expect the average stepsize to keep $h_n\lambda$ close to the boundary of the stability region. In addition, from (3.15), y_n should be close to the span of the dominant eigenvector(s) of A.

An important feature of the stepsize-error relation at the stability boundary is the way old errors are propagated to the next step. For a varying stepsize (3.16) becomes

$$\begin{aligned} \hat{e}_{n+1} &= E(h_nA)y_n = E(h_nA)P(h_{n-1}A)y_{n-1} \\ &= E(h_nA)P(h_{n-1}A)E^{-1}(h_{n-1}A)\hat{e}_n. \end{aligned} \tag{3.17}$$

This recurrence relation is normally hard to analyze, but for certain choices of A and error norm, it is possible to find "equilibrium states" for h_n, r_n and y_n. We will use the first nonvanishing variation to derive a stepsize-error model in two cases where it is possible to obtain these states [Gustafsson, 1991].

A Scalar Problem. Consider the scalar problem one gets by setting $A = \lambda$. The constant stepsize h_s, where $|P(h_s\lambda)| = 1$, leads to the "stationary" solution $|y_{n+1}| = |y_n|$. Consider small perturbations around the equilibrium state h_s, i.e. $h_n = h_s(1 + \kappa_n)$ [Hall, 1985]. Then (3.17) can be written

$$\begin{aligned} \hat{e}_{n+1} &= E\big(h_s\lambda(1+\kappa_n)\big)\frac{P\big(h_s\lambda(1+\kappa_{n-1})\big)}{E\big(h_s\lambda(1+\kappa_{n-1})\big)}\hat{e}_n \\ &\approx E(h_s\lambda)\,(1+\kappa_nC_E)\,\frac{P(h_s\lambda)\,(1+\kappa_{n-1}C_P)}{E(h_s\lambda)\,(1+\kappa_{n-1}C_E)}\hat{e}_n \\ &\approx P(h_s\lambda)\,(1+\kappa_n)^{C_E}\,(1+\kappa_{n-1})^{-C_E+C_P}\,\hat{e}_n \\ &= P(h_s\lambda)\left(\frac{h_n}{h_s}\right)^{C_E}\left(\frac{h_{n-1}}{h_s}\right)^{-C_E+C_P}\hat{e}_n \end{aligned} \tag{3.18}$$

where we have used $1 + \kappa C \approx (1+\kappa)^C$ for small κ and

$$C_E(h_s\lambda) = h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}, \qquad C_P(h_s\lambda) = h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)}$$

3.6 Experimental Verification of the Models

The process models (3.12) and (3.22) are variational approximations of (2.3) about $h_n = 0$ and $h_n = h_s$. The approximations are valid in a neighborhood around the approximation points, but the previous derivations do not indicate how large these neighborhoods are. Somewhere in the interval $[0, h_s]$ there is a transition from (3.12) to (3.22). The models, however, do not tell where in the interval the transition occurs or how it takes place.

System identification [Ljung, 1987; Söderström and Stoica, 1989] can be used to partly answer these questions. The stepsize and error sequence obtained from solving any appropriate differential equation can be recorded and used to fit a dynamical relation between $\log h_n$ and $\log r_n$. The identification is done in closed loop and in the general case it may be hard to distinguish between the effects from the stepsize ($\log h_n$) and the disturbance ($\log \varphi_n$ or $\log h_s$) [Ljung, 1987, pp. 365–366]. By using data sequences from a time interval where the disturbance, i. e. ω_n and ς_n in (3.15), stays almost constant the problem can be overcome.

A Nonlinear Problem Dominated by a Real Eigenvalue

The Robertson problem [Enright *et al.*, 1975, Problem D2]

$$\begin{aligned} \dot{y}_1 &= -0.04y_1 + 0.01y_2y_3 & y_1(0) &= 1.0 \\ \dot{y}_2 &= 400y_1 - 100y_2y_3 - 3000y_2^2 & y_2(0) &= 0.0 \\ \dot{y}_3 &= 30y_2^2 & y_3(0) &= 0.0 \end{aligned} \tag{3.27}$$

is a stiff problem originating from reaction kinetics. The first 0.3 seconds of its solution are shown in Figure 3.11. At $t = 0$ the eigenvalues of the Jacobian are 0, 0, and −0.040. Due to mass balance one of the eigenvalues stays at the origin for all t. During the initial transient ($0 \le t \le 0.005$) one eigenvalue moves from the origin to −2190, while the other moves from −0.040 to −0.4046. For $t > 0.005$ the small eigenvalue moves slowly towards the origin. At $t = 0.3$ it has reached −0.3650. The large eigenvalue starts by moving towards the origin and then changes direction and moves outwards again. Over the time range $0.005 < t < 2$ the change is less than 2 % in magnitude.

The Robertson problem is quite stiff and most explicit methods will be inefficient for any part of the solution other than the initial transient. After the transient the stepsize used in an explicit method will

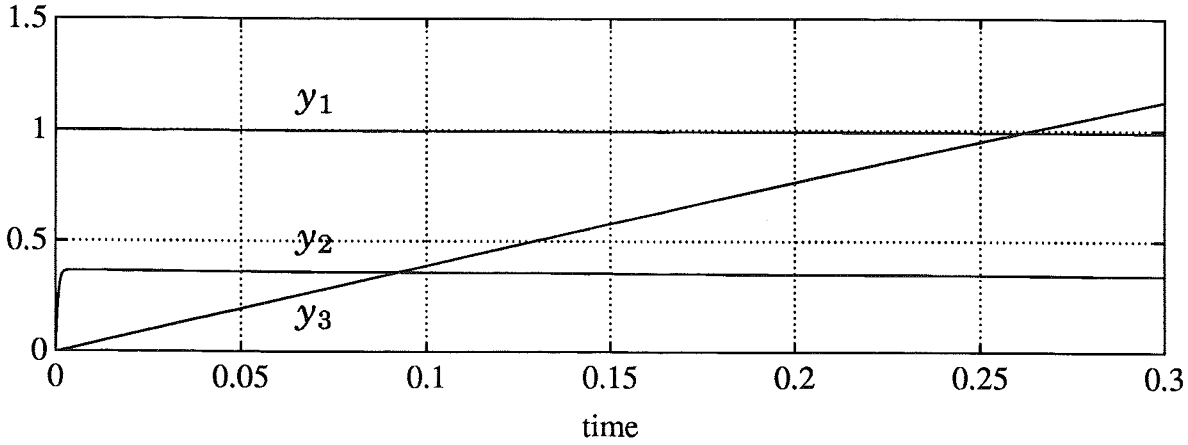


Figure 3.11 The solution of the Robertson problem (3.27).

be almost completely governed by the large negative eigenvalue $\lambda_{\max}$. This eigenvalue moves only slowly with time and $\log \varphi$ will be almost constant.

Identification of Model Parameters for DOPRI(4)5

The differential equation (3.27) was solved with $\varepsilon = 10^{-6}, 10^{-7}, \ldots, 10^{-12}$ using DOPRI(4)5 (cf. Appendix A or [Dormand and Prince, 1980]) in XEPS mode. A mixed absolute-relative 2-norm (2.14) with $\eta = 0.01$ and $\tilde{y} = |y|$ was used. The Robertson problem belongs to the class of problems where the standard stepsize selection rule (2.22) may cause oscillations in the error control loop, cf. Figure 1.1. To prevent this the new controller described in Chapter 4 was used. After an initial transient the disturbance φ, cf. ω_n and ς_n in (3.15), reaches an almost constant value and since $\varepsilon \approx \varphi h^k$ the stepsize stays essentially constant too, cf. Figure 3.12. The larger the value of ε, the larger this constant stepsize. This holds true for tolerances below 10^{-7}, while for larger values of ε the stepsize reaches h_s and is restricted by stability. We have $h_s\lambda_{\max} = 1.51 \cdot 10^{-3} \cdot (-2190) = -3.31$, which corresponds to the value of $\partial S \cap \mathbb{R}^-$ for DOPRI(4)5, cf. Figure A.1 in Appendix A. The error control loop is stable for all values of ε.

The described behavior makes the Robertson problem ideal for identification. By chosing different tolerance levels we can get stationary behavior in the error control loop at several points in the interval $[0, h_s]$. The disturbance φ_n is almost constant and its effect can be removed by high pass filtering the data prior to identification.

The differential equation was solved a second time with an added perturbation on the set-point ε of the error control loop. From $t > 0.1$,

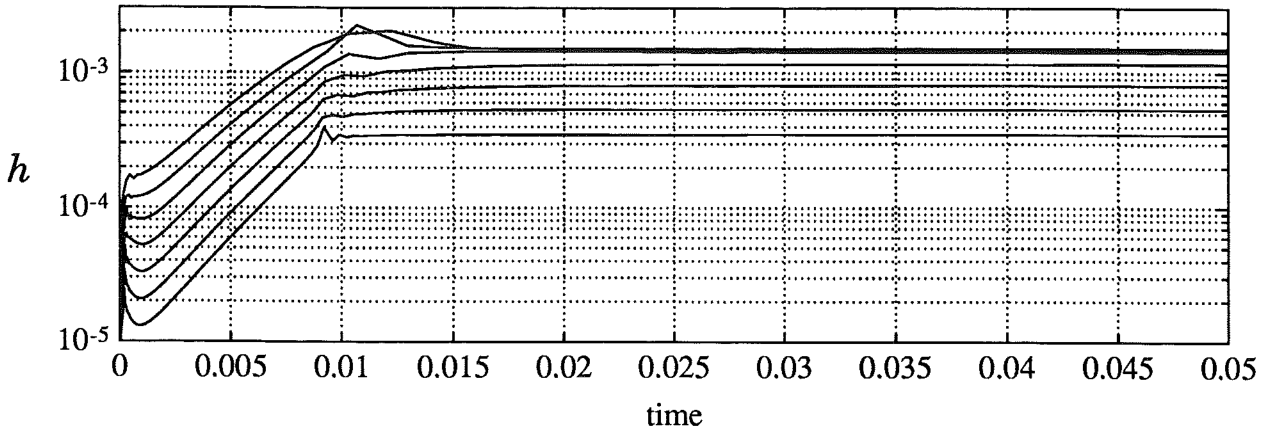


Figure 3.12 The stepsize for different tolerances when solving the Robertson problem (3.27) using DOPRI(4)5. The curves come in order, i. e. the lower one corresponds to $\varepsilon = 10^{-12}$, the second from the bottom corresponds to $\varepsilon = 10^{-11}$, and so on. The stepsize is practically constant for $t > 0.02$. For $\varepsilon = 10^{-7}$ and $\varepsilon = 10^{-6}$ the stationary value of the stepsize is identical and the curves overlap, implying that the stepsize is limited by numerical stability.

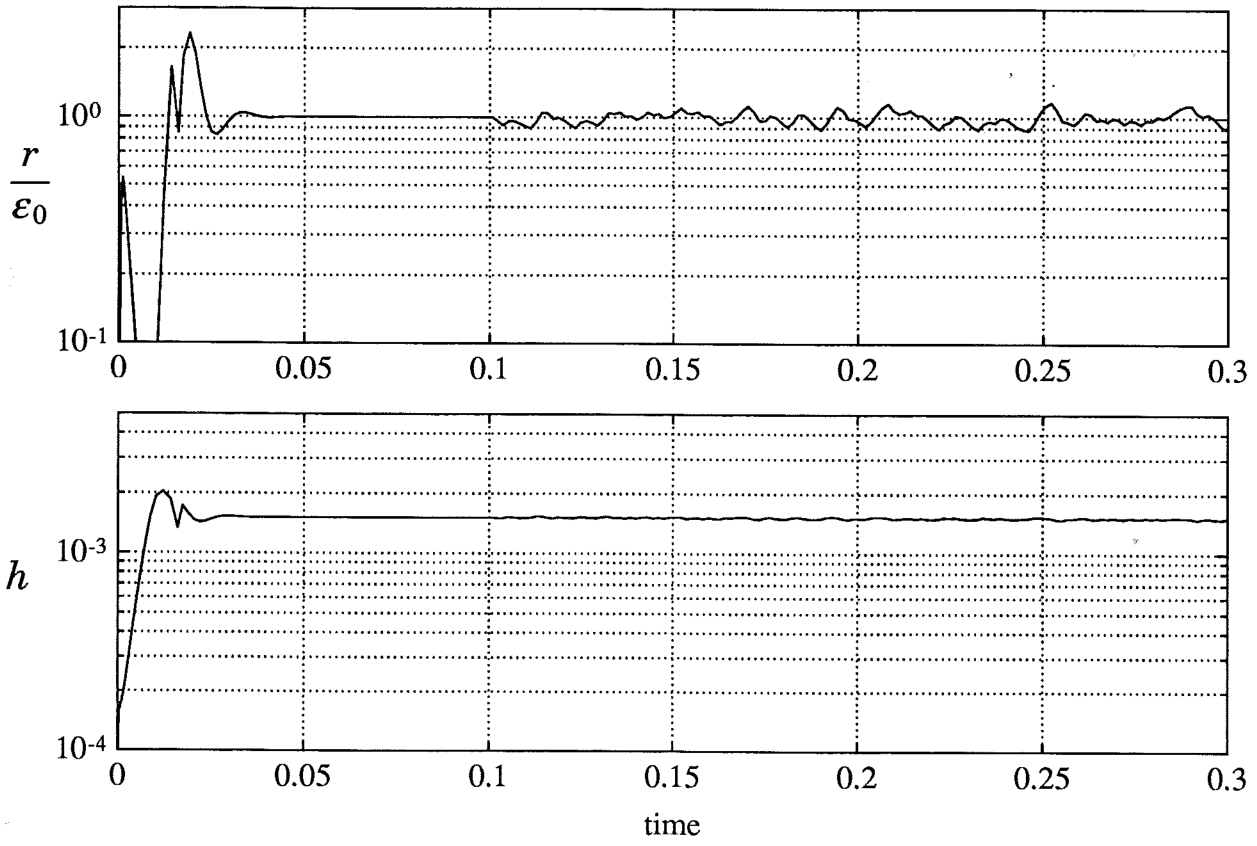


Figure 3.13 Part of the stepsize and error estimate sequence recorded from the identification experiment with DOPRI(4)5 and $\varepsilon_0 = 10^{-6}$. The irregularities in the data sequences for $t > 0.1$ show the effect of the perturbation of ε, and are not due to poor error control.

		$G_{p1}(q) = \frac{k}{q}$	$G_{p2}(q) = \frac{k(\beta_0 q + \beta_1)}{q(q+\alpha)}$		
ε_0	$h_n\lambda_{\max}$	k	$k\beta_0$	$k\beta_1$	α
10^{-6}	−3.31	–	5.85	0.226	−1.00
10^{-7}	−3.29	–	5.85	0.211	−0.978
10^{-8}	−3.14	–	5.87	0.0100	−0.816
10^{-9}	−2.55	–	5.87	−0.417	−0.337
10^{-10}	−1.78	5.78	–	–	–
10^{-11}	−1.18	5.51	–	–	–
10^{-12}	−0.772	5.34	–	–	–

Table 3.1 The identified transfer functions from $\log h_n$ to $\log r_n$. The integration method was DOPRI(4)5 using XEPS.

after the transient has died out completely, an excitation signal was added by perturbing $\log \varepsilon$ according to $\log \varepsilon = \log \varepsilon_0 + 0.05\Delta\varepsilon_n$. Here $\Delta\varepsilon_n$ was a PRBS (pseudo random binary signal) sequence [Ljung, 1987; Söderström and Stoica, 1989] alternating between +1 and −1. The perturbation was small causing the stepsize to vary only a few percent around its stationary value. For each value of ε_0 the stepsize h_n and the error estimate r_n were recorded and stored. The simulation interval was chosen so that more than 1000 integration steps were taken. During the data logging there were no rejected steps. Figure 3.13 shows part of the stepsize and the error estimate recorded for $\varepsilon_0 = 10^{-6}$.

The large eigenvalue of the differential equation moves and thereby introduces a slow drift in the recorded data sequences. The drift is small, only a few percent, but not negligible compared to the variations in $\log h_n$ and $\log r_n$ due to the perturbation of ε. The drift was removed by detrending the data sequences before the identification [Ljung, 1987, pp. 387–389]. For each pair of data signals (stepsize and error sequence) an ARMA-model from $\log h_n$ to $\log r_n$ was identified using the identification toolbox in PRO-MATLAB [Mat, 1990]. Different statistical tests as well as simulation using the obtained models were used to decide upon reasonable model orders [Ljung, 1987; Söderström and Stoica, 1989]. The identified transfer functions are listed in Table 3.1.

The identified parameters are in very close agreement with the theoretical results. DOPRI(4)5 has, theoretically, k equal to 5 when using XEPS. The negative real axis intersects ∂S at −3.31, and at this point

$C_E = 5.85$ and $C_P = 6.07$. According to (3.12) and (3.22) one would hence expect the models

$$G_{p1}(q) = \frac{5.0}{q}, \qquad G_{p2}(q) = \frac{5.85q + 0.228}{q(q-1.00)}, \tag{3.28}$$

which agree well with the practical results for $h_n\lambda_{\max}$ small and $h_n\lambda_{\max}$ on $\partial\mathcal{S}$, cf. Table 3.1.

For $h_n\lambda_{\max} = -0.772$ the following model was obtained:

$$\log r_n = \frac{5.34}{q} \log h_n. \tag{3.29}$$

One would expect the value 5.0, cf. (3.28), instead of 5.34. The discrepancy is explained by the fact that $h_n\lambda_{\max}$ differs significantly from zero. As a result, one does not observe the asymptotic behavior but a slightly modified one. By assuming that the behavior of the nonlinear equation is completely governed by $\lambda_{\max}$ it is possible to analytically estimate the modified behavior. We obtain

$$\hat{e}_{n+1} \approx E(h_n\lambda_{\max})y_n, \qquad r_{n+1} \approx |E(h_n\lambda_{\max})| \, \|y_n\|.$$

It should then approximately hold that

$$\begin{aligned} \frac{\partial \log r_{n+1}}{\partial \log h_n} &= h_n \frac{\partial \log r_{n+1}}{\partial h_n} \approx h_n \frac{\partial}{\partial h_n} \log\left(|E(h_n\lambda_{\max})| \, \|y_n\|\right) \\ &= h_n\lambda_{\max} \frac{E'(h_n\lambda_{\max})}{E(h_n\lambda_{\max})} = C_E(h_n\lambda_{\max}). \end{aligned} \tag{3.30}$$

For $h_n\lambda_{\max} = -0.772$ the formula (3.30) evaluates to 5.28. This is in almost perfect agreement with the value 5.34, cf. (3.29), obtained from identification (the polynomials $P(z)$ and $E(z)$ for DOPRI(4)5 are given in Appendix A). Note that this small discrepancy is due to our assumption that $\lambda_{\max}$ dominates the behavior and that in (3.30) we differentiate the approximation for r_{n+1}. The polynomial $E(z)$ is, however, sufficiently smooth to allow this operation. As $h_n\lambda_{\max}$ increases, higher order terms in $E(z)$ will play a larger role. Hence (3.30) predicts the gain 5.55 for $h_n\lambda_{\max} = -1.78$, while the identification gives 5.78.

As $h_n\lambda_{\max}$ is further increased it will approach $\partial\mathcal{S}$ where now a model of the form (3.22) is expected. Although not verified by theoretical derivations the identification indicates a gradual change from (3.12)

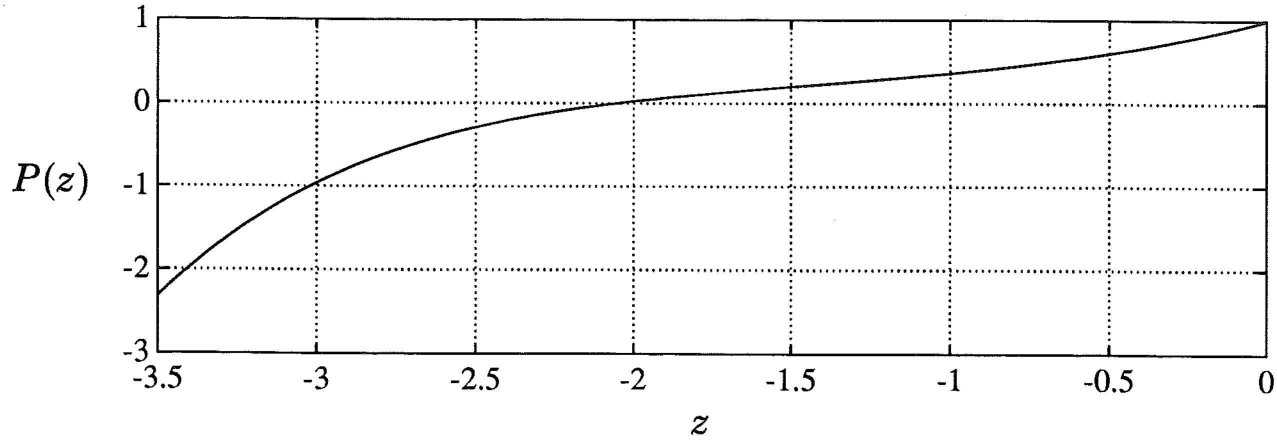


Figure 3.14 The value of $P(z)$ in RKF4(5). When z approaches the boundary of the stability region ($z \approx 3.0$) the value of $P(z)$ is negative.

to (3.22). As an example consider $h_n\lambda_{\max} = -2.55$. For this value the identification resulted in

$$\log r_n = \frac{5.87q - 0.417}{q(q - 0.337)} \log h_n.$$

Identification of Model Parameters for RKF4(5)

The identification procedure was repeated for RKF4(5) using EPS. The method coefficients (cf. Appendix A) and the models (3.12) and (3.22) predict

$$G_{p1}(q) = \frac{5.0}{q}, \qquad G_{p2}(q) = \frac{5.53q + 0.354}{q(q - 1.00)}. \tag{3.31}$$

Between these models we expect the same type of gradual change as for DOPRI(4)5. This is confirmed by the identification results (see Table 3.2), except for some strange results around $\varepsilon = 10^{-8}$.

The reason for the deviation in the identification is that RKF4(5) is a method where $P(z)$ takes negative values as z approaches the boundary of the stability region (cf. Figure 3.14). As a result the numerical solution will, for large stepsizes, oscillate around the true solution. This affects the error estimate, since the solution enters the calculation of the error norm (2.14), and we get an oscillative perturbation of the error estimate. This perturbation may be amplified or damped depending on the dynamic properties of the error control loop. The dependence is very complicated since the perturbation enters nonlinearly through the error norm. The effect can be seen in Figure 3.15, which depicts the same situation as Figure 3.13 but for $\varepsilon_0 = 10^{-8}$ and RKF4(5) using EPS. In our

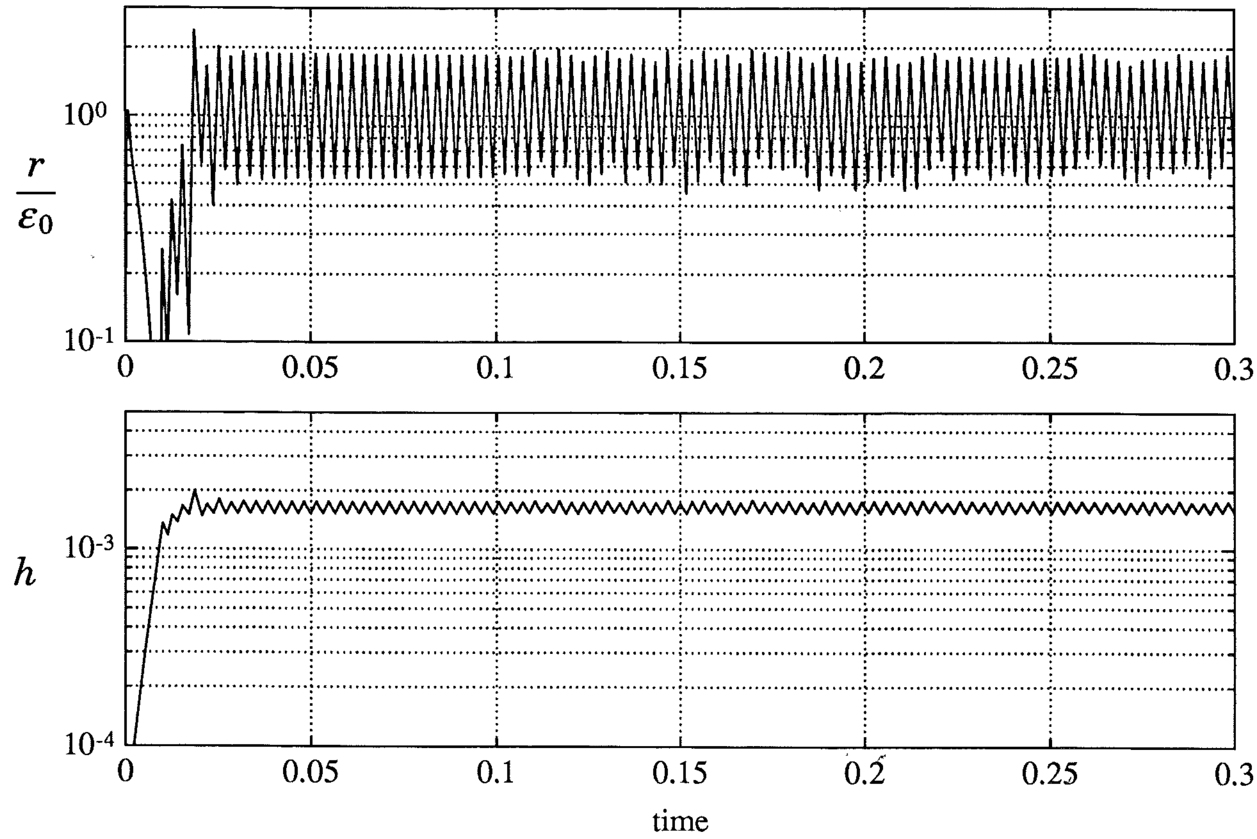


Figure 3.15 Part of the stepsize and error estimate sequence recorded from the identification experiment with RKF4(5) and $\varepsilon_0 = 10^{-8}$. For the stepsizes used $P(z)$ takes a value close to -1, which causes an oscillatory disturbance in the control loop. The perturbation on ε is started at $t = 0.1$, but it can hardly be observed due to the severe intrinsic oscillations. Compare with the behavior for DOPRI(4)5 in Figure 3.13.

experiment we have chosen ε/tol small to prevent rejected steps, and in a realistic practical setting the oscillations depicted in Figure 3.15 would lead to many rejected steps.

The disturbance has rather large amplitude (compared to the variations due to the perturbation of ε) and it is very concentrated in frequency. Practically all its energy is concentrated to the Nyquist frequency, corresponding to an alternating sequence $(-1)^n$. The disturbance is fed back and the oscillation appears in both $\log h_n$ and $\log r_n$. The identification will try to make the model capture the behavior due to the disturbance, and the result is models as in Table 3.2. The influence of the disturbance can be lowered by low pass filtering the data sequences prior to the identification. We used a first order digital Butterworth filter with half the Nyquist frequency as cut-off. The filter has a zero in -1. The corresponding identification results are shown in Table 3.3.

		$G_{p1}(q) = \frac{k}{q}$	$G_{p2}(q) = \frac{k(\beta_0 q + \beta_1)}{q(q+\alpha)}$		
ε_0	$h_n\lambda_{\max}$	k	$k\beta_0$	$k\beta_1$	α
10^{-6}	−3.02	–	5.54	0.350	−1.00
10^{-7}	−3.02	–	5.24	−0.588	−0.910
10^{-8}	−3.01	–	11.5	−5.40	−0.985
10^{-9}	−2.46	–	5.78	−0.146	0.492
10^{-10}	−1.68	5.42	–	–	–
10^{-11}	−1.09	5.37	–	–	–
10^{-12}	−0.705	5.26	–	–	–

Table 3.2 The identified transfer functions from $\log h_n$ to $\log r_n$. The integration method was RKF4(5) using EPS.

		$G_{p1}(q) = \frac{k}{q}$	$G_{p2}(q) = \frac{k(\beta_0 q + \beta_1)}{q(q+\alpha)}$		
ε_0	$h_n\lambda_{\max}$	k	$k\beta_0$	$k\beta_1$	α
10^{-6}	−3.02	–	5.53	0.355	−1.00
10^{-7}	−3.02	–	5.48	0.359	−0.995
10^{-8}	−3.01	–	5.67	0.279	−0.983
10^{-9}	−2.46	–	5.63	−0.587	0.357
10^{-10}	−1.68	5.28	–	–	–
10^{-11}	−1.09	5.35	–	–	–
10^{-12}	−0.705	5.25	–	–	–

Table 3.3 The identified transfer functions from $\log h_n$ to $\log r_n$. The integration method was RKF4(5) using EPS. The data was low pass filtered prior to the identification. The filter was a first order digital Butterworth filter with half the Nyquist frequency as cut-off. The filter has a zero in −1.

General Observations

The identification results obtained for DOPRI(4)5 and RKF4(5) applied to a differential equation dominated by a negative real eigenvalue are quite typical. The model (3.12) is valid for a large span of stepsizes, although most of the methods behave as if k was slightly larger than the true method order. The transition to the model (3.22) takes place fairly close to the boundary of the stability region. It is not until h_n is close to h_s that we recognize the behavior predicted by (3.22).

For integration methods like RKF4(5), where $P(h_s\lambda) = -1$, the situation is more complicated. When the method operates in the nonasymptotic region, i. e. where $P(h\lambda)$ is negative and starts approaching -1, the numerical solution will oscillate causing a perturbation of the error estimate. This perturbation depends on the differential equation and the stepsize in a very complicated way. In some cases the perturbation will strongly affect the error control. The underlying linear model does not, however, change. This can be seen from identification based on low pass filtered data.

A high frequency disturbance may be present also when integrating differential equations dominated by complex eigenvalues. As the stepsize approaches the boundary of the stability region there will be a transition from (3.12) to (3.22). The error norm is normally not aligned with the differential equation, i. e. a nonnormal Jacobian. This will result in a perturbation to the error estimate similar to the one obtained for $P(h_s\lambda) = -1$. As before, the perturbation causes a dependence between the stepsize and the error estimate, which may dominate over the one predicted by the linear model (3.22). The frequency of the disturbance depends on the position of the dominating eigenvalues.

3.7 Conclusions

The stepsize-error relation can often be modeled quite satisfactorily using the standard asymptotic model (2.21). It is, however, important not to assume φ constant as in (3.2). The variations in φ can be captured quite well with a model were the change in $\log \varphi$ is assumed constant. All in all, we get, (3.10) and (3.12),

$$\log r_{n+1} = k \log h_n + \log \varphi_n, \qquad \log \varphi_n = \log \varphi_{n-1} + \nabla \log \varphi_{n-1}.$$

The k value experienced during integration is normally close to the order of the error estimator. There are, however, practical situations where its value may be very different. This is difficult to model since during normal integration a change in k cannot be distinguished from a change in φ. At repeated rejected steps, k can be estimated by (3.13), i. e.

$$\hat{k} = \frac{\log r_{n+1} - \log r_n}{\log h_n - \log h_{n-1}},$$

where (r_{n+1}, h_n) and (r_n, h_{n-1}) are errors and stepsizes from two consecutive rejected steps.

When stability limits the stepsize the model (3.12) is not sufficient. A better description is (3.22)

$$\log r_{n+1} = \frac{k(\beta_0 q + \beta_1)}{q(q-1)} (\log h_n - \log h_s),$$

where the parameters β_0 and β_1 depend on the integration method and the dominating eigenvalue of the differential equation. The stepsize h_s makes $|P(h_s \lambda)| = 1$.

As the stepsize is increased from 0 to h_s our practical experiments indicate a gradual change from (3.12) to (3.22). The error controller must be robust enough to handle this process variation. In addition, methods where $P(z)$ is negative close to the stability boundary may in certain cases introduce a large oscillating disturbance in the error control loop. This disturbance causes a fluctuation in the error estimate that cannot be predicted with neither (3.12) nor (3.22).

4

Control of Explicit Runge-Kutta Methods

The control problem in an explicit Runge-Kutta method may seem deceptively straightforward. The error controller only has to decide whether to accept or reject the current integration step, and then choose a new stepsize based on the error estimate. A severe complication, however, is the change in process behavior when stability restricts the stepsize. Satisfactory error control in this situation conflicts with good attenuation of the influence of a varying φ.

Large process variations calls for a *robust* controller, cf. the discussion in [Franklin *et al.*, 1986, pp. 521–525]. We generalize the standard controller and arrive at a new controller that provides a reasonable trade off between performance and robustness to process variations. The controller is easy to implement and provides improved overall performance.

4.1 Problems with the Standard Controller

To gain insight we will start by analyzing the properties of the error control loop (cf. Figures 2.7 and 4.1) that results when combining the standard controller (2.22) with the two process models (3.12) and (3.22).

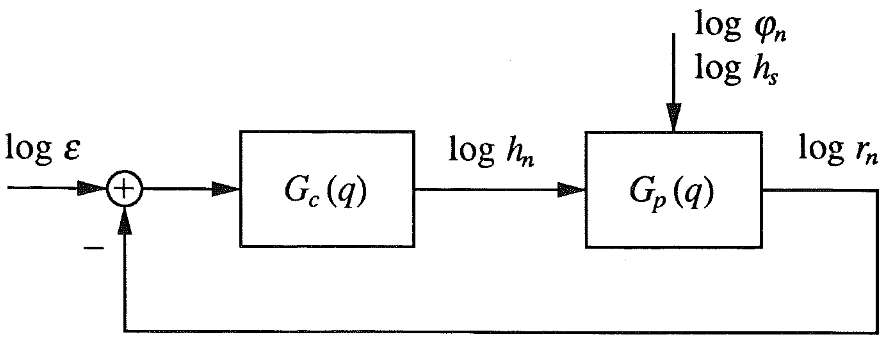


Figure 4.1 The error control loop where the transfer function $G_p(q)$ represents the process and $G_c(q)$ the controller. The differential equation acts as an external perturbation $\log\varphi_n$ (or $\log h_s$).

Using logarithmic variables the standard controller (2.22) can be written

$$\log h_n = \log h_{n-1} + \frac{1}{k}(\log\varepsilon - \log r_n) \tag{4.1}$$

which is recognized as a *discrete-time integrating* controller [Gustafsson *et al.*, 1988] with *integration gain* $k_I = 1/k$, cf. [Åström and Wittenmark, 1990, Section 8.3] about discrete-time integrating controllers in general. The set-point is $\log\varepsilon$, the measured variable $\log r_n$, and the control variable (and controller state) is $\log h_n$.

In the following we will keep the integration gain as a free parameter to investigate its influence on the stepsize control loop. Furthermore, to facilitate the analysis, (4.1) is rewritten as

$$\log h_n = G_{c1}(q)(\log\varepsilon - \log r_n), \qquad G_{c1}(q) = k_I\frac{q}{q-1}. \tag{4.2}$$

Similarly, for ease of reference, we restate the equations for the two processes (3.12) and (3.22), i.e.

$$\begin{aligned} \log r_n &= G_{p1}(q)\log h_n + q^{-1}\log\varphi_n, & G_{p1}(q) &= kq^{-1},\\ \log r_n &= G_{p2}(q)(\log h_n - \log h_s), & G_{p2}(q) &= \frac{k(\beta_0 q + \beta_1)}{q(q-1)}. \end{aligned}$$

Asymptotically Small Stepsizes

For asymptotically small stepsizes the process is well approximated by (3.12). The same model structure is valid both for EPS and EPUS. By combining (3.12) and (4.2) the control loop (see Figure 4.1) can be written

$$\log r_n = G_\varepsilon(q)\log\varepsilon + G_\varphi(q)\log\varphi_n \tag{4.3}$$

where

$$\begin{aligned} G_\varepsilon(q) &= \frac{G_{c1}(q)G_{p1}(q)}{1+G_{c1}(q)G_{p1}(q)} = \frac{kk_I}{q-1+kk_I}, \\ G_\varphi(q) &= \frac{1}{q(1+G_{c1}(q)G_{p1}(q))} = \frac{q-1}{q(q-1+kk_I)} \end{aligned} \tag{4.4}$$

are transfer functions from tolerance and disturbance to error estimate, respectively.

The characteristic equation (the denominator of $G_\varepsilon(q)$) has a root at $1 - kk_I$. This pole determines the stability as well as the transient properties of the closed loop system. The difference operator $q-1$ in the numerator of $G_\varphi(q)$ will remove constant components in $\log \varphi_n$ at a rate determined by the position of the pole. Consequently, if the closed loop system is stable, r_n will eventually approach ε since $G_\varepsilon(1) = 1$.

Choosing $k_I = 1/k$, as is normally done in the standard controller, places the pole at the origin. This corresponds to so called *deadbeat* control [Åström and Wittenmark, 1990, p. 138] and makes the system as fast as possible. For this choice a constant disturbance is compensated in a single step, but at the price of making the stepsize try to compensate fast fluctuations in the error estimate. Moving the pole along the real axis towards 1, makes the system slower and the stepsize sequence will in general be smoother. Making the system slower will, however, also degrade the ability to attenuate the influence from fast changes in φ. The position of the pole is thus a trade off between response time and sensitivity, which makes the value of k_I a design parameter that should not be regarded as given by $1/k$.

