Automatic control for adaptive time stepping in electrical circuit simulation

Verhoeven, A.

Award date:
2004

TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Mathematics and Computing Science

MASTER'S THESIS

Automatic control for adaptive time stepping in electrical circuit simulation

by
A.Verhoeven

Supervisor: prof.dr.ir. M.L.J.Hautus, dr.E.J.W.ter Maten

Eindhoven, December 19, 2003

Abstract

The group Philips Electronic Design & Tools, Analogue Simulation provides the circuit simulator Pstar, which can perform a transient analysis on electrical circuits.

DAE solvers are used to find the numerical solution of this transient analysis. Pstar uses the Nordsieck version of the BDF method as DAE solver. For onestep methods, the local truncation error only depends on the last stepsize, but for multistep methods, this error is also dependent on previous stepsizes in a nonlinear way. Adaptive stepsize control and order control is used to control these errors of the numerical solution.

In connection with the articles of prof.Söderlind, this MSc Thesis investigates the possibilities of a control-theoretic approach to stepsize control. The purpose was to design smoother controllers, such that also the error and stepsize-sequences become smoother, which is useful for optimization. For onestep methods, it appears that the stepsize control process can be viewed as a digital linear control system for the logarithms of the errors and steps. This means that the logarithm of the error depends linearly on the logarithms of the previous stepsizes and an additional disturbance, which is nearly independent on the stepsizes. From a control-theoretic point of view, the goal is to keep the error close to a reference level with use of the stepsizes as input. For the BDF-method, which is a multistep method, this process can only be approximated by such linear control system. There are two approximation levels: the first level corresponds with the onestep model and the second level is derived by linearization techniques.

If the stepsize control process is correctly modelled, a finite order digital linear controller can be designed. This design depends on the wanted controller specifications, such as the poles of the closed loop dynamics. Also the adaptivity or prediction order of the stepsize controller can be determined. Furthermore, the controller can be designed with filter properties with respect to the error or the stepsize sequences. Because the computation of the control parameters can be rather complex, this has been implemented.

Constraint validation appears to be an important property of a stepsize controller, because it guarantees that the next stepsize will be accepted, if the process model is correct and the disturbance varies not too much. It is also possible to use nonlinear controllers with the same linearized behaviour, which don't need the assumption that the process model is valid for constraint validation.

From the results, it appears that for nearly all tested circuits, it is possible to design better linear controllers than the deadbeat controller, which is used by Pstar itself. For stiff problems, constraint validation is a necessary property, because it reduces the numbers of rejections. However, for non-stiff problems, one could use smooth controllers with all poles equal, which don't have oscillatory behaviour. For some circuits, the controllers based on model two perform better. It appears that stepsize filters can be preferred above error filters. The stepsize filters are useful, if the error and stepsize sequences have high-frequent behaviour. Predictive control worked badly, in comparison with the other controllers.

Further research could be done on the combination of order and stepsize control, control of the Newton-Raphson method and the application of optimal control and system identification. Also the used error estimate could be important, because it must satisfy the process model for the local truncation error. Adaptive stepsize control in combination with multirate has to be investigated more profoundly.

Preface

This MSc Thesis describes the results of my MSc project for the Technische Universiteit Eindhoven. The project has been performed in cooperation with Philips Electronic Design & Tools / Analogue Simulation (ED&T/AS), which is headed by Jan van Gerwen. I have worked on the circuit simulator Pstar, which is provided by ED&T/AS. My supervisors were Malo Hautus[1] and Jan ter Maten[2].

I would like to thank everybody, who supported me to write this MSc Thesis. In particular, I mean Jan ter Maten and Malo Hautus for their kind support, during the whole project. I am also gratefully to Bob Mattheij[3] and Theo Beelen[4], who spent a lot of time with reviewing this thesis. Furthermore, Jos Peters and Roland Hermans are thanked for their help with the numerical experiments for Pstar.

Besides these people, I would like also to call explicitly Jan van Gerwen, Els van Duren, Jaap Fijnvandraat and also all other members of the group ED&T/AS.

[1]Professor System Theory at the Department of Mathematics and Computing Science of the TU/e.

[2]Employee of ED&T/AS and scientist Scientific Computing at the Department of Mathematics and Computing Science of the TU/e

[3]Professor Scientific Computing at the Department of Mathematics and Computing Science of the TU/e.

[4]Project leader of Pstar.

Contents

Chapter 1

Introduction

1.1 Motivation for the project

Integrated circuits or chips are (large) electrical circuits, which react on input sources. The dynamics of the chip determine the behaviour of the chip. These chips can be used as controllers or other signal processors. Nowadays, they are very important and are used in many applications[1]. Circuit designers make chip designs such that the resulting chip has the wanted behaviour. To reduce the costs, software packages are made, that can analyse chip designs. These tools can perform several types of circuit analysis, such as direct current analysis, small signal analysis or transient analysis. In practice, transient analysis is an important analysis, because then the full nonlinear dynamics of the circuit are analysed. The circuit simulators use mathematical circuit models, which can be derived from the theory of electrical circuits. These models can be described by differential-algebraic equations.

Because Philips is also a chip manufacturer, it needs also tools to analyse the designed integrated circuits. The group Philips Electronic Design & Tools, Analogue Simulation provides several tools, among others the analog circuit simulator Pstar.

The graduating project "Automatic Control for Adaptive Time Stepping in Electrical Circuit Simulation" has been performed to improve the transient integration by Pstar. I have studied the possibilities of automatic control for adaptive time stepping for solving the differential-algebraic equations of an electrical circuit.

1.2 Formulation of the problem

Differential-algebraic equations (DAE's) describe the dynamic behaviour of electrical circuits. There are several numerical procedures to compute the numerical solution of these DAE's. All these methods discretize the time and compute the solution on the time grid with the help of integration methods. Of course, this numerical solution will not be equal to the exact solution. With the stepsizes, it is possible to control the errors. Therefore, the most numerical solvers use control laws for the stepsizes. It seems however that little mathematical attention is payed to this control logic, in contrast with the discretization methods. That is the reason why Gustaf Söderlind[2] proposed a control-theoretic approach of time step control, which is called adaptive time-stepping. He claims that integration methods with use of adaptive time-stepping yield smoother stepsize sequences, fewer rejected steps,

[1] See Fig. 1.1.

[2] Professor at the Department Numerical Analysis of Lund University

Figure 1.1: An AD-converter for audio of ANALOG DEVICES.

more efficiency, while the total work will not grow significantly. It is also possible to design the stepsize controller for special purposes as higher order of adaptivity (for smooth ODE problems) or filtering the high-frequency error components (for non-smooth problems). This controlled timestep variations are also claimed to be less sensitive with respect to parameter variations than in classical time integration procedures. Hence, the results obtained with automatic control will be more robust and better suited for optimization purposes than before.

The main purpose of the project was to explore the possibilities of automatic control for adaptive time stepping for solving the DAE of an electrical circuit. A comparison has been made between the results of a classical solver and an adaptive solver.

1.3 Guideline to document

In chapter two the dynamics of electrical circuits have been formulated as a DAE. It is considered how the DAE can be derived from the physical laws for the electrical circuits. Furthermore, several kinds of circuit analysis have been enumerated. Afterwards, in chapter three, the numerical analysis of DAE's is summarized. At first, some theoretical aspects of DAE's have been summarized. Afterwards, several numerical methods for DAE's are considered, such as Runge Kutta methods, Linear Multistep Methods and BDF-methods. Special cases have also been considered, e.g. the TR-BDF2-method, NDF-schemes and multirate. Besides the algorithms for this methods, also the estimation and asymptotic behaviour of the errors have been studied. These error estimations and error models are important for adaptive stepsize control. This main topic has been presented in detail in chapter four. First, the classical approach is shown. Afterwards, it is studied how control theory can be applied to control the local errors. To verify the results, numerical experiments have been done with Pstar and MATLAB-DAE-solvers for several electrical circuits. These results are shown in chapter five. Finally, in chapter six the conclusion of the project and proposals for further research can be found. Some additional information has been added in the appendices.

Chapter 2

Dynamics of electrical circuits

2.1 Introduction

Nowadays, circuit simulation[1] plays an important role in the analysis of circuit systems. Because computers can make very complicated computations, it is possible to predict the behaviour of the designed circuit.

There are several types of analysis, e.g. direct current analysis, small signal analysis, pole-zero analysis, transient analysis and periodic steady-state analysis. In practice, transient analysis is a very important analysis. In this case, the behaviour of the system will be computed over a time interval $[0, T]$ as the solution of the DAE

$$\frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}.$$

The functions $\mathbf{q}, \mathbf{j}$ may be non-linear functions of $\mathbf{x}$. The explicit dependence on t usually comes from sources. Firstly, you can have an ordinary initial value problem (IVP), secondly you can have a two-point boundary problem (BVP) for periodic steady-state problems. For an IVP, the initial solution at $t = 0$ usually is the solution of $\mathbf{j}(0, \mathbf{x}) = \mathbf{0}$. In section two, the functions $\mathbf{q}$ and $\mathbf{j}$ will be derived by use of Modified Nodal Analysis. Afterwards, a few important kinds of circuit analysis will be summarized in section three.

2.2 Mathematical model of electrical circuits

2.2.1 Theory of electrical circuits

In general, electric circuits consist of electric components and connecting conductors. Each location in the circuit has its voltage or potential V. If two locations in the network have different potentials, there is a voltage difference v between these locations. A voltage difference will result in a current i to the highest potential, that will level out the voltages. Normally, these currents and voltages depend on the location and time. But because the electrical resistance of the conductors is negligible, only the junctions of the conductors are important.

Consider the next variables:

- The currents through the electrical components;

[1] For more information about this subject, see [7, 37]. For a short overview of the most important theory, the MSc-Theses [21, 30] are useful.

- The nodal voltages at the junctions (nodes);
- The voltage differences[2] across the electrical components.

Together with the equations that relate these variables, these variables are sufficient to describe an electric circuit. It is even possible to describe the circuits with less variables, but the equations have been derived for these variables.

It is possible to consider these circuits as graphs of two types of elements: nodes and branches. The branches represent the components, while the nodes represent the junctions of the conductors. Each branch is connected with a number of nodes. These nodes are divided into positive and negative nodes. The voltage difference v across a branch is equal to $V^+ - V^-$, where V^+ and V^- are the potentials of the positive and negative node, respectively. The direction of a positive current is always to the positive nodes. If the current is negative, it means a positive current to the negative nodes.

Balance laws

There are two types equations from physics, which together describe the circuit.

First of all, there are the balance laws or the laws of Kirchhoff. These equations have a topological character, because they do not depend on the type of the components, but only on the topology of the circuit.

Kirchhoff's Current Law (KCL):	The algebraic sum of all branch currents leaving a node is zero at all instants of time.
Kirchhoff's Voltage Law (KVL):	The algebraic sum of all voltage differences around any closed loop of a network is zero at all instants of time.

Both laws are based on the Maxwell equations[3], which are consequences of the conservation of charge and energy.

Constitutive relations

The constitutive relations[4] (CR)depend on the type of the component. There are many types of components, but here only the basic circuit components have been considered. Let $\mathbf{i}_b$ and $\mathbf{v}_b$ be the vectors, which consists of all branch currents and voltage differences across the branches. Then, in general, all equations of the components can be described in an implicit way:

$$f(\mathbf{v}_b, \frac{d\mathbf{v}_b}{dt}, \mathbf{i}_b, \frac{d\mathbf{i}_b}{dt}, t) = 0.$$

More precisely in the circuits that are considered here, all equations are of the next two types:

$$i_k = f(\tilde{\mathbf{v}}_b, \frac{d\tilde{\mathbf{v}}_b}{dt}, \tilde{\mathbf{i}}_b, \frac{d\tilde{\mathbf{i}}_b}{dt}, t), \quad \text{I}$$
$$v_k = f(\tilde{\mathbf{v}}_b, \frac{d\tilde{\mathbf{v}}_b}{dt}, \tilde{\mathbf{i}}_b, \frac{d\tilde{\mathbf{i}}_b}{dt}, t), \quad \text{II}$$

where $\tilde{\mathbf{v}}_b$ and $\tilde{\mathbf{i}}_b$ are equal to $\mathbf{v}_b$ and $\mathbf{i}_b$ without the variables v_k and i_k. The components with equations of type I are called current-defined and voltage-controlled components, while the other components are called voltage-defined and current-controlled components.

[2]These variables are also called branch voltages.

[3]In [22], it has been shown how these laws can be derived from the Maxwell equations.

[4]These equations are often also mentioned as branch equations.

It appears that all components of electrical circuits can be modelled by means of some basic components. The behaviour of more complex components, such as transistors, can be described with compact models, which consist only of these basic components. In the next paragraphs, the dynamics of these basic components are described. Also the pictures of the components in network diagrams used, are shown.

Current source

A simple current source generates a given time-dependent current between its two nodes, independent of the state of the circuit.

$$i_j = i^*(t).$$

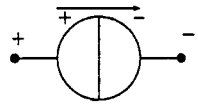


Figure 2.1: Picture of a current source

Voltage source

A simple voltage source generates a given time-dependent voltage difference, also independent of the state of the circuit.

$$v_e = v^*(t).$$

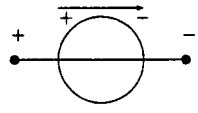


Figure 2.2: Picture of a voltage source

Capacitor

An ideal capacitor stores an amount of electrical charge q_c without loss. A change of the charge results in a current through a capacitor, because $i_c = \frac{dq_c}{dt}$. Furthermore the voltage difference across the capacitor is given by $q_c = f(v_c)$. Hence the CR is:

$$i_c = \frac{d}{dt} f(v_c).$$

If the capacitor is linear with capacity C, the CR is:

$$i_c = C \frac{dv_c}{dt}.$$

Nonlinear capacitors occur in the compact models that describe the behaviour of transistor models.

Figure 2.3: Picture of a capacitor

Inductor

An inductor generates a voltage difference, because of self-inductance. A change of the flux linkage ϕ_l will result in a voltage difference, because $v_l = \frac{d\phi}{dt}$. Furthermore the flux linkage depends on the current: $\phi_l = f(i_l)$. Hence the CR is:

$$v_l = \frac{d}{dt} f(i_l).$$

If the inductor is linear with inductance L, the CR is:

$$v_l = L\frac{di_l}{dt}.$$

Figure 2.4: Picture of an inductor

Resistor

A resistor reduces the energy in a circuit. In contrast to the inductor and capacitor, the behaviour of a resistor is not dynamic. If the resistor is current-defined, its CR is:

$$i_r = f(v_r).$$

while the CR of a voltage-defined resistor is:

$$v_r = f(i_r).$$

The diode junction is a frequently used nonlinear resistor with CR:

$$i_d = i_d^0(e^{\alpha v_d} - 1)$$

with i_d^0 the diode saturation current and $\alpha \approx 40$ a physical parameter. This model is used in the compact models for transistors. If the resistor is linear with resistance R, the CR is:

$$i_r = \frac{v_r}{R}.$$

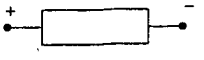


Figure 2.5: Picture of a resistor

Controlled components

Electrical circuits may also contain controlled components. Normally the dynamics of component k depend on the current i_k, the voltage difference v_k and thus of the potentials of the connected nodes V_k^1, V_k^2. The equations of the branches may have the next representations:

$$\begin{array}{rcccl} i_k & = & f(v_k) & = & f(V_k^1 - V_k^2), \\ V_k^1 - V_k^2 & = & v_k & = & g(i_k). \end{array}$$

But it is also possible to add feedback controllers to the circuit. In this case, the dynamics of each component may depend on all variables. Now the branch equations may have the following representations:

$$\begin{array}{rcccl} i_k & = & f(v_k, \mathbf{v}_b, \mathbf{i}_b) & = & f(V_k^1 - V_k^2, \mathbf{v}_b, \mathbf{i}_b), \\ V_k^1 - V_k^2 & = & v_k & = & g(i_k, \mathbf{v}_b, \mathbf{i}_b), \end{array}$$

where $\mathbf{v}_b$ and $\mathbf{i}_b$ consist of all voltage differences and currents. With this type of branch equation, oscillators can be described.

2.2.2 Network analysis

In the previous section, the equations are derived which describe the circuit. In this section, the functions $\mathbf{q}$ and $\mathbf{j}$ will be derived, such that:

$$\frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}.$$

Assume that the network[5] consists of n nodes and b branches. Structurally this can be stored in the matrix $A \in \mathbb{R}^{n \times b}$, which is defined by:

$$A_{ij} = \begin{cases} 1 & \text{if branch } j \text{ is incident at node } i \text{ and node } i \text{ is positive node of } j, \\ -1 & \text{if branch } j \text{ is incident at node } i \text{ and node } i \text{ is negative node of } j, \\ 0 & \text{if branch } j \text{ is not incident at node } i. \end{cases}$$

Introduce the vector $\mathbf{V}_n \in \mathbb{R}^n$ which contains the nodal voltages, arranged in the same order as the rows of A. Introduce furthermore the vector $\mathbf{i}_b \in \mathbb{R}^b$ and $\mathbf{v}_b \in \mathbb{R}^b$ which contain the branch currents and branch voltage differences in the same order as the columns of A.

With A, $\mathbf{V}_n$, $\mathbf{v}_b$ and $\mathbf{i}_b$ it is easy to formulate KCL and KVL:

$$\begin{array}{ll} A\mathbf{i}_b = \mathbf{0}, & \text{(KCL)}, \\ A^T\mathbf{V}_n = \mathbf{v}_b. & \text{(KVL)}. \end{array}$$

Besides the balance laws, also the constitutive relations for several components have to be used. Assume that the circuit only consists of current-defined components. In that case, for each component, the CR are of the form:

$$i_k = f(\mathbf{v}_b, t) = \frac{d}{dt}q(t, \mathbf{v}_b) + j(t, \mathbf{v}_b).$$

Because all constitutive relations are of this type, the state vector $\mathbf{i}_b$ satisfies the next equation:

$$\mathbf{i}_b = \tfrac{d}{dt}\tilde{\mathbf{q}}(t, \mathbf{v}_b) + \tilde{\mathbf{j}}(t, \mathbf{v}_b), \quad \text{CR}$$

[5]This theory has been derived from [10, 21, 30].

with $\tilde{\mathbf{q}}, \tilde{\mathbf{j}} : \mathbb{R} \times \mathbb{R}^b \to \mathbb{R}^b$ functions, which describe all constitutive relations.

In matrix-vector notation, the equations of the circuits are:

$$\begin{aligned} A\mathbf{i}_b &= \mathbf{0}, && \text{(KCL)} \\ A^T\mathbf{V}_n &= \mathbf{v}_b, && \text{(KVL)} \\ \mathbf{i}_b &= \tfrac{d}{dt}\tilde{\mathbf{q}}(t, \mathbf{v}_b) + \tilde{\mathbf{j}}(t, \mathbf{v}_b). && \text{(CR)} \end{aligned}$$

Left-multiplying the third equation with A and use of KCL gives:

$$\mathbf{0} = A\mathbf{i}_b = \frac{d}{dt}A\tilde{\mathbf{q}}(t, \mathbf{v}_b) + A\tilde{\mathbf{j}}(t, \mathbf{v}_b).$$

Furthermore, it is possible to eliminate $\mathbf{v}_b$ by means of KVL, which results in:

$$\mathbf{0} = \frac{d}{dt}A\tilde{\mathbf{q}}(t, A^T\mathbf{V}_n) + A\tilde{\mathbf{j}}(t, A^T\mathbf{V}_n). \tag{2.1}$$

The result is a system of n equations with n unknowns. Because the column sum is zero for all columns of A, this matrix must be singular[6]. This means that $\mathbf{V}_n$ is not uniquely determined by equation (2.1), because only the voltage differences are important. This is the reason why at least one node may be grounded. Assume, the k-th node is grounded at V^*. In that case, the k-th row of A and the k-th co-ordinate of $\mathbf{V}_n$ have to be removed, resulting in $\hat{A}$ and $\hat{V}_n$. This will lead to a new system, with $n-1$ equations and unknows, with $\mathbf{e}_k \in \mathbb{R}^n$ the k-th unit vector:

$$\mathbf{0} = \frac{d}{dt}\hat{A}\tilde{\mathbf{q}}(t, \hat{A}^T\hat{\mathbf{V}}_n + V^*A^T\mathbf{e}_k) + \hat{A}\tilde{\mathbf{j}}(t, \hat{A}^T\hat{\mathbf{V}}_n + V^*A^T\mathbf{e}_k).$$

Now define the state vector $\mathbf{x} = \hat{\mathbf{V}}_n$ and functions $\mathbf{q}, \mathbf{j} : \mathbb{R} \times \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$ with

$$\mathbf{q}(t, \mathbf{x}) = \hat{A}\tilde{\mathbf{q}}(t, \hat{A}^T\mathbf{x} + V^*A^T\mathbf{e}_k),$$

$$\mathbf{j}(t, \mathbf{x}) = \hat{A}\tilde{\mathbf{j}}(t, \hat{A}^T\mathbf{x} + V^*A^T\mathbf{e}_k).$$

Indeed, the circuit is described with the next differential-algebraic equation:

$$\frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}. \tag{2.2}$$

This version of network analysis is also called "Nodal analysis". This method is only possible if all components are current-defined, which, in general, is not the case. In the next section, this method will be slightly adapted to treat also voltage-defined components, e.g. voltage sources and inductors.

2.2.3 Modified nodal analysis

Because Modified nodal analysis is an extension of Nodal analysis, it uses the notations from Nodal Analysis. But now it is assumed that the circuit contains current-defined (type I) as well as voltage-defined (type II) types of components. We will show that it is possible to derive a similar system of equations of the form (2.2), where $\mathbf{x}$ consists of all nodal voltages $\mathbf{V}_n$ and the currents through the voltage-defined elements.

[6] In [21], it has been proved that the rank of the topology matrix for a connected circuit is $n-1$.

Assume there are b_1 branches of type I and b_2 branches of type II, thus $b_1 + b_2 = b$. The vectors $\mathbf{v}_b$ and $\mathbf{i}_b$ are split into two parts:

$$\mathbf{v}_b = \begin{pmatrix} \mathbf{v}_{b_1} \\ \mathbf{v}_{b_2} \end{pmatrix}, \quad \mathbf{i}_b = \begin{pmatrix} \mathbf{i}_{b_1} \\ \mathbf{i}_{b_2} \end{pmatrix}.$$

Furthermore, also the matrix A is partitioned into two parts:

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix}.$$

In the new variables, the balance laws can be written in the next form:

$$A_1\mathbf{i}_{b_1} + A_2\mathbf{i}_{b_2} = \mathbf{0}, \quad \text{(KCL)}$$

$$\left.\begin{aligned} A_1^T\mathbf{V}_n &= \mathbf{v}_{b_1}, \\ A_2^T\mathbf{V}_n &= \mathbf{v}_{b_2}. \end{aligned}\right\} \quad \text{(KVL)}$$

By definition, the constitutive relations of the two types of components can be formulated as

$$\left.\begin{aligned} \mathbf{i}_{b_1} &= \tfrac{d}{dt}\tilde{\mathbf{q}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}) + \tilde{\mathbf{j}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}), \\ \mathbf{v}_{b_2} &= \tfrac{d}{dt}\bar{\mathbf{q}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}). \end{aligned}\right\} \quad \text{CR}$$

Left-multiplying the first equation with A_1 and using KCL results in:

$$-A_2\mathbf{i}_{b_2} = \frac{d}{dt}A_1\tilde{\mathbf{q}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}) + A_1\tilde{\mathbf{j}}(t, \mathbf{v}_{b_1}, \mathbf{v}_{b_2}, \mathbf{i}_{b_2}).$$

Using KVL, we can express $\mathbf{v}_{b_1}$ and $\mathbf{v}_{b_2}$ in terms of $\mathbf{V}_n$. This yields:

$$\left.\begin{aligned} -A_2\mathbf{i}_{b_2} &= \tfrac{d}{dt}A_1\tilde{\mathbf{q}}(t, A_1^T\mathbf{V}_n, A_2^T\mathbf{V}_n, \mathbf{i}_{b_2}) + A_1\tilde{\mathbf{j}}(t, A_1^T\mathbf{V}_n, A_2^T\mathbf{V}_n, \mathbf{i}_{b_2}), \\ A_2^T\mathbf{V}_n &= \tfrac{d}{dt}\bar{\mathbf{q}}(t, A_1^T\mathbf{V}_n, A_2^T\mathbf{V}_n, \mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t, A_1^T\mathbf{V}_n, A_2^T\mathbf{V}_n, \mathbf{i}_{b_2}). \end{aligned}\right\}$$

This system has $n + b_2$ equations and variables, but is again undetermined. Grounding node k at V^* and introducing $\hat{A}$ and $\hat{V}_n$ in the same way as in the previous paragraph, for an ordinary circuit, the system is described by the next equations[7] :

$$\left.\begin{aligned} -\hat{A}_2\mathbf{i}_{b_2} &= \tfrac{d}{dt}\hat{A}_1\tilde{\mathbf{q}}(t, X_1, X_2, \mathbf{i}_{b_2}) + \hat{A}_1\tilde{\mathbf{j}}(t, X_1, X_2, \mathbf{i}_{b_2}), \\ X_2 &= \tfrac{d}{dt}\bar{\mathbf{q}}(t, X_1, X_2, \mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t, X_1, X_2, \mathbf{i}_{b_2}). \end{aligned}\right\},$$

where $X_1 = \hat{A}_1^T\hat{\mathbf{V}}_n + V^*A_1^T\mathbf{e}_k$ and $X_2 = \hat{A}_2^T\hat{\mathbf{V}}_n + V^*A_2^T\mathbf{e}_k$.

Define the state vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T \end{pmatrix}^T = \begin{pmatrix} \hat{\mathbf{V}}_n^T & \mathbf{i}_{b_2}^T \end{pmatrix}^T$ and the functions $\mathbf{q}, \mathbf{j} : \mathbb{R} \times \mathbb{R}^{n-1} \times \mathbb{R}^{b_2} \to \mathbb{R}^{n-1+b_2}$, such that

$$\mathbf{q}(t, \mathbf{x}) = \begin{pmatrix} \hat{A}_1\tilde{\mathbf{q}}(t, \hat{A}_1^T\mathbf{x}_1 + V^*A_1^T\mathbf{e}_k, \hat{A}_2^T\mathbf{x}_1 + V^*A_2^T\mathbf{e}_k, \mathbf{x}_2) \\ \bar{\mathbf{q}}(t, \hat{A}_1^T\mathbf{x}_1 + V^*A_1^T\mathbf{e}_k, \hat{A}_2^T\mathbf{x}_1 + V^*A_2^T\mathbf{e}_k, \mathbf{x}_2) \end{pmatrix},$$

$$\mathbf{j}(t, \mathbf{x}) = \begin{pmatrix} \hat{A}_1\tilde{\mathbf{j}}(t, \hat{A}_1^T\mathbf{x}_1 + V^*A_1^T\mathbf{e}_k, \hat{A}_2^T\mathbf{x}_1 + V^*A_2^T\mathbf{e}_k, \mathbf{x}_2) + \hat{A}_2\mathbf{x}_2 \\ \bar{\mathbf{j}}(t, \hat{A}_1^T\mathbf{x}_1 + V^*A_1^T\mathbf{e}_k, \hat{A}_2^T\mathbf{x}_1 + V^*A_2^T\mathbf{e}_k, \mathbf{x}_2) - \hat{A}_2^T\mathbf{x}_1 - V^*A_2^T\mathbf{e}_k \end{pmatrix}.$$

[7] In general, a DAE does not have an unique solution.

As in the previous case, the circuit is mathematically described by the next differential-algebraic equation:

$$\frac{d}{dt}\mathbf{q}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}. \tag{2.3}$$

The Jacobian matrices of $\mathbf{q}$ and $\mathbf{j}$ are denoted by:

$$C(t,\mathbf{x}) = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}(t,\mathbf{x}), \quad G(t,\mathbf{x}) = \frac{\partial \mathbf{j}}{\partial \mathbf{x}}(t,\mathbf{x}). \tag{2.4}$$

These matrix functions are very important, because they are used by the numerical solvers of DAE's.

2.2.4 Contributions of components to q and j

The components listed in the previous section, are the building blocks of an electrical circuit. In the same way, the functions $\mathbf{q}$ and $\mathbf{j}$ can be composed from the elementary functions q and j, which belong to the components. The variables of the current-defined components are the nodal voltages at their positive and negative nodes, while the voltage-defined components also need the current. On each node, the sum of all contributions of the components which are connected to that node has to be zero. Because the current-equations do not contribute to the nodes, they belong to their components and they all have to be zero.

In Tab.(2.1), the local functions $\mathbf{q}, \mathbf{j}$ are shown for the basic components.

Component:	Variables:	$\mathbf{q}$	$\mathbf{j}$
Current Source	$\begin{pmatrix} v^+ \\ v^- \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} i^*(t) \\ -i^*(t) \end{pmatrix}$
Voltage Source	$\begin{pmatrix} v^+ \\ v^- \\ i_V \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} i_V \\ -i_V \\ v^+ - v^- - v^*(t) \end{pmatrix}$
Nonlinear capacitor	$\begin{pmatrix} v^+ \\ v^- \end{pmatrix}$	$\begin{pmatrix} f(v^+ - v^-) \\ -f(v^+ - v^-) \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$
Linear capacitor	$\begin{pmatrix} v^+ \\ v^- \end{pmatrix}$	$\begin{pmatrix} Cv^+ - Cv^- \\ Cv^- - Cv^+ \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$
Nonlinear inductor	$\begin{pmatrix} v^+ \\ v^- \\ i_L \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ -f(i_L) \end{pmatrix}$	$\begin{pmatrix} i_L \\ i_L \\ v^+ - v^- \end{pmatrix}$
Linear inductor	$\begin{pmatrix} v^+ \\ v^- \\ i_L \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ -Li_L \end{pmatrix}$	$\begin{pmatrix} i_L \\ -i_L \\ v^+ - v^- \end{pmatrix}$
Nonlinear current-defined resistor	$\begin{pmatrix} v^+ \\ v^- \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} f(v^+ - v^-) \\ -f(v^+ - v^-) \end{pmatrix}$
Nonlinear voltage-defined resistor	$\begin{pmatrix} v^+ \\ v^- \\ i_r \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} i_r \\ -i_r \\ v^+ - v^- - f(i_r) \end{pmatrix}$
Linear resistor	$\begin{pmatrix} v^+ \\ v^- \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{R}v^+ - \frac{1}{R}v^- \\ \frac{1}{R}v^- - \frac{1}{R}v^+ \end{pmatrix}$

Table 2.1: Local functions $\mathbf{q}$ and $\mathbf{j}$ for the components.

Besides the local functions $\mathbf{q}$ and $\mathbf{j}$, also the local Jacobian matrices $C = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}$ and $G = \frac{\partial \mathbf{j}}{\partial \mathbf{x}}$ are important. In Tab.(2.2) the local matrices C and G are shown for the basic components. Because all components have two or three variables, the matrices are $2 \times 2-$ or $3 \times 3-$ matrices. Also the behaviour of more complex components, such as transistors, can be described by $\mathbf{q}$- and $\mathbf{j}$-functions[8].

Component:	C	G
Current Source	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
Voltage Source	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}$
Nonlinear capacitor	$\begin{pmatrix} f'(v^+ - v^-) & -f'(v^+ - v^-) \\ -f'(v^+ - v^-) & f'(v^+ - v^-) \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
Linear capacitor	$\begin{pmatrix} C & -C \\ -C & C \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
Nonlinear inductor	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -f'(i_L) \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}$
Linear inductor	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -L \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}$
Nonlinear current-defined resistor	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} f'(v^+ - v^-) & -f'(v^+ - v^-) \\ -f'(v^+ - v^-) & f'(v^+ - v^-) \end{pmatrix}$
Nonlinear voltage-defined resistor	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & -f'(v^+ - v^-) \end{pmatrix}$
Linear resistor	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{pmatrix}$

Table 2.2: Local matrices C and G for the components.

We want to know the contributions of each branch to the complete matrices C and G.

According to the previous section, the state of the circuit is

$$\mathbf{x} = \begin{pmatrix} \mathbf{V}_n \\ \mathbf{i}_{b_2} \end{pmatrix}.$$

Consider a branch b with connected nodes i and j.

[8]For many components, the compact models are available at www.philipssemiconductors.com.

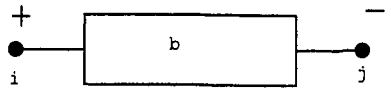


Figure 2.6: Picture of branch b.

Thus the voltages of the connected nodes are V_i and V_j, while the voltage difference v_b across the branch is equal to $V_i - V_j$. Because the nodal voltages belong to the state, there exist numbers k and l, such that $V_i = x_k = \mathbf{e}_k^T\mathbf{x}$ and $V_j = x_l = \mathbf{e}_l^T\mathbf{x}$. Introduce the vector $\mathbf{e} = \mathbf{e}_k - \mathbf{e}_l$, then $v_b = x_k - x_l = \mathbf{e}^T\mathbf{x}$. If the branch is voltage-defined, then the current through the branch, i_b, belongs to the state of the circuit. So there exists a number m, such that $i_b = \mathbf{x}_m = \mathbf{e}_m^T\mathbf{x}$. With use of the vectors $\mathbf{e}$ and $\mathbf{e}_m$, it is possible to get a short notation for the contributions $\Delta\mathbf{q}$, $\Delta\mathbf{j}$ of each branch to the complete functions $\mathbf{q}$ and $\mathbf{j}$ and the contributions ΔC, ΔG to the complete matrices C and G. In Tab.(2.3), these contributions are shown.

Component:	$\Delta\mathbf{q}$	$\Delta\mathbf{j}$	ΔC	ΔG
Current Source	$\mathbf{0}$	$i^*(t)\mathbf{e}$	0	0
Voltage Source	$\mathbf{0}$	$\mathbf{x}_m\mathbf{e}+$ $(x_l - x_k - v^*(t))\mathbf{e}_m$	0	$\mathbf{e}\mathbf{e}_m^T + \mathbf{e}_m\mathbf{e}^T$
Nonlinear capacitor	$f(x_k - x_l)\mathbf{e}$	$\mathbf{0}$	$f'(x_k - x_l)\mathbf{e}\mathbf{e}^T$	0
Linear capacitor	$(Cx_k - Cx_l)\mathbf{e}$	$\mathbf{0}$	$C\mathbf{e}\mathbf{e}^T$	0
Nonlinear inductor	$-f(x_m)\mathbf{e}_m$	$x_m\mathbf{e} + (x_k - x_l)\mathbf{e}_m$	$-f'(i_L)\mathbf{e}_m\mathbf{e}_m^T$	$\mathbf{e}\mathbf{e}_m^T + \mathbf{e}_m\mathbf{e}^T$
Linear inductor	$-Lx_m\mathbf{e}_m$	$x_m\mathbf{e} + (x_k - x_l)\mathbf{e}_m$	$-L\mathbf{e}_m\mathbf{e}_m^T$	$\mathbf{e}\mathbf{e}_m^T + \mathbf{e}_m\mathbf{e}^T$
Nonlinear current-defined resistor	$\mathbf{0}$	$f(x_k - x_l)\mathbf{e}$	0	$f'(x_k - x_l)\mathbf{e}\mathbf{e}^T$
Nonlinear voltage-defined resistor	$\mathbf{0}$	$x_m\mathbf{e}+$ $(x_k - x_l - f(x_m))\mathbf{e}_m$	0	$\mathbf{e}\mathbf{e}_m^T + \mathbf{e}_m\mathbf{e}^T$ $-f'(x_k - x_l)\mathbf{e}_m\mathbf{e}_m^T$
Linear resistor	$\mathbf{0}$	$\frac{1}{R}(x_k - x_l)\mathbf{e}$	0	$\frac{1}{R}\mathbf{e}\mathbf{e}^T$

Table 2.3: Contributions of branches to $\mathbf{q}$, $\mathbf{j}$, C and G.

In practice, this table is used to derive the functions $\mathbf{q}$, $\mathbf{j}$ and the matrices C and G. The method described in the previous section provides the theoretical background, but is not used in an explicit way. If there are only current-defined branches and for all components f' is positive, it is clear that the Jacobian matrices C and G are symmetric semi-positive definite for all t and $\mathbf{x}$.

2.3 Analysis of electrical circuits

After deriving the model of an electrical circuit, which is determined by the functions $\mathbf{q}$ and $\mathbf{j}$, one is able to analyse the circuit. In this section, several types of analysis-methods are shown.

2.3.1 Direct current analysis

Direct current analysis computes the steady-state solution $\mathbf{x}_0$ of the circuit. In a steady-state there are only time-invariant equations. This means that

$$\begin{aligned}\mathbf{q}(t, \mathbf{x}) &= \mathbf{q}_{DC}(\mathbf{x}),\\ \mathbf{j}(t, \mathbf{x}) &= \mathbf{j}_{DC}(\mathbf{x}).\end{aligned}$$

Furthermore the steady-state solution has the property:

$$\dot{\mathbf{x}}_{DC} = \mathbf{0}.$$

So, the steady-state equation is the next algebraic equation:

$$\mathbf{j}_{DC}(\mathbf{x}_{DC}) = \mathbf{0}.$$

In general, this is a nonlinear equation, which can be solved by e.g. the Newton-Raphson method[9]. If the equation is linear, Gaussian elimination is sufficient. Note that the matrix $G_{DC} = \frac{\partial \mathbf{j}_{DC}}{\partial \mathbf{x}}$ is important, because it is the matrix of all linear systems which have to be solved during the Newton-Raphson method.

2.3.2 Small signal analysis

This type of analysis[10] considers the effect of applying small signal perturbations $\mathbf{e}(t)$, with $\mathbf{e}(0) = \mathbf{0}$, to the equation of the DC-solution.

$$\frac{d}{dt}\mathbf{q}_{DC}(\mathbf{x}) + \mathbf{j}_{DC}(\mathbf{x}) - \mathbf{e}(t) = \mathbf{0}. \tag{2.5}$$

At $t = 0$, the solution is equal to the steady-state solution $\mathbf{x}_{DC}$, while the dynamic behaviour is caused by an independent small sine-wave excitation $\mathbf{e}(t)$, which is added to the circuit as a source function.

2.3.3 Transient analysis

This type of analysis[11] is very important, because the real nonlinear circuit, is simulated. In general, one is interested in the behaviour of the system on the time interval $[0, T]$. Normally the initial state $\mathbf{x}_0$ is known. It could be the steady-state solution, but this is not necessary.

For transient analysis, the next initial value problem (IVP) has to be solved:

$$\begin{cases} \frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}, \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases}$$

[9] This numerical method has been described in section 3.5.

[10] For more information, the reader is referred to [30].

[11] Information of the numerical background of the transient analysis with the circuit simulator Pstar can be found in [9].

Because it is a differential algebraic equation, not all initial states are allowed. If $\mathbf{q}$ is independent on t, the steady state solution at $t = 0$ is always a consistent initial solution. This IVP is also important for periodic steady state analysis, because for example the shooting method uses IVP-solvers. This IVP can be solved with numerical tools as LMM-methods or Runge Kutta methods. Because of the algebraic equations and the possible stiff behaviour, it is necessary to use implicit methods. In chapter 3, the numerical tools for the transient analysis are investigated.

2.3.4 Periodic steady-state analysis

In many circuits, there exists a periodic solution. This solution is often called the periodic steady state. In this case, instead of an initial value, a periodicity constraint is specified. In the periodic steady state, the functions $\mathbf{q}$ and $\mathbf{j}$ are periodic functions with respect to t.

The next two-point boundary value problem (BVP) has to be solved:

$$\begin{cases} \frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}, \\ \mathbf{x}(0) = \mathbf{x}(T), \end{cases}$$

where T is the period of the solution. If the functions $\mathbf{q}$ and $\mathbf{j}$ are periodic, the period T can be derived. But if that is not the case, it is still possible to have periodic solutions. Sometimes T is known, but T could also be an additional unknown of the system. In that case, the free oscillator problem[12] has to be solved.

These problems, with periodicity constraints, are more difficult to solve, because there are no initial conditions. If the period T is known, one could use e.g. a shooting method to solve this problem. For unknown period, an eigenvalue problem has to be solved, which is more difficult. However, for linear systems it is sufficient to determine the eigenvalues of the system, which determine the possible periods.

[12]For more information about this subject, the reader is referred to [22].

Chapter 3

Numerical analysis of differential-algebraic equations

3.1 Introduction

In previous chapter, it has been explained how the dynamics of electrical circuits can be mathematically modelled. For a transient analysis of a circuit, the initial value problem of a DAE has to be solved. This chapter will consider IVP's for DAE's and some common methods, which can be used to get approximated solutions.

There are several classical methods to compute the numerical solution of this DAE. All these methods discretize the time and compute the solution on the time grid with the help of integration methods. The timesteps have to be chosen in such a way that the global error will be small enough. Because it is hard to compute the global error, the local error is controlled.

First, in section two, some theoretical properties of DAE's will be mentioned. Afterwards, in section three, the transient analysis of ODE's with Runge Kutta methods has been studied. Also the Linear Multistep Methods have been introduced for fixed stepsizes. In section three, the transient analysis of DAE's has been considered. Here, some useful techniques have been considered, which can be used to avoid problems because of the algebraic equations. Afterwards, the LMM- and BDF-methods for variable stepsizes have been considered. Besides the numerical schemes, also the asymptotic models and estimates for the local discretization error are studied. It appears that the error model of multistep-methods with variable stepsizes are much more difficult than for one step methods. Also some special cases have been studied, such as the NDF-method, the TR-BDF2 method and the multirate approach. Finally, in section five, it has been considered how the nonlinear equations for implicit integration methods can be solved.

3.2 Theory of differential-algebraic equations.

3.2.1 Initial value problem

Consider a certain dynamical system[1] with state vector $\mathbf{x} \in \mathbb{R}^d$, which satisfies next differential-algebraic equation:

$$\begin{cases} \frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}, \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \tag{3.1}$$

[1] For more information about this subject, the reader is referred to [17, 18, 26, 28].

The functions $\mathbf{q} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ and $\mathbf{j} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ can be derived from physical laws. If the system is an electrical circuit, Modified Nodal Analysis[2] can be used to find $\mathbf{q}$ and $\mathbf{j}$. The solution $\mathbf{x}(t)$ of (3.1) describes the dynamic behaviour of the system for a known initial value, for example the steady state.

A special case of the DAE's is the ordinary differential equation. In that case, a transient analysis computes the solution of the next initial value problem:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}), \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \tag{3.2}$$

with $\mathbf{f} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$. A standard theorem states that if $\mathbf{f}(t, \mathbf{x})$ is continuous on $[0, T]$ and Lipschitz continuous with respect to $\mathbf{x}$, there exists[3] an unique continuously differentiable solution.

Unfortunately, most dynamical systems can not be represented by ODE's. In general, it is hard to determine, whether or not an analytical solution exists.

There are more representations of (3.1). Expanding the derivative of $\mathbf{q}$ results in:

$$C(t, \mathbf{x})\dot{\mathbf{x}} + \frac{\partial \mathbf{q}}{\partial t}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}. \tag{3.3}$$

From this representation , it follows that if $C(t, \mathbf{x})$ is invertible for all $\mathbf{x}$, the DAE can be written as an ODE. But in practice, $C(t, \mathbf{x})$ will be almost always singular, because of the algebraic equations. Then, the solution has to satisfy a number of algebraic equations. Because these algebraic equations also apply in $t = 0$, a proper initial solution has also to satisfy the algebraic equations. Such initial solution is called consistent.

This means also that not all initial values are consistent. However, if the initial solution is equal to the steady state, there are no problems, because the steady state always satisfies the algebraic equations.

In general, it is not possible to find the analytic solution in closed form. Hence, it is important to use accurate, robust and efficient tools to approximate this solution. The time is discretized in small timepoints, while for each timepoint the DAE is approximated by a numerical integration scheme. Every timestep, a small local error has been made, which affect the global error.

3.2.2 Stability

Besides the solvability, also the global stability of the system is very important.

Definition 3.1 *Consider the perturbated IVP of (3.1) with initial value $\hat{x}_0$ and solution $\hat{x}(t)$. The system is stable if:*

$$\forall_{\epsilon>0}\ \exists_{\delta>0}\ \|\hat{x}_0 - x_0\| < \delta \ \Rightarrow\ \forall_t\ \|\hat{x}(t) - x(t)\| < \epsilon.$$

Thus, stability ensures that the difference between the exact and the approximated solution remains small, if the initial value is changed. This is a useful property, because the local discretization errors of an integration method can be considered as perturbations of the initial value for that timepoint.

For many physical systems like electrical circuits, the time-dependent behaviour is only caused by source functions. This means that these systems can be described by the next DAE, where $\mathbf{u}(t)$ is an input function, which only depends on t.

$$\begin{cases} \frac{d}{dt}\mathbf{q}(\mathbf{x}) + \mathbf{j}(\mathbf{x}) + \mathbf{u}(t) = \mathbf{0}, \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \tag{3.4}$$

[2]This method has been described on page 13.

[3]See [11].

In general, it is difficult to check the global stability. A possible approach is to check the local stability around the initial steady state. In this case, the stability of the linearised systems at all possible times and states is determined. For a linear time-invariant system, it is well-known[4] that the system is stable, if the Jacobian matrix is a stable matrix. This means that all eigenvalues of the Jacobian have strict negative real parts.

Theorem 3.1 *Let*[5] x_0 *be the steady state of 3.4, with* $j(x_0) = 0$. *Consider the linearised homogeneous system around* x_0

$$C\dot{y} + Gy = 0, \tag{3.5}$$

with $C = \frac{\partial}{\partial \dot{x}} q(x_0)$ *and* $G = \frac{\partial}{\partial x} j(x_0)$. *This system is stable if all roots of the next equation have strict negative real part:*

$$det(sC + G) = 0.$$

If G *is invertible and if* $G^{-1}C$ *is a stable matrix, then this condition has been satisfied. If (3.5) is stable, then the nonlinear system is locally stable around* x_0.

However, this approach is only valid for small input signals. Therefore, also next approach can be used to determine whether the system is stable[6].

Theorem 3.2 *Consider the system (3.4). The matrix functions* $C(x)$ *and* $G(x)$ *are the Jacobians of the functions* q *and* j. *Assume that there exists a positive definite matrix* P, *such that for all* x *next matrix function is also positive definite:*

$$G(x)^T PC(x) + C(x)^T PG(x) > 0, \tag{3.6}$$

$$\dot{u}^T PC(x)\dot{x} + \dot{x}^T C(x)^T P\dot{u} \geq 0. \tag{3.7}$$

Then, the system is stable. Note that the first condition is sufficient for time-independent input signals.

Proof This theorem can be proved with use of the positive definite Lyapunov function

$$V(\mathbf{x}) = \frac{d}{dt}[\mathbf{q}(\mathbf{x})]^T P \frac{d}{dt}[\mathbf{q}(\mathbf{x})].$$

It is well known[7] that if there exists a positive definite matrix P, such that $\dot{V}(\mathbf{x}) < 0$, the system is stable. Because of the DAE, it follows that

$$\begin{aligned} \dot{V}(\mathbf{x}) &= \tfrac{d^2}{dt^2}[\mathbf{q}(\mathbf{x})]^T P \tfrac{d}{dt}[\mathbf{q}(\mathbf{x})] + \tfrac{d}{dt}[\mathbf{q}(\mathbf{x})]^T P \tfrac{d^2}{dt^2}[\mathbf{q}(\mathbf{x})] \\ &= -\tfrac{d}{dt}[\mathbf{j}(\mathbf{x}) + \mathbf{u}]^T P \tfrac{d}{dt}[\mathbf{q}(\mathbf{x})] - \tfrac{d}{dt}[\mathbf{q}(\mathbf{x})]^T P \tfrac{d}{dt}[\mathbf{j}(\mathbf{x}) + \mathbf{u}] \\ &= -\dot{\mathbf{x}}^T \left(G(\mathbf{x})^T PC(\mathbf{x}) + C(\mathbf{x})^T PG(\mathbf{x})\right)\dot{\mathbf{x}} - (\dot{\mathbf{u}}^T PC(\mathbf{x})\dot{\mathbf{x}} + \dot{\mathbf{x}}^T C(\mathbf{x})^T P\dot{\mathbf{u}}). \end{aligned}$$

Thus, if there exists a positive definite matrix P, such that

$$\begin{aligned} G(\mathbf{x})^T PC(\mathbf{x}) + C(\mathbf{x})^T PG(\mathbf{x}) > 0, \\ \dot{\mathbf{u}}^T PC(\mathbf{x})\dot{\mathbf{x}} + \dot{\mathbf{x}}^T C(\mathbf{x})^T P\dot{\mathbf{u}} > 0, \end{aligned}$$

it follows indeed that the system is stable.

[4] See [28].

[5] This theorem has been derived from [28].

[6] See [28].

[7] See [28].

□

If $G(\mathbf{x})$ is invertible for all $\mathbf{x}$, (3.6) is equivalent to next condition:

$$\exists_{P>0} PA(\mathbf{x}) + A(\mathbf{x})^T P > 0,$$

where $A(\mathbf{x}) = C(\mathbf{x})G(\mathbf{x})^{-1}$. Because of Lyapunov's theorem, this means that for all $\mathbf{x}$, $C(x)G(\mathbf{x})^{-1}$ must be a stable matrix. Note that this property is sufficient, but not necessary for stability.

With special choice $P = I$, it follows that

$$G(x)^T C(x) + C(x)^T G(x) > 0 \implies \text{stability for time-invariant circuit.}$$

If $C(x)$ and $G(x)$ are symmetric positive definite, this condition is satisfied. In [21], it has been proved that electrical circuits with only non-controlled, current-defined elements, have symmetric semi-positive definite Jacobians. Thus, in that case, it follows that $G(x)^T C(x) + C(x)^T G(x) = 2C(x)^T G(x)$. So, if C and G are invertible, the circuit is globally stable. Because, in general, C is not invertible, this is not the case.

3.2.3 Index of DAE's

The general form of an DAE[8] can be described as:

$$F(t, \mathbf{x}, \dot{\mathbf{x}}) = \mathbf{0}. \tag{3.8}$$

This equation consists of algebraic and differential equations. If $\frac{\partial F}{\partial \dot{\mathbf{x}}}$ is invertible, the DAE can be transformed into an ODE. If F represents the dynamics of an electrical circuit, in general, this will not be the case.

But after differentiating the DAE sufficient times and replacing the algebraic equations by the extra derived differential equations, it can become possible.

Definition 3.2 *The (global) index*[9] ν *of the DAE (3.8) is the necessary amount of differentiations to get an ODE.*

Clearly, ODE's have index $\nu = 0$. For (3.1), it follows that $\nu = 0$ if $C(t, \mathbf{x})$ is invertible. But in general, it is hard to determine the global index of a system.

Definition 3.3 *Consider the matrix pencils:* $\lambda C(t, \boldsymbol{x}) + G(t, \boldsymbol{x})$ *with* $\lambda \in \mathbb{C}$. *The DAE (3.1) is solvable if* $det(\lambda C(t, \boldsymbol{x}) + G(t, \boldsymbol{x}))$ *is not identically zero for all* λ. *If* $G(t, \boldsymbol{x})$ *is invertible and if* $-\frac{1}{\lambda}$ *is an eigenvalue of the matrix* $C(t, \boldsymbol{x})G(t, \boldsymbol{x})^{-1}$, *the pencils are never invertible.*

In next section, it will appear that this property also ensures that the system is solvable.

Theorem 3.3 *Consider*[10] *the invertible matrix pencil* $\lambda_n C(t_n, \boldsymbol{x}_n) + G(t_n, \boldsymbol{x}_n)$. *Then there exist non-singular matrices* P_n *and* Q_n, *such that*

$$P_n C(t_n, \boldsymbol{x}_n) Q_n = \begin{pmatrix} I & 0 \\ 0 & N_n \end{pmatrix}, \quad P_n G(t_n, \boldsymbol{x}_n) Q_n = \begin{pmatrix} A_n & 0 \\ 0 & I \end{pmatrix}. \tag{3.9}$$

[8] For more information about this subject, the reader is referred to [5, 8, 26, 36].

[9] See [26].

[10] This theorem has been derived from [5].

Here, I is the identity matrix, while N_n is a nilpotent matrix. The nilpotency index μ_n is defined as:

$$\mu_n = \min\{k \in \mathbb{N} : N_n^k = 0\}.$$

If $C(t_n, x_n)$ is invertible, define $\mu_n = 0$, because then N_n is empty. The nilpotency index is also called the local index of the DAE (3.1).

Definition 3.4 *Consider the DAE (3.1) with exact solution $x(t)$. Because of theorem (3.3), it follows that there exist invertible matrices P_n and Q_n such that $P_nC(t_n, x(t_n))Q_n$ and $P_nG(t_n, x_n)Q_n$ can be written like (3.9). Then, the local index μ_n at t_n is equal to the nilpotency index of N_n.*

Note that there is no close relationship between the local index and the global index.

The DAE is called semi-explicit if it can be written as:

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{x}, \mathbf{y}), \\ \mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{y}). \end{cases}$$

The benefit of this type is the strict separation between the differential equations and the algebraic equations.

After introducing the new variable $\mathbf{y} = \mathbf{q}(t, \mathbf{x})$, the DAE (3.1) can be written in the semi-explicit form:

$$\begin{cases} \dot{\mathbf{y}} = -\mathbf{j}(t, \mathbf{x}), \\ \mathbf{0} = \mathbf{y} - \mathbf{q}(t, \mathbf{x}). \end{cases}$$

3.3 Numerical methods for ODE's

3.3.1 General integration methods

Because, in general, ODE's are easier to solve than DAE's, this section will mention some methods[11] for problem (3.2). It is assumed that the exact solution $\mathbf{x}(t)$ exists on $[0, T]$. The time interval $[0, T]$ is discretized into a timegrid $\{t_n, n = 0, \ldots, N\}$, with $h_n = t_n - t_{n-1}$ the n-th timestep. The numerical approximation at $t = t_n$ is denoted by $\mathbf{x}_n$. Now, (3.2) is approximated with use of numerical discretization or integration methods. In general, this results in a large system of N nonlinear equations with N unknowns. Usually, it is possible to compute $\mathbf{x}_n$ for $n \in \{1, \ldots, N\}$ in a recursive manner:

$$\mathbf{g}(t_n, \mathbf{x}_{n-k}, \ldots, \mathbf{x}_n) = \mathbf{0}. \tag{3.10}$$

If $\mathbf{x}_n$ only depends on the previous solution $\mathbf{x}_{n-1}$, the method is an one step method, e.g. the Runge Kutta methods. Multistep Methods use more history, which makes them more complex to analyse.

Definition 3.5 *Consider the numerical solutions $\{x_n, n = 0, \ldots, N\}$ at the timegrid $\{t_n, n = 0, \ldots, N\}$.*

- *The global error e_n is equal to $x(t_n) - x_n$.*
- *The local discretization error (LDE) is equal to the residue of (3.10), after inserting the exact solution: $\delta_n = g(t_n, x(t_{n-k}), \ldots, x(t_n))$.*

[11] For more information about the numerical analysis of ODE's, see [11, 16, 26, 31].

