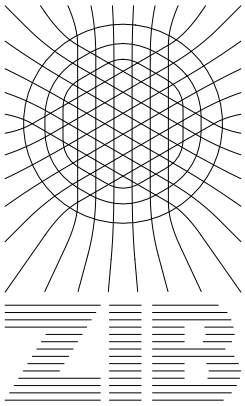
Konrad-Zuse-Zentrum für Informationstechnik Berlin



Andreas Hohmann

Inexact Gauss Newton Methods for Parameter Dependent Nonlinear Problems

TR 93-13 (December 1993)

Inexact Gauss Newton Methods for Parameter Dependent Nonlinear Problems

Andreas Hohmann

Konrad–Zuse–Zentrum für Informationstechnik Berlin, Heilbronner Strasse 10,
D-10711 Berlin-Wilmersdorf, Federal Republic of Germany

Abstract

A new approach to inexact Gauss Newton methods for the solution of underdetermined nonlinear problems is presented. It is based on a convergence theorem being invariant under affine transformations of the domain and results in an easily implementable accuracy matching strategy for the arising linear subproblems which guarantees the quadratic convergence. Thanks to the weak assumptions on the given nonlinear problem, the results provide a general framework for multilevel Newton and continuation methods. As an example, a new multilevel Newton h-p collocation method for boundary value problems of ordinary differential equations is developed. It combines the inexact Newton method with a linear collocation solver using adaptive refinement and variable orders. The performance of the resulting C++ class library Cocon is demonstrated by some numerical examples including singular perturbed problems. In addition, the new method is applied to a realistic railway bogie model in which a branch of periodic solutions emanates from a branch of fixed points at a Hopf bifurcation.

Contents

Introduction

It seems beyond doubt that Newton's method (often called Newton-Raphson method, particularly in the engineering literature) is the most successful method for the numerical solution of nonlinear problems provided with some differentiability. Because its idea of successive linearization is so fundamental, there are many possible applications. Consequently, any list of these possibilities must remain incomplete. We only refer to the numerical analysis of dynamical systems regarding fixed points, periodic solutions, and bifurcations (see e.g. Allgower and Georg [1], Doedel [35] [36], Keller [46], Parker and Chua [52], Rheinboldt [55], Seydel [60] to name but a few) and computational mechanics (see e.g. Crisfield [17], Zienkiewicz and Taylor [69] and the references herein).

The huge range of applications has attracted much interest in mathematical and engineering research. Thus, many fundamental insights like the (local) quadratic convergence are common knowlegde to be found in any textbook about numerics. Moreover, there are dozens of variations of Newton's method invented to reduce the effort expended on the classical iteration, such as the simplified Newton method and various updating techniques for the Jacobian.

As the convergence of these *local* methods is restricted to a sometimes small neighbourhood of the solution, *global* generalizations have been developed. The two most prominent are the damped Newton method and the continuation methods, where the latter not only enlarge the region of convergence but also allow new applications such as parameter studies and bifurcation analysis. Further, continuation (in combination with some Newton process) is widely accepted and part of many numerical software packages.

Convergence theorems for Newton's method and its derivates mainly exist in two kinds of formulations (mathematically almost equivalent). The more 'classical' formulation uses Lipschitz constants for the Jacobian and bounds on its inverse and the initial guess. For an excellent and rather complete collection of convergence theorems of this kind we refer to the textbook by Ortega and Rheinboldt [51]. As an alternative, the convergence theorems may be formulated in *affine invariant* terms as introduced by Deuflhard and Heindl [32]. These formulations employ computationally available (or

estimable) terms that rule the convergence behaviour. We shall discuss this point in more detail in part I.

In this thesis we shall deal with inexact Newton methods. The basic idea is simple and, of course, not new at all. Considering a large nonlinear problem, we have to solve in each step of the Newton process a (large) linear problem. But it appears to be wasted time to solve all these linear problems exactly, since the Newton iterates themselves are only approximations of the solution. So, we would expect that for iterates far from the solution it is sufficient to solve the associated linear problem rather inaccurately, whereas we have to spend more and more effort as the Newton iterates approach the solution. Thus, the principle of inexact Newton methods is to solve the linear subproblems only as accurately as necessary. By a careful choice of the accuracies for the linear subproblems it is even possible to maintain the quadratic convergence of Newton's method.

Although the idea is very simple and already theoretically quite well understood (see e.g. Dembo, Eisenstat and Steihaug [22]), it is surprisingly hardly ever applied. In addition, the few applications (see e.g. Bank and Rose [10], [11], Bank and Chan [8], Bank and Mittelmann [9]) reveal a gap between the underlying analytic results and the formulae used in the actual codes. In our oppinion this situation has mainly two explanations. Firstly, the theoretical results are often not suitable for application. The crucial point for an inexact method is that the algorithm has to be able to control itself. This means that the theory has to provide simple formulae for the accuracies for the linear subproblems that only depend on cheaply available information. But the typical analytic constants used in the convergence theory cannot be cheaply computed in the actual algorithm for the most interesting classes of problems.

Secondly, inexact Newton methods make no sense for fixed discretizations and direct solvers prevailing in the huge software packages in engineering. For an inexact method to be efficient, weak accuracy requirements for the linear subproblems must pay off in some way. One way is to employ different discretizations which, in turn, have to be finer the more accuracy is required which leads to more and more degrees of freedom. Alternatively, or in addition, we may use iterative solvers for the arising (large and often sparse) linear systems.

On the other hand there are many applications where the inexactness is

just inavoidable. As an example, the integrators used in multiple shooting always introduce a discretization error that has to be controlled in some way. Nonetheless many standard codes use rather crude heuristics (like 0.1 times the required accuracy for the nonlinear problem). A more satisfactory control mechanism for multiple shooting has been developed by Bock [13] [14] in connection with the parameter identification code PARFIT. It is based on the heuristic that the error of the inexact Newton correction has to be below the norm of the next correction and needs estimates for the norm of the Jacobian.

The preceding discussion leads us now to the formulation of a clear program of what is required to fully exploit the idea of an inexact Newton method. This may be summerized as follows:

- We have to analyze the inexact Newton method with regard to an algorithmic realization and a wide range of applications.
- This analysis should result in an accuracy matching mechanism for the linear subproblems based on computationally available terms.
- Moreover, we have to derive cheap and sharp estimates for the required problem dependent constants.
- The results should be applicable to realistic ('real life') problems.
- In view of the wide range of applications the inexact method must certainly be easy to embed in a continuation framework.

The first point means that we have to use weak (realistic) assumptions (like differentiability in a weak sense) on the given problem. The last claim reminds us to keep an eye on parameter dependent problems which are in fact *underdetermined* if viewed as nonlinear equations.

Having referred to the technique of using different discretizations to exploit the weak accuracy requirements, we already touched quite an exciting application of inexact Newton techniques. Why not apply the results to the infinite dimensional problem and solve the (still infinite dimensional) linear subproblems to the accuracy prescribed by the surrounding inexact iteration? This so-called *quasilinearization* or *multilevel Newton* approach to nonlinear problems seems to be very attractive since it separates the linear from the

nonlinear task. If we know how to solve linear problems of a particular kind, the multilevel Newton method tells us how to solve nonlinear ones without altering the linear solver. Although tempting, this idea has been infrequently applied (see, e.g., Scott and Watts [57] who combine quasilinearization with a shooting technique for the linear problems) which may be due to the accuracy matching problem for inexact Newton methods. Note that in this infinite dimensional context, weak assumptions on the given problem are even more important, since they decide whether the inexact method is applicable or not.

As a class of problems to be solved by the new techniques, we have chosen boundary value problems for ordinary differential equations (for short: BVPs for ODEs). On the one hand, this class is simple enough with regard to the programming effort needed for a new method. On the other hand, it already has a rich structure and includes a wide range of applications. Of particular interest for our investigations were periodic boundary conditions for autonomous ODEs since they present a natural continuation of preliminary research concerning stationary solutions of dynamical systems (see [39]).

As discretization methods, we use multiple shooting and collocation with variable orders and stepsizes (so-called h-p collocation). Regarding multiple shooting we only sketch some theoretical questions since numerical results are already available in the diploma thesis by C. Wulff [66]. The new h-p collocation method consequently realizes a multilevel Newton method applied to the nonlinear integral equations describing the BVP.

The thesis is divided into three parts devoted to the inexact Newton theory, the h-p multilevel Newton collocation method and the numerical results, respectively. Part I introduces the abstract framework of the inexact Newton method. We give in particular two different formulations, *affine invariant* and (as a new concept) *affine contravariant*. Part I also contains two examples confirming the underlying analytic results: the application of the inexact Newton method to multiple shooting, and a very simple facility for implicit discretization methods that increases their robustness drastically. Part II gives a complete account of the multilevel Newton h-p collocation method: the integral formulations of the given BVP, the rather technical computation of local and global collocation solutions, the h-p and multilevel Newton strategies, as well as the multilevel continuation process. Part III, finally, sketches some ideas concerning the often neglected implementation issue,

and demonstrates the numerical impact of the new techniques. The method was realized as a C++ class library called COCON, incorporating the solver for parameter dependent BVPs, the continuation method and the handling of Hopf bifurcations. As a 'real life' problem, we compute the limit cycles of a railway bogie model, presenting the classical situation of a branch of periodic solutions emanating from a branch of equilibria at a Hopf bifurcation.

A note on notation. In order to avoid an inflation of symbols and indices we often use the same letter (even in the same section) for different matters, if the meaning is clear from the context. As an example, the letter b not only denotes the right boundary of the (global or local) interval, but also the weight vector of a Runge Kutta scheme (where we adopted the standard notation from numerical integration). For convenience we listed most symbols on page 110.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to P. Deuflhard whose support and encouragement made this work possible. Being an abolute newcomer in numerics when I started working at the Zuse Center (ZIB), he influenced my attitude towards scientific computing from the very beginning. It is his dream of a general purpose multilevel Newton method that is behind the new algorithm. Moreover, it required only a rather short step from his affine invariant convergence theorems to arrive at the results presented in this thesis. He certainly will forgive me the use of residuals rather than errors in the multilevel Newton strategy.

I also thank U. Ascher and G. Bader for intense and very informative discussions during their visits at the Zuse Center this summer which helped me a lot to envision the pros and cons of the new collocation method.

Many thanks to the staff at the ZIB, in particular F. Bornemann, for leaving no analytical question unanswered; M. Wulkow (now CiT), for discussing the h-p strategy and sharing some classes; G. Zumbusch, for also discussing the h-p issue; C. Wulff (now FU Berlin), for our joint work on periodic solutions; K. Gatermann, for our joint work on symmetry exploitation and for taking many tasks off my shoulders; and, last but not least, C. Schütte, for

many helpful ideas and for enduring my impatience.

I. Newton's Method

Newton's method is well accepted as the method of choice for a large variety of nonlinear problems such as boundary value problems of ODEs (multiple shooting or collocation) or PDEs (finite elements). In many applications it is combined with some continuation process which not only makes it a global method in contrast to the local convergence of Newton's method, but also allows the examination of the qualitative behaviour of the problem depending on parameters.

Often the Newton correction cannot be computed directly but has to be approximated. This may be due to the fact that the nonlinear function itself or its Jacobian are only approximately at hand, or, because the arising linear system is too large to be solved directly, so that an iterative method has to be employed. Therefore, the question arises, which conditions have to be imposed on the approximate Newton corrections in order to preserve the good convergence properties of Newton's method. Looking at an algorithmic realization, we would like to know how to control the accuracy of the inexact Newton corrections to obtain the desired speed of convergence. Thus, our real aim is an accuracy matching for the inexact Newton method based on algorithmically available terms.

Inexact Newton methods have been attacked by many authors, we only mention Bank and Rose [10], Dembo, Eisenstat and Steihaug [22], Ypma [68], Deuflhard [28] and the references therein. Dembo et. al. obtained precise results on how to control the relative residual of the correction equation to obtain a prescribed order $1 < q \leq 2$ of convergence. Unfortunately, they use problem dependent constants, such as bounds for the Jacobian and its inverse, that in many applications cannot be estimated algorithmically or tend to grow with the dimension of the problem (e.g., for successively finer discretizations of nonlinear PDE's).

Deuflhard and Heindl start with a convergence theorem for Newton's method that is invariant with respect to linear transformations of the image space of the nonlinear mapping and therefore only uses the norm in the domain. Due to this *affine invariant* approach, it is possible to give bounds for the relative error of the Newton correction which can be estimated algorithmically. In [28] these methods are extended to the damped (or global) Newton

method. The main difficulty of this approach is the control of the relative error, which is in many cases much more complicated than the control of the relative residual.

We shall try to combine the advantages of both approaches to obtain algorithmically available bounds for the relative residuals. To this end, we only have to transfer the affine invariant theorems into results which are invariant with respect to affine transformations of the domain and therefore only use the norm given in the image space.

This part is organized as follows. In the first section we discuss the invariance properties of Newton's method with respect to affine transformations of the domain and image space. This will lead us in Section 2 to so-called *affine contravariant* convergence theorems which we compare to the affine invariant formulations. In Section 3 we show how these theorems transfer to inexact Newton methods. Both invariant formulations directly lead to computationally available estimates for the problem dependent constants. Instead of trying to develop a damped variant of the theory for highly nonlinear problems, we present in Section 4 a continuation process for parameter dependent problems based on the inexact Newton method for underdetermined nonlinear equations.

1 Invariance Properties

We want to solve a nonlinear problem

$$F(x) = 0 \ ,$$

where F is a differentiable mapping. To simplify the presentation in this section, we only consider the finite dimensional situation, i.e.,

$$F : \mathbb{R}^n \longrightarrow \mathbb{R}^m \ ,$$

although all considerations transfer to a mapping of an open subset $D \subset X$ of a Banach space X in another Banach space Y. In order to solve the nonlinear equation we apply Newton's method

$$x_{k+1} = x_k + \Delta x_k \quad \text{for } k = 0, 1, \ldots,$$

for some initial guess x_0, where the *Newton correction* Δx_k satisfies the *Newton equation*

$$F'(x_k)\Delta x_k = -F(x_k) \quad \text{for } k = 0, 1, \ldots. \tag{1.1}$$

If $F'(x_k)$ is invertible, the Newton correction is uniquely determined by (1.1). It may also occur, that the Newton equation results in an underdetermined linear system, so that we have to choose a particular solution by an additional normalization requirement. In fact, we will meet this situation in the continuation methods discussed in Section 4.

Before analyzing Newton's method, we would like to draw attention to the affine invariance properties of the nonlinear problem. To this end let $\mathrm{Aff}(\mathbb{R}^n)$ denote the ring of affine isomorphisms of $\mathbb{R}^n$, i.e., each transformation $T \in \mathrm{Aff}(\mathbb{R}^n)$ is given by

$$T : \mathbb{R}^n \to \mathbb{R}^n, \quad Tx = Ax + b,$$

for some invertible linear mapping $A = T' \in \mathrm{GL}(n)$ and $b = T(0) \in \mathbb{R}^n$. Then $\mathrm{Aff}(\mathbb{R}^m)$ acts on the nonlinear mappings $F : \mathbb{R}^n \to \mathbb{R}^m$ by the covariant transformation

$$F \mapsto T_* F := T \circ F \quad \text{for } T \in \mathrm{Aff}(\mathbb{R}^m),$$

satisfying $(ST)_* = S_* T_*$ for $S, T \in \mathrm{Aff}(\mathbb{R}^m)$. On the other hand $\mathrm{Aff}(\mathbb{R}^n)$ acts by the contravariant transformation

$$F \mapsto T^* F := F \circ T \quad \text{for } T \in \mathrm{Aff}(\mathbb{R}^n),$$

satisfying $(ST)^* = T^* S^*$ for $S, T \in \mathrm{Aff}(\mathbb{R}^n)$. Obviously, the covariant transformation T_* does not alter the solution of the nonlinear problem $F(x) = 0$, since

$$F(x) = 0 \iff TF(x) = (T_* F)(x) = T(0) \; .$$

What is changed, is the solution's *characterization*. This is different from the contravariant transformation T^*. Since

$$F(x) = 0 \iff (T^* F)(T^{-1} x) = 0 \; ,$$

we do not change the nonlinear problem but the *representation* of its solution x.

Now let us turn to Newton's method and its invariance properties. Since the method is defined in terms of the linearization of F, it is in fact invariant

with respect to affine transformations of the domain and image space. More precisely, if we consider the covariant transformation T_* for $T \in \mathrm{Aff}(\mathbb{R}^m)$ and apply Newton's method to the transformed equation $T_*F(x) - T(0) = 0$, the Newton iterates x_k remain the same, since $(T_*F)' = T'F'$ and hence

$$F'(x)\Delta x = -F(x) \iff (T_*F)'(x)\Delta x = -(T_*F(x) - T(0)).$$

On the other hand, the iterates transform in the same way as the solution does, if we apply the contravariant transformation T^* for $T \in \mathrm{Aff}(\mathbb{R}^n)$, since $(T^*F)'(x) = F'(Tx)T'$ and therefore

$$F'(x)\Delta x = -F(x) \iff F'(x)A(A^{-1}\Delta x) = -F(x),$$

where $A = T' \in \mathrm{GL}(n)$.

2 Exact Newton Methods

Naturally, convergence theorems for Newton's method should reflect in some way its affine invariance properties. In other words, the characterizing quantities, such as bounds on the initial guess x_0 and the Jacobian F', should be invariant with respect to linear transformations of the domain or image space of F. Obviously, it is impossible to maintain both invariance properties, since the transformations change the norm of the spaces and thus the measure of convergence. Thus, we have two points of view: *affine invariant* and *affine contravariant*, preserving the invariance under the covariant and contravariant transformations, respectively. We shall see that both invariant formulations automatically lead to descriptions of the nonlinearity in computationally available terms. Accordingly, we present the convergence theorems hand in hand with the computational estimates for the involved characteristic constants which are used in the actual algorithms.

In the sequel, we always assume that *X and Y are Banach spaces and $D \subset X$ an open convex subset of X.*

2.1 Affine Invariant Approach

Deuflhard and Heindl [32] formulated a so-called *affine invariant* convergence theorem which is invariant with respect to transformations of the image space, i.e., to the covariant transformations T_* for $T \in \mathrm{Aff}(Y)$. In a simplified version the result by Deuflhard and Heindl reads as follows.

THEOREM 1. *Let $F : D \subset X \to X$ be a continously Fréchet-differentiable mapping, such that $F'(x)$ is continously invertible for all $x \in X$. Moreover, we require F' to meet the affine invariant Lipschitz condition*

$$\|F'(y)^{-1}(F'(x + t(y - x)) - F'(x))(y - x)\| \leq t\omega \|y - x\|^2 \tag{2.1}$$

for all $x, y \in D$, $t \in [0, 1]$ and some $\omega > 0$. If the initial guess x_0 satisfies

$$\omega \|\Delta x_0\| = \omega \|F'(x_0)^{-1} F(x_0)\| < 2 \, . \tag{2.2}$$

and the Newton iteration x_k stays in the domain D, then the iteration converges to a solution x of $F(x) = 0$. The convergence is quadratic in the sense

that

$$\|\Delta x_{k+1}\| \leq C \|\Delta x_k\|^2 .$$

for some constant $C \geq 0$.

Proof. For convenience we recall the proof. The whole key to almost all convergence proofs for Newton's method is the judicious application of the fundamental theorem of calculus. By the definition of the Newton correction, we have

$$\begin{aligned} F(x_{k+1}) &= F(x_k + \Delta x_k) - F(x_k) - F'(x_k)\Delta x_k \\ &= \int_0^1 (F'(x_k + t\Delta x_k) - F'(x_k))\Delta x_k \, \mathrm{dt} \ . \end{aligned} \tag{2.3}$$

Using the affine invariant Lipschitz condition (2.1), we can estimate the next Newton correction by

$$\begin{aligned} \|\Delta x_{k+1}\| &= \|F'(x_{k+1})^{-1}F'(x_k)\| \\ &\leq \int_0^1 \|F'(x_{k+1})^{-1}(F'(x_k + t\Delta x_k) - F'(x_k))\Delta x_k\| \, \mathrm{dt} \\ &\leq \frac{\omega}{2}\|\Delta x_k\|^2 . \end{aligned} \tag{2.4}$$

Introducing the so-called Kantorovitch quantities $h_k := \omega\|\Delta x_k\|$ this corresponds to

$$h_{k+1} \leq \frac{1}{2}h_k^2 . \tag{2.5}$$

Requiring $h_1 < h_0$ leads to the initial condition (2.2) which implies $h_{k+1} \leq \theta h_k$ with $\theta := h_0/2 < 1$. Hence, the Newton corrections converge quadratically to zero. This implies that the Newton iterates x_k form a Cauchy sequence converging to some $x_* := \lim_{k\to\infty} x_k \in X$ which obviously is a solution of F. □

REMARK 1. Using (2.5) it is easy to see that the Newton iteration stays in the ball of radius

$$\rho = \|\Delta x_0\| \sum_{j=0}^{\infty} \left(\frac{h_0}{2}\right)^{2^j-1} \leq \frac{\|\Delta x_0\|}{1 - h_0/2}$$

with center x_0. Thus, we may abandon the condition that the iteration stays in D by the assumption that D contains this ball.

Note that the Lipschitz constant ω for the Jacobian remains the same if we substitute F by the transformed mapping $T_*F = TF$. That is why we call it an *affine invariant* Lipschitz constant. The affine invariance implies that only the norm in the domain X is used.

Computational Estimates. To verify the claimed applicability of the affine invariant formulation, we have to derive cheap estimates for the characteristic constants, i.e., the Kantorovitch quantities h_k. According to the initial condition (2.2), they can be used to check the convergence of the iteration by the *affine invariant monotonicity test*

$$h_k < h_{\max} := 2 \; . \tag{2.6}$$

In addition, they may be employed to control the surrounding method, e.g., the stepsize of a continuation process (see Section 4 and [24]). Using the basic inequality (2.4) we may locally estimate ω and the Kantorovitch quantity h_k by

$$[\omega] := 2\frac{\|\Delta x_{k+1}\|}{\|\Delta x_k\|^2} \leq \omega \quad \text{and} \quad [h_k] := 2\frac{\|\Delta x_{k+1}\|}{\|\Delta x_k\|} \leq h_k \; . \tag{2.7}$$

Here, we follow Deuflhard [24] using brackets $[b]$ to denote a computationally available estimate of a quantity b.

REMARK 2. Note that these estimates are rather sharp since the nonlinearity is characterized by the variation of the directional derivative of F, where the directions are in fact the Newton corrections Δx_k. This property is lost in the affine invariant formulation of inexact methods (see Section 3.1).

Substituting the analytic quantity h_k in (2.6) by its computational estimate $[h_k]$, we arrive at the easily implementable monotonicity test

$$\|\Delta x_{k+1}\| < \|\Delta x_k\| \; . \tag{2.8}$$

We observe however that this monotonicity test unfortunately involves the Newton correction Δx_{k+1} of the next step which is obsolete if the convergence test fails. This fact motivated Deuflhard to introduce the so-called *simplified Newton corrections* $\bar{\Delta} x_{k+1}$ defined by

$$F'(x_k)\bar{\Delta} x_{k+1} = -F(x_{k+1}) \; .$$

If a direct solver is used, the effort to compute $\bar{\Delta}x_{k+1}$ is negligible compared to the effort expended on the Newton correction Δx_k, since the decomposition of the Jacobian $F'(x_k)$ is already at hand. Moreover, if we suppose that the Lipschitz condition (2.1) also holds for $F'(x)^{-1}$ in place of $F'(y)^{-1}$, the simplified correction again meets the inequality

$$\|\bar{\Delta}x_{k+1}\| \;\leq\; \frac{\omega}{2}\|\Delta x_k\|^2 \,.$$

As a consequence, we may substitute the next Newton correction Δx_{k+1} in the Kantorovitch estimate (2.7) and the monotonicity test (2.8) by its simplified counterpart $\bar{\Delta}x_{k+1}$ leading to the more favourable convergence check

$$\|\bar{\Delta}x_{k+1}\| < \|\Delta x_k\| \,. \tag{2.9}$$

If this condition is violated the iteration is stopped without wasting time on the computation of Δx_{k+1}.

2.2 Affine Contravariant Approach

We next formulate a so-called *affine contravariant* convergence theorem which preserves the invariance with respect to the contravariant transformations T^*, $T \in \mathrm{Aff}(X)$, of the domain. This approach implies the exclusive use of the norm in the image space Y. In fact, we do not prove convergence of the Newton iterates x_k, but only show that the Newton residuals $F(x_k)$ converge quadratically to zero.

THEOREM 2. *Let $F : D \subset X \to Y$ be a Gâteaux-differentiable mapping. Moreover, we assume that there is an $\omega > 0$ such that F' meets the affine contravariant Lipschitz condition*

$$\|(F'(y) - F'(x))(y - x)\| \leq \omega \, \|F'(x)(y - x)\|^2 \tag{2.10}$$

for all $x, y \in D$. If the initial guess x_0 satisfies

$$\omega \, \|F(x_0)\| < 2 \,. \tag{2.11}$$

and the Newton iteration x_k stays in D, then the residuals $F(x_k)$ converge quadratically to zero.

Proof. The proof is almost a copy of the previous one. Using (2.3) and the affine contravariant Lipschitz condition (2.10), we can estimate the next residual by

$$\begin{aligned} \|F(x_{k+1})\| &\leq \int_0^1 \|(F'(x_k + t\Delta x_k) - F'(x_k))\Delta x_k\| \, \mathrm{dt} \\ &\leq \frac{\omega}{2}\|F'(x_k)\Delta x_k\|^2 \\ &= \frac{\omega}{2}\|F(x_k)\|^2 \, . \end{aligned} \tag{2.12}$$

We now define the residual oriented Kantorovitch quantities h_k as $h_k := \omega\|F(x_k)\|$ and obtain again

$$h_{k+1} \leq \frac{1}{2}h_k^2 \, .$$

Requiring $h_1 < h_0$ now leads to the initial condition (2.11) which implies $h_{k+1} \leq \theta h_k$ with $\theta := h_0/2 < 1$. Hence, the Kantorovitch quantities and thereby the residuals converge quadratically to zero. □

Analogous to Theorem 1 the *affine contravariant* Lipschitz constant ω as defined by (2.10) remains as it was, if we substitute F by $T^*F = F \circ T$ for a transformation $T \in \mathrm{Aff}(X)$ of the domain. (Of course, we have to transform the domain D at the same time.)

We would like to emphasize the rather weak differentiability assumption on F compared to Theorem 1. We only require F to be Gâteaux differentiable and need neither the Fréchet derivative nor its inverse. This fact becomes important in the infinite dimensional context of a Newton multilevel method. Furthermore, note that the Newton correction $\Delta x_k = x_{k+1} - x_k$ may be *any* solution of the linearized problem

$$F'(x_k)\Delta x_k = -F(x_k) \, . \tag{2.13}$$

On the other hand, it may even occur that the iterates do not converge despite the fact that the residual becomes zero.

REMARK 3. If the nonlinear problem $F(x) = 0$ is underdetermined, i.e., $\ker F'(x) \neq \{0\}$, there is no finite ω satisfying the affine contravariant Lipschitz condition (2.10). But the convergence result is still valid, if we restrict

the Newton corrections Δx_k in (2.13) and the differences $y - x$ in (2.10) to suitable subspaces $V \subset X$ with $V \cap \ker F'(x) = \{0\}$. If X is a Hilbert space, we may e.g. choose the orthogonal complement of the kernel $\ker F'(x)$. By this means we substitute the inverse of the Jacobian by its Moore Penrose pseudo inverse (see Ben-Israel and Greville [12]) leading to the *Gauss Newton method* for underdetermined problems. In this way, theorem 2 contains the underdetermined case without any additional effort. In fact, the image oriented approach seems to be quite natural in this context, since we are interested in any solution of the problem, not a particular one.

Computational Estimates. As in the affine invariant formulation, we can easily estimate ω and the affine contravariant Kantorovitch quantity h_k. This time using the inequality (2.12), we get

$$[\omega] := 2\frac{\|F(x_{k+1})\|}{\|F(x_k)\|^2} \leq \omega \quad \text{and} \quad [h_k] := 2\frac{\|F(x_{k+1})\|}{\|F(x_k)\|} \leq h_k \;, \tag{2.14}$$

leading to the simple monotonicity test

$$\|F(x_{k+1})\| < \|F(x_k)\| \;.$$

Here, we observe one of the advantages of the affine contravariant approach. We do not need the next Newton correction (or its simplified version) to check the convergence of the iteration but only the residual at the new iterate x_{k+1}. This fact will become more important in the inexact case.