Stepsizes Restricted by Stability

The system description (4.3) and (4.4) is not valid if the magnitude of φ_n gets too small. A small φ_n leads to a large stepsize, and if φ_n is sufficiently small the stepsize will be limited by numerical stability. Using the model (3.22) together with the standard controller (4.2), the closed loop system can be written

$$\begin{aligned} \log r_n &= G_\varepsilon(q)\log\varepsilon + G_{h_s}(q)\log h_s \\ G_\varepsilon(q) &= \frac{G_{c1}(q)G_{p2}(q)}{1+G_{c1}(q)G_{p2}(q)} = \frac{kk_I(\beta_0 q+\beta_1)}{q^2+(kk_I\beta_0-2)q+1+kk_I\beta_1} \\ G_{h_s}(q) &= -\frac{G_{p2}(q)}{1+G_{c1}(q)G_{p2}(q)} = -\frac{k(q-1)(\beta_0 q+\beta_1)}{q(q^2+(kk_I\beta_0-2)q+1+kk_I\beta_1)} \end{aligned} \tag{4.5}$$

Here $G_\varepsilon(1) = 1$ and $G_{h_s}(1) = 0$. Therefore $\log h_s$, which is constant or slowly varying, will be removed and eventually $\log r_n$ will equal $\log \varepsilon$, provided the closed loop system is stable.

The transient behavior as well as the stability of the control system is governed by the roots of

$$q^2 + (kk_I\beta_0 - 2)q + 1 + kk_I\beta_1 = 0. \tag{4.6}$$

The system is stable if these roots are inside the unit circle. In [Hall, 1985; Hall, 1986] another stability test is derived, consisting of an eigenvalue check for a 2 by 2 matrix. In the special case $k_I = 1/k$, the characteristic equation of that matrix equals the polynomial in (4.6).

When choosing $k_I = 1/k$ the roots of (4.6) are inside the unit circle for the (β_0, β_1) values depicted by the triangle in Figure 4.2. The plot also shows (β_0, β_1) pairs calculated from (3.26) with $h_s\lambda$ varying along the boundary of the stability regions of the explicit Runge-Kutta methods in Appendix A (cf. also Figures A.1–A.2 and Tables A.1–A.2 in Appendix A). As is readily seen, the values of (β_0, β_1) are often *outside* the stability region. The resulting instability in the error control loop is the cause of the misbehavior in Example 1.1.

The stability problem cannot be solved by altering the value of k_I. As seen in (4.6) changing k_I will merely scale the triangle in Figure 4.2, and we will not succeed in stabilizing the cases where $\beta_1 > 0$. These cases are important. As an example consider solving a differential equation dominated by a negative real eigenvalue. The process dynamics is then determinated by (β_0, β_1) evaluated at the intersection of $\partial\mathcal{S}$ and the negative real axis. Many integration methods have $\beta_1 > 0$ for this case (cf. Tables A.1–A.2 in Appendix A).

The Time-Variability of the Stepsize-Error Relation

When using the standard controller the error control loop is often unstable for $h_n \approx h_s$. The instability makes the error grow (possibly causing rejected steps) and the controller will reduce the stepsize to bring the error down to ε. The reduction of the stepsize moves $h_n\lambda$ inside $\mathcal{S}$ and the process changes behavior from (3.22) to (3.12), making the system regain stability. The error decreases and the controller may respond by increasing the stepsize, again placing $h_n\lambda$ on $\partial\mathcal{S}$. The cycle repeats itself creating an oscillatory stepsize sequence. The phenomenon is a consequence of the stationary value of the disturbance φ_n corresponding to

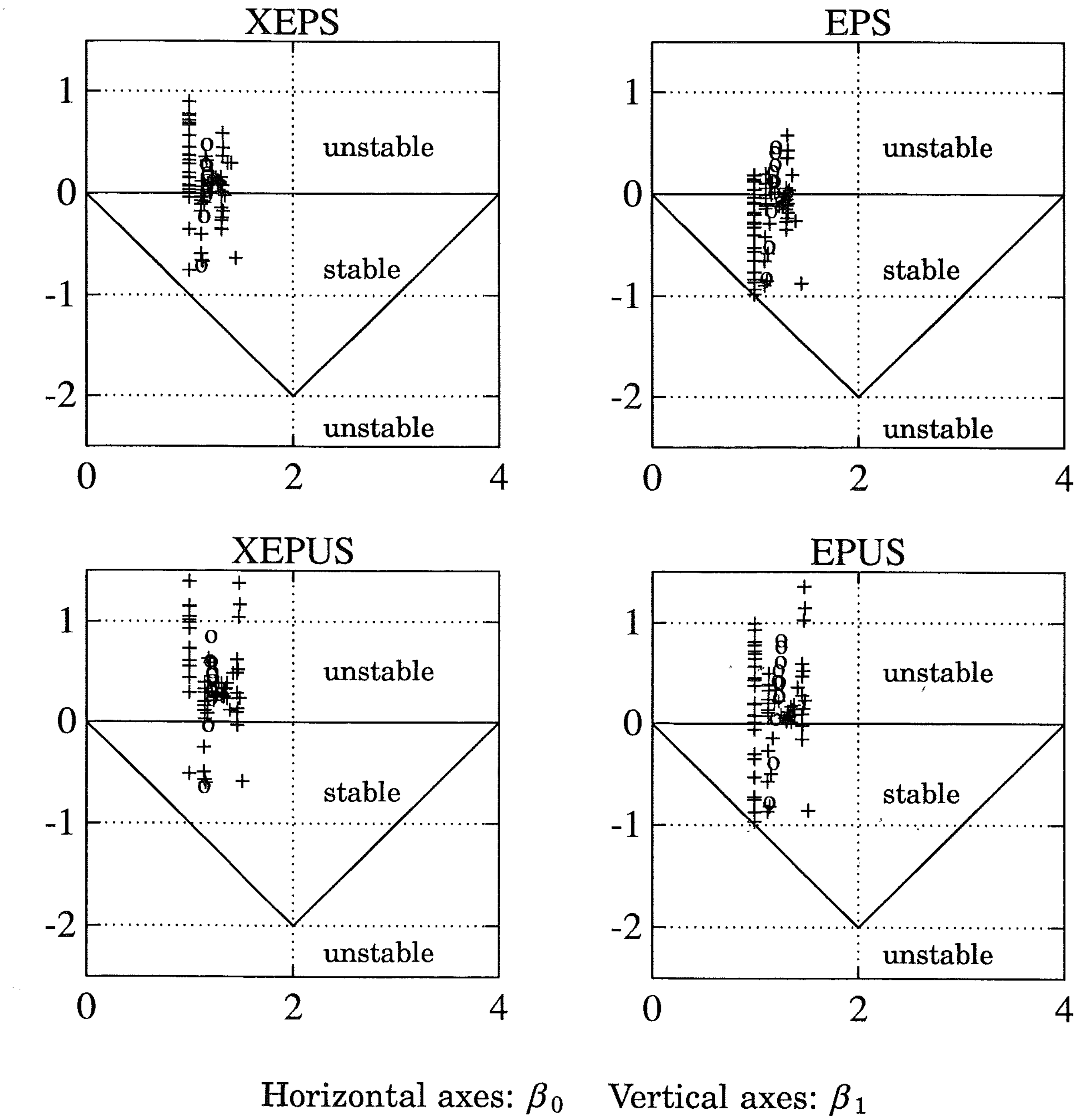


Figure 4.2 The standard controller with $k_I = 1/k$ results in a stable error control loop if (β_0, β_1) is inside the plotted triangle. This is seldom the case as seen from the plotted (β_0, β_1) values. The (β_0, β_1) values are calculated from (3.26) with $h_s\lambda$ varying along the boundary of the stability regions of the explicit Runge-Kutta methods depicted in Figures A.1 and A.2 in Appendix A. For each method 12 values where used for $h_s\lambda$. They were chosen linearly spaced with $\pi/2 < \arg(h_s\lambda) < \pi$. The 'o' corresponds to values for DOPRI45. Changing the value of k_I will not make the stability region extend above $\beta_1 = 0$, and is therefore not a solution to the stability problem. A pure integrating controller (4.2), and hence the standard stepsize selection rule, must be discarded in the case $\beta_1 > 0$.

a stepsize that is outside the region where the process model (3.12) is valid.

The error control loop may oscillate even in cases where the controller stabilizes both (3.12) and (3.22). The changes in the stepsize make the process behavior vary between (3.12) and (3.22), and this variation may by itself cause a type of "oscillation" in the error control loop.

4.2 Control Objectives

The overall objective of the controller is to make possible an efficient and accurate integration of the differential equation. To get a more concrete objective we will interpret this as:

- The controller should keep r close to ε.
- The controller structure should be simple.

The set-point ε of the controller is chosen based on the user-specified *tol*. If the error r is larger than *tol* the step is rejected. How well the first objective is fulfilled will therefore determine how close ε can be chosen to *tol*. A smooth error sequence will decrease the risk of a rejected step and allow ε to be chosen closer to *tol*, leading to fewer steps needed to carry out the integration.

The second objective is important to minimize overhead, and also to facilitate integration startup and restart after a rejected step. A complicated controller with many internal states will be difficult to restart based on the information available after a rejected step.

To fulfill the first objective, it is reasonable to try to achieve the following:

- Good disturbance attenuation for the process model (3.12), i.e. r should not vary too much when φ varies.
- A stable closed loop system for reasonable (β_0, β_1) values in the model (3.22).

These two properties are in conflict. The former calls for a fast controller with accurate prediction of the φ sequence, while the latter demands improved stabilizing properties.

The attenuation of φ is a trade off between variations in r and variations in h. Completely rejecting φ means having a perfect model for its variation, and letting h vary accordingly. Any extra filtering or limit on the stepsize will result in φ affecting r. A characteristic of many control problems is that rather large disturbance attenuation can be achieved by moderate control signal variations, while completely rejecting the disturbance is much more expensive (in terms of control signal variation), a general reference is [Boyd and Barrat, 1991]. We will strive to attenuate the effect from φ as much as possible and not put too much restrictions on the variation of h. The stepsize variations caused by an unstable error control loop (4.5) are, however, different. They do not reduce, but rather increase, the variations in r, and should of course be avoided.

4.3 Changing the Controller Structure

The properties of the closed loop system depend on the controller as well as on the process, and one may change either one to improve the behavior of the system. Higham and Hall [Higham and Hall, 1987] approach the problem by changing the process, viz. the integration algorithm. When constructing an explicit Runge-Kutta method there is some freedom in the choice of parameters, cf. (2.9) in Example 2.7. Normally this freedom is used to minimize error coefficients or to maximize the stability region of the method, but Higham and Hall exploit it to change C_E and C_P so that the closed loop system is stable when the standard controller is used.

It is our opinion that a better way to approach the problem is to change the controller. Then the freedom in choice of parameters in the method can be used to improve its numerical properties, while the instability in the error control loop is solved by improving the controller, i.e. the stepsize selection rule.

A PI Controller

The process model valid on the stability boundary (3.22) can be satisfactorily controlled with a controller on the form

$$G_{c2}(q) = k_I \frac{q}{q-1} + k_P = \frac{(k_I + k_P)q - k_P}{q-1}. \qquad (4.7)$$

This controller is by no means arbitrary. When rewriting the standard controller (2.22) on the form (4.1) it is recognized as a discrete-time

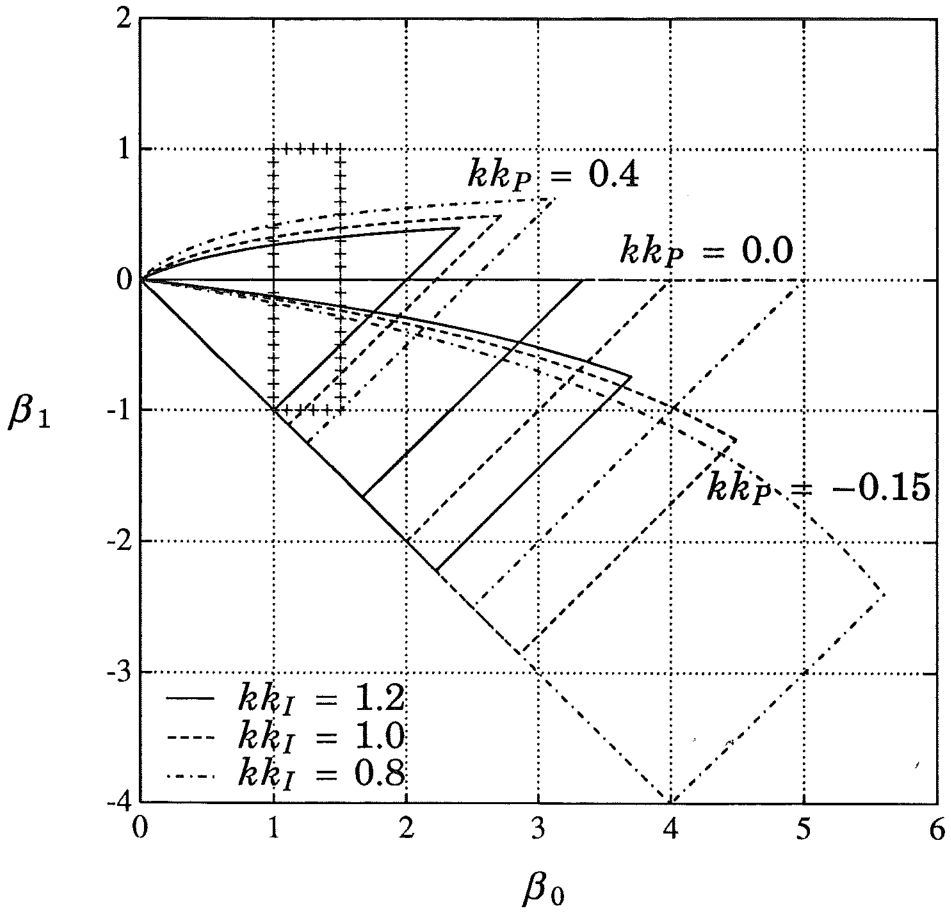


Figure 4.3 Changing the controller parameters k_I and k_P affects which values on (β_0, β_1) the controller it manages to stabilize in the model (3.22). Altering k_I changes the size of the (β_0, β_1) region while k_P mainly affects its orientation. The area inside the (+) region corresponds to (β_0, β_1) values taken from the explicit Runge-Kutta methods in Appendix A. To cover this area it seems as if k_P must be positive while k_I have to be reduced from its original value. Also compare with the original stability area depicted in Figure 4.2.

integrating controller [Gustafsson *et al.*, 1988]. Once this is realized, the modification to a discrete-time *proportional-integral* (or PI) controller as in (4.7) is straightforward. A PI controller provides, in general, better stabilizing properties than an I controller [Åström and Hägglund, 1988; Franklin *et al.*, 1986; Hairer and Wanner, 1991].

The (β_0, β_1) values that the PI controller may stabilize depend on the values of kk_I and kk_P. The standard I controller corresponds to $kk_I = 1$ and $kk_P = 0$ and gives the triangular stability region depicted in Figure 4.2. Choosing kk_I and kk_P differently affects the stability region as demonstrated in Figure 4.3. The (β_0, β_1) values for DOPRI45 (the 'o' in Figure 4.2 are inside the box marked with + in Figure 4.3. It turns out that the (β_0, β_1) values for the other Runge-Kutta methods, cf. Figure

4.2, behave similarly and designing a good controller for DOPRI45 will lead to a good controller also for the other methods. Figure 4.3 indicates that to cover the box we must have $kk_I < 1$ and $kk_P > 0$.

The improved stability for the model (3.22) comes with a price. With the PI controller the ability to attenuate φ in (3.12) is traded for stabilization of (3.22). This is typical for most robust control designs. When enlarging the set of process models the controller is supposed to handle, some of the performance often has to be sacrificed. Ideally, we would like our new controller to have improved attenuation of the disturbance φ. Chapter 3 suggested a φ model that assumes linear change in $\log \varphi$. For satisfactory attenuation of a ramp disturbance a controller needs to include two integrators [Åström and Wittenmark, 1990, p. 134]. Such a controller would be perfectly natural in the region where (3.12) holds, but the two integrators make it very difficult to stabilize (3.22).

One way to resolve the trade off between φ attenuation and stability would be to use different controllers depending on if (3.12) or (3.22) holds. A major problem with this approach is to know when to switch between the different controllers. There are different suggestions for how to detect when an explicit method encounters stiffness [Shampine, 1991; Hairer and Wanner, 1991, pp. 22–25], and these algorithms do a good job of determining that the method operates on the boundary of the stability region. They are, however, less accurate in deciding when the switch from (3.12) or (3.22) takes place, and may therefore miss the correct time to switch controllers. Changing controllers introduces a time variability into the stepsize control loop, and this action may by itself provoke stepsize oscillations. We have, consequently, decided to use a fixed controller and tune it so that it performs satisfactorily under all operating conditions.

Explicit Formulation of the PI controller

The PI controller (4.7) can be reformulated in a way that resembles the standard controller (2.22). Some manipulations applied to

$$\log h_n = G_{c2}(q)(\log \varepsilon - \log r_n)$$

yield [Gustafsson *et al.*, 1988]

$$h_{n+1} = \left(\frac{\varepsilon}{r_{n+1}}\right)^{k_I} \left(\frac{r_n}{r_{n+1}}\right)^{k_P} h_n. \tag{4.8}$$

From this expression it is clear that the proportional term is equivalent to taking the most recent development of r into account when deciding upon h_{n+1}. It is also clear that this type of controller is trivial to implement in existing ODE codes.

4.4 Tuning the Controller Parameters

The controller parameters should be chosen so that the closed loop system behaves well for both the asymptotic model (3.12) as well as for the model valid on the boundary of the stability region (3.22). One way to make the choice is to use a pole placement strategy [Åström and Wittenmark, 1990], e. g. we study the closed loop poles in the two cases and choose parameters so that they are positioned at locations corresponding to good closed loop behavior.

Combining the PI controller (4.7) with (3.12) and (3.22) makes the closed loop system read

$$\begin{aligned}
\log r_n &= G_\varepsilon(q)\log\varepsilon + G_\varphi(q)\log\varphi_n \\
G_\varepsilon(q) &= \frac{G_{c2}(q)G_{p1}(q)}{1+G_{c2}(q)G_{p1}(q)} = \frac{(kk_I+kk_P)q - kk_P}{q^2+(kk_I+kk_P-1)q-kk_P} \\
G_\varphi(q) &= \frac{1}{q\big(1+G_{c2}(q)G_{p1}(q)\big)} = \frac{q-1}{q^2+(kk_I+kk_P-1)q-kk_P}
\end{aligned}$$

and

$$\begin{aligned}
\log r_n &= G_\varepsilon(q)\log\varepsilon + G_{h_s}(q)\log h_s \\
G_\varepsilon(q) &= \frac{G_{c2}(q)G_{p2}(q)}{1+G_{c2}(q)G_{p2}(q)} \\
&= \frac{\beta_0(kk_I+kk_P)q^2+(kk_I\beta_1+kk_P(\beta_1-\beta_0))q-kk_P\beta_1}{q^3+\big(\beta_0(kk_P+kk_I)-2\big)q^2+\big(kk_I\beta_1+kk_P(\beta_1-\beta_0)+1\big)q-kk_P\beta_1} \\
G_{h_s}(q) &= -\frac{G_{p2}(q)}{1+G_{c2}(q)G_{p2}(q)} \\
&= \frac{-k(q-1)(\beta_0 q+\beta_1)}{q^3+\big(\beta_0(kk_P+kk_I)-2\big)q^2+\big(kk_I\beta_1+kk_P(\beta_1-\beta_0)+1\big)q-kk_P\beta_1}
\end{aligned}$$

respectively. The closed loop poles are given by the roots of the two

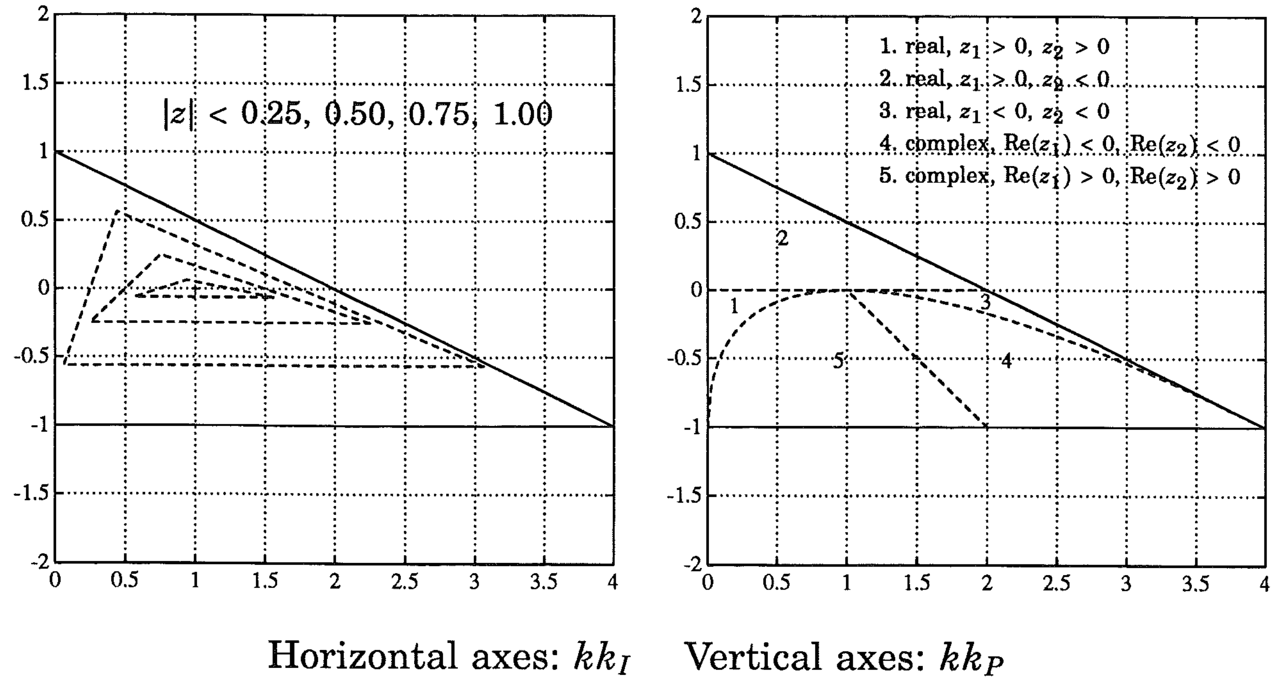


Figure 4.4 The plot to the left demonstrates how the maximum absolute value of the roots of (4.9a) depends on kk_I and kk_P, while the plot to the right depcits the character of the two roots. The kk_I and kk_P values needed to stabilize (3.22), i. e. (4.9b), typically belong in region 2.

characteristic equations

$$q^2 + (kk_I + kk_P - 1)q - kk_P = 0 \tag{4.9a}$$

$$\begin{aligned} &q^3 + \big(\beta_0(kk_P + kk_I) - 2\big)q^2 \\ &\qquad + \big(kk_I\beta_1 + kk_P(\beta_1 - \beta_0) + 1\big)q - kk_P\beta_1 = 0. \end{aligned} \tag{4.9b}$$

From Figure 4.3 we know that compared to the standard controller kk_I has to be decreased and kk_P increased in order to stabilize the relevant (β_0, β_1) area. This parameter change will most likely reduce the performance for the standard model (3.12), i. e. (4.9a). The model (3.12) describes, however, the most common situation, and any parameter adjustment should retain good error control properties for this case.

We start by investigating how a change in kk_I and kk_P affects the poles of the error control for the standard model (3.12). Figure 4.4 depicts the properties of the roots of (4.9a) for different values of kk_I and kk_P. To handle (4.9b) we need $kk_I < 1$ and $kk_P > 0$, which makes (4.9a) have two real roots of opposite sign. Having a closed loop pole on the real negative axis is not desirable. It corresponds to an oscillatory time response, which may lead to over- or undershoot when controlling r_n about ε. Too large an overshoot will make r_n exceed tol and the result will be a rejected step.

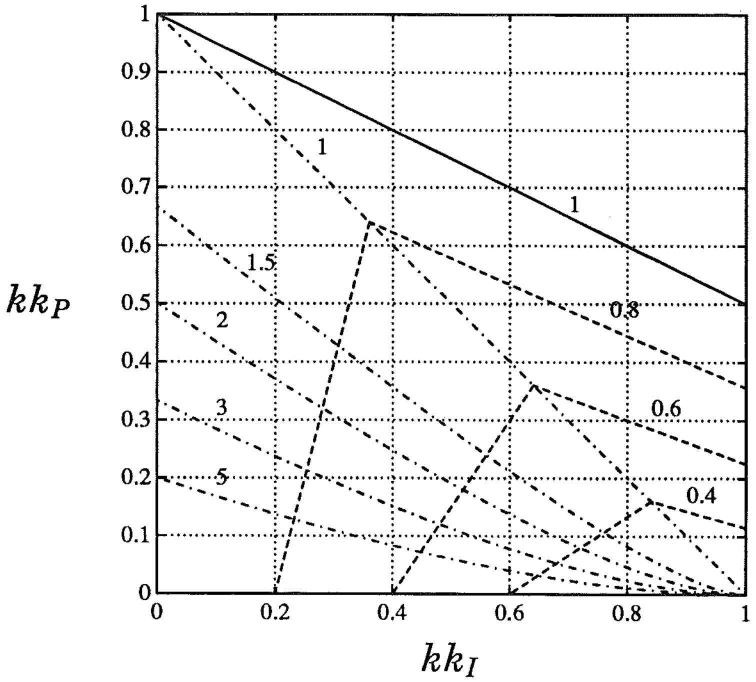


Figure 4.5 The plot shows the upper left corner of the stability region in Figure 4.4. The properties of the roots of (4.9a) are depicted as a function of kk_I and kk_P. The full line shows the border of the stability region, the dashed lines depict regions where the maximum absolute value of the two poles is below the values written beside the lines, and the dash-dotted lines show where the magnitude of the positive pole is α times larger than the magnitude of the negative pole. The value of α is indicated beside the lines.

If aiming at stabilizing (4.9b) it is, unfortunately, not possible to avoid the pole on the negative real axis in the normal case (4.9a). Its effect on the closed loop system can, however, be decreased, by choosing the controller parameters so that the positive pole has larger magnitude than the negative. The situation is demonstrated in terms of kk_I and kk_P values in Figure 4.5. To provide good error control for the case (3.12) the poles of (4.9a) should be close to 0, i. e. deadbeat control, with the positive pole dominating over the negative. This corresponds to kk_I close to 1 and kk_P as small as possible. With this in mind we turn to the model (3.22) and investigate what parameter values it takes to provide good pole locations also for (4.9b).

The case (3.22) when numerical stability restricts the stepsize is difficult. It is not sufficient to make the poles of (4.9b) stable for the relevant (β_0, β_1) values. Ideally, the closed loop poles should be well inside the unit circle to provide good damping. If this is not the case, any disturbance that excite the error control loop would give rise to a slowly decaying (possibly oscillatory) perturbation of the stepsize sequence. As a result the risk of rejected steps increases.

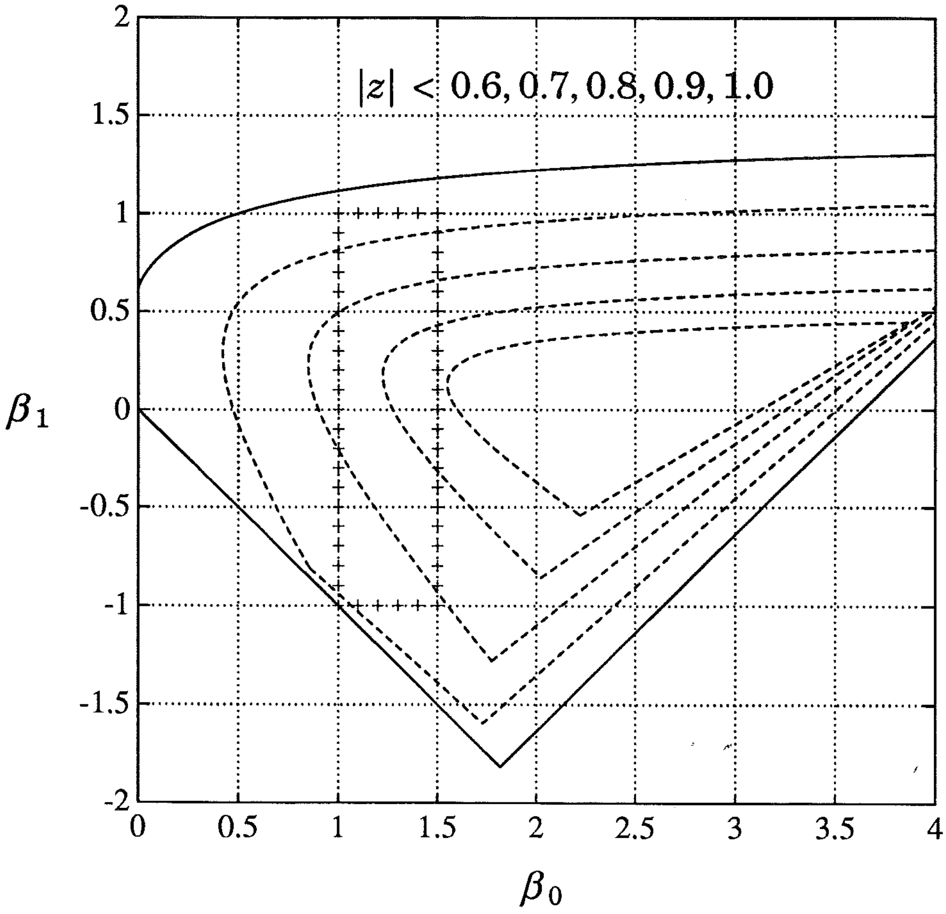


Figure 4.6 The values $kk_I = 0.3$ and $kk_P = 0.4$ results in a PI controller that manages to stabilize the relevant (β_0, β_1)-region (the box marked with '+'). The full line depicts the border of the stability region and the dashed lines the level curves for the maximum absolute value of the roots of (4.9b), i. e. $|z| < 0.6, 0.7, 0.8, 0.9$. The controller not only stabilizes but also manages to provide reasonable damping.

By varying the values of kk_I and kk_P and investigating the resulting stability regions, cf. Figure 4.3, it was found that the values

$$kk_I = 0.3, \qquad kk_P = 0.4, \tag{4.10}$$

result in an error controller with sensible damping for most relevant (β_0, β_1) values. The resulting stability region, in terms of (β_0, β_1), is shown in Figure 4.6. The values (4.10) seem reasonable also for the normal case (3.12). The closed loop will be more sluggish than with the deadbeat action of the standard stepsize selection rule (2.22), but not so slow that it seriously impairs the performance.

The values of k_I and k_P are a trade off between different properties of the error control loop. From the discussion above it should be clear that the values suggested in (4.10) are not the only possible ones. As we will see in Section 4.6, these values work well in practice. It cannot, however, be excluded that some fine-tuning might improve the performance for specific integration methods.

4.5 Restart Strategies

When a step is rejected the next step to be taken is a retry, and from the last attempt it is known what to expect ahead. The error from the rejected step can be used to calculate an approximate $\log \varphi_n$. This quantity is then used to calculate a new stepsize h_n^* just as was done in the standard controller, i. e.

$$h_n^* = \left(\frac{\varepsilon}{r_{n+1}}\right)^{1/k} h_n. \tag{4.11}$$

The stepsize from (4.11) is normally of appropriate size to restart the integration. It may, however, happen that the error again is too large, resulting in another rejected step. One reason could be that the stepsize-error relation locally has a lower k than the expected (see Chapter 3). For explicit methods this is, normally, an unlikely case, and we do not include any special measures in the controller, e. g. estimation of k as in (3.13).

A Predicting Restart Strategy

The standard controller (2.22), and in a sense also the PI controller (4.8), is derived assuming $\log \varphi$ constant or slowly varying. The performance will, consequently, not be acceptable for problems where $\log \varphi$ changes rapidly (cf. Example 1.2). A solution is to use a controller that predicts changes in $\log \varphi$. This approach leads to a controller with more than one integrator (cf. Chapter 5), which makes it difficult to stabilize the system on $\partial \mathcal{S}$, i. e. the model (3.22). Inside $\mathcal{S}$, however, such a controller works very well.

Although the stepsize-error relation on $\partial \mathcal{S}$ in practice excludes the use of a controller that predicts φ, the prediction idea can be used after a rejected step. A rejected step can be interpreted as a failure to capture a large increase in φ. Since φ includes a lot of structure it seems likely that φ will continue to increase also in the following steps. Part of this increase can be anticipated by having the stepsize decrease appropriately after the rejected step.

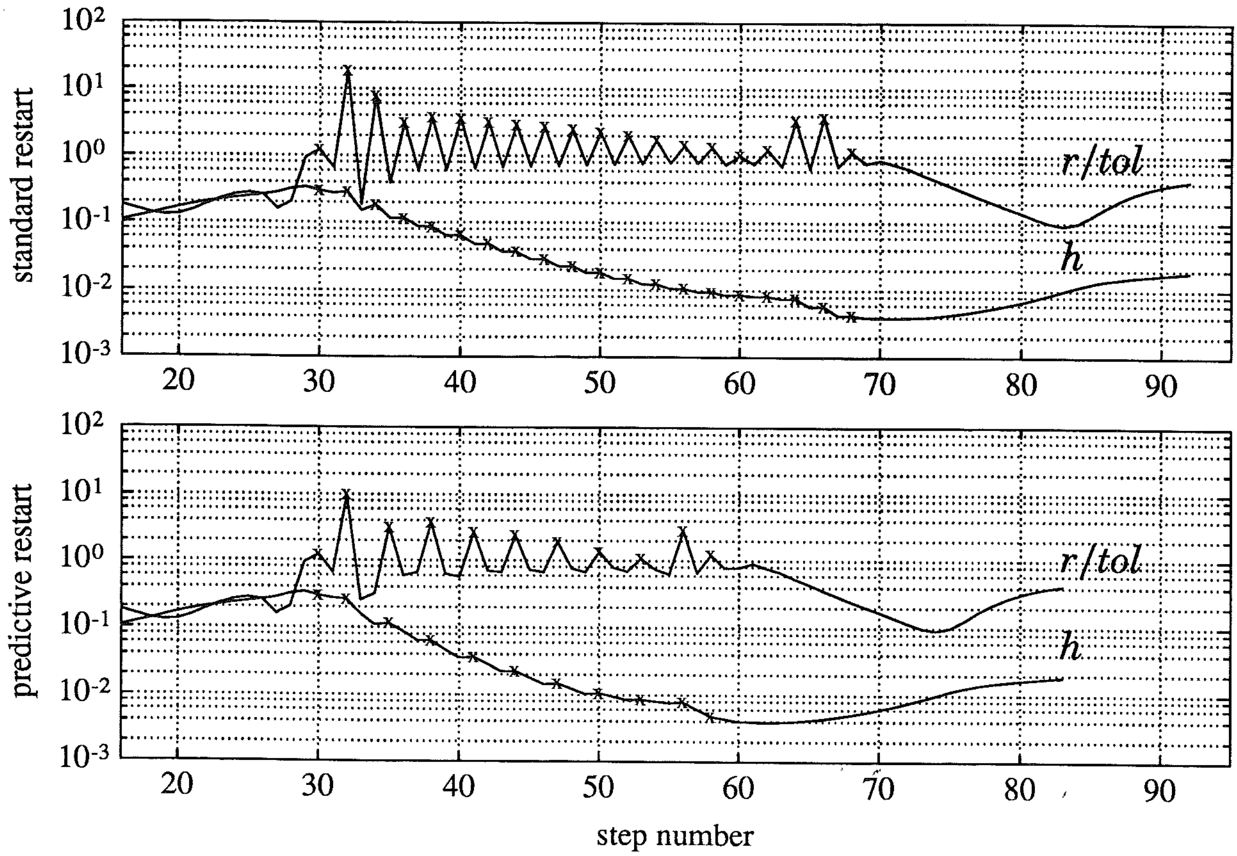


Figure 4.7 This plot depicts some variables from the time interval $t \in [1.0, 5.0]$ when integrating the Brusselator (Example 4.1). The upper plot originates from an integration using the standard restart strategy (4.11), cf. Figure 1.2, while the lower shows the improvement when using a restart strategy that predicts the increase in φ. The rejected steps are indicated by 'x'.

After a rejected step we calculate and apply h_n^* from (4.11). The quotient h_n/h_n^* is a measure of the increase in φ and probably also a good guess of how large increase to expect in the next step. Hence, we preschedule a similar stepsize change also in the next step, cf. (3.10) and Figure 3.7. The internal state of the controller can be updated to achieve this end.

EXAMPLE 4.1—Restart strategy in DOPRI(4)5

The described restart strategy was used when integrating the Brusselator [Hairer *et al.*, 1987, p. 112]

$$\begin{aligned}\dot{y}_1 &= A + y_1^2 y_2 - (B+1)y_1\\ \dot{y}_2 &= By_1 - y_1^2 y_2\end{aligned}$$

with $A = 2$, $B = 8$, $y_1(0) = 1$, and $y_2(0) = 4$. The controller (4.8) with $kk_I = 0.3$ and $kk_P = 0.4$, and a mixed absolute-relative 2-norm (2.14) with $\eta = 10^{-2}$, $\tilde{y} = |y|$, $tol = 5 \cdot 10^{-6}$, and $\varepsilon/tol = 0.8$ was used. The Brusselator was solved in Example 1.2 and its solution is depicted in Figure 1.2. In the time interval $t \in [3.0, 4.8]$ there are many rejected steps. When introducing the predictive restart strategy (cf. Figure 4.7) the number of rejected steps was decreased by almost 50% (from 20 to 11). Each rejected step is now normally followed by (at least) two accepted steps. The first accepted step is explained by (4.11) while the second is due to the special update of the controller state. □

The predicting restart strategy normally performs well, and reduces the total number of integration steps by a few percent. Even though the strategy is based on the asymptotic model (3.12), it often performs well also when stability restricts the stepsize. One notable exception, however, is the combination of integration methods where $P(h_s\lambda) = -1$ and a tolerance which puts the stepsize-error relation in the region where the transition between (3.12) and (3.22) takes place. The sign of $P(h\lambda)$ causes an oscillation in the error control loop. This φ "variation" cannot be predicted by the model (3.10), and the predicting restart strategy may cause an increase in the number of rejected steps. The predicting restart strategy is therefore a poor choice for methods like RKF4(5) and DOPRI(7)8, and to get comparable results we will not use it in any of the numerical tests.