- *Let x_n^* be the solution of the numerical scheme (3.10), if all previous solutions are exact.*

$$g(t_n, x(t_{n-k}), \ldots, x(t_{n-1}), x_n^*) = 0.$$

Then, the local error is equal to $d_n = x(t_n) - x_n^*$.

A numerical scheme is called consistent with order p, if for constant stepsize h, the local discretization error $\delta_n = O(h^{p+1})$. The scheme is convergent with order p, if also the global errors satisfy $\|\mathbf{e}_n\| = O(h^p)$. Theorem (3.4) gives some important relations between the local and global error.

Theorem 3.4 *Assume that method (3.10) is consistent with order p. Then, the local error satisfies*

$$\left(\frac{\partial g}{\partial x_n}\right) d_n = \delta_n.$$

The global error satisfies the following equation:

$$\frac{\partial g}{\partial x_{n-k}} e_{n-k} + \ldots + \frac{\partial g}{\partial x_n} e_n \doteq \delta_n.$$

This is equivalent to

$$\left(\frac{\partial g}{\partial x_n}\right) e_n \doteq -\left(\frac{\partial g}{\partial x_{n-k}} e_{n-k} + \ldots + \frac{\partial g}{\partial x_{n-1}} e_{n-1}\right) + \left(\frac{\partial g}{\partial x_n}\right) d_n.$$

Proof For the local error, it holds that

$$\begin{aligned} \mathbf{0} &= \mathbf{g}(t_n, \mathbf{x}(t_{n-k}), \ldots, \mathbf{x}(t_{n-1}), \mathbf{x}_n^*) \\ &= \delta_n - \frac{\partial \mathbf{g}}{\partial \mathbf{x}_n} \mathbf{d}_n. \end{aligned}$$

Because of the definition of δ_n and (3.10), it follows that

$$\begin{aligned} \mathbf{0} &= \mathbf{g}(t_n, \mathbf{x}_{n-k}, \ldots, \mathbf{x}_n) \\ &\doteq \delta_n - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}_{n-k}} \mathbf{e}_{n-k} + \ldots + \frac{\partial \mathbf{g}}{\partial \mathbf{x}_n} \mathbf{e}_n\right). \end{aligned}$$

□

Note that there exists a close relationship between the local error and the global error. It appears that the global error also depends on the previous global errors. Although theorem (3.4) holds, this asymptotic relation can only be used to estimate the global error at the next timepoint.

Because the rather complex behaviour of the global error, the LDE or the local error are controlled instead. If a method is consistent with order p, it is possible to control the local discretization error with use of the stepsizes. Then, it is necessary to use an estimate $\hat{\delta}_n$ of the LDE with $\hat{\delta}_n = \delta_n + O(h^{p+2})$.

If a method is convergent, this means that the global errors tend to zero, if $h \to 0$. However, in practice the stepsizes are always positive numbers. In that case, stability is also an important property of a numerical scheme.

Definition 3.6 *A method is called absolutely stable*[12] *for a given step size h and a given differential equation if the change due to a perturbation of size ϵ in one of the mesh values x_n is not larger than ϵ in all subsequent values x_m, $m > n$.*

[12] See [11].

In practice, the absolute stability is investigated for the test equation $\dot{y} = \lambda y$ with $\lambda \in \mathbb{C}$. This results in a stability region $S \subset \mathbb{C}$, with

$$h\lambda \in S \Rightarrow \text{Method is stable for linear test equation with eigenvalue } \lambda.$$

Definition 3.7 *A method is called $A(\alpha)$-stable with $0 < \alpha < \frac{\pi}{2}$ if*

$$S \supseteq S_\alpha = \{z : |arg(-z)| < \alpha, z \neq 0\}.$$

A method is called absolutely stable (A-stable) or inconditionally stable if it is $A(\frac{\pi}{2})$-stable.

3.3.2 Runge Kutta methods

Description of the method

One step methods have the nice property that using variable stepsizes makes no difference. Runge Kutta methods are one step methods, but they use more advanced quadrature formulas to improve the accuracy of the results. The ODE is integrated on the interval $[t_{n-1}, t_n]$, so

$$\mathbf{x}_n - \mathbf{x}_{n-1} = \int_{t_{n-1}}^{t_n} \mathbf{f}(\tau, \mathbf{x}(\tau))d\tau.$$

But now the integral is approximated by a quadrature formula of higher order:

$$\int_{t_{n-1}}^{t_n} \mathbf{f}(\tau, \mathbf{x}(\tau))d\tau = h_n \sum_{m=1}^{s} \beta_m \mathbf{f}(t_{n-1,m}, \mathbf{x}_{n-1,m}).$$

This requires intermediate values $\mathbf{x}_{n-1,m}$ for which we apply a similar quadrature formula. This yields:

$$\begin{cases} t_{n-1,m} = t_{n-1} + \rho_m h_n, \\ \mathbf{x}_{n-1,m} = \mathbf{x}_{n-1} + h_n \sum_{l=1}^{s} \gamma_{m,l} \mathbf{f}(t_{n-1,l}, \mathbf{x}_{n-1,l}). \end{cases}$$

Define the next vectors in $\mathbb{R}^s$:

$$\begin{aligned} \mathbf{e} &= (1, \ldots, 1)^T, \\ \beta &= (\beta_1, \ldots, \beta_s)^T, \\ \rho &= (\rho_1, \ldots, \rho_s)^T. \end{aligned}$$

Define the matrix $\Gamma \in \mathbb{R}^{ss}$ by:

$$\Gamma = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1s} \\ \vdots & \ddots & \vdots \\ \gamma_{s1} & \cdots & \gamma_{ss} \end{pmatrix}.$$

Now the RK formula is determined by the vectors β and ρ and the matrix Γ. A compact notation is the Butcher matrix denoted by:

$$\begin{array}{c|c} \rho & \Gamma \\ \hline & \beta^T \end{array}.$$

If the method is explicit, the matrix Γ is a strictly lower triangular matrix.

Now, define the vector $\tilde{t}_{n-1} \in \mathbb{R}^s$ and the stage vector $Y_{n-1} \in \mathbb{R}^{sd}$ by

$$\tilde{t}_{n-1} = \begin{pmatrix} t_{n-1} + \rho_1 h_n \\ \vdots \\ t_{n-1} + \rho_s h_n \end{pmatrix}, \quad Y_{n-1} = \begin{pmatrix} \mathbf{x}_{n-1,1} \\ \vdots \\ \mathbf{x}_{n-1,s} \end{pmatrix}.$$

Thus, the stage vector Y_{n-1} consists of the approximated solutions at the intermediate timepoints. Define also the function $\bar{\mathbf{f}} : \mathbb{R}^s \times \mathbb{R}^{sd} \to \mathbb{R}^{sd}$ by

$$\bar{\mathbf{f}}(t, Y) = \begin{pmatrix} \mathbf{f}(t_1, (Y_1, \ldots, Y_d)^T) \\ \vdots \\ \mathbf{f}(t_s, (Y_{(s-1)d+1}, \ldots, Y_{sd})) \end{pmatrix}.$$

Then the Runge Kutta method performs the next iteration step:

- Solve the stage vector Y_{n-1} from the nonlinear algebraic equation

$$Y_{n-1} = \mathbf{e} \otimes \mathbf{x}_{n-1} + (\Gamma \otimes I_d) h_n \bar{\mathbf{f}}(\bar{t}_{n-1}, Y_{n-1}). \tag{3.11}$$

- Compute the next solution $\mathbf{x}_n$ with the explicit one step formula:

$$\mathbf{x}_n = \mathbf{x}_{n-1} + (\beta^T \otimes I_d) h_n \bar{\mathbf{f}}(\bar{t}_{n-1}, Y_{n-1}). \tag{3.12}$$

In general, this results in very large systems, which has to be solved. Therefore, often the matrix Γ will has a special form. For DIRK methods, Γ is a lower triangular matrix. In this case, s systems of normal size has to be solved. If all diagonal elements are equal, only one matrix factorisation has to be performed, which reduces the amount of work. A disadvantage of RK methods is the high numer of function evaluations, which is needed.

Convergence

Because of the definition of subsection (3.3.1), the local discretization error is the residue of the Runge Kutta method, after inserting the exact solution. The Runge Kutta method is called consistent with order p if this local discretization error is $O(h_n^{p+1})$. Note that the numerical approximations at the intermediate timepoints do not need to be consistent with the same order.

Theorem 3.5 *An RK-method with s stages, can be consistent with order $p \leq s$. Only for $s \in \{1, \ldots, 4\}$ there exist schemes with $p = s$. Furthermore, RK-methods have the next property.:*

An RK-method is consistent with order p. $\Leftrightarrow$ *An RK-method is convergent with order p.*

If an RK-method is consistent with order larger or equal to one, the parameters have to obey the next relations:

$$\begin{cases} \beta^T \boldsymbol{e} = 1, \\ \Gamma \boldsymbol{e} = \rho. \end{cases}$$

Proof See for example [26]. It is assumed that $\mathbf{f}$ is Lipschitz continuous.

□

Because of the equation (3.12), it follows that an RK-method can be described in the next way:

$$\mathbf{x}_n = \mathbf{x}_{n-1} + h_n \Phi(t_n, \mathbf{x}_{n-1}, h_n).$$

Then the local discretization error δ_n is equal to:

$$\delta_n = [\mathbf{x}(t_n) - \mathbf{x}(t_{n-1}) - h_n \Phi(t_n, \mathbf{x}(t_{n-1}), h_n)].$$

If the order of the method is equal to p, then for each n there exists a $C_{p,n}$, such that

$$\delta_n = C_{p,n} h_n^{p+1} + O(h_n^{p+2}).$$

Theorem 3.6 *Consider a higher order RK-method with*

$$\bar{x}_n = x_{n-1} + h_n \bar{\Phi}(t_n, x_{n-1}, h_n)$$

and local discretization error:

$$\bar{\delta}_n = [x(t_n) - x(t_{n-1}) - h_n \bar{\Phi}(t_n, x(t_{n-1}), h_n)] = O(h_n^{p+2}).$$

Now, the LDE can be estimated by

$$\hat{\delta}_n = \bar{x}_n - x_n.$$

Proof It follows that $\bar{\delta}_n = \delta_n - C_{p,n} h_n^{p+1} + O(h_n^{p+2})$, so

$$C_{p,n} h_n^{p+1} + O(h_n^{p+2}) = \delta_n - \bar{\delta}_n = \bar{\Phi}(t_n, \mathbf{x}(t_{n-1}), h_n) - \Phi(t_n, \mathbf{x}(t_{n-1}), h_n) = \frac{1}{h_n}(\bar{\mathbf{x}}_n - \mathbf{x}_n).$$

Thus

$$\delta_n = \hat{\delta}_n + O(h_n^{p+2}).$$

□

Usually, an RK-method is embedded in such higher order method, to make error estimate easy. This higher order method is allowed to be less stable than the lower order one, because only one step is performed.

Now, the global error satisfies:

$$\mathbf{e}_n = \mathbf{x}_n - \mathbf{x}(t_n) = \mathbf{x}_{n-1} + h_n \Phi(t_n, \mathbf{x}_{n-1}, h_n) - \mathbf{x}(t_{n-1}) - h_n \Phi(t_n, \mathbf{x}(t_{n-1}), h_n) + \delta_n \doteq \mathbf{e}_{n-1} + \delta_n.$$

Because the Runge Kutta method is an one step method, the global error increment only depends on δ_n.

Applying the RK-method to the linear test-equation $\dot{y} = \lambda y$ gives[13]

$$y_n = \phi(\bar{h}) y_{n-1} = (1 + \beta^T \bar{h}(I_d - \bar{h}\Gamma)^{-1}\mathbf{e}) y_{n-1}.$$

with $\bar{h} = h\lambda$. The method is stable in $\bar{h}$ if $|\phi(\bar{h})| \leq 1$. The region of stability[14] S is defined by:

$$S = \{\bar{h} \in \mathbb{C} : |\phi(\bar{h})| \leq 1\}$$

with boundary region

$$\partial S = \{\bar{h} \in \mathbb{C} : |\phi(\bar{h})| = 1\}.$$

Examples

Below some Butcher matrices of common RK-methods are shown:

[13] See [26].

[14] This stability region has been defined on page 25.

Table 3.1: Butcher matrices of RK-methods.

Method :	Butcher matrix	order
Euler forward	$\begin{array}{c\|c} 0 & 0 \\ \hline & 1 \end{array}$	1
Euler backward	$\begin{array}{c\|c} 1 & 1 \\ \hline & 1 \end{array}$	1
Trapezoidal method	$\begin{array}{c\|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	2
Heun2	$\begin{array}{c\|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	2
Heun3	$\begin{array}{c\|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$	3
RK Fehlberg4	$\begin{array}{c\|cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 \\ \frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 \\ 1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 \\ \hline & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} \end{array}$	4
RK Fehlberg5	$\begin{array}{c\|ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\ \frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\ 1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\ \frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \\ \hline & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \end{array}$	5

In Fig.(3.1), the stability regions of these methods are drawn. Because for all $\bar{h}$ on the boundary ∂S of S, $|\phi(\bar{h})| = 1$, we have:

$$\forall_{\bar{h}\in\partial S}\exists_{\theta\in[-\pi,\pi]}\phi(\bar{h}) = e^{i\theta}.$$

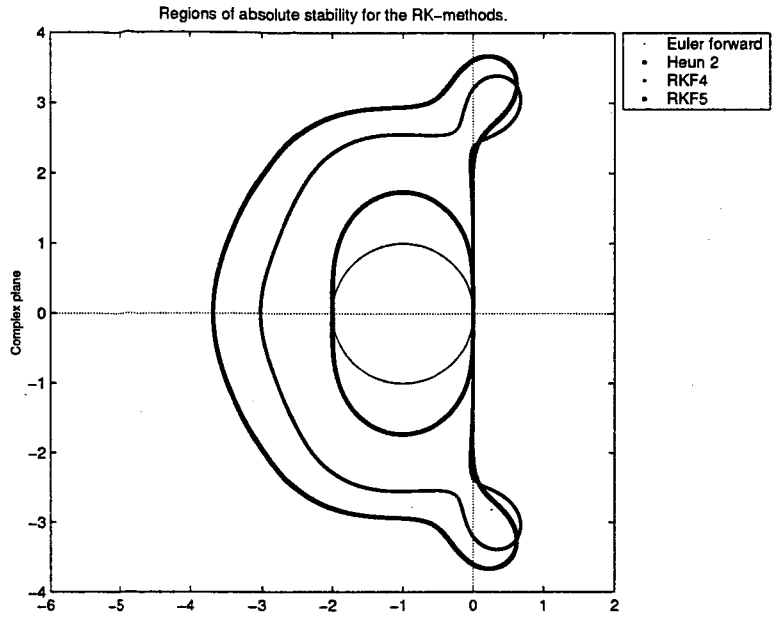


Figure 3.1: The stability regions of the explicit Runge Kutta methods. The methods are stable inside the contours. From small to large, the stability regions belong to Euler Forward, Heun 2, RKF4 and RKF5.

From this figure, it is clear that these RK-methods are not inconditionally stable[15]. In [26], it has been proved that all inconditionally stable RK-methods are also implicit methods. Because the examples are explicit, they are not inconditionally stable. This means also that numerical solutions with high frequencies will be amplified.

3.3.3 Linear Multistep Methods with fixed step

Besides one step methods, it is also possible to use Linear Multistep Methods. In contradiction to RK-methods, they do not use only the last numerical solution. This leads to smoother solutions, but can cause problems with discontinuities. For variable stepsizes, it is more difficult to analyse them, because now the local errors depend also on the previous stepsizes. Below, first the LMM-methods with fixed timesteps $h_n = h$ have been considered, while on page 44, the LMM-methods with variable stepsizes has been investigated. Consider the initial value problem (3.2). A general representation of the LMM-method to approximate the solution is:

$$\sum_{m=0}^{k} \rho_m \mathbf{x}_{n-m+k} = h \sum_{m=0}^{k} \sigma_m \mathbf{f}(t_{n-m+k}, \mathbf{x}_{n-m+k}). \tag{3.13}$$

A k-step method needs k given initial values $\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}$ to start. If only $\mathbf{x}_0$ is known, the other values can be computed iteratively with methods with increasing amount of steps. Implicit schemes, which are used, have the property that $\sigma_k \neq 0$.

[15]This concept has been defined in Def.(3.7).

Define the vector $\mathbf{r}_n \in \mathbb{R}^n$, which is known at $t = t_n$.

$$\mathbf{r}_n = -\left(\frac{1}{h}\sum_{m=0}^{k-1} \rho_m \mathbf{x}_{n+m-k} - \sum_{m=0}^{k-1} \sigma_m \mathbf{f}(t_{n+m-k}, \mathbf{x}_{n+m-k})\right)$$

Then each timestep, the next nonlinear algebraic equation has to be solved:

$$\frac{\rho_k}{h}\mathbf{x}_n - \sigma_k \mathbf{f}(t_n, \mathbf{x}_n) = \mathbf{r}_n$$

With for example the Newton-Raphson method, this equation can be solved. A proper initial guess for the Newton process would be $\mathbf{x}_n^0 = \mathbf{x}_{n-1}$, but better predictions can be made by means of extrapolation.

Two important classes of LMM-methods are the Adams-Moulton methods and the BDF methods.

Adams-Moulton

Define for $i \in \{0, \ldots, k\}$ the Lagrange interpolation-polynomials $L_i(t)$ with

$$L_i(t_{n+m-k}) = \delta_{im}.$$

Then on $[t_{n-k}, t_n]$:

$$\mathbf{f}(t, \mathbf{x}) \doteq \sum_{m=0}^{k} L_m(t)\mathbf{f}(t_{n+m-k}, \mathbf{x}_{n+m-k}).$$

Integrating the ODE on the interval $[t_{n-1}, t_n]$ gives:

$$\mathbf{x}_n - \mathbf{x}_{n-1} = \int_{t_{n-1}}^{t_n} \mathbf{f}(\tau, \mathbf{x}(\tau))d\tau \doteq \sum_{m=0}^{k}\left(\int_{t_{n-1}}^{t_n} L_m(\tau)d\tau\right)\mathbf{f}(t_{n+m-k}, \mathbf{x}_{n+m-k}).$$

Hence for ρ_m, σ_m in equation (3.13) we derive:

$$\forall_{m\in\{0,\ldots,k\}}\ \rho_m = \begin{cases} 1 & m \in \{k-1, k\}, \\ 0 & \text{otherwise}, \end{cases}$$
$$\forall_{m\in\{0,\ldots,k\}}\ \sigma_m = \frac{1}{h}\int_{t_{n-1}}^{t_n} L_m(\tau)d\tau$$

and

$$\mathbf{x}_n - \mathbf{x}_{n-1} \doteq \sum_{m=0}^{k} h\sigma_m \mathbf{f}(t_{n+m-k}, \mathbf{x}_{n+m-k}).$$

BDF

With the same interpolation polynomials, it is also possible to approximate the derivative. The solution obeys the next approximation:

$$\mathbf{x}(t) \doteq \sum_{m=0}^{k} L_m(t)\mathbf{x}_{n+m-k}.$$

Differentiating the ODE in $t = t_n$, we get:

$$h\mathbf{f}(t_n, \mathbf{x}_n) = h_n \left.\frac{d}{dt}\mathbf{x}(t)\right|_{t=t_n} \doteq \sum_{m=0}^{k} h\left(\left.\frac{d}{dt}L_m(t)\right|_{t=t_n}\right)\mathbf{x}_{n+m-k}.$$

Hence for ρ_m, σ_m in equation (3.13) we derive:

$$\forall_{m\in\{0,\ldots,k\}}\ \rho_m = h\ \frac{d}{dt}L_m(t)\Big|_{t=t_n},$$
$$\forall_{m\in\{0,\ldots,k\}}\ \sigma_m = \begin{cases} 1 & m = k, \\ 0 & \text{otherwise} \end{cases}$$

and

$$h\mathbf{f}(t_n, \mathbf{x}_n) \doteq \sum_{m=0}^{k} \rho_m \mathbf{x}_{n+m-k}.$$

Convergence

Define the polynomials:

$$\rho(z) = \textstyle\sum_{m=0}^{k} \rho_m z^m, \quad \sigma(z) = \sum_{m=0}^{k} \sigma_m z^m.$$

Consider the shift-operator q with[16] $(q\ \mathbf{x})_n = \mathbf{x}_{n+1}$. Now it is possible to describe an LMM-method with use of the shift-operator as follows:

$$\rho(q)\mathbf{x}_{n-k} = h\sigma(q)\mathbf{f}(t_{n-k}, \mathbf{x}_{n-k}). \tag{3.14}$$

Theorem 3.7 *An LMM-method is consistent with order p if and only if the polynomials $\rho(z)$ and $\sigma(z)$ satisfy next equations:*

$$\begin{gathered} \rho(1) = 0, \\ \rho'(1) - \sigma(1) = 0, \\ \forall_{s\in\{2,\ldots,p\}} \textstyle\sum_{m=0}^{k} \left[\rho_m (m-k)^s - s\sigma_m (m-k)^{s-1}\right] = 0. \end{gathered} \tag{3.15}$$

The local discretization error[17] *(LDE) of a consistent LMM-method with order p and fixed stepsizes can be written like:*

$$\delta_n = C_p h^{p+1} x^{(p+1)}(t_n) + O(h^{p+2})$$

with $C_p = \sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^{p+1}}{(p+1)!} - \sigma_m \frac{(m-k)^p}{p!}\right]$. *For BDF-methods,* $C_p = -\frac{1}{p+1} = -\frac{1}{k+1}$[18].

Proof The local discretization error (LDE) δ_n is equal to

$$\delta_n = \rho(q)\mathbf{x}(t_{n-k}) - h\sigma(q)\mathbf{f}(t_{n-k}, \mathbf{x}(t_{n-k})).$$

Then, δ_n can be approximated in the next way:

$$\begin{aligned} \delta_n &= \textstyle\sum_{m=0}^{k} \rho_m \mathbf{x}(t_{n+m-k}) - h \sum_{m=0}^{k} \sigma_m \mathbf{f}(t_{n+m-k}, \mathbf{x}(t_{n+m-k})) \\ &= \textstyle\sum_{m=0}^{k} \rho_m \mathbf{x}(t_n) + h \sum_{m=0}^{k} [\rho_m (m-k) - \sigma_m]\, \dot{\mathbf{x}}(t_n) + h^2 \sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^2}{2} - \sigma_m (m-k)\right] \ddot{\mathbf{x}}(t_n) \\ &\quad + \ldots + h^p \textstyle\sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^p}{p!} - \sigma_m \frac{(m-k)^{p-1}}{(p-1)!}\right] \mathbf{x}^{(p)}(t_n) + O(h^{p+1}). \end{aligned} \tag{3.16}$$

Because an LMM-method is consistent with order p if $\delta_n = O(h^{p+1})$, indeed $\rho(z)$ and $\sigma(z)$ has to satisfy the equations (3.15).

[16] This is often also denoted by $q\mathbf{x}_n = \mathbf{x}_{n+1}$.

[17] This concept has been defined on page 24.

[18] See [31]

□

An estimate for the local discretization error can be derived as follows:

$$\delta_n = C_p(\mathbf{x}_n - \mathbf{x}_n^0) + O(h^{p+2}),$$

where $\mathbf{x}_n^0 = p_n(t_n)$ with p_n the interpolation polynomial on the $(p+1)$ last time grid points[19]. So, the difference between the initial estimate and the final solution of the Newton-Raphson method can be used to estimate the LDE in a cheap way.

Definition 3.8 *Consider a polynomial $P(z)$. Then, $P(z)$ obeys the root condition, if*[20]*:*

$$\begin{cases} \forall_{z\in\mathbb{C}} P(z) = 0 \Rightarrow |z| \leq 1, \\ \forall_{z\in\mathbb{C}} (P(z) = 0 \wedge z \textit{ is not simple.}) \Rightarrow |z| < 1. \end{cases} \tag{3.17}$$

The LMM-method is called root-stable if $\rho(z)$ obeys the root condition (3.17).

Theorem 3.8 *An LMM-scheme is convergent with order k for an ordinary ODE, if it is both consistent with order k and root stable.*

Proof See [26, 31].

□

Because convergent LMM-methods are root-stable, it always holds for these LMM-methods that $\rho'(1) \neq 0$, because 1 can only be a simple root of $\rho(z)$. Then, it follows from the equation (3.15) that also $\sigma(1) \neq 0$.

With help of the estimate of the local discretization error, it is possible to give an asymptotic estimate of the global error[21]. Define the function $\mathbf{d}$ which obeys the next IVP:

$$\begin{cases} \dot{\mathbf{d}} = \frac{\partial}{\partial \mathbf{x}}\mathbf{f}(t, \mathbf{x})\mathbf{d} + C_p\mathbf{x}^{(p)}(t), \\ \mathbf{d}(0) = \mathbf{0}. \end{cases}$$

Then $\mathbf{e}_n = \mathbf{d}(t_n, \mathbf{x}(t_n))h^{p+1} + O(h^{p+2})$. So, it follows that the behaviour of the global error depends on the local discretization error, but also on the stability of the ODE. For stable ODE's it is much easier to control the global error than for unstable ODE's.

To check the absolute stability, the LMM-method is performed for the linear test equation, which results in:

$$\rho(q)y_{n-k} = \bar{h}\sigma(q)y_{n-k}$$

with $\bar{h} = \lambda h$. Now, the characteristic polynomial $\pi(z, \bar{h})$ of the numerical scheme is equal to $\rho(z) - \bar{h}\sigma(z)$. Then

$$\pi(q, \bar{h})y_{n-k} = 0.$$

The method is absolutely stable (A-stable) in $\bar{h}$ if $\pi(z, \bar{h})$ obeys the root condition[22]. Note that root stability is equivalent to absolutely stability in $\bar{h} = 0$.

[19] This subject has been explained in more detail on page 48.

[20] Note that a root is called simple if its multiplicity is one.

[21] See [26].

[22] See equation (3.17).

Now it is possible to derive stability regions[23] for the LMM-methods with:

$$S = \{\bar{h} \in \mathbb{C} : \pi(z, \bar{h}) \text{ satisfies the root condition.}\}. \tag{3.18}$$

There exists a theorem[24], which states that the order of absolutely stable LMM-methods is at most 2, while the order of $A(\alpha)$-stable[25] methods is at most 6. In practice, the stability regions for special nonlinear functions f may be quite different from these stability regions. But after linearizing f at $x = x^*$, the eigenvalues of $\frac{\partial f}{\partial x}\Big|_{x=x^*}$ can be computed. The timestep h has to be chosen, such that for all eigenvalues $h\lambda \in S$.

Examples

Below the polynomials $\rho(z)$ and $\sigma(z)$ are listed for some well-known methods with constant stepsizes. In general, the coefficients depend on the step sizes, but if the step sizes are constant, $\rho(z)$ and $\sigma(z)$ are polynomials, independent of the step size.

Table 3.2: Polynomials $\rho(z)$ and $\sigma(z)$ for some Adams-Moulton methods

Method :	$\rho(z)$	$\sigma(z)$	order	C_p
Euler Backward (Adams-Moulton 0)	$z-1$	z	1	$-\frac{1}{2}$
Trapezoidal method (Adams-Moulton 1)	$z-1$	$\frac{1}{2}(z+1)$	2	$-\frac{1}{12}$
Adams-Moulton 2	z^2-z	$\frac{1}{12}(5z^2+8z-1)$	3	$-\frac{1}{24}$

Note that both Euler Backward and the Trapezoidal method are one-step methods.

Table 3.3: Polynomials $\rho(z)$ and $\sigma(z)$ for some BDF methods

Method :	$\rho(z)$	$\sigma(z)$	*order*	C_p
Euler Backward (BDF1)	$z-1$	z	1	$-\frac{1}{2}$
BDF2	$\frac{1}{2}(3z^2-4z+1)$	z^2	2	$-\frac{1}{3}$
BDF3	$\frac{1}{6}(11z^3-18z^2+9z-2)$	z^3	3	$-\frac{1}{4}$
BDF4	$\frac{1}{12}(25z^4-48z^3+36z^2-16z+3)$	z^4	4	$-\frac{1}{5}$
BDF5	$\frac{1}{60}(137z^5-300z^4+300z^3-200z^2+75z-12)$	z^5	5	$-\frac{1}{6}$
BDF6	$\frac{1}{60}(147z^6-360z^5+450z^4-400z^3+225z^2-72z+10)$	z^6	6	$-\frac{1}{7}$

Note that the error constant C_p is smaller in the case of the Adams-Moulton-methods. However, this method is not absolutely stable. In the Figures (3.2) and (3.3), the stability regions of these methods have been drawn. For the boundary ∂S of the stability region S, which has been defined in equation 3.18, it holds that

$$\partial S = \{\bar{h} \in \mathbb{C} : \text{All roots of } \pi(z, \bar{h}) = \rho(z) - \bar{h}\sigma(z) \text{ have absolute values equal to one.}\}.$$

[23] These stability regions have been defined on page 26.

[24] From Dahlquist, see [26].

[25] Absolutely and $A(\alpha)$-stability has also been introduced on page 26.

Thus, it follows that

$$\forall_{\bar{h}\in\partial S}\ \exists_{\theta\in[-\pi,\pi]}\ \bar{h} = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}.$$

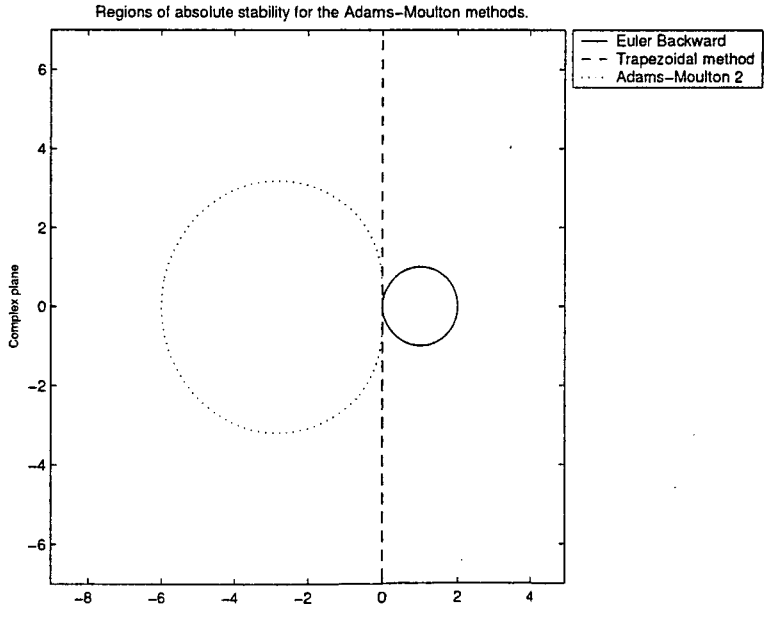


Figure 3.2: The Euler Backward method is absolutely stable outside the circle, the trapezoidal method is absolutely stable in the negative half plane, while the Adams-Moulton-methods of higher order are absolutely stable inside their stability regions.

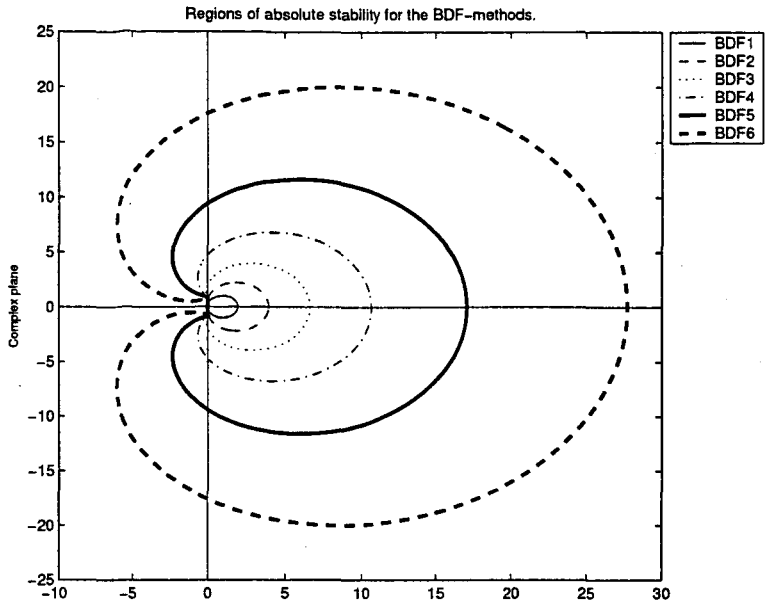


Figure 3.3: All BDF methods are absolutely stable for the test equation outside their indicated closed contour.

From the stability-regions in Figures (3.3) and (3.2), it is clear that BDF1 and BDF2 are inconditionally stable, while BDF3 up to BDF6 are at most $A(\alpha)$-stable[26] due to the Dahlquist-barrier[27]. The Trapezoidal is the linear one-step method with the highest possible order and the smallest discretization error constant C_p. Furthermore, its numerical stability corresponds well with the stability of the linear test equation. The Adams-Moulton 2 method is not $A(\alpha)$ stable, because its stability region is inside the contour.

In the case of stiff equations, it is possible that there exist eigenvalues λ with $\text{Re}[\lambda] << 1$. If the stability-region of the numerical method is not $A(\alpha)$ stable, the stepsize h has to be chosen very small to get $h\lambda \in S$. Because the BDF-methods with $k \in \{1, \ldots, 6\}$ are $A(\alpha)$-stable, these methods are often used for stiff problems.

Besides the stability, also the damping of oscillatory solutions is an important property. The BDF methods damp the solution for high frequencies, but for lower frequencies all higher order methods amplify the solution. The Trapezoidal method has no damping, while for $|\omega| > \omega_0$ the solutions are amplified with the Adams-Moulton 2 method.

Theoretically, the Trapezoidal method has very nice properties. However, in practice one wants damping for very large frequencies to filter high frequency errors. Furthermore, the Trapezoidal method gives damped oscillations if $\text{Re}[\bar{h}] < 1$.

[26] These concepts have been defined in Def.(3.7).

[27] See [26].

3.3.4 TR-BDF2 method

Description

A successful numerical method[28] is the combination of the trapezoidal one-step method and the BDF2 two-step method. First an internal stage vector $\mathbf{x}^*_{n-1}$ is computed with the TR-method from the previous $\mathbf{x}_{n-1}$. Afterwards the next solution value $\mathbf{x}_n$ is computed with the BDF2 method using $\mathbf{x}_{n-1}$ and $\mathbf{x}^*_{n-1}$. Generally, $\mathbf{x}^*_{n-1}$ is the approximation at $t_{n-1} + \frac{1}{2}\omega h$ with $\omega \in (0, 2)$. Note that the coefficients for the BDF2-method will depend on ω.

First, $\mathbf{x}^*_{n-1}$ is computed, where $h_1 = \omega\frac{1}{2}h$.

$$\mathbf{x}^*_{n-1} - \mathbf{x}_{n-1} = h_1(\frac{1}{2}\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1}) + \frac{1}{2}\mathbf{f}(t^*_{n-1}, \mathbf{x}^*_{n-1})).$$

Afterwards, $\mathbf{x}_n$ is computed, where $h_2 = (1 - \omega\frac{1}{2})h$.

$$(2 - \frac{1}{2}\omega)\mathbf{x}_n - \frac{2}{\omega}\mathbf{x}^*_{n-1} + \frac{(\omega-2)^2}{2\omega}\mathbf{x}_{n-1} = h_2\mathbf{f}(t_n, \mathbf{x}_n).$$

The TR-BDF2-method can also be viewed as an implicit Runge-Kutta method with 3 stages[29]. The only difference is that the TR-BDF2 method also uses the intermediate stage solutions, while a Runge-Kutta method only uses the final solution. Because

$$\begin{aligned}\mathbf{x}^*_{n-1} &= \mathbf{x}_{n-1} + \tfrac{\omega}{4}h\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1}) + \tfrac{\omega}{4}h\mathbf{f}(t^*_{n-1}, \mathbf{x}^*_{n-1}),\\ \mathbf{x}_n &= \mathbf{x}_{n-1} + \tfrac{1}{\omega-4}h\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1}) + \tfrac{1}{\omega-4}h\mathbf{f}(t^*_{n-1}, \mathbf{x}^*_{n-1}) + \tfrac{2-\omega}{4-\omega}h\mathbf{f}(t_n, \mathbf{x}_n),\end{aligned}$$

it can be derived that the method has the next Butcher matrix:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{\omega}{2} & \frac{\omega}{4} & \frac{\omega}{4} & 0 \\ 1 & \frac{1}{4-\omega} & \frac{1}{4-\omega} & 1 - \frac{2}{4-\omega} \\ \hline & \frac{1}{4-\omega} & \frac{1}{4-\omega} & 1 - \frac{2}{4-\omega} \end{array}.$$

If one wants all diagonal terms the same, we get:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \gamma & d & d & 0 \\ 1 & w & w & d \\ \hline & w & w & d \end{array}$$

with $\omega = \omega^* = 4 - 2\sqrt{2}$, $\gamma = \frac{\omega^*}{2}$, $d = \frac{\omega^*}{4}$ and $w = \frac{\sqrt{2}}{4}$. The benefit of this special case is the fact that the same Newton iteration matrix can be used to evaluate all implicit stages, because the diagonal terms are the same. It can be proved[30] that the error can be estimated with use of the next higher order method with Butcher matrix

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \gamma & d & d & 0 \\ 1 & w & w & d \\ \hline & \frac{1-w}{3} & \frac{3w+1}{3} & \frac{d}{3} \end{array}.$$

Because both RK-methods have the same Γ and ρ, the LDE can be computed in a cheap way.

[28] See [20].

[29] See [20].

[30] See [20].

Local discretization error

Because the TR-BDF2 method is a Runge Kutta method, the method does not need to be consistent at t^*_{n-1}. It appears that this implies that the TR-BDF2 has a smaller error constant than the BDF2-method and the Trapezoidal method.

Theorem 3.9 *The local discretization error of the TR-BDF2 method satisfies next equation:*

$$\delta_n \doteq C_k(\omega)h^3x^{(3)}(t_n).$$

with

$$C_k(\omega) = \frac{-1}{48}(3\omega^2 - 8\omega + 8).$$

Proof Assume that $\mathbf{x}_{n-1} = \mathbf{x}(t_{n-1})$ is exact. Then it follows that

$$\mathbf{x}^*_{n-1} \doteq \mathbf{x}(t^*_{n-1}) + (I - h_1\frac{1}{2}\mathbf{J}(t^*_{n-1}, \mathbf{x}(t^*_{n-1})))^{-1}\frac{1}{12}h_1^3\mathbf{x}^{(3)}(t^*_{n-1}).$$

Furthermore,

$$\begin{aligned}
\delta_n &\doteq (2 - \tfrac{1}{2}\omega)\mathbf{x}(t_n) - \tfrac{2}{\omega}\mathbf{x}(t^*_{n-1}) + \tfrac{(\omega-2)^2}{2\omega}\mathbf{x}(t_{n-1}) - h_2\mathbf{f}(t_n, \mathbf{x}(t_n)) \\
&\quad -\tfrac{2}{\omega}(I - h_1\tfrac{1}{2}\mathbf{J}(t^*_{n-1}, \mathbf{x}(t^*_{n-1})))^{-1}\tfrac{1}{12}h_1^3\mathbf{x}^{(3)}(t^*_{n-1}) \\
&\doteq \tfrac{1}{3\omega-6}h_2^3\mathbf{x}^{(3)}(t_n) - \tfrac{2}{\omega}(I - h_1\tfrac{1}{2}\mathbf{J}(t^*_{n-1}, \mathbf{x}(t^*_{n-1})))^{-1}\tfrac{1}{12}h_1^3\mathbf{x}^{(3)}(t^*_{n-1}) \\
&\doteq \tfrac{1}{3\omega-6}(1 - \tfrac{1}{2}\omega)^3h^3\mathbf{x}^{(3)}(t_n) - \tfrac{2}{\omega}(I - \tfrac{1}{4}\omega h\mathbf{J}(t^*_{n-1}, \mathbf{x}(t^*_{n-1})))^{-1}\tfrac{1}{12}(\tfrac{1}{2}\omega)^3h^3\mathbf{x}^{(3)}(t_n).
\end{aligned}$$

Because $(I - \frac{1}{4}\omega h\mathbf{J}(t^*_{n-1}, \mathbf{x}(t^*_{n-1})))^{-1} = I + O(h)$, it follows that

$$\begin{aligned}
\delta_n &= (\tfrac{1}{3\omega-6}(1 - \tfrac{1}{2}\omega)^3 - \tfrac{2}{\omega}\tfrac{1}{12}(\tfrac{1}{2}\omega)^3)h^3\mathbf{x}^{(3)}(t_n) + O(h^4) \\
&= \tfrac{-1}{48}(3\omega^2 - 8\omega + 8)h^3\mathbf{x}^{(3)}(t_n) + O(h^4) \\
&= C_k(\omega)h^3\mathbf{x}^{(3)}(t_n) + O(h^4).
\end{aligned}$$

□

Because $C_k(\omega) = \frac{-3}{48}(\omega - \frac{4}{3})^2 - \frac{1}{18}$, the error constant is minimal for $\omega = \frac{4}{3}$, with $C_k(\frac{4}{3}) = \frac{-1}{18} \approx -0.0556$. Note, that this method is more efficient than the normal Trapezoidal method. However, this method is a Runge-Kutta method with two stages. Therefore, this method needs two matrix factorisations. Suppose that the Jacobian $\mathbf{J}$ does not depends on t and $\mathbf{x}$, then both Jacobians are equal if $1 - \frac{1}{2}h_1\mathbf{J} = \lambda(2 - \frac{1}{2}\omega - h_2\mathbf{J})$. This is the case for $\omega^* = 4 - 2\sqrt{2}$ and now only one LU-decomposition suffices. For this case, the error constant is just a bit larger, with $C_k(\omega^*) = \frac{2}{3}\sqrt{2} - 1 \approx -0.0572$.

Compared with the BDF2-method, the numer of stepsizes for the TR-BDF2-method is reduced to 56%, for EPUS-control. However, the TR-BDF2 method needs one extra function evaluation and one extra matrix factorisation, if $\omega \neq \omega^*$.

Stability

The stability-region of the TR-BDF2 as RK-method has been shown in Fig.(3.4) for $\omega \in \{0.5, 1, \omega^*, 1.5\}$ together with the stability-regions of the Trapezoidal method and the BDF2 method.

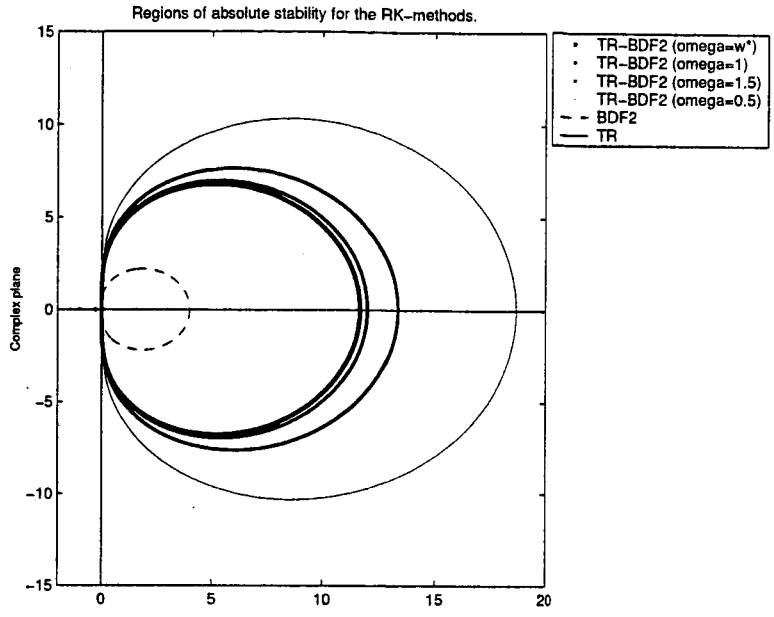


Figure 3.4: The stability region of the TR-BDF2 methods compared with the stability region of the Trapezoidal method and the BDF2-method. Because they are implicit methods, the methods are stable outside the contour.

Because the TR-BDF2 method is implicit, the high frequencies will be damped. In Fig.(3.5), the amplification factor has been shown for several cases. From this figure, it is clear that the TR-BDF2 has a filter property that is better than the BDF2-method.

Therefore, the TR-BDF2-method is recommended for circuits with oscillatory behaviour. The high-frequent noise will be damped, while the lower-frequent oscillations are better conserved than the BDF2-method.

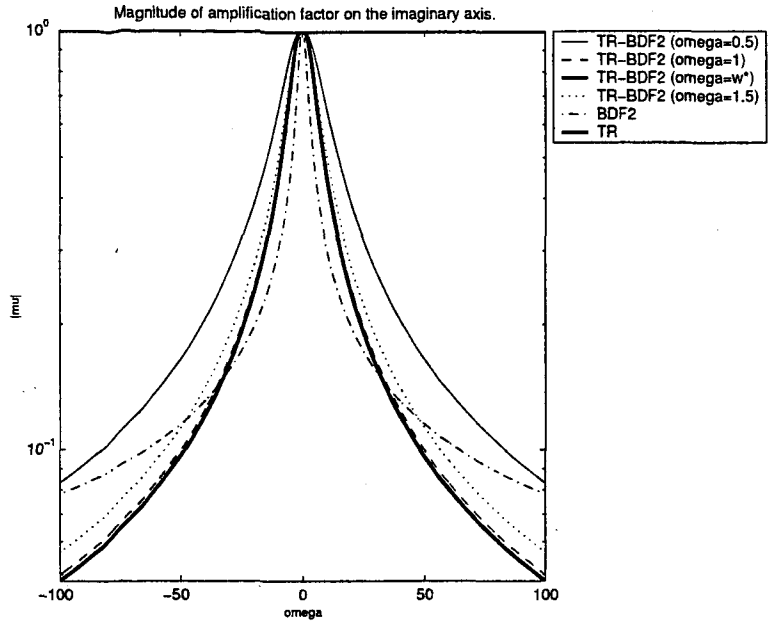


Figure 3.5: The filter behaviour of the TR-BDF2 methods, compared to the BDF2 method.

In Fig.(3.6), for $\omega \in (0, 2)$, the maximal frequencies have been shown for fixed maximal damping constants. Clearly, for $\omega \approx 2$ or $\omega \approx 0$, the damping becomes much smaller. Also for the BDF2-method, the maximal frequencies have been shown.

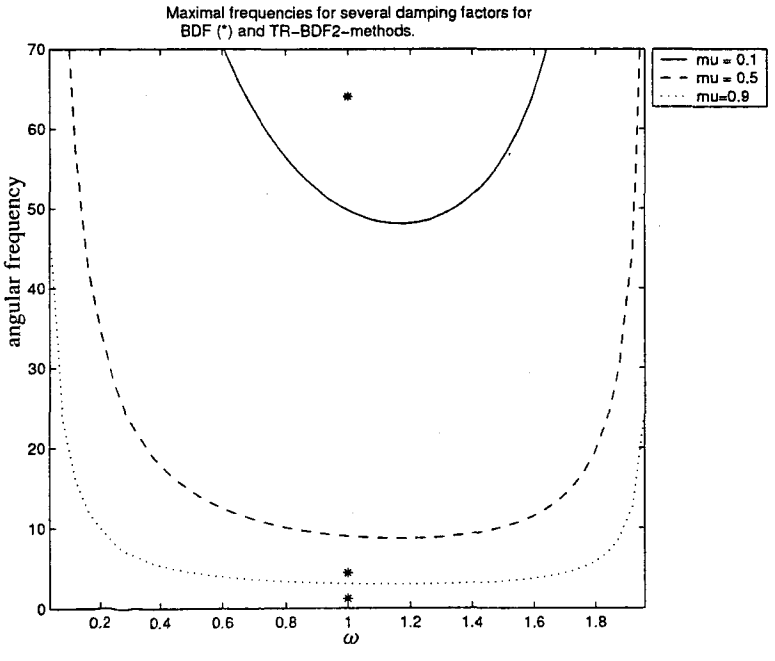


Figure 3.6: The maximal oscillation frequencies for damping factors 0.1, 0.5 and 0.9. The maximal frequencies for the BDF2 method have been identified with the marker *.

3.4 Numerical methods for DAE's

In this section, the Linear Multistep Methods[31] for the DAE (3.1) will be studied. First, some theoretical results for LMM-methods[32] for schemes for DAE's have been shown. Afterwards, the LMM-methods with variable step have been investigated. As special cases, the BDF- and NDF-methods are considered.

3.4.1 Convergent numerical schemes for DAE's

Transferability of DAE

Consider the index-1 DAE

$$C(t, \mathbf{x})\dot{\mathbf{x}} + \frac{\partial \mathbf{q}}{\partial t}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}. \tag{3.19}$$

Introduce the projections $Q(t)$ on the kernel $\mathcal{N}(C(t, \mathbf{x}))$ and $P(t)$ on the range of $C(t, \mathbf{x})$ with

$$\begin{cases} \forall_x\ C(t, \mathbf{x})Q(t)\mathbf{x} = \mathbf{0}, \\ P(t) = I - Q(t). \end{cases}$$

Now the differentiated variables are $P(t)\mathbf{x}$, while $Q(t)\mathbf{x}$ are the algebraic variables. Note that $\forall_x C(t, \mathbf{x})P(t)\mathbf{x} = C(t, \mathbf{x})\mathbf{x}$. Because P and Q are projections, they also satisfy the next equations:

$$\begin{array}{rcl} Q^2(t) & = & Q(t) \\ P^2(t) & = & P(t) \\ Q(t)P(t) & = & P(t)Q(t) = 0 \end{array} \tag{3.20}$$

Definition 3.9 *The DAE (3.19) is transferable*[33] *when*

1. q_t, *G and C are continuous for all t and* $\mathbf{x}$.
2. *The nullspace of C depends only on t.*
3. *C has constant rank.*
4. *The projector* $Q(t)$ *is smooth.*
5. $C(t, \mathbf{x}) + G(t, \mathbf{x})Q(t)$ *is invertible.*
6. G^{-1} *is bounded.*
7. *The DAE (3.19) has a smooth solution.*

[31] For more information about this subject, the reader is referred to [5]. Here also Runge Kutta methods for DAE's have been considered.

[32] In [5], also the convergence of Runge Kutta methods for DAE's has been studied.

[33] See [5].

Numerical schemes with projections for DAE's

There holds

$$C(t,\mathbf{x})P(t)\dot{\mathbf{x}} + \frac{\partial \mathbf{q}}{\partial t}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = C(t,\mathbf{x})\dot{\mathbf{x}} + \frac{\partial \mathbf{q}}{\partial t}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}). \tag{3.21}$$

Thus, the DAE (3.19) can be written as follows:

$$\begin{cases} P(t)\dot{\mathbf{x}} - \mathbf{y} = \mathbf{0}, \\ C(t,\mathbf{x})\mathbf{y} + \mathbf{q}_t(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}. \end{cases} \tag{3.22}$$

Note that now the second equation has become algebraic. An LMM-method which treats this problem is:

$$\begin{cases} P_n(\frac{1}{h_n}\rho(q)\mathbf{x}_{n-k} - \sigma(q)\mathbf{y}_{n-k}) - Q_n\mathbf{y}_n = \mathbf{0}, \\ C(t_n,\mathbf{x}_n)P_n\mathbf{y}_n + \mathbf{q}_t(t_n,\mathbf{x}_n) + \mathbf{j}(t_n,\mathbf{x}_n) = \mathbf{0}. \end{cases} \tag{3.23}$$

Introduce for each time step the vector $\mathbf{r}_n$ with

$$\mathbf{r}_n = P_n(-\frac{1}{h_n}\sum_{m=0}^{k-1}\rho_m\mathbf{x}_{n-k+m} + \sum_{m=0}^{k-1}\sigma_m\mathbf{y}_{n-k+m})$$

Then for each time step, the next equations have to be solved:

$$\begin{cases} P_n(\frac{\rho_k}{h_n}\mathbf{x}_n - \sigma_k\mathbf{y}_n) - Q_n\mathbf{y}_n = \mathbf{r}_n, \\ C(t_n,\mathbf{x}_n)P_n\mathbf{y}_n + \mathbf{q}_t(t_n,\mathbf{x}_n) + \mathbf{j}(t_n,\mathbf{x}_n) = \mathbf{0}. \end{cases}$$

Because of the properties[34] of $Q(t)$ and $P(t)$, it follows that $Q_n\mathbf{y}_n = \mathbf{0}$. Furthermore, $\mathbf{y}_n$ satisfies $P_n\mathbf{y}_n = \frac{1}{\sigma_k}(\frac{1}{h_n}P_n\rho_k\mathbf{x}_n - \mathbf{r}_n)$. After eliminating $P_n\mathbf{y}_n$, the second equation yields:

$$C(t_n,\mathbf{x}_n)\frac{1}{\sigma_k}(\frac{1}{h_n}P_n\rho_k\mathbf{x}_n - \mathbf{r}_n) + \mathbf{q}_t(t_n,\mathbf{x}_n) + \mathbf{j}(t_n,\mathbf{x}_n) = \mathbf{0}.$$

Because $C(t,\mathbf{x})P(t)\mathbf{x} = C(t,\mathbf{x})\mathbf{x}$, this is equivalent to the next equation:

$$C(t_n,\mathbf{x}_n)(\rho_k\mathbf{x}_n - h_n\sigma_k\mathbf{r}_n) + h_n\sigma_k\mathbf{q}_t(t_n,\mathbf{x}_n) + h_n\sigma_k\mathbf{j}(t_n,\mathbf{x}_n) = \mathbf{0}.$$

Because of (3.20), the equation $P(t)\dot{\mathbf{x}} - \mathbf{y} = \mathbf{0}$ is equivalent to

$$P(t)[(P(t)\mathbf{x})' - P'(t)\mathbf{x} - \mathbf{y}] - Q(t)\mathbf{y} = \mathbf{0}.$$

Now, the DAE (3.19) can also be written as:

$$\begin{cases} P(t)[(P(t)\mathbf{x})' - P'(t)\mathbf{x}] - P(t)\mathbf{y} - Q(t)\mathbf{y} = \mathbf{0}, \\ C(t,\mathbf{x})\mathbf{y} + \mathbf{q}_t(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}. \end{cases}$$

This formulation is more general, since it only requires $P\mathbf{x}$ to be differentiable instead of $\mathbf{x}$. With the next LMM-method, it can be solved.

$$\begin{cases} P_n\left(\frac{1}{h_n}\rho(q)[P_{n-k}\mathbf{x}_{n-k}] - \sigma(q)[P'_{n-k}\mathbf{x}_{n-k} + P_{n-k}\mathbf{y}_{n-k}]\right) - Q_n\mathbf{y}_n = \mathbf{0}, \\ C(t_n,\mathbf{x}_n)\mathbf{y}_n + \mathbf{q}_t(t_n,\mathbf{x}_n) + \mathbf{j}(t_n,\mathbf{x}_n) = \mathbf{0}. \end{cases} \tag{3.24}$$

[34] See equation (3.20).