2.3 Example: Stepsize Bounds for Implicit Discretization Methods

A common device in linearly implicit discretization methods is a convergence test for the (first) Newton iteration used to solve the characteristic nonlinear equation of the implicit scheme. This test is often based on the quotient of the Newton corrections, where the next Newton correction may be substituted by so-called simplified Newton corrections, cf. Deuflhard [27]. We will demonstrate that it might be a good idea to substitute the quotient of the corrections by the quotient of the residuals, as inspired by Theorem 2. At least for the code Eulsim, based on the linearly implicit Euler discretization with extrapolation, we obtain much better results with less effort. More

importantly, the code becomes more robust for highly nonlinear problems and weak accuracy requirements, which are of most importance in technical applications, and which so far have posed a major problem for these codes.

The implicit Euler discretization for an ordinary differential equation

$$x' = f(x,t)$$

and the stepsize τ is given by the formula

$$x_{k+1} = x_k + \tau f(x_{k+1}, t_{k+1}) \approx x(t_{k+1}), \quad t_k = t_0 + k\tau \; .$$

Equivalently, we have to solve in each time step the nonlinear equation

$$F_k(x) := F_k(x,\tau) := x - x_k - \tau f(x, t_k + \tau) = 0 \, . \tag{2.15}$$

The linearly implicit Euler discretization is just the first step of an inexact Newton method for the nonlinear problem $F_k(x) = 0$ (or, equivalently, for the parameter dependent problem $F_k(x,\tau) = 0$ with τ fixed, i.e., "fixed parametrization"). The Jacobian

$$F_k'(x) = I - \tau f_x(x, t_k + \tau)$$

is substituted by

$$J := I - \tau A, \quad \text{where} \quad A := f_x(x_0, t_0) \, .$$

For autonomous systems this is in fact the exact Jacobian for the first step. As initial guess for the solution x_{k+1} of $F_k(x) = 0$, we take the last step x_k, finally leading to the linearly implicit Euler formula

$$x_{k+1} = x_k + \Delta_k, \quad \text{where} \quad \Delta_k := -J^{-1} F(x_k) = (I - \tau A)^{-1} f(x_k, t_k + \tau) \; .$$

One strategy to test the convergence of the Newton method, as proposed e.g. by Deuflhard in [27], is to compute the additional simplified Newton correction

$$\bar{\Delta}_{k+1} := -J^{-1} F(x_{k+1}) = -(I - \tau A)^{-1} \Delta_k + \Delta_{k+1}$$

and to test the quotient

$$\theta_k := \frac{\|\bar{\Delta}_{k+1}\|}{\|\Delta_k\|} = \frac{\|J^{-1} F(x_{k+1})\|}{\|J^{-1} F(x_k)\|} \; .$$

by

$$\theta_k < \theta_{\max} \tag{2.16}$$

for some $\theta_{\max} < 1$. On the other hand, Theorem 2 inspires to check the quotient of the residuals

$$\mu_k := \frac{\|F(x_{k+1})\|}{\|F(x_k)\|} = \frac{\|\Delta_k - \tau f(x_{k+1}, t_{k+1})\|}{\|\tau f(x_k, t_{k+1})\|} ,$$

(μ for "monotonicity coefficient"), which corresponds for $k = 0$ up to a factor 2 to the a posteriori estimate $[h_0]$ of the Kantorovitch quantity h_0. According to the monotonicity test (2.6), we require μ_k to satisfy

$$\mu_k < \mu_{\max} := 1 . \tag{2.17}$$

If this condition is violated, the stepsize control developed in Section 4 tells us how to reduce the time step τ. Since the linearly implicit Euler corresponds to fixed parametrization, i.e., a predictor of order 1, we have to reduce the stepsize τ in this case by

$$\tau_{\text{new}} = \rho \frac{\mu_{\max}}{\mu_k} \cdot \tau_{\text{old}} ,$$

where $\rho < 1$ is some safety factor, say, $\rho = 0.5$. Note that in contrast to the test of the Newton correction, this simple monotonicity test involves no additional solution of the linear system with matrix J, but only a vector subtraction and the two norms. For non autonomous systems we have to compute an additional right hand side $f(x_{k+1}, t_{k+1})$. Therefore, we can afford this monotonicity test in each step, at least for autonomous equations, leading to a surprisingly robust behavior of the resulting code.

Example. As a small but sufficiently complicated example we take the (one cell) Brusselator (cf. for example [54]), a system of two autononous differential equations

$$x_1' = a - (b+1)x_1 + x_1^2 x_2 \tag{2.18}$$

$$x_2' = b x_1 - x_1^2 x_2 \tag{2.19}$$

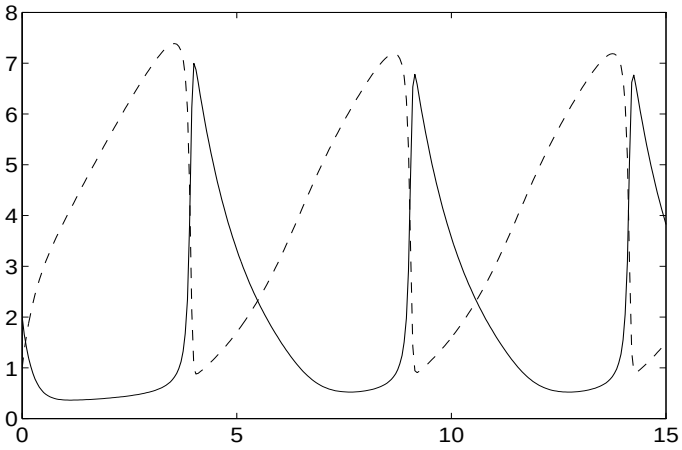


FIG. 1. Solution of the Brusselator (2.18), x_1 and x_2 (dashed) versus t

Figure 1 shows the solution. We compare the old code EULSIM with the error oriented monotonicity test to the new one with the monotonicity device (2.17). To get comparable result, we use a relatively scaled norm in both cases. Table 1 shows the results for different demands tol on the (relatively scaled) absolute error. We chose the parameters $a = 2$ and $b = 10$ and

	error test (2.16)			residual test (2.17)		
tol	steps	#f	error	steps	#f	error
10^{-5}	50	1061		48	974	
0.0025	41	283	0.0057	45	348	0.0027
0.005	41	268	0.012	50	272	0.0056
0.01	37	189	0.041	47	206	0.012
0.02	36	149	0.036	40	163	0.034
0.04	33	116	0.11	34	124	0.059
0.08	31	99	1.6	44	118	0.059
0.16	34	85	8.5	39	88	0.10

TABLE 1. Performance of EULSIM for the Brusselator (2.18)

the initial value is $x = (2, 1)$. Integrating from $t = 0$ to $t = 15$, we used

the solution with tol $= 10^{-5}$ as the reference solution and then doubled the required accuacy in each run beginning with tol $= 0.0025$. #f denotes the numbers of function evaluations of the ODE's right hand side.

The results show that even for an allowed error of more than 10 % the solution obtained with the residual oriented monotonicity test is qualitively correct. The code recognizes the nonlinearity independently from the accuracy demand. In fact, the nonlinearity rules the behaviour of the integrator for accuracies tol > 0.16, so that the costs do not decrease any more. We would also like to point out the (almost) monotonous behaviour of the error over the whole span of $0.0025 <$ tol < 0.16, demonstrating the robustness gained by the monotonicity test.

Example. As a second example consider the following system of five autonomous differential equations modelling a chemical oscillator (cf. Seelig [58] and [59]).

$$\begin{aligned} x_1' &= j - k_1 x_1 - k_4 x_1 x_4 + k_{-4}(E_{\text{tot}} - x_4 - x_5) \\ x_2' &= k_1 x_1 - k_2 x_2 \\ x_3' &= k_2 x_2 - k_3 x_3 - k_5 x_3 (E_{\text{tot}} - x_4 - x_5) + (k_{-5} + k_6) x_5 \\ x_4' &= -k_4 x_1 x_4 + k_{-4}(E_{\text{tot}} - x_4 - x_5) + k_6 x_5 \\ x_5' &= k_5 x_3 (E_{\text{tot}} - x_4 - x_5) - (k_{-5} + k_6) x_5 \end{aligned} \tag{2.20}$$

We compute the solution for the parameters $E_{\text{tot}} = 1$, $k_1 = k_2 = k_3 = 1$, $k_4 = k_{-5} = 2000$, $j = 100$, $k_{-4} = k_5 = 100$, $k_6 = 600$, and the initial guess $x_{\text{start}} = (9, 7, 5, 0.01, 0.16)$ from $t = 0$ to $t = 3$. Table 2 presents the results. Here the two versions of the extrapolation code show quite a different behaviour. The residual oriented monotonicity test leads to much more integration steps for the same required accuracy leading to better results (the true error meets almost the required accuracy) but also needing more effort. On the other hand the gained accuracy per number of function evaluations is still comparable to the error oriented monotonicity test. Moreover, the latter could not be called robust for tolerances tol > 0.1 and the code even fails to produce a solution for tol $= 0.15$, since the number of steps exceeded 1000. Using the original scaling, the code failes for tol $= 0.08$ and tol $= 0.04$ but gives a result for tol $= 0.16$, so that its behaviour seems to be more or less random for weak accuracy requirements.

	error test (2.16)			residual test (2.17)		
tol	steps	#f	error	steps	#f	error
10^{-5}	17	328		59	478	
0.0025	12	106	0.0051	63	176	0.0027
0.005	11	78	0.013	50	131	0.0066
0.01	10	48	0.016	37	96	0.014
0.02	9	32	0.2	34	88	0.051
0.04	6	20	0.26	38	93	0.046
0.08	9	27	0.38	37	93	0.10
0.16	failed			34	84	0.10

TABLE 2. Performance of EULSIM for the chemical oscillator (2.20)

3 INEXACT NEWTON METHODS

In an *inexact Newton method* we do not solve the Newton equation exactly, but compute an approximate solution s_k and obtain the *inexact Newton iterates*

$$x_{k+1} = x_k + s_k \quad \text{for } k = 0, 1, \ldots.$$

We define the *inner residuals* r_k by

$$F'(x_k)s_k = -F(x_k) + r_k \quad \text{for } k = 0, 1, \ldots \tag{3.1}$$

not to be confused with the *outer residuals* $F(x_k)$. In many applications the inexact Newton corrections s_k result from an iterative solver for the Newton equation, i.e., an *inner iteration* in contrast to Newton's method itself as the *outer iteration*. But we may also think of any other approximate solution of the linear equation. Of particular interest are the quasilinearization or multilevel Newton methods, where Newton's method is defined in an infinite dimensional setting. Here, we have to solve the arising infinite dimensional linear subproblems by some discretization.

Obviously, the crucial point is an appropriate matching between Newton's method (as the outer iteration) and the linear solver (the inner iteration). We want to spend as little effort as possible for the linear problems but without loosing the quadratic convergence property of the Newton iteration.

The easiest way to control the inexact Newton corrections s_k is to give upper bounds η_k for the relative residuals

$$\frac{\|r_k\|}{\|F(x_k)\|} \leq \eta_k \ . \tag{3.2}$$

This requirement was used by Dembo, Eisenstat and Steihaug [22]. They employ a Hölder condition on F' as well as bounds on $F'(x)$ and $F'(x)^{-1}$ to achieve a convergence order $1 < q \leq 2$ for the inexact Newton method, if the inner residuals r_k tend to zero with the same order. They end up with the accuracy matching strategy

$$\eta_k \leq c \, \|F(x_k)\|^{q-1}$$

for some constant $c > 0$. The choice of the constant c remains unclear due to the fact that it is rather difficult if not impossible to apply their theorems quantitatively in an actual code since there are in general no means to cheaply estimate the required constants. Following Dembo et al., Bank and Rose [10] [11] use the strategy

$$\eta_k = \eta_0 \left(\frac{\|F(x_k)\|}{\|F(x_0)\|} \right)^{q-1} = \eta_{k-1} \left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|} \right)^{q-1}$$

Here, the more or less heuristic choice of η_0 has a major influence on the whole iteration process.

We now turn to affine (contra-) invariant convergence theorems which are more suitable for applications as discussed above. In order to obtain a smooth transition from exact to inexact Newton methods, we introduce a parameter $0 \leq \beta \leq 1$ describing the "exactness" of the method. Setting $\beta = 0$ corresponds to the exact Newton iteration (cf. Theorems 1 and 2), while $\beta = 1$ distributes the error in equal parts on the Newton iteration and the inexact solution of the Newton equation. Thus, β may be viewed as an embedding parameter from exact to inexact Newton methods.

3.1 Affine Invariant Approach

We start with a rather concise form of an affine invariant convergence theorem for the inexact Newton method which extends previous results by Deuflhard (restricted to a class of linear perturbations of the Jacobian).

THEOREM 3. *Let $F : D \subset X \to X$ be a continously Fréchet-differentiable mapping, such that $F'(x)$ is continously invertible for all $x \in D$. Moreover, we require F' to satisfy the affine invariant Lipschitz condition*

$$\|F'(x)^{-1}(F'(y) - F'(z))\| \leq \omega \|y - z\| \quad \text{for all } x, y, z \in D, \tag{3.3}$$

for some $\omega > 0$. Introducing the Kantorovitch quantities $h_k := \omega \|s_k\|$, we assume that the error of the inexact Newton corrections s_k is bounded by

$$\|s_k - \Delta x_k\| \leq \varepsilon_k \|s_k\|, \quad \text{where } \varepsilon_k := \frac{\beta}{2} \frac{h_k}{1 + h_k} \tag{3.4}$$

for some $0 \leq \beta \leq 1$. If the initial guess x_0 satisfies

$$h_0 < h_{\max} := \frac{2 - \beta}{1 + \beta} \tag{3.5}$$

and the inexact Newton iteration x_k stays in the domain D, then it converges quadratically to a solution x_*, $F(x_*) = 0$.

Proof. To estimate $F(x_{k+1})$ in the inexact case, we have to extend the standard integral expression by the inner residual r_k:

$$\begin{aligned} F(x_{k+1}) &= F(x_k + s_k) - F(x_k) - F'(x_k)s_k + r_k \\ &= \int_0^1 (F'(x_k + ts_k) - F'(x_k))s_k \, \mathrm{dt} \; + r_k \; . \end{aligned} \tag{3.6}$$

The Lipschitz condition (3.3) gives us the bound

$$\begin{aligned} \|F'(x_{k+1})^{-1}F'(x_k)\| &\leq \|F'(x_{k+1})^{-1}(F'(x_k) - F'(x_{k+1}))\| + \|I\| \\ &\leq 1 + \omega \|s_k\|. \end{aligned}$$

Applied to the inner residual r_k, we get

$$\begin{aligned} \|F'(x_{k+1})^{-1}r_k\| &= \|F'(x_{k+1})^{-1}F'(x_k)(s_k - \Delta x_k)\| \\ &\leq (1 + \omega \|s_k\|) \|s_k - \Delta x_k\| \; . \end{aligned}$$

Thus we obtain for the Newton correction

$$\|\Delta x_{k+1}\| = \|F'(x_{k+1})^{-1}F(x_{k+1})\|$$

$$
\begin{aligned}
&= \|\int_0^1 F'(x_{k+1})^{-1}(F'(x_k+ts_k)-F'(x_k))s_k \ \mathrm{dt} \ + F'(x_{k+1})^{-1}r_k\| \\
&\leq \int_0^1 \|F'(x_{k+1})^{-1}(F'(x_k+ts_k)-F'(x_k))s_k\| \ \mathrm{dt} \ + \|F'(x_{k+1})^{-1}r_k\| \\
&\leq \frac{\omega}{2}\|s_k\|^2 + (1+\omega\,\|s_k\|)\,\|s_k-\Delta x_k\| \\
&\leq h_k\frac{\|s_k\|}{2} + \varepsilon_k(1+h_k)\,\|s_k\| \ = \ (1+\beta)h_k\frac{\|s_k\|}{2}\,.
\end{aligned}
$$

Since $\varepsilon_k \leq \beta/2$, we have

$$
\|s_k\| \ \leq \ \frac{1}{1-\varepsilon_k}\|\Delta x_k\| \ \leq \ \frac{2}{2-\beta}\|\Delta x_k\|\,.
$$

Thus, we are able to estimate the inexact Newton corrections by

$$
\|s_{k+1}\| \ \leq \ \frac{2}{2-\beta}\|\Delta x_{k+1}\| \ \leq \ \frac{1+\beta}{2-\beta}h_k\,\|s_k\|\,.
$$

Equivalently, the quatities h_k satisfy

$$
h_{k+1} \ \leq \ \frac{1+\beta}{2-\beta}\,h_k^2\,.
$$

Requiring $h_1 < h_0$ leads to the initial condition (3.5). If this condition is fulfilled, then the h_k obviously converge quadratically to zero. Hence, the inexact Newton iterates form a Cauchy sequence converging to some point $x_* = \lim_{k\to\infty} x_k \in X$ which immediately turns out to be a solution of F, i.e., $F(x_*) = 0$. □

REMARK 4. For simplicity we only considered the quadratic convergence of the inexact Newton process as the best possible order of convergence. Thus we had to impose the Lipschitz condition (3.3) on the Jacobian F'. A corresponding Hölder condition

$$
\|F'(x)^{-1}(F'(y)-F'(z))\| \leq \omega\,\|y-z\|^p \ \text{ for all } x,y,z \in D,
$$

for some $0 < p \leq 1$ would lead to a convergence order of $1+p \in (1,p]$.

Computational Estimates. In the context of an inexact method, the Kantorovitch quantities are to be used threefold:

- to check the convergence by the monotonicity test

$$h_k < h_{\max}\,, \tag{3.7}$$

- to control the linear solver by the accuracy requirement

$$\|s_k - \Delta x_k\| \le \varepsilon_k \|s_k\|, \quad \varepsilon_k = \frac{\beta}{2}\frac{h_k}{1+h_k}\,, \tag{3.8}$$

- to control a surrounding method, e.g., the steplength of a continuation process.

The proof of Theorem 3 offers us a basic inequality involving the inexact Newton corrections s_k and the Kantorovitch quantities h_k, namely

$$\|s_{k+1}\| \le \frac{1+\beta}{2(1-\varepsilon_{k+1})} h_k \|s_k\| \tag{3.9}$$

or equivalently

$$h_{k+1} \le \frac{1+\beta}{2(1-\varepsilon_{k+1})} h_k^2\,. \tag{3.10}$$

Analogous to the estimates in Section 2, this inequality leads to the *a posteriori Kantorovitch estimates*

$$[h_k] := \frac{2(1-\varepsilon_{k+1})}{1+\beta}\frac{\|s_{k+1}\|}{\|s_k\|} \le h_k\,. \tag{3.11}$$

REMARK 5. Note however that this estimate may be quite rough, since it only contains information about local directional derivatives, whereas the convergence Theorem 3 uses global inequalities to describe the nonlinearity. We shall see in Section 3.2 that this problem is resolved in the affine contravariant formulation.

Regarding the monotonicity test (3.7) we encounter the same problem as in the exact case. The inexact Newton corrections s_{k+1} involved in the estimate (3.11) is obsolete if the convergence test fails. As proposed by Deuflhard

[28], this problem can be circumvented by introducing the *simplified inexact Newton corrections* $\bar{s}_{k+1}$ characterized by

$$\|\bar{s}_{k+1} - \bar{\Delta}x_{k+1}\| \le \bar{\varepsilon}_{k+1}\|\bar{\Delta}x_{k+1}\|$$

for some $\bar{\varepsilon}_{k+1} < \beta/2$, say $\bar{\varepsilon}_{k+1} := \beta/4$. As in the proof of Theorem 3 we get the inequality

$$\|\bar{s}_{k+1}\| \le \frac{1+\beta}{2(1-\bar{\varepsilon}_{k+1})}\|h_k\bar{s}_k\|$$

leading to the a posteriori estimate

$$[h_k] := \frac{2(1-\bar{\varepsilon}_{k+1})}{1+\beta}\frac{\|\bar{s}_{k+1}\|}{\|s_k\|} \le h_k\,. \tag{3.12}$$

Note that the effort to compute $\bar{s}_{k+1}$ is again almost negligible compared to the effort expended on the next inexact correction s_{k+1}, because we only require a very moderate relative accuracy of $\bar{\varepsilon}_{k+1} = \beta/4$. Moreover, in the context of an iterative linear solver, the simplified correction $\bar{s}_{k+1}$ may be exploited as initial guess for the next correction s_{k+1} (as realized in the code Giant [28]) assuming that the iterative solver rewards good initial guesses.

The simplified inexact Newton correction helps us with the (a posteriori) monotonicity test (3.7) but not with the accuracy matching (3.8), since we have to know some estimate of h_k *before* s_k is computed. Here, we follow again Deuflhard and employ the quadratic convergence property (3.10) to define the *a priori Kantorovitch estimate*

$$[h_k] := \frac{1+\beta}{2(1-\bar{\varepsilon}_{k-1})}[h_{k-1}]^2\,. \tag{3.13}$$

Note that we cannot prove an inequality like $[h_k] \le h_k$ as for the a posteriori estimate. Substituting the analytic quantity h_k in (3.8) by its computationally available a priori estimate $[h_k]$, we arrive at the matching strategy

$$[\varepsilon_k] := \frac{\beta}{2}\frac{[h_k]}{1+[h_k]}\,, \tag{3.14}$$

3.2 Affine Contravariant Approach

Theorem 3 leaves us with some problems. First of all, we do not know how to control the relative error of the inexact Newton correction s_k. Considering

a (discretized) nonlinear PDE, most iterative solvers give us in fact only the inner residual r_k, so that the only chance to obtain reliable information about the error $F'(x_k)^{-1}r_k$ is to use a good (theoretically well understood) preconditioner. Unfortunately, preconditioners of this kind so far exist only for (almost) elliptic problems.

Furthermore, we would prefer a directional derivative in the Lipschitz condition (3.3) as in Theorem 1, since we algorithmically always estimate directional derivatives. But the operator norm in (3.3) seems to be unavoidable, since we have to estimate $\|F'(x_{k+1})^{-1}F'(x_k)\|$. Fortunately, the affine contravariant approach overcomes these difficulties, as we will see now.

THEOREM 4. *Let $F : D \subset X \to Y$ be Gâteaux-differentiable, satisfying the Lipschitz condition*

$$\|(F'(y) - F'(x))(y - x)\| \le \omega \|F'(x)(y - x)\|^2 \quad \text{for all } x, y \in D, \tag{3.15}$$

for some $\omega > 0$. Defining $h_k := \omega \|F'(x_k)s_k\|$, let the inner residuals r_k be bounded by

$$\|r_k\| \le \varepsilon_k \|F'(x_k)s_k\|, \quad \text{where } \varepsilon_k := \frac{\beta}{2} \min(1, h_k) \tag{3.16}$$

for some $0 \le \beta \le 1$. If the initial guess x_0 satisfies

$$h_0 < h_{\max} := \frac{2 - \beta}{1 + \beta}. \tag{3.17}$$

and the inexact Newton iteration x_k stays in D, then the residuals $F(x_k)$ converge quadratically to zero.

Proof. We first note that (3.16) implies for $\varepsilon_k < 1$

$$\|F(x_k)\| \le (1 + \varepsilon_k) \|F'(x_k)s_k\| \quad \text{and} \quad \|F'(x_k)s_k\| \le \frac{1}{1 - \varepsilon_k} \|F(x_k)\|. \tag{3.18}$$

Using again the integral representation (3.6) of $F(x_{k+1})$ and the Lipschitz condition (3.15), we obtain

$$\|F(x_{k+1})\| \quad \le \quad \int_0^1 \|(F'(x_k + ts_k) - F'(x_k))s_k\| \, \mathrm{d}t + \|r_k\|$$

$$\begin{aligned} &\leq \frac{\omega}{2}\|F'(x_k)s_k\|^2 + \varepsilon_k\|F'(x_k)s_k\| \\ &\leq \omega\frac{1+\beta}{2}\|F'(x_k)s_k\|^2 \\ &\leq \frac{1+\beta}{2}h_k\,\|F'(x_k)s_k\| \,. \end{aligned}$$

Since $\varepsilon_k \leq \beta/2$, we have

$$\|F'(x_k)s_k\| \leq \frac{2}{2-\beta}\,\|F(x_k)\| \,.$$

Therefore, the Kantorovitch quantities h_k meet the same inequality as in Theorem 3, since

$$h_{k+1} = \omega\,\|F'(x_{k+1})s_{k+1}\| \leq \frac{2}{2-\beta}\omega\,\|F(x_{k+1})\| \leq \frac{1+\beta}{2-\beta}h_k^2 \,. \tag{3.19}$$

Again, the monotonicity requirement $h_1 < h_0$ leads to the initial condition (3.17). Thereby, we force the sequence h_k to converge geometrically, and, by (3.19), even quadratically to zero. Because of (3.18) the same holds then for the (outer) residuals. □

REMARK 6. Note that the Kantorovitch quantity $h_k = \omega\|F'(x_k)s_k\|$ coincides in the exact case $\beta = 0$ with the quantity defined in Theorem 2.

Of course, we can use (3.18) to derive a matching stategy for the relative residuals

$$\|r_k\| \leq \eta_k\,\|F(x_k)\|, \quad \text{where} \quad \eta_k := \frac{\varepsilon_k}{1+\varepsilon_k} \,,$$

which obviously implies (3.16). This argument also holds in what follows: Up to $O(\varepsilon_k)$ the residual norm $\|F(x_k)\|$ and $\|F'(x_k)s_k\|$ are interchangeable.

Computational Estimates. The estimates for the affine contravariant Kantorovitch quantities h_k are derived in complete analogy to the affine invariant approach. The only difference is that we do not need any simplified corrections, because the (outer) residuals are always one step ahead. From Theorem 4 we learn the inequalities

$$\|F(x_{k+1})\| \leq \frac{1+\beta}{2(1-\varepsilon_k)}h_k\,\|F(x_k)\| \tag{3.20}$$

and

$$h_{k+1} \leq \frac{1+\beta}{2(1-\varepsilon_{k+1})} h_k^2 \,. \tag{3.21}$$

The first inequality (3.20) leads to the affine contravariant *a posteriori Kantorovitch estimate*

$$[h_k] := \frac{2(1-\varepsilon_k)}{1+\beta} \frac{\|F(x_{k+1})\|}{\|F(x_k)\|} \leq h_k \,. \tag{3.22}$$

In contrast to the affine invariant approach, we only need the outer residual of the current and the last step. Moreover, recalling the proof of Theorem 4, we observe that only local derivatives in direction of the inexact Newton correction s_k were involved. Hence, the a posteriori estimate (3.22) is despite its simplicity a sharp estimate for the local Kantorovich quantity. For the accuracy matching (3.16) we need as in Section 3.1 an estimate for h_k before the correction s_k is computed. Employing the quadratic convergence property (3.21), we derive the a priori estimate

$$[h_k] := \frac{1+\beta}{2(1-\varepsilon_{k-1})} [h_{k-1}]^2 \,. \tag{3.23}$$

Thus, we arrive at the affine contravariant matching strategy

$$[\varepsilon_k] := \frac{\beta}{2} \min(1, [h_k]) \,. \tag{3.24}$$

REMARK 7. In the algorithmic realization, the accuracy requirements should be coupled with a lower bound (depending on the prescribed accuracy for the solution of the nonlinear problem) to avoid too strict conditions for the last Newton iteration.

Discussion

We have derived two convergence theorems for the inexact Newton process that reflect its affine invariance properties. While both points of view are almost interchangeable for the exact Newton method, the affine contravariant approach has some structural advantages in the inexact case that allow an easy and precise accuracy control for the linear subproblems. The weak

differentiability assumptions not only make it applicable to a wide range of nonlinear problems but also lead to better estimates for the analytic constants that rule the convergence behaviour.

Note however that our investigations aimed at well- or underdetermined nonlinear problems, where the solution is characterized by a *vanishing residual*. In the overdetermined case this is no longer true: The residual at a solution is in general different from zero, and, more importantly, not known in advance. Here, a *vanishing Newton correction* indicates a solution. Not surprisingly, this line of thought led to the affine invariant convergence theorems by Deuflhard and Heindl and the affine invariant monotonicity test.

