Acta Numerica (2010), pp. 287–449
doi:10.1017/S096249291000005X

Verification methods: Rigorous results using floating-point arithmetic

Siegfried M. Rump
Institute for Reliable Computing,
Hamburg University of Technology,
Schwarzenbergstraße 95, 21071 Hamburg, Germany
and
Visiting Professor at Waseda University,
Faculty of Science and Engineering,
3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan
E-mail: rump@tu-harburg.de

A classical mathematical proof is constructed using pencil and paper. However, there are many ways in which computers may be used in a mathematical proof. But 'proof by computer', or even the use of computers in the course of a proof, is not so readily accepted (the December 2008 issue of the *Notices of the American Mathematical Society* is devoted to formal proofs by computer).

In the following we introduce verification methods and discuss how they can assist in achieving a mathematically rigorous result. In particular we emphasize how floating-point arithmetic is used.

The goal of verification methods is ambitious. For a given problem it is proved, with the aid of a computer, that there exists a (unique) solution of a problem within computed bounds. The methods are constructive, and the results are rigorous in every respect. Verification methods apply to data with tolerances as well, in which case the assertions are true for all data within the tolerances.

Non-trivial problems have been solved using verification methods. For example, Tucker (1999) received the 2004 EMS prize awarded by the European Mathematical Society for giving 'a rigorous proof that the Lorenz attractor exists for the parameter values provided by Lorenz. This was a long-standing challenge to the dynamical system community, and was included by Smale in his list of problems for the new millennium. The proof uses computer estimates with rigorous bounds based on higher dimensional interval arithmetics.' Also, Sahinidis and Tawaralani (2005) received the 2006 Beale–Orchard–Hays Prize for their package BARON, which 'incorporates techniques from automatic differentiation, interval arithmetic, and other areas to yield an automatic, modular, and relatively efficient solver for the very difficult area of global optimization.'

This review article is devoted to verification methods and consists of three parts of similar length. In Part 1 the working tools of verification methods are discussed, in particular floating-point and interval arithmetic; my findings in Section 1.5 (Historical remarks) seem new, even to experts in the field.

In Part 2, the development and limits of verification methods for finite-dimensional problems are discussed in some detail. In particular, we discuss how verification is *not* working. For example, we give a probabilistic argument that the so-called interval Gaussian elimination (IGA) does not work even for (well-conditioned) random matrices of small size. Verification methods are discussed for problems such as dense systems of linear equations, sparse linear systems, systems of nonlinear equations, semi-definite programming and other special linear and nonlinear problems, including M-matrices, finding simple and multiple roots of polynomials, bounds for simple and multiple eigenvalues or clusters, and quadrature. The necessary automatic differentiation tools to compute the range of gradients, Hessians, Taylor coefficients, and slopes are also introduced.

Concerning the important area of optimization, Neumaier (2004) gave in his *Acta Numerica* article an overview on global optimization and constraint satisfaction methods. In view of the thorough treatment there, showing the essential role of interval methods in this area, we restrict our discussion to a few recent, complementary issues.

Finally, in Part 3, verification methods for infinite-dimensional problems are presented, namely two-point boundary value problems and semilinear elliptic boundary value problems.

Throughout the article, many examples of the *inappropriate* use of interval operations are given. In the past such examples contributed to the dubious reputation of interval arithmetic (see Section 1.3), whereas they are, in fact, simply a misuse.

One main goal of this review article is to introduce the principles of the design of verification algorithms, and how these principles differ from those for traditional numerical algorithms (see Section 1.4).

Many algorithms are presented in executable MATLAB/INTLAB code, providing the opportunity to test the methods directly. INTLAB, the MATLAB toolbox for reliable computing, was, for example, used by Bornemann, Laurie, Wagon and Waldvogel (2004) in the solution of half of the problems of the SIAM 10×10-digit challenge by Trefethen (2002).

CONTENTS

PART ONE
Fundamentals

1. Introduction

It is not uncommon for parts of mathematical proofs to be carried out with the aid of computers. For example,

- the non-existence of finite projective planes of order 10 by Lam, Thiel and Swiercz (1989),
- the existence of Lyons' simple group (of order $2^8 \cdot 3^7 \cdot 5^6 \cdot 7 \cdot 11 \cdot 31 \cdot 37 \cdot 67$) by Gorenstein, Lyons and Solomon (1994),
- the uniqueness of Thompson's group and the existence of O'Nan's group by Aschbacher (1994)

were proved with substantial aid of digital computers. Those proofs have in common that they are based on integer calculations: no 'rounding errors' are involved. On the other hand,

- the proof of universality for area-preserving maps (Feigenbaum's constant) by Eckmann, Koch and Wittwer (1984),
- the verification of chaos in discrete dynamical systems by Neumaier and Rage (1993),
- the proof of the double-bubble conjecture by Hass, Hutchings and Schlafly (1995),
- the verification of chaos in the Lorenz equations by Galias and Zgliczynski (1998),
- the proof of the existence of eigenvalues below the essential spectrum of the Sturm–Liouville problem by Brown, McCormack and Zettl (2000)

made substantial use of proof techniques based on floating-point arithmetic (for other examples see Frommer (2001)). We do not want to philosophize to what extent such proofs are rigorous, a theme that even made it into *The New York Times* (Browne 1988). Assuming a computer is *working according to its specifications*, the aim of this article is rather to present methods providing rigorous results which, in particular, use floating-point arithmetic.

We mention that there are possibilities for performing an entire mathematical proof by computer (where the ingenuity is often with the programmer). There are many projects in this direction, for example proof assistants like *Coq* (Bertot and Castéran 2004), theorem proving programs such as *HOL* (Gordon 2000), combining theorem provers and interval arithmetic (Daumas, Melquiond and Muñoz 2005, Hölzl 2009), or the ambitious project FMathL by Neumaier (2009), which aims to formalize mathematics in a very general way.

A number of proofs for non-trivial mathematical theorems have been carried out in this way, among them the Fundamental Theorem of Algebra, the impossibility of trisecting a 60° angle, the prime number theorem, and Brouwer's Fixed-Point Theorem.

Other proofs are routinely performed by computers, for example, integration in finite terms: for a given function using basic arithmetic operations and elementary standard functions, Risch (1969) gave an algorithm for deciding whether such an integral exists (and eventually computing it). This algorithm is implemented, for example, in Maple (2009) or Mathematica (2009) and solves any such problem in finite time. In particular the proof of non-existence of an integral in closed form seems appealing.

1.1. Principles of verification methods

The methods described in this article, which are called *verification methods* (or *self-validating methods*), are of quite a different nature. It will be discussed how *floating-point arithmetic* can be used in a rigorous way. Typical problems to be solved include the following.

- Is a given matrix non-singular? (See Sections 1.6, 10.5, 10.8.)
- Compute error bounds for the minimum value of a function $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$. (See Sections 8, 11.6.)
- Compute error bounds for a solution of a system of nonlinear equations $f(x) = 0$. (See Section 13.)
- Compute error bounds for the solution of an ordinary or partial differential equation. (See Sections 15, 16.)

Most verification methods rely on a good initial approximation. Although the problems (and their solutions) are of different nature, they are solved by the following common

> Design principle of verification methods:
> Mathematical theorems are formulated whose assumptions are verified with the aid of a computer. (1.1)

A verification method is the interplay between mathematical theory and practical application: a major task is to derive the theorems and their assumptions in such a way that the verification is likely to succeed. Mostly those theorems are sufficient criteria: if the assumptions are satisfied, the assertions are true; if not, nothing can be said.[1] The verification of the assumptions is based on *estimates* using floating-point arithmetic. Following Hadamard, a problem is said to be well-posed if it has a unique solution which depends continuously on the input data. A verification method *solves*

[1] In contrast, methods in computer algebra (such as Risch's algorithm) are *never-failing*: the correct answer will be computed in finite time, and the maximally necessary computing time is estimated depending on the input.

a problem by *proving* existence and (possibly local) uniqueness of the solution. Therefore, the inevitable presence of rounding errors implies the

Solvability principle of verification methods:
Verification methods solve well-posed problems. (1.2)

As a typical example, there are efficient verification methods to prove the non-singularity of a matrix (see Section 1.6); however, the proof of singularity is outside the scope of verification methods because in every open neighbourhood of a matrix there is a non-singular matrix.

There are partial exceptions to this principle, for example for integer input data or, in semidefinite programming, if not both the primal and dual problem have distance zero to infeasibility, see Section 14.2. After regularizing an ill-posed problem, the resulting well-posed problem can be treated by verification algorithms.

There will be many examples of these principles throughout the paper. The ambitious goal of verification methods is to produce *rigorous error bounds*, correct in a mathematical sense – taking into account all possible sources of errors, in particular rounding errors.

Furthermore, the goal is to derive verification algorithms which are, for certain classes of problems, not much slower than the best numerical algorithms, say by at most an order of magnitude. Note that comparing computing times is not really fair because the two types of algorithms deliver results of different nature.

Part 1 of this review article, Sections 1 to 9, introduces tools for verification methods. From a mathematical point of view, much of this is rather trivial. However, we need this bridge between mathematics and computer implementation to derive successful verification algorithms.

Given that numerical algorithms are, in general, very reliable, one may ask whether it is necessary to compute verified error bounds for numerical problems. We may cite William (Vel) Kahan, who said that 'numerical errors are rare, rare enough not to worry about all the time, yet not rare enough to ignore them.' Moreover, problems are not restricted to numerical problems: see the short list at the beginning.

Besides this I think it is at the core of mathematics to produce true results. Nobody would take it seriously that Goldbach's conjecture is likely to be true because in trillions of tests no counter-example was found.

Verification methods are to be sharply distinguished from approaches *increasing* reliability. For example, powerful stochastic approaches have been developed by La Porte and Vignes (1974), Vignes (1978, 1980), Stewart (1990) and Chatelin (1988). Further, Demmel *et al.* (2004) have proposed a very well-written linear system solver with improved iterative refinement, which proves reliable in millions of examples. However, none of these approaches claims to produce always true results.

As another example, I personally believe that today's standard function libraries produce floating-point approximations accurate to at least the second-to-last bit. Nevertheless, they cannot be used 'as is' in verification methods because there is no *proof* of that property.[2] In contrast, basic floating-point operations $+, -, \cdot, /, \sqrt{\cdot}$, according to IEEE 754, are *defined precisely*, and are accurate to the last bit.[3] Therefore verification methods willingly use floating-point arithmetic, not least because of its tremendous speed.

1.2. Well-known pitfalls

We need designated tools for a mathematically rigorous verification. For example, it is well known that a small residual of some approximate solution is not sufficient to prove that the true solution is anywhere near it. Similarly, a solution of a discretized problem need not be near a solution of the continuous problem.

Consider, for example, Emden's equation $-\Delta u = u^2$ with Dirichlet boundary conditions on a rectangle with side lengths f and $1/f$, which models an actively heated metal band. It is theoretically known, by the famous result by Gidas, Ni and Nirenberg (1979) on symmetries of positive solutions to semilinear second-order elliptic boundary value problems, that there is a unique non-trivial centro-symmetric solution. However, the discretized equation, dividing the edges into 64 and 32 intervals for $f = 4$, has the solution shown in Figure 1.1. The height of the peak is normed to 4 in the figure; the true height is about 278. The norm of the residual divided by the norm of the solution is about $4 \cdot 10^{-12}$.

Note this is a *true* solution of the discretized equation computed by a verification method described in Section 13. Within the computed (narrow) bounds, this solution of the nonlinear system is unique. The nonlinear system has other solutions, among them an approximation to the solution of the exact equation.

The computed true solution in Figure 1.1 of the discretized equation is far from symmetric, so according to the theoretical result it cannot be near the solution of the continuous problem. Methods for computing rigorous inclusions of infinite-dimensional problems will be discussed in Sections 15 and 16.

It is also well known that if a computation yields similar approximations in various precisions, this approximation need not be anywhere near the true

[2] In Section 7 we briefly discuss how to take advantage of the fast floating-point standard functions.

[3] Admittedly assuming that the actual implementation follows the specification, a principal question we will address again.

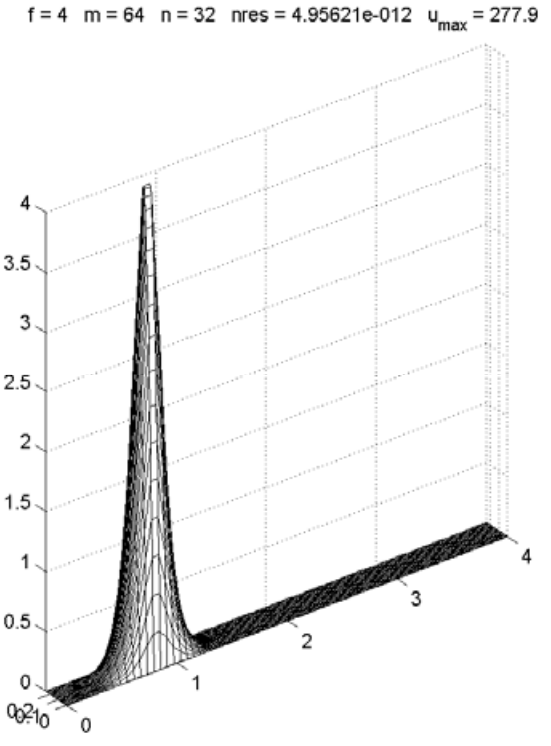

Figure 1.1. True solution of discretized, but spurious solution of the continuous Emden equation $-\Delta u = u^2$.

solution. To show this I constructed the arithmetic expression (Rump 1994)

$$f = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}, \tag{1.3}$$

with $a = 77617$ and $b = 33096$, in the mid-1980s for the arithmetic on IBM S/370 mainframes. In single, double and extended precision[4] corresponding to about 8, 17 and 34 decimal digits, respectively, the results are

$$\begin{array}{ll} \text{single precision} & f \approx 1.172603\cdots \\ \text{double precision} & f \approx 1.1726039400531\cdots \\ \text{extended precision} & f \approx 1.172603940053178\cdots, \end{array} \tag{1.4}$$

whereas the true value is $f = -0.827386\cdots = a/2b - 2$.

The true sum of the main part in (1.3) (everything except the last fraction) is -2 and subject to heavy cancellation. Accidentally, in all precisions the floating-point sum of the main term cancels to 0, so the computed result is just $a/2b$. Further analysis has been given by Cuyt, Verdonk, Becuwe and Kuterna (2001) and Loh and Walster (2002).

[4] Multiplications are carried out to avoid problems with exponential and logarithm.

Almost all of today's architectures follow the IEEE 754 arithmetic standard. For those the arithmetic expression

$$f = 21 \cdot b \cdot b - 2 \cdot a \cdot a + 55 \cdot b \cdot b \cdot b \cdot b - 10 \cdot a \cdot a \cdot b \cdot b + \frac{a}{2b} \tag{1.5}$$

with $a = 77617$ and $b = 33096$ yields the same (wrong) results as in (1.4) when computing in single, double and extended precision (corresponding to about 7, 16 and 19 decimal digits precision, respectively). The reason is the same as for (1.3), and the true value is again

$$f = -0.827386\cdots = \frac{a}{2b} - 2.$$

1.3. The dubious reputation of interval arithmetic

One of the tools we use is interval arithmetic, which bears a dubious reputation. Some exponents in the interval community contributed to a history of overselling intervals as a panacea for problems in scientific computing. In fact, like almost every tool, interval arithmetic is no panacea: if used in a way it should not be used, the results may be useless. And there has been a backlash: the result is that interval techniques have been under-utilized.

It is quite natural that claims were criticized, and the criticism was justified. However, only the critique of the claims and of inappropriate use of interval arithmetic is appropriate; the extension of the criticism to interval arithmetic as a whole is understandable, but overstated.

One of the aims of this article is to show that by using interval arithmetic *appropriately*, certain non-trivial problems can be solved (see the abstract); for example, in Bornemann *et al.* (2004), half of the problems of the SIAM 10×10-digit challenge by Trefethen (2002) were (also) solved using INTLAB.

Consider the following well-known fact:

$$\text{The (real or complex) roots of } x^2 + px + q = 0 \text{ are } x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}. \tag{1.6}$$

It is also well known that, although mathematically correct, this innocent formula may produce poor approximations if not used or programmed appropriately. For example, for $p = 10^8$ and $q = 1$, the MATLAB statements

```
>> p=1e8; q=1; x1tilde=-p/2+sqrt(p^2/4-q)
```

produce

```
x1tilde =
 -7.4506e-009
```

The two summands $-p/2$ and $+\sqrt{p^2/4 - q}$ are almost equal in magnitude, so although using double-precision floating-point arithmetic corresponding to 16 decimal figures of precision, the cancellation leaves no correct digits. A naive user may trust the result because it comes without warning.

All this is well known. It is also well known that one should compute the root of smaller magnitude by $q/(-p/2-\sqrt{p^2/4-q})$ (for positive p). Indeed

```
>> p=1e8; q=1; x1=q/(-p/2-sqrt(p^2/4-q))
```
(1.7)

produces correctly

```
x1 =
 -1.0000e-008
```

We want to stress explicitly that we think that, in this example, interval arithmetic is by no means better than floating-point arithmetic: an inclusion of x_1 computed by (1.6) yields the wide interval $[-1.491, -0.745]\cdot 10^{-8}$. This would be a typical example of inappropriate use of interval arithmetic.[5] But (1.7) yields a good, though not optimal inclusion for positive p.

We also want to stress that we are not arguing against the use of floating-point arithmetic or even trying to imply that floating-point calculations are not trustworthy per se. On the contrary, every tool should be used appropriately. In fact, verification methods depend on floating-point arithmetic to derive rigorous results, and we will use its speed to solve larger problems.

Because it is easy to use interval arithmetic inappropriately, we find it necessary to provide the reader with an easy way to check the claimed properties and results. Throughout the article we give many examples using INTLAB, developed and written by Rump (1999*a*), the MATLAB (2004) toolbox for reliable computing. INTLAB can be downloaded freely for academic purposes, and it is entirely written in MATLAB. For an introduction, see Hargreaves (2002). Our examples are given in executable INTLAB code; we use Version 6.0.

Recently a book by Moore, Kearfott and Cloud (2009) on verification methods using INTLAB appeared; Rohn (2009*b*) gives a large collection of verification algorithms written in MATLAB/INTLAB.

1.4. Numerical methods versus verification methods

A main purpose of this article is to describe how verification methods work and, in particular, how they are designed. There is an essential difference from numerical methods. Derivations and estimates valid over the field of real numbers frequently carry over, in some way, to approximate methods using floating-point numbers.

Sometimes care is necessary, as for the *pq*-formula (1.6) or when solving least-squares problems by normal equations; but usually one may concentrate on the real analysis, in both senses of the term. As a rule of thumb,

[5] One might think that this wide interval provides information on the sensitivity of the problem. This conclusion is not correct because the wide intervals may as well be attributed to some overestimation by interval arithmetic (see Sections 5, 6, 8).

straight replacement of real operations by floating-point operations is not unlikely to work.

For verification methods it is almost the other way around, and many examples of this will be given in this article. Well-known terms such as 'convergence' or even 'distributivity' have to be revisited. As a rule of thumb, straight replacement of real operations by interval operations is likely *not* to work.

We also want to stress that interval arithmetic is a convenient *tool* for implementing verification methods, but is by no means mandatory. In fact, in Sections 3 or 10.8.1 fast and rigorous methods will be described using solely floating-point arithmetic (with rounding to nearest mode). However, a rigorous analysis is sometimes laborious, whereas interval methods offer convenient and simple solutions; see, for example, the careful analysis by Viswanath (1999) to bound his constant concerned with random Fibonacci sequences, and the elegant proof by Oliveira and de Figueiredo (2002) using interval operations. In Section 9.2 we mention possibilities other than traditional intervals for computing with sets.

1.5. Historical remarks

Historically, the development of verification methods has divided into three major steps.

First, interval operations were defined in a number of papers such as Young (1931), Dwyer (1951) and Warmus (1956), but without proper rounding of endpoints and without any applications.

A second, major step was to *solve problems* using interval arithmetic. In an outstanding paper, his Master's Thesis at the University of Tokyo, Sunaga (1956) introduced:

- the interval lattice, the law of subdistributivity, differentiation, gradients, the view of intervals as a topological group, *etc.*,
- infimum–supremum and midpoint-radius arithmetic theoretically and with outward rounded bounds, real and complex including multi-dimensional intervals,
- the inclusion property (5.16), the inclusion principle (5.17) and the subset property,
- the centred form (11.10) as in Section 11.6, and subdivision to improve range estimation,
- the interval Newton procedure (Algorithm 6.1),
- verified interpolation for accurate evaluation of standard functions,
- fast implementation of computer arithmetic by redundant number representation, *e.g.*, addition in two cycles (rediscovered by Avizienis (1961)),
- inclusion of definite integrals using Simpson's rule as in Section 12, and
- the solution of ODEs with stepwise refinement.

His thesis is (handwritten) in Japanese and difficult to obtain. Although Sunaga (1958) summarized some of his results in English (see also Markov and Okumura (1999)), the publication was in an obscure journal and his findings received little attention.

Interval arithmetic became popular through the PhD thesis by Moore (1962), which is the basis for his book (Moore 1966). Overestimation by interval operations was significantly reduced by preconditioning proposed by Hansen and Smith (1967).

Sunaga finished his thesis on February 29, 1956. As Moore (1999) states that he conceived of interval arithmetic and some of its ramifications in the spring of 1958, the priority goes to Sunaga.

Until the mid-1970s, however, either known error estimates (such as for Simpson's rule) were computed with rigour, or assumed inclusions were refined, such as by Krawczyk (1969*a*). However, much non-trivial mathematics was developed: see Alefeld and Herzberger (1974).

The existence tests proposed by Moore (1977) commence the third and major step from interval to verification methods. Now fixed-point theorems such as Brouwer's in finite dimensions or Schauder's in infinite dimensions are used to certify that a solution to a problem exists within given bounds. For the construction of these bounds an iterative scheme was introduced in Rump (1980), together with the idea to include not the solution itself but the error with respect to an approximate solution (see Section 10.5 for details). These techniques are today standard in all kinds of verification methods.

An excellent textbook on verification methods is Neumaier (1990). Moreover, the introduction to numerical analysis by Neumaier (2001) is very much in the spirit of this review article: along with the traditional material the tools for rigorous computations and verification methods are developed. Based on INTLAB, some alternative verification algorithms for linear problems are described in Rohn (2005).

Besides INTLAB, which is free for academic use, ACRITH (1984) and ARITHMOS (1986) are commercial libraries for algorithms with result verification. Both implement the algorithms in my habilitation thesis (Rump 1983).

1.6. A first simple example of a verification method

A very simple first example illustrates the DESIGN PRINCIPLE (1.1) and the SOLVABILITY PRINCIPLE (1.2), and some of the reasoning behind a verification method.

Theorem 1.1. Let matrices $A, R \in \mathbb{R}^{n\times n}$ be given, and denote by I the $n \times n$ identity matrix. If the spectral radius $\varrho(I - RA)$ of $I - RA$ is less than 1, then A is non-singular.

Proof. If A is singular, then $I - RA$ has an eigenvalue 1, a contradiction. □

If the assumption $\varrho(I - RA) < 1$ is satisfied, then it is satisfied for all matrices in a small neighbourhood of A. This corresponds to the Solvability principle (1.2). Note that verification of singularity of a matrix is an ill-posed problem: an arbitrarily small change of the input data may change the answer.

Theorem 1.1 is formulated in such way that its assumption $\varrho(I-RA) < 1$ is likely to be satisfied for an approximate inverse R of A calculated in floating-point arithmetic. This is the interplay between the mathematical theorem and the practical application.

Note that in principle the matrix R is arbitrary, so neither the 'quality' of R as an approximate inverse nor its non-singularity need to be checked. The only assumption to be verified is that an upper bound of $\varrho(I - RA)$ is less than one. If, for example, $\|I - RA\|_\infty < 1$ is rigorously verified, then Theorem 1.1 applies. One way to achieve this is to estimate the individual error of each floating-point operation; this will be described in the next section.

Also note that all numerical experience should be used to design the mathematical theorem, the assumptions to be verified, and the way to compute R. In our particular example it is important to calculate R as a 'left inverse' of A: see Chapter 13 in Higham (2002) (for the drastic difference of the residuals $\|I - RA\|$ and $\|I - AR\|$ see the picture on the front cover of his book). For the left inverse it is proved in numerical analysis that, even for ill-conditioned matrices, it is likely that $\varrho(I - RA) < 1$ is satisfied.

Given that this has been done with care one may ask: What is the validity and the value of such a proof? Undoubtedly the computational part lacks beauty, surely no candidate for 'the Book'. But is it rigorous?

Furthermore, a proof assisted by a computer involves many components such as the programming, a compiler, the operating system, the processor itself, and more. A purely mathematical proof also relies on trust, however, but at least every step *can* be checked.

The trust in the correctness of a proof assisted by a computer can be increased by extensive testing. Verification algorithms allow a kind of testing which is hardly possible for numerical algorithms. Suppose a problem is constructed in such a way that the true solution is π. An approximate solution $p = 3.14159265358978$ would hardly give any reason to doubt, but the wrong 'inclusion' $[3.14159265358978, 3.14159265358979]$ would reveal an error. In INTLAB we obtain a correct inclusion of π by

```
>> P = 4*atan(intval(1))
intval P =
[   3.14159265358979,   3.14159265358980]
```

or simply by `P = intval('pi')`.

1 000 000	
− 999 999.9	3
0.1	

Figure 1.2. Erroneous result on pocket calculators.

The deep mistrust of 'computer arithmetic' as a whole is nourished by examples such as the following. For decades, floating-point arithmetic on digital computers was not precisely defined. Even worse, until very recent times the largest computers fell into the same trap as practically all cheap pocket calculators[6] still do today: calculating 1 000 000–999 999.93 results in 0.1 rather than 0.07 on an 8-decimal-digit calculator. This is because both summands are moved into the 8-digit accumulator, so that the last digit 3 of the second input disappears.

The reason is that the internal accumulator has no extra digit, and the effect is catastrophic: the relative error of a single operation is up to 100%. For other examples and details see, for example, the classical paper by Goldberg (1991).

It is a trivial statement that the floating-point arithmetic we are dealing with has to be precisely defined for any rigorous conclusion. Unfortunately, this was not the case until 1985. Fortunately this is now the case with the IEEE 754 arithmetic standard.

The floating-point arithmetic of the vast majority of existing computers follows this standard, so that the result of every (single) floating-point operation is *precisely defined*. Note that for composite operations, such as dot products, intermediate results are sometimes accumulated in some extended precision, so that floating-point approximations may differ on different computers. However, the error estimates to be introduced in the next section remain valid. This allows mathematically rigorous conclusions, as will be shown in the following.

2. Floating-point arithmetic

A floating-point operation approximates the given real or complex operations. For simplicity we restrict the following discussion to the reals. An excellent and extensive treatment of various aspects of floating-point arithmetic is Muller *et al.* (2009). For another, very readable discussion see Overton (2001).

Let a finite subset $\mathbb{F} \subseteq \mathbb{R} \cup \{-\infty, +\infty\}$ be given, where $\infty \in \mathbb{F}$ and $-\mathbb{F} = \mathbb{F}$. We call the elements of $\mathbb{F}$ *floating-point numbers*. Moreover, set $\texttt{realmax} := \max\{|f| : f \in \mathbb{F} \cap \mathbb{R}\}$.

[6] 8 or 10 or 12 decimal digits without exponent.

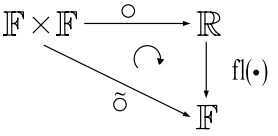

Figure 2.1. Definition of floating-point arithmetic through rounding.

For $a, b \in \mathbb{F}$, a floating-point operation $a \,\tilde{\circ}\, b : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ with $\circ \in \{+, -, \cdot, /\}$ should approximate the real result $a \circ b$. Using a rounding $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$, a natural way to define floating-point operations is

$$a \,\tilde{\circ}\, b := \mathrm{fl}(a \circ b) \quad \text{for all} \quad a, b \in \mathbb{F}. \tag{2.1}$$

In other words, the diagram in Figure 2.1 commutes. Note that

$$a \,\tilde{\circ}\, b = a \circ b \quad \text{whenever} \quad a \circ b \in \mathbb{F}.$$

There are several ways to define a rounding $\mathrm{fl}(\cdot)$. One natural way is *rounding to nearest*, satisfying

$$|\mathrm{fl}(x) - x| = \min\{|f - x| : f \in \mathbb{F}\} \quad \text{for all } x \in \mathbb{R} \text{ with } |x| \le \texttt{realmax}. \tag{2.2}$$

Real numbers larger than `realmax` in absolute value require some special attention. Besides this, such a rounding is optimal in terms of approximating a real number by a floating-point number. Note that the only freedom is the result of $\mathrm{fl}(x)$ for x being the midpoint between two adjacent floating-point numbers. This ambiguity is fixed by choosing the floating-point number with zero last bit in the mantissa.[7]

This defines uniquely the result of all floating-point operations in rounding to nearest mode, and (2.1) together with (2.2) is exactly the definition of floating-point arithmetic in the IEEE 754 standard in rounding to nearest mode. For convenience we denote the result of a floating-point operation by $\mathrm{fl}(a \circ b)$. In terms of minimal relative error the definition is best possible.

In the following we use only IEEE 754 double-precision format, which corresponds to a relative rounding error unit of $\mathtt{u} := 2^{-53}$. It follows for operations $\circ \in \{+, -, \cdot, /\}$ that

$$a \circ b = \mathrm{fl}(a \circ b) \cdot (1 + \varepsilon) + \delta \quad \text{with} \quad |\varepsilon| \le \mathtt{u}, |\delta| \le \frac{1}{2}\eta, \tag{2.3}$$

where $\eta := 2^{-1074}$ denotes the underflow unit. Furthermore, we always have $\varepsilon\delta = 0$ and, since $\mathbb{F} = -\mathbb{F}$, taking an absolute value causes no rounding error.

For addition and subtraction the estimates are particularly simple, because the result is exact if underflow occurs:

$$a \circ b = \mathrm{fl}(a \circ b) \cdot (1 + \varepsilon) \quad \text{with} \quad |\varepsilon| \le \mathtt{u} \quad \text{for} \quad \circ \in \{+, -\}. \tag{2.4}$$

[7] Called 'rounding tie to even'.

Such inequalities can be used to draw rigorous conclusions, for example, to verify the assumption of Theorem 1.1.[8]

In order to verify $\|I - RA\|_\infty < 1$ for matrices $R, A \in \mathbb{F}^{n\times n}$, (2.3) can be applied successively. Such estimates, however, are quite tedious, in particular if underflow is taken into account. Much simplification is possible, as described in Higham (2002), (3.2) and Lemma 8.2, if underflow is neglected, as we now see.

Theorem 2.1. Let $x, y \in \mathbb{F}^n$ and $c \in \mathbb{F}$ be given, and denote by $\mathrm{fl}(\sum x_i)$ and $\mathrm{fl}(c - x^T y)$ the floating-point evaluation of $\sum_{i=1}^n x_i$ and $c - x^T y$, respectively, in any order. Then

$$\left|\mathrm{fl}\Big(\sum x_i\Big) - \sum x_i\right| \le \gamma_{n-1} \sum |x_i|, \tag{2.5}$$

where $\gamma_n := \frac{n\mathbf{u}}{1-n\mathbf{u}}$. Furthermore, provided no underflow has occurred,

$$|\mathrm{fl}(c - x^T y) - (c - x^T y)| \le \gamma_n |x|^T |y|, \tag{2.6}$$

where the absolute value is taken entrywise.

To obtain a computable bound for the right-hand side of (2.6), we abbreviate $e := |x|^T|y|$, $\tilde{e} := \mathrm{fl}(|x|^T|y|)$ and use (2.6) to obtain

$$|e| \le |\tilde{e}| + |\tilde{e} - e| \le |\tilde{e}| + \gamma_n |e|, \tag{2.7}$$

and therefore

$$|\mathrm{fl}(c - x^T y) - (c - x^T y)| \le \gamma_n |e| \le \frac{\gamma_n}{1-\gamma_n} |\tilde{e}| = \frac{1}{2}\gamma_{2n} \cdot \mathrm{fl}(|x|^T|y|). \tag{2.8}$$

This is true provided no over- or underflow occurs. To obtain a computable bound, the error in the floating-point evaluation of γ_n has to be estimated as well. Denote

$$\tilde{\gamma}_n := \mathrm{fl}(n \cdot \mathbf{u}/(1 - n \cdot \mathbf{u})).$$

Then for $n\mathbf{u} < 1$ neither $N := \mathrm{fl}(n \cdot \mathbf{u})$ nor $\mathrm{fl}(1 - N)$ causes a rounding error, and (2.3) implies

$$\gamma_n \le \tilde{\gamma}_{n+1}.$$

Therefore, with (2.8), estimating the error in the multiplication by e and observing that division by 2 causes no rounding error, we obtain

$$|\mathrm{fl}(c - x^T y) - (c - x^T y)| \le \mathrm{fl}\left(\frac{1}{2}\tilde{\gamma}_{2n+2} \cdot \tilde{e}\right) \quad \text{with} \quad \tilde{e} = \mathrm{fl}(|x|^T|y|). \tag{2.9}$$

[8] The following sample derivation of floating-point estimates serves a didactic purpose; in Section 5 we show how to avoid (and improve) these tedious estimates.

Applying this to each component of $I - RA$ gives the entrywise inequality

$$|\mathrm{fl}(I - RA) - (I - RA)| \le \mathrm{fl}\left(\left(\frac{1}{2}\tilde{\gamma}_{2n+2}\right) \cdot E\right) =: \beta \quad \text{with} \quad E := \mathrm{fl}(|R||A|). \tag{2.10}$$

Note that the bound is valid for all $R, A \in \mathbb{F}^{n\times n}$ with $(2n+2)\mathtt{u} < 1$ provided that no underflow has occurred. Also note that β is a computable matrix of floating-point entries.

It is clear that one can continue like this, eventually obtaining a computable and rigorous upper bound for $\bar{c} := \|I - RA\|_\infty$. Note that $\bar{c} < 1$ *proves* (R and) A to be non-singular. The proof is rigorous provided that $(2n+2)\mathtt{u} < 1$, that no underflow occurred, and that the processor, compiler, operating system, and all components involved work to their specification.

These rigorous derivations of error estimates for floating-point arithmetic are tedious. Moreover, for each operation the worst-case error is (and has to be) assumed. Both will be improved in the next section.

Consider the model problem with an $n \times n$ matrix A_n based on the following pattern for $n = 3$:

$$A_n = \begin{pmatrix} 1 & \frac{1}{4} & \frac{1}{7} \\ \frac{1}{2} & \frac{1}{5} & \frac{1}{8} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{9} \end{pmatrix} \quad \text{for} \quad n = 3. \tag{2.11}$$

Now the application of Theorem 1.1 is still possible but even more involved since, for example, $\frac{1}{3}$ is not a floating-point number. Fortunately there is an elegant way to obtain rigorous and sharper error bounds using floating-point arithmetic, even for this model problem with input data not in $\mathbb{F}$.

Before we come to this we discuss a quite different but very interesting method for obtaining not only rigorous but exact results in floating-point arithmetic.

3. Error-free transformations

In the following we consider solely rounding to nearest mode, and we assume that no overflow occurs. As we have seen, the result of every floating-point operation is uniquely defined by (2.1). This not only allows error estimates such as (2.3), but it can be shown that the error of every floating-point operation is itself a floating-point number:[9]

$$x = \mathrm{fl}(a \circ b) \quad \Rightarrow \quad x + y = a \circ b \quad \text{with} \quad y \in \mathbb{F} \tag{3.1}$$

for $a, b \in \mathbb{F}$ and $\circ \in \{+, -, \cdot\}$. Remarkably, the error y can be calculated using only basic floating-point operations. The following algorithm by Knuth (1969) does it for addition.

[9] For division, $q, r \in \mathbb{F}$ for $q := \mathrm{fl}(a/b)$ and $a = qb + r$, and for the square root $x, y \in \mathbb{F}$ for $x = \mathrm{fl}(\sqrt{a})$ and $a = x^2 + y$.

Algorithm 3.1. Error-free transformation of the sum of two floating-point numbers:

$$\begin{aligned}&\text{function } [x, y] = \text{TwoSum}(a, b)\\ &\quad x = \text{fl}(a + b)\\ &\quad z = \text{fl}(x - a)\\ &\quad e = \text{fl}(x - z)\\ &\quad f = \text{fl}(b - z)\\ &\quad y = \text{fl}(\text{fl}(a - e) + f)\end{aligned}$$

Theorem 3.2. For any two floating-point numbers $a, b \in \mathbb{F}$, the computed results x, y of Algorithm 3.1 satisfy

$$x = \text{fl}(a + b) \quad \text{and} \quad x + y = a + b. \tag{3.2}$$

Algorithm 3.1 needs six floating-point operations. It was shown by Dekker (1971) that the same can be achieved in three floating-point operations if the input is sorted by absolute value.

Algorithm 3.3. Error-free transformation of the sum of two sorted floating-point numbers:

$$\begin{aligned}&\text{function } [x, y] = \text{FastTwoSum}(a, b)\\ &\quad x = \text{fl}(a + b)\\ &\quad y = \text{fl}(\text{fl}(a - x) + b)\end{aligned}$$

Theorem 3.4. The computed results x, y of Algorithm 3.3 satisfy (3.2) for any two floating-point numbers $a, b \in \mathbb{F}$ with $|a| \geq |b|$.

One may prefer Dekker's Algorithm 3.3 with a branch and three operations. However, with today's compiler optimizations we note that Knuth's Algorithm 3.1 with six operations is often faster: see Section 9.1.

Error-free transformations are a very powerful tool. As Algorithms 3.1 and 3.3 transform a pair of floating-point numbers into another pair, Algorithm 3.5 transforms a vector of floating-point numbers into another vector without changing the sum.

Algorithm 3.5. Error-free vector transformation for summation:

$$\begin{aligned}&\text{function } q = \text{VecSum}(p)\\ &\quad \pi_1 = p_1\\ &\quad \text{for } i = 2 : n\\ &\quad\quad [\pi_i, q_{i-1}] = \text{TwoSum}(p_i, \pi_{i-1})\\ &\quad \text{end for}\\ &\quad q_n = \pi_n\end{aligned}$$

As in Algorithm 3.1, the result vector q splits into an approximation q_n of $\sum p_i$ and into error terms $q_{1\ldots n-1}$ without changing the sum.

We use the convention that for floating-point numbers p_i, $\mathrm{fl}_\uparrow\big(\sum p_i\big)$ denotes their recursive floating-point sum starting with p_1.

Theorem 3.6. For a vector $p \in \mathbb{F}^n$ of floating-point numbers let $q \in \mathbb{F}^n$ be the result computed by Algorithm 3.5. Then

$$\sum_{i=1}^{n} q_i = \sum_{i=1}^{n} p_i. \tag{3.3}$$

Moreover,

$$q_n = \mathrm{fl}_\uparrow\left(\sum_{i=1}^{n} p_i\right) \quad \text{and} \quad \sum_{i=1}^{n-1} |q_i| \le \gamma_{n-1} \sum_{i=1}^{n} |p_i|, \tag{3.4}$$

where $\gamma_k := k\mathtt{u}/(1-k\mathtt{u})$ using the relative rounding error unit $\mathtt{u}$.

Proof. By Theorem 3.2 we have $\pi_i + q_{i-1} = p_i + \pi_{i-1}$ for $2 \le i \le n$. Summing up yields

$$\sum_{i=2}^{n} \pi_i + \sum_{i=1}^{n-1} q_i = \sum_{i=2}^{n} p_i + \sum_{i=1}^{n-1} \pi_i \tag{3.5}$$

or

$$\sum_{i=1}^{n} q_i = \pi_n + \sum_{i=1}^{n-1} q_i = \sum_{i=2}^{n} p_i + \pi_1 = \sum_{i=1}^{n} p_i. \tag{3.6}$$

Moreover, $\pi_i = \mathrm{fl}(p_i + \pi_{i-1})$ for $2 \le i \le n$, and $q_n = \pi_n$ proves $q_n = \mathrm{fl}_\uparrow\big(\sum p_i\big)$. Now (2.3) and (2.5) imply

$$|q_{n-1}| \le \mathtt{u}|\pi_n| \le \mathtt{u}(1+\gamma_{n-1}) \sum_{i=1}^{n} |p_i|.$$

It follows by an induction argument that

$$\sum_{i=1}^{n-1} |q_i| \le \gamma_{n-2} \sum_{i=1}^{n-1} |p_i| + \mathtt{u}(1+\gamma_{n-1}) \sum_{i=1}^{n} |p_i| \le \gamma_{n-1} \sum_{i=1}^{n} |p_i|. \tag{3.7}$$

□

This is a rigorous result using only floating-point computations. The vector p is transformed into a new vector of $n-1$ error terms $q_1, \ldots, q_{n-1}$ together with the floating-point approximation q_n of the sum $\sum p_i$.

A result 'as if' computed in quadruple precision can be achieved by adding the error terms in floating-point arithmetic.

Theorem 3.7. For a vector $p \in \mathbb{F}^n$ of floating-point numbers, let $q \in \mathbb{F}^n$ be the result computed by Algorithm 3.5. Define

$$e := \mathrm{fl}\left(\sum_{i=1}^{n-1} q_i\right), \tag{3.8}$$

where the summation can be executed in any order. Then

$$\left| q_n + e - \sum_{i=1}^{n} p_i \right| \le \gamma_{n-2}\gamma_{n-1} \sum_{i=1}^{n} |p_i|. \tag{3.9}$$

Further,

$$|\tilde{s} - s| \le \mathtt{u}|s| + \gamma_n^2 \sum_{i=1}^{n} |p_i|, \tag{3.10}$$

for $\tilde{s} := \mathrm{fl}(q_n + e)$ and $s := \sum_{i=1}^{n} p_i$.

Proof. Using (3.3), (2.5) and (3.4), we find

$$\left| q_n + e - \sum_{i=1}^{n} p_i \right| = \left| e - \sum_{i=1}^{n-1} q_i \right| \le \gamma_{n-2} \sum_{i=1}^{n-1} |q_i| \le \gamma_{n-2}\gamma_{n-1} \sum_{i=1}^{n} |p_i| \tag{3.11}$$

and, for some $|\varepsilon| \le \mathtt{u}$,

$$|\mathrm{fl}(q_n + e) - s| = |\varepsilon s + (q_n + e - s)(1 + \varepsilon)| \le \mathtt{u}|s| + \gamma_n^2 \sum_{i=1}^{n} |p_i| \tag{3.12}$$

by (2.4) and (3.11). □

Putting things together, we arrive at an algorithm using solely double-precision floating-point arithmetic but achieving a result of quadruple-precision quality.

Algorithm 3.8. Approximating some $\sum p_i$ in double-precision arithmetic with quadruple-precision quality:

```
function res = Sums(p)
    π_1 = p_1;  e = 0
    for i = 2 : n
        [π_i, q_{i-1}] = TwoSum(p_i, π_{i-1})
        e = fl(e + q_{i-1})
    end for
    res = fl(π_n + e)
```

This algorithm was given by Neumaier (1974) using FastTwoSum with a branch. At the time he did not know Knuth's or Dekker's error-free transformation, but derived the algorithm in expanded form. Unfortunately, his paper is written in German and did not receive a wide audience.

Note that the pair (π_n, e) can be used as a result representing a kind of simulated quadruple-precision number. This technique is used today in the XBLAS library (see Li *et al.* (2002)).

Similar techniques are possible for the calculation of dot products. There are error-free transformations for computing $x + y = a \cdot b$, again using only floating-point operations. Hence a dot product can be transformed into a sum without error provided no underflow has occurred.

Once dot products can be computed with quadruple-precision quality, the residual of linear systems can be improved substantially, thus improving the quality of inclusions for the solution. We come to this again in Section 10.4. The precise dot product was, in particular, popularized by Kulisch (1981).

4. Directed roundings

The algebraic properties of $\mathbb{F}$ are very poor. In fact it can be shown under general assumptions that for a floating-point arithmetic neither addition nor multiplication can be associative.

As well as rounding to nearest mode, IEEE 754 allows other rounding modes,[10] $\mathrm{fldown}, \mathrm{flup} : \mathbb{R} \to \mathbb{F}$, namely rounding towards $-\infty$ mode and rounding towards $+\infty$ mode:

$$\begin{aligned} \mathrm{fldown}(a \circ b) &:= \max\{f \in \mathbb{F} : f \le a \circ b\}, \\ \mathrm{flup}(a \circ b) &:= \min\{f \in \mathbb{F} : a \circ b \le f\}. \end{aligned} \tag{4.1}$$

Note that the inequalities are always valid for all operations $\circ \in \{+, -, \cdot, /\}$, including possible over- or underflow, and note that $\pm\infty$ may be a result of a floating-point operation with directed rounding mode. It follows for all $a, b \in \mathbb{F}$ and $\circ \in \{+, -, \cdot, /\}$ that

$$\mathrm{fldown}(a \circ b) = \mathrm{flup}(a \circ b) \quad \Longleftrightarrow \quad a \circ b \in \mathbb{F}, \tag{4.2}$$

a nice mathematical property. On most computers the operations with directed roundings are particularly easy to execute: the processor can be set into a specific rounding mode such as to nearest, towards $-\infty$ or towards $+\infty$, so that *all* subsequent operations are executed in this rounding mode until the next change.

Let $x, y \in \mathbb{F}^n$ be given. For $1 \le i \le n$ we have

$$s_i := \mathrm{fldown}(x_i \cdot y_i) \le x_i \cdot y_i \le \mathrm{flup}(x_i \cdot y_i) =: t_i.$$

Note that $s_i, t_i \in \mathbb{F}$ but $x_i \cdot y_i \in \mathbb{R}$ and, in general, $x_i \cdot y_i \notin \mathbb{F}$. It follows that

$$d_1 := \mathrm{fldown}\Big(\sum s_i\Big) \le x^T y \le \mathrm{flup}\Big(\sum t_i\Big) =: d_2, \tag{4.3}$$

where $\mathrm{fldown}(\sum s_i)$ indicates that all additions are performed with rounding towards $-\infty$ mode. The summations may be executed in any order.

[10] IEEE 754 also defines rounding towards zero. This is the *only* rounding mode available on cell processors, presumably because it is fast to execute and avoids overflow.

The function `setround` in INTLAB performs the switching of the rounding mode, so that for two column vectors $\mathtt{x}, \mathtt{y} \in \mathbb{F}^n$ the MATLAB/INTLAB code

```
setround(-1)              % rounding downwards mode
d1 = x'*y
setround(1)               % rounding upwards mode
d2 = x'*y
```

calculates floating-point numbers $\mathtt{d1}, \mathtt{d2} \in \mathbb{F}$ with $\mathtt{d1} \leq x^T y \leq \mathtt{d2}$. Note that these inequalities are rigorous, including the possibility of over- or underflow. This can be applied directly to verify the assumptions of Theorem 1.1. For given $A \in \mathbb{F}^{n \times n}$, consider the following algorithm.

Algorithm 4.1. Verification of the non-singularity of the matrix `A`:

```
R = inv(A);               % approximate inverse
setround(-1)              % rounding downwards mode
C1 = R*A-eye(n);          % lower bound for RA-I
setround(1)               % rounding upwards mode
C2 = R*A-eye(n);          % upper bound for RA-I
C = max(abs(C1),abs(C2)); % upper bound for |RA-I|
c = norm( C , inf );      % upper bound for ||RA-I||_inf
```

We claim that $\mathtt{c} < 1$ proves that `A` and `R` are non-singular.

Theorem 4.2. Let matrices $A, R \in \mathbb{F}^{n \times n}$ be given, and let `c` be the quantity computed by Algorithm 4.1. If $\mathtt{c} < 1$, then A (and R) are non-singular.

Proof. First note that

$$\mathrm{fldown}(a - b) \leq a - b \leq \mathrm{flup}(a - b) \quad \text{for all} \quad a, b \in \mathbb{F}.$$

Combining this with (4.3) and observing the rounding mode, we obtain

$$\mathtt{C1} \leq RA - I \leq \mathtt{C2}$$

using entrywise inequalities. Taking absolute value and maximum do not cause a rounding error, and observing the rounding mode when computing the ∞-norm together with $\varrho(I - RA) \leq \|I - RA\|_\infty = \| \, |RA - I| \, \|_\infty \leq \|\mathtt{C}\|_\infty$ proves the statement. □

Theorem 4.2 is a very simple first example of a verification method. According to the DESIGN PRINCIPLE OF VERIFICATION METHODS (1.1), $\mathtt{c} < 1$ is verified with the aid of the computer. Note that this proves non-singularity of all matrices within a small neighbourhood $U_\epsilon(A)$ of A.

According to the SOLVABILITY PRINCIPLE OF VERIFICATION METHODS (1.2), it is possible to verify non-singularity of a given matrix, but not singularity. An arbitrarily small perturbation of the input matrix may change the answer from yes to no. The verification of singularity excludes the use of estimates: it is only possible using exact computation.

Note that there is a trap in the computation of `C1` and `C2`: Theorem 4.2 is not valid when replacing $\mathtt{R} * \mathtt{A} - \mathtt{eye(n)}$ by $\mathtt{eye(n)} - \mathtt{R} * \mathtt{A}$ in Algorithm 4.1. In the latter case the multiplication and subtraction must be computed in opposite rounding modes to obtain valid results.

This application is rather simple, avoiding tedious estimates. However, it does not yet solve our model problem (2.11) where the matrix entries are not floating-point numbers. An elegant solution for this is interval arithmetic.