Introduce for each time step the vector $\mathbf{r}_n$ with

$$r_n = P_n \left(-\frac{1}{h_n} \sum_{m=0}^{k-1} \rho_m [P_{n-k+m} \mathbf{x}_{n-k+m}] + \sum_{m=0}^{k-1} \sigma_m [P'_{n-k+m} \mathbf{x}_{n-k+m} + P_{n-k+m} \mathbf{y}_{n-k+m}] \right).$$

Now, for each time step, the next equations have to be solved:

$$\begin{cases} P_n \left(\frac{\rho_k}{h_n} \mathbf{x}_n - \sigma_k P'_n \mathbf{x}_n - \sigma_k \mathbf{y}_n \right) - Q_n \mathbf{y}_n = r_n, \\ C(t_n, \mathbf{x}_n) P_n \mathbf{y}_n + \mathbf{q}_t(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}. \end{cases}$$

Again, it is possible to eliminate $P_n \mathbf{y}_n$, which results in:

$$C(t_n, \mathbf{x}_n) \left(\frac{\sigma_k}{\rho_k h_n} P_n \mathbf{x}_n - P_n P'_n \mathbf{x}_n + \frac{1}{\sigma_k} \mathbf{r}_n \right) + \mathbf{q}_t(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}.$$

Theorem 3.10 *If the DAE is transferable*[35] *and the LMM-method is stable and consistent with order p for ODE's, then*

1. *Method (3.24) applied to the DAE has also order p.*
2. *Method (3.23) applied to the DAE has order at least one. Generally, the order is less than p, but the BDF-methods have still the same order p.*

One leg methods

Besides the LMM-methodws, there is also a family of integration methods, which are called one-leg methods. For an ODE, their numerical scheme is equal to:

$$\frac{1}{h_n} \rho(q) \mathbf{x}_{n-k} = \mathbf{f}(\sigma(q) t_{n-k}, \sigma(q) \mathbf{x}_{n-k}).$$

Note that these methods use only one function evaluation per step. It is clear that the BDF-methods also are one-leg methods. It appears that one-leg methods can be extended to DAE's in a natural way. For the equation (3.19), we get:

$$C(\sigma(q) t_{n-k}, \sigma(q) \mathbf{x}_{n-k}) \frac{1}{h_n} \rho(q) \mathbf{x}_{n-k} + \mathbf{q}_t(\sigma(q) t_{n-k}, \sigma(q) \mathbf{x}_{n-k}) + \mathbf{j}(\sigma(q) t_{n-k}, \sigma(q) \mathbf{x}_{n-k}) = \mathbf{0}.$$

For these methods, the next theorem holds.

Theorem 3.11 *If a one-leg method is consistent*[36] *with order p for an ODE, it is also consistent with order p for the DAE (3.19).*

This implies that BDF-methods do not need the projections to converge with the same order.

[35] This theorem has been derived from [5], page 69.

[36] This theorem has been derived from [5], page 71.

3.4.2 Linear Multistep Methods

Introduction

In the previous section, the LMM-methods have been studied for fixed stepsizes. In this section, the same method will be considered for variable stepsizes[37]. Introduce the coefficients $\xi_{n,m}$, which are useful to analyse the method, if the stepsizes are not fixed.

$$\xi_{n,m} = \frac{t_n - t_{n-m}}{h_n}. \tag{3.25}$$

For fixed stepsizes, it holds that $\xi_{n,m} = m$.

A general representation of the LMM-methods with variable step and fixed order is:

$$\sum_{m=0}^{k} \rho_{n,m}\mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h_n \sum_{m=0}^{k} \sigma_{n,m}\mathbf{j}(t_{n-m}, \mathbf{x}_{n-m}) = \mathbf{0}.$$

In general, the coefficients $\rho_{n,m}$ and $\sigma_{n,m}$ are dependent on the stepsizes. Only for fixed stepsizes, they are constant for all n. The LMM-methods with k steps need k initial values. In each iteration, the next nonlinear equation has to be solved:

$$\rho_{n,0}\mathbf{q}(t_n, \mathbf{x}_n) + h_n\sigma_{n,0}\mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{r}_n. \tag{3.26}$$

Here, $\mathbf{r}_n$ is a vector, which is already known:

$$\mathbf{r}_n = \sum_{m=1}^{k} \rho_{n,m}\mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h_n \sum_{m=1}^{k} \sigma_{n,m}\mathbf{j}(t_{n-m}, \mathbf{x}_{n-m}).$$

If the Jacobians C and G of the functions $\mathbf{q}$ and $\mathbf{j}$ exist, this equation is often solved by the Newton-Raphson method. The Jacobian of this nonlinear equation is equal to $\mathbf{J}(t, \mathbf{x}) = \rho_{n,0}C(t, \mathbf{x}) + h_n\sigma_{n,0}G(t, \mathbf{x})$. To perform this method, it is necessary that the matrix pencil $\mathbf{J}(t_n, \mathbf{x}_n)$ is invertible for all n. Because C is not invertible, this means that $\sigma_{n,0}$ must be nonzero. So, the DAE of the equation (3.1) may not be solvable[38] with explicit LMM methods.

Predictor Corrector method

For LMM-methods, polynomials are attractive to store the previous data.

Assume that $k+1$ solutions $\{\mathbf{x}_{n-k-1}, \ldots, \mathbf{x}_{n-1}\}$ are already known at the time grid $\{t_{n-k-1}, \ldots, t_{n-1}\}$ and $h_{n-1} = t_{n-1} - t_{n-2}$. Then, the necessary history is stored by the "corrector" polynomials $C_{q,n}(t) : \mathbb{R} \to \mathbb{R}^n$ and $C_{j,n}(t) : \mathbb{R} \to \mathbb{R}^n$ of degree k, which satisfy the next conditions.

$$\begin{cases} C_{q,n-1}(t_{n-m}) &= \mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}), \quad m \in \{1, \ldots, k+1\}, \\ C_{j,n-1}(t_{n-m}) &= \mathbf{j}(t_{n-m}, \mathbf{x}_{n-m}). \end{cases} \tag{3.27}$$

These corrector polynomials can be used to predict the solution $\mathbf{x}_n$ at t_n. Therefore, the predictor polynomials $P_{q,n}(t) : \mathbb{R} \to \mathbb{R}^n$ and $P_{j,n}(t) : \mathbb{R} \to \mathbb{R}^n$ of degree k are defined by:

$$\begin{aligned} P_{q,n}(t) &= C_{q,n-1}(t), \\ P_{j,n}(t) &= C_{j,n-1}(t). \end{aligned} \tag{3.28}$$

[37] For more information about LMM-methods with variable stepsizes, see [40].

[38] The solvability of a DAE has been introduced on page 23.

The corrector polynomials $C_{q,n}(t)$ and $C_{j,n}(t)$ are the corrected versions of $P_{q,n}(t)$ and $P_{j,n}(t)$, which have the correct value at t_n. More precisely, we express the corrector polynomials in the predictor polynomials and the Lagrange basis polynomial $L_n(t)$, with

$$\begin{aligned} L_n(t_{n-m}) &= 0, \quad m \in \{1, \ldots, k\}, \\ L_n(t_n) &= 1. \end{aligned} \tag{3.29}$$

Now, the corrector polynomials are equal to:

$$\begin{cases} C_{q,n}(t) = P_{q,n}(t) + (\mathbf{q}(t_n, \mathbf{x}_n) - P_{q,n}(t_n))L_n(t), \\ C_{j,n}(t) = P_{j,n}(t) + (\mathbf{j}(t_n, \mathbf{x}_n) - P_{j,n}(t_n))L_n(t). \end{cases} \tag{3.30}$$

So, the corrector (C) and predictor (P) polynomials are equal on the timepoints $\{t_{n-k}, \ldots, t_{n-1}\}$. Furthermore, $C_{q,n}(t_n)$ and $C_{j,n}(t_n)$ are the correct values at t_n, while $P_{q,n}(t_n)$ and $P_{j,n}(t_n)$ are the predicted values based on extrapolation of the $k+1$ previous values.

In practice, also the predictor and corrector polynomials for $\mathbf{x}$ are stored. These polynomials can be used to determine an initial guess for the Newton-Raphson method.

Now, the DAE is approximated by

$$\frac{d}{dt} C_{q,n}(t) + C_{j,n}(t) = \mathbf{0}. \tag{3.31}$$

From $C_{q,n}(t_n)$ and $C_{j,n}(t_n)$, the next solution $\mathbf{x}_n$ can be determined. There are several ways to compute $\mathbf{x}_n$, which result in several kinds of numerical schemes.

Nordsieck representation

Introduction The Nordsieck representation[39] corresponds closely to the Predictor Corrector method. The previous solutions are represented by a polynomial. Furthermore, changing stepsizes becomes easy, when using Nordsieck vectors. Without loss of generality, we only consider $C_{q,n}(t)$ and $P_{q,n}(t)$. For $C_{j,n}(t)$ and $P_{j,n}(t)$ similar results can be obtained.

The corrector polynomial $C_{q,n-1}(t)$ can be written like a truncated Taylor series around t_{n-1}:

$$C_{q,n-1}(t) = \sum_{m=0}^{k} \left(\frac{h_{n-1}^m \frac{d^m}{dt^m} C_{q,n-1}(t_{n-1})}{m!} \right) \left(\frac{t - t_{n-1}}{h_{n-1}} \right)^m.$$

The Nordsieck vector $\bar{\mathbf{C}}_{q,n-1} \in \mathbb{R}^{n \times k+1}$ of this polynomial contains all coefficients of this polynomial, i.e.

$$\bar{\mathbf{C}}_{q,n-1} = \left(C_{q,n-1}(t_{n-1}), h_{n-1} \frac{d}{dt} C_{q,n-1}(t_{n-1}), \frac{h_{n-1}^2}{2} \frac{d^2}{dt^2} C_{q,n-1}(t_{n-1}), \ldots, \frac{h_{n-1}^k}{k!} \frac{d^k}{dt^k} C_{q,n-1}(t_{n-1}) \right).$$

If the state dimension n is larger than one, in fact $\bar{\mathbf{C}}_{q,n-1}$ is a matrix. Now, the polynomial and its Nordsieck vector are related by next equation:

$$C_{q,n-1}(t) = \sum_{m=0}^{k} \bar{\mathbf{C}}_{q,n-1,m} \left(\frac{t - t_{n-1}}{h_{n-1}} \right)^m. \tag{3.32}$$

[39] See [11, 32, 40].

Predictor polynomial Because of the equation (3.28), it follows that $P_{q,n}(t) = C_{q,n-1}(t)$. This means for the Nordsieck vectors that $\bar{\mathbf{P}}_{j,n} = \bar{\mathbf{C}}_{q,n}$. Because only $\bar{\mathbf{C}}_{q,n-1}$ is known, the Nordsieck vector $\bar{\mathbf{P}}_{q,n}$ of $P_{q,n}(t)$ has to be computed.

Theorem 3.12 *Define the matrices* $H_{k,\omega} \in \mathbb{R}^{k+1\times k+1}$ *and* $P_k \in \mathbb{R}^{k+1\times k+1}$ *with*

$$H_{k,\omega} = \begin{pmatrix} 1 & & & \\ & \omega & & \\ & & \ddots & \\ & & & \omega^k \end{pmatrix}, P_k = \begin{pmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ 1 & 3 & 3 & 1 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ 1 & k & \binom{k}{2} & \binom{k}{3} & \dots & 1 \end{pmatrix}.$$

Here P_k *is the Pascal matrix with*

$$P_k(i,j) = \begin{cases} \binom{i+1}{j-1} & j \in \{0,\dots,i\}, \\ 0 & j > i. \end{cases}$$

Then[40]

$$\bar{\boldsymbol{P}}_{q,n} = \bar{\boldsymbol{C}}_{q,n-1} \cdot H_{k,\omega_n} \cdot P_k, \tag{3.33}$$

where $\omega_n = \frac{h_n}{h_{n-1}}$.

Proof Because of (3.28), it follows for the predictor polynomial $P_{q,n}(t)$ that

$$\begin{aligned} P_{q,n}(t) &= \sum_{m=0}^{k} \bar{\mathbf{C}}_{q,n-1,m} \left(\tfrac{t-t_{n-1}}{h_{n-1}}\right)^m \\ &= \sum_{m=0}^{k} \bar{\mathbf{C}}_{q,n-1,m} \left(\tfrac{t-t_n}{h_n} + 1\right)^m \left(\tfrac{h_n}{h_{n-1}}\right)^m \\ &= \sum_{m=0}^{k} \bar{\mathbf{C}}_{q,n-1,m} \left(\tfrac{h_n}{h_{n-1}}\right)^m \sum_{l=0}^{m} \binom{m}{l} \left(\tfrac{t-t_n}{h_n}\right)^l. \end{aligned}$$

Because

$$\{(m,l) : 0 \le m \le k, 0 \le l \le m\} = \{(m,l) : 0 \le l \le k, l \le m \le k\},$$

it follows for $P_{q,n}(t)$ that

$$\begin{aligned} P_{q,n}(t) &= \sum_{l=0}^{k} \left(\sum_{m=l}^{k} \binom{m}{l} \left(\tfrac{h_n}{h_{n-1}}\right)^m \bar{\mathbf{C}}_{q,n-1,m}\right) \left(\tfrac{t-t_n}{h_n}\right)^l \\ &= \sum_{l=0}^{k} \bar{\mathbf{P}}_{q,n,l} \left(\tfrac{t-t_n}{h_n}\right)^l. \end{aligned}$$

So,

$$\bar{\mathbf{P}}_{q,n,l} = \sum_{m=l}^{k} \binom{m}{l} \left(\frac{h_n}{h_{n-1}}\right)^m \bar{\mathbf{C}}_{q,n-1,m} = \bar{\mathbf{C}}_{q,n-1,l} \binom{l}{l} \left(\frac{h_n}{h_{n-1}}\right)^l + \dots + \bar{\mathbf{C}}_{q,n-1,k} \binom{k}{l} \left(\frac{h_n}{h_{n-1}}\right)^k.$$

□

With this property, it is easy to change stepsizes. Note that $\bar{\mathbf{P}}_{q,n}$ and $\bar{\mathbf{C}}_{q,n}$ always represent k-th degree polynomials.

[40] The matrix $\bar{\mathbf{P}}_{q,n} \in \mathbb{R}^{n\times k+1}$ is expressed as a matrix multiplication of the matrices $\bar{\mathbf{C}}_{q,n-1}$, H_{k,ω_n} and P_k.

Corrector polynomial Finally, to get the Nordsieck representation of $C_{q,n}(t)$, the Nordsieck vector $\bar{\mathbf{L}}_n$ of $L_n(t)$ at t_n is needed. $L_n(t)$ can[41] be written as

$$\begin{aligned} L_n(t) &= \tfrac{t-t_{n-1}}{t_n-t_{n-1}} \cdot \tfrac{t-t_{n-2}}{t_n-t_{n-2}} \cdot \ldots \cdot \tfrac{t-t_{n-k}}{t_n-t_{n-k}} \\ &= \left(\tfrac{1}{\xi_{n,1}}\tfrac{t-t_n}{h_n}+1\right) \cdot \left(\tfrac{1}{\xi_{n,2}}\tfrac{t-t_n}{h_n}+1\right) \cdot \ldots \cdot \left(\tfrac{1}{\xi_{n,k}}\tfrac{t-t_n}{h_n}+1\right) \\ &= 1+\left(\tfrac{1}{\xi_{n,1}}+\ldots+\tfrac{1}{\xi_{n,k}}\right)\left(\tfrac{t-t_n}{h_n}\right)+\ldots+\left(\tfrac{1}{\xi_{n,1}} \cdot \ldots \cdot \tfrac{1}{\xi_{n,k}}\right)\left(\tfrac{t-t_n}{h_n}\right)^k. \end{aligned} \tag{3.34}$$

On this manner, the Nordsieck vector $\bar{\mathbf{L}}_n$ can be determined, such that

$$L_n(t) = \sum_{m=0}^{k} \bar{\mathbf{L}}_{n,m} \left(\frac{t-t_n}{h_n}\right)^m.$$

Because of the relation between the predictor and corrector polynomials[42], it follows that

$$C_{q,n}(t) = \sum_{m=0}^{k} \left(\bar{\mathbf{P}}_{q,n,m} + (\mathbf{q}(t_n, \mathbf{x}_n) - P_{q,n}(t_n))\bar{\mathbf{L}}_{n,m}\right)\left(\frac{t-t_n}{h_n}\right)^m$$

and

$$\bar{\mathbf{C}}_{q,n} = \bar{\mathbf{P}}_{q,n} + (\mathbf{q}(t_n, \mathbf{x}_n) - P_{q,n}(t_n))\bar{\mathbf{L}}_n.$$

Changing the number of stepsizes Besides the stepsizes, also the number of steps k can be used to control the errors. In that case, the degree of the predictor and corrector polynomials must be adapted. Furthermore, if discontinuities occur, one has to use an one step method to restart the integration process.

Thus, if the previous corrector polynomials have degree k_{n-1}, the next predictor polynomials have different degree. If the number of stepsizes remains equal, it still holds that $P_{q,n}(t) = C_{q,n-1}(t)$ and $P_{j,n}(t) = C_{j,n-1}(t)$. This also holds, if $k_n > k_{n-1}$, because there is not sufficient previous data available.

Thus, if $k_n > k_{n-1}$, the predictor polynomal still has degree k_{n-1}. However, the dimension of the Nordsieck vector gets dimension k_n with $\bar{\mathbf{P}}_{n,k_n} = \mathbf{0}$. But, if the number of stepsizes decreases, the next predictor polynomials must be of lower degree. In this case, the predictor and corrector polynomials are not equal, which makes this case more complex. In [40], this subject has been investigated in more detail.

Local discretization error

Asymptotic behaviour On page 24, the local discretization error is defined as the residue of the numerical scheme, after inserting the exact solution. The next theorem can be used to determine the order of an LMM-method.

Theorem 3.13 *The next two propositions are equivalent.*

- *The method is of order p if the local discretization error obeys the next asymptotic behaviour:*

$$\delta_n = C_{p,n} h_n^{p+1} \boldsymbol{q}^{(p+1)}(t_n, x(t_n)) + O(h_n^{p+2}). \tag{3.35}$$

[41] The coefficients $\xi_{n,m}$ have been defined on page 44.

[42] See the equation (3.30).

The error constant $C_{p,n}$ is equal to:

$$C_{p,n} = (-1)^{p+1} \frac{\sum_{m=1}^{p} \xi_{n,m}^{p+1} \rho_{n,m} + (p+1) \sum_{m=0}^{p} \xi_{n,m}^{p} \sigma_{n,m})}{(p+1)!}. \tag{3.36}$$

- *The method coefficients satisfy the next properties*[43]*:*

$$\begin{array}{rrcl} s=0: & \sum_{m=0}^{p} \rho_{n,m} & = & 0, \\ s=1: & \sum_{m=1}^{p} \xi_{n,m} \rho_{n,m} + \sum_{m=0}^{p} \sigma_{n,m}) & = & 0, \\ 2 \le s \le p: & \sum_{m=1}^{p} \xi_{n,m}^{s} \rho_{n,m} + s \sum_{m=1}^{p} \xi_{n,m}^{s-1} \sigma_{n,m}) & = & 0, \\ s=p+1: & \sum_{m=1}^{p} \xi_{n,m}^{s} \rho_{n,m} + s \sum_{m=1}^{p} \xi_{n,m}^{s-1} \sigma_{n,m}) & \neq & 0. \end{array} \tag{3.37}$$

Proof Because of the DAE, it follows for $s \ge 0$ that $\mathbf{j}^{(s)}(t_n, \mathbf{x}(t_n)) = -\mathbf{q}^{(s+1)}(t_n, \mathbf{x}(t_n))$. Now, the asymptotic behaviour of the LDE δ_n is equal to

$$\begin{array}{rcl} \delta_n & = & \sum_{m=0}^{p} \rho_{n,m} \mathbf{q}(t_{n-m}, \mathbf{x}(t_{n-m})) + h_n \sum_{m=0}^{p} \sigma_{n,m} \mathbf{j}(t_{n-m}, \mathbf{x}(t_{n-m})) \\ & = & \sum_{m=0}^{p} \left[\rho_{n,m} (\mathbf{q}(t_n, \mathbf{x}(t_n)) - \xi_{n,m} h_n \mathbf{q}^{(1)}(t_n, \mathbf{x}(t_n)) + \frac{1}{2} \xi_{n,m}^2 h_n^2 \mathbf{q}^{(2)}(t_n, \mathbf{x}(t_n)) + \ldots \right] \\ & + & h_n \sum_{m=0}^{p} \left[\sigma_{n,m} (\mathbf{j}(t_n, \mathbf{x}(t_n)) - \xi_{n,m} h_n \mathbf{j}^{(1)}(t_n, \mathbf{x}(t_n)) + \ldots \right] \\ & = & \sum_{m=0}^{p} \rho_{n,m} \mathbf{q}(t_n, \mathbf{x}(t_n)) - h_n \sum_{m=0}^{p} (\xi_{n,m} \rho_{n,m} + \sigma_{n,m}) \mathbf{q}^{(1)}(t_n, \mathbf{x}(t_n)) \\ & + & \frac{h_n^2}{2} \sum_{m=1}^{p} (\xi_{n,m}^2 \rho_{n,m} + 2 \xi_{n,m} \sigma_{n,m}) \mathbf{q}^{(2)}(t_n, \mathbf{x}(t_n)) + \ldots \\ & + & (-1)^{p+1} \frac{h_n^{p+1}}{(p+1)!} \sum_{m=1}^{p} (\xi_{n,m}^{p+1} \rho_{n,m} + (p+1) \xi_{n,m}^{p} \sigma_{n,m}) \mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n^{p+2}). \end{array}$$

□

If the coefficients satisfy the conditions of the equation (3.37), the method is consistent with order p. Note that in general, the error constant $C_{p,n}$ depends on the previous stepsizes. But for fixed stepsizes, the coefficients and error constant are uniquely defined, independent of the stepsizes.

Estimate of the local discretization error The local discretization error can asymptotically be modelled by the equation (3.35). It appears that δ_n can be approximated by $\hat{\delta}_n$, such that $\delta_n - \hat{\delta}_n = O(h_n^{p+2})$. If the Predictor Corrector method is used, this can easily be done by means of the difference between the predicted and corrected values at t_n. To prove this, a well known extrapolation theorem[44] has been used.

Theorem 3.14 *Let $P(t)$ be an interpolation polynomial of degree l, which interpolates the $l+1$ values $\{x(t_{n-l-1}), \ldots, x(t_{n-1})\}$. Then*

$$x(t_n) - P(t_n) = (t_n - t_{n-l-1}) \cdots (t_n - t_{n-1}) \frac{x^{(l+1)}(\tau)}{(l+1)!}, \quad \tau \in (t_{n-l-1}, t_n).$$

This theorem can be used to estimate the local discretization error.

Theorem 3.15 *Let $P_{q,n}(t)$ be the predictor polynomial and $C_{q,n}(t)$ the corrector polynomial for the values of $\boldsymbol{q}(t, \boldsymbol{x})$. Assume that $C_{q,n}^{(p+1)}(t_n) = \boldsymbol{q}^{(p+1)}(t_n, \boldsymbol{x}(t_n)) + O(h_n)$. Then, the estimate $\hat{\delta}_n$ has sufficient accuracy. Consider the LDE estimate $\hat{\delta}_n$ with*

$$\hat{\delta}_n := C_{p,n} \frac{(p+1)!}{\xi_{n,p+1} \cdots \xi_{n,1}} (C_{q,n}(t_n) - P_{q,n}(t_n)). \tag{3.38}$$

Then, this estimator has sufficient accuracy, which implies that $\hat{\delta}_n = \delta_n + O(h_n^{k+2})$.

[43] Because $\xi_{n,0}^{s} = 0$, if $s > 0$, in that case this term is removed.

[44] This theorem has been derived from [25].

Proof Because $P_{q,n}(t)$ is an interpolation polynomial of degree p for the numerical solution, it follows from Theorem 3.14 that

$$\begin{aligned} P_{q,n}(t_n) - C_{q,n}(t_n) &= \xi_{n,p+1} \cdots \xi_{n,1} h_n^{p+1} \frac{C_{q,n}^{(p+1)}(\tau)}{(p+1)!} \quad \tau \in (t_{n-p-1}, t_n) \\ &= \xi_{n,p+1} \cdots \xi_{n,1} h_n^{p+1} \frac{C_{q,n}^{(p+1)}(t_n)}{(p+1)!} + O(h_n^{p+2}). \end{aligned}$$

Because $C_{q,n}(t_n) = \mathbf{q}(t_n, \mathbf{x}(t_n)) + O(h_n)$, this is equivalent to the next equation:

$$P_{q,n}(t_n) - C_{q,n}(t_n) = \xi_{n,p+1} \cdots \xi_{n,1} h_n^{p+1} \frac{\mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n))}{(p+1)!} + O(h_n^{p+2}).$$

Thus, it follows from the equation (3.35) and the definition of $\hat{\delta}_n$ that $\hat{\delta}_n = \delta_n + O(h_n^{k+2})$.

□

Local and global error

On page 24, the local error is defined. Below, we will relate the local error of LMM-methods to the local discretization error. Furthermore, its asymptotic behaviour is investigated. From Theorem 3.4, it follows that for LMM-methods, the relation between the local error and the LDE is equal to:

$$\mathbf{J}(t_n, \mathbf{x}(t_n))\mathbf{d}_n \doteq \delta_n. \tag{3.39}$$

where $\mathbf{J}(t, \mathbf{x}) = \rho_{n,0} C(t, \mathbf{x}) + h_n \sigma_{n,0} G(t, \mathbf{x})$ is also the Jacobian, which is used to solve the implicit equation (3.26).

Because for the ODE (3.2), we have that $\mathbf{J}(t, \mathbf{x}) = \rho_{n,0} I + h_n \sigma_{n,0} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(t, \mathbf{x})$, it follows from equation (3.39) that the local error is of order p if the LDE is of order p. However, in general this is not true for DAE's, if the Jacobian $C(t, \mathbf{x})$ is not invertible.

Theorem 3.16 *Assume that $\delta_n = O(h_n^{p+1})$ and that the DAE is solvable[45]. If the local indices[46] at all timepoints are smaller or equal than μ, it follows that $\mathbf{d}_n = O(h_n^{p+1-\mu})$. For linear time invariant systems, a sharper result holds that for $\mu > 0$ $\mathbf{d}_n = O(h_n^{p+2-\mu})$.*

Proof Assume that $\delta_n = O(h_n^{p+1})$. Because the DAE is solvable, it follows that for all n the matrix pencils $\lambda_n C(t_n, \mathbf{x}(t_n)) + G(t_n, \mathbf{x}(t_n))$ are not singular for all λ. Assume that the local index[47] of the invertible matrix pencil $\lambda_n C(t_n, \mathbf{x}(t_n)) + G(t_n, \mathbf{x}(t_n))$ is equal to μ_n. Because of the equation (3.39), it follows for the local error that

$$[\lambda_n C(t_n, \mathbf{x}(t_n)) + G(t_n, \mathbf{x}(t_n))]\, \mathbf{d}_n \doteq \frac{1}{h_n \sigma_{n,0}} \delta_n$$

where $\lambda_n = \frac{\rho_{n,0}}{h_n \sigma_{n,0}} = O(h_n)$. From theorem (3.4), it follows that there exist invertible matrices P_n and Q_n, such that

$$P_n\, [\lambda_n C(t_n, \mathbf{x}(t_n)) + G(t_n, \mathbf{x}(t_n))]\, Q_n = \begin{pmatrix} \lambda_n I + A_n & 0 \\ 0 & \lambda_n N_n + I \end{pmatrix}.$$

[45] The solvability of a DAE has been explained on page 23.

[46] The local index has been defined on page 24.

[47] The local index has been defined on page 24.

Because μ_n is the nilpotency index[48] of the matrix N_n, this means[49] that

$$[\lambda_n C(t_n, \mathbf{x}(t_n)) + G(t_n, \mathbf{x}(t_n))]^{-1} = Q_n \begin{pmatrix} (\lambda_n I + A_n)^{-1} & 0 \\ 0 & (\lambda_n N_n + I)^{-1} \end{pmatrix} P_n =$$
$$Q_n \begin{pmatrix} \frac{1}{\lambda_n}(I - \frac{1}{\lambda_n}A_n + \ldots) & 0 \\ 0 & I - \lambda_n N_n + \ldots + (-1)^{\mu_n - 1}\lambda_n^{\mu_n - 1} N_n^{\mu_n - 1} \end{pmatrix} P_n$$

Thus,

$$\mathbf{d}_n = Q_n \begin{pmatrix} \frac{1}{\rho_{n,0}} I & 0 \\ 0 & \frac{1}{h_n \sigma_{n,0}} I + \ldots + (-1)^{\mu_n - 1} \frac{\rho_{n,0}^{\mu_n - 1}}{h_n^{\mu_n} \sigma_{n,0}^{\mu_n}} N_n^{\mu_n - 1} \end{pmatrix} P_n \delta_n + O(h_n^{-\mu_n} \delta_n). \tag{3.40}$$

Assume that $\delta_n = O(h_n^{p+1})$ and that for all n $\mu_n \geq \mu$. Then, indeed it follows that $\mathbf{d}_n = O(h_n^{p+1-\mu})$.

For linear time invariant systems, it holds that $\mathbf{q}(t, \mathbf{x}) = C\mathbf{x}$. This property can be used, because then $\delta_n = C_{p,n} h_n^{p+1} C\mathbf{x}^{(p+1)}(t_n)$[50]. Because of theorem (3.4), the Jacobian matrix C satisfies

$$P_n C Q_n = \begin{pmatrix} I & 0 \\ 0 & N_n \end{pmatrix}.$$

Thus

$$\delta_n = P_n^{-1} \begin{pmatrix} I & 0 \\ 0 & N_n \end{pmatrix} Q_n^{-1} C_{p,n} h_n^{p+1} \mathbf{x}^{(p+1)}(t_n).$$

Inserting this expression in equation (3.40) results in

$$\mathbf{d}_n \doteq Q_n \begin{pmatrix} \frac{1}{\rho_{n,0}} I & 0 \\ 0 & \frac{1}{h_n \sigma_{n,0}} N_n + \ldots + (-1)^{\mu_n - 2} \frac{\rho_{n,0}^{\mu_n - 2}}{h_n^{\mu_n - 1} \sigma_{n,0}^{\mu_n - 1}} N_n^{\mu_n - 1} \end{pmatrix} Q_n^{-1} C_{p,n} h_n^{p+1} \mathbf{x}^{(p+1)}(t_n).$$

Assume again that $\delta_n = O(h_n^{p+1})$ and that for all n $\mu_n \geq \mu$. If $\mu > 0$, indeed it follows that $\mathbf{d}_n = O(h_n^{p+2-\mu})$. But, if $\mu = 0$, the order of $\mathbf{d}_n$ is also equal to p.

□

Thus, the local indices are very important for the order of the local error. Fortunately, many physical system have indices smaller or equal than one. The reduction of the asymptotic order of the local errors is caused by the hidden constraints, which can occur in higher index systems.

Next, we consider the global error, which is defined on page 24 as the difference at t_n between the exact solution and the numerical solution.

Theorem 3.17 *The global error sequence satisfies next recurrent relation:*

$$\sum_{m=0}^{p} \left(\rho_{n,m} C(t_{n-m}, \boldsymbol{x}(t_{n-m})) + h_n \sigma_{n,m} G(t_{n-m}, \boldsymbol{x}(t_{n-m}))\right) \boldsymbol{e}_{n-m} \doteq \delta_n. \tag{3.41}$$

This is equivalent to next relationship between the global and local errors:

$$\boldsymbol{J}(t_n, \boldsymbol{x}(t_n)) \boldsymbol{e}_n \doteq -\sum_{m=1}^{p} \left(\rho_{n,m} C(t_{n-m}, \boldsymbol{x}(t_{n-m})) + h_n \sigma_{n,m} G(t_{n-m}, \boldsymbol{x}(t_{n-m}))\right) \boldsymbol{e}_{n-m} + \boldsymbol{J}(t_n, \boldsymbol{x}(t_n)) \boldsymbol{d}_n. \tag{3.42}$$

[48] $\mu_n = \min\{k \in \mathbb{N} : N_n^k = 0\}$.

[49] If PAQ = B, then it follows that $Q^{-1} A^{-1} P^{-1} = B^{-1} and A^{-1} = Q B^{-1} P$.

[50] See Theorem 3.13.

Proof This is a result from Theorem 3.4.

□

From the equation (3.42), it follows that there is a strong relationship between the global and local error. This means that the order of the global errors is also dependent on the index of the system.

The global error consists of the local error and an additional error, which is caused by the error propagation. The stability of the LMM-method determines, whether the global error remains bounded or not. If the method is strongly stable, the global error is nearly equal to the local error.

3.4.3 Backward Difference Method

Next, we consider the BDF method more closely. In this section, this method will be investigated for variable stepsizes[51].

The BDF-method is a special kind of an LMM method, which solves equation (3.31) by means of evaluating at t_n, which results in:

$$\frac{d}{dt}C_{q,n}(t_n) + \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}. \tag{3.43}$$

This means that the BDF-method is an interpolation method, which approximates the derivative by a backward difference.

Theorem 3.18 *Consider a k-degree polynomial $P(t)$. Then, there exist unique coefficients $\rho_{n,0}, \ldots, \rho_{n,k}$, such that*

$$h_n \frac{dP}{dt}(t_n) = \rho_{n,0}P(t_n) + \ldots + \rho_{n,k}P(t_{n-k}).$$

Because $P_{q,n}(t)$ and $C_{q,n}(t)$ have both degree k, this theorem yields

$$\begin{aligned} h_n \tfrac{d}{dt} P_{q,n}(t_n) &= \rho_{n,0}P_{q,n}(t_n) + \ldots + \rho_{n,k}P_{q,n}(t_{n-k}), \\ h_n \tfrac{d}{dt} C_{q,n}(t_n) &= \rho_{n,0}C_{q,n}(t_n) + \ldots + \rho_{n,k}C_{q,n}(t_{n-k}). \end{aligned}$$

The definition of $C_{q,n}(t)$ implies

$$\begin{aligned} h_n \tfrac{d}{dt} C_{q,n}(t_n) &= \rho_{n,0}\mathbf{q}(t_n, \mathbf{x}_n) + \rho_{n,1}P_{q,n}(t_{n-1}) + \ldots + \rho_{n,k}P_{q,n}(t_{n-k}) \\ &= h_n \tfrac{d}{dt} P_{q,n}(t_n) + \rho_{n,0}\left[\mathbf{q}(t_n, \mathbf{x}_n) - P_{q,n}(t_n)\right]. \end{aligned}$$

Using this identity in the equation (3.43) we obtain the next nonlinear equation in $\mathbf{x}_n$:

$$h_n \frac{d}{dt}P_{q,n}(t_n) + \rho_{n,0}\left[\mathbf{q}(t_n, \mathbf{x}_n) - P_{q,n}(t_n)\right] + h_n\mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}.$$

Define $\alpha_0 = \rho_{n,0}$ and $\beta_0 = h_n \frac{d}{dt}P_{q,n}(t_n) - \rho_{n,0}P_{q,n}(t_n)$. Then $\mathbf{x}_n$ is defined by:

$$\alpha_0\mathbf{q}(t_n, \mathbf{x}_n) + h_n\mathbf{j}(t_n, \mathbf{x}_n) + \beta_0 = \mathbf{0}.$$

This equation can be solved by the Newton Raphson method. The coefficients α_0 and β_0 can easily be computed from the Nordsieck vectors. From $\bar{\mathbf{L}}_n$ the coefficient α_0 can be determined. Because $L_n(t)$ has also degree k,

$$h_n \frac{d}{dt}L_n(t_n) = \rho_{n,0}L_n(t_n) + \ldots + \rho_{n,k}L_n(t_{n-k}) = \rho_{n,0}$$

[51] Much theory is derived from [40].

Clearly, for α_0 and β_0, there holds that

$$\begin{cases} \alpha_0 &= \bar{\mathbf{L}}_{n,1}, \\ \beta_0 &= \bar{\mathbf{P}}_{n,1} - \alpha_0 \bar{\mathbf{P}}_{n,0}. \end{cases}$$

For this method it is sufficient to store only the previous values of $\mathbf{q}(t, \mathbf{x})$, so only $P_{q,n}(t)$ and $C_{q,n}(t)$ are necessary. However, if one would like to use also extrapolation for $\mathbf{x}$ as an initial guess for the Newton-Raphson method, these polynomials are not sufficient, because $\mathbf{q}(t, \mathbf{x})$ is not invertible.

Order of the method

It appears that the order p of the local discretization errors of the BDF-methods with k steps is equal to k. This means that for $k > 0$, all methods are consistent for ODE's. However, for DAE's, it is still possible that the local error does not have the same order as the LDE[52] is not consistent, because of the hidden constraints.

Theorem 3.19 *The BDF-method with k steps has a local discretization error of order $p = k$, such that*

$$\delta_n = C_{p,n} h_n^{p+1} \boldsymbol{q}^{(p+1)}(t_n, \boldsymbol{x}(t_n)) + O(h_n^{p+2}).$$

Proof Consider for $s \in \{0, \ldots, k\}$ the polynomial $Q_s(t)$ of degree lower or equal than k with

$$Q_s(t) = (\frac{t_n - t}{h_n})^s.$$

So

$$Q_s(t_{n-m}) = \xi_m^s, \quad m \in \{0, \ldots, k\}.$$

Then, it follows that for all $s \in \{0, \ldots, k\}$

$$h_n \frac{d}{dt} Q_s(t_n) = \rho_{n,0} Q_s(t_n) + \ldots + \rho_{n,k} Q_s(t_{n-k}).$$

From this property, it can be derived that

$$\begin{array}{rrcl} s = 0: & \rho_{n,0} + \ldots + \rho_{n,k} & = & 0, \\ s = 1: & \rho_{n,1}\xi_{n,1} + \ldots + \rho_{n,k}\xi_{n,k} & = & -1, \\ 1 < s \leq k: & \rho_{n,1}\xi_{n,1}^s + \ldots + \rho_{n,k}\xi_{n,k}^s & = & 0, \\ s = k+1: & \rho_{n,1}\xi_{n,1}^s + \ldots + \rho_{n,k}\xi_{n,k}^s & \neq & 0. \end{array}$$

Furthermore, for BDF-methods:

$$\begin{aligned} \sigma_{n,0} &= 1, \\ \sigma_{n,m} &= 0 \quad m \in \{1, \ldots, k\}. \end{aligned}$$

Because of (3.37), it follows that the local discretization error is $O(h_n^{k+1})$.

□

[52] This has been proved on page 49.

Estimation of the local discretization error

Because of Theorem 3.13, it follows that the local discretization error for a BDF-method with order $p = k$ satisfies the model of equation (3.35). Again, this error can be estimated by means of prediction[53], which results in the estimate $\hat{\delta}_n$

We define the Vandermonde matrix $M_{n,p}$ by

$$M_{n,p} = \begin{pmatrix} 1 & \dots & 1 & 1 \\ 0 & \xi_{n,1} & \dots & \xi_{n,p} \\ 0 & \xi_{n,1}^2 & \dots & \xi_{n,p}^2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \xi_{n,1}^p & \dots & \xi_{n,p}^p \end{pmatrix}.$$

Then, the coefficients $\rho_{n,0}, \dots, \rho_{n,p}$ can be computed as follows:

$$M_{n,p} \cdot \begin{pmatrix} \rho_{n,0} \\ \vdots \\ \rho_{n,p} \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3.44}$$

So, it is possible to compute the error constant $C_{p,n}$ and estimate $\hat{\delta}_n$ by means of solving this linear system. Although the condition number of the matrices $M_{n,p}$ becomes very large, in practice it always holds that $p \leq 6$.

Because of (3.34), it follows that

$$\frac{1}{\xi_{n,p} \cdots \xi_{n,1}} = \bar{\mathbf{L}}_{n,p}.$$

Thus, $\hat{\delta}_n$ can also be computed with use of the Nordsieck vectors:

$$\hat{\delta}_n = C_{p,n} \tfrac{(p+1)!}{\xi_{n,p+1}} \bar{\mathbf{L}}_{n,p} (\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}). \tag{3.45}$$

Although, this estimater can be used, it needs the computation of the error constant $C_{p,n}$. The next theorem gives a cheap method to estimate the local discretization error.

Theorem 3.20 *For the BDF-method of order $p = k$, the error constant $C_{p,n}$ satisfies*

$$C_{p,n} = -\frac{\xi_{n,1} \cdots \xi_{n,p}}{(p+1)!}$$

This means that

$$\begin{aligned} \hat{\delta}_n &= \tfrac{-1}{\xi_{n,p+1}} (\bar{C}_{q,n,0} - \bar{P}_{q,n,0}), \\ \delta_n &= -\tfrac{1}{(p+1)!} \xi_{n,1} \cdots \xi_{n,p} h_n^{p+1} \boldsymbol{q}^{(p+1)}(t_n, x(t_n)) + O(h_n^{p+2}). \end{aligned}$$

[53] See Theorem 3.15.

Proof Consider the Lagrange polynomial $L_n^*(t)$ of degree $k+1$ with

$$\begin{aligned} \bar{L}_n(t_{n-m}) &= 0, \quad m \in \{1, \ldots, p+1\}, \\ \bar{L}_n(t_n) &= 1. \end{aligned}$$

So,

$$L_n^*(t) = \frac{t - t_{n-p-1}}{t_n - t_{n-p-1}} L_n(t) = (\frac{1}{\xi_{n,p+1}}(\frac{t - t_n}{h_n}) + 1) L_n(t)$$

and

$$L_n(t) - L_n^*(t) = \frac{-1}{\xi_{n,p+1}}(\frac{t - t_n}{h_n}) L_n(t).$$

Consider for $s \geq 1$ the polynomials $Q_s(t) = (\frac{t_n - t}{h_n})^s$ with the property that $Q_s(t_{n-m}) = \xi_m^s$. Thus, it also holds that

$$\rho_{n,0} Q_{p+1}(t_n) + \ldots + \rho_{n,p} Q_{p+1}(t_{n-p}) = \rho_{n,1} \xi_{n,1}^{p+1} + \ldots + \rho_{n,p} \xi_{n,p}^{p+1}.$$

Furthermore

$$L_n^*(t) = \sum_{m=0}^{p+1} \bar{\mathbf{L}}_{n,m}^* (-1)^m Q_m(t).$$

We recall the proof of Theorem 3.19: it follows that

$$\begin{aligned} \rho_{n,0} L_n^*(t_n) + \ldots + \rho_{n,p} L_n^*(t_{n-p}) &= \\ \textstyle\sum_{m=0}^{p+1} \bar{\mathbf{L}}_{n,m}^* (-1)^m \left(\rho_{n,0} Q_m(t_n) + \ldots + \rho_{n,p} Q_m(t_{n-p}) \right) &= \\ \bar{\mathbf{L}}_{n,1}^* + (-1)^{p+1} \bar{\mathbf{L}}_{n,p+1}^* \left(\rho_{n,1} \xi_{n,1}^{p+1} + \ldots + \rho_{n,p} \xi_{n,p}^{p+1} \right). \end{aligned}$$

Because of the definition of $L_n^*(t)$, it follows that

$$\rho_{n,0} L_n^*(t_n) + \ldots + \rho_{n,p} L_n^*(t_{n-p}) = \rho_{n,0} = \bar{\mathbf{L}}_{n,1}.$$

Thus,

$$\begin{aligned} (-1)^{p+1} \bar{\mathbf{L}}_{n,p+1}^* \left(\rho_{n,1} \xi_{n,1}^{p+1} + \ldots + \rho_{n,p} \xi_{n,p}^{p+1} \right) &= \\ \bar{\mathbf{L}}_{n,1} - \bar{\mathbf{L}}_{n,1}^* = \tfrac{-1}{\xi_{n,p+1}} \bar{\mathbf{L}}_{n,0} = \tfrac{-1}{\xi_{n,p+1}}. \end{aligned}$$

So, for the error constant $C_{p,n}$ in the equation (3.36), it follows that

$$C_{p,n} = (-1)^{p+1} \frac{\sum_{m=1}^{p} \xi_{n,m}^{p+1} \rho_{n,m}}{(p+1)!} = \frac{-1}{\xi_{n,p+1}} \frac{1}{(p+1)! \bar{\mathbf{L}}_{n,p+1}^*}.$$

Because $\bar{\mathbf{L}}_{n,p+1}^* = \frac{1}{\xi_{n,p+1}} \bar{\mathbf{L}}_{n,p}$, it follows that

$$C_{p,n} = \frac{-1}{(p+1)! \bar{\mathbf{L}}_{n,p}},$$

which together with the equation (3.34) proves the theorem. Because of the equation (3.45), it immediately follows that $\hat{\delta}_n = \frac{-1}{\xi_{n,p+1}} (\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0})$.

□

3.4.4 Numerical Difference Methods

Description of the method

It has become clear that implicit BDF-methods are important, because of their stability regions. However, for higher order, the stability regions decrease. Klopfenstein[54] studied a related family of formulas: the Numerical Difference Formulas, where one takes care with the prediction of the LDE. It is possible to adapt the BDF-method, which results in the next NDF-scheme[55]:

$$\alpha_0 \mathbf{q}(t_n, \mathbf{x}_n) + h_n \mathbf{j}(t_n, \mathbf{x}_n) + \beta_0 = \kappa_n(\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}). \tag{3.46}$$

For $\kappa_n = 0$, this NDF-scheme is equal to the original BDF method. Each iteration, the next nonlinear equation has to be solved:

$$(\alpha_0 - \kappa_n)\mathbf{q}(t_n, \mathbf{x}_n) + h_n \mathbf{j}(t_n, \mathbf{x}_n) + \beta_0 + \kappa_n \bar{\mathbf{P}}_{q,n,0} = \mathbf{0}.$$

It is still possible to estimate the LDE, because for a k-step method with order $p = k$, it holds that

$$\delta_n^{NDF} = \delta_n^{BDF} - \kappa_n(\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}) = -(\frac{1}{\xi_{n,p+1}} + \kappa_n)(\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}) + O(h_n^{p+2}).$$

With use of the free parameter κ_n, one can get better schemes with lower discretization errors. For this scheme, it follows that

$$\delta_n^{NDF} = \delta_n^{BDF} - \kappa_n(\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}) = (1 + \xi_{n,p+1}\kappa_n)\delta_n^{BDF} + O(h_n^{p+2}).$$

Take $\kappa_n = \frac{p+1}{\xi_{n,p+1}}\kappa$, with κ a constant number. This means that

$$\delta_n^{NDF} = (1 + \kappa(p+1))\delta_n^{BDF} + O(h_n^{p+2}).$$

If $\kappa = \frac{\lambda - 1}{p+1}$, the discretization error is simply multiplied by λ:

$$\|\delta_n^{NDF}\| = |\lambda| \|\delta_n^{BDF}\|. \tag{3.47}$$

Theorem 3.21 *If the order is equal to $p = k$, this means that*

$$\frac{|h_n^{NDF}|}{|h_n^{BDF}|} \leq |\lambda|^{\frac{-1}{p+1}} \Rightarrow \|\delta_n^{NDF}\| = \|\delta_n^{BDF}\|.$$

Proof From Theorem 3.20, it follows that

$$\delta_n^{BDF} = -\frac{1}{(p+1)!}\xi_{n,1} \cdots \xi_{n,p}(h_n^{BDF})^{p+1}\mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n^{p+2}).$$

Because $\xi_{n,m} = \frac{t_n - t_{n-m}}{h_n}$ remains equal if all stepsizes are multiplied with a constant factor, it follows that

$$\frac{\|\delta_n^{NDF}\|}{\|\delta_n^{BDF}\|} \doteq |\lambda| \left(\frac{|h_n^{NDF}|}{|h_n^{NDF}|}\right)^{p+1}.$$

Assume that $\frac{|h_n^{NDF}|}{|h_n^{BDF}|} \leq |\lambda|^{\frac{-1}{p+1}}$, then

$$\frac{\|\delta_n^{NDF}\|}{\|\delta_n^{BDF}\|} \leq 1.$$

[54] See [32].

[55] In appendix C, it has been shown how this scheme can be implemented in Pstar.

Table 3.4: Prediction polynomials $\mu(z)$ which can be used for method with order p.

p :	$\mu(z)$
1	$2z-1$
2	$3z^2-3z+1$
3	$4z^3-6z^2+4z-1$
4	$5z^4-10z^3+10z^2-5z+1$
5	$6z^5-15z^4+20z^3-15z^2+6z-1$
6	$7z^6-21z^5+35z^4-35z^3+21z^2-7z+1$

□

The factor $|\lambda|^{\frac{-1}{p+1}}$ is called the Step Ratio Percent (SRP). If λ is small, the NDF-method performs better, while the order is increased if $\lambda = 0$.

Besides the local accuracy, also the stability is important. Therefore, the stability of the NDF-methods will be investigated for the test equation.

Now, the choice of κ is a trade-off between the stability and the error. Consider the test equation $\dot{\mathbf{y}} = \lambda \mathbf{y}$ and perform the NDF-method with constant stepsize $h_n = h$. Note that for fixed stepsizes, $\kappa_n = \kappa$. Denoting the predictor for y_n by y_n^0, method (3.46) is described by next equation.

$$\sum_{m=0}^{k} \rho_m y_{n-k+m} - h\sigma_m \lambda y_n - \kappa(y_n - y_n^0) = \mathbf{0}.$$

The predictor y_n^0 is the result of the extrapolation of the previous numerical solutions. This prediction can be described with use of the polynomial $\mu(z)$ and the shift-operator q:

$$y_n^0 = \mu(q) y_{n-p-1}.$$

In Tab.(3.4), these polynomials $\mu(z)$ have been shown for orders $p \in \{1, \ldots, 6\}$.

Introduce the modified polynomials $\bar{\rho}(z)$ and $\bar{\sigma}(z)$ with

$$\begin{aligned} \bar{\rho}(z) &= z\rho(z) - \kappa(z^{p+1} - \mu(z)), \\ \bar{\sigma}(z) &= z\sigma(z). \end{aligned}$$

After introducing $\bar{h} = \lambda h$, the NDF-method performs the next method:

$$\bar{\rho}(q) y_{n-p-1} - \bar{h}\sigma(q) y_{n-p-1} = \mathbf{0}.$$

So, the NDF-version is a $(p+1)$-step method. To restrict the number of steps to k, it is necessary to use a lower order extrapolation polynomial $\mu(z)$.

Define the polynomial $\bar{\pi}(z, \bar{h}) = \bar{\rho}(z) - \bar{h}\sigma(z))$. The stability-region S of this method is defined as

$$S = \{\bar{h} \in \mathbb{C} : \bar{\pi}(z, \bar{h}) \text{is root stable.}\}$$

The BDF-methods are unconditionaly stable for $p \in \{1, 2\}$. We want to maintain this property for the NDF-methods. Because of the equation (3.47), it follows that for $\lambda = 0$ and $\kappa = \frac{-1}{p+1}$, the order is increased, while for $\lambda = \frac{1}{2}$ and $\kappa = \frac{-1}{2p+2}$ the error terms are halved.

Euler Backward

Consider the fixed step version of the Euler Backward method with $p = k = 1$. Then, the method is described by the next polynomials:

$$\begin{aligned}
&\rho(z) = z - 1,\\
&\sigma(z) = z,\\
&\mu(z) = 2z - 1,\\
&\bar{\rho}(z) = z(z-1) - \kappa(z^2 - \mu(z)),\\
&\bar{\sigma}(z) = z^2.
\end{aligned}$$

So for $\theta \in [0, \pi]$, it can be computed that

$$\mathrm{Re}[\frac{\bar{\rho}(e^{i\phi})}{\bar{\sigma}(e^{i\phi})}] = 2(1 + 2\kappa\cos(\phi))\sin(\frac{\phi}{2})^2.$$

Thus, if $\kappa \in [-\frac{1}{2}, \frac{1}{2}]$, the method is unconditionally stable. This implies that for $\kappa = \frac{-1}{2}$ the error constant will be zero, while the unconditionally stability is maintained. Then, $\bar{\rho}(z)$ and $\bar{\sigma}(z)$ are equal to

$$\bar{\rho}(z) = \frac{3}{2}z^2 - 2z + \frac{1}{2}, \quad \bar{\sigma}(z) = z^2.$$

This is just the BDF2-method and thus it is not necessary to perform this version of the Euler Backward method. In practice, however, this BDF2-version can be easier to implement, because no order control is needed to use the BDF2-method.

In the case $\kappa = \frac{-1}{6}$, the error constant is halved.

BDF2 method

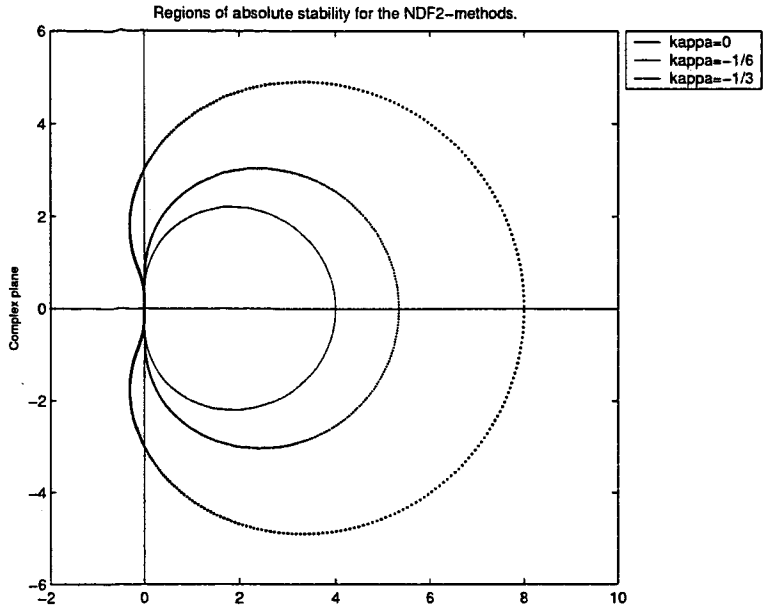
Now consider the fixed step version of the BDF2 method with $p = k = 2$. Then, the method is described by the next polynomials:

$$\begin{aligned}
&\rho(z) = \tfrac{1}{2}(3z^2 - 4z + 1),\\
&\sigma(z) = z^2,\\
&\mu(z) = 3z^2 - 3z + 1,\\
&\bar{\rho}(z) = \tfrac{1}{2}z(3z^2 - 4z + 1) - \kappa(z^3 - \mu(z)),\\
&\bar{\sigma}(z) = z^3.
\end{aligned}$$

For $\theta \in [0, \pi]$, it can be computed that

$$\mathrm{Re}[\frac{\bar{\rho}(e^{i\phi})}{\bar{\sigma}(e^{i\phi})}] = (\cos(\phi) - 1)^2(1 + 2\kappa + 4\kappa\cos(\phi)).$$

If $\kappa \in [-\frac{1}{6}, \frac{1}{2}]$, the adapted version will be unconditionally stable. Now, the optimal choice is $\kappa = \frac{-1}{6}$, such that the error constant will be halved. In this case, the error term will be halved, while the SRP is equal to 1.26. Thus a 3 step method of order 2 is founded, with a smaller error term, while it is still unconditionally stable. In Fig.(3.4.4), the boundaries of the stability region for $\kappa \in \{-\frac{1}{3}, -\frac{1}{6}, 0\}$ has been shown.