3.3 Example: Accuracy Matching for Multiple Shooting

In the multiple shooting context, both the function F and the Jacobian $F'(x)$ are approximated by some numerical integrator. Hence, we have to translate the matching strategy of the inexact Newton method to appropriate accuracy demands for the integrators used to compute the flow and the propagation matrices. To our knowledge, the only adaptive accuracy control for shooting methods has been developed by Bock [14] [13] for his parameter identification code Parfit solving overdetermined nonlinear problems. It is based on the (affine invariant) *heuristic* that the error $\|s_k - \Delta x_k\|$ should be below the norm $\|\Delta x_{k+1}\|$ of the next correction. Using an a priori estimate for $\|\Delta x_{k+1}\|$ similar to (3.13) this heuristic leads to an implementable accuracy control mechanism. This affine invariant approach necessitates estimates for the norm of the (pseudo) inverse of the Jacobian. Based on the affine invariant inexact Newton Theorem 3, we developed an accuracy control for multiple shooting applied to underdetermined problems, particularly to parameter dependent problems and periodic solutions of autonomous ODEs. The resulting strategy and numerical examples are documented in the diploma thesis by C. Wulff [66].

Proceeding, we present the affine contravariant analogon and derive an accuracy matching based on the control of the inner residuals. We shall see that this strategy is much simpler and needs no information about the norm of the Jacobian or its inverse.

We consider the two point boundary value problems given by the first order

ODE

$$x' = f(x,t) \quad \text{on} \quad [a,b] \tag{3.25}$$

and the nonlinear two point boundary conditions

$$r(x(a), x(b)) = 0\,, \tag{3.26}$$

where f and r are continuously differentiable mapping

$$f : X \times \mathbb{R} \longrightarrow X, \quad r : X \times X \longrightarrow Z\,,$$

with $X = \mathbb{R}^n$, $Z = \mathbb{R}^m$ (typically $n = m$). The extension to multi-point boundary conditions is straight forward. According to the multiple shooting approach, we choose some partition

$$a = t_1 < t_2 < \cdots < t_k < t_{k+1} = b$$

of the basic interval $[a,b]$, where t_i are the *shooting nodes*. By $\Phi^{t,s} : X \to X$ we denote the flow of the ODE (3.25) from s to t, i.e.,

$$D_t \Phi^{t,s}(x) = f(\Phi^{t,s}(x), t) \quad \text{and} \quad \Phi^{t,t}(x) = x$$

for all $x \in X$. Thus, we can reformulate the BVP given by (3.25) and (3.26) as the finite dimensional nonlinear equation $F(x) = 0$, where the nonlinear function

$$F : X^k \longrightarrow X^{k-1} \times Z$$

is defined as

$$F(x) = F(x_1, \ldots, x_k) := \begin{pmatrix} \Phi^{t_2,t_1}(x_1) - x_2 \\ \vdots \\ \Phi^{t_k,t_{k-1}}(x_{k-1}) - x_k \\ r(x_1, x_{k+1}) \end{pmatrix},$$

where $x_{k+1} := \Phi^{t_{k+1},t_k}(x_k)$. Its Jacobian is the cyclic matrix

$$F'(x) = \begin{bmatrix} W_1 & -I & & \\ & \ddots & \ddots & \\ & & W_{k-1} & -I \\ B_1 & & & B_2 W_k \end{bmatrix}$$

where we used the common notation

$$W_i := D_x \Phi^{t_{i+1},t_i}(x_i) \quad \text{and} \quad B_i := D_i r(x_1, x_{k+1})\,.$$

Appropriate Norms. A central point for the accuracy matching is the choice of the correct norms on X^k and $X^{k-1} \times Z$. Standard modern ODE solvers, such as adaptive extrapolation or Runge-Kutta methods, allow us to control the accuracy of the approximate flow $\tilde{\Phi}^{t,s}(x)$ in a relatively scaled norm, i.e.

$$\|\tilde{\Phi}^{t,s}(x) - \Phi^{t,s}(x)\|_{\text{scal}} \leq \text{tol}, \quad \text{where} \quad \|\tilde{\Phi}^{t,s}(x)\|_{\text{scal}} = 1 \,,$$

for some prescribed $tol > 0$. Thus, aiming at an implementable accuracy control, we provide the domain X^k with the norm

$$\|(x_1, \ldots, x_k)\|^2 := \frac{1}{k} \sum_{i=1}^{k} \|x_i\|_{\text{scal}}^2 \tag{3.27}$$

and the image space $X^{k-1} \times Z$ with the norm

$$\|(x_1, \ldots, x_{k-1}, z)\|^2 := \frac{1}{k} \left(\sum_{i=1}^{k-1} \|x_i\|_{\text{scal}}^2 + \|z\|^2 \right) \,. \tag{3.28}$$

Here, $\|z\|$ is the user defined norm on Z for the boundary values $r(u, v)$.

REMARK 8. Of course, the scaled norm $\|\cdot\|_{\text{scal}}$ varies from step to step (of the Newton iteration), since it is computed during the integration process. In praxis, however, this variation is quite small due to the local character of the Newton method.

Accuracy Control of the Flow. As starting point we take the absolute accuracy demand for the inner residuals

$$\|r\| \leq \delta, \quad \text{where} \quad r = F'(x)s + F(x) \,.$$

For ease of notation, we drop the index k of the Newton iteration. We first distribute δ in equal parts on the function F and the derivative $F'(x)s$ and therefore require

$$\|\tilde{F}(x) - F(x)\| \leq \delta_F \tag{3.29}$$

for the approximation $\tilde{F}$ of the function F and

$$\|F'(x)s - \tilde{F}(x)\| \leq \delta_A \tag{3.30}$$

for the inexact correction s, where $\delta_F = \delta_A = \delta/2$. To translate the demand (3.29) to the flow, we observe that

$$\|\tilde{F}_i(x) - F_i(x)\| \leq \begin{cases} \|\tilde{\Phi}_i(x_i) - \Phi_i(x_i)\| & \text{for } i = 1, \dots, k-1 \\ \|B_2\| \, \|\tilde{\Phi}_k(x_k) - \Phi_k(x_k)\| & \text{for } i = k \end{cases}$$

where we used the abbreviation $\Phi_i := \Phi^{t_{i+1},t_i}$. Let δ_{Φ_i} denote the accuracy for the i-th flow Φ_i, i.e.,

$$\|\tilde{\Phi}_i(x_i) - \Phi_i(x_i)\| \leq \delta_{\Phi_i} \, .$$

Due to the definition (3.28) of the norm on the image space, we meet the accuracy requirement (3.29) by setting

$$\delta_{\Phi_i} := \begin{cases} \delta_F & \text{for } i = 1, \dots, k-1 \\ \delta_F / \|B_2\| & \text{for } i = k \end{cases}$$

Accuracy Control of the Linearized Flow. To derive a matching strategy for the linearized flow, we assume that the propagation matrices W_i are approximated using a numerical integrator that guarantees

$$\|\tilde{W}_i - W_i\| \leq \delta_{W_i}$$

for a prescribed absolute accuracy $\delta_{W_i} > 0$. Here, the norm is the operator norm corresponding to the scaled norms at x_i and x_{i+1}, respectively. The arising cyclic linear system

$$\tilde{F}'(x)s = -\tilde{F}(x), \quad \text{where} \quad \tilde{F}'(x) = \begin{bmatrix} \tilde{W}_1 & -I & & \\ & \ddots & \ddots & \\ & & \tilde{W}_{k-1} & -I \\ B_1 & & & B_2\tilde{W}_k \end{bmatrix} \tag{3.31}$$

is then solved directly either via block Gaussian elimination or using a direct sparse solver. Compared to the discretization error of the numerical integration, the round off error of the linear system solver is negligible so that we may suppose that the solution of (3.31) is exact. As a consequence, the residual

$$\tilde{r} := F'(x)s + \tilde{F}(x)$$

is given by

$$\tilde{r} = F'(x)s - \tilde{F}'(x)s = \begin{pmatrix} (W_1 - \tilde{W}_1)s_1 \\ \vdots \\ (W_{k-1} - \tilde{W}_{k-1})s_{k-1} \\ B_2(W_k - \tilde{W}_k)s_k \end{pmatrix} .$$

Its norm can be estimated by

$$\|\tilde{r}\|^2 \leq \frac{1}{k} \left(\sum_{i=1} k - 1 \|W_i - \tilde{W}_i\|^2 \, \|s_i\|^2 + (\|B_2\| \, \|W_k - \tilde{W}_k\| \, \|s_k\|)^2 \right) .$$

Therefore, we meet the accuracy demand (3.30), which now reads

$$\|\tilde{r}\| \leq \delta_A \, ,$$

if we set

$$\delta_{W_i} := \begin{cases} \delta_A / \|s_i\| & \text{for } i = 1, \ldots, k-1 \\ \delta_A / (\|B_2\| \, \|s_k\|) & \text{for } i = k \, . \end{cases} \tag{3.32}$$

We observe however that (3.32) involves the yet unknown corrections s_i. But remembering the a posteriori estimate $[h_k]$ in the affine invariant setting (see Section 3.1), we can use the same trick and substitute the correction s_i in (3.32) by its simplified counterpart $\bar{s}_i$, thereby arriving at an easily implementable accuracy control for multiple shooting methods. This strategy was realized as part of the Cocon package for boundary value problems of ODEs. For periodic boundary conditions, it turns out to be almost twice as efficient (with respect to function evaluations) as the old strategy.

4 Stepsize Control for Continuation Methods

This section extends the ideas developed in the preceding section to continuation methods for parameter dependent problems

$$F(z,\lambda) = 0, \quad \text{where} \quad F : D \subset Z \times \Lambda \longrightarrow Y\,, \tag{4.1}$$

where Z is a Banach space isomorphic to Y and Λ is some finite dimensional parameter space, $\dim \Lambda = p$. We follow the implicit parametrization idea of Deuflhard, Fiedler and Kunkel [30] and consider instead of (4.1) the underdetermined nonlinear problem in $X = Z \times \Lambda$

$$F(x) = 0, \quad \text{where} \quad F : D \subset X \longrightarrow Y\,. \tag{4.2}$$

In order to obtain a well defined p-dimensional solution manifold $\mathcal{M} \subset \mathcal{D}$, we have to require (see e.g. Fink and Rheinboldt [38]) that F is a Fredholm mapping of index p with $\dim \ker F'(x) = p$ for all solutions $x \in D$, $F(x) = 0$. Thus, $0 \in Y$ is a regular value of F and (4.2) defines a p-dimensional manifold $\mathcal{M}$ in $D \subset X$. If $\bar{x} \in \mathcal{M}$ is a solution, $F(\bar{x}) = 0$, we can find a local parametrization

$$x : \Sigma \longrightarrow D \subset X, \quad \sigma \longmapsto x(\sigma)$$

of $\mathcal{M}$ in an open neighbourhood $U \subset X$ of $\bar{x}$ (where the notation $\sigma \in \Sigma$ was chosen for 'stepsize'). More precisely, Σ is an open neighbourhood of $0 \in \mathbb{R}^p$, and x a differentiable mapping, such that

$$\mathcal{M} \cap U = x(\Sigma) \quad \text{and} \quad x(0) = \bar{x}\,.$$

Observe however that this approach does not include problems that are underdetermined by their very nature, e.g., due to some symmetry property. As a prototype, we think of the periodic solutions of a parameter dependent autonomous ODE. Here, the time invariance of the ODE leads to the S^1-symmetry of the periodic solutions and thus to an underdetermined problem with the period T as an additional implicit parameter. Analytically, it is fairly easy to reduce this symmetry, e.g., by intoducing an appropriate Poincaré section. Numerically, there is in general no canonical a priori choice for such a Poincaré section so that it is more feasible to tackle the underdetermined problem directly as proposed by Deuflhard [26]. Fortunately, the

affine contravariant approach needs no uniqueness of the solution and thus directly incorporates the underdetermined case.

To derive a steplength selection strategy for predictor-corrector methods in the affine contravariant setting, we define a general predictor for p-dimensional solution manifolds as follows.

DEFINITION 1. Let $\bar{x} \in D$ be a solution of F, $F(\bar{x}) = 0$. A *predictor* for the nonlinear problem $F(x) = 0$ at $\bar{x}$ is a mapping

$$\hat{x} : \Sigma \longrightarrow D \subset X \ ,$$

defined in an open neighbourhood Σ of $0 \in \mathbb{R}^p$ such that $\hat{x}(0) = \bar{x}$ and

$$\|F(\hat{x}(\sigma))\| \leq C\phi(\sigma) \ \text{ for all } \sigma \in \Sigma \tag{4.3}$$

for some constant $C > 0$ and a continuous function $\phi : \Sigma \to \mathbb{R}$ with $\phi(0) = 0$. The predictor $\hat{x}$ has *order* $q > 0$, if $\hat{x}$ fulfills condition (4.3) for $\phi(\sigma) = \|\sigma\|^q$, i.e.,

$$\|F(\hat{x}(\sigma))\| \leq C\|\sigma\|^q \, .$$

As an example, we consider the standard tangential predictor

$$\hat{x}(\sigma) = \bar{x} + \sigma t$$

for $p = 1$, where t is a normalized tangent at $\bar{x}$, i.e., $F'(\bar{x})t = 0$ and $\|t\| = 1$. For twice differentiable mappings F, it has obviously order $q = 2$, since

$$\|F(\hat{x}(\sigma))\| = \underbrace{\|F(\bar{x}) + \sigma F'(\bar{x})t\|}_{=0} + \sigma^2 O(\|F''(\bar{x})t^2\|) \, .$$

Substituting the predictor characterization (4.3) for the initial guess in the inexact Newton Theorem 4, we can derive a bound for the *maximal feasible stepsize* for which we may expect convergence of the inexact Newton iteration.

THEOREM 5. *Let $F : D \subset X \to Y$ be a Gâteaux differentiable mapping such that the Lipschitz condition (3.15) holds for some $\omega > 0$ as in Theorem 4. In addition, let $\hat{x}$ be a predictor for $F(x) = 0$ at a solution $\bar{x} \in D$. Then the inexact Newton method converges for the initial guess $\hat{x}(\sigma)$ for all stepsizes $\sigma \in \Sigma$ satisfying*

$$\phi(\sigma) < \frac{2-\beta}{2\omega C}\, h_{\max}\,, \quad \textit{where} \quad h_{\max} = \frac{2-\beta}{1+\beta} \tag{4.4}$$

as in Theorem 4.

Proof. Substituting $\hat{x}(\sigma)$ for the initial guess x_0 in Theorem 4 yields

$$h_0 = \omega \|F'(x_0)s_0\| \le \frac{2}{2-\beta}\omega\|F(x_0)\| \le \frac{2\omega C}{2-\beta}\,\phi(\sigma)\,, \tag{4.5}$$

leading to the stepsize bound (4.4). □

For a predictor of order q, this leads in particular to the stepsize bound

$$\|\sigma\| \le \sqrt[q]{\frac{(2-\beta)h_{\max}}{2\omega C}}\,.$$

As for the Kantorovitch quantities h_k, we have to look for computationally available estimates for ωC to get an implementable stepsize control. Using the a posteriori estimate $[h_0]$ and inequality (4.5), we derive the *a posteriori estimate*

$$[\omega C] := \frac{2-\beta}{2}\frac{[h_0]}{\phi(\sigma)} \le \frac{2-\beta}{2}\frac{h_0}{\phi(\sigma)} \le \omega C\,.$$

For order q predictors this expression simplifies to

$$[\omega C] = \frac{2-\beta}{2}\frac{[h_0]}{\|\sigma\|^q}\,.$$

Instead of constructing an additional *a priori estimate* for ωC (probably using second order information like the curvature), we simply use the a posteriori estimate of the last continuation step. This strategy has proven to be sufficiently efficient in praxis.

Thus, we arrive at a stepsize control mechanism for inexact continuation methods: Given a solution $\bar{x}$ and a predictor $\hat{x}$, look for a stepsize σ satisfying

$$\phi(\sigma) < \frac{(2-\beta)h_{\max}}{2\,[\omega C]} \tag{4.6}$$

and start the inexact Newton iteration with initial guess $\hat{x}(\sigma)$. If the iteration fails, contrary to expectation, use the new Kantorovitch estimate to compute $[\omega C]$ and reduce the stepsize in order to satisfy (4.6). For scalar parameters, $p = 1$, and an order q predictor this strategy reduces to the well known stepsize correction formula (cf. [24], [30])

$$\sigma_{\text{new}} = \sqrt[q]{\frac{h_{\max}}{[h_0]}} \cdot \sigma_{\text{old}} \ .$$

II. Adaptive Multilevel Newton h-p Collocation

The two most successful approaches to the numerical solution of boundary value problems of ODEs seem to be the *local* approach by *multiple shooting* and the *global* one by *collocation*. Probably one of the best codes using multiple shooting is the Mulcon/Percon family by Deuflhard, Fiedler and Kunkel [31] and the recent version by Hohmann and Wulff [66] which incorporates the affine invariant accuracy matching strategy based on Theorem 3. The collocation approach is best represented by the continuation code Auto by Doedel and Kernevez [35] [36] and in particular Colcon by Bader and Kunkel [7] which is based on the famous collocation code Colsys by Ascher, Christiansen, Russell [2] [3] and Bader (Colnew [6]). Both collocation codes, Auto and Colcon, use collocation at Gaussian nodes and a similar adaptive mesh selection technique based on the equidistribution of the local discretization error. The main difference is that Auto is restricted to a fixed number of collocation points and provides no global accuracy control, whereas Colcon adaptively chooses the number of nodes necessary to achieve the required accuracy. Due to the fixed number of degrees of freedom, Auto is prone to effects caused by the discretization which may lead to even qualitatively wrong results (often called spurious solutions, see e.g. [55]). In this respect, Colcon with its fully adaptive mesh selection is much more reliable. In addition, Bader and Kunkel transfer the Gauss Newton continuation process as developed in [30] to the infinite dimensional setting of the BVP. Despite this infinite dimensional formulation, it is then only applied to the finite dimensional nonlinear problems obtained by the collocation discretization. In fact, they had no choice since they only considered the exact Gauss Newton process which is not realizable in the infinite dimensional case.

Not surprisingly, the accuracy matching and continuation philosophy of the new method to be presented below shares many ideas with the new (inexact) versions of the multiple shooting codes Mulcon and Percon. On the other hand, we also used collocation at Gaussian nodes but in a multilevel Newton setting, i.e., applied to the linear subproblems arising in the inexact Gauss Newton iteration corresponding to the infinite dimensional problem.

Another related method is the Newton multilevel continuation code PLT-

MGC by Bank and Chan [8] (see also Bank and Mittelmann [9]) for quasilinear elliptic PDEs in two dimensions. Here, the nonlinear problems obtained after discretization are on each level solved by an approximate damped Newton process (already mentioned in part I, see [11]) using the nested iteration strategy (full multigrid). The continuation method is a purely finite dimensional one based on the coarse grid discretization. Thus the user has to provide a coarse grid that is already fine enough to reflect the qualitative structure of the solution.

Comparing multiple shooting and collocation it is useful to discern two kinds of BVPs. Following a phrasing by Deuflhard, we call them *time-like* and *space-like* BVPs. The former are supposed to have a distinguished direction in the independent variable, as is the case if it really models physical time. Conversely, we call a BVP space-like, if no such direction is marked out or the qualitative behaviour of the solution (like stability properties, etc.) strongly changes along the given interval. This situation typically occurs if the independent variable models physical space, i.e., the BVP should be interpreted as a onedimensional PDE. Although this characterization has to remain vague, it may help to understand the different behaviour of the two methods. It is no surprise that multiple shooting works best for time-like problems since it depends on numerical integration of the underlying ODE in a prescribed direction. So, it is generally accepted that multiple shooting is an efficient method for time-like problems and strict accuracy demands, but needs good initial data (or lots of numerical insight) to converge at all. On the other hand, collocation (of fixed low order) is actually often used to produce these initial data, since the Newton method belonging to it converges much better. However, it becomes less efficient when it comes to strong accuracy demands. Moreover, space-like BVPs are mainly the realm of symmetric collocation methods. Here, their global point of view and the symmetry of the discretization really pays off. As an example, it is very hard to solve a transition layer problem (cf. example 6, 7) by the shooting method, even if it is known where the layer occurs (cf. Maier [48]), though the method is competitive for periodic solutions of ODEs as a typical example of a time-like BVP (cf. Wulff [66]).

The advantage of multiple shooting methods concerning small tolerances lies in the variable orders of the numerical integrators used to compute the flow and to solve the variational equation. Thus, it is indeed quite natural

to think of a collocation method with variable local orders which combines the advantages of both methods. This line of thought has already attracted much interest in the PDE community (cf. [40], [23], [50]), where the so-called h-p methods become more and more popular. The same idea also led to the very efficient numerical methods for countable systems of ODEs (so-called CODEs, cf. [67] and the new code MEDICI [61]). As we will see, we can borrow a lot from the experiences gained in these fields.

As the second main ingredient of our method, we apply the multilevel Newton idea, i.e., the discretization and the linearization by Newton's method change places. In contrast to the common approach, we first apply the inexact Newton method, as described in the first part, to the infinite dimensional problem and then discretize the linear subproblems. Thus, we separate, in a way, the nonlinear difficulties from the linear ones. The h-p collocation for the arising linear BVPs is based on local refinement by bisection and variable local orders. We preferred this standard approach from finite elements to regridding techniques, because we can store up local information due to the linearity of the BVP. Furthermore, the transition to BVPs in two or more dimensions seems to be much easier. Altogether, this leads to a new method for BVPs of ODEs that looks quite promising with regard to the problems tested so far.

A crucial point for the method is the correct formulation of the BVP as a nonlinear problem $F(x) = 0$. In Section 5 we discuss two formulations which correspond to the two kinds of problems introduced above. Time-like problems directly lead to a nonlinear Volterra integral equation, whereas a Fredholm equation, which automatically incorporates the boundary conditions, is more suitable for space-like problems. Unfortunately, the Fredholm formulation relies on the existence of a particular Green's function, so that we have to claim non degenerate linear boundary conditions (as usual for theoretical investigations). The formulation of the nonlinear problem governs the other parts of the method, from the local tasks down to the choice of local basis functions and the solution of the arising global linear systems.

Although the Volterra approach is in many aspects just a special case (corresponding to initial value boundary conditions) of the Fredholm formulation, in Section 6 we start the presentation of the collocation method with the former which is very close to the standard collocation method. Hence, we collect a series of facts from standard collocation theory. We use in par-

ticular Runge Kutta basis functions and discuss the recursive construction and usage of Runge Kutta schemes of collocation type.

The next section generalizes these ideas to the Fredholm formulation of the given BVP. Because we are confined to non degenerate linear boundary condition in this case, we consider (in contrast to Section 6) ODEs of m-th order so as to include the most interesting examples. The Fredholm approach canonically leads to a new local representation of the solution (using its m-th derivative and the local boundary condition) that is symmetric with respect to the boundary conditions and reduces to the Runge Kutta representation for initial value boundary conditions (i.e., the Volterra approach). Moreover, the global linear systems are most easily solved by a new successive elimination of local boundary conditions.

Based on this material, we present the main parts of the h-p collocation method for linear problems like the h-p error model, the residual estimator and the h-p (refinement and order selection) strategy. Here, only the local problems for the Volterra and the Fredholm approach differ.

5 Global Integral Formulations for BVPs

We consider a nonlinear boundary value problem for a first order ODE. For ease of notation we restrict ourselves to two point BVPs. The generalization to multi-point BVPs is straightforward. For the same reason we postpone the parameter dependent case to Section 5.3. Hence, we are looking for a solution of the ODE

$$x' = f(x,t) \tag{5.1}$$

subject to the (in general nonlinear) boundary conditions

$$r(x) = r(x(a), x(b)) = 0\,, \tag{5.2}$$

where f and r are continuously differentiable functions

$$f : X \times \mathbb{R} \longrightarrow X, \quad r : X \times X \longrightarrow Z$$

in $X = \mathbb{R}^n$ and $Z = \mathbb{R}^k$ (typically $k = n$).

The main task in this section is to derive an appropriate formulation of the BVP as a nonlinear problem $F(x) = 0$ that goes well with the collocation approach. The multiple shooting idea (see Section 3.3) automatically leads to

a finite dimensional nonlinear problem involving the flow of the ODE which is then approximated by an ODE solver. This is different from collocation as a global discretization method. The construction of a suitable grid for the solution, which implies a finite dimensional approximation of the problem, is part of the solution process itself. Thus, in contrast to multiple shooting, we first have to formulate the BVP in an infinite dimensional setting and then to discretize this problem. Actually, this line of thought leads further. Once we have the infinite dimensional problem at hand, we may also apply Newton's method in its inexact version to the infinite dimensional mapping, if it is Gâteaux differentiable. This leaves us with a sequence of *linear* subproblems (here: linear BVPs) that may be easier to solve by the collocation discretization to an accuracy prescribed by the surrounding Newton method.

5.1 Volterra Formulation

The first idea that comes to mind is to use the differential formulation (5.1) directly, defining

$$F(x) := \begin{pmatrix} x' - f(x,t) \\ r(x) \end{pmatrix} .$$

Remember, however, that a collocation solution is only continuous in general (e.g., for Gaussian nodes), so that x' is not defined over the whole interval. Hence, avoiding differentiation, we integrate the differential equation (5.1) arriving at the nonlinear Volterra equation (of second kind)

$$V(x) = 0, \quad \text{where} \quad V(x)(t) := x(t) - x(a) - \int_a^t f(x(s), s)\, ds\,. \tag{5.3}$$

Thus, the original BVP is equivalent (for differentiable x) to

$$F(x) := \begin{pmatrix} V(x) \\ r(x) \end{pmatrix} = 0\ .$$

Let us look for appropriate function spaces such that F is a well defined continuous and Gâteaux differentiable mapping. By $L^p = L^p(I)$ we denote the space of L^p functions $I \to \mathbb{R}^n$ on the finite interval $I = [a, b]$.

LEMMA 1. *Let $f : \mathbb{R}^n \times I \to \mathbb{R}^n$ be a continuous function statisfying*

$$\|f(x,t)\| \leq c(1 + \|x\|^{p/q}) \tag{5.4}$$

on $\mathbb{R}^n \times I$ for some $c \geq 0$ and $p > 1$, where $q = p/(p-1)$ is the dual index. Then, the Volterra operator

$$V(x)(t) := x(t) - x(a) - \int_a^t f(x(s), s)\, ds$$

is a continuous mapping $V : L^p \to L^s$ for all $s \leq p$.

Proof. The condition at infinity (5.4) guarantees that the Nemytskii operator $(Nx)(t) := f(x(t), t)$ maps L^p continuously into L^q (cf. [20]). Since $q > 1$ and $L^q \subset L^1$, the operator G defined by

$$(Gx)(t) := \int_a^t f(x(t), t)\, dt$$

is a continuous map

$$G : L^p \xrightarrow{N} L^q \xrightarrow{\int} L^\infty \subset L^s \ .$$

Moreover, we have $L^p \subset L^s$ and thus the Volterra operator is a continuous map $L^p \to L^s$. □

The lemma tells us that we have to choose L^∞ as the domain, if we want to include all functions f which are uniformly bounded with respect to x by some polynomial. Consequently, we assume in the sequel that f and its total derivative f' are bounded in x by

$$\|f(x,t)\| \leq c(1 + \|x\|^k) \ \text{ and } \ \|f'(x,t)\| \leq c(1 + \|x\|^k) \tag{5.5}$$

on $\mathbb{R}^n \times I$ for some $c, k \geq 0$. On the other hand, we may choose for the image space any L^p, $1 < p \leq \infty$. Moreover, we have to keep in mind that the boundary function r defined in (5.2) is only continuous on $C(I) \subset L^\infty$. Hence, the nonlinear function F, describing the nonlinear BVP, is a continuous function

$$F : C(I) \longrightarrow L^p \times Z, \quad x \longmapsto \begin{pmatrix} V(x) \\ r(x) \end{pmatrix}$$

for any $1 < p \leq \infty$. For the image space we have chosen $p = 2$, i.e., $L^2 \times Z$, since many iterative solvers (such as GMRES [56] and BICGSTAB [64]) are based on a Hilbert image space. Moreover, the effort for the computation of the corresponding norm via an appropriate quadrature formula is still feasible.

REMARK 9. An alternative would have been a formulation using the Sobolev space $H^1(I)$ as proposed by Bader and Kunkel [7]. Numerical experience has shown, however, that the differentiability imposed on the functions should be as weak as possible. If the initial guess for the solution of the nonlinear problem is a smooth function (e.g. a constant one), the derivatives often become worse at the beginning of the Newton process. Furthermore, the effort for the computation of the corresponding norm would almost double.