5. Operations with sets

There are many possibilities for defining operations on sets of numbers, most prominently the power set operations. Given two sets $X, Y \subseteq \mathbb{R}$, the operations $\circ : \mathbb{PR} \times \mathbb{PR} \to \mathbb{PR}$ with $\circ \in \{+, -, \cdot, /\}$ are defined by

$$X \circ Y := \{x \circ y : x \in X, y \in Y\}, \tag{5.1}$$

where $0 \notin Y$ is assumed in the case of division. The input sets X, Y may be interpreted as *available information* of some quantity. For example, $\pi \in [3.14, 3.15]$ and $e \in [2.71, 2.72]$, so, with only this information at hand,

$$d := \pi - e \in [0.42, 0.44] \tag{5.2}$$

is true and the best we can say. The operations are optimal, the result is the minimum element in the infimum lattice $\{\mathbb{PR}, \cap, \cup\}$. However, this may no longer be true when it comes to composite operations and if operations are executed one after the other. For example,

$$d + e \in [3.13, 3.16] \tag{5.3}$$

is again true and the best we can say using only the information $d \in [0.42, 0.44]$; using the definition of d reveals $d + e = \pi$.

5.1. Interval arithmetic

General sets are hardly representable. The goal of implementable operations suggests restricting sets to intervals. Ordinary intervals are not the only possibility: see Section 9.2.

Denote[11] the set of intervals $\{[\underline{x}, \overline{x}] : \underline{x}, \overline{x} \in \mathbb{R}, \underline{x} \le \overline{x}\}$ by $\mathbb{IR}$. Then, provided $0 \notin Y$ in the case of division, the result of the power set operation $X \circ Y$ for $X, Y \in \mathbb{IR}$ is again an interval, and we have

$$[\underline{x}, \overline{x}] \circ [\underline{y}, \overline{y}] := [\min(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}),\ \max(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y})]. \tag{5.4}$$

For a practical implementation it becomes clear that in most cases it can be decided *a priori* which pair of the bounds $\underline{x}, \overline{x}, \underline{y}, \overline{y}$ lead to the lower and

[11] A standardized interval notation is proposed by Kearfott *et al.* (2005).

upper bound of the result. In the case of addition and subtraction this is particularly simple, namely

$$\begin{aligned} X + Y &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\ X - Y &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}]. \end{aligned} \tag{5.5}$$

For multiplication and division there are case distinctions for X and Y depending on whether they are entirely positive, entirely negative, or contain zero. In all but one case, namely where both X and Y contain zero for multiplication, the pair of bounds is *a priori* clear. In that remaining case $0 \in X$ and $0 \in Y$ we have

$$[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] := [\min(\underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}),\ \max(\underline{x} \circ \underline{y}, \overline{x} \circ \overline{y})]. \tag{5.6}$$

As before, we assume from now on that a denominator interval does not contain zero. We stress that *all* results remain valid without this assumption; however, various statements become more difficult to formulate without giving substantial new information.

We do not discuss complex interval arithmetic in detail. Frequently complex discs are used, as proposed by Sunaga (1958). They were also used, for example, by Gargantini and Henrici (1972) to enclose roots of polynomials. The implementation is along the lines of real interval arithmetic. It is included in INTLAB.

5.2. Overestimation

A measure for accuracy or overestimation by interval operations is the diameter $d(X) := \overline{x} - \underline{x}$. Obviously,

$$d(X + Y) = d(X) + d(Y); \tag{5.7}$$

the diameter of the sum is the sum of the diameters. However,

$$d(X - Y) = (\overline{x} - \underline{y}) - (\underline{x} - \overline{y}) = d(X) + d(Y), \tag{5.8}$$

so the diameter of the sum and the difference of intervals cannot be smaller than the minimum of the diameters of the operands. In particular,

$$d(X - X) = 2 \cdot d(X). \tag{5.9}$$

This effect is not due to interval arithmetic but occurs in power set operations as well: see (5.2) and (5.3).

This can also be seen when writing an interval X in Gaussian notation $x \pm \Delta x$ as a number with a tolerance:

$$\begin{aligned} (x \pm \Delta x) + (y \pm \Delta y) &= (x + y) \pm (\Delta x + \Delta y), \\ (x \pm \Delta x) - (y \pm \Delta y) &= (x - y) \pm (\Delta x + \Delta y). \end{aligned} \tag{5.10}$$

That means the absolute errors add, implying a large relative error if the

result is small in absolute value. This is exactly the case when catastrophic cancellation occurs. In contrast, neglecting higher-order terms,

$$\begin{aligned}(x \pm \Delta x) \cdot (y \pm \Delta y) &\sim x \cdot y \pm (\Delta x \cdot |y| + |x| \Delta y),\\ \frac{x \pm \Delta x}{y \pm \Delta y} = \frac{(x \pm \Delta x) \cdot (y \mp \Delta y)}{y^2 - \Delta y^2} &\sim \frac{x}{y} \pm \frac{\Delta x \cdot |y| + |x| \Delta y}{y^2},\end{aligned} \tag{5.11}$$

so that

$$\frac{\Delta(x \cdot y)}{|x \cdot y|} = \frac{\Delta(x/y)}{|x/y|} = \frac{\Delta x}{|x|} + \frac{\Delta y}{|y|}. \tag{5.12}$$

This means that for multiplication and division the relative errors add. Similarly, not much overestimation occurs in interval multiplication or division.

Demmel, Dumitriu, Holtz and Koev (2008) discuss in their recent *Acta Numerica* paper how to evaluate expressions to achieve accurate results in linear algebra. One major sufficient (but not necessary) condition is the No inaccurate cancellation principle (NIC). It allows multiplications and divisions, but additions (subtractions) only on data with the same (different) sign, or on input data.

They use this principle to show that certain problems can be solved with high accuracy if the structure is taken into account. A distinctive example is the accurate computation of the smallest singular value of a Hilbert matrix of, say, dimension $n = 100$ (which is about 10^{-150}) by looking at it as a Cauchy matrix.

We see that the No inaccurate cancellation principle (NIC) means precisely that replacement of every operation by the corresponding interval operation produces an accurate result. This is called 'naive interval arithmetic', and it is, in general, bound to fail (see Section 6).

In general, if no structure in the problem is known, the sign of summands cannot be predicted. This leads to the

Utilize input data principle of verification methods:
Avoid re-use of computed data; use input data where possible. (5.13)

Our very first example in Theorem 1.1 follows this principle: the verification of $\varrho(I - RA) < 1$ is based mainly on the input matrix A.

Once again we want to stress that the effect of *data dependency* is not due to interval arithmetic but occurs with power set operations as well. Consider two enclosures $X = [3.14, 3.15]$ and $Y = [3.14, 3.15]$ for π. Then

$$X - Y = [-0.01, +0.01] \quad \text{or} \quad X/Y = [0.996, 1.004]$$

is (rounded to 3 digits) the best we can deduce using the given information; but when adding the information that both X and Y are inclusions for π, the results can be sharpened into 0 and 1, respectively.

5.3. Floating-point bounds

Up to now the discussion has been theoretical. In a practical implementation on a digital computer the bounds of an interval are floating-point numbers. Let $\mathbb{IF} \subset \mathbb{IR}$ denote the set $\{[\underline{x}, \overline{x}] : \underline{x}, \overline{x} \in \mathbb{F}, \underline{x} \leq \overline{x}\}$. We define a rounding $\diamond : \mathbb{IR} \to \mathbb{IF}$ by

$$\diamond([\underline{x}, \overline{x}]) := [\mathrm{fldown}(\underline{x}), \mathrm{flup}(\overline{x})], \tag{5.14}$$

and operations $\diamondsuit : \mathbb{IF} \times \mathbb{IF} \to \mathbb{IF}$ for $X, Y \in \mathbb{IF}$ and $\circ \in \{+, -, \cdot, /\}$ by

$$X \diamondsuit Y := \diamond(X \circ Y). \tag{5.15}$$

Fortunately this definition is straightforward to implement on today's computers using directed roundings, as introduced in Section 4. Basically, definitions (5.4) and (5.6) are used where the lower (upper) bound is computed with the processor switched into rounding downwards (upwards) mode. Note that the result is best possible.

There are ways to speed up a practical implementation of scalar interval operations. Fast C++ libraries for interval operations are PROFIL/BIAS by Knüppel (1994, 1998). Other libraries for interval operations include Intlib (Kearfott, Dawande, Du and Hu 1992) and C-XSC (Klatte *et al.* 1993). The main point for this article is that interval operations with floating-point bounds are rigorously implemented.

For vector and matrix operations a fast implementation is mandatory, as discussed in Section 9.1.

5.4. Infinite bounds

We defined $-\infty$ and $+\infty$ to be floating-point numbers. This is particularly useful for maintaining (5.4) and (5.14) without nasty exceptions in the case of overflow. Also, special operations such as division by a zero interval can be consistently defined, such as by Kahan (1968).

We feel this is not the main focus of interest, and requires too much detail for this review article. We therefore assume from now on that all intervals are finite. Once again, all results (for example, computed by INTLAB) remain rigorous, even in the presence of division by zero or infinite bounds.

5.5. The inclusion property

For readability we will from now on denote interval quantities by bold letters $\mathbf{X}, \mathbf{Y}, \ldots$. Operations between interval quantities are always interval operations, as defined in (5.15). In particular, an expression such as

$$\mathbf{R} = (\mathbf{X} + \mathbf{Y}) - \mathbf{X}$$

is to be understood as

$$\mathbf{Z} = \mathbf{X} + \mathbf{Y} \quad \text{and} \quad \mathbf{R} = \mathbf{Z} - \mathbf{X},$$

where both addition and subtraction are interval operations. If the order of execution is ambiguous, assertions are valid for any order of evaluation.

We will frequently demonstrate examples using INTLAB. Even without familiarity with the concepts of MATLAB it should not be difficult to follow our examples; a little additional information is given where necessary to understand the INTLAB code.

INTLAB uses an operator concept with a new data type `intval`. For $\mathtt{f}, \mathtt{g} \in \mathbb{F}$, the 'type casts' `intval(f)` or `infsup(f,g)` produce intervals $[\mathtt{f}, \mathtt{f}]$ or $[\mathtt{f}, \mathtt{g}]$, respectively. In the latter case $\mathtt{f} \leq \mathtt{g}$ is checked. Operations between interval quantities and floating-point quantities `f` are possible, where the latter are automatically replaced by the interval $[\mathtt{f}, \mathtt{f}]$. Such a quantity is called a *point interval*. With this the above reads

```
R = (X+Y)-X;
```

as executable INTLAB code, where `X` and `Y` are interval quantities.

The operator concept with the natural embedding of $\mathbb{F}$ into $\mathbb{IF}$ implies that an interval operation is applied if at least one of the operands is of type `intval`. Therefore,

```
X1 = 1/intval(3);
X2 = intval(1)/3;
X3 = intval(1)/intval(3);
```

all have the same result, namely $[\mathrm{fldown}(1/3), \mathrm{flup}(1/3)]$. The most important property of interval operations is the

$$\begin{aligned} &\textsc{Inclusion property:}\\ &\text{Given } \mathbf{X}, \mathbf{Y} \in \mathbb{IF},\ \circ \in \{+, -, \cdot, /\} \text{ and any } x, y \in \mathbb{R}\\ &\text{with } \ x \in \mathbf{X}, y \in \mathbf{Y}, \ \text{ it is true that } x \circ y \in \mathbf{X} \circ \mathbf{Y}. \end{aligned} \tag{5.16}$$

For a given arithmetic expression $f(x_1, \ldots, x_n)$, we may replace each operation by its corresponding interval operation. Call that new expression $F(x_1, \ldots, x_n)$ the *natural interval extension* of f. Using the natural embedding $x_i \in \mathbf{X}_i$ with $\mathbf{X}_i := [x_i, x_i] \in \mathbb{IF}$ for $i \in \{1, \ldots, n\}$, it is clear that $f(x_1, \ldots, x_n) \in f(\mathbf{X}_1, \ldots, \mathbf{X}_n)$ for $x_i \in \mathbb{F}$. More generally, applying the Inclusion property (5.16) successively, we have the

$$\begin{aligned} &\textsc{Inclusion principle:}\\ &x_i \in \mathbb{R}, \mathbf{X}_i \in \mathbb{IF} \text{ and } x_i \in \mathbf{X}_i \Longrightarrow f(x_1, \ldots, x_n) \in F(\mathbf{X}_1, \ldots, \mathbf{X}_n). \end{aligned} \tag{5.17}$$

These remarkable properties, the Inclusion property (5.16) and the Inclusion principle (5.17), due to Sunaga (1956, 1958) and rediscovered by Moore (1962), allow the estimation of the range of a function over a given domain in a simple and rigorous way. The result will always be true; however, much overestimation may occur (see Sections 6, 8 and 11.6).

Concerning infinite bounds, (5.17) is also true when using INTLAB; however, it needs some interpretation. An invalid operation such as division by two intervals containing zero leads to the answer `NaN`. This symbol *Not a Number* is the MATLAB representation for invalid operations such as $\infty - \infty$. An interval result `NaN` is to be interpreted as 'no information available'.

6. Naive interval arithmetic and data dependency

One may try to replace each operation in some algorithm by its corresponding interval operation to overcome the rounding error problem. It is a true statement that the true value of each (intermediate) result is included in the corresponding interval (intermediate) result. However, the direct and naive use of the INCLUSION PRINCIPLE (5.17) in this way will almost certainly fail by producing wide and useless bounds.

Consider $f(x) = x^2 - 2$. A Newton iteration $x_{k+1} = x_k - (x_k^2 - 2)/2x_k$ with starting value x_0 not too far from $\sqrt{2}$ will converge rapidly. After at most 5 iterations, any starting value in $[1, 2]$ produces a result with 16 correct figures, for example, in executable MATLAB code:

```
>> x=1; for i=1:5, x = x - (x^2-2)/(2*x), end
x = 1.500000000000000
x = 1.416666666666667
x = 1.414215686274510
x = 1.414213562374690
x = 1.414213562373095
```

Now consider a naive interval iteration starting with $\mathbf{X}_0 := [1.4, 1.5]$ in executable INTLAB code.

Algorithm 6.1. Naive interval Newton procedure:

```
>> X=infsup(1.4,1.5);
>> for i=1:5, X = X - (X^2-2)/(2*X), end
intval X =
[    1.3107,    1.5143]
intval X =
[    1.1989,    1.6219]
intval X =
[    0.9359,    1.8565]
intval X =
[    0.1632,    2.4569]
intval X =
[  -12.2002,    8.5014]
```

Rather than converging, the interval diameters increase, and the results are of no use. This is another typical example of inappropriate use of interval

arithmetic. The reason for the behaviour is data dependency. This is not due to interval arithmetic, but, as has been noted before, the results would be the same when using power set operations. It corresponds to the rules of thumb mentioned in Section 1.4.

Instead of an inclusion of

$$\{x - (x^2 - 2)/(2x) : x \in \mathbf{X}_k\}, \tag{6.1}$$

naive interval arithmetic computes in the kth iteration an inclusion of

$$\{\xi_1 - (\xi_2^2 - 2)/(2\xi_3) : \xi_1, \xi_2, \xi_3 \in \mathbf{X}_k\}. \tag{6.2}$$

To improve this, the Newton iteration is to be redefined in an appropriate way, utilizing the strengths of interval arithmetic and diminishing weaknesses (see Moore (1966)).

Theorem 6.2. Let a differentiable function $f : \mathbb{R} \to \mathbb{R}$, $\mathbf{X} = [x_1, x_2] \in \mathbb{IR}$ and $\tilde{x} \in \mathbf{X}$ be given, and suppose $0 \notin f'(\mathbf{X})$. Using interval operations, define

$$N(\tilde{x}, \mathbf{X}) := \tilde{x} - f(\tilde{x})/f'(\mathbf{X}). \tag{6.3}$$

If $N(\tilde{x}, \mathbf{X}) \subseteq \mathbf{X}$, then $\mathbf{X}$ contains a unique root of f. If $N(\tilde{x}, \mathbf{X}) \cap \mathbf{X} = \emptyset$, then $f(x) \neq 0$ for all $x \in \mathbf{X}$.

Proof. If $N(\tilde{x}, \mathbf{X}) \subseteq \mathbf{X}$, then

$$x_1 \leq \tilde{x} - f(\tilde{x})/f'(\xi) \leq x_2 \tag{6.4}$$

for all $\xi \in \mathbf{X}$. Therefore $0 \notin f'(\mathbf{X})$ implies

$$(f(\tilde{x}) + f'(\xi_1)(x_1 - \tilde{x})) \cdot (f(\tilde{x}) + f'(\xi_2)(x_2 - \tilde{x})) \leq 0$$

for all $\xi_1, \xi_2 \in \mathbf{X}$, and in particular $f(x_1) \cdot f(x_2) \leq 0$. So there is a root of f in $\mathbf{X}$, which is unique because $0 \notin f'(\mathbf{X})$.

Suppose $\hat{x} \in \mathbf{X}$ is a root of f. By the Mean Value Theorem there exists $\xi \in X$ with $f(\tilde{x}) = f'(\xi)(\tilde{x} - \hat{x})$ or $\hat{x} = \tilde{x} - f(\tilde{x})/f'(\xi) \in N(\tilde{x}, \mathbf{X})$, and the result follows. □

For a univariate function f it is not difficult to certify that an interval contains a root of f. However, to verify that a certain interval does *not* contain a root is not that simple or obvious.

Theorem 6.2 and in particular (6.3) are suitable for application of interval arithmetic. Let $\mathbf{X}$ be given. The assumptions of Theorem 6.2 are verified as follows.

(1) Let $\mathbf{F}$ and $\mathbf{Fs}$ be interval extensions of f and f', respectively.
(2) If $\mathbf{Fs}(\mathbf{X})$ does not contain zero, then $f'(x) \neq 0$ for $x \in \mathbf{X}$.
(3) If $\tilde{x} - \mathbf{F}(\tilde{x})/\mathbf{Fs}(\mathbf{X}) \in \mathbf{X}$, then $\mathbf{X}$ contains a unique root of f.
(4) If $\{\tilde{x} - \mathbf{F}(\tilde{x})/\mathbf{Fs}(\mathbf{X})\} \cap \mathbf{X} = \emptyset$, then $\mathbf{X}$ contains no root of f.

Step (1) is directly solved by writing down the functions in INTLAB. We then define the computable function

$$\mathbf{N}(\tilde{x}, \mathbf{X}) := \tilde{x} - \mathbf{F}(\tilde{x})/\mathbf{Fs}(\mathbf{X}) : \mathbb{F} \times \mathbb{IF} \to \mathbb{IF}, \tag{6.5}$$

where all operations are interval operations with floating-point bounds. Then always $N(\tilde{x}, \mathbf{X}) \subseteq \mathbf{N}(\tilde{x}, \mathbf{X})$, so that the assumptions of Theorem 6.2 can be confirmed on the computer.

After verifying step (2) for the initial interval $\mathbf{X} := [1, 2]$, we obtain the following results:

```
>> X=infsup(1,2);
   for i=1:4, xs=intval(mid(X)); X = xs - (xs^2-2)/(2*X), end
intval X =
[   1.37499999999999,   1.43750000000001]
intval X =
[   1.41406249999999,   1.41441761363637]
intval X =
[   1.41421355929452,   1.41421356594718]
intval X =
[   1.41421356237309,   1.41421356237310]
```

Starting with the wide interval $[1, 2]$, an accurate inclusion of the root of f is achieved after 4 iterations. Note that the type cast of `xs` to type `intval` by `xs = intval(mid(X))` is mandatory. Using `xs = mid(X)` instead, the computation of `xs^2-2` would be performed in floating-point arithmetic. Moreover, as in Theorem 6.2, `xs` does not need to be the exact midpoint of `X`; only `xs` $\in$ `X` is required. This is true for the `mid`-function. Note that $\big[\tilde{x} - \mathbf{F}(\tilde{x})/\mathbf{F}(\mathbf{X})\big] \cap \mathbf{X}$ contains the root of f as well; in our example, however, it makes no difference.

Also note that all output of INTLAB is rigorous. This means that a *displayed* lower bound is less than or equal to the computed lower bound, and similarly for the upper bound.

For narrow intervals we have found another form of display useful:

```
>> format _; X=infsup(1,2);
   for i=1:4, xs=intval(mid(X)); X = xs - (xs^2-2)/(2*X), end
intval X =
[   1.37499999999999,   1.43750000000001]
intval X =
   1.414___________
intval X =
   1.41421356______
intval X =
   1.41421356237309
```

A true inclusion is obtained from the display by subtracting and adding 1

to the last displayed digit. For example, a true inclusion of the last iterate is

$$[1.41421356237308, 1.41421356237310].$$

This kind of display allows us to grasp easily the accuracy of the result. In this particular case it also illustrates the quadratic convergence.

7. Standard functions and conversion

The concepts of the previous section can be directly applied to the evaluation of standard functions. For example,

$$\sin(x) \in x - x^3/3! + x^5/5! + \frac{x^7}{7!}[-1, 1] \tag{7.1}$$

is a true statement for all $x \in \mathbb{R}$. A remarkable fact is that (7.1) can be applied to interval input as well:

$$\sin(x) \in \left\{ \mathbf{X} - \mathbf{X}^3/3! + \mathbf{X}^5/5! + \frac{\mathbf{X}^7}{7!}[-1, 1] \right\} \cap [-1, 1] \tag{7.2}$$

is a true statement for all $x \in \mathbf{X}$. Of course, the quality of such an inclusion may be weak. Without going into details, we mention that, among others, for

- exp, log, sqrt
- sin, cos, tan, cot and their inverse functions
- sinh, cosh, tanh, coth and their inverse functions

(7.3)

interval extensions are implemented in INTLAB. Much care is necessary to choose the right approximation formulas, and in particular how to evaluate them. In consequence, the computed results are mostly accurate to the last digit. The algorithms are based on a table-driven approach carefully using the built-in functions at specific sets of points and using addition theorems for the standard functions. When no addition theorem is available (such as for the inverse tangent), other special methods have been developed.

Those techniques for standard functions are presented in Rump (2001*a*) with the implementational details. In particular a fast technique by Payne and Hanek (1983) was rediscovered to evaluate sine and cosine accurately for very large arguments. All standard functions in (7.3) are implemented in INTLAB also for complex (interval) arguments, for point and for interval input data. The implementation of the latter follows Börsken (1978).

Other techniques for the implementation of elementary standard functions were given by Braune (1987) and Krämer (1987). Interval implementations for some higher transcendental functions are known as well, for example for the Gamma function by Krämer (1991).

7.1. Conversion

Some care is necessary when converting real numbers into intervals. When executing the statement

```
X = intval(3.5)
```

MATLAB first converts the input '3.5' into the nearest floating-point number f, and then defines `X` to be the point interval $[f, f]$. In this case indeed $f = 3.5$, because 3.5 has a finite binary expansion and belongs to $\mathbb{F}$. However, the statement

```
X = intval(0.1)
```

produces a point interval *not* containing the real number $0.1 \notin \mathbb{F}$. The problem is that the routine `intval` receives as input the floating-point number nearest to the real number 0.1, because conversion of $0.1 \in \mathbb{R}$ into $\mathbb{F}$ has already taken place. To overcome this problem, the conversion has to be performed by INTLAB using

```
X = intval('0.1')
```

The result is an interval truly containing the real number 0.1. Similar considerations apply to transcendental numbers. For example,

```
>> Y = sin(intval(pi))
intval Y =
  1.0e-015 *
[   0.12246467991473,   0.12246467991474]
```

is an inclusion of $\sin(f)$, where f is the nearest floating-point number to π. Note that `Y` does not contain zero. The command

```
>> sin(intval('pi'))
intval ans =
  1.0e-015 *
[  -0.32162452993533,   0.12246467991474]
```

uses an inclusion of the transcendental number π as argument for the sine, hence the inclusion of the sine must contain zero. The statement `sin(4 * atan(intval(1)))` would also work.

8. Range of a function

The evaluation of the interval extension of a function results in an inclusion of the range of the function over the input interval. Note that this is achieved by a mere evaluation of the function, without further knowledge such as extrema, Lipschitz properties and the like. Sometimes it is possible to rearrange the function to improve this inclusion.

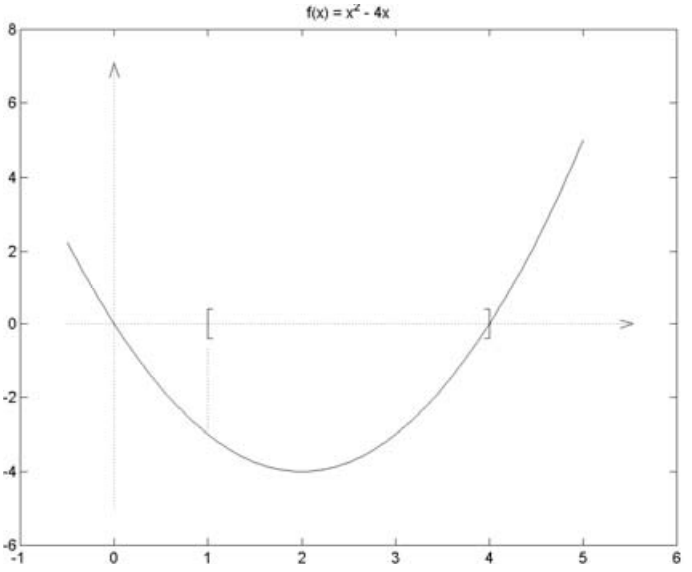

Figure 8.1. Graph of $f(x) = x^2 - 4x$.

8.1. Rearrangement of a function

Consider a simple model function

$$f(x) := x^2 - 4x \quad \text{over} \quad \mathbf{X} := [1, 4]. \tag{8.1}$$

From the graph of the function in Figure 8.1, the range is clearly $\{f(x) : x \in \mathbf{X}\} = [-4, 0]$. The straightforward interval evaluation yields

```
>> X = infsup(1,4); X^2-4*X
intval ans =
[  -15.0000,   12.0000]
```

Now $f(x) = x(x-4)$, so

```
>> X = infsup(1,4); X*(X-4)
intval ans =
[  -12.0000,    0.0000]
```

is an inclusion as well. But also $f(x) = (x-2)^2 - 4$, and this yields the exact range:

```
>> X = infsup(1,4); (X-2)^2-4
intval ans =
[   -4.0000,    0.0000]
```

Note that the real function $f : \mathbb{R} \to \mathbb{R}$ is manipulated, *not* the interval extension $\mathbf{F} : \mathbb{IF} \to \mathbb{IF}$. Manipulation of expressions including intervals should be done with great care; for example, only $(\mathbf{X}+\mathbf{Y})\cdot\mathbf{Z} \subseteq \mathbf{X}\cdot\mathbf{Y}+\mathbf{X}\cdot\mathbf{Z}$

is valid. Also note that the interval square function is the interval extension of $s(x) := x^2$ and is therefore evaluated by

$$\mathbf{X}^2 := \big[\mathbf{X} \cdot \mathbf{X}\big] \cap [0, \infty], \tag{8.2}$$

the bounds of which are easily calculated by a case distinction, whether $\mathbf{X}$ is completely positive, negative or contains zero.

It is not by accident that $f(x) = (x-2)^2 - 4$ yields the exact range of $f(\mathbf{X})$ (neglecting possible overestimation by directed rounding). In fact, it achieves this for every input interval $\mathbf{X}$. The reason is that the input interval occurs only once, so that there is no dependency. Because each individual operation computes the best-possible result, this is true for the final inclusion as well (apparently this was known to Sunaga (1956); it was formulated in Moore (1966, p. 11)).

Theorem 8.1. Let a function $f : \mathbb{R}^n \to \mathbb{R}$ be given, and let $\mathbf{F} : \mathbb{IR}^n \to \mathbb{IR}$ be its interval extension. If f is evaluated by some arithmetic expression and each variable x_i occurs at most once, then

$$\mathbf{F}(\mathbf{X}_1, \ldots, \mathbf{X}_n) = \{f(x_1, \ldots, x_n) : x_i \in \mathbf{X}_i \quad \text{for} \quad 1 \le i \le n\}, \tag{8.3}$$

for all $\mathbf{X}_1, \ldots, \mathbf{X}_n \in \mathbb{IF}$.

Unfortunately there seems to be no general recipe for how to rearrange a function in order to minimize overestimation. It may well be that in one expression the input variables occur less often than in another, and yet the overestimation is worse.

8.2. Oscillating functions

Consider more involved functions. One may expect that for functions with many extrema, an overestimation of the range is more likely. Consider

$$f(x) := \sin\left(\frac{2x^2}{\sqrt{\cosh x}} - x\right) - \operatorname{atan}(4x+1) + 1 \quad \text{over} \quad X := [0, 4]. \tag{8.4}$$

The graph of this function is shown in Figure 8.2. One verifies that the minimum and maximum are achieved near $x_1 = 0.408$ and $x_2 = 1.556$, so that

$$\{f(x) : x \in \mathbf{X}\} \subseteq [-0.2959, 0.5656] \tag{8.5}$$

is a true statement and best possible in 4 digits. The direct interval evaluation yields

```
>> f = inline('sin(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1');
   Y = f(infsup(0,4))
intval Y =
[   -1.5121,    1.2147]
```

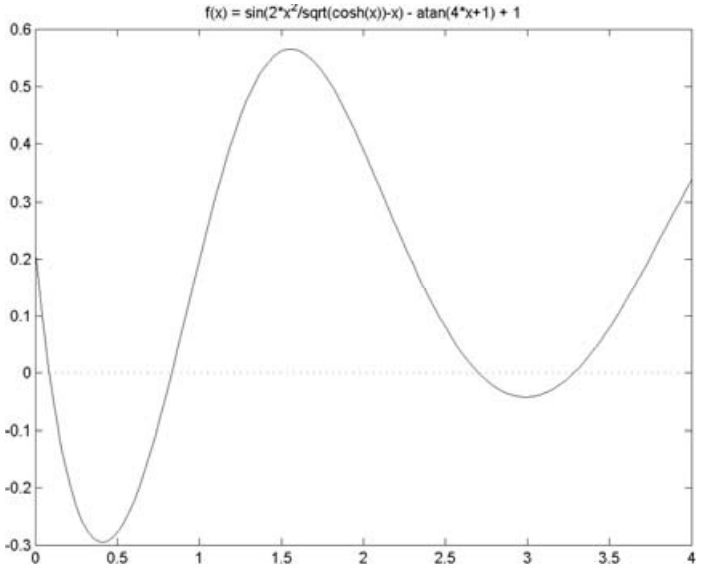

Figure 8.2. Graph of $f(x)$ as defined in (8.4).

which is a true result with some overestimation. Next we change the function by simply replacing the sine function by the hyperbolic sine:

$$g(x) := \sinh\left(\frac{2x^2}{\sqrt{\cosh x}} - x\right) - \operatorname{atan}(4x+1) + 1 \quad \text{over} \quad X := [0,4]. \tag{8.6}$$

The graph of this function is shown in Figure 8.3. One might think that the range of this function would be even simpler to estimate. The best-possible inclusion to 4 digits is

$$\{g(x) : x \in \mathbf{X}\} \subseteq [-0.2962, 6.7189], \tag{8.7}$$

whereas the interval evaluation yields the gross overestimate

```
>> g = inline('sinh(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1');
   format short e
   Y = g(infsup(0,4))
intval Y =
[ -2.7802e+001,  3.9482e+013]
```

Note that the interval evaluation of the standard functions is practically best possible. The reason is once again data dependency. The function

$$h_1(x) := \frac{2x^2}{\sqrt{\cosh\, x}}$$

is not too far from $h_2(x) := x$ over $[0,4]$, so that the range of the argument $h_3(x) := h_1(x) - x$ of the hyperbolic sine is $h_3([0,4]) \subseteq [-0.1271, 2.6737]$.

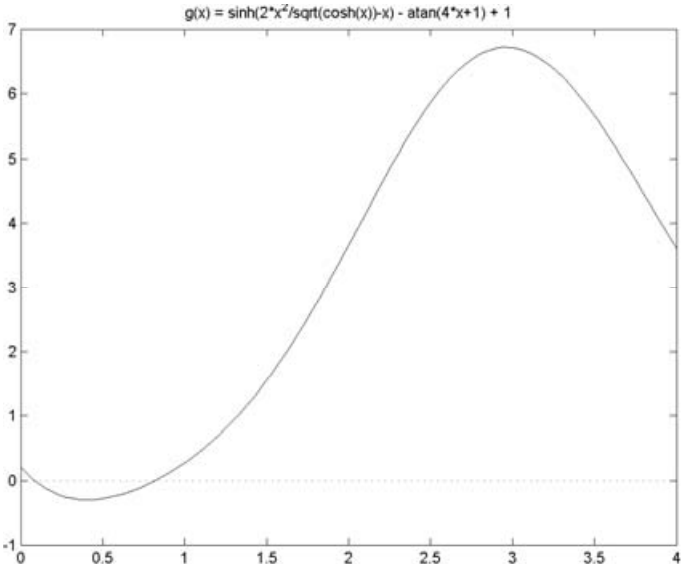

Figure 8.3. Graph of $g(x)$ as defined in (8.6).

However,

$$\left\{ \frac{2\xi_1^2}{\sqrt{\cosh\ \xi_2}} - \xi_3 : \xi_i \in [0,4] \right\} = [-4, 32].$$

This overestimate is due to data dependency, and the result using power set operations is the same. Then this overestimate is amplified by the hyperbolic sine and produces the observed gross overestimate for the function g defined in (8.6).

In contrast, in the first function f defined in (8.4), the sine function reduces the overestimate, since the output of the interval sine function is always bounded by $[-1, 1]$. We come to this again in Section 11.6.

8.3. Improved range estimation

Let a real function $f : \mathbb{R} \to \mathbb{R}$ and its interval extension $\mathbf{F} : \mathbb{IF} \to \mathbb{IF}$ be given. For a given $\mathbf{X} = [a, b] \in \mathbb{IF}$, we always have $f(\mathbf{X}) \subseteq \mathbf{F}(\mathbf{X})$, so that $0 \notin \mathbf{F}(\mathbf{X})$ implies that f has no roots in $\mathbf{X}$. Applying this to an interval extension $\mathbf{Fs} : \mathbb{IF} \to \mathbb{IF}$ of f' means that f is monotone on $\mathbf{X}$ provided $0 \notin \mathbf{Fs}(\mathbf{X})$. In this case

$$0 \notin \mathbf{Fs}(\mathbf{X}) \quad \Rightarrow \quad f(\mathbf{X}) = \{f(x) : x \in \mathbf{X}\} = f(a) \cup f(b). \tag{8.8}$$

Hence a straightforward way to improve the range estimation $f(\mathbf{X})$ is to apply (8.8) if $0 \notin \mathbf{Fs}(\mathbf{X})$ is true, and otherwise to apply a bisection scheme.

Table 8.1. Range inclusion for g as in (8.6), using bisection scheme and (8.8).

Bisection depth	Function evaluations	Range
4	31	$[-0.6415, 19.1872]$
5	53	$[-0.4864, 11.2932]$
6	73	$[-0.3919, 8.7007]$
7	93	$[-0.3448, 7.6441]$
8	115	$[-0.3206, 7.1664]$
9	137	$[-0.3084, 6.9389]$
10	159	$[-0.3023, 6.8280]$
true range		$[-0.2962, 6.7189]$
`fminbnd`	19	$]-0.2961, 6.7189[$

Doing this for the function g defined in (8.6) for different bisection depths yields the results displayed in Table 8.1.

In the second-to-last row the true range is displayed. So, as may be expected, with increasing bisection depth the range estimation improves at the price of more function evaluations.

Note that, despite possible overestimation, interval evaluation has two advantages. First, the estimates come without further knowledge of the function, and second, all range estimates are rigorous. Using pure floating-point arithmetic, this rigour is hardly achievable. We may use some nonlinear function minimization, for example the function `fminbnd` in MATLAB. According to the specification it attempts to find a local minimizer of a given function within a specified interval. Applying this to g and $-g$ yields the result displayed in the last row of Table 8.1. Obviously, with a few function evaluations the true range is found.

Note that in principle the local minimization approach finds an *inner* inclusion of the true range,[12] and it may occur that the global minimum within the given interval is missed. Applying the same scheme to the function f as in (8.4), we obtain the results displayed in Table 8.2.

Again, increasing bisection depth increases the accuracy of the range estimation. However, the minimization function `fminbnd` fails to find the global minimum near 0.4 and underestimates the true range. The displayed bounds are calculated using the default values. Setting the maximum number of function evaluations by `fminbnd` to 1000 and the termination tolerance on

[12] Provided function evaluations are rigorous.

Table 8.2. Range inclusion for f as in (8.4), using bisection scheme and (8.8).

Bisection depth	Function evaluations	Range
4	31	$[-1.0009, 0.6251]$
5	59	$[-0.5142, 0.5920]$
6	91	$[-0.3903, 0.5772]$
7	101	$[-0.3442, 0.5717]$
8	113	$[-0.3202, 0.5686]$
9	127	$[-0.3081, 0.5671]$
10	137	$[-0.3020, 0.5664]$
true range		$[-0.2959, 0.5656]$
`fminbnd`	18	$]-0.0424, 0.5656[$

the minimizer and the function value to 10^{-12} increases the number of function evaluations to 21 but produces the same result.

The approach described in this section (as well as the Newton iteration in Theorem 6.2) requires an interval extension of the derivative. It is not satisfactory for this to be provided by a user. In fact, the range of the first and higher derivatives in the univariate as well as the multivariate case can be computed automatically. This will be described in Section 11.

Moreover, in Section 11.6 derivatives and slopes will be used to improve the range estimation for narrow input intervals; see also Section 13.1. Further background information, together with a challenge problem, is given by Neumaier (2010).

9. Interval vectors and matrices

In order to treat multivariate functions, we need to define interval vectors and matrices. An interval vector $\mathbf{X}$ may be defined as an n-tuple with interval entries such that

$$\mathbf{X} := \{x \in \mathbb{R}^n : x_i \in \mathbf{X}_i \quad \text{for} \quad 1 \le i \le n\}.$$

Obviously an interval vector is the Cartesian product of (one-dimensional) intervals. There is another, equivalent definition using the partial ordering $\le$ on $\mathbb{R}^n$, namely $\mathbf{X} = [\underline{x}, \overline{x}]$ for $\underline{x}, \overline{x} \in \mathbb{R}^n$ with

$$\mathbf{X} = \{x \in \mathbb{R}^n : \underline{x} \le x \le \overline{x}\}, \tag{9.1}$$

where comparison of vectors and matrices is always to be understood componentwise. Both representations are useful. Since they are equivalent we use $\mathbb{IR}^n$ to denote interval vectors. The set of interval matrices $\mathbb{IR}^{n\times n}$

is defined similarly. The lower and upper bound of an interval quantity $\mathbf{X}$ is denoted by $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$, respectively.

Interval operations extend straightforwardly to vectors and matrices by replacement of each individual (real) operation by its corresponding interval operation. For example,

$$\mathbf{A} = \begin{pmatrix} [-2,1] & -2 \\ [0,2] & [1,3] \end{pmatrix} \text{ and } \mathbf{X} = \begin{pmatrix} [-3,2] \\ 4 \end{pmatrix} \text{ yield } \mathbf{AX} = \begin{pmatrix} [-12,-2] \\ [-2,16] \end{pmatrix}. \tag{9.2}$$

As before, we use the natural embedding of $\mathbb{R}$ into $\mathbb{IR}$. This concept of point intervals extends to vectors and matrices as well, and the names point vector or point matrix are used, respectively. Note that for $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and $\mathbf{X} \in \mathbb{IR}^n$ the inclusion property (5.16) implies

$$A \in \mathbf{A}, x \in \mathbf{X} \quad \Rightarrow \quad Ax \in \mathbf{AX}. \tag{9.3}$$

As before, we assume interval operations to be used if at least one operand is an interval quantity. Next consider two special cases. First, let an interval matrix $\mathbf{A} \in \mathbb{IR}^{n\times n}$ and a vector $x \in \mathbb{R}^n$ be given. Then Theorem 8.1 implies

$$\mathbf{A}x = \{Ax : A \in \mathbf{A}\}. \tag{9.4}$$

That means there is no overestimation; the interval operation and the power set operation coincide. The reason is that the components $\mathbf{A}_{ij}$ of $\mathbf{A}$, the only occurring intervals, are used only once in $\mathbf{A}x$.

Secondly, consider a real matrix $A \in \mathbb{R}^{n\times n}$ and an interval vector $\mathbf{X} \in \mathbb{IR}^n$. Now in the interval multiplication $A\mathbf{X}$ each interval component $\mathbf{X}_i$ is used n times, so in general the interval product $A\mathbf{X}$ will be an overestimation by the power set operation:

$$\{Ax : x \in \mathbf{X}\} \subseteq A\mathbf{X}. \tag{9.5}$$

However, for the computation of each *component* $(A\mathbf{X})_i$ of the result, each interval component $\mathbf{X}_j$ is used only once, so componentwise there is no overestimation. In other words, the resulting interval vector $A\mathbf{X}$ is narrowest possible, that is,

$$A\mathbf{X} = \bigcap \{\mathbf{Z} \in \mathbb{IR} \,:\, Ax \in \mathbf{Z}, x \in \mathbf{X}\} =: \operatorname{hull}\big(\{Ax : x \in X\}\big). \tag{9.6}$$

This is called the *interval hull*. Consider

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{X} = \begin{pmatrix} [2,3] \\ [1,3] \end{pmatrix}. \tag{9.7}$$

Then Figure 9.1 illustrates the result of the power set operation $\{Ax : x \in \mathbf{X}\}$ and the interval operation $A\mathbf{X}$. The power set operation is a linear transformation of the axis-parallel rectangle $\mathbf{X}$.

This creates a pitfall. Consider a system of linear equations $Ax = b$ for $A \in \mathbb{R}^{n\times n}$ and $b, x \in \mathbb{R}^n$. Using an approximate solution $\tilde{x}$ computed

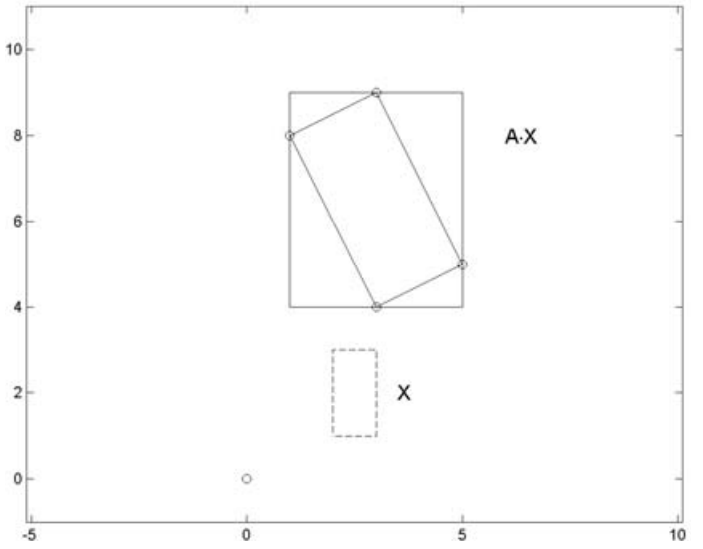

Figure 9.1. Power set and interval product for A and $\mathbf{X}$ as in (9.7).

by some standard algorithm, define an interval vector $\mathbf{X}$ containing $\tilde{x}$, for example $\mathbf{X} := [1-e, 1+e] \cdot \tilde{x}$ for some small $0 < e \in \mathbb{R}$. Because $A\mathbf{X}$ is narrowest possible, one might conclude that $\mathbf{X}$ contains the solution $A^{-1}b$ if $b \in A\mathbf{X}$. It is true that $Ax \in A\mathbf{X}$ for all $x \in \mathbf{X}$; however,

$$\forall\, y \in A\mathbf{X} \;\; \exists\, x \in \mathbf{X} \;:\; y = Ax \quad \text{is not true} \tag{9.8}$$

(see Figure 9.1 for the data in (9.7)). For example, $x := (4,8)^T \in A\mathbf{X} = ([1,5],[4,9])^T$, but $A^{-1}x = (3.2, 2.4)^T \notin \mathbf{X}$.

9.1. Performance aspects

The definition of interval vector and matrix operations by replacement of each individual (real) operation by its corresponding interval operation is theoretically useful, but it implies a severe performance impact. This is because on today's architectures dramatic speed-up is obtained by instruction-level parallelism and by avoiding cache-misses. On the contrary, branches, in particular, may slow down an algorithm significantly.

Our definition of the product of two interval matrices $\mathbf{A}, \mathbf{B} \in \mathbb{IF}^{n\times n}$, say, implies that in the inner loop an interval product $\mathbf{A}_{ik}\mathbf{B}_{kj}$ has to be computed. This requires at least two branches to decide whether the input intervals are non-negative, non-positive or contain zero, and two switches of the rounding mode to compute the lower and upper bound. In the worst case of two intervals both containing zero in the interior, four products must be computed in total.

Table 9.1. Performance in GFLOPs for LAPACK matrix multiplication (DGEMM), Gaussian elimination with partial pivoting (DGETRF) and with total pivoting (DGETC2); one branch is counted as one FLOP.

n	DGEMM	DGETRF	DGETC2
2000	121	68	0.20
4000	151	114	0.15
6000	159	128	0.12
8000	163	142	0.14
10000	166	150	0.11

But a pure counting of floating-point operations does not show how well a code can be optimized, and in particular it hides the disastrous effect of branches. As an example of how branches may slow down a computation we refer to the well-known LAPACK package described in Anderson *et al.* (1995). All routines are well written by the best experts in the field. We compare the routines DGEMM for multiplying two matrices, Gaussian elimination with partial pivoting DGETRF and Gaussian elimination with total pivoting DGETC2. For Table 9.1 we count each addition, multiplication and branch as 1 FLOP and display the GFLOPs achieved for all routines. The environment is a PC with four Quad-Core AMD Opteron 8393 SE with 3.1 GHz clock speed, thus 16 cores in total. Each core may execute up to four operations in parallel, so that the peak performance is $64 \times 3.1 = 198.4$ GFLOPs.[13]

Note that if all codes could be equally well optimized and there were no time penalty for branches, all GFLOP counts in Table 9.1 would be equal. However, we observe a slow-down by a factor well over 1000 for dimension $n = 10\,000$. Similarly a significant slow-down is observed when implementing the interval matrix product according to the theoretical definition by case distinctions.

For a fast implementation of interval vector and matrix operations, Rump (1999*b*) analysed that the conversion into midpoint-radius form is useful. We defined an interval by its left and right bound. Equivalently, we may use

$$\mathbf{X} = \langle m, r \rangle := \{x \in \mathbb{R} : m - r \leq x \leq m + r\}, \tag{9.9}$$

and similarly for vectors and matrices using the componentwise partial ordering in $\mathbb{R}^n$ and $\mathbb{R}^{n \times n}$. Sometimes the midpoint-radius is advantageous

[13] Thanks to Takeshi Ogita for performing the tests.

because the minimum radius is not restricted to the distance of adjacent floating-point numbers, but tiny radii are possible as well.

First, consider the product $\mathbf{A} \cdot B$ of an interval matrix $\mathbf{A} = [\underline{A}, \overline{A}] \in \mathbb{IF}^{m \times k}$ and a real matrix $B \in \mathbb{F}^{k \times n}$. Setting $mA := 0.5(\underline{A} + \overline{A})$ and $rA := 0.5(\overline{A} - \underline{A})$, a little thought reveals

$$\mathbf{A} \cdot B = \langle mA, rA \rangle \cdot B = \langle mA \cdot B,\ rA \cdot |B| \rangle. \tag{9.10}$$

For the computer implementation note that both the conversion of the interval matrix $\mathbf{A}$ into midpoint-radius form and the products $mA \cdot B$ and $rA \cdot |B|$ are subject to rounding errors. Fortunately this problem can be solved in a simple way.

Algorithm 9.1. Fast matrix multiplication $[\texttt{Ainf}, \texttt{Asup}] \cdot \texttt{B}$:

```
setround(1)
mA = 0.5*(Ainf+Asup);
rA = mA - Ainf;
rC = rA*abs(B);
Csup = mA*B + rC;
setound(-1)
Cinf = mA*B - rC;
```

The elegant conversion from left and right bound to midpoint-radius form is due to Oishi (1998). Algorithm 9.1 ensures that, for the computed floating-point matrices `mA` and `rA`,

$$\texttt{mA} - \texttt{rA} \le \texttt{Ainf} \le \texttt{Asup} \le \texttt{mA} + \texttt{rA}. \tag{9.11}$$

Moreover, $\texttt{abs(B)} = |\texttt{B}|$ because $\mathbb{F} = -\mathbb{F}$, and the setting of the rounding mode yields

$$0 \le \texttt{rA} \cdot |\texttt{B}| \le \texttt{rC} \tag{9.12}$$

for the floating-point matrices `rA`, `B` and `rC`, so that

$$\texttt{Cinf} \le \texttt{mA} \cdot |\texttt{B}| - \texttt{rC} \le \texttt{mA} \cdot |\texttt{B}| + \texttt{rC} \le \texttt{Csup} \quad \text{and} \quad \mathbf{A} \cdot B \subseteq [\texttt{Cinf}, \texttt{Csup}]. \tag{9.13}$$

Note that not only are various branches and switchings of the rounding mode omitted but the main work reduces to 3 multiplications of real matrices. Therefore, the very fast BLAS routines can be used, which are again much faster than an ordinary 3-loop implementation.

Secondly, consider the product $\mathbf{A} \cdot \mathbf{B}$ of two interval matrices $\mathbf{A} = [mA - rA, mA + rA] \in \mathbb{IF}^{m \times k}$ and $\mathbf{B} = [mB - rB, mB + rB] \in \mathbb{IF}^{k \times n}$. Again it is easy to see that

$$\begin{aligned} &\mathbf{A} \cdot \mathbf{B} \subseteq [mC - rC, mC + rC] \quad \text{for} \\ &\qquad mC := mA \cdot mB \quad \text{and} \quad rC := rA \cdot (|mB| + rB) + |mA| \cdot rB. \end{aligned} \tag{9.14}$$

Note that now there is some overestimation due to the fact that rB is used

Table 9.2. Computing time for real-interval and interval-interval matrix multiplication in C, floating-point multiplication normed to 1.

