1 Introduction

This chapter is an elementary introduction into the general theme of this book. We start from the historical root, Newton's method for scalar equations (Section 1.1): the method can be derived either *algebraically*, which leads to *local* Newton methods only (see Chapter 2), or *geometrically*, which leads to *global* Newton methods via the topological Newton path (see Chapter 3).

Section 1.2 contains the *key to the basic understanding of this monograph*. First, four affine invariance classes are worked out, which represent the four basic strands of this treatise:

- *affine covariance*, which leads to *error* norm controlled algorithms,
- *affine contravariance*, which leads to *residual* norm controlled algorithms,
- *affine conjugacy*, which leads to *energy* norm controlled algorithms, and
- *affine similarity*, which may lead to *time* step controlled algorithms.

Second, the affine invariant local estimation of affine invariant Lipschitz constants is set as the central *paradigm* for the construction of adaptive Newton algorithms.

In Section 1.3, we fix terms for various Newton-type methods to be named throughout the book: ordinary and simplified Newton method, Newton-like methods, inexact Newton methods, quasi-Newton methods, quasilinearization, and inexact Newton multilevel methods.

In Section 1.4, details are given for the iterative linear solvers `GMRES`, `PCG`, `CGNE`, and `GBIT` to an extent necessary to match them with finite dimensional inexact Newton algorithms. In view of function space oriented inexact Newton algorithms, we also revisit multiplicative, additive, and cascadic multigrid methods emphasizing the role of adaptive error control therein.

1.1 Newton-Raphson Method for Scalar Equations

Assume we have to solve the scalar equation

$$f(x) = 0$$

with an appropriate guess x^0 of the unknown solution x^* at hand.

P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, Springer Series in Computational Mathematics 35,
DOI 10.1007/978-3-642-23899-4_1, © Springer-Verlag Berlin Heidelberg 2011

Algebraic approach. We use the *perturbation*

$$\Delta x = x^* - x^0$$

for Taylor's expansion

$$0 = f(x^0 + \Delta x) = f(x^0) + f'(x^0)\Delta x + O(|\Delta x|^2).$$

Upon dropping terms of order higher than linear in the perturbation, we arrive at the approximate equation

$$f'(x^0)\Delta x \approx -f(x^0),$$

which, assuming $f'(x^0) \neq 0$, leads to the precise equation

$$x^1 - x^0 = \Delta x^0 = -\frac{f(x^0)}{f'(x^0)}$$

for a first correction of the starting guess. From this, an *iterative* scheme is constructed by repetition

$$x^{k+1} = \Phi(x^k) = x^k - \frac{f(x^k)}{f'(x^k)}\,,\ k = 0, 1, \dots\ .$$

If we study the contraction mapping Φ in terms of a *contraction factor* Θ, we arrive at

$$\Theta = \max_{x \in I} \Phi'(x) = \max_{x \in I} \frac{f(x) f''(x)}{(f'(x))^2}$$

with I an appropriate interval containing x^*. From this, we have at least *linear* convergence

$$|x^{k+1} - x^*| \leq \Theta |x^k - x^*|$$

in a neighborhood of x^*, where $\Theta < 1$. In passing we note that this contraction factor Θ remains unchanged, if we rescale the equation according to

$$\alpha f(\beta y) = 0\,,\ \alpha\beta \neq 0\,,\ x = \beta y\,.$$

An extension of this kind of observation to rather general nonlinear problems will lead to fruitful theoretical and algorithmical consequences below. For starting guesses x^0 'sufficiently close' to x^* even *quadratic* convergence of the iterates can be shown in the sense that

$$|x^{k+1} - x^*| \leq C |x^k - x^*|^2\,,\ k = 0, 1, 2 \dots\ .$$

The algebraic derivation in terms of the linear perturbation treatment carries over to rather general nonlinear problems up to operator equations such as boundary value problems for ordinary or partial differential equations.

Geometric approach. Looking at the *graph* of $f(x)$—as depicted in Figure 1.1—any root can be interpreted as the intersection of this graph with the real axis. Since this intersection cannot be constructed other than by tedious sampling of f, the graph of $f(x)$ is replaced by its *tangent* $p(x)$ in x_0 and the first iterate x_1 is defined as the intersection of the tangent with the real axis. Upon repeating this geometric process, the close-by solution point x^* can be constructed up to any desired accuracy. By geometric insight, the iterative process will converge *globally* for *convex* (or concave) f—which includes the case of arbitrarily 'bad' initial guesses as well! At first glance, this geometric derivation seems to be restricted to the scalar case, since the graph of $f(x)$ is a typically one-dimensional concept. A careful examination of the subject in more than one dimension, however, naturally leads to a topological path called *Newton path*—see Section 3.1.4 below.

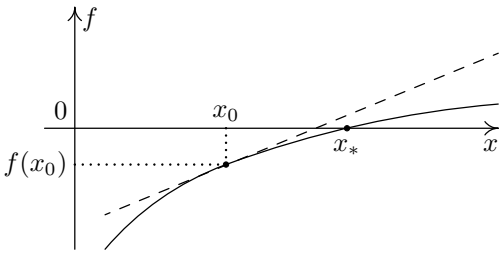

Fig. 1.1. Geometric interpretation: Newton's method for a scalar equation.

Historical Note. Strictly speaking, Newton's method could as well be named as Newton-Raphson-Simpson method—as elaborated in recent articles by N. Kollerstrom [134] or T.J. Ypma [203]. According to these careful historical studies, the following facts seem to be agreed upon among the experts:

- In the year 1600, Francois Vieta (1540–1603) had (first?) designed a *perturbation* technique for the solution of the scalar polynomial equations, which supplied one decimal place of the unknown solution per step via the explicit calculation of successive polynomials of the successive perturbations. It seems that this method had also been detected independently by al-Kāshī and simplified around 1647 by Oughtred.
- Isaac Newton (1643–1727) got to know Vieta's method in 1664. Up to 1669 he had improved it by *linearizing* these successive polynomials. As an example, he discussed the numerical solution of the cubic polynomial

$$f(x) := x^3 - 2x - 5 = 0\,.$$

Newton first noted that the integer part of the root is 2 setting $x_0 = 2$. Next, by means of $x = 2 + p$, he obtained the polynomial equation

$$p^3 + 6p^2 + 10p - 1 = 0\,.$$

Herein he neglected terms higher than first order and thus put $p \approx 0.1$. He inserted $p = 0.1 + q$ and constructed the polynomial equation

$$q^3 + 6.3q^2 + 11.23q + 0.061 = 0\,.$$

Again he neglected terms higher than linear and found $q \approx -0.0054$. Continuation of the process one more step led him to $r \approx 0.00004853$ and therefore to the third iterate

$$x_3 = x_0 + p + q + r = 2.09455147\,.$$

Note that the relations $10p - 1 = 0$ and $11.23q + 0.061 = 0$ given above correspond precisely to

$$p = x_1 - x_0 = -f(x_0)/f'(x_0)$$

and to

$$q = x_2 - x_1 = -f(x_1)/f'(x_1)\,.$$

As the example shows, he had also observed that by keeping all decimal places of the corrections, the number of accurate places would *double* per each step—i.e., *quadratic convergence*. In 1687 (Philosophiae Naturalis Principia Mathematica), the first nonpolynomial equation showed up: it is the well-known equation from astronomy

$$x - e\sin(x) = M$$

between the *mean anomaly* M and the *eccentric anomaly* x. Here Newton used his already developed polynomial techniques via the series expansion of *sin* and *cos*. However, no hint on the derivative concept is incorporated!

- In 1690, Joseph Raphson (1648–1715) managed to avoid the tedious computation of the successive polynomials, playing the computational scheme back to the original polynomial; in this now fully *iterative* scheme, he also kept all decimal places of the corrections. He had the feeling that his method differed from Newton's method at least by its derivation.
- In 1740, Thomas Simpson (1710–1761) actually introduced derivatives ('fluxiones') in his book 'Essays on Several Curious and Useful Subjects in Speculative and Mix'd Mathematicks, Illustrated by a Variety of Examples'. He wrote down the true *iteration* for one (nonpolynomial) equation and for a system of two equations in two unknowns thus making the correct extension to *systems* for the first time. His notation is already quite close to our present one (which seems to go back to J. Fourier).

Throughout this book, we will use the name 'Newton-Raphson method' only for scalar equations. For general equations we will use the name 'Newton method'—even though the name 'Newton-Simpson method' would be more appropriate in view of the just described historical background.

1.2 Newton's Method for General Nonlinear Problems

In contrast to the preceding section, we now approach the general case. Assume we have to solve a nonlinear operator equation

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

wherein $F : D \subset X \to Y$ for Banach spaces X, Y endowed with norms $\|\cdot\|_X$ and $\|\cdot\|_Y$. Let F be at least once continuously differentiable. Suppose we have a starting guess x^0 of the unknown solutions x^* at hand. Then *successive linearization* leads to the general Newton method

$$F'(x^k)\Delta x^k = -F(x^k)\,,\ x^{k+1} = x^k + \Delta x^k\,, \qquad k = 0, 1, \ldots\,. \tag{1.1}$$

Obviously, this method attacks the solution of a nonlinear problem by solving a *sequence of linear problems of the same kind.*

1.2.1 Classical convergence theorems revisited

A necessary assumption for the solvability of the above linear problems is that the derivatives $F'(x)$ are *invertible* for all occurring arguments. For this reason, standard convergence theorems typically require a-priori that the inverse $F'(x)^{-1}$ exists and is bounded

$$\|F'(x)^{-1}\|_{Y\to X} \le \beta < \infty\,,\ x \in D\,, \tag{1.2}$$

where$\|\cdot\|_{Y\to X}$ denotes an *operator norm.* From a computational point of view, such a theoretical quantity β defined over the domain D seems to be hard to get, apart from rather simple examples. Sampling of *local* estimates like

$$\|F'(x^0)^{-1}\|_{Y\to X} \le \beta_0 \tag{1.3}$$

seems to be preferable, but is still quite expensive. Moreover, a well-known rule in Numerical Analysis states that the actual computation of inverses should be avoided. Rather, such a condition should be monitored implicitly in the course of solving linear systems with specific right hand sides.