Since we required f and r to be C^1 functions, where $D_x f$ is as well bounded by some polynomial, we see by the same arguments that F is Gâteaux-differentiable. The derivative

$$F'(x) : C(I) \longrightarrow L^2 \times Z$$

at $x \in C(I)$ in direction $v \in C(I)$ is given by

$$F'(x)v = \begin{pmatrix} V'(x)v \\ r'(x)v \end{pmatrix} \tag{5.6}$$

where

$$(V'(x)v)(t) = v(t) - v(a) - \int_a^t D_x f(x(s), s)v(s)\, ds$$

and

$$r'(x)v = D_1 r(x(a), x(b))v(a) + D_2 r(x(a), x(b))v(b)\,.$$

Hence, the nonlinear problem $F(x) = 0$ meets the formal assumptions of the affine contravariant convergence theorems and thus is ready to be solved by an inexact Newton method (in the infinite dimensional setting). Here, the equation

$$F'(x)v = -F(x)$$

for the Newton correction v at x is the linear BVP given by the boundary condition

$$D_1 r(x(a), x(b))v(a) + D_2 r(x(a), x(b))v(b) + r(x(a), x(b)) = 0$$

and the linear (non autonomous) differential equation

$$\dot{v}(t) = -(\dot{x}(t) - f(x(t), t)) + D_x f(x(t), t)v(t)\,. \tag{5.7}$$

Thus we are lead to linear BVPs of the form

$$\begin{aligned} f(x,t) &= A(t)x + g(t) \\ r(x(a), x(b)) &= B_1 x(a) + B_2 x(b) + d = 0 \end{aligned} \tag{5.8}$$

where

$$A \in C(I, L(X)), \quad g \in C(I, X),$$

are time dependent mappings and

$$B_1, B_2 \in L(X, Z), \quad \text{and} \quad d \in Z\,.$$

This kind of problem is to be attacked by the h-p collocation solver. More precisely, we have to solve linear BVPs of the form (5.8) up to a prescribed accuracy, measured in the L^2-norm of the residual.

5.2 Fredholm Formulation

The Volterra formulation has, despite its simplicity, a major drawback. By taking the integral from the left boundary a to t, we introduce an artificial *direction*, although we praised the symmetry of collocation methods (at symmetric node) in the introduction. We are able to maintain this symmetry by switching over to the Fredholm formulation of the BVP which includes the boundary conditions in a natural way. Unfortunately, this approach relies on the existence of a Green's function, so that we have to claim *linear non degenerate* boundary conditions. In order to include the most interesting space-like problems we now consider ODEs of m-th order. Hence, we are looking for a solution of the differential equation

$$x^{(m)} = f(x,t) \;\text{ on } I = [a,b], \text{ where } f : \mathbb{R}^{mn} \times \mathbb{R} \to \mathbb{R}^n, \tag{5.9}$$

subject to the *linear non degenerate boundary conditions*

$$\beta x = B_1 x(a) + B_2 x(b) = z \in Z = \mathbb{R}^{mn}, \tag{5.10}$$

where $B_1, B_2 \in \mathbb{R}^{mn \times mn}$ and the restriction of β on the space $\mathbb{P}_{m-1} = \ker D^m$ of polynomials of degree less than m induces an isomorphism

$$\beta : \mathbb{P}_{m-1} \xrightarrow{\cong} Z = \mathbb{R}^{mn}. \tag{5.11}$$

Here, we almost abuse notation denoting by x not only the functions $x(t) \in \mathbb{R}^n$ but also (depending on the context) the column vector $x(t) \in \mathbb{R}^{mn}$ of the values of x and its derivatives $x', \ldots, x^{(m-1)}$ up to order $m-1$. Hence, equation (5.9) reads in full prose

$$x^{(m)}(t) = f(x(t), x'(t), \ldots, x^{(m-1)}(t), t)$$

and the boundary condition looks more explicitly like

$$\beta x = B_1 \begin{pmatrix} x(a) \\ x'(a) \\ \vdots \\ x^{(m-1)}(a) \end{pmatrix} + B_2 \begin{pmatrix} x(b) \\ x'(b) \\ \vdots \\ x^{(m-1)}(b) \end{pmatrix}.$$

For order $m = 1$ the non degeneracy condition means that the sum $B_1 + B_2$ of the boundary matrices is an invertible matrix $B_1 + B_2 \in \mathrm{GL}(n)$. The non degeneracy condition comes as no surprise since it is also the basic assumption for the analysis of collocation in the fundamental paper of de Boor and Swartz [18]. It guarantees the existence of a Green's function

$$G : [a, b]^2 \longrightarrow \mathbb{R}^{n \times n}$$

for the linear differential operator $L = D^m$ and the homogeneous boundary conditions $\beta x = 0$ characterized by

a) $\partial_t^m G(t, s) = \delta(t - s)$ on $[a, b]^2$

b) $\beta G(\cdot, s) = 0 \in \mathbb{R}^{mn}$ for all $s \in [a, b]$.

As above, we think of $G(t,s)$ also as a function in $\mathbb{R}^{mn\times n}$ being the column matrix $G = (G, \partial_t G, \ldots, \partial_t^{m-1} G)$ of G and its first partial derivatives up to order $m-1$. Following [18], we denote by

$$P : C^{m-1}(I) \longrightarrow \mathbb{P}_{m-1}$$

the canonical projection defined by $\beta P x = \beta x$. Hence, we have

$$x(t) = (Px)(t) + \int_a^b G(t,s) x^{(m)}(s)\, ds \quad \text{for all } x \in H_1^m(I)\,, \tag{5.12}$$

where $H_1^m(I)$ is just the space of functions for which the integral is well defined, i.e.,

$$H_1^m(I) := \{x \in C^{m-1}(I) \mid \ x^{(m-1)} \text{ abs. cont. and } x^{(m)} \in L_1(I)\}\,.$$

By (5.12) and the characteristic properties of Green's function we see at once that the BVP given by (5.9) and (5.10) is equivalent to the nonlinear equation $Fx = 0$, where F is the nonlinear Fredholm operator

$$(Fx)(t) := x(t) - (\beta^{-1} z)(t) - \int_a^b G(t,s) f(x,s)\, ds\,. \tag{5.13}$$

Assuming that f and its derivative are polynomially bounded, we see by the same arguments as for the Volterra operator (cf. Lemma 1) that F is a well defined Gâteau differentiable mapping

$$F : C^{m-1}(I) \longrightarrow L^2\,.$$

The derivative

$$F'(x) : C^{m-1}(I) \longrightarrow L^2$$

at $x \in C^{m-1}(I)$ in direction $v \in C^{m-1}(I)$ is given by

$$(F'(x)v)(t) = v(t) - \int_a^b G(t,s) D_x f(x,s)\, ds \tag{5.14}$$

REMARK 10. We would like to emphasize the fact that the Fredholm formulation directly incorporates the boundary conditions. Thus, it avoids the problem of choosing the correct norm on the product space $L^2 \times Z$ and $\mathbb{R}^{(k-1)n} \times Z$ occuring in the Volterra and multiple shooting formulation, respectively.

For convenience we give some examples of linear non degenerate boundary conditions and the corresponding projections and Green's functions.

Affine transformation onto the unit interval. We first recall some transformation properties with respect to the affine transformation

$$\psi : [a,b] \to [0,1], \quad t \mapsto \frac{t-a}{b-a}$$

of an interval $[a,b]$ onto the unit interval $[0,1]$ and its inverse

$$\phi = \psi^{-1} : [0,1] \to [a,b], \quad \tau \mapsto a + \tau(b-a)\,.$$

Let $\bar{\beta}$ denote the boundary map

$$\bar{\beta}x := B_1 x(0) + B_2 x(1)$$

with respect to the unit interval $[0,1]$ and $\bar{P}$ the corresponding projection, i.e.,

$$\bar{P} : C^{m-1}[0,1] \to \mathbb{P}_{m-1}, \quad \bar{\beta}\bar{P}x = \bar{\beta}x\,.$$

Moreover, suppose that $\bar{G}$ is Green's function for D^m and $\bar{\beta} = 0$. Then, we obtain (by trivial computation) the projection and Green's function corresponding to the interval $[a,b]$ by

$$Px = \bar{P}(x \circ \phi) \circ \psi \quad \text{and} \quad G(t,s) = (b-a)^{m-1} G(\psi(t), \psi(s))\,. \tag{5.15}$$

Hence, we only have to consider the unit interval and obtain the terms for any interval $[a,b]$ by a simple and cheap transformation (which will become important for the algorithm).

EXAMPLE 1. *First order BVP.* The linear boundary map is of the form

$$\beta x = B_1 x(a) + B_2 x(b)\,,$$

where $B_1, B_2 \in \mathbb{R}^{n\times n}$ and $\ker D = \mathbb{P}_0$ are the constant functions. Since

$$\beta(Px) = \beta x \;\Leftrightarrow\; (B_1 + B_2)Px = \beta x$$

the projection $P : C[a,b] \to \mathbb{P}_0$ is given by

$$P = (B_1 + B_2)^{-1}\beta x\,,$$

which is well defined iff the boundary condition β is non degenerate. Moreover, Green's function for the first order differential operator $Lx = x'$ and homogeneous boundary conditions $\beta = 0$ is easily established as the locally constant function

$$G(t,s) = \begin{cases} G_1 & \text{for } s \le t \\ G_2 & \text{for } s > t \end{cases}$$

where $G_1 = (B_1 + B_2)^{-1} B_1$ and $G_2 = G_1 - I$.

EXAMPLE 2. *Second order BVP with standard boundary conditions.* Let us consider the most common two point boundary conditions for second order problems, where the solution is fixed at the left and right boundary by given $x(a)$ and $x(b)$. This corresponds to the boundary map $\beta x = B_1 x(a) + B_2 x(b)$, where

$$B_1 = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad B_2 = \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix} \in \mathbb{R}^{2n \times 2n}.$$

With respect to the unit interval $[0,1]$, the projection $\bar{P} : C^1[0,1] \to \mathbb{P}_1$ and Green's function $\bar{G}$ for $Lx = x''$ are given by

$$(\bar{P}x)(t) = (1-t)x(0) + tx(1)$$

and

$$\bar{G}(t,s) = \begin{cases} s(t-1) & \text{for } s \le t \\ t(s-1) & \text{for } s > t \end{cases}$$

For an arbitrary interval $[a,b]$ this yields

$$(Px)(t) = \frac{b-t}{b-a} x(a) + \frac{t-a}{b-a} x(b)$$

and

$$G(t,s) = (b-a)\bar{G}(\psi(t),\psi(s)) = \frac{1}{b-a} \begin{cases} (s-a)(t-b) & \text{for } s \le t \\ (t-a)(s-b) & \text{for } s > t \end{cases}$$

EXAMPLE 3. *Initial value problem of m-th order.* As a trivial case, we think of an initial value problem for a m-th order ODE, where the left value $x(a), x'(x), \ldots, x^{(m-1)}(a)$ is given. By Taylor's formula,

$$x(t) = \sum_{k=0}^{m-1} \frac{1}{k!} x^{(k)}(a)(t-a)^k + \frac{1}{(m-1)!} \int_a^b x^{(m)}(s)(t-s)_+^{m-1} \, ds$$

we have

$$(Px)(t) = \sum_{k=0}^{m-1} \frac{1}{k!} x^{(k)}(a)(t-a)^k$$

and

$$G(t,s) = \frac{1}{(m-1)!}(t-s)_+^{m-1}\,.$$

Here, $(t-\cdot)_+^k$ denotes as usual the truncated monomial

$$(t-s)_+^k = \begin{cases} (t-s)^k & \text{for } s \le t \\ 0 & \text{for } s > t \end{cases}$$

5.3 Parameter Dependent Problems

So far we have neglected any parameter dependence of the differential equation and/or the boundary condition, although one of our objectives is the study of parameter dependent systems via continuation. This was mainly due to the fact that the parameter dependence spoils the formulae with nasty and lengthy expressions. This section gives a brief account of how to deal with parameters. In the following more technical sections we will always incorporate the parameter dependent case as necessary for the algorithmic realization. So, this section also fixes some notation for the forthcoming parts. For simplicity we only consider the Volterra approach and first order problems; ODEs of m-th order and the Fredholm formulation are handled in total analogy.

A parameter dependent (two point) BVP is given by a differential equation

$$x' = f(x,t,\lambda)$$

and a (in general nonlinear) boundary condition

$$r(x,\lambda) = r(x(a),x(b),\lambda) = 0\,,$$

where f and r are now continuously differentiable functions

$$f : X \times \mathbb{R} \times \Lambda \to X \quad \text{and} \quad r : X \times X \times \Lambda \to Z$$

defined on the product spaces with some finite dimensional parameter space $\Lambda = \mathbb{R}^q$. This yields the equivalent nonlinear Volterra equation

$$F(x,\lambda) = \begin{pmatrix} V(x,\lambda) \\ r(x,\lambda) \end{pmatrix} = 0\,,$$

where

$$V(x,\lambda)(t) = x(t) - x(a) - \int_a^t f(x(s), s, \lambda)\, ds\,.$$

In order to obtain a well defined Gâteaux differentiable mapping

$$F : C(I) \times \Lambda \to L^2 \times Z$$

we have again to assume that f and $D_x f$ are polynomially bounded in x for all $\lambda \in \Lambda$. The derivative

$$F'(x,\lambda) : C(I) \times \Lambda \to L^2 \times Z$$

at (x,λ) in direction $(v,\mu) \in C(I) \times \Lambda$ is then given by

$$V'(x,\lambda)\begin{pmatrix} v \\ \mu \end{pmatrix}(t) = v(t) - v(a) - \int_a^t \left[D_x f(x(s), s, \lambda)v(s) + (D_\lambda f(x(s), s, \lambda)\mu\right] ds$$

and

$$r'(x,\lambda)\begin{pmatrix} v \\ \mu \end{pmatrix} = B_1 v(a) + B_2 v(b) + C\mu\,,$$

where

$$B_i = D_i r(x(a), x(b), \lambda) \;\text{ for } i = 1, 2 \text{ and } \; C = D_\lambda r(x(a), x(b), \lambda)\,.$$

Thus, we now have to consider linear parameter dependent BVPs of the form

$$\begin{aligned} f(x,t,\lambda) &= A(t)x + P(t)\lambda + g(t) \\ r(x,\lambda) &= B_1 x(a) + B_2 x(b) + C\lambda + d = 0\,, \end{aligned}$$

where in addition to the terms in (5.8) we have $P \in C(I, L(\Lambda, X))$ and $C \in L(\Lambda, Z)$.

The rest of this section shows how free boundary value problems and in particular periodic boundary value problems may be incorporated into the present framework of parameter dependent BVPs.

Free boundary value problems. To include parameter dependent free boundary value problems, we use the affine transformation of $[a, b]$ on the standard interval $[0, 1]$

$$x(t) \to z(s), \quad \text{where} \quad t = a + s(b-a), \ s \in [0,1]$$

Introducing the extended parameter $\mu = (a, b, \lambda)$ we obtain an equivalent BVP without free boundaries consisting of the differential equation

$$z'(s) = g(z(s), s, \mu) \ , \quad \text{where} \quad g(z, s, \lambda) = (b-a) f(z, a + s(b-a), \lambda) \ ,$$

and the original boundary conditions

$$r(z(0), z(1), \mu) = r(z(0), z(1), a, b, \lambda) = 0 \, .$$

Obviously, this transformation implies for the derivatives with respect to the additional parameters a, b that

$$\begin{aligned} D_a g(x, s, \mu) &= -f(z, t, \lambda) + (b-a) D_t f(z, t, \lambda)(1-s) \qquad (5.16) \\ D_b g(x, s, \mu) &= f(z, t, \lambda) + (b-a) D_t f(z, t, \lambda) s \ , \end{aligned}$$

while the other derivatives of f have to be multiplied by $b - a$ to get the derivatives of g. Note that the second terms in (5.16) vanish for autonomous ODEs.

Periodic boundary value problems. Of particular interest for our investigations are periodic boundary conditions

$$r(x(0), x(T)) = x(0) - x(T) = 0 \ .$$

Here, the derivatives of the (linear) boundary conditions are extremely simple, $B_1 = I$ and $B_2 = -I$, and we have no explicit dependency on the parameter, i.e., $C = 0$. In the non autonomous case, the period T is given by the necessarily T-periodic right hand side f of the ODE, i.e., $f(x, t) = f(x, t + T)$. For autonomous equations, the period is an implicit free boundary, which we may handle as above considering the autonomous differential equation

$$z' = g(z, \mu) := T f(z, \lambda)$$

with the augmented parameter $\mu = (\lambda, T)$. The resulting derivatives are

$$D_z g = T D_x f \quad \text{and} \quad D_\mu = [D_T g, D_\lambda g] = [f, T D_\lambda f] \ .$$

6 Collocation: Volterra Approach

Proceeding, we consider the computation of the collocation solution on a fixed mesh with variable orders. We define an *h-p grid* $\Delta = (\{t_i\}, \{p_i\})$ as a partition

$$a = t_1 < t_2 < \cdots < t_{N+1} = b$$

of the basic interval $[a, b]$ into N subintervals $J_i = [t_i, t_{i+1}]$ of length $h_i = t_{i+1} - t_i$ and local orders p_i for $1 \leq i \leq N$. The collocation approach consists of two stages:

a) *local collocation:* On each subinterval J_i, we look for a polynomial that satisfies the differential equation at p_i inner collocation nodes, typically Gaussian nodes.

b) *global collocation:* The local polynomials have to define a continuous overall solution that satisfies the boundary condition. Hence, we have to claim continuity at the global mesh points t_i and the boundary conditions at a, b.

Since we shall present these two aspects in seperated sections, we use almost the same notation for the global mesh and the inner nodes, as depicted in figure 2. In the 'local' sections, we denote the local interval by $[a, b]$ and the inner collocation nodes by $t_1, \ldots, t_p$. We start our discussion of collocation

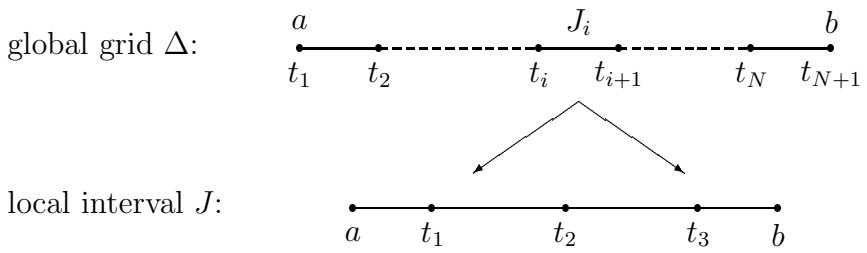


Fig. 2. Global and local collocation nodes (here: $p_i = 3$)

methods by recalling the standard techniques based on Runge Kutta basis functions. This standard approach, solving local initial value problems fitted together by continuity conditions, seems to be almost ideal for the Volterra

formulation (and thus, time-like BVPs). Since it is equivalent to multiple shooting using single step integrators, we may even use well-known techniques like block Gaussian elimination to solve the global linear systems. In Section 7 we derive the analogon for the Fredholm approach (hence, space-like BVPs) using a local boundary representation of the collocation solution. Aiming at an h-p collocation method we always keep an eye on variable orders on the local intervals. Moreover, thinking of a multilevel Newton method, it is sufficient to consider linear problems.

6.1 Runge Kutta Schemes of Collocation Type

A (collocation type) *Runge Kutta scheme* (A, b, c) of stage $p \geq 1$ on the interval $[0, 1]$ is given by p *nodes*

$$0 \leq c_1 < c_2 < \cdots < c_p \leq 1 \ ,$$

and the corresponding *weights* $b = (b_j)$ and *Runge Kutta coefficients* $A = (a_{ij})$ for $1 \leq i, j \leq p$, given by

$$b_j = \int_0^1 L_j(t) \ \mathrm{dt} \quad \text{and} \quad a_{ij} = \int_0^{c_i} L_j(t) \ \mathrm{dt} \ ,$$

where $L_1, \ldots, L_p \in \mathbb{P}_{p-1}$ are the Lagrange polynomials with respect to the nodes $c_1, \ldots, c_p$, i.e., $L_i(c_j) = \delta_{ij}$. We first discuss the so-called Runge Kutta basis which is the most suitable polynomial basis for collocation. Here, the general reference is [4]. Further, we present a simple recursive mechanism to compute the corresponding Runge Kutta coefficients, which is particularly suited for the successively increasing orders of the h-p collocation.

6.1.1 Runge Kutta basis

The *Runge Kutta basis* $\Phi = \{\phi_0, \ldots, \phi_p\}$ of the polynomial space $\mathbb{P}_p$ corresponding the nodes $c_1, \ldots, c_p$ is defined by

a) $\phi_0 \equiv 1$

b) $\phi_j(0) = 0$, $\phi_j'(c_i) = \delta_{ij}$ for $1 \leq i, j \leq p$.

By condition b) the derivatives ϕ_j' coincide for $j > 0$ with the Lagrange polynomials $\phi_j' = L_j$. Thus, we can evaluate ϕ_j according to

$$\phi_j(t) = \phi_j(t) - \phi_j(0) = \int_0^t \phi_j'(t)\ \mathrm{dt} = \int_0^t L_j(t)\ \mathrm{dt} \quad \text{for } j > 0.$$

For the nodes $t = c_i$ and $t = 1$, we recover in particular the Runge Kutta coefficients

$$\phi_j(1) = \int_0^1 L_j(t)\ \mathrm{dt} = b_j \ \text{ and } \ \phi_j(c_i) = \int_0^{c_i} L_j(t)\ \mathrm{dt} = a_{ij}\ .$$

Considering a polynomial $u \in \mathbb{P}_p$ in its *Runge Kutta representation* $u = \sum_{j=0}^p u_j \phi_j$, this implies $(1 \le i \le p)$

$$u(0) = u_0, \quad u(1) = u_0 + \sum_{j=1}^p b_j u_j, \quad u(c_i) = u_0 + \sum_{j=1}^p a_{ij} u_j \ \text{ and } \ u'(c_i) = u_i\ .$$

Of course, all these considerations transfer to a general interval $[a, b]$ of length $h = b - a$ using the affine transformation

$$[0, 1] \longrightarrow [a, b], \quad s \longmapsto a + sh\ .$$

The resulting nodes are $t_i = a + c_i h$ for $1 \le i \le p$ and the corresponding Runge Kutta coefficients

$$b_j^{[a,b]} = h b_j \ \text{ and } \ a_{ij}^{[a,b]} = h a_{ij}\ .$$

6.1.2 Computation of Runge Kutta schemes of collocation type

In this section we describe a simple recursive procedure to compute the Runge Kutta coefficients at Gaussian points for arbitrary order.

Computation of Gaussian nodes. To start with, we recall the well known method to compute the Gaussian points $c_1^{(n)}, \ldots, c_n^{(n)}$ as the zeros of the Legendre polynomials $P_n \in \mathbb{P}_n$ (transformed to $[0, 1]$). The Legendre polynomials may be defined by the three term recurrence formula $P_0 \equiv 1$, $P_1(x) = x$, and

$$k P_k(x) = (2k - 1) x P_{k-1}(x) - (k - 1) P_{k-2}(x) \ \text{ for } k \ge 2.$$

We consider the slightly more general case of a three term recurrence formula

$$\alpha_k^2 P_k(x) = (x - \beta_k) P_{k-1}(x) - \gamma_k^2 P_{k-2}(x) \quad \text{for } k \geq 2. \tag{6.1}$$

For the Legendre polynomials we have in particular

$$\beta_k = 0, \quad \alpha_k^2 = \frac{k}{2k-1} \quad \text{and} \quad \gamma_k^2 = \frac{k-1}{2k-1} \, .$$

In matrix notation this formula (6.1) reads $r = xp - T_n p$, where

$$r = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ \alpha_n^2 P_n(x) \end{pmatrix}, \quad p = \begin{pmatrix} P_0(x) \\ \vdots \\ \vdots \\ \vdots \\ P_{n-1}(x) \end{pmatrix},$$

and

$$T_n = \begin{pmatrix} \beta_1 & \alpha_1^2 & & & \\ \gamma_2^2 & \beta_2 & \alpha_2^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-1}^2 & \beta_{n-1} & \alpha_{n-1}^2 \\ & & & \gamma_n^2 & \beta_n \end{pmatrix} .$$

Hence, the problem of finding the zeros of P_n is equivalent to the eigenvalue problem of T_n, i.e.,

$$P_n(x) = 0 \iff T_n p = xp \, .$$

The matrix T_n can be transformed to an equivalent symmetric matrix $\hat{T}_n = DT_nD^{-1}$ by a diagonal transformation $D = \operatorname{diag}(d_1, \ldots, d_n)$, where $d_1 = 1$ and $d_k = \alpha_{k-1} d_{k-1}/\gamma_k$ for $2 \leq k \leq n$. We thus obtain the symmetric tridiagonal matrix

$$\hat{T}_n = \begin{pmatrix} \beta_1 & \alpha_1\gamma_2 & & & \\ \alpha_1\gamma_2 & \beta_2 & \alpha_2\gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{n-2}\gamma_{n-1} & \beta_{n-1} & \alpha_{n-1}\gamma_n \\ & & & \alpha_{n-1}\gamma_n & \beta_n \end{pmatrix} ,$$

whose (real and distinct) eigenvalues may be easily calculated by a standard QR method. Once given the zeros

$$-1 < x_1^{(n)} < x_2^{(n)} < \ldots < x_n^{(n)} < 1$$

of P_n as the ordered eigenvalues of $\hat{T}_n$, we obtain the Gaussian nodes $c_1^{(n)}, \ldots, c_n^{(n)}$ with respect to the interval $[0,1]$ via

$$c_k^{(n)} = \frac{x_k^{(n)} + 1}{2} \ .$$

Computation of Runge Kutta coefficients. We now assume that the Gaussian nodes $c_j^{(n)}$ are given for arbitrary stages n. The Runge Kutta coefficients are defined by

$$b_j^{(n)} = \int_0^1 L_j(t) \ \mathrm{dt} \quad \text{and} \quad a_{ij}^{(n)} = \int_0^{c_i^{(n)}} L_j(t) \ \mathrm{dt} \ , \tag{6.2}$$

where $L_1^{(n)}, \ldots, L_n^{(n)} \in \mathbb{P}_{n-1}$ are the Lagrange polynomials with respect to the nodes $c_1^{(n)}, \ldots, c_n^{(n)}$. Furthermore, we know that the Gaussian quadrature rule

$$\hat{I}_n(f) := \sum_{j=1}^{n} b_j^{(n)} f(c_j^{(n)}) \approx \int_0^1 f(t) \ \mathrm{dt}$$

is exact for polynomials f of degree $\leq 2n-1$. Hence, we can evaluate the integral expressions (6.2) for $n > 1$ using the Gaussian quadrature $\hat{I}_{n-1}$. The first coefficients $b_1^{(1)}$ and $a_{11}^{(1)}$ are of cause the trivial ones

$$b_1^{(1)} = \int_0^1 \underbrace{L_1(t)}_{=1} \ \mathrm{dt} = 1 \quad \text{and} \quad a_{11}^{(1)} = \int_0^{c_1^{(1)}} \underbrace{L_1(t)}_{=1} \ \mathrm{dt} = c_1^{(1)} \ .$$

Thus, we have all coefficients for arbitrary orders at hand.

REMARK 11. Once we know the Gaussian Runge Kutta formulae, we may use the same procedure to compute different Runge Kutta schemes of collocation type like the Lobatto schemes or the collocation schemes defined in Section 7.1.