Dimension	Real × interval matrix		Interval × interval matrix	
	Traditional	(9.10)	Traditional	(9.14)
100	11.3	3.81	110.2	5.36
200	12.4	3.53	131.1	4.94
500	12.3	3.35	134.2	4.60
1000	20.1	3.25	140.0	4.45
1500	23.2	3.18	142.6	4.32
2000	23.6	3.14	144.6	4.25

twice in the computation of rC, and the product of the midpoints $mA \cdot mB$ is not the midpoint of the product $\mathbf{A} \cdot \mathbf{B}$. For example, $[1,3] \cdot [5,9] = [5,27]$ is the usual interval product identical to the power set operation, but $\langle 2,1\rangle \cdot \langle 7,2\rangle = \langle 14,13\rangle = [1,27]$ using (9.14).

However, something strange happens, namely that the *computed* quantities $[mC-rC, mC+rC]$ are sometimes more narrow than $\mathbf{A}\cdot\mathbf{B}$, contradicting (9.14). This is due to outward rounding, as in the following example:

$$[2.31, 2.33] \cdot [3.74, 3.76] = [8.6394, 8.7608] \subseteq [8.63, 8.77] \tag{9.15}$$

is the best-possible result in a 3-digit decimal arithmetic, whereas (9.14) yields

$$\begin{aligned} \langle 2.32, 0.01\rangle \cdot \langle 3.75, 0.01\rangle &= \langle 8.7, 0.01 \cdot (3.75 + 0.01) + 2.32 \cdot 0.01\rangle \\ &= \langle 8.7, 0.0608\rangle, \end{aligned} \tag{9.16}$$

which has a 13% smaller diameter than the result in (9.14).

An implementation of (9.14) along the lines of Algorithm 9.1 is straightforward. Note that the main costs are 4 real matrix multiplications, which are, as before, very fast using BLAS routines.

We first compare the new approaches in (9.10) and (9.14) with a C implementation along the traditional lines. With the time for one ordinary BLAS matrix multiplication in floating-point normed to 1, Table 9.2 displays the timing for different dimensions.[14] It clearly shows the advantage of the new approach, which almost achieves the theoretical ratio.

This method, as in (9.10) and (9.14), is in particular mandatory for a MATLAB implementation: although much effort has been spent on diminishing the effects of interpretation, loops may still slow down a computation

[14] Thanks to Viktor Härter for performing the tests.

Table 9.3. Computing time for interval matrix multiplication in INTLAB, floating-point multiplication normed to 1.

Dimension	3-loop	Rank-1 update	Mid-rad by (9.14)
50	610659.1	81.3	6.4
100		79.6	5.2
200		100.5	4.7
500		173.2	4.3
1000		261.7	4.1

significantly, in particular when user-defined variables such as intervals are involved. The fastest way to compute a tight inclusion without overestimation, as in (9.14), seems to be via rank-1 updates. Both possibilities, tightest and fast inclusions, are available in INTLAB via a system variable.[15]

Table 9.3 shows the timing for the product of two interval matrices for a standard 3-loop approach, for the tight inclusion using rank-1 updates and the fast inclusion via (9.14). The computing time for an ordinary product of two real matrices of the same size is again normed to 1.

As can be seen, the overhead for the tight inclusion increases with the dimension, whereas the time ratio for the fast inclusion approaches the theoretical factor 4. The standard 3-loop approach is unacceptably slow even for small dimensions.

As in the scalar case, interval matrices are a convenient way to deal with matrices not representable in floating-point arithmetic. For example, Algorithm 9.2 produces an inclusion `A` of the matrix in our model problem (2.11).

Algorithm 9.2. Computation of an interval matrix $\mathtt{A} \in \mathbb{IF}^{n \times n}$ containing the real matrix as in (2.11):

```
A = zeros(n);
for i=1:n, for j=1:n
  A(i,j) = intval(1)/(i+(j-1)*n);
end, end
```

Then, for example, the matrix product $\mathtt{A} * \mathtt{A}$ surely contains the square of the original matrix in (2.11). In Section 10 we discuss how to compute an inclusion of the solution of a system of linear equations, in particular when the matrix is not representable in floating-point arithmetic.

[15] This is done by `intvalinit('SharpIVmult')` and `intvalinit('FastIVmult')`, and applies only if both factors are thick interval quantities, *i.e.*, the lower and upper bound do not coincide.

9.2. Representation of intervals

The Cartesian product of a one-dimensional interval suggests itself as a representation of interval vectors. Although operations are easy to define and fast to execute, there are drawbacks.

In particular, there is no 'orientation': the boxes are always parallel to the axes which may cause overestimation. The simplest example is the 2-dimensional unit square rotated by 45: the best inclusion increases the radii by a factor $\sqrt{2}$. This is known as the *wrapping effect*, one of the major obstacles when integrating ODEs over a longer time frame.

As an alternative to interval vectors, Beaumont (2000) considers oblique boxes $Q\mathbf{X}$ for orthogonal Q, where the product is not executed but Q and $\mathbf{X}$ are stored separately.

Another approach involving ellipsoids is defined by Neumaier (1993) (see also Kreinovich, Neumaier and Xiang (2008)), where the ellipsoid is the image of the unit ball by a triangular matrix; for interesting applications see Ovseevich and Chernousko (1987).

Andrade, Comba and Stolfi (1994) define an affine arithmetic, apparently used in other areas, by representing an interval by $x_0 + \sum x_i \mathbf{E}_i$ for $x_i \in \mathbb{R}$ and $\mathbf{E}_i = [-1, 1]$. Often this seems efficient in combatting overestimation; see de Figueiredo and Stolfi (2004).

For each representation, however, advantages are counterbalanced by increased computational costs for interval operations.

PART TWO
Finite-dimensional problems

10. Linear problems

The solution of systems of linear equations is one of the most common of all numerical tasks. Therefore we discuss various aspects of how to compute verified error bounds, from dense systems to large systems, input data with tolerances, estimation of the quality to NP-hard problems. Almost-linear problems such as the algebraic eigenvalue problem are discussed in Section 13.4. We start by showing that a naive approach is bound to fail.

10.1. The failure of the naive approach: interval Gaussian elimination (IGA)

The most commonly used algorithm for a general dense linear system $Ax = b$ is factoring A by Gaussian elimination. From an LU decomposition, computed with partial pivoting, the solution can be directly computed.

Once again it is a true statement that replacement of each operation in this process by its corresponding interval operation produces a correct inclusion of the result – if not ended prematurely by division by an interval containing zero. This naive approach is likely to produce useless results for almost any numerical algorithm, and Gaussian elimination is no exception.

Unfortunately, it is still not fully understood why Gaussian elimination works so successfully in floating-point arithmetic. A detailed and very interesting argument was given by Trefethen and Schreiber (1990). They investigate in particular why partial pivoting suffices despite worst-case exponential growth factors. Their model can be used to shed some light on the failure of IGA.

Let $A^{(1)} := A \in \mathbb{R}^{n\times n}$ be given, and denote by $A^{(k)}$ the modified matrix before step k of Gaussian elimination (we assume the rows of A are already permuted so that the partial pivots are already in place). Then $A^{(k)} = L^{(k)} \cdot A^{(k-1)}$ with $L^{(k)}_{ik} = \varphi^{(k)}_i := -A^{(k)}_{ik}/A^{(k)}_{kk}$ for $k+1 \le i \le n$, or

$$A^{(k+1)}_{ij} = A^{(k)}_{ij} - \varphi^{(k)}_i A^{(k)}_{kj} \quad \text{for} \quad k+1 \le i,j \le n. \tag{10.1}$$

Now suppose Gaussian elimination is performed in interval arithmetic. For intervals $\mathbf{X}, \mathbf{Y} \in \mathbb{IR}$ it is easy to see that

$$\mathtt{rad}(\mathbf{XY}) \ge \mathtt{rad}(\mathbf{X}) \cdot |\mathtt{mid}(\mathbf{Y})| + |\mathtt{mid}(\mathbf{X})| \cdot \mathtt{rad}(\mathbf{X})$$

(Neumaier 1990, Proposition 1.6.5).[16] Using this and (5.8), a very crude

[16] Note the similarity to $(uv)' = u'v + uv'$.

estimate for the radius of the new elements of the elimination is

$$\mathtt{rad}(A_{ij}^{(k+1)}) \geq \mathtt{rad}(A_{ij}^{(k)}) + |\varphi_i^{(k)}| \cdot \mathtt{rad}(A_{kj}^{(k)}). \tag{10.2}$$

In matrix notation this takes the form

$$\mathtt{rad}(A^{(k+1)}) \geq |L^{(k)}| \cdot \mathtt{rad}(A^{(k)}),$$

valid for the first k rows of the upper triangle of $A^{(k+1)}$ and for the lower right square of $A^{(k+1)}$, *i.e.*, for indices $k+1 \leq i, j \leq n$. Finally we obtain

$$\mathtt{rad}(U) \geq \text{upper triangle}\big[|L^{(n-1)}| \cdot \ldots \cdot |L^{(2)}| \cdot |L^{(1)}| \cdot \mathtt{rad}(A)\big]. \tag{10.3}$$

A unit lower triangular matrix with non-zero elements below the diagonal only in column k is inverted by negating those elements. Hence $|L^{(k)}|^{-1} = \langle L^{(k)} \rangle$ using Ostrowski's comparison matrix,

$$\langle A \rangle_{ij} := \begin{cases} |A_{ij}| & \text{for } i = j, \\ -|A_{ij}| & \text{otherwise.} \end{cases} \tag{10.4}$$

It follows that

$$|L^{(n-1)}| \cdot \ldots \cdot |L^{(2)}| \cdot |L^{(1)}| = \big[\langle L^{(1)} \rangle \cdot \langle L^{(2)} \rangle \cdot \ldots \cdot \langle L^{(n-1)} \rangle\big]^{-1} = \langle L \rangle^{-1},$$

and (10.3) implies

$$\mathtt{rad}(U) \geq \text{upper triangle}\big[\langle L \rangle^{-1} \cdot \mathtt{rad}(A)\big]. \tag{10.5}$$

It is known that for a random matrix $\|\langle L \rangle^{-1}\| / \|L^{-1}\|$ is large. However, the L factor of Gaussian elimination is far from random. One reason is that L is generally well-conditioned, even for ill-conditioned A, despite the fact that Viswanath and Trefethen (1998) showed that random lower triangular matrices are generally ill-conditioned.

Adopting the analysis of Gaussian elimination by Trefethen and Schreiber (1990), for many classes of matrices we can expect the multipliers, *i.e.*, the elements of L, to have mean zero with standard deviation

$$\sigma(L_{ik}) \approx \frac{1}{W(m)} \left(1 - \frac{\sqrt{2/\pi} W(m) \mathrm{e}^{-W(m)^2/2}}{\operatorname{erf}(W(m)/\sqrt{2})}\right)^{1/2} \sim \left(\frac{1}{2\log(m\sqrt{2/\pi})}\right)^{1/2}, \tag{10.6}$$

where $m = n + 1 - k$ for partial pivoting, and $W(m)$ denotes the 'winner function' for which

$$W(m) \approx \alpha \left(1 - \frac{2\log\alpha}{1+\alpha^2}\right)^{1/2} + \mathcal{O}\left(\frac{1}{\log m}\right)$$

with $\alpha := \sqrt{2\log(m\sqrt{2/\pi})}$. The expected absolute value of the multipliers is about 0.6745 times the standard deviation. Hence we can compute a matrix $\mathcal{L}$ which, in general, is not too far from $|L|$, and can compute $\Phi := \|\langle \mathcal{L} \rangle^{-1}\|$.

Table 10.1. Lower bound for the amplification factor for interval Gaussian elimination (IGA).

	$n = 20$	$n = 50$	$n = 100$	$n = 150$	$n = 170$
Φ	$1.0 \cdot 10^2$	$1.6 \cdot 10^5$	$1.4 \cdot 10^{10}$	$7.6 \cdot 10^{14}$	$5.4 \cdot 10^{16}$
$\|\langle L\rangle^{-1}\|$	$2.6 \cdot 10^2$	$2.2 \cdot 10^6$	$4.6 \cdot 10^{13}$	$2.7 \cdot 10^{19}$	$5.3 \cdot 10^{21}$

After the first elimination step of interval Gaussian elimination, the inevitable presence of rounding errors produces an interval matrix $\mathbf{A}^{(2)}$ with $\mathtt{rad}(\mathbf{A}^{(2)}) \geq \mathtt{u}|A^{(2)}|$. Thus Φ is a lower bound for the amplification of radii in IGA.

Table 10.1 lists this lower bound Φ for different dimensions. In addition, $\|\langle L\rangle^{-1}\|$ for the factor L of some random matrix is displayed. For small dimension the amplification Φ exceeds $\mathtt{u}^{-1}$, which means breakdown of IGA. These estimates are very crude; in practice the behaviour is even worse than the second line of Table 10.1, as explained in a moment.

The model does not apply to special matrices such as M-matrices or diagonally dominant matrices. In fact one can show that IGA does not break down for such matrices. However, for such matrices very fast methods are available to solve linear systems with rigour; see Section 10.9.

In what follows we monitor the practical behaviour of IGA. For general linear systems the condition number is usually reflected in the U factor. Thus it seems reasonable to monitor the relative error of the last component U_{nn}. We define the relative error of an interval $\mathbf{X} \in \mathbb{IF}$ by

$$\mathtt{relerr}(\mathbf{X}) := \begin{cases} |\frac{\mathtt{rad}(\mathbf{X})}{\mathtt{mid}(\mathbf{X})}| & \text{if } 0 \notin \mathbf{X}, \\ \mathtt{rad}(\mathbf{X}) & \text{otherwise.} \end{cases} \tag{10.7}$$

For randomly generated matrices with normally distributed random entries, Gaussian elimination with partial pivoting is applied to generate $\mathbf{L}$ and $\mathbf{U}$ factors. All operations are interval operations, and the pivot element is the one with largest *mignitude* $\mathrm{mig}(\mathbf{X}) := \min\{|x| : x \in \mathbf{X}\}$. This is about the best one can do.

In Table 10.2 we display the time ratio for a Gaussian elimination for a matrix of the same size in pure floating-point arithmetic, and the median relative error of $\mathbf{U}_{nn}$ over 100 samples. If some interval $\mathbf{U}_{ii}$ contains 0 we call that a failure. The percentage of failures is listed as well, and the time ratio and relative errors displayed are the means over successful examples. As can be seen, the relative error of $\mathbf{U}_{nn}$ increases with the dimension, and for dimensions about 60 to 70 the approach fails completely. Note that this is true even though random matrices are well known to be reasonably

Table 10.2. Results for interval Gaussian elimination (IGA) for random matrices and random orthogonal matrices.

Dimension	Random matrices			Random orthogonal matrices		
	Time ratio	`relerr`(U_{nn})	Failure %	Time ratio	`relerr`(U_{nn})	Failure %
10	173.9	3.4e-013	0	171.6	1.3e-014	0
20	320.5	2.7e-009	0	320.9	6.3e-012	0
30	432.3	2.1e-007	0	419.5	2.5e-009	0
40	482.2	1.0e-004	0	497.7	1.0e-006	0
50	407.0	4.9e-002	2	434.6	4.4e-004	0
60	454.0	6.6e-001	96	414.2	1.4e-001	4
70			100			100

well-conditioned. Moreover, not only do the results become useless, but the approach is also very slow due to many branches and rounding switches, as explained in Section 9.1.[17]

The reason is solely the number of consecutive interval operations, permanently violating the Utilize input data principle (5.13). Data dependencies quickly produce wide and useless results. Interval arithmetic is not to blame for this: the result would be the same with power set operations applied in this way.

To confirm that even the mild condition numbers of random matrices do not contribute, similar data for randomly generated orthogonal matrices are also displayed in Table 10.2.

The picture changes when the input data have a specific structure, such as A being an M-matrix. Then no overestimation occurs, because the No inaccurate cancellation principle (NIC) is satisfied. However, the time penalty persists, and the methods in Section 10.9 should be used.

To emphasize the point, we take a randomly generated 3×3 matrix and multiply it by another random matrix several times in interval arithmetic. Note that all matrices are point matrices (left and right bounds coincide), and that every factor is a new random matrix. All entries of the intermediate products are summed up, and the relative error of this sum is displayed. The following code produces a semi-logarithmic graph of the result.

Algorithm 10.1. Product of random matrices:

```
imax = 65; y = zeros(1,imax); A = intval(randn(3));
for i=1:imax, A=A*randn(3); y(i)=relerr(sum(A(:))); end
close, semilogy(1:imax,y)
```

[17] Note that the code was vectorized where possible.

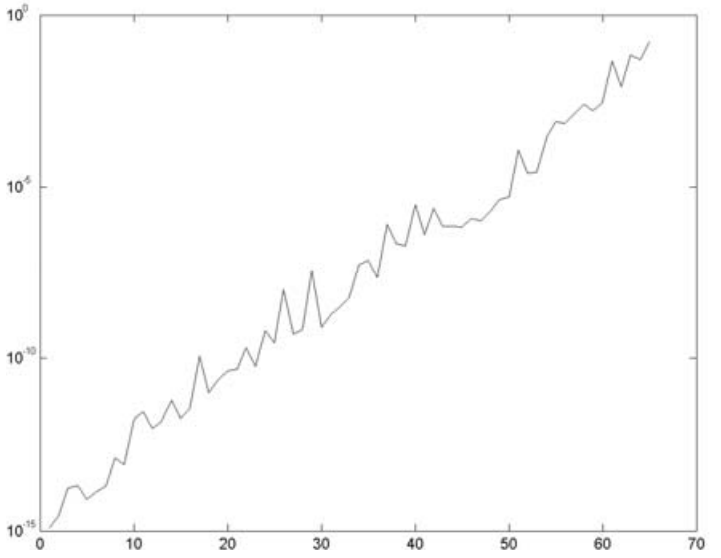

Figure 10.1. Overestimation by naive interval arithmetic and dependencies.

A typical graph of the result is shown in Figure 10.1. As can be seen, rounding errors add from step to step, resulting in an exponentially growing relative error. A rule of thumb roughly predicts an increase of the relative error by a factor 2^K after K iterations. Since $2^{48} \sim 10^{16}$ and double precision corresponds to about 16 decimal places of precision, a relative error 1 can be expected after some 50 iterations.

10.2. Partial pivoting

The method of choice for a dense system of linear equations is Gaussian elimination with partial pivoting. Although it is known that the growth factor may increase exponentially with the dimension, the corresponding examples seemed to be constructed and not occurring in practice. In 1993 Wright (1993) showed practical examples with exponential growth factor. For example, integrating

$$\dot{x} = x - 1 \quad \text{for} \quad x(0) = x(40)$$

and applying a trapezoid rule results in a linear system $Ax = b$, the solution of which is obviously the vector of ones. For given n, the following MATLAB code, given by Foster (1994),

```
T = 40; h = T/(n-1);
b = -(0:n-1)'*h;
A = - h * tril(ones(n),-1);
```

```
A = A + (1-h/2) * diag(ones(n,1));
A(:,1) = -(h/2)*ones(n,1);
A(1,1) = 1; A(:,n) = -ones(n,1);
A(n,n) = -(h/2);
```

computes A and b. Up to $n = 60$ the solution is very accurate, but for larger n the exponential increase of the growth factor produces meaningless results. For example, the components $\mathtt{x}_{62\ldots65}$ of $\mathtt{x} = \mathtt{A}\backslash\mathtt{b}$ for $n = 65$ are

```
-23.272727272727273
                  0
-93.090909090909093
  1.000000000000000
```

rather than all ones. The condition number of the matrix is less than 30, so the disastrous effect of rounding errors is only due to the large growth factor. Remarkably, no warning is given by MATLAB, so a user might trust the computed values. It is also well known that one extra residual iteration in working precision produces a backward stable result. Doing this, even for large dimension, produces accurate results.

There seems little potential for converting floating-point Gaussian elimination into a verification algorithm, because known error estimates need an upper bound for the condition number. There are condition estimators as well, but there is strong evidence that a reliable condition estimator costs as much as a reliable computation of A^{-1}. In fact Demmel, Diament and Malajovich (2001) have shown that reliably computing a bound for $\|A^{-1}\|$ has at least the cost of testing whether the product of two $n \times n$ matrices is zero, which in turn is believed to actually cost as much as computing the product.

10.3. Preconditioning

One way to follow the Utilize input data principle (5.13) and to avoid successive operations on computed data is preconditioning by some approximate inverse R. This very important principle was proposed by Hansen and Smith (1967), analysed in Ris (1972), and its optimality in a certain sense shown by Neumaier (1984).

It is an unwritten rule in numerical analysis never to compute the inverse of a matrix, especially not to solve a linear system $Ax = b$. Indeed, the direct solution by Gaussian elimination is not only faster but also produces more accurate results than multiplication of b by a computed inverse. For the purpose of verification other rules apply, in particular to rely only on the input data if possible. There are other (and faster) verification approaches using factorizations of A; see Oishi and Rump (2002). However, preconditioning by some approximate inverse is of superior quality.

Given a linear system $Ax = b$ with non-singular A, some $R \in \mathbb{R}^{n\times n}$ and some $\tilde{x} \in \mathbb{R}^n$, there follows

$$\|A^{-1}b - \tilde{x}\| = \|\big(I - (I - RA)\big)^{-1}R(b - A\tilde{x})\| \leq \frac{\|R(b - A\tilde{x}\|}{1 - \|I - RA\|} \tag{10.8}$$

provided $\|I - RA\| < 1$. We stress that there are no mathematical assumptions on R: if A is too ill-conditioned and/or R is of poor quality, then $\|I - RA\| < 1$ is not satisfied. Moreover, if $\|I - RA\| < 1$ is satisfied, we conclude *a posteriori* that A (and R) is non-singular. This proves the following.

Theorem 10.2. Let $A, R \in \mathbb{R}^{n\times n}$ and $b, \tilde{x} \in \mathbb{R}^n$ be given, and denote by I the $n \times n$ identity matrix. If $\|I - RA\| < 1$ for some matrix norm, then A is non-singular and

$$\|A^{-1}b - \tilde{x}\| \leq \frac{\|R(b - A\tilde{x})\|}{1 - \|I - RA\|}. \tag{10.9}$$

In particular, the ∞-norm is useful because it directly implies componentwise error bounds on the solution, and it is easy to calculate. This theorem is especially suited to deriving error estimates using interval arithmetic.

10.4. Improved residual

Note that the quality of the bound in (10.9) depends directly on the size of the residual $\|b - A\tilde{x}\|$ and can be improved by some residual iteration on $\tilde{x}$. If dot products can be calculated accurately, an inclusion of the solution $A^{-1}b$ accurate to the last bit can be computed provided $\|I - RA\| < 1$.

This can be done[18] by multi-precision packages, such as, for example, the MPFR-package of Fousse *et al.* (2005), or based on 'error-free transformations', as discussed in Section 3. Recently the latter techniques were used to derive very fast algorithms for computing rigorous error bounds of sums and dot products of vectors of floating-point numbers to arbitrary accuracy, for example by Zielke and Drygalla (2003), Zhu, Yong and Zheng (2005), Rump, Ogita and Oishi (2008) and Rump (2009). The algorithms are particularly fast because there are no branches, and only floating-point operations in one working precision are used. INTLAB contains reference implementations, which however, severely suffer from interpretation overhead.

Some improvement can be achieved by the 'poor man's residual' algorithm `lssresidual`. It is based on the following algorithm by Dekker for splitting a floating-point number into some higher- and lower-order part.

[18] Sometimes it is preferable to solve the given linear system using a multi-precision package from the beginning.

Algorithm 10.3. Error-free splitting of a floating-point number into two parts:

$$\begin{array}{ll} \text{function } [x, y] = \text{Split}(a) & \\ \quad c = \text{fl}(\varphi \cdot a) & \% \ \varphi = 2^s + 1 \\ \quad x = \text{fl}(c - \text{fl}(c - a)) & \\ \quad y = \text{fl}(a - x) & \end{array}$$

As a result $a = x + y$ for all $a \in \mathbb{F}$, and in 53-bit precision x and y have at most $53 - s$ and $s - 1$ significant bits. In particular, for $s = 27$ a 53-bit number is split into two 26-bit parts, which can be multiplied in floating-point arithmetic without error. The trick is that the sign bit is used as an extra bit of information.

Algorithm 10.4. Improved computation of the residual of a linear system (poor man's residual):

```
function res = lssresidual(A,x,b)
   factor = 68719476737;   % heuristically optimal splitting 2^36+1
   C = factor*A;
   Abig = C - A;
   A1 = C - Abig;          % upper part of A, first 17 bits
   A2 = A - A1;            % A = A1+A2 exact splitting
   x = -x; y = factor*x;
   xbig = y - x;
   x1 = y - xbig;          % upper part of -x, first 17 bits
   x2 = x - x1;            % -x = x1+x2 exact splitting
   res = (A1*x1+b)+(A1*x2+A2*x);
```

This algorithm `lssresidual` splits `A` and `x` into $17 + 35$ bits, so that the product `A1` $*$ `x1` does not cause a rounding error if the elements of neither `A` nor `x` cover a wide exponent range.

`lssresidual` is a cheap way to improve the residual, and thus the quality of the solution of a linear system, also for numerical algorithms without verification. The command `intvalinit('ImprovedResidual')` sets a flag in INTLAB so that `verifylss` uses this residual improvement. It is applicable for a matrix residual $I - RA$ as well.

10.5. Dense linear systems

Before discussing other approaches for verification, we show the computed bounds by Theorem 10.2 for a linear system with matrix as in (2.11) of dimension $n = 9$. Note that, except for $n = 2$, these matrices are more ill-conditioned than the well-known and notoriously ill-conditioned Hilbert matrices. The right side is computed so that the solution $A^{-1}b$ is the vector of ones. This is done by the following executable code, first for the floating-

point matrix nearest to the original matrix:

```
n = 9; A = 1./reshape(1:n^2,n,n); b = A*ones(n,1);
R = inv(A); xs = A\b;
d = norm(eye(n)-R*intval(A),inf);
if d<1
  midrad( xs , mag(norm(R*(b-A*intval(xs)),inf)/(1-d)) )
end
```

Note that the quantity `d` is estimated in interval arithmetic and is itself of type interval. The magnitude of an interval $\mathbf{X}$ is defined by $\mathrm{mag}(\mathbf{X}) := \max\{|x| : x \in \mathbf{X}\} \in \mathbb{F}$. The result is as follows:

```
intval ans =
[    0.8429,    1.1571]
[    0.8429,    1.1571]
[    0.8429,    1.1571]
[    0.8428,    1.1569]
[    0.8441,    1.1582]
[    0.8384,    1.1526]
[    0.8514,    1.1656]
[    0.8353,    1.1495]
[    0.8455,    1.1596]
```

The quality of the inclusion seems poor, but it corresponds roughly to the condition number $\mathrm{cond}(A) = 4.7 \cdot 10^{14}$. This is the largest dimension of the matrix as in (2.11), for which Theorem 10.2 is applicable. For $n = 10$ the condition number of A is $8.6 \cdot 10^{16}$.

For a completely rigorous result there are two flaws. First, as has been mentioned, we do not use the true, real matrices as in (2.11), but floating-point approximations. Second, the right-hand side b is computed in floating-point arithmetic as well, which means that the vector of ones is likely *not* the solution of the linear system. Both problems are easily solved by the following code:

```
n = 9; A = 1./intval(reshape(1:n^2,n,n)); b = A*ones(n,1);
R = inv(A.mid); xs = A.mid\b.mid;
d = norm(eye(n)-R*A,inf);
if d<1
  midrad( xs , mag(norm(R*(b-A*xs),inf)/(1-d)) )
end
```

Now the interval matrix `A` contains the original real matrix as in (2.11). The assertions of Theorem 10.2 are valid for all matrices within the interval matrix `A`, in particular for the original real matrix. This statement includes the non-singularity as well as the error bounds. The result is shown in Table 10.3; the true solution is again the vector of all ones.

Table 10.3. Inclusion by Theorem 10.2 for a 9×9 linear system with matrix as in (2.11).

```
intval ans =
[    0.6356,    1.3644]
[    0.6356,    1.3644]
[    0.6356,    1.3644]
[    0.6357,    1.3645]
[    0.6344,    1.3632]
[    0.6403,    1.3691]
[    0.6267,    1.3555]
[    0.6435,    1.3723]
[    0.6329,    1.3617]
```

The diameter is even worse because now the set of solutions $A^{-1}b$ is included for all $A \in \mathtt{A}$ and $b \in \mathtt{b}$. However, it may equally well be attributed to overestimation by interval arithmetic. But it is possible to estimate the overestimation, as will be described in Section 10.6.

A drawback to this approach is that $\|I - RA\| < 1$ is mandatory. In order to be easily and reliably computable there is not much choice for the matrix norm. If the matrix is badly scaled this causes problems. It is possible to incorporate some kind of scaling in the verification process. The corresponding inclusion theorem is based on a lemma by Rump (1980). The following simple proof is by Alefeld.

Lemma 10.5. Let $\mathbf{Z}, \mathbf{X} \in \mathbb{IR}^n$ and $\mathbf{C} \in \mathbb{IR}^{n \times n}$ be given. Assume

$$\mathbf{Z} + \mathbf{C}\mathbf{X} \subset \mathrm{int}(\mathbf{X}), \tag{10.10}$$

where $\mathrm{int}(\mathbf{X})$ denotes the interior of $\mathbf{X}$. Then, for every $C \in \mathbf{C}$ the spectral radius $\varrho(C)$ of C is less than one.

Proof. Let $C \in \mathbf{C}$ be given and denote $\mathbf{Z} = [\underline{\mathbf{Z}}, \overline{\mathbf{Z}}]$. Then, following (9.10) yields

$$\mathbf{Z} + C\mathbf{X} = C \cdot \mathrm{mid}(X) + [\underline{\mathbf{Z}} - |C| \cdot \mathrm{rad}(X), \overline{\mathbf{Z}} + |C| \cdot \mathrm{rad}(X)] \subset \mathrm{int}(X),$$

and therefore

$$|C| \cdot \mathrm{rad}(X) < \mathrm{rad}(X).$$

Since $\mathrm{rad}(X)$ is a positive real vector, Perron–Frobenius theory yields $\varrho(C) \leq \varrho(|C|) < 1$. □

Based on this we can formulate a verification method, *i.e.*, a theorem the assumptions of which can be verified on digital computers. The theorem was formulated by Krawczyk (1969*a*) as a refinement of a given inclusion

of $A^{-1}b$, and modified as an existence test by Moore (1977). Our exposition follows Rump (1980).

Theorem 10.6. Let $A, R \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $\mathbf{X} \in \mathbb{IR}^n$ be given, and denote by I the $n \times n$ identity matrix. Assume

$$Rb + (I - RA)\mathbf{X} \subset \operatorname{int}(\mathbf{X}). \tag{10.11}$$

Then the matrices A and R are non-singular and $A^{-1}b \in Rb + (I - RA)\mathbf{X}$.

Proof. By Lemma 10.5 the spectral radius of $C := I - RA$ is less than one. Hence the matrices R and A are non-singular, and the iteration $x^{(k+1)} := z + Cx^{(k)}$ with $z := Rb$ converges to $(I - C)^{-1}z = (RA)^{-1}Rb = A^{-1}b$ for every starting point $x^{(0)} \in \mathbb{R}^n$. By (10.11), $x^{(k)} \in \mathbf{X}$ for any $k \geq 0$ if $x^{(0)} \in \mathbf{X}$, and the result follows. □

Condition (10.11) is the only one to be verified on the computer. To build an efficient verification algorithm, we need three improvements introduced in Rump (1980). First, condition (10.11) is still a sufficient criterion for given $\mathbf{X}$, and the problem remains of how to determine a suitable candidate. The proof suggests an interval iteration,

$$\mathbf{X}^{(k+1)} := Rb + (I - RA)\mathbf{X}^{(k)}, \tag{10.12}$$

so that with $\mathbf{X}^{(k+1)} \subseteq \operatorname{int}(\mathbf{X}^{(k)})$ the assumption of Theorem 10.6 is satisfied for $\mathbf{X} := \mathbf{X}^{(k)}$.

This is not sufficient, as trivial examples show. Consider a 1×1 linear system with $b = 0$, a poor 'approximate inverse' R so that $1 - RA = -0.6$ and a starting interval $\mathbf{X}^{(k)} := [-1, 2]$. Obviously the solution 0 is enclosed in every iterate, but $\mathbf{X}^{(k+1)}$ never belongs to $\operatorname{int}(\mathbf{X}^{(k)})$.

So, secondly, it needs a so-called *epsilon-inflation*, that is, to enlarge $\mathbf{X}^{(k)}$ intentionally before the next iteration. This was first used in Caprani and Madsen (1978); the term was coined and the properties of the method analysed in Rump (1980). One possibility is

$$\begin{aligned} \mathbf{Y} &:= \mathbf{X}^{(k)} \cdot [0.9, 1.1] + [-e, e], \\ \mathbf{X}^{(k+1)} &:= Rb + (I - RA)\mathbf{Y}, \end{aligned} \tag{10.13}$$

with some small constant e and to check $\mathbf{X}^{(k+1)} \subseteq \operatorname{int}(\mathbf{Y})$.

The third improvement is that it is often better to calculate an inclusion of the difference of the true solution to an approximate solution $\tilde{x}$. For linear systems this is particularly simple, by verifying

$$\mathbf{Y} := R(b - A\tilde{x}) + (I - RA)\mathbf{X} \subset \operatorname{int}(\mathbf{X}) \tag{10.14}$$

to prove $A^{-1}b \in \tilde{x} + \mathbf{Y}$. Nowadays these three improvements are standard tools for nonlinear and infinite-dimensional problems (see Sections 13, 15 and 16).

Table 10.4. Inclusion by Algorithm 10.7 for a 9×9 linear system with matrix as in (2.11).

```
intval X =
[    0.9999,    1.0001]
[    0.9999,    1.0001]
[    0.9998,    1.0002]
[    0.9955,    1.0050]
[    0.9544,    1.0415]
[    0.8363,    1.1797]
[    0.6600,    1.3095]
[    0.7244,    1.3029]
[    0.8972,    1.0935]
```

In summary, we obtain the following verification algorithm presented in Rump (1980) for general dense systems of linear equations with point or interval matrix and right-hand side.[19]

Algorithm 10.7. Verified bounds for the solution of a linear system:

```
function XX = VerifyLinSys(A,b)
  XX = NaN;                      % initialization
  R = inv(mid(A));               % approximate inverse
  xs = R*mid(b);                 % approximate solution
  C = eye(dim(A))-R*intval(A); % iteration matrix
  Z = R*(b-A*intval(xs));
  X = Z; iter = 0;
  while iter<15
    iter = iter+1;
    Y = X*infsup(0.9,1.1) + 1e-20*infsup(-1,1);
    X = Z+C*Y;                   % interval iteration
    if all(in0(X,Y)), XX = xs + X; return; end
  end
```

Theorem 10.8. If Algorithm 10.7 ends successfully for a given interval matrix $\mathbf{A} \in \mathbb{IR}^{n \times n}$ and interval right-hand side $\mathbf{b} \in \mathbb{IR}^n$, then the following is true. All matrices $A \in \mathbf{A}$ are non-singular, and the computed XX satisfies

$$\Sigma(\mathbf{A}, \mathbf{b}) := \{x \in \mathbb{R}^n \,:\, Ax = b \ \text{ for some } A \in \mathbf{A}, b \in \mathbf{b}\} \subseteq \mathtt{XX}. \qquad (10.15)$$

Proof. The proof follows by applying Theorem 10.6 to fixed but arbitrary $A \in \mathbf{A}$ and $b \in \mathbf{b}$ and the INCLUSION PROPERTY (5.16). □

Note that the result is valid for a matrix right-hand side $\mathbf{b} \in \mathbb{IR}^{n \times k}$ as well. In particular, it allows one to compute an inclusion of the inverse of a

[19] The routine in0(X,Y) checks X ⊂ int(Y) componentwise.

Table 10.5. Computing time of Algorithm 10.7 for dense point and interval linear systems, Gaussian elimination in floating-point arithmetic normed to 1.

Dimension	Point data	Interval data
100	7.3	8.2
200	6.4	8.2
500	6.8	9.9
1000	7.0	9.5
2000	7.7	10.4

matrix $A \in \mathbb{R}^{n\times n}$, and also of the set of inverses $\{A^{-1} : A \in \mathbf{A}\}$ including the proof of non-singularity of all $A \in \mathbf{A}$. It was shown by Poljak and Rohn (1993) that the latter is an NP-hard problem. Rohn (2009*a*) gives 40 necessary and sufficient criteria for non-singularity of an interval matrix.

Theorem 10.8 is superior to Theorem 10.2. For our model problem in (2.11) for $n = 9$ with the interval matrix $\mathbf{A}$ containing the true real matrix, for example, we obtain $\mathbf{C} := I - R\mathbf{A}$ with $\mathrm{norm}(\mathrm{mag}(\mathbf{C})) = 0.0651$ but $\varrho(\mathrm{mag}(\mathbf{C})) = 0.0081$. Accordingly the result of Algorithm 10.7, as displayed in Table 10.4, is superior to that in Table 10.3. Recall that the condition number $\|A^{-1}\|\|A\|$ of the midpoint matrix is again increased by taking the (narrowest) interval matrix including the matrix (2.11) for $n = 9$. For $n = 10$ we have $\mathrm{norm}(\mathrm{mag}(\mathbf{C})) = 10.29$ but $\varrho(\mathrm{mag}(\mathbf{C})) = 0.94$. Therefore Theorem 10.2 is not applicable, whereas with Theorem 10.8 an inclusion is still computed. Due to the extreme[20] condition number $8.2 \cdot 10^{16}$ of the midpoint matrix, however, the quality is very poor.

Epsilon-inflation is theoretically important as well. It can be shown that basically an inclusion will be computed by Algorithm 10.7 if and only if $\varrho(|I - RA|) < 1$. Note the similarity to the real iteration $x^{(k+1)} := Rb + (I - RA)x^{(k)}$, which converges if and only if $\varrho(I - RA) < 1$. In a practical application there is usually not too much difference between $\varrho(|I - RA|)$ and $\varrho(I - RA)$. Also note that the only matrix norms which can be easily estimated are the 1-norm, the ∞-norm and the Frobenius norm, for which the norm of C and $|C|$ coincides.

The main computational effort in Algorithm 10.7 is the computation of `R` and `C`. Using (9.10) and (9.14) this corresponds to the cost of 3 or 4 real matrix multiplications, respectively. The cost of Gaussian elimination is one third of one real matrix multiplication, so we can expect the verification algorithm to take 9 or 12 times the computing time of Gaussian elimination

[20] Computed with a symbolic package corresponding to infinite precision.

Table 10.6. Quality of inclusions $\mu/\mathrm{cond}(A)$ of Algorithm 10.7 for point data.

	$n = 100$	$n = 200$	$n = 500$	$n = 1000$	$n = 2000$
$\mathrm{cond}(A) = 10^{10}$	$6.7 \cdot 10^{-16}$	$2.1 \cdot 10^{-15}$	$6.3 \cdot 10^{-15}$	$2.0 \cdot 10^{-14}$	$6.0 \cdot 10^{-14}$
$\mathrm{cond}(A) = 10^{11}$	$5.8 \cdot 10^{-16}$	$1.5 \cdot 10^{-15}$	$8.4 \cdot 10^{-15}$	$2.1 \cdot 10^{-14}$	$4.6 \cdot 10^{-14}$
$\mathrm{cond}(A) = 10^{12}$	$8.1 \cdot 10^{-16}$	$1.9 \cdot 10^{-15}$	$7.7 \cdot 10^{-15}$	$1.5 \cdot 10^{-14}$	$5.0 \cdot 10^{-14}$
$\mathrm{cond}(A) = 10^{13}$	$5.2 \cdot 10^{-16}$	$2.1 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$2.3 \cdot 10^{-14}$	$6.4 \cdot 10^{-14}$
$\mathrm{cond}(A) = 10^{14}$	$8.4 \cdot 10^{-16}$	$3.0 \cdot 10^{-15}$	$7.5 \cdot 10^{-15}$	$9.6 \cdot 10^{-13}$	failed

for point or interval input matrices, respectively. This is confirmed by the measured computing time in Table 10.5, where the ratio is a little better than expected since BLAS routines for matrix multiplication are, in contrast to Gaussian elimination, likely to achieve nearly peak performance.

The sensitivity of the solution of a linear system is measured by the condition number. Computing in double precision corresponding to 16 decimal digits precision, for $\mathrm{cond}(A) = 10^k$ we cannot expect more than about $16-k$ correct digits of the solution. What is true for the pure floating-point algorithm is also true for the verification process. If μ is the median of the relative errors of an inclusion computed by Algorithm 10.7, then the ratio $\mu/\mathrm{cond}(A)$ should be constantly 10^{-16}. For randomly generated matrices with specified condition number, this is confirmed in Table 10.6.

The practical implementation `verifylss` in INTLAB of a verification algorithm for dense systems of linear equations is based on Algorithm 10.7.

10.6. Inner inclusion

The traditional condition number for a linear system $Ax = b$ is defined by

$$\kappa_{E,f}(A,x) := \lim_{\epsilon \to 0} \sup \left\{ \frac{\|\Delta x\|}{\epsilon \|x\|} : (A + \Delta A)(x + \Delta x) = b + \Delta b,\ \Delta A \in \mathbb{R}^{n \times n}, \right.$$
$$\left. \Delta b \in \mathbb{R}^n, \|\Delta A\| \le \epsilon \|E\|,\ \|\Delta b\| \le \epsilon \|f\| \right\}, \qquad (10.16)$$

and it is known (Higham 2002) that

$$\kappa_{E,f}(A,x) = \|A^{-1}\|\,\|E\| + \frac{\|A^{-1}\|\,\|f\|}{\|x\|}. \qquad (10.17)$$

This is the sensitivity of the solution with respect to infinitesimally small perturbations of the input data. The solution set $\Sigma(\mathbf{A}, \mathbf{b})$ in (10.15) characterizes the sensitivity for finite perturbations. As shown by Kreinovich, Lakeyev and Noskov (1993) and Rohn (1994), the computation of the

interval hull of $\Sigma(A,b)$ is an NP-hard problem, even when computed only to a certain precision.

Algorithm 10.7 computes an inclusion $\mathbf{X}$ of the solution set, thus upper bounds for the sensitivity of $A^{-1}b$ with respect to finite perturbations of A and b. Estimating the amount of overestimation of $\mathbf{X}$ yields lower bounds for the sensitivity, and this is possible by 'inner' inclusions of $\Sigma(\mathbf{A},\mathbf{b})$.

For interval input data, besides the solution set defined in (10.15), the 'tolerable (or restricted) solution set' (see Shary (2002))

$$\Sigma_{\forall\exists}(\mathbf{A},\mathbf{b}) := \{x \in \mathbb{R} \,:\, \forall A \in \mathbf{A}\ \exists b \in \mathbf{b} \text{ with } Ax = b\} = \{x \in \mathbb{R}^n \,:\, \mathbf{A}x \subseteq \mathbf{b}\} \tag{10.18}$$

and the 'controllable solution set'

$$\Sigma_{\exists\forall}(\mathbf{A},\mathbf{b}) := \{x \in \mathbb{R} \,:\, \forall b \in \mathbf{b}\ \exists A \in \mathbf{A} \text{ with } Ax = b\} = \{x \in \mathbb{R}^n \,:\, \mathbf{A}x \supseteq \mathbf{b}\} \tag{10.19}$$

are also of interest. We restrict our attention to what is sometimes called the 'united solution set' (10.15).

An inclusion $\mathbf{X} \in \mathbb{IF}$ of the solution set defined in (10.15) computed by Algorithm 10.7 may be wide due to overestimation and/or due to the sensitivity of the problem. The best-possible computable inclusion $\mathbf{Y} \in \mathbb{IF}$ is

$$\mathbf{Y} := \bigcap\{\mathbf{Z} \in \mathbb{IF} \,:\, \Sigma(\mathbf{A},\mathbf{b}) \subseteq \mathbf{Z}\} = \mathrm{hull}(\Sigma(\mathbf{A},\mathbf{b})). \tag{10.20}$$

The following theorem shows how to estimate the overestimation of $\mathbf{X}$ with respect to $\mathbf{Y}$. Such 'inner' inclusions were introduced by Neumaier (1987, 1989). Here we follow Rump (1990), who showed how to obtain them practically without additional computational effort.

Theorem 10.9. Let $\mathbf{A} \in \mathbb{IR}^{n\times n}, \mathbf{b} \in \mathbb{IR}^n$, $\tilde{x} \in \mathbb{R}^n$ and $\mathbf{X} \in \mathbb{IR}^n$ be given. Define

$$\mathbf{Z} := R(\mathbf{b} - \mathbf{A}\tilde{x}) \quad \text{and} \quad \mathbf{\Delta} := (I - R\mathbf{A})\mathbf{X}, \tag{10.21}$$

and suppose

$$\mathbf{Z} + \mathbf{\Delta} \subset \mathrm{int}(\mathbf{X}). \tag{10.22}$$

Then R and every matrix $A \in \mathbf{A}$ is non-singular, and $\mathbf{Y}$ as defined in (10.20) satisfies

$$\tilde{x} + \underline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \underline{\mathbf{Y}} \le \tilde{x} + \underline{\mathbf{Z}} + \overline{\mathbf{\Delta}} \quad \text{and} \quad \tilde{x} + \overline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \overline{\mathbf{Y}} \le \tilde{x} + \overline{\mathbf{Z}} + \overline{\mathbf{\Delta}}. \tag{10.23}$$

Proof. Let $A \in \mathbf{A}$ and $b \in \mathbf{b}$ be given. Applying Theorem 10.6 to the linear system $Ay = b - A\tilde{x}$ implies that A and R are non-singular, and $y = A^{-1}b - \tilde{x} \in \mathbf{Z} + \mathbf{\Delta}$. Since A and b are chosen arbitrarily and $\mathbf{Z} + \mathbf{\Delta} = [\underline{\mathbf{Z}} + \underline{\mathbf{\Delta}}, \overline{\mathbf{Z}} + \overline{\mathbf{\Delta}}]$, this proves $\Sigma(\mathbf{A},\mathbf{b}) \subseteq \tilde{x} + \mathbf{Z} + \mathbf{\Delta}$ and the first and last inequality in (10.23). Using (10.22), it follows that $\Sigma(\mathbf{A},\mathbf{b}) - \tilde{x} \subseteq \mathbf{X}$.

Table 10.7. Ratio of diameter of inner and outer inclusions for $n = 100$ and interval data.

Width of input	$n = 50$	$n = 100$	$n = 200$	$n = 500$	$n = 1000$
10^{-12}	1.000	1.000	1.000	1.000	1.000
10^{-8}	1.000	1.000	1.000	1.000	1.000
10^{-6}	1.000	0.998	0.991	0.955	0.954
10^{-5}	0.996	0.992	0.974	0.710	0.641
10^{-4}	0.962	0.684	0.502	failed	failed

Furthermore,

$$\begin{aligned}
&\{\tilde{x} + R(b - A\tilde{x}) \,:\, A \in \mathbf{A}, b \in \mathbf{b}\} \\
&\quad = \{A^{-1}b - (I - RA)(A^{-1}b - \tilde{x}) \,:\, A \in \mathbf{A}, b \in \mathbf{b}\} \\
&\quad \subseteq \Sigma(\mathbf{A}, \mathbf{b}) - \{(I - RA)(A^{-1}b - \tilde{x}) \,:\, A \in \mathbf{A}, b \in \mathbf{b}\} \\
&\quad \subseteq \Sigma(\mathbf{A}, \mathbf{b}) - \{(I - RA_1)(A_2^{-1}b - \tilde{x}) \,:\, A_1, A_2 \in \mathbf{A}, b \in \mathbf{b}\} \\
&\quad \subseteq \Sigma(\mathbf{A}, \mathbf{b}) - (I - R\mathbf{A})(\Sigma(\mathbf{A}, \mathbf{b}) - \tilde{x}) \\
&\quad \subseteq \Sigma(\mathbf{A}, \mathbf{b}) - \mathbf{\Delta},
\end{aligned}$$

where all set operations, even on interval quantities, are power set operations. If $U \subseteq V$ for two sets $U, V \in \mathbb{R}^n$, then $\mathrm{hull}(U) \subseteq \mathrm{hull}(V)$, and (9.4) and (9.6) imply

$$\tilde{x} + [\underline{\mathbf{Z}}, \overline{\mathbf{Z}}] = \tilde{x} + \mathbf{Z} \subseteq \mathbf{Y} - \mathbf{\Delta} = [\underline{\mathbf{Y}} - \overline{\mathbf{\Delta}}, \overline{\mathbf{Y}} - \underline{\mathbf{\Delta}}]. \tag{10.24}$$

This finishes the proof. □

Special care is necessary when applying this on the computer because lower *and upper* bounds are required in (10.23) for $\underline{\mathbf{Z}}$ and for $\overline{\mathbf{Z}}$. Using directed roundings and (9.6), this is not difficult. Note that for the outer and the inner bounds of $\mathbf{Y}$ only an outer inclusion of $\mathbf{\Delta}$ is necessary. Therefore, the inner bounds in Theorem 10.9 come with an extra effort of $\mathcal{O}(n^2)$ operations, so almost for free.