In order to study the convergence properties of the above Newton iteration, some *second derivative* information is needed, as already stated in the scalar equation case (Section 1.1 above). The classical standard form to include this information is via a *Lipschitz condition* of the type

$$\|F'(x) - F'(\bar{x})\|_{X\to Y} \le \gamma \|x - \bar{x}\|_X \,, \;\; x, \bar{x} \in D \,. \tag{1.4}$$

With this additional assumption, the *operator perturbation lemma* (sometimes also called Banach perturbation lemma) proves the existence of some upper bound β such that

$$\|F'(x)^{-1}\|_{Y\to X} \le \beta \le \frac{\beta_0}{1 - \beta_0 \gamma \|x - x^0\|_X}$$

for

$$\|x - x^0\|_X < \frac{1}{\beta_0 \gamma} \,, \;\; x \in D \,.$$

The proof is left as Exercise 1.1. Classical convergence theorems for Newton's method use certain combinations of these assumptions.

Newton-Kantorovich theorem. This first classical convergence theorem for Newton's method in abstract spaces (see [127, 163]) requires assumptions (1.3) and (1.4) to show *existence* and *uniqueness* of a solution x^* as well as quadratic convergence of the Newton iterates within a neighborhood characterized by a so-called *Kantorovich quantity*

$$h_0 := \|\Delta x^0\|_X \; \beta_0 \gamma < \tfrac{1}{2}$$

and a corresponding convergence ball around x^0 with radius $\rho_0 \sim 1/\beta_0\gamma$. This theorem is also the standard tool to prove the classical *implicit function theorem*—compare Exercise 1.2.

Newton-Mysovskikh theorem. This second classical convergence theorem (see [155, 163]) requires assumptions (1.2) and (1.4) to show *uniqueness* (not existence!) and quadratic convergence within a neighborhood characterized by the slightly different quantity

$$h_0 := \|\Delta x^0\|_X \; \beta\gamma < 2$$

and a corresponding convergence ball around x^0 with radius $\rho \sim 1/\beta\gamma$.

Both theorems seem to require the actual computation of the Lipschitz constant γ. However, such a quantity is certainly hard if not hopeless to compute in realistic nonlinear problems. Moreover, even computational local estimates of β and γ are typically far off any use in practical applications. That is why, for quite a time, people believed that convergence results are of theoretical interest only, but not of any value for the actual implementation of Newton algorithms. An illustrating simple example is given as Exercise 2.3.

This undesirable gap between convergence analysis and algorithm construction has been the motivation for the present book. As will become apparent, the key to closing this gap is supplied by *affine invariance in both convergence theory and algorithmic realization.*

1.2.2 Affine invariance and Lipschitz conditions

In order to make the essential point clear enough, it is sufficient to regard simply systems of nonlinear equations, which means that $X = Y = \mathbb{R}^n$ for fixed dimension $n > 1$ and the same norm in X and Y. Recall Newton's method in the form

$$F'(x^k)\Delta x^k = -F(x^k),\ x^{k+1} = x^k + \Delta x^k \quad k = 0, 1, \ldots .$$

Scaling. In sufficiently complex problems, scaling or re-gauging of variables (say, from *km* to *miles*) needs to be carefully considered. Formally speaking, with preselected nonsingular *diagonal* scaling matrices D_L, D_R for left and right scaling, we may write

$$(D_L F'(x^k) D_R)(D_R^{-1}\Delta x^k) = -D_L F(x^k)$$

for the scaled linear system. Despite its formal equivalence with (1.1), all standard norms used in Newton algorithms must now be replaced by *scaled norms* such that (dropping the iteration index k)

$$\|\Delta x\|\,, \|F\|\,, \|F + F'(x)\Delta x\| \ \longrightarrow\ \|D_R^{-1}\Delta x\|\,, \|D_L F\|\,, \|D_L(F + F'(x)\Delta x)\|\,.$$

With the change of norms comes a change of the criteria for the acceptance or rejection of new iterates. The effect of scaling on the iterative performance of Newton-type methods is a sheet lightning of the more general effects caused by affine invariance, which are the topic of this book.

Affine transformation. Let $A, B \in \mathbb{R}^{n\times n}$ be *arbitrary nonsingular* matrices and study the affine transformations of the nonlinear system as

$$G(y) = AF(By) = 0\,,\ x = By\,.$$

Then Newton's method applied to $G(y)$ reads

$$G'(y^k)\Delta y^k = -G(y^k),\ y^{k+1} = y^k + \Delta y^k \quad k = 0, 1, \ldots .$$

With the relation

$$G'(y^k) = AF'(x^k)B$$

and starting guess $y^0 = B^{-1}x^0$ we immediately obtain

$$x^k = By^k\,,\ k = 0, 1, \ldots .$$

Obviously, the iterates are invariant under transformation of the image space (by A)—an invariance property described by *affine covariance*. Moreover, they are transformed just as the whole original space (by B)—a property denoted by *affine contravariance*.

It is only natural to require that the above affine invariance properties are inherited by any theoretical characterization. As it turns out, the inheritance of the full invariance property is impossible. That is why we restrict our study to four special invariance classes.

Affine covariance. In this setting, we keep the domain space of F fixed ($B = I$) and look at the *whole class of problems*

$$G(x) = AF(x) = 0$$

that is generated by the class $\mathrm{GL}(n)$ of nonsingular matrices A. The Newton iterates are the same all over the whole class of nonlinear problems. For this reason, an affine covariant theory about their convergence must be possible. Upon revisiting the above theoretical assumptions (1.2), (1.3), and (1.4) we now obtain

$$\|G'(x)^{-1}\| \le \beta(A)\,,\ \|G'(x^0)^{-1}\| \le \beta_0(A)\,,\ \|G'(x) - G'(\bar{x})\| \le \gamma(A)\|x - \bar{x}\|\,.$$

Application of the classical convergence theorems then yields convergence balls with radius, say

$$\rho(A) \sim 1/\beta(A)\gamma(A)\,.$$

Compared with $\beta(I)$, $\gamma(I)$ we obtain (assuming best possible theoretical bounds)

$$\beta(A) \le \beta(I)\|A^{-1}\|\,,\ \gamma(A) \le \gamma(I)\|A\|$$

and therefore

$$\beta(A)\gamma(A) \le \beta(I)\gamma(I)\,\mathrm{cond}(A)\,. \tag{1.5}$$

For $n > 1$ we have $\mathrm{cond}(A) \ge 1$, even unbounded for $A \in \mathrm{GL}(n)$. Obviously, by a mean choice of A we can make the classical convergence balls shrink to nearly zero!

Fortunately, careful examination of the proof of the Newton-Kantorovich theorem shows that assumptions (1.3) and (1.4) can be telescoped to the requirement

$$\|F'(x^0)^{-1}\big(F'(x) - F'(\bar{x})\big)\| \le \omega_0\|x - \bar{x}\|\,, \qquad x, \bar{x}, x^0 \in D\,. \tag{1.6}$$

The thus defined Lipschitz constant ω_0 is affine covariant, since

$$\begin{aligned} G'(x^0)^{-1}\big(G'(x) - G'(\bar{x})\big) &= \big(AF'(x^0)\big)^{-1}A\big(F'(x) - F'(\bar{x})\big) \\ &= F'(x^0)^{-1}\big(F'(x) - F'(\bar{x})\big) \end{aligned}$$

so that both sides of (1.6) are independent of A. This definition of ω_0 (assumed best possible) still has the disadvantage of containing an operator norm on the left side—which, however, is unavoidable, because the operator perturbation lemma is required in the proof. Examination of the Newton-Mysovskikh theorem shows that assumptions (1.2) and (1.4) can also be telescoped to an affine covariant Lipschitz condition, which this time only contains vector norms (and directional derivatives):

$$\big\|F'(x)^{-1}\big(F'(\bar{x}) - F'(x)\big)(\bar{x} - x)\big\| \le \omega\|\bar{x} - x\|^2\,, \quad x, \bar{x} \in D\,. \tag{1.7}$$

This assumption allows a clean affine covariant theory about the local quadratic convergence of the Newton iterates including local uniqueness of the solution x^*—see Section 2.1 below. Moreover, this type of theorem will be the stem from which a variety of computationally useful convergence theorems branch off.

Summarizing, any affine covariant convergence theorems will lead to results in terms of *iterates* $\{x^k\}$, *correction norms* $\|\Delta x^k\|$ or *error norms* $\|x^k - x^*\|$.

Bibliographical Note. For quite a while, *affine covariance* held only in very few convergence theorems for local Newton methods, among which are Theorem 6. (1.XVIII) in the book of Kantorovich/Akhilov [127] from 1959, part of the theoretical results by J.E. Dennis [52, 53], or an interesting early paper by H.B. Keller [129] from 1970 (under the weak assumption of just Hölder continuity of $F'(x)$). None of these authors, however, seems to have been fully aware of the importance of this invariance property, since all of them neglected this aspect in their later work.

A systematic approach toward affine covariance, then simply called affine invariance, has been started in 1972 by the author in his dissertation [59], published two years later in [60]. His initial motivation had been to overcome severe difficulties in the actual application of Newton's method within multiple shooting—compare Section 7.1 below. In 1979, this approach has been transferred to convergence theory in a paper by P. Deuflhard and G. Heindl [76]. Following the latter paper, T. Yamamoto has preserved affine covariance in his subtle convergence estimates for Newton's method—see, e.g., his starting paper [202] and work thereafter. Around that time H.G. Bock [29, 31, 32] also joined the affine invariance crew and slightly improved the theoretical characterization from [76]. The first affine covariant convergence proof for inexact Newton methods is due to T.J. Ypma [203].