6.2 Local Collocation

In this section we discuss the local collocation on a interval $[a, b]$ of length $h = b - a$ with the Runge Kutta scheme (A, b, c) of order p and the nodes $t_j = a + c_j h$, $1 \leq j \leq p$. Since we think of a Newton multilevel method, we only consider linear problems of the form defined in (5.8), i.e.,

$$f(x, t, \lambda) = A(t)x + P(t)\lambda + g(t) \; .$$

Our task may be described as follows: For a given $x_0 \in X$, compute the coefficients u_j of the polynomial $u = \sum_{j=0}^{p} u_j \phi \in \mathbb{P}_p$ with respect to the Runge Kutta basis $\Phi = \{\phi_0, \ldots, \phi_p\}$ satisfying

a) $u(a) = x_0$

b) $u'(t_i) = f(u(t_i), t_i, \lambda)$ for $1 \leq i \leq p$.

i.e., u is a polynomial approximation of the ODE's solution with initial value x_0 that solves the differential equation at the p nodes $t_1, \ldots, t_p$. By the definition of the Runge Kutta basis, we see that a) is equivalent to $u_0 = x_0$, while the second condition b) reads

$$\begin{aligned} u_i &= f(u_0 + h \sum_{j=1}^{p} a_{ij} u_j, t_i, \lambda) \\ &= A(t_i)u_0 + h \sum_{j=1}^{p} a_{ij} A(t_i) u_j + P(t_i)\lambda + g(t_i) \end{aligned}$$

for $1 \leq i \leq p$. Using the common matrix notation (cf. [4]), we obtain the linear system

$$Mu = Au_0 + P\lambda + g \; ,$$

where

$$M = I - h \begin{pmatrix} a_{11}A(t_1) & \cdots & a_{1p}A(t_1) \\ \vdots & & \vdots \\ a_{p1}A(t_p) & \cdots & a_{pp}A(t_p) \end{pmatrix} \; , \quad u = \begin{pmatrix} u_1 \\ \vdots \\ u_p \end{pmatrix} \; , \quad A = \begin{pmatrix} A(t_1) \\ \vdots \\ A(t_p) \end{pmatrix}$$

and P, g are defined in the same way as A. In particular, we get the value $x_1 = u(b)$ at the right boundary by

$$x_1 = x_0 + h \sum_{j=1}^{p} b_j u_j = Rx_0 + S\lambda + r \ ,$$

where $R = I + hDM^{-1}A$, $D = [b_1 I, \ldots, b_p I]$, $S = hDM^{-1}P$ and $r = hDM^{-1}g$. In this way, we have defined for each local interval $[a, b]$ and order p the affine *local collocation map*

$$\Psi : X \times \Lambda \longrightarrow X, \quad (x, \lambda) \longmapsto Rx + S\lambda + r \ ,$$

which in a way substitutes the flow used in multiple shooting methods, while the linear part R replaces the propagation matrix (the linearized flow).

6.3 Global Collocation

We now consider a grid $\Delta = (\{t_i\}, \{p_i\})$ given by a partition

$$a = t_1 < t_2 < \cdots < t_{N+1} = b$$

of the basic interval $[a, b]$ into N subintervals $J_i = [t_i, t_{i+1}]$ of length $h_i = t_{i+1} - t_i$ and local orders p_i for $1 \leq i \leq N$. By $\mathbb{P}_\Delta$ we denote the space of piecewise polynomials with corresponding degrees on the subintervals, i.e.,

$$u \in \mathbb{P}_\Delta \iff u|_{J_i} \in \mathbb{P}_{p_i} \text{ for all } i = 1, \ldots, N.$$

Our global collocation task is to find a continuous piecewise polynomial $u \in \mathbb{P}_\Delta$ that satisfies the boundary conditions and the local collocation conditions on each subinterval J_i. Equivalently, we look on each interval J_i for a polynomial $u_i \in \mathbb{P}_{p_i}$ such that

a) the overall solution is continous, i.e.,

$$u_{i-1}(t_i) = u_i(t_i) \text{ for } i = 2, \ldots, N$$

b) each u_i satisfies the differential equation at the local collocation points, i.e.,

$$u_i'(t) = f(u_i(t), t, \lambda)$$

for $t = t_i + c_j h_i$, $1 \leq i \leq N$ and $1 \leq j \leq p_i$,

c) the boundary conditions are fullfilled, i.e.,

$$r(u^{(1)}(a), u^{(N)}(b), \lambda) = 0 \ .$$

Again, we only consider the linear problem

$$f(x,t,\lambda) = A(t)x + P(t)\lambda + g(t) \ \text{ and } \ r(u,v,\lambda) = B_1 u + B_2 v + C\lambda - z \ .$$

On each subinterval J_i we have the affine local collocation map

$$\Psi_i : x_i \longmapsto x_{i+1} = R_i x_i + S_i \lambda + r_i \ ,$$

derived in Section 6.2, which guarantees the pointwise collocation conditions b). Hence, the global collocation problem is equivalent to the cyclic linear system

$$\underbrace{\begin{bmatrix} R_1 & -I & & & S_1 \\ & \ddots & \ddots & & \vdots \\ & & R_{N-1} & -I & S_{N-1} \\ B_1 & & & B_2 R_N & B_2 S_N + C \end{bmatrix}}_{=:H} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_N \\ \lambda \end{pmatrix} = - \begin{pmatrix} r_1 \\ \vdots \\ r_{N-1} \\ B_2 r_N - z \end{pmatrix} \ , \tag{6.3}$$

where the matrix H describes a linear mapping

$$H : X^N \times \Lambda \longrightarrow X^{N-1} \times Z \ .$$

Condensing and expanding. For problems of medium difficulty the cyclic linear system $Hy = b$, $y = (x, \lambda)$ given by (6.3) can be solved in the same way as in the multiple shooting context. We define a linear *expansion map*

$$R : X \times \Lambda \longrightarrow X^N \times \Lambda, \ \ (u, \lambda) \longmapsto R(u, \lambda) = (x, \lambda)$$

by the recurrence formula

$$x_1 = u \ \text{ and } \ x_{i+1} = R_i x_i - b_i + S_i \lambda \ \text{ for } i = 1, \ldots, N-1.$$

By the usual blockwise Gaussian elimination we can easily show that the original system is equivalent to a so-called *condensed system*. More precisely, we have

$$Hy = b \iff H_c \begin{pmatrix} u \\ \lambda \end{pmatrix} = b_c \text{ and } y = R(u, \lambda) \ ,$$

where $H_c = [E_c, S_c]$ is the condensed matrix given by $E_c = B_1 + B_2 R_N \cdots R_1$ and

$$S_c = C + B_2(S_N + R_N S_{N-1} + R_N R_{N-1} S_{N-2} + \cdots + R_N \cdots R_2 S_1) \ ,$$

and b_c is the condensed right hand side

$$b_c = b_N + B_2(R_N b_{N-1} + R_N R_{N-1} b_{N-2} + \cdots + R_N \cdots R_2 b_1) \ .$$

As in the multiple shooting context, this elimination technique becomes unstable if the local initial value problems are poorly conditioned and/or the number of intervals is too large. To detect this situation we can employ the condition estimates based on the iterative refinement sweeps according to Deuflhard and Bader [29]. If this situation has been detected, the global collocation system must be solved by a direct sparse solver such as SOLVERBLOK by de Boor and Weiss [19] for almost block diagonal systems (requiring separated boundary conditions) or a general sparse solver like MA28, MA42 by Duff [37].

7 Collocation: Fredholm Approach

The basic idea of the Runge Kutta basis as sketched in Section 6.2 is to characterize the local collocation polynomials $x \in \mathbb{P}_{p+m-1}$ by their m-th derivative $y = x^{(m)} \in \mathbb{P}_{p-1}$ (in Section 6 we always had $m = 1$) and left initial value $x(t_i) \in \mathbb{R}^{mn}$. The polynomial $y \in \mathbb{P}_{p-1}$ is then represented via its values at the p collocation nodes leading to the Lagrange basis. We see at once that the characterization by the left initial value introduces an asymmetry, since the local polynomials are 'fixed' at the left boundary of the corresponding subinterval $J_i = [t_i, t_{i+1}]$ as already observed by Bader [6]. Hence, we look for a more adequate local representation, subject to the following demands:

- It should respect the reflectional symmetry of the BVP in the independent variable, i.e.,

$$[a, b] \longrightarrow [b, a], \quad t \longmapsto \tau := a + b - t \; . \tag{7.1}$$

- The evaluation of the local polynomial should be cheap.
- The representation has to be stable.

The symmetry of the representation with respect to (7.1) means that it should make no difference whether we solve the original problem

$$x'(t) = f(x(t), t), \quad B_1 x(a) + B_2 x(b) = z \tag{7.2}$$

or the one obtained after applying the reflection (7.1),

$$\tilde{x}'(\tau) = -f(\tilde{x}(\tau), a + b - \tau), \quad B_2 \tilde{x}(b) + B_1 \tilde{x}(a) = z \; . \tag{7.3}$$

(For ease of notation we used an first order ODE.) Note that the boundary matrices B_1 and B_2 change places.

REMARK 12. We would like to point out that this kind of symmetry is different from Bader's notion in [6], where he considers a symmetry with respect to the values $x(t_i)$ and $x(t_{i+1})$, which does not involve the boundary condition.

By the symmetry of the Runge Kutta formulae of collocation type, the local collocation polynomials remain the same, if we reverse the direction of integration. Moreover, the m-th derivative $x^{(m)}$ of the collocation solution transforms canonically with the factor $(-1)^m$. The same is true for the characterization of $x^{(m)}$ by its values at the (symmetric) collocation nodes. Furthermore, we see that we have to incorporate the boundary matrices B_i in order to get a totally symmetric representation of the local collocation polynomials.

Fortunately, every non degenerate linear boundary map

$$\beta_i x = B_1^{(i)} x(t_i) + B_2^{(i)} x(t_{i+1}) \in \mathbb{R}^{mn} \tag{7.4}$$

gives us an isomorphism

$$\mathbb{P}_{p+m-1} \to \mathbb{P}_{p-1} \times \mathbb{R}^{mn}, \quad x \mapsto (y, z) = (x^{(m)}, \beta_i x)\,. \tag{7.5}$$

and thus a $1-1$ representation of the local polynomials $x \in \mathbb{P}_{p+m-1}$ defined on the local interval $J_i = [t_i, t_{i+1}]$ by its m-th derivative $x^{(m)}$ and the boundary value $\beta_i x \in \mathbb{R}^{mn}$. Hence, the question arises which local boundary map β_i is appropriate. The three criteria for an adequate local representation lead us to the boundary map

$$B_1^{(i)} := B_1 \quad \text{and} \quad B_2^{(i)} := B_2 \tag{7.6}$$

defined by the *global* boundary matrices B_1 and B_2. First of all we see that this characterization is symmetric, since B_1 and B_2 change places as we apply the transformation (7.1). Moreover, the effort for the evaluation of the local polynomials is still feasible, because the local boundary maps only differ by an affine transformation of the underlying interval (see below). Finally, numerical evidence shows that the local boundary representation is a stable characterization of the collocation polynomials.

We call the representation (7.5) with the local boundary map (7.6) the *local boundary representation* of the collocation solution.

REMARK 13. Obviously, this representation is strongly connected with the necessarily non degenerate boundary conditions of the Fredholm approach. In addition, we see that the standard Runge Kutta representation corresponds to (left) initial value boundary conditions ($B_1 = I$, $B_2 = 0$).

7.1 Collocation Schemes

To describe the local collocation problem we make use of the following lifting operators (cf. [18]). By D^{-m} we denote the integral operator

$$D^{-m} : C(I) \to C^m(I), \quad (D^{-m}y)(t) := \int_a^b G(t,s)y(s)\,ds$$

corresponding to the homogeneous boundary conditions $\beta = 0$. By the characteristic properties of Green's function, $y^{(-m)}(t) = (D^{-m}y)(t)$ may be equivalently defined by

$$D^m y^{(-m)} = y \quad \text{and} \quad \beta y^{(-m)} = 0\,.$$

By $D^{-m,z} := \beta^{-1}z + D^{-m}$ we denote its inhomogeneous counterpart corresponding to the boundary condition $\beta = z$, i.e., $y^{(-m,z)} = D^{-m,z}y$ satisfies

$$D^m y^{(-m,z)} = y \quad \text{and} \quad \beta y^{(-m,z)} = z\,.$$

Restricted to polynomials $y \in \mathbb{P}_{p-1}$, the operator $D^{-m,z}$ is just the inverse of the isomorphism (7.5). Using this lifting operator, the BVP is obviously equivalent to the nonlinear equation $Hy = 0$ for the m-th derivative $y = x^{(m)}$, where

$$(Hy)(t) = y(t) - f(y^{(-m,z)}, t)\,.$$

This characterization of the BVP was the starting point for the theoretical investigation in [18].

Since we are going to compute y in its Lagrange representation $y = \sum_{j=1}^p y_j L_j$, where $L_1, \ldots, L_p$ are the Lagrange polynomials for the nodes $t_1, \ldots, t_p$, we need the liftings $L_j^{(-m)}$ with respect to the boundary map β. Therefore, we introduce so-called *collocation schemes* which generalize the Runge Kutta schemes of collocation type (corresponding to left initial value boundary conditions). In addition, we define so-called *boundary schemes* describing the inverse β^{-1} of the boundary map.

DEFINITION 2. Let β be a non degenerate linear boundary condition on $[a,b]$, $G(t,s)$ the corresponding Green's function for D^m and $L_1, \ldots, L_p$ the

Lagrange polynomials for p distinct nodes $t_1, \ldots, t_p \in [a,b]$. We call the functions

$$\gamma_j : [a,b] \to \mathbb{R}^{mn \times n}, \quad \gamma_j(t) := L_j^{(-m)}(t) = \int_a^b G(t,s) L_j(s)\, ds$$

collocation schemes for β and $\{t_i\}$ on $[a,b]$. By $\Gamma(t)$ we denote the composed matrix

$$\Gamma(t) := [\gamma_1(t), \ldots, \gamma_p(t)] \in \mathbb{R}^{mn \times pn}\,.$$

The *boundary scheme for β on $[a,b]$* is defined as the function

$$C : [a,b] \to \mathbb{R}^{mn \times mn}$$

describing the inverse of the boundary map, i.e.,

$$(\beta^{-1} z)(t) = C(t) z \;\text{ for all } z \in \mathbb{R}^{mn}\,.$$

We recover the Runge Kutta coefficients a_{ij} as the collocation scheme $\gamma_j(t_i)$ evaluated at the nodes t_i, if we substitute for β the initial value boundary conditions:

$$a_{ij} = \int_a^{t_i} L_j(s)\, ds = \int_a^b (t_i - s)_+^0 L_j(s)\, ds = \gamma_j(t_i)\,.$$

Before we derive the local collocation equations in terms of the local boundary representation, we briefly look at the transformation properties of the collocation schemes with respect to affine transformations of the underlying interval $[a,b]$. This is quite a decisive point for our algorithm, since we want to compute the collocation schemes only once for a given boundary condition (and not for each interval). Using the notation from Section 5.2, we denote in addition the Lagrange polynomials for the nodes $c_i = \psi(t_i) \in [0,1]$ by $\bar{L}_j \in \mathbb{P}_{p-1}$ and the corresponding collocation schemes by $\bar{\gamma}_j$. Using the transformation properties of Green's function, we have

$$\partial_t^k G(t,s) = (b-a)^{m-1-k} D_1^k \bar{G}(\psi(t), \psi(s))$$

and

$$\begin{aligned} \int_a^b \partial_t^k G(t,s) L_j(s)\, ds &= (b-a)^{m-1-k} \int_a^b D_1^k \bar{G}(\psi(t),\psi(s)) L_j(s)\, ds \\ &= (b-a)^{m-k} \int_0^1 \partial_t^k \bar{G}(t,s) \bar{L}_j(s)\, ds\,. \end{aligned}$$

Hence, setting $h := b - a$ we obtain the collocation scheme γ_j for $[a, b]$ from the scheme for the unit interval $[0, 1]$ by the simple transformation

$$\gamma_j = \operatorname{diag}(h^m I, \ldots, hI)\, \bar{\gamma}_j \,.$$

The transformation of the boundary schemes depends of course on the boundary condition itself. Let us give two examples.

EXAMPLE 4. *First order BVP (see Example 1).* The inverse of the boundary map $\beta x = B_1 x(a) + B_2 x(b)$, where $B_1 + B_2 \in \mathrm{GL}(n)$, is simply

$$\beta^{-1} : \mathbb{R}^{mn} \to \mathbb{P}_{m-1} = \mathbb{P}_0 \,, \quad (\beta^{-1} z)(t) = (B_1 + B_2)^{-1} z \,.$$

Hence, the boundary scheme is

$$C(t) = (B_1 + B_2)^{-1} \in \mathbb{R}^{n \times n} \,.$$

EXAMPLE 5. *Second order BVP with standard boundary conditions (see Example 2).* Here, the inverse of the boundary map is given by

$$\beta^{-1} : \mathbb{R}^{mn} \to \mathbb{P}_{m-1}, \quad (\beta^{-1} z)(t) = \frac{b-t}{b-a} z_1 + \frac{t-a}{b-a} z_2 \,.$$

Therefore, we have

$$C(t) = \beta^{-1}(t) = \frac{1}{b-a} \begin{bmatrix} (b-t)I & (t-a)I \\ -I & I \end{bmatrix} \in \mathbb{R}^{2n \times 2n} \,.$$

7.2 Local Collocation

The local callocation task for the local boundary representation is to find a polynomial $x \in \mathbb{P}_{m+p-1}$ satisfying

a) $x^{(m)}(t_i) = f(x(t_i), t_i, \lambda)$ for $i = 1, \ldots, p$

b) $\beta x = z$,

where $t_1, \ldots, t_p \in [a, b]$ are p distinct nodes and $z \in \mathbb{R}^{mn}$ the local boundary value. Equivalently, we may look for a polynomial $y \in \mathbb{P}_{p-1}$ (being the m-th derivative $y = x^{(m)}$ of x) satisfying

$$y(t_i) = f(y^{(-m,z)}(t_i), t_i, \lambda) \quad \text{for } i = 1, \ldots, p, \tag{7.7}$$

where

$$y^{(-m,z)}(t) = (\beta^{-1}z)(t) + \int_a^b G(t,s)y(s)\,ds$$

is the lifting introduced above. Due to the multilevel Newton context we consider again only the linear case

$$f(x,t,\lambda) = A(t)x + P(t)\lambda + g(t)\ .$$

Using the collocation scheme defined above we may compute the lifting $y^{(-m,z)}$ of some polynomial $y = \sum_{j=1}^{p} y_j L_j \in \mathbb{P}_{p-1}$ explicitly by

$$y^{(-m,z)}(t) = C(t)z + \sum_{j=1}^{p} \gamma_j(t)y_j = C(t)z + \Gamma(t)y.$$

Here, we identify y with the column vector $(y_1,\ldots,y_p) \in \mathbb{R}^{pn}$ of its Lagrange coefficients. Since

$$\begin{aligned} y(t_i) &= A(t_i)y^{(-m,z)}(t_i) + P(t_i)\lambda + g(t_i) \\ &= A(t_i)(C(t_i)z + \Gamma(t_i)y) + P(t_i)\lambda + g(t_i) \end{aligned}$$

for $i = 1,\ldots,p$, the local collocation equation (7.7) reads like

$$My = ACz + P\lambda + g\ ,$$

where

$$M = I - \begin{bmatrix} A(t_1)\Gamma(t_1) \\ \vdots \\ A(t_p)\Gamma(t_p) \end{bmatrix} \in \mathbb{R}^{pn\times pn}\ , \quad AC = \begin{bmatrix} A(t_1)C(t_1) \\ \vdots \\ A(t_p)C(t_p) \end{bmatrix} \in \mathbb{R}^{pn\times mn}\ ,$$

$$P = \begin{bmatrix} P(t_1) \\ \vdots \\ P(t_p) \end{bmatrix} \in \mathbb{R}^{pn\times q} \ \text{ and } \ g = \begin{bmatrix} g(t_1) \\ \vdots \\ g(t_p) \end{bmatrix} \in \mathbb{R}^{pn}\,.$$

Accordingly, the solution $y \in \mathbb{R}^{pn}$ of the local collocation equation is

$$y = Rz + S\lambda + r\ ,$$

where $R = M^{-1}AC \in \mathbb{R}^{pn\times mn}$, $S = M^{-1}P \in \mathbb{R}^{pn\times q}$ and $r = M^{-1}g \in \mathbb{R}^{pn}$. To actually evaluate the lifted solution $x = y^{(-m,z)}$, we again have to apply the collocation schemes to obtain

$$\begin{aligned} x(t) &= C(t)z + \Gamma(t)y \\ &= (C(t)z + \Gamma(t)R)z + \Gamma(t)S\lambda + \Gamma(t)r\ . \end{aligned}$$

For the values of x at the (local) boundaries this leads in particular to

$$x(a) = Uz + U^{(\lambda)}\lambda + u \ \text{ and } \ x(b) = Vz + V^{(\lambda)}\lambda + v$$

where

$$U = C(a)\Gamma(a)R, \ \ U^{(\lambda)} = \Gamma(a)S, \ \ u = \Gamma(a)r$$

and

$$V = C(b)\Gamma(b)R, \ \ V^{(\lambda)} = \Gamma(b)S, \ \ v = \Gamma(b)r\ .$$

7.3 Global Collocation

Having derived the local collocation solutions with respect to the local boundary representation, it is fairly easy to set up the global collocation system. We consider as in Section 6.3 a grid $\Delta = (\{t_i\}, \{p_i\})$ on the basic interval $[a, b]$ consisting of N intervals $J_i = [t_i, t_{i+1}]$ of length $h_i = t_{i+1} - t_i$ and orders p_i. By $\Delta + k$ we denote the grid

$$\Delta + k := (\{t_i\}, \{p_i + k\})$$

obtained by adding the integer k to the local orders. Accodingly, $\mathbb{P}_{\Delta+k}$ is space of piecewise polynomials of local degrees $p_i + k$, i.e.,

$$u \in \mathbb{P}_{\Delta+k} \iff u|_{J_i} \in \mathbb{P}_{p_i+k} \ \text{ for all } i = 1, \ldots, N.$$

7.3.1 Derivation of the global collocation system

Including m-th order problems, we require the solution $x \in \mathbb{P}_{\Delta+m-1}$ to be $m-1$ times continuously differentiable. Therefore, we are looking for polynomials $x_i \in \mathbb{P}_{p_i+m-1}$ on J_i such that

a) the overall solution is in $C^{m-1}[a, b]$, i.e.,

$$x_{i-1}(t_i) = x_i(t_i) \in \mathbb{R}^{mn} \ \text{ for } i = 2, \ldots, N$$

b) the local collocation conditions

$$x_i^{(m)}(t) = f(x_i(t), t, \lambda) = A(t)x_i(t) + P(t)\lambda + g(t)$$

are satisfied for $t = t_i + c_j h_i$, $1 \le i \le N$ and $1 \le j \le p_i$,

c) the (global) boundary condition is fullfilled, i.e.,

$$B_1 x_1(a) + B_2 x_N(b) = z\ .$$

We are going to compute the global solution in terms of the local boundary representation

$$x_i = y_i^{(-m,z_i)}$$

where $y_i \in \mathbb{P}_{p_i-1}$ are the m-th derivatives and $z_i \in \mathbb{R}^{mn}$ the local boundary values with respect to the local boundary conditions

$$\beta_i x = B_1 x(t_i) + B_2 x(t_{i+1})\,.$$

As a consequence, we have to substitute x_i in the global collocation conditions by $y_i^{(-m,z_i)}$ and obtain in particular the local collocation equations

$$y_i(t) = f(y_i^{(-m,z_i)}(t), t, \lambda)$$

as in Section 7.2. Using the results of the last section, the continuity condition a) is equivalent to

$$V_{i-1}z_{i-1} + V_{i-1}^{(\lambda)}\lambda + v_{i-1} = U_i z_i + U_i^{(\lambda)}\lambda + u_i \in \mathbb{R}^{mn}$$

for $i = 2, \ldots, N$. Here, the indices refer of course to the local intervals J_i. Similarly, the global boundary condition c) reads

$$B_1(U_1 z_1 + U_1^{(\lambda)}\lambda + u_1) + B_2(V_N z_N + V_N^{(\lambda)}\lambda + v_N) = z\ .$$

Thus, we arrive at the following global collocation system in terms of the local boundary values $z_1, \ldots, z_N \in \mathbb{R}^{mn}$:

$$\begin{bmatrix} V_1 & -U_2 & & & & V_1^{(\lambda)} - U_2^{(\lambda)} \\ & V_2 & -U_3 & & & V_2^{(\lambda)} - U_3^{(\lambda)} \\ & & \ddots & \ddots & & \vdots \\ & & & V_{N-1} & -U_N & V_{N-1}^{(\lambda)} - U_N^{(\lambda)} \\ B_1U_1 & & & & B_1V_N & B_1U_1^{(\lambda)} + B_2V_N^{(\lambda)} \end{bmatrix} \begin{pmatrix} z_1 \\ \vdots \\ \vdots \\ \vdots \\ z_N \\ \lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ \vdots \\ b_N \end{pmatrix},$$

where

$$\begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} = \begin{pmatrix} u_2 - v_1 \\ \vdots \\ \vdots \\ u_N - v_{N-1} \\ z - B_1 u_1 - B_2 v_N \end{pmatrix}$$

7.3.2 Solution by elimination of local boundary values

The global collocation system can be solved using an appropriate sparse solve. This is, in particular, true for separated boundary conditions leading to the 'almost block diagonal' form of the global collocation matrix as for the standard approach using Runge Kutta basis functions (care has to be taken in the parameter dependent case, especially near turning points, e.g., using some deflation technique, cf. [15]).

Since we are going to construct the collocation grid via local refinement and order increasing, the following method may provide an efficient alternative. We exploit the hierarchical structure of the collocation grid using an *elimination of local variables* (here, local boundary values) up to the coarse grid, whose collocation system is then solved directly. One advantage of this approach, in the context of the adaptive h-p collocation to be described below, is that we only have to update the local collocation and elimination matrices where some refinement or change of order has taken place. Moreover, the elimination process may be interpreted to have successively solved local boundary conditions up the global one.

For the local elimination we only have to consider two intervals (obtained by bisection in the h-p algorithm), i.e., the case $N = 2$, which reads as

$$\underbrace{\begin{bmatrix} V_1 & -U_2 \\ B_1 U_1 & B_2 V_2 \end{bmatrix}}_{=: W} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 0 \\ z \end{pmatrix} + \underbrace{\begin{pmatrix} V_1^{(\lambda)} - U_2^{(\lambda)} \\ B_1 U_1^{(\lambda)} + B_2 V_N^{(\lambda)} \end{pmatrix}}_{=: N} \lambda + \underbrace{\begin{pmatrix} u_2 - v_1 \\ -B_1 u_1 - B_2 v_2 \end{pmatrix}}_{=: w}$$

For the local elimination we require W to be invertible (else we stop the local elimination and leave the intervals to the global solver) leading to

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = W^{-1} \begin{pmatrix} 0 \\ z \end{pmatrix} + W^{-1} N \lambda + W^{-1} w \; .$$

Setting

$$W^{-1} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$$

this yields

$$z_1 = W_{12}z + W_1N\lambda + W_1w \quad \text{and} \quad z_2 = W_{22}z + W_2N\lambda + W_2w \; .$$

As a consequence, we obtain the values at left and right boundaries by

$$\begin{aligned} x(a) &= U_1z_1 + U_1^{(\lambda)}\lambda + u_1 \\ &= U_1(W_{12}z + W_1N\lambda + W_1w) + U_1^{(\lambda)}\lambda + u_1 \\ &= \underbrace{U_1W_{12}}_{=U} z + \underbrace{(U_1W_1N + U_1^{(\lambda)})}_{=U^{(\lambda)}} \lambda + \underbrace{U_1W_1w + u_1}_{=u} \end{aligned}$$

and

$$\begin{aligned} x(b) &= V_2z_2 + V_2^{(\lambda)}\lambda + v_2 \\ &= V_2(W_{22}z + W_2N\lambda + W_2w) + V_2^{(\lambda)}\lambda + v_2 \\ &= \underbrace{V_2W_{22}}_{=V} z + \underbrace{(V_2W_2N + V_2^{(\lambda)})}_{=V^{(\lambda)}} \lambda + \underbrace{V_2W_2w + v_2}_{=v} \; . \end{aligned}$$

and thus eliminated z_1, z_2. Concerning the computational effort note that we have to invert an $2n \times 2n$ matrix in each elimination step meaning roughly $8n^3$ multiplications. If the grid was constructed from a single interval by successive bisection ($N = 2^k$ for some $k \in \mathbb{N}$) this has to be done $N-1$ times leading to an effort of $O(Nn^3)$. Because every direct solver has at least to touch the N dense blocks of size $n \times n$ using some $O(n^3)$ elimination process, our new successive elimination method is up to a constant factor optimal for a direct method. As already mentioned above, the situation is actually much better in the context of the h-p collocation, since we only have to update the local matrices changed by refinement or order increasing.