4.6 Numerical Examples

To evaluate the new PI controller we have run a large set of numerical tests. The objectives have been many: to verify the properties of the PI controller, to check the parameter values chosen for k_I and k_P, as well as finding a reasonable way to choose ε based on tol. Several different integration methods (RKF2(3), RKF4(5), DOPRI(4)5, and DOPRI(7)8) were used. All methods were implemented in a common software framework [Gustafsson, 1992], which allows for a fair comparative study where parameters, strategies, etc., may be varied one at a time, under controlled experimental conditions. The test problems (see below) were integrated for a large set of tolerance levels and parameter (k_I, k_P, ε/tol) values. All in all, to obtain a reasonably dense evaluation of the parameter space more than 100 000 integrations were done.

The Test Problems

Three different test problems have been used in the evaluation. The problems have very different properties and they have been chosen so that different types of stepsize-error relations will be experienced during the integration. Although the solutions (cf. Figures 4.8–4.10) look deceptively simple, we shall see that the special characteristics of these problems give quite different results.

Problem 1. The van der Pol oscillator [Hairer *et al.*, 1987, pp. 107, 236]

$$\begin{aligned} \dot{y}_1 &= y_2, & y_1(0) &= 2 \\ \dot{y}_2 &= \sigma\left(1-y_1^2\right)y_2 - y_1, & y_2(0) &= 0 \end{aligned}$$

is a standard numerical test problem (cf. Example 3.1). It has a periodic solution with different properties along the trajectory. The solution contains both smooth parts, as well as transient parts where φ changes quickly. For $\sigma = 10$ the smooth part is mildly stiff. The problem is studied in the time interval $0 \le t \le 15$.

Problem 2. The Robertson problem [Enright *et al.*, 1975, Problem D2]

$$\begin{aligned} \dot{y}_1 &= -0.04y_1 + 0.01y_2y_3 & y_1(0) &= 1.0 \\ \dot{y}_2 &= 400y_1 - 100y_2y_3 - 3000y_2^2 & y_2(0) &= 0.0 \\ \dot{y}_3 &= 30y_2^2 & y_3(0) &= 0.0 \end{aligned}$$

is a stiff problem originating from chemical reaction kinetics (cf. Section 3.6). After the initial transient, the problem is dominated by one large negative eigenvalue, and for moderate tolerances the stepsize gets restricted by stability. The problem is studied in the time interval $0 \le t \le 0.5$.

Problem 3. The differential equation [Hairer and Wanner, 1991, p. 26]

$$\begin{aligned} \dot{y}_1 &= -2000\left(1 + y_1\cos t + y_2\sin t\right) & y_1(0) &= 1 \\ \dot{y}_2 &= -2000\left(1 - y_1\sin t + y_2\cos t\right) & y_2(0) &= 0 \end{aligned}$$

has eigenvalues that move slowly on a large circle from -2000 to $\pm 2000i$, as t changes from 0 to $\pi/2$. For moderate tolerances stability will restrict the stepsize, and integrating the problem gives an idea of the average behavior of the error control loop when it operates on $\partial\mathcal{S}$.

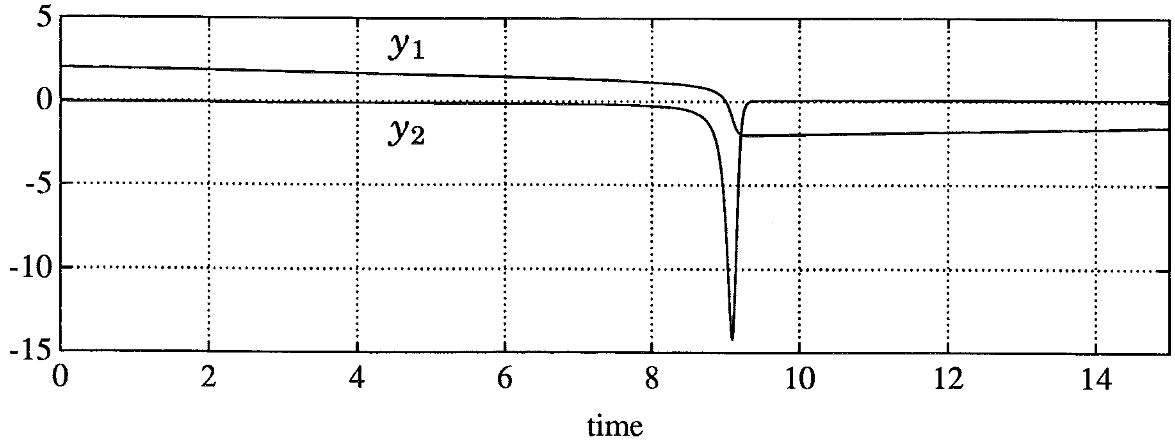


Figure 4.8 The solution to the van der Pol oscillator with $\sigma = 10$ (test problem 1).

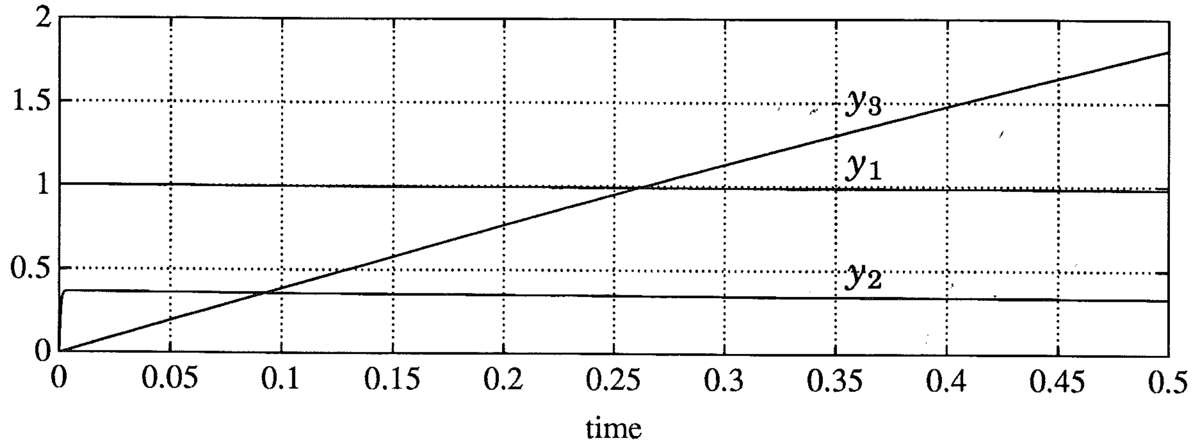


Figure 4.9 The solution to the Robertson problem (test problem 2).

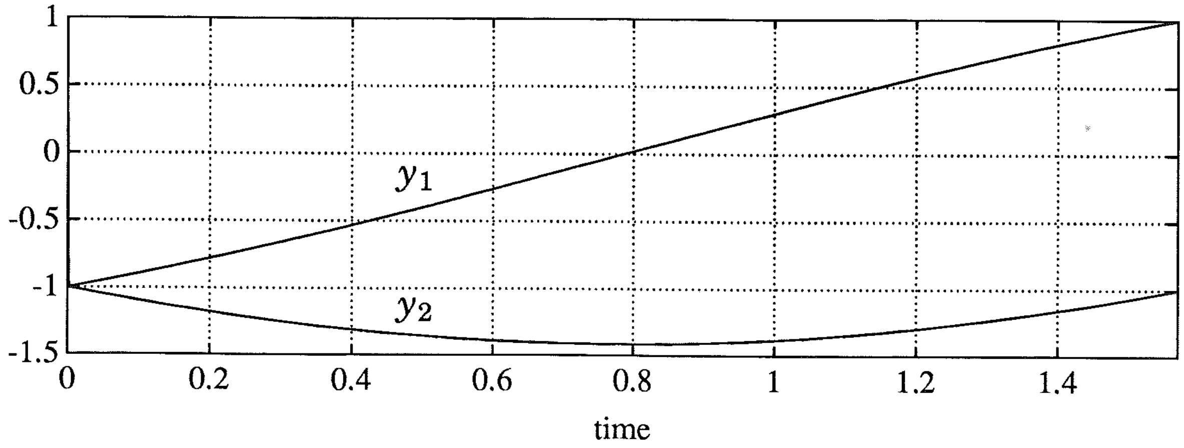


Figure 4.10 The solution to the test problem with varying complex eigenvalues (test problem 3).

When integrating the three problems a mixed absolute-relative 2-norm (2.14) with $\eta = 10^{-4}$ and $\tilde{y} = |y|$ was used. The error control was run in EPS mode.

The error controller included various safety nets:

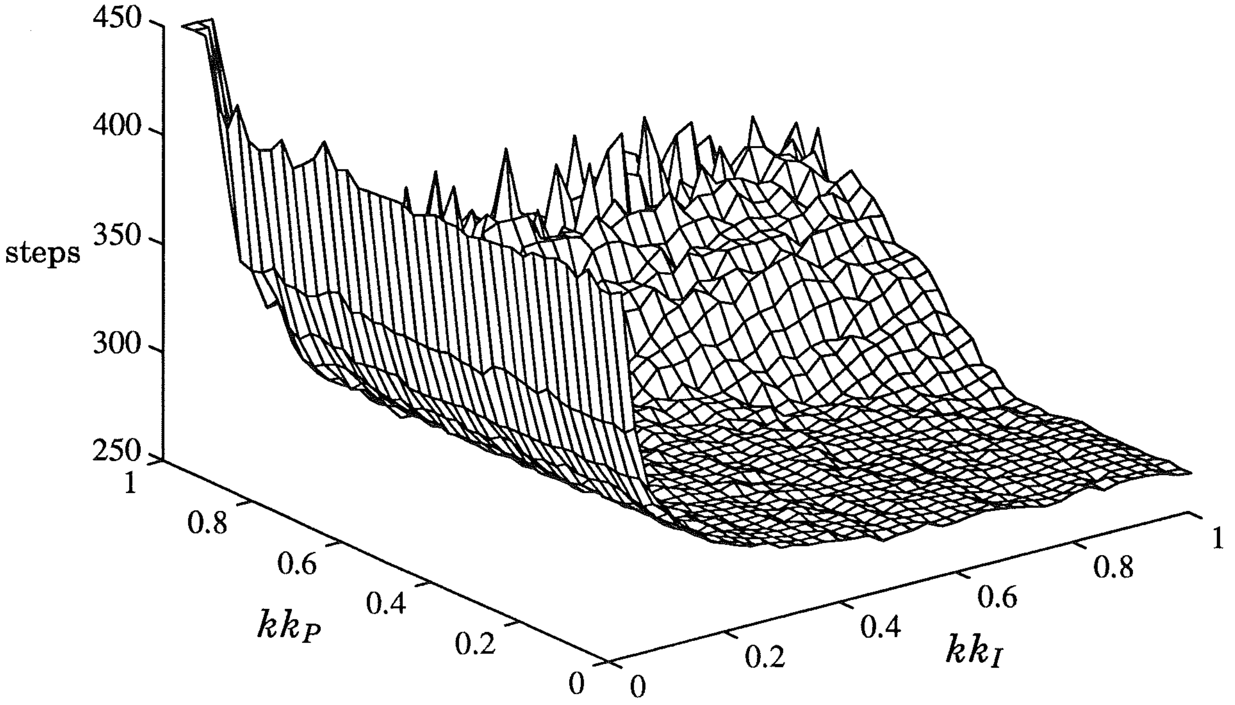
- The stepsize is not allowed to grow more than a factor $10^{1/k}$ in one step. This corresponds to an error increase by a factor of 10.
- The factors $(\varepsilon/r_{n+1})^{k_I}$ and $(r_n/r_{n+1})^{k_P}$ in (4.8) are restricted to the interval [0.01, 100].

Controller Parameters

First we present the results from some numerical integrations that aim at confirming the parameter choice $kk_I = 0.3$, $kk_P = 0.4$, cf. (4.10), that was made for the PI controller. It is impossible to present all the individual results, and instead we concentrate on some typical situations. In all the simulations the set-point ε was chosen as a factor 0.8 of the rejection level *tol*, i. e. $\varepsilon/tol = 0.8$.

Integrating Problem 1 with DOPRI(4)5 and $tol = 10^{-6}$ corresponds to a case where the stepsize almost never is restricted by stability. Hence, the standard controller (2.22) is expected to work reasonably well. This is confirmed by the plots in Figure 4.11. The standard controller ($kk_I = 1$, $kk_P = 0$) results in a minimum number of integration steps, but as can be seen the exact values of kk_I and kk_P are not too critical. In this case, the performance penalty of having $kk_I = 0.3$, $kk_P = 0.4$ is an approximate 5 % increase of the number of steps. For certain parameter values there is a very large increase in the number of steps. These values lie outside or close to the border of the stability region depicted in Figure 4.5. Although the error control loop gets unstable, the logic for rejected steps saves the situation, and it is still possible to proceed (although inefficiently) with the integration.

Problem 2 is dominated by a large negative eigenvalue and for moderate tolerances the stepsize soon reaches the value h_s. If the integration method has $\beta_1 > 0$ (cf. Figure 4.2) the error control loop ends up unstable with the standard controller. The larger β_1 the more severe the instability gets. To demonstrate the effect DOPRI4(5) has been run without local extrapolation at $tol = 10^{-6}$. This method has a large β_1 value ($\beta_1 = 0.47$), cf. Table A.1. From Figure 4.12 it is obvious that a proportional term in the controller substantially reduces the total number of integration steps. Again we note that the exact values for kk_I and



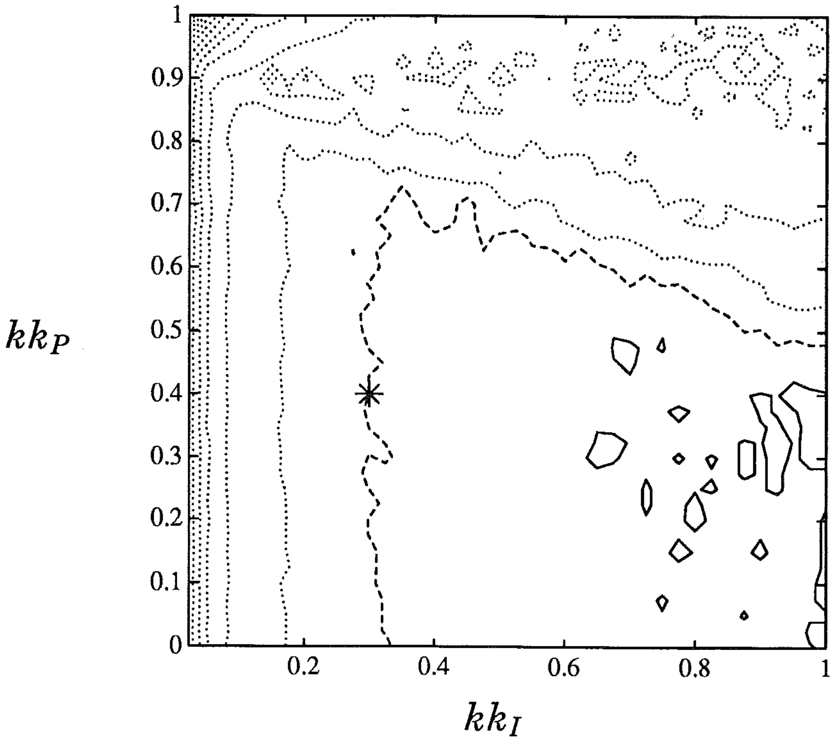


Figure 4.11 Total number of integration steps for Problem 1 as a function of kk_I and kk_P. The lower plot depicts the level curves for the surface in the upper plot. The full line represents the parameter values that give less than 1 % increase in number of steps compared to the minimum. The dashed line represents an increase by 5 %, while the 10 %, 15 %, 20 %, and so on, are plotted with dotted lines. Problem 1 corresponds to a case where the stepsize-error relation is rather well described by the asymptotic model (3.12). The standard controller (kk_I = 1, kk_P = 0) performs very well, but as can be seen the exact values of kk_I and kk_P are not too critical. The star marks the parameter choice $kk_I = 0.3$, $kk_P = 0.4$, cf. (4.10).

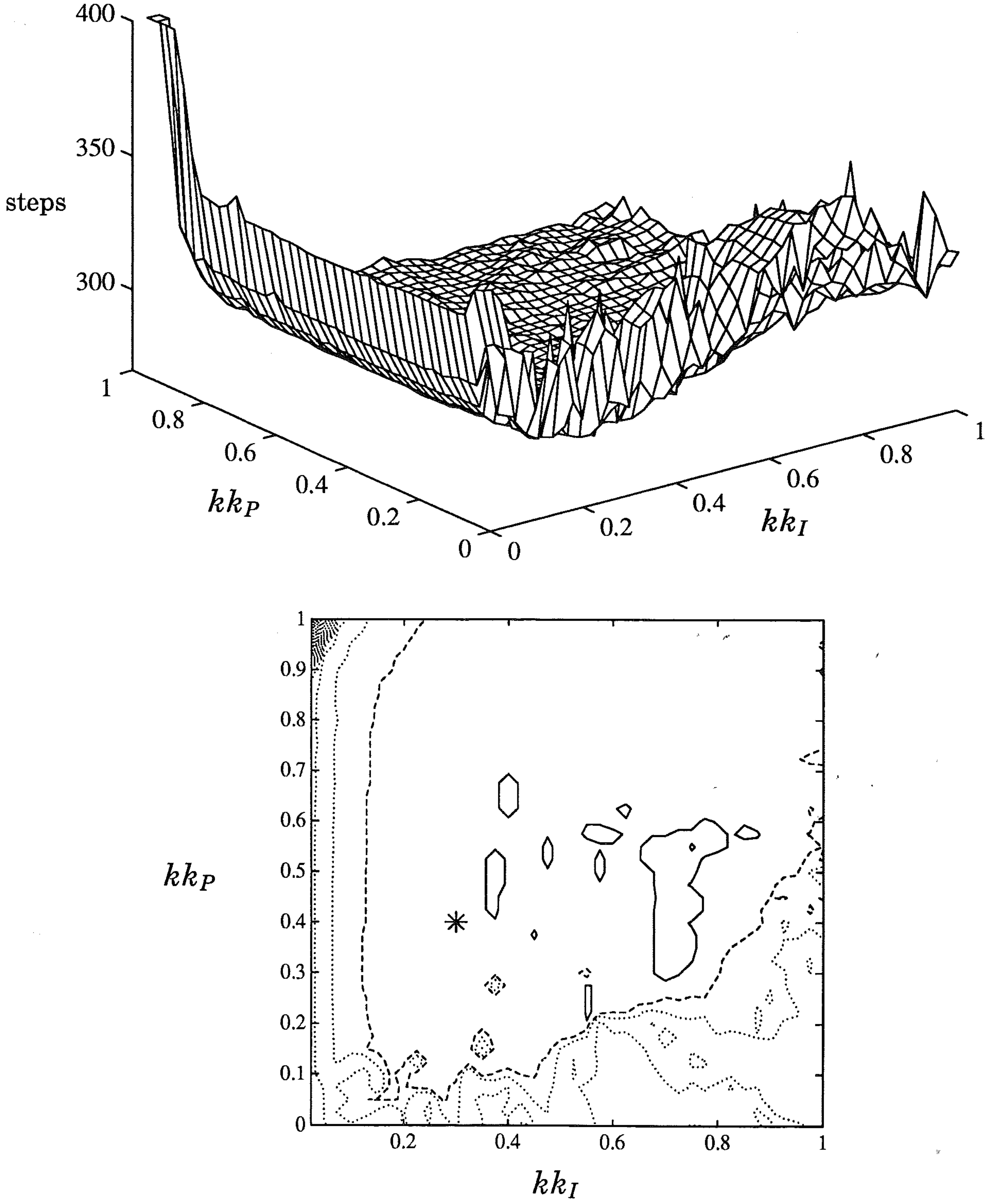


Figure 4.12 Total number of integration steps for Problem 2 as a function of kk_I and kk_P. The lower plot depicts the level curves for the surface in the upper plot. The full line represents the parameter values that give less than 1 % increase in number of steps compared to the minimum. The dashed line represents an increase by 5 %, while the 10 %, 15 %, 20 %, and so on, are plotted with dotted lines. Problem 2 corresponds to a case where the stepsize gets restricted by stability. It is quite clear that to get a well behaved error control loop the controller should include a proportional term. The star marks the parameter choice $kk_I = 0.3$, $kk_P = 0.4$, cf. (4.10).

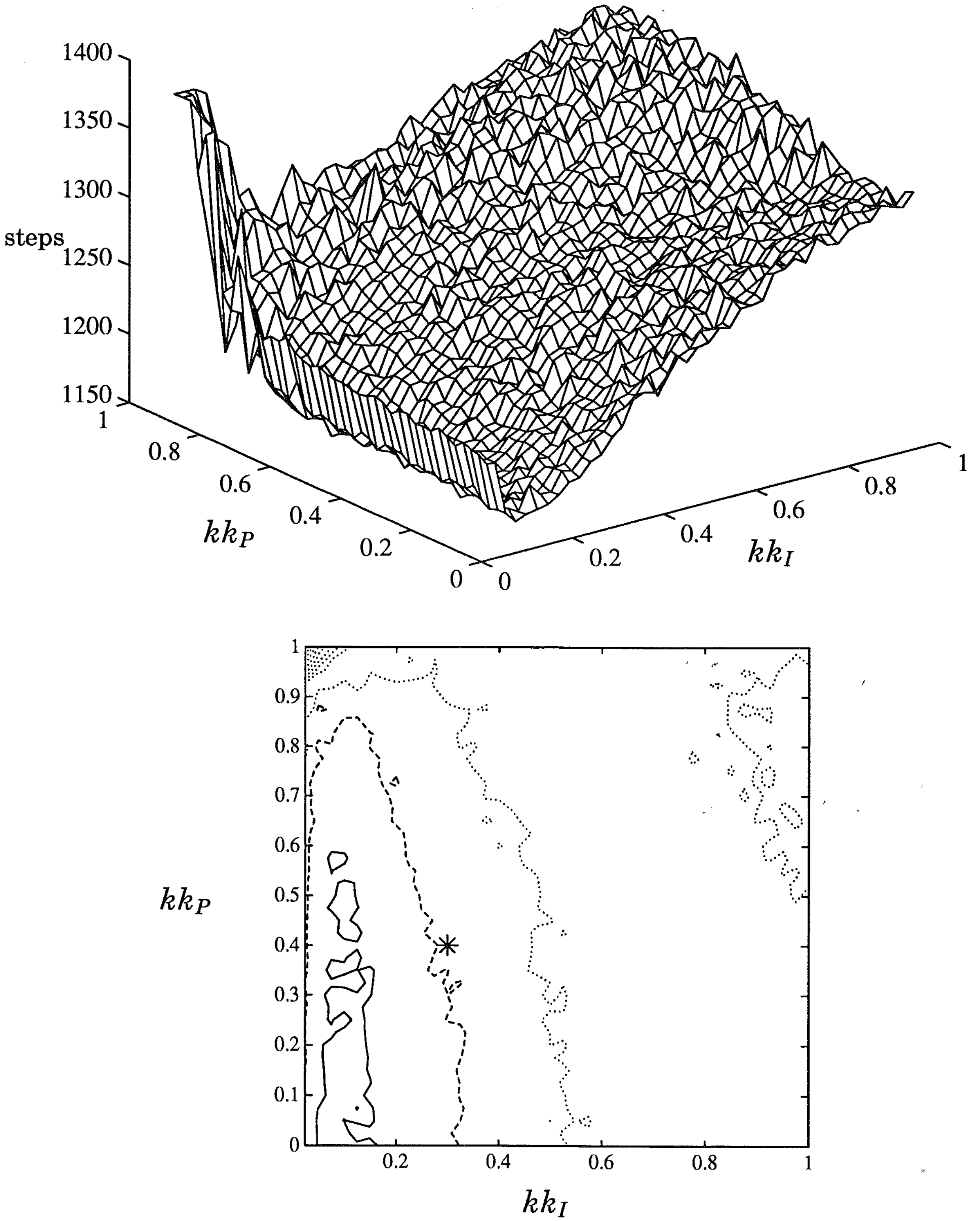


Figure 4.13 Total number of integration steps for Problem 3 as a function of kk_I and kk_P. The lower plot depicts the level curves for the surface in the upper plot. The full line represents the parameter values that give less than 1 % increase in number of steps compared to the minimum. The dashed line represents an increase by 5 %, while the 10 %, 15 %, 20 %, and so on, are plotted with dotted lines. Problem 3 corresponds to a case where the stepsize gets restricted by stability. The complex eigenvalues of the Jacobian in combination with the mixed absolute-relative error norm cause a rather strong perturbation signal. This situation seems to favor a controller that uses strong averaging, i. e. small kk_I and kk_P. The star marks the parameter choice $kk_I = 0.3$, $kk_P = 0.4$, cf. (4.10).

kk_P are not too critical.

The result from the integration of Problem 3 is more difficult to interpret. Figure 4.13 depicts the situation for DOPRI(4)5 and $tol = 10^{-6}$. For this tolerance the stepsize gets restricted by stability. The eigenvalues of the Jacobian in Problem 3 moves on a large circle from -2000 to $\pm 2000i$. The forcing function is smooth and the complex eigenvalues will cause the numerical solution to "oscillate" around the slowly varying stationary solution. Through the mixed absolute-relative error norm this oscillation results in a perturbation causing rather large variations in the error estimate and the stepsize. The resulting number of rejected steps is quite high. The stepsize-error relation is governed by an underlying dynamics corresponding to (3.22), but due to the perturbation from the nonaligned norm this dynamics does not always dominate. The results in Figure 4.13 advocates a controller with small values on kk_I and kk_P. This corresponds to a slow controller which averages over many previous steps before deciding on the current stepsize. With the perturbation that is present this may be a rather wise thing to do.

The results shown in Figures 4.11–4.13 are typical. There is not one single parameter value that gives optimum performance for all three cases. The standard controller is performing well in some cases, but very poorly in other. The choice $kk_I = 0.3$, $kk_P = 0.4$ is a compromise resulting in a controller with reasonable performance for all cases.

The Choice of Set-Point

The relation between the error controller set-point ε and the rejection level tol is an important choice that affects the efficiency of the integration method. If ε is chosen too close to tol there will be many rejected steps, while a value that is much smaller than tol results in many extra integration steps. In order to investigate the effect of the choice of ε a large set of tests were done, combining different integration methods and differential equations with different parameter settings. We will present some typical results.

Problem 1 was integrated for different values of ε/tol using DOPRI(4)5 and $tol = 10^{-6}$. The result is shown in Figure 4.14, and corresponds well with the behavior we expect. It demonstrates that for normal problems the PI controller requires a few more integration steps. As ε/tol approaches 1 the predictive restart strategy improves the efficiency of the integration. Regarding the choice of ε/tol Figure 4.14 seems to indicate $\varepsilon/tol \approx 0.8$. The minimum is, however, rather broad, and the exact value is probably not too critical.

The situation is somewhat different for Problem 2. Figure 4.15 depicts the result from applying DOPRI(4)5 and *tol* $= 10^{-6}$. The standard controller results in an unstable error control loop, increasing the number of required integration steps. The PI controller stabilizes the system and reduces the number of integration steps. Its behavior agrees completely with what was seen in Figure 4.14.

The integrations of Problem 2 were redone but instead using DOPRI(7)8 and *tol* $= 10^{-8}$. This integration method belongs to the ones where $P(z)$ gets negative when z approaches the stability boundary. For this type of methods we get strong oscillations, due to the interaction between the error norm and the stepsize error dynamics, at the transition between the models (3.12) and (3.22). In this specific example the PI controller behaves worse than the standard controller, cf. Figure 4.16.

Figure 4.17 shows the result from integrating Problem 3 with DOPRI(4)5 and *tol* $= 10^{-6}$. As for the case in Figure 4.16 these results advocate a lower value for ε/tol. The reason is that the oscillation in the control loop makes it beneficial to have the set-point farther away from the rejection level. In the case in Figure 4.17 we do not, however, have the problem with a negative value on $P(h\lambda)$, and in spite of the oscillations caused by the nonaligned norm, the model (3.22) captures essential properties of the stepsize-error relation. The PI controller therefore behaves better than the standard controller.

When combining the results from all the integrations it seems reasonable to have $\varepsilon/tol = 0.8$.

Smoothness of the Stepsize and Error Sequence

The numerical examples so far have been biased towards efficiency. By providing a stable error control loop also when the stepsize reaches h_s the PI controller reduces the number of rejected steps and thus increases efficiency. The price is a slight increase in the number of integration steps needed for the standard case (3.12).

Apart from decreasing the number of rejected steps the PI controller also provides a smoother error and stepsize sequence for (3.22). This was demonstrated in Example 1.1, an example we now return to.

EXAMPLE 4.2—Control system example

The signal in Figure 1.1 comes from a simulation of a small control system consisting of a continuous-time PID-controller [Åström and Häg-

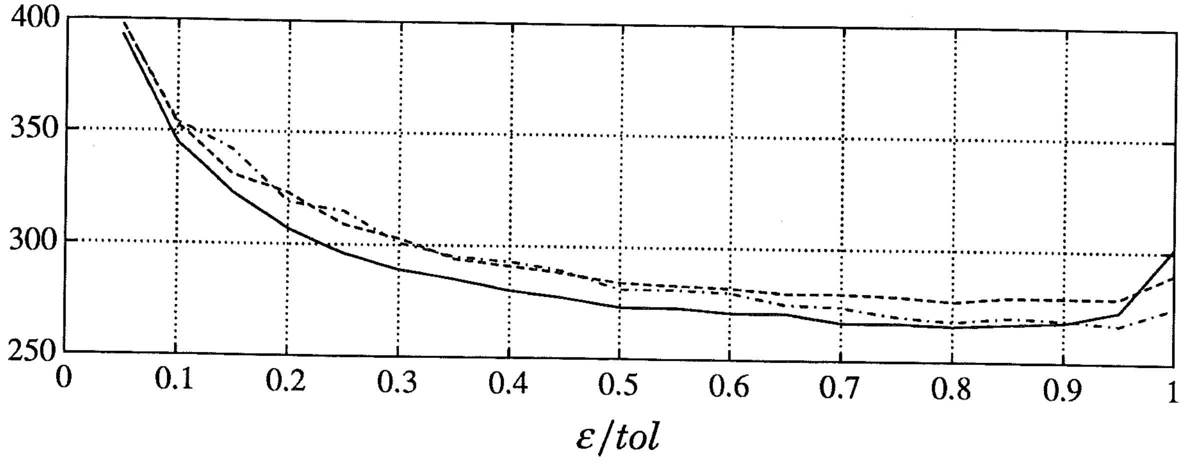


Figure 4.14 Total number of steps as a function of ε/tol, when integrating Problem 1 using DOPRI(4)5 and $tol = 10^{-6}$. The full line corresponds to the standard controller while the dashed line is the result for the PI controller. The dash-dotted line depicts the result when using the PI controller with the predicting restart strategy. This restart strategy is most effective when ε/tol approaches 1, since the set-point then is close to the rejection level. The PI controller is slightly slower than the standard controller and thus requires a few more steps to integrate the problem. For a minimum amount of integration steps $\varepsilon/tol \approx 0.8$.

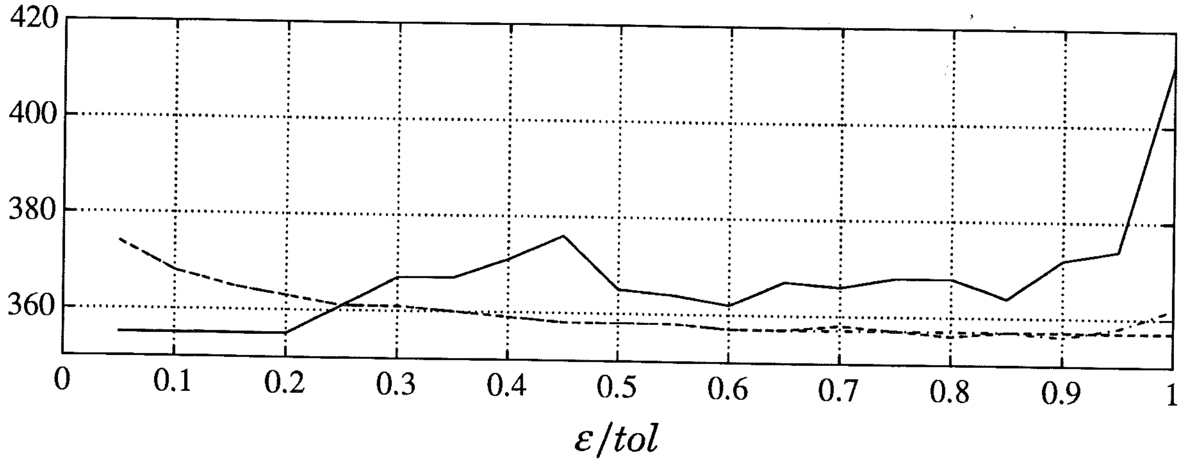


Figure 4.15 Total number of steps as function of ε/tol, when integrating Problem 2 using DOPRI(4)5 and $tol = 10^{-6}$. The full line corresponds to the standard controller while the dashed line is the result for the PI controller. The dash-dotted line depicts the result when using the PI controller with the predicting restart strategy. The standard controller results in an unstable closed loop system, which increases the number of steps needed to finish the integration. The PI controller is more effective, and results in a minimum number of steps for $\varepsilon/tol \approx 0.8$.

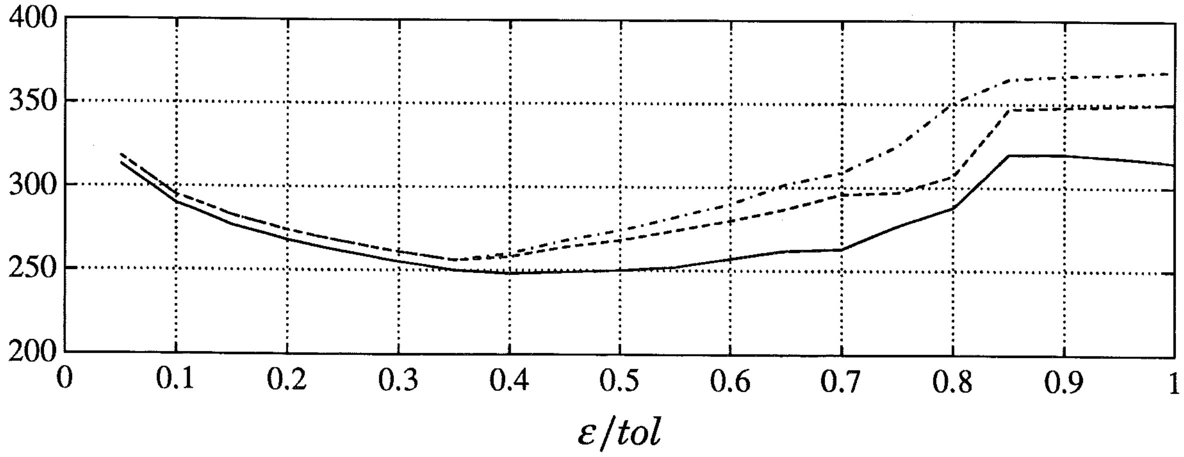


Figure 4.16 The same situation as in Figure 4.14, but using DOPRI(7)8 and *tol* = 10^{-8}. This method has $P(h_s\lambda) = -1$ and the tolerance level is chosen to provoke the situation with the perturbation due to the error norm described in connection with the identification for RKF4(5) in Section 3.6. In this case the standard controller is better than the PI, and the predicting restart strategy makes things even worse. The situation must, however, be considered exceptional.

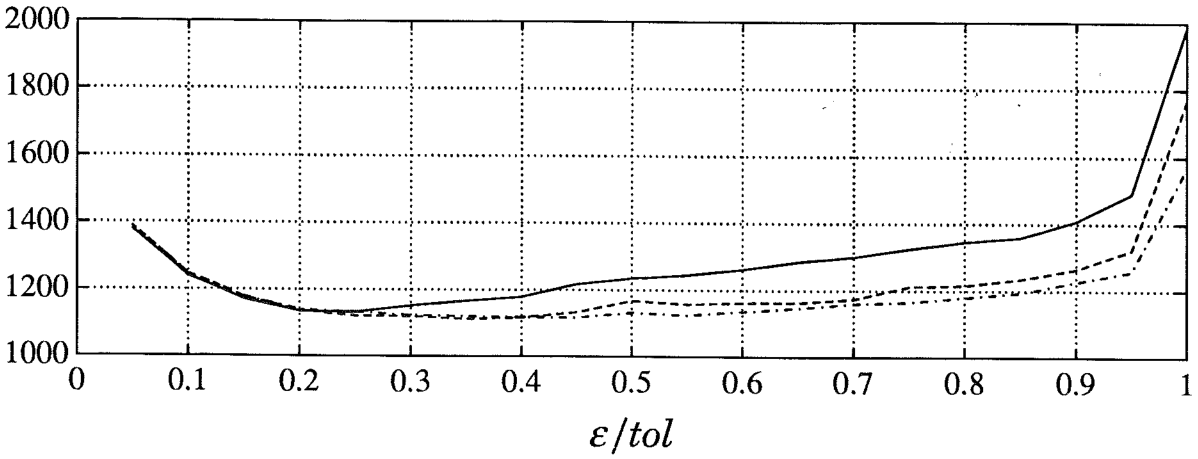


Figure 4.17 Total number of steps as a function of ε/tol, when integrating Problem 3 using DOPRI(4)5 and *tol* = 10^{-6}. Although the nonaligned norm makes the stepsize-error relation deviate from the boundary model (3.22), the PI controller (dashed line) still behaves better than the standard controller (full line). The predicting restart strategy improves the situation slightly. Due to the oscillations it would be beneficial to have ε/tol smaller than the normal choice 0.8.

glund, 1988; Franklin *et al.*, 1986; Hairer and Wanner, 1991] and a fourth order process. The variable y is the output from the process and y_{PID} (which is the variable shown in Figure 1.1) is the output from the controller (and thus the input to the process). The system is described by

$$\begin{aligned} y &= \frac{1}{(\mathcal{D}+1)^4}\, y_{\mathrm{PID}} && \text{(process)} \\ y_{\mathrm{PID}} &= k\left(y_r - y + \frac{1}{\mathcal{D}T_i}(y_r - y) - \frac{\mathcal{D}T_d}{\mathcal{D}T_d/N+1}\, y\right) && \text{(controller)} \\ y_r &= 1 && \text{(reference signal)} \end{aligned}$$

with $\mathcal{D}$ being the differential operator. The parameter values $k = 0.87$, $T_i = 2.7$, $T_d = 0.69$, and $N = 30$ yield a PID-controller well tuned for the process. Figure 4.18 shows some signals from the simulation.