Affine contravariance. This setting is dual to the preceding one: we keep the image space of F fixed ($A = I$) and consider the *whole class of problems*

$$G(y) = F(By)\,,\ x = By\,,\ B \in \mathrm{GL}(n)$$

that is generated by the class $\mathrm{GL}(n)$ of nonsingular matrices B. Consequently, a common convergence theory for the whole problem class will not lead to statements about the Newton iterates $\{y^k\}$, but only about the *residuals* $\{F(x^k)\}$, which are independent of any choice of B. Once more, the classical conditions (1.2) and (1.4) can be telescoped, this time in image space terms only:

$$\big\|\big(F'(\bar{x}) - F'(x)\big)(\bar{x} - x)\big\| \le \omega \|F'(x)(\bar{x} - x)\|^2\,. \tag{1.8}$$

Observe that both sides are independent of B, since, for example

$$G'(y)(\bar{y} - y) = F'(x)B(\bar{y} - y) = F'(x)(\bar{x} - x)\,.$$

A Newton-Mysovskikh type theorem on the basis of such a Lipschitz condition will lead to convergence results in terms of *residual norms* $\|F(x^k)\|$.

Bibliographical Note. The door to *affine contravariance* in the Lipschitz condition has been opened by A. Hohmann in his dissertation [120] , wherein he exploited it for the construction of a residual based inexact Newton method within an adaptive collocation method for ODE boundary value problems—compare Section 7.4 below.

At first glance, the above dual affine invariance classes seem to be the only ones that might be observed in actual computation. At second glance, however, certain couplings between the linear transformations A and B may arise, which are discussed next.

Affine conjugacy. Assume that we have to solve the *minimization problem*

$$f(x) = \min\,,\ f : D \subset \mathbb{R}^n \to \mathbb{R}$$

for a functional f, which is *convex* in a neighborhood D of the minimum point x^*. Then this problem is equivalent to solving the nonlinear equations

$$F(x) = \operatorname{grad} f(x) = f'(x)^T = 0\,,\ x \in D\,.$$

For such a gradient mapping F the Jacobian $F'(x) = f''(x)$ is *symmetric* and certainly *positive semi-definite*. Moreover, assume that $F'(x)$ is *strictly positive definite* so that $F'(x)^{1/2}$ can be defined. This also implies that f is *strictly convex*. Upon transforming the minimization problem to

$$g(y) = f(By) = \min\,,\ x = By\,,$$

we arrive at the transformed equations

$$G(y) = B^T F(By) = 0$$

and the transformed Jacobian

$$G'(y) = B^T F'(x) B\,,\ x = By\,.$$

The Jacobian transformation is *conjugate*, which motivates the name of this special affine invariance. Due to Sylvester's theorem (compare [151]), it conserves the index of inertia, so that all G' are symmetric and strictly positive definite. Affine conjugate theoretical terms are, of course, functional values $f(x)$ and, in addition, so–called *local energy products*

$$(u, v) = u^T F'(x) v\,,\ u, v, x \in D\,.$$

Just note that energy products are invariant under this kind of affine transformation, since

$$u, v, x \to \bar{u} = Bu, \bar{v} = Bv, x = By$$

implies

$$u^T G'(y) v = \bar{u}^T F'(x) \bar{v} \,.$$

Local energy products induce *local energy norms*

$$\|F'(x)^{1/2} u\|^2 = (u, u) = u^T F'(x) u \,, \;\; u, x \in D \,.$$

In this framework, telescoping the theoretical assumptions (1.2) and (1.4) leads to an affine conjugate Lipschitz condition

$$\|F'(x)^{-1/2} \big(F'(\bar{x}) - F'(x)\big)(\bar{x} - x)\| \le \omega \|F'(x)^{1/2} (\bar{x} - x)\|^2 \,. \tag{1.9}$$

Affine conjugate convergence theorems will lead to results in terms of *functional values* $f(x)$ and *energy norms of corrections* $\|F'(z)^{1/2} \Delta x^k\|$ or *errors* $\|F'(z)^{1/2}(x^k - x^*)\|$.

Bibliographical Note. The concept of *affine conjugacy* dates back to P. Deuflhard and M. Weiser, who, in 1997, defined and exploited it for the construction of an adaptive Newton multilevel FEM for nonlinear elliptic PDEs—see [84, 85] and Section 8.3.

Affine similarity. This invariance principle is more or less common in the differential equation community—apart perhaps from the name given here. Consider the case that the solution of the nonlinear system $F(x) = 0$ can be interpreted as *steady state* or *equilibrium point* of the *dynamical system*

$$\dot{x} = F(x) \,. \tag{1.10}$$

Arbitrary affine transformation

$$A\dot{x} = AF(x) = 0$$

here affects both the domain and the image space of F in the same way—of course, differentiability with respect to time differs. The corresponding problem class to be studied is then

$$G(y) = AF(A^{-1} y) = 0 \,, \;\; y = Ax \,,$$

which gives rise to the Jacobian transformation

$$G'(y) = AF'(x) A^{-1} \,.$$

This *similarity* transformation (which motivates the name affine similarity) is known to leave the *Jacobian eigenvalues* λ invariant. Note that a theoretical characterization of *stability* of the equilibrium point involves their real parts $\Re(\lambda)$. In fact, an upper bound of these real parts, called the *one-sided* Lipschitz constant, will serve as a substitute of the Lipschitz constant of F, which

is known to restrict the analysis to nonstiff differential equations. As an affine similar representative, we may formally pick the (possibly complex) *Jordan canonical form* J, known to consist of elementary Jordan blocks for each separate eigenvalue. Let the Jacobian at any selected point $\hat{x}$ be decomposed such that

$$F'(\hat{x}) = T(\hat{x})J(\hat{x})T(\hat{x})^{-1} = TJT^{-1}\,,$$

which implies

$$G'(\hat{y}) = AF'(x)A^{-1} = (AT)J(AT)^{-1}\,.$$

Consequently, any theoretical results phrased in terms of the *canonical norm*

$$|\cdot| := \|T^{-1}\cdot\|$$

will meet the requirement of affine similarity. We must, however, remain aware of the fact that numerical Jordan decomposition may be *ill-conditioned*, whenever eigenvalue clusters arise—a property, which is reflected in the size of $\mathrm{cond}(T)$. With this precaution, an affine similar approach will be helpful in the analysis of *stiff* initial value problems for ODE's (see Chapter 6).

In contrast to the other invariance classes, note that here not only Newton's iteration exhibits the correct affine similar pattern, but also any *fixed point iteration* of the type

$$x^{k+1} = x^k + \alpha_k F(x^k)\,,$$

assuming the parameters α_k are chosen by some affine similar criterion. Hence, any linear combination of Newton and fixed point iteration may be considered as well: this leads to an iteration of the type

$$\left(I - \tau F'(x^k)\right)(x^{k+1} - x^k) = \tau F(x^k)\,,$$

which is nothing else than a *linearly implicit Euler discretization* of the above ordinary differential equation (1.10) with timestep τ to be adapted. As worked out in Section 6.4, such a *pseudo-transient continuation* method can be safely applied only, if the equilibrium point is dynamically *stable*—a condition anyway expected from geometrical insight. As a 'first choice', we then arrive at the following Lipschitz condition

$$|\left(F'(\bar{x}) - F'(x)\right)u| \le \omega|\bar{x} - x||u|\,.$$

Unfortunately, the canonical norm is computationally not easily available and at the same time may suffer from ill-conditioning—reflected in the size of $\mathrm{cond}(T)$. Therefore, upon keeping in mind that in affine similar problems domain and image space of F have the same transformation behavior, we are led to realize a 'second best' choice: we may switch from the canonical norm $|\cdot|$ to the standard norm $\|\cdot\|$ thus obtaining a Lipschitz condition of the structure

$$\|\left(F'(\bar{x}) - F'(x)\right)u\| \le \omega\|\bar{x} - x\|\cdot\|u\|\,.$$

However, in this way we lose the affine similarity property in the definition of ω, which means we have to apply careful scaling at least. In passing, we note that here the classical Lipschitz condition (1.4) arises directly from affine invariance considerations; however, a bounded inverse assumption like (1.2) is not needed in this context, but replaced by other conditions.

Scaling invariance. Scaling as discussed at the beginning of this section is a special affine transformation. In general, we will want to realize a scaling invariant algorithm, i.e. an algorithm that is invariant under the choice of units in the given problem. Closer examination shows that the four different affine invariance classes must be treated differently.

In an *affine covariant* setting, the formal assumption $B = I$ will certainly cover any fixed scaling transformation of the type $B = D$ so that 'dimensionless' variables

$$y = D^{-1}x\,,\ D = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)\,,\ \alpha_i > 0$$

are used at least inside the codes (internal scaling). For example, with components $x = (x_1, \ldots, x_n)$, *relative* scaling could mean any a-priori choice like

$$\alpha_i = |x_i^0|\,,\quad \text{if } |x_i^0| \neq 0$$

or an iterative adaptation like

$$\alpha_i^{k+1} = \max\bigl\{|x_i^k|\,,\ |x_i^{k+1}|\bigr\}\,.$$

Whenever these choices guarantee $\alpha_i > 0$, then scaling invariance is assured: to see this, just re-scale the components of x according to

$$x_i \longrightarrow \hat{x}_i = \beta_i x_i\,,$$

which implies

$$\alpha_i \longrightarrow \hat{\alpha}_i = \beta_i \alpha_i$$

and leaves

$$y_i = \frac{\hat{x}_i}{\hat{\alpha}_i} = \frac{x_i}{\alpha_i}$$

unchanged. In reality, however, *absolute threshold values* $\alpha_{\min} > 0$ have to be imposed in the form, say

$$\bar{\alpha}_i = \max\{\alpha_i, \alpha_{\min}\}$$

to avoid overflow for values close to zero. By construction, such threshold values spoil the nice scaling invariance property, unless they are defined for dimensionless components of the variable y.

In an *affine contravariant* setting, scaling should be applied in the image space of F, which means for the residual components

$$F \to G = D^{-1}F$$

with appropriately chosen diagonal matrix D.