For interval input $\mathbf{A}, \mathbf{b}$, Theorem 10.9 yields an inner inclusion $\tilde{x} + [\underline{\mathbf{Z}} + \overline{\mathbf{\Delta}}, \overline{\mathbf{Z}} + \underline{\mathbf{\Delta}}]$, provided the radius of $\mathbf{\Delta}$ is no larger than the radius of $\mathbf{Z}$. Since $\mathbf{X}$ is the potential inclusion of the distance of $\tilde{x}$ to the true solution, $\mathbf{\Delta}$ is the product of two small quantities and can be expected to be of small radius.

In Table 10.7 we display the median of the componentwise ratios of the inner inclusion to the radius of the outer inclusion $\tilde{x} + \mathbf{Z} + \mathbf{\Delta}$. The matrices are generated by `midrad(randn(n), r * rand(n))` for `r` as given in the first column. This corresponds approximately to the relative precision of the matrix components. The right-hand side is computed by `b = A * ones(n, 1)`.

Table 10.8. Comparison of Monte Carlo and verification method for $n = 100$.

$K = 10$	$K = 100$	$K = 1000$	$K = 10000$
0.084	0.153	0.215	0.266

For increasing dimensions the ratio of the radii of inner and outer inclusions is not too far from 1 if the relative precision of the input data is not too large. For a relative precision of 10^{-4} the verification algorithm fails for dimension 500 and 1000, presumably because singular matrices slip into the input interval matrix. For these cases the spectral radius of $\mathrm{mag}(I - R\mathbf{A})$ is 1.15 and 2.18, respectively.

For relative precision 10^{-6} the radius of the inner inclusion is at least 95% of the outer inclusion, *i.e.*, the computed inclusion $\mathbf{X}$ by Algorithm 10.7 is not too far from the best possible inclusion $\mathbf{Y}$ as in (10.20). A Monte Carlo approach underestimates $\mathbf{Y}$ in principle. For fixed dimension $n = 100$ we choose K vertex matrices A and right-hand sides b, solve the linear system $Ax = b$ and form the narrowest interval vector $\mathbf{Z}$ containing all those solutions. We compare this with the solution $\mathbf{X}$ computed by Algorithm 10.7. In Table 10.8 the median of the quotient of the radii of $\mathbf{Z}$ and $\mathbf{X}$ is displayed for different values of K.

The computational effort for the Monte Carlo method is about $K/10$ times as much as for verification; however, improvement is slowly increasing with K. Note that the result of Algorithm 10.7 is of high quality and proved to be correct.

However, there is a major drawback to verification routines applied to interval data. All interval input is regarded as *independent* of each other. Simple dependencies such as symmetry are often difficult to take account of in verification algorithms. For linear systems this is possible, even for general linear dependencies of the input data, however, at the cost of roughly doubling the computational effort (see Section 10.7). In contrast, rather general, also nonlinear, dependencies are easily taken into account with a Monte Carlo approach.

The solution set was characterized by Oettli and Prager (1964) as

$$x \in \Sigma(\mathbf{A}, \mathbf{b}) \quad \Leftrightarrow \quad |\mathrm{mid}(\mathbf{A})x - \mathrm{mid}(\mathbf{b})| \leq \mathtt{rad}(\mathbf{A})|x| + \mathtt{rad}(\mathbf{b}).$$

Moreover, the shape of the intersection of $\Sigma(\mathbf{A}, \mathbf{b})$ with some orthant is precisely prescribed by rewriting the problem as an LP-problem, as follows.

Let $S \in \mathbb{R}^{n \times n}$ be a signature matrix, *i.e.*, a diagonal matrix with $|S_{ii}| = 1$ for $1 \leq i \leq n$. The set $\mathcal{O} := \{x \in \mathbb{R}^n \,:\, Sx \geq 0\}$ defines an orthant in $\mathbb{R}^n$.

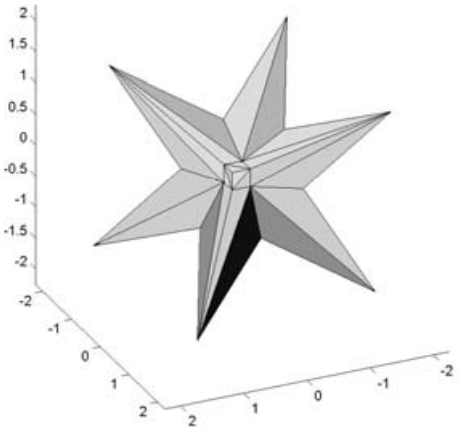

Figure 10.2. Solution set $\Sigma(\mathbf{A},\mathbf{b})$ for $\mathbf{A},\mathbf{b}$ as in (10.27).

For $\mathbf{A} := [mA - rA, mA + rA]$ and $x \in \mathcal{O}$ it follows that

$$\mathbf{A}x = [mA - rA, mA + rA]x = mA \cdot x + [-rA \cdot Sx, rA \cdot Sx] \qquad (10.25)$$

because $Sx \geq 0$. As explained in (9.4), there is no overestimation in the computation of $\mathbf{A}x$; interval and power set operations coincide. Therefore

$$\Sigma(\mathbf{A},\mathbf{b}) \cap \mathcal{O} = \{x \in \mathbb{R}^n : Sx \geq 0,\ (mA - rA \cdot S)x \leq \overline{\mathbf{b}},\ (mA + rA \cdot S)x \geq \underline{\mathbf{b}}\}. \qquad (10.26)$$

Hence the intersection of $\Sigma(\mathbf{A},\mathbf{b})$ with every orthant is obtained by solving some LP-problems, the number of which is polynomial in the dimension n. However, there are 2^n orthants, and $\Sigma(\mathbf{A},\mathbf{b})$ may have a non-empty intersection with each orthant although all matrices in $\mathbf{A}$ are non-singular. Here is the reason for the NP-hardness. Jansson (1997) gave an algorithm to compute exact bounds for $\Sigma(\mathbf{A},\mathbf{b})$ with computing time not necessarily exponential in the dimension, and Jansson and Rohn (1999) gave a similar algorithm for checking non-singularity of an interval matrix. Both algorithms, however, are worst-case exponential.

The INTLAB function `plotlinsol(A,b)` computes the graph of the solution set of a 2- or 3-dimensional interval linear system. For example (taken from Neumaier (1990, p. 96 *ff.*)),

$$\mathbf{A} = \begin{pmatrix} 3.5 & [0,2] & [0,2] \\ [0,2] & 3.5 & [0,2] \\ [0,2] & [0,2] & 3.5 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} [-1,1] \\ [-1,1] \\ [-1,1] \end{pmatrix} \qquad (10.27)$$

produce the solution set $\Sigma(\mathbf{A},\mathbf{b})$, as shown in Figure 10.2. Because of the

wide input intervals, the inclusion

```
intval X =
[   -8.6667,    8.6667]
[   -8.6667,    8.6667]
[   -8.6667,    8.6667]
```

computed by `verifylss` overestimates the interval hull $[-1.7648, 1.7648] \cdot (1,1,1)^T$ of the true solution set $\Sigma(\mathbf{A}, \mathbf{b})$, and the inner inclusion by Theorem 10.9 is empty. Note that the true solution set of the preconditioned linear system using the true inverse $R := \mathrm{mid}(A)^{-1}$ is

$$\mathrm{hull}\big(\Sigma(R\mathbf{A}, R\mathbf{b})\big) = [-8.6667, 8.6667] \cdot (1,1,1)^T,$$

so that the reason for the overestimation by `verifylss` is solely the preconditioning.

10.7. Data dependencies

In practice, matrices are often algebraically structured, for example symmetric or Toeplitz. Denote by M_n^{struct} a set of matrices such that $A, B \in M_n^{\mathrm{struct}} \subseteq \mathbb{R}^{n\times n}$ implies $A+B \in M_n^{\mathrm{struct}}$. Then Higham and Higham (1992*b*) generalize the usual condition number (10.16) into a structured condition number by

$$\kappa_{E,f}^{\mathrm{struct}}(A,x) := \tag{10.28}$$
$$\limsup_{\epsilon\to 0}\left\{ \frac{\|\Delta x\|}{\epsilon\|x\|} : (A+\Delta A)(x+\Delta x) = b+\Delta b,\ \Delta A \in M_n^{\mathrm{struct}},\ \Delta b \in \mathbb{R}^n, \|\Delta A\| \le \epsilon\|E\|,\ \|\Delta b\| \le \epsilon\|f\| \right\},$$

where the spectral norm is used. It includes general matrices (no structure), *i.e.*, $M_n^{\mathrm{struct}} = \mathbb{R}^{n\times n}$.

For symmetric matrices, Bunch, Demmel and Van Loan (1989) showed that the unstructured and structured condition numbers are equal, *i.e.*, among the worst-case perturbations for A is a symmetric one; for many other structures Rump (2003*b*) showed that there is also no – or not much – difference. For some structures, however, such as Toeplitz matrices, Rump and Sekigawa (2009) showed that there exist $A_n \in M_n^{\mathrm{Toep}}$ and $b \in \mathbb{R}^n$ such that $\kappa_{A_n,b}^{\mathrm{Toep}} / \kappa_{A_n,b} < 2^{-n}$.

Note that this happens only for a specific right-hand side: the *matrix* condition number for general and structured perturbations is the same for many structures. More precisely, define

$$\kappa^{\mathrm{struct}}(A) := \tag{10.29}$$
$$\limsup_{\epsilon\to 0}\left\{ \frac{\|(A+\Delta A)^{-1} - A^{-1}\|}{\epsilon\|A^{-1}\|} :\ \Delta A \in M_n^{\mathrm{struct}}, \|\Delta A\| \le \epsilon\|A\| \right\}.$$

Then Rump (2003*b*) showed that

$$\kappa^{\rm struct}(A) = \kappa(A) = \|A^{-1}\| \cdot \|A\| \tag{10.30}$$

for general, symmetric, persymmetric, skewsymmetric, Toeplitz, symmetric Toeplitz, Hankel, persymmetric Hankel, and circulant structures. Moreover, define the structured distance to the nearest singular matrix by

$$\delta^{\rm struct}(A) := \min\{\alpha : \ \Delta A \in M_n^{\rm struct}, \|\Delta A\| \le \alpha\|A\|, A + \Delta A \text{ singular}\}. \tag{10.31}$$

The (absolute) distance to the nearest singular matrix in the spectral norm is the smallest singular value, so that $\delta(A) = [\|A^{-1}\| \cdot \|A\|]^{-1} = \kappa(A)^{-1}$ for general perturbations. Rump (2003*b*) showed a remarkable generalization of this famous result by Eckart and Young (1936), namely

$$\delta^{\rm struct}(A) = \kappa^{\rm struct}(A)^{-1} \tag{10.32}$$

for all structures mentioned above.

As in Section 10.6, the sensitivity of a linear system with respect to finite structured perturbations can be estimated by inner inclusions. What is straightforward for Monte Carlo-like perturbations requires some effort in verification methods.

Let $\mathbf{A}$ be an interval matrix such that $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for all i, j. The *symmetric solution set* is defined by

$$\Sigma^{\rm sym}(\mathbf{A}, \mathbf{b}) := \{x \in \mathbb{R}^n \ : \ Ax = b \quad \text{for} \quad A^T = A \in \mathbf{A}, b \in \mathbf{b}\}.$$

Due to Jansson (1991), inclusions of the symmetric solution set can be computed. Consider the following example taken from Behnke (1989):

$$\mathbf{A} = \begin{pmatrix} 3 & [1,2] \\ [1,2] & 3 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} [10, 10.5] \\ [10, 10.5] \end{pmatrix}. \tag{10.33}$$

Computed inner and outer inclusions for the solution set $\Sigma(\mathbf{A}, \mathbf{b})$ and the symmetric solution set $\Sigma^{\rm sym}(\mathbf{A}, \mathbf{b})$ are shown in Figure 10.3. To compute outer and inner inclusions for the symmetric solution we adapt Theorem 10.9. For data without dependencies we know that

$$\mathbf{b} - \mathbf{A}\tilde{x} \ = \ \{b - A\tilde{x} : \ A \in \mathbf{A}, b \in \mathbf{b}\}, \tag{10.34}$$

by (9.4), and

$$R(\mathbf{b} - \mathbf{A}\tilde{x}) \ = \ \mathrm{hull}\big(\{R(b - A\tilde{x}) : \ A \in \mathbf{A}, b \in \mathbf{b}\}\big), \tag{10.35}$$

by (9.6), so $\mathbf{Z}$ is best possible. As we have seen, $\mathbf{\Delta}$ in (10.21) is usually small compared to $\mathbf{Z}$. So the main task is to obtain a sharp inclusion of

$$\mathbf{Z}^{\rm sym} := \mathrm{hull}\big(\{R(b - A\tilde{x}) : \ A^T = A \in \mathbf{A}, b \in \mathbf{b}\}\big). \tag{10.36}$$

In view of Theorem 8.1 we use the symmetry to rearrange the computation

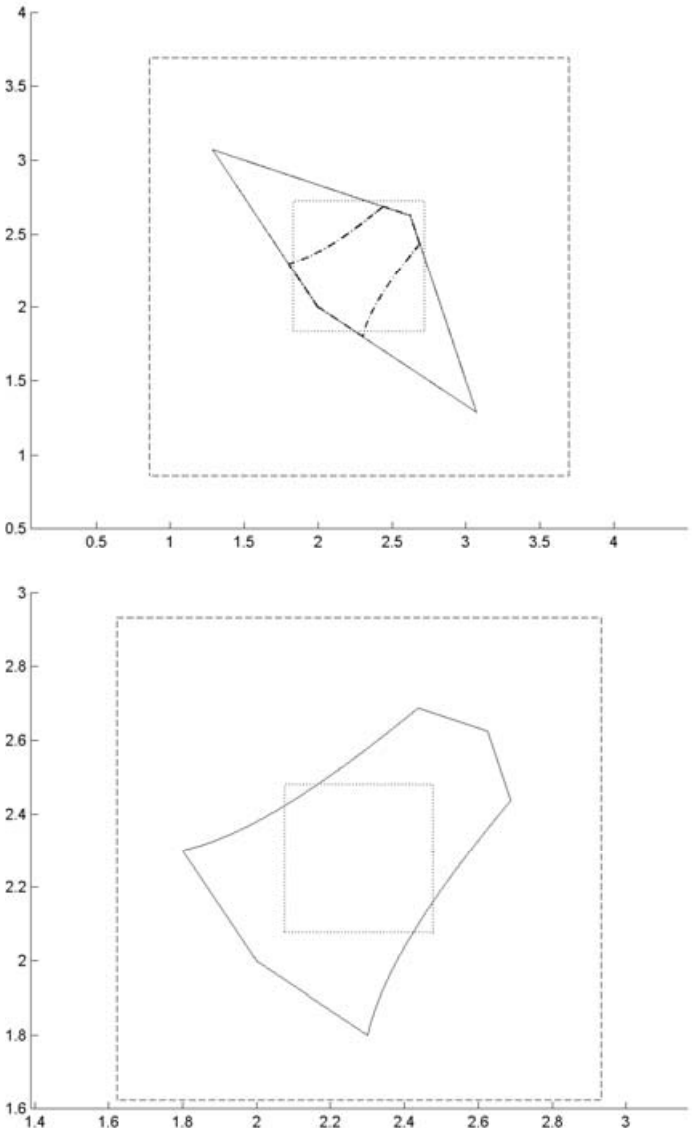

Figure 10.3. Inner and outer inclusions for the solution set $\Sigma(\mathbf{A}, \mathbf{b})$ and the symmetric solution set $\Sigma^{\mathrm{sym}}(\mathbf{A}, \mathbf{b})$ for $\mathbf{A}, \mathbf{b}$ as in (10.33).

so that each interval quantity occurs only once:

$$\big(R(b-A\tilde{x})\big)_i = \sum_{j=1}^{n} R_{ij}(b_j - A_{jj}\tilde{x}_j) - \sum_{\substack{j,k=1 \\ j<k}}^{n} (R_{ij}\tilde{x}_k + R_{ik}\tilde{x}_j)A_{jk}. \quad (10.37)$$

It follows that inserting $\mathbf{A}$ and $\mathbf{b}$ into (10.37) produces no overestimation, resulting in the componentwise (sharp) computation of $\mathbf{Z}_i^{\mathrm{sym}}$. Therefore we may proceed as in the proof of Theorem 10.9, to obtain the following.

Theorem 10.10. Let $\mathbf{A} \in \mathbb{IR}^{n\times n}$, $\mathbf{b} \in \mathbb{IR}^n$, $\tilde{x} \in \mathbb{R}^n$ and $\mathbf{X} \in \mathbb{IR}^n$ be given. Define

$$\mathbf{\Delta} := (I - R\mathbf{A})\mathbf{X}, \quad (10.38)$$

and let $\mathbf{Z}$ be computed componentwise by (10.37). Furthermore, suppose

$$\mathbf{Z} + \mathbf{\Delta} \subset \mathrm{int}(\mathbf{X}). \quad (10.39)$$

Then R and every matrix $A^T = A \in \mathbf{A}$ is non-singular, and

$$\mathbf{Y} := \mathrm{hull}\big(\Sigma^{\mathrm{sym}}(\mathbf{A}, \mathbf{b})\big)$$

satisfies

$$\tilde{x} + \underline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \underline{\mathbf{Y}} \le \tilde{x} + \underline{\mathbf{Z}} + \overline{\mathbf{\Delta}} \quad \text{and} \quad \tilde{x} + \overline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \overline{\mathbf{Y}} \le \tilde{x} + \overline{\mathbf{Z}} + \overline{\mathbf{\Delta}}. \quad (10.40)$$

Theorem 10.10 was used to compute the data in Figure 10.3. The idea of computing a sharp inclusion of $R(\mathbf{b} - \mathbf{A}\tilde{x})$ for symmetric $\mathbf{A}$ was extended by Rump (1994) to general linear dependencies in the matrix and the right-hand side as follows.

The following exposition borrows an idea by Higham and Higham (1992*a*), which they used to analyse the sensitivity of linear systems with data dependency. Denote by $\mathrm{vec}(A) \in \mathbb{R}^{mn}$ the columns of a matrix $A \in \mathbb{R}^{m\times n}$ stacked into one vector, and let $A \otimes B \in \mathbb{R}^{mp\times nq}$ be the Kronecker product of $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{pq}$.

Let $\varphi : \mathbb{R}^k \to \mathbb{R}^{n\times n}$ be a matrix-valued function depending linearly on parameters $p \in \mathbb{R}^k$, so that $A(p) := \varphi(p)$ implies

$$A(p)_{ij} = [c^{(ij)}]^T p \quad (10.41)$$

for vectors $c^{(ij)} \in \mathbb{R}^k$. If $\Phi \in \mathbb{R}^{n^2\times k}$ denotes the matrix with $\{(i-1)n+j\}$th row $[c^{(ij)}]^T$, then

$$\mathrm{vec}(A(p)) = \Phi p. \quad (10.42)$$

The set $M_n^{\mathrm{struct}} := \{A(p) : p \in \mathbb{R}^k\}$ defines a set of structured matrices. Obviously, $A, B \in M_n^{\mathrm{struct}}$ implies $A + B \in M_n^{\mathrm{struct}}$, so that the sensitivity with respect to infinitesimally small structured perturbations is characterized by (10.28), and can be computed according to Higham and Higham (1992*b*).

Table 10.9. Number of parameters for different structures.

Structure	Number of parameters
symmetric	$n(n+1)/2$
real skew-symmetric	$n(n-1)/2$
Toeplitz	$2n-1$
symmetric Toeplitz	n
circulant	n

Next consider finite perturbations generalizing componentwise structured perturbations discussed in Rump (2003*c*).

For fixed dimension, the matrix Φ depends only on the given structure. For example, symmetry is modelled by $\Phi^{\rm sym}$ with $k = n(n+1)/2$ rows and one non-zero entry 1 per row. For common structures the number k of parameters is shown in Table 10.9.

A right-hand side $b(p)$ linear depending on p is modelled in the same way by some matrix $\Psi \in \mathbb{R}^{n\times k}$ with $b(p) = \Psi p$. For an interval vector $\mathbf{p} \in \mathbb{R}^k$, define

$$A(\mathbf{p}) := \{A(p) : p \in \mathbf{p}\} \quad \text{and} \quad b(\mathbf{p}) := \{b(p) : p \in \mathbf{p}\}. \tag{10.43}$$

Then (9.6) implies

$$\Phi\mathbf{p} = \text{hull}(A(\mathbf{p})) \quad \text{and} \quad \Psi\mathbf{p} = \text{hull}(b(\mathbf{p})). \tag{10.44}$$

As for (10.37), one verifies, for $p \in \mathbb{R}^k$,

$$R\big(b(p) - A(p)\tilde{x}\big) = \big(R\Psi - (\tilde{x}^T \otimes R)\Phi\big)p. \tag{10.45}$$

Since $R\Psi - (\tilde{x}^T \otimes R)\Phi \in \mathbb{R}^{n\times k}$ is a point matrix, it follows by (9.4) that

$$\big(R\Psi - (\tilde{x}^T \otimes R)\Phi\big)\mathbf{p} = \text{hull}\big\{R\big(b(p) - A(p)\tilde{x}\big) : p \in \mathbf{p}\big\}. \tag{10.46}$$

As for Theorems 10.9 and 10.10, this is the foundation for calculating outer and inner inclusions for systems of linear equations, with general linear dependencies in the coefficients of the matrix and the right-hand side.

Theorem 10.11. For $k, n \in \mathbb{N}$, let $\Phi \in \mathbb{R}^{k\times n^2}$, $\Psi \in \mathbb{R}^{k\times n}$, $\mathbf{p} \in \mathbb{IR}^k$, $\tilde{x} \in \mathbb{R}^n$ and $\mathbf{X} \in \mathbb{IR}^n$ be given. Define

$$\mathbf{Z} := \big(R\Psi - (\tilde{x}^T \otimes R)\Phi\big)\mathbf{p} \quad \text{and} \quad \mathbf{\Delta} := (I - R\mathbf{A})\mathbf{X} \tag{10.47}$$

and

$$A(\mathbf{p}) := \{A : \text{vec}(A) = \Phi p,\ p \in \mathbf{p}\} \quad \text{and} \quad b(\mathbf{p}) := \{\Psi p : p \in \mathbf{p}\}. \tag{10.48}$$

Suppose

$$\mathbf{Z} + \mathbf{\Delta} \subset \mathrm{int}(\mathbf{X}). \tag{10.49}$$

Then R and every matrix $A \in A(\mathbf{p})$ is non-singular, and the hull of the structured solution set

$$\mathbf{Y} := \mathrm{hull}\big(\{x \in \mathbb{R}^n : Ax = b, A \in A(\mathbf{p}), b \in b(\mathbf{p})\}\big) \tag{10.50}$$

satisfies

$$\tilde{x} + \underline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \underline{\mathbf{Y}} \le \tilde{x} + \underline{\mathbf{Z}} + \overline{\mathbf{\Delta}} \quad \text{and} \quad \tilde{x} + \overline{\mathbf{Z}} + \underline{\mathbf{\Delta}} \le \overline{\mathbf{Y}} \le \tilde{x} + \overline{\mathbf{Z}} + \overline{\mathbf{\Delta}}. \tag{10.51}$$

Note again that special care is necessary in the computation of $\mathbf{Z}$ in (10.47): inner bounds for $\mathbf{Z}$ are necessary for inner bounds of $\mathbf{Y}$, whereas for both outer and inner bounds of $\mathbf{Y}$, outer bounds for $\mathbf{\Delta}$ are sufficient.

The exact, in general, nonlinear shape (see Figure 10.3) of the solution set for symmetric and other linear dependencies has been characterized by Alefeld, Kreinovich and Mayer (1997, 2003).

10.8. Sparse linear systems

For general dense linear systems, verification algorithms generally compute an inclusion if Gaussian elimination in floating-point arithmetic provides at least one correct digit. The computing time for the verification algorithm is within an order of magnitude of Gaussian elimination. Unfortunately, for general sparse linear systems such an algorithm is not known, one major open problem of verification algorithms.

For sparse linear systems Algorithm 10.7 is, in general, not suitable because the inverse of a sparse matrix is likely to be dense. For symmetric positive definite matrices, verified error bounds can be computed effectively using the following methods by Rump (1994). But already for indefinite symmetric problems, the known approaches may fail for reasonable condition number and/or require extensive computational time. Moreover, the field of verification algorithms based on iterative methods is basically a blank slate.

For didactic purposes, we let there be given a symmetric positive definite matrix A. Later it is seen that this significant assumption is *proved a posteriori* by the proposed algorithm.

Let $A \in \mathbb{R}^{n\times n}$ and $b, \tilde{x} \in \mathbb{R}^n$ be given. As for (10.8) we note that

$$\|A^{-1}b - \tilde{x}\|_2 \le \|A^{-1}\|_2 \cdot \|b - A\tilde{x}\|_2 = \frac{\|b - A\tilde{x}\|_2}{\sigma_n(A)}, \tag{10.52}$$

where $\|\cdot\|_2$ denotes the spectral norm and $\sigma_1(A) \ge \cdots \ge \sigma_n(A)$ denote the singular values of A. Thus a lower bound on the smallest singular value of A yields a bound on the error of $\tilde{x}$. We use the following well-known perturbation result given by Wilkinson (1965, p. 99 *ff.*).

Lemma 10.12. Let the eigenvalues of a symmetric matrix A be denoted by $\lambda_1(A) \geq \cdots \geq \lambda_n(A)$. Let symmetric $A, E \in \mathbb{R}^{n\times n}$ be given. Then

$$\lambda_i(A) + \lambda_n(E) \leq \lambda_i(A+E) \leq \lambda_i(A) + \lambda_1(E) \quad \text{for} \quad 1 \leq i \leq n. \tag{10.53}$$

Let $\alpha \in \mathbb{R}$ be an approximation of $\sigma_n(A)$, and set $\tilde{\alpha} := 0.9 \cdot \alpha$. Further, let $G \in \mathbb{R}^{n\times n}$ and define $E := A - \tilde{\alpha}I - G^TG$. Then Lemma 10.12 implies

$$\lambda_n(A - \tilde{\alpha}I) \geq \lambda_n(G^TG) + \lambda_n(E) \geq -\|E\|_2 \quad \text{or} \quad \lambda_n(A) \geq \tilde{\alpha} - \|E\|_2 := \mu. \tag{10.54}$$

If $\mu > 0$ it follows that A is symmetric positive definite and

$$\sigma_n(A) \geq \mu. \tag{10.55}$$

For practical applications, it suffices to compute an upper bound for $\|E\|_2$ which is readily obtained by evaluating $\|A - \tilde{\alpha}I - G^TG\|_1$ in interval arithmetic, or with the methods given in Section 10.11. The obvious choice for G is an approximate Cholesky factor of $A - \tilde{\alpha}I$. Note that only the input data is used corresponding to the UTILIZE INPUT DATA PRINCIPLE (5.13).

The Cholesky decomposition allows one to estimate all rounding errors effectively to obtain an *a priori* bound without calculating $A - \tilde{\alpha}I - G^TG$, resulting in an efficient verification method.

10.8.1. Verification of positive definiteness

Following Demmel (1989) we use *pure* floating-point arithmetic to verify positive definiteness of $A - \tilde{\alpha}I$ and to obtain a lower bound $\sigma_n(A) \geq \tilde{\alpha}$. It is another typical example of the design of a verification method following the DESIGN PRINCIPLE (DP) (1.1) and the UTILIZE INPUT DATA PRINCIPLE (5.13). It is much faster than (10.54), albeit with a more restricted range of applicability.

When executing Cholesky decomposition in floating-point arithmetic and, for the moment, barring underflow, successive applications of Theorem 2.1 imply, for the computed factor $\tilde{G}$,

$$\tilde{G}^T\tilde{G} = A + \Delta A \quad \text{with} \quad |\Delta A| \leq \operatorname{diag}(\gamma_2, \ldots, \gamma_{n+1})|\tilde{G}^T||\tilde{G}|, \tag{10.56}$$

where $\gamma_k := k\mathtt{u}/(1 - k\mathtt{u})$ is defined as in Theorem 2.1. Cholesky decomposition is a very stable algorithm, and no pivoting is necessary, so the size of $|\Delta A|$ can be estimated by $\tilde{G}$ and the input data. If the Cholesky decomposition runs to completion, then $a_{jj} \geq 0$, and one can show

$$\|\tilde{g}_j\|_2^2 = \tilde{g}_j^T\tilde{g}_j \leq a_{jj} + \gamma_{j+1}\,|\tilde{g}_j^T|\,|\tilde{g}_j| = a_{jj} + \gamma_{j+1}\,\tilde{g}_j^T\,\tilde{g}_j, \tag{10.57}$$

where $\tilde{g}_j$ denotes the jth column of $\tilde{G}$. Therefore

$$\|\tilde{g}_j\|_2^2 \leq (1 - \gamma_{j+1})^{-1}a_{jj} =: d_j,$$

Table 10.10. Smallest singular value for which `isspd` verifies positive definiteness.

Full matrices			Sparse matrices		
Dimension	$\alpha/\|A\|_2$	Time (sec)	Dimension	$\alpha/\|A\|_2$	Time (sec)
100	$1.5 \cdot 10^{-13}$	0.003	5000	$3.5 \cdot 10^{-11}$	0.06
200	$5.9 \cdot 10^{-13}$	0.009	10000	$1.2 \cdot 10^{-10}$	0.22
500	$3.6 \cdot 10^{-12}$	0.094	20000	$5.1 \cdot 10^{-10}$	1.0
1000	$1.4 \cdot 10^{-11}$	0.38	50000	$3.2 \cdot 10^{-9}$	11.9
2000	$5.6 \cdot 10^{-11}$	2.1	100000	$9.1 \cdot 10^{-9}$	84

and setting $\overline{d}_j := (\gamma_{j+1}(1-\gamma_{j+1})^{-1}\, a_{jj})^{1/2}$ and $k := \min(i,j)$, we have

$$|\Delta a_{ij}| \le \gamma_{k+1}\, |\tilde{g}_i^T|\, |\tilde{g}_j| \le \gamma_{i+1}^{1/2}\, \|\tilde{g}_i\|_2\, \gamma_{j+1}^{1/2}\, \|\tilde{g}_j\|_2 \le \overline{d}_i\, \overline{d}_j, \tag{10.58}$$

so that $\overline{d} := (\overline{d}_1, \ldots, \overline{d}_n)$ implies

$$\|\Delta A\| \le \| \, |\Delta A| \, \| \le \|\overline{d}\, \overline{d}^T\| = \overline{d}^T\, \overline{d} = \sum_{j=1}^{n} \gamma_{j+1}(1-\gamma_{j+1})^{-1}\, a_{jj}. \tag{10.59}$$

Lemma 10.13. Let symmetric $A \in \mathbb{F}^{n\times n}$ be given, and suppose Cholesky decomposition, executed in floating-point arithmetic satisfying (2.3), runs to completion. Then, barring over- and underflow, the computed $\tilde{G}$ satisfies

$$\tilde{G}^T\tilde{G} = A + \Delta A \quad \text{with} \quad \|\Delta A\| \le \sum_{j=1}^{n} \varphi_{j+1} a_{jj}, \tag{10.60}$$

where $\varphi_k := \gamma_k(1-\gamma_k)^{-1}$.

Note that corresponding to the Solvability principle of verification methods (1.2) positive definiteness (and indefiniteness) can be verified if the smallest singular value is not too small in magnitude. For a singular matrix, neither is possible in general.

With some additional work covering underflow, Lemma 10.13 leads to algorithm `isspd` in INTLAB: a quantity $\tilde{\alpha}$ based on $\sum_{j=1}^{n} \varphi_{j+1} a_{jj}$ in (10.60) and covering underflow can be computed *a priori*, and if the floating-point Cholesky decomposition applied to $A - \tilde{\alpha} I$ runs to completion, then A is proved to be positive definite.

For an interval input matrix $\mathbf{A}$, algorithm `isspd` verifies positive definiteness of every symmetric matrix within $\mathbf{A}$. But rather than working with the interval matrix, the following lemma is used.

Lemma 10.14. Let $\mathbf{A} = [M - R, M + R] \in \mathbb{IR}^{n\times n}$ with symmetric matrices $M, R \in \mathbb{R}^{n\times n}$ be given. If, for some $c \in \mathbb{R}$, $M - cI$ is positive definite for $\|R\|_2 \le c$, then every symmetric matrix $A \in \mathbf{A}$ is positive definite.

Table 10.11. Results and ratio of computing time for sparse linear systems.

Dimension	Double-precision residual		Improved residual	
	Inclusion of A_{11}^{-1}	Time ratio	Inclusion of A_{11}^{-1}	Time ratio
10000	0.302347266456_	5.0	0.3023472664558_	6.5
40000	0.30234727323__	4.7	0.302347273226_	6.3
250000	0.3023472737___	4.2	0.30234727367__	5.3

Moreover, for every positive vector $x \in \mathbb{R}^n$,

$$\|R\|_2 \le \sqrt{\mu} \quad \text{with} \quad \mu := \max_i \frac{(R^T(Rx))_i}{x_i}. \tag{10.61}$$

Proof. Lemma 10.12 implies that $|\sigma_i(A+\Delta)-\sigma_i(A)| \le \|\Delta\|_2$ for symmetric $A, \Delta \in \mathbb{R}^{n\times n}$, and $\|\Delta\|_2 \le \| \, |\Delta| \, \|_2$ proves that all $A \in \mathbf{A}$ are positive definite. Collatz (1942) showed that μ is an upper bound for the Perron root of (the non-negative matrix) $R^T R$, and the result follows. □

The applicability of Lemma 10.13 is tested by generating a numerically singular symmetric matrix A by $\mathtt{B} = \mathtt{randn}(\mathtt{n}, \mathtt{n}-1); \mathtt{A} = \mathtt{B} * \mathtt{B}'$; and finding the smallest $\alpha > 0$ for which positive definiteness of $A - \alpha I$ can be verified. The results for dense and sparse matrices (with about 5 non-zero elements per row) are displayed in Table 10.10; computations are performed on a 1.6 GHz laptop in MATLAB.

Estimating $\Delta A = \tilde{G}^T\tilde{G} - A$ in (10.60) using interval arithmetic takes more computing time, but positive definiteness can be verified for $\alpha/\|A\|_2$ almost of size `u`.

10.8.2. Solution of sparse linear systems with s.p.d. matrix

As before, let $\alpha \in \mathbb{R}$ be an approximation of $\sigma_n(A)$, and denote $\tilde{\alpha} := 0.9{\cdot}\alpha$. If positive definiteness of $A - \tilde{\alpha} I$ can be verified (using `isspd`), then $\|A\|_2 \ge \tilde{\alpha}$ and (10.52) can be applied.

This algorithm is implemented in `verifylss` in INTLAB for sparse input data. For a five-point discretization A of the Laplace equation on the unit square, we compute A_{11}^{-1} and display the results in Table 10.11.

The time for the MATLAB call $A\backslash b$ is normed to 1, and the results are shown with residual computed in double precision and with the improved residual by Algorithm 10.4. For this well-conditioned linear system, the floating-point approximation is also accurate. For some test matrices from the Harwell–Boeing test case library, the numbers are given in Table 10.12 (the name of the matrix, the dimension, number of non-zero elements, time for MATLAB $A\backslash b$ normed to 1, and median relative error of the inclusion are displayed using the improved residual).

Table 10.12. Computational results for Harwell–Boeing test matrices.

Matrix	Dimension	Non-zero elements	Time ratio	Median relative error of inclusion
bcsstk15	3948	117816	7.69	$1.3 \cdot 10^{-5}$
bcsstk16	4884	290378	11.41	$2.9 \cdot 10^{-7}$
bcsstk17	10974	428650	9.37	$4.1 \cdot 10^{-5}$
bcsstk18	11948	149090	9.50	$5.0 \cdot 10^{-5}$
bcsstk21	3600	26600	7.00	$2.0 \cdot 10^{-11}$
bcsstk23	3134	45178	7.85	$1.8 \cdot 10^{-3}$
bcsstk24	3562	159910	15.02	$6.8 \cdot 10^{-5}$
bcsstk25	15439	252241	7.49	$1.3 \cdot 10^{-3}$
bcsstk28	4410	219024	13.13	$3.5 \cdot 10^{-7}$

For matrices with small bandwidth, however, the verification is significantly slower than the pure floating-point computation. For example, computing inclusions of the first column of A^{-1} for the classic second difference operator on n points gives the results in Table 10.13. For such special matrices, however, specialized verification methods are available.

However, no stable and fast verification method for reasonably ill-conditioned matrices is known for symmetric (or general) matrices. There are some methods based on an LDL^T-decomposition along the lines of (10.54) discussed in Rump (1994); however, the method is only stable with some pivoting, which may, in turn, produce significant fill-in. The problem may also be attacked using the normal equations $A^T Ay = A^T b$, but this squares the condition number. To be more specific, we formulate the following challenge.

Challenge 10.15. Derive a verification algorithm which computes an inclusion of the solution of a linear system with a general symmetric sparse matrix of dimension 10 000 with condition number 10^{10} in IEEE 754 double precision, and which is no more than 10 times slower than the best numerical algorithm for that problem.

For more challenges see Neumaier (2002).

10.9. Special linear systems

If the matrix of a linear system has special properties, adapted methods may be used. For example, the discretization of an elliptic partial differential equation leads in general to an M-matrix, *i.e.*, a matrix of the form $A := cI - B$ with non-negative B and $\varrho(B) < c$.

Table 10.13. Results and ratio of computing time for the classic second difference operator.

Dimension	Time ratio	Median relative error of inclusion
10000	79.6	$1.24 \cdot 10^{-7}$
30000	94.0	$2.99 \cdot 10^{-6}$
100000	92.8	$6.25 \cdot 10^{-5}$
300000	151.9	$6.03 \cdot 10^{-4}$
1000000	147.0	$1.86 \cdot 10^{-2}$

In this case it is well known that $A^{-1} > 0$ and thus $\|A^{-1}\|_\infty = \|A^{-1}e\|_\infty$ for the vector e of all ones. Hence for $\tilde{y} \in \mathbb{R}^n$ it follows that

$$\|A^{-1}\|_\infty = \|\tilde{y} + A^{-1}(e - A\tilde{y})\|_\infty \leq \|\tilde{y}\| + \|A^{-1}\|_\infty \|e - A\tilde{y}\|_\infty$$

(see Ogita, Oishi and Ushiro (2001)), and therefore, along the lines of (10.8),

$$\|A^{-1}b - \tilde{x}\|_\infty \leq \|A^{-1}\|_\infty \|b - A\tilde{x}\|_\infty \leq \frac{\|\tilde{y}\|_\infty}{1 - \|e - A\tilde{y}\|_\infty} \|b - A\tilde{x}\|_\infty \quad (10.62)$$

for $\tilde{x} \in \mathbb{R}^n$. For given approximate solutions $\tilde{y}$ of $Ay = e$ and $\tilde{x}$ of $Ax = b$, this is an $\mathcal{O}(n^2)$ bound of good quality.

An interval matrix $\mathbf{A}$ is called an H-matrix if $\mathbf{A}v > 0$ for some positive vector v. Note that by (9.4) the interval and power set product $\mathbf{A}v$ coincide. Moreover, Ostrowski's comparison matrix $\langle \mathbf{A} \rangle \in \mathbb{R}^{n \times n}$ for interval matrices is defined by

$$\langle \mathbf{A} \rangle_{ij} := \begin{cases} \min\{|\alpha| \,:\, \alpha \in \mathbf{A}_{ij}\} & \text{for } i = j, \\ -\max\{|\alpha| \,:\, \alpha \in \mathbf{A}_{ij}\} & \text{otherwise.} \end{cases} \quad (10.63)$$

In fact, all matrices $A \in \langle \mathbf{A} \rangle$ are M-matrices. It follows that

$$|A^{-1}| \leq \langle \mathbf{A} \rangle^{-1} \quad \text{for all} \quad A \in \mathbf{A} \quad (10.64)$$

(see Neumaier (1990)).

If the midpoint matrix of $\mathbf{A}$ is a diagonal matrix, *i.e.*, off-diagonal intervals are centred around zero, then the exact solution set can be characterized. This remarkable result is known as the Hansen–Bliek–Rohn–Ning–Kearfott–Neumaier enclosure of an interval linear system. Recall that for a general linear system with interval matrix $\mathbf{A}$, it is an NP-hard problem to compute narrow bounds for the solution set defined in (10.15) (Rohn and Kreinovich 1995).

Theorem 10.16. Let $\mathbf{A} \in \mathbb{IR}^{n\times n}$ be an H-matrix, and let $\mathbf{b} \in \mathbb{IR}^n$ be a right-hand side. Define

$$u := \langle \mathbf{A}\rangle^{-1}|\mathbf{b}| \in \mathbb{R}^n, \quad d_i := (\langle \mathbf{A}\rangle^{-1})_{ii} \in \mathbb{R}, \tag{10.65}$$

and

$$\alpha_i := \langle \mathbf{A}\rangle_{ii} - 1/d_i, \quad \beta_i := u_i/d_i - |\mathbf{b}_i|. \tag{10.66}$$

Then the solution set $\Sigma(\mathbf{A},\mathbf{b}) = \{x \in \mathbb{R}^n \,:\, Ax = b \text{ for } A \in \mathbf{A}, b \in \mathbf{b}\}$ is contained in the interval vector $\mathbf{X}$ with components

$$\mathbf{X}_i := \frac{\mathbf{b}_i + [-\beta_i, \beta_i]}{\mathbf{A}_{ii} + [-\alpha_i, \alpha_i]}. \tag{10.67}$$

Moreover, if the midpoint matrix of $\mathbf{A}$ is diagonal, then

$$\mathrm{hull}(\Sigma(\mathbf{A},\mathbf{b})) = \mathbf{X}.$$

In practice, Theorem 10.16 is applied to a preconditioned linear system with matrix $R\mathbf{A}$ and right-hand side $R\mathbf{b}$, where R is an approximation of the inverse of $\mathrm{mid}(\mathbf{A})$.

It is likely that the midpoint of $R\mathbf{A}$ is not far from the identity matrix. However, even using exact arithmetic with $R := \mathrm{mid}(\mathbf{A})^{-1}$ so that $\mathrm{mid}(R\mathbf{A}) = I$, the solution set $\Sigma(R\mathbf{A}, R\mathbf{b})$ is only a superset of $\Sigma(\mathbf{A},\mathbf{b})$ due to data dependencies. So this approach offers no means to attack the original NP-hard problem.

This approach is slower than Algorithm 10.7; therefore it is used in `verifylss` in INTLAB as a 'second stage' if the verification using Algorithm 10.7 fails.

10.10. The determinant

As another example for the DESIGN PRINCIPLE OF VERIFICATION METHODS (1.1), consider the determinant of a matrix.

About the worst one can do is to perform Gaussian elimination in interval arithmetic (IGA). Instead one may proceed as follows:

```
[L,U,p] = lu(A,'vector');        % approximate factorization of A
Linv = inv(L); Uinv = inv(U);    % approximate preconditioners
C = Linv*(A(p,:)*intval(Uinv)); % inclusion of preconditioned matrix
D = prod(gershgorin(C));         % inclusion of det(Linv*A(p,:)*Uinv)
```

The first statement calculates approximate factors `L`, `U` such that $\mathtt{A}(\mathtt{p},:) \approx \mathtt{LU}$ for a permutation vector `p`. Multiplying $\mathtt{A}(\mathtt{p},:)$ from the left and right by approximate inverses of `L` and `U` results approximately in the identity matrix, the determinant of which can be estimated by the (complex interval) product of the Gershgorin circles.

Most of the computations are performed in floating-point arithmetic: only the computation of `C` and `D` is performed in interval arithmetic. A typical result for a 1000×1000 random matrix is

```
intval D =
<   1.00000000000008 +  0.00000000000000i,  0.00000062764860>
```

where the complex result is displayed by midpoint and radius. Since the input matrix was real, so must be the determinant resulting in

$$\det(\texttt{Linv} * \texttt{A(p,:)} * \texttt{Uinv}) \in [0.99999937, 1.00000063].$$

Now $\det(\texttt{L}) = \det(\texttt{Linv}) = 1$ and $\det(A)$ (or, better, $\log\det(A)$ to avoid over- or underflow) is easily and accurately computed.

Note that the approach applies to interval matrices as well, by computing an approximate decomposition of the midpoint matrix and otherwise proceeding as before.

10.11. The spectral norm of a matrix

Any non-trivial vector $x \in \mathbb{R}^n$ yields a lower bound of $\|A\|_2$ by evaluating $\|Ax\|_2/\|x\|_2$ in interval arithmetic, so the best numerical algorithm at hand may be used to compute a suitable x. However, an upper bound seems non-trivial.

If A is symmetric and $\alpha \approx \|A\|_2$ is given, then Lemma 10.13 may be applied to verify that $\tilde{\alpha}I - A$ and $\tilde{\alpha}I + A$ are positive definite for some $\tilde{\alpha} > \alpha$, thus verifying $\|A\|_2 \le \tilde{\alpha}$.

Let a general matrix A be given, together with an approximation $\alpha \approx \|A\|_2$. Using perturbation bounds similar to Lemma 10.12, it is not difficult to compute an inclusion of a singular value of A near α. However, there is no proof that this is the largest singular value.

But in this case there is no problem using $A^H A$ in the above approach, *i.e.*, verifying that $\tilde{\alpha}^2 I - A^H A$ is positive definite to prove $\|A\|_2 \le \tilde{\alpha}$; the squared condition number has no numerical side effect.

The cost of a verified upper bound of $\|A\|_2$, however, is $\mathcal{O}(n^3)$, whereas a few power iterations on $A^H A$ require some $\mathcal{O}(n^2)$ operations and usually lead to an accurate approximation of $\|A\|_2$. For a verified upper bound, the standard estimations $\|A\|_2 \le \sqrt{\|A\|_1 \|A\|_\infty}$ or $\|A\|_2 \le \|A\|_F$ are sometimes weak. Also, $\|A\|_2 \le \||A|\|_2$ together with (10.61) is often weak.

Challenge 10.17. Given $A \in \mathbb{R}^{n\times n}$, derive a verification algorithm to compute an upper bound for $\|A\|_2$ with about 1% accuracy in $\mathcal{O}(n^2)$ operations.

A verified lower bound is easy to compute in $\mathcal{O}(n^2)$ operations, but I find it hard to believe that there is such a method for an upper bound of $\|A\|_2$.

11. Automatic differentiation

For enclosing solutions of systems of nonlinear equations we need to approximate the Jacobian of nonlinear functions and to compute the range of the Jacobian over some interval.

The method of 'automatic differentiation' accomplishes this. Because this is mandatory for the following, we want to make at least a few remarks here. For a thorough discussion see Rall (1981), Corliss *et al.* (2002) and the *Acta Numerica* article by Griewank (2003).

11.1. Gradients

The method was found and forgotten several times, starting in the 1950s. When giving a talk on automatic differentiation in the 1980s, the audience would usually split in two groups, one not understanding or believing in the method and the other knowing it. The reason is that it seems, at first sight, not much more than the classical differentiation formulas $(uv)' = u'v + uv'$ or $g(f(x))' = g'(f(x))f'(x)$.

One way to understand it is similar to the concept of differential fields. Define a set $\mathcal{D}$ of pairs $(a, \alpha) \in \mathbb{R}^2$, and define operations $+, -, \cdot, /$ on $\mathcal{D}$ by

$$\begin{aligned} (a,\alpha) \pm (b,\beta) &:= (a + b, \alpha \pm \beta), \\ (a,\alpha) \cdot (b,\beta) &:= (a \cdot b, \alpha b + a\beta), \\ (a,\alpha)/(b,\beta) &:= (a/b, (\alpha - a\beta/b)/b), \end{aligned} \tag{11.1}$$

with non-zero denominator assumed in the case of division. Let a differentiable function $f : \mathbb{R} \to \mathbb{R}$ be given by means of an arithmetic expression $\mathfrak{A}$ in one independent variable x, so that $\mathfrak{A}(x) = f(x)$.

When replacing constants $c \in \mathbb{R}$ in $\mathfrak{A}$ by $(c, 0)$, then evaluating $\mathfrak{A}((\tilde{x}, 1))$ for some $\tilde{x} \in \mathbb{R}$ using (11.1) yields a pair (r, ρ) with property $f(\tilde{x}) = r$ and $f'(\tilde{x}) = \rho$, provided no division by 0 has occurred.

Using a programming language with an operator concept, the implementation is particularly simple, basically implementing (11.1). Standard functions are easily added, for example,

$$\begin{aligned} \mathrm{e}^{(b,\beta)} &:= (\mathrm{e}^b, \beta\,\mathrm{e}^b), \\ \cos(b,\beta) &:= (\cos(b), -\beta\sin(b)). \end{aligned} \tag{11.2}$$

Replacing all operations again by the corresponding interval operations and successively applying the inclusion property (5.16), it is clear that an inclusion of the range of a function and its derivative over some interval $\mathbf{X}$ is obtained as well. For example, for the function f given in (8.4), an inclusion of the range of f and f' over $\mathbf{X} := [2.4, 2.5]$ is obtained by

```
f = inline('sin(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1');
Y = f(gradientinit(intval('[2.4,2.5]')))
```

```
intval gradient value Y.x =
[   -0.2435,    0.3591]
intval gradient derivative(s) Y.dx =
[   -1.1704,   -0.0736]
```

The function `gradientinit(x)` initializes the gradient operator, *i.e.*, replaces constants c by $(c, 0)$ and replaces the argument x by $(x, 1)$. It follows that $-1.1704 \leq f'(x) \leq -0.0736$ for all $x \in [2.4, 2.5]$. Note that the notation `intval('[2.4,2.5]')` is necessary to obtain a true inclusion of $\mathbf{X}$ because $2.4 \notin \mathbb{F}$. As always, the inclusions are rigorous but may be subject to overestimation.