The system has one fast eigenvalue at -40 (corresponding to the filtering of the D-part in the controller) and five well-damped eigenvalues with magnitudes approximately equal to one. The stepsize soon reaches h_s, where the standard stepsize selection rule (2.22) results in an unstable error control loop, and the stepsize and error estimate oscillate, cf. Figure 4.19. The PI controller instead gives a smooth stepsize and error sequence. The two integrations give on average the same global error, but the solution is qualitatively improved with the PI controller, cf. Figures 1.1 and 4.19. □

4.7 The Complete Controller

To summarize this chapter, an outline of the code implementing the new controller is presented in Listing 4.1. We give two versions: one with the predicting restart strategy and one with the standard restart strategy. We suggest using the one with predicting restart for methods where $P(h_s\lambda) = 1$ and the one with standard restart when $P(h_s\lambda) = -1$.

The controller, which should be called after each step in the integration routine, calculates the stepsize to be used in the next step. As before, h is the stepsize, and r the corresponding error estimate. The variables h_{acc} and r_{acc} are used to store the stepsize and the error from the most recent accepted step. Occasionally, the error estimator may produce an unusually small (or large) value, thus advocating a very

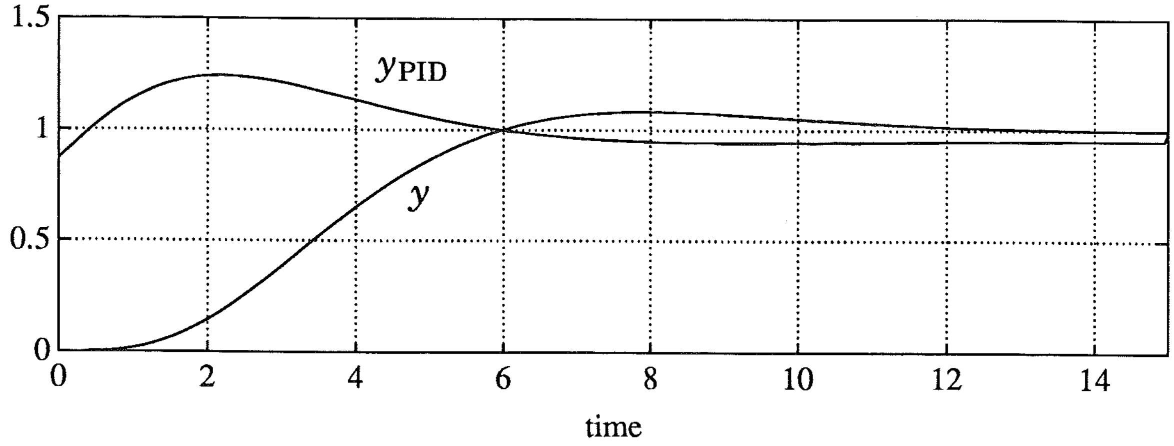


Figure 4.18 The control signal y_{PID} and the process output y from a simulation of the control system in Example 4.2.

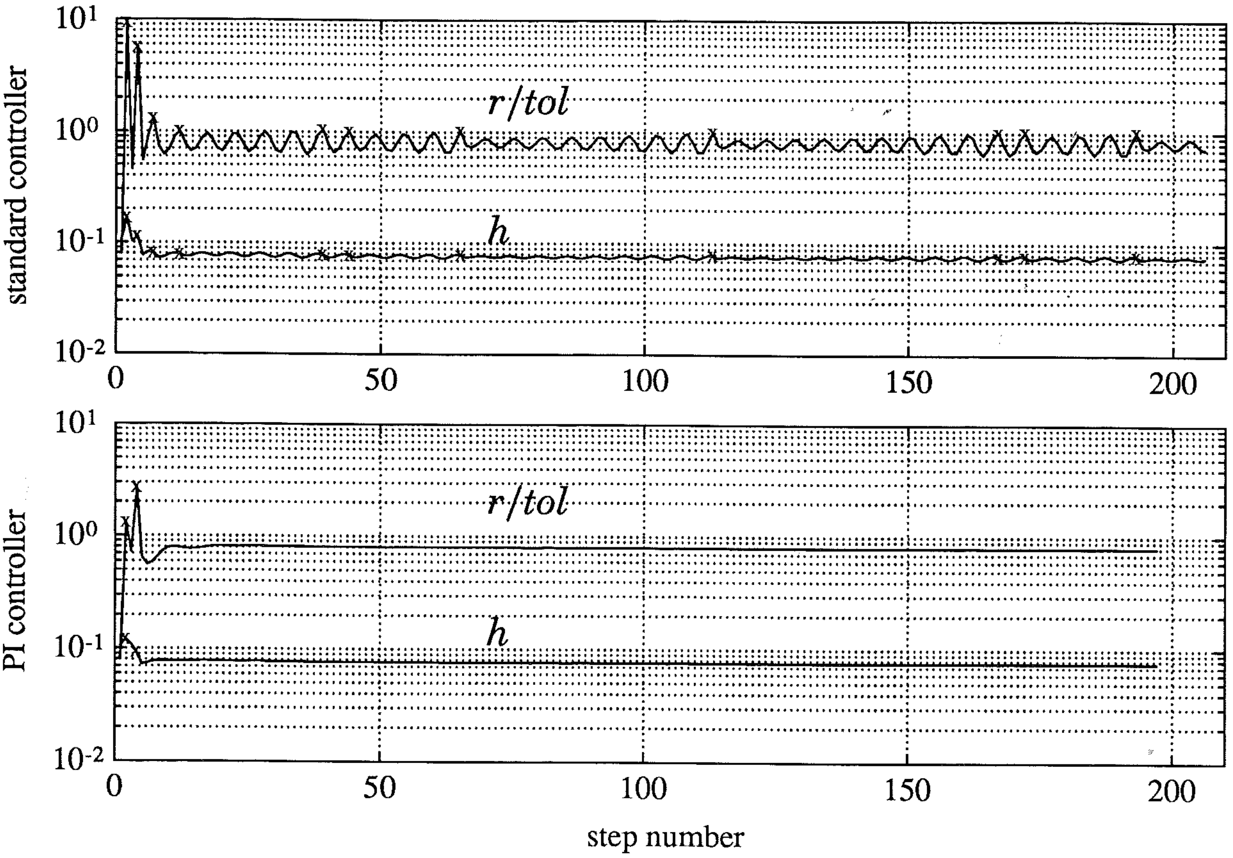


Figure 4.19 The error estimate and the stepsize sequence resulting when simulating the control system in Example 4.2. The upper plot clearly demonstrates how the standard stepsize selection rule leads to an unstable error control loop. Rejected steps are indicated with 'x'. By instead using a PI controller ($kk_I = 0.3$, $kk_P = 0.4$) for choosing the stepsize one obtains a smooth error and stepsize sequence.

```
if cur_step_accept then
    if prev_step_reject then
        $h := h \cdot \frac{h}{h_{\mathrm{acc}}}$
    endif
    $h := \left(\frac{\varepsilon}{r}\right)^{k_I} \left(\frac{r_{\mathrm{acc}}}{r}\right)^{k_P} h$
    $h_{\mathrm{acc}} := h$
    $r_{\mathrm{acc}} := r$
else
    $h := \left(\frac{\varepsilon}{r}\right)^{1/k} h$
endif
```

```
if cur_step_accept then
    $h := \left(\frac{\varepsilon}{r}\right)^{k_I} \left(\frac{r_{\mathrm{acc}}}{r}\right)^{k_P} h$
    $r_{\mathrm{acc}} := r$
else
    $h := \left(\frac{\varepsilon}{r}\right)^{1/k} h$
endif
```

Listing 4.1 An outline of the code needed to implement the PI controller. The listing to the left also includes the predicting restart strategy after rejected steps.

large stepsize change. For robustness the controller should (as usual) include some limitation on such large changes. Also, it is important to avoid overflow or underflow in expressions as r_{acc}/r.

The parameter choice

$$k_I = \frac{0.3}{k}, \qquad k_P = \frac{0.4}{k}, \qquad \varepsilon = 0.8\,tol,$$

gives an error controller that works well with many explicit Runge-Kutta methods and most types of differential equations.

Explicit Runge-Kutta Methods that are Difficult to Control

In many situations, both in the control design as well as when trying to model the stepsize-error relation, special consideration had to be taken for integration methods that have $P(h\lambda)$ negative at the intersection of the negative real axis and $\partial\mathcal{S}$. This property leads to a stepsize-error relation that is more complicated, which in turn makes the integration method more difficult to control. From a control point of view a negative $P(h_s\lambda)$ is a property one should try to avoid when constructing integration methods.

5

Control of Implicit Runge-Kutta Methods

It is considerably more complicated to design a controller for an implicit integration method than an explicit. The key problem is to find a controller that combines good error control with an efficient strategy for the equation solver. What is efficient in the equation solver depends strongly on the differential equation, e. g. number of states, nonlinearity, rapidly/slowly varying solution. This makes the class of implicit Runge-Kutta methods less homogeneous than the class of explicit ones, and for efficiency the tuning of the controller will have to depend on the type of differential equation.

The structure of the controller to be presented here is similar to the one of traditional controllers (cf. Section 2.5), and may be viewed as in Figure 5.1. The error controller keeps the integration error at a prescribed level by adjusting the stepsize. The error depends also on the quality of the stage values Y, but by keeping the iteration error small, i. e. below τ, the error estimate r will be dominated by the error contribution that may be affected by adjusting the stepsize h. Our new error controller is based on the improved asymptotic stepsize-error relation derived in Chapter 3, and achieves better error control by predicting changes in φ.

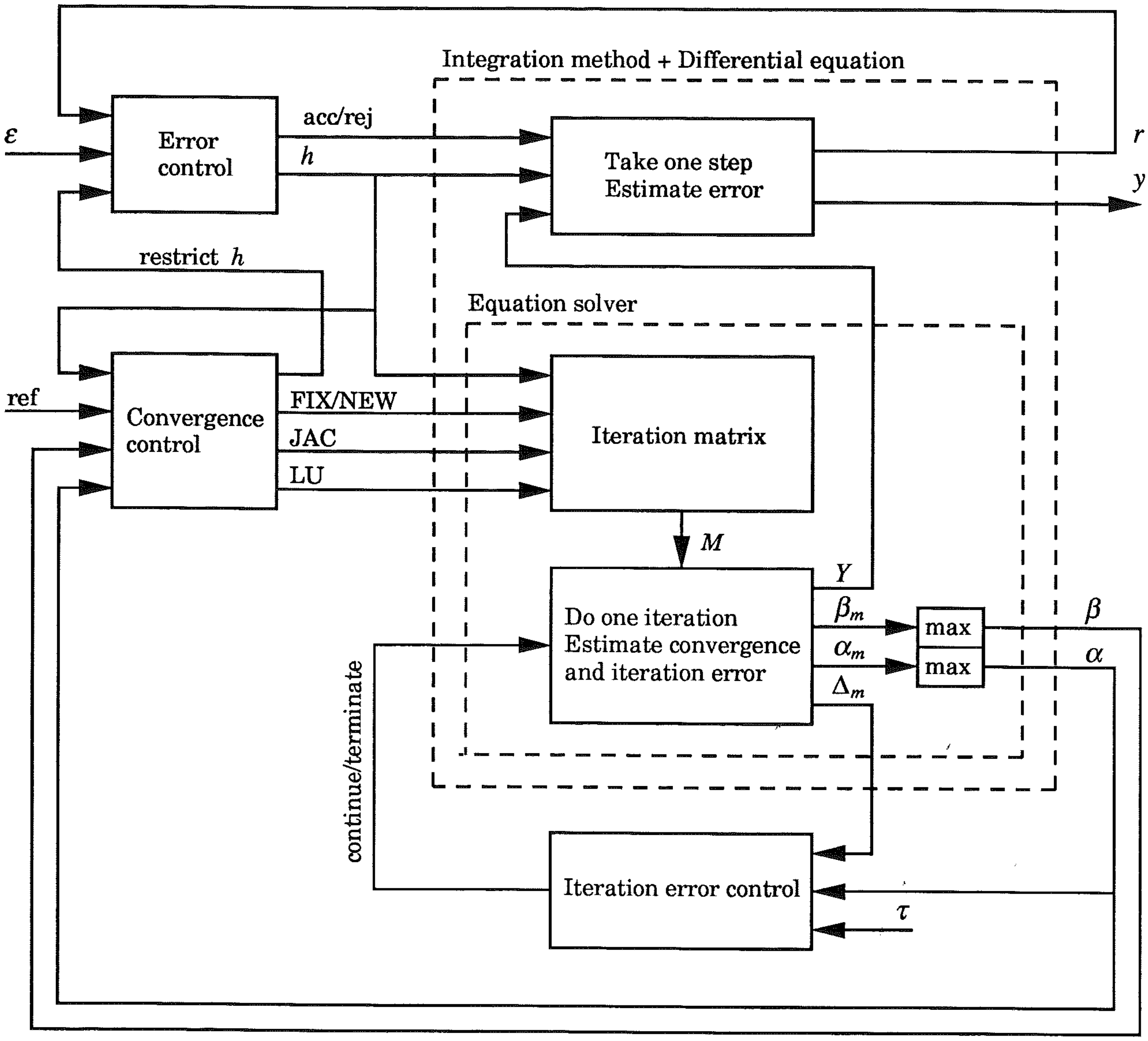


Figure 5.1 A block diagram view of the control strategy in an implicit Runge-Kutta method.

The convergence of the iteration in the equation solver is secured by the convergence controller. This controller is quite different from the error controller in that it makes decisions that are binary in nature instead of adjusting some continuous control signal. The convergence is affected by the choice of stepsize and traditional convergence controllers often prohibit small stepsize changes. The smoothness of the error control is improved by removing some of these restrictions. We present a convergence control strategy where small stepsize changes do not lead to excessive operations on the iteration matrix.

The organization of this chapter is similar to the block structure of Figure 5.1. We will start by discussing the error control loop. Then follows a short section on the control of the iteration error. Some of the signals from this loop are used as measurements for the convergence controller, and we examine their quality before continuing with the convergence control loop. Finally, some issues regarding the overall control strategy are discussed.

5.1 A Predictive Error Controller

In an implicit integration method there are many input signals, cf. Figure 5.1, all directly or indirectly affecting the error estimate r. By making the iteration error small, i. e. τ small compared to the error set-point ε (say $\tau < 0.01\,\varepsilon$), the dominating dependence will be the one between the stepsize h and the error estimate r, and r may be controlled by adjusting h appropriately.

From Examples 3.1 and 3.4 we know that the stepsize-error model can be improved by adding a model for the changes in φ. In the explicit case this could, however, not be exploited (cf. Section 4.3). The PI controller that was needed to handle the case when numerical stability restricts the stepsize, i.e. $h = h_s$, is quite different from a controller that predicts changes in φ. Many implicit methods are designed to be L-stable and the problems related to stability restricting the stepsize need not be considered when designing the error controller. This is true also for most methods possessing the weaker property A-stability. An exception is when ∞ belongs to ∂S, i. e. $|P(\infty)| = 1$, but this must here, in this general context, be regarded as a poorly designed implicit method.

Being able to predict φ_n accurately is an essential part of improved error control. We will start by designing an observer that, based on old data and the model (3.10), predicts the value of φ_n. From the prediction it is straightforward to calculate the new stepsize, and we will arrive at a stepsize selection rule on the form

$$h_{n+1} = \frac{h_n}{h_{n-1}} \left(\frac{\varepsilon}{r_{n+1}} \right)^{k_2/k} \left(\frac{r_n}{r_{n+1}} \right)^{k_1/k} h_n, \tag{5.1}$$

where k_1 and k_2 are parameters related to the observer.

The selection rule (5.1) uses data from previous steps to decide upon the new stepsize. At a rejected step more information is available, and (5.1) will be augmented with some logic reflecting this.

An Observer for the Disturbance

In order to use (3.10) to predict the disturbance, $\log \varphi_{n-1}$ and $\nabla \log \varphi_{n-1}$ have to be estimated from past data. Should their value be based on the last point only, or should some type of averaging be used? One way to approach this problem is to use an observer (for a discussion about observers see e.g. [Åström and Wittenmark, 1990, Chapter 9.3]). Introduce

$$x(n) = \begin{pmatrix} x_1(n) & x_2(n) \end{pmatrix}^T = \begin{pmatrix} \log \varphi_n & \nabla \log \varphi_n \end{pmatrix}^T,$$

and assume $\log \varphi$ linearly varying, i.e. $\nabla \log \varphi$ constant. Then

$$x(n) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(n-1).$$

Let $\hat{x}(n|n)$ denote the estimate of $x(n)$ given data up to n. The observer that estimates $\hat{x}(n|n)$ based on $\log \varphi_n$ ($\log \varphi_n$ can be calculated from r_{n+1} and h_n) then reads

$$\begin{aligned} \hat{x}(n|n) &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \hat{x}(n-1|n-1) \\ &\quad + \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} \left(\log \varphi_n - \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \hat{x}(n-1|n-1) \right) \qquad (5.2) \\ &= \begin{pmatrix} 1-k_1 & 1-k_1 \\ -k_2 & 1-k_2 \end{pmatrix} \hat{x}(n-1|n-1) + \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} \log \varphi_n, \end{aligned}$$

where k_1 and k_2 are parameters determining the dynamics of the observer. The observer can, in this case, be interpreted as a geometrically weighted average of old measured values. The design parameters k_1 and k_2 determine how much the old measurements should influence the estimate. Taking $k_1 = k_2 = 1$ bases the estimation on the most recent measurements only.

Using the forward shift operator q, (5.2) can be written

$$\hat{x}(n-1|n-1) = \begin{pmatrix} \dfrac{k_1 q^2 + (k_2 - k_1)q}{q^2 + (-2 + k_1 + k_2)q + 1 - k_1} \\ \dfrac{k_2 q^2 - k_2 q}{q^2 + (-2 + k_1 + k_2)q + 1 - k_1} \end{pmatrix} \log \varphi_{n-1},$$

and from $\hat{x}(n-1|n-1)$ we predict $\log \hat{\varphi}_n$ as

$$\begin{aligned}\log \hat{\varphi}_n &= \begin{pmatrix} 1 & 1 \end{pmatrix} \hat{x}(n-1|n-1) \\ &= \frac{(k_1+k_2)q^2 - k_1 q}{q^2 + (-2+k_1+k_2)q + 1 - k_1} \log \varphi_{n-1}.\end{aligned} \tag{5.3}$$

A Stepsize Selection Rule Using the Disturbance Prediction

Given a prediction of the disturbance it is natural to choose the stepsize from the "asymptotic" formula, cf. (3.1),

$$\log h_n = k^{-1}(\log \varepsilon - \log \hat{\varphi}_n). \tag{5.4}$$

Using the expression for $\log \hat{\varphi}_n$ (5.3) and the fact that $\log \varepsilon$ is constant,

$$\log h_n = \frac{1}{k} \frac{(k_1+k_2)q^2 - k_1 q}{q^2 + (-2+k_1+k_2)q + 1 - k_1} (\log \varepsilon - \log \varphi_{n-1}). \tag{5.5}$$

Finally, we substitute $\log \varphi_{n-1} = \log r_n - kq^{-1} \log h_n$ in (5.5), and solve for $\log h_n$,

$$\log h_n = \frac{1}{k} \frac{(k_1+k_2)q^2 - k_1 q}{(q-1)^2} (\log \varepsilon - \log r_n). \tag{5.6}$$

As a consequence of assuming $\log \varphi$ linearly varying the controller includes a double integrator. This is called the *internal-model principle*, and says that the disturbance dynamics appears in the controller when controlling a system including an unstable disturbance model [Åström and Wittenmark, 1990, pp. 400–401].

It is worth noting the similarity between (5.6) and the PI controller (4.7) recommended for explicit Runge-Kutta methods. By writing (5.6) as

$$\frac{q-1}{q} \log h_n = \frac{1}{k} \frac{(k_1+k_2)q - k_1}{q-1} (\log \varepsilon - \log r_n),$$

we see that in (5.6) it is the *stepsize change* that is governed by a PI controller and not the stepsize itself as in (4.7). This difference is important, and it would not be possible to use (5.6) when numerical stability restricts the stepsize, i.e. for a process model on the form (3.22).

The controller (5.6) can be rewritten as (5.1), which reveals that the controller forms the new stepsize based on extrapolation of old errors

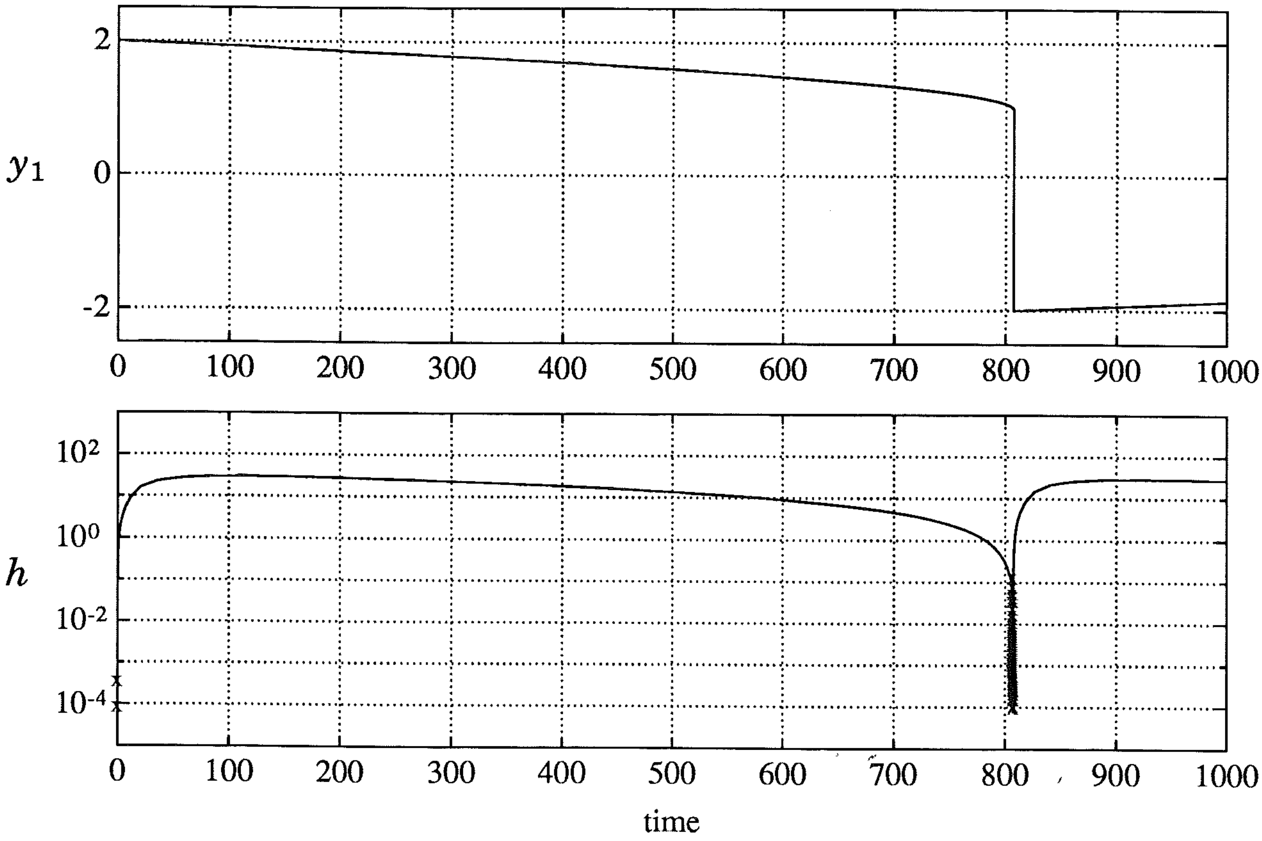


Figure 5.2 The upper plot depicts the solution of the first component y_1 of the stiff van der Pol problem ($\sigma = 1000$) in Example 5.1. The problem was solved using HW-SDIRK(3)4 and standard stepsize control (2.22). The lower diagram shows the required stepsize sequence. During the fast state transition there are many rejected steps (indicated with crosses).

and stepsizes. In that respect it is similar to the controllers derived in [Watts, 1984] and [Zonneveld, 1964].

EXAMPLE 5.1—Predictive stepsize control
We return to the van der Pol oscillator in Example 3.1, and make the problem stiff by choosing $\sigma = 1000$. The solution for this σ-value has the same general form as the one for smaller σ (cf. Figure 4.8), but the peaks in y_2 are more pronounced and the state transitions are very fast. The problem was solved (cf. Figure 5.2) using HW-SDIRK(3)4, an SDIRK method by Hairer and Wanner (cf. Appendix A or [Hairer and Wanner, 1991, p. 107]). A mixed absolute-relative 2-norm (2.14) with $\eta = 10^{-4}$ and $\tilde{y} = |y|$ was used. The tolerance was set to $tol = 10^{-4}$ and $\varepsilon = 0.8\,tol$. To remove potential effects from the equation solver, modified Newton iterations were used with a new Jacobian at every step. The norm of the iteration error was required to be less than $0.01\,\varepsilon$.

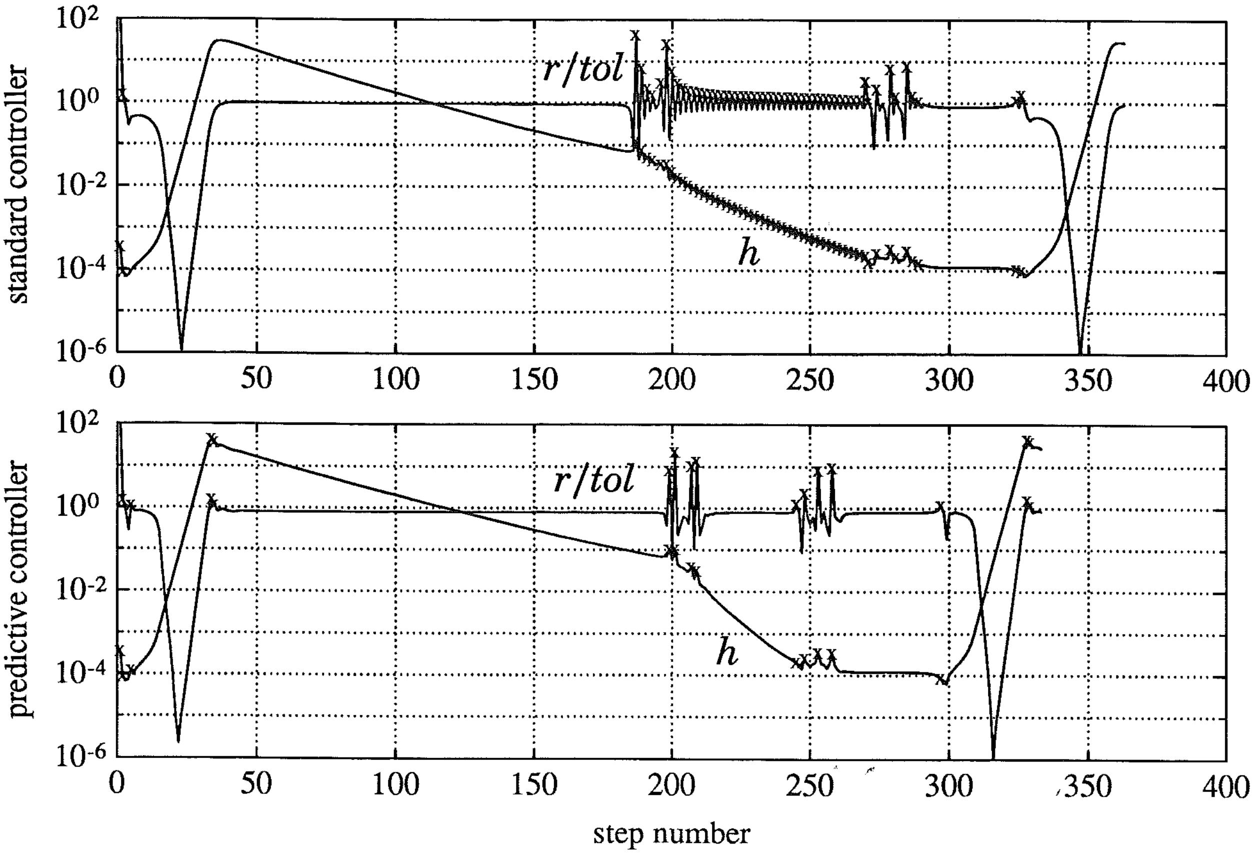


Figure 5.3 Stepsize and error sequences arising when solving the stiff van der Pol oscillator in Example 5.1 using HW-SDIRK(3)4. The upper diagram depicts the situation when using the standard controller (2.22), i. e. the stepsize and error sequence from Figure 5.2, while the lower diagram corresponds to the predictive controller (5.1) augmented with some logic to handle rejected steps. When using the standard controller almost every second step is rejected (indicated with crosses) during the first part of the fast state transition (step 180–280). In contrast, the predictive controller quickly adjusts to the changes in φ and as a result gives superior error control and fewer integration steps.

The standard controller (2.22) works rather well except at the fast state transition (step 180–330 in the upper plot in Figure 5.3). Here the controller fails to track the changes in φ, resulting in many rejected steps. In contrast, the predictive controller ((5.1) with $k_1 = k_2 = 1$), depicted in the lower plot of Figure 5.3, quickly captures the changes in φ and as a result gives superior error control and fewer rejected steps.

During a short period after the initial transient (steps 15–30 in the upper plot of Figure 5.3) and after the fast state transition (steps 340–355) the error drops very low. For robustness the error controller does not allow a stepsize increase larger than $10^{1/k}$, i. e. a change that would increase the error by more than a factor of 10. In the case at hand this limit prevents the controller from increasing the stepsize fast enough to keep $r = \varepsilon$. □

The inability of the standard controller (2.22) to handle fast changes in φ is clearly demonstrated in Example 5.1. The same phenomenon is responsible for the misbehavior in Example 1.2. The problem is resolved by instead using the predictive error controller (5.1).

Rejected Steps

If r_{n+1} gets too large the current step has to be rejected. A new stepsize needs to be determined and the observer should be updated with the information from the rejected step. The predictive controller has two internal states, $\hat{x}_1$ and $\hat{x}_2$, cf. (5.2), which makes the restart more involved than for the standard controller. The situation depends on whether the previous step was rejected or not, and it is beneficial to have different restart strategies for these two cases.

First rejected step. The information from the rejected step makes it possible to calculate φ_n, and as in the explicit case, cf. (4.11), it is natural to use the standard stepsize selection rule to calculate a new stepsize.

Updating the observer state $\hat{x}$ (5.2) is more difficult. One step does not contain enough information to set both components of the observer state vector to their "correct" values, and we have to choose. A likely cause for the rejection is φ changing differently than predicted (cf. Example 3.1 and Figure 3.2), and it is therefore reasonable to use the information to calculate a new value for $\hat{x}_2$ ($\nabla \log \hat{\varphi}$).

The described strategy for rejected steps results in the expression

$$\hat{x}(n|n) = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \hat{x}(n-1|n-1) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \log \varphi_{\mathrm{rej}}, \tag{5.7}$$

where $\log \varphi_{\mathrm{rej}}$ is calculated from the rejected step. After this special update the stepsize is determined by using $\log \varphi_{\mathrm{rej}}$ in (5.4). The strategy is, actually, equivalent to (4.11), but as a side affect the observer states are set to better track the changes in the following steps.

EXAMPLE 5.2—Restart after one rejected step
Again return to the van der Pol oscillator, and specifically the lower plot in Figure 5.3. The predictive controller assumes a constant change in $\log \varphi$, and will as a result have problems when this assumption is not valid. The rejected steps around 200 occur just at the transition from the flat part of the solution to the fast state transition, the steps at 250

correspond to the peak of y_2, and, finally, the step at 300 is an effect of the change between absolute and relative norm when y_2 passes zero. In all these cases there are large changes in $\nabla \log \varphi$. After one or two rejected steps the predictive controller captures the situation and the integration continues successfully. □

Successive rejected steps. It may happen that there are successive rejected steps. In this case the standard asymptotic model motivating (4.11) fails to capture the situation. One reason may be that the excess error was caused by a solution component in the nonasymptotic region, leading to a stepsize-error relation with a lower value of k than expected. The standard stepsize selection rule then results in a long sequence of rejected steps (cf. the example in [Hairer and Wanner, 1991, pp. 122–123]). The efficiency of the restart is greatly improved by forming an estimate $\hat{k}$ (3.13), and using it in the normal stepsize selection rule at rejected steps (4.11). For robustness the value of $\hat{k}$ should be limited to (say) $[0.1, k]$, and the stepsize should not be allowed to decrease by more than a factor of (say) 10.

Using the estimated $\hat{k}$ is a kind of patch that reduces the number of rejected steps, but it does not solve the main problem. Decreasing the stepsize by several orders of magnitude in order to restart leads to very inefficient integration. The conclusion to be drawn is to use integration methods which do not unnecessarily excite the nonasymptotic modes, i. e. stiffly accurate integration methods [Cash, 1979; Kværnø, 1988; Hairer *et al.*, 1989; Hairer and Wanner, 1991].

When not being able to trust the value of k there is no way to calculate φ, and we cannot make a correct update of the observer state $\hat{x}$. We choose a conservative strategy and delay the updating until after the first accepted step. At this stage φ is calculated and the state components are reset as $\hat{x}_1 = \log \varphi$, $\hat{x}_2 = 0$. This corresponds to using the standard stepsize selection rule (2.22) at the first accepted step after successive rejections.

EXAMPLE 5.3—Restart after successive rejected steps

SIMPLE(2)3 (see Appendix A) is an SDIRK where $P(\infty) \neq 0$. It is quite prone to order reduction. Figure 5.4 depicts the stepsize sequence arising when solving the stiff van der Pol oscillator of Example 5.1. Several times the stepsize drops by a factor of 10^4; every time accompanied with a long sequence of rejected steps. Predictive error control, i. e. (5.1) with $k_1 = k_2 = 1$, was used, but the artifacts in Figure 5.4 are similar when

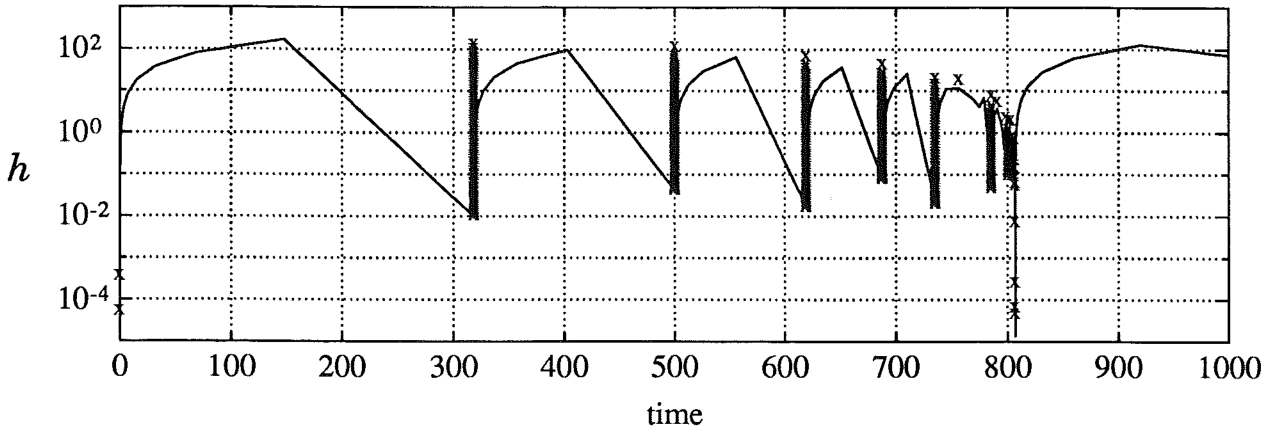


Figure 5.4 Stepsize sequence arising when using SIMPLE(2)3 to solve the stiff van der Pol oscillator in Example 5.1 (compare with Figure 5.2). The integration is very inefficient due to the very large stepsize reductions required to restart the integration method after a rejected step.