For *affine similarity*, simultaneous scaling should be applied in both domain and image space

$$x\,,\ F \to y = D^{-1}x\,,\ G = D^{-1}F\,.$$

Finally, the *affine conjugate* energy products can be verified to be scaling invariant already by construction.

Further affine invariance classes. The four affine invariance classes mentioned so far actually represent the dominant classes of interest. Beyond these, certain combinations of these classes play a role in problems with appropriate substructures, each of which gives rise to one of the 'grand four'. As an example take optimization with equality constraints, which may require affine covariance or contravariance in the constraints, but affine conjugacy in the functional—see, e.g., the recent discussion [193] by S. Volkwein and M. Weiser.

1.2.3 The algorithmic paradigm

The key question treated in this book is how theoretical results from convergence analysis can be exploited for the construction of *adaptive* Newton algorithms. The key answer to this question is to realize *affine invariant computational estimates* of *affine invariant Lipschitz constants* that are cheaply available in the course of the algorithms. The realization is done as follows:

We identify some *theoretical local Lipschitz constant* ω defined over a nonempty domain D such that

$$\omega = \sup_{x,y,z\in D} g(x,y,z) \tag{1.11}$$

in terms of some scalar expression $g(x,y,z)$ that will only contain affine invariant terms. For ease of writing, we will mostly just write

$$g(x,y,z) \le \omega \text{ for all } x,y,z \in D\,,$$

even though we mean the best possible estimates (1.11) to characterize nonlinearity by virtue of Lipschitz constants. Once such a g has been selected, we exploit it by defining some corresponding *computational local estimate* according to

$$[\omega] = g(\hat{x},\hat{y},\hat{z}) \text{ for specific } \hat{x},\hat{y},\hat{z} \in D\,.$$

By construction, $[\omega]$ and ω share the same affine invariance property and satisfy the relation

$$[\omega] \le \omega\,.$$

Illustrating example. For the affine covariant Lipschitz condition (1.6) we have

$$\omega_0 = \sup_{x,y \in D} g(x, y, x^0) = \frac{\|F'(x^0)^{-1} \left(F'(x) - F'(y)\right) \|}{\|x - y\|} . \tag{1.12}$$

As a local affine covariant estimate, we may choose

$$[\omega_0] = g(x^1, x^0, x^0) = \frac{\|F'(x^0)^{-1} \left(F'(x^1) - F'(x^0)\right) \|}{\|x^1 - x^0\|} \tag{1.13}$$

in terms of the anyway computed Newton iterates x^0, x^1. In actual implementation, we will apply estimates different from (1.12) and (1.13), but preferable in the algorithmic context. The art in this kind of approach is to find out, among many possible theoretical characterizations, those ones that give rise to 'cheap and suitable' computational estimates and, in turn, lead to the construction of efficient algorithms.

There remains some gap $\omega - [\omega] \geq 0$, which can be reduced by appropriate reduction of the domain D. As will turn out, efficient adaptive Newton algorithms can be constructed, if $[\omega]$ catches at least one leading binary digit of ω—for details see the various *bit counting lemmas* scattered all over the book.

Remark 1.1 If the paradigm were realized *without* a strict observation of affine invariance of Lipschitz constants and estimates, then undesirable geometrical distortion effects (like those described in detail in (1.5)) would lead to totally unrealistic estimates and thus could not be expected to be a useful basis for any efficient algorithm.

Bibliographical Note. The general *paradigm* described here was, in an intuitive sense, already employed by P. Deuflhard in his 1975 paper on adaptive damping for Newton's method [63]. In 1979, the author formalized the whole approach introducing the notation $[\cdot]$ for computational estimates and exploited it for the construction of adaptive continuation methods [61]. Early on, H.G. Bock also took up the paradigm in his work on multiple shooting techniques for parameter identification and optimal control problems [29, 31, 32].

1.3 A Roadmap of Newton-type Methods

There is a large variety of Newton-type methods, which will be discussed in the book and therefore named and briefly sketched here.

Ordinary Newton method. For general nonlinear problems, the classical ordinary Newton method reads

$$F'(x^k)\Delta x^k = -F(x^k)\,,\ x^{k+1} = x^k + \Delta x^k\,, \qquad k = 0, 1, \ldots\,. \tag{1.14}$$

For $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ a Jacobian (n, n)-matrix is required. Sufficiently accurate Jacobian approximations can be computed by symbolic differentiation or by numerical differencing—see, for example, the automatic differentiation due to A. Griewank [112].

The above form of the linear system deliberately reflects the actual sequence of computation: first, compute the Newton corrections Δx^k, then improve the iterates x^k to obtain x^{k+1}—to avoid possible cancellation of significant digits, which might occur, if we solve for the new iterates x^{k+1} directly.

Simplified Newton method. This variant of Newton's method is characterized by keeping the initial derivative throughout the whole iteration:

$$F'(x^0)\overline{\Delta x}^k = -F(x^k)\,,\ x^{k+1} = x^k + \overline{\Delta x}^k\,,\ k = 0, 1, \ldots\,.$$

Compared to the ordinary Newton method, computational cost per iteration is saved—at the possible expense of increasing the number of iterations and possibly decreasing the convergence domain of the thus defined iteration.

Newton-like methods. This type of Newton method is characterized by the fact that, in finite dimension, the Jacobian matrices are either replaced by some fixed 'close by' Jacobian $F'(z)$ with $z \neq x^0$, or by some approximation so that

$$M(x^k)\delta x^k = -F(x^k)\,, \quad x^{k+1} = x^k + \delta x^k\,, \qquad k = 0, 1, \ldots\,.$$

As an example, deliberate 'sparsing' of a large Jacobian, which means dropping of 'weak couplings', will permit the use of a *direct sparse solver* for the Newton-like corrections and therefore possibly help to reduce the work per iteration; if really only weak couplings are dropped, then the total iteration pattern will not deteriorate significantly.

Exact Newton methods. Any of the finite dimensional Newton-type methods requires the numerical solution of the linear equations

$$F'(x^k)\Delta x^k = -F(x^k)\,.$$

Whenever *direct* elimination methods are applicable, we speak of *exact* Newton methods. However, naive application of direct elimination methods may cause serious trouble, if scaling issues are ignored.

Bibliographical Note. There are numerous excellent books on the numerical solution of linear systems—see, e.g., the classic by G.H. Golub and C.F. van Loan [107]. Programs for direct elimination in *full* or *sparse mode* can be found in the packages `LAPACK` [5], `SPARSPAK` [100], or [27]. As a rule, these codes leave the scaling issue to the user—for good reasons, since the user will typically know the specifications behind the problem that define the necessary scaling.

Local versus global Newton methods. Local Newton methods require 'sufficiently good' initial guesses. Global Newton methods are able to compensate for bad initial guesses by virtue of damping or adaptive trust region strategies. Exact global Newton *codes* for the solution of nonlinear equations are named `NLEQ` plus a characterizing suffix. We give details about

- `NLEQ-RES` for the residual based approach,
- `NLEQ-ERR` for the error oriented approach, or
- `NLEQ-OPT` for convex optimization.

Inexact Newton methods. For extremely large scale nonlinear problems the arising linear systems for the Newton corrections can no longer be solved directly ('exactly'), but must be solved *iteratively* ('inexactly')—which gives the name *inexact* Newton methods. The whole scheme then consists of an *inner iteration* (at Newton step k)

$$\begin{aligned} &F'(x^k)\ \delta x_i^k = -F(x^k) + r_i^k\,, \qquad k = 0, 1, \dots\,, \\ &x_i^{k+1} = x^k + \delta x_i^k\,, \qquad\qquad\qquad\quad i = 0, 1, \dots, i_{\max}^k \end{aligned} \tag{1.15}$$

in terms of *residuals* r_i^k and an *outer iteration* where, given x^0, the iterates are defined as

$$x^{k+1} = x_i^{k+1} \quad \text{for} \quad i = i_{\max}^k,\ k = 0, 1, \dots\ .$$

Compared with the exact Newton corrections in (1.14), *errors* $\delta x_i^k - \Delta x^k$ arise. Throughout the book, we will mostly drop the inner iteration index i for ease of notation.

In an *adaptive* inexact Newton method, the accuracy of the inner iteration should be matched to the outer iteration, preferably such that the Newton convergence pattern is essentially unperturbed—which means an appropriate control of $i_{\max}$ above. Criteria for the choice of the *truncation index* $i_{\max}$ depend on affine invariance, as will be worked out in detail. With this aspect in mind, inexact Newton methods are sometimes also called *truncated Newton methods*.

Inexact global Newton *codes* for the solution of large scale nonlinear equations are named `GIANT` plus a suffix characterizing the combination with an inner iterative solver. The name GIANT stands for Global Inexact Affine invariant Newton Techniques. We will work out details for

- `GIANT-GMRES` for the residual based approach,
- `GIANT-CGNE` and `GIANT-GBIT` for the error oriented approach, or
- `GIANT-PCG` for convex optimization.

As for the applied iterative solvers, see Section 1.4 below.

Preconditioning. A compromise between direct and iterative solution of the arising linear Newton correction equations is obtained by direct elimination of 'similar' linear systems, which can be used in a wider sense than just scaling as mentioned above. For its characterization we write

$$C_L F'(x^k) C_R C_R^{-1} \delta x_i^k = -C_L \left(F(x^k) - r_i^k \right) , \qquad i = 0, 1, \ldots i_{\max} \tag{1.16}$$

or, equivalently, also

$$C_L F'(x^k) C_R C_R^{-1} \left(\delta x_i^k - \Delta x_i^k \right) = C_L r_i^k , \qquad i = 0, 1, \ldots i_{\max} .$$

Consequently, within the algorithms any residual or error norms need to be replaced by their preconditioned counterparts

$$\| r_i^k \| , \| \delta x_i^k - \Delta x_i^k \| \quad \longrightarrow \quad \| C_L r_i^k \| , \| C_R^{-1} \left(\delta x_i^k - \Delta x_i^k \right) \| .$$

Matrix-free Newton methods. Linear iterative solvers within inexact Newton methods only require the evaluation of Jacobian matrix vector products so that numerical difference approximations

$$F'(x) v \doteq \frac{F(x + \delta v) - F(x)}{\delta}$$

can be conveniently realized. Note, however, that the quality of such directional difference approximations will heavily depend on the choice of the relative deviation parameter δ and the mantissa length of the used arithmetic. A numerically stable realization will use *automatic differentiation* as suggested by A. Griewank [112].