In an environment with operator concept such as MATLAB we can conveniently define a function and, depending on the input argument, obtain an approximation or an inclusion of the range of the function and the derivative. From the definitions (11.1) and (11.2) it is clear that the computing time for the function value and its derivative is less than about 5 times the computing time for only the function value.

Some care is necessary if a program contains branches. For example,

```
function y = f(x)
  if x==3, y=9; else y=x^2; end
```

is an unusual but correct implementation for the function $f(x) = x^2$. A straightforward automatic differentiation program will, however, deliver $f'(3) = 0$.

Applying the discussed principles to n independent variables $x_1, \ldots, x_n$ successively, an approximation and also an inclusion of the gradient or the Hessian of a function $f : \mathbb{R}^n \to \mathbb{R}$ is obtained. This is called the forward mode. The computing time for the gradient, however, is up to $5n$ times that for a function evaluation.

11.2. Backward mode

A major breakthrough of automatic differentiation was the so-called backward mode, in which the time to compute the function *and* gradient is not more than 5 times as much as only the evaluation of the function. This is independent of the dimension n. The idea is as follows.

Assume we are given an arithmetic expression $\mathfrak{A}$ to evaluate a function $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$. Denote the result of each intermediate operation in $\mathfrak{A}$ by x_i for $n+1 \leq i \leq n+m$, so that the evaluation consists of m steps and x_{n+m} is the final result $f(x)$. Collecting the initial vector x and the vector of intermediate results in one vector, the evaluation of $\mathfrak{A}(x)$ corresponds to successive computation of

$$y^{(k)} = \Phi_k(y^{(k-1)}) \quad \text{for} \quad k = 1, \ldots, m, \tag{11.3}$$

with the initial vector $y^{(0)} = (x, 0)^T \in \mathbb{R}^{n+m}$ and $x_{n+k} = e_{n+k}^T y^{(k)}$. The

functions $\Phi_k : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ correspond to a basic arithmetic operation or standard function on already computed quantities. In other words, Φ_k depends only on components $1, \ldots, n+k-1$ of $y^{(k-1)}$. Therefore

$$f(x) = e_{n+m}^T \cdot \Phi_m \circ \ldots \circ \Phi_1 \left(\begin{pmatrix} x \\ 0 \end{pmatrix} \right), \tag{11.4}$$

with 0 denoting a vector of m zeros, and

$$\nabla f(x) = e_{n+m}^T \cdot \Phi_m'(y^{(m-1)}) \cdots \Phi_1'(y^{(0)}) \cdot \begin{pmatrix} I \\ 0 \end{pmatrix}, \tag{11.5}$$

with 0 denoting the $m \times n$ zero matrix. The Jacobians $\Phi_k'(y^{(k-1)}) \in \mathbb{R}^{(n+m)\times(n+m)}$ have a very simple structure corresponding to the operations they are associated with. Note that m corresponds to the number of intermediate steps in the evaluation of $\mathfrak{A}$, and n corresponds to the number of unknowns.

One can see that the forward mode corresponds to the evaluation of (11.5) from right to left, so that each operation is a $(p \times p)$ times $(p \times n)$ matrix–matrix multiplication with $p := n + m$. However, when evaluating (11.5) from left to right, each operation is a $(p \times p)$ matrix multiplied by a p-vector from the left. This is the backward mode.

The remarkable speed of backward automatic differentiation comes at a price. While the forward mode is straightforward to implement with an operator concept at hand, the implementation of the backward mode is involved. However, packages are available that transform a computer program for evaluating a function into another program for evaluating the function together with the gradient and/or Hessian (in the fast backward mode); see, for example, Bischof, Carle, Corliss and Griewank (1991). For the range estimation of derivatives, however, forward differentiation sometimes gives much better enclosures than those from the backward mode.

11.3. Hessians

Automatic differentiation in forward mode for gradients and Hessians is implemented in INTLAB. Consider, for example, the function $f : \mathbb{R}^2 \to \mathbb{R}$:

$$\begin{aligned} f(x,y) := \exp(\sin(50x)) + \sin(60\exp(y)) + \sin(70\sin(x)) + \sin(\sin(80y)) \\ - \sin(10(x+y)) + (x^2+y^2)/4. \end{aligned} \tag{11.6}$$

It was Problem 4 of the 10×10-digit challenge by Trefethen (2002) to compute the global minimum of this function over $\mathbb{R}^2$. The INTLAB code

```
f = inline('exp(sin(50*x(1))) + sin(60*exp(x(2)))
       + sin(70*sin(x(1))) + sin(sin(80*x(2)))
       - sin(10*(x(1)+x(2))) + (x(1)^2+x(2)^2)/4');
X = verifynlss(f,[-0.02;0.2],'h')
Y = f(hessianinit(X))
```

evaluates the function value, gradient and Hessian of f over the given `X`, resulting in

```
intval X =
  -0.02440307969437
   0.21061242715535
intval Hessian value Y.x =
  -3.3068686474752_
intval Hessian first derivative(s) Y.dx =
  1.0e-011 *
  -0.0_____________   0._______________
intval Hessian second derivative(s) Y.hx =
  1.0e+003 *
   5.9803356010662_   0.09578721471459
   0.09578721471459   9.895778741947__
```

The call `X = verifynlss(f,[-0.02;0.2],'h')` proves that `X` is an inclusion of a stationary point (see Section 13), and by Gershgorin's theorem, every 2×2 matrix included in the Hessian is positive definite. This proves that `X` contains a strict local minimizer.

Consider the test problem

$$f(x) := \sum_{i=1}^{n-10} \frac{x_i}{x_{i+10}} + \sum_{i=1}^{n} (x-1)^2 - \sum_{i=1}^{n-1} x_i x_{i+1} = \text{ Min!}, \tag{11.7}$$

with initial guess $\tilde{x} := (1, \ldots, 1)^T$. Given `f` implementing the function in (11.7), the code

```
n = 2000;
X = verifynlss(@f,ones(n,1),'h');
H = f(hessianinit(X));
isMin = isspd(H.hx)
```

computes for $n = 2000$ an inclusion `X` of a solution of the nonlinear system $\nabla f(x) = 0$, *i.e.*, of a stationary point $\hat{x}$ of f, to at least 10 decimal places. Using `isspd` as described in Section 10.8, the result `isMin` $= 1$ verifies that every matrix $A \in$ `H.hx` is positive definite, in particular the Hessian of f at $\hat{x}$. Therefore, f has a strict (local) minimum in `X`. The Hessian has bandwidth 10, and the Gershgorin circles contain 0.

11.4. Taylor coefficients

To obtain higher-order derivatives, Taylor coefficients can be computed along the previous lines. Here we follow Rall (1981); see also Moore (1966, Section 11). Let two functions $f, g : \mathbb{R} \to \mathbb{R}$ with $f, g \in \mathcal{C}^K$ be given, and denote their respective Taylor coefficients at some $\tilde{x} \in \mathbb{R}$ by

$$a_k := \frac{1}{k!} f^{(k)}(\tilde{x}) \quad \text{and} \quad b_k := \frac{1}{k!} g^{(k)}(\tilde{x}),$$

for $0 \le k \le K$. Then the Taylor coefficients of selected composite functions are as follows:

$$\begin{array}{ll} \text{operation} & \text{Taylor coefficient} \\ c = f \pm g & c_k = a_k \pm b_k \\ c = f \cdot g & c_k = \sum_{j=0}^{k} a_j b_{k-j} \\ c = f/g & c_k = \frac{1}{b_0}\big(a_k - \sum_{j=1}^{k} b_j c_{k-j}\big) \\ c = \exp(f) & c_k = \frac{1}{k}\sum_{j=1}^{k} j a_j c_{k-j}. \end{array} \tag{11.8}$$

As before, assume a function f is given by means of an arithmetic expression. Then, initializing constants c by $(c, 0, \ldots, 0)$ and the independent variable x by $(\tilde{x}, 1, 0, \ldots, 0)$, and replacement of each operation or standard function by the corresponding Taylor operation or standard function, result in a vector $(r_0, \ldots, r_K)$ of Taylor coefficients of f at $\tilde{x}$.

In INTLAB, Taylor operations for all functions listed in (7.3) are implemented. Again, replacement of the argument by an interval $\mathbf{X}$ computes inclusions of the Taylor coefficients $\frac{1}{k!}f^{(k)}(\tilde{x})$ for $\tilde{x} \in \mathbf{X}$. For example, for the function f given in (8.4), an inclusion of the range of the Taylor coefficients up to order 4 over $\mathbf{X} := [2.4, 2.5]$ is obtained by

```
f = inline('sin(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1');
Y = f(taylorinit(intval('[2.4,2.5]'),4))
intval Taylor value Y.t =
[   -0.2435,    0.3591]
[   -1.0692,   -0.1034]
[   -0.3933,    1.1358]
[   -0.5512,    1.7307]
[   -2.2706,    1.0008]
```

Note that the inclusion $[-1.0692, -0.1034]$ for $f'([2.4, 2.5])$ is narrower than before.

Along similar lines, 'automatic Lipschitz estimates' can also be defined.

11.5. Slopes

Yet another approach uses automatic slopes, introduced by Krawczyk and Neumaier (1985). For $f : \mathbb{R}^m \to \mathbb{R}^n$, let $\tilde{x} \in \mathbb{R}^m$ and $\mathbf{X} \in \mathbb{IR}^m$ be given. The triple $(\mathbf{C}, \mathbf{R}, \mathbf{S}) \in \mathbb{IR}^n \times \mathbb{IR}^n \times \mathbb{IR}^{n\times m}$ of 'centre, range and slope' is a slope expansion with respect to f, $\tilde{x}$ and $\mathbf{X}$ if $f(\tilde{x}) \in \mathbf{C}$, $\{f(x) : x \in \mathbf{X}\} \subseteq \mathbf{R}$, and

$$f(x) \in f(\tilde{x}) + \mathbf{S}(x - \tilde{x}) \quad \text{for all} \quad x \in \mathbf{X}. \tag{11.9}$$

Note that $\tilde{x} \in \mathbf{X}$ is not necessary. An automatic slope package, which is contained in INTLAB, initializes constants c by the point interval $(c, c, 0)$, and the ith independent variable by $(\tilde{x}_i, \mathbf{X}_i, e_i)$. This satisfies (11.9), and

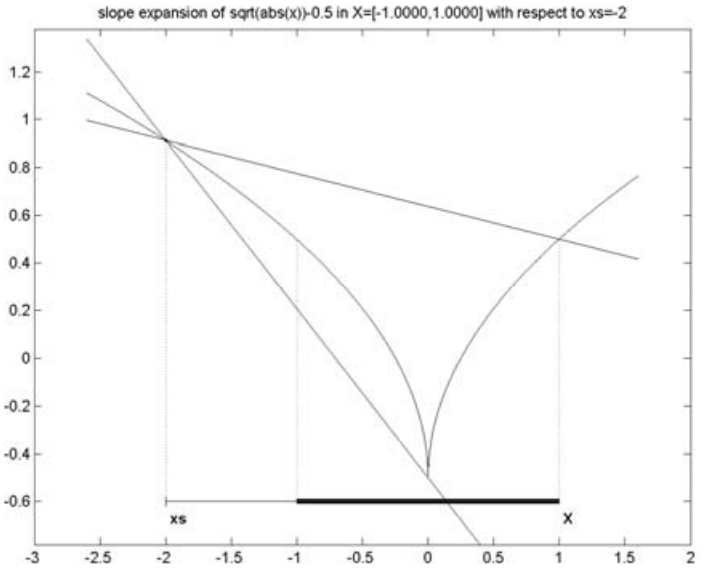

Figure 11.1. Visualization of slope for $f(x) = \sqrt{|x|} - 0.5$ and $\tilde{x} = -2$, $\mathbf{X} = [-1, 1]$.

automatic slopes are computed by defining arithmetic operations and standard functions appropriately.

As an example for $f(x) = \sqrt{|x|} - 0.5$ and $\tilde{x} = -2$, $\mathbf{X} = [-1, 1]$, the slope expansion is computed by

```
f=inline('sqrt(abs(x))-0.5'); y=f(slopeinit(-2,infsup(-1,1)))
slope intval center y.c =
    0.9142
slope intval range y.r =
[   -0.5000,    0.5000]
slope intval slope y.s =
[   -0.7072,   -0.1380]
```

and visualized in Figure 11.1.[21] The derivative over $\mathbf{X}$ is $[-\infty, \infty]$, but the slope is finite. Slopes offer some possibility for computing inclusions when the Jacobian contains singular matrices; the practical use, however, seems to be limited.

11.6. Range of a function

Gradients and slopes may be used to improve bounds for the range of a function over narrow intervals. For, given $f : \mathbb{R} \to \mathbb{R}$ and $\mathbf{X} = [m - r, m + r]$

[21] Generated by `slopeplot('sqrt(abs(x))-0.5',-2,infsup(-1,1),[],10000)`.

Table 11.1. Range estimation and overestimation for the function f as in (8.4) and $\mathrm{mid}(\mathbf{X}) = 2$.

$\mathrm{rad}(\mathbf{X})$	True range $\mathbf{R}$	$\mathrm{d}(\mathbf{X1})/\mathrm{d}(\mathbf{R})$	$\mathrm{d}(\mathbf{X2})/\mathrm{d}(\mathbf{R})$	$\mathrm{d}(\mathbf{X3})/\mathrm{d}(\mathbf{R})$
10^{-6}	[0.3904, 0.3905]	5.86	1.00	1.00
10^{-4}	[0.3903, 0.3906]	5.86	1.00	1.00
10^{-2}	[0.3839, 0.3969]	5.86	1.21	1.03
0.1	[0.3241, 0.4520]	5.81	3.56	1.31
0.5	[0.0794, 0.5656]	4.22	18.1	3.16
1	[−0.0425, 0.5656]	3.49	148	3.49
2	[−0.2959, 0.5656]	3.17	2041	3.17

$\in \mathbb{IR}$, we obviously have

$$\mathrm{Range}(f, \mathbf{X}) := \{f(x) : x \in \mathbf{X}\} \subseteq f(m) + f'(\mathbf{X})[-r, r]. \tag{11.10}$$

Hansen (1969) showed that the overestimation of this 'centred form' converges quadratically to zero for small radii of $\mathbf{X}$. However, the overestimation also increases quadratically for larger radii.

Another possibility for bounding $\mathrm{Range}(f, \mathbf{X})$ uses slopes, as described in Section 11.5. The following code computes inclusions for $\mathrm{Range}(f, \mathbf{X})$ by directly using interval operations, by the centred form (11.10) and by slopes:

```
X1 = f(X);                      % naive interval arithmetic
y = f(gradientinit(m));         % gradient inclusion of f(m)
Y = f(gradientinit(X));         % gradient inclusion of f'(X)
X2 = y.x + Y.dx*infsup(-r,r);   % inclusion by centred form
y = f(slopeinit(m,X));          % slope w.r.t. (m,X)
X3 = y.r;                       % inclusion by slopes
```

We compute the overestimation by means of the ratio of diameters of the computed inclusion and the true range for our model function f in (8.4). Table 11.1 displays the results for input interval $\mathbf{X}$ with $\mathrm{mid}(\mathbf{X}) = 2$ and different radii $\mathrm{rad}(\mathbf{X})$. For small radii both the centred form and the slope inclusion are almost optimal, whereas naive interval evaluation shows some overestimation. For larger radii it is the other way around for the centred form, whereas direct evaluation and the slope inclusion show moderate overestimation.

For arbitrarily wide input interval, the range of the sine function is always bounded by $[-1, 1]$. While oscillations may be a problem for numerical approximations, they can be advantageous for interval evaluation.

The picture changes completely when changing the first sine function in the definition (8.4) of f into the hyperbolic sine, which is then the function

Table 11.2. Range estimation and overestimation for the function g as in (8.6) and $\mathrm{mid}(\mathbf{X}) = 2$.

$\mathrm{rad}(\mathbf{X})$	True range $\mathbf{R}$	$\mathrm{d}(\mathbf{X1})/\mathrm{d}(\mathbf{R})$	$\mathrm{d}(\mathbf{X2})/\mathrm{d}(\mathbf{R})$	$\mathrm{d}(\mathbf{X3})/\mathrm{d}(\mathbf{R})$
10^{-6}	[3.6643, 3.6644]	6.33	1.00	1.00
10^{-4}	[3.6639, 3.6649]	6.33	1.00	1.00
10^{-2}	[3.6166, 3.7122]	6.34	1.18	1.04
0.1	[3.1922, 4.1434]	7.15	4.3	1.35
0.5	[1.5495, 5.8700]	89	778	4.28
1	[0.2751, 6.7189]	$5.6 \cdot 10^4$	$5.0 \cdot 10^6$	36.5
2	[−0.2962, 6.7189]	$5.6 \cdot 10^{12}$	$9.8 \cdot 10^{15}$	$3.3 \cdot 10^{11}$

defined in (8.6). From the graphs in Figure 8.2 and Figure 8.3 we know that both functions behave similarly over the interval $[0, 4]$. However, interval evaluation for the same data as in Table 11.1 for the function g produces the results shown in Table 11.2.

Clearly interval arithmetic is no panacea, particularly for wide input intervals. It is the main goal of verification methods to use appropriate mathematical tools to avoid such situations. For more details see Ratschek and Rokne (1984) or Neumaier (1990).

12. Quadrature

A direct application of the possibility of computing inclusions of Taylor coefficients is the inclusion of an integral. Often it suffices to merely implement a known error estimate using interval arithmetic. This is one (of the not so common) examples where naive evaluation in interval arithmetic is applicable. The following error estimate for Kepler's Faßregel (published in 1615, but commonly known as Simpson's [1710–1761] formula) was already known to James Stirling [1692–1770]; the computation of verified error bounds appeared in Sunaga (1956).

Theorem 12.1. For $[a, b] \in \mathbb{IR}$ let $f : [a, b] \to \mathbb{R}$ with $f \in \mathcal{C}^4$ be given. For even $n \in \mathbb{N}$, define $x_i := a + ih$ for $0 \le i \le n$ and $h := (b - a)/n$. Then

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}\big(f(x_0)+4f(x_1)+2f(x_2)+\cdots+4f(x_{n-1})+f(x_n)\big)-E \quad (12.1)$$

with

$$E := \frac{h^4}{180}(b-a)f^{(4)}(\xi) \quad \text{for some} \quad \xi \in [a, b]. \quad (12.2)$$

For a given function f, for a, b and some n the application is straightforward, for example by the following code.

Table 12.1. Integral of f and g as in (8.4) and (8.6).

Function	Approximation by `quad`		Verification by `verifyquad`	
	Approximation	Time (sec)	Inclusion	Time (sec)
f as in (8.4)	0.563112654015933	0.025	0.563112^{60}_{58}	0.410
g as in (8.6)	12.711373479890620	0.030	12.7113^{80}_{64}	0.363

Algorithm 12.2. Computation of `X` including $\int_a^b f(x)\,\mathrm{d}x$:

```
D = b-intval(a);  H = D/n;
x = a + intval(0:n)/n*D;
w=2*ones(1,n+1); w(2:2:n)=4; w(1)=1; w(n+1)=1;
V = sum( H/3 * w .* f(intval(x)) );
Y = f(taylorinit(infsup(a,b),4));   % inclusion of approximation
E = H^4*D/180 * Y.t(4);             % error term
X = V - E;                          % inclusion of integral
```

A more sophisticated algorithm, `verifyquad`, is implemented in INTLAB based on a Romberg scheme with error term and automatic choice of n, depending on the behaviour of the function.

If the input function is well behaved, the verification algorithm is slower than an approximate routine. For example, the integral of f and g as defined in (8.4) and (8.6) over $[0, 4]$ is approximated by the MATLAB routine `quad` and included by `verifyquad` using default parameters. The results are shown in Table 12.1.[22]

One of the main problems of a numerical integration routine, namely the stopping criterion, is elegantly solved by interval arithmetic: The verification algorithm stops (to increase n, for example) when the inclusion is (provably) good enough or does not improve. As an example, consider

$$\int_0^8 \sin(x + \mathrm{e}^x)\,\mathrm{d}x \tag{12.3}$$

with the results displayed in Table 12.2. Note that $\mathrm{e}^8 \approx 949\pi$.

Note that no digit of the MATLAB approximation is correct, and no warning is given by MATLAB. Presumably, some internal results in the MATLAB routine `quad` were sufficiently close to make `quad` stop without warning. As in (1.4) for the example in (1.3), even coinciding results computed in different ways give no guarantee of correctness.

[22] Note that the MATLAB routine `quad` requires the function to be specified in 'vectorized' form, *e.g.*, f as in (8.4) by
`f=vectorize(inline('sin(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1'));`

Table 12.2. Integral of f as in (12.3) using the MATLAB routine `quad`, Algorithm 12.2 and `verifyquad`.

		Result	Time (sec)
	`quad`	0.2511	1.77
Algorithm 12.2	$n = 2^{10}$	$[-0.47, 1.01]$	0.07
	$n = 2^{12}$	0.32^{9}_{2}	0.12
	$n = 2^{14}$	0.347^{42}_{39}	0.17
	$n = 2^{16}$	0.347400^{3}_{1}	0.60
	$n = 2^{18}$	0.3474001725	2.26
	`verifyquad`	0.3474001^{8}_{6}	2.66

The presented Algorithm 12.2 is a first step. More serious algorithms handle singularities within the range of integration and may integrate through the complex plane: see Corliss and Rall (1987), Petras (2002), Okayama, Matsuo and Sugihara (2009) and Yamanaka, Okayama, Oishi and Ogita (2009).

13. Nonlinear problems

Let a nonlinear system $f(x) = 0$ with differentiable function $f : \mathbf{D} \to \mathbb{R}^n$ with $\mathbf{D} \in \mathbb{IR}^n$ be given. We assume a MATLAB program `f` be given such that `f(x)` evaluates $f(x)$. Using the INTLAB operators, according to the type of the argument `x`, an approximation or inclusion of the function value or gradient or Hessian is computed.

Let $\tilde{x} \in \mathbf{D}$ be given. Denote the Jacobian of f at x by $J_f(x)$. Then, by the n-dimensional Mean Value Theorem, for $x \in \mathbf{D}$, there exist $\xi_1, \ldots, \xi_n \in x \underline{\cup} \tilde{x}$, the convex union of x and $\tilde{x}$, with

$$f(x) \;=\; f(\tilde{x}) + \begin{pmatrix} \nabla f_1(\xi_1) \\ \cdots \\ \nabla f_n(\xi_n) \end{pmatrix} \big(x - \tilde{x}\big), \tag{13.1}$$

using the component functions $f_i : \mathbf{D}_i \to \mathbb{R}$. As is well known, the ξ_i cannot, in general, be replaced by a single ξ, so that the matrix in (13.1) is only row-wise equal to some Jacobian J_f of f.

For $X \in \mathbb{PR}^n$, recall that $\mathrm{hull}(X) \in \mathbb{IR}^n$ is defined by

$$\mathrm{hull}(X) \;:=\; \bigcap \{\mathbf{Z} \in \mathbb{IR}^n \;:\; X \subseteq \mathbf{Z}\}. \tag{13.2}$$

For $x, \tilde{x} \in \mathbf{D}$, also $\mathbf{X} := \mathrm{hull}(x \underline{\cup} \tilde{x}) \subseteq \mathbf{D}$, and the inclusion property (5.16) implies

$$\begin{pmatrix} \nabla f_1(\xi_1) \\ \cdots \\ \nabla f_n(\xi_n) \end{pmatrix} \in J_f(\mathbf{X}) \quad \text{with} \quad J_f(\mathbf{X}) := \mathrm{hull}\big\{ J_f(x) : x \in \mathbf{X} \big\} \tag{13.3}$$

for all $\xi_1, \ldots, \xi_n \in \mathbf{X}$. Therefore, using interval operations, the Mean Value Theorem can be written in an elegant way.

Theorem 13.1. Let there be given continuously differentiable $f : \mathbf{D} \to \mathbb{R}^n$ with $\mathbf{D} \in \mathbb{IR}^n$ and $x, \tilde{x} \in \mathbf{D}$. Then

$$f(x) \ \in \ f(\tilde{x}) + J_f(\mathbf{X})\big(x - \tilde{x}\big) \tag{13.4}$$

for $\mathbf{X} := \mathrm{hull}(x \underline{\cup} \tilde{x})$.

This allows an interval Newton's method similar to the univariate version in Theorem 6.2. The proof is taken from Alefeld (1994).

Theorem 13.2. Let differentiable $f : \mathbf{X} \to \mathbb{R}^n$ with $\mathbf{X} \in \mathbb{IR}^n$ be given. Suppose all matrices $M \in J_f(\mathbf{X})$ are non-singular, and define, for some $\tilde{x} \in \mathbf{X}$,

$$N(\tilde{x}, \mathbf{X}) := \{ \tilde{x} - M^{-1} f(\tilde{x}) : M \in J_f(\mathbf{X}) \}. \tag{13.5}$$

If $N(\tilde{x}, \mathbf{X}) \subseteq \mathbf{X}$, then $\mathbf{X}$ contains a unique root $\hat{x}$ of f in $\mathbf{X}$. If $N(\tilde{x}, \mathbf{X}) \cap \mathbf{X} = \emptyset$, then $f(x) \neq 0$ for all $x \in \mathbf{X}$. Moreover, $\hat{x} \in N(\tilde{x}, \mathbf{X})$.

Proof. Using

$$f(x) - f(\tilde{x}) = \int_0^1 \frac{\mathrm{d}}{\mathrm{d}t} f(\tilde{x} + t(x - \tilde{x}))\, \mathrm{d}t,$$

it follows that

$$f(x) - f(\tilde{x}) = M_x (x - \tilde{x}) \quad \text{for} \quad M_x := \int_0^1 \frac{\partial f}{\partial x}(\tilde{x} + t(x - \tilde{x}))\, \mathrm{d}t \in J_f(\mathbf{X}) \tag{13.6}$$

for all $x \in \mathbf{X}$. The function

$$g(x) \ := \ \tilde{x} - M_x^{-1} f(\tilde{x}) : \mathbf{X} \to \mathbb{R}^n \tag{13.7}$$

is continuous, and by assumption $\{ g(x) : x \in \mathbf{X} \} \subseteq \mathbf{X}$. Therefore Brouwer's Fixed-Point Theorem implies existence of $\hat{x} \in \mathbf{X}$ with

$$g(\hat{x}) = \hat{x} = \tilde{x} - M_{\hat{x}}^{-1} f(\tilde{x}) \in N(\tilde{x}, \mathbf{X}),$$

so that (13.6) implies $f(\hat{x}) = 0$. Furthermore, the root $\hat{x}$ is unique in $\mathbf{X}$ by the non-singularity of all $M \in J_f(\mathbf{X})$. Finally, if $f(y) = 0$ for $y \in \mathbf{X}$, then $-f(\tilde{x}) = M_y (y - \tilde{x})$ by (13.6), so that $y = \tilde{x} - M_y^{-1} f(\tilde{x}) \in N(\tilde{x}, \mathbf{X})$. □

To apply Theorem 13.2, first an inclusion $\mathbf{J} \in \mathbb{IF}^{n\times n}$ of $J_f(\mathbf{X})$ is computed by automatic differentiation. Then an inclusion of $\boldsymbol{\Delta}$ of the solution set $\Sigma(\mathbf{J}, f(\tilde{x}))$ defined in (10.15) is computed by a verification method described in Section 10. Note that this implies in particular that all matrices $M \in J_f(\mathbf{X})$ are non-singular. If $\tilde{x} - \boldsymbol{\Delta} \subseteq \mathbf{X}$, then Theorem 13.2 implies existence and uniqueness of a root $\hat{x}$ of f in $\mathbf{X}$ (and in $\tilde{x} - \boldsymbol{\Delta}$).

For a practical implementation, first an approximate solution $\tilde{x}$ should be improved by numerical means. The better $\tilde{x}$, the smaller is the residual $f(\tilde{x})$ and the more likely is an inclusion.

Next an interval refinement $\mathbf{X}^{(k+1)} := N(\tilde{x}, \mathbf{X}^{(k)}) \cap \mathbf{X}^{(k)}$ may be applied, starting with the first inclusion $\mathbf{X}^{(0)} := \mathbf{X}$.

However, this requires at each step the solution of an interval linear system. Therefore, it is in practice often superior to use the following modification (Rump 1983) of an operator given by Krawczyk (1969*a*).

Theorem 13.3. Let there be given continuously differentiable $f : D \to \mathbb{R}^n$ and $\tilde{x} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{IR}^n$, $R \in \mathbb{R}^{n\times n}$ with $0 \in \mathbf{X}$ and $\tilde{x} + \mathbf{X} \subseteq D$. Suppose

$$S(\mathbf{X}, \tilde{x}) := -Rf(\tilde{x}) + \big\{I - RJ_f(\tilde{x} + \mathbf{X})\big\}\mathbf{X} \subseteq \operatorname{int}(\mathbf{X}). \tag{13.8}$$

Then R and all matrices $M \in J_f(\tilde{x} + \mathbf{X})$ are non-singular, and there is a unique root $\hat{x}$ of f in $\tilde{x} + S(\mathbf{X}, \tilde{x})$.

Proof. Define $g : \mathbf{X} \to \mathbb{R}^n$ by $g(x) := x - Rf(\tilde{x}+x)$. Then, using $\tilde{x} \in \tilde{x}+\mathbf{X}$ and Theorem 13.1 implies that

$$g(x) = x - R\big(f(\tilde{x}) + M_{\tilde{x}+x}x\big) = -Rf(\tilde{x}) + \big\{I - RM_{\tilde{x}+x}\big\}x \in S(\mathbf{X}, \tilde{x}) \subseteq \mathbf{X} \tag{13.9}$$

for $x \in \mathbf{X}$. By Brouwer's Fixed-Point Theorem there exists a fixed point $\hat{x} \in \mathbf{X}$ of g, so that $Rf(\tilde{x} + \hat{x}) = 0$. Moreover, $\hat{x} = g(\hat{x}) \in S(\mathbf{X}, \tilde{x})$ by (13.9). Now Lemma 10.5 applied to (13.8) implies that R and every matrix $M \in J_f(\tilde{x} + \mathbf{X})$ is non-singular, and therefore $f(\tilde{x} + \hat{x}) = 0$. The non-singularity of all $M \in J_f(\tilde{x} + \mathbf{X})$ implies that f is injective over $\tilde{x} + \mathbf{X}$. □

Much along the lines of Algorithm 10.7 with the improvements therein, the following algorithm computes an inclusion of a solution of the nonlinear system given by a function `f` near some approximation `xs`.

Algorithm 13.4. Verified bounds for the solution of a nonlinear system:

```
function XX = VerifyNonLinSys(f,xs)
  XX = NaN;                          % initialization
  y = f(gradientinit(xs));
  R = inv(y.dx);                     % approximate inverse of J_f(xs)
  Y = f(gradientinit(intval(xs)));
  Z = -R*Y.x;                        % inclusion of -R*f(xs)
  X = Z; iter = 0;
```

Table 13.1. Solution of (13.10) using MATLAB's `fsolve` and INTLAB's `verifynlss`.

Dimension	`fsolve` Median relative error	`verifynlss` Maximum relative error	`verifynlss` Ratio computing time
50	$5.7 \cdot 10^{-14}$	$6.1 \cdot 10^{-16}$	1.10
100	$5.8 \cdot 10^{-10}$	$6.8 \cdot 10^{-16}$	0.70
200	$8.0 \cdot 10^{-8}$	$5.7 \cdot 10^{-16}$	0.40
500	$2.5 \cdot 10^{-9}$	$5.7 \cdot 10^{-16}$	0.15
1000	$2.1 \cdot 10^{-7}$	$8.4 \cdot 10^{-16}$	0.13
2000	$2.2 \cdot 10^{-8}$	$8.1 \cdot 10^{-16}$	0.11

```
while iter<15
  iter = iter+1;
  Y = hull( X*infsup(0.9,1.1) + 1e-20*infsup(-1,1) , 0 );
  YY = f(gradientinit(xs+Y));    % YY.dx inclusion of J_f(xs+Y)
  X = Z + (eye(n)-R*YY.dx)*Y;    % interval iteration
  if all(in0(X,Y)), XX = xs + X; return; end
end
```

As has been mentioned before, the initial approximation $\tilde{x}$ should be improved by some numerical algorithm. This is included in the algorithm `verifynlss` in INTLAB for solving systems of nonlinear equations. It has been used for the verified solution of the discretized Emden's equation shown in Figure 1.1, and the inclusion of a stationary point of f in (11.6) and (11.7).

The nonlinear function in Algorithm 13.4 may depend on parameters $p \in \mathbb{R}^k$, so that an inclusion $\mathbf{X} \in \mathbb{IR}^n$ of a solution of $f(p, x) = 0$ is computed. If an interval vector $\mathbf{p} \in \mathbb{IR}^k$ is specified for the parameter vector and an inclusion $\mathbf{X}$ is computed, then for all $p \in \mathbf{p}$ there exists a unique solution of $f(p, x) = 0$ in $\mathbf{X}$. This is an upper bound for the sensitivity of the solution with respect to finite perturbations of the parameters. Lower bounds for the sensitivity, *i.e.*, inner inclusions of the solution set as in Section 10.6, can be computed as well: see Rump (1990).

As another example, consider the discretization of

$$3\ddot{y}y + \dot{y}^2 = 0 \quad \text{with} \quad y(0) = 0,\ y(1) = 20, \tag{13.10}$$

given by Abbott and Brent (1975). The true solution is $y = 20x^{0.75}$. As initial approximation we use equally spaced points in $[0, 20]$. The results for different dimensions, comparing `fsolve` of the MATLAB R2009b Optimization toolbox and `verifynlss` of INTLAB, are displayed in Table 13.1.

The third column displays the maximum relative error of the inclusion by `verifynlss`. The inclusions are very accurate, so the relative error of the approximation by `fsolve` can be estimated.

It seems strange that the inclusion is more accurate than the approximation. However, `verifynlss` itself performs a (simplified) Newton iteration, which apparently works better than `fsolve`.

In this particular example the verification is even faster than the approximative routine, except for dimension $n = 50$. This, however, is by no means typical. Note that an inclusion of the solution of the discretized problem (13.10) is computed; verification algorithms for infinite-dimensional problems are discussed in Sections 15 and 16.

The interval Newton method in Theorem 13.2 requires the solution of a linear system with interval matrix. We note that assumption (13.8) of Theorem 13.3 can be verified by solving a linear system with point matrix and interval right-hand side. Define $[mC - rC, mC + rC] := J_f(\tilde{x} + \mathbf{X})$, assume mC is non-singular and $\mathbf{Y}$ is an inclusion of $\Sigma(mC, -f(\tilde{x}) + [-rC, rC]\mathbf{X})$. Then $\mathbf{Y} \subset \mathrm{int}(\mathbf{X})$ implies (13.8).

To see this, set $R := mC^{-1}$ and observe

$$-mC^{-1}f(\tilde{x}) + \big\{I - mC^{-1}[mC + \Delta]\big\}x = mC^{-1}\big\{-f(\tilde{x}) - \Delta x\big\}$$

for $x \in \mathbb{R}^n$ and $\Delta \in \mathbb{R}^{n\times n}$. Applying this to $x \in \mathbf{X}$ and using $|\Delta| \le rC$ proves the assertion. Note that the non-singularity of mC follows when computing $\mathbf{Y}$ by one of the algorithms in Section 10.

This little derivation is another example of how to prove a result when interval quantities are involved: the assertion is shown for fixed but arbitrary elements out of the interval quantities and the INCLUSION PRINCIPLE (5.17) is used.

Although Algorithm 13.4 works well on some problems, for simple circumstances it is bound to fail. For example, if $\mathbf{X}$ is wide enough that $J_f(\mathbf{X})$ or $J_f(\tilde{x} + \mathbf{X})$ contain a singular matrix, then Theorem 13.2 or Theorem 13.3 is not applicable, respectively, because the non-singularity of every such matrix is proved.

One possibility for attacking this is to use automatic slopes; however, the practical merit of bounding the solution of nonlinear problems seems limited. When interval bounds get wider, slopes and their second-order variants (see Kolev and Mladenov (1997)) become more and more useful.

13.1. Exclusion regions

Verification methods based on Theorem 13.3 prove that a certain interval vector $\mathbf{X} \in \mathbb{IR}^n$ contains a unique root of a nonlinear function $f : \mathbb{R}^n \to \mathbb{R}^n$. A common task, in particular in computer algebra, is to find *all* roots of f within some region. In this case a major problem is to prove that a certain region does *not* contain a root.

Jansson (1994) suggested the following. The box $\mathbf{X}$ is widened into a box $\mathbf{Y} \in \mathbb{IR}^n$ with $\mathbf{X} \subseteq \mathbf{Y}$. If, for example by Theorem 13.3, it can be verified that $\mathbf{Y}$ contains a unique root of f as well, then $f(x) \neq 0$ for all $x \in \mathbf{Y} \backslash \mathbf{X}$. This method proves to be useful for diminishing the so-called cluster effect (Kearfott 1997) in global optimization; see also Neumaier (2004). For verification of componentwise and affine invariant existence, uniqueness, and non-existence regions, see Schichl and Neumaier (2004).

13.2. Multiple roots I

Theorem 13.3 proves that the inclusion interval contains a unique root of the given function. This implies in particular that no inclusion is possible for a multiple root.

There are two ways to deal with a multiple root or a cluster of roots. First, it can be proved that a slightly perturbed problem has a true multiple root, where the size of the perturbation is estimated as well. Second, it can be proved that the original problem has k roots within computed bounds. These may be k simple roots, or one k-fold, or anything in-between. The two possibilities will be discussed in this and the following section.

The proof that the original problem has a k-fold root is possible in exact computations such as computer algebra systems, *e.g.*, Maple (2009), but it is outside the scope of verification methods for $k \geq 2$, by the SOLVABILITY PRINCIPLE OF VERIFICATION METHODS (1.2).

We first consider double roots. In the univariate case, for given $f : \mathbb{R} \to \mathbb{R}$ define

$$g(x, \varepsilon) := \begin{pmatrix} f(x) - \varepsilon \\ f'(x) \end{pmatrix} \quad \text{with Jacobian} \quad J_g(x, \varepsilon) = \begin{pmatrix} f'(x) & -1 \\ f''(x) & 0 \end{pmatrix}. \tag{13.11}$$

Then the function $F(x) := f(x) - \hat{\varepsilon}$ satisfies $F(\hat{x}) = F'(\hat{x}) = 0$ for a root $(\hat{x}, \hat{\varepsilon})^T \in \mathbb{R}^2$ of g. The nonlinear system $g(x, \varepsilon) = 0$ can be solved using Theorem 13.3, evaluating the derivatives by the methods described in Section 11.

In the multivariate case, following Werner and Spence (1984), let a function $f = (f_1, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ be given and let $\hat{x} \in \mathbb{R}^n$ be such that $f(\hat{x}) = 0$ and the Jacobian $J_f(\hat{x})$ of f at $\hat{x}$ is singular or almost singular. Adding a smoothing parameter ε, we define $F_\varepsilon : \mathbb{R}^{n+1} \to \mathbb{R}^n$ by

$$F_\varepsilon(x) = f(x) - \varepsilon e_k, \tag{13.12}$$

for fixed k, where e_k denotes the kth column of the $n \times n$ identity matrix. In addition, the Jacobian of f is forced to be singular by

$$J_f(x) y = 0 \tag{13.13}$$

for $y = (y_1, \ldots, y_{m-1}, 1, y_{m+1}, \ldots, y_n)^T$ and some fixed m. This defines

a nonlinear system in $2n$ unknowns $g(x, \varepsilon, y) = 0$ with

$$g(x, \varepsilon, y) = \begin{pmatrix} F_\varepsilon(x) \\ J_f(x)y \end{pmatrix}, \tag{13.14}$$

and again it can be solved using Theorem 13.3 by evaluating the derivatives using the methods described in Section 11. For $g(\hat{x}, \hat{\varepsilon}, \hat{y}) = 0$, it follows that $F_{\hat{\varepsilon}}(\hat{x}) = 0$ and $\det J_{F_{\hat{\varepsilon}}}(\hat{x}) = 0$ for the perturbed nonlinear function F_ε with $\varepsilon := \hat{\varepsilon}$ in (13.12). The problem was also discussed by Kanzawa and Oishi (1999*b*, 1999*a*) and Rump and Graillat (2009), where some heuristics are given on the choice of k and m.

The case of multiple roots of univariate functions will first be explained for a triple root. Let a function $f : \mathbb{R} \to \mathbb{R}$ be given with $f^{(i)}(\tilde{x}) \approx 0$ for $0 \le i \le 2$ and $f'''(\tilde{x})$ not too small. Assume $\mathbf{X}$ is an inclusion of a root of f'' near $\tilde{x}$ computed by Algorithm 13.4, so that there is $\hat{x} \in \mathbf{X}$ with $f''(\hat{x}) = 0$. Moreover, $\hat{x}$ is a simple root of f'', so $f'''(\hat{x}) \neq 0$.

Compute $\mathbf{E}_0, \mathbf{E}_1 \in \mathbb{IR}$ with

$$f'(m) + f''(\mathbf{X})(\mathbf{X} - m) \subseteq \mathbf{E}_0 \quad \text{and} \quad f(m) + f'(\mathbf{X})(\mathbf{X} - m) - \mathbf{E}_0\mathbf{X} \subseteq \mathbf{E}_1 \tag{13.15}$$

for some $m \in \mathbf{X}$. Then $\hat{\varepsilon}_0 := f'(\hat{x}) \in \mathbf{E}_0$ and $\hat{\varepsilon}_1 := f(\hat{x}) - f'(\hat{x})\hat{x} \in \mathbf{E}_1$, and the function

$$F(x) := f(x) - \hat{\varepsilon}_0 x - \hat{\varepsilon}_1 \tag{13.16}$$

satisfies

$$F(\hat{x}) = F'(\hat{x}) = F''(\hat{x}) = 0 \quad \text{and} \quad F'''(\hat{x}) \neq 0, \tag{13.17}$$

so that $\hat{x}$ is a triple root of the perturbed function F in $\mathbf{X}$.

Following Rump and Graillat (2009), this approach can be applied to compute an inclusion of a k-fold root of a perturbed function.

Theorem 13.5. Let $f : \mathbb{R} \to \mathbb{R}$ with $f \in \mathcal{C}^{k+1}$ be given. Assume $\mathbf{X} \in \mathbb{IR}$ is an inclusion of a root $\hat{x}$ of $f^{(k-1)}$. Let $\mathbf{E}_j \in \mathbb{IR}$ be computed by

$$\mathbf{E}_j = f^{(k-2-j)}(m) + f^{(k-1-j)}(\mathbf{X})(\mathbf{X} - m) - \sum_{\nu=0}^{j-1} \frac{\mathbf{E}_\nu}{(j-\nu)!}\mathbf{X}^{j-\nu} \tag{13.18}$$

for some $m \in \mathbf{X}$. Then, for $0 \le j \le k-2$ there exist $\hat{\varepsilon}_j \in \mathbf{E}_j$ with

$$f^{(k-2-j)}(\hat{x}) = \hat{\varepsilon}_j + \sum_{\nu=0}^{j-1} \frac{\hat{\varepsilon}_\nu}{(j-\nu)!}\hat{x}^{j-\nu}. \tag{13.19}$$

Define $F : \mathbb{R} \to \mathbb{R}$ by

$$F(x) := f(x) - \sum_{\nu=0}^{k-2} \frac{\hat{\varepsilon}_\nu}{(k-2-\nu)!}\, x^{k-2-\nu}. \tag{13.20}$$

Table 13.2. Expansion of g_5, in total 7403 characters.

```
g_5 =
inline('1+40*sin(x)*cos(x^2/cosh(x)^(1/2))^2*atan(4*x+1)^3
                        ... ... ...
-240*cos(x)^2*sin(x^2/cosh(x)^(1/2))^2*cos(x^2/cosh(x)^(1/2))^2
  *sin(x)*atan(4*x+1)')
```

Table 13.3. Inclusions of a k-fold root of perturbed g_k by `verifynlss2` near $\tilde{x} = 0.82$.

k	$\mathbf{X}$	mag($\mathbf{E}_0$)	mag($\mathbf{E}_1$)	mag($\mathbf{E}_2$)	mag($\mathbf{E}_3$)
1	0.81990735694295_3^5				
2	0.8199073569_2^4	$1.28 \cdot 10^{-13}$			
3	0.819907356_3^6	$1.30 \cdot 10^{-11}$	$1.24 \cdot 10^{-11}$		
4	0.8199073_6^8	$8.85 \cdot 10^{-10}$	$1.61 \cdot 10^{-9}$	$1.93 \cdot 10^{-9}$	
5	failed				

Then $F^{(j)}(\hat{x}) = 0$ for $0 \le j \le k-1$. If the inclusion $\mathbf{X}$ is computed by a verification method based on Theorem 13.3, then the multiplicity of the root $\hat{x}$ of F in $\mathbf{X}$ is exactly k.

The methods described in this section are implemented by `verifynlss2` in INTLAB. As an example, consider our model function g in (8.6). To obtain multiple roots, we define $g_k := g(x)^k$. Using the symbolic toolbox of MATLAB, the symbolic expression

```
expand((sinh(2*x^2/sqrt(cosh(x))-x)-atan(4*x+1)+1)^k)
```

is specified for g_k, that is, the expanded version of $g(x)^k$. For example, part of the expression for g_5 with a total length of the string of 7403 characters is shown in Table 13.2.

The call `verifynlss2(g_k, 0.82, k)` attempts to compute an inclusion $\mathbf{X}$ of a k-fold root of

$$G(x) := g(x) - \epsilon_0 - \epsilon_1 x - \cdots - \epsilon_{k-2} x^{k-2}$$

near $\tilde{x} = 0.82$ and inclusions $\mathbf{E}_\nu$ of the perturbations ϵ_ν for $0 \le \nu \le k-2$. Note that, compared with (13.20), the true coefficients $\epsilon_\nu = \hat{\varepsilon}_\nu (k-2-\nu)!$ of $x^{k-2-\nu}$ are included. Since, by Figure 8.3, there is a root of g near $\tilde{x} = 0.82$, so there is a k-fold root of g_k. The computational results are shown in Table 13.3.

The test functions are based on the function g in (8.6). When using f in (8.4) instead, the results are a little more accurate, and the inclusion of the 5-fold root does not fail. Note that the results are always verified to be correct; failing means that no rigorous statement can be made.

Another approach to multiple roots of nonlinear equations going beyond Theorem 13.5 was presented by Kearfott, Dian and Neumaier (2000). They derive an efficient way to compute the topological degree of nonlinear functions that can be extended to an analytic function (or to a function that can be approximated by an analytic function).

13.3. Multiple roots II

The second kind of verification method for a multiple or a cluster of roots is to verify that, counting multiplicities, there are exactly k roots within some interval. Note that this does not contradict the SOLVABILITY PRINCIPLE OF VERIFICATION METHODS (1.2).

Neumaier (1988) gives a sufficient criterion for the univariate case. He shows that if

$$\left|\mathrm{Re}\,\frac{f^{(k)}(z)}{k!}\right| > \sum_{i=0}^{k-1}\left|\frac{f^{(i)}(\tilde z)}{i!}\right| r^{i-k} \tag{13.21}$$

is satisfied for all z in the disc $D(\tilde z, r)$, then f has exactly k roots in D.

We now discuss the approach by Rump and Oishi (2009). They showed how to omit the $(k-1)$st summand on the right of (13.21), and computed sharper expressions for the left-hand side. In particular, they gave a constructive scheme for how to find a suitable disc D.

Let a function $f : D_0 \to \mathbb{C}$ be given, which is analytic in the open disc D_0. Suppose some $\tilde x \in D_0$ is given such that $\tilde x$ is a numerically k-fold zero, *i.e.*,

$$f^{(\nu)}(\tilde x) \approx 0 \quad \text{for} \quad 0 \le \nu < k \tag{13.22}$$

and $f^{(k)}(\tilde x)$ not too small. Note that the latter is not a mathematical assumption to be verified. For $z, \tilde z \in D_0$, denote the Taylor expansion by

$$f(z) = \sum_{\nu=0}^{\infty} c_\nu (z-\tilde z)^\nu \quad \text{with} \quad c_\nu = \frac{1}{\nu!} f^{(\nu)}(\tilde z). \tag{13.23}$$

Let $X \subset D_0$ denote a complex closed disc near $\tilde x$ such that $f^{(k-1)}(\hat x) = 0$ for some $\hat x \in X$. It can be computed, for example, by `verifynlss` in INTLAB applied to $f^{(k-1)}(x) = 0$.

We aim to prove that some closed disc $Y \subset D_0$ with $X \subseteq Y$ contains exactly k roots of f. First f is expanded with respect to $\hat x$ and the series is split into

$$f(y) = q(y) + g(y)(y-\hat x)^k \quad \text{and} \quad g(y) = c_k + e(y) \tag{13.24}$$

with

$$q(y) = \sum_{\nu=0}^{k-2} c_\nu (y-\hat{x})^\nu \quad \text{and} \quad e(y) = \sum_{\nu=k+1}^{\infty} c_\nu (y-\hat{x})^{\nu-k}. \tag{13.25}$$

Note that g is holomorphic in D_0, and that $c_{k-1} = 0$ by assumption. The minimum of $|g(y)|$ on Y can be estimated by the maximum of the remainder term $|e(y)|$. This is possible by the following, apparently not so well-known version of a complex Mean Value Theorem due to Darboux (1876).[23]

Theorem 13.6. Let holomorphic $f : D_0 \to \mathbb{C}$ in the open disc D_0 be given and $a, b \in D_0$. Then, for $1 \le p \le k+1$ there exists $0 \le \Theta \le 1$ and $\omega \in \mathbb{C}, |\omega| \le 1$ such that, for $h := b - a$ and $\xi := a + \Theta(b-a)$,

$$f(b) = \sum_{\nu=0}^{k} \frac{h^\nu}{\nu!} f^{(\nu)}(a) + \omega \frac{h^{k+1}}{k!} \frac{(1-\Theta)^{k-p+1}}{p} f^{(k+1)}(\xi). \tag{13.26}$$

The following proof is due to Bünger (2008).