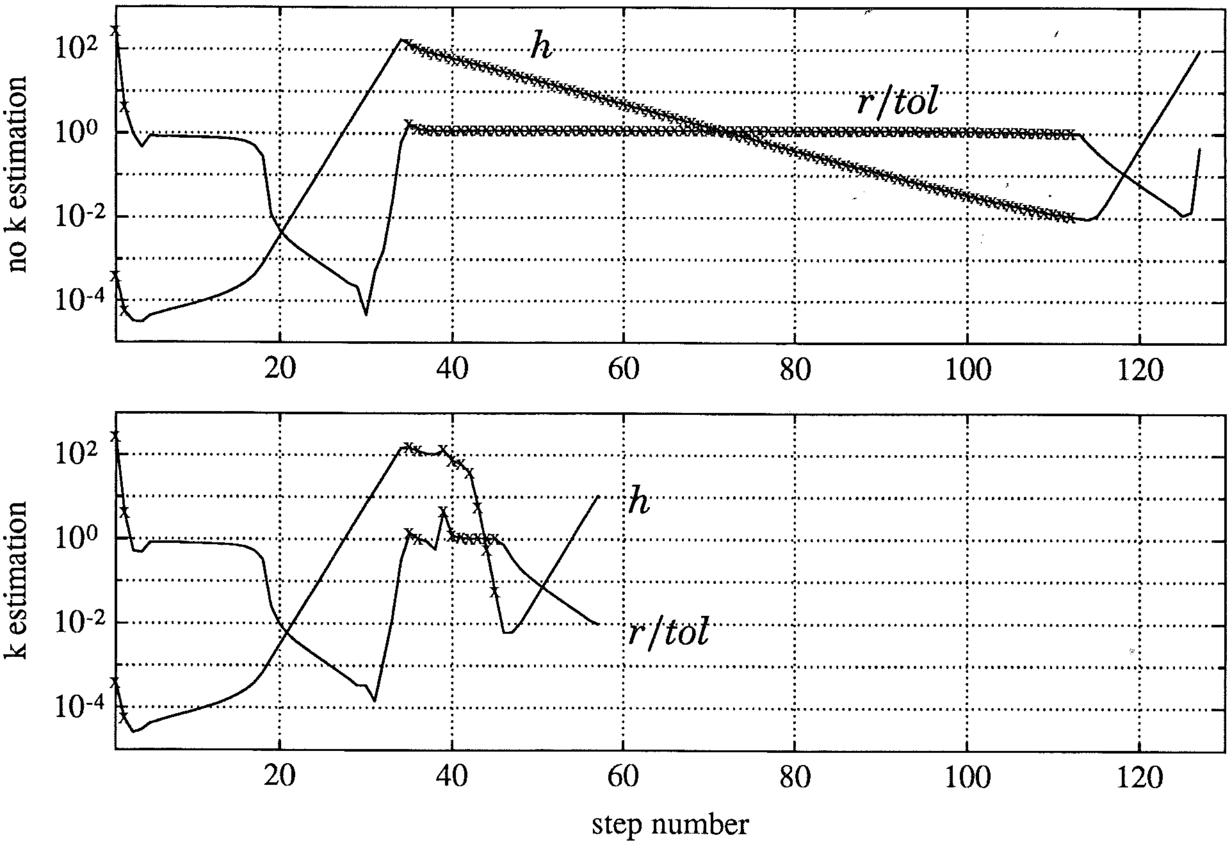


Figure 5.5 The upper plot depicts the stepsize and error sequence from the time interval $t \in [0, 400]$ of Figure 5.4. The stepsize-error relation at the large stepsize drop disobeys the normal asymptotic model (2.21), resulting in many rejected steps before the controller manages to restart the integration. By using estimates of k after successive rejections, the restart strategy can be made more efficient, greatly reducing the number of rejected steps. This is demonstrated in the lower plot.

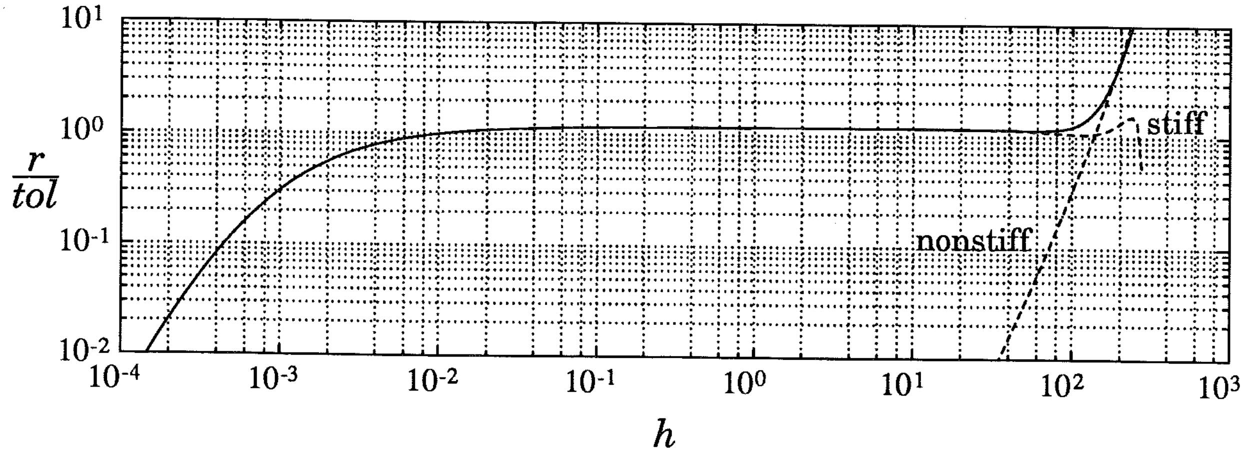


Figure 5.6 The plot depicts the stepsize-error relation at the long sequence of rejected steps at $t \approx 318$ in Figure 5.4. The stepsize may change by almost four orders of magnitude without substantially affecting the integration error [Hairer and Wanner, 1991, pp. 122-123]. The error can be decomposed into two components (dashed lines) corresponding to the eigenvectors of the Jacobian. The component corresponding to the stiff eigenvalue is in the nonasymptotic region of the stepsize-error relation, which explains the difficulty of affecting the error with reasonable stepsize changes.

using the standard error controller (2.22).

The stepsize and error sequences from the first of these drops are depicted in Figure 5.5. Almost 80 rejected steps are experienced before the integration finally succeeds in restarting. As Figure 5.6 clearly demonstrates, the stepsize-error relation disobeys the asymptotic relation (3.12) normally assumed. If using a restart strategy where k is estimated as described above, the integration can be restarted after a few number of rejected steps. It is, however, important to note that the strategy only reduces the number of rejected steps. The stepsize still has to be reduced with the same amount to restart the integration.

During the stiff part of the solution the two eigenvalues of the Jacobian differ considerably in magnitude. At $t \approx 318$ they are $9.2 \cdot 10^{-4}$ and $-2.1 \cdot 10^3$, respectively. The solution mode corresponding to the large eigenvalue is very small and has been moved far out in the nonasymptotic region, cf. Figure 3.9. Its error contribution is small and the stepsize error relation is normally dominated by the mode corresponding to the small eigenvalue. Through a complicated interaction between the differential equation and the integration method the stiff mode gets excited and its error contribution is no longer negligible, leading to the situation depicted in the Figures 5.4–5.6, [Hairer and Wanner, 1991, pp. 434–435].

By projecting the error estimate on the eigenvectors of the Jacobian it is possible to obtain the error components corresponding to the two modes. Figure 5.6 depicts these components as dashed lines, and it is clear that one of the modes is within the asymptotic region while the other is in the nonasymptotic region. The nonasymptotic component is uncontrollable using reasonable stepsize changes.

The stiff mode is less excited when using the L-stable method HW-SDIRK(3)4. Its error contribution remains small, and the integration proceeds without the need for drastic stepsize reductions during the stiff part of the solution, cf. Figure 5.2. □

Explicit Formulation of the Predictive Error Controller

To summarize the error controller obtained so far, an outline of the code needed to implement it is presented in Listing 5.1. The controller is incomplete in that it does not address the effect the stepsize choice has on the convergence in the equation solver. The stepsize calculated for error control needs to be restricted so that the convergence is not jeopardized. Moreover, various additional safety nets need to be included, e. g. discard unreasonably large stepsize changes, discard unreasonable values of k_{est}, protect against underflow/overflow, etc.

The variables h_{acc}, r_{acc}, h_{rej}, and r_{rej} are used to store the stepsize and error from the most recent accepted and rejected step, respectively.

It may happen that the equation solver fails to produce a new solution point. It is then not possible to calculate a valid error estimate r and the new stepsize has to be based on the observed behavior of the equation solver.

The code in Listing 5.1 prevents the use of the predictive control when convergence restricts the stepsize (*conv_restrict*). We will discuss this further in Section 5.3.

Choosing Controller Parameters

The parameters k_1 and k_2 influence the dynamics of the controller, and hence also the closed loop system. Using the expression for $\log h_n$ (5.6) in

$$\log r_n = k \log h_{n-1} + \log \varphi_{n-1}$$

together with the fact that $\log \varepsilon$ is constant yields

$$\log r_n = \log \varepsilon + \frac{(q-1)^2}{q(q^2 + (-2 + k_1 + k_2)q + 1 - k_1)} \log \varphi_n .$$

```
if iteration_successfully_completed then
    if current_step_accept then
        if first_step or first_after_succ_rej or conv_restrict then
            $h_r := \left(\frac{\varepsilon}{r}\right)^{1/k} h$
        else
            $h_r := \frac{h}{h_{\mathrm{acc}}} \left(\frac{\varepsilon}{r}\right)^{k_2/k} \left(\frac{r_{\mathrm{acc}}}{r}\right)^{k_1/k} h$
        endif
        $r_{\mathrm{acc}} := r$
        $h_{\mathrm{acc}} := h$
    else
        if successive_rejects then
            $k_{\mathrm{est}} := \frac{\log r/r_{rej}}{\log h/h_{\mathrm{rej}}}$
            $h_r := \left(\frac{\varepsilon}{r}\right)^{1/k_{\mathrm{est}}} h$
        else
            $h_r := \left(\frac{\varepsilon}{r}\right)^{1/k} h$
        endif
        $r_{\mathrm{rej}} := r$
        $h_{\mathrm{rej}} := h$
    endif
    $h :=$ restrict($h_r$)
else
    ⋮
endif
```

Listing 5.1 An outline of the code needed to implement the predictive controller. The restriction of h_r accentuates that the final choice of stepsize has to be coordinated with the convergence control. The algorithm has to be augmented with various safety nets, e.g. discard unreasonably large stepsize changes, discard unreasonable values of k_{est} protect against underflow/overflow.

By choosing k_1 and k_2 it is possible to position the two transfer function poles arbitrarily. Their location will determine how $\log \varphi_n$ affects $\log r_n$. In Figure 5.7 it is demonstrated how the properties of the poles depend on the values of k_1 and k_2. We strive for a fast system with well-damped poles [Åström and Wittenmark, 1990, Section 3.6 and Figure

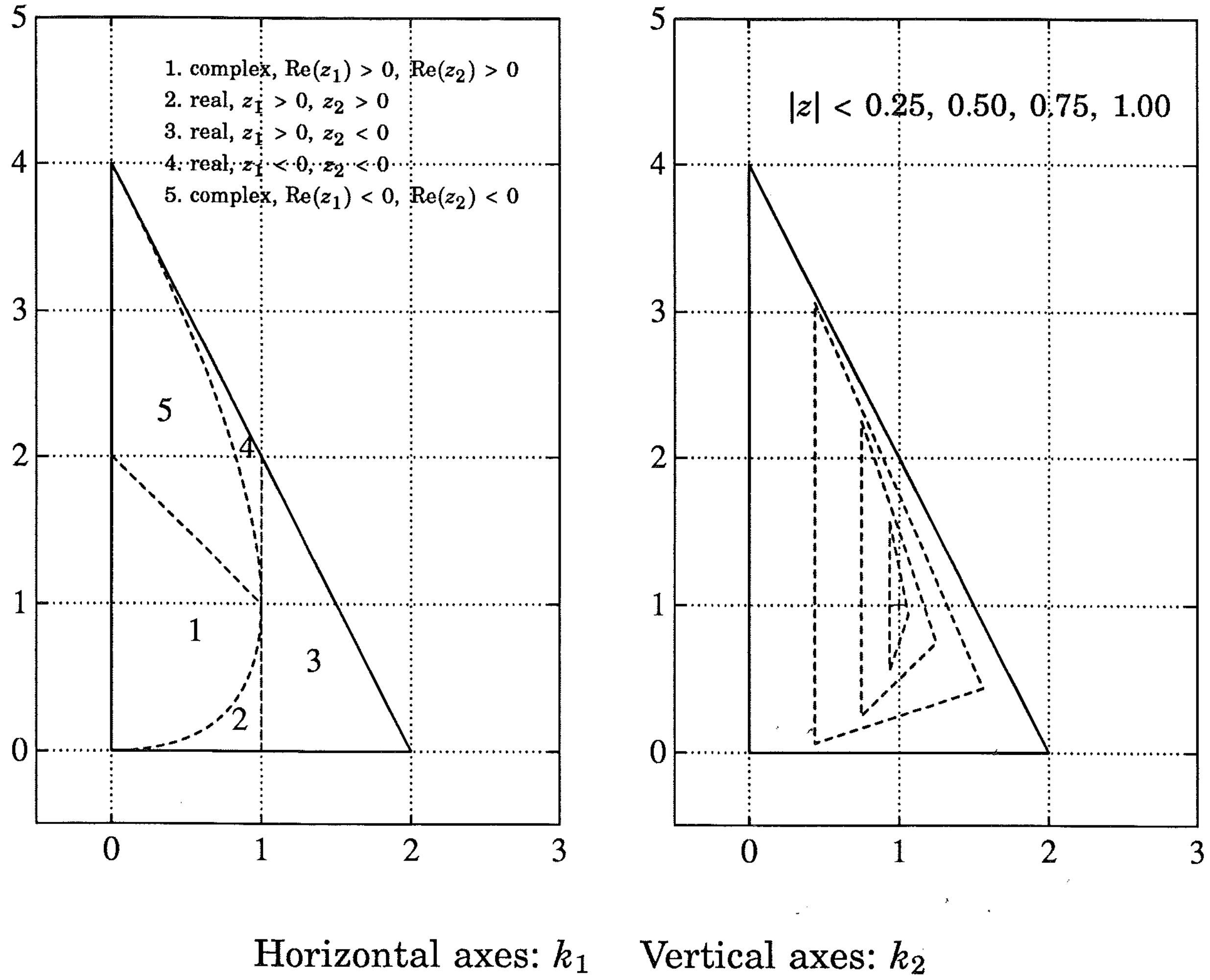


Figure 5.7 The properties of the poles in the transfer function from $\log \varphi_n$ to $\log r_n$ as a function of the parameters k_1 and k_2.

10.2], which suggests having both k_1 and k_2 in the neighborhood of 1 or slightly below.

Just studying pole locations is not enough when choosing values for k_1 and k_2. The nature of $\log \varphi$ is problem dependent and before the values for k_1 and k_2 are set, they have to be tested on real problems. Such tests, however, do seem to advocate the choice

$$k_1 = 1, \qquad k_2 = 1. \tag{5.8}$$

corresponding to the closed loop system having a double pole at the origin, i. e. deadbeat control. The following example shows some typical results.

EXAMPLE 5.4—Choosing values for k_1 and k_2.
We return to the Brusselator problem in Examples 1.2 and 4.1. The problem was solved using HW-SDIRK(3)4. A mixed absolute-relative 2-norm (2.14) with $\eta = 0.01$ and $\tilde{y} = |y|$ was used. The tolerance was set to $tol = 10^{-5}$ and $\varepsilon = 0.8\, tol$. To remove potential effects from the equation solver, modified Newton iterations were used with a new Jacobian at every step. The norm of the iteration error was required to be less than $0.01\,\varepsilon$.

Figure 5.8 depicts the total number of integration steps as a function of k_1 and k_2. The minimum is 128 steps, which should be compared with the 164 steps it takes to solve the problem with the standard controller (2.22). The minimum is quite broad, and the exact values of k_1 and k_2 are not too critical. It seems reasonable to have $k_1 = k_2 = 1$. □

Continuous Stepsize Variations

Most algorithms for selecting stepsize restrict the stepsize variations, cf. Figure 5.1. A limitation on too large stepsize changes is natural, since occasionally the error estimator may produce an unusually large (or small) value. Many algorithms, however, also prevent stepsize changes if they are too small. This nonlinearity is introduced into the feedback loop in order to improve efficiency by reducing the number of factorizations of the iteration matrix when using modified Newton [Nørsett and Thomsen, 1987; Hairer and Wanner, 1991, p.134].

An effect of preventing small stepsize changes is that once a change is allowed it will usually be rather large, cf. Example 2.11. Such changes often lead to transient effects that may differ from what is predicted by the stepsize-error and stepsize-convergence models available. Small changes produce a smoother behavior of the error with respect to the set-point ε of the error control. This is of importance if tolerance proportionality is to be achieved.

The prevention of small stepsize changes is often introduced in an asymmetric way, i. e. a stepsize decrease is readily accepted (in order to reduce the risk for rejection) while a stepsize increase has to be substantial to be considered. As intended this reduces the number of factorizations, but if the number of small stepsize decreases is large there may still be many unnecessary factorizations. As an example consider the stepsize sequence in the lower plot of Figure 5.3. In every step from step 40 to step 200 there is a stepsize decrease by approximately 4 %.

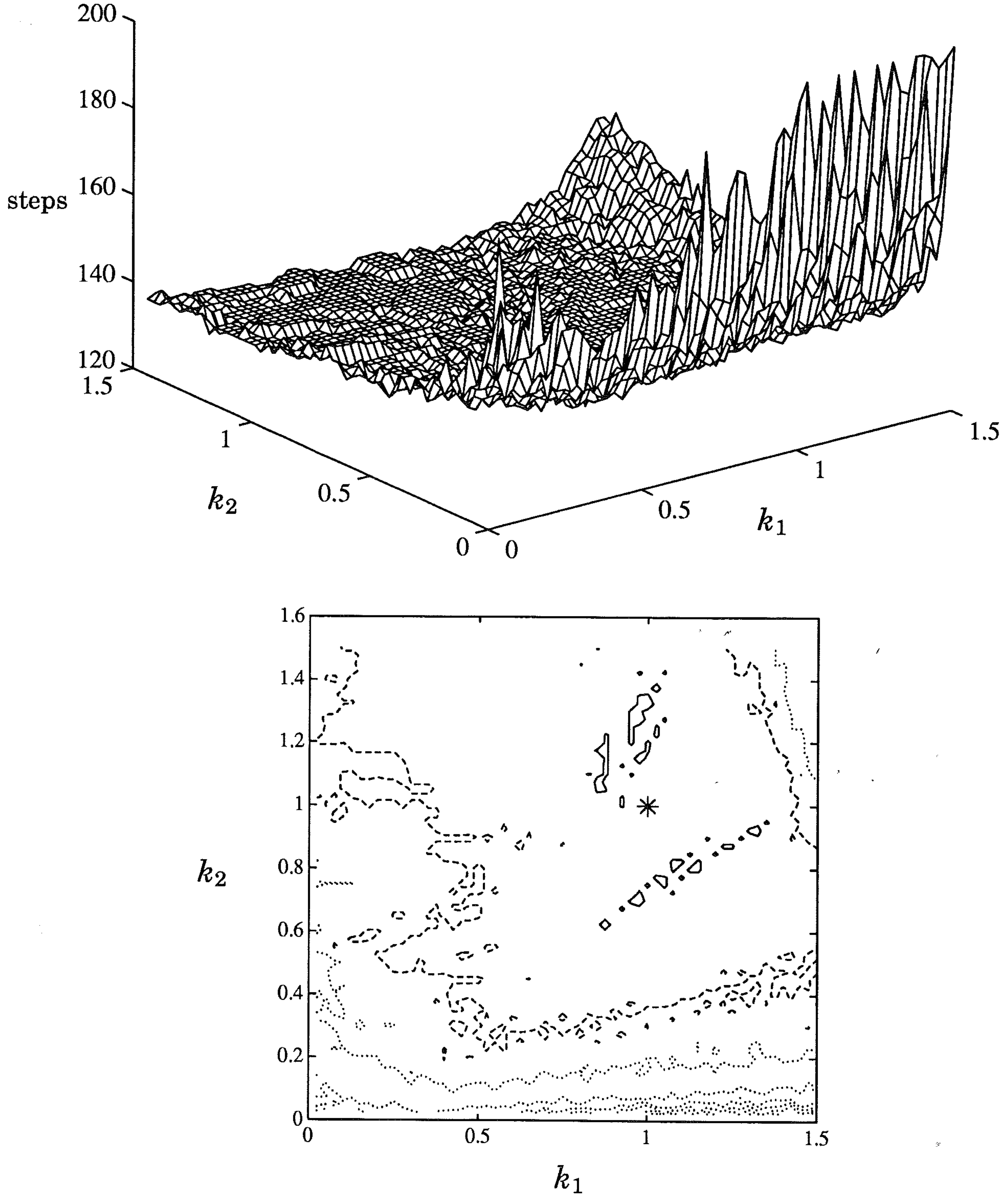


Figure 5.8 Total number of integration steps as a function of k_1 and k_2 when solving the Brusselator in Example 5.4. The lower plot depicts the level curves for the surface in the upper plot. The full line represents the parameter values that give less than 1 % increase in number of steps compared to the minimum (128 steps). The dashed line represents an increase by 5 %, while the 10 %, 15 %, 20 %, and so on, are plotted with dotted lines. The values of k_1 and k_2 should be about 1 for a minimum number of steps, but the exact values are not too critical. The predictive controller performs better than the standard, which requires 164 integration steps. The star marks the parameter choice $k_1 = 1$, $k_2 = 1$, cf. (5.8).

Preventing the change is not reasonable since it corresponds to an error growth by almost 20 % per step. On the other hand it is questionable if the changes are large enough to call for a refactorization of the iteration matrix at every step.

If the stepsize is allowed to vary more freely it is of course infeasible (and unnecessary) to factorize the iteration matrix at every stepsize change. Most changes are small and will not impair the convergence rate. This, however, calls for a different strategy regarding the factorizations, a matter that we will return to in Section 5.3.

Although we advocate removing some of the normally used restrictions on stepsize change, this is by no means a requisite for using the predictive controller. The predictive controller, expressed on the form (5.1), calculates a new stepsize based on the actual restricted stepsizes and the corresponding error estimates. A prevented stepsize change does not cause any extra problems apart from making the error estimate deviate unnecessarily from the error control set-point.

5.2 The Iteration Error in the Equation Solver

The equation solver is a basic building block in an implicit integration method. During the integration we need to assure that it operates efficiently and that its output is of sufficient quality. In this section we will discuss some aspects of these requirements, mainly concerning the iteration error control loop in Figure 5.1. The results form a basis for the design of the convergence controller in Section 5.3.

Most of what will be discussed is well known [Shampine, 1980]. Still it needs to be discussed, since many control strategies rely (maybe too) strongly on the asymptotic properties of the iteration. In addition, the connection between the iteration error, i.e. the quality of Y, and the integration error is not well understood. We will mention some current research that might force a revision of the iteration error control strategy.

For notational simplicity consider the case, e.g. SDIRK, where the iterations are done stagewise (the arguments and results are similar for the general case (2.24a)). At every integration step an iterative scheme is used to solve a problem on the form

$$Y_i = \psi_i + \gamma_i h_n f(Y_i), \tag{5.9}$$

where Y_i is the i:th stage value (2.23), ψ_i a constant depending on (t_n, y_n) and previous stage values, and γ_i is the i:th diagonal element of $\mathcal{A}$. In the sequel the index i will be suppressed. As noted earlier (Section 2.4), under certain conditions (5.9) has a unique solution Y^* [Kraaijevanger and Schneid, 1991], and we expect the iteration to rapidly converge towards this point.

The iteration can be written

$$Y^{m+1} = G(Y^m) = (I - \gamma h'J)^{-1}(\psi + \gamma h_n f(Y^m) - \gamma h'JY^m) \qquad (5.10)$$

where h_n is the current stepsize, h' the stepsize for which the iteration matrix $M = I - \gamma h'J$ was formed, and J an approximation of the Jacobian $\partial f/\partial y$. By setting $J = 0$ we obtain fixed point iteration.

If the initial point Y^0 is close to Y^*, and h_n and h' are small enough, then G is a contraction and (5.10) converges linearly to Y^* [Ortega and Rheinboldt, 1970, 10.1.3, 10.1.4], [Alexander, 1991]. Roughly speaking, the better J approximates $\partial f/\partial y$ the less stringent asymptotic demands on h_n and h' for convergence.

The Convergence Rate

In practice the convergence of (5.10) has to be supervised. A possibly divergent process should be terminated as soon as possible. In addition, if the convergence rate is known it can be used to estimate the current iteration error as well as the number of iterations needed to bring this error down below τ.

The convergence rate may be estimated as (2.35)

$$\alpha = \max_m \alpha_m = \frac{\|Y^{m+1} - Y^m\|}{\|Y^m - Y^{m-1}\|} = \frac{\|\Delta_m\|}{\|\Delta_{m-1}\|}. \qquad (5.11)$$

The estimate (5.11) is an underestimate of the Lipschitz constant L of G, and the iteration need not be contractive although $\alpha < 1$. The asymptotic convergence rate of the iteration (5.10) is $\rho(G'(Y^*))$, where $\rho(\cdot)$ denotes the spectral radius [Ortega and Rheinboldt, 1970, 10.1.4]. In practice (5.11) may, however, assume values much larger or smaller than $\rho(G'(Y^*))$. There are several possible reasons for this: the iteration may not yet have reached the asymptotic phase, or the norm used to calculate (5.11) may differ from the one needed to observe the rate $\rho(G'(Y^*))$.

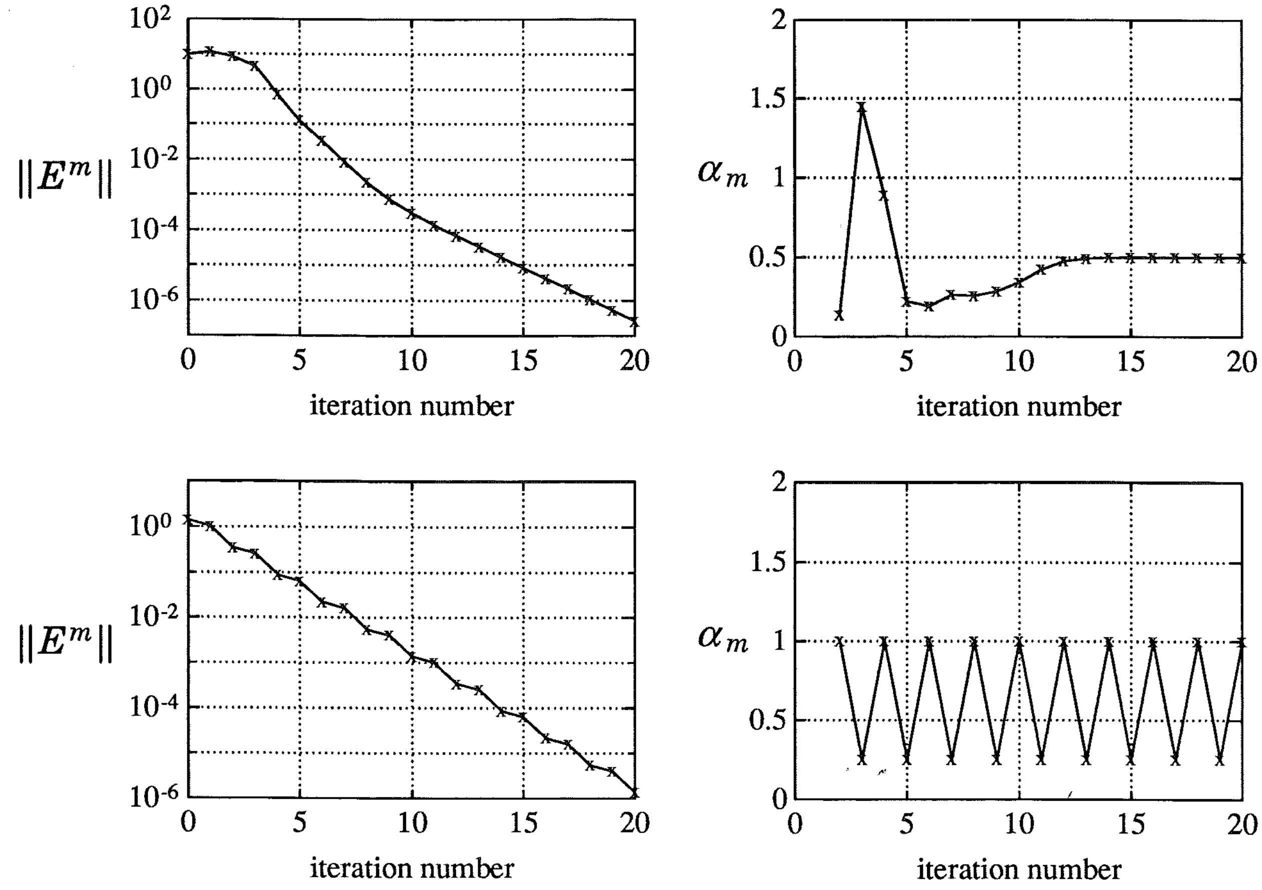


Figure 5.9 This figure demonstrates how the iteration in Example 5.5 converges towards the fixed point Y^* at the origin. The upper plots correspond to the nonlinear equation and the lower plots correspond to the linear equation. As can be seen the convergence rate estimate α_m may deviate considerably from the spectral radius α, which in both cases equals 0.5.

EXAMPLE 5.5—The quality of the convergence estimate

Consider the two iterations

$$\begin{pmatrix} Y_1^{m+1} \\ Y_2^{m+1} \end{pmatrix} = \begin{pmatrix} Y_1^m\left(-\frac{1}{2} - \frac{1}{10}Y_2^m\right) \\ Y_2^m\left(-\frac{1}{4} + \frac{1}{10}\left(Y_1^m - Y_2^m\right)\right) \end{pmatrix}, \qquad \begin{pmatrix} Y_1^0 \\ Y_2^0 \end{pmatrix} = \begin{pmatrix} 1 \\ 10 \end{pmatrix}$$

$$\begin{pmatrix} Y_1^{m+1} \\ Y_2^{m+1} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{4} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} Y_1^m \\ Y_2^m \end{pmatrix}, \qquad \begin{pmatrix} Y_1^0 \\ Y_2^0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Both equations have a fixed point at the origin. Figure 5.9 depicts the norm of the error $E^m = Y^m - Y^*$ and the corresponding estimate α_m as the iteration converges towards Y^*. The standard 2-norm is used. Note that the rate estimate is not available until after the second iteration.

The first problem is nonlinear and its Jacobian has the eigenvalues $-1/2$ and $-1/4$ at Y^*. The iteration of the equation goes through three phases. The first, $m \in [1, 4]$, is a nonlinear transient, where the iteration converges although not being strictly contractive in the chosen norm. The second phase, $m \in [5, 10]$, is a linear transient governed by the $-1/4$ eigenvalue, and the iteration converges with approximately the rate $1/4$. After this fast mode has decayed the slower mode corresponding to the eigenvalue $-1/2$ will dominate, and the iteration enters phase 3, $m \in [11, 20]$. This phase demonstrates the asymptotic properties predicted by $\rho(G'(Y^*))$, and we observe how the convergence rate estimate α_m approaches $1/2$.

The second problem is linear and its Jacobian has eigenvalues $\pm i/2$. The Jacobian is, however, not a normal matrix, and the norm is not "aligned" with the eigenvectors. As a result we will not observe the asymptotic convergence rate $1/2$ (cf. the norm discussion in Section 3.5).

□

The behavior in Example 5.5 demonstrates that the convergence estimate (5.11) may differ considerably from the asymptotic value, and in particular underestimates L.. The nonlinear transient may often be avoided by having a good predictor for the starting value Y^0, but due to norm misalignment and/or eigenvalues of different magnitude the estimate α_m may still be larger or smaller than the theoretical asymptotic value. Shampine comments that too much attention is paid to the asymptotic behavior [Shampine, 1980]. He argues, very convincingly, that for robustness one should insist on the iteration being contractive. If any $\alpha_m > 1$ the iteration should be terminated. Moreover, the largest observed α_m should be taken as estimate α for the convergence rate, cf. (5.11) and Figure 5.1.

Some algorithms do not check the convergence of (5.10). Instead, if any $||\Delta_m||$ is small enough the corresponding iterate is accepted. Shampine demonstrates [Shampine, 1980] that such a strategy may erroneously accept a solution point from a divergent iteration. Therefore a minimum of two iterations should always be done, so that an estimate α_m can be formed. The iterate should not be accepted if $\alpha_m > 1$, irrespective of the size of $||\Delta_m||$.

The Iteration Error

The deviation E^m between the final iterate Y^m and Y^* should ideally be small. It is well known that [Ortega and Rheinboldt, 1970, 12.1.2]

$$||E^m|| \leq \frac{L}{1-L}||\Delta_m||$$

with L being the Lipschitz constant of G, cf. (5.10). To assure $\|E^m\| \leq \tau$ we may test if

$$\frac{L}{1-L}\|\Delta_m\| \leq \tau. \tag{5.12}$$

In practical computations it is common to replace the test (5.12) with the (weaker) requirement

$$\frac{\alpha}{1-\alpha}\|\Delta_m\| \leq \tau. \tag{5.13}$$

with α as the convergence rate estimate (5.11). Although (5.13) does not imply $\|E^m\| \leq \tau$, this is usually not a severe limitation [Söderlind, 1984a]. Moreover, since $\|E^m\| < L\|E^{m-1}\|$, it is reasonable to use

$$\frac{\alpha^{1+m_{\text{remain}}}}{1-\alpha}\|\Delta_m\| \leq \tau \quad \Rightarrow \quad m_{\text{remain}} \geq \frac{\log\left(\frac{1-\alpha}{\alpha}\right) + \log\frac{\tau}{\|\Delta_m\|}}{\log\alpha} \tag{5.14}$$

as an estimate of the remaining number of iterations m_{remain} needed to bring the iteration error below τ. If m_{remain} is too large the iteration should be terminated and a new try with updated stepsize and/or iteration matrix should be made. Often a maximum of 7–10 iterations [Hairer and Wanner, 1991, p. 131] is accepted.

When using (5.13) and (5.14) we must anticipate that α_m may be a misleading estimate for α, and in particular, could be too small. This is the motive for using the largest α_m as estimate α for the convergence rate and demanding the iteration to be contracting.

Recently it has been shown [Jackson *et al.*, 1992] that the order of a Runge-Kutta method is connected to the number of iterations performed in the equation solver; specifically, a minimum number of iterations are needed to recover the order of the underlying discretization formula. The exact number depends on the type of problem (stiff – nonstiff), the order of the interpolation polynomial used to obtain Y^0, and the type of iteration used in the equation solver. The result suggests that in addition to checking the iteration error a minimum number of iterations should be done. This number is often well below the maximum number of iterations (7–10) normally allowed.

Fixing the number of iterations in the equation solver will probably make the iteration error change more smoothly from step to step. The iteration error contributes to the error estimate (2.32), and one may hope for a smoother error estimate as well. A smooth error sequence makes the error control easier. However, when the convergence of the iteration is changed (a new Jacobian and/or a refactorization of the iteration matrix), the iteration error (and thus the error estimate) may change abruptly. This has to be considered in the error control algorithm, i. e. it may be difficult to predict changes in φ when the iteration matrix is updated.

The iteration error propagates from step to step. If $P(\infty) \neq 0$ or $P(z)$ tends too slowly to 0 as $z \to \infty$, these errors may accumulate and eventually be large enough to dominate the integration error [Arévalo, 1992]. The picture is not yet clear, but the importance of having $P(z) \to 0$ as $z \to \infty$ is again demonstrated.

The Control Strategy for the Iteration Error

In our control algorithm we have chosen a conservative version of the traditional strategies [Shampine, 1980; Hairer and Wanner, 1991, pp. 130–131]. The main parts are:

- The acceptance test of the iterate Y^{m+1} is based on the norm of the displacement, i. e. $\|\Delta_m\|$, to be consistent with the local error control.
- To be able to check the convergence a minimum of two iterations are always done.
- The largest α_m is taken as estimate α for the convergence rate.
- An iterate is accepted when (5.13) is fulfilled, and all $\alpha_m < 1$.
- The iteration is terminated if any $\alpha_m > 1$.
- The number of remaining iterations is estimated with (5.14), and the iteration is terminated if this indicates a total of more than 10 iterations.

From the discussion above we want to stress that the processes governing the iteration error effect on the integration error is not fully understood. The outcome from on-going research [Jackson *et al.*, 1992; Arévalo, 1992] may very well force a revision of the current control strategy regarding the equation solver and the iteration error.

5.3 Convergence Control

While proceeding with the integration the controller needs to supervise the equation solver and make decisions like (cf. Figure 5.1): which equation solver should be used, when should the iteration matrix be evaluated and factorized (in case of modified Newton iterations), and what restrictions should be put on the stepsize to assure convergence? This part of the controller is vital, and efficient integration hinges on finding a good control strategy.

Although the strategy presented here is quite similar to current algorithms, e.g. [Shampine, 1980; Nørsett and Thomsen, 1987; Alexander, 1991; Hairer and Wanner, 1991, p. 134], there are some things worth noting:

- Small stepsize changes are normally prevented in order to reduce the number of factorizations of the iteration matrix. We remove this restriction since a smooth stepsize sequence leads to smoother error control.
- Most control strategies refactorize the iteration matrix at every stepsize change. When allowing small stepsize changes this is infeasible (and unnecessary). We derive a bound on the convergence in terms of the relative stepsize change, and use this to decide when to factorize. The same bound helps determining when to reevaluate the Jacobian.
- The switch from fixed point iteration to modified Newton is done based on an inefficiency measure that relates the stepsize needed to assure convergence and the one that would give $r = \varepsilon$. The switch back is based on a traditional estimate of the local stiffness of the differential equation.
- When the stepsize has to be restricted to assure convergence we use $\alpha_{\text{ref}} = 0.4$ as set-point for the convergence rate. This value will be shown to minimizes the amount of work needed to complete the integration.