8 Adaptivity Issues

8.1 Residual Estimation

In the framework of the inexact Newton method, we have to be able to compute the norms $\|F(x_k)\|$ and $\|r_k\|$ of the outer and inner residuals, respectively. Obviously, the computation of the residual norms of the linear subproblems is a special case of its nonlinear counterpart. In our particular situation, we have either the nonlinear function

$$F : C(I) \longrightarrow L^2 \times Z, \quad x \longmapsto \begin{pmatrix} V(x) \\ r(x) \end{pmatrix},$$

defined by the Volterra operator

$$V(x)(t) := x(t) - x(a) - \int_a^t f(x(s), s)\, ds \tag{8.1}$$

and the boundary function

$$r(x) := r(x(a), x(b)) ,$$

or the Fredholm formulation $F : C(I) \to L^2$, where

$$(Fx)(t) = x(t) - (\beta^{-1} z)(t) - \int_a^b G(t, s) f(x(s), s)\, ds\,. \tag{8.2}$$

For ease of notation, we ignore the parameter λ in this section. The norm $\|r(x)\|$ poses no problem, since it is the (problem dependent) norm in the finite dimensional space $Z = \mathbb{R}^m$. However, it is impossible to compute the L^2-norm of the Volterra or Fredholm integral exactly using only *discrete information* about the nonlinear function f. Since the Volterra integral is only a special case of the Fredholm integral (with Green's function $G(t, s) = (t - s)_+^{m-1}$), we consider in this section only the latter.

8.1.1 Local residuals

The residuals are not only necessary for the inexact Newton method but also needed to control the local adaptive refinement and order selection. Accordingly, we begin with the simpler task to estimate the L^2 norms of

the *local residuals* Fx, where $x \in \mathbb{P}_{p+m-1}$ and the expression in (8.2) is to be understood locally as in Section 7.2. To derive the residual estimate, we first reformulate the local collocation task in terms of some interpolation operators.

DEFINITION 3. For given p distinct nodes $t_1, \ldots, t_p \in [a, b]$ we denote the interpolation operator by

$$\pi_p : C(I) \to \mathbb{P}_{p-1} , \quad (\pi_p y)(t_i) = y(t_i) \quad \text{for } i = 1, \ldots, p.$$

Moreover, we define the *lifted interpolation* operator $\pi_p^{(-m)}$ by

$$\pi_p^{(-m)} : C^m(I) \to \mathbb{P}_{p+m-1}^{(0)} , \quad \pi_p^{(-m)} x = D^m(\pi_p x^{(m)}) ,$$

where $\mathbb{P}_{p+m-1}^{(0)}$ is the subspace of polynomials satisfying the homogeneous boundary conditions, i.e.,

$$\mathbb{P}_{p+m-1}^{(0)} = \{x \in \mathbb{P}_{p+m-1} \mid \ \beta x = 0\} .$$

More explicitly, the operator $\pi_p^{(-m)}$ looks like

$$(\pi_p^{(-m)} x)(t) = \int_a^b G(t, s)(\pi_p x)(s)\, ds .$$

Using this operator, we can reformulate the local collocation task for a given local boundary value $z \in \mathbb{R}^{mn}$ as follows: look for a polynomial $x \in \mathbb{P}_{p+m-1}$, such that

$$\beta(Fx) = 0 \quad \text{and} \quad \pi_p^{(-m)}(Fx) = 0 .$$

Note that

$$D^m(Fx) = x^{(m)} - fx \quad \text{and} \quad \beta(Fx) = \beta x - z .$$

Regarding this characterization, it might be a good idea to approximate the residual Fx by the lifted interpolant $\pi_k^{(-m)}(Fx)$ at different nodes t_i. In other words, we introduce new collocation points and use these to approximate the residual. We will see that this approximation results in a fairly good residual estimate. In what follows, let $\hat{\pi}_k$ denote the interpolation operator corresponding to distinct nodes $\hat{t}_1, \ldots, \hat{t}_k$. We first show that for certain polynomial like right hand sides of the differential equation and k big enough, the approximation of the residual Fx by $\hat{\pi}_k^{(-m)}(Fx)$ is even exact.

LEMMA 2. *Let $f(x,t)$ be of the form*

$$f(x,t) = A(t)x + g(t)$$

for some polynomials $A \in \mathbb{P}_l$ and $g \in \mathbb{P}_{p+l+2(m-1)}$, where $l \geq 0$. If in addition $k \geq p+l+m$, then

$$\hat{\pi}_k^{(-m)}(Fx) = Fx$$

for all $x \in \mathbb{P}_{p+m-1}$ satisfying $\beta x = 0$.

Proof. The particular form of f guarantees that the Nemytskii operator $(fx)(t) := f(x(t),t)$ maps $\mathbb{P}_{p+m-1}$ in $\mathbb{P}_{p+l+m-1}$. Hence, the integral expression is in $\mathbb{P}_{p+l+2m-1}$. The boundary condition $\beta x = z$ yields $\beta(Fx) = 0$. Since $\hat{\pi}_k^{(-m)}$ is the identity on $\mathbb{P}_{k+m-1}^{(0)}$ and $k \geq p+l+m$, we have the result. ∎

This result justifies the residual estimate

$$\varepsilon_k := [\|Fx\|_{L^2}] := \|\hat{\pi}_k^{(-m)}(Fx)\|_{L^2} \,. \tag{8.3}$$

Note that we had to require f to be polynomially bounded in x anyway for the integral operator to be well defined. Since $\hat{\pi}_k^{(-m)}(Fx)$ is a polynomial, the L^2 norm can be computed exactly, e.g., using an appropriate quadrature formula.

For linear problems we may also exploit the fact that $\pi_p^{(-m)}(Fx)$ vanishes and us a *saturation property* to derive the common inequalities for this kind of estimates:

LEMMA 3. *Let f be linear and let $x \in \mathbb{P}_{p+m-1}$ be the local collocation solution. Moreover, assume that the saturation property*

$$\|\hat{\pi}_k^{(-m)}(Fx) - Fx\| \;\leq\; \theta \,\|\pi_p^{(-m)}(Fx) - Fx\|$$

is satisfied for some $0 \leq \theta < 1$. Then we have

$$(1-\theta)\,\|Fx\| \;\leq\; \|\hat{\pi}_k^{(-m)}(Fx)\| \;\leq\; (1+\theta)\,\|Fx\|\,.$$

Proof. We use nothing but the triangle inequality to obtain

$$\begin{aligned} \|\hat{\pi}_k^{(-m)}(Fx)\| &\leq \|Fx\| + \|\hat{\pi}_k^{(-m)}(Fx) - Fx\| \\ &\leq \|Fx\| + \theta\,\|\underbrace{\pi_p^{(-m)}(Fx)}_{=0} - Fx\| \;=\; (1+\theta)\,\|Fx\|\,. \end{aligned}$$

The left inequality follows in the same way. □

Let us briefly depict the actual algorithmic realization of the residual estimate. We restrict ourselves to the most important cases $m = 1, 2$. For linear problems we use $k = p + 2$ and the nodes

$$\{\hat{t}_1, \ldots, \hat{t}_k\} := \{a, t_1, \ldots, t_p, b\}\,,$$

i.e., we simply add the local boundaries to the (Gaussian) nodes of the collocation scheme. Thus, we only have to evaluate the linear right hand side f of the ODE at the nodes of the global grid Δ. Moreover, the computation of the L^2 norm $\|\hat{\pi}_k^{(-m)}(Fx)\|$ becomes rather cheap, since $\pi_p^{(-m)}(Fx) = 0$.

For nonlinear problems we use the nodes of the collocation scheme of order $p + m$, i.e., take $\hat{\pi}_k = \pi_{p+m}$ leading to a very reliable residual estimate. The numerical tests actually showed that the residual estimate is very accurate as soon as the grid contains any information about the solution.

8.1.2 Global residuals

We now consider the approximation of the global residual. To this end we have to transfer the local notions of the last section to the global situation. Thus, we define the global interpolation operator

$$\pi_\Delta : C_\Delta \to \mathbb{P}_{\Delta-1}\,, \quad (\pi_\Delta y)_i = \pi_{p_i} y \in \mathbb{P}_{p_i} \ \text{ for } i = 1, \ldots, N$$

and the lifted interpolation operator

$$\pi_\Delta^{(-m)} : C_\Delta^m \to \mathbb{P}^{(0)}_{\Delta+m-1}\,, \quad \pi_\Delta^{(-m)} x = D^{-m}(\pi_\Delta x^{(m)})\,.$$

Here, C_Δ^m denotes the space of piecewise m times continuously differentiable maps on the grid Δ and $\mathbb{P}^{(0)}_{\Delta+m-1}$ the space of piecewise polynomials $x \in \mathbb{P}_{\Delta+m-1}$ satisfying the global homogeneous boundary conditions $\beta x = 0$. Using this operator, the global collocation task means to look for a polynomial $x \in \mathbb{P}_{\Delta+m-1} \cap C^{m-1}(I)$ such that

$$\beta(Fx) = 0 \ \text{ and } \ \pi_\Delta^{(-m)}(Fx) = 0\,.$$

As in the local case this formulation inspires the approximation of the residual Fx by the lifted interpolant $\hat{\pi}^{(-m)}_{\Delta+k}(Fx)$ for some $k \geq 1$ and local nodes $\hat{t}_1, \ldots, \hat{t}_{p_i+k}$. As above we obtain the following exactness result.

LEMMA 4. *Let $f(x,t)$ be of the form*

$$f(x,t) = A(t)x + g(t)$$

for some polynomials $A \in \mathbb{P}_l$ and $g \in \mathbb{P}_{p+l+2(m-1)}$, where $l \geq 0$ as in lemma 2. If in addition $k \geq l+m$, then

$$\hat{\pi}_{\Delta+k}^{(-m)}(Fx) = Fx$$

for all $x \in \mathbb{P}_{\Delta+m-1} \cap C^{m-1}(I)$ satisfying $\beta x = 0$.

This justifies the global residual estimate

$$\varepsilon_k := [\|Fx\|_{L^2}] := \|\hat{\pi}_{\Delta+k}^{(-m)}(Fx)\|_{L^2}$$

being exact under the assumptions of lemma 4. For linear problems we may again employ $\pi_{\Delta}^{(-m)}(Fx) = 0$ and a saturation property to derive the corresponding inequalities for the global residual estimate.

Let us sketch the algorithmic realization. For the additional local nodes we choose the same as for the local residuals. Now, we have to evaluate

$$(\hat{F}x)(t) := x(t) - (\beta^{-1}z)(t) - \int_a^b G(t,s)(\hat{\pi}_{\Delta+k}fx)(s)\,ds$$

at the local quadrature points needed for the computation of the L^2 norm. Obviously, it is much to costly to compute the global integral expression for each evaluation. Hence, we first evaluate $\hat{F}x$ at the left boundary

$$(\hat{F}x)(a) := x(a) - (\beta^{-1}z)(a) - \int_a^b G(a,s)(\hat{\pi}_{\Delta+k}fx)(s)\,ds$$

and then use the Volterra integral to evaluate $\hat{F}x$ at any t from left to right by

$$(\hat{F}x)(t) := x(a) + \int_a^t (s-t)^{m-1}(\hat{\pi}_{\Delta+k}fx)(s)\,ds\ .$$

Although this formula violates the symmetry of the Fredholm formulation it seems to excusable since it is only employed for the global residual estimate and neither for the actual solution nor the local residual estimate that rules the h-p mechanism.

8.2 H-p Strategy

So far we are able to compute the global collocation solutions of a linear BVP on a fixed grid $\Delta = (\{t_i\}, \{p_i\})$ and to estimate the local as well as the global residuals. We now consider one of the crucial points of any adaptive h-p method: How should the algorithm decide which grid is appropriate to achieve the required accuracy? Of course, we would like to obtain the result with as little effort as possible. Here, we adopt the standard strategy that already led to very efficient codes for PDEs and CODEs (cf. Babuška and Rheinboldt [5]). The main difference is that our objective is the minimization of the residual (as required by the surrounding Newton iteration) rather than the error. Looking at an adaptive mesh selection, the local residual has the advantage that it is big where errors are *created*, in contrast to the local error which may also become large due to transport effects. Thus, a large local residual indicates the neccessity for a local improvement, e.g., by refinement or higher order.

Starting with a coarse initial grid and low orders, we try to *equidistribute the local residual* by locally choosing a higher order or subdividing a subinterval by bisection (see figure 3), leading to a sequence of nested grids, the

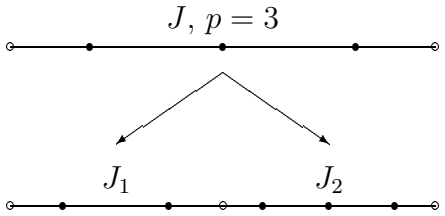


Fig. 3. Bisection of a local interval with new orders $p_1 = 2$ and $p_2 = 3$

so-called *levels*. To obtain the optimal partition of the main interval $[a, b]$ (h strategy) and orders (p strategy), we use *a priori residual estimates* based on an h-p *error model* and minimize *work per accuracy* measured by the product of the residual estimates and the corresponding *amounts of work*. Roughly, we proceed for each level as follows:

algorithm 1.

1. *Compute the local collocation matrices.*
2. *Compute the global collocation solution.*
3. *Compute the local residual estimates.*
4. *Check for convergence (overall residual estimate less than the required accuracy)*
5. *Compute the local coefficients of the error model. (For it may be neccessary to compute a local solution of lower order.)*
6. *Choose for each subinterval the optimal refinement and order.*
7. *Compute a threshold value which is to be the biggest local residual of the next level.*
8. *Apply the optimal refinement and order (chosen in step 6.) to all subintervals whose current residual estimate is bigger than the threshold.*

Steps 6 to 8 are responsible for the equidistribution of the local residual. We only refine a subinterval or increase its order, if the residual is still too big. Observe that the local collocation matrices are kept, if the corresponding subinterval (inclusive its order) remains unaltered. Thus, we directly use information computed on the previous levels. Proceeding further, we give the notions mentioned above a more precise meaning.

Amount of work. To start, we have a look at the amount of work needed for the local collocation on a subinterval (see Sections 6.2 and 7.2). We only count multiplications in terms of $O(pn^2)$, where n is the dimension of the state space X. Accordingly, we have to consider the following operations:

1. Evaluation of $A(t_1), \ldots, A(t_p)$ needing an effort of about Cpn^2, where the constant C depends on the given problem.
2. Computation and LR-decomposition of W: $\frac{1}{3}(pn)^3$
3. Computation of $W^{-1}A$ and $W^{-1}c$: $\frac{mn+1}{2}(pn)^2$

Alltogether, we end up with an amount of work of

$$\mathcal{A}(p) = \frac{1}{3}(pn)^3 + \frac{mn+1}{2}(pn)^2 + Cpn^2$$

for the local collocation on a single interval of order p. For the elimination of a local boundary condition (in case of the Fredholm formulation) we need $26(mn)^3/3$ multiplications which has to be taken into account for refinement.

Error model. Given a subinterval J of length $\bar{h}$ and order $\bar{p}$, we would like to know the local residual obtained for different stepsizes h and orders p. To this end, we construct a *local h-p error model* $\varepsilon(h,p)$ that depends on three parameters to be estimated in the algorithm. By standard collocation theory we know that the error of the collocation solution with respect to the norm $\|\cdot\|_\infty$ is $O(h^p)$ for sufficiently smooth solutions. By continuity, the same estimate holds for the residual measured in the L^2-norm, i.e.,

$$\|Fx\| \leq Ch^p\,,$$

where F is the Fredholm operator as defined in Section 5.2. Of course, the constant C contains bounds for the higher derivatives depending on p so that the error model $\varepsilon(h,p) = Ch^p$ with the single parameter C is not realistic. The standard choice for fixed order methods (e.g. linear finite elements) is

$$\varepsilon(h) = Ch^\gamma$$

including a second parameter $\gamma > 0$. Combining this approach with a third term describing the variable order, we are lead to the h-p error model

$$\varepsilon(h,p) = Ch^\gamma\alpha^p \tag{8.4}$$

depending on the three parameters $C, \alpha, \gamma \geq 0$. The *stepsize coefficient* γ characterizes the influence of refinement while the *order coefficient* α is responsible for order variations. Once we know the coefficients α and γ, we obtain the desired estimate for the local residuals by

$$\varepsilon(h,p) = \varepsilon(\bar{h},\bar{p})\left(\frac{h}{\bar{h}}\right)^\gamma \alpha^{p-\bar{p}}\,.$$

To compute the order coefficient α, we use the residual estimate for the order $p = \bar{p} - 1$ leading to

$$\alpha = \sqrt[p-\bar{p}]{\frac{\varepsilon(\bar{h}, p)}{\varepsilon(\bar{h}, \bar{p})}} = \frac{\varepsilon(\bar{h}, \bar{p})}{\varepsilon(\bar{h}, \bar{p} - 1)} \ .$$

Using this estimate for α, we may employ any other residual estimate to compute γ by

$$\gamma = \log_{h/\bar{h}} \left(\frac{\varepsilon(h, p)}{\varepsilon(\bar{h}, \bar{p})} \alpha^{\bar{p}-p} \right) \ .$$

Optimal order and refinement. Now we have the main tools at hand to choose an optimal order and refinement for the next level. Since the error model is only feasible in a neighbourhood of the current stepsize and order $(\bar{h}, \bar{p})$, we only consider pairs (h, p) from a so-called *order-stepsize window* (cf. the *order* window in [25])

$$W(\bar{h}, \bar{p}) := \{(\bar{h}, \bar{p} + 1\} \cup \left\{ (\bar{h}, p) \mid \ \bar{p}/2 + 1 \leq p \leq \bar{p} \right\} \ .$$

In other words, we either increase the order by one or refine the interval and choose a new order $\bar{p}/2 + 1 \leq p \leq \bar{p}$. We call an h-p pair $(h, p) \in W(\bar{h}, \bar{p})$ *optimal*, if it minimizes the *work per accuracy* measured by the amout of work times the error model (as the expected residual), i.e.,

$$\varepsilon(h, p) \cdot \mathcal{A}(p) \cdot \frac{\bar{h}}{h} \ = \ \text{min!} \tag{8.5}$$

Here, we have to take into account that the local amount of work doubles if we subdivide a subinterval. Moreover, we add the effort for the elimination of the local boundary values in that case. Nonetheless, we neglect the increased effort neccessary to solve the rest of the *global* system, if more subintervals are present. This is part of our quite conservative strategy to use high orders only if they really pay off. We have seen in Section 8.1 that high orders may be very efficient but could also become dangerous if the problem is not sufficiently smooth. Therefore, we prefer refinement to higher orders.

Note that (8.5) is by far not the only choice for a measure of the work per accuracy. The expected residual $\varepsilon(h, p)$ in (8.5) may be substituted by any $\phi(\varepsilon(h, p))$, where ϕ is a monotonously increasing function. As an example,

we may consider the number of binary digits locally gained from one level to the next, i.e.,

$$\log_2(\varepsilon_{\text{last}}/\varepsilon(h,p))$$

and thus try to maximize

$$\log_2(\varepsilon_{\text{last}}/\varepsilon(h,p)) \cdot \frac{h}{\bar{h}} \cdot \frac{1}{\mathcal{A}(p)} = \max! \tag{8.6}$$

Here, we assume that the code may be viewed as a (linear) encoding machine (cf. Deuflhard [25] for the ODE case). In fact, we tried several optimality functions such as (8.6) and got almost identical final h-p grids. What is changed, is the history of the adaptive mesh selection. The simplest strategy (8.5) was the winner requiring less levels to arrive at the final grid that represents the solution.

Refinement and order selection. Our refinement and order selection is based on the principle to change only subintervals whose residual estimate is greater than the maximal residual that we expect for the optimal refinement and order pair. Moreover, we want the local residual of the next level to be at least reduced by the quotient of the required accuracy tol and the current global residual ε. The latter strategy is employed to avoid too many levels. Accordingly, we define a *threshold* value ε_{cut} for the local residuals by

$$\varepsilon_{\text{cut}} := \min\left(\frac{\text{tol}}{\varepsilon}\max_J \varepsilon(J),\ \max_J \varepsilon_{\text{opt}}(J)\right)$$

and apply the optimal refinement and order to all subintervals J satisfying

$$\varepsilon(J) > \varepsilon_{\text{cut}}\ .$$

8.3 Multilevel Newton Method

The last sections give us a black box solver for the solution of a linear BVP up to a prescribed residual. The result is a collocation solution on an adaptively chosen h-p grid. As the final task, we have to fit this linear solver into the framework of the inexact Newton method (in the infinite dimensional setting). In particular, we have to answer the following questions:

a) Which grid should represent the solution of the nonlinear problem?

b) Which initial coarse grids should be given to the linear solver?

c) How can a solution be transferred to another grid?

d) What does it mean to add solutions of different grids?

e) How to implement the multilevel continuation method?

We will see that the answers to these questions are quite obvious if we consequently follow the inexact Newton approach.

Initial Grid for the Linear Solver. To start with the second question, we consider each linear subproblem of the surrounding Newton method as a completely new linear BVP. Accordingly, we leave the selection of an appropriate grid to the adaptive collocation and start all linear subproblems with the same coarse grid, the *basic grid*. In general, this basic grid will be a uniform partition of the main interval into a small number of subintervals and a common initial order, say $p = 1$ or $p = 2$.

Lifting Solutions on Finer Grids. We may define the following *partial ordering* on the set of grids. Let Δ_1 and Δ_2 be to h-p grids on $[a, b]$. We say that Δ_1 is *coarser* than Δ_2, in symbols $\Delta_1 \subset \Delta_2$, if

a) each subinterval of Δ_2 is included in a subinterval of Δ_1

b) the order of any subinterval J of Δ_1 is less than or equal to the smallest order of all subintervals of Δ_2 that are subsets of J.

Equivalently, we say that Δ_2 is *finer* than Δ_1. Obviously, this ordering was chosen in order to obtain the inclusion

$$\mathbb{P}_{\Delta_1} \subset \mathbb{P}_{\Delta_2}$$

of the corresponding spaces of piecewise polynomials, if $\Delta_1 \subset \Delta_2$. Hence, a solution on Δ_1 may be *lifted* to a finer grid Δ_2.

Addition of Solutions of Different Grids. To add to piecewise polynomials on the same grid (with the same orders), we only have to add the local coefficients. Now consider two piecewise polynomials $u_1 \in \mathbb{P}_{\Delta_1}$ and $u_2 \in \mathbb{P}_{\Delta_2}$ living on grids Δ_1 and Δ_2 that result from a common coarse grid by bisection and increasing of local orders. Then, there is a coarsest grid Δ that is finer than Δ_1 and Δ_2, i.e.

$$\Delta_1 \subset \Delta \quad \text{and} \quad \Delta_2 \subset \Delta \; .$$

To construct Δ, we simply have to locally choose the smallest subintervals and highest orders of Δ_1 and Δ_2. Thanks to the common coarse grid and the bisection, there are no overlaps. Hence, the sum of u_1 and u_2 is a piecewise polynomial $u = u_1 + u_2 \in \mathbb{P}_\Delta$ living on Δ whose coefficients may be obtained as the sum of the coefficients of the u_i lifted to Δ.

H-p Grid for the Solution of the Nonlinear Problem. Now the first question is already answered, since the Newton iterates u_k are computed as the sum of the initial solution u_0 and the solutions s_k of the linear subproblems. Hence, the solution of the nonlinear problems is defined on the coarsest grid which is finer than the grid of the initial solution u_0 and all grids needed to solve the linear subproblems. To be consistent with the linear solver, the initial solution should be defined on a grid obtained from the basic grid by bisection.

Multilevel Continuation Method. In the context of a continuation method, the multilevel h-p collocation offers us at each continuation point a collocation solution $x \in \mathbb{P}_{\Delta_x}$ and an h-p tangent $t \in \mathbb{P}_{\Delta_t}$ being the normalized solution of the homogeneous linearized BVP. The grids Δ_x and Δ_t are the (in general different) grids obtained by the nonlinear process for x and the linear h-p collocation for t. The tangential continuation

$$\hat{x} = x + st \in \mathbb{P}_\Delta$$

gives us an initial guess $\hat{x} \in \mathbb{P}_\Delta$ for the next Gauss Newton iteration defined on the union $\Delta = \Delta_x \cup \Delta_t$ of these two grids. However, taking $\hat{x}$ directly as the initial guess would lead to finer and finer grids in the coarse of the continuation process. In order to avoid this, we adopt a technique developed by Wulkow [67] for time dependent problems. We interpolate the prediction

$\hat{x}$ on the grid obtained by merging every two sons of the fine grid. On the resulting intervals we choose the minimum of the orders of the subintervals. Thereby, we avoid too many intervals without loosing much information from one continuation step to the next.

III. Implementation and Numerical Examples

9 Object Oriented Implementation

As is common in the scientific computing field, a large portion of time has been consumed in the implementation of the numerical ideas and as usual this is not reflected in this paper, since a full documentation of the program would be too costly (the whole package presently consists of some 300 pages of source code). As we presented not a single algorithm but a basic method with lots of applications, several numerical tools were needed. The following list does not claim completeness.

- basic matrix and vector operations including some numerical linear algebra, in particular the computation of the Moore Penrose pseudoinverse for the solution of underdetermined linear equations
- integrators for ODEs (and the corresponding variational equations used in multiple shooting), where we used adaptive extrapolation codes
- h-p collocation for linear problems using
  - Runge Kutta and collocation schemes
  - hierarchical h-p grids
  - h-p control mechanism
- abstract inexact Newton method (including the accuracy matching along the lines of Section 3)
- continuation methods for different types of solutions
  - equilibria (finite dimensional nonlinear problem)
  - Hopf points (additional computation of the subspace of the emanating periodic branch)
  - periodic solutions (computed by the multilevel Newton h-p collocation)

In our oppinion, it is almost impossible to combine all these modules in an acceptable time span if not using an object oriented programming environment. The well-known principles of 'good' programming like modularity, data encapsulation, efficiency, simplicity and compactness are most easily achieved using the abstract data and inheritance concepts of object oriented programming. Thus, a large part of the traditional programming task is left to the compiler.

Although these ideas are already widely used and accepted in computer science, object oriented programming is still in its developing stages in scientific computing. This fact is mainly due to the overwhelming variety of very efficient (procedural) FORTRAN libraries, which so far present simply the best numerical codes one can get. Another reason, in our opinion, is that the standardization of object oriented languages like C++ is still in progress and has come to a satisfactory state only recently (say, by the definition of C++ 3.0), a fact which is reflected in the available compilers.

This situation is going to change soon as already well established object oriented numerical class libraries like Rogue Wave's Linpack.h++ (based on the famous LINPACK FORTRAN-based library) show. We have chosen C++ which seems to be the most widespread object oriented programming language as the language of implementation. It also has the advantage that it includes C (with lots of numerical codes available) and shares some very useful features for numerical applications (like built-in complex numbers) with FORTRAN. One of the most outstanding features of C++ for scientific computing is the operator overloading mechanism, which enables us to implement complex algorithms in an abstract way employing almost the mathematical language used to describe it. As an example, we may define arithmetic operators like '+' on grids representing functions in an appropriate adaptively defined function space. Last, but no least, I acted on a personal affinity which I developed for giving 'life' to abstract mathematical objects (e.g., efficiently working C++ classes) which is really good fun.

The whole package is based on a matrix vector library which was developed during the last two years including linear algebra functions like LR and QR decomposition. We did not use commercial libraries like Linpack.h++, since theses libraries were not fully available at the beginning of the project and because the resulting code should be public domain. Some classes use ideas from the well-known 'Numerical Recipes in C' [53] by Press et al., while the

QR decomposition and the computation of the pseudoinverse is based on [34] (see also [33]).

The integration classes are based on a new implementation of the adaptive extrapolation codes EULSIM, EULEX, and DIFEX in C (cf. [45]) which are now available as C++ classes (cf. [44]) incorporating the improved stepsize control and some added functionality like continuous output according to Hairer and Ostermann [41]. The latter is used in the multiple shooting code for the solution of variational equations along a given trajectory.

The continuation module uses data structures developed for a two dimensional continuation algorithm (cf. [43]) which are also used in the new versions of PERCON ([66]) and ALCON ([47]). The collocation part of the program was developed in the course of the mathematical derivation of the h-p collocation method.