Proof. Set $\ell := |b-a|$, which is non-zero without loss of generality, and define a function $g : [0, \ell] \to a \underline{\cup} b$ by $g(t) := a + t\frac{b-a}{\ell}$. Then

$$|g'(t)| = \frac{|b-a|}{\ell} \equiv 1.$$

For

$$F(x) := \sum_{\nu=0}^{k} \frac{(b-x)^\nu}{\nu!} f^{(\nu)}(x),$$

this means

$$\begin{aligned} F'(x) &= f'(x) + \sum_{\nu=1}^{k} -\frac{(b-x)^{\nu-1}}{(\nu-1)!} f^{(\nu)}(x) + \frac{(b-x)^\nu}{\nu!} f^{(\nu+1)}(x) \\ &= \frac{(b-x)^k}{k!} f^{(k+1)}(x). \end{aligned}$$

With this we use $|g'(t)| \equiv 1$ and $|b - g(t)| = \ell - t$ to obtain

$$\begin{aligned} |F(b) - F(a)| = |F(g(\ell)) - F(g(0))| &= \left| \int_0^\ell (F \circ g)'(t)\, \mathrm{d}t \right| \\ &\le \int_0^\ell |F'(g(t))|\, |g'(t)|\, \mathrm{d}t = \int_0^\ell \left| \frac{|b-g(t)|^k}{k!} \right| |f^{(k+1)}(g(t))|\, \mathrm{d}t \\ &= \int_0^\ell \frac{(\ell-t)^k}{k! p (\ell-t)^{p-1}} |f^{(k+1)}(g(t))| p (\ell-t)^{p-1}\, \mathrm{d}t \end{aligned}$$

[23] Thanks to Prashant Batra for pointing this out.

$$\leq \frac{(\ell - t^*)^{k-p+1}}{k!p} |f^{(k+1)}(g(t^*))| \int_0^\ell (-(\ell - t)^p)' \, \mathrm{d}t$$
$$= \frac{(\ell - t^*)^{k-p+1} \ell^p}{k!p} |f^{(k+1)}(g(t^*))|$$

for $1 \leq p \leq k+1$ and some $t^* \in [0, \ell]$. For $\Theta := \frac{t^*}{\ell} \in [0,1]$, the last expression is equal to

$$\frac{\ell^{k+1}(1-\Theta)^{k-p+1}}{k!p} |f^{(k+1)}(a + \Theta(b-a))|,$$

so that there exists complex ω with $|\omega| \leq 1$ and

$$f(b) - f(a) - \sum_{\nu=1}^{k} \frac{(b-a)^\nu}{\nu!} f^{(\nu)}(a) = F(b) - F(a)$$
$$= -\omega \frac{(b-a)^{k+1}}{k!} \frac{(1-\Theta)^{k-p+1}}{p} |f^{(k+1)}(a + \Theta(b-a))|. \qquad \square$$

Using Taylor coefficients as described in Section 11.4, an inclusion of $c_k = \frac{1}{k!} f^{(k)}(\hat{x})$ can be evaluated, and with Theorem 13.6 the remainder term $e(y)$ in (13.24) can be estimated as well.

Note that there is some freedom to choose p. The choice $p = k+1$ gives the traditional-looking expansion

$$f(b) = \sum_{\nu=0}^{k} \frac{h^\nu}{\nu!} f^{(\nu)}(a) + \omega \frac{h^{k+1}}{(k+1)!} f^{(k+1)}(\xi) \quad \text{with} \quad |\omega| \leq 1,$$

so that

$$|e(y)| \leq \frac{|b-a|}{(k+1)!} \max_{z \in \partial Y} |f^{(k+1)}(z)| \quad \forall \, y \in Y. \tag{13.27}$$

For $p = k$ the interval for Θ may be split to obtain $|e(y)| \leq \max(\beta_1, \beta_2)$ with

$$\beta_1 := \frac{|b-a|}{k!} \max_{|y - \hat{x}| \leq \frac{r}{2}} |f^{(k+1)}(y)| \tag{13.28}$$

and

$$\beta_2 := \frac{|b-a|}{2k!} \max_{|y - \hat{x}| \leq r} |f^{(k+1)}(y)|, \tag{13.29}$$

where $r := \max_{y \in Y} |y - \hat{x}|$. By the definition (13.24) this gives a computable lower bound for $|g(y)|$.

Let a polynomial $P(z) \in \mathbb{C}[z]$ with $P(z) = \sum_{\nu=0}^{n} p_\nu z^\nu$ be given with $p_n \neq 0$. The Cauchy polynomial $C(P)$ with respect to P is defined by $C(P) := |p_n x^n| - \sum_{\nu=0}^{n-1} |p_\nu| x^\nu \in \mathbb{R}[x]$. By Descartes' rule of signs, $C(P)$ has exactly one non-negative root, called the Cauchy bound $\overline{C(P)}$. It is well

known that the Cauchy bound is an upper bound for the absolute value of all (real and complex) roots of P:

$$P(z) = 0 \quad \Rightarrow \quad |z| \le \overline{C(P)}. \tag{13.30}$$

In fact, it is the best upper bound taking only the absolute values $|p_\nu|$ into account. Note that the leading coefficient p_n must be non-zero.

The Cauchy bound can be defined for interval polynomials as well. For $\mathfrak{P}(z) \in \mathbb{IK}[z]$ with $\mathfrak{P}(z) = \sum_{\nu=0}^{n} \mathfrak{p}_\nu z^\nu$, $\mathfrak{p}_\nu \in \mathbb{IK}$ and $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, define

$$C(\mathfrak{P}) := \operatorname{mig}(\mathfrak{p}_n)x^n - \sum_{\nu=0}^{n-1} \operatorname{mag}(\mathfrak{p}_\nu)x^\nu \in \mathbb{R}[x], \tag{13.31}$$

where $\operatorname{mig}(\mathfrak{p}_n) := \min\{|\pi| : \pi \in \mathfrak{p}_n\}$ and $\operatorname{mag}(\mathfrak{p}_\nu) := \max\{|\pi| : \pi \in \mathfrak{p}_\nu\}$. Then the unique non-negative root $\overline{C(\mathfrak{P})}$ of $C(\mathfrak{P})$ is a root bound for all polynomials $P \in \mathfrak{P}$:

$$P \in \mathfrak{P} \quad \text{and} \quad P(z) = 0 \quad \Rightarrow \quad |z| \le \overline{C(\mathfrak{P})}. \tag{13.32}$$

The Cauchy bound for real or complex interval polynomials is easily upper-bounded by applying a few Newton iterations on $C(\mathfrak{P})$ starting at some other traditional root bound. Note that the iteration converges quickly to $\overline{C(\mathfrak{P})}$.

Theorem 13.7. Let holomorphic $f : D_0 \to \mathbb{C}$ in the open disc D_0 and fixed $k \in \mathbb{N}$ be given, and closed discs $\mathbf{X}, \mathbf{Y} \subset D_0$ with $\mathbf{X} \subseteq \mathbf{Y}$. Assume there exists $\hat{x} \in \mathbf{X}$ with $f^{(k-1)}(\hat{x}) = 0$. Define $g(y)$ as in (13.24), and let $\mathbf{G} \in \mathbb{IC}$ be a complex disc with $g(y) \in \mathbf{G}$ for all $y \in Y$. Assume $0 \notin \mathbf{G}$, and define the interval polynomial

$$\mathfrak{P}(z) := q(z) + \mathbf{G} \cdot (z - \hat{x})^k \in \mathbb{IC}[z]. \tag{13.33}$$

Denote the closed complex disc with centre m and radius r by $D(m; r)$. Assume that the Cauchy bound $\overline{C(\mathfrak{P})}$ for $\mathfrak{P}$ satisfies

$$D(\hat{x}\,; \overline{C(\mathfrak{P})}) \subset \operatorname{int}(Y). \tag{13.34}$$

Then, counting multiplicities, there are exactly k roots of the function f in $D(\hat{x}\,; \overline{C(\mathfrak{P})})$.

Proof. Define the parametrized set of polynomials

$$P_y(z) := q(z) + g(y)(z - \hat{x})^k \in \mathbb{C}[z]. \tag{13.35}$$

Note that only the leading term depends on the parameter y. By definition (13.24) we have $f(y) = P_y(y)$. Moreover, $P_y \in \mathfrak{P}$ for all $y \in Y$, so that $g(y) \neq 0$ and (13.32) imply that $P_y(z) = 0$ is only possible for $z \in D(\hat{x}\,; \overline{C(\mathfrak{P})})$. Thus (13.34) implies for all $y \in Y$ that $P_y(z) \neq 0$ for all $z \in \partial Y$. Next define

$$P_{y,t}(z) := t \cdot q(z) + g(y)(z - \hat{x})^k \tag{13.36}$$

and the homotopy function

$$h_t(y) := P_{y,t}(y) = t \cdot q(y) + g(y)(y - \hat{x})^k. \tag{13.37}$$

Since q is a polynomial and g is holomorphic, all functions h_t are holomorphic as well. The definition of the Cauchy bound implies

$$\overline{C(P_{y,t})} \le \overline{C(P_y)} \le \overline{C(\mathfrak{P})} \tag{13.38}$$

for all $t \in [0,1]$ and all $y \in Y$. Thus definition (13.37) implies that for all $t \in [0,1]$ we have $h_t(y) \neq 0$ for all $y \in \partial Y$. We conclude that all holomorphic functions h_t must have the same number of roots in Y, in particular h_0 and h_1.

For $t = 0$ we have $h_0(y) = g(y)(y - \hat{x})^k$, which has exactly k roots in Y because $g(y) \neq 0$ for all $y \in Y$. Hence

$$h_1(y) = q(y) + g(y)(y - \hat{x})^k = P_y(y) = f(y)$$

must have exactly k roots in Y. By (13.38), for all $t \in [0,1]$ and all $y \in Y$, all roots of $P_{y,t}(z)$ lie in $D(\hat{x}\,; \overline{C(\mathfrak{P})})$, so in particular the roots of f. This concludes the proof. □

For the applicability of Theorem 13.7 in a verification method, note that the quality of the bound depends directly on the lower bound on $|g(\mathbf{Y})|$, which means by (13.24) on the lower bound of $c_k = \frac{1}{k!} f^{(k)}(\hat{x}) \in \frac{1}{k!} f^{(k)}(\mathbf{X})$.

The direct computation of an inclusion of $\frac{1}{k!} f^{(k)}(\mathbf{X})$ by interval arithmetic can be improved using the centred form

$$c_k \in \frac{1}{k!} f^{(k)}(\tilde{x}) + \frac{1}{(k+1)!} f^{(k+1)}(\mathbf{X}) \cdot (\mathbf{X} - \tilde{x}) \tag{13.39}$$

for $\tilde{x} \in \mathbf{X}$. A suitable choice is some $\tilde{x}$ near the midpoint of $\mathbf{X}$.

The problem remains to find a suitable inclusion interval Y. Note that the inclusion interval is necessarily complex: if the assumptions of Theorem 13.7 are satisfied for some function f, they are by continuity satisfied for a suitably small perturbation of f as well. But an arbitrary small perturbation of f may move a double real root into two complex roots. This is another example of the SOLVABILITY PRINCIPLE OF VERIFICATION METHODS (1.2).

Since $\hat{x} \in \mathbf{X}$ is necessary by assumption, a starting interval may be $\mathbf{Y}^0 := \mathbf{X}$. However, the sensitivity of a k-fold root is $\varepsilon^{1/k}$ for an ε-perturbation of the coefficients (see below). But the quality of the inclusion X of the *simple* root of $f^{(k-1)}$ can be expected to be nearly machine precision.

The polynomial in (13.33), $\mathfrak{P}_{\mathbf{Y}}$, say, depends on $\mathbf{Y}$. The main condition to check is (13.34). Thus a suitable candidate for a first inclusion interval is $\mathbf{Y}^1 := D(\hat{x}\,; \overline{C(\mathfrak{P}_{\mathbf{Y}^0})})$. This already defines an iteration scheme, where $\mathbf{Y}^{m+1} \subset \operatorname{int}(\mathbf{Y}^m)$ verifies the conditions of Theorem 13.7. Equipped with an epsilon-inflation as in (10.13), this is a suitable verification method for

Table 13.4. Inclusions of k roots of the original function g_k by `verifynlss2` near $\tilde{x} = 0.82$ and sensitivity of the root.

k	$\mathfrak{Re}(\mathbf{X})$	rad($\mathbf{X}$)	Sensitivity
1	0.81990735694295_{3}^{5}	$6.32 \cdot 10^{-15}$	$2.14 \cdot 10^{-16}$
2	0.8199907_{0}^{7}	$2.68 \cdot 10^{-7}$	$2.53 \cdot 10^{-8}$
3	0.819_{820}^{995}	$8.71 \cdot 10^{-5}$	$1.25 \cdot 10^{-5}$
4	0.8_{18}^{22}	$1.54 \cdot 10^{-3}$	$2.95 \cdot 10^{-4}$
5	failed		$1.96 \cdot 10^{-3}$

the inclusion of k roots of a univariate nonlinear function. It is implemented in `verifynlss2` in INTLAB.

For computational results we use the same function as in the previous section, namely $g_k := g(x)^k$ for our model function g in (8.6). Again the expanded function as in Table 13.2 is used. The results are displayed in Table 13.4.

Note that by construction, and according to the Solvability principle (1.2), the inclusion is a complex disc. In all examples in Table 13.4 the inclusion was the smallest disc including $\mathfrak{Re}(\mathbf{X})$, *i.e.*, the imaginary part of the midpoint was zero.

Although the inclusions in Table 13.4 may appear wide, we show that they are, when computing in double precision, almost best possible. Let analytic f be given with k-fold root $\hat{z}$. For given ϵ define $\tilde{f}(z) := f(z) - \epsilon$. By continuity, for sufficiently small ϵ there exists small h with $\tilde{f}(\hat{z}+h) = 0$, so that

$$0 = -\epsilon + c_k h^k + \mathcal{O}(h^{k+1}), \tag{13.40}$$

using the Taylor expansion $f(\hat{z}+h) = \sum c_\nu f^{(\nu)}(\hat{z}) h^\nu$. Thus h, the sensitivity of the k-fold root $\hat{z}$, is of the order $(\epsilon/c_k)^{1/k}$ for small ϵ.

Let a function f be given by an arithmetic expression, such as g_5 in Table 13.2. As a course analysis, $f(\tilde{x})$ is evaluated as a floating-point sum $\mathrm{fl}(t_1 + \cdots + t_m)$ of some $t_\mu \in \mathbb{F}$, where $t_\mu := \mathrm{fl}(T_\mu(\tilde{x}))$ is the result of the floating-point evaluation of (possibly large) terms $T_\mu(\tilde{x})$.

Due to rounding errors, the accuracy of $\mathrm{fl}(t_1 + \cdots + t_m)$ is at best $\epsilon := \mathtt{u} \cdot \max_\mu |t_\mu|$ for the relative rounding error unit $\mathtt{u} = 2^{-53}$, ignoring possible sources of errors in the evaluation of the summands $T_\mu(\tilde{x})$.

Thus the inevitable presence of rounding errors creates the sensitivity $h = (\mathtt{u} \cdot \max_\mu |t_\mu|/c_k)^{1/k}$ of a k-fold root. This sensitivity is shown in Table 13.4, and it is not far from the radius of the inclusion. The analysis is confirmed by the floating-point evaluation of g_5 over $[0.816, 0.824]$, as shown in Figure 13.1.

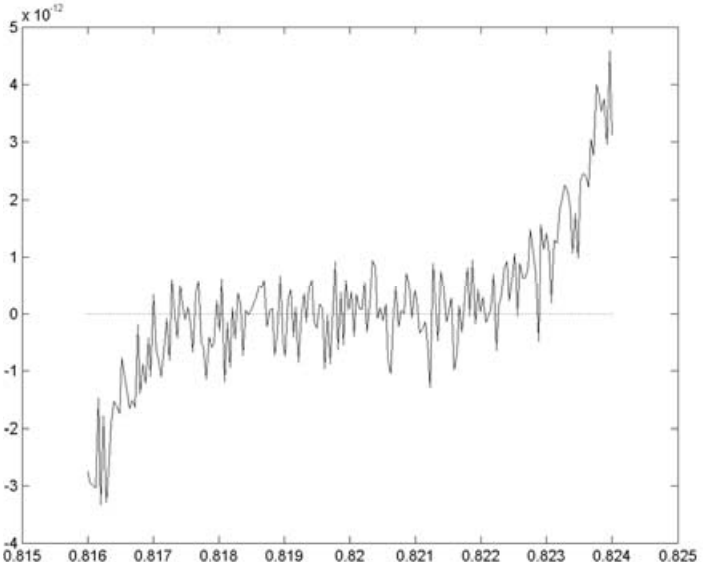

Figure 13.1. Floating-point evaluation of g_5 as in Table 13.2 on 200 equidistant mesh points in [0.816,0.824].

13.4. Simple and multiple eigenvalues

The methods discussed so far can be applied to a variety of particular problems. For example, Theorem 13.3 and Algorithm 13.4 imply a verification algorithm for simple roots of polynomials. For simple roots this is efficient where, of course, the derivative can be computed directly.

For multiple roots of polynomials, Neumaier (2003) and Rump (2003*a*) present a number of specific methods taking advantage of the special structure of the problem.

An eigenvector/eigenvalue pair (x, λ) can be written as a solution of the nonlinear system $Ax - \lambda x = 0$ together with some normalization: see Krawczyk (1969*b*). Again, multiple eigenvalues can be treated by the general approach as in the last section, but it is superior to take advantage of the structure of the problem. The approach in Rump (2001*b*) for non-self-adjoint matrices, to be described in the following, is a further example of how to develop verification methods. We mention that the methods apply *mutatis mutandis* to the generalized eigenvalue problem $Ax = \lambda Bx$.

Another example of the SOLVABILITY PRINCIPLE (1.2) is the following. Suppose $\mathbf{\Lambda} \in \mathbb{IR}$ and $\mathbf{X} \in \mathbb{IR}^k$ have been calculated by a verification algorithm such that $\mathbf{\Lambda}$ contains k not necessarily distinct eigenvalues $\lambda_1, \ldots, \lambda_k$ of a matrix $A \in \mathbb{R}^{n \times n}$, and the columns of $\mathbf{X}$ contain an inclusion of basis vectors of the corresponding invariant subspace. Then Rump and Zemke (2004) show under plausible assumptions that all eigenvalues λ_ν must have

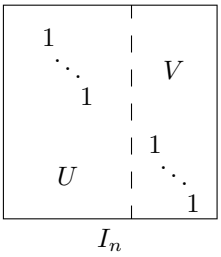

Figure 13.2. Partition of the identity matrix.

geometric multiplicity 1, *i.e.*, have, up to normalization, a unique eigenvector. This is even true for symmetric A. The reason is that for an eigenvalue λ of A of geometric multiplicity $m > 1$, for any eigenvector x in the m-dimensional eigenspace there exists an arbitrarily small perturbation $\tilde{A}$ of A such that λ is a simple eigenvalue of $\tilde{A}$ and x is (up to normalization) the unique eigenvector.

For $A \in \mathbb{K}^{n\times n}$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, let $\tilde{X} \in \mathbb{K}^{n\times k}$ be an approximation to a k-dimensional invariant subspace corresponding to a multiple or a cluster of eigenvalues near some $\tilde{\lambda} \in \mathbb{K}$, such that $A\tilde{X} \approx \tilde{\lambda}\tilde{X}$. As always, there are no *a priori* assumptions on the quality of the approximations $\tilde{X}$ and $\tilde{\lambda}$.

The degree of arbitrariness is removed by freezing k rows of the approximation $\tilde{X}$. If the set of these rows is denoted by v, and by definition $u := \{1, \ldots, n\}\backslash v$, then denote by $U \in \mathbb{R}^{n\times(n-k)}$ the submatrix of the identity matrix with columns in u. Correspondingly, define $V \in \mathbb{R}^{n\times k}$ to comprise of the columns in v out of the identity matrix. Denoting the $n \times n$ identity matrix by I_n, we have $UU^T + VV^T = I_n$, and $V^T\tilde{X} \in \mathbb{K}^{k\times k}$ is the normalizing part of $\tilde{X}$. Note that $U^TU = I_{n-k}$ and $V^TV = I_k$. For example, for $u = \{1, \ldots, n-k\}$, $v = \{n-k+1, \ldots, n\}$ the situation is as in Figure 13.2.

For given $\tilde{X} \in \mathbb{K}^{n\times k}$ and $\tilde{\lambda} \in \mathbb{K}$, suppose

$$AY = YM \quad \text{for } Y \in \mathbb{K}^{n\times k},\ M \in \mathbb{K}^{k\times k}, \tag{13.41}$$

such that Y and $\tilde{X}$ coincide in the normalizing part of $\tilde{X}$: $V^TY = V^T\tilde{X}$. The unknown quantities U^TY and M are collected into $\hat{X} \in \mathbb{K}^{n\times k}$. In other words, $\hat{X}$ will be computed with $U^T\hat{X} = U^TY$ and $V^T\hat{X} = M$. Note that M is not assumed to be diagonal. For $u = \{1, \ldots, n-k\}, v = \{n-k+1, \ldots, n\}$ the situation is as in Figure 13.3. This implies the eigen-equation

$$A(UU^T\hat{X} + VV^T\tilde{X}) = (UU^T\hat{X} + VV^T\tilde{X})V^T\hat{X}, \tag{13.42}$$

such that, according to (13.41), $Y = UU^T\hat{X} + VV^T\tilde{X}$ and $M = V^T\hat{X}$. It can be shown that the following algorithm, Algorithm 13.8, converges quadratically under reasonable conditions.

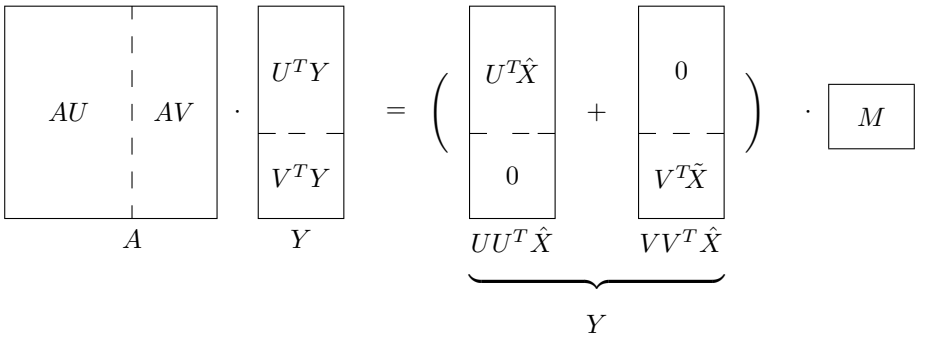

Figure 13.3. Nonlinear system for multiple or clusters of eigenvalues.

Algorithm 13.8. Newton-like iteration for eigenvalue clusters:

$X^0 := UU^T\tilde{X} + \tilde{\lambda}V$
for $\nu = 0, 1, \ldots$
$\lambda_\nu := \mathrm{trace}(V^T X^\nu)/k$
$C_\nu := (A - \lambda_\nu I_n)UU^T - (UU^T X^\nu + VV^T\tilde{X})V^T$
$X^{\nu+1} := UU^T X^\nu + \lambda_\nu V - C_\nu^{-1} \cdot (A - \lambda_\nu I_n)(UU^T X^\nu + VV^T\tilde{X})$

Note that the scalar λ_ν is an approximation to the cluster, and is adjusted in every iteration to be the mean value of the eigenvalues of $V^T X^\nu$. This iteration is the basis for the following verification method.

Theorem 13.9. Let $A \in \mathbb{K}^{n\times n}, \tilde{X} \in \mathbb{K}^{n\times k}, \tilde{\lambda} \in \mathbb{K}, R \in \mathbb{K}^{n\times n}$ and $X \in \mathbb{K}^{n\times k}$ be given, and let U, V partition the identity matrix as defined in Figure 13.2. Define

$$f(X) := -R(A\tilde{X} - \tilde{\lambda}\tilde{X}) + \{I - R\big((A - \tilde{\lambda}I)UU^T - (\tilde{X} + UU^T \cdot X)V^T\big)\} \cdot X. \tag{13.43}$$

Suppose

$$f(\mathbf{X}) \subseteq \mathrm{int}(\mathbf{X}). \tag{13.44}$$

Then there exists $\hat{M} \in \mathbb{K}^{k\times k}$ with $\hat{M} \in \tilde{\lambda}I_k + V^T\mathbf{X}$ such that the Jordan canonical form of $\hat{M}$ is identical to a $k \times k$ principal submatrix of the Jordan canonical form of A, and there exists $\hat{Y} \in \mathbb{K}^{n\times k}$ with $\hat{Y} \in \tilde{X} + UU^T\mathbf{X}$ such that $\hat{Y}$ spans the corresponding invariant subspace of A. We have $A\hat{Y} = \hat{Y}\hat{M}$.

Proof. The continuous mapping $f : \mathbb{K}^n \to \mathbb{K}^n$ defined by (13.43) maps by (13.44) the non-empty, convex and compact set $\mathbf{X}$ into itself. Therefore, Brouwer's Fixed-Point Theorem implies existence of a fixed point $\hat{X} \in \mathbb{K}^n$ with $f(\hat{X}) = \hat{X}$ and $\hat{X} \in \mathbf{X}$. Inserting in (13.43) yields

$$-R\{(A\tilde{X} - \tilde{\lambda}\tilde{X}) + (A - \tilde{\lambda}I)UU^T\hat{X} - (\tilde{X} + UU^T\hat{X})V^T\hat{X}\} = 0. \tag{13.45}$$

Furthermore, (13.43), (13.44) and Lemma 10.5 imply R and every matrix within $\mathbf{B} := (A - \tilde{\lambda}I)UU^T - (\tilde{X} + UU^T \cdot \mathbf{X})V^T \in \mathbb{IK}^{n\times n}$ to be non-singular. Collecting terms in (13.45) yields

$$A(\tilde{X} + UU^T\hat{X}) = (\tilde{X} + UU^T\hat{X})(\tilde{\lambda}I_k + V^T\hat{X})$$

or

$$A\hat{Y} = \hat{Y}\hat{M} \quad \text{for} \quad \hat{Y} := \tilde{X} + UU^T\hat{X} \quad \text{and} \quad \hat{M} := \tilde{\lambda}I_k + V^T\hat{X}.$$

Finally, $(A - \tilde{\lambda}I)UU^T - (\tilde{X} + UU^T\hat{X})V^T \in \mathbf{B}$ is non-singular and has k columns equal to $-\hat{Y}$. Therefore, $\hat{Y}$ has full rank and is a basis for an invariant subspace of A. For $\hat{M} = ZJZ^{-1}$ denoting the Jordan canonical form, $A\hat{Y} = \hat{Y}\hat{M}$ implies $A(\hat{Y}Z) = (\hat{Y}Z)J$. The theorem is proved. □

Note that Theorem 13.9 is applicable for $k = 1, \ldots, n$. For $k = 1$ we have the usual eigenvalue/eigenvector inclusion, basically corresponding to the application of Theorem 13.3 to $Ax - \lambda x = 0$, freezing some component of x. For $k = n$ the maximum spectral radius of $\tilde{\lambda}I + X$, $X \in \mathbf{X}$ is an inclusion of all eigenvalues.

For a practical implementation, $\tilde{X}$ and $\tilde{\lambda}$ are such that $A\tilde{X} \approx \tilde{\lambda}\tilde{X}$, and the matrix R serves as a preconditioner, with (13.43) indicating the obvious choice

$$R \approx \left((A - \tilde{\lambda}I)UU^T - \tilde{X}V^T\right)^{-1}.$$

As before, Theorem 13.9 computes an inclusion $\mathbf{X}$ for the error with respect to $\tilde{\lambda}$ and $\tilde{X}$. For an interval iteration with epsilon-inflation as in (10.13), an initial choice for $\mathbf{X}$ is a small superset of the correction term $-R(A\tilde{X} - \tilde{\lambda}\tilde{X})$.

It remains to compute an inclusion of the eigenvalue cluster, that is, an inclusion of the eigenvalues of $\hat{M}$. Using Gershgorin circles of $\tilde{\lambda}I_k + V^T\mathbf{X}$ would yield quite pessimistic bounds for defective eigenvalues.

For an interval matrix $\mathbf{C} \in \mathbb{K}^{k\times k}$, denote by $|\mathbf{C}| \in \mathbb{R}^{k\times k}$ the matrix of the entrywise maximum modulus of $\mathbf{C}$. Therefore, $|C_{ij}| \le (|\mathbf{C}|)_{ij}$ for every $C \in \mathbf{C}$. Then,

$$\begin{aligned} &\text{for} \quad r := \varrho(|V^T\mathbf{X}|) \quad \text{there are } k \text{ eigenvalues of } A \\ &\text{in} \quad U_r(\tilde{\lambda}) := \{z \in \mathbb{C} : |z - \tilde{\lambda}| \le r\}, \end{aligned} \tag{13.46}$$

where ϱ denotes the spectral radius, in this case the Perron root of $|V^T\mathbf{X}| \in \mathbb{R}^{k\times k}$, which can be estimated as in (10.61). As a matter of principle, the inclusion is complex.

To see (13.46), observe that for $\hat{M} = \tilde{\lambda}I_k + \tilde{M}$, for some $\tilde{M} \in V^T\mathbf{X}$, the eigenvalues of $\hat{M}$ are the eigenvalues of $\tilde{M}$ shifted by $\tilde{\lambda}$, and for any eigenvalue of μ of $\tilde{M}$, Perron–Frobenius theory implies $|\mu| \le \varrho(\tilde{M}) \le \varrho(|\tilde{M}|) \le \varrho(|V^T\mathbf{X}|) = r$. Using (13.46) is especially advantageous for defective eigenvalues.

The matrix $V^T\mathbf{X}$ basically contains error terms except for large off-diagonal quantities characterizing the Jordan blocks. If the error terms are of size ε and off-diagonal elements of size 1, the spectral radius of $|V^T\mathbf{X}|$ is of size $\varepsilon^{1/m}$, where m is the size of the largest Jordan block. Therefore, the radius of the inclusion is of size $\varepsilon^{1/m}$, which corresponds to the sensitivity of defective eigenvalues given by Wilkinson (1965, p. 81).

In turn, this implies that if the distance of an m-fold defective eigenvalue to the rest of the spectrum is of the order $\varepsilon^{1/m}$, then 'numerically' the cluster comprises at least $m+1$ eigenvalues, and for $k = m$ no inclusion is possible.

For the same reason the quality of a numerical algorithm to approximate an eigenvalue corresponding to a $k \times k$ Jordan block will be no better than $\varepsilon^{1/k}$. We demonstrate this with the following example.

Algorithm 13.10. Computing approximations and inclusions of eigenvalues of `A`:

```
n = 8; C = triu(tril(ones(n),1));
H = hadamard(n); A = H'*C*H/8,
[V,D] = eig(A);
e = diag(D), close, plot(real(e),imag(e),'*')
[L,X] = verifyeig(A,mean(e),V)
```

The matrix `C` is unit upper triangular with ones on the first upper diagonal, *i.e.*, one Jordan block to $\lambda = 1$. MATLAB seems to check for triangular matrices because `eig(C)` produces a vector of ones.

The 8×8 Hadamard matrix has integer entries, and $8^{-1}H^TH = I$, also in floating-point arithmetic for $8 = 2^3$. Hence `A` has the same Jordan structure as `C`, *i.e.*, one 8×8 Jordan block to the 8-fold eigenvalue 1.

The computed eigenvalues (in the diagonal of `D`) are shown as asterisks in Figure 13.4. Basically, they are on a circle around 1 with radius 0.1. MATLAB produces those values without error message.

The final statement in Algorithm 13.10 computes the inclusion $\mathcal{C} := \{z \in \mathbb{C} : |z - 1.0002| \leq 0.0355\}$ based on the mean of the eigenvalue approximations and the approximation `V` of the corresponding invariant subspace. Note that `V` is numerically singular because there is only one eigenvector to the 8-fold eigenvalue 1.

Since $\varepsilon^{1/8} = 0.0101$, the quality of $\mathcal{C}$ corresponds to the sensitivity of the 8-fold eigenvalue. Note that the floating-point approximations of the eigenvalue, which may be regarded as of poor quality, represent, in fact, a backward stable result: there is a small perturbation of the input matrix with true eigenvalues near the computed approximations.

It is proved that $\mathcal{C}$ contains 8 eigenvalues of `A`, and the analysis shows that there is not much room for improvement. We note that for $n = 16$ the same approach fails; no inclusion is computed.

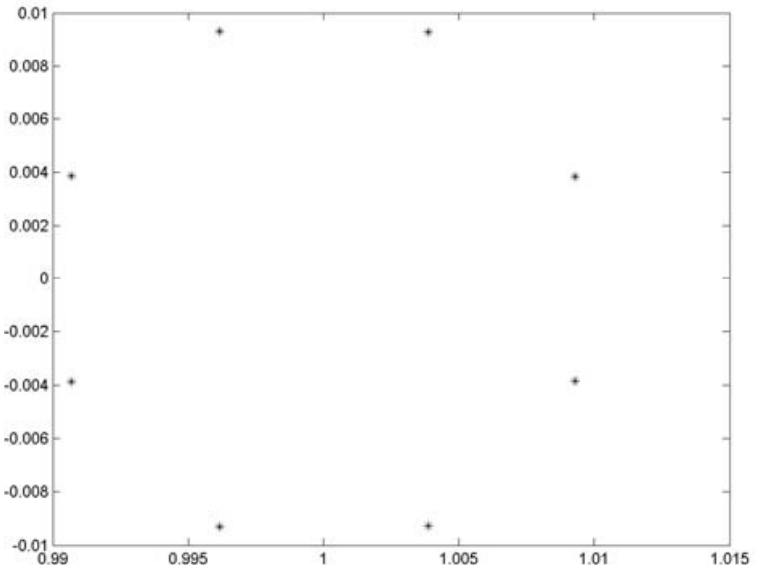

Figure 13.4. The computed eigenvalue approximations for `A` as in Algorithm 13.10.

Applying Theorem 13.9 to an interval matrix $\mathbf{A} \in \mathbb{K}^{n\times n}$ yields an inclusion of eigenvalues and eigenvectors of all $A \in \mathbf{A}$. In this case, as for general nonlinear equations, inner inclusions may be computed as well. Again this corresponds to bounds for the sensitivity with respect to finite perturbations of A.

Moreover, outer and inner inclusions for input matrices out of some structure may be computed as well. This corresponds to structured finite perturbations and generalizes the structured condition numbers for eigenvalues and pseudospectra discussed in Rump (2006).

14. Optimization

As mentioned in the abstract, Sahinidis and Tawaralani (2005) received the 2006 Beale–Orchard–Hays Prize for their global optimization package BARON, which 'incorporates techniques from automatic differentiation, interval arithmetic, and other areas to yield an automatic, modular, and relatively efficient solver for the very difficult area of global optimization' (from the laudatio).

In optimization some important tasks are treated successfully by interval methods which could hardly be solved in any other way. In particular, nonlinear terms in constraint propagation for branch and bound methods, the estimation of the range of a function, and verification of the non-existence

of roots within a domain (see Section 13.1) are reserved to interval methods or related techniques from convex analysis.

Neumaier (2004), in his *Acta Numerica* article, gave a detailed overview on global optimization and constraint satisfaction methods. In view of the thorough treatment there, showing the essential role of interval methods in this area, we restrict our discussion to more recent, complementary issues.

14.1. Linear and convex programming

One might be inclined to presume that convex optimization problems are less affected by numerical problems. However, the NETLIB (2009) suite of linear optimization problems contains practical examples from various areas, and a study by Ordóñez and Freund (2003) revealed that 72% of these real-life problems are ill-conditioned; they show that many commercial solvers fail.

For mixed integer linear programming problems, preprocessing in particular may change the status of the linear program from feasible to infeasible, and *vice versa*. Jansson (2004*b*) and Neumaier and Shcherbina (2004) give methods describing how safe bounds for the solution of linear and mixed integer linear programming problems can be obtained with minimal additional computational effort (also, a simple example is given for which many commercial solvers fail). A generalization of their method will be described below.

In my experience, although straightforward, it is not easy to program a *robust* simplex algorithm. Even for small problems it is not unlikely that an incorrect branch will lead to a sub-optimal result or apparent infeasibility.

Much more demanding is the situation when applying local programming solvers. Tawaralani and Sahinidis (2004) pointed out that nonlinear programming solvers often fail even in solving convex problems. Due to this lack of reliability, as one consequence, they used in their global optimization package BARON only linear instead of nonlinear convex relaxations.

For linear programming, efficient verification methods have been implemented in LURUPA by Keil (2006). This is a C++ package based on PROFIL/BIAS by Knüppel (1994, 1998). It includes a routine for computing rigorous bounds for the condition number. Numerical results for the NETLIB (2009) lp-library, a collection of difficult-to-solve applications, can be found in Keil and Jansson (2006).

For convex optimization problems, too, some postprocessing of computed data allows one to compute rigorous bounds for the result with little additional effort.

14.2. Semidefinite programming

For this review article we restrict the exposition to semidefinite programming problems (SDP), and briefly sketch some promising results in this

direction, following Jansson, Chaykin and Keil (2007). This class is rather extensive, since many nonlinear convex problems can be reformulated within this framework. For a general introduction to semidefinite programming, see Todd (2001).

Consider the (*primal*) *semidefinite program* in block diagonal form,

$$p^* := \min \sum_{j=1}^{n} \langle C_j, X_j \rangle \quad \text{such that} \quad \sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = b_i \quad \text{for} \quad i = 1, \ldots, m, \\ X_j \succeq 0 \quad \text{for} \quad j = 1, \ldots, n, \tag{14.1}$$

where C_j, A_{ij}, and X_j are symmetric $s_j \times s_j$ matrices, $b \in \mathbb{R}^m$, and

$$\langle C, X \rangle = \operatorname{trace}(C^T X) \tag{14.2}$$

denotes the inner product for the set of symmetric matrices. Moreover, $\succeq$ is the Löwner partial ordering, that is, $X \succeq Y$ if and only if $X - Y$ is positive semidefinite.

Semidefinite programming problems generalize linear programming problems as by $s_j = 1$ for $j = 1, \ldots, n$, in which case C_j, A_{ij} and X_j are real numbers. On the other hand, linear and semidefinite programming are special cases of conic programming. This is a universal form of convex programming, and refers to non-smooth problems with linear objective function, linear constraints, and variables that are restricted to a cone.

The Lagrangian dual of (14.1) is

$$d^* := \max b^T y \quad \text{such that} \quad \sum_{i=1}^{m} y_i A_{ij} \preceq C_j \quad \text{for} \quad j = 1, \ldots, n, \tag{14.3}$$

where $y \in \mathbb{R}^m$, so that the constraints $\sum_{i=1}^{m} y_i A_{ij} \preceq C_j$ are linear matrix inequalities (LMI). We use the convention that $p^* = -\infty$ if (14.1) is unbounded and $p^* = \infty$ if (14.1) is infeasible, analogously for (14.3).

It is known (Vandenberghe and Boyd 1996, Ben-Tal and Nemirovskii 2001) that semidefinite programs satisfy weak duality $d^* \leq p^*$, which turns into strong duality if the so-called Slater constraint qualifications are satisfied.

Theorem 14.1. (Strong Duality Theorem)

(a) If the primal problem (14.1) is strictly feasible (*i.e.*, there exist feasible positive definite matrices X_j for $j = 1, \ldots, n$) and p^* is finite, then $p^* = d^*$ and the dual supremum is attained.

(b) If the dual problem (14.3) is strictly feasible (*i.e.*, there exists some $y \in \mathbb{R}^m$ such that $C_j - \sum_{i=1}^{m} y_i A_{ij}$ are positive definite for $j = 1, \ldots, n$) and d^* is finite, then $p^* = d^*$, and the primal infimum is attained.

In general, one of the problems (14.1) and (14.3) may have optimal solutions while its dual is infeasible, or the duality gap may be positive at

optimality. This is in contrast to linear programming, where strong duality is fulfilled without any assumptions.

As has been pointed out by Neumaier and Shcherbina (2004), ill-conditioning is, for example, likely to take place in combinatorial optimization when branch-and-cut procedures sequentially generate linear or semidefinite programming relaxations. Rigorous results in combinatorial optimization can be obtained when solving the relaxations rigorously.

The following results generalize methods for linear programming by Jansson (2004*b*) and Neumaier and Shcherbina (2004). For convex programming problems that cannot be reformulated as semidefinite or conic problems, see Jansson (2004*a*). These techniques can also be generalized for computing rigorous error bounds for infinite-dimensional non-smooth conic optimization problems within the framework of functional analysis: see Jansson (2009).

The main goal is to obtain rigorous bounds by postprocessing already computed data.

Theorem 14.2. Let a semidefinite program (14.1) be given, let $\tilde{y} \in \mathbb{R}^m$, set

$$D_j := C_j - \sum_{i=1}^{m} \tilde{y}_i A_{ij} \quad \text{for } j = 1, \ldots, n, \tag{14.4}$$

and suppose that

$$\underline{d}_j \le \lambda_{\min}(D_j) \quad \text{for} \quad j = 1, \ldots, n. \tag{14.5}$$

Assume further that a primal feasible solution X_j of (14.1) is known, together with upper bounds

$$\lambda_{\max}(X_j) \le \overline{x}_j \quad \text{for} \quad j = 1, \ldots, n \tag{14.6}$$

for the maximal eigenvalues, where $\overline{x}_j$ may be infinite. If

$$\underline{d}_j \ge 0 \quad \text{for those } j \text{ with } \ \overline{x}_j = +\infty, \tag{14.7}$$

then, abbreviating $\underline{d}_j^- := \min(0, \underline{d}_j)$,

$$p^* \ge \inf\left\{ b^T \tilde{y} + \sum_{j=1}^{n} s_j \cdot \underline{d}_j^- \cdot \overline{x}_j \right\} \tag{14.8}$$

is satisfied, and the right-hand side of (14.8) is finite. Moreover, for every j with $\underline{d}_j \ge 0$,

$$\sum_{i=1}^{m} \tilde{y}_i A_{ij} - C_j \preceq 0.$$

Proof. Let $E, Y \in \mathbb{R}^{k \times k}$ be symmetric matrices with

$$\underline{d} \le \lambda_{\min}(E), \quad 0 \le \lambda_{\min}(Y), \quad \text{and} \quad \lambda_{\max}(Y) \le \overline{x}. \tag{14.9}$$

For the eigenvalue decomposition $E = Q\Lambda Q^T$, it follows that

$$\langle E, Y\rangle = \mathrm{trace}(Q\Lambda Q^T Y) = \mathrm{trace}(\Lambda Q^T Y Q) = \sum_{\nu=1}^{k} \lambda_\nu(E) e_\nu^T Q^T Y Q e_\nu.$$

Then (14.9) implies $0 \le e_\nu^T Q^T Y Q e_\nu \le \overline{x}$ and thus, using $\underline{d}^- := \min(0, \underline{d})$,

$$\langle E, Y\rangle \ge k \cdot \underline{d}^- \cdot \overline{x}. \tag{14.10}$$

The definitions (14.4) and (14.1) imply

$$\sum_{j=1}^{n} \langle C_j, X_j\rangle - b^T \tilde{y} = \sum_{j=1}^{n} \langle D_j, X_j\rangle,$$

and application of (14.10) yields

$$\sum_{j=1}^{n} \langle D_j, X_j\rangle \ge \sum_{j=1}^{n} s_j \cdot \underline{d}_j^- \cdot \overline{x}_j,$$

which proves inequality (14.8), and assumption (14.7) ensures a finite right-hand side. The last statement is an immediate consequence of $\lambda_{\min}(D_j) \ge \underline{d}_j \ge 0$. □

Note that Theorem 14.2 includes the case when no information on primal feasible solutions is available. In this case $\overline{x}_j = +\infty$ for all j.

The application of Theorem 14.2 is as follows. The lower bounds for the smallest eigenvalues as in (14.5) are calculated by the methods explained in Section 10.8.1. If (14.7) is satisfied, only (14.8) has to be evaluated. Otherwise, the constraints j violating (14.7) are relaxed by replacing C_j by $C_j - \epsilon_j I$. Then the dual optimal solution $y(\epsilon)$ satisfies the constraints

$$C_j - \sum_{i=1}^{m} y_i(\varepsilon)\, A_{ij} \succeq \varepsilon_j I,$$

increasing the minimal eigenvalues of the new defect

$$D_j(\varepsilon) := C_j - \sum_{i=1}^{m} y_i(\varepsilon) A_{ij}.$$

Some heuristic is applied to choose ϵ_j: see Jansson *et al.* (2007).

Algorithms for computing rigorous lower and upper bounds bounds for the optimal value, existence of optimal solutions and rigorous bounds for ϵ-optimal solutions as well as verified certificates of primal and dual infeasibility have been implemented in VSDP by Jansson (2006), a MATLAB toolbox for verified semidefinite programming solving based on INTLAB.

Numerical results for problems from the SDPLIB collection by Borchers (1999), a collection of large and real-life test problems, were reported by

Jansson (2009). The verification method in VSDP could compute for all problems a rigorous lower bound of the optimal value, and could verify the existence of strictly dual feasible solutions proving that all problems have a zero duality gap. A finite rigorous upper bound could be computed for all well-posed problems, with one exception (Problem hinf2). For all 32 ill-posed problems in the SDPLIB collection, VSDP computed the upper bound $\overline{f}_d = +\infty$ indicating the zero distance to primal infeasibility.

The package SDPT3 by Tütüncü, Toh and Todd (2003) (with default values), for example, gave warnings for 7 problems, where 2 warnings were given for well-posed problems. Hence, no warnings were given for 27 ill-posed problems with zero distance to primal infeasibility.

PART THREE
Infinite-dimensional problems

15. Ordinary differential equations

In the following we briefly summarize current verification methods for initial value problems; two-point boundary value problems will be discussed in the next section in more detail, because the generalization from finite-dimensional problems (Section 13) to the infinite-dimensional case is very natural.

Most approaches proceed iteratively, computing an inclusion $\mathbf{y}_{\nu+1}$ based on an inclusion $\mathbf{y}_\nu$. This verification is performed in two steps: an initial inclusion $\tilde{\mathbf{y}}_{\nu+1}$ followed by a refinement step. Because of the inevitable presence of rounding errors, the first inclusion $\mathbf{y}_1$ will have non-zero width, so that, at each step, an initial value problem has to be integrated over a set of initial values.

This leads to a fundamental problem, the *wrapping effect* already discussed in Section 9.2, similar to the widening of intervals in interval Gaussian elimination (see Section 10.1). To combat this, Moore (1966) proposed a moving coordinate system to reduce the local error. Eijgenraam (1981) proposed using higher-order approximations and inclusions. Instead of directly multiplying the transformation matrices, Lohner (1988) applies a QR-decomposition, an important advantage for long-term integration. His package AWA (Anfangswertaufgaben)[24] is well known.

Most verification methods for ODEs follow this two-step approach (see Nedialkov and Jackson (2000)), and much research has been done on improving the next initial inclusion (the first step), and improving the refinement. For the latter step, common techniques are Taylor series methods, as in AWA, constraint satisfaction methods, which are also used in global optimization (Schichl and Neumaier 2005), and the Hermite–Obreschkoff method by Nedialkov (1999). The latter is the basis for the software package VNODE. Note that these approaches contradict the UTILIZE INPUT DATA PRINCIPLE (5.13).

Finally we mention the one-phase method Cosy Infinity by Berz and Makino (1999), where the solution is directly modelled by an automatic Taylor expansion (see Section 11.4). Their code is not freely available.

However, we also mention the very detailed but also discouraging study by Bernelli azzera, Vasile, Massari and Di Lizia (2004) on the solution of space-related problems. The result is basically that long-term integration is an open problem. This was also formulated as a challenge in Neumaier (2002).

[24] Initial value problems.

15.1. Two-point boundary value problems

The ideas presented in the previous sections on finite-dimensional problems can be used to derive verification methods for infinite-dimensional problems by extending the tools used so far to Banach and Hilbert spaces.

In this section we briefly sketch a verification method for an ODE, whereas in the final section semilinear elliptic PDEs are considered in much more detail. Once more it demonstrates the principle of verification methods: to derive mathematical theorems, the assumptions of which can be verified with the aid of a computer. The result is a computer-assisted proof.

In particular, we are here concerned with a two-point boundary value problem of the form

$$\begin{aligned} -u'' &= ru^N + f, \quad 0 < x < 1, \\ u(0) &= u(1) = 0, \end{aligned} \tag{15.1}$$

for $N \in \mathbb{N}$, $r \in L^\infty(0,1)$ and $f \in L^2(0,1)$. We assume $N \geq 2$; the following is based on Nakao, Hashimoto and Watanabe (2005) and Takayasu, Oishi and Kubo (2009*a*) (the simpler linear case $N = 1$ is handled in Takayasu, Oishi and Kubo (2009*b*); see below).

The verification method transforms the problem into a nonlinear operator equation with a solution operator $\mathcal{K}$ to the Poisson equation

$$\begin{aligned} -u'' &= f, \qquad 0 < x < 1, \\ u(0) &= u(1) = 0. \end{aligned} \tag{15.2}$$

Let $(\cdot,\cdot)$ denote the inner product in $L^2(0,1)$, and let $H^m(0,1)$ denote the L^2-Sobolev space of order m. For

$$H_0^1(0,1) = \{u \in H^1(0,1) : u(0) = u(1) = 0 \text{ in the sense of traces}\}$$

with the inner product (u',v') and norm $\|u\|_{H_0^1} = \|u'\|_{L^2}$, the solution operator $\mathcal{K}$ will be regarded as a bounded linear operator from L^2 into $H^2 \cap H_0^1$.