Each one of these items will be discussed in detail, but first we investigate how the convergence rate α is related to variations in the stepsize and in the Jacobian. The obtained models will be used as a basis for further discussions concerning the convergence control strategy.

Convergence Effects Due to Stepsize and Jacobian Variations

To be able to control the convergence successfully we need to model its dependence on the available control variables, e. g. the stepsize and evaluations and/or refactorizations of the iteration matrix, cf. Figure

5.1. The iteration (5.10) can be rewritten (cf. (2.30)) as

$$E^{m+1} = M^{-1}\left(\gamma h_n \bar{J} - \gamma h' J\right) E^m, \qquad M = (I - \gamma h' J) \tag{5.15}$$

where $\bar{J}$ is a mean value Jacobian [Ortega and Rheinboldt, 1970, 3.2.6]

$$\bar{J} = \int_0^1 \frac{\partial f}{\partial y}(Y^* + \zeta\,(Y^m - Y^*))\, d\zeta.$$

Fixed point iteration and modified Newton iteration behave quite differently and we will treat them separately.

Fixed point iteration. The case of fixed point iteration is trivial. Setting $J = 0$ in (5.15) results in

$$E^{m+1} = \gamma h_n \bar{J} E^m, \tag{5.16}$$

and obviously the convergence rate depends on both the stepsize and the (unknown) Jacobian $\bar{J}$. One may write

$$\alpha = \vartheta h_n, \tag{5.17}$$

where $\vartheta \lesssim \gamma \|\bar{J}\|$. Note the similarity to the stepsize-error model (3.2).

Modified Newton iteration. Consider again the iteration (5.15). The iteration matrix M was formed and factorized using the stepsize h' and the Jacobian approximation J. The convergence of the iteration is governed by the iteration matrix and $\delta(h_n\bar{J}) = h_n\bar{J} - h'J$, i.e. the difference between the current stepsize and Jacobian compared to the values used when forming the iteration matrix. Since J is merely a numerical approximation, the value of $\delta(h_n\bar{J})$ will always differ from zero.

Our aim is to find a bound for α in terms of the relative deviation in $h_n\bar{J}$ from $h'J$. Assuming that J^{-1} exists, (5.15) can be rewritten

$$E^{m+1} = (I - \gamma h' J)^{-1} \gamma h' J (h'J)^{-1} \delta(h_n\bar{J}) E^m.$$

Introduce $\nu = \left\|(I - \gamma h'J)^{-1}\gamma h'J\right\|$, then

$$\alpha \leq \nu \left\|(h'J)^{-1}\delta(h_n\bar{J})\right\|. \tag{5.18}$$

Our intention is to use (5.18) to predict how changes in $\delta(h_n\bar{J})$ affect the convergence rate, but then the value of ν has to be known.

By a generalization [Nevanlinna, 1984; Söderlind, 1986a] of the spectral theory of von Neumann [von Neumann, 1951], we have

$$\mu_H[\gamma h'J] \le \mu < 1 \quad \Rightarrow \quad \left\|(I - \gamma h'J)^{-1}\gamma h'J\right\|_H \le \max_{\mathrm{Re}\, z \le \mu} \left|\frac{z}{1-z}\right|, \tag{5.19}$$

where $\|\cdot\|_H$ is any inner product vector norm and $\mu_H[\cdot]$ denotes the corresponding logarithmic norm [Söderlind, 1984b]. Hence, if $\mu_H[\gamma h'J] \le 1/2$, a rather realistic assumption, then $(I - \gamma h'J)^{-1}\gamma h'J$ is a contraction and $\nu < 1$. One may ask how or to what extent this bound depends on the choice of norm. Since α should (ideally) be evaluated as the asymptotic convergence rate, this dependence is generally weak. However, the actual value will depend on the (unknown) value of $\mu_H[\gamma h'J]$, but this dependence is also weak unless $1/2 < \mu_H[\gamma h'J] < 1$.

When $\|h'J\| \ll 1$, i. e. the nonstiff case, then $\nu \ll 1$ and a bound

$$\alpha < \left\|(h'J)^{-1}\delta(h_n\bar{J})\right\| \tag{5.20}$$

will not be particularly sharp. This is not a serious problem since we do not expect to use modified Newton iteration when the differential equation is nonstiff. In the stiff case we have $\|h'J\| \gg 1$ with J dominated by some large eigenvalue with negative real part. The components in the stiff subspace of J take active part in the iteration (5.15), and for the norm used we expect $\nu \to 1$ as $\|h'J\| \to \infty$. The bound (5.20) will hence normally be sharp, and it loses sharpness only for a nonstiff problem or a problem in transition between nonstiff and stiff, cf. Example 5.9 and Figure 5.15.

If $\bar{J}$ remains constant and only the stepsize varies we can write (5.20) as

$$\alpha \le \left|\frac{\delta h_n}{h'}\right|. \tag{5.21}$$

It can easily be seen, cf. (5.18), that this bound holds also when J is singular, under the single assumption $\mu_H[\gamma h'J] \le 1/2$.

Likewise, if h_n is constant and $\bar{J}$ varies, then

$$\alpha \le \left\|J^{-1}\,\delta\bar{J}\right\|_H \tag{5.22}$$

whenever $\mu_H[\gamma h'J] \le 1/2$. For simultaneous variations in h_n and $\bar{J}$ we neglect higher order variations, and obtain

$$\delta(h_n\bar{J}) \approx J\delta h_n + h'\delta\bar{J},$$

which together with (5.20) leads to

$$\alpha \lesssim \left\| \frac{\delta h_n}{h'} I + J^{-1} \delta \bar{J} \right\|_H \lesssim \left| \frac{\delta h_n}{h'} \right| + \| J^{-1} \delta \bar{J} \|_H , \tag{5.23}$$

provided that $\mu_H[\gamma h' J] \leq 1/2$.

The bounds (5.21) and (5.23) are important. They offer a possibility to predict how a stepsize change will affect the convergence rate. This can (and will later in this section) be used to decide when to factorize the iteration matrix. Likewise, a poor convergence rate, although the stepsize has been kept virtually constant, indicates, cf. (5.23), that the Jacobian has changed and it may be wise to reevaluate the iteration matrix.

Coordinating Stepsize Choice and Convergence Control

In some situations the stepsize has to be restrained in order to assure convergence in the equation solver. The most common case is when fixed point iteration is used and the integration enters a moderately stiff region of the differential equation. The stepsize is the only available control variable affecting the convergence rate, and convergence may be a more restrictive constraint than accuracy.

When using modified Newton the convergence is normally secured by appropriately updating the iteration matrix, i. e. by keeping $\delta(h_n \bar{J})$ small. It may, however, happen that the convergence is poor although the iteration matrix is based on current data. The Jacobian J is an approximation calculated at one solution point, and the distance between the different stage points may be large enough to jeopardize convergence. Reducing the stepsize brings the stage points closer and convergence may be secured.

In both cases described above the stepsize-convergence relation may be modeled as (5.17). This model has the same structure as the standard stepsize-error relation, and assuming ϑ constant, the stepsize should be chosen as

$$h_\alpha = \frac{\alpha_{\text{ref}}}{\alpha} h_n \tag{5.24}$$

to obtain $\alpha = \alpha_{\text{ref}}$ in the next step. If the change in ϑ is substantial between steps, one could envision a predictive controller on the same form as the one derived for error control in Section 5.1.

The main problem with (5.24) is the low quality of the convergence rate estimate α. As was seen in Example 5.5 this estimate may deviate considerably from the asymptotic value. In addition, it behaves nonsmoothly with respect to the stepsize; a small stepsize change may alter the number of iterations needed and thereby cause a quite different estimate. This lack of smoothness manifests itself as variations in ϑ, and it is hardly any use trying a more sophisticated strategy than (5.24).

The stepsize suggested by (5.24) has to be coordinated with the one from the error control. A straightforward way is to choose the minimum, i. e. when using fixed point iteration or experiencing poor convergence in modified Newton in spite of an iteration matrix based on current data, the restriction in Listing 5.1 should be implemented as

$$h := \min(h_r, h_\alpha), \tag{5.25}$$

with h_α taken from (5.24).

A problem with (5.25) and (5.24) is the deadbeat character it gives to the convergence control loop. When convergence restricts the stepsize, i. e. $h_\alpha < h_r$, any value different from α_{ref} will be brought to α_{ref} in one step. Considering the low quality of the convergence rate estimate this may be a too aggressive strategy. Two possibilities for making the strategy smoother are: filter the convergence rate estimate before using it in (5.24) and/or change (5.24) so that the closed loop pole is moved from the origin to some position in the interval [0, 1], cf. Section 4.1 and the discussion on the choice of k_I in the standard stepsize selection rule (2.22). In the simulations in this chapter we will use (5.24) and (5.25) as is.

Convergence restricting the stepsize leads to an error estimate below ε. When $h_\alpha < h_r$ the corresponding error estimate $r \ll \varepsilon$ and it may be dominated by the contribution from the iteration error, cf. (2.32), making r fluctuate wildly. These fluctuations do not provide any information about the variations in φ. Therefore it is not advisable to use the predictive error controller (5.1) to calculate h_r in this case. Hence, we use the standard stepsize selection rule (2.22) whenever the previous stepsize was restricted by h_α. The test on *conv_restrict* achieves this change of controller in Listing 5.1.

Example 5.6—Coordinated error and convergence control

Consider the van der Pol oscillator with $\sigma = 10$ (for the solution see Figure 4.8). The problem was solved using HW-SDIRK(3)4. A mixed absolute-relative 2-norm (2.14) with $\eta = 10^{-4}$ and $\tilde{y} = |y|$ was used. The tolerance was set to $tol = 10^{-4}$ and stepsize control according to

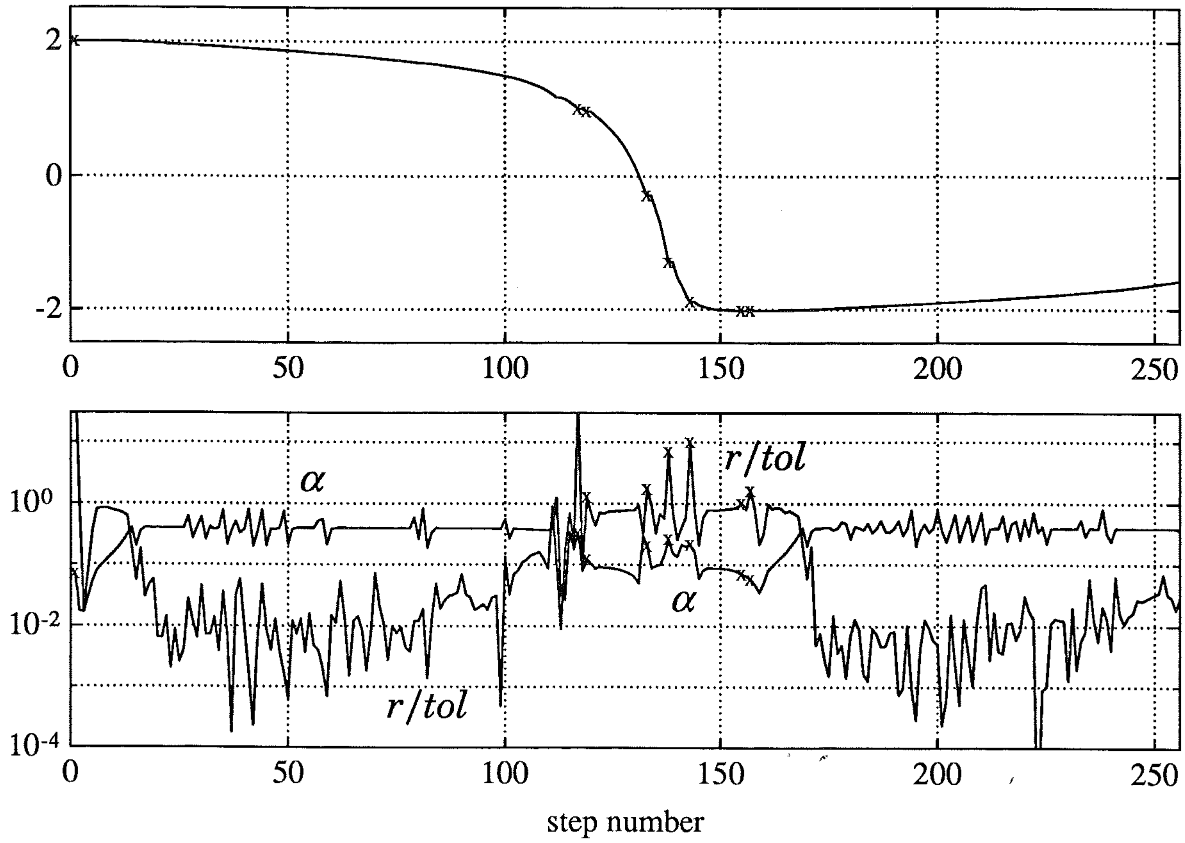


Figure 5.10 The upper plot depicts the solution of the first component y_1 when solving the van der Pol problem in Example 5.6. The curve looks erratic, but this is only due to the time-distortion introduced by plotting y_1 as a function of integration step number. The lower plot shows the error estimate and the estimate of the convergence rate. During the initial transient and the fast state transition it is accuracy that limits the stepsize, while the stepsize is held back to maintain acceptable convergence during the smooth part of the solution. The coefficient ϑ in the stepsize-convergence model varies, and consequently α sometimes "oscillates" around the set-point $\alpha_{\text{ref}} = 0.4$, e.g. steps 25–60 and 180–240. During the state transition there are a couple of rejected steps (indicated with crosses). The cause of these rejected steps is described in Example 5.2.

Listing 5.1 ($k_1 = k_2 = 1$), (5.24), and (5.25) was used with $\varepsilon = 0.8\,tol$. Fixed point iteration with $\alpha_{\text{ref}} = 0.4$ was used at all integration steps, and the norm of the iteration error was required to be less than $0.01\,\varepsilon$.

The problem is mildly stiff during the smooth part of the solution, and the stepsize has to be restricted to assure convergence in the fixed point iteration. Figure 5.10 demonstrates how the controller uses the stepsize for error control and convergence control, cf. (5.25), during different parts of the integration.

Due to variations in the convergence rate estimate it is hard to keep α close to the set-point $\alpha_{\text{ref}} = 0.4$. Figure 5.11 demonstrates the relation

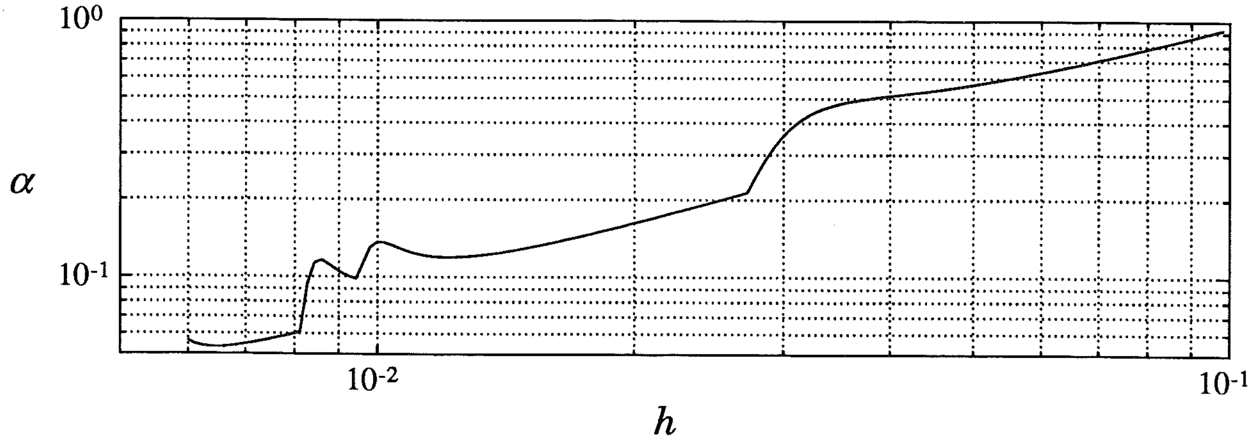


Figure 5.11 The stepsize-convergence relation at step 30 of the integration in Figure 5.10. The overall slope of the curve is about 1 confirming the linear stepsize-convergence dependence (5.17). The proportionality factor ϑ may, however, vary and in this case we have several regions with different ϑ. These regions change from step to step, making it difficult to keep $\alpha = \alpha_{\text{ref}}$.

between the stepsize and the estimate of the convergence rate. The data are taken from step 30 in Figure 5.10. During the actual integration the stepsize was $h \approx 0.055$ at this step, resulting in $\alpha \approx 0.60$. In the next step the controller reduced the stepsize to $h \approx 0.037$ to obtain $\alpha = \alpha_{\text{ref}}$. The value of ϑ, however, changed and the new stepsize led to $\alpha = 0.26$. The variations in ϑ in combination with the deadbeat control strategy are responsible for the oscillations in the convergence control loop. □

Choice of Set-Point for the Convergence Rate

When restricting the stepsize to assure convergence in the equation solver, an important question is what convergence rate, i. e. α_{ref}, the controller should aim for. Of course $0 < \alpha_{\text{ref}} < 1$, but what value gives the most efficient integration? This question can be addressed using an idea due to Söderlind [Söderlind, 1986b].

Assume m is the number of iterations needed to fulfill (5.13). Then, using (5.13) and again substituting α for L,

$$\tau \geq \frac{\alpha}{1-\alpha}\|\Delta_m\| \gtrsim \|E^m\| \approx \alpha^m \|E^0\|, \tag{5.26}$$

where E^0 is the error in the initial guess Y^0. From (5.26)

$$m \gtrsim \frac{\log \tau - \log \|E^0\|}{\log \alpha},$$

and, consequently, the number of function evaluations per unit step is proportional to $\bar{m}$, where (approximately)

$$\bar{m} = \frac{m}{h} = \frac{\log \tau - \log \|E^0\|}{h \log \alpha}.$$

We would like to find the value of α that minimizes $\bar{m}$. The convergence α is proportional to h, therefore

$$\bar{m} \sim \frac{\log \tau - \log \|E^0\|}{\alpha \log \alpha}.$$

The starting value Y^0 is constructed using an interpolation polynomial, which makes the initial error E^0 depend on h (and hence α). The dependence is weak, and hence

$$\bar{m} \sim -\frac{1}{\alpha \log \alpha}, \tag{5.27}$$

where the negative sign is a consequence of assuming $E^0 > \tau$. The expression (5.27) has a minimum at $\alpha = e^{-1} \approx 0.4$, which suggests choosing

$$\alpha_{\text{ref}} = 0.4. \tag{5.28}$$

The function (5.27) is rather flat around the minimum (cf. Figure 5.12), and any value $0.2 < \alpha_{\text{ref}} < 0.5$ would probably be acceptable, with robustness favoring the lower values.

Example 5.7—The effect of convergence rate set-point on efficiency
The van der Pol problem was solved with different values on α_{ref} using exactly the same setup as in Example 5.6. The resulting number of function calls as a function of α_{ref} is shown in Figure 5.12. The values agree very well with the ones predicted by (5.27) (the expression (5.27) was normalized with a constant that makes its minimum equal the minimum number of function calls experienced in practice).

As described in Section 5.2 the equation solver includes a limit on the number of iterations allowed. This makes the values experienced in practice deviate from (5.27) for large α_{ref}. The effect is readily visible in Figure 5.12.

Note that the convergence control is only active when convergence restricts the stepsize, i.e. $h_\alpha < h_r$ in (5.25), and it is only in this case that we try to achieve a convergence rate equal to α_{ref}. □

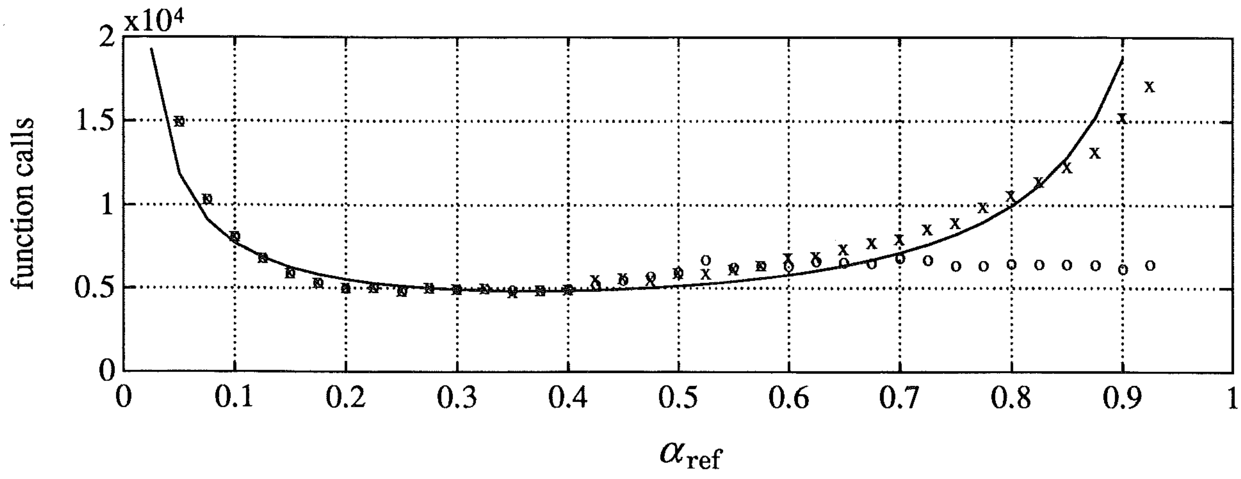


Figure 5.12 Total work (number of function calls) as a function of the convergence rate set-point when integrating the van der Pol problem in Example 5.6. The equation solver includes a limit on the number of allowed iterations. The crosses 'x' and the rings 'o' correspond to setting this limit at 100 and 10, respectively. The actual number of function calls agrees very well with the work predicted by (5.27) (shown as a full line). The deviation for large α_{ref} is a consequence of limiting the number of iterations in the equation solver.

A Strategy for Choosing Equation Solver

To improve efficiency many implicit integration methods try to switch to fixed point iteration during nonstiff regions of the differential equation. The "optimal" switching points depend in a very complicated way on both the differential equation and the integration method. Most strategies therefore do not try to find these points, but rather aim at implementing a simple but robust strategy that performs sufficiently well on a large class of problems. The strategies described in [Shampine, 1981; Nørsett and Thomsen, 1986b] are representative. We will give a short recapitulation and comment on a few issues sometimes overlooked.

The integration is normally started using fixed point iteration. The stepsize will be small to resolve initial transients, and fixed point iteration is likely to converge. In addition, if the problem never turns stiff, the cost of forming the iteration matrix has been saved.

Switching from fixed point to modified Newton. A stepsize that is held back to assure convergence in the fixed point iteration is an indication that it would be beneficial to switch to modified Newton iteration. Shampine compares the stepsize h_r from error control with the stepsize h_α from convergence control, and if the difference is too large a switch is made [Shampine, 1981]. A problem is that what "large enough" is, is in general unknown; it depends on the method as well as the future solution of the differential equation. Often the switch is done hoping that the current observed behavior is part of a trend where the problem is becoming more and more stiff. By switching to modified Newton iteration we will be able to anticipate this change more efficiently.

Suppose that to do the switch we require a possible stepsize increase by a factor ξ. Consequently, we await a situation where $h_r/h_\alpha > \xi$ [Shampine, 1981]. To make probable that the switch to modified Newton will be profitable we would like to have ξ large. It may, however, be difficult to achieve $h_r/h_\alpha > \xi$ in practice, since the iteration error in the equation solver will prevent r from becoming very small in the case where convergence restricts the stepsize. In practice this puts an upper limit on the choice of ξ.

Suppose convergence restricts the stepsize, and that the convergence rate is kept close to α_{ref}, i. e. $h_\alpha \approx h_n$. In this case the standard stepsize selection rule (2.22) is used to calculate h_r, cf. *conv_restrict* in Listing 5.1. Hence

$$\frac{h_r}{h_\alpha} \approx \frac{\left(\frac{\varepsilon}{r_{n+1}}\right)^{1/k} h_n}{h_n} = \left(\frac{\varepsilon}{r_{n+1}}\right)^{1/k} \tag{5.29}$$

and the value of r_{n+1} determines if we will observe $h_r/h_\alpha > \xi$ or not. The slow convergence in combination with $h_\alpha < h_r$ suggest that the error estimate will be dominated by the contribution from the iteration error, cf. (2.32). This can be seen in Figure 5.10, where $r \approx \tau$ ($\tau = 0.01\,\varepsilon$). Setting $r_{n+1} = \tau$ in (5.29) is of course an oversimplification. The relation (2.32), however, indicates that for the switching strategy to work in practice the choice of ξ has to be related to quantities as

$$\xi_1 = \left(\frac{\varepsilon}{\tau}\right)^{1/k}, \qquad \xi_2 = \left(\frac{\varepsilon}{\|(b-\hat{b})^T\mathcal{A}^{-1}\|\alpha_{\text{ref}}\tau}\right)^{1/k}.$$

In the expression for ξ_2 we have assumed that $\alpha = \alpha_{\text{ref}}$ and $\Delta = \tau$. In the case of HW-SDIRK(3)4 and $\tau = 0.01\,\varepsilon$ we have $\xi_1 = 100^{1/4} \approx 3.1$ and $\xi_2 = 2.5^{1/4} \approx 1.3$. In our simulations we have chosen $\xi = 2$.

The strategy proposed in [Nørsett and Thomsen, 1986b] makes a switch as soon as fixed point iteration fails to converge. Such a strategy calls for a rather large α_{ref} or poor convergence control. Otherwise it may happen that the controller manages to keep α below 1, and the switch to modified Newton iteration is never done.

We prefer the strategy suggested by Shampine, provided ξ is chosen in relation to τ/ε, α_{ref}, and the method parameters $\mathcal{A}$, b, and $\hat{b}$. Additional robustness is gained by requiring $h_r/h_\alpha > \xi$ for a few consecutive steps before doing the switch. Another possibility is to not require a fixed value on h_r/h_α to make the switch, but rather accumulate recent values on h_r/h_α and use that as a kind of "inefficiency measure". A switch is done when this measure indicates a too large inefficiency for using fixed point iteration.

Switching from modified Newton to fixed point. In the nonstiff case fixed point iteration is normally cheaper than modified Newton iteration. Most switching strategies therefore try to facilitate the switch from modified Newton to fixed point. To be able to decide if fixed point iteration converges or not we need to know the norm of the Jacobian, cf. (2.25) and (5.16). Some strategies recommend calculating the norm directly [Shampine, 1981], while others estimate it through the ratio between the residual and the displacement in the equation solver (2.36) [Nørsett and Thomsen, 1986b].

We have used the latter alternative with the β taken as the maximum over all iterations and all stages, cf. (2.36). For added robustness we require $\beta < 2$ for a few steps before doing the switch. In addition, at the switch the stepsize is reduced by the factor α_{ref} to assure good convergence.

Hysteresis. All switching strategies need to include some kind of hysteresis. Otherwise it may happen that the solver toggles between fixed point iteration and modified Newton iteration. Some common countermeasures are:

- Do not allow a switch until the switching condition has been fulfilled during a few consecutive steps (say 3).
- When switching from fixed point iteration to modified Newton iteration a switch back is blocked for (say) 5 steps in order to let the controller ramp up the stepsize and enter the region where $\beta > 2$ [Nørsett and Thomsen, 1987].
- When switching from modified Newton to fixed point iteration the old Jacobian is stored. Should a switch back occur within not too many steps then a new Jacobian need not be calculated to form the iteration matrix [Shampine, 1981].

EXAMPLE 5.8—Switching equation solver
Again return to the van der Pol problem and the setup in Example 5.6. Figure 5.13 demonstrates some aspects of the switching strategy described above. The number of integration steps is almost halved by switching to modified Newton iteration during the mildly stiff parts of the solution. A new iteration matrix was formed every integration step.

The integration starts using fixed point iteration. After the initial transient the stepsize soon gets restricted by convergence (steps 12–22) and the error estimate drops. At $t \approx 0.60$ (step 23) r has been sufficiently low for 3 steps and a switch to modified Newton is done. The controller ramps up the stepsize and soon $r \approx \varepsilon$. For simplicity a new iteration matrix is formed at every step, and as a consequence the convergence is very good.

When the integration approaches the fast state transition the stiffness drops. At $t \approx 6.7$ (step 44) the stiffness estimate β has been below 2 for 3 steps and a switch to fixed point iteration is done. At first the stepsize is restricted to assure convergence (steps 45–60), but during the transition at $8.5 < t < 9.4$ (steps 61–109) it is accuracy that determines the stepsize. As the integration proceeds out on the flat part of the solution, the stepsize is again restricted by convergence and a switch to modified Newton iteration is done at $t \approx 9.9$ (step 120). □

A Factorization/Evaluation Strategy for the Iteration Matrix

The last major component of the convergence control remaining to be discussed is the administration of the iteration matrix in the modified Newton iteration. Early implementations of stiff integration methods formed a new iteration matrix at every step. For large systems this may be the dominating part of the computations, and it soon became clear that large savings could be achieved by using the same iteration matrix during several integration steps.

The overall strategy is to use the stepsize to control the error and then update the iteration matrix so that good convergence is achieved with a small number of Jacobian evaluations and iteration matrix factorizations. The trade off between frequent changes to the iteration matrix on the one side, and poor convergence on the other, is delicate. Both extremes are expensive in terms of computation and the optimal balance depends on the integration method as well as on the problem size. Therefore there is not one algorithm that solves all problems, but rather

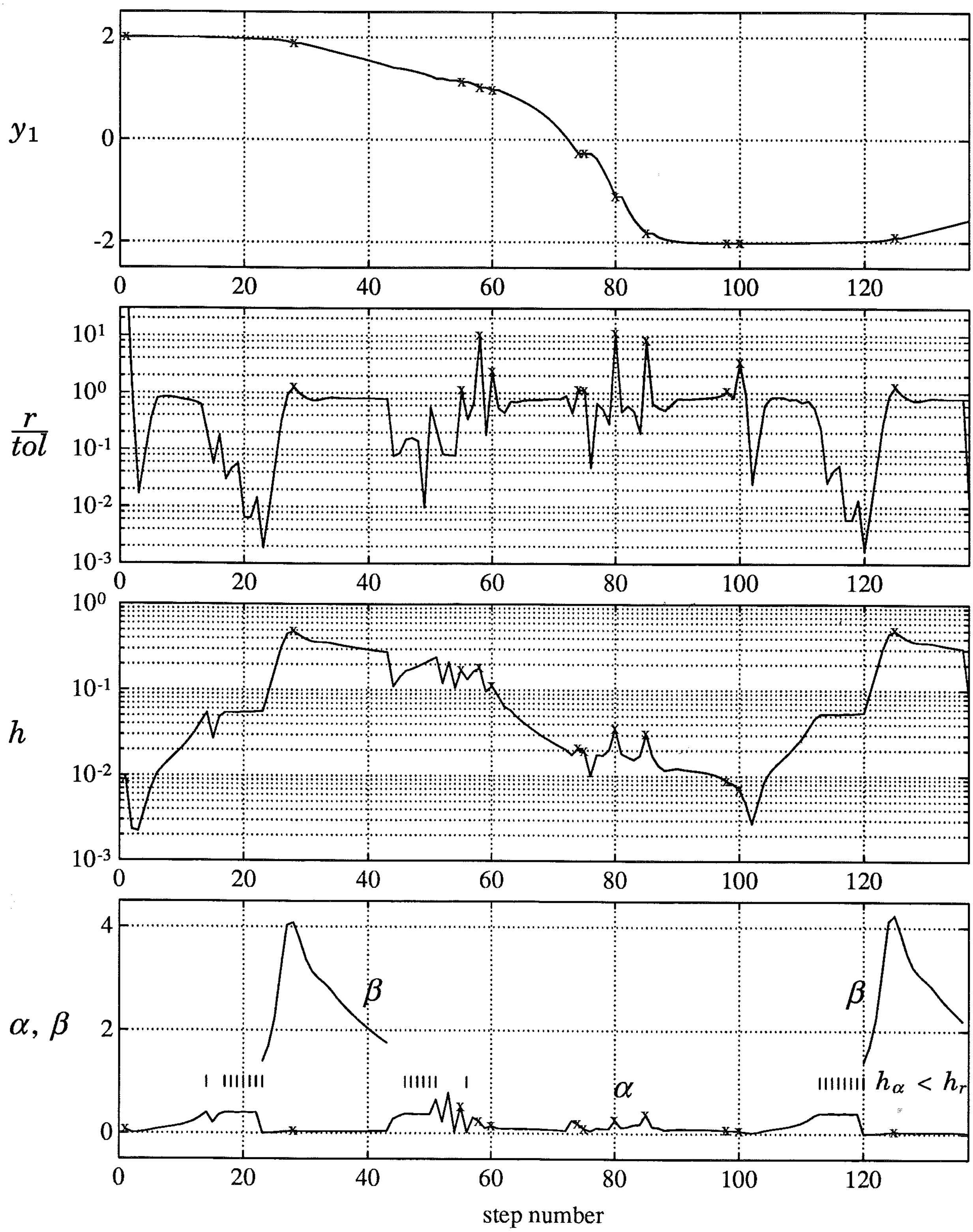


Figure 5.13 The upper plot depicts the solution component y_1 from the van der Pol problem in Example 5.8. The erratic look of the curve is due to the time-distortion introduced by plotting y_1 as a function of step number. The second plot shows the error estimate. Except at fast transitions and when convergence limits the stepsize, the error r is kept close to the set-point. The stepsize plot clearly shows the regions where the stepsize is restricted by convergence. The lower plot depicts the convergence estimate α and the stiffness estimate β. It also indicates (vertical bars) where the standard stepsize selection rule has been used due to $h_\alpha < h_r$ in the previous step (cf. *conv_restrict* in Listing 5.1).

strategies with tuning parameters that need to be adapted to different applications.

We advocate a control strategy that permits the stepsize to change at every integration step. Although the cost of factorizing the iteration matrix can be greatly reduced by storing the Jacobian in Hessenberg form [Enright, 1978; Varah, 1979; Hairer and Wanner, 1991, p. 132], it is still, normally, unacceptable to factorize at every integration step. To find a strategy that does not factorize at every stepsize change we turn to (5.23), i. e.

$$\alpha \lesssim \left|\frac{\delta h_n}{h'}\right| + \|J^{-1}\delta \bar{J}\|_H .$$

For a stiff differential equation we expect this bound to be rather sharp, i. e. $\nu \lesssim 1$ in (5.18), and any stepsize change will directly affect the convergence rate. We monitor the relative stepsize change since the last factorization, and when $\delta h_n/h'$ reaches a threshold α_{LU} a factorization is done. The strategy is preventive in the sense that we try to avoid convergence failures by factorizing whenever planning to do a stepsize change that is likely to jeopardize convergence. Should poor convergence be experienced despite $\delta h_n/h' < \alpha_{\mathrm{LU}}$, say $\alpha > \alpha_{\mathrm{JAC}}$, then this cannot be blamed on the stepsize changes. Instead, cf. (5.23), it must be due to a change in the Jacobian, and a reevaluation of J is called for. What values on α_{LU} and α_{JAC} that lead to efficient integration depend on the differential equation. Typical values lie in the range [0, 0.5], where small values correspond to regarding the iteration matrix operations as cheap.

The strategy is identical after accepted and rejected steps. A step rejection is caused by a too large error estimate, which will be dealt with by changing the stepsize and then retry. There is no reason for special considerations regarding the iteration matrix.

The situation is, however, somewhat different if we fail to get adequate convergence. If the iteration matrix used was based on an old Jacobian it is natural to compute a new one. The difficult question is whether the stepsize should be changed too. Here we will distinguish between three different situations. Suppose we experienced divergence. The analysis of Alexander [Alexander, 1991] points out that one integration step does not turn a rapidly converging process into a divergent one, unless some basic smoothness assumption is failing. He concludes that apart from forming a new Jacobian, the stepsize should also be changed. We change it based on (5.24).

The second situation is when the iteration is converging, but not fast enough to produce a solution within the allowed number of iterations. If the iteration matrix is based on an old Jacobian we calculate a new one but keep the stepsize unchanged. If on the other hand the iteration matrix is based on current data, we have to change the stepsize, and again this is done using (5.24).

The third situation is less common. It may happen that although the iteration matrix is based on current data, and the convergence rate is adequate ($\alpha \leq \alpha_{\text{ref}}$), the equation solver still fails to produce a solution within the allowed number of iterations. The reason is that the starting values for the iteration are too far from the fixed point. In this case we reduce the stepsize by the arbitrary factor 2. In all other situations we keep the stepsize unchanged.

The strategy regarding the iteration matrix can be summerized as in Listing 5.2. The listing also includes the strategy for choosing stepsize after a convergence failure, as well as the restriction of the stepsize when experiencing poor convergence in spite of an iteration matrix based on current data. The quality of the convergence estimate may be poor and as a result unreasonable stepsize changes should be discarded when calculating h_α.

EXAMPLE 5.9—Administrating the iteration matrix

To demonstrate the strategy regarding the iteration matrix we turn to the stiff van der Pol oscillator ($\sigma = 1000$). The problem was solved under the same conditions as in Example 5.1, i. e. we use modified Newton iteration throughout. This time, however, we do not form a new iteration matrix at every step.