General results

Because for higher order, the BDF-methods themselves are not unconditionally stable, it is not necessary to have this property for the NDF-methods. Klopfenstein and Shampine[56] found the next optimal choices for κ:

order p	κ	$\frac{h^*}{h}$	stability angle BDF	stability angle NDF
1	-0.1850	26%	90	90
2	$-\frac{1}{6}$	26%	90	90
3	-0.1509	26%	86	80
4	-0.0865	12%	73	66
5	0	0%	51	51

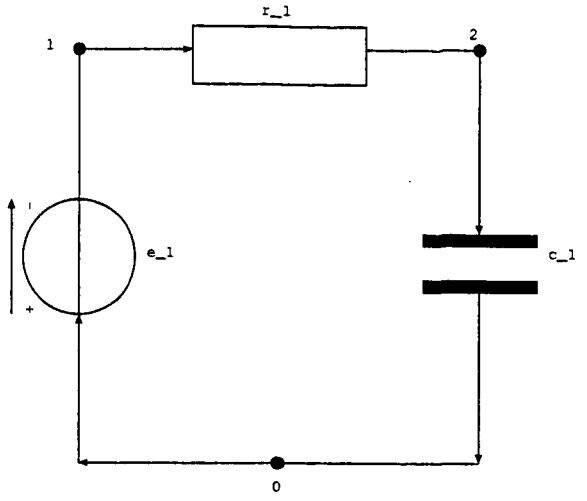
Note that for $p \in \{1, 2, 3\}$, the expected amount of work will decrease with 20%.

It is also possible to use this method for the other LMM methods. However, for the Trapezoidal method, it appears that it is not possible to conserve the unconditional stability, if the error constant is reduced.

Small example

Now, the NDF methods are performed on a small real circuit. Therefore, the NDF-implementation "ode15s" in MATLAB has been used. Consider the next electrical circuit:

[56] See [32].



In Pstar, the circuit model is:

```
title: NDFexample;

circuit;
 f = 50;
 amp=1;
 j_1 (0,1) amp*sin(2*pi*f*t);
 r_1 (1,2) 100ml;
 c_1     (2,0) 100ml;
end;
```

With the program Qstar[57], the circuit equations are derived in MATLAB. It appears that

$$\mathbf{j}(t,\mathbf{x}) = \begin{bmatrix} -\sin(100\pi t) + \frac{x_1 - x_2}{0.1} \\ -\frac{x_1 - x_2}{0.1} \end{bmatrix}, \tag{3.48}$$

$$\mathbf{q}(t,\mathbf{x}) = \begin{bmatrix} 0.0 \\ x_2 \cdot 0.1 \end{bmatrix}. \tag{3.49}$$

Now, this circuit is simulated with initial conditions $x(1) = x(2) = 1$. The routine ode15s is used with fixed stepsizes $h = 10^{-5}$ and $h^* = 1.2510^{-5}$. Because this IVP has an analytic solution, it is possible to show the global errors for $x(1)$ and $x(2)$. These global errors have been shown in Fig.(3.7).

[57] This is an useful tool, which can transform Pstar input files into m-files for MATLAB. For more information, the reader is referred to [21, 22].

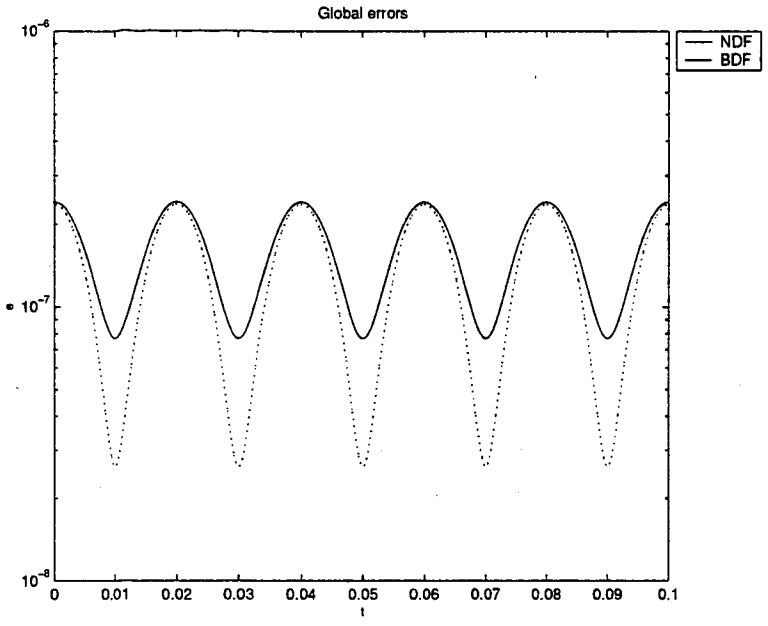


Figure 3.7: Global errors of the NDF-(dashed line) and the BDF-method (solid line).

Indeed, the global errors are about equal, while the NDF-method needs less computational work.

	NDF	BDF
#Stepsizes	8000	10000
#Function evaluations	16003	20003
Time (seconds)	49	63

From the table, it follows that the NDF methods need only 78% of the time, which the BDF methods use.

3.4.5 Multirate approach

Concept of the multirate approach

In chapter two, it has been shown that all electrical circuits can be described by a IVP for the DAE (3.1). All equations represent electrical components, while the variables represent the voltages and currents in the circuit. In all integration methods in the former sections, all equations are solved with the same method and the same stepsize. Furthermore, all variables are discretized at the same time grid. However, the functions $\mathbf{q}$ and $\mathbf{j}$ have an hierarchical structure, because of the topology of the circuits. Some subcircuits may show fast varying behaviour, while other subcircuits have much slower behaviour or show periods with latency, for example digital subcircuits. Also, there are differences with respect to the wanted accuracies. Therefore, it could be attractive to use more time rates, which is called the multirate approach. Below, the multirate approach has been explained for the Euler Backward method. However, it is also possible to perform this approach with other integration methods[58]

The slow part of the circuit is integrated with larger stepsizes $H = mh$.

[58]In the MSc-Thesis [3], the multirate approach has been studied for the ROW methods, which is a special family of Runge Kutta methods.

The variable $\mathbf{x}$ and functions $\mathbf{q}$ and $\mathbf{j}$ are separated in a fast (f) and a slow (s) part.

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_f \\ \mathbf{x}_s \end{pmatrix} \quad \mathbf{q}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{q}_f(t, \mathbf{x}) \\ \mathbf{q}_s(t, \mathbf{x}) \end{pmatrix}, \quad \mathbf{j}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{j}_f(t, \mathbf{x}) \\ \mathbf{j}_s(t, \mathbf{x}) \end{pmatrix}.$$

Then, the DAE is equivalent to

$$\begin{cases} \frac{d}{dt}\mathbf{q}_f(t, \mathbf{x}_f, \mathbf{x}_s) + \mathbf{j}_f(t, \mathbf{x}_f, \mathbf{x}_s) = \mathbf{0}, \\ \frac{d}{dt}\mathbf{q}_s(t, \mathbf{x}_f, \mathbf{x}_s) + \mathbf{j}_s(t, \mathbf{x}_f, \mathbf{x}_s) = \mathbf{0}. \end{cases}$$

Slowest first

The "slowest first" version of multirate integrates first the slowest part, while $\mathbf{x}_f$ is replaced by an extrapolation function, resulting in $\hat{\mathbf{x}}_f(t)$. Afterwards, the fastest part is integrated, while $\mathbf{x}_s$ is replaced by an interpolation function $\hat{\mathbf{x}}_s(t)$. Using Euler Backward as integration method on the multirate timegrid $\{t_k^n = (k-1)H + nh : k = 0, \ldots, N, n = 0, \ldots, m-1\}$ results in:

$$\begin{cases} \mathbf{q}_s(t_k^0, \hat{\mathbf{x}}_f(t_k^0), \mathbf{x}_s(t_k^0)) - \mathbf{q}_s(t_{k-1}^0, \hat{\mathbf{x}}_f(t_{k-1}^0), \mathbf{x}_s(t_{k-1}^0)) + H\mathbf{j}_s(t_k^0, \hat{\mathbf{x}}_f(t_k^0), \mathbf{x}_s(t_k^0)) = \mathbf{0}, \\ \mathbf{q}_f(t_{k-1}^n, \mathbf{x}_f(t_{k-1}^n), \hat{\mathbf{x}}_s(t_{k-1}^n)) - \mathbf{q}_f(t_{k-1}^{n-1}, \mathbf{x}_f(t_{k-1}^{n-1}), \hat{\mathbf{x}}_s(t_{k-1}^{n-1})) + h\mathbf{j}_f(t_{k-1}^n, \mathbf{x}_f(t_{k-1}^n), \hat{\mathbf{x}}_s(t_{k-1}^n)) = \mathbf{0}. \end{cases}$$

Because $\mathbf{x}_f(t_{k-1}^0)$ is already known,

$$\hat{\mathbf{x}}_f(t_{k-1}^0) = \mathbf{x}_f(t_{k-1}^0).$$

However, $\mathbf{x}_f(t_k^0)$ and $\mathbf{x}_s(t_{k-1}^n)$ are not known. Applying constant extrapolation on $\mathbf{x}_f$ and linear interpolation on $\mathbf{x}_s$ gives:

$$\hat{\mathbf{x}}_f(t_k^0) = \mathbf{x}_f(t_{k-1}^0), \quad \hat{\mathbf{x}}_s(t_{k-1}^n) = \mathbf{x}_s(t_{k-1}^0) + \tfrac{t_{k-1}^n - t_{k-1}^0}{H}(\mathbf{x}_s(t_k^0) - \mathbf{x}_s(t_{k-1}^0)).$$

Fastest first

The "fastest first" version of multirate integrates first the fastest part, while $\mathbf{x}_s$ is replaced by an extrapolation function. When the slowest part is integrated, the variable $\mathbf{x}_f$ at t_k^0 is already known, without interpolation.

$$\begin{cases} \mathbf{q}_f(t_{k-1}^n, \mathbf{x}_f(t_{k-1}^n), \hat{\mathbf{x}}_s(t_{k-1}^n)) - \mathbf{q}_f(t_{k-1}^{n-1}, \mathbf{x}_f(t_{k-1}^{n-1}), \hat{\mathbf{x}}_s(t_{k-1}^{n-1})) + h\mathbf{j}_f(t_{k-1}^n, \mathbf{x}_f(t_{k-1}^n), \hat{\mathbf{x}}_s(t_{k-1}^n)) = \mathbf{0}, \\ \mathbf{q}_s(t_k^0, \mathbf{x}_f(t_k^0), \mathbf{x}_s(t_k^0)) - \mathbf{q}_s(t_{k-1}^0, \mathbf{x}_f(t_{k-1}^0), \mathbf{x}_s(t_{k-1}^0)) + H\mathbf{j}_s(t_k^0, \mathbf{x}_f(t_k^0), \mathbf{x}_s(t_k^0)) = \mathbf{0}. \end{cases}$$

To predict $\hat{\mathbf{x}}_s(t_{k-1}^n)$, one can use constant extrapolation:

$$\hat{\mathbf{x}}_s(t_{k-1}^n) = \mathbf{x}_s(t_{k-1}^0).$$

Note that it is not necessary to use interpolation for the fast variables, because they are already known.

However, if stepsize control is used for the large stepsize H and H is rejected, one needs the variable $\mathbf{x}_f$ many small stepsizes ago. Thus, the fast variables have to be backed up, which may not be wanted.

Let p be the order of the original integration method, which is equal to one for the Euler Backward method. Assume that this method is stable and consistent. Furthermore, define q as the order of the interpolation method with $q \geq 1$. In an article of C.W.Gear, it has been proved[59] that the multirate method has a global error, which is $O(h^{\min(p,q)})$, if the the coupling from the fast values to the slow values is small. This implies that it is necessary to use at least constant interpolation ($q = 1$) to remain consistent.

[59] See p490 in [12].

3.5 Newton-Raphson method

3.5.1 Description of the method

Consider the nonlinear equation:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{3.50}$$

Assume that $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$. Define the Jacobian matrix $\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$. Let $\mathbf{x}^k$ be the previous approximation of $\mathbf{x}^*$, then it is possible to use the Taylor approximation of $\mathbf{f}(\mathbf{x}^*)$ around $\mathbf{x}^k$ to get a better approximation:

$$\mathbf{0} = f(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}^k) + \mathbf{J}(\mathbf{x}^k)(\mathbf{x}^* - \mathbf{x}^k) + O(\|\mathbf{x}^* - \mathbf{x}^k\|).$$

Neglecting the higher order term, this results in the Newton-Raphson method, which performs next algorithm to solve equation (3.50):

1. Set the inital value $\mathbf{x}^0$ and the tolerances $\epsilon_1, \epsilon_2, \epsilon_3$.

2. Solve the linear equation $\mathbf{J}(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = -\mathbf{f}(\mathbf{x}^k)$.

3. If $\|\mathbf{f}(\mathbf{x}^k)\| < \epsilon_1$ and $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| < \epsilon_2 + \epsilon_3\|\mathbf{x}^k\|$ stop. Else go to step 2.

It can be proved[60] that for sufficient smooth $\mathbf{f}$ and properly chosen $\mathbf{x}^0$, the algorithm is convergent. Furthermore, if $\mathbf{J}(\mathbf{x}^*)$ is non-singular, the method converges quadratically to the solution.

Define the Hessian matrix

$$\mathbf{H}(\mathbf{x}) = \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x} \partial \mathbf{x}^T} = \begin{pmatrix} \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}_1^2} & \cdots & \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}_1 \partial \mathbf{x}_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}_d \partial \mathbf{x}_1} & \cdots & \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}_d^2} \end{pmatrix},$$

then it follows that

$$\mathbf{f}(\mathbf{x}^k) = \mathbf{J}(\mathbf{x}^*)(\mathbf{x}^k - \mathbf{x}^*) + \frac{1}{2}((\mathbf{x}^k - \mathbf{x}^*) \otimes \mathbf{I}_d)^T \mathbf{H}(\mathbf{x}^*)(\mathbf{x}^k - \mathbf{x}^*) + O(\|\mathbf{x}^k - \mathbf{x}^*\|^3).$$

Define the error $\mathbf{e}^k = \mathbf{x}^* - \mathbf{x}^k$. Then, it follows that

$$\mathbf{J}(\mathbf{x}^*)(\mathbf{e}^{k+1} - \mathbf{e}^k) = \mathbf{J}(\mathbf{x}^*)(\mathbf{x}^k - \mathbf{x}^{k+1}) = \mathbf{f}(\mathbf{x}^k) = -\mathbf{J}(\mathbf{x}^*)\mathbf{e}^k + \frac{1}{2}(\mathbf{e}^k \otimes I_d)^T \mathbf{H}(\mathbf{x}^*)\mathbf{e}^k + O(\|\mathbf{e}^k\|^3).$$

Thus,

$$\mathbf{J}(\mathbf{x}^*)\mathbf{e}^{k+1} = \frac{1}{2}(\mathbf{e}^k \otimes I_d)^T \mathbf{H}(\mathbf{x}^*)\mathbf{e}^k + O(\|\mathbf{e}^k\|^3).$$

Indeed, if $\mathbf{J}(\mathbf{x}^*)$ is non-singular and $\mathbf{H}(\mathbf{x}^*)$ exists, the convergence is quadratically.

[60] See [29].

3.5.2 Application to numerical schemes for DAE's

In the previous sections, it has become clear that for many numerical schemes, each time, the next nonlinear equation has to be solved:

$$\alpha \mathbf{q}(t_n, \mathbf{x}_n) + h_n \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{r},$$

with $\mathbf{r}$ an already known vector. Often, this nonlinear equation is solved by the Newton-Raphson method. There are also other possibilities, such as the Modified Newton-Raphson method or fixed point iteration. These methods are only linear convergent, but need less LU-factorisations.

As initial guess for these iterative methods, the previous solution or an extrapolated prediction is used. This means that the number of iterations is also dependent of the stepsize sequence, because for small stepsizes, the initial guess becomes better. Furthermore, the convergence ratio for the linear convergent methods becomes also better for small stepsizes[61].

[61] For more information about this subject, see [38].

Chapter 4

Adaptive stepsize control

4.1 Introduction

In the previous chapter, the transient analysis of DAE's has been summarized. To control the errors of these integration methods, it is necessary to use adaptive stepsize control. This subject will be studied in more detail in this chapter. It seems that no much mathematical attention has been payed to stepsize control in the past, in contrast to the integration methods. That is the reason why Gustaf Söderlind[1] proposed a control-theoretic approach of timestep control, which is called adaptive time-stepping. He claims that integration methods with use of adaptive time-stepping yield smoother stepsize sequences, fewer rejected steps, more efficiency, while the total work will not grow significantly. It is also possible to design the stepsize controller for special purposes as higher order of adaptivity (for smooth ODE problems) or filtering the high-frequency error components (for non-smooth problems). This controlled timestep variations are also claimed to be less sensitive with respect to parameter variations than in classical time integration procedures. Hence, results obtained with automatic control will be more robust and better suited for optimization purposes than before. This chapter will investigate the possibilities to use control theory with adaptive timestepping. DAE solvers use stepsize control to control the local discretization errors. This subject will be explained in section two. Here also the classical approach will be described. Afterwards, the specifications of a good stepsize controller will be defined. Also the available literature about this subject has been summarized. In section three, the theory of digital linear control systems is studied. This theory can be used to design better stepsize controllers. How this can be done is described in section four.

4.2 Description of adaptive stepsize control

4.2.1 One step methods

As shown before, the LDE can be estimated with use of a reference method of higher order. In each timestep, the next solution is computed with these two methods: $\bar{\mathbf{x}}_n$ and $\mathbf{x}_n$. Then

$$\hat{\delta}_n = \bar{\mathbf{x}}_n - \mathbf{x}_n.$$

Because $\hat{\delta}_n = \delta_n + O(h_n^{p+2})$, it follows that $\hat{\delta}_n$ obeys the next asymptotic behaviour:

$$\hat{\delta}_n = \hat{\Phi}(t_n, \mathbf{x}(t_n))h_n^{p+1}. \tag{4.1}$$

[1]Professor at the Department Numerical Analysis of Lund University

Now, it is possible to perform adaptive step size control, because the LDE can be controlled by the stepsize. It is possible to control the LDE per step (EPS) or per unit step (EPUS). Because the control-theoretic approach to stepsize control is the same for EPS- and EPUS-control, the controlled error estimate $\hat{r}_n$ is introduced. Then, $\hat{r}_n = \|\hat{\delta}_n\| h_n^{P-p-1}$ with $P = p$ (EPS) or $P = p+1$ (EPUS). Let TOL be a certain tolerance. Then with both types of control the next constraints have to be satisfied:

$$\forall_n \hat{r}_n \leq \text{TOL}. \tag{4.2}$$

Introduce for all n

$$\hat{\varphi}_n = \|\hat{\Phi}(t_n, \mathbf{x}_n)\|.$$

Then

$$\|\hat{\delta}_n\| = \hat{\varphi}_n h_n^{p+1}$$

and

$$\hat{r}_n = \hat{\varphi}_n h_n^P. \tag{4.3}$$

Because $\hat{r}_n$ has to satisfy the constraints of the equation (4.2), the stepsize h_n has to satisfy the next inequality:

$$h_n \leq \left(\frac{\text{TOL}}{\hat{\varphi}_n}\right)^{\frac{1}{P}}. \tag{4.4}$$

However, $\hat{\varphi}_n$ is not known, when h_n is computed. This means that $\hat{\varphi}_n$ must be predicted, which can be done by feedback control laws. A safety factor $0 < \theta < 1$ is used, to compensate the prediction error.

If $\hat{\varphi}_n$ is predicted by $\hat{\varphi}_{n-1}$, it follows from the equation (4.3) that the control law (4.4) is equivalent to

$$h_n \leq \left(\frac{\text{TOL}}{\hat{r}_{n-1}}\right)^{\frac{1}{P}} h_{n-1}. \tag{4.5}$$

This upper bound can be used to derive the next elementary control law, with $\epsilon = \theta\text{TOL}$ and $0 < \theta < 1$ a safety factor.

Theorem 4.1 *Consider the next elementary local error control law, with $\epsilon = \theta TOL$ and $0 < \theta < 1$ a safety factor.*

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\frac{1}{P}} h_{n-1}. \tag{4.6}$$

If $\theta\hat{\varphi}_n \leq \hat{\varphi}_{n-1}$, then the constraints (4.2) are satisfied with this control law.

Proof For all n we have that

$$\hat{r}_n = \hat{\varphi}_n h_n^P = \hat{\varphi}_n \left(\frac{\epsilon}{\hat{r}_{n-1}}\right) h_{n-1}^P.$$

Because $\hat{r}_{n-1} = \hat{\varphi}_{n-1} h_{n-1}^P$ and $\epsilon = \theta\text{TOL}$, it follows that

$$\hat{r}_n = \frac{\hat{\varphi}_n}{\hat{\varphi}_{n-1}} \epsilon \leq \text{TOL}.$$

□

From this theorem, it becomes clear that if $\theta\hat{\varphi}_n \leq \hat{\varphi}_{n-1}$, this control law will always satisfy the constraints (4.2) for $\hat{r}$.

4.2.2 BDF methods

General model

For the BDF-methods, it appears that the LDE obeys a more complex model. Below, this model is derived, with also a first order approximation. Note that for BDF-methods, the order p is equal to the number of steps k. Then recalling Theorem 3.20, the LDE can be estimated with use of prediction, i.e.

$$\hat{\delta}_n = \frac{-1}{\xi_{n,p+1}}(\bar{\mathbf{C}}_{q,n,0} - \bar{\mathbf{P}}_{q,n,0}).$$

Because of Theorem 3.20 and the sufficient accuracy of the estimator, the LDE estimate of a BDF-method with order p, satisfies[2]:

$$\begin{aligned} \hat{\delta}_n &= -\tfrac{1}{(p+1)!}\xi_{n,1}\cdots\xi_{n,p}h_n^{p+1}\mathbf{q}^{(p+1)}(t_n,\mathbf{x}(t_n)) + O(h_n^{p+2}) \\ &= -\tfrac{1}{(p+1)!}h_n^2(h_{n-1}+h_n)\cdots(h_{n-p+1}+\cdots+h_n)\mathbf{q}^{(p+1)}(t_n,\mathbf{x}(t_n)) + O(h_n^{p+2}). \end{aligned} \tag{4.7}$$

Define $\hat{\varphi}$, such that

$$\hat{\delta}_n = \frac{1}{p!}h_n^2(h_{n-1}+h_n)\cdots(h_{n-p+1}+\cdots+h_n)\hat{\Phi}(t_n,\mathbf{x}(t_n)).$$

This means that

$$\hat{\Phi}(t_n,\mathbf{x}(t_n)) = -\frac{1}{p+1}\mathbf{q}^{(p+1)}(t_n,\mathbf{x}(t_n)) + O(h_n).$$

Introduce for all n

$$\hat{\varphi}_n = \|\hat{\Phi}(t_n,\mathbf{x}(t_n))\|.$$

Then, because all stepsizes are positive, it follows that

$$\hat{r}_n = \|\hat{\delta}_n\|h_n^{P-p-1} = \hat{\varphi}_n\frac{1}{p!}h_n^{1+P-p}(h_{n-1}+h_n)\cdots(h_{n-p+1}+\cdots+h_n). \tag{4.8}$$

This model describes the behaviour of $\hat{r}_n$ with respect to the stepsizes. It is a rather complex error model, which depends on p previous stepsizes. Note that for one step methods, $\hat{r}_n$ only depends on the last stepsize h_n.

Again, $\hat{r}_n$ must satisfy the constraints

$$\forall_n\hat{r}_n \leq \text{TOL}, \tag{4.9}$$

where TOL is the tolerance level.

First order approximation

Often model (4.8) is simplified, to make the analysis less complex. Introduce for all n

$$\Omega_n = \frac{1}{p!}\xi_{n,1}\cdots\xi_{n,p}.$$

Then, $\hat{r}_n$ also satisfies the next model:

$$\hat{r}_n = \hat{\varphi}_n\Omega_n h_n^P.$$

[2]The coefficients $\xi_{n,m}$ have been defined in the equation (3.25).

Now, it is assumed that Ω_n is nearly independent of the stepsize sequence, so $\Omega_n = 1 + O(h_n)$. Note that for fixed stepsizes, $\Omega_n = 1$. Define $\hat{\psi}_n$ by $\hat{\psi}_n = \hat{\phi}_n \Omega_n$, then

$$\hat{r}_n = \hat{\psi}_n h_n^P. \tag{4.10}$$

This means that $\hat{\psi}_n \doteq \hat{\varphi}_n$ also is nearly independent of the stepsizes. Note that this model is similar to the error model (4.3) of the one step methods.

4.2.3 Stability

Besides the local discretization error, also the stability[3] region of the methods is important. Although the stability region can only be determined for the linear test equation, it is possible to use this region S also for nonlinear DAE's. In that case, at each iteration, the DAE is linearized at t_{n-1}. Then, it is required that $h_n\lambda \in S$ for all eigenvalues λ of the Jacobian matrix.

Of course, it is possible to compute all eigenvalues and to verify whether for all eigenvalues $h\lambda \in S$ holds, with S the stability region for the test equation. Then, we get the next upper bound for h_n, where J_n is the Jacobian matrix at t_{n-1}.

$$h_n \leq \max\{h : \forall_{\lambda \in \sigma(J_n)} h\lambda \in S\}. \tag{4.11}$$

Method (4.11) can be very expensive, because of the computation of all eigenvalues. Therefore, one has to use heuristics. In general, the dominant eigenvalue, which determines the upperbound do not need to be the eigenvalue with maximal amplitude λ_{max}. However, for explicit RK-methods, S has the next form:

$$S = \{\bar{h} : |P(h\lambda)| \leq 1\}.$$

with $P(z)$ a polynomial[4]. In that case, when the stability regions are bounded, a good upper bound is:

$$h_n \leq \max\{h : |P(h\lambda_{max})| \leq 1\}.$$

In practice, it is much cheaper to compute λ_{max} than the whole spectrum of J_n. For high tolerance levels, the small stepsizes will often result in stability, because then $|h\lambda|$ is very small. However, for low tolerance levels and high frequencies, the stability can be very important.

In [13], Gustafsson proposed to use stepsize control also for the stability.

4.2.4 Classical approach

Numerical DAE solvers with stepsize control have always the following structure. The method starts with an initial stepsize, which must be guessed. Then, $\mathbf{x}_n$ is computed from the nonlinear equation and the local discretization error (LDE) $\hat{r}_n$ is estimated. If the LDE is small enough, the step is accepted and the next step is computed. If the step is rejected, the stepsize has to be reduced and $\mathbf{x}_n$ is calculated again. Clearly, a large number of rejections will slow down the performance of the method. Therefore, the stepsizes are computed on base of the smaller tolerance $\epsilon = \theta \mathrm{TOL}$ where $0 < \theta < 1$ is a safety factor. In that case, the stepsizes will become smaller, which reduce the probability of rejections.

A schematic overview of adaptive stepsize control has been shown in Fig.(4.1).

[3] This concept has been introduced on page 26.

[4] See subsection 3.3.2.

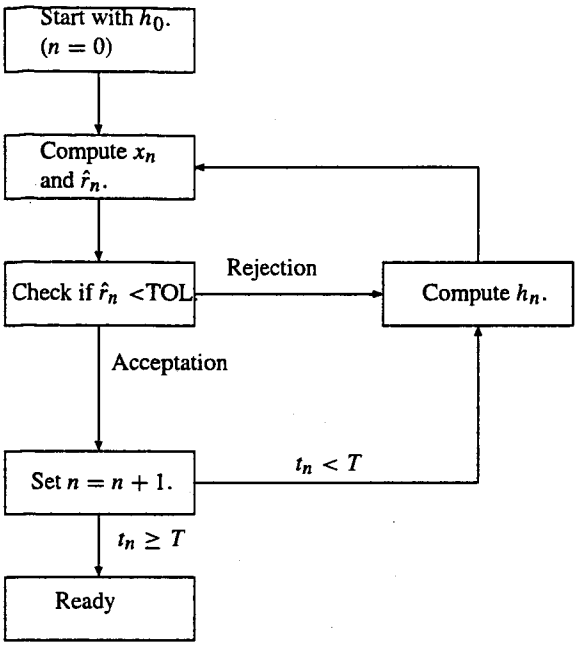


Figure 4.1: Schematic overview of stepsize control for DAE solvers.

In the most classic DAE-solvers, the stepsize controllers use the elementary control law[5] together with some additional nonlinear control actions. The nonlinearity is caused by the many logical if-else statements. However, the logarithmic version of the elementary control law is linear. In practice, the stepsize controller is a combination of a linear and a non-linear part. In the classical case, the linear part is an ordinary deadbeat-controller, while the non-linear part may consist of saturations, dead-zones, memory, etc. The idea of Söderlind[6] is to expand the linear part to a PID-controller with free control parameters and to reduce the non-linear part. Linear controllers result in much smoother stepsize sequences than the original non-linear controllers. In Fig.(4.2), the structure of the classical stepsize controllers is shown.

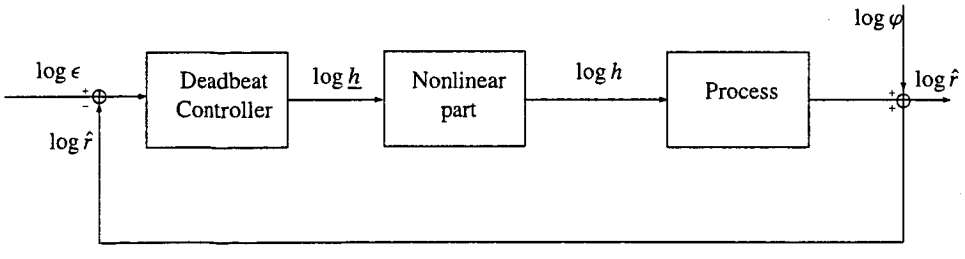


Figure 4.2: Structure of classical approach to stepsize control.

4.2.5 Specifications of a good stepsize controller

Stepsize controllers must preferably satisfy the following specifications:

[5] See equation (4.6).

[6] See [34].

Table 4.1: Important parameters for the computational work of an integration method.

Parameters	Description
C	Work load per Newton iteration
m	Average number of Newton iterations per step
N	Number of timesteps
R	Number of rejections

- The computational work with a stepsize controller must be small.
- The local errors $\hat{r}_n$ must be smaller than TOL.
- The stepsize and error sequences $\hat{r}$ and h must be smooth. If this is the case, it is expected that the local errors also depend smoothly on the circuit parameters. This is important for optimization purposes.

The computational work W will satisfy the next equation, where the parameters are defined in Tab.(4.1)..

$$W = Cm(N + R).$$

Because $m = 1$ for linear systems, it follows that W is small if $N+R$ is small. However, for nonlinear systems, m will become larger if the stepsizes become larger. Thus, the optimal size of the stepsizes is a trade off between the number of stepsizes, rejections and Newton iterations per step.

The second specification will always be satisfied, because the timestep would have been rejected, otherwise. Then other control techniques are used to make h_n smaller, which will also result in smaller errors. But, a low number of rejections will decrease the computational load.

The third specification is a rather vague one, because of the vague notion of smoothness. Of course, it is possible to examine the smoothness by means of a picture. However, with this method, it is hard to compare the smoothness of the results for two stepsize controllers.

Therefore, it has been tried to quantify the smoothness of a finite sequence $\{x_n\}_{n=1}^{N}$ by a number $s(x)$:

$$s(x) = \frac{\sqrt{\sum_{m=1}^{N}(x_m - x_{m-1})^2}}{\|x\|_2}. \tag{4.12}$$

Of course, there might be many other functions, which could indicate the smoothness of a sequence.

It is possible to define a optimality criterion, e.g.

$$J = \alpha W + \beta s(\hat{r}) + \gamma s(h).$$

Afterwards, optimal control could be used to determine the optimal controller.

4.2.6 Available literature

The application of control theory to adaptive stepsize control in numerical transient integration has already been studied by several authors.

Articles by Söderlind: [34, 35] In these articles, Söderlind proposes the use of control theory for adaptive timestepping. For one step methods, it is possible to design digital linear controllers, such that the poles of the closed loop dynamics are placed, while also the controller can have filter properties or higher adaptivity.

PhD thesis by Sjö: [33] In this thesis, A.Sjö studied the LMM-methods and in particular the Adams-Moulton and the BDF-methods. He also investigated the possibilities of control theory for LMM-methods. However, he only considers ordinary differential equations.

Bachelor thesis by Appel: [2] The circuit simulation package TITAN is a software package within the Infineon Technologies AG, which is used for simulating electrical circuits. In 2000, Appel has investigated the next possible approaches:

- a transformation of the local discretization error (LDE) to the local error of voltages and currents;
- a new error estimation of the LDE;
- a new strategy for accepting/refusing timepoints;
- a new stepsize controller (PID or predictive PID[7]);
- a smoother norm for the error.

Furthermore, stepsize control is used to avoid too many Newton-iterations. Numerical tests show that these approaches improve the reliability concerning the local accuracy of output signals. However, only for some test circuits, also the efficiency has been improved. Appel expects that after refined tuning of the parameters, the efficiency is improved for most circuits.

Articles by Gustafsson: [13, 14] In these articles, K.Gustafsson studied the results of PID-controllers) for explicit and implicit Runge Kutta methods. He also studied the possibility to use these stepsize controllers with explicit methods, which are not unconditionally stable.

4.2.7 Order control

Besides the stepsizes, also the order p can be used to control the errors.

For smooth solutions, it is better to use higher order methods, because then it is possible to use larger stepsizes. But if the solution has singularities, a lower order method is more efficient, because it needs less previous results.

The next algorithm is an example of order control, which locally optimizes the magnitudes of the stepsizes.

- Assume that the previous solution is computed with a method of order $p \geq 1$. Compute for the orders $s \in \{p-1, p, p+1\}$ with $s \geq 1$ the stepsizes h_s, such that the discretization error estimates for order s are equal to ϵ.
- Determine s^*, which corresponds with $h_{s^*} = \max\{h_s, s \in \{p-1, p, p+1\} \wedge s \geq 1\}$.
- The next order becomes s^*, such that the stepsize is maximal.

Order control can be more efficient, but has the disadvantage that it results in more complex error models.

[7] These controllers belong to a well-known class of controllers in control theory.

4.3 Theory of digital linear control systems

Because it appears that the numerical integration process can be modelled as a digital linear control system, this section will recall some aspects of them[8].

Consider the input signal $u = \{u_k\}_{k=1}^N$ and a disturbance signal $w = \{w_k\}_{k=1}^N$.

Assume that the state $x = \{x_k\}_{k=1}^N$ satisfies the recurrent relation:

$$x_n + a_1 x_{n-1} + \cdots + a_m x_{n-m} = b_0 u_n + \cdots + b_m u_{n-m}. \tag{4.13}$$

Furthermore, the output signal $y = \{y_k\}_{k=1}^N$ is disturbed by w:

$$y_n = x_n + w_n. \tag{4.14}$$

Introduce the shift-operator q with[9] $qu_n = u_{n+1}$ and $q^{-1}u_n = u_{n-1}$. Then the model (4.13) can also be written as:

$$(1 + a_1 q^{-1} + \cdots + a_m q^{-m})x_n = (b_0 + \cdots + b_m q^{-m})u_n. \tag{4.15}$$

If $N \to \infty$, it appears attractive to describe this model in the z-domain. The z-transform of an infinite signal u is defined by:

$$u(z) = \zeta\{u\} = \sum_{k=1}^{\infty} u_k z^{-k}.$$

Because of the shift property, it follows that

$$\begin{array}{rcccl} \zeta\{q^{-1}u\} & = & \sum_{k=1}^{\infty} u_{k-1} z^{-k} & = & z^{-1}u(z), \\ \zeta\{qu\} & = & \sum_{k=1}^{\infty} u_{k+1} z^{-k} & = & zu(z) - u_1. \end{array}$$

After neglecting the influence of the initial values of x and u, it is clear that in the z-domain, (4.15) is equivalent to:

$$(1 + a_1 z^{-1} + \cdots + a_m z^{-m})x(z) = (b_0 + \cdots + b_m z^{-m})u(z).$$

Now introduce the rational transfer function

$$G(z) = \frac{b_0 + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}}.$$

Then, in the z-domain, the system can be described as:

$$\left\{ \begin{array}{rcl} x(z) & = & G(z)u(z), \\ y(z) & = & x(z) + w(z). \end{array} \right. \tag{4.16}$$

This implies the next input-output relation for y:

$$y(z) = G(z)u(z) + w(z). \tag{4.17}$$

This is called the process model of the modelled system.

[8] For more information about this subject, the reader is referred to [23, 24, 27].

[9] In fact, q is an operator with the property that $(qu)_n = u_{n+1}$.

4.3.1 Controller model

With use of a feedback control law, one can control system (4.17). The input u is fed back by the control error $r - y$, where r is a given reference signal. For a digital linear controller, this feedback law can be described with use of the shift-operator: $u = C(q)(r - y)$, where $C(q)$ denotes the effect of some recursion. In the z-domain, this is equivalent to

$$u(z) = C(z)(r(z) - y(z)) \tag{4.18}$$

This is called the model of the controller. Note that $C(z)$ is a rational function of z. The coefficients are still free and are called the control parameters.

It is possible to define families of controllers, which have fewer free coefficients. A very common[10] family of linear controllers are the PID-controllers with

$$C(z) = k_P \frac{1}{z} + k_I \frac{1}{z-1} + k_D \frac{z-1}{z^2} = \frac{(k_I + k_P)z^2 + (k_D - k_P)z - k_D}{z^2(z-1)}.$$

The numbers k_P, k_I and k_D are the control parameters, called the proportional gain, the integral gain and the derivative gain, respectively. Special cases are the PI-controller ($k_D = 0$) and the I-controller ($k_D = k_P = 0$) with

$$\begin{array}{lll} \text{PI:} & C(z) & = \frac{(k_I+k_P)z-k_P}{z(z-1)} = k_P\frac{1}{z} + k_I\frac{1}{z-1}, \\ \text{I:} & C(z) & = \frac{k_I}{z-1}. \end{array}$$

Another controller family is the family of predictive PI-controllers (also called PC-controllers) with

$$C(z) = \frac{(k_R + k_E)z - k_R}{(z-1)^2}.$$

4.3.2 Closed loop dynamics

Transfer functions

In Fig.(4.3), this digital linear control system has been visualized. The input u is controlled by controller $C(q)$, which depends on the control error $r - y$. This means that if $y = r$, the controller will not perform any action. The behaviour of the state x is described by the process model $G(q)$. Furthermore, the output y is disturbed by an additional output disturbance w. This model is mathematically described by the next equations:

$$\begin{array}{lll} u & = & C(q)(r - y), \\ x & = & G(q)u, \\ y & = & x + w. \end{array} \tag{4.19}$$

[10] These controllers are frequently used in engineering applications. There is much literature about these controllers. More information can be found in [1].

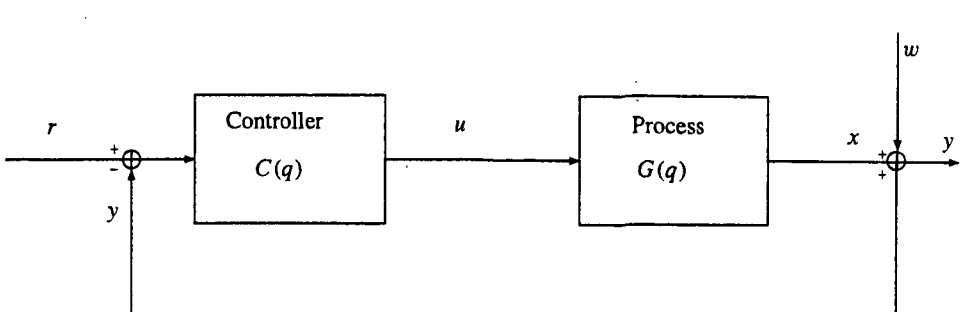


Figure 4.3: Structure of classical approach to stepsize control.

The closed loop dynamics of this feedback controlled system are described by the next closed loop model:

$$\begin{cases} u(z) = U_r(z)r(z) + U_w(z)w(z), \\ y(z) = Y_r(z)r(z) + Y_w(z)w(z). \end{cases}$$

These transfer functions express the sensitivities of the input and output to r and w.

For this case:

$$\begin{aligned} U_r(z) &= \tfrac{C(z)}{1+C(z)G(z)}, \quad U_w(z) = \tfrac{-C(z)}{1+C(z)G(z)}, \\ Y_r(z) &= \tfrac{C(z)G(z)}{1+C(z)G(z)}, \quad Y_w(z) = \tfrac{1}{1+C(z)G(z)}. \end{aligned} \tag{4.20}$$

From these four closed loop transfer functions, it is possible to derive the analytic behaviour of the input and output.

The frequency responses of the system describe the damping of the different frequency components of y. Commonly, the frequency response is measured in dB and shown for $\omega \in [0, \pi]$ in log-log diagrams. For this system, the frequence response of y with respect to w is $|Y_w(e^{i\omega})|$.

Poles

The system is stable[11] if bounded input implies that also the output will remain bounded. It can be proved that if all poles z^* of the closed loop transfer functions satisfy $|z^*| \leq 1$, the system is stable.

Besides stability, the poles also determine the behaviour of the output.

Consider the first order model:

$$(z - z_1^*)y(z) = 0.$$

Then y satisfies the next recurrent relation:

$$y_k = z_1^* y_{k-1}.$$

For each timestep, the solution is multiplied with the complex number $z_1^* = ae^{i\omega}$, such that

$$y_k = a^k e^{ik\omega} y_0. \tag{4.21}$$

Thus, the angles of the complex poles determine the frequencies of the solution. Furthermore, the absolute values are the corresponding damping factors.

[11] In this thesis, only BIBO-stability is considered. This means that an bounded input implies that also the output will be bounded.

Clearly, a and φ have the next properties:

$\lvert a\rvert = 1$	No damping
$\lvert a\rvert \in (0, 1)$	Damping
$a = 0$	The output converges in finite time
$\omega = 0$	No oscillations
$\omega \in (0, \pi)$	Oscillation with period $\frac{2\pi}{\omega}$
$\omega = \pi$	Alternating behaviour

So, for smooth control without oscillations or alternations, the poles must be real and positive. If $a = 0$, deadbeat control is used, which can result in not-smooth behaviour. If $a > 0$, the controller reacts slower, which results in smoother output. However, if $a \to 1$, the controller becomes very slow, which reduces the performance. So, there is a trade-off between poles with large and small absolute values.

Adaptivity and final error

First order adaptivity Assume that the disturbance w is constant, so it follows that $(z-1)w(z) = 0$. Then, if $(z-1)$ is a factor of $Y_w(z)$, we get $Y_w(z)w(z) = 0$. This property is equivalent with $Y_w(1) = 0$. In this case, the controller is adaptive with order one. Because of the final value theorem[12], we have

$$\lim_{n\to\infty} y_n = Y_r(1) \lim_{n\to\infty} r_n + Y_w(1) \lim_{n\to\infty} w_n.$$

If also $Y_r(1) = 1$, the final error is equal to zero. Together with stability, this implies convergence:

$$y_n - r_n \to 0 \quad \text{if } n \to \infty.$$

The same theorem also holds for the input:

$$\lim_{n\to\infty} u_n = U_r(1) \lim_{n\to\infty} r_n + U_w(1) \lim_{n\to\infty} w_n.$$

So, $U_r(1)$ and $U_w(1)$ indicate the global magnitude of the input signal.

Higher order adaptivity It is also possible to construct controllers with higher order adaptivity, which are also able to follow higher order effects, for example linear trends. Assume that the disturbance is a polynomial of degree p, such that:

$$w_n = C_0 + C_1 n + \cdots + C_p n^p.$$

Because

$$(q-1)w_n = w_{n+1} - w_n = C_0 + C_1(n+1) + \cdots + C_p(n+1)^p - (C_0 + C_1 n + \cdots + C_p n^p) = \sum_{l=1}^{p} C_l[(n+1)^l - n^l],$$

is a polynomial of degree $p-1$, it follows that $(z-1)w(z)$ is of degree $p-1$. Because $(q-1)C_0 = 0$, it follows by induction that

$$(z-1)^{p+1} w(z) = 0.$$

Thus, if $Y_w(z)$ is divisible by $(z-1)^{p_A}$ and $w(z)$ is a polynomial of degree $p_A - 1$, we get $Y_w(z)w(z) = 0$. Thus the controller is adaptive with order p_A if $z^* = 1$ is a root of $Y_w(z)$ with multiplicity p_A. Note that this implies for the frequency response that

$$|G_d(e^{i\omega})| = O(|e^{i\omega - 1}|) = O(|\omega|^{p_A}), \quad \omega \to 0.$$

[12] See [23].

Filter properties

So, the controller can be designed such that the output y will converge to the reference signal r. But it is also possible to filter the high frequency components from h, y or the control error $r - y$.

If the controller has a low pass filter property, the filtered signal will not depend on the high frequency components of the non-smooth signal w.

The controller has input filter order p_F with respect to w if

$$|U_w(e^{i\omega})| = O(|\omega - \pi|^{p_F}), \quad \omega \to \pi. \tag{4.22}$$

This means that always the alternations are filtered, while also frequency components near $\omega = \pi$ are removed. Higher filter order results in a smaller transition band between the passed and filtered frequencies. Because

$$\lim_{\omega \to \pi} \frac{e^{i\omega} + 1}{\omega - \pi} = -i,$$

it follows that the equation (4.22) is equivalent to

$$|U_w(e^{i\omega})| = O(|e^{i\omega} + 1|^{p_F}), \quad \omega \to \pi.$$

Because $\lim_{\omega \to \pi} e^{i\omega} = -1$, it follows that

$$|U_w(z)| = O(|z + 1|^{p_F}), \quad z \to -1.$$

Thus, if $U_w(z)$ is divisible by $(z + 1)^{p_F}$, its filter order is equal to p_F. This means that $z^* = -1$ is a zero of $U_w(z)$ with multiplicity p_F.

It is also possible to filter the output signal itself. The controller has output filter order p_R with respect to w if

$$|Y_w(e^{i\omega})| = O(|\omega - \pi|^{p_R}), \quad \omega \to \pi. \tag{4.23}$$

Again, this is satisfied if $Y_w(z)$ is divisible by $(z + 1)^{p_R}$.

4.4 Control-theoretic approach to adaptive timestepping

4.4.1 Introduction

In this chapter, the problem of adaptive stepsize control is approached with control-theoretic techniques. Background literature can be found in [13, 14, 33, 34, 35]. The process model $G(q)$ describes the dependance of the error estimators $\hat{r}_n$ of the stepsizes h_n. It is assumed that it can be written in the next form:

$$\log \hat{r} = G(q) \log h + \log \hat{\varphi}, \tag{4.24}$$

where $\hat{\varphi}$ is viewed as an external disturbance. Now, a control law is stated, where $C(q)$ is the controller model with still free control parameters.

$$\log h = C(q)(\log \epsilon - \log \hat{r}). \tag{4.25}$$

After all, the closed loop dynamics are studied and appropriate control parameters are selected. Thus, the controller only consists of linear control actions. Otherwise, it would not be allowed to use the theoretical results from previous section. In Fig.(4.4), the diagram of this control system is shown.

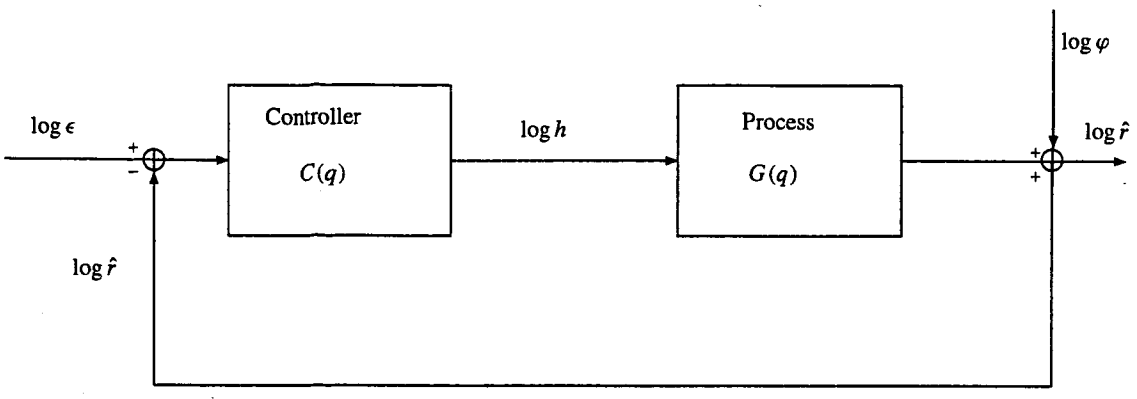


Figure 4.4: Diagram of adaptive stepsize control viewed as a feedback control system.

Because $G(q)$ is necessary to control the system, the process model will be determined first. Afterwards, a controller $C(q)$ will be designed, such that the LTE is controlled correctly. First, the family of PI-controllers will be considered. Then, a general design approach will be studied.

4.4.2 Process models of local discretization error

In the previous chapter, we have studied the error models for the one step methods and the LMM-methods. The one step-methods have simpler error models than the LMM-methods. The LMM-methods have much more complex models for $\hat{r}$, which cannot be written in the form of model (4.24). However, Sjö proposed to use linearization techniques to make it possible to handle also the LMM-methods with control-theoretic techniques. Below, two process models have been introduced. The first model describes $\hat{r}$ for one step methods, such as the RK methods or Euler Backward. Furthermore, it is also a first order approximation for the error model of the BDF-methods. The second model is only true for the BDF-methods and is derived with use of linearization techniques. Probably, this model will be more accurate for the BDF-methods.

Besides these theoretical process models, it is also possible to use system identification to determine $G(q)$. Then, it is also possible to derive a better model for the disturbance $\log \hat{\varphi}$. However, then it is necessary to get data before using the process model.

Process model one

In this case, the first order approximation (4.10) is used as model for $\hat{r}_n$. This is equivalent to

$$\log \hat{r}_n = P \log h_n + \log \hat{\psi}_n. \tag{4.26}$$

So, it follows that the process model is a constant gain:

$$G(q) = P \tag{4.27}$$

Process model two

In this case, the more accurate model (4.8) is used as model for $\hat{r}_n$. Unfortunately, the logarithmic version of this model can not be described as a digital linear control system.

$$\log \hat{r}_n = (1 + P - p) \log h_n + \log(h_{n-1} + h_n) + \cdots + \log(h_{n-p+1} + \cdots + h_n) + \log \hat{\varphi}_n - \log p!. \tag{4.28}$$

Theorem 4.2 *Let* $\boldsymbol{x}^0 = (x_1^0, \cdots, x_n^0)^T$ *be in* $\mathbb{R}^n$, *where* $\boldsymbol{x}^0 > \boldsymbol{0}$. *Introduce* $X = x_1^0 + \cdots + x_n^0$. *Then, for all* $\boldsymbol{x} \in \mathbb{R}^n$ *with* $\boldsymbol{x} > \boldsymbol{0}$:

$$\begin{aligned}\log(x_1 + \cdots + x_n) &= \log(x_1^0 + \cdots + x_n^0) + \tfrac{x_1^0}{X}(\log x_1 - \log x_1^0) + \cdots + \tfrac{x_n^0}{X}(\log x_n - \log x_n^0) \\ &\quad + O(\| \left(\log(\tfrac{x_1}{x_1^0}), \cdots, \log(\tfrac{x_n}{x_n^0})\right) \|^2), \quad \boldsymbol{x} \to \boldsymbol{x}^0.\end{aligned}$$

Proof Define the functions $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^n$, $g^{-1} : \mathbb{R}^n \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}$ by

$$\begin{aligned} f(\mathbf{x}) &= \log(x_1 + \cdots + x_n), \\ g(\mathbf{x}) &= (e^{x_1}, \cdots, e^{x_n})^T, \\ g^{-1}(\mathbf{x}) &= (\log x_1, \cdots, \log x_n)^T, \\ h(\mathbf{x}) &= f(g(\mathbf{x})) = \log(e^{x_1} + \cdots + e^{x_n}). \end{aligned}$$

Introduce the variable $\mathbf{y} = g^{-1}(\mathbf{x})$ and constant vector $\mathbf{y}^0 = g^{-1}(\mathbf{x}^0)$. It follows that $f(\mathbf{x}) = f(g(\mathbf{y})) = h(\mathbf{y})$. Then, it follows from Taylor, that

$$h(\mathbf{y}) = h(\mathbf{y}^0) + \frac{\partial h}{\partial \mathbf{y}}(\mathbf{y}^0)(\mathbf{y} - \mathbf{y}^0) + O(\|\mathbf{y} - \mathbf{y}^0\|^2), \quad \mathbf{y} \to \mathbf{y}^0.$$

Because $\mathbf{y} = g^{-1}(\mathbf{x})$, this is equivalent to:

$$f(\mathbf{x}) = f(\mathbf{x}^0) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^0)\frac{\partial g}{\partial \mathbf{y}}(\mathbf{y}^0)(\mathbf{y} - \mathbf{y}^0) + O(\|g^{-1}(\mathbf{x}) - g^{-1}(\mathbf{x}^0)\|^2), \quad \mathbf{x} \to \mathbf{x}^0.$$

Applying this to our functions, we get

$$f(\mathbf{x}) = f(\mathbf{x}^0) + \frac{1}{x_1^0 + \cdots + x_n^0}\left(e^{y_1^0}(y_1 - y_1^0) + \cdots + e^{y_n^0}(y_n - y_n^0)\right) + O(\|g^{-1}(\mathbf{x}) - g^{-1}(\mathbf{x}^0)\|^2), \quad \mathbf{x} \to \mathbf{x}^0.$$

Hence, because $\mathbf{y} = g^{-1}(\mathbf{x})$:

$$\begin{aligned}\log(x_1 + \cdots + x_n) &= \log(x_1^0 + \cdots + x_n^0) + \tfrac{x_1^0}{X}(\log x_1 - \log x_1^0) + \cdots + \tfrac{x_n^0}{X}(\log x_n - \log x_n^0) \\ &\quad + O(\|g^{-1}(\mathbf{x}) - g^{-1}(\mathbf{x}^0)\|^2), \quad \mathbf{x} \to \mathbf{x}^0.\end{aligned}$$

Because of the definition of g^{-1}, the theorem has been proved.