As an example we document here the class for the abstract inexact Newton method from which all Newton methods in the package are derived. We hope that this example illustrates the improved structural clarity and readability (even for someone not used to the C++ syntax) of the object oriented implementation. Compared to the actual code we only excluded the print options. The following header is used by the other modules:

```
class Newton {
public:
    Newton();

    void Accuracy(Real val) { tol = val; }
    void Exactness(Real beta);

    Real Accuracy() const { return tol; }
    Int Iterations() const { return k; }

    void InitialResidual(Real val) { fFirst = val; }
    void InitialMonotonicity(Real val) { hFirst = val; }

    Real ConvergenceFactor() const;
protected:
    virtual Bool Function(Bool first, Real& norm) = 0;
    virtual Bool Solver() = 0;
    virtual void AddCorrection() = 0;

    Bool Start();
```

```
    Bool Step();
    Bool Solve();

    Real RelativeAccuracy() const { return eps; }
    Real AbsoluteAccuracy() const { return delta; }
private:
    Real tol, beta, eps, delta;
    Real hMin, hMax, hFirst, fFirst, rhoEnd;
    Int k, kMax, red, redMax;
    Bool converged;
    RealVec f, h;

    void PrepareNextStep();
};
```

The pure virtual functions *Function*, *Solver* and *AddCorrection* realize the function evaluation, the (inexact) solution of the linear Newton equation and addition of the Newton correction, respectively. These functions have to be provided by the 'user', i.e., the derived Newton methods. So, the multilevel Newton h-p collocation realizes the linear solver as an h-p collocation operating on grids, whereas the Newton method for finite dimensional problems simply uses an LR or QR decomposition. On the other hand, the user supplied function may employ the relative and absolution accuracy requirements *RelativeAccuracy* and *AbsoluteAccuracy* provided by the abstract method.

The implementation of the inexact Newton method would look like the following:

```
Newton::Newton() :
  kMax(30), r(0, kMax-1), h(-1, kMax+1), redMax(10)
{
    tol  = 10 * sqrtEpsMach;
    beta = 1;
    hMax  = (2-beta)/(1+beta);
    hMin  = 1e-3;
    fFirst = 1;
    hFirst = 0.1;
    rhoEnd = 0.5;
}

void Newton::Exactness(Real val) {
    beta = val;
```

```
    hMax = (2-beta)/(1+beta);
}

Real Newton::ConvergenceFactor() const {
    return (converged || k==0) ? hMax/h(0) : hMax/h(k);
}

Bool Newton::Start() {
    k=0;
    h.Clear();
    f.Clear();
    h(0) = hFirst;
    f(0) = fFirst;
    eps = beta * h(0) / 2;   // first required relative accuracy
    delta = eps * f(0);      // first required absolute accuracy
    Bool done = Function(true, f(0));
    if (done) {
        PrepareNextStep();
        if (converged) h(0) = hMin;
    }
    return done;
}

Bool Newton::Solve() {
    Bool done = Start();
    while (done && !converged) done = Step();
    return done;
}

Bool Newton::Step() {
    if (k>kMax) return false;
    if (!Solver()) return false;
    AddCorrection();
    if (!Function(false, f(k+1)) return false;
    h(k) = hMax * f(k+1) / f(k);
    if (h(k) < hMax) return false;
    h(k+1) = sqr(h(k)) / hMax : hFirst;
    k++;
    PrepareNextStep();
    return true;
}

void Newton::PrepareNextStep() {
    converged = r(k)<=tol;           // check convergence
```

```
    f(k+1) = f(k) * h(k) / hMax;     // a priori residual
    eps = beta * h(k) / 2;           // required relative accuracy
    delta = eps * f(k+1);            // required absolute accuracy
    delta = Max(delta, rhoEnd*tol);  // soften requirement for last step
}
```

10 Illustrative Examples

In this section we give some examples for the new multilevel Newton h-p collocation algorithm, illustrating the adaptive process of the linear and nonlinear iterations. Due to the completely different approach to accuracy control and error (here: residual) estimation, it is hard to compare the new code to the well established collocation codes such as Colsys and Colnew. As an example, the required accuracy is not comparable, since these codes estimate the error in a maximum norm, whereas the new h-p collocation method controls the residual of the nonlinear integral equation in the L^2-norm. Moreover, we meet the problem of comparing an h-p method versus a regridding technique with fixed order. We can always find a situation in which one code beats the other one. We only have to start the regridding methods with a small order for a scalar equation and small tolerances in order to make h-p collocation look superior. This seems unfair. On the other hand, choosing a higher initial order, we lose the main advantage of the h-p method which automatically takes the appropriate local orders. Another problem lies in the different types of problems tackled by the codes. The new code directly solves parameter dependent underdetermined problems (periodic solutions of autonomous ODEs being the prototype), whereas the standard codes do not consider the underdetermined case.

Another problem is posed by the new implementation of the numerical algorithm in C++. In the preceeding section we discussed the impact of the advanced features of modern programming languages on the development of highly abstract adaptive algorithms. The main result was that it takes much less time to implement a new adaptive method in an object oriented environment. But on the other hand, this advantage has its price in that an organizational overhead follows. Obviously, programming languages lacking these features (such as dynamical data structures, operator overloading, etc.) do not need this overhead. This fact particularly pays off for small examples. Thus, a C++ code relies more on the underlying libraries and a posteriori optimization which we have not employed as yet.

Nonetheless, the new code is already competitive for the problems tested so far. This is most significantly demonstrated by the h-p grids obtained in the adaptive process. In particular, the code performs equally well for difficult space-like problems such as singular perturbed equations as well as

for time-like problems such as periodic solutions of autonomous ODEs. Due to the problems discussed above, we do not provide an explicit comparison (e.g. COCON versus COLSYS or COLCON) in terms of CPU time, but give the absolute times for the new code for the examples. In our experience, the new package is at most three times slower than COLSYS.

All calculation were run on a SUN SPARC 10 workstation using the SPARCWORKS 3.0 C++ compiler without optimization. For the graphical presentation we used the invaluable MATLAB system.

We start to examine the numerical results by looking at some model problems that illustrate the performance of the main elements of the new method, the h-p collocation for linear problems and the multilevel Newton method for nonlinear problems based on the Fredholm (Example 7) and the Volterra formulation (Example 8).

EXAMPLE 6. *Linear transition layer.* We consider the second order boundary value problem given by

$$\begin{aligned} &\varepsilon x'' + tx' = -\varepsilon\cos(\pi t) - \pi t\sin(\pi t) \ \text{ on } [-1,1] \\ &x(-1) = -2 \ \text{ and } \ x(1) = 0\,. \end{aligned} \tag{10.1}$$

This is a well-known test problem (see e.g. [4] Example 9.2) whose solution has a rapid transition layer at $t = 0$ for small $0 < \varepsilon \ll 1$. Figure 4 shows the solution for $\varepsilon = 10^{-6}$ (as computed by the h-p collocation). In figure 5 we illustrate the adaptive solution process by a series of h-p grids automatically constructed by the algorithm for $\varepsilon = 10^{-4}$ and a required residual of tol $= 10^{-6}$. The left column shows the approximate collocation solution on the indicated level while the right column displays the corresponding h-p grid. Here, we plotted the local orders versus the midpoints of the intervals. As expected, the algorithm chooses high orders for the scalar problem and the mesh points collect near the boundary layer. The whole solution process takes 2 seconds. Starting with a finer grid of ten intervals, it is less than a second.

EXAMPLE 7. *Nonlinear transition layer.* Our second example is again a a well-known test problem taken from [4] (Example 9.8). We consider the

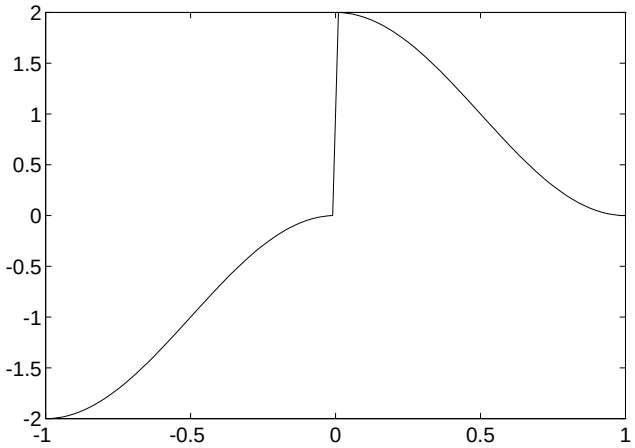


FIG. 4. Solution of (10.1) for $\varepsilon = 10^{-6}$

scalar nonlinear BVP given by

$$\begin{aligned} &\varepsilon x'' + xx' - x = 0 \quad \text{on } [0,1] \\ &x(0) = x(1) = \frac{1}{2}\,. \end{aligned} \tag{10.2}$$

Here, the solution has for small $0 < \varepsilon \ll 1$ a rapid transition layer at $t = 0$. Figure 6 shows the solution for $\varepsilon = 10^{-4}$ which we obtained without continuation and ten initial intervals. Figure 7 present the solution process for $\varepsilon = 10^{-3}$ and a required residual of tol $= 10^{-6}$. We display the iterates of the multilevel Newton method together with the corresponding h-p grids, i.e., the successively finer grids obtained by adding the inexact Newton corrections. Again, high orders are very efficient, since we consider a scalar problem. Moreover, the transition layer at $t = 0$ is easily recognized by the multilevel Newton method. Figure 8 displays the corresponding Newton corrections obtained as the h-p collocation solutions of the linear subproblems. Here we see that the grids for the linear subproblems may be much coarser then the resulting grid of the nonlinear solution (obtained as the union of the correction grids). Hence, the linear systems are much easier to solve than the linear problems obtained by the standard approach 'linearization after discretization'.

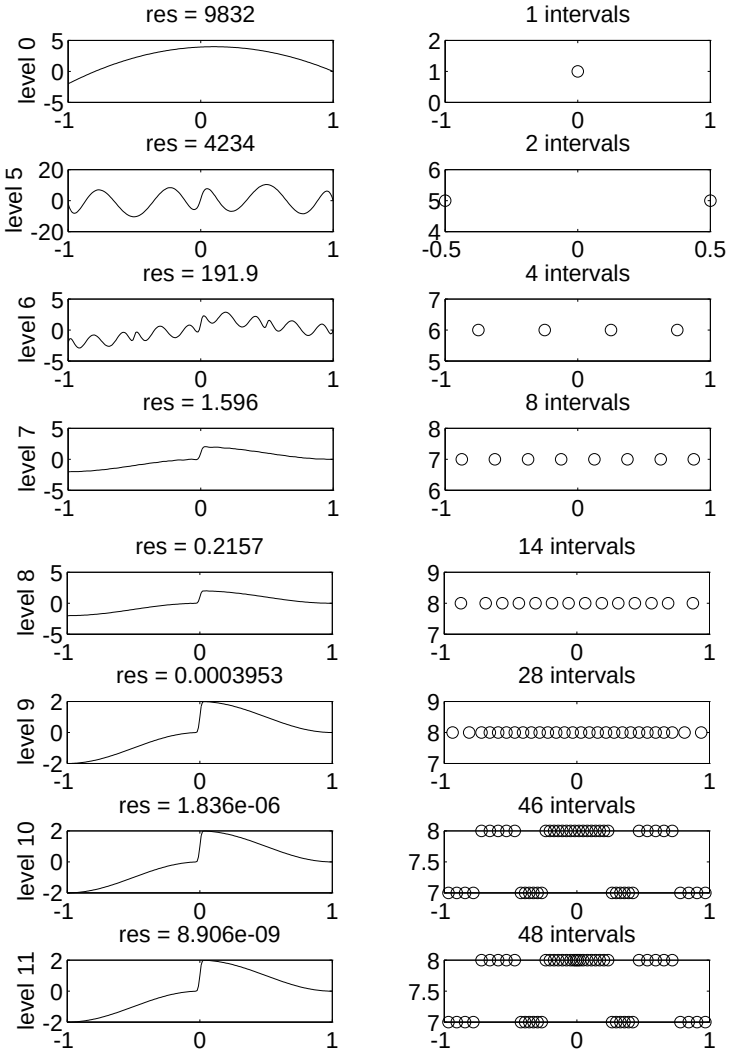


FIG. 5. Adaptively chosen grids for (10.1) with $\varepsilon = 10^{-4}$ and tol $= 10^{-6}$

EXAMPLE 8. *Chemical Oscillator.* Next, we return to the chemical oscillator which we already introduced as a stiff system of five ordinary differential equations (see Section 2, Equation (2.20)). We now want to compute a periodic solution using the multilevel Newton collocation. Since the independent variable models physical time, the Volterra approach is the appropriate formulation. Figure 9 shows the four Newton iterates necessary for a required

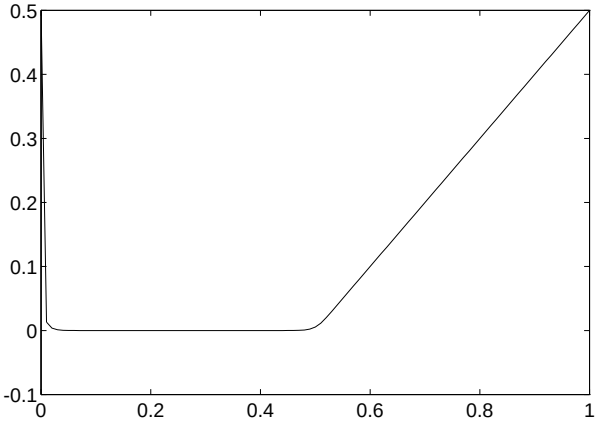


FIG. 6. Solution for (10.2) with $\varepsilon = 10^{-4}$

accuracy tol $= 10^{-3}$. Here, we plotted x_5 versus x_4. As initial guess we took the linear interpolant at 5 points which we obtained by an integration from $t = 0$ to $t = 3$ using the extrapolation code EULSIM. The initial value was $x = (9, 7, 5, 0.05, 0.1)$ and as initial guess for the period we set $T = 3$. The first picture clearly shows that this initial guess is far away from the periodic solution. Using multiple shooting (without damping) we would stand no chance in obtaining a solution. Figure 10 presents the development of the grids in the solution process. The left column displays the component x_4 versus the normalized time and the right column shows the corresponding grids.

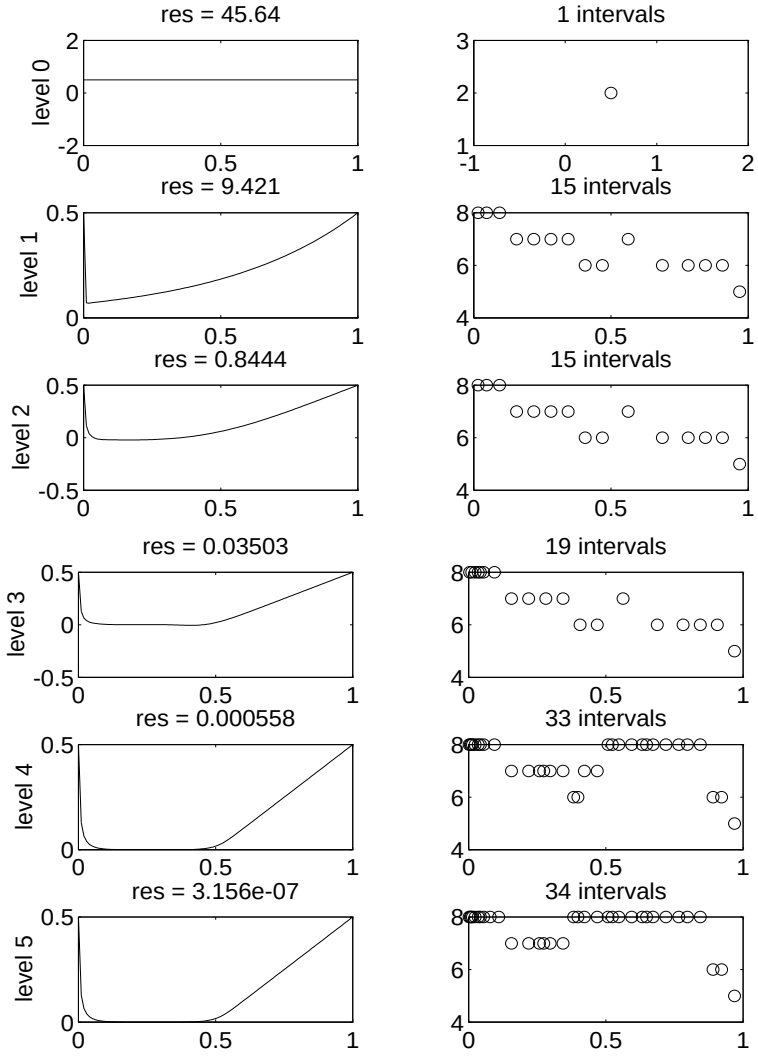


FIG. 7. Adaptively chosen grids for (10.2) with $\varepsilon = 10^{-3}$ and $\text{tol} = 10^{-6}$

11 Limit Cycles of a Railway Bogie

In this section we analyze the dynamical behaviour of a model describing the motion of a bogie on rails. The model was originally proposed by Cooperrider [16]. Our calculations are based on the model used by True and Kaas-Petersen ([63], [62]). For a discussion of different models see also Moelle [49].

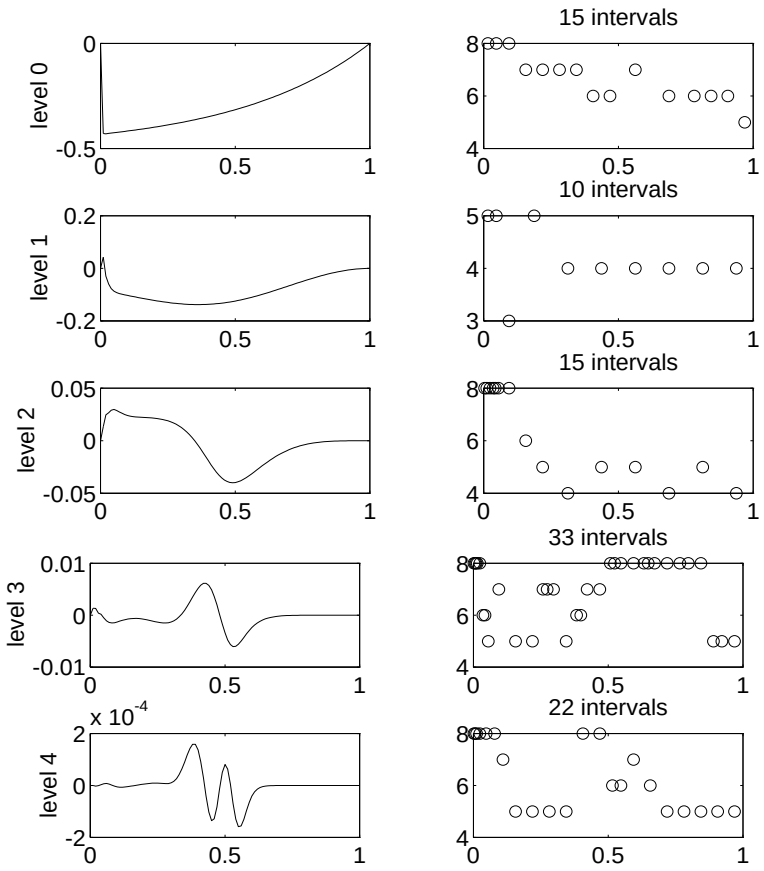


FIG. 8. Newton corrections for (10.2) with $\varepsilon = 10^{-3}$ and $\mathrm{tol} = 10^{-6}$

Mathematically speaking, we meet the classical situation of a periodic solution emanating from a branch of (trivial) fixed points at a Hopf bifurcation.

Figures 11 sketches the bogie model schematically. The Cooperrider model describes a bogie running with constant speed v on a perfect, stiff, level and straight track. We use the coordinate system moving along the track with the constant speed of the vehicle. The wheels, axles and the bogie are assumed stiff and friction is only included in the wheel-rail forces. Table 3 explains the parameters used in the model. The coupling of the frame and the wheelset is described by linear string and damper forces. The nonlinearity is due to the creep forces which are modelled as proposed by Vermeulen and Johnson [65].

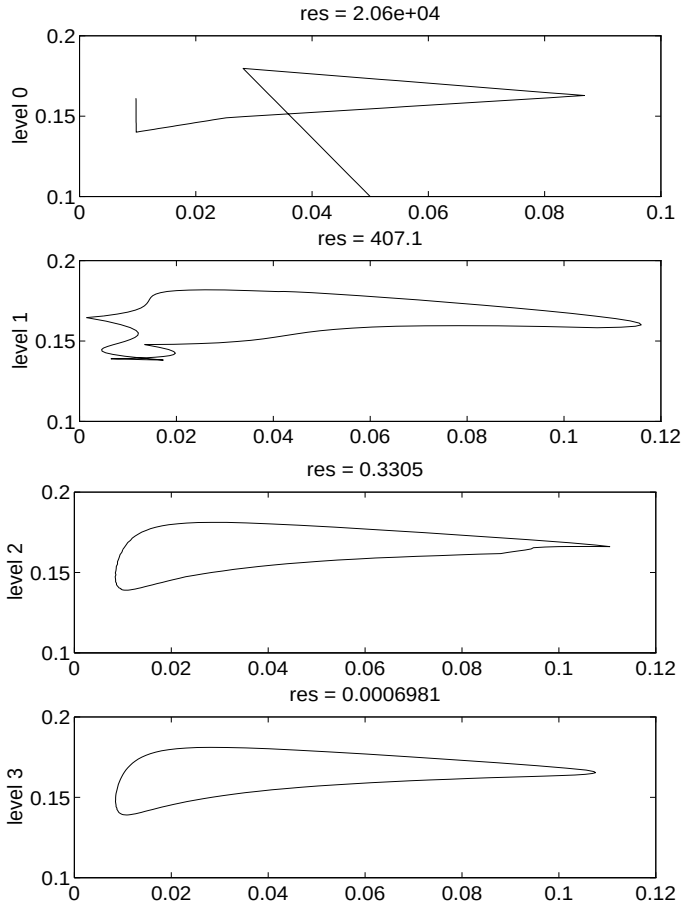


FIG. 9. Newton iterates for the chemical oscillator

If x, y are the normalized creep terms $x = \xi_x/\psi$ and $y = \xi_y/\phi$, the radial creep force is given by

$$G(\xi_R) = \frac{F_R}{\xi_R} = \mu N k_R \begin{cases} 1/u & \text{for } u \geq 3 \\ 1 - \frac{1}{3}u + \frac{1}{27}u^2 & \text{for } u < 3 \end{cases} \tag{11.1}$$

where

$$k_R = \frac{G\pi ab}{\mu N} \ , \quad u = k_R \xi_R$$

and $\xi_R = \sqrt{x^2 + y^2}$ is the radial creep term. This results in the axial creep

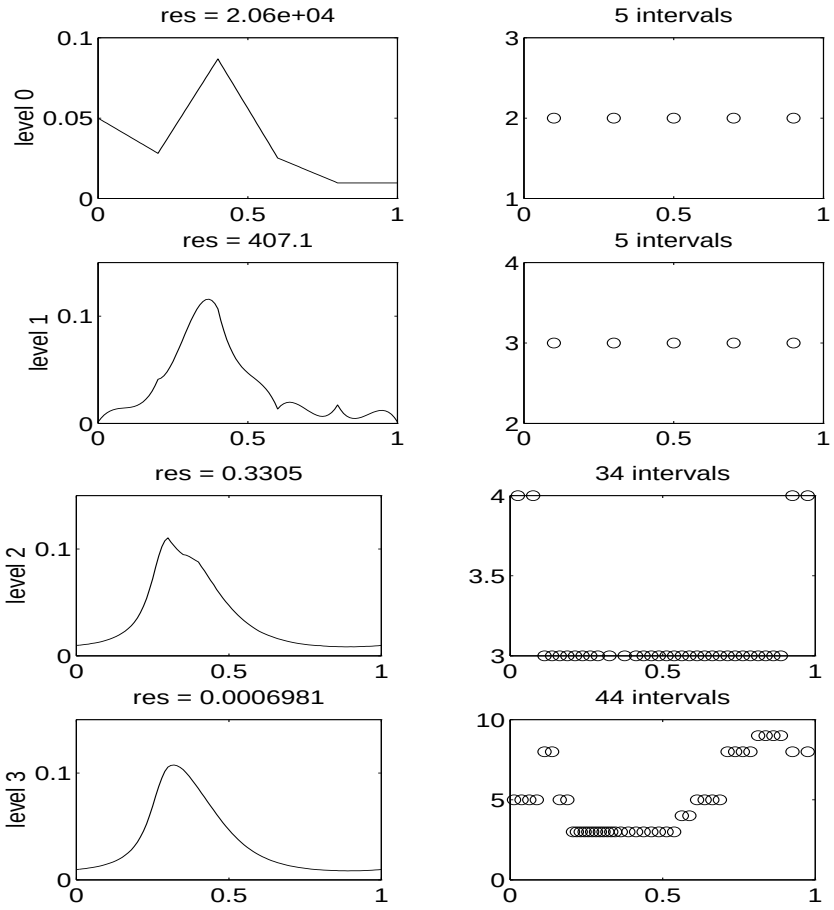


FIG. 10. Adaptively chosen grids for the chemical oscillator

forces

$$F_x = xG(\xi_R) \quad \text{and} \quad F_y = yG(\xi_R) \; .$$

Due to the linear term in the Vermeulen-Johnson model, the creep force term G is not differentiable with respect to (x, y) in $(x, y) = (0, 0)$. This fact necessitates some smoothing procedure near the origin. This might either be done by using the creep force

$$\tilde{G}(\xi) = \tanh u$$

instead of G as in [63] or by smoothing the norm in the radial creep term by some polynomial approximation near the origin. We follow the latter

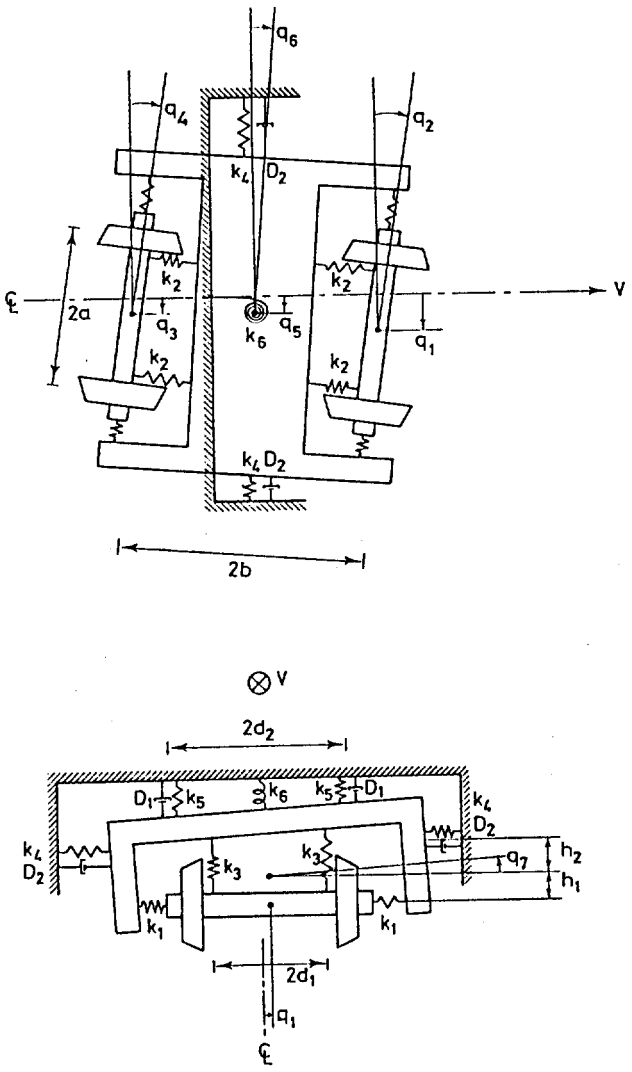


FIG. 11. Model of the railway bogie

approach which seems to be more flexible since the approximation is easily controlled by a small parameter ε. We substitute the squareroot $f(x) = \sqrt{x}$ by the continuously differentiable function $\tilde{f} \in C^1[0, \infty)$ defined by

$$\tilde{f}(x) := \begin{cases} g(x) & \text{for } x \leq \varepsilon \\ \sqrt{x} & \text{for } x \geq \varepsilon \end{cases} ,$$

a	$= 0.716\,m$	half of track gauge
b	$= 1.074\,m$	half of axle gauge
d_1	$= 0.620\,m$	distance spring to center of gravity
d_2	$= 0.680\,m$	
h_1	$= 0.0762\,m$	distance damper to center of gravity
h_2	$= 0.6584\,m$	
λ	$= 0.05$	wheel conicity
μ	$= 0.15$	friction coefficient
δ	$= 0.0091$	clearance between flange and railhead
r_0	$= 0.4572\,m$	centered wheel rolling radius
k_1	$= 1.823\,MN/m$	spring constant, see figure 11
k_2	$= 3.646\,MN/m$	
k_3	$= 3.646\,MN/m$	
k_4	$= 0.1823\,MN/m$	
k_5	$= 0.3333\,MN/m$	
k_6	$= 2.710\,MN/m$	
D_1	$= 20.0\,kN\,s/m$	damper constant
D_2	$= 29.2\,kN\,s/m$	
m_w	$= 1022\,kg$	mass of wheel axle
m_f	$= 2918\,kg$	mass of frame
I_{wy}	$= 678\,kg\,m^2$	moment of inertia for yaw motion of wheel axle
I_{fy}	$= 6780\,kg\,m^2$	moment of inertia for yaw motion of frame
I_{fr}	$= 6780\,kg\,m^2$	moment of inertia for roll motion of frame
ψ	$= 0.54219$	values obtained from Hertz contact theory
ϕ	$= 0.60252$	
$G\pi ab$	$= 6.563 \cdot 10^6\,N$	
μN	$= 10000\,N$	

TABLE 3. Table of parameters used in the bogie model

where $g \in \mathbb{P}_3$ is a cubic polynomial $g(x) = \sum_{i=0}^{3} a_i x^i$ satisfying the interpolation conditions

$$g(0) = g'(0) = 0\ , \quad g(\varepsilon) = f(\varepsilon) = \sqrt{\varepsilon} \ \text{ and } \ g'(\varepsilon) = f'(\varepsilon) = \frac{1}{2\sqrt{\varepsilon}}\ .$$

The resulting coefficients a_i are

$$a_0 = a_1 = 0 \ , \quad a_2 = \frac{5}{2}\varepsilon^{-\frac{3}{2}} \ , \quad a_3 = -\frac{3}{2}\varepsilon^{-\frac{5}{2}} \ .$$

Figure 12 displays the squareroot and its approximation for $\varepsilon = 0.5$ and

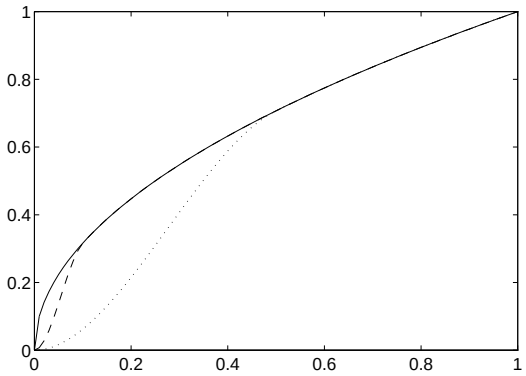


FIG. 12. Smoothing the squareroot near the origin ($\varepsilon = 0.5$ dotted, $\varepsilon = 0.1$ dashed)

$\varepsilon = 0.1$. For the numerical experiments we used $\varepsilon = 10^{-3}$.