Figure 5.14 depicts the result of the, rather arbitrary, choice $\alpha_{\text{LU}} = \alpha_{\text{JAC}} = 0.4$. The number of function evaluations is approximately doubled (7630 vs. 3678) compared to when forming a new iteration matrix at every step, cf. Example 5.1. The number of Jacobian evaluations and factorizations is, however, reduced from 339 down to 22 and 95, respectively. In this case the exact figures are of no importance; they only serve to demonstrate the very significant possibility of trading function evaluations with operations on the iteration matrix.

During the stiff part of the solution (steps 35–220) there are quite a few rejected steps (indicated with 'x', convergence failures are indicated with 'o'). Once (steps 35-36) there are even repeated rejected steps triggering the special restart strategy intended for the case of order reduction. The cause of all these rejected steps is the iteration error in the equation solver. For HW-SDIRK(3)4 the value of $\|(b - \hat{b})^T \mathcal{A}^{-1}\|$ is ap-

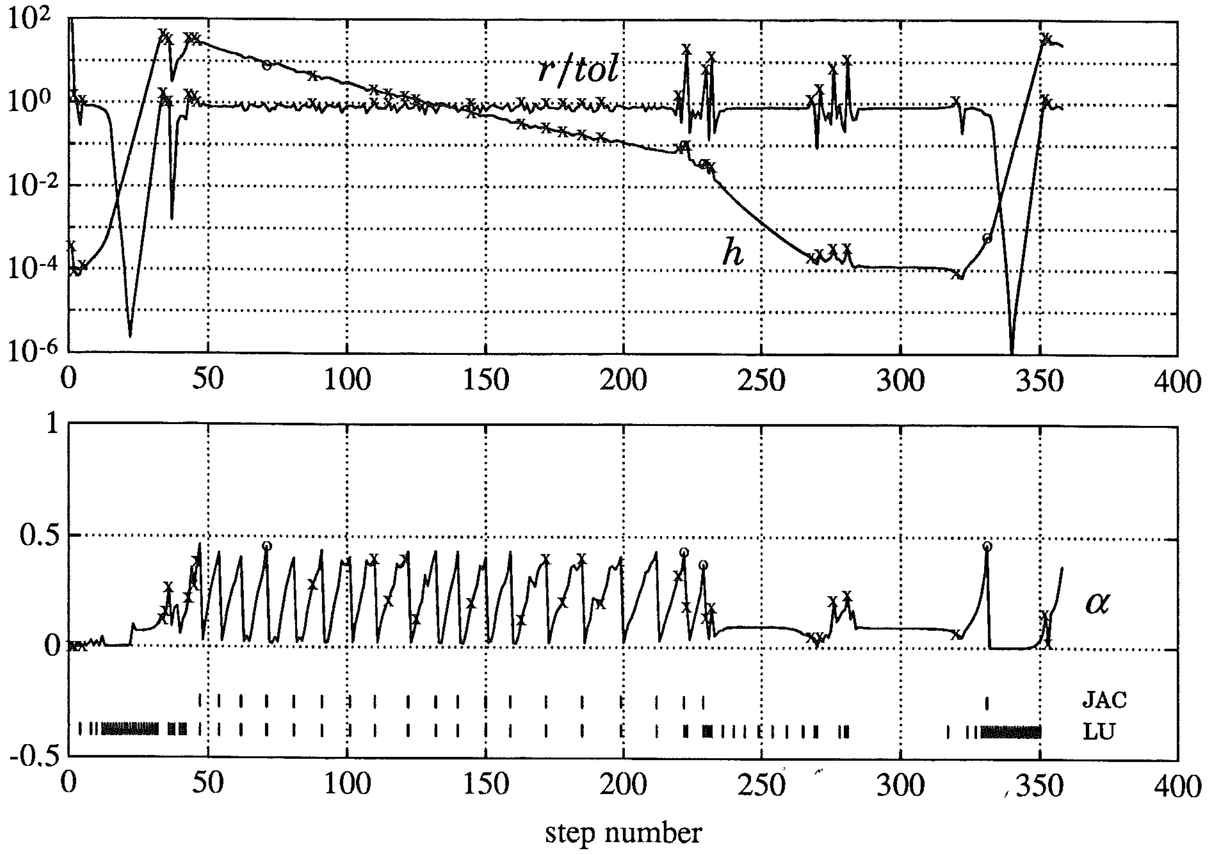


Figure 5.14 A demonstration of the strategy for administrating the iteration matrix discussed in Example 5.9. The parameters α_{LU} and α_{JAC} were set to 0.4. The upper plot shows the error and the stepsize, while the lower indicates when a Jacobian evaluation and/or factorization of the iteration matrix was done. The lower plot also depicts the estimate of the convergence rate. The rejected steps during the stiff part are caused by the iteration error.

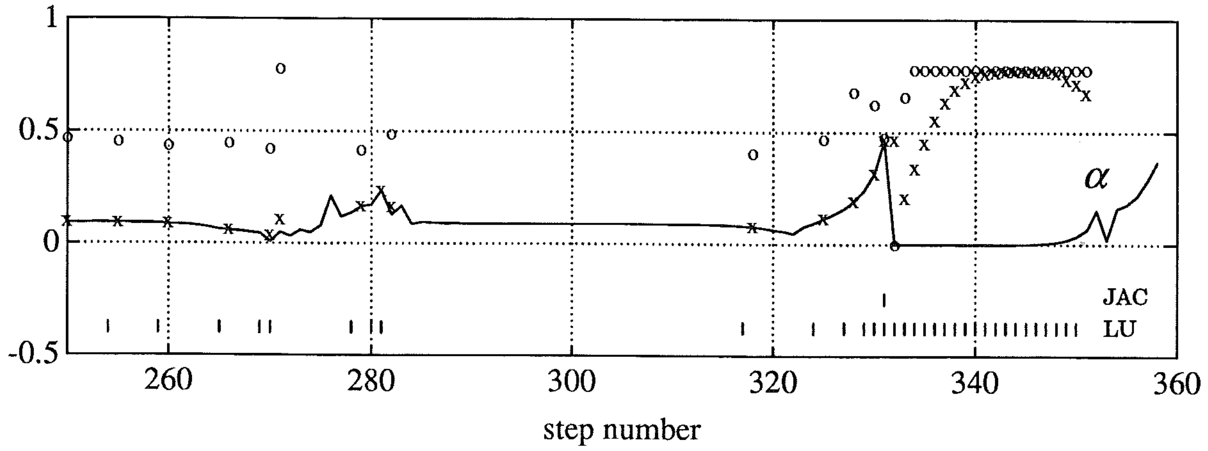


Figure 5.15 A segment of the lower plot in Figure 5.14. At every step where an operation was done on the iteration matrix we have plotted the corresponding value of $\delta h_n/h'$ ('o') and the convergence rate estimate α ('x') one would get if suppressing this operation. It is clear that the factorizations done during the nonstiff period (steps 230–330) are unnecessary ($\nu \ll 1$ in (5.18)). In contrast, every single factorization is important when ramping up the stepsize (steps 330–350) and entering the stiff region.

proximately 100, and with $\tau = 0.01\,\varepsilon$ the contribution from the iteration error is large enough to cause "noisy" variations in r. The rejections can be removed by reducing τ, cf. Figure 5.16 where $\tau = 0.005\,\varepsilon$.

As can be seen from Figure 5.14 there are many factorizations when the stepsize changes fast. When the changes are more moderate the factorizations occur less often, and it seems to be perfectly all right to allow small stepsize changes without corresponding factorizations. In addition, the Jacobian evaluations are quite infrequent and mainly needed during the stiff part. During the stiff part the convergence deteriorates at a faster rate than what could be explained with $\delta h_n/h'$. When α reaches $\alpha_{\mathrm{JAC}} = 0.4$ the Jacobian is evaluated and the convergence rate is immediately improved. The overall behavior agrees well with what could be expected.

From the behavior in Figure 5.14 one may suspect that the iteration matrix is factorized too often. The situation is clarified by Figure 5.15, which depicts a segment of the integration in Figure 5.14. Whenever an operation was to be done on the iteration matrix, we have calculated $\delta h_n/h'$ (plotted as 'o') and the convergence estimate that one would get if suppressing the operation (plotted as 'x'). It is clear that the factorizations during the nonstiff part of the solution (steps 230–330) are unnecessary. The factorizations are triggered by $\delta h_n/h' > \alpha_{\mathrm{LU}}$, but $\nu \ll 1$ in (5.18) and the stepsize change does not affect the convergence (the crosses lie on top of the α curve). The message is to change factorization strategy when the problem is nonstiff (β can be used to detect this situation), or more naturally, switch to fixed point iteration as discussed in Example 5.8, cf. Figure 5.17.

The situation is completely different when ramping up the stepsize and entering the stiff part of the solution (steps 10–30 and 330–350). As can be seen from the crosses in Figure 5.15 the convergence would become poor if omitting any of the factorizations. It is interesting to note how the convergence rate in case of suppressing the factorization (the crosses) approaches the value for $\delta h_n/h'$ (the rings) as the integration enters the stiff region. This is a consequence of $\nu \to 1$ as $\|h'J\| \to \infty$. In conclusion, we may say that in this case it is not a poor factorization strategy that causes the many factorizations, but rather a strict limit on how much the stepsize may increase in one integration step (the error controller will not allow a stepsize change that would increase the error by more than a factor of 10).

The value of τ was lowered to reduce the effect on r from the iter-

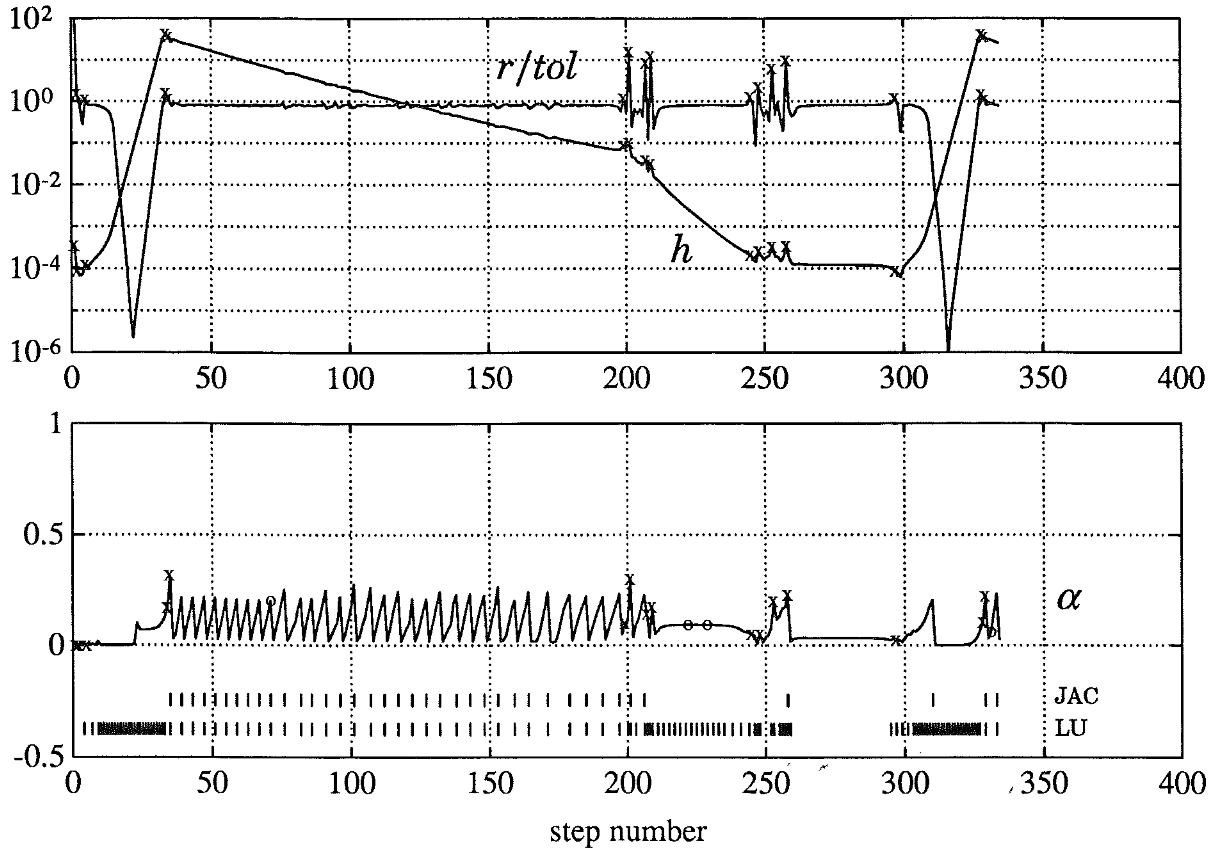


Figure 5.16 The setup here is similar to the one in Figure 5.14. The step rejections during the stiff part were removed by lowering the iteration error ($\tau = 0.005\,\varepsilon$). This, however, calls for better convergence, and α_{LU} and α_{JAC} were changed to 0.2. This increases the number of operations on the iteration matrix.

ation error. As expected the number of rejected steps was reduced, but as a side effect the number of steps with failed convergence. The reason is the limit on a maximum of 10 iterations in the equation solver. Normally, 3–5 iterations are done per stage, but when convergence is poor it may sometimes take more than 10 iterations to bring the iteration error below τ. To improve the situation the values on α_{LU} and α_{JAC} were lowered to 0.2, thus forcing a faster convergence. This removes the nonconverged steps and the result is depicted in Figure 5.16. The number of Jacobian evaluations doubled (41) and the number factorizations increased by approximately 30 % (129), while the number of function evaluations dropped (from 7630 to 5852) due to the improved convergence.

It is only necessary to use modified Newton iteration during the stiff parts of the solution. Figure 5.17 shows what happens when we introduce the previously discussed switching strategy, cf. Example 5.8, and use fixed point iteration during the nonstiff parts of the solution. □

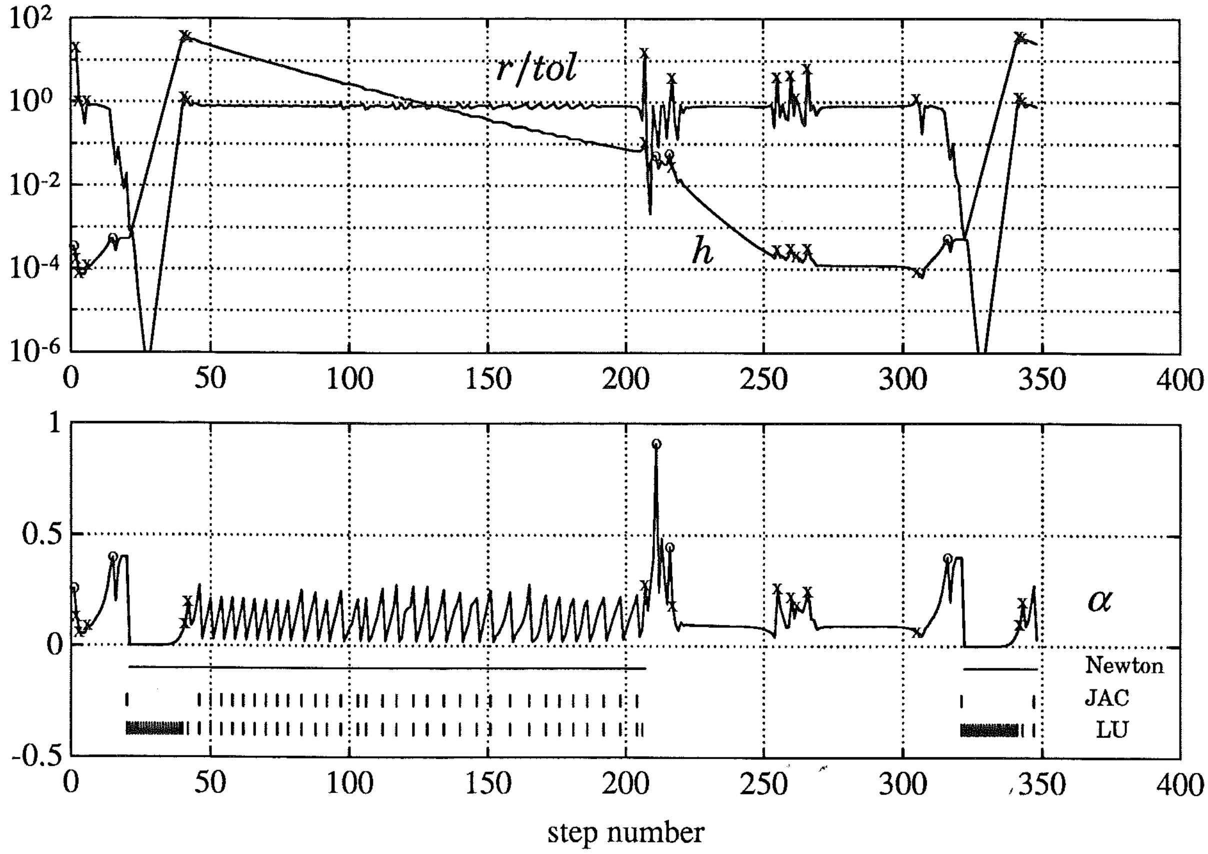


Figure 5.17 The setup for this figure is almost identical to the one in Figure 5.16. The only difference is that modified Newton iteration was used only during the stiff parts of the integration. The switching strategy from Example 5.8 was used.

5.4 The Complete Controller

The complete controller for an implicit Runge-Kutta method includes three main parts:

- An error controller that chooses the stepsize based on the required accuracy.
- A controller that supervises the convergence of the iteration in the equation solver, and terminates when the iteration error is acceptable.
- A convergence controller that chooses the most appropriate equation solver and administrates the iteration matrix so that convergence is obtained in an efficient way. If needed, it also restricts the stepsize not to jeopardize the convergence.

The code for the error controller is outlined in Listing 5.1. Based on data from previous steps the controller predicts changes in φ and thereby pro-

```
if iteration_successfully_completed then
    Calculate $h_r$, cf. Listing 5.1
    Calculate $h_\alpha$, cf. (5.24)
    if fresh_Jacobian and $\alpha > \alpha_{\text{ref}}$ then
        $h := \min(h_r, h_\alpha)$
    else
        $h := h_r$
    endif
    if $\alpha > \alpha_{\text{JAC}}$ then
        Form new Jacobian and factorize iteration matrix
        $h_{\text{LU}} := h$
    elseif $|h - h_{\text{LU}}|/h_{\text{LU}} > \alpha_{\text{LU}}$ then
        Factorize iteration matrix
        $h_{\text{LU}} := h$
    endif
else
    Calculate $h_\alpha$, cf. (5.24)
    if diverging then
        $h := h_\alpha$
    elseif fresh_Jacobian then
        if $\alpha > \alpha_{\text{ref}}$ then
            $h := h_\alpha$
        else
            $h := h/2$
        endif
    endif
    if not_fresh_Jacobian then
        Form new Jacobian
    endif
    Factorize iteration matrix
endif
```

Listing 5.2 An outline of the code needed to implement the evaluation and factorization strategy in modified Newton iteration. The stepsize choice after a step with convergence failure is also described. The parameters α_{JAC} and α_{LU}, govern when a new Jacobian is calculated and when the iteration matrix should be factorized. Their values affect the efficiency of the integration, and hence need to be chosen with care. The algorithm has to be augmented with various safety nets, e. g. discard unreasonably large stepsize changes, protect against underflow/overflow, etc.

```
if $iteration\_successfully\_completed$ then
    Calculate $h_r$, cf. Listing 5.1
    Calculate $h_\alpha$, cf. (5.24)
    $h := \min(h_r, h_\alpha)$
else
    Calculate $h_\alpha$, cf. (5.24)
    if $diverging$ then
        $h := h_\alpha$
    else
        $h := h/2$
    endif
endif
```

Listing 5.3 An outline of the code needed to coordinate the accuracy and convergence requirement on the stepsize when using fixed point iteration. The stepsize h_α is calculated from (5.24). Again, in a real implementation various safety nets have to be included, e.g. discard unreasonably large stepsize changes when calculating h_α, protect against underflow/overflow, etc.

duces a new stepsize that is more likely to achieve $r_{n+1} = \varepsilon$ than the stepsize suggested by the standard stepsize selection rule. The predictive controller may be considered as a general improvement over current stepsize selection rules, and we recommend using it no matter if the convergence controller impose limits on small stepsize changes or not. The controller includes logic that improves startup after rejected steps, which is especially important in cases demanding large stepsize decrease to bring a fast mode into the asymptotic region of the integration method. This situation may arise due to poor method properties. The correct solution in this case is to change methods, i.e. use a stiffly accurate integration method, instead of trying to improve the control.

The choice of strategy for the second part, the supervision of the iteration error, is fairly standard, and was described in Section 5.2. The end of that section gives specific details about the implementation. The lack of novelty in our choice of control strategy does not imply supremity of current algorithms, but rather demonstrates an uncertainty. The dynamic relation between iteration error and integration error is not well understood, and on-going research may very well force a revision of this part of the control strategy.

The third part, the convergence controller, is important mainly for efficiency reasons. It should secure good convergence with as little computations as possible. Listing 5.2 describes the strategy used in case of modified Newton iteration. The strategy for fixed point iteration is simpler, and is shown in Listing 5.3. The strategy for changing equation solver has to be added to what is described in Listings 5.1–5.3. This is straightforward: after every accepted step we simply check whether the switching conditions, cf. Section 5.3 are fulfilled, and if this is the case a switch is made.

Controller Parameters

The complete controller includes several parameters that have to be set. The parameters k_1 and k_2 in the error controller are fairly independent of integration method and differential equation. The choice $k_1 = k_2 = 1$ results in good overall performance. As set-point for the error control we recommend $\varepsilon = 0.8\,tol$, although not having done as extensive tests as for the explicit case.

Turning to the parameters regarding the iteration error and the convergence control it is no longer possible to make the choice independently of integration method and differential equation. Consider the choice of τ. As noted in [Hairer and Wanner, 1991, p. 131] its value affects the efficiency of the method. In addition, a large value on τ results in a nonsmooth component in r that impairs the performance of the stepsize selection rule. The choice $\tau = 0.005\,\varepsilon$ works well for HW-SDIRK(3)4. It is, however, not possible to directly translate this value to other methods, cf. (2.33). A different method is likely to call for a different value of τ.

The parameter values in the convergence controller, i. e. α_{JAC} and α_{LU}, can be used to optimize the performance of the integration method for a class of differential equations. The decisions concerning the administration of the iteration matrix depend on the relative cost of evaluating the Jacobian, of reducing it to Hessenberg form, of factoring the iteration matrix, and of taking one iteration in the equation solver. If forming the iteration matrix is cheap, this should be done often in order to maintain good convergence, thus minimizing the total number of iterations. If, on the other hand, the iteration matrix operations are expensive compared to one iteration, we should instead try to minimize the number of changes of the iteration matrix. For optimal performance a careful tuning analysis will be required.

Despite the complexity of the tuning of α_{JAC} and α_{LU} it is possible to give some general recommendations. The maximum reasonable value of α_{ref} (say 0.5), cf. Figure 5.12, sets an upper limit on α_{JAC} and α_{LU}.

It is of no use having an iteration matrix strategy that accepts a convergence rate that is worse than what is known to be efficient. Large values on α_{JAC} and α_{LU} are only of interest when the administration of the iteration matrix is expensive compared to one iteration, i. e. for problems with many states and a full Jacobian. For small and medium sized problems it is more reasonable to have (say) $\alpha_{\text{JAC}} = \alpha_{\text{LU}} = 0.1$ or 0.2. Having $\alpha_{\text{JAC}} > \alpha_{\text{LU}}$ is motivated when the Jacobian is stored on special form, so that a factorization is relatively cheap compared to the Jacobian evaluation. The choice of α_{JAC} and α_{LU} also affects the average number of iterations needed in the equation solver. A strategy that accepts slow convergence thus has to be accompanied with fairly large upper limit on the number of iterations.

6

Conclusions

This thesis has presented a feedback control viewpoint of the algorithms for supervision and parameter adjustment in numerical integration methods. It provides a different, but rewarding, perspective on some of the central issues in the implementation of an integration method. The methodology can be applied to any integration method. The detailed treatment has, however, been focused on Runge-Kutta methods. We have derived new controllers both for the explicit and the implicit case. They give improved overall performance at little extra expense.

Methodology

Before turning to specific results there are a few general aspects of our approach that we would like to stress:

- The approach taken in this thesis is traditional viewed as an application of feedback control theory. It is, however, novel from the numerical analysis point of view. The traditional design of stepsize adjustment rules and supervision algorithms for numerical integration methods is based on heuristic principles and asymptotic arguments. Our approach provides a framework where the problem can be addressed in a systematic and structured way.
- We have implemented the integration methods, control strategies etc., in a common software framework. This allows for fair comparative studies where parameters, strategies, etc., are varied, one at

a time, under controlled experimental conditions.

- A small number of carefully selected test problems have been used. They suffice for developing, testing, and verifying the control strategies. The main objective has not been a performance/efficiency study, but rather an investigation of how to design a coherent strategy for the control of numerical ODE solvers.

Separating Process and Controller

A conceptually important step in our approach is the separation of process (numerical integration method) and controller. We would like to stress the benefits of this separation. A reasonably complete controller, cf. Chapters 4 and 5, becomes rather involved, cf. Figure 5.1. Treating it as a separate object makes both the controller design and the implementation more straightforward. The separation is a significant advantage when tuning the controller for optimal performance, as well as when trying to confirm that the code really implements the intended algorithms.

Improved Process Models

A first step in the control analysis is to obtain models that accurately describe the input-output relations of the process. An important relation is the one between stepsize and error estimate. Our modeling, cf. Chapter 3, has given the following insight:

- The static asymptotic stepsize-error model normally used is inappropriate in many situations. Dynamic models are needed.
- It is not possible to capture the stepsize-error behavior in one single model. We have distinguished between three cases: the asymptotic region, the nonasymptotic region (implicit methods), and stepsize restricted by numerical stability (explicit methods).
- In the asymptotic region we have $r_{n+1} = \varphi_n h_n^k$. The differential equation influences the stepsize-error relation only through an estimate φ of the norm of the principal error function. By modeling the variations in φ we arrive at better predictions of the error resulting from a specific stepsize choice.
- In the nonasymptotic region the error relates to the stepsize through a different exponent than the value k normally assumed. In the case of repeated rejected steps we estimate the value of the exponent and use it in our model of the stepsize-error relation.

- When the stepsize is restricted by numerical stability the stepsize-error relation behaves very differently compared to the normal case. We have derived a simple dynamic model that captures this behavior. The parameters in the model depend on the coefficients of the integration method and the dominating eigenvalues of the differential equation.

The models provide an accurate description of the stepsize-error relation. They have been verified by system identification techniques as well as in numerical tests.

To be able to control the equation solver in an implicit method we need a model that relates the convergence rate of the iteration to the process input variables. A new model that describes the relationship between the convergence rate and changes in the stepsize and the Jacobian has been derived, cf. Chapter 5. The model has a structure that makes it directly applicable in the design of the administration strategy for the iteration matrix.

The Feedback Control Analysis

The new process models have been combined with the standard controller, allowing an analysis of the resulting closed loop system. The analysis provides insight and suggests possible improvements to the controller. In particular we have noted the following:

- The standard stepsize selection rule can be interpreted as a pure integrating controller. This immediately suggests generalizations to controllers with better stabilizing properties, e. g. PI or PID control.
- The standard controller is based on the static asymptotic process model. The situations where the controller performs poorly are directly related to the static model not being able to accurately describe the process behavior.
- The standard choice of controller parameters is based on the idea of instant correction (deadbeat control). Whenever there is a deviation from the desired behavior, an attempt is made to fully correct this in the following integration step. This is merely one possibility of many when deciding the dynamics of the control loop.

Improved Control Algorithms and Strategies

Based on the knowledge we have acquired from the process modeling and the feedback control analysis we have derived new and improved control algorithms, cf. Chapters 4 and 5.

Explicit Runge-Kutta methods. The main complication in the choice of stepsize for an explicit Runge-Kutta method is the change of process dynamics when numerical stability restricts the stepsize. We have designed a controller that can handle this and still retain good performance in the asymptotic region. The controller is the equivalent of a PI controller, i. e.

$$h_{n+1} = \left(\frac{\varepsilon}{r_{n+1}}\right)^{k_I} \left(\frac{r_n}{r_{n+1}}\right)^{k_P} h_n. \tag{6.1}$$

Choosing the controller parameters as $k_I = 0.3/k$ and $k_P = 0.4/k$ provides good performance both for the standard case, and for the case when numerical stability restricts the stepsize.

Implicit Runge-Kutta methods. In most implicit methods the stepsize will not be limited by numerical stability. The controller can instead be chosen to gain other properties. We have used the model derived for the evolution of the principal error function to design a controller that tracks changes in the differential equation and its solution. This controller takes the form

$$h_{n+1} = \frac{h_n}{h_{n-1}} \left(\frac{\varepsilon}{r_{n+1}}\right)^{k_2/k} \left(\frac{r_n}{r_{n+1}}\right)^{k_1/k} h_n. \tag{6.2}$$

It achieves substantially improved error control. Choosing the controller parameters as $k_1 = k_2 = 1$ provides good overall performance.

The efficiency of an implicit integration method is closely related to the control strategy used for its equation solver. Based on the model of the equation solver dynamics we have designed a new strategy that:

- allows for "continuous" stepsize variations, and hence smoother error control, without excessive operations on the iteration matrix,
- implements a robust strategy for switching between fixed-point iteration and modified Newton iteration,
- anticipates poor convergence by appropriately scheduling the operations on the iteration matrix, thus reducing convergence failures, and

- achieves efficient integration also when convergence restricts the stepsize by an optimal choice of the set-point α_{ref} for the convergence rate.

Extensions and Future Research

There are several directions in which the results of this thesis can be extended. We have mainly studied Runge-Kutta methods, but the methodology used is not limited to this case. It is no doubt possible to derive models for other types of integration methods, e. g. multistep methods. Once a model is obtained it can be used to analyze and improve the controller.

Another interesting question is to what extent the controller should depend on the differential equation. We have been able to derive general error controllers which do not rely on specific information about the differential equation. This is not the case for the control strategy concerning the equation solver. For efficiency this strategy should depend on the properties of the problem, e. g. problem size, sparse/nonsparse or banded Jacobian, cost of function evaluation compared to Jacobian evaluation and factorization cost, etc. A controller that depends on the process is quite common in control theory, e. g. adaptive controllers, self-tuning controllers, gain-scheduling, etc. It is probably not possible to directly apply any of these techniques, but the underlying ideas could be exploited to obtain a controller that will adapt its strategy to the structure of the differential equation being solved.

Many numerical methods, e. g. partial differential equation solvers, optimization routines, etc., have a structure similar to that of an integration method. They all need supervision and sensible parameter choice to function properly. It may be beneficial to investigate to what extent the feedback control analogy could be used to improve efficiency and accuracy also for these types of numerical methods.

7

References

ALEXANDER, R. (1991): “The modified Newton method in the solution of stiff ordinary differential equations.” *Mathematics of Computation*, **57**, pp. 673–701.

ARÉVALO, C. (1992): Private communication.

ARMSTRONG, N. (1969): “Journey to the moon.” Pickwick International Inc. Ltd., London.

BICKART, T. A. (1977): “An efficient solution process for implicit Runge-Kutta methods.” *SIAM Journal on Numerical Analysis*, **14:6**, pp. 1022–1027.

BOYD, S. P. and C. H. BARRAT (1991): *Linear Controller Design: Limits of Performance.* Prentice Hall, Englewood Cliffs, New Jersey.

BRENAN, K. E., S. L. CAMPBELL, and L. R. PETZOLD (1989): *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* North-Holland.

BURRAGE, K. (1978): “A special family of Runge-Kutta methods for solving stiff differential equations.” *BIT (Nordisk Tidskrift för Informationsbehandling)*, **18**, pp. 22–41.

BURRAGE, K., J. C. BUTCHER, and F. H. CHIPMAN (1980): “An implementation of singly-implicit Runge-Kutta methods.” *BIT (Nordisk Tidskrift för Informationsbehandling)*, **20**, pp. 326–340.

BUTCHER, J. C. (1976): "On the implementation of implicit Runge-Kutta methods." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **16**, pp. 237–240.

BUTCHER, J. C. (1987): *The Numerical Analysis of Ordinary Differential Equations*. John Wiley & Sons.

BYRNE, G. D. and A. C. HINDMARSH (1987): "Stiff ODE solvers: A review of current and coming attractions." *Journal of Computational Physics*, **70:1**, pp. 1–62.

CARROLL, L. (1897): *Alice's Adventures in Wonderland*. Macmillan 6s. standard edition, 86th Thousand.

CASH, J. R. (1979): "Diagonally implicit Runge-Kutta formulae with error estimators." *Journal of the Institute of Mathematics and its Applications*, **24**, pp. 293–301.

CASH, J. R. (1985): "Stiff methods." In AIKEN, Eds., *Stiff Computation*, chapter 3, pp. 1–16. Oxford University Press. International Conference on Stiff Computation, April 12–14, 1982, Park City, Utah.

CHAR, B. W., K. O. GEDDES, G. H. GONNET, M. B. MONAGAN, and S. M. WATT (1988): *Maple – Reference Manual*. Symbolic Computation Group, Department of Computer Sciences, University of Waterloo, Waterloo, Ontario, Canada, fifth edition.

DAHLQUIST, G. (1963): "A special stability problem for linear multistep methods." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **3**, pp. 27–43.

DAHLQUIST, G., Å. BJÖRK, and N. ANDERSSON (1974): *Numerical Methods*. Prentice-Hall, Englewood Cliffs, New Jersey.

DORMAND, J. R. and P. J. PRINCE (1980): "A family of embedded Runge-Kutta formulae." *Journal of Computational and Applied Mathematics*, **6:1**, pp. 19–26.

DORMAND, J. R. and P. J. PRINCE (1986): "Runge-Kutta triples." *Computers and Mathematics with Applications A*, **12A:9**, pp. 1007–1017.

EHLE, B. L. (1969): "On Padé approximations to the exponential function and *A*-stable methods for the numerical solution of initial value problems." Research Report CSRR 2010, Dept. AACS, University of Waterloo, Ontario, Canada.

ENRIGHT, W. H. (1978): "Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations."

ACM Transactions on Mathematical Software, **4:2**, pp. 127–136.

ENRIGHT, W. H. (1989a): "Analysis of error control strategies for continuous Runge-Kutta methods." *SIAM Journal on Numerical Analysis*, **26:3**, pp. 588–599.

ENRIGHT, W. H. (1989b): "A new error-control for initial value solvers." *Applied Mathematics and Computation*, **31**, pp. 288–301.

ENRIGHT, W. H., T. E. HULL, and B. LINDBERG (1975): "Comparing numerical methods for stiff systems of ODE:s." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **15**, pp. 10–48.

EULER, L. (1768): *Institutionum Calculi Integralis.* Volumen Primum, Opera Omnia vol. XI.

FEHLBERG, E. (1969): "Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems." Technical Report 315, NASA. Extract published in Computing vol 6, pp. 61–71.

FRANKLIN, G. F., J. D. POWELL, and A. EMAMI-NAEINI (1986): *Feedback Control Systems.* Addison-Wesley.

FRANKLIN, G. F., J. D. POWELL, and M. L. WORKMAN (1990): *Digital Control of Dynamic Systems.* Addison-Wesley, second edition.

GEAR, C. W. (1971): *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, New Jersey.

GUSTAFSSON, K. (1991): "Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods." *ACM Transactions on Mathematical Software*, **17:4**, pp. 533–554.

GUSTAFSSON, K. (1992): "An object-oriented framework for the implementation of numerical integration methods." Technical report, Department of Automatic Control, Lund Institute of Technology. In preparation.

GUSTAFSSON, K., M. LUNDH, and G. SÖDERLIND (1988): "A PI stepsize control for the numerical solution of ordinary differential equations." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **28:2**, pp. 270–287.

HAIRER, E., C. LUBICH, and M. ROCHE (1989): *The Numerical Solution of Differential-Algebraic System by Runge-Kutta Methods*, volume 1409 of *Lecture Notes in Mathematics.* Springer-Verlag.

HAIRER, E., S. P. NØRSETT, and G. WANNER (1987): *Solving Ordinary*

Differential Equations I – Nonstiff Problems, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag.

Hairer, E. and G. Wanner (1991): *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag.

Hall, G. (1985): "Equilibrium states of Runge-Kutta schemes: Part I." *ACM Transactions on Mathematical Software*, **11:3**, pp. 289–301.

Hall, G. (1986): "Equilibrium states of Runge-Kutta schemes: Part II." *ACM Transactions on Mathematical Software*, **12:3**, pp. 183–192.

Hall, G. and D. J. Higham (1988): "Analysis of stepsize selection schemes for Runge-Kutta codes." *IMA Journal of Numerical Analysis*, **8**, pp. 305–310.

Higham, D. J. (1989a): "Robust defect control with Runge-Kutta schemes." *SIAM Journal on Numerical Analysis*, **26:5**, pp. 1175–1183.

Higham, D. J. (1989b): "Runge-Kutta defect control using Hermite-Birkhoff interpolation." Technical Report 221/89, Department of Computer Science, University of Toronto. Submitted to SIAM Journal on Scientific and Statistical Computing.

Higham, D. J. (1990): "Global error versus tolerance for explicit Runge-Kutta methods." Technical Report 233/90, Department of Computer Science, University of Toronto.

Higham, D. J. and G. Hall (1987): "Embedded Runge-Kutta formulae with stable equilibrium states." NA report 140, Department of Mathematics, University of Manchester.

Higham, D. J. and G. Hall (1989): "Runge-Kutta equilibrium theory for a mixed relative/absolute error measure." Technical Report 218/89, Department of Computer Science, University of Toronto.