□

Now, this theorem is repeatedly applied to the higher order model (4.28), with for all n, $h_n^0 = h$, i.e. the equidistant case.

$$\begin{aligned}\log \hat{r}_n \doteq\ & (1 + P - p)\log h_n + \underbrace{\log 2 + \frac{1}{2}\log h_{n-1} + \frac{1}{2}\log h_n}_{\approx \log(h_{n-1} + h_n)} \\ & + \cdots + \underbrace{\log p + \frac{1}{p}\log h_{n-p+1} + \cdots + \frac{1}{p}\log h_n}_{\approx \log(h_{n-p+1} + \cdots + h_n)} + \log \hat{\varphi}_n - \log p!.\end{aligned} \tag{4.29}$$

So, it follows that

$$\log \hat{r}_n \doteq (1 + P - p + \frac{1}{2} + \cdots + \frac{1}{p})\log h_n + (\frac{1}{2} + \cdots + \frac{1}{p})\log h_{n-1} + \cdots + \frac{1}{p}\log h_{n-p+1} + \log \hat{\varphi}_n.$$

p	P	$G(q)$
1	2	2
2	3	$\frac{5}{2}+\frac{1}{2}q^{-1}$
3	4	$\frac{17}{6}+\frac{5}{6}q^{-1}+\frac{1}{3}q^{-2}$
4	5	$\frac{37}{12}+\frac{13}{12}q^{-1}+\frac{7}{12}q^{-2}+\frac{1}{4}q^{-3}$
5	6	$\frac{197}{60}+\frac{77}{60}q^{-1}+\frac{47}{60}q^{-2}+\frac{9}{20}q^{-3}+\frac{1}{5}q^{-4}$
6	7	$\frac{69}{20}+\frac{29}{20}q^{-1}+\frac{19}{20}q^{-2}+\frac{37}{60}q^{-3}+\frac{11}{30}q^{-4}+\frac{1}{6}q^{-5}$

Table 4.2: Table with the process models for the BDF-methods with EPUS-control.

Thus, a dynamic higher order model has been derived, which can be viewed as a digital linear control system. Define for $m \in \mathbb{N}$ the numbers $\gamma_m = \sum_{n=1}^{m} \frac{1}{n}$. Then, the model can also be written as

$$\log \hat{r}_n \doteq (P - p + \gamma_p)\log h_n + (\gamma_p - \gamma_1)\log h_{n-1} + \cdots + (\gamma_p - \gamma_{p-1})\log h_{n-p+1} + \log \hat{\varphi}_n.$$

Thus, the process model is equal to:

$$\begin{aligned} G(q) &= \frac{(P-p+\gamma_p)q^{p-1}+(\gamma_p-\gamma_1)q^{p-2}+\cdots+(\gamma_p-\gamma_{p-1})}{q^{p-1}} \\ &= P - p + \gamma_p + \frac{\gamma_p-\gamma_1}{q} + \cdots + \frac{\gamma_p-\gamma_{p-1}}{q^{p-1}}. \end{aligned} \tag{4.30}$$

In Tab.(4.2), the process models for EPUS control ($P = p+1$) and $p \in \{1, \ldots, 6\}$ have been shown. Note that for the Euler Backward method, with $p = 1$, the linearized model corresponds with the fixed step model from subsection 4.4.2. However, for higher order BDF-methods, these higher order process models are dynamic, in contrast to the fixed step model level. Because of the equation (4.29), it is clear that always $G(1) = P$.

4.4.3 PI-control

Deadbeat controller

Consider the first order process model, which is controlled by the next control law with initial stepsize h_1:

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\frac{1}{P}} h_{n-1}. \tag{4.31}$$

This law is equivalent to the next equation:

$$\log h_n = \log h_{n-1} + \frac{1}{P}(\log \epsilon - \log \hat{r}_{n-1}). \tag{4.32}$$

Only if $\hat{r}_{n-1} = \epsilon$, the stepsize remains the same. The final solution for h_n is:

$$\log h_n = \log h_0 + \frac{1}{P}\sum_{m=0}^{n-1}(\log \epsilon - \log \hat{r}_m).$$

Because of the asymptotics assumption[13], $\hat{r}$ has the property that $\hat{r}_n = \hat{\varphi}_n^P h_n^P$, which is equivalent to

$$\log \hat{r}_n = P \log h_n + \log \hat{\varphi}_n. \tag{4.33}$$

[13] See subsections 4.2.1 and 4.2.2.

This equation is called the open loop process model. The error estimation $\hat{r}_n$ depends on the input h_n and the disturbance $\hat{\varphi}_n$. In the closed loop dynamics, the input satisfies the control law (4.31). After inserting the process model (4.33) in the control law (4.32), we get:

$$\log h_n = \frac{1}{P}(\log \epsilon - \log \hat{\varphi}_{n-1}). \tag{4.34}$$

Note that it is not possible to use $\hat{r}_n$ itself to compute h_n, because $\hat{r}_n$ is the error estimate of the integration method with stepsize h_n. Therefore, only the previous used error estimates can be used, e.g. $\hat{r}_{n-1}$. This control law is also called deadbeat control. It uses constant extrapolation for the disturbance $\hat{\phi}_n$. Thus, h_n does not depend on former timesteps, but only on the current prediction of the local discretization error (LDE).

Integral controller

Now, instead of the fixed power $\frac{1}{P}$, a free control parameter k_I is used.

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{k_I} h_{n-1}, \tag{4.35}$$

This is called the integral controller with k_I the integral gain. For $k_I = \frac{1}{P}$ it is equal to the elementary control law, or deadbeat control law. Using the same open loop process model (4.33) as before, the closed loop dynamics become:

$$\log h_n = (1 - Pk_I) \log h_{n-1} + k_I(\log \epsilon - \log \hat{\varphi}_{n-1}), \tag{4.36}$$

with the following characteristic equation for the homogeneous part of the recursion in $\log h_n$:

$$q = 1 - Pk_I$$

The dynamics are stable if for all roots z^* of the characteristic equation holds that $|z^*| \leq 1$. This means in this case that $|1 - Pk_I| \leq 1$. From equation (4.36), it is clear that the control paramter k_I affects the behaviour of the error and stepsize sequence. At each new timepoint, the stepsize is computed with the factor $1 - Pk_I$. Thus, if $|1 - Pk_I| > 1$, the closed loop dynamics become instable. With the following conditions for k_I, the closed loop behaviour can be predicted.

$$\begin{array}{rcl} Pk_I \in [0, 2] & \Leftrightarrow & \text{stability,} \\ Pk_I \in (1, 2) & \Leftrightarrow & \text{fast, oscillatory control,} \\ Pk_I = 1 & \Leftrightarrow & \text{deadbeat control / elementary control,} \\ Pk_I \in (0, 1) & \Leftrightarrow & \text{slow, smooth control.} \end{array}$$

For smooth stepsize sequences, it is clear that one has to choose $Pk_I \in (0, 1)$. If $Pk_I > 1$, the factor $1 - Pk_I$ becomes negative, which results in alternating behaviour. The solution of the control law (4.36) is:

$$\log h_n = (1 - Pk_I)^n \log h_0 + k_I \sum_{m=1}^{n} (1 - Pk_I)^{n-m} (\log \epsilon - \log \hat{\varphi}_{m-1}).$$

So, the stepsize depends on the sum of all past control errors. Because this sum can be viewed as a discretized integral for the continuous case, this controller is called an integral controller.

Proportional-integral controller

Consider the proportional-integral[14]. local error control law with initial stepsize h_1:

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{k_I} \left(\frac{\hat{r}_{n-2}}{\hat{r}_{n-1}}\right)^{k_P} h_{n-1} \tag{4.37}$$

with k_I the integral gain and k_P the proportional gain. Note that for $k_P = 0$, we get the integral controller, which has been introduced before. This controller can also be described in logarithmic form:

$$\log h_n - \log h_{n-1} = k_I \log \epsilon - (k_I + k_P) \log \hat{r}_{n-1} + k_P \log \hat{r}_{n-2}.$$

With use of the shift-operator, this equation is equivalent to:

$$(q^2 - q) \log h = k_I \log \epsilon + (k_P - (k_I + k_P)q) \log \hat{r}.$$

Because $\log \epsilon$ is constant, it follows that $\log h = C(q)(\log \epsilon - \log \hat{r})$, where

$$C(q) = \frac{(k_P - (k_I + k_P)q)}{q^2 - q} = k_I \frac{1}{q-1} + k_P \frac{1}{q}.$$

Note that the two previous controllers are special cases of this PI controller. Assuming again that the process model (4.33) is valid for all n, the closed loop dynamics become:

$$\log h_n = (1 - Pk_I - Pk_P) \log h_{n-1} + Pk_P \log h_{n-2} + k_I (\log \epsilon - \log \hat{\varphi}_{n-1}) + k_P (\log \hat{\varphi}_{n-2} - \log \hat{\varphi}_{n-1}),$$

with characteristic equation

$$q^2 - (1 - Pk_I - Pk_P)q - Pk_P = 0. \tag{4.38}$$

The roots of this characteristic equation are the poles of the closed loop system. If the roots of this equation have absolute values smaller or equal than one, the closed loop dynamics are stable. The roots of this equation are:

$$\begin{aligned} q_1 &= \tfrac{1}{2}(1 - Pk_I - Pk_P) - \tfrac{1}{2}\sqrt{(1 - Pk_I - Pk_P)^2 + 4Pk_P}, \\ q_2 &= \tfrac{1}{2}(1 - Pk_I - Pk_P) + \tfrac{1}{2}\sqrt{(1 - Pk_I - Pk_P)^2 + 4Pk_P}. \end{aligned} \tag{4.39}$$

If one wants the roots r_1 and r_2, the next values have to be chosen for the control parameters:

$$Pk_I = 1 - r_1 - r_2 + r_1 r_2, \quad Pk_P = -r_1 r_2. \tag{4.40}$$

It is possible to derive conditions for the control parameters, such that the closed loop system will be stable. Furthermore, the control parameters also affect the dynamic behaviour of the controlled system. With a lot of algebra, it can be derived that the stability region W_{PI} is equal to

$$W_{PI} = \{(k_I, k_P) : 0 \leq Pk_I \leq 4, -1 \leq Pk_P \leq 1 - \frac{1}{2} Pk_I\}.$$

The next proof is a short version, which does not describe the derivation of W_{PI}.

Theorem 4.3 • *If $(k_I, k_P) \in W_{PI}$, the closed loop dynamics are stable.*

[14] See section 4.3.

- *If* $4Pk_P + (1 - Pk_I - Pk_P)^2 > 0$, *the poles are real and different. Both poles are equal, if* $4Pk_P + (1 - Pk_I - Pk_P)^2 = 0$. *If* $4Pk_P + (1 - Pk_I - Pk_P)^2 < 0$, *the poles form a conjugate complex pole-pair.*

Proof Assume that the closed loop dynamics are stable, so for the poles, it holds that $|r_1| \leq 1$ and $|r_2| \leq 1$.

- First, it is proved that W_{PI} is the stability-region.
  - Assume that both poles r_1 and r_2 are real. Thus, from equation (4.40), it follows that $Pk_I = 1 - r_1 - r_2 + r_1 r_2$ and $Pk_P = -r_1 r_2$.
    - This means that $Pk_I = (1 - r_1)(1 - r_2) \in [0, 4]$.
    - For Pk_P, it holds that $Pk_P = -r_1 r_2 \in [-1, 1]$. Because $2Pk_P + Pk_I - 2 = -1 - r_1 - r_2 - r_1 r_2 = -(1 + r_1)(1 + r_2) \leq 0$, the control parameters must also satisfy the equation $Pk_P \leq 1 - \frac{1}{2} Pk_I$.
  - Assume that the poles are a conjugate complex pole pair $(r_1, r_2) = (re^{i\phi}, re^{-i\phi})$. It follows that $Pk_I = 1 - re^{i\phi} - re^{-i\phi} + r^2 = 1 - 2\cos(\phi) + r^2$ and $Pk_P = -r^2$.
    - Now, it follows that $Pk_I = (1 - re^{i\phi})(1 - re^{-i\phi}) = 1 - 2r\cos(\phi) + r^2 \leq 4$. Because $Pk_I \geq 1 - 2r + r^2 = (1 - r)^2 \geq 0$, it follows that $Pk_I \in [0, 4]$.
    - For Pk_P, it follows that $Pk_P = -r^2 \in [-1, 0]$.
- Because $4Pk_P + (1 - Pk_I - Pk_P)^2$ is the discriminant of the characteristic equation, the second part of the theorem 4.3 follows immediately.

□

In Fig.(4.5), the stability region W_{PI} is shown. From (4.39), it follows that W_{PI} can be divided in five subsets. These subsets have the next characteristic behaviour:

I	r_1 and r_2 are real, while only r_1 has a negative real part. Furthermore $\|r_1\| < \|r_2\|$.
II	r_1 and r_2 are real, while again only r_1 has a negative real part. But now $\|r_1\| > \|r_2\|$.
III	r_1 and r_2 are real and have positive real parts.
IV	r_1 and r_2 are real and have negative real parts.
V	r_1 and r_2 are a conjugate complex pole-pair.

Thus, the location of the control parameters determines also the dynamic behaviour of the stepsize and error sequence.

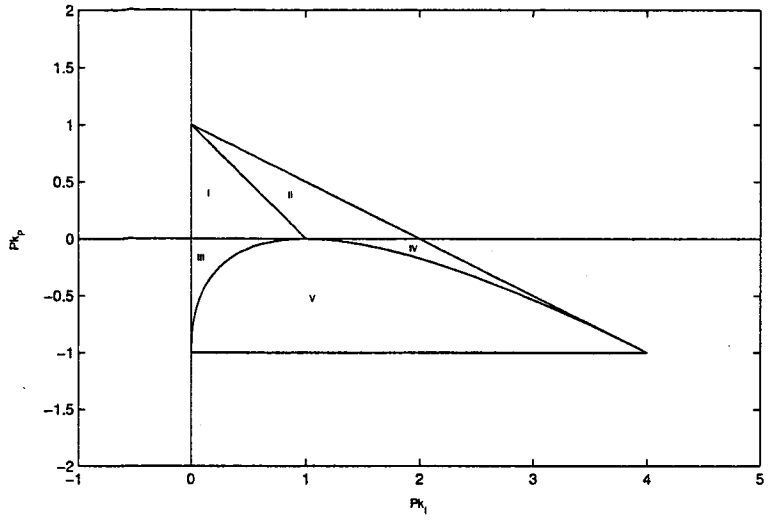


Figure 4.5: The five parts of the stability-region for the PI-controller for the process model one. The stability-region W_{PI} is bounded by the contours $Pk_I = 0$, $Pk_P = -1$ and $Pk_P = 1 - \frac{1}{2}Pk_I$. Part I and II are divided by the contour $Pk_P = 1 - Pk_I$, while part V is situated below the contour $4Pk_P + (1 - Pk_I - Pk_P)^2 = 0$.

From section 4.3, it follows that there is no oscillating behaviour, if there are only real poles. This is the case in the parts I,II,III and IV, where$4Pk_P + (1 - Pk_I - Pk_P)^2 \geq 0$. However, one can still have alternating behaviour with real poles, if at least one pole is negative[23]. From the equation (4.39), it is clear that for the real poles, where the discriminant $4Pk_P + (1 - Pk_I - Pk_P)^2$ is positive, there is always at least one negative pole, if $(1 - Pk_I - Pk_P)^2 + 4Pk_P > (1 - Pk_I - Pk_P)^2$, which is equivalent to $Pk_P > 0$. Thus, part III is indeed the unique part of W_{PI} with no alternations and oscillations. This means that PI-controllers with $(k_I, k_P) \in W_{PI}$ will result in smooth and stable controllers.

Therefore, theoretically, the parameters have to be chosen in the part III. Besides the part III, also the part I is interesting, because then the alternating component is damped faster than the other component.

Combined PI-control

Consider the stability region for the PI controllers:

$$W_{PI} = \{(k_I, k_P) : 0 \leq Pk_I \leq 4, -1 \leq Pk_P \leq 1 - \frac{1}{2}Pk_I\}.$$

It appears that the constraints are still satisfied if the control parameters are in the part I of the stability region W_{PI}.

Theorem 4.4 *Consider the PI control law.*

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{k_I} \left(\frac{\hat{r}_{n-2}}{\hat{r}_{n-1}}\right)^{k_P} h_{n-1}$$

with $(k_I, k_P) \in W_{PI}$. *The safety factor* ϵ *is equal to* θTOL, *where TOL is the tolerance level and* $0 < \theta < 1$ *the safety factor. If* $\theta^{Pk_I}\hat{\varphi}_n \leq \hat{\varphi}_{n-1}$ *and the control parameters are in the part I of the*

stability region, where $1 - Pk_I - Pk_P \geq 0$ *and* $Pk_P \geq 0$, *then the constraints are satisfied with this control law and the asymptotic process model.*

Proof Assume that $\hat{r}_1$ <TOL, so the first step is feasible. Let $n \in \mathbb{N}$ and suppose that $\hat{r}_{n-2}$ <TOL and $\hat{r}_{n-1}$ <TOL. Because $Pk_P \geq 0$ and $1 - Pk_I - Pk_P \geq 0$, it follows that

$$\hat{r}_n = \hat{\varphi}_n h_n^P = \hat{\varphi}_n \theta^{Pk_I} \hat{r}_{n-1}^{-Pk_I - Pk_P} \hat{r}_{n-2}^{Pk_P} h_{n-1}^P \mathrm{TOL}^{Pk_I} = \theta^{Pk_I} \frac{\hat{\varphi}_n}{\hat{\varphi}_{n-1}} \hat{r}_{n-1}^{1-Pk_I-Pk_P} \hat{r}_{n-2}^{Pk_P} \mathrm{TOL}^{Pk_I} \leq \mathrm{TOL}.$$

□

Note that induction is needed to prove the constraint validation for the PI-controller. This is necessary, because the PI-controller has memory, which can be used to make the error and stepsize sequence smoother. However, if one stepsize is rejected, the assumption that all previous stepsizes are smaller than ϵ does not hold. Therefore, it is better to use different controllers for the next four possible cases: Introduce the four different types of PI-controllers[15]:

- AA-controller if $1 - Pk_I - Pk_P \geq 0$ and $Pk_P \geq 0$,
- RA-controller if $1 - Pk_I - Pk_P \geq 0$ and $Pk_P \leq 0$,
- AR-controller if $1 - Pk_I - Pk_P \leq 0$ and $Pk_P \geq 0$,
- RR-controller if $1 - Pk_I - Pk_P \leq 0$ and $Pk_P \leq 0$.

The AA-controller can be used if the last two stepsizes have been (A)ccepted, the RA-controller can be used if the previous stepsize has been (R)ejected, but the last stepsize has been (A)ccepted, etc. Note that these conditions are conditions for the coefficients of the characteristic polynomial $z^2 - (1 - Pk_I - Pk_P)z - Pk_P$. Thus, the controllers are also determined by the two poles.

Controller:	$r_1 + r_2$	$r_1 r_2$
AA	≥ 0	≤ 0
RA	≥ 0	≥ 0
AR	≤ 0	≤ 0
RR	≤ 0	≥ 0

An example of these controllers with poles r_1 and r_2 is:

Controller:	r_1	r_2
AA	r	$-r$
RA	r	r
AR	r	$-r$
RR	r	r

Now, there are only two different controllers.

Because of stability, it must hold that $r \in (0, 1)$. Large r will reduce the amount of rejections, but also results in slower action.

Theorem 4.5 *Assume again that* $\theta^{Pk_I} \hat{\varphi}_n \leq \hat{\varphi}_{n-1}$.

[15] A=Accepted, R=Rejected

- *If $\hat{r}_{n-2}$ <TOL and $\hat{r}_{n-1}$ <TOL, the last two stepsizes are accepted. In that case the AA-controller gives constraint validation.*
- *If $\hat{r}_{n-2}$ <TOL, but $\hat{r}_{n-1}$ >TOL, the last stepsize has been rejected. Then the AR-controller gives constraint validation.*
- *If $\hat{r}_{n-2}$ >TOL and $\hat{r}_{n-1}$ >TOL, the last two stepsizes are rejected. In that case, the RR-controller gives constraint validation.*
- *If $\hat{r}_{n-2}$ >TOL, but $\hat{r}_{n-1}$ <TOL, the last stepsize is accepted. Then, the RA-controller gives constraint validation.*

Proof For the PI-controller, it follows that

$$\hat{r}_n \leq \hat{r}_{n-1}^{1-Pk_I-Pk_P}\hat{r}_{n-2}^{Pk_P}\mathrm{TOL}^{Pk_I}.$$

- If $\hat{r}_{n-2}$ <TOL and $\hat{r}_{n-1}$ <TOL, then it follows for the AA-controller that

$$\hat{r}_n \leq \mathrm{TOL}.$$

- If $\hat{r}_{n-2}$ <TOL, but $\hat{r}_{n-1}$ >TOL, it follows that $\hat{r}_{n-1}^{-1} < \mathrm{TOL}^{-1}$. Thus, with the AR-controller, we get

$$\hat{r}_n \leq (\hat{r}_{n-1}^{-1})^{Pk_I+Pk_P-1}\hat{r}_{n-2}^{Pk_P}\mathrm{TOL}^{Pk_I} \leq \mathrm{TOL}.$$

- If $\hat{r}_{n-2}$ >TOL and $\hat{r}_{n-1}$ >TOL, then it follows that $\hat{r}_{n-1}^{-1} < \mathrm{TOL}^{-1}$ and $\hat{r}_{n-2}^{-1} < \mathrm{TOL}^{-1}$. Now, it is clear that the RR-controller gives constraint validation.
- If $\hat{r}_{n-2}$ >TOL, but $\hat{r}_{n-1}$ <TOL, it is clear that the RA-controller gives constraint validation.

□

Of course, now the closed loop dynamics become very complex and are very difficult to analyse. However, because the number of rejections will be much smaller, than in the ordinary case, the AA-controller will control the most stepsizes. Therefore, the closed loop dynamics are mainly determined by the closed loop dynamics with the AA-controller.

Predictive PI -control

In the previous sections, it is shown that it is possible to use PI-control for adaptive stepsize control. The main assumption is:

$$\theta^{Pk_I}\hat{\varphi}_n \leq \hat{\varphi}_{n-1}.$$

If the DAE has stiff behaviour, this assumption does not need to hold. This means that in that case, it is not sufficient to use first order extrapolation to predict $\hat{\varphi}_n$. So, then higher order extrapolation is used, which also needs former $\hat{\varphi}_k$'s. Because $\hat{r}_n = \hat{\varphi}_n h_n^P$, the ideal control law results in:

$$h_n = \left(\frac{\epsilon}{\hat{\varphi}_n}\right)^{\frac{1}{P}}.$$

Introduce the second order predictor for $\log\hat{\varphi}_n$ with

$$\log\hat{\varphi}_n \doteq 2\log\hat{\varphi}_{n-1} - \log\hat{\varphi}_{n-2}.$$

This results in the second order control law:

$$\frac{h_n}{h_{n-1}} = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\frac{1}{P}} \left(\frac{\hat{r}_{n-2}}{\hat{r}_{n-1}}\right)^{\frac{1}{P}} \frac{h_{n-1}}{h_{n-2}}. \tag{4.41}$$

This control law appears to be second order adaptive[16] with respect to $\hat{\varphi}$, which implies the ability to neglect linear trends. This law can be generalized to the next predictive controller:

$$\frac{h_n}{h_{n-1}} = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{k_E} \left(\frac{\hat{r}_{n-2}}{\hat{r}_{n-1}}\right)^{k_R} \frac{h_{n-1}}{h_{n-2}} \tag{4.42}$$

with k_E and k_R the control parameters. Because this is a second order controller, it needs two initial stepsizes h_0 and h_1. This controller can also be described in logarithmic form:

$$\log h_n - 2\log h_{n-1} + \log h_{n-2} = k_E \log \epsilon - (k_E + k_R)\log \hat{r}_{n-1} + k_R \log \hat{r}_{n-2}.$$

With use of the shift-operator, this equation is equivalent to:

$$(q-1)^2 \log h = k_E \log \epsilon + (k_R - (k_E + k_R)q)\log \hat{r}.$$

Because $\log \epsilon$ is constant, it follows that $\log h = C(q)(\log \epsilon - \log \hat{r})$, where

$$C(q) = \frac{(k_R - (k_E + k_R)q)}{(q-1)^2} = \frac{1}{q-1}\left(k_E \frac{q}{q-1} + k_R\right).$$

Note that this controller consists of a double integrator, which now is able to follow linear trends. Because $z^* = 1$ is a root with multiplicity 2, this controller is adaptive with order 2. For the fixed step process model, the closed loop dynamics are determined by the next second order characteristic equation:

$$z^2 - (2 - Pk_E - Pk_R)z + 1 - Pk_R = 0$$

Because for $Pk_E = Pk_I$ and $Pk_R = 1 + Pk_P$, we retrieve the characteristic equation for the PI-controller, the PC-controller is stable if $(Pk_E, Pk_R) \in W_{PC}$ with

$$W_{PC} = \{(k_E, k_R) = (k_I, k_P + \frac{1}{P}) : (k_I, k_P) \in W_{PI}\}$$

with W_{PI} the stability region of the PI-controller. To place the poles r_1 and r_2, the next control parameters have to be chosen[17]:

$$Pk_E = 1 - r_1 - r_2 + r_1 r_2, \quad Pk_R = 1 - r_1 - r_2.$$

4.4.4 Design of finite order digital linear controller

General approach

Assume that the transfer function of the process model of order M is equal to $G(z)$, with $G(z)$ a rational function of z. Define the polynomials $K(z)$ and $L(z)$, such that

$$G(z) = \frac{L(z)}{K(z)} = \frac{\lambda_0 z^M + \cdots + \lambda_M}{z^M + \kappa_1 z^{M-1} + \cdots + \kappa_M}. \tag{4.43}$$

[16] The adaptivity of a controller has been introduced in section 4.3. Second order adaptivity means that the controller can also follow linear trends of $\hat{\varphi}$.

[17] See the equation (4.40).

Note that for models (4.27) and (4.30), we have $M = 0$ and $M = p - 1$, respectively. Similarly, the controller of order N is defined by the rational function $C(z)$. Introduce the polynomials $A(z)$ and $B(z)$, such that

$$C(z) = \frac{B(z)}{A(z)} = \frac{\beta_0 z^{N-1} + \cdots + \beta_{N-1}}{z^N + \alpha_1 z^{N-1} + \cdots + \alpha_N}. \tag{4.44}$$

So, the controller has $2N$ free parameters. Note that $B(z)$ has a lower degree than $A(z)$, because only the previous errors and stepsizes are known. It is even allowed that $B(z) = 0$, when the stepsize controller will only use the previous stepsizes. The closed loop dynamics are described by the next equations:

$$\begin{cases} \log h = U_r(q) \log \epsilon + U_w(q) \log \hat{\varphi}, \\ \log \hat{r} = Y_r(q) \log \epsilon + Y_w(q) \log \hat{\varphi}. \end{cases} \tag{4.45}$$

Here, $U_w(q)$ and $Y_w(q)$ describe the sensitivity of the stepsize and error sequence to the disturbance $\hat{\varphi}$.

Define the z-transforms

$$\begin{aligned} \zeta\{\log h\} &= u(z), \quad \zeta\{\log \epsilon\} = r(z), \\ \zeta\{\log \hat{r}\} &= y(z), \quad \zeta\{\log \hat{\varphi}\} = w(z). \end{aligned}$$

The closed loop dynamics in equation (4.45) can be described in the z-domain by

$$\begin{cases} u(z) = U_r(z) r(z) + U_w(z) w(z), \\ y(z) = Y_r(z) r(z) + Y_w(z) w(z). \end{cases}$$

with

$$\begin{aligned} U_r(z) &= \frac{B(z)K(z)}{A(z)K(z)+B(z)L(z)}, \quad U_w(z) = \frac{-B(z)K(z)}{A(z)K(z)+B(z)L(z)}, \\ Y_r(z) &= \frac{B(z)L(z)}{A(z)K(z)+B(z)L(z)}, \quad Y_w(z) = \frac{A(z)K(z)}{A(z)K(z)+B(z)L(z)}. \end{aligned} \tag{4.46}$$

So, the poles of the system are determined by the $N + M$ roots of the characteristic equation

$$A(z)K(z) + B(z)L(z) = 0.$$

With $N + M$ parameters, it is possible to place these poles to the stable poles $\{r_1, \ldots, r_{N+M}\}$.

Besides the poles, also the adaptivity and filter properties are important. The next theorem is useful for the design of $C(q)$.

Theorem 4.6 *Assume that the closed loop dynamics are stable. If $K(1) \neq 0$ and $K(-1) \neq 0$, the next statements are true:*

- *The controller is adaptive with adaptivity order p_A if $(z-1)^{p_A}$ is a divisor of $A(z)$.*
- *If $p_A \geq 1$, there is no final error. Furthermore, if $\log \hat{\varphi}$ is a polynomial of degree $p_A - 1$, there is no control error at all.*
- *Assume that $\log \hat{\varphi}_n \to \log \hat{\varphi}^*$ for $n \to \infty$, then the input $\log h_n$ converges to $\frac{1}{G(1)}(\log \epsilon - \log \hat{\varphi}^*)$ for $n \to \infty$. For models 1 and 2, this means that $\log h_n \to \frac{1}{p}(\log \epsilon - \log \hat{\varphi}^*)$, for $n \to \infty$.*
- *The controller is a stepsize filter of order p_F if $(z+1)^{p_F}$ is a divisor of $B(z)$.*
- *The controller is an error filter of order p_R if $(z+1)^{p_R}$ is a divisor of $A(z)$.*

Proof

- The controller is adaptive with adaptivity order p_A if $(z-1)^{p_A}$ is a divisor of $Y_w(z)$. Because the system is stable, this means that the denominator is not divisible by $(z-1)$. So, it follows that the controller has adaptivity order p_A if $(z-1)^{p_A}$ is a divisor of the numerator $A(z)K(z)$. Because also $K(1) \neq 0$, it follows that $(z-1)^{p_A}$ must be a divisor of $A(z)$.

- The final error is zero if $Y_r(1) = 1$. Because $p_A \geq 1$, this is always true, because then $A(1) = 0$. If $\log \hat{\varphi}$ is a polynomial of degree $p_A - 1$, it follows that $Y_w(q) \log \hat{\varphi} = 0$, so $\log \hat{r} = Y_r(q) \log \epsilon = \log \epsilon$.

- Because of the final value theorem[18], $\log h_n \rightarrow U_r(1) \log \epsilon + U_w(1) \log \hat{\varphi}^*$, for $n \rightarrow \infty$. Because $p_A \geq 1$, it follows that $A(1) = 0$. Hence, $U_r(1) = \frac{K(1)}{L(1)} = \frac{1}{G(1)} = \frac{1}{P}$.

- The stepsize filter order is p_F if $(z+1)^{p_F}$ is a divisor of $U_w(z)$. For stable systems, the denominator is not divisible by $(z+1)$. Hence, $(z+1)^{p_F}$ is a divisor of $B(z)K(z)$. If also $K(-1) \neq 0$, it follows that $(z+1)^{p_F}$ must be a divisor of $B(z)$.

- The error filter order is p_R if $(z+1)^{p_R}$ is a divisor of $Y_w(z)$. Analogous to the stepsize filter, it follows that $(z+1)^{p_R}$ must be a divisor of $A(z)$.

□

Note that the assumption for $K(z)$ is satisfied for the two process models of subsection 4.4.2.

There are $2N$ free parameters, which must be designed. Assume that $N \geq M$. After placing the $N+M$ poles, there are still $N-M$ free parameters left. These parameters can be chosen on base of the adaptivity and filter order. For the orders, the next inequalities have to be satisfied:

Theorem 4.7 *Assume that a system with process model (4.43) of order M is controlled by a controller with controller model (4.44) of order N, such that the closed loop dynamics are stable. Then, N, p_A, p_F and p_R have to satisfy the following conditions:*

$$\begin{cases} N \geq M, \\ p_A + p_F + p_R \leq N - M, \\ 1 \leq p_A \leq N, \\ 0 \leq p_R \leq N - p_A, \\ 0 \leq p_F \leq N - 1, \\ p_R = 0 \vee p_F = 0. \end{cases}$$

Proof

- Because $N+M$ poles must be placed, there must be at least $N+M$ free parameters. So, $2N \geq N+M$ or $N \geq M$.

- With the remaining $N-M$ parameters, only $N-M$ additional conditions can be added.

- To get an first order adaptive controller, it must hold that $p_A \geq 1$.

- Because both p_A and p_R depend on the roots of $A(q)$, it follows that $p_A + p_R \leq N$.

- Assume that $p_F > 0$ and $p_R > 0$. Then $B(-1) = 0$ and $A(-1) = 0$. Thus, -1 is a pole of the closed loop system, because $A(-1)K(-1) + B(-1)L(-1) = 0$. This contradicts with the stability of the system. Thus, since always $p_F \geq 0$, $p_R \geq 0$, we must have $p_F = 0$ or $p_R = 0$.

[18] This theorem has been derived from [23].

□

Note that the difference $D = N - M$ determines the grades of freedom for the controller. Because always $M \geq 0$, it follows that $N \geq D$. In the next table, for small values of D, the possibilities are shown. A controller of order N for a process model of order M with poles $\{r_1, \ldots, r_{N+M}\}$ will be denoted by $H^D_{p_A p_F p_R}(r_1, \ldots, r_{N+M})$. For these controllers, it always holds that $N = M + D$, which means that the closed loop dynamics are determined by $2M + D$ poles.

Controller	D	N	p_A	p_F	p_R
$H^1_{100}(r_1, \ldots, r_{2M+1})$	1	$M+1$	1	0	0
$H^2_{110}(r_1, \ldots, r_{2M+2})$	2	$M+2$	1	1	0
$H^2_{101}(r_1, \ldots, r_{2M+2})$	2	$M+2$	1	0	1
$H^2_{200}(r_1, \ldots, r_{2M+2})$	2	$M+2$	2	0	0
$H^3_{120}(r_1, \ldots, r_{2M+3})$	3	$M+3$	1	2	0
$H^3_{102}(r_1, \ldots, r_{2M+3})$	3	$M+3$	1	0	2
$H^3_{210}(r_1, \ldots, r_{2M+3})$	3	$M+3$	2	1	0
$H^3_{201}(r_1, \ldots, r_{2M+3})$	3	$M+3$	2	0	1
$H^3_{300}(r_1, \ldots, r_{2M+3})$	3	$M+3$	3	0	0

Controller $H^1_{100}(r_1, \ldots, r_{2M+1})$ is the standard controller, which is used most often. The controllers $H^2_{110}(r_1, \ldots, r_{2M+2})$ and $H^2_{101}(r_1, \ldots, r_{2M+2})$ have additional filter properties, while controller $H^2_{200}(r_1, \ldots, r_{M+2})$ is a predictive controller. The other controllers are higher order controllers, such that the filter orders and adaptivity can be increased.

Constraint validation

In the previous subsection, a general approach for control design has been studied. The controllers have the property that $\hat{r} \to \epsilon$ if $n \to \infty$ and $\hat{\varphi}$ is constant. Furthermore, if $\hat{\varphi}$ is a polynomial of degree $p_A - 1$, the controller will produce no control errors at all. In practice, this is never the case, so $\hat{r}$ can have deviations. If the deviation of $\hat{r}_n$ is too large, the step h_n is rejected. To reduce the number of rejections, the safety factor $0 < \theta < 1$ plays an important rôle. The controller is designed to control the difference $\log \hat{r} - \log \epsilon$ with $\epsilon = \theta \mathrm{TOL}$. The step is rejected if $\hat{r}_n > \mathrm{TOL}$. Rejections are not wanted, because then other controllers are used, which disturb the control process. Also, the speed of the DAE solver will decrease, if many rejections occur.

It is possible to analyse the behaviour of the controller with use of the closed loop dynamics.

Theorem 4.8 *Consider process model (4.43) of order M, which is controlled by a first order adaptive controller with controller model (4.44) of order N, such that the closed loop dynamics are stable. The safety factor ϵ is equal to θTOL, where TOL is the tolerance level and $0 < \theta < 1$ the safety factor. Define the polynomials $R(z)$ and $S(z)$ of degree $N + M$, such that*

$$\begin{aligned} S(z) &= A(z)K(z) &&= z^{N+M} + \sigma_1 z^{N+M-1} + \cdots + \sigma_{N+M}, \\ R(z) &= A(z)K(z) + B(z)L(z) &&= z^{N+M} + \rho_1 z^{N+M-1} + \cdots + \rho_{N+M}. \end{aligned}$$

Assume that the next conditions are satisfied:

- *The disturbance $\hat{\varphi}$ satisfies the inequality:*

$$\theta^{R(1)} \hat{\varphi}_n \hat{\varphi}_{n-1}^{\sigma_1} \cdots \hat{\varphi}_{n-N-M}^{\sigma_{N+M}} \leq 1. \tag{4.47}$$

- *The coefficients of $R(z)$ satisfy: $\rho_i \leq 0, \quad i \in \{1, \ldots, N+M\}$.*
- *The previous $N+M$ stepsizes have been accepted.*
- *The real behaviour of the error estimate $\hat{r}$ is sufficient modelled by the process model (4.43).*

Then, the next stepsize will be accepted.

Proof Because of the closed loop dynamics, it follows that

$$y(z) - r(z) = Y_w(z)(w(z) - r(z)).$$

Then $R(z)(y(z) - r(z)) = S(z)(w(z) - r(z))$. For $\hat{r}$, this means that

$$\left(\frac{\hat{r}_n}{\epsilon}\right)\left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-N-M}}{\epsilon}\right)^{\rho_{N+M}} = \left(\frac{\hat{\varphi}_n}{\epsilon}\right)\left(\frac{\hat{\varphi}_{n-1}}{\epsilon}\right)^{\sigma_1} \cdots \left(\frac{\hat{\varphi}_{n-N-M}}{\epsilon}\right)^{\sigma_{N+M}}.$$

Because $p_A \geq 1$, it follows that $A(1) = 0$. This means that $S(1) = 1 + \sigma_1 + \cdots + \sigma_{N+M} = 0$. Thus,

$$\left(\frac{\hat{r}_n}{\mathrm{TOL}}\right)\left(\frac{\hat{r}_{n-1}}{\mathrm{TOL}}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-N-M}}{\mathrm{TOL}}\right)^{\rho_{N+M}} = \theta^{R(1)}\hat{\varphi}_n\hat{\varphi}_{n-1}^{\sigma_1} \cdots \hat{\varphi}_{n-N-M}^{\sigma_{N+M}}.$$

So,

$$\hat{r}_n = \theta^{R(1)}\hat{\varphi}_n\hat{\varphi}_{n-1}^{\sigma_1} \cdots \hat{\varphi}_{n-N-M}^{\sigma_{N+M}} \left(\frac{\hat{r}_{n-1}}{\mathrm{TOL}}\right)^{-\rho_1} \cdots \left(\frac{\hat{r}_{n-N-M}}{\mathrm{TOL}}\right)^{-\rho_{N+M}} \mathrm{TOL}.$$

Because of the assumptions, it follows that $\hat{r}_n \leq \mathrm{TOL}$.

□

Note that only the second condition in theorem (4.8) depends on the used controller.

Definition 4.1 *Assume that controller (4.44) satisfies the second condition in Theorem 4.8. This property is called: constraint validation.*

Controllers with constraint validation will never give rejections, if the process model is accurate enough and if the disturbance $\hat{\varphi}$ satisfies a certain inequality. Usually, this inequality is easier satisfied, if the safety factor θ tends to zero.

Note that $R(z)$ is the characteristic polynomial of the closed loop system. This means that the coefficients of $R(z)$ depend on the pole positions. If one wants a controller with constraint validation, not all pole combinations are allowed. For example, it is not possible to design a controller with constraint validation, while all poles are equal. This means that there are no controllers without oscillations with constraint validation.

Theorem 4.9 *Consider process model (4.43) of order M, which is controlled by a controller with controller model (4.44) of order N, such that the closed loop dynamics are stable. The safety factor ϵ is equal to θTOL, where TOL is the tolerance level and $0 < \theta < 1$ the safety factor. Assume that*

- *The disturbance $\hat{\varphi}$ satisfies the inequality: $\theta^{R(1)}\hat{\varphi}_n\hat{\varphi}_{n-1}^{\sigma_1} \cdots \hat{\varphi}_{n-N-M}^{\sigma_{N+M}} \leq 1$.*
- *The previous $N+M$ stepsizes have been accepted.*
- *The process model (4.43) is correct.*

Then, the next assumptions are true.

- *If the poles are equal to* $\{r, \ldots, r\}$ *with* $r \neq 0$ *and* $N + M > 1$, *there is no constraint validation. Only, if the closed loop dynamics have one pole* $(N + M = 1)$, *the controller has constraint validation.*
- *If the poles are equal to* $\{re^{\frac{2k\pi i}{N+M}} : k = 0, \ldots, N + M - 1\}$ *with* $r \in [0, 1)$, *there is constraint validation.*
- *The deadbeat controller with all poles equal to zero always has constraint validation.*

Proof

- If all poles are equal, it follows that

$$R(z) = (z - r)^{N+M} = z^{N+M} - (N + M)rz^{N+M-1} + \binom{N+M}{2} r^2 z^{N+M-2} + \cdots + (-r)^{N+M}.$$

  This characteristic polynomial has coefficients, which are alternately positive and negative, if $r \neq 0$ and $N + M > 1$. Only if $N + M = 1$, the characteristic polynomial $R(z) = z - r_1$ has only one negative coefficient.

- For the second case, it follows that

$$R(z) = z^{N+M} - r^{N+M}.$$

  Indeed, this polynomial satisfies the condition for constraint validation.

- This is a special case of case 2, with $r = 0$.

□

So, it is possible to design a controller with constraint validation, adaptivity and filter properties. Furthermore, the absolute value of the poles is still free. If one wants also to design the angles of the poles, the controller will not have constraint validation. Thus, there is a trade-off between controllers with constraint validation and controllers without oscillations. It is also possible to use an optimality criterion to choose the pole positions.

Computation of the control parameters

In the previous subsection, several design properties have been shown. Now we will show how in practice the control parameters can be computed. Consider the system with process model (4.43) of order M. This process is controlled by controller (4.44), which must have the next properties:

- The order of the controller must be equal to $N = M + p_A + p_F + p_R$.
- The $N + M$ poles are equal to $\{r_1, \ldots, r_{N+M}\}$ and are stable.
- The adaptivity order is equal to p_A with $1 \leq p_A \leq N$.
- The filter orders are equal to p_R and p_F with $0 \leq p_R \leq N - p_A$, $0 \leq p_F \leq N - 1$ and $p_R = 0 \vee p_F = 0$.

Assume that N, p_A, p_F and p_R satisfy these conditions. The characteristic polynomial $R(z)$ is equal to $(z - r_1) \cdots (z - r_{N+M})$. Then, it follows that

$$A(z)K(z) + B(z)L(z) = R(z).$$

Because of the other properties, we can write $A(z) = (z-1)^{p_A}(z+1)^{p_R}\tilde{A}(z)$ and $B(z) = (z+1)^{p_F}\tilde{B}(z)$. Thus, we have

$$(z-1)^{p_A}(z+1)^{p_R}\tilde{A}(z)K(z) + (z+1)^{p_F}\tilde{B}(z)L(z) - R(z) = 0. \tag{4.48}$$

This monic polynomial of degree $N + M$ at the left-hand side is equal to zero, if all coefficients are equal to zero. On this manner, the $N + M$ free control parameters can be computed. For the process model one, it is easy to compute the control parameters for arbitrary P. However, for the process model two, the computations can become rather complex. In that case, one can use e.g. Mathematica[19] or MATLAB.

Below, for the both process models and $p \in \{1, 2\}$, the control parameters for controller $H^1_{100}(r_1, \ldots, r_{2M+1})$ have been computed. For the process model one, also the second order controllers have been investigated, while the order is not restricted to 2.

Process model one

For the process model one, we have[20] that $K(q) = 1$, $L(q) = P$ and $G(q) = \frac{L(q)}{K(q)} = P$. It always holds that $M = 0$ and $D = N - M = N$, so from Theorem 4.4.4, it follows that $N \geq p_A + p_F + p_R$. The control parameters will be designed for a variable order $P \in \{p, p+1\}$.

Standard controller For the controller $H^1_{100}(r_1)$ with $p_A = 1$ and $p_F = p_R = 0$, it holds that $D = N \geq 1$. For $N = 1$, it follows that

$$\begin{aligned} A(z) &= (z-1)\tilde{A}(z), \\ \tilde{A}(z) &= 1, \\ B(z) &= \tilde{B}(z) = \beta_0, \\ R(z) &= z - r_1. \end{aligned}$$

Because $K(z) = 1$ and $L(z) = P$, the equation (4.48) implies that

$$(z-1) + P\beta_0 = z - r_1.$$

Thus, it follows that $\beta_0 = \frac{1-r_1}{P}$. Because $A(z) = z - 1$, it also holds that $\alpha_1 = -1$.

Stepsize filter For the controller $H^2_{110}(r_1, r_2)$ with $p_A = p_F = 1$ and $p_R = 0$, it holds that $D = N \geq 2$. For $N = 2$, it follows that

$$\begin{aligned} A(z) &= (z-1)\tilde{A}(z), \\ \tilde{A}(z) &= z + \tilde{\alpha}_1, \\ B(z) &= (z+1)\tilde{B}(z), \\ \tilde{B}(z) &= \tilde{\beta}_0, \\ R(z) &= (z - r_1)(z - r_2). \end{aligned}$$

[19]This computer algebra package is able to compute the exact values.

[20]See the equation (4.43).

The equation (4.48) implies for this stepsize filter

$$(z-1)(z+\tilde{\alpha}_1)+P(z+1)\tilde{\beta}_0=(z-r_1)(z-r_2).$$

Thus, it follows that

$$\begin{cases} \tilde{\alpha}_1+P\tilde{\beta}_0 &= 1-r_1-r_2, \\ P\tilde{\beta}_0-\tilde{\alpha}_1 &= r_1r_2. \end{cases}$$

So, $\tilde{\alpha}_1=\frac{1}{2}(1-r_1-r_2-r_1r_2)$ and $\tilde{\beta}_0=\frac{1}{2P}(1-r_1-r_2+r_1r_2)$.

Error filter For the controller $H^2_{101}(r_1,r_2)$ with $p_A=p_R=1$ and $p_F=0$, it also holds that $D=N\geq 2$. For $N=2$, it follows that

$$\begin{gathered} A(z)=(z-1)(z+1)\tilde{A}(z), \\ \tilde{A}(z)=1, \\ B(z)=\tilde{B}(z)=\beta_0z+\beta_1, \\ R(z)=(z-r_1)(z-r_2). \end{gathered}$$

The equation (4.48) implies for this error filter

$$(z-1)(z+1)+P(\beta_0z+\beta_1)=(z-r_1)(z-r_2).$$

Now, it follows that $\beta_0=\frac{-r_1-r_2}{P}$ and $\beta_1=\frac{1+r_1r_2}{P}$. Because $A(z)=z^2-1$, it also holds that $\alpha_1=0$ and $\alpha_2=-1$.

Predictive controller For the controller $H^2_{200}(r_1,r_2)$ with $p_A=2$ and $p_F=p_R=0$, it also holds that $D=N\geq 2$. For $N=2$, it follows that

$$\begin{gathered} A(z)=(z-1)^2\tilde{A}(z), \\ \tilde{A}(z)=1, \\ B(z)=\tilde{B}(z)=\beta_0z+\beta_1, \\ R(z)=(z-r_1)(z-r_2). \end{gathered}$$

The equation (4.48) implies for this error filter

$$(z-1)^2+P(\beta_0z+\beta_1)=(z-r_1)(z-r_2).$$

So, $\beta_0=\frac{2-r_1-r_2}{P}$ and $\beta_1=\frac{r_1r_2-1}{P}$. Because $A(z)=(z-1)^2$, it also holds that $\alpha_1=-2$ and $\alpha_2=1$.

Process model two

For the process model two, the order is equal to $M=p-1$. For $p=1$, the process model is similar to the process model one. Now, for $p=2$ and $P\in\{2,3\}$, the control parameters for the standard controller $H^1_{100}(r_1,r_2,r_3)$ with $p_A=1$ and $p_F=p_R=0$ will be computed. In this case[21], $G(q)=P-\frac{1}{2}+\frac{1}{2q}$, which implies that $K(q)=q$ and $L(q)=(P-\frac{1}{2})q+\frac{1}{2}$. Now, $M=1$, $D\geq 1$ and $N\geq 2$, which implies that the closed loop dynamics have at least three poles

[21] See page 78.

Controller	Constraint validation	$\lvert r\rvert$	$\tilde{\alpha}_1$	β_0	β_1
$H^1_{100}(0)$	true	0	0	$\frac{1}{2}$	0
$H^2_{110}(0,0)$	true	0	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$
$H^2_{101}(0,0)$	true	0	1	0	$\frac{1}{2}$
$H^2_{200}(0,0)$	true	0	-1	1	$-\frac{1}{2}$
$H^1_{100}(\frac{1}{2})$	true	$\frac{1}{2}$	0	$\frac{1}{4}$	0
$H^2_{110}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$-\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
$H^2_{101}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	1	$-\frac{1}{2}$	$\frac{5}{8}$
$H^2_{200}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	-1	$\frac{1}{2}$	$-\frac{3}{8}$
$H^1_{100}(\frac{1}{2})$	true	$\frac{1}{2}$	0	$\frac{1}{4}$	0
$H^2_{110}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	$\frac{5}{8}$	$\frac{3}{16}$	$\frac{3}{16}$
$H^2_{101}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	1	0	$\frac{3}{8}$
$H^2_{200}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	-1	1	$-\frac{5}{8}$

Table 4.3: Table with the control parameters for the EPUS-version of the first order one step method.

For $N = 2$, it follows that

$$\begin{aligned} A(z) &= (z-1)\tilde{A}(z), \\ \tilde{A}(z) &= z + \tilde{\alpha}_1, \\ B(z) &= \tilde{B}(z) = \beta_0 z + \beta_1, \\ R(z) &= (z-r_1)(z-r_2)(z-r_3). \end{aligned}$$

Because $K(z) = z$ and $L(z) = (P - \frac{1}{2})z + \frac{1}{2}$, the equation (4.48) implies that

$$(z-1)(z+\tilde{\alpha}_1)z + (\beta_0 z + \beta_1)((P-\frac{1}{2})z + \frac{1}{2}) = (z-r_1)(z-r_2)(z-r_3).$$

Equating the coefficients at both sides yields

$$\left\{ \begin{array}{rcl} \tilde{\alpha}_1 + \beta_0(P-\frac{1}{2}) & = & 1 - r_1 - r_2 - r_3, \\ \frac{1}{2}\beta_0 + \beta_1(P-\frac{1}{2}) - \tilde{\alpha}_1 & = & r_1 r_2 + r_1 r_3 + r_2 r_3, \\ \frac{1}{2}\beta_1 & = & -r_1 r_2 r_3. \end{array} \right.$$

Hence, it follows that

$$\left\{ \begin{array}{rcl} \beta_0 & = & \frac{(1-r_1)(1-r_2)(1-r_3)+2Pr_1r_2r_3}{P}, \\ \beta_1 & = & -2r_1 r_2 r_3, \\ \tilde{\alpha}_1 & = & 1 - r_1 - r_2 - r_3 - (P-\frac{1}{2})\beta_0. \end{array} \right.$$

Computed controllers with Mathematica

In Tab.(4.3), the control parameters are shown for the EPUS-version of the Euler-Backward method with $p = 1$ and $P = 2$. This first order one step method can be modelled by the fixed step model without linearization. In Tables (4.4) and (4.5), for order $p = 2$ and $P = p + 1$ (EPUS), the control parameters are shown for some designed controllers[22]. Each time, the control parameters are computed for the deadbeat controller, the smooth controller and the controller with constraint validation.

[22]This has been done with use of Mathematica

Controller	Constraint validation	$\|r\|$	$\bar{\alpha}_1$	β_0	β_1
$H^1_{100}(0)$	true	0	0	$\frac{1}{3}$	0
$H^2_{110}(0,0)$	true	0	$\frac{1}{2}$	$\frac{1}{6}$	$\frac{1}{6}$
$H^2_{101}(0,0)$	true	0	1	0	$\frac{1}{3}$
$H^2_{200}(0,0)$	true	0	-1	$\frac{2}{3}$	$-\frac{1}{3}$
$H^1_{100}(\frac{1}{2})$	true	$\frac{1}{2}$	0	$\frac{1}{6}$	0
$H^2_{110}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$-\frac{1}{8}$	$\frac{1}{24}$	$\frac{1}{24}$
$H^2_{101}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	1	$-\frac{1}{3}$	$\frac{5}{12}$
$H^2_{200}(\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	-1	$\frac{1}{3}$	$-\frac{1}{4}$
$H^1_{100}(\frac{1}{2})$	true	$\frac{1}{2}$	0	$\frac{1}{6}$	0
$H^2_{110}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	$\frac{5}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
$H^2_{101}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	1	0	$\frac{1}{4}$
$H^2_{200}(\frac{1}{2},-\frac{1}{2})$	true	$\frac{1}{2}$	-1	$\frac{2}{3}$	$-\frac{5}{12}$

Table 4.4: Table with the control parameters for the second order BDF2 method with the process model one.

Controller	Constraint validation	$\|r\|$	$\bar{\alpha}_1$	$\bar{\alpha}_2$	β_0	β_1	β_2
$H^1_{100}(0,0,0)$	true	0	$\frac{1}{6}$	0	$\frac{1}{3}$	0	0
$H^2_{110}(0,0,0,0)$	true	0	$\frac{7}{12}$	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{6}$	0
$H^2_{101}(0,0,0,0)$	true	0	$\frac{29}{24}$	$\frac{5}{24}$	$-\frac{1}{12}$	$\frac{5}{12}$	0
$H^2_{200}(0,0,0,0)$	true	0	$-\frac{29}{36}$	$-\frac{7}{36}$	$\frac{13}{18}$	$-\frac{7}{18}$	0
$H^1_{100}(\frac{1}{2},\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$-\frac{59}{48}$	0	$\frac{7}{24}$	$-\frac{1}{4}$	0
$H^2_{110}(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$-\frac{137}{192}$	$\frac{157}{192}$	$-\frac{11}{96}$	$\frac{1}{96}$	$\frac{1}{8}$
$H^2_{101}(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$\frac{941}{384}$	$\frac{557}{384}$	$-\frac{265}{192}$	$\frac{245}{192}$	$\frac{1}{8}$
$H^2_{200}(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2})$	false	$\frac{1}{2}$	$-\frac{941}{576}$	$\frac{365}{576}$	$\frac{73}{288}$	$-\frac{103}{288}$	$\frac{1}{8}$
$H^1_{100}(\frac{1}{2},\frac{1}{2}e^{\frac{2}{3}\pi i},\frac{1}{2}e^{-\frac{2}{3}\pi i})$	true	$\frac{1}{2}$	$-\frac{17}{48}$	0	$\frac{13}{24}$	$-\frac{1}{4}$	0
$H^2_{110}(\frac{1}{2},\frac{1}{2}e^{\frac{1}{2}\pi i},-\frac{1}{2},\frac{1}{2}e^{-\frac{1}{2}\pi i})$	true	$\frac{1}{2}$	$\frac{19}{64}$	$-\frac{15}{64}$	$\frac{9}{32}$	$\frac{5}{32}$	$-\frac{1}{8}$
$H^2_{101}(\frac{1}{2},\frac{1}{2}e^{\frac{1}{2}\pi i},-\frac{1}{2},\frac{1}{2}e^{-\frac{1}{2}\pi i})$	true	$\frac{1}{2}$	$\frac{113}{128}$	$-\frac{15}{128}$	$\frac{3}{64}$	$\frac{25}{64}$	$-\frac{1}{8}$
$H^2_{200}(\frac{1}{2},\frac{1}{2}e^{\frac{1}{2}\pi i},-\frac{1}{2},\frac{1}{2}e^{-\frac{1}{2}\pi i})$	true	$\frac{1}{2}$	$-\frac{113}{192}$	$-\frac{79}{192}$	$\frac{61}{96}$	$-\frac{19}{96}$	$-\frac{1}{8}$

Table 4.5: Table with the control parameters for the second order BDF2 method with the process model two.