The flange forces are modelled by stiff linear springs with a dead band leading to the (continuous but not differentiable) flange force

$$F(u) = \begin{cases} k_0(U-\delta) & \text{for } U > \delta \\ 0 & \text{for } -\delta \le U \le \delta \\ k_0(U+\delta) & \text{for } U < -\delta \end{cases} \ .$$

In contrast to [63] we do not need to smooth the flange force (e.g., by some exponential expression), since it poses no problem for the algorithm. We end up with seven second order ODEs for the seven degrees of freedom $q_1, \ldots, q_7$.

$$\begin{aligned} 0 &= m_w q_1'' + A_1 + 2F_{xf} + F(q_1) \\ 0 &= I_{wy} q_2'' + A_3 + 2aF_{yf} \\ 0 &= m_w q_3'' + A_2 + 2F_{xr} + F(q_3) \\ 0 &= I_{wy} q_4'' + A_4 + 2aF_{yr} \\ 0 &= m_f q_5'' - A_1 - A_2 + A_5 \\ 0 &= I_{fy} q_6'' - bA_1 + bA_2 - A_3 - A_4 + A_6 \end{aligned}$$

$$0 = I_{fr}q_7'' - h_1A_1 - h_2A_2 - h_2A_5 + A_7$$

where

$$\begin{aligned}
A_1 &= 2k_1(q_1 - q_5 - bq_6 - h_1q_7) \\
A_2 &= 2k_1(q_3 - q_5 + bq_6 - h_1q_7) \\
A_3 &= 2k_2d_1^2(q_2 - q_6) \\
A_4 &= 2k_2d_1^2(q_4 - q_6) \\
A_5 &= 2D_2(q_5' - h_2q_7') + 2k_4(q_5 - h_2q_7) \\
A_6 &= k_6q_6 \\
A_7 &= 2D_1d_2^2q_7' + 2k_5d_2^2q_7 + 4k_3d_1^2q_7
\end{aligned}$$

Here, F_{xf} and F_{yf} are the creep forces for the front axle corresponding to the creep terms

$$\xi_x = q_1'/v - q_2 \quad \text{and} \quad \xi_y = aq_2'/v + \lambda q_1/r_0$$

whereas F_{xr} and F_{yr} are the creep forces for the rear axle obtained from the creep terms

$$\xi_x = q_3'/v - q_4 \quad \text{and} \quad \xi_y = aq_4'/v + \lambda q_3/r_0 \ .$$

Considered as a first order system, we are lead to a dynamical system of 14 autonomous first order ODEs depending on the speed v as a scalar parameter. This formulation is the starting point for the fully automatic numerical simulation. Beginning with some fixed point (here: the trivial stationary solution at low speed) the algorithm follows the branch of equilibria, looks for Hopf bifurcations (via the standard eigenvalue criterium), computes the approximate periodic solution near the Hopf point and uses this to follow the branch of periodic solutions by the h-p collocation method.

For the results presented below we considered the velocity range $v \in [50, 190]$. We first require an accuracy of $tol = 5 \cdot 10^{-3}$ on the (relatively scaled) residual. In figure 14 we display the corresponding bifurcation diagram, where the amplitude of the first component q_1 is plotted versus the velocity v. The actually computed periodic solutions are marked by a circle. The algorithm detects and computes a Hopf bifurcation at $v = 68.6\, m/s$.

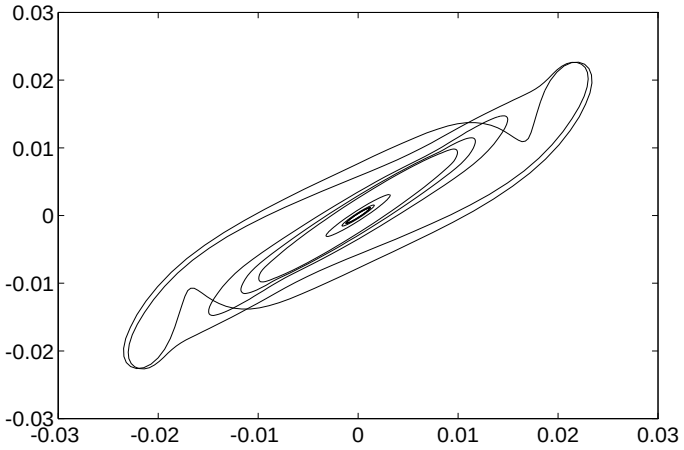


FIG. 13. Periodic solutions of the bogie model, q_3 versus q_1

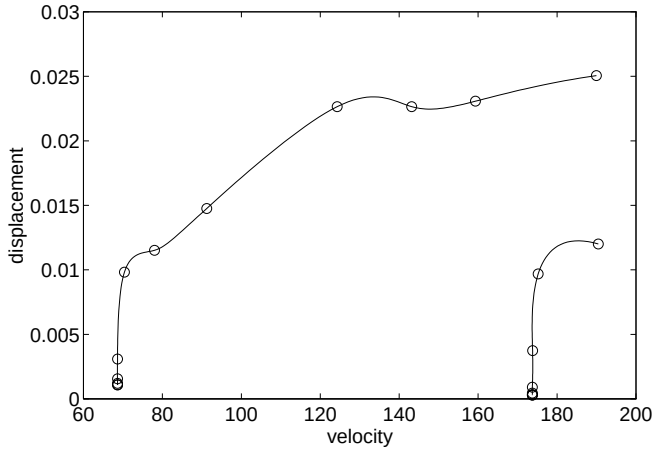


FIG. 14. Bifurcation diagram for the bogie model ($tol = 5 \cdot 10^{-3}$)

Using the periodic solution of the linearization near the Hopf point as initial guess, the code follows the branch of periodic solution to the right boundary of the given parameter interval. Next, the second Hopf bifurcation at $v = 173\,m/s$ is detected and computed, followed by the calculation of the new path of periodic solutions. The computation of the whole diagram takes less than two minutes.

In figure 13 we show the first nine periodic solutions on the first branch.

Since the trajectories become more and more complicated, we use in figure 15 a three dimensional presentation of the whole branch of periodic solutions. Here, we display the components q_1 and q_3 versus the velocity v. It definetely

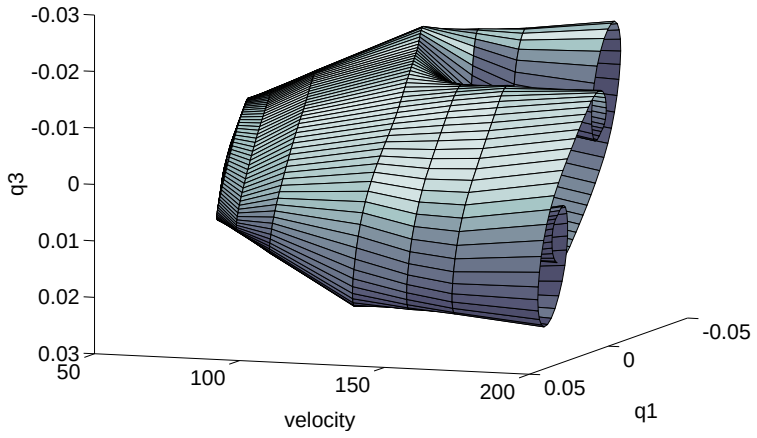


FIG. 15. Periodic solutions of the bogie model, q_3 and q_1 versus v ($tol = 5 \cdot 10^{-3}$)

documents the large continuation steps due to large convergence radius of the multilevel Newton method. There was only one steplength reduction. Next we required an accuracy of $tol = 5 \cdot 10^{-4}$. The associated bifurcation diagram is shown in figure 16. Due to the smaller tolerance we obtain a better resolution of the qualitative change of the solution at $v \approx 130\,m/s$ which is also reflected in the three dimensional representation in figure 17.

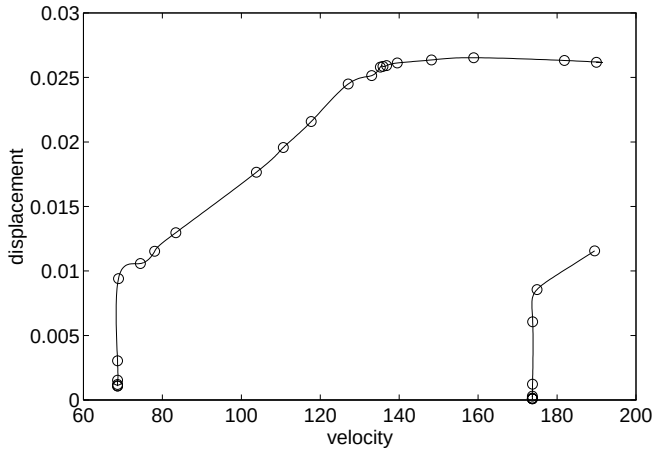


FIG. 16. Bifurcation diagram for the bogie model ($tol = 5 \cdot 10^{-4}$)

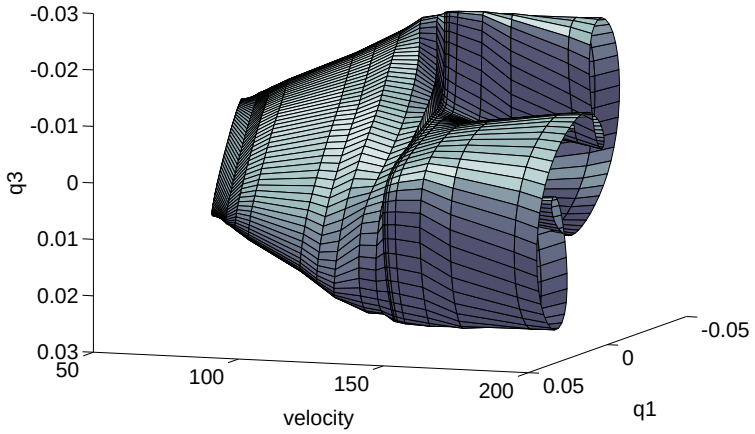


FIG. 17. Periodic solutions of the bogie model, q_3 and q_1 versus v ($tol = 5 \cdot 10^{-4}$)

Conclusions

We presented a new and complete framework for inexact Gauss Newton methods applied to underdetermined nonlinear problems. Starting with Deuflhard's fundamental insight that there is a close relationship between invariance properties and the algorithmical realizability, we were led to affine invariant and affine contravariant convergence theorems for the inexact Newton method that directly resulted in easily implementable accuracy matching strategies for the linear subproblems. In addition, these characterizations of the nonlinearity allow an embedding of inexact Gauss Newton methods into a general continuation framework including estimates for the optimal preditor stepsizes.

While the error oriented affine invariant approach is appropriate whenever the error of the linear solver is cheaply available, conversely the new affine contravariant approach appears to be the natural concept for underdetermined problems. One of its features is its weak differentiability assumption, so that it only requires a Gâteaux differentiable mapping satisfying a directional Lipschitz condition. Furthermore, the demands on the inner residuals of the linear subproblems translate very easily to almost any linear solver. This was demonstrated by the application within an existing multiple shooting code and the full derivation of a new adaptive h-p collocation code for BVPs of ODEs.

The latter code was developed as a prototype for the most attractive application of inexact Gauss Newton methods: The quasilinearization or multilevel Newton methods. We have seen that the crucial point lies in the correct formulation of the nonlinear problem. For space-like BVPs the nonlinear Fredholm equation is the most suitable formulation which directly incorporates the (necessarily non degenerate) boundary conditions. For time-like problems and more general boundary conditions the Volterra formulation represents an alternative. Once the nonlinear problem is well defined, we can concentrate on the linear subproblems. With this linearity we can save up local information so that the standard adaptive mesh selection techniques for PDEs transfer efficiently to collocation. Thus we were led to the new h-p collocation using local refinement and variable local orders, the new local boundary representation which respects the symmetry of the BVP, and the local elimination process. The resulting multilevel Newton h-p collocation

method, although only meant as an easy application of the multilevel Newton idea, is already competitive with the well established collocation codes based on regridding. More importantly, the algorithm has no difficulties in determining boundary layers anywhere on the given interval without any special treatment and is equally efficient for space-like and time-like problems.

Last but not least, the resulting class library COCON shows that modern software technology such as object oriented programming languages is able to cut the costs for the developement of advanced numerical algorithms drastically.

Of course this is not the end of the story but just the beginning. We think that the results are reason enough to think of more applications and enhencements. Thus, we would like to draw attention to some perspectives.

- So far, we have only used the multilevel Newton method for one-dimensional BVPs. The next step would be to tackle 2D and 3D problems and to combine efficient linear solvers (like, e.g., the CASCADE code for elliptic problems) with the inexact Gauss Newton method.

- Moreover, we used different discretizations for the linear subproblems but *direct* (sparse) solvers for the linear systems. If the number of degrees of freedom becomes very large, they have to be substituted by *iterative* methods. Fortunately, the embedding in the inexact Newton context poses no problems since the residual is easily available for most iterative solvers.

- Another extension of the method will be its combination with simplified and updating methods. This may again increase the efficiency since most effort is spent in the last Newton step. We have taken this problem into account by imposing a lower bound on the absolute accuracy, but a simplified approach for example might be even better.

- Unfortunately the symmetry issue was beyond the scope of this work. Obviously, most reduction techniques known from finite dimensional nonlinear systems (see e.g. Dellnitz and Werner [21], Healey [42], or Gatermann and H. [39]) can be transferred to the new situation. This applies to the symmetry of equilibria as well as to symmetric periodic solutions.

Symbols

$\mathbb{R}$	real numbers
$\mathbb{R}^{m\times n}$	real $m \times n$ matrices
$L(X, Y)$	space of continuous linear maps from a Banach space X in another Banach space Y
(A, b, c)	Runge Kutta scheme of collocation type, see page 55
$\gamma_j(t), \Gamma(t)$	collocation scheme, see page 65
$C(t)$	boundary scheme, see page 65
β	linear non degenerate boundary condition, see page 47
$M \in \mathbb{R}^{pn\times pn}$	local matrix of the implicit Runge Kutta discretization, see pages 59, 68
$B_\rho(x)$	open ball $\{y \mid \|x - y\| < \rho\}$ of radius ρ and center x
$\bar{B}_\rho(x)$	closure of $B_\rho(x)$
$C^n(U)$	n-times continuously differentiable maps $f : U \to \mathbb{R}$
$C^n[a, b]$	n-times continuously differentiable maps on $[a, b]$
$\mathrm{GL}(n)$	invertible $n \times n$ matrices $\{A \in \mathbb{R}^{n\times n} \mid \det(A) \neq 0\}$
A^+	Moore Penrose pseudo inverse of a matrix $A \in \mathbb{R}^{m\times n}$
$N(A)$, $\ker A$	kernel of a linear map, $N(A) = \{x \mid Ax = 0\}$
$R(A)$, $\mathrm{im}\, A$	range of a linear map
$\mathbb{P}_n$	polynomials of degree less or equal to n (with coefficients in some Banach space)
L_i	Lagrange polynomials
$[\varepsilon]$	estimate for ε
Δ, $\Delta + k$	h-p grids $\Delta = (\{t_i\}, \{p_i\})$ and $\Delta + k = (\{t_i\}, \{p_i + k\})$, see page 69
$\mathbb{P}_\Delta$, $\mathbb{P}_{\Delta+k}$	corresponding spaces of piecewise polynomials Δ and $\Delta + k$, see pages 69
D^{-m}	lifting operator w.r.t. the homogeneous boundary conditions $\beta = 0$, see page 65

$D^{-m,z}$	lifting operator w.r.t. the boundary conditions $\beta = z$, see page 65
π_k	interpolation operator at k nodes
$\pi_k^{(-m)}$	lifted interpolation operator at k nodes, see page 74

References

[1] E. L. Allgower and K. Georg. *Numerical Continuation Methods.* Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[2] U. Ascher, J. Christiansen, and R. D. Russell. A collocation solver for mixed order systems of boundary value problems. *Math. Comput.*, 33:659–679, 1979.

[3] U. Ascher, J. Christiansen, and R. D. Russell. Collocation software for boundary value ode's. *ACM Trans. Math. Software*, 7:209–222, 1981.

[4] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[5] I. Babuška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15:736–754, 1978.

[6] G. Bader. Solution of boundary value problems by collocation methods. Habilitationsschrift, Universität Heidelberg, 1988.

[7] G. Bader and P. Kunkel. Continuation and collocation for parameter-dependent boundary value problems. *SIAM J. Sci. Stat. Comput.*, 10(1):72–88, 1989.

[8] R. E. Bank and T. F. Chan. PLTMGC: A Multi-grid Continuation Program for Parameterized Nonlinear Elliptic Systems. *SIAM J. Sci. Stat. Comput.*, 7(2):540–559, 1986.

[9] R. E. Bank and H. D. Mittelmann. Continuation and multi-grid for nonlinear elliptic system. In *Multigrid methods II, Proc. 2nd Eur. Conf., Cologne / Ger. 1985*, pages 23–37. Lect. Notes Math. 1228, 1986.

[10] R. E. Bank and D. J. Rose. Global approximate newton methods. *Numer. Math.*, 37:279–295, 1981.

[11] R. E. Bank and D. J. Rose. Analysis of a multilevel iterative method for nonlinear finite element equations. *Math. Comp.*, 39:453–465, 1982.

[12] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. John Wiley, New York, 1974.

[13] H. G. Bock. Recent Advances in Parameter Identification Techniques for O.D.E. In Deuflhard and Hairer, editors, *Progress in Scientific Computing 2*. Birkhäuser, Boston, 1983.

[14] H. G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*. PhD thesis, Universität zu Bonn, 1985.

[15] T. F. Chan. Deflation techniques and block-elimination algorithms for solving bordered singular systems. *SIAM J. Sci. Stat. Comp.*, 5:121–134, 1984.

[16] N. K. Cooperrider. The Hunting Behavior of Conventional Railway Trucks. *J. Eng. for Industry*, pages 1–10, 1971.

[17] M. A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures*. John Wiley and Sons, Chichester, New York, 1991.

[18] C. de Boor and B. Swartz. Collocation At Gaussian Nodes. *SIAM J. Numer. Anal.*, 10(4):582–606, 1973.

[19] C. de Boor and R. Weiss. SOLVEBLOK: A package for solving almost block diagonal linear systems. *ACM TOMS*, 6:80–87, 1980.

[20] K. Deimling. *Nonlinear Functional Analysis*. Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[21] M. Dellnitz and B. Werner. Computational methods for bifurcation problems with symmetries — with special attention to steady state and hopf bifurcation points. In H. D. Mittelmann, D. Roose, editor, *Continuation Techniques and Bifurcation Problems*, pages 97–123. ISNM **92**, Birkhäuser, Basel, 1990.

[22] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[23] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward a universal h-p adaptive finite element strategy, Part 1. Constrained approximation and data structure. *Comput. Methods Appl. Mech. Engrg.*, 77:79–112, 1989.

[24] P. Deuflhard. A Stepsize Control for Continuation Methods and its Special Application to Multiple Shooting Techniques. *Numer. Math.*, 33:115–146, 1979.

[25] P. Deuflhard. Order and stepsize control in extrapolation methods. *Numer. Math.*, 41:399–422, 1983.

[26] P. Deuflhard. Computation of Periodic Solutions of Nonlinear ODEs. *BIT*, 24:456–466, 1984.

[27] P. Deuflhard. Uniqueness Theorems for Stiff ODE initial Value Problems. In *Proceedings 13th Biennial Conference on Numerical Analysis*, pages 74–88. University of Dundee, 1989.

[28] P. Deuflhard. Global Inexact Newton Methods for Very Large Scale Nonlinear Problems. *Impact of Computing in Science and Engineering*, 3(4):366–393, 1991.

[29] P. Deuflhard and G. Bader. Multiple shooting techniques revisited. In P. Deuflhard and B. Enquist, editors, *Numerical treatment of inverse problems in differential and Integral Equations*. Birkhäuser, Boston, 1983.

[30] P. Deuflhard, B. Fiedler, and P. Kunkel. Efficient Numerical Pathfollowing beyond Critical Points. *SIAM J. Numer. Anal.*, 18:949–987, 1987.

[31] P. Deuflhard, B. Fiedler, and P. Kunkel. Numerical Pathfollowing beyond Critical Points in ODE Models. In P. Deuflhard and B. Enquist, editors, *Proc. Large Scale Scientific Computing*. Birkhäuser, Boston, 1987.

[32] P. Deuflhard and G. Heindl. Affine invariant convergence theorems for newton's method and extensions to related methods. *SIAM J. Numer. Anal.*, 16:1–10, 1979.

[33] P. Deuflhard and A. Hohmann. *Numerische Mathematik I.* Walter de Gruyter, Berlin, New York, 2nd edition, 1993.

[34] P. Deuflhard and W. Sautter. On rank-deficient pseudoinverses. *Lin. Alg. Appl.*, 29:91–111, 1980.

[35] E. J. Doedel. AUTO: A program for the automatic bifurcation analysis of autonomous systems. *Congr. Numer.*, 30:265–284, 1981.

[36] E. J. Doedel. AUTO: Software for continuation and bifurcation problems in ordinary differential equations. Technical report, California Institute of Technology, Pasadena, 1986.

[37] I. S. Duff. Ma28 - a set of fortran subroutines for sparse unsymmetric linear equations. Report AERE-R.8730, Harwell, 1980.

[38] J. P. Fink and W. C. Rheinboldt. On the discretization error of parametrized nonlinear equations. *SIAM J. Numer. Anal.*, 20(4):732–746, 1983.

[39] K. Gatermann and A. Hohmann. Symbolic Exploitation of Symmetry in Numerical Pathfollowing. *Impact of Computing in Science and Engineering*, 3(4):330–365, 1991.

[40] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in one dimension, Part 1 to 3. *Numer. Math.*, 49:577–683, 1986.

[41] E. Hairer and A. Ostermann. Dense ouput for extrapolation methods. *Numer. Math*, 58:419–439, 1990.

[42] T. Healey. A group-theoretic approach to computational bifurcation problems with symmetry. *Comput. Methods Appl. Mech. Engrg.*, 67:257–295, 1988.

[43] A. Hohmann. An adaptive continuation method for implicitly defined surfaces. Preprint SC 91–20, Konrad-Zuse-Zentrum, Berlin, 1991.

[44] A. Hohmann. An Implementation of Extrapolation Codes in C++. Technical Report TR 93–8, Konrad-Zuse-Zentrum, Berlin, 1993.

[45] A. Hohmann and C. Wulff. Modular design of extrapolation codes. Technical Report TR 92-5, Konrad-Zuse-Zentrum, Berlin, 1992.

[46] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. In P. H. Rabinowitz, editor, *Application of Bifurcation Theory*, pages 359–384. Academic Press, New York, London, 1977.

[47] C. Klein-Robbenhaar. Numerische Pfadverfolgung für große dünnbesetzte Systeme. Diplomarbeit, Freie Universität Berlin, 1993.

[48] M. R. Maier. *Numerische Lösung singulär gestörter Randwertprobleme mit Anwendung auf Halbleitermodelle*. PhD thesis, TU München, Institut für Mathematik, TUM-M8216, 1982.

[49] Dirk Moelle. *Digitale Grenzzykelrechnung zur Untersuchung der Stabilität von Eisenbahndrehgestellen unter dem Einfluß von Nichtlinearitäten*. PhD thesis, Institut für Luft- und Raumfahrt, Technische Universität Berlin, 1990.

[50] J. T. Oden, L. Demkowizc, W. Rachowicz, and T. A. Westermann. Toward a universal h-p adaptive finite element strategy, Part 2. A posteriori error estimation. *Comput. Methods Appl. Mech. Engrg.*, 77:113–180, 1989.

[51] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.

[52] T. S. Parker and L. O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, New York, Berlin, Heidelberg, 1989.

[53] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, New York, et al., 1988.

[54] I. Prigogine and R. Lefever. Symmetry breaking instabilities in dissipative systems II. *J. Chem. Phys.*, 48:1695–1701, 1968.

[55] W. C. Rheinboldt. *Numerical analysis of parametrized nonlinear equations*. John Wiley and Sons, New York, 1986.

[56] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[57] M. R. Scott and H. A. Watts. Superposition, orthonormalization, quasi-linearization and two-point boundary-value problems. In G. Goos and J. Hartmanis, editors, *Codes for Boundary-Value Problems in Ordinary Differential Equations*, pages 109–121. Springer-Verlag, Berlin, Heidelberg, 1979.

[58] F. F. Seelig. Unrestricted Harmonic Balance I. Theory and Computer Program for Time-Dependent Systems. *Y. Naturforsch.*, 359:1054–1061, 1980.

[59] F. F. Seelig. Unrestricted Harmonic Balance II. Application to Stiff ODE's in Exzyme Catalysis. *J. of Math. Biology*, 12:187–198, 1981.

[60] R. Seydel. *From equilibrium to chaos. Practical bifurcation and stability analysis*. Elsevier, New York, 1988.

[61] R. Telgmann and M. Wulkow. MEDICI Benutzerhandbuch. CiT, Wiefelstede, 1993.

[62] H. True. Railway Vehicle Chaos and Asymmetric Hunting. *Proc. 12th IAVSD Symposium, Vehicle System Dynamics*, 19:625–637, 1990.

[63] H. True and C. Kaas-Petersen. A Bifurcation Analysis of Nonlinear Oscillations in Railway Vehicles. *Proc. 8th IAVSD Symposium, Vehicle System Dynamics*, 13:655–665, 1984.

[64] H. A. van der Vorst. BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13(2):613–644, 1992.

[65] P. J. Vermeulen and K. L. Johnson. Contact of Nonspherical Elastic Bodies Transmitting Tangential Forces. *J. Appl. Mech.*, 31:338–340, 1964.

[66] C. Wulff. Numerische Pfadverfolgung von periodischen Lösungen mit Symmetrie. Diplomarbeit, Freie Universität Berlin, 1993.

[67] M. Wulkow. Adaptive Treatment of Polyreactions in Weighted Sequence Spaces. *Impact of Computing in Science and Engineering*, 4(2):153–193, 1992.

[68] T. J. Ypma. Local convergence of inexact newton methods. *SIAM J. Numer. Anal.*, 21:583–590, 1984.

[69] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method, Volume 2, Solid and Fluid Mechanics, Dynamics and Non-linearity.* McGraw-Hill Book Company, London, New York, 1991.