Secant method. For *scalar* equations, say $f(x) = 0$, this type of method is derived from Newton's method by substituting the tangent by the secant

$$f'(x^k + \delta x_k) \longrightarrow \frac{f(x^k + \delta x_k) - f(x^k)}{\delta x_k} = j_{k+1}$$

and computing the correction as

$$\delta x_{k+1} = -\frac{f(x^{k+1})}{j_{k+1}} , \; x^{k+1} = x^k + \delta x_k .$$

The thus constructed secant method is known to converge locally *superlinearly.*

Quasi-Newton methods. This class of methods extends the secant idea to systems of equations. In this case only a so-called *secant condition*

$$J\delta x_k = F(x^{k+1}) - F(x^k) \tag{1.17}$$

can be imposed, wherein J represents some Jacobian approximation to be specified. The above condition does not determine a unique J, but a whole class of matrices. If we recur to the previous quasi-Newton step as

$$J_k \delta x_k = -F(x^k)\,,$$

we may select special *Jacobian rank-1 updates* as

$$J_{k+1} = J_k + \frac{F(x^{k+1})z^T}{z^T \delta x_k}\,, \qquad z \in \mathbb{R}^n\,,\ z \neq 0\,,$$

where the vector z is arbitrary, in principle. As will be shown below in detail, the specification of z is intimately linked with affine invariance. Once z has been specified, the next quasi-Newton step

$$J_{k+1}\delta x_{k+1} = -F(x^{k+1})$$

is determined. In the best case, *superlinear* local convergence can be shown to hold again. A specification to *linear* systems is the algorithm `GBIT` described in Section 1.4.4 below.

Gauss-Newton methods. This type of method applies to nonlinear least squares problems, whether unconstrained or constrained. The method requires the nonlinear least squares problems to be statistically well-posed, characterized either as 'small residual' (Section 4.2.1) or as 'adequate' problems (Section 4.3.2). For this problem class, *local* Gauss-Newton methods are appropriate, when 'sufficiently good' initial guesses are at hand, while *global* Gauss-Newton methods are used, when only 'bad initial guesses' are available. In the statistics community Gauss-Newton methods are also called *scoring methods*.

Quasilinearization. Infinite dimensional Newton methods for operator equations are also called *Newton methods in function space* or quasilinearization. The latter name stems from the fact that the nonlinear operator equation is solved via a sequence of corresponding linearized operator equations. Of course, the linearized equations for the Newton corrections can only be solved *approximately*. Consequently, *inexact* Newton methods supply the correct theoretical frame, within which now the 'truncation errors' represent *approximation errors*, typically *discretization errors*.

Inexact Newton multilevel methods. We reserve this term for those multilevel schemes, wherein the arising infinite dimensional linear Newton systems are approximately solved by some linear multilevel or multigrid method; in such a setting, Newton methods act in function space. The highest degree of sophistication of an inexact Newton multilevel method would be an *adaptive* Newton multilevel method, where the approximation errors are controlled within an abstract framework of inexact Newton methods.

Multilevel Newton methods. Unfortunately, the literature is often not unambiguous in the choice of names. In particular, the name 'Newton multigrid method' is often given to schemes, wherein a finite dimensional Newton multigrid method is applied on each level—see, e.g., the classical textbook [113] by W. Hackbusch or the more recent treatment [135] by R. Kornhuber, who uses advanced functional analytic tools. In order to avoid confusion, such a scheme will here be named 'multilevel Newton method'.

Nonlinear multigrid methods. For the sake of clarity, it may be worth mentioning that 'nonlinear multigrid methods' are not Newton methods, but fixed point iteration methods, and therefore not treated within the scope of this book.

Bibliographical Note. The classic among the textbooks for the numerical solution of finite dimensional systems of nonlinear equations has been the 1970 book of J.M. Ortega and W.C. Rheinboldt [163]. It has certainly set the state of the art for quite a long time. The monograph [177] by W.C. Rheinboldt guides into related more recent research areas. The popular textbook [132] by C.T. Kelley offers a nice introduction into finite dimensional inexact Newton methods—see also references therein. The technique of 'preconditioning' is usually attributed to O. Axelsson—see his textbook [11] and references therein. Multigrid Newton methods are worked out in detail in the meanwhile classic text of W. Hackbusch [113]; a detailed convergence analysis of such methods for certain smooth as well as a class of non-smooth problems has been recently given by R. Kornhuber [135].

1.4 Adaptive Inner Solvers for Inexact Newton Methods

As stated in Section 1.3 above, *inexact* Newton methods require the linear systems for the Newton corrections to be solved *iteratively*. Different affine invariance concepts naturally go with different concepts for the iterative solution. In particular, recall that

- *residual* norms go with *affine contravariance*,
- *error* norms go with *affine covariance*,

- *energy* norms go with *affine conjugacy*.

For the purpose of this section, let the inexact Newton system (1.15) be written as

$$Ay_i = b - r_i, \qquad i = 0, 1, \ldots i_{\max}$$

in terms of iterative approximations y_i for the solution y and iterative residuals r_i. In order to control the number $i_{\max}$ of iterations, several *termination criteria* may be realized:

- Terminate the iteration as soon as the *residual norm* $\|r_i\|$ is small enough.
- Terminate the iteration as soon as the iterative *error norm* $\|y - y_i\|$ is small enough.
- If the matrix A is symmetric positive definite, terminate the iteration as soon as the *energy norm* $\|A^{1/2}(y - y_i)\|$ of the error is small enough.

In what follows, we briefly sketch some of the classical iterative linear solvers with particular emphasis on appropriate termination criteria for use within inexact Newton algorithms. We will restrict our attention to those iterative solvers, which minimize or, at least, reduce

- the residual norm (`GMRES`, Section 1.4.1),
- the energy norm of the error (`PCG`, Section 1.4.2), and
- the error norm (`CGNE`, Section 1.4.3, and `GBIT`, Section 1.4.4).

We include the less known solver `GBIT`, since it is a *quasi-Newton method* specialized to the solution of *linear* systems.

Preconditioning. This related issue deals with the iterative solution of systems of the kind

$$C_L A C_R C_R^{-1} y_i = C_L(b - r_i), \qquad i = 0, 1, \ldots i_{\max}\,, \tag{1.18}$$

where left preconditioner C_L and right preconditioner C_R arise. A proper choice of preconditioner will exploit information from the problem class under consideration and often crucially affect the convergence speed of the iterative solver.

Bi-CGSTAB. Beyond the iterative algorithms selected here, there are numerous further ones of undoubted merits. An example is the iterative solver `Bi-CG` and its stabilized variant `Bi-CGSTAB` due to H.A. van der Vorst [189]. This solver might actually be related to affine similarity as treated above in Section 1.2; as a consequence, this code would be a natural candidate within an inexact pseudo–continuation method (see Section 6.4.2). However, this combination of inner and outer iteration would require a rather inconvenient norm (Jordan canonical norm). That is why we do not incorporate this candidate here. However, further work along this line might be promising.

Bibliographical Note. A good survey on many aspects of the iterative solution of linear equation systems can be found in the textbook [181] by Y. Saad. Preconditioning techniques are described, e.g., in the textbook [11] by O. Axelsson.

Multilevel discretization. For the adaptive realization of inexact Newton methods in function space, discretizations on successively finer levels play the role of the inner iteration. That is why we additionally treat linear multigrid methods in Section 1.4.5 below. Skipping any technical details here, multilevel methods permit an adaptive control of discretization errors on each level—for example, see Section 7.3.3 on Fourier-Galerkin methods for periodic orbit computation, Section 7.4.2 on polynomial collocation methods for ODE boundary value problems, and Section 8.3 on adaptive multigrid methods for elliptic PDEs.

1.4.1 Residual norm minimization: `GMRES`

A class of iterative methods aims at the successive reduction of the residual norms $\|r_i\|$ for increasing index i. Outstanding candidates among these are those solvers that even *minimize the residual norms* over some Krylov subspace—such as `GMRES` and `CGNR`. Since algorithm `GMRES` requires less matrix/vector multiplies per step, we focus our attention on it here.

Algorithm `GMRES`. Given an initial approximation $y_0 \approx y$ compute the initial residual $r_0 = b - Ay_0$. Set $\beta = \|r_0\|_2\,, v_1 = r_0/\beta\,, V_1 = v_1$.
For $i = 1, 2, \ldots, i_{\max}$:

I. Orthogononalization:	$\hat{v}_{i+1} = Av_i - V_i h_i$ where $h_i = V_i^T A v_i$
II. Normalization:	$v_{i+1} = \hat{v}_{i+1}/\|\hat{v}_{i+1}\|_2$
III. Update:	$V_{i+1} = (V_i\ v_{i+1})$
	$H_i = \begin{pmatrix} H_{i-1} & h_i \\ 0 & \|\hat{v}_{i+1}\|_2 \end{pmatrix}$
	H_i is an $(i+1, i)$-Hessenberg matrix (for $i = 1$ drop the left block column)
IV. Least squares problem for z_i :	$\|\beta e_1 - H_i z\| = \min$
V. Approximate solution ($i = i_{\max}$):	$y_i = V_i z_i + y_0$

Array storage. Up to iteration step i, the above implementation requires to store $i + 2$ vectors of length n.

Computational amount. In each iteration step i, there is one matrix/vector multiply needed. Up to step i, the Euclidean inner products sum up to $\sim i^2 n$ flops.