Implementation

If the coefficients of the controller $C(q)$ are derived, this controller can be used to compute the stepsizes. Assume that $p_A \geq 1$, then there exists a polynomial $\bar{A}(q)$ of degree $N-1$, such that $A(q) = (q-1)\bar{A}(q) = (q-1)(q^{N-1} + \bar{\alpha}_1 q^{N-2} + \ldots + \bar{\alpha}_{N-1})$. Now,

$$(q-1)\bar{A}(q)\log h = B(q)(\log \epsilon - \log \hat{r}).$$

This is equivalent to the control law

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\beta_0} \cdots \left(\frac{\epsilon}{\hat{r}_{n-N}}\right)^{\beta_{N-1}} \left(\frac{h_{n-1}}{h_{n-2}}\right)^{-\bar{\alpha}_1} \cdots \left(\frac{h_{n-N+1}}{h_{n-N}}\right)^{-\bar{\alpha}_{N-1}} h_{n-1}. \tag{4.49}$$

This controller can easily be implemented in a numerical DAE solver. It needs the N previous stepsizes $h_{n-1}, \ldots, h_{n-N}$ and the N previous error estimators $\hat{r}_{n-1}, \ldots, \hat{r}_{n-N}$. Controllers with large N have the disadvantage of a large history. After discontinuities, the integration order decreases to one, which implies that the process model is no longer valid. This implies that another stepsize controller has to be used, which can not use the previous stepsizes and errors. If after rejections, other control techniques are used, these results do not satisfy the closed loop dynamics.

Since the behaviour of the error estimate is not exactly modelled by the process models, it is not completely sure that controller (4.49) maintains its properties, such as adaptivity and filter-properties, for the integration process itself.

4.4.5 Nonlinear controller for the process model two

In the previous sections, the nonlinear process model has been linearized. Afterwards a linear controller has been designed, which can be used to control the nonlinear process. However, it is also possible to use a nonlinear controller, which can also be linearized.

Consider the process model two[23]. Then the ideal stepsize would satisfy the next nonlinear equation, because then $\hat{r}_n = \epsilon$.

$$\hat{\varphi}_n \frac{1}{p!} h_n^{1+P-p}(h_{n-1} + h_n) \cdots (h_{n-p+1} + \cdots + h_n) = \epsilon. \tag{4.50}$$

Because $\hat{\varphi}_n$ is not known yet, it is not possible to use this controller. Therefore, $\hat{\varphi}_n$ must be predicted by $\hat{\varphi}_n^P$ with use of the previous values of $\hat{r}$ and h. This will result in the next nonlinear controller:

$$h_n^{1+P-p}(h_{n-1} + h_n) \cdots (h_{n-p+1} + \cdots + h_n) = \frac{p!\epsilon}{\hat{\varphi}_n^P}. \tag{4.51}$$

So, h_n is the root of a polynomial of degree P. It can be shown that h_n can be computed from this equation, because each polynomial of this type has only one real positive root. From the error model (4.8) and the equation (4.51), it follows that the error estimate will be equal to

$$\hat{r}_n = \hat{\varphi}_n \frac{1}{p!} h_n^{1+P-p}(h_{n-1} + h_n) \cdots (h_{n-p+1} + \cdots + h_n) = \frac{\hat{\varphi}_n}{\hat{\varphi}_n^P}\epsilon. \tag{4.52}$$

[23] See the equation 4.8.

Theorem 4.10 *Assume that $\hat{\varphi}_n^P$ is computed with the next linear control law:*

$$\log \hat{\varphi}^P = \log \epsilon + C(q)G(q)(\log \hat{r} - \log \epsilon). \tag{4.53}$$

Then, the linearized closed loop system is equal to the process model two, which is controlled by the linear controller $\log h = C(q)(\log \epsilon - \log \hat{r})$.

Proof The linearized version of the nonlinear controller (4.51) is

$$G(q) \log h + \log \hat{\varphi}^P = \log \epsilon.$$

If $\log \hat{\varphi}^P$ satisfies the linear relation (4.53), it indeed follows that

$$\log h = C(q)(\log \epsilon - \log \hat{r}).$$

□

Because of the equation (4.52), it follows that

$$\log \hat{\varphi} = \log \hat{r} + C(q)G(q)(\log \hat{r} - \log \epsilon) = \log \hat{\varphi}^P - (\log \epsilon - \log \hat{r}). \tag{4.54}$$

Thus, if $\log \hat{r} \to \log \epsilon$, also $\log \hat{\varphi}^P \to \log \hat{\varphi}$ for $n \to \infty$. Furthermore, the difference only depends on the last control error.

Note that this approach also covers the process model one, but then the process model is already linear. This means that in that case, the nonlinear controller (4.51) corresponds exactly to the linearized

Implementation

Using the definitions on page 85, it follows that the control law (4.53) for $\log \hat{\varphi}^P$ is equivalent to

$$A(q)K(q)(\log \hat{\varphi}^P - \log \epsilon) = B(q)L(q)(\log \hat{r} - \log \epsilon).$$

Assume that $p_A \geq 1$, so again $A(q) = (q-1)\tilde{A}(q)$ and $A(q) \log \epsilon = 0$. Let $S(q)$ and $R(q)$ be equal to $S(q) = A(q)K(q)$ and $R(q) = A(q)K(q) + B(q)L(q)$, then it follows from (4.54)

$$\begin{aligned} q^{N+M} \log \hat{\varphi}^P &= (q^{N+M} - S(q)) \log \hat{\varphi}^P + B(q)L(q)(\log \hat{r} - \log \epsilon) \\ &= (q^{N+M} - S(q)) \log \hat{\varphi} + (S(q) - q^{N+M} + B(q)L(q))(\log \hat{r} - \log \epsilon) \\ &= (q^{N+M} - S(q)) \log \hat{\varphi} + (R(q) - q^{N+M})(\log \hat{r} - \log \epsilon). \end{aligned} \tag{4.55}$$

Thus, the control law reads

$$\hat{\varphi}_n^P = \hat{\varphi}_{n-1}^{-\sigma_1} \cdots \hat{\varphi}_{n-N}^{-\sigma_{N+M}} \left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-M-N}}{\epsilon}\right)^{\rho_{N+M}}. \tag{4.56}$$

Because for process model two[24], $K(q) = q^M$, it follows that $S(q) = q^M A(q)$ and

$$q^{N+M} \log \hat{\varphi}^P = q^M (q^N - A(q)) \log \hat{\varphi} + (R(q) - q^{N+M})(\log \hat{r} - \log \epsilon).$$

[24] See page 78.

Now, the control law reads

$$\hat{\varphi}_n^P = \hat{\varphi}_{n-1}^{-\alpha_1} \cdots \hat{\varphi}_{n-N}^{-\alpha_N} \left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-M-N}}{\epsilon}\right)^{\rho_{N+M}}. \tag{4.57}$$

Note that for deadbeat control, just extrapolation is used, because then $R(z) = z^{N+M}$. So, in general, the nonlinear controller performs the next algorithm:

$$\begin{cases} \hat{\varphi}_n^P = \hat{\varphi}_{n-1}^{-\sigma_1} \cdots \hat{\varphi}_{n-N}^{-\sigma_{N+M}} \left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-M-N}}{\epsilon}\right)^{\rho_{N+M}}, \\ h_n^{1+P-p}(h_{n-1} + h_n) \cdots (h_{n-p+1} + \cdots + h_n) = \frac{p!\epsilon}{\hat{\varphi}_n^P}. \end{cases} \tag{4.58}$$

The next theorem is analogous to Theorem 4.8, which indicates whether the controller will have constraint validation[25]

Theorem 4.11 *Consider the nonlinear process model (4.43) of order M, which is controlled by the controller (4.58) with $p_A \geq 1$, such that the linearized closed loop dynamics are stable and correspond to the linear feedback law $\log h = C(q)(\log \epsilon - \log \hat{r})$ with controller model (4.44). The safety factor ϵ is equal to θTOL, where TOL is the tolerance level and $0 < \theta < 1$ the safety factor. Assume that the next conditions are satisfied:*

- *The disturbance $\hat{\varphi}$ satisfies the inequality: $\theta^{R(1)} \hat{\varphi}_n \hat{\varphi}_{n-1}^{\sigma_1} \cdots \hat{\varphi}_{n-N-M}^{\sigma_{N+M}} \leq 1$.*
- *The coefficients of $R(z)$ satisfy: $\rho_i \leq 0, \quad i \in \{1, \ldots, N+M\}$.*
- *The previous $N + M$ stepsizes have been accepted.*

Then, the next stepsize will be accepted.

Proof If controller (4.58) is applied, it follows from (4.52) that

$$\hat{r}_n = \frac{\hat{\varphi}_n}{\hat{\varphi}_n^P} \epsilon.$$

Because of the equation (4.57), it follows that

$$\hat{\varphi}_n^P \geq 1.$$

So, indeed

$$\hat{r}_n \leq \epsilon.$$

□

Note that this nonlinear controller does not need the assumption that the linearized model is correct.

[25] This concept has been defined in Def.4.1.

Example 1

Consider the BDF2-method with process model 2 and $P = 3$, and which is controlled by a $H^2_{100}(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$-controller. Thus, $M = 1$, $N = 2$, $p_A = 1$, $p_F = p_R = 0$ and

$$G(q) = \frac{L(q)}{K(q)} = \frac{\frac{5}{2}q + \frac{1}{2}}{q}. \tag{4.59}$$

The three poles are equal to $\frac{1}{2}$, which means that

$$R(q) = (q - \frac{1}{2})^3 = q^3 - \frac{3}{2}q^2 + \frac{3}{4}q - \frac{1}{8}.$$

From Theorem 4.9, it follows that this controller does not have constraint validation. Therefore, Theorem 4.11 does not hold for this controller, because the second condition is not satisfied. However, because all poles are equal, there will be no oscillating behaviour. With Mathematica, it can be computed[26] that for this controller, it holds that

$$C(q) = \frac{B(q)}{A(q)} = \frac{\frac{7}{24}q - \frac{1}{4}}{(q-1)(q - \frac{59}{48})}.$$

The control law for ϕ_n^P is equal to

$$\hat{\varphi}_n^P = \hat{\varphi}_{n-1}^{-\alpha_1}\hat{\varphi}_{n-2}^{-\alpha_2}\left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\sigma_1}\left(\frac{\hat{r}_{n-2}}{\epsilon}\right)^{\sigma_2}\left(\frac{\hat{r}_{n-3}}{\epsilon}\right)^{\sigma_3}, \tag{4.60}$$

where

$$\begin{aligned} \alpha_1 &= -\tfrac{107}{48}, \\ \alpha_2 &= \tfrac{59}{48}, \\ \sigma_1 &= -\tfrac{3}{2}, \\ \sigma_2 &= \tfrac{3}{4}, \\ \sigma_3 &= -\tfrac{1}{8} \end{aligned}$$

Now, each iteration h_n is computed from the next nonlinear equation:

$$h_n^2(h_{n-1} + h_n) = \frac{2\epsilon}{\hat{\varphi}_n^P}. \tag{4.61}$$

Example 2

Because of Theorem 4.9, it follows that the controller $H^1_{100}(\frac{1}{2}, \frac{1}{2}e^{\frac{2}{3}\pi i}, \frac{1}{2}e^{-\frac{2}{3}\pi i})$ has constraint validation. Again, it holds that $M = 1$, $N = 2$, $p_A = 1$ and $p_F = p_R = 0$. Also $G(q)$ is equal to the process model in (4.59), but now the poles are different, which implies that

$$R(q) = q^3 - \frac{1}{8}.$$

Now, $\hat{\varphi}_n^P$ is computed with control law (4.60), where

$$\begin{aligned} \alpha_1 &= -\tfrac{65}{48}, \\ \alpha_2 &= \tfrac{17}{48}, \\ \sigma_1 &= 0, \\ \sigma_2 &= 0, \\ \sigma_3 &= -\tfrac{1}{8}. \end{aligned}$$

Again, each iteration h_n is computed from the equation (4.61).

[26] See page 91.

Chapter 5

Numerical experiments

5.1 Introduction

This chapter contains the results of some numerical experiments, that have been performed to verify the previous theoretical results for stepsize control. Several stepsize control strategies have been investigated for the next three applications:

- The MATLAB-implementation "ode45" of a Runge Kutta method[1],
- The MATLAB-implementation "BDFcontrol" of the BDF method[2],
- The circuit simulator Pstar[3]

The MATLAB-implementations are just investigated for some rather small examples, while for Pstar also some larger real circuits have been tested. With use of Qstar[4] it is possible to get the describing circuit functions $\mathbf{q}$ and $\mathbf{j}$, as well as the Jacobians C and G for MATLAB from a circuit node list. In section two, the new control-theoretic approach has been tested for the Runge Kutta method on a small example. Afterwards, in section three, the new controllers have been tested on the MATLAB-implementation "BDFcontrol" of the BDF method for two small testcircuits. Also, it has been shown how the DAE's can be derived for these circuits. In section four, these experiments have been repeated for the circuit simulator Pstar itself. Now, also some larger circuits have been tested.

5.2 Experiments with the Runge Kutta method in MATLAB

5.2.1 Description of the method

To test the stepsize controllers on the Runge Kutta method, an adapted version of the MATLAB-routine "ode45.m" has been used. This is an implementation of a Dormand-Prince pair with orders 4 and 5 and Butcher matrices as in Tab.(5.1):

[1] This integration method has been described in subsection 3.3.2.

[2] This integration method has been described in subsection 3.4.3.

[3] The stepsize mechanism of this circuit simulator has been described in appendix A.

[4] This is an useful tool, which can transform Pstar input files into m-files for MATLAB. For more information, the reader is referred to [21, 22].

Table 5.1: Butcher matrices of ode45.

Method :	Butcher matrix	order
RK 4	$\begin{array}{c\|cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & 0 & 0 & 0 \\ \frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & 0 & 0 & 0 \\ \frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & 0 & 0 \\ 1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & 0 \\ \hline & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} \end{array}$	4
RK 5	$\begin{array}{c\|ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & 0 & 0 & 0 & 0 \\ \frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & 0 & 0 & 0 & 0 \\ \frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & 0 & 0 & 0 \\ 1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & 0 & 0 \\ 1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\ \hline & \frac{743}{8043} & 0 & \frac{275}{618} & \frac{1321}{1920} & -\frac{4851}{12997} & \frac{121}{700} & -\frac{1}{40} \end{array}$	5

This pair consists of a fourth order RK-method with an embedded fifth order RK-method, which is used as reference method to estimate the error. Because the first six stages of the reference method are equal to the stage vector of the original method, it follows that

$$\bar{Y}_{n-1} = (Y_{n-1} \quad \bar{Y}_7)$$

and

$$\bar{\mathbf{x}}_n = \mathbf{x}_{n-1} + (\bar{\beta}^T \otimes I_d) h_n \bar{\mathbf{f}}(\bar{t}_{n-1}, Y_{n-1}) + \bar{\beta}_7 h_n \mathbf{f}(\bar{t}_7, \bar{Y}_7).$$

This means for the LTE that:

$$\hat{\delta}_n = \bar{\mathbf{x}}_n - \mathbf{x}_n = ((\bar{\beta} - \beta)^T \otimes I_d) h_n \bar{\mathbf{f}}(\bar{t}_{n-1}, Y_{n-1}) + \bar{\beta}_7 h_n \mathbf{f}(\bar{t}_7, \bar{Y}_7).$$

Introduce the vector σ by

$$\sigma = \left(\bar{\beta}_1 - \beta_1, \ldots, \bar{\beta}_6 - \beta_6, \bar{\beta}_7\right).$$

Then

$$\hat{\delta}_n = \sigma^T \otimes I_d h_n \bar{\mathbf{f}}(\bar{t}_{n-1}, \bar{Y}_{n-1}).$$

In the MATLAB-implementation, EPS-control is used, which means that $P = 4$.

The stepsize control of the ode45-routine is a combination of linear and non-linear control. If h_n is rejected, then

$$\begin{aligned} h_n^{(1)} &= (\tfrac{\epsilon}{\hat{r}_n})^{\frac{1}{5}} h_n, \\ h_n^{(2)} &= \begin{cases} \frac{1}{2} h_n & \text{if previous stepsize has also been rejected,} \\ \frac{1}{10} h_n & \text{if previous stepsize has been accepted and } h_n' < \frac{1}{10} h_n, \\ h_n^{(1)} & \text{otherwise,} \end{cases} \\ h_n &= \max\{h_{\min}, h_n^{(2)}\}. \end{aligned} \tag{5.1}$$

Here $\epsilon = \theta \text{TOL}$ where $\theta \approx 0.32$. If h_n is accepted, then

$$\begin{aligned} h_n^{(1)} &= (\tfrac{\epsilon}{\hat{r}_n})^{\frac{1}{3}} h_n, \\ h_n^{(2)} &= \begin{cases} 5h_n & \text{if } h_n^{(1)} > 5h_n, \\ h_n^{(1)} & \text{otherwise,} \end{cases} \\ h_{n+1} &= \max\{h_{\min}, \min\{h_{\max}, h_n^{(2)}\}\}. \end{aligned} \tag{5.2}$$

In the both cases, $h_n^{(1)}$ is computed with use of a linear deadbeat controller[5], while for the final timestep also non-linear controllers are used.

Because this method is an explicit method, there are no implicit equations, which have to be solved.

5.2.2 Test problem

PI-controllers and PC-controllers with $\theta = 0.3$

Consider the IVP for the the Van de Pol equation on [0, 100]:

$$\begin{gathered} \ddot{y} + 10(y^2 - 1)\dot{y} + y = \sin(100t), \\ y(0) = 0, \dot{y}(0) = 1. \end{gathered} \tag{5.3}$$

In Fig.(5.1), the solution has been shown.

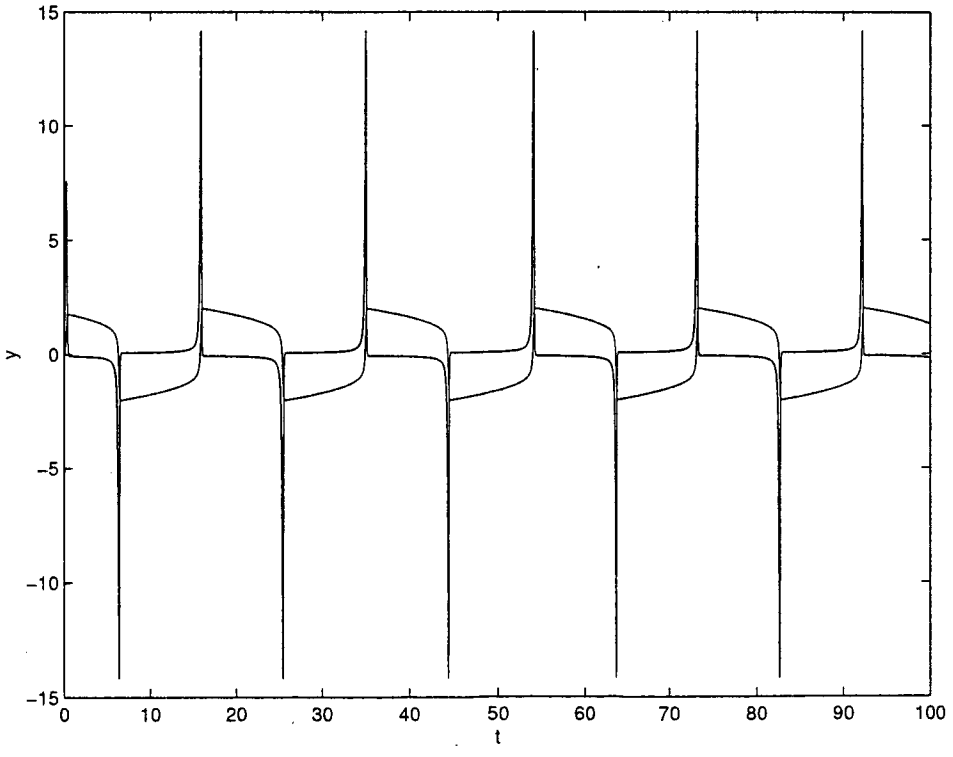


Figure 5.1: Solution of the Van de Pol equation.

[5]This controller has been described in paragraph 4.4.3.

Case	#timesteps	#rejections	#function evaluations
PI(1,0)	14392	1739	96787
PI(0.36,-0.16)	13685	148	82999
Combined PI	14614	410	90145
PC(1,1)	15519	2909	110569
PC(0.36,0.84)	14642	2105	100483

Table 5.2: Numerical results for the Runge Kutta method with $\theta = 0.3$.

Note that this problem is rather stiff, while the influence of the source term is of small order.

This ODE has been solved with the Runge Kutta method from the previous subsection with the tolerance level TOL = 10^{-5}. The safety factor was set to 0.3, because of the stiff behaviour. The next controllers[6] has been considered:

1. PI-control (deadbeat) with the control parameters $(Pk_I, Pk_P) = (1, 0)$ and the corresponding poles $(r_1, r_2) = (0, 0)$.
2. PI-control with the control parameters $(Pk_I, Pk_P) = (0.36, -0.16)$ and the corresponding poles $(r_1, r_2) = (0.4, 0.4)$.
3. Combined PI-controller with constraint validation.
4. PC-control (deadbeat) with the control parameters $(Pk_E, Pk_R) = (1, 1)$ and the corresponding poles $(r_1, r_2) = (0, 0)$.
5. PC-control with the control parameters $(Pk_E, Pk_R) = (0.36, 0.84)$ and the corresponding poles $(r_1, r_2) = (0.4, 0.4)$.

The combined[7] PI-controller uses also linear PI-control, if the previous stepsize has been rejected. The other controllers use the original controller (5.1) in that case. The PC-controllers are predictive PI-controllers, which can follow linear trends of the disturbance $\hat{\varphi}$.

In Tab.(5.2), the computational work has been compared. Note that for this example, the combined controller is not efficient, because θ is much smaller. Now, the smooth PI controller needs the smallest amount of work, while the PC-controllers are rather slowly.

For the cases 1,2,3 and 4, the stepsize- and error sequences have been shown. Because of the periodic stiff behaviour, all controllers fail in the stiff regions. Despite the small influence of the source term on the solution of the ODE, the stepsize and error sequences get oscillating behaviour. It is clear that the PI-controller of case 2 generates smoother results. From Fig.(5.5), it is visible that for the predictive PC-controller of case 4, the error follows better the reference level ϵ.

[6] In subsection 4.4.3, these controllers have been described.

[7] See subsection 4.4.3.

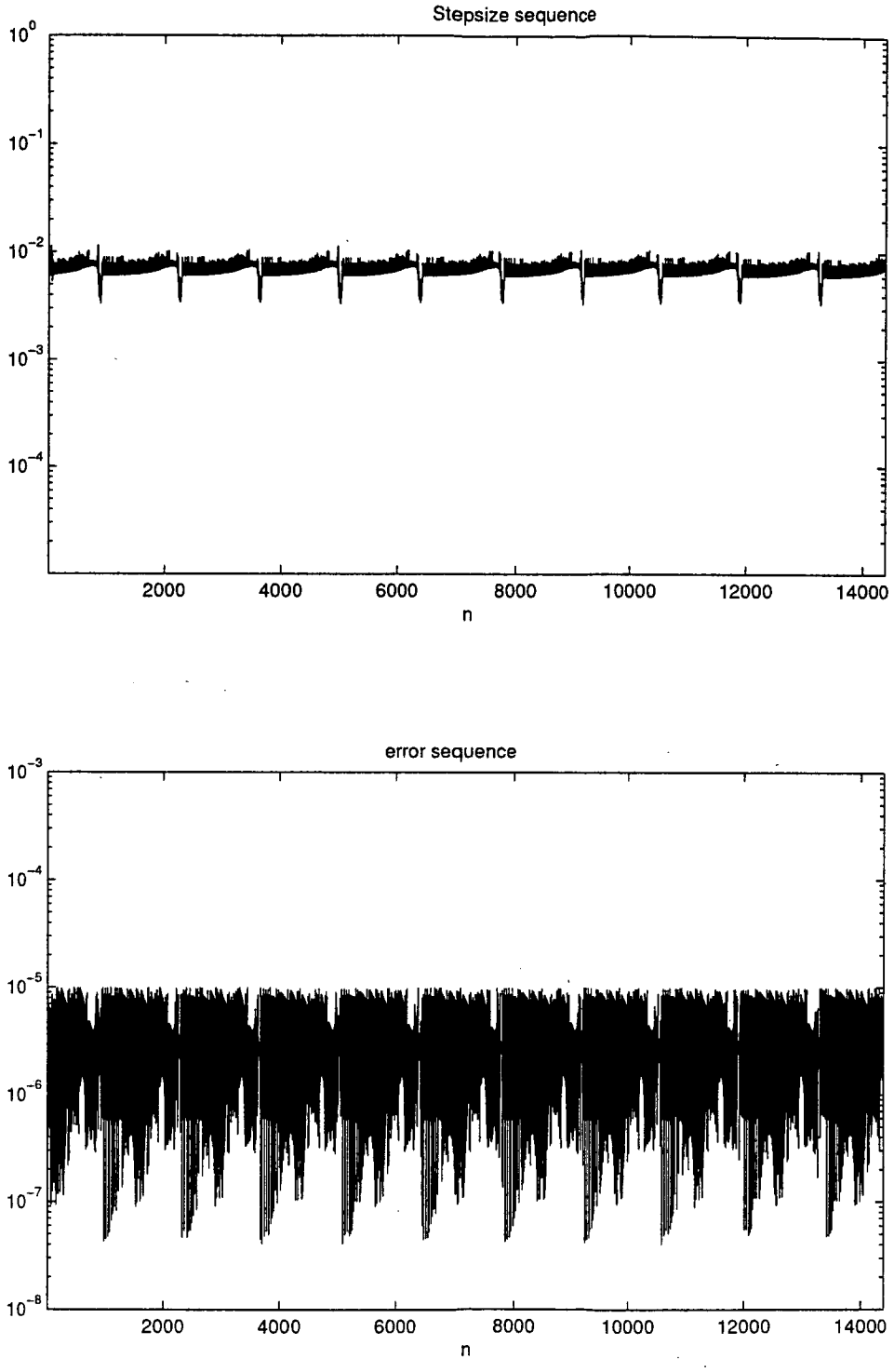


Figure 5.2: Results for the PI deadbeat controller.

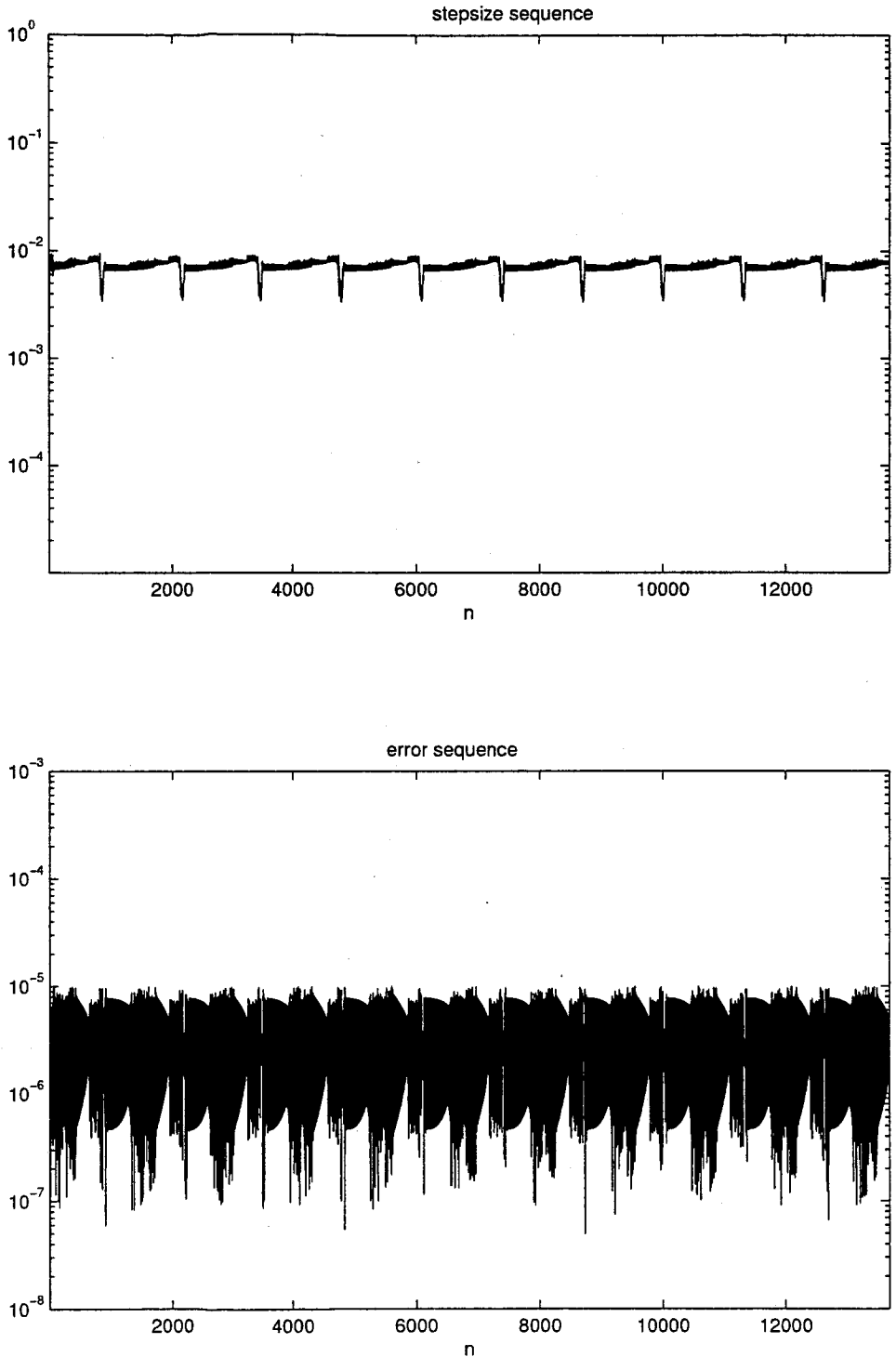
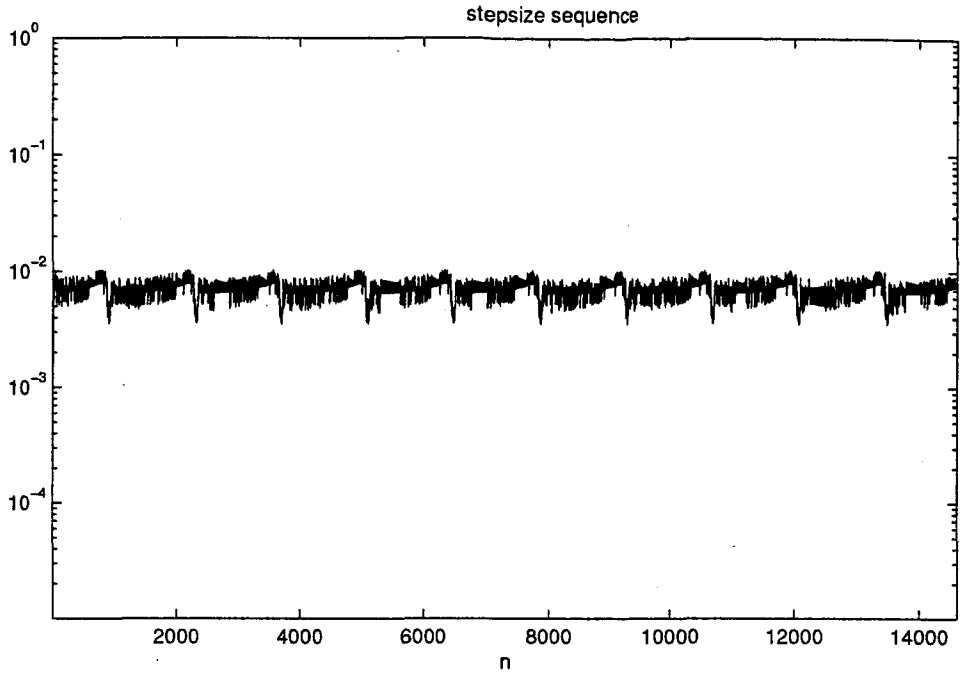


Figure 5.3: Results for the PI controller with $(Pk_I, Pk_P) = (0.36, -0.16)$.



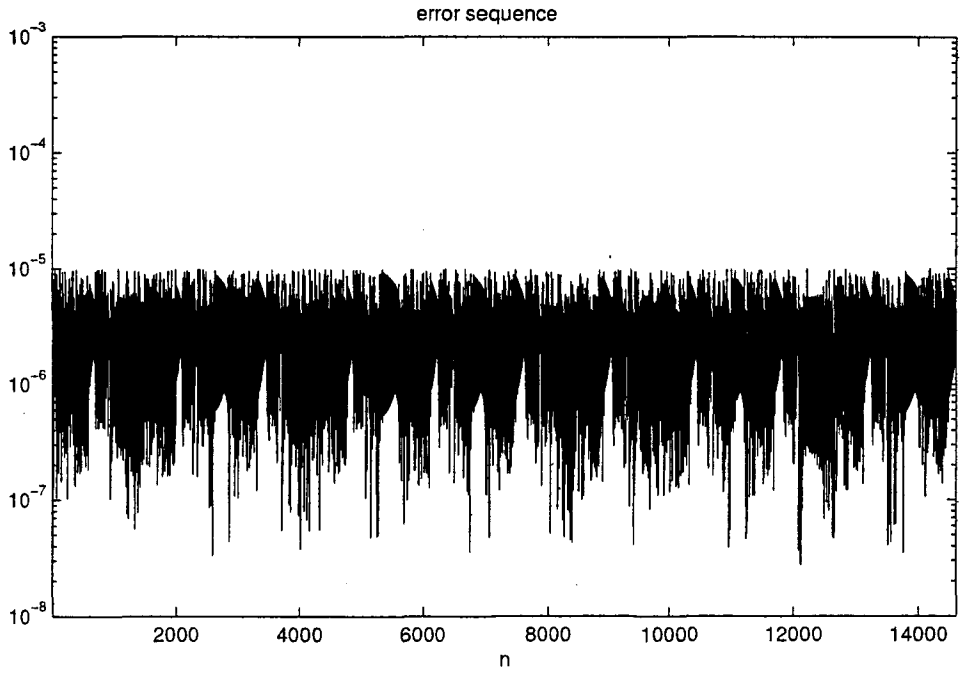


Figure 5.4: Results for the combined PI controller.

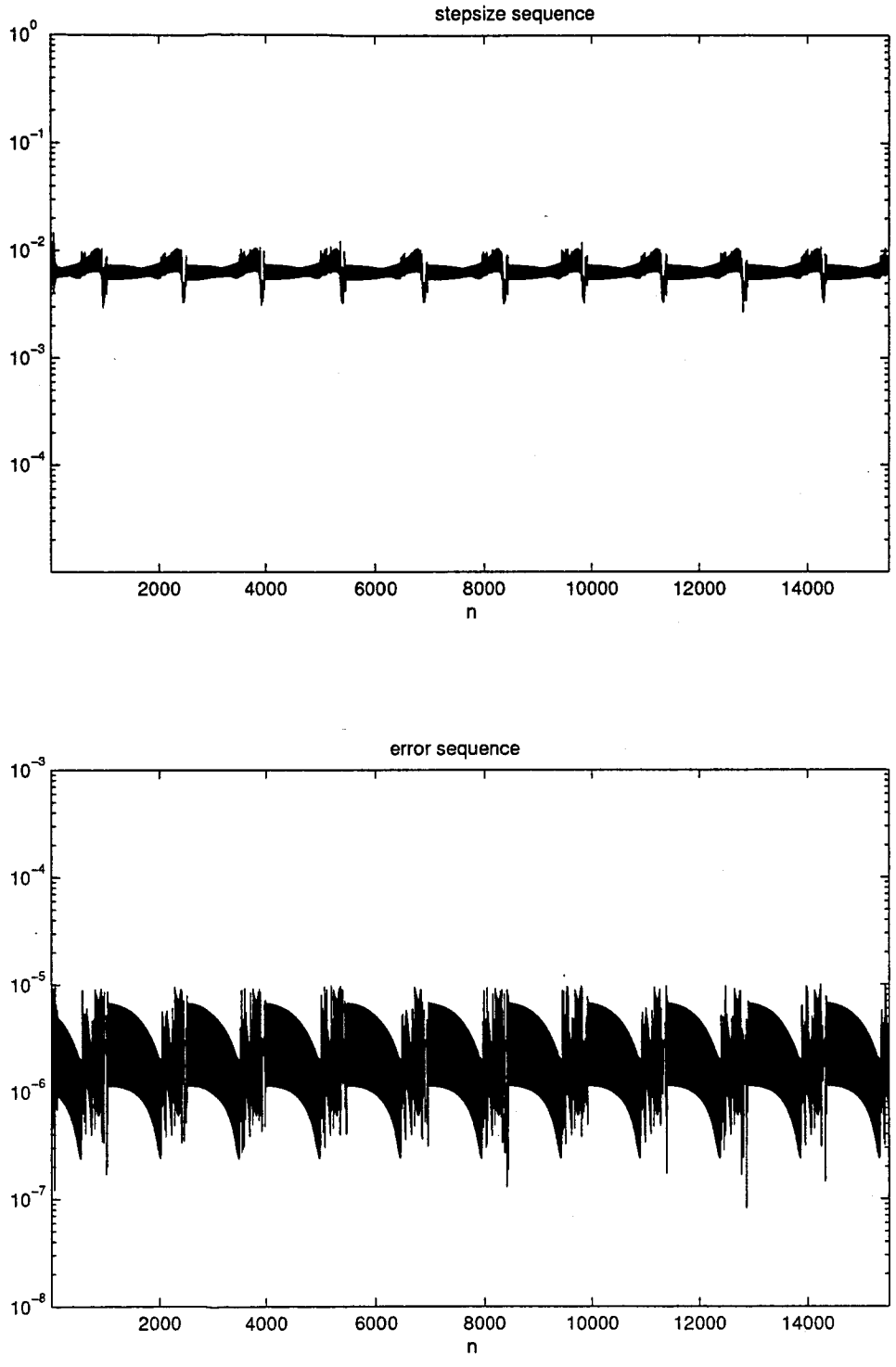


Figure 5.5: Results for the PC deadbeat controller.

To investigate the influence of the safety factor θ, in Tab.(5.3), the computational work has also been shown for $\theta = 0.8$. Note that all controllers need less stepsizes, because of the higher safety

Case	#timesteps	#rejections	#function evaluations
PI(1,0)	11943	5479	104533
PI(0.36,-0.16)	11977	12497	146845
Combined PI	13049	2465	93085
PC(1,1)	12711	4703	104485
PC(0.36,0.84)	12252	7572	118945

Table 5.3: Numerical results for the Runge Kutta method with $\theta = 0.8$.

factor. However, because also the numbers of rejctions have been increased, all controllers are less efficient for this example. Note that for this safety factor, case 3 results in few rejections, in comparison with the other cases. Clearly, the safety factor has a large influence on the dynamic behaviour of the controlled errors.

5.3 Experiments with the BDF-method in MATLAB

5.3.1 Description of the method

The MATLAB-routine "BDFcontrol.m" is an Predictor Corrector implementation of a BDF-method[8].

We have investigated the BDF-method with fixed order p with $p = 4$. For the BDF methods, EPUS control has been used, which implies that $P = p + 1$. For multistep methods with process model two[9], there is a difference between the linearized controllers and the nonlinear controllers[10]. Therefore, for all testcases with process model 2, also its nonlinear equivalent version will also be performed.

It appears that controllers with constraint validation[11] have better results, because other controllers do not converge, because of too small timesteps. Therefore, only the H^1_{100}-controller is tested without this property. In the previous section, it has become clear that also this value has a large impact. To reduce the number of test cases, only one value for θ has been investigated. For $\theta = 0.5$, the next test cases[12] will be considered for the MATLAB-routine "BDFcontrol.m".

[8] In subsections 3.4.2 and 3.4.3, the theoretical background of this method has been described.

[9] See subsection 4.4.2.

[10] See subsection 4.4.5.

[11] See page 88.

[12] The linear controllers have been notated like on page 88 by $H^D_{p_A p_F p_R}(r_1, \ldots, r_{N+M})$.

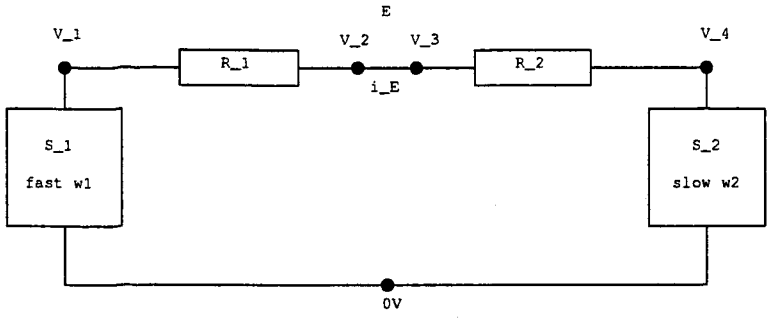


Figure 5.6: Hierarchical electrical circuit with two subcircuits S_1 and S_2 and additional elements R_1, R_2 and E.

1. $H^1_{100}(0)$ (deadbeat control)
2. $H^1_{100}(\frac{1}{2})$ (smooth I-control)
3. PI($\frac{1}{2}$, $\frac{1}{2}$) (smooth PI-control without oscillations)
4. PI($\frac{1}{2}$, $-\frac{1}{2}$) (PI-control with constraint validation)
5. Combined[13] PI-control
6. $H^2_{200}(0)$ (predictive deadbeat control)
7. $H^2_{110}(0)$ (deadbeat stepsize filter)
8. $H^2_{101}(0)$ (deadbeat error filter)
9. $H^2_{200}(\frac{1}{2})$ (predictive smooth control)
10. $H^2_{110}(\frac{1}{2})$ (smooth stepsize filter)
11. $H^2_{101}(\frac{1}{2})$ (smooth error filter)
12. $H^1_{100}(\frac{1}{2})$ (smooth I-control, model 2)
13. $H^2_{200}(\frac{1}{2})$ (predictive smooth control, model 2)
14. $H^2_{110}(\frac{1}{2})$ (smooth stepsize filter, model 2)
15. $H^2_{101}(\frac{1}{2})$ (smooth error filter, model 2)
16. $H^1_{100}(\frac{1}{2})$ (nonlinear smooth I-control, model 2)
17. $H^2_{200}(\frac{1}{2})$ (nonlinear predictive smooth control, model 2)
18. $H^2_{110}(\frac{1}{2})$ (nonlinear smooth stepsize filter, model 2)
19. $H^2_{101}(\frac{1}{2})$ (nonlinear smooth error filter, model 2)

For all testcases, no additional nonlinear or logic control actions have been applied, because they can disturb the designed behaviour. The combined PI-controller uses also PI-control after rejections. The other controllers use in that case the simple control law:

$$h_n = 0.5h_n. \tag{5.4}$$

5.3.2 Linear electrical circuit

Derivation of the DAE

Structure of the circuit Consider the hierarchical circuit, shown in Fig.(5.6). It consists of subcircuits S_1 and S_2, with terminal variables V_1 and V_2. The elements R_1, R_2 and short E are used to connect these subcircuits, but are not part of the original circuit. In Fig.(5.7), the structure of both subcircuits is shown. The functions $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{j}_1$ and $\mathbf{j}_2$ are the contributions of S_1 and S_2 to $\mathbf{q}$ and $\mathbf{j}$. For

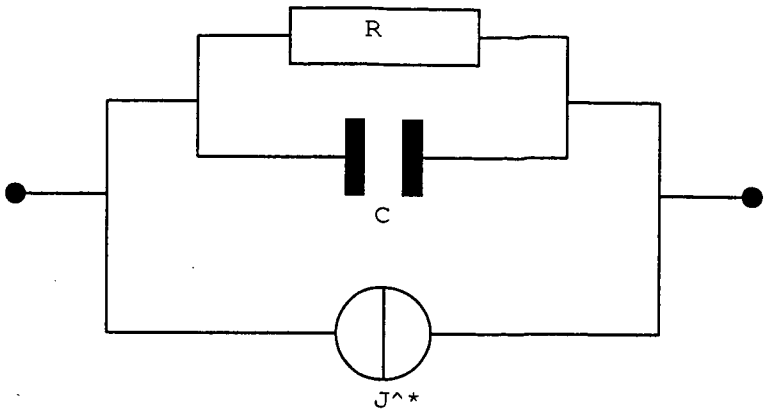


Figure 5.7: Structure of the subcircuits.

Parameters	Value
T	0.08
ω_1	$\frac{5}{2}\pi \cdot 10^3$
ω_2	$\frac{1}{4}\pi \cdot 10^3$
R	10
C	10^{-3}
R_1	1
R_2	1

Table 5.4: Table with the used parameters, with $\omega_1 = 10\omega_2$.

this example, we have

$$\begin{aligned} \mathbf{q}_1(t,x) &= -Cx, \quad \mathbf{j}_1(t,x) = -\tfrac{1}{R}x + \sin(\omega_1 t), \\ \mathbf{q}_2(t,x) &= -Cx, \quad \mathbf{j}_2(t,x) = -\tfrac{1}{R}x + \sin(\omega_2 t). \end{aligned}$$

Note that the frequency of the current source is the only difference between S_1 and S_2. In Tab.(5.4), the used parameters are shown.

Mathematical model This circuit is determined by the state variables $\mathbf{x} = (V_1, V_2, i_E, V_3, V_4)$, where the corresponding equations at the flat parent circuit level are given by:

$$\begin{aligned} \tfrac{d}{dt}\mathbf{q}_1(t, V_1) + \mathbf{j}_1(t, V_1) &= \tfrac{1}{R_1}(V_1 - V_2), \\ \tfrac{1}{R_1}(V_1 - V_2) &= i_E, \\ V_2 - V_3 &= 0, \\ i_E &= \tfrac{1}{R_2}(V_3 - V_4), \\ \tfrac{d}{dt}\mathbf{q}_2(t, V_4) + \mathbf{j}_2(t, V_4) &= \tfrac{1}{R_2}(V_4 - V_3). \end{aligned}$$

Define the functions $\mathbf{q}$ and $\mathbf{j}$ with

$$\mathbf{q}(t,\mathbf{x}) = \begin{pmatrix} \mathbf{q}_1(t,V_1) \\ 0 \\ 0 \\ 0 \\ \mathbf{q}_2(t,V_4) \end{pmatrix}, \quad \mathbf{j}(t,\mathbf{x}) = \begin{pmatrix} \mathbf{j}_1(t,V_1) - \frac{1}{R_1}(V_1 - V_2) \\ \frac{1}{R_1}(V_1 - V_2) - i_E \\ V_2 - V_3 \\ i_E - \frac{1}{R_2}(V_3 - V_4) \\ \mathbf{j}_2(t,V_4) - \frac{1}{R_2}(V_4 - V_3) \end{pmatrix}.$$

Then, the circuit is modelled by the next DAE:

$$\frac{d}{dt}\mathbf{q}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}.$$

Eliminating V_2, V_3 and i_E shows that $\bar{\mathbf{x}} = (V_1, V_4)$ satsifies the ODE:

$$\frac{d\bar{\mathbf{x}}}{dt} = \begin{pmatrix} -\alpha & \beta \\ \beta & -\alpha \end{pmatrix}\bar{\mathbf{x}} + \frac{1}{C}\begin{pmatrix} \sin(\omega_1 t) \\ \sin(\omega_2 t) \end{pmatrix},$$

where

$$\begin{cases} \alpha &= \frac{1}{RC} + \frac{1}{C(R_1+R_2)}, \\ \beta &= \frac{1}{C(R_1+R_2)}. \end{cases}$$

Now, V_2, V_3 and i_E are determined by $\bar{\mathbf{x}}$:

$$\begin{aligned} V_2 &= V_3 = \tfrac{R_2}{R_1+R_2}V_1 + \tfrac{R_1}{R_1+R_2}V_4, \\ i_E &= \tfrac{1}{R_1+R_2}V_1 - \tfrac{1}{R_1+R_2}V_4. \end{aligned}$$

The initial solution is the steady-state solution, so $\mathbf{x}(0) = \mathbf{0}$.

Exact solution Because this circuit obeys a linear ODE, its exact solution can be derived. This is useful, because then for the numerical experiments also the global errors can be computed. It follows that:

$$\begin{aligned} V_1 &= a_1\cos(\omega_1 t) + a_2\cos(\omega_2 t) + b_1\sin(\omega_1 t) + b_2\sin(\omega_2 t) - \\ &\quad (a_1+a_2)e^{-\alpha t}\cosh(\beta t) - (c_1+c_2)e^{-\alpha t}\sinh(\beta t), \\ V_4 &= c_1\cos(\omega_1 t) + c_2\cos(\omega_2 t) + d_1\sin(\omega_1 t) + d_2\sin(\omega_2 t) - \\ &\quad (c_1+c_2)e^{-\alpha t}\cosh(\beta t) - (a_1+a_2)e^{-\alpha t}\sinh(\beta t). \end{aligned}$$

The parameters are shown in Tab.(5.5). In Fig.(5.8), the exact solution for all unknowns is shown, where the values in Tab.(5.4) are used. Note that the solution is affected by the high-frequent and the low-frequent current sources.

Numerical results

This circuit has been solved with BDFcontrol with the tolerance level TOL = 10^{-4}. Because the testcircuit is linear, only one Newton iteration per step is used.

In Tab.(5.6), the numerical results for the several controllers have been compared. The number of accepted stepsizes, the rejections and the total number of Newton iterations have been shown. Besides the computational work, which is proportional to the number of Newton iterations, also the smoothness of the results has been quantified[14]. A sequence is smooth, if the smoothness function tends to

[14] See equation (4.12) in subsection 4.2.5.

Parameters	Value	Example
α	$\frac{1}{RC} + \frac{1}{C(R_1+R_2)}$	600
β	$\frac{1}{C(R_1+R_2)}$	500
a_1	$\frac{-\omega_1(\omega_1^2+\alpha^2+\beta^2)}{C(\omega_1^2+(\alpha-\beta)^2)(\omega_1^2+(\alpha+\beta)^2)}$	-0.1261
a_2	$\frac{-2\omega_2\alpha\beta}{C(\omega_2^2+(\alpha-\beta)^2)(\omega_2^2+(\alpha+\beta)^2)}$	-0.4115
b_1	$\frac{\alpha(\omega_1^2+\alpha^2-\beta^2)}{C(\omega_1^2+(\alpha-\beta)^2)(\omega_1^2+(\alpha+\beta)^2)}$	0.0096
b_2	$\frac{-\beta(\omega_2^2-\alpha^2+\beta^2)}{C(\omega_2^2+(\alpha-\beta)^2)(\omega_2^2+(\alpha+\beta)^2)}$	-0.2213
c_1	$\frac{-2\omega_1\alpha\beta}{C(\omega_1^2+(\alpha-\beta)^2)(\omega_1^2+(\alpha+\beta)^2)}$	-0.0012
c_2	$\frac{-\omega_2(\omega_2^2+\alpha^2+\beta^2)}{C(\omega_2^2+(\alpha-\beta)^2)(\omega_2^2+(\alpha+\beta)^2)}$	-0.8414
d_1	$\frac{-\beta(\omega_1^2-\alpha^2+\beta^2)}{C(\omega_1^2+(\alpha-\beta)^2)(\omega_1^2+(\alpha+\beta)^2)}$	-0.0079
d_2	$\frac{\alpha(\omega_2^2+\alpha^2-\beta^2)}{C(\omega_2^2+(\alpha-\beta)^2)(\omega_2^2+(\alpha+\beta)^2)}$	0.3808

Table 5.5: The parameters of the exact solution, using the values from Tab.(5.4).

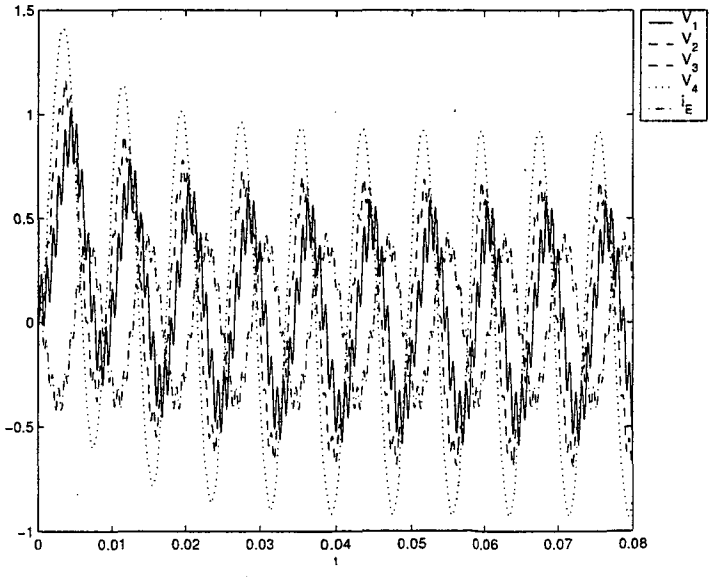


Figure 5.8: Exact solution for the linear electrical circuit in Fig.(5.6).

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1	517	35	551	1.03	0.18
2	484	0	483	0.95	0.08
3	491	0	490	0.72	0.06
4	520	38	557	1.07	0.21
5	512	49	560	1.08	0.18
6	605	182	786	1.21	0.35
7	483	0	482	0.88	0.07
8	502	15	516	0.90	0.28
9	608	120	727	1.05	0.33
10	486	0	485	0.87	0.07
11	487	0	486	0.84	0.26
12	550	63	612	0.77	0.20
13	602	151	752	1.10	0.44
14	486	0	485	1.00	0.14
15	551	66	616	0.96	0.23
16	522	55	576	0.94	0.25
17	590	198	787	1.11	0.37
18	488	3	490	1.01	0.16
19	522	14	535	0.86	0.30

Table 5.6: Numerical results for the linear circuit in Fig.(5.6).

zero. Clearly, deadbeat control is not very good in this case. It needs more computational work, while also the stepsize and error sequences are not smooth. Simple I-control (case 2) performs very well, but produces not a very smooth error sequence. However, the smooth PI-controller without oscillations (case 3) appears to perform very well, because it combines a small number Newton iterations with smooth results. Also the first order stepsize filters and error filters (case 7, 10, 11) are very efficient. Also cases 14 and 18 are stepsize filters, which perform very well. It appears that the stepsize controllers are more efficient than the error filters. The predictive controllers (cases 6, 9, 13, 17) have many rejections, which is probably caused by problems with linear extrapolation.

Note that for all controllers, the stepsize sequence is much smoother than the error sequence.

From Tab.(5.7), it becomes clear that the smooth controllers also result in smaller global errors. Clearly, the PI($\frac{1}{2}$, $\frac{1}{2}$)-controller (case 3), generates the smoothest error and stepsize sequences. Then, the smoothness is equal to 0.72 and 0.06 respectively, which is smaller than for the other cases. Note that for all testcases, the stepsize sequences are much smoother than the error sequences.