Since the embedding $H^2 \hookrightarrow H^1$ is compact, $\mathcal{K}$ is also a compact linear operator from H_0^1 to H_0^1.

Recall that the eigenvalue problem

$$\begin{aligned} -u'' &= \lambda u, \qquad 0 < x < 1, \\ u(0) &= u(1) = 0, \end{aligned} \tag{15.3}$$

has eigenvalues $\lambda_k = k^2\pi^2$, so the minimal eigenvalue is $\lambda_{\min} = \pi^2$. The usual Rayleigh quotient estimate for $u \in H^2 \cap H_0^1$ of (15.3) implies

$$\lambda_{\min}(u,u) \leq (u',u') = (-u'',u) \leq \|u''\|_{L^2}\|u\|_{L^2},$$

hence

$$\|u\|_{L^2} \le \frac{1}{\pi}\|u'\|_{L^2} = \frac{1}{\pi}\|u\|_{H_0^1} \quad \text{and} \quad \|u\|_{H_0^1} \le \frac{1}{\pi}\|u''\|_{L^2}, \tag{15.4}$$

which means in particular

$$\|\mathcal{K}\|_{\mathcal{L}(L^2,H_0^1)} \le \frac{1}{\pi}. \tag{15.5}$$

Note that $\|u\|_{L^2} \le \frac{1}{\pi}\|u\|_{H_0^1}$ holds for all $u \in H_0^1$ since $H^2 \cap H_0^1$ is dense in H_0^1 (and the embedding $H^1 \hookrightarrow L^2$ is compact).

Recall that, for every $u \in H_0^1$,

$$|u(x)| = \left|\int_0^x u'\,\mathrm{d}t\right| \le \int_0^x |u'|\,\mathrm{d}t \quad \text{and} \quad |u(x)| = \left|\int_1^x u'\,\mathrm{d}t\right| \le \int_x^1 |u'|\,\mathrm{d}t,$$

which implies

$$2|u(x)| \le \int_0^1 |u'|\,\mathrm{d}t \le \|u'\|_{L^2} = \|u\|_{H_0^1},$$

and thus

$$\|u\|_\infty \le \frac{1}{2}\|u\|_{H_0^1}. \tag{15.6}$$

For numerical approximations we will use piecewise cubic finite element basis functions. For this purpose consider the cubic polynomials

$$\begin{aligned} N_0(x) &= (1-x)^2(1+2x), \\ N_1(x) &= x(1-x)^2, \\ N_2(x) &= x^2(3-2x), \\ N_3(x) &= -x^2(1-x), \end{aligned}$$

which form a basis of the vector space P_3 of all real polynomials of degree ≤ 3. Note that for $p \in P_3$ the corresponding dual basis $N_i^* \in P_3^*$, $0 \le i \le 3$, is given by

$$\begin{aligned} N_0^*(p) &:= p(0), \\ N_1^*(p) &:= p'(0), \\ N_2^*(p) &:= p(1), \\ N_3^*(p) &:= p'(1), \end{aligned}$$

so that $p = p(0)N_0 + p'(0)N_1 + p(1)N_2 + p'(1)N_3$.

Now let $n \in \mathbb{N}$, $h := 1/(n+1)$ and $x_i := ih$, $-1 \le i \le n+2$. Then $x_0, \dots, x_{n+1}$ is an equidistant partition of the interval $[0,1]$. The functions

$$\psi_{i,0}(x) = \begin{cases} N_2(\frac{x-x_{i-1}}{h}), & x \in [x_{i-1}, x_i] \cap [0,1], \\ N_0(\frac{x-x_i}{h}), & x \in]x_i, x_{i+1}] \cap [0,1], \\ 0, & \text{otherwise}, \end{cases} \qquad 0 \le i \le n+1,$$

$$\psi_{i,1}(x) = \begin{cases} hN_3(\frac{x-x_{i-1}}{h}), & x \in [x_{i-1}, x_i] \cap [0,1], \\ hN_1(\frac{x-x_i}{h}), & x \in]x_i, x_{i+1}] \cap [0,1], \\ 0, & \text{otherwise}, \end{cases} \qquad 0 \le i \le n+1,$$

are $2(n+2)$ linearly independent H^2-conforming finite element basis functions. We choose the ordering

$$(\psi_{0,0}, \psi_{0,1}, \psi_{1,0}, \psi_{1,1}, \dots, \psi_{n,0}, \psi_{n,1}, \psi_{n+1,1}, \psi_{n+1,0}) =: (\phi_0, \dots, \phi_{m+1}),$$

where $m := 2(n+1)$, and denote the spanned $(m+2)$-dimensional space by

$$\overline{X}_n := \operatorname{span}\{\phi_0, \dots, \phi_{m+1}\} \subset H^2. \tag{15.7}$$

The subspace spanned by the H_0^1-conforming finite element basis functions $(\phi_1, \dots, \phi_m)$ is denoted by

$$X_n := \operatorname{span}\{\phi_1, \dots, \phi_m\} \subset H^2 \cap H_0^1. \tag{15.8}$$

Note that the coefficients $v_0, \dots, v_{m+1}$ of a function

$$v = \sum_{i=0}^{m+1} v_i \phi_i \in \overline{X}_n$$

can be easily computed by $v_{2i} = v(x_i)$, $v_{2i+1} = v'(x_i)$, $i = 0, \dots, n$, and $v_m = v'(1)$, $v_{m+1} = v(1)$. The orthogonal projection $\mathcal{P}_n : H_0^1 \to X_n$ is defined by

$$(u' - (\mathcal{P}_n u)', \phi') = 0 \quad \text{for all} \quad u \in H_0^1 \quad \text{and all} \quad \phi \in X_n, \tag{15.9}$$

and the dual projection $\mathcal{Q}_n : H^2 \to \overline{X}_n$ is defined by

$$\mathcal{Q}_n(u) := \sum_{i=0}^{n} (u(x_i)\phi_{2i} + u'(x_i)\phi_{2i+1}) + u'(1)\phi_m + u(1)\phi_{m+1}, \tag{15.10}$$

where in (15.10), as usual, $u \in H^2$ is regarded as a member of $C^1([0,1])$ in which H^2 is compactly embedded. We recall basic estimates of the finite element projection errors. Even though well known in general, these estimates often do not occur explicitly for the one-dimensional case in standard textbooks. Thus, for completeness, the proofs are included.

Theorem 15.1. (FEM-projection error bounds) For $u \in H^2$ the following inequalities hold:

(a) $\|u - \mathcal{P}_n u\|_{H_0^1} \le \|u - \mathcal{Q}_n u\|_{H_0^1}$ if $u \in H^2 \cap H_0^1$,
(b) $\|u - \mathcal{Q}_n u\|_{L^2} \le \frac{h}{\pi}\|u - \mathcal{Q}_n u\|_{H_0^1}$,
(c) $\|u - \mathcal{Q}_n u\|_{H_0^1} \le \frac{h}{\pi}\|(u - \mathcal{Q}_n u)''\|_{L^2} \le \frac{h}{\pi}\|u''\|_{L^2}$,
(d) $\|u - \mathcal{Q}_n u\|_{H_0^1} \le \frac{h^2}{\pi^2}\|u'''\|_{L^2}$ if $u \in H^3 \cap H_0^1$.

Proof. **(a)** Since $u \in H^2 \cap H_0^1$, $\mathcal{Q}_n u \in X_n$ and therefore $\mathcal{P}_n u - \mathcal{Q}_n u \in X_n$. It follows from the definition of the orthogonal projection (15.9) that $((u - \mathcal{P}_n u)', (\mathcal{P}_n u - \mathcal{Q}_n u)') = 0$, which gives the Pythagorean identity

$$\|u - \mathcal{Q}_n u\|_{H_0^1}^2 = \|u - \mathcal{P}_n u\|_{H_0^1}^2 + \|\mathcal{P}_n u - \mathcal{Q}_n u\|_{H_0^1}^2 \ge \|u - \mathcal{P}_n u\|_{H_0^1}^2.$$

(b) The function $e := u - \mathcal{Q}_n u$ fulfils $e(x_i) = 0$ for each $i \in \{0, \dots, n+1\}$. Hence each function $e(x_i + hx)$, $i \in \{0, \dots, n\}$, belongs to H_0^1. Now (15.4) supplies

$$\begin{aligned}\int_0^1 e(x_i + hx)^2 \,\mathrm{d}x = \|e(x_i + hx)\|_{L^2}^2 &\le \frac{1}{\pi^2}\|(e(x_i + hx))'\|_{L^2}^2 \\ &= \frac{h}{\pi^2}\int_0^1 h e'(x_i + hx)^2 \,\mathrm{d}x = \frac{h}{\pi^2}\int_{x_i}^{x_{i+1}} e'(x)^2 \,\mathrm{d}x.\end{aligned}$$

Thus

$$\begin{aligned}\|e\|_{L^2}^2 &= \sum_{i=0}^{n}\int_{x_i}^{x_{i+1}} e(x)^2 \,\mathrm{d}x = h\sum_{i=0}^{n}\int_{x_i}^{x_{i+1}} e\left(x_i + h\cdot\frac{x - x_i}{h}\right)^2 \frac{1}{h}\,\mathrm{d}x \\ &= h\sum_{i=0}^{n}\int_0^1 e(x_i + hx)^2 \,\mathrm{d}x \le \frac{h^2}{\pi^2}\sum_{i=0}^{n}\int_{x_i}^{x_{i+1}} e'(x)^2 \,\mathrm{d}x = \frac{h^2}{\pi^2}\|e\|_{H_0^1}^2.\end{aligned}$$

(c) The function $d := (u - \mathcal{Q}_n u)'$ fulfils $d(x_i) = 0$ for each $i \in \{0, \dots, n+1\}$. Therefore $d \in H_0^1$ as $u, \mathcal{Q}_n u \in H^2$, and $\|d\|_{L^2} \le \frac{h}{\pi}\|d'\|_{L^2}$ follows by the same computation as in (b) with d instead of e. This is the first part,

$$\|u - \mathcal{Q}_n u\|_{H_0^1} \le \frac{h}{\pi}\|(u - \mathcal{Q}_n u)''\|_{L^2},$$

of the inequality in (c). Since $\mathcal{Q}_n u$ is a piecewise cubic polynomial,

$$(\mathcal{Q}_n u|_{[x_i, x_{i+1}]})'''' = 0$$

supplies

$$\begin{aligned}&\frac{1}{2}\big(-\|(u - \mathcal{Q}_n u)''\|_{L^2}^2 + \|u''\|_{L^2}^2 - \|(\mathcal{Q}_n u)''\|_{L^2}^2\big) \qquad (15.11)\\ &\quad = \int_0^1 (\mathcal{Q}_n u)''(u - \mathcal{Q}_n u)'' \,\mathrm{d}x\end{aligned}$$

$$= \sum_{i=0}^{n} [(\mathcal{Q}_n u)''(u - \mathcal{Q}_n u)']_{x_i}^{x_{i+1}} - [(\mathcal{Q}_n u)'''(u - \mathcal{Q}_n u)]_{x_i}^{x_{i+1}} + \int_{x_i}^{x_{i+1}} (\mathcal{Q}_n u)''''(\mathcal{Q}_n u)\,\mathrm{d}x = 0.$$

Thus

$$\|(u - \mathcal{Q}_n u)''\|_{L^2}^2 = \|u''\|_{L^2}^2 - \|(\mathcal{Q}_n u)''\|_{L^2}^2 \le \|u''\|_{L^2}^2,$$

giving the second part of the asserted inequality.

(d) If $u \in H^3 \cap H_0^1$, then using (15.11), the Cauchy–Schwarz inequality and (c) yield

$$\|(u - \mathcal{Q}_n u)''\|_{L^2}^2 = \int_0^1 u''(u - \mathcal{Q}_n u)''\,\mathrm{d}x = -\int_0^1 u'''(u - \mathcal{Q}_n u)'\,\mathrm{d}x$$
$$\le \|u'''\|_{L^2} \|(u - \mathcal{Q}_n u)'\|_{L^2} \le \frac{h}{\pi} \|u'''\|_{L^2} \|(u - \mathcal{Q}_n u)''\|_{L^2}.$$

Therefore

$$\|(u - \mathcal{Q}_n u)''\|_{L^2} \le \frac{h}{\pi} \|u'''\|_{L^2},$$

which, inserted into (c), gives

$$\|u - \mathcal{Q}_n u\|_{H_0^1} \le \frac{h^2}{\pi^2} \|u'''\|_{L^2}. \qquad \square$$

Applying the solution operator $\mathcal{K}$ for (15.2), the problem (15.1) is transformed into $u = \mathcal{K}(ru^N + f)$, or equivalently into the nonlinear operator equation

$$F(u) := u - \mathcal{K} r u^N - \mathcal{K} f = 0. \tag{15.12}$$

Based on an approximate finite element solution $\hat{u}$, the Newton–Kantorovich theorem is applicable. Note that $\hat{u}$ can be computed by any standard (approximate) numerical method.

Theorem 15.2. (Newton–Kantorovich) Let F be given as in (15.12), and assume the Fréchet derivative $F'(\hat{u})$ is non-singular and satisfies

$$\|F'(\hat{u})^{-1} F(\hat{u})\|_{H_0^1} \le \alpha \tag{15.13}$$

for some positive α. Furthermore, assume

$$\|F'(\hat{u})^{-1}(F'(v) - F'(w))\|_{\mathcal{L}(H_0^1, H_0^1)} \le \omega \|v - w\|_{H_0^1} \tag{15.14}$$

for some positive ω and for all $v, w \in U_{2\alpha}(\hat{u}) := \{z \in H_0^1, \|z - \hat{u}\|_{H_0^1} < 2\alpha\}$. If

$$\alpha\omega \le \frac{1}{2}, \tag{15.15}$$

then $F(u) = 0$ has a solution u satisfying

$$\|u - \hat{u}\|_{H_0^1} \le \rho := \frac{1 - \sqrt{1 - 2\alpha\omega}}{\omega}. \tag{15.16}$$

Within this bound, the solution is unique.

Since $\mathcal{K} : H_0^1 \to H_0^1$ is a compact operator, $r \in L^\infty(0,1)$ and $\hat{u} \in X_n \subset L^\infty$, the operator

$$\mathcal{T} : H_0^1 \to H_0^1, \quad v \mapsto \mathcal{K}Nr\hat{u}^{N-1}v \tag{15.17}$$

is compact as well. The verification method is based on the computation of constants C_1, C_2 and C_3 with

$$\|F'(\hat{u})^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)} = \|(I - \mathcal{T})^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)} \le C_1, \tag{15.18}$$

$$\|F(\hat{u})\|_{H_0^1} = \|\hat{u} - \mathcal{K}r\hat{u}^N - \mathcal{K}f\|_{H_0^1} \le C_2, \tag{15.19}$$

where I denotes the identity on H_0^1, and

$$\|F'(v) - F'(w)\|_{\mathcal{L}(H_0^1,H_0^1)} \le C_3\|v - w\|_{H_0^1} \quad \text{for all} \quad v, w \in U_{2\alpha}(\hat{u}). \tag{15.20}$$

If (15.15) is satisfied for

$$\alpha := C_1C_2 \quad \text{and} \quad \omega := C_1C_3, \tag{15.21}$$

then (15.16) follows. The main part, the estimation of C_1, uses the following result by Oishi (2000).

Theorem 15.3. Let $\mathcal{T} : H_0^1 \to H_0^1$ be a compact linear operator. Assume that $\mathcal{P}_n\mathcal{T}$ is bounded by

$$\|\mathcal{P}_n\mathcal{T}\|_{\mathcal{L}(H_0^1,H_0^1)} \le K, \tag{15.22}$$

that the difference between $\mathcal{T}$ and $\mathcal{P}_n\mathcal{T}$ is bounded by

$$\|\mathcal{T} - \mathcal{P}_n\mathcal{T}\|_{\mathcal{L}(H_0^1,H_0^1)} \le L, \tag{15.23}$$

and that the finite-dimensional operator $(I - \mathcal{P}_n\mathcal{T})|_{X_n} : X_n \to X_n$ is invertible with

$$\|(I - \mathcal{P}_n\mathcal{T})|_{X_n}^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)} \le M. \tag{15.24}$$

If $(1 + MK)L < 1$, then the operator $(I - \mathcal{T}) : H_0^1 \to H_0^1$ is invertible and

$$\|(I - \mathcal{T})^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)} \le \frac{1 + MK}{1 - (1 + MK)L}.$$

Proof. Using the finite-dimensional operator $(I - \mathcal{P}_n\mathcal{T})|_{X_n}^{-1} : X_n \to X_n$, one can show by direct computation that

$$(I - \mathcal{P}_n\mathcal{T})^{-1} = I + (I - \mathcal{P}_n\mathcal{T})|_{X_n}^{-1}\mathcal{P}_n\mathcal{T}.$$

The assumptions imply that this inverse operator is bounded by

$$\begin{aligned}\|(I-\mathcal{P}_n\mathcal{T})^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)} &\le 1+\|(I-\mathcal{P}_n\mathcal{T})|_{X_n}^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)}\cdot\|\mathcal{P}_n\mathcal{T}\|_{\mathcal{L}(H_0^1,H_0^1)}\\ &\le 1+MK.\end{aligned}$$

Moreover, for $\phi\in H_0^1$ and $\psi:=(I-\mathcal{T})\phi$, using

$$\phi=\phi-(I-\mathcal{P}_n\mathcal{T})^{-1}(I-\mathcal{T})\phi+(I-\mathcal{P}_n\mathcal{T})^{-1}\psi,$$

we have

$$\begin{aligned}\|\phi\|_{H_0^1} &\le \|I-(I-\mathcal{P}_n\mathcal{T})^{-1}(I-\mathcal{T})\|_{\mathcal{L}(H_0^1,H_0^1)}\|\phi\|_{H_0^1}+\|(I-\mathcal{P}_n\mathcal{T})^{-1}\psi\|_{H_0^1}\\ &\le \|(I-\mathcal{P}_n\mathcal{T})^{-1}\|_{\mathcal{L}(H_0^1,H_0^1)}\|\mathcal{T}-\mathcal{P}_n\mathcal{T}\|_{\mathcal{L}(H_0^1,H_0^1)}\|\phi\|_{H_0^1}+(1+MK)\|\psi\|_{H_0^1}\\ &\le (1+MK)L\|\phi\|_{H_0^1}+(1+MK)\|\psi\|_{H_0^1}, \qquad (15.25)\end{aligned}$$

and therefore

$$\|\phi\|_{H_0^1}\le\frac{1+MK}{1-(1+MK)L}\|(I-\mathcal{T})\phi\|_{H_0^1}.$$

This implies that $(I-\mathcal{T}):H_0^1\to H_0^1$ is injective, and by the Fredholm alternative it has a bounded inverse satisfying $\phi=(I-\mathcal{T})^{-1}\psi$. The theorem follows. □

To estimate C_1 by Theorem 15.3, three constants K, L and M are needed, which can be computed as follows. Using (15.4) and (15.5), we obtain for $u\in H_0^1$

$$\|\mathcal{P}_n\mathcal{T}u\|_{H_0^1}\le\|\mathcal{T}u\|_{H_0^1}\le\frac{1}{\pi}N\|r\hat{u}^{N-1}\|_\infty\|u\|_{L^2}\le\frac{1}{\pi^2}N\|r\hat{u}^{N-1}\|_\infty\|u\|_{H_0^1},$$

which implies that (15.22) holds for

$$K:=\frac{1}{\pi^2}N\|r\hat{u}^{N-1}\|_\infty. \qquad (15.26)$$

Furthermore, applying the error bound

$$\|(I-\mathcal{P}_n)v\|_{H_0^1}\le\frac{h}{\pi}\|v''\|_{L^2}$$

for $v\in H^2\cap H_0^1$ (see Theorem 15.1(a), (c)), we obtain, using (15.4),

$$\|(\mathcal{T}-\mathcal{P}_n\mathcal{T})u\|_{H_0^1}\le\frac{h}{\pi}\|(\mathcal{T}u)''\|_{L^2}=\frac{h}{\pi}N\|r\hat{u}^{N-1}u\|_{L^2}\le\frac{h}{\pi^2}N\|r\hat{u}^{N-1}\|_\infty\|u\|_{H_0^1}.$$

Hence (15.23) holds for

$$L:=\frac{h}{\pi^2}N\|r\hat{u}^{N-1}\|_\infty=hK. \qquad (15.27)$$

If $r \in W^{1,\infty}$, the L^∞-Sobolev space of order 1, then $\mathcal{T}u \in H^3 \cap H_0^1$, and using Theorem 15.1(a), (d) gives

$$\begin{aligned}\|(\mathcal{T} - \mathcal{P}_n\mathcal{T})u\|_{H_0^1} &\\ &\le \frac{h^2}{\pi^2}\|(\mathcal{T}u)'''\|_{L^2} = \frac{h^2}{\pi^2}N\|(r\hat{u}^{N-1}u)'\|_{L^2}\\ &\le \frac{h^2}{\pi^2}N\left(\frac{1}{\pi}\|r'\hat{u}^{N-1} + (N-1)r\hat{u}^{N-2}\hat{u}'\|_\infty + \|r\hat{u}^{N-1}\|_\infty\right)\|u\|_{H_0^1},\end{aligned}$$

and (15.23) holds for the $O(h^2)$ bound

$$L := \frac{h^2}{\pi^2}N\left(\frac{1}{\pi}\|r'\hat{u}^{N-1} + (N-1)r\hat{u}^{N-2}\hat{u}'\|_\infty + \|r\hat{u}^{N-1}\|_\infty\right). \tag{15.28}$$

Condition (15.24) requires us to calculate a constant M satisfying

$$\left\|(I - \mathcal{P}_n\mathcal{T})|_{X_n}^{-1}\left(\sum_{i=1}^m v_i\phi_i\right)\right\|_{H_0^1} \le M\left\|\sum_{i=1}^m v_i\phi_i\right\|_{H_0^1} \quad \text{for all } \ v_1,\dots,v_m \in \mathbb{R},$$

and thus

$$\sum_{i,j=1}^m v_iv_j\int_0^1 \psi_i'\psi_j'\,\mathrm{d}x \le M^2\sum_{i,j=1}^m v_iv_j\int_0^1 \phi_i'\phi_j'\,\mathrm{d}x, \tag{15.29}$$

where

$$\psi_i = \sum_{j=1}^m \alpha_{ij}\phi_j := (I - \mathcal{P}_n\mathcal{T})|_{X_n}^{-1}\phi_i, \quad i.e., \quad (I - \mathcal{P}_n\mathcal{T})\psi_i = \phi_i.$$

Taking inner H_0^1-products with ϕ_k, $k = 1,\dots,m$, the latter is equivalent to

$$\sum_{j=1}^m \alpha_{ij}\left[(\phi_j',\phi_k') - ((\mathcal{T}\phi_j)',\phi_k')\right] = (\phi_i',\phi_k'). \tag{15.30}$$

By partial integration,

$$((\mathcal{T}\phi_j)',\phi_k') = -((\mathcal{T}\phi_j)'',\phi_k) = N(r\hat{u}^{N-1}\phi_j,\phi_k).$$

Thus, defining $A = (\alpha_{ij})_{1\le i,j\le m}$ and

$$\begin{aligned}R &:= ((\phi_j',\phi_k') - N(r\hat{u}^{N-1}\phi_j,\phi_k))_{1\le j,k\le m},\\ G &:= ((\phi_i',\phi_k'))_{1\le i,k\le m},\end{aligned} \tag{15.31}$$

(15.30) reads $AR = G$, whence $A = GR^{-1}$. Note that all integrals have to be bounded rigorously. This can be done, for example, using the INTLAB algorithm `verifyquad` described in Section 12. Also note that the invertibility of R, which is equivalent to the invertibility of the operator

$(I - \mathcal{P}_n\mathcal{T})|_{X_n}$, has to be verified. The methods described below ensure this. Now

$$(\psi_i', \psi_j') = \sum_{k,l=1}^{m} \alpha_{ik}\alpha_{jl}(\phi_k', \phi_l') = (AGA^T)_{ij} = (GR^{-1}GR^{-1}G)_{ij}.$$

Insertion into (15.29) gives the equivalent condition

$$v^T(GR^{-1}GR^{-1}G)v \le M^2 v^T G v \quad \text{for all} \quad v \in \mathbb{R}^m,$$

and thus (15.24) holds for

$$M := \sqrt{\lambda_{\max}}, \tag{15.32}$$

with $\lambda_{\max}$ denoting the largest (in absolute value) eigenvalue of the matrix eigenvalue problem

$$GR^{-1}GR^{-1}Gv = \lambda Gv. \tag{15.33}$$

Since M is equal to the spectral radius of $R^{-1}G$, an upper bound can be computed in INTLAB, for example, as follows. First, `C = verifylss(R,G)` (see Section 10.5) verifies that R is non-singular and computes an inclusion `C` of $R^{-1}G$, so that $M \le \|R^{-1}G\|_2 \le \sqrt{\|R^{-1}G\|_\infty \|R^{-1}G\|_1}$ and

$$\texttt{Mb} = \texttt{mag}(\texttt{sqrt}(\texttt{norm}(\texttt{C},\texttt{inf}) * \texttt{norm}(\texttt{C},1))) \quad \text{implies} \quad M \le \texttt{Mb}. \tag{15.34}$$

Another way to estimate M is by a (verified) similarity transformation and Gershgorin circles as follows:

```
[V,D] = eig(mid(C));
Y = verifylss(V,C*V);
Mb = max(mag(gershgorin(Y)));
```

Again $M \le$ `Mb` holds. Neither approach uses the symmetry of R and G. Alternatively, note that for a complex eigenvalue/eigenvector pair (v, λ) of (15.33) it follows that

$$\overline{v}^T GR^{-1}GR^{-1}Gv = \lambda \overline{v}^T G v. \tag{15.35}$$

But $\overline{v}^T G v > 0$ because G is positive definite, and the left-hand side of (15.35) is real because $GR^{-1}GR^{-1}G$ is symmetric, so that λ must be real and non-zero. Moreover, λ is an eigenvalue of $(R^{-1}G)^2$, so $\lambda > 0$, and, because G is positive definite, $\lambda' G - GR^{-1}GR^{-1}G$ is positive semidefinite if and only if $\lambda' \ge \lambda_{\max}$. The latter bounds are often significantly superior to (15.34).

Thus, choosing λ' a little larger than an approximation of $\lambda_{\max}$, and verifying that $\lambda' G - GR^{-1}GR^{-1}G$ is positive semidefinite by algorithm `isspd` discussed in Section 10.8.1, proves $M \le \sqrt{\lambda'}$. We mention that the latter two approaches yield significantly better estimates of M than (15.34).

With K, L, M computed according to (15.26), (15.27) (or (15.28) if $r \in W^{1,\infty}$) and (15.32), a constant

$$C_1 := \frac{1+MK}{1-(1+MK)L} \tag{15.36}$$

satisfying (15.18) is obtained via Theorem 15.3, provided $(1+MK)L < 1$ (which requires h to be sufficiently small).

It follows from (15.5) that

$$\begin{aligned}\|\hat{u} - \mathcal{K}(r\hat{u}^N) - \mathcal{K}f\|_{H_0^1} &= \|\mathcal{K}(-\hat{u}'' - r\hat{u}^N - f)\|_{H_0^1} \\ &\le \frac{1}{\pi}\|\hat{u}'' + r\hat{u}^N + f\|_{L^2} =: C_2. \end{aligned}\tag{15.37}$$

The constant C_2 is expected to be small if the approximation $\hat{u}$ is sufficiently accurate.

In the following, another estimate for a defect bound $\|\hat{u} - \mathcal{K}(r\hat{u}^N + f)\|_{H_0^1} \le C_2$ is given, which naturally leads to a Newton method for finding $\hat{u}$. For simplicity we assume from now on that $r, f \in H^2$, so that the dual projection $\mathcal{Q}_n$ can be applied to these functions. The main idea is to replace $\|\hat{u} - \mathcal{K}(r\hat{u}^N + f)\|_{H_0^1}$ by a projected finite element defect, namely by $\|\hat{u} - \mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1}$. Note that $r\hat{u}^N \in H^2$, since H^2 is an algebra in the one-dimensional case. Using the triangle inequality, Theorem 15.1(a), (d) and (15.5) imply

$$\begin{aligned}&\|\hat{u} - \mathcal{K}(r\hat{u}^N + f)\|_{H_0^1} \\ &\quad\le \|\hat{u} - \mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1} + \|\mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f) - \mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1} \\ &\qquad + \|\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f) - \mathcal{K}(r\hat{u}^N + f)\|_{H_0^1} \\ &\quad= \|\hat{u} - \mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1} + \|(\mathcal{P}_n - I)\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1} \\ &\qquad + \|\mathcal{K}(\mathcal{Q}_n - I)(r\hat{u}^N + f)\|_{H_0^1} \\ &\quad\le \|\hat{u} - \mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1} + \frac{h^2}{\pi^2}\|(\mathcal{Q}_n(r\hat{u}^N + f))'\|_{L^2} \\ &\qquad + \frac{1}{\pi}\|(\mathcal{Q}_n - I)(r\hat{u}^N + f)\|_{L^2}.\end{aligned}$$

The three summands in the last line are abbreviated by

$$\begin{aligned}C'_{21} &:= \|\hat{u} - \mathcal{P}_n\mathcal{K}\mathcal{Q}_n(r\hat{u}^N + f)\|_{H_0^1}, \\ C_{22} &:= \frac{h^2}{\pi^2}\|(\mathcal{Q}_n(r\hat{u}^N + f))'\|_{L^2}, \\ C_{23} &:= \frac{1}{\pi}\|(\mathcal{Q}_n - I)(r\hat{u}^N + f)\|_{L^2}.\end{aligned}$$

Furthermore, define the mass and stiffness matrices

$$H := ((\phi_i, \phi_j))_{1\le i\le m,\ 0\le j\le m+1} \in \mathbb{R}^{m,m+2},$$
$$\overline{G} := ((\phi_i', \phi_j'))_{0\le i,j\le m+1} \in \mathbb{R}^{m+2,m+2}.$$

Recall that the symmetric positive definite matrix

$$G = ((\phi_i', \phi_j'))_{1\le i,j\le m} \in \mathbb{R}^{m,m},$$

defined in (15.31), is the inner submatrix of $\overline{G}$ of order m.

Let $\overline{u}_h = (u_0, \ldots, u_{m+1}) \in \mathbb{R}^{m+2}$ denote the vector representation of $u \in \overline{X}_n$ with respect to the basis $(\phi_0, \ldots, \phi_{m+1})$, and analogously let $u_h = (u_1, \ldots, u_m) \in \mathbb{R}^m$ denote the vector representation of $u \in X_n$ with respect to the basis $(\phi_1, \ldots, \phi_m)$ of X_n. For $v \in \overline{X}_n$ and $u := \mathcal{P}_n\mathcal{K}v \in X_n$, for each $i = 1, \ldots, m$ the definition of the orthogonal projection (15.9) yields

$$\begin{aligned}(Gu_h)_i &= \sum_{j=1}^{m} u_j(\phi_i', \phi_j') = (\phi_i', u') = (\phi_i', (\mathcal{P}_n\mathcal{K}v)') = (\phi_i', (\mathcal{K}v)') \\ &= -(\phi_i, (\mathcal{K}v)'') = (\phi_i, v) = \sum_{j=0}^{m+1} v_j(\phi_i, \phi_j) = (H\overline{v}_h)_i.\end{aligned}$$

Hence $u_h = G^{-1}H\overline{v}_h$ shows that the matrix representation of $\mathcal{P}_n\mathcal{K}|_{\overline{X}_n} : \overline{X}_n \to X_n$ with respect to the bases $(\phi_0, \ldots, \phi_{m+1})$ of $\overline{X}_n$ and $(\phi_1, \ldots, \phi_m)$ of X_n is $G^{-1}H$. Now define $v := \mathcal{Q}_n(r\hat{u}^N + f)$. Then

$$\begin{aligned}C_{21}' &= \|\hat{u} - \mathcal{P}_n\mathcal{K}v\|_{H_0^1} = [(\hat{u}_h - G^{-1}H\overline{v}_h)^t G(\hat{u}_h - G^{-1}H\overline{v}_h)]^{\frac{1}{2}} \qquad (15.38)\\ &= [(G\hat{u}_h - H\overline{v}_h)^t G^{-1}(G\hat{u}_h - H\overline{v}_h)]^{\frac{1}{2}} \le \frac{1}{\sqrt{\lambda_{\min}(G)}}\|G\hat{u}_h - H\overline{v}_h\|_2,\end{aligned}$$

where $\lambda_{\min}(G)$ denotes the smallest eigenvalue of the symmetric positive definite matrix G. A positive lower bound $\texttt{lambda} \le \lambda_{\min}(G)$ can be estimated and verified using INTLAB, for example, by the following pattern:

```
lambda = 0.99*min(eig(mid(G)));
I_m = eye(m);
while not(isspd(G-lambda*I_m))
    lambda = 0.99*lambda;
end
```

Here `isspd` verifies that a symmetric matrix is positive definite; see Section 10.8.1. Note that the loop was never executed in our examples.

Having computed `lambda` successfully, define

$$C_{21} := \frac{1}{\sqrt{\texttt{lambda}}}\|G\hat{u}_h - H\overline{v}_h\|_2.$$

Then the constant

$$C_2 := C_{21} + C_{22} + C_{23} \tag{15.39}$$

is an upper bound for $\|\hat{u} - \mathcal{K}(r\hat{u}^N + f)\|_{H_0^1}$. Besides

$$C_{22} = \frac{h^2}{\pi^2}\sqrt{\overline{v}_h^t \overline{G}\overline{v}_h},$$

we also have that

$$C_{23} = \frac{1}{\pi}\|(\mathcal{Q}_n - I)(r\hat{u}^N + f)\|_{L^2} \le \frac{h^2}{\pi^3}\|(r\hat{u}^N + f)''\|_{L^2}$$

(see Theorem 15.1(b), (c)) is of order $O(h^2)$, so that in general C_2 cannot be expected to be of higher order.

Line (15.38) suggests a Newton method for determining an approximate solution $\hat{u}$, which is described below. Define

$$g : \mathbb{R}^m \to \mathbb{R}^m, \quad u \mapsto Gu - H\overline{v}_h,$$

where

$$v := \mathcal{Q}_n\left(r\left(\sum_{i=1}^{m} u_i\phi_i\right)^N + f\right) \in \overline{X}_n,$$

with vector representation $\overline{v}_h = (v_0, \ldots, v_{m+1}) \in \mathbb{R}^{m+2}$. For $\mathcal{Q}_n f,\ \mathcal{Q}_n r \in \overline{X}_n$ with vector representations

$$\begin{aligned}\overline{f}_h &= (f(0), f'(0), f(h), f'(h), \ldots, f(1-h), f'(1-h), f'(1), f(1)) \\ &= (f_0, \ldots, f_{m+1}),\end{aligned}$$

and similarly $\overline{r}_h = (r_0, \ldots, r_{m+1})$, respectively, it follows, utilizing $N \ge 2$, for $i \in \{0, \ldots, m+1\}$, that

$$v_i = \begin{cases} f_i, & \text{if } i \in \{0, 1, m, m+1\}, \\ r_i u_i^N + f_i, & \text{if } i \in \{2, \ldots, m-2\} \text{ is even}, \\ r_i u_{i-1}^N + N r_{i-1} u_{i-1}^{N-1} u_i + f_i, & \text{if } i \in \{3, \ldots, m-1\} \text{ is odd}. \end{cases}$$

Hence the Jacobian J of g in $u \in \mathbb{R}^m$ reads, in MATLAB notation,

```
J = G;
for j = [2:2:m-2]
  J(:,j) = J(:,j)-N*(H(:,j)*r(j)*u(j)^(N-1)...
                    +H(:,j+1)*(r(j+1)*u(j)^(N-1)...
                    +(N-1)*r(j)*u(j)^(N-2)*u(j+1)));
end
for j = [3:2:m-1]
  J(:,j) = J(:,j)-N*H(:,j)*r(j-1)*u(j-1)^(N-1);
end
```

and a Newton step improves an approximation u into $u - du$ with $du := J^{-1}(Gu - H\overline{v}_h)$. In practice, of course, du is calculated as an approximate solution of the linear system with matrix J and right-hand side $Gu - H\overline{v}_h$.

Finally, to compute a constant C_3 satisfying (15.20), we use (15.5) and (15.6) to obtain, for $v, w, \varphi \in H_0^1$,

$$\begin{aligned}
&\|(F'(v) - F'(w))\varphi\|_{H_0^1} \\
&\quad = \|\mathcal{K}Nr(v^{N-1} - w^{N-1})\varphi\|_{H_0^1} \le \frac{N}{\pi}\|r(v^{N-1} - w^{N-1})\varphi\|_{L^2} \\
&\quad \le \frac{N}{\pi}\|r(v^{N-1} - w^{N-1})\|_{L^2}\|\varphi\|_\infty \le \frac{N}{2\pi}\|r(v^{N-1} - w^{N-1})\|_{L^2}\|\varphi\|_{H_0^1}.
\end{aligned}$$

If

$$\|v - \hat{u}\|_{H_0^1} < 2\alpha, \quad \|w - \hat{u}\|_{H_0^1} < 2\alpha,$$

and thus

$$\|v - \hat{u}\|_\infty < \alpha, \quad \|w - \hat{u}\|_\infty < \alpha$$

by (15.6), this means, using (15.4), that

$$\begin{aligned}
&\|F'(v) - F'(w)\|_{\mathcal{L}(H_0^1, H_0^1)} \\
&\quad \le \frac{N}{2\pi}\|r(v^{N-1} - w^{N-1})\|_{L^2} \\
&\quad = \frac{N}{2\pi}\|r(v^{N-2} + v^{N-3}w + \cdots + vw^{N-3} + w^{N-2})(v - w)\|_{L^2} \\
&\quad \le \frac{N}{2\pi}\|r\|_\infty(\|v\|_\infty^{N-2} + \|v\|_\infty^{N-3}\|w\|_\infty + \cdots + \|w\|_\infty^{N-2})\|v - w\|_{L^2} \\
&\quad \le \frac{N(N-1)}{2\pi^2}\|r\|_\infty(\|\hat{u}\|_\infty + \alpha)^{N-2}\|v - w\|_{H_0^1},
\end{aligned}$$

which gives

$$C_3 := \frac{N(N-1)}{2\pi^2}\|r\|_\infty(\|\hat{u}\|_\infty + \alpha)^{N-2}. \tag{15.40}$$

If $\alpha\omega = C_1^2 C_2 C_3 \le \frac{1}{2}$, then the Newton–Kantorovich theorem implies that there exists a unique solution $u \in \overline{U_\rho(\hat{u})}$ satisfying the two-point boundary value problem (15.1), with ρ defined in (15.16).[25]

As a numerical example, consider

$$\begin{aligned}
&-u'' = (x+2)^3 u^3 - \cos 2\pi x, \quad 0 < x < 1, \\
&u(0) = u(1) = 0.
\end{aligned} \tag{15.41}$$

[25] Even though excluded in the beginning, the linear case $N = 1$ can be treated in a similar but more direct way resulting in $\|u - \hat{u}\|_{H_0^1} \le \alpha = C_1 C_2$. The constant C_3 is not needed.

Table 15.1. Guaranteed error estimates.

		Number of grid points			
		65	129	257	513
u_{-1}	$\alpha\omega$	0.85	$1.88 \cdot 10^{-1}$	$4.56 \cdot 10^{-2}$	$1.14 \cdot 10^{-2}$
	ρ	–	$2.39 \cdot 10^{-3}$	$5.41 \cdot 10^{-4}$	$1.34 \cdot 10^{-4}$
u_0	$\alpha\omega$	$4.77 \cdot 10^{-5}$	$1.18 \cdot 10^{-5}$	$2.95 \cdot 10^{-6}$	$7.37 \cdot 10^{-7}$
	ρ	$1.11 \cdot 10^{-4}$	$2.79 \cdot 10^{-5}$	$6.97 \cdot 10^{-6}$	$1.74 \cdot 10^{-6}$
u_1	$\alpha\omega$	0.59	$1.34 \cdot 10^{-1}$	$3.27 \cdot 10^{-2}$	$8.19 \cdot 10^{-3}$
	ρ	–	$2.09 \cdot 10^{-3}$	$4.88 \cdot 10^{-4}$	$1.21 \cdot 10^{-4}$

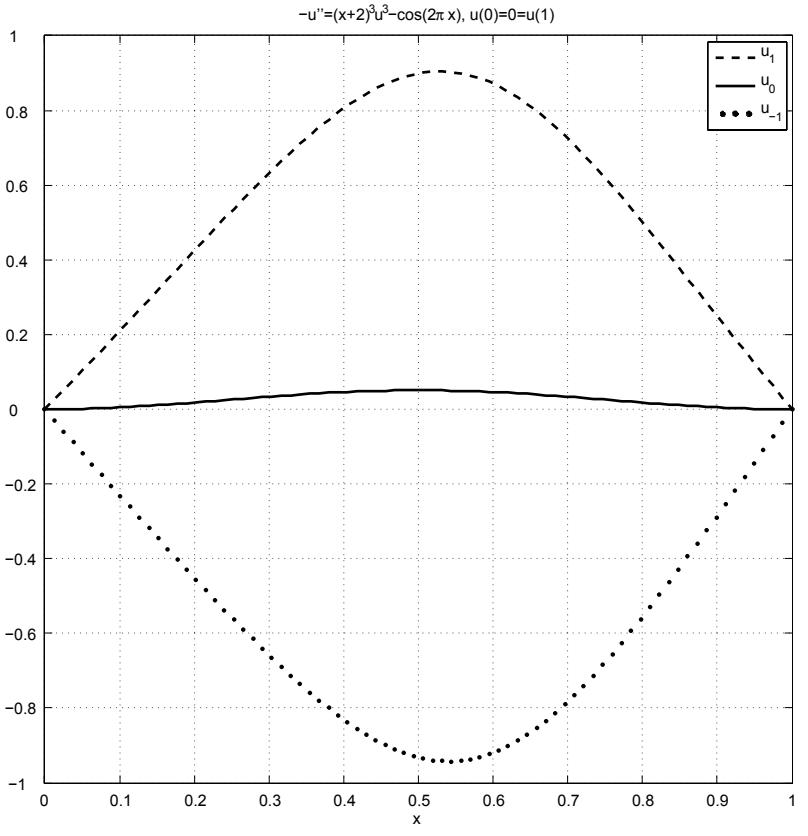

Figure 15.1. Verified inclusions for three distinct solutions of (15.41)

Three approximate solutions $\hat{u}_{-1}$, $\hat{u}_0$, $\hat{u}_1$ are obtained by some Newton iterates starting with $\hat{u}^0_{-1} \equiv -1$, $\hat{u}^0_0 \equiv 0$, $\hat{u}^0_1 \equiv 1$, respectively. The computed bounds for $\alpha\omega$ and ρ are displayed in Table 15.1. If the number of grid points $n+1$ is at least 129, then $\alpha\omega < 0.5$ in all three cases. It follows that there exist solutions u_{-1}, u_0 and u_1 of (15.41) in the ball centred at $\hat{u}_i$ with corresponding radius ρ, *i.e.*, $\|u_i - \hat{u}_i\|_{H^1_0} \le \rho$ and therefore

$$\|u_i - \hat{u}_i\|_\infty \le \frac{1}{2}\rho \quad \text{for} \quad i = -1, 0, 1.$$

Since ρ is sufficiently small, these solutions are pairwise distinct. Note that $\|u_{-1}\|_\infty \approx 0.91$, $\|u_0\|_\infty \approx 0.05$ and $\|u_1\|_\infty \approx 0.95$, so that for all three solutions the relative error near the extremum is about 10^{-4}.

Figure 15.1 displays the $\|\cdot\|_\infty$-inclusions for these three solutions. The radius ρ is so small that the upper and lower bounds seem to lie on one single line. At first glance Figure 15.1 might suggest $u_{-1} = -u_1$, which is of course not the case, on checking (15.41) for symmetries. We do not know whether there are more solutions of (15.41) besides u_1, u_0, u_{-1}.

16. Semilinear elliptic boundary value problems (by Michael Plum, Karlsruhe)

In this final section we will describe in more detail a verification method for semilinear elliptic boundary value problems of the form

$$-\Delta u(x) + f(x, u(x)) = 0 \quad (x \in \Omega), \quad u(x) = 0 \quad (x \in \partial\Omega), \tag{16.1}$$

with $\Omega \subset \mathbb{R}^n$ denoting some given domain, and $f : \Omega \times \mathbb{R} \to \mathbb{R}$ some given nonlinearity.

Such problems have been (and still are) extensively studied in the differential equations literature, and they have a lot of applications, for instance in mathematical physics. Often they serve as model problems for more complex mathematical situations.

Starting perhaps with Picard's successive iterations at the end of the nineteenth century,various analytical methods and techniques have been (and are being) developed to study existence and multiplicity of solutions to problem (16.1), among them variational methods (including mountain pass methods), index and degree theory, monotonicity methods, fixed-point methods, and more. However, many questions remain open, offering opportunities for computer-assisted proofs and verification methods to supplement these purely analytical approaches.

As for finite-dimensional problems, we start with an approximate solution $\tilde{u}$ in some suitable function space and rewrite (16.1) as a boundary value problem for the error $v = u - \tilde{u}$. This is transformed into an equivalent fixed-point equation,

$$v \in X, \quad v = T(v), \tag{16.2}$$

in a Banach space X, and some fixed-point theorem is applied as in Sections 13 or 15.

In the finite-dimensional case, Brouwer's Fixed-Point Theorem was the easiest choice. Here we use its generalization to Banach spaces, that is, Schauder's Fixed-Point Theorem, provided that some compactness properties are available, or Banach's Fixed-Point Theorem if we are ready to accept an additional contraction condition. The existence of a solution v^* of (16.2) in some suitable set $V \subset X$ then follows, provided that

$$T(V) \subset V. \tag{16.3}$$

Consequently, $u^* := \tilde{u} + v^*$ is a solution of (16.1) (which gives the desired existence result), and $u^* \in \tilde{u} + V$ is an enclosure for u^*.

So the crucial condition to be verified, for some suitable set V, is (16.3). In the corresponding finite-dimensional situation, for given V the left-hand side of (16.3) was evaluated more or less in closed form, since there the operator $T : \mathbb{R}^n \to \mathbb{R}^n$ was composed of computable or enclosable terms such as the given nonlinear function, its Jacobian, and so forth.

In the infinite-dimensional situation, the operator T will in principle be built in a similar way, with the Jacobian now replaced by the elliptic differential operator L obtained by linearization of the left-hand side of (16.1). Evaluating or enclosing L^{-1} (as is needed if T is of Newton type, as in 13.2) is, however, not directly possible in general, and it is unclear how an approximation $C \approx L^{-1}$ should be computed such that $I - CL$ or $I - LC$ has a norm less than 1 (as in 13.3).

Therefore – choosing a Newton-type operator T – a normwise bound for L^{-1}, and also for the other ingredients of T, will be used instead. Again there is an analogy to the finite-dimensional case. If the system dimension is too large and hence the effort for enclosing – or even approximating – an inverse matrix A^{-1} is too high, a lower bound for the smallest singular value of A is used as in Section 10.8.1, which obviously corresponds to a norm bound for A^{-1}. If a norm ball V (centred at the origin) is chosen as a candidate for (16.3), then (16.3) results in an inequality involving the radius of V and the norm bounds indicated above. Since these bounds are computable, either directly or by additional computer-assisted means (such as the bound for $\|L^{-1}\|$), the resulting sufficient inequality for (16.3) can be checked.

It is important to remark that this inequality will indeed be satisfied if the approximate solution $\tilde{u}$ has been computed with sufficient accuracy, and if $\|L^{-1}\|$ is not too large (*i.e.*, if the situation is 'sufficiently non-degenerate'). Both conditions also appear in the finite-dimensional case.

We remark that Nagatou, Nakao and Yamamoto (1999), Nakao (1993) and Nakao and Yamamoto (1995) have chosen an approach avoiding the direct computation of a bound for $\|L^{-1}\|$. They use a finite-dimensional

projection of L, which can be treated by the linear algebra verifying tools explained in the previous sections. However, the (infinite-dimensional) projection error also needs to be bounded in a suitable way, as in Nakao (1988) or Nakao *et al.* (2005), which is certainly possible for not too complicated domains Ω, but may be problematic, *e.g.*, for unbounded domains.[26]

16.1. Abstract formulation

In order to see the structural essence of the proposed enclosure methods for problem (16.1), and in particular its analogy to finite-dimensional problems, we first study (16.1) in the abstract form

$$\text{Find } u \in X \text{ satisfying } \mathcal{F}(u) = 0, \tag{16.4}$$

with $(X, \langle\cdot,\cdot\rangle_X)$ and $(Y, \langle\cdot,\cdot\rangle_Y)$ denoting two real Hilbert spaces, and $\mathcal{F} : X \to Y$ some Fréchet-differentiable mapping. Concrete choices of X and Y will be discussed in the next two subsections.