As already stated, this algorithm *minimizes* the residual norms over the Krylov subspace

$$\mathcal{K}_i(r_0, A) = \text{span}\,\{r_0, \ldots, A^{i-1} r_0\}\,.$$

By construction, the inner residuals will *decrease monotonically*

$$\|r_{i+1}\|_2 \leq \|r_i\|_2\,.$$

Therefore, a reasonable *inner termination criterion* will check whether the final residual $\|r_i\|_2$ is 'small enough'. Moreover, starting with arbitrary initial guess y_0 and initial residual $r_0 \neq 0$, we have the *orthogonality relation* (in terms of the Euclidean inner product $\langle \cdot\,, \cdot \rangle$)

$$\langle r_i, r_i - r_0 \rangle = 0\,,$$

which directly implies that

$$\|r_0\|_2^2 = \|r_0 - r_i\|_2^2 + \|r_i\|_2^2 \tag{1.19}$$

throughout the inner iteration. If we define

$$\eta_i = \frac{\|r_i\|_2}{\|r_0\|_2}\,,$$

then we will generically have $\eta_i < 1$ for $i > 0$ and

$$\eta_{i+1} < \eta_i\,, \quad \text{if} \quad \eta_i \neq 0\,.$$

This implies that, after a number of iterations, any adaptive truncation criterion

$$\eta_i \leq \bar{\eta}$$

for a prescribed threshold value $\bar{\eta} < 1$ can be met. In passing we note that then (1.19) can be rewritten as

$$\|r_0 - r_i\|_2^2 = (1 - \eta_i^2)\|r_0\|_2^2\,. \tag{1.20}$$

These detailed results are applied in Sections 2.2.4 and 3.2.3.

Preconditioning. Finally, if (1.18) is applied, then the Euclidean norms of the *preconditioned residuals* $\bar{r}^k = C_L r^k$ are iteratively *minimized* in `GMRES`, whereas C_R only affects the rate of convergence. Therefore, if strict residual minimization is aimed at, then only *right* preconditioning should be implemented, which means $C_L = I$.

Bibliographical Note. This iterative method has been designed as a rather popular code by Y. Saad and M.H. Schultz [182] in 1986; an earlier often overlooked derivation has been given by G.I. Marchuk and Y.A. Kuznetsov [146] already in 1968.

1.4.2 Energy norm minimization: PCG

With A *symmetric positive definite*, we are able to define the *energy product* $(\cdot,\cdot)$ and its induced *energy norm* $\|\cdot\|_A$ by

$$(u,v) = \langle u, Av\rangle, \quad \|u\|_A^2 = (u,u)$$

in terms of the Euclidean inner product $\langle\cdot,\cdot\rangle$. Let $B \approx A^{-1}$ denote some preconditioning matrix, assumed to be also *symmetric positive definite*. Usually the numerical realization of $z = Bc$ is much simpler and faster than the solution of $Ay = b$. Formally speaking, we may specify some $C_L = C_R^T = B^{1/2}$. This specification does not affect the energy norms, but definitely the speed of convergence of the iteration. Any *preconditioned conjugate gradient* (PCG) method reads:

Algorithm PCG. For given approximation $y_0 \approx y$ compute the initial residual $r_0 = b - Ay_0$ and the preconditioned residual $\bar r_0 = Br_0$. Set $p_0 = \bar r_0$ $\sigma_0 = \langle r_0, \bar r_0\rangle = \|r_0\|_B^2$.
For $i = 0, 1, \ldots, i_{\max}$:

$$\begin{aligned}
\alpha_i &= \frac{\|p_i\|_A^2}{\sigma_i} \\
y_{i+1} &= y_i + \frac{1}{\alpha_i} p_i \\
\gamma_i^2 &= \frac{\sigma_i}{\alpha_i} \qquad \text{(energy error contribution} \quad \|y_{i+1} - y_i\|_A^2) \\
r_{i+1} &= r_i - \frac{1}{\alpha_i} A p_i, \quad \bar r_{i+1} = B r_{i+1} \\
\sigma_{i+1} &= \|r_{i+1}\|_B^2, \qquad \beta_{i+1} = \frac{\sigma_{i+1}}{\sigma_i} \\
p_{i+1} &= \bar r_{i+1} + \beta_{i+1} p_i.
\end{aligned}$$

Array storage. Up to iteration step i, ignoring any preconditioners, the above implementation requires to store only 4 vectors of length n.

Computational amount. In each iteration step i, there is one matrix/vector multiply needed. Up to step i, the Euclidean inner products sum up to $\sim 5in$ flops.

This iteration successively *minimizes* the energy error norm $\|y - y_i\|_A$ within the associated Krylov subspace

$$\mathcal{K}_i(r_0, A) = \text{span}\{r_0, \dots, A^{i-1}r_0\}\,.$$

By construction, we have the orthogonality relations (also called *Galerkin conditions*)

$$(y_i - y_0, y_{i+m} - y_i)_A = 0\,, \quad m = 1, \dots,$$

which imply the orthogonal decompositions (with $m = 1$)

$$\|y_{i+1} - y_0\|_A^2 = \|y_{i+1} - y_i\|_A^2 + \|y_i - y_0\|_A^2 \tag{1.21}$$

and (with $m = n - i$ and $y_n = y$)

$$\|y - y_0\|_A^2 = \|y - y_i\|_A^2 + \|y_i - y_0\|_A^2\,. \tag{1.22}$$

From (1.21) we easily derive that

$$\|y_i - y_0\|_A^2 = \sum_{j=0}^{i-1} \|y_{j+1} - y_j\|_A^2 = \sum_{j=0}^{i-1} \gamma_j^2\,. \tag{1.23}$$

Together with (1.22), we then obtain

$$\epsilon_i = \|y - y_i\|_A^2 = \sum_{j=i}^{n-1} \gamma_j^2\,. \tag{1.24}$$

Estimation of PCG error. Any adaptive *affine conjugate* inexact Newton algorithm will require a reasonable estimate for the errors ϵ_i to be able to exploit the theoretical convergence results. Note that the monotonicity

$$\|y_i - y_0\|_A \le \|y_{i+1} - y_0\|_A \le \cdots \le \|y - y_0\|_A$$

can be derived from (1.21)—a saturation effect easily observable in actual computation. There are two basic methods to estimate the error.

(I) Assume we have a computable upper bound

$$\bar{\epsilon}_0 \ge \|y - y_0\|_A^2\,,$$

such as the ones suggested by G.H. Golub and G. Meurant [104] or by B. Fischer [92]. Then, with (1.22) and (1.23), we obtain the computable upper bound

$$\epsilon_i \le \bar{\epsilon}_0 - \sum_{j=0}^{i-1} \gamma_j^2 = [\epsilon_i]\,.$$

(II) As an alternative (see [68]), we may exploit the structure of (1.24) via the lower bound

$$[\epsilon_i] = \sum_{j=i}^{i+m} \gamma_j^2 \le \epsilon_i \tag{1.25}$$

for some sufficiently large index $m > 0$—which means to continue the iteration further just for the purpose of getting some error estimate. In the case of 'fast' convergence (usually for 'good' preconditioners only), few terms in the sum will suffice. Typically, we use this technique, since it does not require any choice of an upper bound $\bar{\epsilon}_0$.

Both techniques inherit the *monotonicity* $[\epsilon_{i+1}] \leq [\epsilon_i]$ from $\epsilon_{i+1} \leq \epsilon_i$. Hence, generically, after a number of iterations, any adaptive truncation criterion

$$[\epsilon_i] \leq \overline{\epsilon}$$

for a prescribed threshold value $\overline{\epsilon}$ can be met. In the inexact Newton-PCG algorithms to be worked out below we will use the *relative* energy error norms defined, for $i > 0$, as

$$\delta_i = \frac{\|y - y_i\|_A}{\|y_i\|_A} \approx \frac{\sqrt{[\epsilon_i]}}{\|y_i\|_A} .$$

Whenever $y_0 = 0$, then (1.21) implies the monotone increase $\|y_{i+1}\|_A \geq \|y_i\|_A$ and therefore the monotone decrease $\delta_{i+1} \leq \delta_i$. This guarantees that any relative error criterion

$$\delta_i \leq \overline{\delta}$$

can be met. Moreover, in this case we have the relation

$$\|y_i\|_A^2 = (1 + \delta_i^2)\|y\|_A^2 . \tag{1.26}$$

For $y_0 \neq 0$, the above monotonicities and (1.26) no longer hold. This option, however, is not used in the inexact Newton-PCG algorithms to be derived in Sections 2.3.3 and 3.4.3 below.

1.4.3 Error norm minimization: CGNE

Another class of iterative solvers aims at the successive reduction of the (possibly scaled) Euclidean error norms $\|y - y_i\|$ for $i = 0, 1, \ldots$. Among these the ones that *minimize the error norms* over some Krylov subspace play a special role. For nonsymmetric Jacobian matrices the outstanding candidate with this feature seems to be CGNE. For its economic implementation, we recommend Craig's variant (see, e.g., [181]), which reads:

Algorithm CGNE. Given an initial approximation y_0, compute the initial residual $r_0 = b - Ay_0$ and set $p_0 = 0, \beta_0 = 0, \sigma_0 = \|r_0\|^2$.

For $i = 1, 2, \ldots, i_{\max}$:

$$
\begin{aligned}
p_i &= A^T r_{i-1} + \beta_{i-1} p_{i-1} \\
\alpha_i &= \sigma_{i-1} / \|p_i\|^2 \\
\gamma_{i-1}^2 &= \alpha_i \sigma_{i-1} \qquad \text{(Euclidean error contribution} \quad \|y_i - y_{i-1}\|^2) \\
y_i &= y_{i-1} + \alpha_i p_i \\
r_i &= r_{i-1} - \alpha_i A p_i \\
\sigma_i &= \|r_i\|^2 \\
\beta_i &= \sigma_i / \sigma_{i-1}
\end{aligned}
$$

Array storage. Up to iteration step i, the above implementation requires to store only 3 vectors of length n.