To compare the results of the smoothest controller (case three) with the classical deadbeat controller (case one), in Figures (5.9), (5.10), (5.11) and (5.12), the stepsize and error sequences have been plotted. Clearly, testcase three generates much smoother results than testcase one. The stepsize sequence for the smooth PI-controller is still constant for this ODE. The smooth PI-controller has irregular initial behaviour of the error sequence.

Case	V_1	V_2	V_3	V_4	i_E
1	1.21e-01	9.02e-02	9.02e-02	6.44e-02	3.22e-02
2	5.55e-02	2.99e-02	2.99e-02	1.28e-02	2.86e-02
3	6.15e-02	3.30e-02	3.30e-02	1.04e-02	2.89e-02
4	1.06e-01	7.71e-02	7.71e-02	5.20e-02	3.06e-02
5	7.91e-02	4.28e-02	4.28e-02	2.27e-02	3.78e-02
6	1.22e-01	8.51e-02	8.51e-02	5.77e-02	4.06e-02
7	6.27e-02	3.60e-02	3.60e-02	1.70e-02	3.54e-02
8	1.00e-01	6.43e-02	6.43e-02	4.36e-02	4.68e-02
9	1.13e-01	7.68e-02	7.68e-02	5.09e-02	4.25e-02
10	5.66e-02	3.03e-02	3.03e-02	1.23e-02	2.65e-02
11	1.06e-01	7.44e-02	7.44e-02	4.87e-02	3.24e-02
12	7.03e-02	4.05e-02	4.05e-02	1.56e-02	2.97e-02
13	9.61e-02	6.56e-02	6.56e-02	4.56e-02	3.48e-02
14	5.92e-02	3.42e-02	3.42e-02	1.66e-02	2.63e-02
15	5.81e-02	3.35e-02	3.35e-02	2.07e-02	2.69e-02
16	2.57e-01	2.08e-01	2.08e-01	1.66e-01	5.33e-02
17	1.04e-01	7.62e-02	7.62e-02	5.18e-02	5.06e-02
18	9.43e-02	5.47e-02	5.47e-02	2.96e-02	3.96e-02
19	8.17e-02	5.08e-02	5.08e-02	3.05e-02	3.51e-02

Table 5.7: Global errors for the linear electrical circuit in Fig.(5.6).

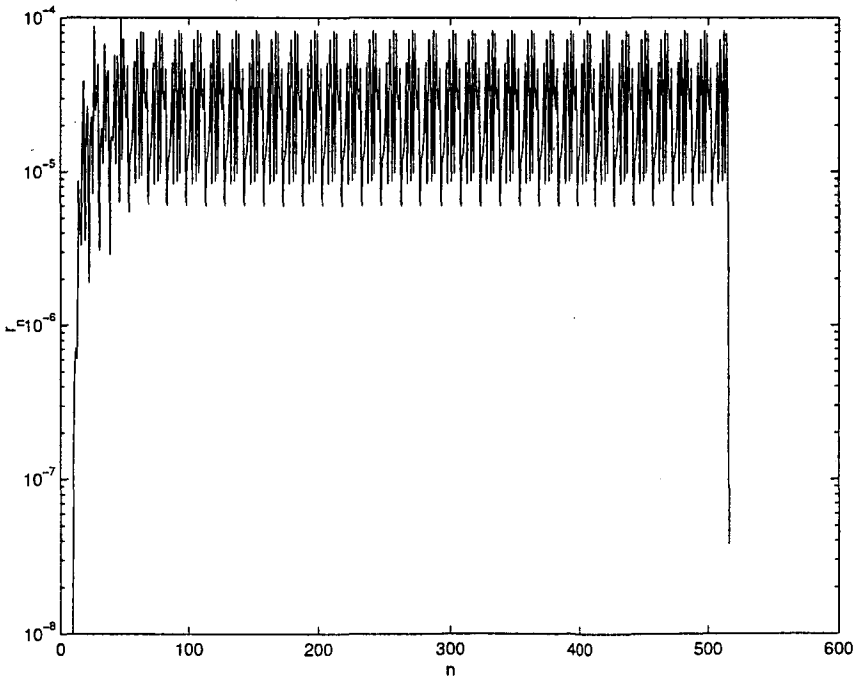


Figure 5.9: Error sequence for the linear circuit with the deadbeat controller (case 1).

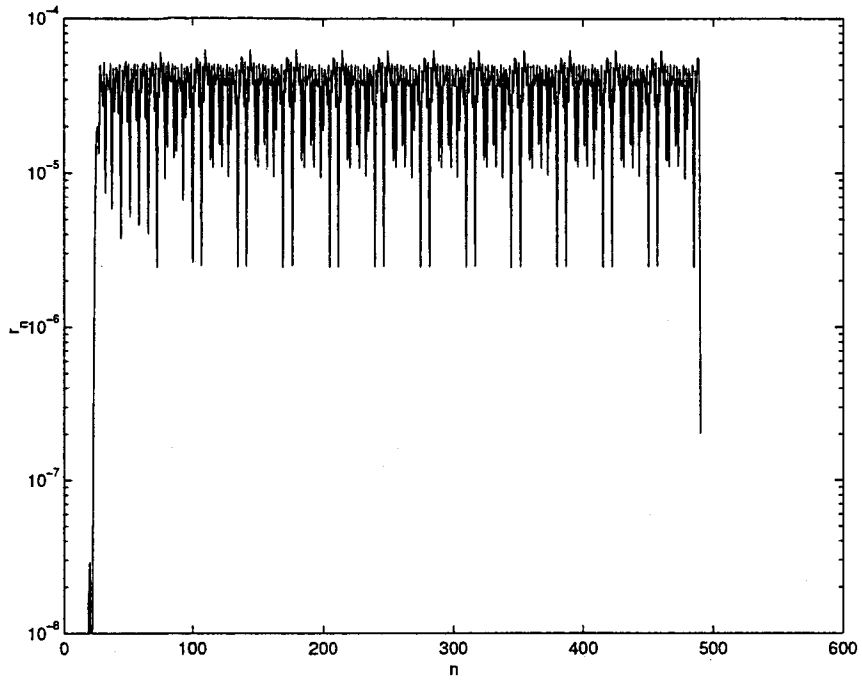


Figure 5.10: Error sequence for the linear circuit with smooth PI control without oscillations (case 3).

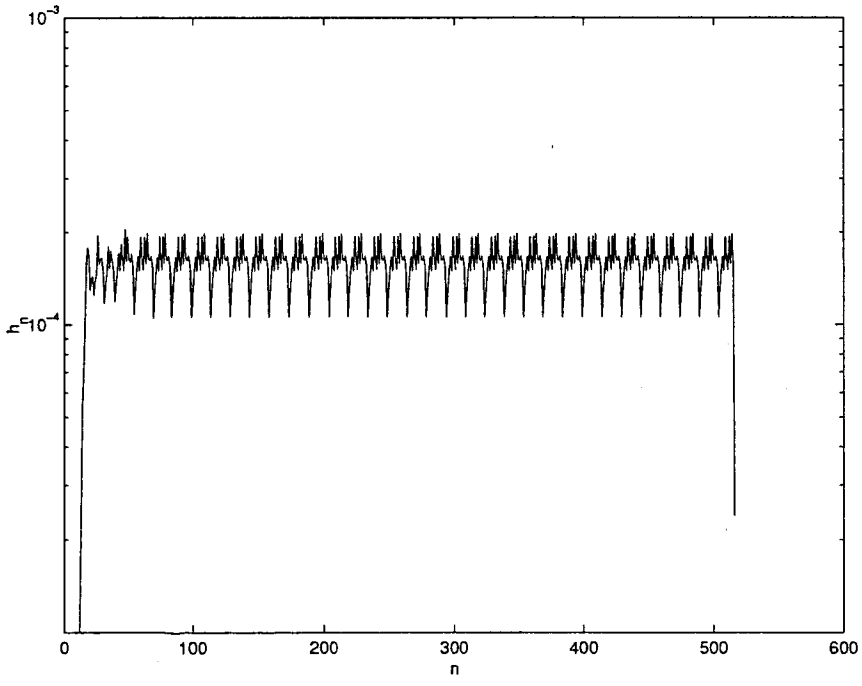


Figure 5.11: Stepsize sequence for the linear circuit with the deadbeat controller (case 1).

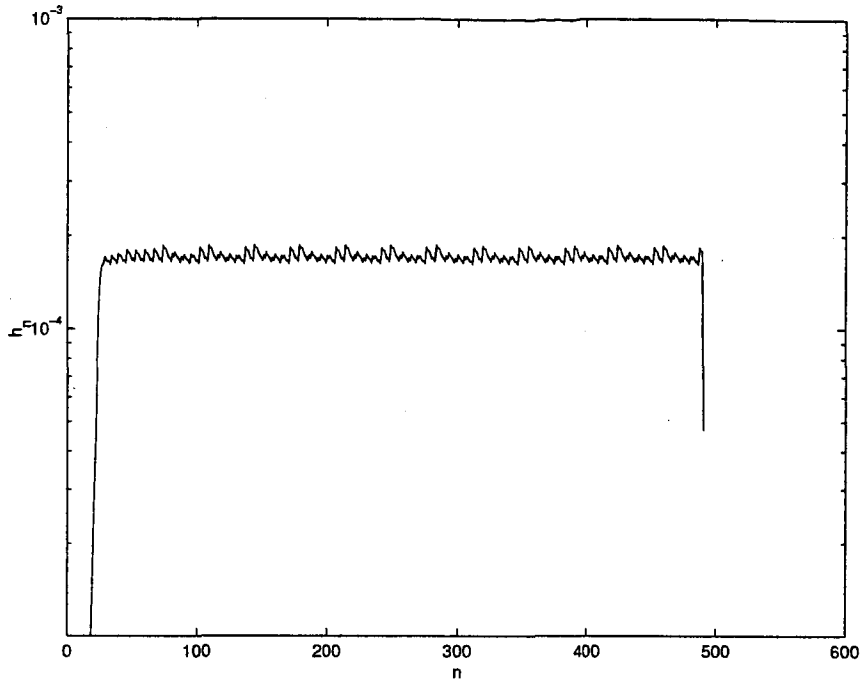


Figure 5.12: Stepsize sequence for the linear circuit with smooth PI control without oscillations (case 3).

5.3.3 Van de Pol equation

Derivation of the DAE

Circuit Consider the small electrical circuit in Fig.(5.13):

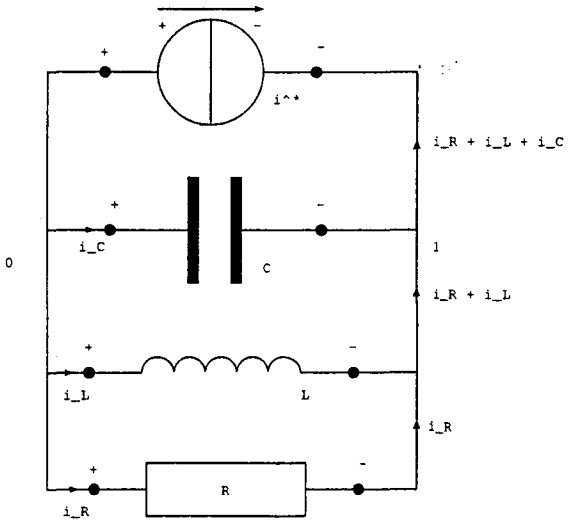


Figure 5.13: Van de Pol oscillator.

Parameters	Value
T	100
C	10^{-3}
L	10^3
R	$\frac{10^2}{3}$
i^*	1

Table 5.8: Table with the used parameters

with the next constitutive relations:

$$\begin{aligned} i_J &= i^*, \\ i_C &= C\dot{v}_C, \\ v_L &= L\tfrac{d}{dt}i_L, \\ i_R &= \tfrac{v_R}{R}(\tfrac{v_R^2}{3} - 1). \end{aligned}$$

In Tab.(5.8), the used parameters have been shown.

Mathematical model For this circuit, it follows from the Kirchhoff laws that $i^* + i_C + i_L + i_R = 0$ and $V_1 = -v_C = -v_L = -v_R$. The circuit is determined by the state variables $\mathbf{x} = (V_1, i_L)$, with the corresponding equations:

$$\begin{gathered} i^* - C\dot{V}_1 + i_L - \tfrac{V_1}{R}(\tfrac{V_1^2}{3} - 1) = 0, \\ L\tfrac{d}{dt}i_L + V_1 = 0. \end{gathered}$$

Define the functions $\mathbf{q}$ and $\mathbf{j}$ with

$$\mathbf{q}(t, \mathbf{x}) = \begin{pmatrix} -CV_1 \\ -Li_L \end{pmatrix}, \quad \mathbf{j}(t, \mathbf{x}) = \begin{pmatrix} i^* + i_L - \frac{V_1}{R}(\frac{V_1^2}{3} - 1) \\ -V_1 \end{pmatrix}.$$

Then, the circuit is modelled by the next nonlinear DAE:

$$\frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}.$$

The circuit can also be described by the next ODE, where $v = V_1$.

$$C\ddot{v} + \frac{1}{R}(v^2 - 1)\dot{v} + \frac{1}{L}v = \frac{di^*}{dt}.$$

If $LC = 1$, $\frac{di^*}{dt} = 0$ and $RC = \frac{1}{\sigma}$, this ODE describes the so-called Van de Pol oscillator:

$$\ddot{v} + \sigma(v^2 - 1)\dot{v} + v = 0.$$

If R, C and L satisfy the values of Tab.(5.8), it follows that $\sigma = 30$.

For the initial solution, it holds that $\mathbf{x}(0) = (0, 1)$.

In Fig.(5.14), an approximated solution has been shown. Clearly, this circuit has a rather stiff behaviour.

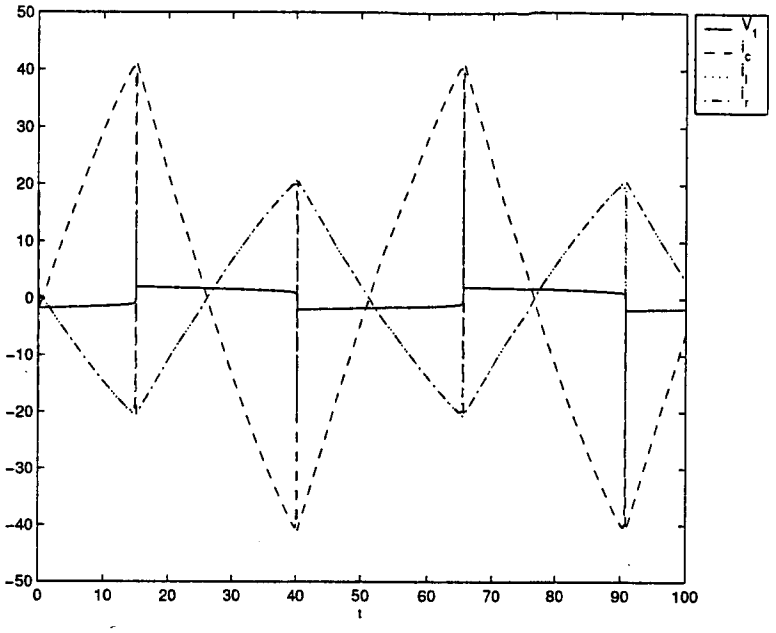


Figure 5.14: The approximated solution for the circuit for the Van de Pol equation.

Numerical results

Also this nonlinear circuit has been solved with BDFcontrol with the tolerance level TOL = 10^{-4}. Now, more than one Newton iteration per step is used. In Tab.(5.9), the numerical results have been compared for all tested controllers. Now the combined PI controller (case 5) results into the smoothest results. It appears also to be rather efficient compared to the other cases. Only the PI-controller with constraint validation needs less Newton iterations. Besides these controllers, also simple I-control (case 2) and the deadbeat stepsize filter (case 7) are more efficient than deadbeat control (case 1).

In Figures (5.15) and (5.16), the error sequences are shown for testcases 1 and 5. Because of the numerical results in Tab.(5.9), it is expected that the error sequence of case 5 with smoothness 0.39 would be smoother than the error sequence of case 1, with smoothness 0.54. However, from Figures (5.15) and (5.16), this smoothness difference is not visible. This means that it is not always possible to quantify the smoothness of the sequences.

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1	462	31	923	0.54	0.21
2	482	40	919	0.63	0.24
3	537	46	950	0.73	0.31
4	458	26	882	0.52	0.21
5	449	36	883	0.39	0.22
6	715	180	1477	0.89	0.22
7	460	44	905	0.65	0.26
8	510	93	1049	0.73	0.36
9	697	146	1366	0.91	0.28
10	479	59	947	0.74	0.31
11	488	75	968	0.76	0.34
12	466	51	932	0.68	0.27
13	676	119	1326	0.88	0.29
14	472	63	943	0.71	0.31
15	614	175	1297	0.88	0.36
16	495	130	1140	0.58	0.22
17	603	90	1214	0.86	0.25
18	507	163	1212	0.68	0.21
19	674	273	1611	0.73	0.31

Table 5.9: Numerical results for the Van de Pol equation.

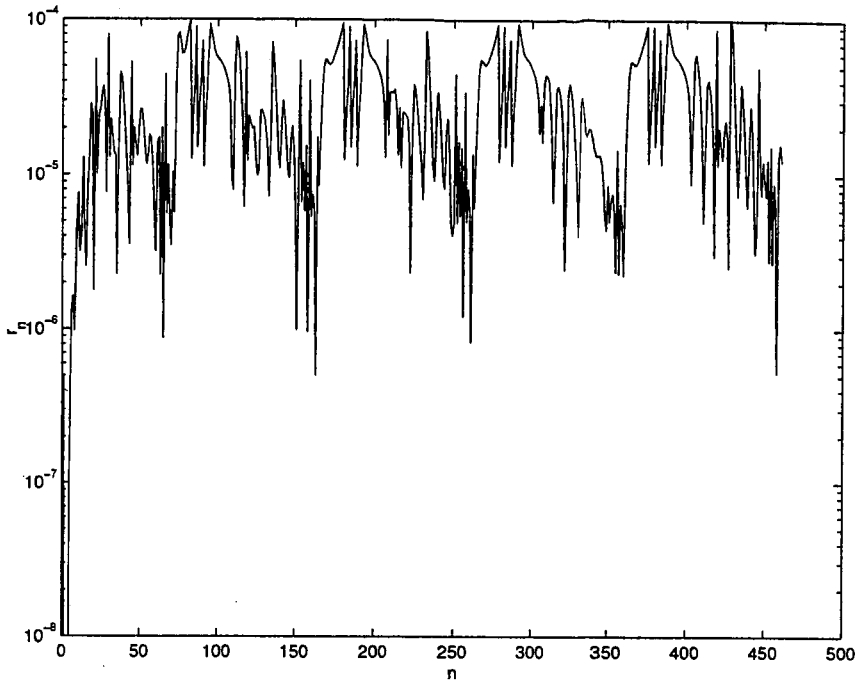


Figure 5.15: Error sequence for the Van de Pol oscillator with the deadbeat controller (case 1).

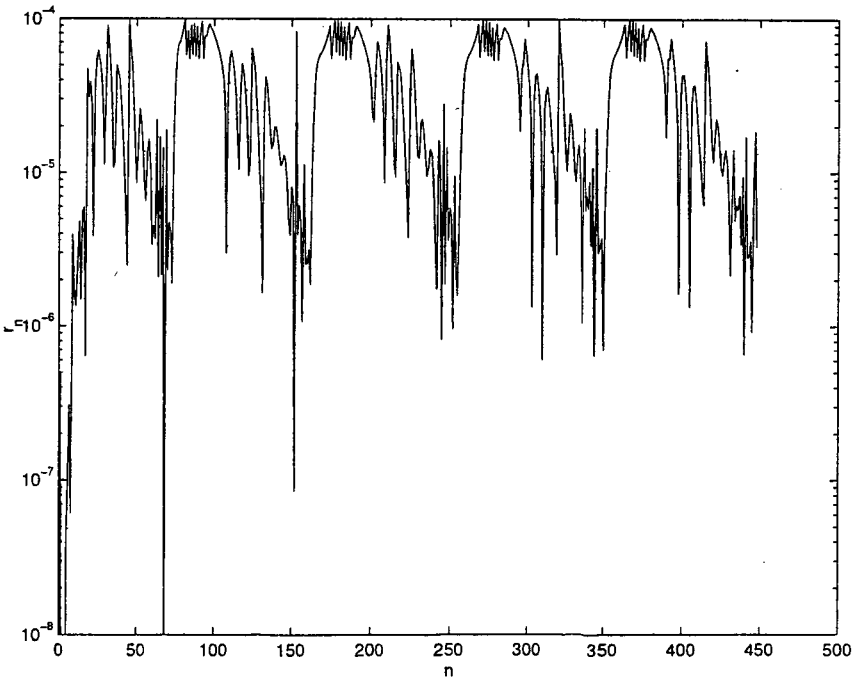


Figure 5.16: Error sequence for the Van de Pol oscillator with the combined PI-controller (case 5).

5.4 Experiments with the circuit simulator Pstar

5.4.1 Description of the method

From the previous subsection, it appears that digital linear control performs well, compared to deadbeat control. However, it has only be tested on two small examples. To compare the results for larger testcircuits, the tested controllers have also to be used for the circuit simulator Pstar[15]. With use of the Pstar-MATLAB-interface[16], it is possible to use MATLAB-functions in Pstar. This interface has been used to test the results of adaptive stepsize controllers for Pstar itself itself. Below, all tested electrical circuits are shown.

1. Linear circuit
2. Van de Pol oscillator
3. perf_mos7_qubic_6953 (from the albuquerque group)
4. bim2 (from the performance group)
5. sram (from the performance group)
6. perf_mos9_c100_7342a (from the roza group)

The testcircuit sram has also be used by C.Bomhof in his PhD-thesis [4].

These testcircuits will be simulated by means of Pstar, while several stepsize controllers are used.

It appears that order control is very important for the behaviour of the controllers. If the order changes too much, the process models are not valid, which can disturb the results. Also, it appears that for higher orders, the Newton process converges less easy. Therefore, for some testcases, the order is bounded at maximum order $p = 3$. Furthermore, the order can only decrease to one, if there are discontinuities. As safety factor, $\theta_n = f_{release}$ has been chosen[17]. Because deadzones can disturb the behaviour of the controller, they have been removed[18]. For the two small testcircuits, the following testcases[19] will be considered:

[15] In appendix A, the original time stepping algorithm of Pstar has been described.

[16] In appendix B, the Pstar-MATLAB-interface and its application have been described in more detail.

[17] This releasefactor of Pstar is described in appendix A.

[18] To remove the deadzones in Pstar, some lines must be changed in the source code.

[19] The linear controllers have been notated like on page 88 by $H^D_{p_A p_F p_R}(r_1, \ldots, r_{N+M})$.

1. Pstar stepsize mechanism only with changed order control
2. $H^1_{100}(0)$ (deadbeat control)
3. $H^1_{100}(\frac{1}{2})$ (smooth I-control)
4. PI($\frac{1}{2}, \frac{1}{2}$) (smooth PI-control without oscillations)
5. PI($\frac{1}{2}, -\frac{1}{2}$) (PI-control with constraint validation)
6. $H^2_{200}(0)$ (predictive deadbeat control)
7. $H^2_{110}(0)$ (deadbeat stepsize filter)
8. $H^2_{101}(0)$ (deadbeat error filter)
9. $H^2_{200}(\frac{1}{2})$ (predictive smooth control)
10. $H^2_{110}(\frac{1}{2})$ (smooth stepsize filter)
11. $H^2_{101}(\frac{1}{2})$ (smooth error filter)
12. $H^1_{100}(\frac{1}{2})$ (smooth I-control, model 2)
13. $H^2_{200}(\frac{1}{2})$ (predictive smooth control, model 2)
14. $H^2_{110}(\frac{1}{2})$ (smooth stepsize filter, model 2)
15. $H^2_{101}(\frac{1}{2})$ (smooth error filter, model 2)
16. $H^1_{100}(\frac{1}{2})$ (nonlinear smooth I-control, model 2)
17. $H^2_{200}(\frac{1}{2})$ (nonlinear predictive smooth control, model 2)
18. $H^2_{110}(\frac{1}{2})$ (nonlinear smooth stepsize filter, model 2)
19. $H^2_{101}(\frac{1}{2})$ (nonlinear smooth error filter, model 2)

Because for the process model one, it is not necessary to use linearization techniques to derive an approximated linear model, the linear controller is equal to its nonlinear version[20].

Because Pstar itself uses the maximum order five and variable order, also some cases are investigated with the original order controller of Pstar. Now, the original settings are used, such as deadzones and variable order. Again, $\theta_n = f_{release}$, except for case one. In that case, the original release factor of Pstar is used with $\theta_n = 0.5 f_{release}$. Because of the variable order, these cases need less computational work. However, sometimes the other cases are more efficient, because of nonspecified discontinuities. After too many rejections, Pstar decides that a nonspecified discontinuity occurs. In that case, Pstar starts again with the Euler Backward method with order one. It appears that these nonspecified discontinuities occur more frequently for higher order methods. This can destroy the higher efficiency of these higher-order methods, because of the many restarts. The following testcases will be considered:

1v. Ordinary Pstar stepsize mechanism
2v. $H^1_{100}(\frac{1}{2})$ (smooth I-control)
3v. PI($\frac{1}{2}, -\frac{1}{2}$) (PI-control with constraint validation)
4v. $H^2_{110}(0)$ (deadbeat stepsize filter)
5v. $H^1_{100}(\frac{1}{2})$ (smooth I-control, model 2)
6v. $H^2_{110}(\frac{1}{2})$ (smooth stepsize filter, model 2)
7v. $H^1_{100}(\frac{1}{2})$ (nonlinear smooth I-control, model 2)

5.4.2 Linear electrical circuit

First, the testcases with fixed order have been tested. In Tab.(5.10), the numerical results are shown. Because of the dynamic release factor, Pstar needs less computational work than the MATLAB-implementation, but the local errors are also larger. Note that case four is the smoothest version, but it needs quite a lot of Newton iterations. The cases 7, 10, 11, 12 and 14 have much better perfor-

[20] See subsection 4.4.5.

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1	1258	222	1480	1.17	0.57
2	1106	243	1349	1.07	0.43
3	1277	198	1475	0.82	0.38
4	1737	0	1737	0.36	0.06
5	1261	356	1617	0.92	0.44
6	1038	623	1661	1.14	0.87
7	1038	33	1071	0.61	0.18
8	1141	1	1142	0.82	0.24
9	990	609	1599	1.14	0.83
10	1053	0	1053	0.57	0.10
11	1198	0	1198	0.75	0.22
12	1015	0	1015	1.01	0.32
13	1142	684	1826	1.07	0.80
14	1136	0	1136	0.69	0.18
15	1440	0	1440	1.05	0.39
16	939	753	1692	1.12	0.75
17	4421	29	4450	1.07	0.85
18	981	811	1792	1.03	0.82
19	524	486	1010	1.09	0.88

Table 5.10: Numerical results for the linear electrical circuit with fixed order.

mance, compared to the other testcases. This is caused by the filter properties or the more accurate process model. Although case 19 seems to be very efficient, from Tab.(5.11), it is clear that this controller results in large global errors. Also case 9, where predictive control is used, results in too large global errors.

Also the testcases with variable order will be discussed. For this circuit, without discontinuities, these cases are more efficient, because of the higher maximal order.

From Tab.(5.12), it becomes clear that for these testcases, case 1v is the most efficient one. But case 4 with fixed order still generates smoother error and stepsize sequences. Because of the smaller maximum order, the linear controllers with fixed order have much more rejections and are less efficient. However, case 12 with fixed order is still more efficient than Pstar. Furthermore, its global errors are much smaller than for case 1v. However, the stepsize control mechanism of Pstar appears to be rather smooth, compared to case 12 with fixed order.

In Figures (5.17) and (5.18), the error sequences have been shown for cases 1v (variable order) and 4 (fixed order). Note that the error sequence of case four still has oscillating behaviour, despite the much better smoothness. These oscillations are caused by the high frequent input source, which will always occur in the stepsize and error sequences. However, the controllers with oscillating behaviour can also produce oscillating or alternating behaviour, which is not caused by the exact solution. Because the period of the oscillations of case 4 is larger than two (period of alternations), its smoothness is smaller.

Case	V_1	V_2	V_3	V_4	i_E
1	1.12e-02	6.20e-03	6.20e-03	3.80e-03	6.38e-03
2	1.54e-02	1.02e-02	1.02e-02	6.21e-03	6.09e-03
3	1.19e-02	7.13e-03	7.13e-03	4.30e-03	5.39e-03
4	2.19e-03	1.19e-03	1.19e-03	4.27e-04	1.03e-03
5	3.15e-02	2.10e-02	2.10e-02	1.49e-02	1.13e-02
6	9.93e-01	7.98e-01	7.98e-01	7.28e-01	2.67e-01
7	1.63e-02	1.15e-02	1.15e-02	7.59e-03	6.03e-03
8	9.63e-03	6.29e-03	6.29e-03	3.79e-03	4.03e-03
9	1.54e+00	1.06e+00	1.06e+00	8.42e-01	4.79e-01
10	1.31e-02	8.31e-03	8.31e-03	4.56e-03	4.77e-03
11	7.19e-03	4.36e-03	4.36e-03	2.26e-03	3.08e-03
12	1.23e-02	7.79e-03	7.79e-03	4.44e-03	4.51e-03
13	7.98e-01	5.84e-01	5.84e-01	4.45e-01	3.28e-01
14	8.21e-03	5.08e-03	5.08e-03	3.06e-03	4.03e-03
15	5.71e-03	3.96e-03	3.96e-03	2.48e-03	1.85e-03
16	5.77e-01	4.12e-01	4.12e-01	2.85e-01	2.00e-01
17	4.15e-01	2.66e-01	2.66e-01	1.22e-01	1.50e-01
18	7.04e-01	5.19e-01	5.19e-01	3.83e-01	2.63e-01
19	1.72e+00	1.42e+00	1.42e+00	1.17e+00	3.66e-01

Table 5.11: Global errors for the linear electrical circuit with fixed order.

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	993	41	1034	0.67	0.18
2v	1023	22	1045	0.57	0.10
3v	1093	111	1204	0.95	0.25
4v	1086	153	1239	0.92	0.27
5v	1163	178	1341	0.81	0.43
6v	1173	197	1370	0.99	0.33
7v	1404	918	2322	1.00	0.69

Table 5.12: Numerical results for the linear electrical circuit with variable order.

Case	V_1	V_2	V_3	V_4	i_E
1v	3.00e-02	2.31e-02	2.31e-02	1.80e-02	1.16e-02
2v	2.06e-02	1.42e-02	1.42e-02	1.06e-02	7.14e-03
3v	3.34e-02	2.71e-02	2.71e-02	2.18e-02	9.31e-03
4v	4.63e-02	3.29e-02	3.29e-02	2.52e-02	1.74e-02
5v	3.68e-02	2.57e-02	2.57e-02	2.01e-02	1.59e-02
6v	5.34e-02	4.11e-02	4.11e-02	3.54e-02	1.67e-02
7v	3.26e-01	2.39e-01	2.39e-01	2.01e-01	1.11e-01

Table 5.13: Global errors for the linear electrical circuit with variable order.

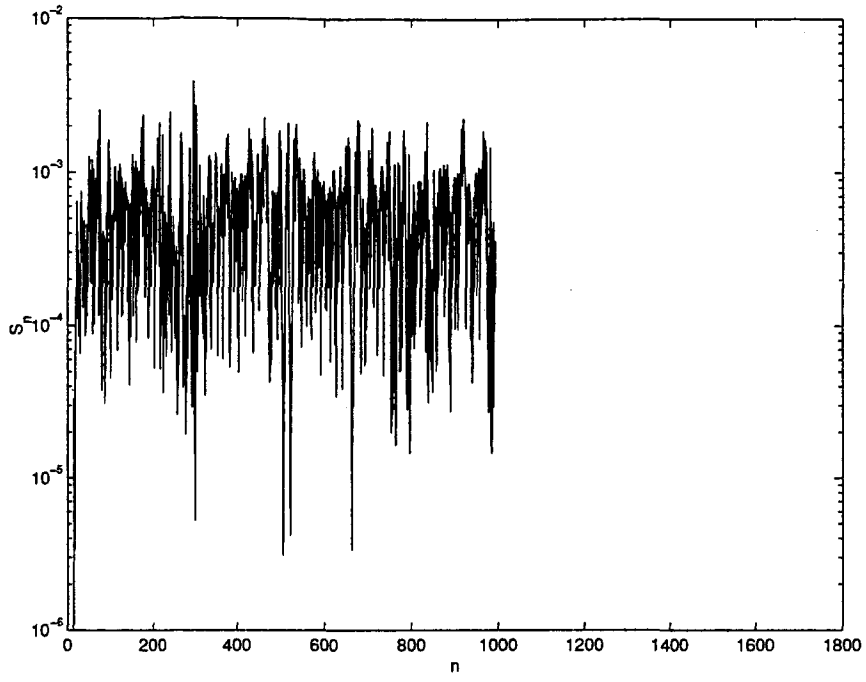


Figure 5.17: Error sequence for the linear circuit with Pstar (case 1v).

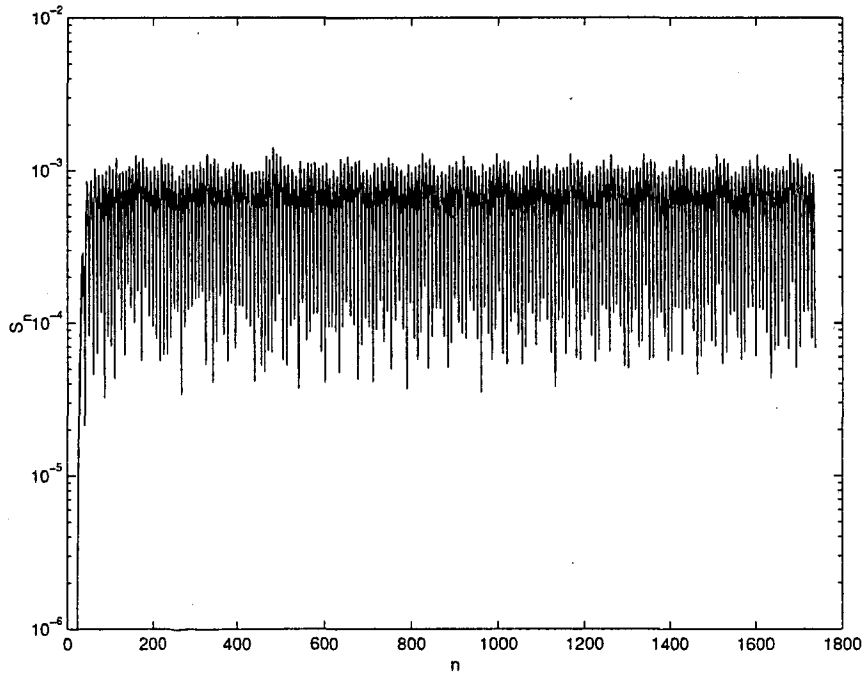


Figure 5.18: Error sequence for the linear circuit with the smooth PI-controller without oscillations (case 4).

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1	262	47	815	0.91	0.36
2	240	49	671	0.94	0.34
3	245	35	630	0.66	0.30
4	231	82	1348	1.28	0.28
5	239	61	1058	1.06	0.35
6	210	84	1175	1.29	0.44
7	233	66	744	1.14	0.25
8	161	65	1820	1.31	0.28
9	281	108	1576	1.34	0.41
10	238	60	1045	1.12	0.29
11	171	73	1391	1.32	0.28
12	235	55	885	1.08	0.31
13	247	99	1208	1.21	0.41
14	221	67	1333	1.07	0.26
15	119	50	1248	1.39	0.29
16	204	95	1565	1.20	0.29
17	325	60	1114	1.05	0.36
18	181	77	1288	1.30	0.28
19	135	75	2361	1.28	0.30

Table 5.14: Numerical results for the Van de Pol oscillator with fixed order.

5.4.3 Van de Pol equation

The Van de Pol equation has been solved with Pstar. From the Tables (5.14) and (5.15), it becomes clear that for the Van de Pol oscillator, case 2v with variable order is the most efficient one. Note that this is caused by the smaller number of Newton iterations per step, compared to deadbeat control. However, case 3 with fixed order, generates smoother stepsize and error sequences.

In Figures (5.19) and (5.20), the error sequences are shown for the cases 1v and 3. Also, for these cases, the sequence of ϵ has been shown, which is dynamic, because ϵ is equal to θ_nTOL.

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	258	36	705	0.81	0.35
2v	262	32	571	0.89	0.33
3v	256	42	644	0.69	0.43
4v	246	63	711	0.76	0.33
5v	246	46	676	0.76	0.30
6v	249	59	842	0.86	0.32
7v	275	51	783	1.18	0.34

Table 5.15: Numerical results for the Van de Pol oscillator with variable order.

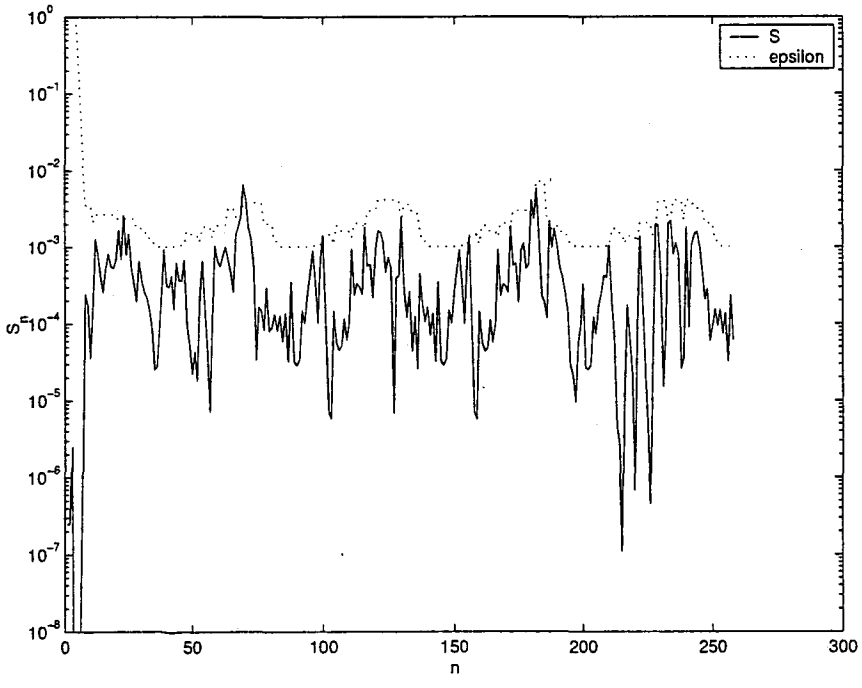


Figure 5.19: Error sequence and the dynamic reference level ϵ for the Van de Pol oscillator with Pstar (case 1v).

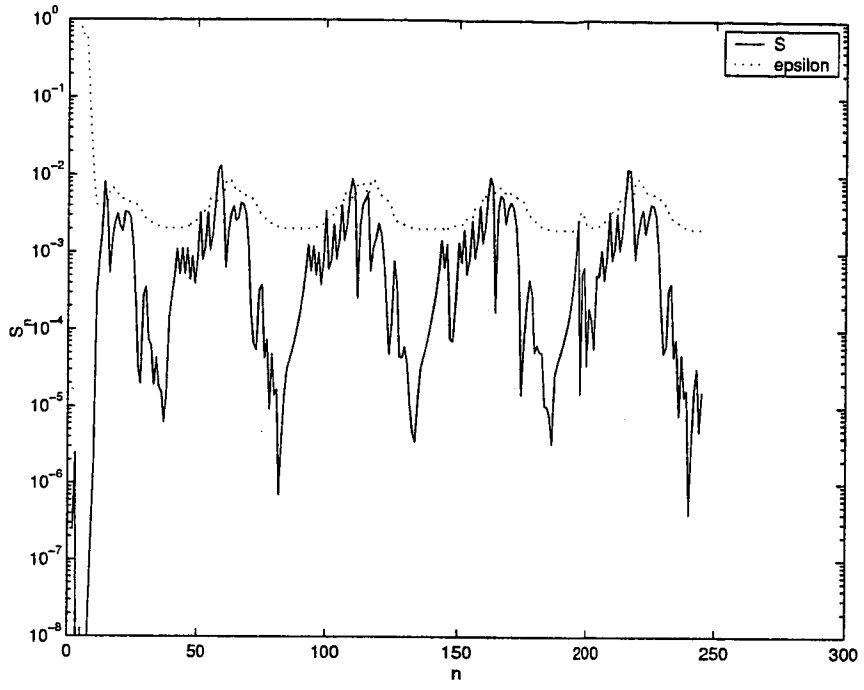


Figure 5.20: Error sequence and dynamic reference level ϵ for the Van de Pol oscillator with smooth I-control (case 3).

In subsection 4.4.5, it has been shown that linear control is equivalent to the prediction of the disturbance $\hat{\varphi}_n$. In Figures (5.21) and (5.22), the disturbance and its predicted values have been shown for cases 1 and 3 with fixed order. Clearly, deadbeat control uses constant extrapolation to predict φ_n, while smoother controllers use also the previous control errors. This implies that the smoother controllers don't depend on small fluctuations of $\hat{\varphi}_n$. This means that the predictor $\hat{\varphi}^P$ for case 1 is more sensitive to sudden fluctuations of $\hat{\varphi}$.

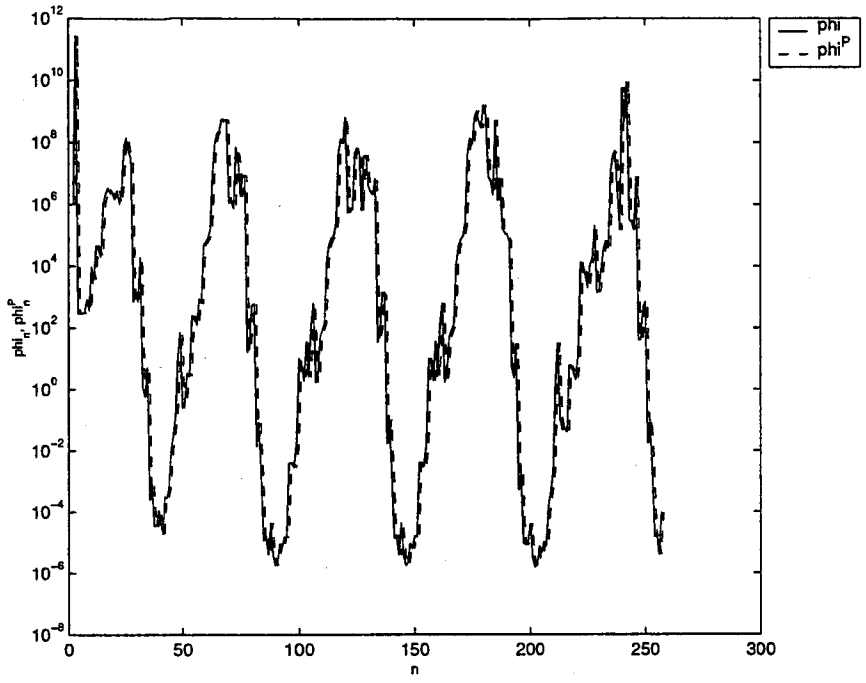


Figure 5.21: Prediction for the disturbance sequence with deadbeat control (case 1).

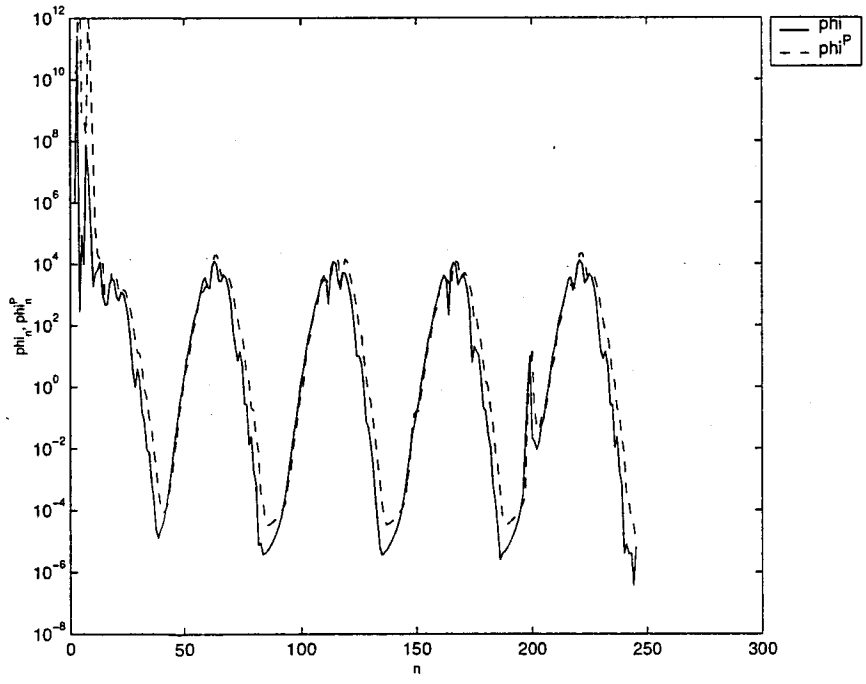


Figure 5.22: Prediction for the disturbance sequence with smooth I-control (case 3).

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	6465	947	43232	0.85	0.58
2v	6934	777	40234	0.79	0.48
3v	7927	862	43527	0.96	1.03
4v	6036	1505	50033	0.86	0.55
5v	6423	714	39619	0.74	0.85
6v	6099	1222	45929	0.78	0.57
7v	5598	940	38596	0.76	0.91
3	5253	801	88089	0.74	0.47

Table 5.16: Numerical results for perf_mos7_qubic_6953.

5.4.4 perf_mos7_qubic_6953

In Tab.(5.16), the numerical results are shown for the testcircuit perf_mos7_qubic_6953. First, some testcases with variable order are investigated. Afterwards, only case 3 with fixed order has been run. Clearly, variable order is more efficient for this case. Controlling with smooth I-control (case 2v, 5v or 7v) is more efficient than the original Pstar controller (case 1v). Also they generate smoother error and stepsize sequences. However, case 3 with fixed order is still the smoothest version. Note that for this example, the controllers which are based on process model two need less Newton iterations.

In Figures (5.23) and (5.24), the error sequences are shown for case 1v (Pstar) and case 5v.

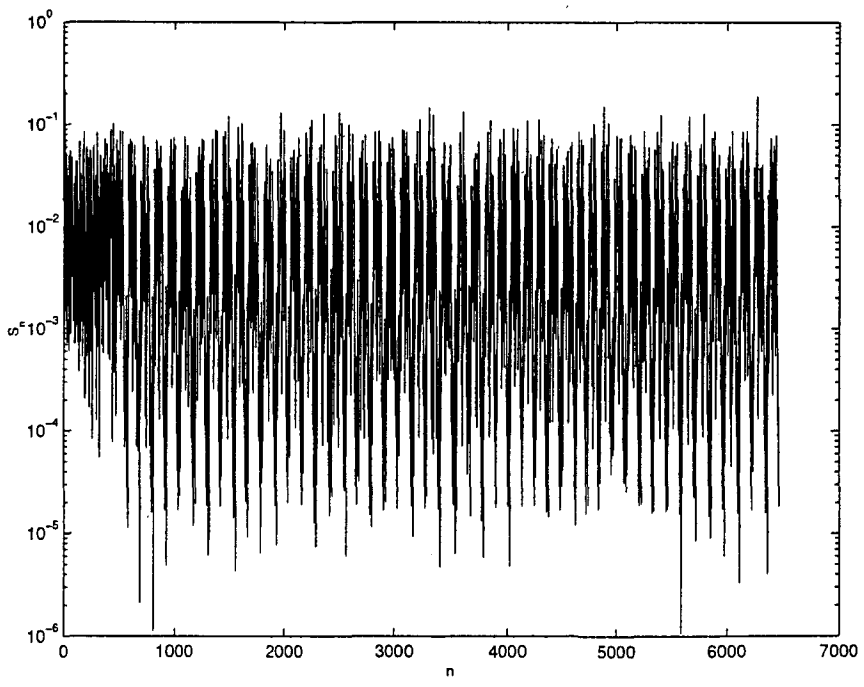


Figure 5.23: Error sequence for perf_mos7_qubic_6953 with Pstar (case 1v).

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	5641	108	8777	1.01	0.09
2v	6026	108	9283	0.94	0.09
3v	5886	148	9377	0.73	0.11
4v	5561	240	9256	0.94	0.09
5v	5645	132	8914	0.83	0.09
6v	5497	212	9986	1.38	0.09
7v	5876	233	9671	0.99	0.12
3	5788	102	9875	0.69	0.09

Table 5.17: Numerical results for bim2.

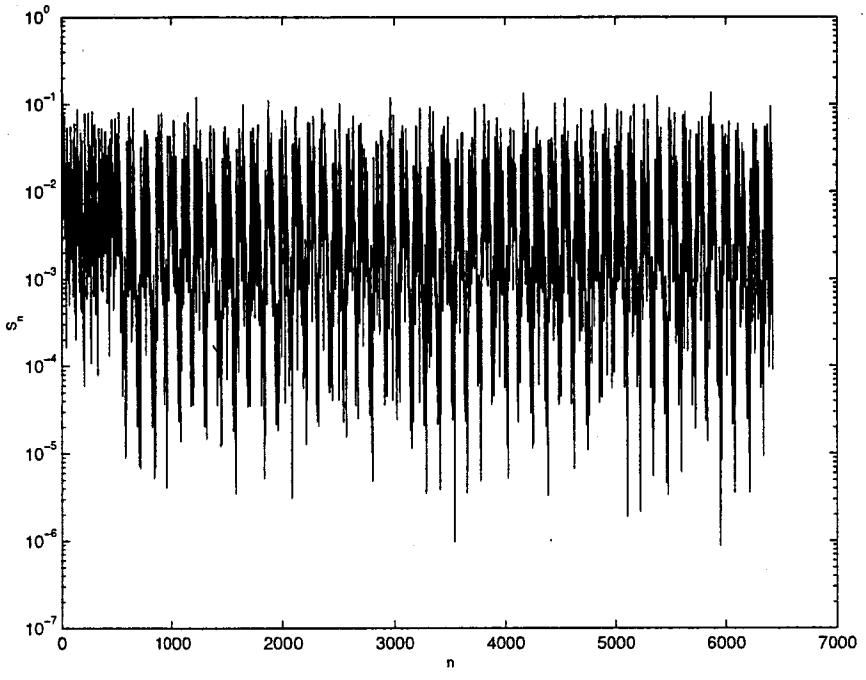


Figure 5.24: Error sequence for perf_mos7_qubic_6953 with smooth I-control, based on model 2 (case 5v).

5.4.5 bim2

In Tab.(5.17), the numerical results for this testcircuit have been shown. For this case, it appears that Pstar (case 1v) is the most efficient version. However, the other controllers generate smoother error sequences.

In Figures (5.25) and (5.26), the error sequences have been shown for testcases 1v and 3. If the stepsizes become very small, the smooth testcase 3 generates a much smoother error sequence. Probably, this is caused by the numerical errors for the very small stepsizes. which are used by Pstar (case 1v).

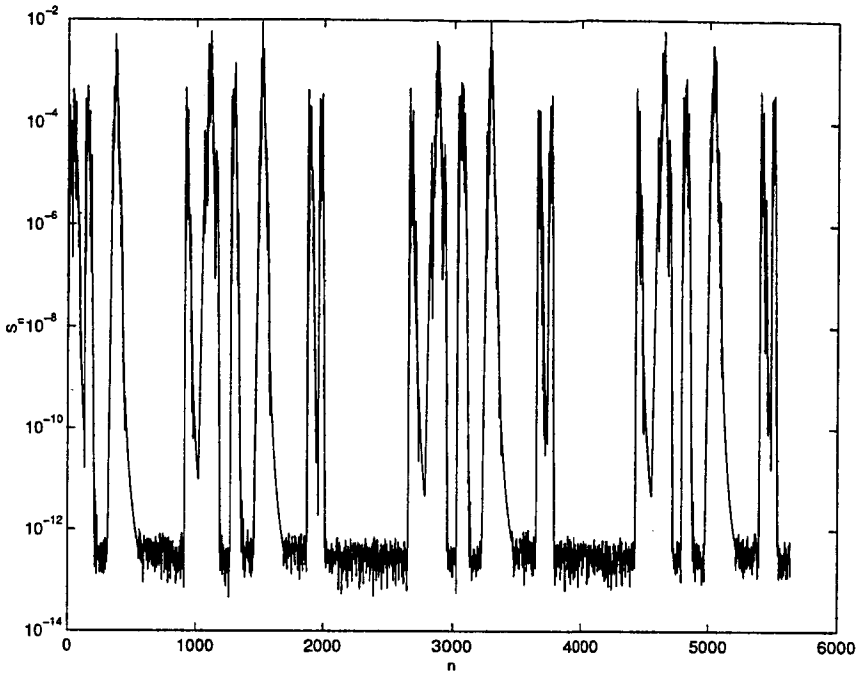


Figure 5.25: Error sequence for bim2 with Pstar (case 1v).

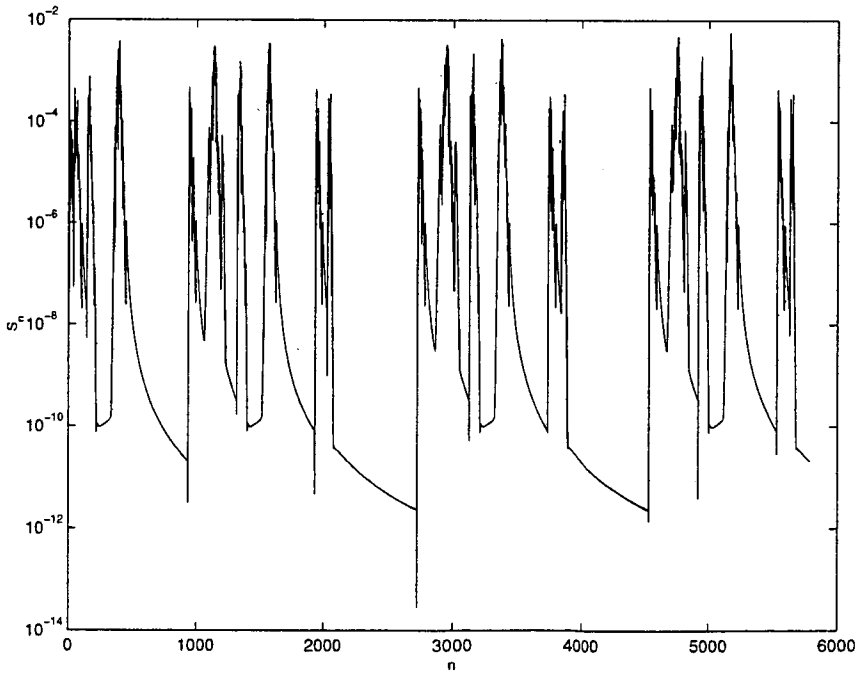


Figure 5.26: Error sequence for bim2 with smooth I-control (case 3).