Hindmarsh, A. C. (1983): "ODEPACK, a systematized collection of ODE solvers." In Stepleman, Eds., *Scientific Computing*, pp. 55–64. North-Holland, Amsterdam.

Houbak, N., S. P. Nørsett, and P. G. Thomsen (1985): "Displacement or residual test in the application of implicit methods for stiff problems." *IMA Journal of Numerical Analysis*, **5**, pp. 297–305.

Jackson, K. R., A. Kværnø, and S. P. Nørsett (1992): "The order of Runge-Kutta formulas when an iterative method is used to compute the internal stage values." Technical report, Institute for Mathematics, University of Trondheim, Norway. In preparation.

KRAAIJEVANGER, J. F. B. M. and J. SCHNEID (1991): "On the unique solvability of the Runge-Kutta equations." *Numerische Mathematik*, **59**, pp. 129–157.

KVÆRNØ, A. (1988): *Runge-Kutta Methods for the Numerical Solution of Differential-Algebraic Equations of Index 1.* PhD thesis, Institute for Mathematics, University of Trondheim, Norway.

LJUNG, L. (1987): *System Identification: Theory for the User.* Prentice-Hall.

THE MATHWORKS INC., Cochituate Place, 24 Prime Parkway, Natick, MA 01760, USA (1990): *PRO-MATLAB – User's Guide.*

VON NEUMANN, J. (1951): "Eine Spektraltheorie für allgemeine Operatoren eines unitären Raumes." *Mathematische Nachrichten*, **4**, pp. 258–281.

NEVANLINNA, O. (1984): "Matrix valued versions of a result of von Neumann with an application to time discretization." Technical Report HTKK-MAT-A224, Institute of Mathematics, Helsinki University of Technology.

NØRSETT, S. P. and P. G. THOMSEN (1984): "Embedded SDIRK-methods of basic order three." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **24**, pp. 634–646.

NØRSETT, S. P. and P. G. THOMSEN (1986a): "Local error control in SDIRK-methods." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **26**, pp. 100–113.

NØRSETT, S. P. and P. G. THOMSEN (1986b): "Switching between modified Newton and fix-point iteration for implicit ODE-solvers." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **26**, pp. 339–348.

NØRSETT, S. P. and P. G. THOMSEN (1987): *User Guide for SIMPLE – a stiff system solver.*

NØRSETT, S. P. and P. G. THOMSEN (1990): "SIMPLE – the FORTRAN code."

ORTEGA, J. M. and W. C. RHEINBOLDT (1970): *Iterative Solution of Nonlinear Equations in Several Variables.* Computer Science and Applied Mathematics. Academic Press, New York.

PRINCE, P. J. and J. R. DORMAND (1981): "High order embedded Runge-

Kutta formulae." *Journal of Computational and Applied Mathematics*, **7:1**, pp. 67–76.

PROTHERO, A. and A. ROBINSON (1974): "On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations." *Mathematics of Computation*, **28:125**, pp. 145–162.

SHAMPINE, L. F. (1975): "Stiffness and nonstiff differential equation solvers." In COLLATZ, Eds., *Numerische Behandlung von Differentialgleichungen*, pp. 287–301. Birkhäuser Verlag, Basel.

SHAMPINE, L. F. (1977): "Local error control in codes for ordinary differential equations." *Applied Mathematics and Computation*, **3**, pp. 189–210.

SHAMPINE, L. F. (1980): "Implementation of implicit formulas for the solution of ODE's." *SIAM Journal on Scientific and Statistical Computing*, **1:1**, pp. 103–118.

SHAMPINE, L. F. (1981): "Type-insensitive ODE codes based on implicit *A*-stable formulas." *Mathematics of Computation*, **36:154**, pp. 499–510.

SHAMPINE, L. F. (1985a): "Interpolation for Runge-Kutta methods." *SIAM Journal on Numerical Analysis*, **22:5**, pp. 1014–1027.

SHAMPINE, L. F. (1985b): "Introduction to stiffness." In AIKEN, Eds., *Stiff Computation*, chapter 1, pp. 1–16. Oxford University Press. International Conference on Stiff Computation, April 12–14, 1982, Park City, Utah.

SHAMPINE, L. F. (1986): "Some practical Runge-Kutta formulas." *Mathematics of Computation*, **46:173**, pp. 135–150.

SHAMPINE, L. F. (1991): "Diagnosing stiffness for Runge-Kutta methods." *SIAM Journal on Scientific and Statistical Computing*, **2:12**, pp. 260–272.

SHAMPINE, L. F. and C. W. GEAR (1979): "A user's view of solving stiff ordinary differential equations." *SIAM Review*, **21:1**, pp. 1–17.

SKEEL, R. D. (1986): "Thirteen ways to estimate global error." *Numerische Mathematik*, **48**, pp. 1–20.

STETTER, H. J. (1973): *Analysis of Discretization Methods for Ordinary Differential Equations*. Springer-Verlag, Berlin, Heidelberg, New York.

STETTER, H. J. (1976): "Considerations concerning a theory for ODE-solvers." In BURLISH *et al.*, Ed., *Numerical Treatment of Differential Equations: Proc. Oberwolfach*, volume 631 of *Lecture Notes in Mathematics*, pp. 188–200, Berlin. Springer.

STETTER, H. J. (1980): "Tolerance proportionality in ODE-codes." In MÄRZ, Eds., *Proceedings of the Second Conference on Numerical Treatment of Ordinary Differential Equations*, volume 32 of *Seminarberichte*, Berlin. Humboldt University.

SÖDERLIND, G. (1984a): "An error bound for fixed-point iterations." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **24**, pp. 391–393.

SÖDERLIND, G. (1984b): "On nonlinear difference and differential equations." *BIT (Nordisk Tidskrift för Informationsbehandling)*, **24**, pp. 667–680.

SÖDERLIND, G. (1986a): "Bounds on nonlinear operators in finite-dimensional banach spaces." *Numerische Mathematik*, **50**, pp. 27–44.

SÖDERLIND, G. (1986b):. "Folklore and fudge factors in the implementation of fixed-point and Newton iterations for nonlinear ODEs.". Talk presented at "Workshop on the Numerical Solution of Stiff ODEs", Sept 15–17, NTH, Trondheim, Norway.

SÖDERLIND, G. (1987):. "Numerical analysis of ordinary differential equations.". Lecture Notes from a course given 1986–1987 at Lund Institute of Technology.

SÖDERLIND, G. (1991): Private communication.

SÖDERSTRÖM, T. and P. STOICA (1989): *System Identification*. Prentice-Hall International.

VARAH, J. M. (1979): "On the efficient implementation of implicit Runge-Kutta methods." *Mathematics of Computation*, **33:146**, pp. 557–561.

VERNER, J. H. (1978): "Explicit Runge-Kutta mehtods with estimates of the local truncation error." *SIAM Journal on Numerical Analysis*, **15:4**, pp. 772–790.

WATTS, H. A. (1982): "Starting step size for an ODE solver." *Journal of Computational and Applied Mathematics*, **9**, pp. 177–191.

WATTS, H. A. (1984): "Step size control in ordinary differential equation solvers." *Transactions of the Society for Computer Simulation*, **1:1**, pp. 15–25.

ZONNEVELD, J. A. (1964): *Automatic Numerical Integration*. Math. centre tracts no. 8, Matematisch Centrum, Amsterdam.

ÅSTRÖM, K. J. and T. HÄGGLUND (1988): *Automatic Tuning of PID Controllers.* Instrument Society of America.

ÅSTRÖM, K. J. and B. WITTENMARK (1990): *Computer Controlled Systems – Theory and Design.* Prentice-Hall, Englewood Cliffs, New Jersey, second edition.

A

Some Common Runge-Kutta Methods

This is a short summary of the main Runge-Kutta methods used and/or mentioned in the text. Many of the methods include an imbedded error estimator. We use the notation name$(p_1)p_2$, where name is the name of a Runge-Kutta method and p_1, p_2 are the orders of the two formulas. The formula without parenthesis is the one recommended for solution update.

In the methods with embedded error estimator we let y refer to the low order formula and $\hat{y}$ to the high order formula. The rational/polynomial functions $P(z)$ and $\hat{P}(z)$, cf. (2.18) and (2.19), are the updating formulas that result when applying the method to the linear test equation, i. e. $\dot{y} = \lambda y$, $z = h\lambda$. They relate to y and $\hat{y}$, respectively. The rational/polynomial function $E(z) = P(z) - \hat{P}(z)$, cf. (2.19), is the error estimate for the same equation.

The methods are given in terms of their Butcher tableaux, i. e.

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \dots & a_{1s} \\
c_2 & a_{21} & a_{22} & \dots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\
\hline
y & b_1 & b_2 & \dots & b_s \\
\hline
\hat{y} & \hat{b}_1 & \hat{b}_2 & \dots & \hat{b}_s
\end{array}$$

where

$$\begin{aligned}
\dot{Y}_i &= f(t_n + c_i h_n, y_n + h_n \sum_{j=1}^{s} a_{ij} \dot{Y}_j), \qquad i = 1 \dots s \\
y_{n+1} &= y_n + h_n \sum_{j=1}^{s} b_j \dot{Y}_j \\
\hat{y}_{n+1} &= y_n + h_n \sum_{j=1}^{s} \hat{b}_j \dot{Y}_j \\
\hat{e}_{n+1} &= y_{n+1} - \hat{y}_{n+1} \\
t_{n+1} &= t_n + h_n.
\end{aligned}$$

A.1 Explicit Methods

Explicit Euler

Explicit Euler [Euler, 1768; Hairer *et al.*, 1987, p. 32] is the simplest explicit method. It is of first order and does not include an error estimator.

$$\begin{array}{c|c}
0 & 0 \\
\hline
y & 1
\end{array} \qquad P(z) = 1 + z$$

Modified Euler

Modified Euler is a second order method without error estimator.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline y & \frac{1}{2} & \frac{1}{2} \end{array} \qquad P(z) = 1 + z + \tfrac{1}{2}z^2$$

Midpoint Method

The midpoint method [Hairer *et al.*, 1987, p. 131] is a second order method without error estimator.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline y & 0 & 1 \end{array} \qquad P(z) = 1 + z + \tfrac{1}{2}z^2$$

RKF1(2)

RKF1(2) [Fehlberg, 1969; Hairer *et al.*, 1987, pp. 174–175] is one of the simplest methods including an embedded error estimator. The method is a combination of explicit Euler and modified Euler.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline y & 1 & 0 \\ \hline \hat{y} & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{aligned} P(z) &= 1 + z \\ \hat{P}(z) &= 1 + z + \tfrac{1}{2}z^2 \\ E(z) &= -\tfrac{1}{2}z^2 \end{aligned}$$

RKF2(3)

RKF2(3) is an embedded Runge-Kutta method due to Fehlberg [Fehlberg, 1969; Hairer *et al.*, 1987, pp. 169–170].

$$
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
\frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\
\hline
y & \frac{1}{2} & \frac{1}{2} & 0 \\
\hline
\hat{y} & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}
\end{array}
$$

$$
\begin{aligned}
P(z) &= 1+z+\tfrac{1}{2}z^2 \\
\hat{P}(z) &= 1+z+\tfrac{1}{2}z^2+\tfrac{1}{6}z^3 \\
E(z) &= -\tfrac{1}{6}z^3
\end{aligned}
$$

RKF(2)3B

RKF2(3)B is an alternative order 2/3 method derived by Fehlberg [Fehlberg, 1969; Hairer *et al.*, 1987, pp. 169–170]. Its error coefficients are more than 100 times smaller than the ones of RKF2(3).

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\
\frac{27}{40} & \frac{-189}{800} & \frac{729}{800} & 0 & 0 \\
1 & \frac{214}{891} & \frac{1}{33} & \frac{650}{891} & 0 \\
\hline
y & \frac{214}{891} & \frac{1}{33} & \frac{650}{891} & 0 \\
\hline
\hat{y} & \frac{41}{162} & 0 & \frac{800}{1053} & \frac{-1}{78}
\end{array}
$$

$$
\begin{aligned}
P(z) &= 1+z+\tfrac{1}{2}z^2+\tfrac{117}{704}z^3 \\
\hat{P}(z) &= 1+z+\tfrac{1}{2}z^2+\tfrac{1}{6}z^3-\tfrac{3}{1408}z^4 \\
E(z) &= -\tfrac{1}{2112}z^3+\tfrac{3}{1408}z^4
\end{aligned}
$$

RKF4(5)

RKF4(5) is probably the most used method in the set of embedded Runge-Kutta methods designed by Fehlberg [Fehlberg, 1969; Hairer *et al.*, 1987, pp. 169–170].

$$
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\
\frac{12}{13} & \frac{1932}{2197} & \frac{-7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & \frac{-845}{4104} & 0 & 0 \\
\frac{1}{2} & \frac{-8}{27} & 2 & \frac{-3544}{2565} & \frac{1859}{4104} & \frac{-11}{40} & 0 \\
\hline
y & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & \frac{-1}{5} & 0 \\
\hline
\hat{y} & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & \frac{-9}{50} & \frac{2}{55}
\end{array}
$$

$$
\begin{aligned}
P(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1}{104}z^5 \\
\hat{P}(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1}{120}z^5 + \tfrac{1}{2080}z^6 \\
E(z) &= \tfrac{1}{780}z^5 - \tfrac{1}{2080}z^6
\end{aligned}
$$

DOPRI(4)5

Dormand and Prince have derived a family of embedded Runge-Kutta methods with stage reuse and local extrapolation. One of the most used is DOPRI(4)5 [Dormand and Prince, 1980, method RK5(4)7M], [Hairer *et al.*, 1987, p. 171].

$$
\begin{array}{c|ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & 0 & 0 & 0 & 0 \\
\frac{4}{5} & \frac{44}{45} & \frac{-56}{15} & \frac{32}{9} & 0 & 0 & 0 & 0 \\
\frac{8}{9} & \frac{19372}{6561} & \frac{-25360}{2187} & \frac{64448}{6561} & \frac{-212}{729} & 0 & 0 & 0 \\
1 & \frac{9017}{3168} & \frac{-355}{33} & \frac{46732}{5247} & \frac{49}{176} & \frac{-5103}{18656} & 0 & 0 \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & \frac{-2187}{6784} & \frac{11}{84} & 0 \\
\hline
y & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & \frac{-92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
\hline
\hat{y} & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & \frac{-2187}{6784} & \frac{11}{84} & 0
\end{array}
$$

$$
\begin{aligned}
P(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1097}{120000}z^5 + \tfrac{161}{120000}z^6 + \tfrac{1}{24000}z^7 \\
\hat{P}(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1}{120}z^5 + \tfrac{1}{600}z^6 \\
E(z) &= \tfrac{97}{120000}z^5 - \tfrac{13}{40000}z^6 + \tfrac{1}{24000}z^7
\end{aligned}
$$

VERN5(6)

VERN5(6) belongs to a family of Runge-Kutta methods designed by Verner [Verner, 1978, Table 5]. It is intended to be used without local extrapolation.

$$
\begin{array}{c|cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{18} & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{6} & \frac{-1}{12} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{2}{9} & \frac{-2}{81} & \frac{4}{27} & \frac{8}{81} & 0 & 0 & 0 & 0 & 0 \\
\frac{2}{3} & \frac{40}{33} & \frac{-4}{11} & \frac{-56}{11} & \frac{54}{11} & 0 & 0 & 0 & 0 \\
1 & \frac{-369}{73} & \frac{72}{73} & \frac{5380}{219} & \frac{-12285}{584} & \frac{2695}{1752} & 0 & 0 & 0 \\
\frac{8}{9} & \frac{-8716}{891} & \frac{656}{297} & \frac{39520}{891} & \frac{-416}{11} & \frac{52}{27} & 0 & 0 & 0 \\
1 & \frac{3015}{256} & \frac{-9}{4} & \frac{-4219}{78} & \frac{5985}{128} & \frac{-539}{384} & 0 & \frac{693}{3328} & 0 \\
\hline
y & \frac{3}{80} & 0 & \frac{4}{25} & \frac{243}{1120} & \frac{77}{160} & \frac{73}{700} & 0 & 0 \\
\hline
\hat{y} & \frac{57}{640} & 0 & \frac{-16}{65} & \frac{1377}{2240} & \frac{121}{320} & 0 & \frac{891}{8320} & \frac{2}{35}
\end{array}
$$

$$
\begin{aligned}
P(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1}{120}z^5 + \tfrac{7}{6480}z^6 \\
\hat{P}(z) &= 1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3 + \tfrac{1}{24}z^4 + \tfrac{1}{120}z^5 + \tfrac{1}{720}z^6 + \tfrac{1}{6480}z^7 \\
E(z) &= -\tfrac{1}{3240}z^6 - \tfrac{1}{6480}z^7
\end{aligned}
$$

DOPRI(7)8

DOPRI(7)8 is another method in the Dormand–Prince family [Prince and Dormand, 1981; Hairer *et al.*, 1987, p. 193]. It is designed to minimize the error coefficients of the 8th order approximation.

The Butcher tableau of this method is a little too large to reproduce. We refer to [Prince and Dormand, 1981; Hairer *et al.*, 1987, p. 195] for the coefficients. Moreover, the coefficents of DOPRI(7)8 are rational approximations with relative accuracy $\approx 10^{-18}$, which makes the polynomials $P(z)$ and $\hat{P}(z)$ deviate slightly from the expected Taylor expansions also for low order z-terms.

$$\begin{aligned}
P(z) &= 1 \\
&\quad + \left(1+7.7110\cdot 10^{-19}\right) z + \left(\tfrac{1}{2!} - 2.5991\cdot 10^{-18}\right) z^2 + \left(\tfrac{1}{3!} - 5.1606\cdot 10^{-19}\right) z^3 \\
&\quad + \left(\tfrac{1}{4!} - 1.7147\cdot 10^{-20}\right) z^4 + \left(\tfrac{1}{5!} - 1.3876\cdot 10^{-19}\right) z^5 + \left(\tfrac{1}{6!} - 2.2932\cdot 10^{-20}\right) z^6 \\
&\quad + \left(\tfrac{1}{7!} - 3.0332\cdot 10^{-21}\right) z^7 + \left(\tfrac{1}{8!} + 2.4267\cdot 10^{-7}\right) z^8 + \left(\tfrac{1}{9!} - 1.7467\cdot 10^{-7}\right) z^9 \\
&\quad + \left(\tfrac{1}{10!} + 4.1705\cdot 10^{-9}\right) z^{10} + \left(\tfrac{1}{11!} - 1.4102\cdot 10^{-8}\right) z^{11} + \left(\tfrac{1}{12!} - 2.1898\cdot 10^{-9}\right) z^{12} \\
\hat{P}(z) &= 1 \\
&\quad + \left(1-3.6853\cdot 10^{-18}\right) z + \left(\tfrac{1}{2!} - 4.2503\cdot 10^{-18}\right) z^2 + \left(\tfrac{1}{3!} - 2.2402\cdot 10^{-18}\right) z^3 \\
&\quad + \left(\tfrac{1}{4!} - 5.4901\cdot 10^{-19}\right) z^4 + \left(\tfrac{1}{5!} - 8.5307\cdot 10^{-20}\right) z^5 + \left(\tfrac{1}{6!} - 2.6136\cdot 10^{-20}\right) z^6 \\
&\quad + \left(\tfrac{1}{7!} - 2.1926\cdot 10^{-21}\right) z^7 + \left(\tfrac{1}{8!} - 2.5859\cdot 10^{-22}\right) z^8 + \left(\tfrac{1}{9!} - 3.6039\cdot 10^{-9}\right) z^9 \\
&\quad + \left(\tfrac{1}{10!} - 3.3253\cdot 10^{-8}\right) z^{10} + \left(\tfrac{1}{11!} - 6.6239\cdot 10^{-10}\right) z^{11} + \left(\tfrac{1}{12!} - 2.2911\cdot 10^{-9}\right) z^{12}
\end{aligned}$$

$$\begin{aligned}
E(z) = {} & \left(4.4564\cdot 10^{-18} z + 1.6512\cdot 10^{-18} z^2 + 1.7242\cdot 10^{-18} z^3 + 5.3186\cdot 10^{-19} z^4\right. \\
& \left. - 5.3455\cdot 10^{-20} z^5 + 3.2043\cdot 10^{-21} z^6 - 8.4060\cdot 10^{-22} z^7\right) + 2.4267\cdot 10^{-7} z^8 \\
& - 1.7107\cdot 10^{-7} z^9 + 3.7424\cdot 10^{-8} z^{10} - 1.3439\cdot 10^{-8} z^{11} + 1.0132\cdot 10^{-10} z^{12}
\end{aligned}$$

A.2 Implicit Methods

Implicit Euler

Implicit Euler is the simplest implicit method [Hairer *et al.*, 1987, p.199]. It does not include an error estimator.

$$\begin{array}{c|c} 1 & 1 \\ \hline y & 1 \end{array} \qquad\qquad P(z) = \frac{1}{1-z}$$

Implicit midpoint

The implicit midpoint rule is a combination of an explicit and an implicit Euler step [Hairer *et al.*, 1987, p. 199].

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline y & 1 \end{array} \qquad\qquad P(z) = \frac{2+z}{2-z}$$

Trapezoidal Rule

The trapezoidal rule is similar to the implicit midpoint rule, but here the explicit and the implicit Euler steps are taken in reversed order [Hairer *et al.*, 1987, p. 199]. With stage reuse the Trapezoidal rule is a one-stage method.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline y & \frac{1}{2} & \frac{1}{2} \end{array} \qquad P(z) = \frac{2+z}{2-z}$$

SIMPLE(2)3

SIMPLE(2)3 is a code developed by Nørsett and Thomsen [Nørsett and Thomsen, 1984; Nørsett and Thomsen, 1987]. It implements an A-stable and B-stable (but not L-stable) SDIRK of order 2/3.

$$\begin{array}{c|ccc} \frac{5}{6} & \frac{5}{6} & 0 & 0 \\ \frac{29}{108} & \frac{-61}{108} & \frac{5}{6} & 0 \\ \frac{1}{6} & \frac{-23}{183} & \frac{-33}{61} & \frac{5}{6} \\ \hline y & \frac{25}{61} & \frac{36}{61} & 0 \\ \hline \hat{y} & \frac{26}{61} & \frac{324}{671} & \frac{1}{11} \end{array}$$

$$\begin{aligned} P(z) &= \frac{36-24z-17z^2}{36-60z+25z^2} \\ \hat{P}(z) &= \frac{216-324z+18z^2+91z^3}{216-540z+450z^2-125z^3} \\ E(z) &= \frac{-6z^3}{216-540z+450z^2-125z^3} \end{aligned}$$

HW-SDIRK(3)4

HW-SDIRK(3)4 is an L-stable implicit method developed by Hairer and Wanner, [Hairer and Wanner, 1991, pp. 107].

$$\begin{array}{c|ccccc} \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{11}{20} & \frac{17}{50} & \frac{-1}{25} & \frac{1}{4} & 0 & 0 \\ \frac{1}{2} & \frac{371}{1360} & \frac{-137}{2720} & \frac{15}{544} & \frac{1}{4} & 0 \\ 1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\ \hline y & \frac{59}{48} & \frac{-17}{96} & \frac{225}{32} & \frac{-85}{12} & 0 \\ \hline \hat{y} & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \end{array}$$

$$P(z) = \frac{768 - 96z^2 - 16z^3 + 10z^4}{768 - 768z + 288z^2 - 48z^3 + 3z^4}$$
$$\hat{P}(z) = \frac{3072 - 768z - 384z^2 + 32z^3 + 28z^4}{3072 - 3840z + 1920z^2 - 480z^3 + 60z^4 - 3z^5}$$
$$E(z) = \frac{28z^4 - 10z^5}{3072 - 3840z + 1920z^2 - 480z^3 + 60z^4 - 3z^5}$$

A.3 Method Properties

An important property of an integration method is its stability region. The stability regions of the methods listed above are depicted in Figure A.1.

For the explicit methods, we are interested in the behavior when stability limits the stepsize. In this case the stepsize-error relation can be modeled as, cf. (3.22) in Chapter 3,

$$G_{p2}(q) = \frac{k(\beta_0 q + \beta_1)}{q(q-1)}$$

with (3.23)

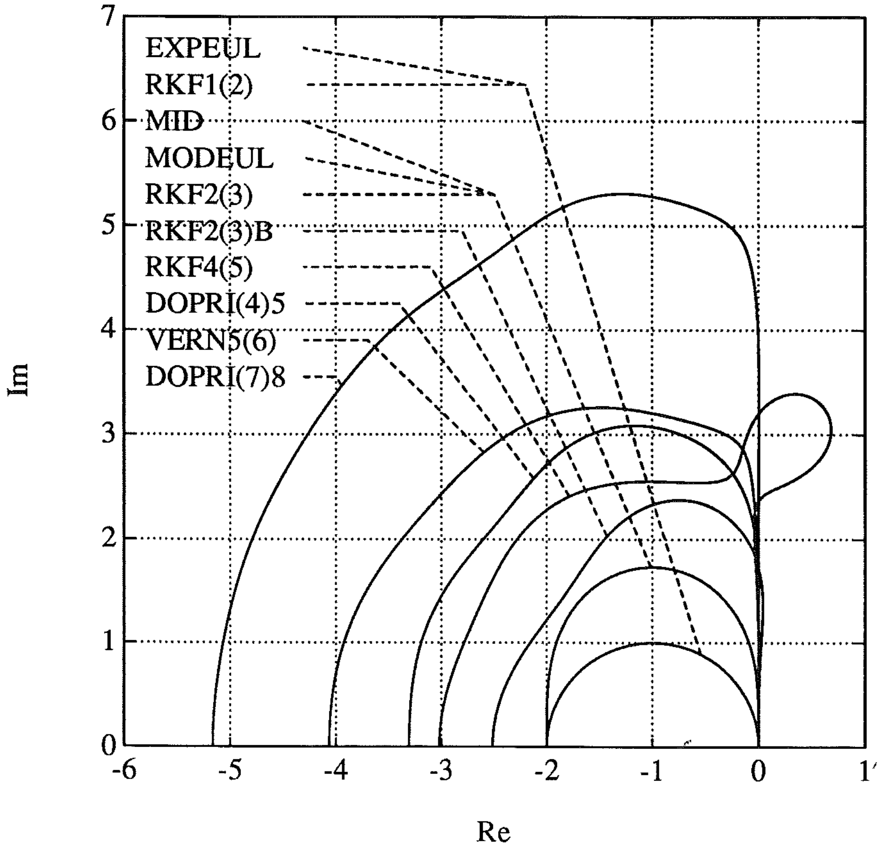
$$\beta_0 = \begin{cases} C_E/k, & \text{EPS} \\ (C_E - 1)/k, & \text{EPUS} \end{cases}, \qquad \beta_1 = \begin{cases} (C_P - C_E)/k, & \text{EPS} \\ (C_P - C_E + 1)/k, & \text{EPUS} \end{cases}$$

and (3.26)

$$C_E(h_s\lambda) = \operatorname{Re}\left(h_s\lambda \frac{E'(h_s\lambda)}{E(h_s\lambda)}\right), \qquad C_P(h_s\lambda) = \operatorname{Re}\left(h_s\lambda \frac{P'(h_s\lambda)}{P(h_s\lambda)}\right).$$

The parameters β_0 and β_1 vary along the border of the stability region, cf. Figure A.2. As can be seen β_0 normally stays in the interval $[1, 1.5]$ when $\theta = \arg(h_s\lambda)$ varies from $\pi/2$ to π. The variations in β_1 are larger, but it normally stays in $[-1, 1]$ when $\theta = \arg(h_s\lambda)$ varies from $\pi/2$ to π. When using XEPUS some of the low order methods (RKF(1)2, RKF(2)3, and RKF(2)3B) have a β_1 value that gets larger than 1. The variations are in general larger for (X)EPUS than (X)EPS.

The values of β_0 and β_1 are of particular interest for $\theta = \pi$, since this is what is experienced when solving a differential equation dominated by a negative real eigenvalue. The corresponding parameter values are listed in Tables A.1 and A.2 .



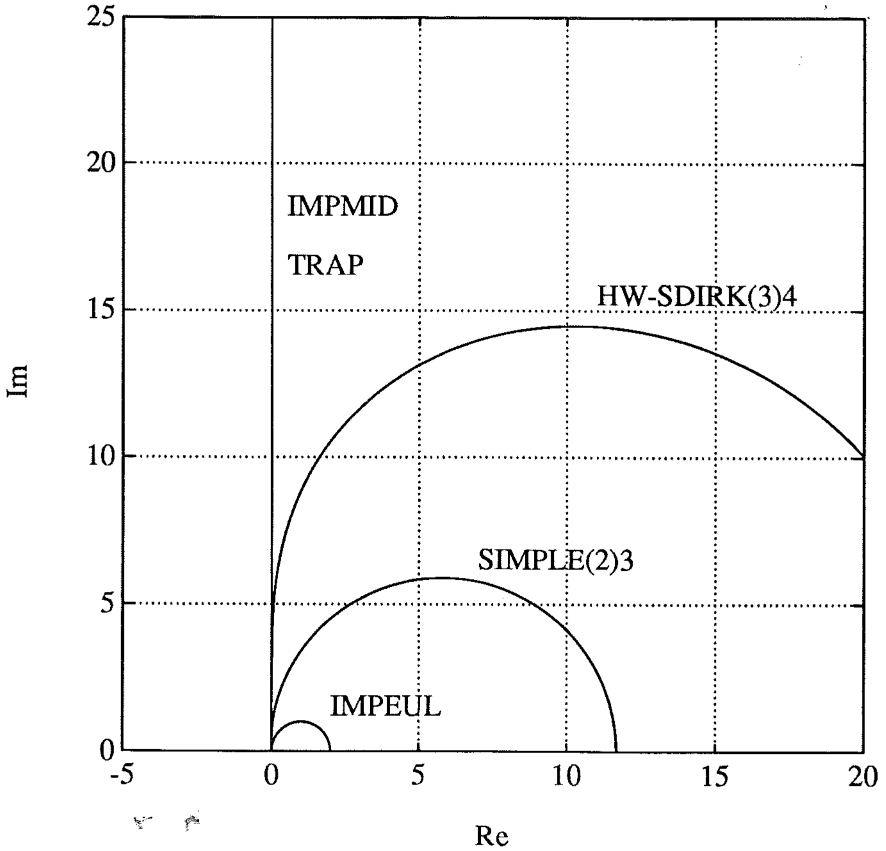


Figure A.1 Stability regions of the explicit (upper plot) and the implicit (lower plot) methods in Sections A.1 and A.2. Only the part of the stability region that includes the origin is shown.

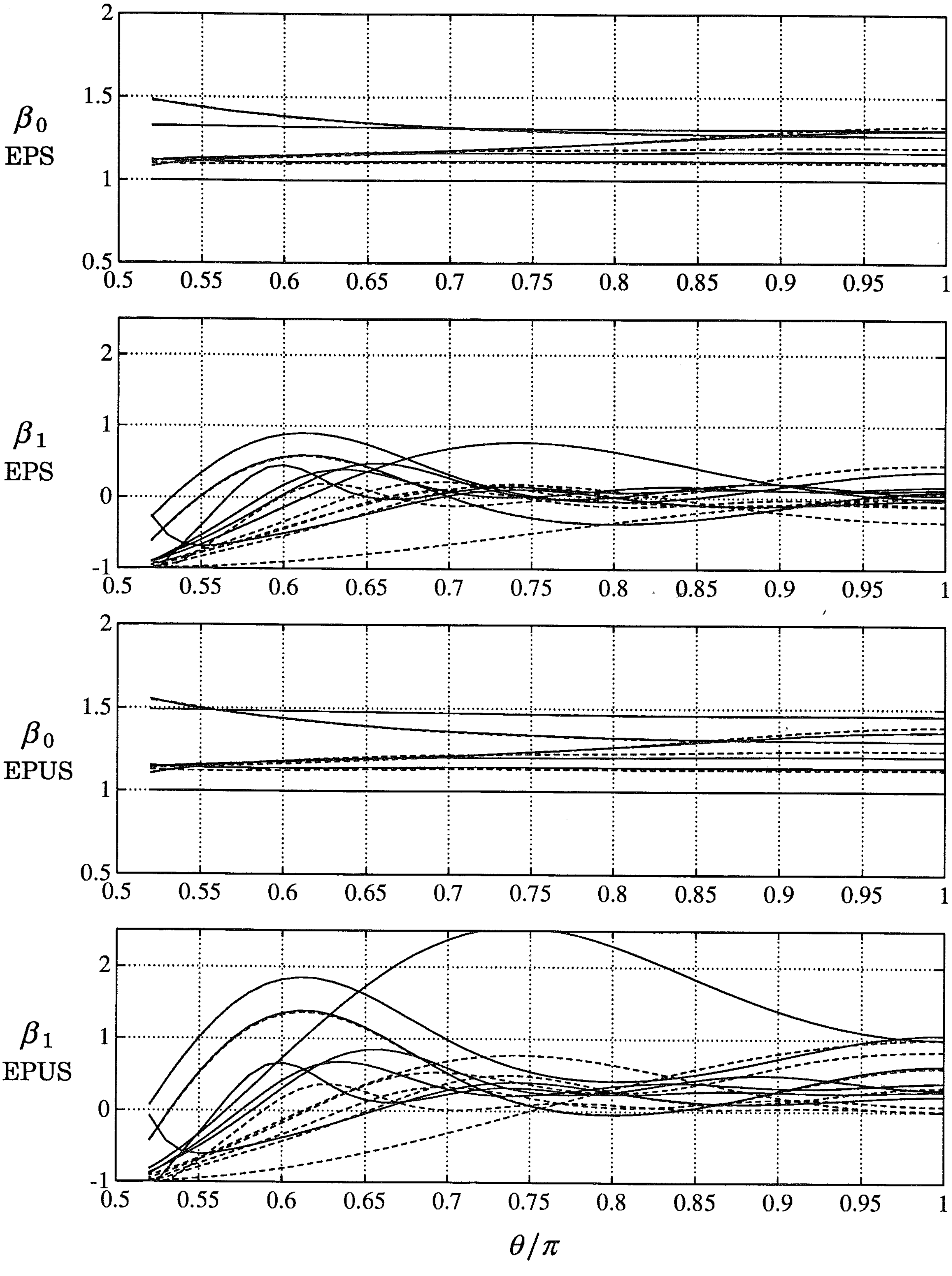


Figure A.2 The parameters β_0 and β_1 of the explicit methods in Section A.1 plotted as a function of $\theta = \arg(h_s\lambda)$. The upper two plots correspond to using the method in EPS mode, and the lower two plots to EPUS mode. The full lines relate to the case when the high order formula is used for solution update (local extrapolation), and the dashed lines relate to the low order formula.

	XEPS		EPS	
Method	β_0	β_1	β_0	β_1
RKF12	1.00	0.000	1.00	−0.000607
RKF23	1.00	0.376	1.00	−0.333
RKF23B	1.31	0.0885	1.31	0.0708
RKF45	1.12	−0.0303	1.11	0.0704
DOPRI45	1.17	0.0450	1.20	0.468
VERN56	1.30	0.164	1.33	−0.103
DOPRI78	1.27	0.120	1.27	−0.113

Table A.1 The values for β_0 and β_1 (3.23) at the intersection of the boundary of the stability region and the negativer real axis. The values are given for error control in EPS mode.

	XEPUS		EPUS	
Method	β_0	β_1	β_0	β_1
RKF12	1.00	1.00	1.00	0.999
RKF23	1.00	1.06	1.00	0.000
RKF23B	1.46	0.633	1.46	0.606
RKF45	1.14	0.212	1.13	0.338
DOPRI45	1.21	0.306	1.25	0.835
VERN56	1.36	0.396	1.39	0.0766
DOPRI78	1.31	0.280	1.31	0.0133

Table A.2 The values for β_0 and β_1 (3.23) at the intersection of the boundary of the stability region and the negativer real axis. The values are given for error control in EPUS mode.

B

List of Notations

$\mathcal{A}$, b, c	coefficients of Runge-Kutta method
C_E, C_P	parameters in stepsize-error model on $\partial\mathcal{S}$
e, $\hat{e}$	local truncation error and local truncation error estimate
E	error polynomial/rational in discretization of test equation
h, h_s	stepsize, stepsize that puts $h\lambda$ on $\partial\mathcal{S}$
J, $\bar{J}$	Jacobian, mean value Jacobian
k	exponent in stepsize-error model
k_I, k_P	controller parameters (explicit case)
k_1, k_2	controller parameters (implicit case)
M	iteration matrix
p	convergence order of integration method
P, $\hat{P}$	stability polynomial/rational in discretization of test equation
q	forward shift operator
r	norm of local truncation error estimate
$\mathcal{S}$, $\partial\mathcal{S}$	stability region and boundary of stability region
tol	accuracy requirement
Y	stage values

α	convergence rate estimate
α_{LU}, α_{JAC}	controller parameters (implicit case)
α_{ref}	set-point for convergence rate
β	stiffness estimate
β_1, β_2	parameters in stepsize-error model on $\partial\mathcal{S}$
γ	diagonal element in $\mathcal{A}$
Δ	displacement
ε	set-point for error estimate
η	parameter in error norm
ϑ	parameter in covergence rate model
ν	parameter in convergence rate model
ξ	parameter in strategy for switching equation solver
τ	set-point for iteration error
ϕ	principal error function
φ	norm of principal error function estimate
∇	backward difference operator

> "I think I should understand that better," Alice said very politely, "if I had it written down: but I'm afraid I ca'n't quite follow it as you say it."
>
> "That's nothing to what I could say if I chose," the Duchess replied, in a pleased tone.
>
> "Pray don't trouble yourself to say it any longer than that," said Alice.
>
> Lewis Carroll, *Alice's Adventures in Wonderland*