Computational amount. In each iteration step i, there are *two* matrix/vector multiplies needed. Up to step i, the Euclidean inner products sum up to $\sim 5in$ flops.

This iteration successively *minimizes* the Euclidean norms $\|y - y_i\|$ within the Krylov subspace

$$\mathcal{K}_i(A^T r_0, A^T A) = \text{span}\{A^T r_0, \dots, (A^T A)^{i-1} A^T r_0\}\,.$$

By construction, we have the orthogonality relations (also: *Galerkin conditions*)

$$(y_i - y_0, y_{i+m} - y_i) = 0\,, \quad m = 1, \dots,$$

which imply the orthogonal decomposition (with $m = 1$)

$$\|y_{i+1} - y_0\|^2 = \|y_{i+1} - y_i\|^2 + \|y_i - y_0\|^2 \tag{1.27}$$

and (with $m = n - i$ and $y_n = y$)

$$\|y - y_0\|^2 = \|y - y_i\|^2 + \|y_i - y_0\|^2\,. \tag{1.28}$$

From (1.27) we easily derive that

$$\|y_i - y_0\|^2 = \sum_{j=0}^{i-1} \|y_{j+1} - y_j\|^2 = \sum_{j=0}^{i-1} \gamma_j^2\,. \tag{1.29}$$

Together with (1.28), we then obtain

$$\epsilon_i = \|y - y_i\|^2 = \sum_{j=i}^{n-1} \gamma_j^2\,. \tag{1.30}$$

Estimation of CGNE error. Any adaptive *affine covariant* inexact Newton algorithm will require a reasonable estimate for the errors ϵ_i to be able to exploit the theoretical convergence results. Again, the saturation effect from the monotonicity

$$\|y_i - y_0\| \leq \|y_{i+1} - y_0\| \leq \cdots \leq \|y - y_0\|$$

can be derived from (1.27).

There are two basic methods to estimate the error within CGNE.

(I) Assume we have a computable upper bound

$$\bar{\epsilon}_0 \geq \|y - y_0\|^2$$

in the spirit of those suggested by G.H. Golub and G. Meurant [104] or by B. Fischer [92]. From this, with (1.28) and (1.29), we obtain the computable upper bound

$$\epsilon_i \leq \bar{\epsilon}_0 - \sum_{j=0}^{i-1} \gamma_j^2 = [\epsilon_i] \,.$$

(II) As an alternative, transferring an idea from [68], we may exploit the structure of (1.30) to look at the lower bound

$$[\epsilon_i] = \sum_{j=i}^{i+m} \gamma_j^2 \leq \epsilon_i \tag{1.31}$$

for some sufficiently large index $m > 0$—which means to continue the iteration further just for the purpose of getting some error estimate. In the case of 'fast' convergence (for 'sufficiently good' preconditioner, see below), only few terms in the sum will be needed. Typically, we use this second technique.

Both techniques inherit the *monotonicity* $[\epsilon_{i+1}] \leq [\epsilon_i]$ from $\epsilon_{i+1} \leq \epsilon_i$. After a number of iterations, any adaptive truncation criterion

$$[\epsilon_i] \leq \overline{\epsilon}$$

for a prescribed threshold value $\overline{\epsilon}$ can generically be met. In the inexact Newton-ERR algorithms to be worked out below we will use the *relative* error norms defined, for $i > 0$, as

$$\delta_i = \frac{\|y - y_i\|}{\|y_i\|} \approx \frac{\sqrt{[\epsilon_i]}}{\|y_i\|} \,. \tag{1.32}$$

Whenever $y_0 = 0$, then (1.27) implies the monotonicities $\|y_{i+1}\| \geq \|y_i\|$ and $\delta_{i+1} \leq \delta_i$. The latter one guarantees that the relative error criterion

$$\delta_i \leq \overline{\delta} \tag{1.33}$$

can be eventually met. For this initial value we also have the relation

$$\|y_i\|^2 = (1 + \delta_i^2)\|y\|^2 . \tag{1.34}$$

These detailed results enter into the presentation of local inexact Newton-ERR methods in Section 2.1.5.

For $y_0 \neq 0$, the above monotonicities as well as the relation (1.34) no longer hold. This situation occurs in the global inexact Newton-ERR method to be derived in Section 3.3.4. Since the $\|y_i\|$ eventually approach $\|y\|$, we nevertheless require the relative truncation criterion (1.33).

Preconditioning. Finally, if (1.18) is applied, then the norms of the iterative *preconditioned errors* $C_R^{-1}(y - y_i)$ are minimized. Therefore, if *strict* unscaled error minimization is aimed at, then only *left* preconditioning should be realized. In addition, if 'good' preconditioners C_R or C_L are available (resulting in 'fast' convergence), then the simplification

$$\|C_R^{-1}(y - y_i)\| \approx \|C_R^{-1}(y_{i+1} - y_i)\|$$

will be sufficient.

Remark 1.2 Numerical experiments with large discretized PDEs in Section 8.2.1 document a poor behavior of `CGNE`, which seems to stem from a rather sensitive dependence on the choice of preconditioner. Generally speaking, a preconditioner, say B, is expected to reduce the condition number $\kappa(J)$ to some $\kappa(BJ) \ll \kappa(J)$. However, as the algorithm `CGNE` works on the normal equations, the characterizing dependence is on $\kappa^2(BJ) \gg \kappa(BJ)$. In contrast to this behavior, the preconditioned `GMRES` just depends on $\kappa(BJ)$ as the characterizing quantity.

1.4.4 Error norm reduction: `GBIT`

The quasi-Newton methods already mentioned in Section 1.3 can be specified to apply to *linear* systems as well. Following the original paper by P. Deuflhard, R. Freund, and A. Walter [74], a special *affine covariant* rank-1 update can be chosen, which turns out to be Broyden's 'good' update [40]. Especially for linear systems, an optimal line search is possible, which then gives the algorithm `GBIT` (abbreviation for Good Broyden ITerative solver for linear systems).

Preconditioner improvement. The main idea behind this algorithm is to improve any (given) initial preconditioner $B_0 \sim A$, or $H_0 \sim A^{-1}$, respectively, successively to $B_i \sim A, H_i \sim A^{-1}$. Let $E_i = I - A^{-1}B_i$ denote the preconditioning error, then each iterative step can be shown to realize some new preconditioner such that

$$\|E_{i+1}\|_2 \le \|E_i\|_2 \,, \quad i = 0, 1, \ldots \,.$$

Error reduction. In [74], this algorithm has been proven to converge under the sufficient assumption

$$\|E_0\|_2 < \tfrac{1}{3} \,, \tag{1.35}$$

in the sense that

$$\|y_{i+1} - y\| < \|y_i - y\| \,. \tag{1.36}$$

Moreover, asymptotic *superlinear* convergence can even be shown. Numerical experience shows that an assumption weaker than (1.35) might do, but there is no theoretical justification of such a statement yet.

The actual implementation of the algorithm does not store the improved preconditioners H_i explicitly, but exploits the Sherman-Morrison formula to obtain a cheap recursion. The following implementation is a recent slight improvement over the algorithm `GB` suggested in [74]—essentially replacing the iterative stepsize $t_i = \tau_i$ therein by some modification (see below and Exercise 1.4 for $\tau_{\max} = 1$).

Algorithm `GBIT`. Given an initial guess y_0, an initial preconditioner $H_0 \sim A^{-1}$, and some inner product $\langle u, v \rangle$.
Initialization:

$$\begin{aligned} r_0 &= b - Ay_0 \\ \Delta_0 &= H_0 r_0 \\ \sigma_0 &= \langle \Delta_0, \Delta_0 \rangle \end{aligned}$$

Iteration loop $i = 0, 1, \ldots, i_{\max}$:

$$\begin{aligned} q_i &= A\Delta_i \\ \zeta_0 &= H_0 q_i \end{aligned}$$

Update loop $m = 0, \ldots, i-1$ (for $i \ge 1$):

$$\zeta_{m+1} = \zeta_m + \frac{\langle \Delta_m, \zeta_m \rangle}{\sigma_m} \left(\Delta_{m+1} - (1 - t_m)\Delta_m \right)$$

$$\begin{aligned} z_i &= \zeta_i \\ \gamma_i &= \langle \Delta_i, z_i \rangle \\ \tau_i &= \sigma_i / \gamma_i \\ \text{if} \quad \tau_i &< \tau_{\min} : \quad \textbf{restart} \\ t_i &= \tau_i \end{aligned}$$

$$
\begin{array}{llll}
\text{if} & t_i & > & \tau_{\max}: \quad t_i = 1 \\
& y_{i+1} & = & y_i + t_i \Delta_i \\
& (r_{i+1} & = & r_i - t_i q_i) \\
& \Delta_{i+1} & = & (1 - t_i + \tau_i)\Delta_i - \tau_i z_i \\
& \sigma_{i+1} & = & \langle \Delta_{i+1}, \Delta_{i+1} \rangle \\
& \epsilon_i & = & \frac{1}{2}\sqrt{\sigma_{i-1} + 2\sigma_i + \sigma_{i+1}} \\
\text{if} & \epsilon_i & \leq & \rho \|y_{i+1}\| \cdot \mathrm{ERRTOL}: \quad \textbf{solution found}
\end{array}
$$

The parameters $\tau_{\min}, \tau_{\max}$ are set internally such that $0 < \tau_{\min} \ll 1, \tau_{\max} \geq 1$, the safety factor $\rho < 1$ and the error tolerance ERRTOL are user prescribed values.

Array storage. Up to iteration step i, the above recursive implementation requires to store the $i + 3$ vectors

$$\Delta_0, \ldots, \Delta_i, q, z \equiv \zeta \,.$$

of length n.

Computational amount. In each iteration step i, the computational work is dominated by one matrix/vector multiply, one solution of a preconditioned system ($\zeta_0 = H_0 q$), and $\sim 2i \cdot n$ flops for the Euclidean inner product. Up to step i, this sums up to i preconditioned systems and $\sim i^2 n$ flops.