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	1406	83	5604	1.15	0.49
2v	1657	73	5994	1.18	0.47
3v	1640	104	6038	1.31	0.80
4v	1275	216	5894	1.16	0.51
5v	1376	107	5585	1.17	0.50
6v	1313	153	5523	1.18	0.52
7v	1395	175	6269	1.17	0.56
3	1347	110	6100	1.07	0.49

Table 5.18: Numerical results for sram.

Case	# stepsizes	# rejections	# Newton iterations	Smoothness of local error sequence	Smoothness of stepsize sequence
1v	2750	97	8029	1.07	0.64
2v	2912	98	8262	1.07	0.67
3v	2994	94	8382	1.09	0.72
4v	2731	121	8079	1.03	0.58
5v	2788	119	8180	1.05	0.63
6v	2782	119	8171	1.06	0.61
7v	3677	123	9336	1.09	0.70
3	2553	101	7854	1.00	0.58

Table 5.19: Numerical results for perf_mos9_c100_7342a.

5.4.6 sram

In Tab.(5.18), the numerical results for the testcircuit sram are shown. Cases 5v and 6v, which are based on process model two appear to be more efficient than Pstar. All controllers produce non-smooth error sequences, but case 3 with fixed order is again the smoothest one.

5.4.7 perf_mos9_c100_7342a

For the rather large testcircuit perf_mos9_c100_7342a from the roza group, it appears from Tab.(5.19) that case 3 with fixed order is the most efficient one. This has been caused by the nonspecified discontinuities, which occur if the maximal order is equal to five. For this testcircuit, all controllers produce non-smooth results. From Tab.(5.19), it would follow that case 3 also produces smoother error and stepsize sequences.

In Figures (5.27), (5.28), (5.29) and (5.30), the error and stepsize sequences of case 1v (Pstar) and case 3 with fixed order, have been shown. Although the error sequence for testcase 3 should be more smooth than the other error sequence, this is not visible from the Figures. The influence of the discontinuities is clearly visible, because then the stepsize controller starts again with very small timesteps. Because the smooth controller (case 3) needs less stepsizes than the stepsize mechanism of Pstar (case 1v), it seems that the timepoints of these discontinuities are not equal.

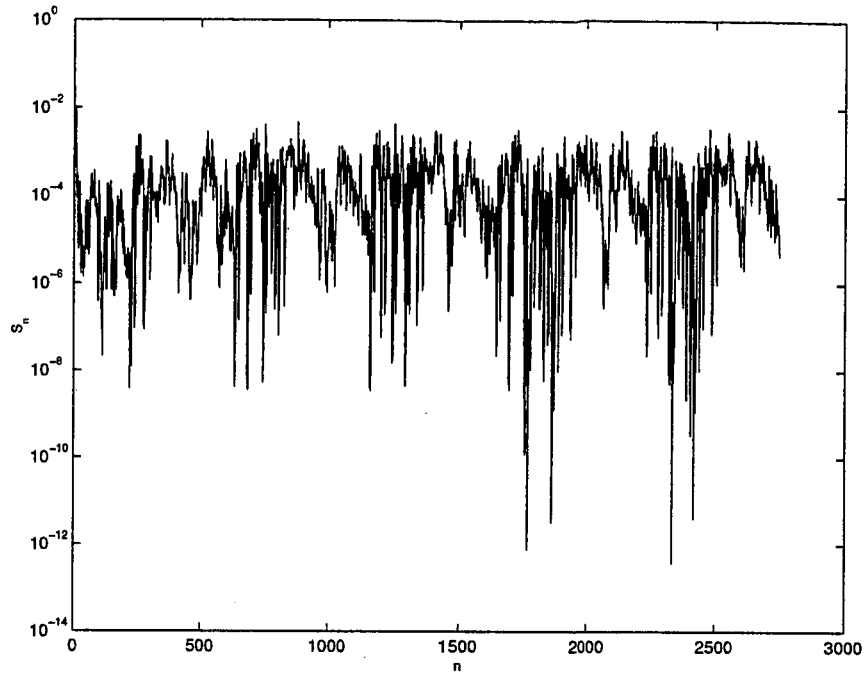


Figure 5.27: Error sequence for perf_mos9_c100_7342a with Pstar (case 1v).

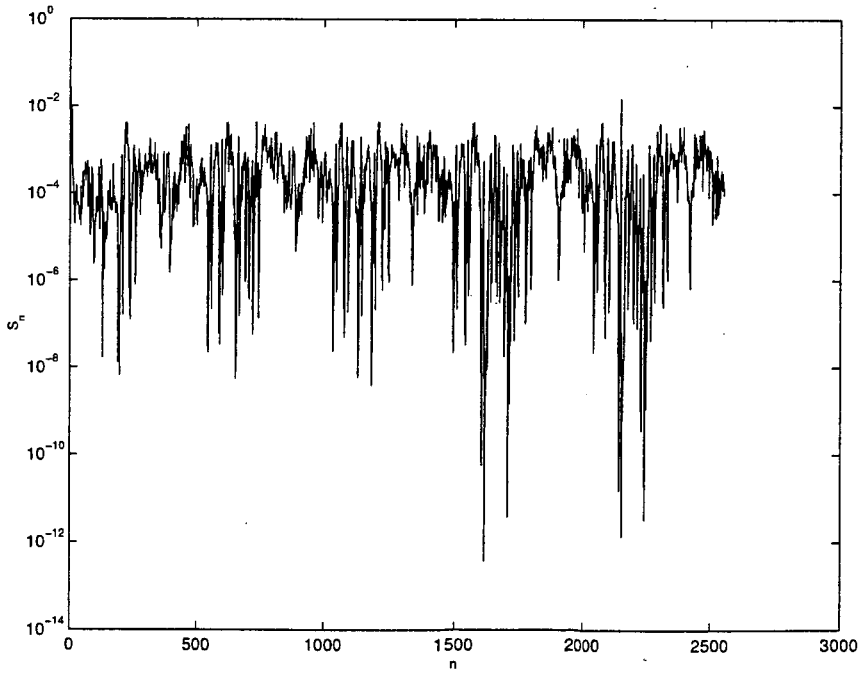


Figure 5.28: Error sequence for perf_mos9_c100_7342a with smooth I-control (case 3).

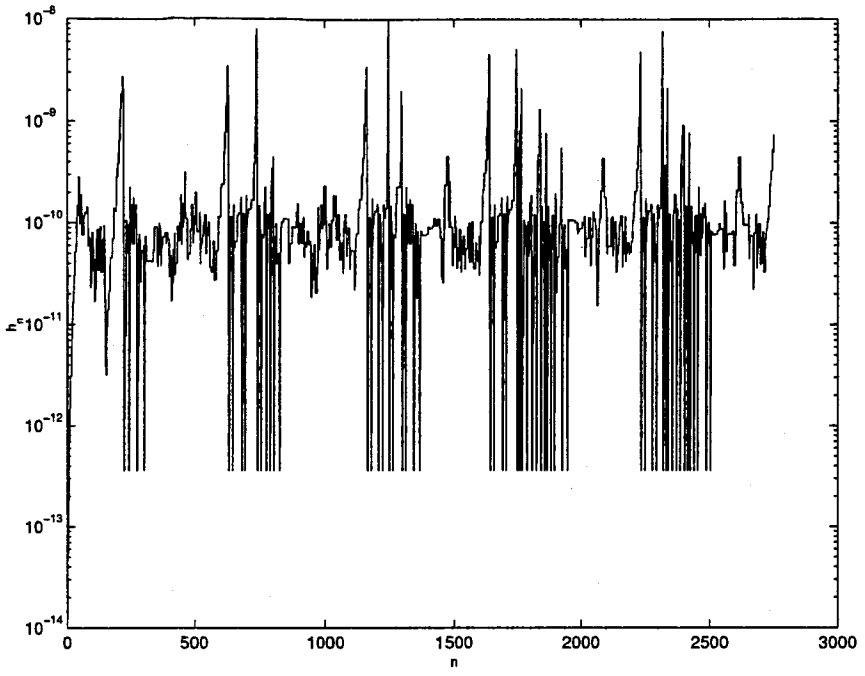


Figure 5.29: Stepsize sequence for perf_mos9_c100_7342a with Pstar (case 1v).

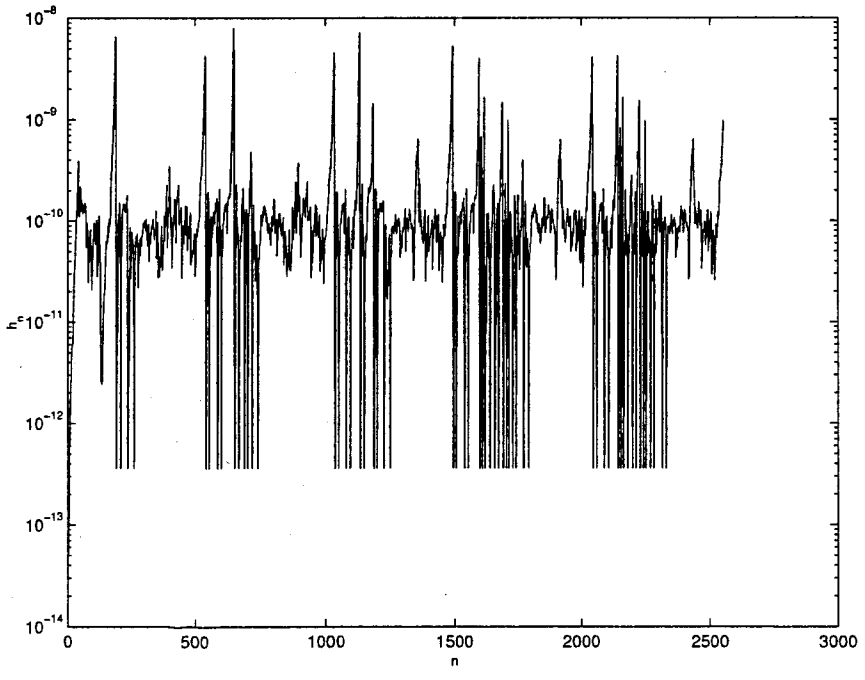


Figure 5.30: Stepsize sequence for perf_mos9_c100_7342a with smooth I-control (case 3).

Chapter 6

Conclusion

6.1 Conclusion

From the chapters three and four, the next conclusions can be drawn.

- For the one step methods, the local error estimate $\hat{r}_n$ depends only on the last stepsize h_n. For the LMM-methods and the BDF methods, $\hat{r}_n$ depends also on the previous stepsizes. This means that adaptive timestepping is more difficult for the BDF methods.

- Adaptive timestepping for DAE solvers can be viewed as a digital linear control system. In logarithmic form, $\log \hat{r}$ depends linearly on $\log h$ and $\log \hat{\varphi}$, where $\hat{\varphi}$ is a disturbance, which is nearly independent on h. Therefore, it is possible to use a control-theoretic approach to adaptive stepsize control. If after an accepted step, the next stepsize must be computed, this can be done by means of a digital linear control law. In practice, often additional nonlinear or logic control actions are used, which can disturb the designed properties.

- For one step methods, the local error estimate satisfies a very simple process model. In general, for LMM methods, it is not possible to describe the behaviour of $\log \hat{r}$ as a linear system. As first order approximation, the model for one step methods can be used. With linearization techniques, a more accurate model can be derived.

- If the process model is available, it is possible to design digital linear controllers of finite order for this control system, which must keep the control error $\log r - \log \epsilon$ near zero. Here, the wanted accuracy ϵ is equal to $\epsilon = \theta \mathrm{TOL}$, where TOL is the tolerance level. The safety factor $0 < \theta < 1$ can be used to avoid too many rejections.

- On the basis of the characteristic equation, the poles of the closed loop system can be placed. To ensure stability, the poles must have absolute values, which are smaller than one. However, the poles also determine the behaviour of the stepsize and error sequences, such as oscillating or alternating behaviour. If all poles are equal, the results will be very smooth, because there will be no oscillations at all.

- Besides the poles, it is also possible to design the adaptivity of the controller. For all controllers, the order of adaptivity p_A must be larger or equal to one. This means that there are no control errors, if the disturbance $\hat{\varphi}$ is constant. Controllers with higher order adaptivity can also follow polynomial trends of the disturbance. This can be useful, if $\hat{\varphi}$ has not-smooth behaviour.

- Controllers can also be designed as filters of the alternating behaviour of the stepsize or error sequence.

- An important property of stepsize controllers is the constraint validation. This means that the next stepsize will not be rejected, if the process model is correct and $\hat{\varphi}$ satisfies the constraint of the equation (4.47). If the poles are equal to $\{re^{\frac{2k\pi i}{N+M}} : k = 0, \ldots, N+M-1\}$ with $r \in [0, 1)$, this is always the case. However, smooth controllers with all poles are equal to $r \neq 0$, do not have this property.

- A special family of linear controllers are the PI-controllers, which only depend on two free control parameters. Their predictive versions are called the PC-controllers. It is also possible to use PI control after a rejected step, which will remove the not-smooth behaviour of the stepsize and error sequences because of the rejections. A combination of PI controllers, which is also used after rejections is called a combined PI controller.

- It is also possible to use a nonlinear version of the linear control law, which results in the same linearized closed loop dynamics. This nonlinear controller will have the advantage that they also have constraint validation if the linearized process model is not correct.

These theoretical results have been verified in the chapter five. It appears that:

- Indeed, generally, the deadbeat controller generates less smooth error and stepsize sequences. For the small circuits, they are also less efficient than other smoother controllers for the tested methods "ode45", "BDFcontrol" and Pstar. Also for the larger circuits "perf_mos7_qubic_6953", "sram" and "perf_mos9_c100_7342a", it is possible to design better stepsize controllers. Only for the circuit "bim2", Pstar remains the optimal choice.

- The safety factor θ plays an important role, because it determines the number of rejections and stepsizes.

- Also the release factor is important, because it makes the tolerance level dynamic and can reduce the number of rejections. For small stepsizes, the tolerance level is very high, which makes larger stepsizes and safety factors possible.

- The combined PI controller appears to be rather efficient for stiff equations. For the stiff example for the RK method and the Van de Pol oscillator for the BDF method, its results were very good, compared to the other tested controllers.

- For Pstar, the order control has large influence on the results. If the original dynamic order control of Pstar is used with the maximal order five, the process models will be not correct. Therefore, the linear controllers do not satisfy their designed properties. However, if the order is fixed with maximal order three, their results are much better. For smooth circuits, the dynamic order control will be more efficient, because of the higher order. But it appears that there occur more nonspecified discontinuities, if the maximal order is equal to five. This reduces the efficiency of the original order control, because then the order is decreased to one.

- Controllers with higher order adaptivity perform rather bad, in comparison to the other controllers. Probably, this is caused by the errors of the linear extrapolation.

- Stepsize filters perform better than error filters. They are useful, if high-frequent oscillations occur, e.g. in the linear electrical circuit.

- Constraint validation is important if stiff DAE's are solved, because then the number of rejections will be reduced. However, for smooth problems, one could use controllers with equal poles, which do not have oscillating behaviour. Because these controllers do not have constraint validation, they can not used for problems with many rejections.

6.2 Further research

The next subjects could be investigated more profoundly.

- Up to now, for the controller design, it was assumed that the order is kept fixed. From the experiments, it is clear that sometimes it is better to use variable order, because then larger steps can be used for smooth parts, which increases the speed. Therefore, in practice a combination of stepsize and order control is used. However, the process model of the behaviour of this combined control system is not clear yet.
- Pstar uses a voltage based criterion as error estimate, but in general, this norm is not equivalent to the local error or the local discretization error. Probably, this error estimate does not satisfy the process models of the local discretization error.
- It may be possible to make the safety factor θ and the tolerance level ϵ dynamic. Because $\epsilon = \theta \mathrm{TOL}$, it follows that

  $$\log \epsilon_n = \log \theta_n \log \mathrm{TOL}$$

  Now, it is possible to use feedback control for $\log \theta$, for example:

  $$\log \theta = \log \theta^* + G_T(q)(\log \mathrm{TOL} - \log \hat{r})$$

  Of course, this makes the closed loop dynamics much more complex, but it is still linear. The choice of θ^* remains the question. Pstar uses already a dynamic safety factor, which is proportional to the release factor, but this release factor depends not linearly on the stepsize.
- Because there are many criteria, which have to be optimized, it could be helpful to use optimization techniques. Because the closed loop dynamics are linear, it is also possible to describe them as a state space model. The smoothness of the stepsize- and the error sequence can be characterized as

  $$\sum_{k=1}^{N} (\hat{r}_k - \hat{r}_{k-1})^2 \quad \sum_{k=1}^{N} (\hat{h}_k - \hat{h}_{k-1})^2$$

  The constraint $\sum_{k=1}^{N} h_k = T$ could be described as $\sum_{k=1}^{N} \log h_k = \bar{T}$. Theoretically, now it is possible to use LQ control or other techniques from optimal control to derive a feedback law.
- Maybe, system identification would be useful to determine better process models for the integration methods. Then, the coefficients of $G(z)$ are determined on base of an available stepsize and error sequence.
- Because the Newton-Raphson method is an important part of the implicit integration methods, the influence of stepsize control on this method could be investigated. Maybe, it is possible to control also the number of Newton iterations.
- Because multirate appears to be a hot item, nowadays, it is also recommended to investigate adaptive stepsize control in combination with multirate more profoundly.

Appendix A

Implementation of stepsize controller in Pstar

A.1 Important parameters

Pstar uses the next debug parameters with default values.

$$
\begin{array}{rcll}
\tau_{abs,min\_step} & = & \text{ABS\_MIN\_TIMESTEP} & (10^{-13}), \\
\epsilon_{rel,int} & = & \text{REL\_EPS\_TIME\_INTERVAL} & (5 \cdot 10^{-7}), \\
\tau_{rel,min\_step} & = & \text{REL\_MIN\_TIMESTEP} & (5 \cdot 10^{-11}), \\
\tau_{rel,release} & = & \text{REL\_RELEASE\_TIMESTEP} & (2 \cdot 10^{-4}), \\
\tau_{rel,max\_step} & = & \text{REL\_MAX\_TIMESTEP} & (2 \cdot 10^{-2}), \\
\tau_{abs,acc\_volts} & = & \text{REL\_ACC} & (5 \cdot 10^{-3}), \\
\epsilon_{min} & = & \text{EPS\_STEPSIZE\_LOWER\_FACTOR} & 0.8, \\
\epsilon_{max} & = & \text{EPS\_STEPSIZE\_UPPER\_FACTOR} & 2, \\
\omega_{min} & = & \text{MIN\_STEPSIZE\_FACTOR} & 0.1, \\
\omega_{max} & = & \text{MAX\_STEPSIZE\_FACTOR} & 5, \\
\omega^*_{max} & = & \text{MAX\_STEPSIZE\_FACTOR\_ADJUST} & 3, \\
\mu & = & \text{STEPSIZE\_ADJUSTED\_FACTOR} & 0.1, \\
\text{RTOL} & = & \text{MAX\_ALLOWED\_RATIO\_LTE\_EMAX} & 2.
\end{array}
$$

The start and end time of the time interval are equal to t_s and t_e respectively. The first two discontinuity points greater than t_s are denoted by $t_{dis,1}$ and $t_{dis,2}$. Furthermore, t_1 is the first time point for which output is required. Then, the first time interval length, maximum scale factor and scale factor respectively are defined by:

$$
\begin{array}{rcll}
\Delta t_{first\_int} & = & \min(t_1 - t_s, t_{dis,1} - t_s, t_{dis,2} - t_{dis,1}) & , \\
\sigma_{max} & = & 0.05 \frac{\Delta t_{first\_int}}{\max(\epsilon_{rel,step}, \epsilon_{rel,int})} & (10^5 \Delta t_{first\_int}), \\
\sigma & = & \min(T, \sigma_{max}). &
\end{array}
$$

Then the minimum, maximum, release and start timestep are defined by respectively

$$
\begin{array}{rcll}
\Delta t_{min} & = & \sigma\,\tau_{rel,min\_step} & (5\cdot 10^{-11}\sigma),\\
\Delta t_{max} & = & T\,\tau_{rel,max\_step} & (2\cdot 10^{-2}T),\\
\Delta t_{release} & = & T\,\tau_{rel,release} & (2\cdot 10^{-4}T),\\
\Delta t_{start} & = & \max(\sigma^2\tau_{rel,min\_step},\,\sigma\,\tau_{rel,min\_step},\,\tau_{abs,min\_step}) & (10^{-11}\max(5\max(\sigma^2,\sigma),10^{-2})).
\end{array}
$$

The parameters ϵ_{step} and ϵ_{int}, used in discontinuous timesteps, are defined by:

$$
\begin{aligned}
\epsilon_{step} &= \max(\sigma\,\epsilon_{rel,step},\,\tau_{abs,min\_step}),\\
\epsilon_{int} &= \max(\sigma\,\epsilon_{rel,int},\,\tau_{abs,min\_step}).
\end{aligned}
$$

Let M be the **numform** in Pstar (default, M = 4). Now, the absolute tolerance is defined by

$$
\epsilon^{V}_{abs} = \tau_{abs,acc\_volts}10^{4-M} = 5\cdot 10^{1-M}. \tag{A.1}
$$

The dynamic release time factor is given by

$$
f_{release} = 1+\frac{\Delta t_{release}}{h_{n-1}} = 1+2\cdot 10^{-4}\frac{T}{h_{n-1}}. \tag{A.2}
$$

A.2 Time stepping algorithm

A.2.1 Error estimation

Pstar uses an Predictor Corrector implementation of the BDF method[1]. There are two versions of error-control, which can be performed by Pstar:

1. Discretization error based criterion;
2. Voltage based criterion;

The first version controls the discretization error itself, while the second version only considers the voltage differences. It is also possible to use local error or global error transformations, such that instead of the LDE, the local error or global error is controlled.

Default, the voltage based criterion is used, because this corresponds better with the wants of the designer.

This version use the next error estimators, which are based on the voltage differences. The voltages $v^{m,(0)}$ are the predicted values, derived by use of extrapolation. For $\mathcal{N}=1$, the ∞-norm is used, while for the default value $\mathcal{N}=2$, an adapted 2-norm is used, where N is the number of voltages.

$$
\begin{aligned}
S_n &= \max_m\left(\left|v^m(t_n)-v^{m,(0)}\right|\right) \quad \text{if} \quad \mathcal{N}=1,\\
S_n &= \sqrt{\tfrac{1}{N}\textstyle\sum_m\left(v^m(t_n)-v^{m,(0)}\right)^2} \quad \text{if} \quad \mathcal{N}=2.
\end{aligned}
$$

It can be proved that if $C(t,\mathbf{x})$ is constant, $\|C\| >> \|h_n G(t_n,\mathbf{x}_n)\|$ and the currents are negligible:

$$
\begin{aligned}
S_n &\approx \tfrac{1}{|C_p|}\|\mathbf{d}_n\|_\infty \quad \text{if} \quad \mathcal{N}=1,\\
S_n &\approx \tfrac{1}{|C_p|\sqrt{N}}\|\mathbf{d}_n\|_2 \quad \text{if} \quad \mathcal{N}=2,
\end{aligned}
$$

[1] See the subsections 3.4.2 and 3.4.3.

where $\mathbf{d}_n$ is the local error[2] of the BDF-method. Now, Pstar scales these ratiosums:

$$\begin{aligned} \bar{S}_n &= \frac{S_n}{\epsilon^V_{abs} f_{release}} \quad \text{if} \quad \mathcal{N} = 1, \\ \bar{S}_n &= \frac{5S_n}{\epsilon^V_{abs} f_{release}} \quad \text{if} \quad \mathcal{N} = 2. \end{aligned}$$

A.2.2 Adaptive stepsize control in Pstar

Notation of Pstar

The stepsize is accepted, if

$$\bar{S}_n \leq RTOL(=2).$$

If the stepsize h_n is accepted, Pstar determines the next step ratio:

$$\omega_{n+1} = \left(\frac{1}{\bar{S}_n}\right)^{\frac{1}{p+1}}. \tag{A.3}$$

Afterwards, some nonlinear control parts are used to get safe.

$$\begin{aligned} \omega_{n+1} &= \max(\min(\omega_n, \omega_{max}), \omega_{min}), \\ (\epsilon_{min} < \omega_{n+1} < \epsilon_{max}) &\Rightarrow \omega_{n+1} = 1, \\ h_{n+1} &= \omega_{n+1} h_n, \\ h_{n+1} &= \max(\min(h_{n+1}, h_{max}), h_{min}). \end{aligned}$$

The first three times, after a rejection, $\omega^*_{max}(=3)$ is used instead of $\omega_{max}(=5)$. If h_n is rejected or if the Newton method did not converge, Pstar tries again with a smaller timestep.

$$h_n = \mu h_n.$$

If output is requested at given output points, interpolation is necessary to give the output. In that case, the stepsize ratio will also be dependent on the interpolation error.

Corresponding notation with chapter four.

It is possible to cast the above procedure in a control law as described in the proceeding section. Define the absolute tolerance level TOL and safety factor θ_n by

$$\begin{aligned} \text{TOL} &= \frac{\epsilon^V_{abs}}{5}\text{RTOL} \qquad (= 5 \cdot 10^{-M}), \\ \theta_n &= f_{release}\frac{1}{\text{RTOL}} \quad (= 0.5 + 10^{-4}\frac{T}{h_n}). \end{aligned}$$

This means that default $S_n = \theta_n \text{TOL} \bar{S}_n$. Then, controller (A.3) is equivalent with the next deadbeat-controller

$$h_{n+1} = \left(\frac{\theta_n \text{TOL}}{S_n}\right)^{\frac{1}{p+1}} h_n.$$

This step is accepted, if

$$S_n \leq f_{release}\text{TOL}.$$

Note that the controller (A.3) uses a dynamic release time factor $f_{release}$, which depends on h_n. This factor controls the rejection tolerance level and the controller. For small stepsizes, this factor will be

[2] See Def.3.5.

larger, which implies less rejections, because of the lower tolerance level, and larger stepsizes. Thus, $f_{release}$ reduces the amount of work, because the work is proportional to the numer of stepsizes and rejections.

For large stepsizes, the release time factor is almost equal to one. Only for stepsizes, smaller than $2 \cdot 10^{-4}T$, $f_{release}$ will become larger than two, to avoid very small stepsizes.

A.3 New stepsize controller for Pstar

If a digital linear controller[3] is used, it follows from chapter four, that this controller has the next structure:

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\beta_0} \cdots \left(\frac{\epsilon}{\hat{r}_{n-N}}\right)^{\beta_{N-1}} \left(\frac{h_{n-1}}{h_{n-2}}\right)^{-\tilde{\alpha}_1} \cdots \left(\frac{h_{n-N+1}}{h_{n-N}}\right)^{-\tilde{\alpha}_{N-1}} h_{n-1}. \tag{A.4}$$

The control parameters depend on the process model and the wanted specifications of the controller. To implement this controller, the variables $h_{n-N+1}, \ldots, h_{n-1}, S_{n-N+1}, \ldots, S_{n-1}$ must be stored. Afterwards, it is sufficient to change the last line of the function block 'TRU_lte_and_stepsize_factor'. The PI-controller[4] can be implemented as follows, because it only needs the last two errors.

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{k_I} \left(\frac{\hat{r}_{n-2}}{\hat{r}_{n-1}}\right)^{k_P} h_{n-1}. \tag{A.5}$$

```
p_tr_cb->step_factor[p_tr_cb->level_ind] = F_pow( *p_ratiosum, kI ,
&status) *  F_pow( *p_previous_ratiosum / *p_ratiosum, kP , &status) ;
```

In general, the linear controller (A.4) has to be implemented as a C function. Afterwards, this function can be used to compute the next step ratio factor. If the nonlinear controller[5] is used, it follows

$$\begin{cases} \hat{\phi}_n^P = \hat{\phi}_{n-1}^{-\sigma_1} \cdots \hat{\phi}_{n-N}^{-\sigma_{N+M}} \left(\frac{\hat{r}_{n-1}}{\epsilon}\right)^{\rho_1} \cdots \left(\frac{\hat{r}_{n-M-N}}{\epsilon}\right)^{\rho_{N+M}}, \\ h_n^{1+P-k}(h_{n-1}+h_n)\cdots(h_{n-k+1}+\ldots.+h_n) = \frac{k!\epsilon}{\hat{\phi}_n^P}. \end{cases} \tag{A.6}$$

Now, $\hat{\phi}_n^P$ can be computed with the C function 'LinearControl'. Because h_n is the real positive root of a polynomial, a polynomial solver has to be implemented.

It is also possible to use the MATLAB library with use of the Pstar_MATLAB_interface. This has been described in appendix B.

A.4 Algorithm

Below, the structure of the program Pstar has been shown, when a transient analysis is performed. It is assumed that no mixed signals are used, but an ordinary transient analysis. Note that these diagrams are not complete, but only contain the most important parts. Furthermore, it is assumed that there are no discontinuities or compulsory timepoints.

In Fig.(A.1), the structure of the important files for the transient analysis have been shown.

[3] See page 95.
[4] See page 80.
[5] See page 95.

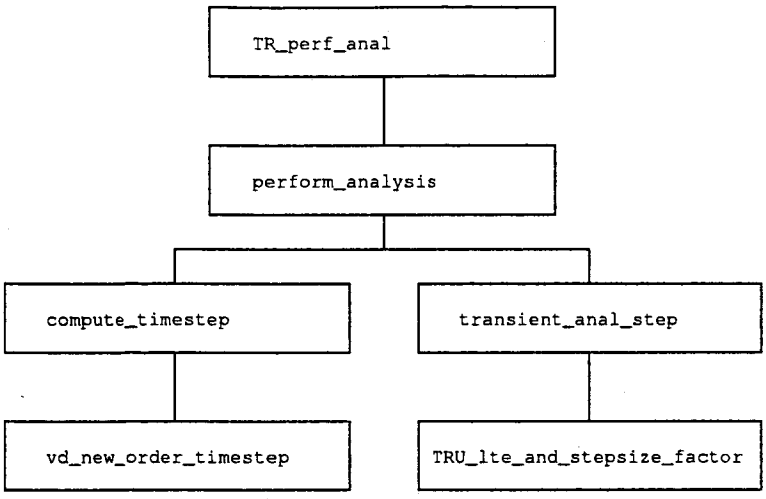


Figure A.1: Hierarchy of the transient routines in Pstar

Appendix B

Pstar-MATLAB-interface

The Pstar_MATLAB_interface is an useful library to test MATLAB-software for Pstar itself. It can be used for stepsize control, but for other purposes as well.

B.1 Installation

- Install first a private version of Pstar. Furthermore, the changed files must be rewritable.
- The files "memory.c", "Makefile" and "Makefile.general" must be changed.

  The files "pstar_matlab_interface.c" and "pstar_matlab_interface.h" contain the necessary code and have to be added.
- The file "pstar_matlab_interface.h" must be included in "panacea.c". Also, the commands "PMI_open_engine ()" and "PMI_close_engine ()" have to be added before and after the command "do{ PAN-panacea ...}". These commands open and close MATLAB.
- The file "pstar_matlab_interface.h" must also be included in the file, where MATLAB must be called. For an overview about the possible calls to MATLAB, the reader is referred to the file "pstar_matlab_interface.h".
- After editing, Pstar can be compiled and linked with the command

  ```
  make
  ```

  A debug version is made by

  ```
  make dbg
  ```

  The next command is necessary to perform these actions:

  ```
  cadenv glib
  ```
- To use this private version, the command "privpstar" or "privdbgpstar'" must be used. Furthermore, the Pstar-files must be located at directories with the same hierarchical position as the directory `.../obj`.
- The location of the MATLAB-scripts is free. With the command

```
export PSTAR_MATLAB_DATA=<address>
```

this location can be added to the path.

B.2 Application to PI control

The Pstar_MATLAB_interface can be used to investigate stepsize controllers for Pstar. Instead of the original stepsize control mechanism, a smoother PI controller is used.

In the function block "TRU_lte_and_stepsize_factor" in the file "transient_utils.c" a new stepsize controller is used. Normally, the stepratio is computed on the next manner:

```
p_tr_cb->step_factor[p_tr_cb->level_ind] = F_pow ( *p_ratiosum,
-1.0/(p_tr_cb->int_order + p_tr_cb->level + 1.0), &status);
```

A new controller can be implemented, by means of adding the next commands:

```
PMI_put_real("barSn_1",*p_ratiosum);
PMI_put_real("hn_1",p_tr_cb->timestep);
PMI_put_real("thetan_1",0.5 * p_tr_cb->release_timestep_factor);
PMI_put_real("TOL",0.4 * PAR_ABS_ACC_VOLTS);
PMI_put_real("k",p_tr_cb->int_order + p_tr_cb->level);
PMI_do_command("[Sn_1,omegan] =
MATLAB_stepcontrol(barSn_1,hn_1,thetan_1,TOL,k);");
PMI_get_real("omegan",&p_tr_cb->step_factor[p_tr_cb->level_ind]);
```

Note that skipping the last line means that the original stepsize controller of Pstar is used.

While `hn_1` is not accepted yet, `omegan` is already computed. Only if `barSn_1 < RTOL`, `hn_1` can be stored. This can be done by adding the next commands at the end of the routine "transient_anal_step" in the file "transient.c", just above the command "update_timestep_table (); ":

```
PMI_do_command("Set_global(Sn_1,hn_1);");
```

With this command, the stepsize and error sequence are stored as global variables in the workspace of MATLAB. These variables can be stored, by means of adding the next commands in the routine "perform_analysis" in the file "transient.c", just after the large `while`– loop:

```
PMI_do_command("Store_history;");
```

To make it possible to call the MATLAB-functions, the directory of these scripts must be added to the path.

```
export PSTAR_MATLAB_DATA=
/home/averhoev/matlab/Scripts/Pstar_matlab_stepsizecontrol
```

The function "Matlab_stepcontrol.m" has the next structure:

```
function [Sn_1,omegan] = MATLAB_stepcontrol(barSn_1,...);

global h S

Sn_1  = thetan_1 * TOL * barSn_1;
```

```
epsilon = thetan_1 * TOL;

htemp   = [h,hn_1];
Stemp   = [S,Sn_1];

i = length(htemp);

kkI = 1;
kkP = 0;

hn

if (i > 1)
  hn = PIcontrol(epsilon,htemp(i),Stemp(i),Stemp(i - 1),kkI,kkP,k + 1);
else
  hn = PIcontrol(epsilon,htemp(i),Stemp(i),epsilon,kkI,kkP,k + 1);
end

omegan  = hn / htemp(i);
```

Here, the script "PIcontrol.m" is the implementation of a stepsize controller.

```
function y = PIcontrol(epsilon,hn_1,rn_1,rn_2,kkI,kkP,k);

kI = kkI / k;
kP = kkP / k;

y  = (epsilon / rn_1)^kI * (rn_2 / rn_1)^kP * hn_1;
```

To be able to store `hn_1` and `Sn_1` in the workspace of MATLAB, the function "Set_global.m" is called:

```
function Set_global(Sn_1,hn_1);

global h S

h = [h,hn_1];
S = [S,Sn_1];
```

To store these global variables, "Store_history.m" is used.

```
function Store_history;

global h S

save filename h S
```

After running pstar, `h` and `S` can be retrieved in MATLAB with "`load filename`".

B.3 Other possibilities

Besides the local errors and the stepsizes, it is also possible to export the orders, the number of Newton iterations and other statistics to MATLAB. Afterwards, it is easy to analyse these data in MATLAB. If a large test set of stepsize controllers has to be tested, this can easily be done by means of a Unix shell script. The tested controller is indicated by a variable TESTCASE, which is exported. In Pstar, this variable can be retrieved and exported to MATLAB. There, the function "Matlab_stepcontrol.m" will be also dependent on this variable. With a switch statement, the corresponding control action will be performed. The function "Store_history.m" will store the global variables in an unique file, which corresponds with this testcase. The same can also be performed, if one wants to test a large group of testcircuits. Then, also the variable TESTCIRCUIT is exported.

Appendix C

Implementation of NDF-method in Pstar

C.1 Implementing NDF-methods in Pstar

The subject of this Appendix is to describe how in the Philips circuit simulator Pstar the BDF time integration method (Backward Differentiation Formula) can be enhanced to perform as the corresponding NDF time integration method[1] (Numerical Differentiation Formula) [32].
The methods will be denoted by $\mathrm{BDF}_{\mathrm{Pstar}}$ and $\mathrm{NDF}_{\mathrm{Pstar}}$.
The differential-algebraic equation (DAE) to solve is:

$$\begin{cases} \frac{d}{dt}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0} \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{C.1}$$

C.1.1 BDF in Pstar

The BDF-method as implemented in Pstar is based on:

$$\begin{aligned} \mathrm{BDF}_{\mathrm{Pstar}}(\mathbf{x}_n) &= \frac{1}{h}\sum_{m=0}^{k} \rho_m \mathbf{q}(t_{n-k+m}, \mathbf{x}_{n-k+m}) + \mathbf{j}(t_n, \mathbf{x}_n) \\ &= \frac{\rho_k}{h}\mathbf{q}(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) + \alpha_n \\ &= \lambda * \mathbf{q}(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) + \alpha_n \qquad \text{(C.2)} \\ \alpha_n &= \frac{1}{h}\sum_{m=0}^{k-1} \rho_m q(t_{n-k+m}, x_{n-k+m}) \qquad \text{(C.3)} \\ \lambda &= \frac{\rho_k}{h} \qquad \text{(C.4)} \end{aligned}$$

[1] This adapted integration method has been investigated in subsection 3.4.4.

Denoting the exact solution of (C.1) by $\mathbf{x}^*(t)$, the discretization error $\Delta^{\text{BDF}}_{\text{Pstar}}$ is based on

$$\begin{aligned}
\Delta^{\text{BDF}}_{\text{Pstar},n} &= \frac{1}{h}\delta^{\text{BDF}}_n \\
&= \frac{1}{h}\text{BDF}_{\text{Pstar}}(\mathbf{x}^*_n) \\
&= \tilde{C}_{k,k}h^k\mathbf{q}^{(k+1)}(t_n, \mathbf{x}^*_n) \\
&= \frac{C_{k,k}}{h}[\mathbf{q}(t_n, \mathbf{x}^*_n) - \mathbf{q}^0_n] \qquad \text{(C.5)} \\
&\approx \frac{C_{k,k}}{h}\frac{\delta\mathbf{q}}{\delta\mathbf{x}}[\mathbf{x}^*_n - \mathbf{x}^0_n], \quad \text{where} \qquad \text{(C.6)} \\
C_{k,k} &= -1/(k+1). \qquad \text{(C.7)}
\end{aligned}$$

Here $\mathbf{q}^0_n$ and $\mathbf{x}^0_n$ have been obtained by k-th order extrapolation exploiting the Nordsieck-vectors for $\mathbf{q}$ and for $\mathbf{x}$, respectively.

Relation to Pstar datastructure fields

The next table relates quantities to fields in Pstar's datastructure

h	`p_tr_cb->timestep`
λ	`p_tr_cb->laplace_var = 1/p_tr_cb->timestep * BDF[k][1]`
$C_{k,k}$	`p_tr_cb->BDF_coeff[int_order][int_order] * const_factorial_table[int_order]`

The term α_n (`alpha_n`) and the extrapolated value $\mathbf{q}^0_n$ (`q_n^{0}`) are elegantly determined by exploiting the Nordsieck-vectors for $\mathbf{q}$.
Nordsieck-vectors for $\mathbf{q}$ of (reactive) elements are used as:

```
p_tr_info    = p_element->ext.ndr.p_tr_info
p_nord_vect  = p_tr_info->p_pred_nords_vect
q_n^{0}      = p_nord_vect[0]
alpha_n      = p_nord_vect[1] / p_tr_cb->timestep -
               p_nord_vect[0] * p_tr_cb->laplace_var
p_tr_info->int_value = alpha_n
q_n          = p_tr_info->output_value
```

Here, the value $\mathbf{q}_{n,i}$ (`q_n`) for the i-th local **reactive element** has been calculated after Newton convergence by a recursive call to SRE_eval_out_var and is thus available for estimation of the discretization error.

For **devices** (transistor models), the corresponding quantities are found as follows (cf in module transient_utils.c: control_init, control_predictor):

```
p_anal_device = ...
p_device      = p_anal_device->p_min_device
n_dev_react_elems = p_device->p_def->n_react_elem
p_tr_dev_info = &(p_anal_device->p_transient_info[0])
for (i=0;
```

```
        i<n_dev_react_elems ;
        p_tr_dev_info++, i++) {

        predict_nordsieck( ... )
        p_nord_vect = p_tr_dev_info->p_react_info->p_pred_nords_vect
        q_n^{0}     = p_nord_vect[0]
        \alpha_n    = p_nord_vect[1] / p_tr_cb->timestep -
                      p_nord_vect[0] * p_tr_cb->laplace_var
        p_tr_dev_info->int_value = \alpha_n
        q_n         = p_tr_dev_info->p_output_value
        [is automatically updated when evaluating reactive branches in
        a device]
}
```

Note that for devices, $\mathbf{q}_n$ contains the value $\mathbf{q}(t_n, \mathbf{x}_n^{j-1})$ where j is the Newton iteration counter and where $\mathbf{x}_n^j$ is the converged value. Because the occurrence of devices always causes the hierarchical branches that contain them, to be considered as non-linear, the deviation between $\mathbf{q}(t_n, \mathbf{x}_n^{j-1})$ and $\mathbf{q}(t_n, \mathbf{x}_n^j)$ may be assumed to be acceptibly small.

C.1.2 NDF in Pstar

In subsection 3.4.4, it has been shown that for the next dynamic coefficient κ_n, the LTE is multiplied by the factor λ.

$$\kappa_n = (p+1)\frac{h_n}{t_n - t_{n-m}}\kappa, \quad \kappa = \frac{\lambda - 1}{p+1}. \tag{C.8}$$

However, because Pstar uses the Vdiff-norm and another error estimate, something may be different.

The NDF-method to be implemented in Pstar will be based on:

$$\begin{aligned}
\mathrm{NDF}_{\mathrm{Pstar}}(\mathbf{x}_n) &= \mathrm{BDF}_{\mathrm{Pstar}}(\mathbf{x}_n) - \frac{\kappa_n}{h}.(\mathbf{q}_n - \mathbf{q}_n^0) \\
&= \frac{1}{h}\sum_{m=0}^{k} \rho_m \mathbf{q}(t_{n-k+m}, \mathbf{x}_{n-k+m}) + \mathbf{j}(t_n, \mathbf{x}_n) - \frac{\kappa_n}{h}.(\mathbf{q}_n - \mathbf{q}_n^0) \\
&= \frac{\rho_k - \kappa_n}{h}\mathbf{q}(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) + \beta_n \\
&= \lambda * \mathbf{q}(t_n, \mathbf{x}_n) + \mathbf{j}(t_n, \mathbf{x}_n) + \beta_n, \quad \text{where} \qquad \text{(C.9)} \\
\beta_n &= \frac{1}{h}\sum_{m=0}^{k-1} \rho_m q(t_{n-k+m}, x_{n-k+m}) + \frac{\kappa_n}{h}\mathbf{q}_n^0 \qquad \text{(C.10)} \\
&= \alpha_n + \frac{\kappa_n}{h}\mathbf{q}_n^0 \qquad \text{(C.11)} \\
\lambda &= \frac{\rho_k - \kappa_n}{h} \qquad \text{(C.12)}
\end{aligned}$$

The discretization error $\Delta^{\text{NDF}}_{\text{Pstar}}$ is based on

$$\begin{aligned}
\Delta^{\text{NDF}}_{\text{Pstar},n} &= \Delta^{\text{BDF}}_{\text{Pstar},n} - \frac{\kappa_n}{h}.(\mathbf{q}_n - \mathbf{q}^0_n) \\
&\approx \frac{C_{k,k} - \kappa_n}{h}[\mathbf{q}(t_n, \mathbf{x}^\star_n) - \mathbf{q}^0_n] && \text{(C.13)} \\
&\approx \frac{C_{k,k} - \kappa_n}{h}\frac{\delta \mathbf{q}}{\delta \mathbf{x}}[\mathbf{x}^\star_n - \mathbf{x}^0_n] && \text{(C.14)} \\
C_{k,k} &= -1/(k+1). && \text{(C.15)}
\end{aligned}$$

Some remarks apply when modifying Pstar code for BDF to allow for NDF.

- At some actions concerning Nordsieck-vectors (prediction, correction), the parameter `laplace_var` should have the same value as for BDF.
- Clearly $\beta_n = \alpha_n + \frac{\kappa_n}{h}\mathbf{q}^0_n$, where α_n is determined by the Nordsieck-vectors of the BDF-method.
- When performing matrix assembly it is convenient to set `laplace_var` to `laplace_var` - $\frac{\kappa_n}{h}$.
- When estimating the discretization error a term $-\frac{\kappa_n}{h}$ should be used in the coefficient as used for BDF.
- It is not possible to use NDF in the first step, or when starting from a discontinuity. In Pstar, this is flagged in `tr_cb.disc_timestep`.

List of used abbreviations

Abbrevation	Description
CR	Constitutive Relation
KCL	Kirchhoff's Current Law
KVL	Kirchhoff's Voltage Law
DC-analysis	Direct Current analysis: steady-state solution
AC-analysis	Alternating Current analysis: solution of linearized system for a small sine-wave excitation.
TR-analysis	Transient analysis
PSS-analysis	Periodic Steady-State analysis
DAE	Differential Algebraic Equation
ODE	Ordinary Differential Equation
IVP	Initial Value Problem
LMM	Linear Multistep Method
RK	Runge Kutta
BDF	Backward Difference Method
NDF	Numerical Difference Method: adapted BDF-method with a smaller error constant
SRP	Step Ratio Percent for the NDF-method
TR-BDF2	Combination of the Trapezoidal method and the BDF2-method
LDE	Local Discretization Error
EPS	Error Per Step
EPUS	Error Per Unit Step
I-control	Integral control
PI-control	Proportional Integral control
PID-control	Proportional Integral Derivative control
PC-control	Predictive PI-Control

List of frequently used symbols

Symbols	Description
v, V, i	voltage difference, nodal voltage and current
$t, \mathbf{x}$	time and state vector of circuit
$\mathbf{q}, \mathbf{j}, C, G$	charge and current functions and their Jacobians of a circuit
$\mathbf{f}, J$	function and its Jacobian, which represents a general ODE
$t_n, h_n, \mathbf{x}_n$	n-th timepoint timestep and numerical approximation
$\mathbf{d}_n, \delta_n, \mathbf{e}_n$	local error, local truncation error and global error at timepoint t_n.
$\hat{\delta}_n$	estimate of LTE
ν, μ_n	global index and local index at t_n
Γ, ρ, β	matrix and vectors, which describe a Runge Kutta method
$S, \partial S$	stability region and its boundary of an integration method
$\rho(z), \sigma(z)$	polynomials, which represent an LMM method
q	shift operator
$C_{p,n}$	error constant for integration method of order p at t_n
$\xi_{n,m}$	useful coefficient, such that $t_n - t_{n-m} = \xi_{n,m} h_n$
$C_{q,n}, C_{j,n}, P_{q,n}, P_{j,n}$	Corrector and predictor polynomials for $\mathbf{q}$ and $\mathbf{j}$
$\bar{C}_{q,n}, \bar{C}_{j,n}, \bar{P}_{q,n}, \bar{P}_{j,n}$	Nordsieck vectors of $C_{q,n}, C_{j,n}, P_{q,n}, P_{j,n}$
$L_n, \bar{L}_n$	Lagrange interpolation polynomial and its Nordsieck vector
$\mathbf{q}_f, \mathbf{j}_f, \mathbf{x}_f, \mathbf{q}_s, \mathbf{j}_s, \mathbf{x}_s$	$\mathbf{q}, \mathbf{j}$ and $\mathbf{x}$ for fast (active) and slow (latent) parts of circuit
TOL, θ, ϵ	tolerance level, safety factor and reference level for controller
k, p	number of steps and order of an integration method
$\hat{r}_n$	controlled error, often based on the norm of $\hat{\delta}_n$
$\hat{\phi}_n, \hat{\phi}_n^P$	disturbance and its prediction
P	order of the controlled error $\hat{r}_n$ for EPS and EPUS control.
k_I, k_P, k_E, k_R	control parameters of PI-controller and PC-controller
r	pole of closed loop dynamics
p_A, p_F, p_R	order of adaptivity, stepsize filter order and error filter order
$G(z), L(z), K(z)$	transfer function of process model, with $G(z) = \frac{L(z)}{K(z)}$
$C(z), B(z), A(z)$	transfer function of process model, with $C(z) = \frac{B(z)}{A(z)}$
N, M	degrees of $A(z)$ and $K(z)$
$\bar{A}(z)$	polynomial for adaptive controller with $A(z) = (z-1)\bar{A}(z)$
$\bar{\alpha}_n, \beta_n$	coefficients of $\bar{A}(z)$ and $B(z)$ and control parameters
$R(z)$	characteristic polynomial with $R(z) = A(z)K(z) + B(z)L(z)$
$S(z)$	polynomial with $S(z) = A(z)K(z)$
$\sigma_n, \rho_n, \alpha_n$	coefficients of $S(z)$, $R(z)$ and $A(z)$ and control parameters
$H_{p_A,p_F,p_R}(r_1, \ldots, r_{N+M})$	short notation for digital linear controller

Bibliography

[1] K.J.AASTROEM, T.HAEGGLUND *PID Controllers* Research Triangle Park: Instrument Society of America, 1995.

[2] T.S.APPEL *A new timestep control in the circuit simulation package TITAN.* Bachelor thesis, 2000.

[3] A.BARTEL *Generalised Multirate: Two ROW-type Versions for Circuit Simulation.* MSc Thesis, TU Darmstadt & IWRMM Universität Karlsruhe, 2000.

[4] C.BOMHOF *Iterative and parallel methods for linear systems, with applications in circuit simulation.* PhD Thesis, Utrecht University, 2000.

[5] K.E.BRENAN, S.L.CAMPBELL, L.R.PETZOLD *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* SIAM, 1996.

[6] S.M.A.BRUIN. *Modified extended bdf applied to circuit equations.* MSc thesis, Vrije Universiteit Amsterdam, 2001. Report, Philips ED&T/Analogue Simulation, 2001.

[7] L.O. CHUA, P.M. LIN *Computer aided analysis of electric circuits: algorithms and computational techniques*, first ed. Prentice Hall, 1975.

[8] D.ESTÉVEZ SCHWARZ *Consistent initialization for index-2 differential algebraic equations and its application to circuit simulation.* PhD Thesis, Humboldt-Universität zu Berlin, 2000.

[9] J.G.FIJNVANDRAAT, E.J.W.TER MATEN *Transient analysis with Pstar.* Report, Philips ED&T/Analogue Simulation, 2000.

[10] J.G.FIJNVANDRAAT, S.H.M.J.HOUBEN, E.J.W.TER MATEN, J.M.F.PETERS *Time domain analog circuit simulation.* Article, Journal of Computational Methods in Sciencs and Engineering, 2003.

[11] C.W.GEAR *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice Hall, 1971.

[12] C.W.GEAR, D.R.WELLS *Multirate Linear Multistep Methods.* BIT 24:484:502, 1984.

[13] K.GUSTAFSSON *Control Theoretic Techniques for Stepsize Selection in Explicit Runge-Kutta Methods.* ACM TOMS 17:533-554, 1991.

[14] K.GUSTAFSSON *Control Theoretic Techniques for Stepsize Selection in Implicit Runge-Kutta Methods.* ACM TOMS 20:496-517, 1994.

[15] K.GUSTAFSSON, G.SÖDERLIND *Control strategies for the iterative solution of nonlinear equations in ode solvers* SIAM J.Sc.Comp. 18:23-40, 1997.

[16] E.HAIRER, S.P.NØRSETT,G.WANNER *Solving Ordinary Differential Equations I.* Springer-Verlag, 1987,

[17] H.J.C.HUIJBERTS ANALYTISCHE ASPECTEN VAN GEWONE DIFFERENTIAALVERGELIJKINGEN Syllabus TU/e.

[18] M.HAUTUS SYSTEEMTHEORIE Syllabus TU/e, 2000.

[19] M.HONKALA, J.ROOS, M.VALTONEN *New multilevel Newton-Raphson method for parallel circuit simulation* Article, Helsinki University of Technology, 2001.

[20] M.E.HOSEA, L.F.SHAMPINE *Analysis and implementation of TR-BDF2.* Applied Numerical Mathematics 20, pp 21-37, 1996.

[21] S.H.M.J.HOUBEN *Algorithms for Periodic Steady State Analysis on Electric Circuits.* MSc thesis, Technische Universiteit Eindhoven, 2 1999. Report, Philips ED&T/Analogue Simulation, 1999.

[22] S.H.M.J.HOUBEN *Circuits in motion, The numerical simulation of electrical oscillators.* PhD Thesis, Technische Universiteit Eindhoven, 2003.

[23] R.ISERMANN *Digital Control Systems* Springer-Verlag, 1981.

[24] B.C.KUO *Digital Control Systems* Holt-Saunders International Editions, 1980.

[25] R.M.M.MATTHEIJ *Inleiding Numerieke Analyse* Syllabus TU/e, 1998.

[26] R.M.M.MATTHEIJ, J.MOLENAAR *Ordinary differential equations in theory and practice.* Wiley, 1996.

[27] H.G.TER MORSCHE *Wiskundige methoden in de signaalverwerking.* Syllabus TU/e.

[28] H.NIJMEIJER, A.J.VAN DER SCHAFT *Nonlinear Dynamical Control Systems* Springer-Verlag, 1990.

[29] J.M.ORTEGA *Numerical Analysis, a second course.* Academic Press, 1972.

[30] J.ROMMES *Jacobi-davidson methods and preconditioning with applications in pole-zero analysis.* MSc thesis, Utrecht University, 05 2002. Report, Philips ED&T/Analogue Simulation, 2002.

[31] L.F.SHAMPINE *Numerical solution of ordinary differential equations.* Chapman & Hall, 1994.

[32] L.F.SHAMPINE, M.W.REICHELT *The MATLAB ode suite.* SIAM J.N.Scient.Comp. Vol.18-1,pp1-22,1997.

[33] A.SJÖ *Analysis of computational algorithms for linear multistep methods.* PhD Thesis, Lund University, 1999.

[34] G.SÖDERLIND *Automatic control and adaptive time-stepping* Article, 2001.

[35] G.SÖDERLIND *Digital filters in adaptive time-stepping* ACM Tr. on Math.Softw.,Vol.V,No.N,pp 1-24, Sept. 2000.

[36] C.TISCHENDORF *Solution of index-2 differential algebraic equations and its application in circuit simulation.* PhD Thesis, Humboldt-Universität zu Berlin, Logos Verlag Berlin, 1996.

[37] M.E.V.VALKENBURG *Network analysis*, third ed. Prentice Hall, 1974.

[38] A.VERHOEVEN To be published

[39] A.W.WESTERBERG, P.C.PIELA *Equational-based process modeling.* Carnegie Mellon University, 1994.

[40] M.C.J. VAN DER WIEL *Numerical time integration techniques for circuit simulation in Pstar.* Report, Philips ED&T/Analogue Simulation, 1992.