Inner product. Apart from the Euclidean inner product $\langle u, v \rangle = u^T v$ any scaled version such as $\langle u, v \rangle = (D^{-1}u)^T D^{-1} v$, with D a diagonal scaling matrix, will be applicable—and even preferable. For special problems like discrete PDE boundary value problems certain discrete L^2-products and norms are recommended.

Error estimation and termination criterion. By construction, we obtain the relation

$$y_i - y = \Delta_i - E_i \Delta_i \,,$$

which, under the above preconditioning assumption, certainly implies the estimation property

$$\left(1 - \frac{\|E_i \Delta_i\|}{\|\Delta_i\|}\right) \|\Delta_i\| \leq \|y_i - y\| \leq \left(1 + \frac{\|E_i \Delta_i\|}{\|\Delta_i\|}\right) \|\Delta_i\| \,.$$

Hence, the true error can be roughly estimated as

$$\|y_i - y\| \approx \|\Delta_i\| = \sqrt{\sigma_i} \,.$$

In order to suppress possible outliers, an average of the kind

$$\epsilon_i = \frac{1}{2}\sqrt{\sigma_{i-1} + 2\sigma_i + \sigma_{i+1}} \tag{1.37}$$

is typically applied for $i > 1$. This estimator leads us to the relative termination criterion

$$\epsilon_i \leq \rho \|y_{i+1}\| \cdot \text{ERRTOL}\,,$$

as stated above.

Note that the above error estimator cannot be shown to inherit the monotonicity property (1.36). Consequently, this algorithm seems to be less efficient than `CGNE` within the frame of Newton-ERR algorithms. Surprisingly, this expectation is not at all in agreement with numerical experiments—see, for instance, Section 8.2.1.

Remark 1.3 On top of `GBIT` we also applied ideas of D.M. Gay and R.B. Schnabel [97] about successive orthogonalization of update vectors to construct some projected 'good' Broyden method for linear systems. The corresponding algorithm `PGBIT` turned out to require only slightly less iterations, but significantly more array storage and computational amount—and is therefore omitted here.

1.4.5 Linear multigrid methods

In Newton methods for *operator equations* the corresponding iterates and solutions live in appropriate *infinite* dimensional function spaces. For example, in steady state partial differential equations (PDEs), the solutions live in some Sobolev space—like H^α—depending on the prescribed boundary conditions. It is an important mathematical paradigm that any such infinite dimensional space should not just be represented by a single finite dimensional space of possibly high dimension, but by a *sequence of finite dimensional subspaces with increasing dimension.*

Consequently, any infinite dimensional Newton method will be realized via a *sequence* of finite dimensional linearized systems

$$Ay_j = b + r_j, \qquad j = 0, 1, \ldots j_{\max}\,,$$

where the residuals r_j represent *approximation errors*, mostly *discretization errors.* Each of the subsystems is again solved iteratively, which gives rise to the question of accuracy matching of discretization versus iteration. This is the regime of linear *multigrid* or *multilevel* methods—see, e.g., the textbook of W. Hackbusch [113] and Chapter 8 below on Newton multilevel methods.

Adaptivity. In quite a number of application problems rather localized phenomena occur. In this case, uniform grids are by no means optimal, which, in turn, also means that the classical multigrid methods on uniform grids could not be regarded as optimal. For this reason, multigrid methods on *adaptive* grids have been developed quite early, probably first by R.E. Bank

[18] in his code PLTMG and later in the family UG of parallel codes [22, 21] by G. Wittum, P. Bastian, and their groups.

Independent of the classical multigrid methods, a multilevel method based on conjugate gradient iteration with some *hierarchical basis* (HB) *preconditioning* had been suggested for elliptic PDEs by H. Yserentant [204]. An *adaptive* 2D version of the new method had been first designed and implemented by P. Deuflhard, P. Leinen, and H. Yserentant [78] in the code KASKADE. A more mature version including also 3D has been worked out by F. Bornemann, B. Erdmann, and R. Kornhuber [36]. The present version of KASKADE [23] contains the original HB-preconditioner for 2D and the more recent BPX-preconditioner due to J. Xu [200, 39] for 3D.

Additive versus multiplicative multigrid methods. In the interpretation of multigrid methods as abstract Schwarz methods as given by J. Xu [201], which the author prefers to adopt, the classical multigrid methods are now called *multiplicative multigrid methods*, whereas the HB- or BPX-preconditioned conjugate gradient methods are called *additive multigrid methods*. In general, any difference in speed between additive or multiplicative multigrid methods is only marginal, since the bulk of computing time is anyway spent in the evaluation of the stiffness matrix elements and the right hand side elements. For the orientation of the reader: UG is nearly exclusively multiplicative, PLTMG is predominantly multiplicative with some additive options, KASKADE is predominantly additive with some multiplicative code for special PDE eigenvalue problems.

Cascadic multigrid methods. These rather recent multigrid methods can be understood as a confluence of additive and multiplicative multigrid methods. From the additive point of view, cascadic multigrid methods are characterized by the simplest possible preconditioner: either no or just a diagonal preconditioner is applied; as a distinguishing feature, coarser levels are visited more often than finer levels—to serve as preconditioning substitutes. From the multiplicative side, cascadic multigrid methods may be understood as multigrid methods with an increased number of smoothing iterations on coarser levels, but without any coarse grid corrections. A first algorithm of this type, the *cascadic conjugate gradient method* (algorithm CCG) had been proposed by the author in [68]. First rather restrictive convergence results were due to V. Shaidurov [185]. The general cascadic multigrid method class with arbitrary inner iterations beyond conjugate gradient methods has been presented by F. Bornemann and P. Deuflhard [35].

Just to avoid mixing terms: *cascadic* multigrid methods are different from the code KASKADE, which predominantly realizes *additive* multigrid methods.

Local error estimators. Any efficient implementation of *adaptive* multigrid methods (additive, multiplicative, cascadic) must be based on cheap *local*

error estimators or, at least, *local error indicators*. In the best case, these are derived from theoretical *a-posteriori error estimates*. These estimates will be local only, if local (right hand side) perturbations in the given problem remain local—i.e., if the Greens' function of the PDE problem exhibits local behavior. As a consequence of this elementary insight, *adaptive* multigrid methods will be essentially applicable to linear or nonlinear elliptic problems (among the stationary PDE problems). A comparative assessment of the different available local error estimators has been given by F. Bornemann, B. Erdmann, and R. Kornhuber in [37]. In connection with any error estimator, the local extrapolation method due to I. Babuška and W.C. Rheinboldt [14] can be applied. The art of refinement is quite established in 2D (see the 'red' and 'green' refinements due to R.E. Bank et al. [20]) and still under further improvement in 3D.

Summarizing, adaptive multilevel methods for linear PDEs play a dominant role in the frame of adaptive Newton multilevel methods for nonlinear PDEs—see, e.g., Section 8.3.

Exercises

Exercise 1.1 Given a nonlinear C^1-mapping $F : X \to Y$ over some domain $D \subset X$ for Banach spaces X, Y, each endowed with some norm $\|\cdot\|$. Assume a Lipschitz condition of the form

$$\|F'(x) - F'(y)\| \leq \gamma \|x - y\|\,,\ x, y \in D\,.$$

Let the derivative at some point x^0 have a bounded inverse with

$$\|F'(x^0)^{-1}\| \leq \beta_0\,.$$

Show that then, for all arguments $x \in D$ in some open ball $S(x^0, \rho)$ with $\rho = \dfrac{1}{\beta_0 \gamma}$, there exists a bounded derivative inverse with

$$\|F'(x)^{-1}\| \leq \frac{\beta_0}{1 - \beta_0 \gamma \|x - x^0\|}\,.$$

Exercise 1.2 Usual proofs of the implicit function theorem apply the Newton-Kantorovich theorem—compare Section 1.2. Revisit this kind of proof in any available textbook in view of affine covariance. In particular, replace condition (1.3) for a locally bounded inverse and Lipschitz condition (1.4) by some affine covariant Lipschitz condition like (1.6), which defines

some local affine covariant Lipschitz constant ω_0. Formulate the thus obtained affine covariant implicit function theorem. Characterize the class of problems, for which $\omega_0 = \infty$.

Exercise 1.3 Consider the scalar monomial equation

$$f(x) = x^m - a = 0\,.$$

We want to study the convergence properties of Newton's method. For this purpose consider the general corresponding contraction term

$$\Theta(x) = \frac{f f''}{f'^2}\,.$$

Verify this expression in general and calculate it for the specific case. What kind of convergence occurs for $m \neq 1$? How could the Newton method be 'repaired' such that quadratic convergence still occurs? Why is this, in general, not a good idea?

Hint: Study the convergence properties under small perturbations.

Exercise 1.4 Consider the linear iterative solver `GBIT` described in Section 1.4.4. In the notation introduced there, let the iterative error be written as $e_i = y - y_i$.

a) Verify the recursive relation

$$e_{i+1} = (1 - t_i)e_i + t_i\overline{E}_i e_i\,,$$

where

$$\overline{E}_i = -(I - E_i)^{-1}E_i\,.$$

b) Show that, under the assumption $\|\overline{E}_i\| < 1$ on a 'sufficiently good' preconditioner, any stepsize choice $0 < t_i \leq 1$ will lead to convergence, i.e.,

$$\|e_{i+1}\| < \|e_i\|, \quad \text{if } e_i \neq 0\,.$$

c) Verify that $\|E_{i+1}\|_2 \leq \|E_i\|_2$ holds, so that $\|E_0\|_2 < \frac{1}{2}$ implies $\|\bar{E}_i\|_2 < 1$ for all indices $i = 0, 1, \ldots$.

d) Compare the algorithm `GBIT` for the two steplength strategies $t_i = \tau_i$ and $t_i = \min(1, \tau_i)$ at your favorite linear system with nonsymmetric matrix A.