Let $\tilde{u} \in X$ denote some *approximate* solution to (16.4) (computed, *e.g.*, by numerical means), and denote by

$$L := \mathcal{F}'(\tilde{u}) : X \to Y, \quad x \mapsto L[x] \tag{16.5}$$

the Fréchet derivative of $\mathcal{F}$ at $\tilde{u}$, *i.e.*, $L \in \mathcal{B}(X, Y)$ (the Banach space of all bounded linear operators from X to Y) and

$$\lim_{\substack{h \in X\setminus\{0\} \\ h \to 0}} \frac{1}{\|h\|_X} \|\mathcal{F}(\tilde{u}+h) - \mathcal{F}(\tilde{u}) - L[h]\|_Y = 0.$$

Suppose that constants δ and K, and a non-decreasing function $g : [0,\infty) \to [0,\infty)$, are known such that

$$\|\mathcal{F}(\tilde{u})\|_Y \le \delta, \tag{16.6}$$

i.e., δ bounds the *defect* (residual) of the approximate solution $\tilde{u}$ to (16.4),

$$\|u\|_X \le K \, \|L[u]\|_Y \quad \text{for all} \quad u \in X, \tag{16.7}$$

i.e., K bounds the *inverse* of the linearization L,

$$\|\mathcal{F}'(\tilde{u}+u) - \mathcal{F}'(\tilde{u})\|_{\mathcal{B}(X,Y)} \le g(\|u\|_X) \quad \text{for all} \quad u \in X, \tag{16.8}$$

i.e., g majorizes the modulus of continuity of $\mathcal{F}'$ at $\tilde{u}$, and

$$g(t) \to 0 \quad \text{as} \quad t \to 0 \tag{16.9}$$

(which, in particular, requires $\mathcal{F}'$ to be continuous at $\tilde{u}$).

The concrete computation of such δ, K, and g is the main challenge in our approach, with particular emphasis on K. We will however not address these

[26] Part of the following is taken from Plum (2008).

questions on the abstract level, but postpone them to the more specific case of the boundary value problem (16.1). For now, assume that (16.6)–(16.9) hold true.

In order to obtain a suitable fixed-point formulation (16.3) for our problem (16.4), the operator L must be *onto* because $L^{-1} : Y \to X$ will be used. (Note that L is one-to-one by (16.7).) There are two alternative ways to do this, both suited to the later treatment of problem (16.1).

(1) *The compact case.* Suppose that $\mathcal{F}$ admits a splitting

$$\mathcal{F} = L_0 + \mathcal{G}, \tag{16.10}$$

with a *bijective* linear operator $L_0 \in \mathcal{B}(X, Y)$, and a *compact* and Fréchet-differentiable operator $\mathcal{G} : X \to Y$, with compact Fréchet derivative $\mathcal{G}'(\tilde{u})$. Then the Fredholm alternative holds for the equation $L[u] = r$ (where $r \in Y$), and since L is one-to-one by (16.7), it is therefore onto.

(2) *The dual and symmetric case.* Suppose that $Y = X'$, the (topological) *dual* of X, *i.e.*, the space of all bounded linear functionals $l : X \to \mathbb{R}$. $X'(= \mathcal{B}(X, \mathbb{R}))$ is a Banach space endowed with the usual operator sup-norm. Indeed, this norm is generated by the inner product (which therefore makes X' a Hilbert space)

$$\langle r, s\rangle_{X'} := \langle \Phi^{-1}[r], \Phi^{-1}[s]\rangle_X \quad (r, s \in X'), \tag{16.11}$$

where $\Phi : X \to X'$ is the canonical isometric isomorphism given by

$$(\Phi[u])[v] := \langle u, v\rangle_X \ (u, v \in X). \tag{16.12}$$

To ensure that $L : X \to Y = X'$ is onto, we make the additional assumption that $\Phi^{-1}L : X \to X$ is *symmetric* with respect to $\langle\cdot,\cdot\rangle_X$, which by (16.12) amounts to the relation

$$(L[u])[v] = (L[v])[u] \quad \text{for all} \ \ u, v \in X. \tag{16.13}$$

This implies the denseness of the range $(\Phi^{-1}L)(X) \subset X$: given any u in its orthogonal complement, we have, for all $v \in X$,

$$0 = \langle u, (\Phi^{-1}L)[v]\rangle_X = \langle (\Phi^{-1}L)[u], v\rangle_X,$$

and hence $(\Phi^{-1}L)[u] = 0$, which implies $L[u] = 0$ and thus $u = 0$ by (16.7).

Therefore, since Φ is isometric, the range $L(X) \subset X'$ is *dense*. To prove that L is onto, it remains to show that $L(X) \subset X'$ is *closed*. For this purpose, let $(L[u_n])_{n\in\mathbb{N}}$ denote some sequence in $L(X)$ converging to some $r \in X'$. Then (16.7) shows that $(u_n)_{n\in\mathbb{N}}$ is a Cauchy sequence in X. With $u \in X$ denoting its limit, the boundedness of L implies $L[u_n] \to L[u]$ $(n \to \infty)$. Thus, $r = L[u] \in L(X)$, proving closedness of $L(X)$.

We are now able to formulate and prove our main theorem, which is similar to the Newton–Kantorovich theorem.

Theorem 16.1. Let δ, K, g satisfy conditions (16.6)–(16.9). Suppose that some $\alpha > 0$ exists such that

$$\delta \leq \frac{\alpha}{K} - G(\alpha), \tag{16.14}$$

where $G(t) := \int_0^t g(s)\,\mathrm{d}s$. Moreover, suppose that:

(1) the *compact case* applies, or

(2) the *dual and symmetric case* applies, and we have the additional condition

$$Kg(\alpha) < 1. \tag{16.15}$$

Then, there exists a solution $u \in X$ of the equation $\mathcal{F}(u) = 0$ satisfying

$$\|u - \tilde{u}\|_X \leq \alpha. \tag{16.16}$$

Remark 1. Due to (16.9), $G(t) = \int_0^t g(s)\,\mathrm{d}s$ is *superlinearly* small as $t \to 0$. Therefore, the crucial condition (16.14) is indeed satisfied for some 'small' α if K is 'moderate' (*i.e.*, not too large) and δ is sufficiently small, which means, according to (16.6), that the approximate solution $\tilde{u}$ to problem (16.4) must be computed with *sufficient accuracy*, and (16.14) tells us *how* accurate the computation has to be.

Remark 2. To prove Theorem 16.1, the (abstract) *Green's operator* L^{-1} will be used to reformulate problem (16.4) as a fixed-point equation, and some fixed-point theorem will be applied. If the space X were finite-dimensional, *Brouwer's* Fixed-Point Theorem would be most suitable for this purpose. In the application to differential equation problems such as (16.1), however, X has to be infinite-dimensional, whence Brouwer's Theorem is not applicable. There are two choices.

(S) We use the generalization of Brouwer's theorem to infinite-dimensional spaces, *i.e.*, *Schauder's* Fixed-Point Theorem, which explicitly requires additional compactness properties (holding automatically in the finite-dimensional case). In our later application to (16.1), this compactness is given by compact embeddings of Sobolev function spaces, provided that the domain Ω is bounded (or at least has finite measure).

(B) We use *Banach's* Fixed-Point Theorem. No compactness is needed, but the additional contraction condition (16.15) is required. Due to (16.9), this condition is, however, not too critical if α (computed according to (16.14)) is 'small'. This option includes unbounded domains Ω.

Proof of Theorem 16.1. We rewrite problem (16.4) as

$$L[u-\tilde{u}] = -\mathcal{F}(\tilde{u}) - \big\{\mathcal{F}(u) - \mathcal{F}(\tilde{u}) - L[u-\tilde{u}]\big\},$$

which due to the bijectivity of L amounts to the equivalent fixed-point equation

$$v \in X, \quad v = -L^{-1}\big[\mathcal{F}(\tilde{u}) + \{\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}) - L[v]\}\big] =: T(v) \tag{16.17}$$

for the error $v = u - \tilde{u}$. We show the following properties of the fixed-point operator $T : X \to X$:

(i) $T(V) \subset V$ for the closed, bounded, non-empty, and convex norm ball

$$V := \{v \in X : \|v\|_X \le \alpha\},$$

(ii) T is continuous and compact (in case (1)) or contractive on V (in case (2)), respectively.

Then, Schauder's Fixed-Point Theorem (in case (1)) or Banach's Fixed-Point Theorem (in case (2)), respectively, gives a solution $v^* \in V$ of the fixed-point equation (16.17), whence by construction $u^* := \tilde{u} + v^*$ is a solution of $\mathcal{F}(u) = 0$ satisfying (16.16).

To prove (i) and (ii), we first note that for every differentiable function $f : [0,1] \to Y$, the real-valued function $\|f\|_Y$ is differentiable almost everywhere on $[0,1]$, and $(\mathrm{d}/\mathrm{d}t)\|f\|_Y \le \|f'\|_Y$ almost everywhere on $[0,1]$. Hence, for every $v, \tilde{v} \in X$,

$$\begin{aligned}
&\|\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}+\tilde{v}) - L[v-\tilde{v}]\|_Y \qquad (16.18)\\
&\quad = \int_0^1 \frac{\mathrm{d}}{\mathrm{d}t}\, \|\mathcal{F}(\tilde{u} + (1-t)\tilde{v} + tv) - \mathcal{F}(\tilde{u}+\tilde{v}) - tL[v-\tilde{v}]\|_Y \,\mathrm{d}t\\
&\quad \le \int_0^1 \|\{\mathcal{F}'(\tilde{u} + (1-t)\tilde{v} + tv) - L\}[v-\tilde{v}]\|_Y \,\mathrm{d}t\\
&\quad \le \int_0^1 \|\mathcal{F}'(\tilde{u} + (1-t)\tilde{v} + tv) - L\|_{\mathcal{B}(X,Y)} \,\mathrm{d}t \cdot \|v-\tilde{v}\|_X\\
&\quad \le \int_0^1 g(\|(1-t)\tilde{v} + tv\|_X) \,\mathrm{d}t \cdot \|v-\tilde{v}\|_X,
\end{aligned}$$

using (16.5) and (16.8) at the last step. Choosing $\tilde{v} = 0$ in (16.18) we obtain, for each $v \in X$,

$$\begin{aligned}
\|\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}) - L[v]\|_Y &\le \int_0^1 g(t\|v\|_X) \,\mathrm{d}t \;\cdot\; \|v\|_X \qquad (16.19)\\
&= \int_0^{\|v\|_X} g(s) \,\mathrm{d}s = G(\|v\|_X).
\end{aligned}$$

Furthermore, (16.18) and the fact that g is non-decreasing imply, for all $v, \tilde{v} \in V$,

$$\begin{aligned} \|\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}+\tilde{v}) - L[v-\tilde{v}]\|_Y \\ \leq \int_0^1 g((1-t)\|\tilde{v}\|_X + t\|v\|_X)\,\mathrm{d}t \cdot \|v-\tilde{v}\|_X \\ \leq g(\alpha)\|v-\tilde{v}\|_X. \end{aligned} \tag{16.20}$$

To prove (i), let $v \in V$, *i.e.*, $\|v\|_X \leq \alpha$. Now (16.17), (16.7), (16.6), (16.19), and (16.14) imply

$$\begin{aligned} \|T(v)\|_X &\leq K\|\mathcal{F}(\tilde{u}) + \{\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}) - L[v]\}\|_Y \\ &\leq K(\delta + G(\|v\|_X)) \leq K(\delta + G(\alpha)) \leq \alpha, \end{aligned}$$

which gives $T(v) \in V$. Thus, $T(V) \subset V$.

To prove (ii), suppose first that the *compact case* applies. So (16.10), which in particular gives $L = L_0 + \mathcal{G}'(\tilde{u})$, and (16.17) imply

$$T(v) = -L^{-1}\big[\mathcal{F}(\tilde{u}) + \{\mathcal{G}(\tilde{u}+v) - \mathcal{G}(\tilde{u}) - \mathcal{G}'(\tilde{u})[v]\}\big] \quad \text{for all} \quad v \in X,$$

whence continuity and compactness of T follow from continuity and compactness of $\mathcal{G}$ and $\mathcal{G}'(\tilde{u})$, and the boundedness of L^{-1} ensured by (16.7).

If the *dual and symmetric case* applies, (16.17), (16.7), and (16.20) imply, for $v, \tilde{v} \in V$,

$$\begin{aligned} \|T(v) - T(\tilde{v})\|_X &= \|L^{-1}\{\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}+\tilde{v}) - L[v-\tilde{v}]\}\|_X \\ &\leq K\|\mathcal{F}(\tilde{u}+v) - \mathcal{F}(\tilde{u}+\tilde{v}) - L[v-\tilde{v}]\|_Y \\ &\leq Kg(\alpha)\|v-\tilde{v}\|_X, \end{aligned}$$

whence (16.15) shows that T is contractive on V. This completes the proof of Theorem 16.1. □

In the following two subsections, the abstract approach developed in this section will be applied to the elliptic boundary value problem (16.1). This can be done in (essentially two) different ways, *i.e.*, by different choices of the Hilbert spaces X and Y, resulting in different general assumptions (*e.g.*, smoothness conditions) to be made for the 'data' of the problem and the numerical approximation $\tilde{u}$, and different conditions (16.6)–(16.8), (16.14), (16.15), as well as different 'results', *i.e.*, existence statements and error bounds (16.16).

At this point, we briefly report on some other applications of our abstract setting which cannot be discussed in more detail in this article.

For *parameter-dependent problems* (where $\mathcal{F}$ in (16.4), or f in (16.1), depends on an additional parameter λ), one is often interested in *branches* $(u_\lambda)_{\lambda\in I}$ of solutions. By additional perturbation techniques, Plum (1995) generalized the presented method to a verification method for such solution branches, as long as the parameter interval I defining the branch is compact.

Such branches may, however, contain *turning points* (where a branch 'returns' at some value λ^*) or *bifurcation points* (where several – usually two – branches cross each other). Near such points, the operator L defined in (16.5) is 'almost' singular, *i.e.*, (16.7) holds only with a very large K, or not at all, and our approach breaks down. However, there are means to overcome these problems.

In the case of (simple) turning points, Plum (1994) used the well-known method of augmenting the given equation by a *bordering* equation (as in Section 13.2): the 'new' operator $\mathcal{F}$ in (16.4) contains the 'old' one *and* the bordering functional, and the 'new' operator L is regular near the turning point if the bordering equation has been chosen appropriately.

In the case of (simple) symmetry-breaking bifurcations, Plum (1996) includes in a first step the symmetry in the spaces X and Y, which excludes the symmetry-breaking branch and regularizes the problem, whence an existence and enclosure result for the symmetric branch can be obtained. In a second step, the symmetric branch is excluded by some transformation (similar to the Lyapunov–Schmidt reduction), and defining a corresponding new operator $\mathcal{F}$ an existence and enclosure result can also be obtained for the symmetry-breaking branch.

Lahmann and Plum (2004) treated *non-self-adjoint eigenvalue problems*, again using bordering equation techniques normalizing the unknown eigenfunction. So $\mathcal{F}$ now acts on pairs (u, λ), and is defined via the eigenvalue equation and the (scalar) normalizing equation. In this way it was possible to give the first known instability proof of the Orr–Sommerfeld equation with Blasius profile, which is a fourth-order ODE eigenvalue problem on $[0, \infty)$.

Also (other) *higher-order* problems are covered by our abstract setting. Breuer, Horák, McKenna and Plum (2006) could prove the existence of 36 travelling wave solutions of a fourth-order nonlinear beam equation on the real line. Biharmonic problems (with $\Delta\Delta u$ as leading term) are currently being investigated by B. Fazekas; see also Fazekas, Plum and Wieners (2005).

16.2. Strong solutions

We first study the elliptic boundary value problem (16.1) under the additional assumptions that f and $\partial f/\partial y$ are continuous on $\bar{\Omega} \times \mathbb{R}$, that the domain $\Omega \subset \mathbb{R}^n$ (with $n \leq 3$) is *bounded* with Lipschitz boundary, and H^2-*regular* (*i.e.*, , for each $r \in L^2(\Omega)$, and that the Poisson problem $-\Delta u = r$ in Ω, $u = 0$ on $\partial\Omega$ has a unique solution $u \in H^2(\Omega) \cap H_0^1(\Omega)$).

Here and in the following, $L^2(\Omega)$ denotes the Hilbert space of all (equivalence classes of) square-integrable Lebesgue-measurable real-valued functions on Ω, endowed with the inner product

$$\langle u, v\rangle_{L^2} := \int_\Omega uv \,\mathrm{d}x,$$

and $H^k(\Omega)$ is the Sobolev space of all functions $u \in L^2(\Omega)$ with weak derivatives up to order k in $L^2(\Omega)$. $H^k(\Omega)$ is a Hilbert space with the inner product

$$\langle u, v\rangle_{H^k} := \sum_{\substack{\alpha\in\mathbb{N}_0^n \\ |\alpha|\le k}} \langle D^\alpha u, D^\alpha v\rangle_{L^2},$$

where $|\alpha| := \alpha_1 + \cdots + \alpha_n$, which can also be characterized as the completion of $C^\infty(\bar\Omega)$ with respect to $\langle\cdot,\cdot\rangle_{H^k}$. Replacement of $C^\infty(\bar\Omega)$ by $C_0^\infty(\Omega)$ (with the subscript 0 indicating compact support in Ω), yields, by completion, the Sobolev space $H_0^k(\Omega)$, which incorporates the vanishing of all derivatives up to order $k-1$ on $\partial\Omega$ in a weak sense.

Note that *piecewise* C^k-smooth functions u (*e.g.*, form functions of finite element methods) belong to $H^k(\Omega)$ if and only if they are (globally) in $C^{k-1}(\overline{\Omega})$.

For example, the assumption that Ω is H^2-regular is satisfied for C^2- (or $C^{1,1}$-)*smoothly* bounded domains (Gilbarg and Trudinger 1983), and also for *convex* polygonal domains $\Omega \subset \mathbb{R}^2$ (Grisvard 1985); it is *not* satisfied, *e.g.*, for domains with re-entrant corners, such as the L-shaped domain $(-1,1)^2 \setminus [0,1)^2$.

Under the assumptions made, we can choose the spaces

$$X := H^2(\Omega) \cap H_0^1(\Omega), \quad Y := L^2(\Omega), \tag{16.21}$$

and the operators

$$\mathcal{F} := L_0 + \mathcal{G}, \quad L_0[u] := -\Delta u, \quad \mathcal{G}(u) := f(\cdot, u), \tag{16.22}$$

whence indeed our problem (16.1) amounts to the abstract problem (16.4). Moreover, $L_0 : X \to Y$ is bijective by the assumed unique solvability of the Poisson problem, and clearly bounded, that is, in $\mathcal{B}(X,Y)$. Finally, $\mathcal{G} : X \to Y$ is Fréchet-differentiable with derivative

$$\mathcal{G}'(u)[v] = \frac{\partial f}{\partial y}(\cdot, u)v, \tag{16.23}$$

which follows from the fact that $\mathcal{G}$ has this derivative as an operator from $C(\bar\Omega)$ (endowed with the maximum norm $\|\cdot\|_\infty$) into itself, and that the embeddings $H^2(\Omega) \hookrightarrow C(\bar\Omega)$ and $C(\bar\Omega) \hookrightarrow L^2(\Omega)$ are bounded. In fact, $H^2(\Omega) \hookrightarrow C(\bar\Omega)$ is even a *compact* embedding by the famous Sobolev–Kondrachov–Rellich Embedding Theorem (Adams 1975) (and since $n \le 3$),

which shows that $\mathcal{G}$ and $\mathcal{G}'(u)$ (for any $u \in X$) are compact. Thus, the *compact case* (see (16.10)) applies.

For the application of Theorem 16.1, we are therefore left to comment on the computation of constants δ, K and a function g satisfying (16.6)–(16.9) (in the setting (16.21), (16.22)). But first, some comments on the computation of the approximate solution $\tilde{u}$ are necessary.

16.2.1. Computation of $\tilde{u}$

Since $\tilde{u}$ is required to be in $X = H^2(\Omega) \cap H_0^1(\Omega)$, it has to satisfy the boundary condition *exactly* (in the sense of being in $H_0^1(\Omega)$), and it needs to have weak derivatives in $L^2(\Omega)$ up to order 2. If finite elements are used, this implies the need for C^1-elements (*i.e.*, *globally* C^1-smooth finite element basis functions), which is a drawback at least on a technical level. (In the alternative approach proposed in the next subsection, this drawback is avoided.)

If $\Omega = (0, a) \times (0, b)$ is a *rectangle*, there are, however, many alternatives to finite elements, for example polynomial or trigonometric polynomial basis functions. In the latter case, $\tilde{u}$ is given in the form

$$\tilde{u}(x_1, x_2) = \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_{ij} \sin\left(i\pi \frac{x_1}{a}\right) \sin\left(j\pi \frac{x_2}{b}\right), \tag{16.24}$$

with coefficients α_{ij} to be determined by some numerical procedure. Such a procedure usually consists of a Newton iteration, together with a Ritz–Galerkin or a collocation method, and a linear algebraic system solver, which possibly incorporates multigrid methods. To *start* the Newton iteration, a rough initial approximation is needed, which may be obtained by path-following methods, or by use of the numerical mountain pass algorithm proposed by Choi and McKenna (1993).

An important remark is that, no matter how $\tilde{u}$ is given or which numerical method is used, there is *no* need for any *rigorous* computation at this stage, and therefore the whole variety of numerical methods applies.

16.2.2. Defect bound δ

Computing some δ satisfying (16.6) means, due to (16.21) and (16.22), computing an upper bound for (the square root of)

$$\int_\Omega \left[-\Delta\tilde{u} + f(\cdot, \tilde{u})\right]^2 \mathrm{d}x, \tag{16.25}$$

which should be 'small' if $\tilde{u}$ is a 'good' approximate solution. In some cases this integral can be calculated in closed form, by hand or by computer algebra routines, for example if f is polynomial and $\tilde{u}$ is piecewise polynomial (as it is if finite element methods have been used to compute it), or if $f(x, \cdot)$

is polynomial and both $f(\cdot, y)$ and $\tilde{u}$ (as in (16.24)) are trigonometric polynomials. The resulting formulas have to be evaluated *rigorously*, to obtain a true upper bound for the integral in (16.25).

If closed-form integration is impossible, a *quadrature formula* should be applied, possibly piecewise, to the integral in (16.25), using, for example, the methods described in Section 12. To obtain a true upper bound for the integral, in addition a *remainder term bound* for the quadrature formula is needed, which usually requires rough $\|\cdot\|_\infty$-bounds for some higher derivatives of the integrand. These rough bounds can be obtained, for example, by subdividing Ω into (many) small boxes, and performing interval evaluations of the necessary higher derivatives over each of these boxes (which gives true supersets of the function value ranges over each of the boxes, and thus, by union, over Ω).

16.2.3. Bound K for L^{-1}

The next task is the computation of a constant K satisfying (16.7), which, due to (16.21)–(16.23), means

$$\|u\|_{H^2} \le K\|L[u]\|_{L^2} \quad \text{for all} \quad u \in H^2(\Omega) \cap H_0^1(\Omega), \tag{16.26}$$

where $L : H^2(\Omega) \cap H_0^1(\Omega) \to L^2(\Omega)$ is given by

$$L[u] = -\Delta u + cu, \quad c(x) := \frac{\partial f}{\partial y}(x, \tilde{u}(x)) \ (x \in \bar{\Omega}). \tag{16.27}$$

The first (and most crucial) step towards (16.26) is the computation of a constant K_0 such that

$$\|u\|_{L^2} \le K_0\|L[u]\|_{L^2} \quad \text{for all} \quad u \in H^2(\Omega) \cap H_0^1(\Omega). \tag{16.28}$$

Choosing some constant lower bound $\underline{c}$ for c on $\bar{\Omega}$, and using the compact embedding $H^2(\Omega) \hookrightarrow L^2(\Omega)$, we find by standard means that $(L - \underline{c})^{-1} : L^2(\Omega) \to L^2(\Omega)$ is compact, symmetric, and positive definite, and hence has a $\langle\cdot,\cdot\rangle_{L^2}$-orthonormal and complete system $(\varphi_k)_{k\in\mathbb{N}}$ of eigenfunctions $\varphi_k \in H^2(\Omega) \cap H_0^1(\Omega)$, with associated sequence $(\mu_k)_{k\in\mathbb{N}}$ of (positive) eigenvalues converging monotonically to 0. Consequently, $L[\varphi_k] = \lambda_k\varphi_k$ for $k \in \mathbb{N}$, with $\lambda_k = \mu_k^{-1} + \underline{c}$ converging monotonically to $+\infty$. Series expansion yields, for every $u \in H^2(\Omega) \cap H_0^1(\Omega)$,

$$\begin{aligned}\|L[u]\|_{L^2}^2 &= \sum_{k=1}^{\infty} \langle L[u], \varphi_k\rangle_{L^2}^2 = \sum_{k=1}^{\infty} \langle u, L[\varphi_k]\rangle_{L^2}^2 = \sum_{k=1}^{\infty} \lambda_k^2 \langle u, \varphi_k\rangle_{L^2}^2 \\ &\ge \Big(\min_{j\in\mathbb{N}} \lambda_j^2\Big) \sum_{k=1}^{\infty} \langle u, \varphi_k\rangle_{L^2}^2 = \Big(\min_{j\in\mathbb{N}} \lambda_j^2\Big)\|u\|_{L^2}^2,\end{aligned}$$

which shows that (16.28) holds if (and only if) $\lambda_j \neq 0$ for all $j \in \mathbb{N}$, and

$$K_0 \geq \left(\min_{j \in \mathbb{N}} |\lambda_j| \right)^{-1}. \tag{16.29}$$

Thus, bounds for the eigenvalue(s) of L neighbouring 0 are needed to compute K_0. Such *eigenvalue bounds* can be obtained by computer-assisted means of their own. For example, *upper* bounds to $\lambda_1, \ldots, \lambda_N$ (with $N \in \mathbb{N}$ given) are easily and efficiently computed by the *Rayleigh–Ritz* method (Rektorys 1980), as follows. Let $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_N \in H^2(\Omega) \cap H_0^1(\Omega)$ denote linearly independent trial functions, for example approximate eigenfunctions obtained by numerical means, and form the matrices

$$A_1 := (\langle L[\tilde{\varphi}_i], \tilde{\varphi}_j \rangle_{L^2})_{i,j=1,\ldots,N}, \quad A_0 := (\langle \tilde{\varphi}_i, \tilde{\varphi}_j \rangle_{L^2})_{i,j=1,\ldots,N}.$$

Then, with $\Lambda_1 \leq \cdots \leq \Lambda_N$ denoting the eigenvalues of the matrix eigenvalue problem

$$A_1 x = \Lambda A_0 x,$$

which can be enclosed by the methods given in Section 13.4, the Rayleigh–Ritz method gives

$$\lambda_i \leq \Lambda_i \quad \text{for} \quad i = 1, \ldots, N.$$

However, to compute K_0 via (16.29), *lower* eigenvalue bounds are also needed, which constitute a more complicated task than upper bounds. The most accurate method for this purpose was proposed by Lehmann (1963), and its range of applicability improved by Goerisch (Behnke and Goerisch 1994). Its numerical core is again (as in the Rayleigh–Ritz method) a matrix eigenvalue problem, but the accompanying analysis is more involved.

In particular, in order to compute lower bounds to the first N eigenvalues, a *rough* lower bound to the $(N+1)$st eigenvalue must already be known. This *a priori* information can usually be obtained via a *homotopy method* connecting a simple 'base problem' with known eigenvalues to the given eigenvalue problem, such that all eigenvalues increase (index-wise) along the homotopy; details are given by Plum (1997) and Breuer *et al.* (2006).

Finding a base problem for the eigenvalue problem $L[u] = \lambda u$, and a suitable homotopy connecting them, is often possible along the following lines. If Ω is a bounded rectangle (whence the eigenvalues of $-\Delta$ on $H_0^1(\Omega)$ are known), choose a constant lower bound $\underline{c}$ for c on Ω, and the coefficient homotopy

$$c_s(x) := (1-s)\underline{c} + sc(x), \quad (x \in \Omega, 0 \leq s \leq 1).$$

Then, the family of eigenvalue problems

$$-\Delta u + c_s u = \lambda^{(s)} u \ \text{ in } \ \Omega, \quad u = 0 \quad \text{on } \ \partial\Omega$$

connects the explicitly solvable constant-coefficient base problem ($s = 0$) to the problem $L[u] = \lambda u$ ($s = 1$), and the eigenvalues increase in s, since the Rayleigh quotient does, by Poincaré's min-max principle.

If Ω is not a rectangle (or a ball), we can first choose a rectangle Ω_0 containing Ω, and a domain deformation homotopy between Ω_0 and Ω, to enclose the (first M) eigenvalues of $-\Delta$ on $H_0^1(\Omega)$: see, *e.g.*, Plum and Wieners (2002). Then, the above coefficient homotopy is applied in a second step.

Once a constant K_0 satisfying (16.28) is known, the desired constant K (satisfying (16.26)) can relatively easily be calculated by explicit *a priori estimates*. With $\underline{c}$ again denoting a constant lower bound for c, we obtain by partial integration, for each $u \in H^2(\Omega) \cap H_0^1(\Omega)$,

$$\|u\|_{L^2}\|L[u]\|_{L^2} \geq \langle u, L[u]\rangle_{L^2} = \int_\Omega (|\nabla u|^2 + cu^2)\,\mathrm{d}x \geq \|\nabla u\|_{L^2}^2 + \underline{c}\|u\|_{L^2}^2,$$

which implies, together with (16.28), that

$$\|\nabla u\|_{L^2} \leq K_1\|L[u]\|_{L^2}, \quad \text{where } K_1 := \begin{cases} \sqrt{K_0(1-\underline{c}K_0)} & \text{if } \underline{c}K_0 \leq \frac{1}{2}, \\ \frac{1}{2\sqrt{\underline{c}}} & \text{otherwise.} \end{cases} \tag{16.30}$$

To complete the H^2-bound required in (16.26), the L^2-norm of the (Frobenius matrix norm of the) Hessian matrix u_{xx} of $u \in H^2(\Omega) \cap H_0^1(\Omega)$ is to be estimated. If Ω is convex (as we shall assume now), then

$$\|u_{xx}\|_{L^2} \leq \|\Delta u\|_{L^2} \quad \text{for all } \ u \in H^2(\Omega) \cap H_0^1(\Omega); \tag{16.31}$$

see, *e.g.*, Grisvard (1985) and Ladyzhenskaya and Uralt́seva (1968). For the non-convex case, see Grisvard (1985) and Plum (1992). Now, with $\bar{c}$ denoting an additional upper bound for c, we choose

$$\mu := \max\left\{0, \frac{1}{2}(\underline{c} + \bar{c})\right\},$$

and calculate

$$\|\Delta u\|_{L^2} \leq \|-\Delta u + \mu u\|_{L^2} \leq \|L[u]\|_{L^2} + \|\mu - c\|_\infty \|u\|_{L^2}.$$

Using that $\|\mu - c\|_\infty = \max\{-\underline{c}, \frac{1}{2}(\bar{c} - \underline{c})\}$, and combining with (16.28) results in

$$\|\Delta u\|_{L^2} \leq K_2\|L[u]\|_{L^2}, \quad \text{where } K_2 := 1 + K_0 \max\left\{-\underline{c}, \frac{1}{2}(\bar{c} - \underline{c})\right\}. \tag{16.32}$$

Now, (16.28), (16.30) and (16.32) give (16.26) as follows. For quantitative purposes, we use the modified inner product

$$\langle u, v\rangle_{H^2} := \gamma_0\langle u, v\rangle_{L^2} + \gamma_1\langle \nabla u, \nabla v\rangle_{L^2} + \gamma_2\langle \Delta u, \Delta v\rangle_{L^2} \tag{16.33}$$

(with positive weights $\gamma_0, \gamma_1, \gamma_2$) on X, which, due to (16.31), and to the obvious reverse inequality $\|\Delta u\|_{L^2} \le \sqrt{n}\|u_{xx}\|_{L^2}$, is equivalent to the canonical one. Then, (16.26) obviously holds for

$$K := \sqrt{\gamma_0 K_0^2 + \gamma_1 K_1^2 + \gamma_2 K_2^2}, \tag{16.34}$$

with K_0, K_1, K_2 from (16.28), (16.30) and (16.32).

16.2.4. Local Lipschitz bound g for $\mathcal{F}'$
By (16.21), (16.22) and (16.23), condition (16.8) reads

$$\left\| \left[\frac{\partial f}{\partial y}(\cdot, \tilde{u} + u) - \frac{\partial f}{\partial y}(\cdot, \tilde{u}) \right] v \right\|_{L^2} \le g(\|u\|_{H^2}) \|v\|_{H^2} \tag{16.35}$$
$$\text{for all } \; u, v \in H^2(\Omega) \cap H_0^1(\Omega).$$

We start with a monotonically non-decreasing function $\tilde{g} : [0, \infty) \to [0, \infty)$ satisfying

$$\left| \frac{\partial f}{\partial y}(x, \tilde{u}(x) + y) - \frac{\partial f}{\partial y}(x, \tilde{u}(x)) \right| \le \tilde{g}(|y|) \quad \text{for all } \; x \in \Omega, \; y \in \mathbb{R}, \tag{16.36}$$

and $\tilde{g}(t) \to 0$ as $t \to 0+$. In practice, such a function $\tilde{g}$ can usually be calculated by hand if a bound for $\|\tilde{u}\|_\infty$ is available, which in turn can be computed by interval evaluations of $\tilde{u}$ over small boxes (as described at the end of Section 16.2.2). Using $\tilde{g}$, the left-hand side of (16.35) can be bounded by

$$\tilde{g}(\|u\|_\infty) \|v\|_{L^2}, \tag{16.37}$$

leaving us to estimate both the norms $\|\cdot\|_{L^2}$ and $\|\cdot\|_\infty$ by $\|\cdot\|_{H^2}$. With ρ^* denoting the smallest eigenvalue of

$$-\Delta u = \rho u, \quad u \in H^2(\Omega) \cap H_0^1(\Omega),$$

we obtain by eigenfunction expansion that

$$\|\nabla u\|_{L^2}^2 = \langle u, -\Delta u \rangle_{L^2} \ge \rho^* \|u\|_{L^2}^2, \quad \|\Delta u\|_{L^2}^2 \ge (\rho^*)^2 \|u\|_{L^2}^2,$$

and thus, by (16.33),

$$\|u\|_{L^2} \le [\gamma_0 + \gamma_1 \rho^* + \gamma_2 (\rho^*)^2]^{-\frac{1}{2}} \|u\|_{H^2} \quad \text{for all } \; u \in H^2(\Omega) \cap H_0^1(\Omega). \tag{16.38}$$

Furthermore, in Plum (1992, Corollary 1) constants C_0, C_1, C_2 are calculated which depend on Ω in a rather simple way, allowing explicit computation, such that

$$\|u\|_\infty \le C_0 \|u\|_{L^2} + C_1 \|\nabla u\|_{L^2} + C_2 \|u_{xx}\|_{L^2} \quad \text{for all } \; u \in H^2(\Omega) \cap H_0^1(\Omega),$$

whence by (16.31) and (16.33) we obtain

$$\|u\|_\infty \le \left[\gamma_0^{-1}C_0^2 + \gamma_1^{-1}C_1^2 + \gamma_2^{-1}C_2^2\right]^{\frac{1}{2}} \|u\|_{H^2} \quad \text{for all} \quad u \in H^2(\Omega) \cap H_0^1(\Omega). \tag{16.39}$$

Using (16.38) and (16.39) in (16.37), we find that (16.35) (and (16.9)) hold for

$$g(t) := [\gamma_0 + \gamma_1\rho^* + \gamma_2(\rho^*)^2]^{-\frac{1}{2}} \tilde{g}\big(\left[\gamma_0^{-1}C_0^2 + \gamma_1^{-1}C_1^2 + \gamma_2^{-1}C_2^2\right]^{\frac{1}{2}} t\big). \tag{16.40}$$

16.2.5. A numerical example

Consider the problem

$$\Delta u + u^2 = s \cdot \sin(\pi x_1)\sin(\pi x_2) \ \ (x = (x_1, x_2) \in \Omega := (0,1)^2), \ \ u = 0 \ \text{ on } \ \partial\Omega. \tag{16.41}$$

The results reported here were established in Breuer, McKenna and Plum (2003).

Since the 1980s it had been conjectured in the PDE community that problem (16.41) has at least four solutions for sufficiently large $s > 0$.

For $s = 800$, indeed four essentially different approximate solutions could be computed by the numerical mountain pass algorithm developed by Choi and McKenna (1993), where 'essentially different' means that none of them is an elementary symmetry transform of another one. Using finite Fourier series of the form (16.24), and a Newton iteration in combination with a collocation method, the accuracy of the mountain pass solutions was improved, resulting in highly accurate approximations $\tilde{u}_1, \ldots, \tilde{u}_4$ of the form (16.24).

Then the described verification method was applied to each of these four approximations, and the corresponding four inequalities (16.14) were successfully verified, with four error bounds $\alpha_1, \ldots, \alpha_4$. Therefore, Theorem 16.1 guarantees the existence of four solutions

$$u_1, \ldots, u_4 \in H^2(\Omega) \cap H_0^1(\Omega)$$

of problem (16.41) such that

$$\|u_i - \tilde{u}_i\|_{H^2} \le \alpha_i \quad \text{for} \quad i = 1, \ldots, 4.$$

Using the embedding inequality (16.39), in addition

$$\|u_i - \tilde{u}_i\|_\infty \le \beta_i \quad \text{for} \quad i = 1, \ldots, 4 \tag{16.42}$$

for $\beta_i := [\gamma_0^{-1}C_0^2 + \gamma_1^{-1}C_1^2 + \gamma_2^{-1}C_2^2]^{\frac{1}{2}}\alpha_i$. Finally, it is easy to check on the basis of the numerical data that

$$\|S\tilde{u}_i - \tilde{u}_j\|_\infty > \beta_i + \beta_j \quad \text{for} \quad i, j = 1, \ldots, 4, \quad i \ne j$$

for each elementary (rotation or reflection) symmetry transformation S of the square Ω, so that $u_1, \ldots, u_4$ are indeed *essentially different*.

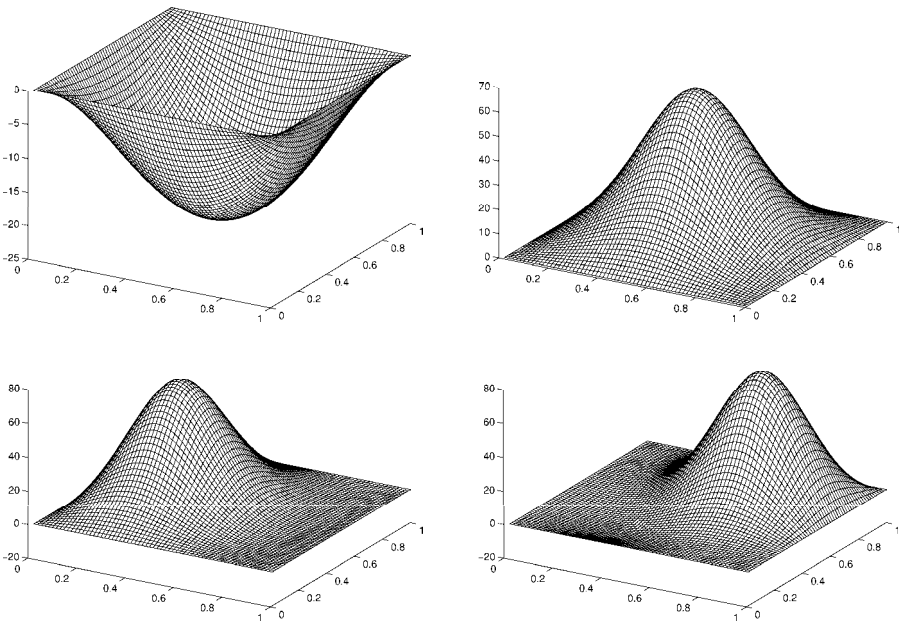

Figure 16.1. Four solutions to problem (16.41), $s = 800$.

Figure 16.1 shows plots of $\tilde{u}_1, \ldots, \tilde{u}_4$ (one might also say $u_1, \ldots, u_4$, since the error bounds β_i are much smaller than the 'optical accuracy' of the figure). The first two solutions are fully symmetric (with respect to reflection at the axes $x_1 = \frac{1}{2}, x_2 = \frac{1}{2}, x_1 = x_2, x_1 = 1 - x_2$), while the third is symmetric only with respect to $x_2 = \frac{1}{2}$, and the fourth only with respect to $x_1 = x_2$. Table 16.1 shows the defect bounds δ (see (16.6), (16.25)), the constants K satisfying (16.7) (or (16.26)), and the $\|\cdot\|_\infty$-error bounds β (see (16.42)) for the four solutions.

We wish to remark that, two years after publication of this result, Dancer and Yan (2005) gave a more general analytical proof (which we believe was stimulated by Breuer *et al.* (2003)); they even proved that the number of solutions of problem (16.41) becomes unbounded as $s \to \infty$.

16.3. Weak solutions

We will now investigate problem (16.1) under weaker assumptions on the domain $\Omega \subset \mathbb{R}^n$ and on the numerical approximation method, but stronger assumptions on the nonlinearity f, compared with the 'strong solutions' approach described in the previous subsection. Ω is now allowed to be any (bounded or unbounded) domain with Lipschitz boundary. We choose the spaces

$$X := H_0^1(\Omega), \quad Y := H^{-1}(\Omega) \tag{16.43}$$

Table 16.1. Enclosure results for problem (16.41).

Approximate solution	Defect bound δ	K (see (16.26))	Error bound β
$\tilde{u}_1$	0.0023	0.2531	$5.8222 \cdot 10^{-4}$
$\tilde{u}_2$	0.0041	4.9267	0.0228
$\tilde{u}_3$	0.0059	2.8847	0.0180
$\tilde{u}_4$	0.0151	3.1436	0.0581

for our abstract setting, where $H^{-1}(\Omega) := (H_0^1(\Omega))'$ denotes the topological dual space of $H_0^1(\Omega)$, *i.e.*, the space of all bounded linear functionals on $H_0^1(\Omega)$, where $H_0^1(\Omega)$ is endowed with the inner product

$$\langle u, v\rangle_{H_0^1} := \langle \nabla u, \nabla v\rangle_{L^2} + \sigma \langle u, v\rangle_{L^2} \tag{16.44}$$

(with some parameter $\sigma > 0$ to be chosen later), and $H^{-1}(\Omega)$ with the 'dual' inner product given by (16.11), with Φ from (16.12).

To interpret our problem (16.1) in these spaces, we first need to define Δu (for $u \in H_0^1(\Omega)$), or more generally, $\operatorname{div} \rho$ (for $\rho \in L^2(\Omega)^n$), as an element of $H^{-1}(\Omega)$. This definition simply imitates partial integration: the functional $\operatorname{div} \rho : H_0^1(\Omega) \to \mathbb{R}$ is given by

$$(\operatorname{div} \rho)[\varphi] := -\int_\Omega \rho \cdot \nabla \varphi \ \mathrm{d}x \quad \text{for all} \ \ \varphi \in H_0^1(\Omega), \tag{16.45}$$

implying in particular that

$$|(\operatorname{div} \rho)[\varphi]| \le \|\rho\|_{L^2} \|\nabla \varphi\|_{L^2} \le \|\rho\|_{L^2} \|\varphi\|_{H_0^1},$$

whence $\operatorname{div} \rho$ is indeed a *bounded* linear functional, and

$$\|\operatorname{div} \rho\|_{H^{-1}} \le \|\rho\|_{L^2}. \tag{16.46}$$

Using this definition of $\Delta u (= \operatorname{div}(\nabla u))$, it is easy to check that the canonical isometric isomorphism $\Phi : H_0^1(\Omega) \to H^{-1}(\Omega)$ defined in (16.12) is now given by (note that (16.44))

$$\Phi[u] = -\Delta u + \sigma u \quad (u \in H_0^1(\Omega)), \tag{16.47}$$

where $\sigma u \in H_0^1(\Omega)$ is interpreted as an element of $H^{-1}(\Omega)$, as explained in the following.

Next, we give a meaning to a *function* being an element of $H^{-1}(\Omega)$, in order to define $f(\cdot, u)$ in (16.1) (and σu in (16.47)) in $H^{-1}(\Omega)$. For this

purpose, let $\mathcal{L}$ denote the linear space consisting of all (equivalence classes of) Lebesgue-measurable functions $w : \Omega \to \mathbb{R}$ such that

$$\sup\left\{ \frac{1}{\|\varphi\|_{H_0^1}} \int_\Omega |w\varphi| \, \mathrm{d}x : \varphi \in H_0^1(\Omega) \setminus \{0\} \right\} < \infty. \tag{16.48}$$

For each $w \in \mathcal{L}$, we can define an associated linear functional $\ell_w : H_0^1(\Omega) \to \mathbb{R}$ by

$$\ell_w[\varphi] := \int_\Omega w\varphi \, \mathrm{d}x \quad \text{for all} \quad \varphi \in H_0^1(\Omega).$$

ℓ_w is bounded due to (16.48) and hence in $H^{-1}(\Omega)$. Identifying $w \in \mathcal{L}$ with its associated functional $\ell_w \in H^{-1}(\Omega)$, it follows that

$$\mathcal{L} \subset H^{-1}(\Omega), \tag{16.49}$$

and $\|w\|_{H^{-1}}$ is less than or equal to the left-hand side of (16.48), for every $w \in \mathcal{L}$.

To get a better impression of the functions contained in $\mathcal{L}$, recall that Sobolev's Embedding Theorem (Adams 1975, Theorem 5.4) gives $H_0^1(\Omega) \subset L^p(\Omega)$, with bounded embedding $H_0^1(\Omega) \hookrightarrow L^p(\Omega)$ (*i.e.*, there exists some constant $C_p > 0$ such that $\|u\|_{L^p} \le C_p \|u\|_{H_0^1}$ for all $u \in H_0^1(\Omega)$), for each

$$p \in [2, \infty) \quad \text{if} \quad n = 2 \quad \text{and} \quad p \in \left[2, \frac{2n}{n-2}\right] \quad \text{if} \quad n \ge 3. \tag{16.50}$$

Here, $L^p(\Omega)$ denotes the Banach space of (equivalence classes of) Lebesgue-measurable functions $u : \Omega \to \mathbb{R}$ with finite norm

$$\|u\|_{L^p} := \left[\int_\Omega |u|^p \, \mathrm{d}x \right]^{\frac{1}{p}}. \tag{16.51}$$

With p in the range (16.50), and p' denoting its dual number (*i.e.*, $p^{-1} + (p')^{-1} = 1$), Hölder's inequality, combined with the above embedding, yields that, for all $w \in L^{p'}(\Omega)$,

$$\int_\Omega |w\varphi| \, \mathrm{d}x \le \|w\|_{L^{p'}} \|\varphi\|_{L^p} \le C_p \|w\|_{L^{p'}} \|\varphi\|_{H_0^1},$$

implying $w \in \mathcal{L}$, and $\|w\|_{H^{-1}} \le C_p \|w\|_{L^{p'}}$. Consequently,

$$L^{p'}(\Omega) \subset \mathcal{L}, \tag{16.52}$$

and (note that (16.49)) the embedding $L^{p'}(\Omega) \hookrightarrow H^{-1}(\Omega)$ is bounded, with the same embedding constant C_p as in the 'dual' embedding $H_0^1(\Omega) \hookrightarrow L^p(\Omega)$. Note that the range (16.50) for p amounts to the range

$$p' \in (1, 2] \quad \text{if} \quad n = 2 \quad \text{and} \quad p' \in \left[\frac{2n}{n+2}, 2\right] \quad \text{if} \quad n \ge 3 \tag{16.53}$$

for the dual number p'. By (16.52), the linear span of the union of all $L^{p'}(\Omega)$, taken over p' in the range (16.53), is a subspace of $\mathcal{L}$, and this subspace is in fact all of $\mathcal{L}$, which is needed (and accessed) in practical applications.

Coming back to our problem (16.1), we now simply require that

$$f(\cdot, u) \in \mathcal{L} \quad \text{for all} \quad u \in H_0^1(\Omega), \tag{16.54}$$

in order to define the term $f(\cdot, u)$ as an element of $H^{-1}(\Omega)$. Our abstract setting requires, furthermore, that

$$\mathcal{F} : \left\{ \begin{array}{ccc} H_0^1(\Omega) & \to & H^{-1}(\Omega) \\ u & \mapsto & -\Delta u + f(\cdot, u) \end{array} \right. \tag{16.55}$$

is Fréchet-*differentiable*. Since $\Delta : H_0^1(\Omega) \to H^{-1}(\Omega)$ is linear and bounded by (16.46), this amounts to the Fréchet-differentiability of

$$\mathcal{G} : \left\{ \begin{array}{ccc} H_0^1(\Omega) & \to & H^{-1}(\Omega) \\ u & \mapsto & f(\cdot, u). \end{array} \right. \tag{16.56}$$

For this purpose, we require (as in the previous subsection) that $\partial f/\partial y$ is continuous on $\bar{\Omega} \times \mathbb{R}$. But in contrast to the 'strong solutions' setting, this is not sufficient here; the main reason is that $H_0^1(\Omega)$ does not embed into $C(\bar{\Omega})$. We need additional *growth restrictions* on $f(x, y)$ or $(\partial f/\partial y)(x, y)$ as $|y| \to \infty$.

An important (but not the only) admissible class consists of those functions f which satisfy

$$f(\cdot, 0) \in \mathcal{L}, \tag{16.57}$$

$$\frac{\partial f}{\partial y}(\cdot, 0) \quad \text{is a bounded function on } \Omega, \tag{16.58}$$

$$\left| \frac{\partial f}{\partial y}(x, y) - \frac{\partial f}{\partial y}(x, 0) \right| \le c_1 |y|^{r_1} + c_2 |y|^{r_2} \quad (x \in \Omega, \quad y \in \mathbb{R}), \tag{16.59}$$

with non-negative constants c_1, c_2, and with

$$0 < r_1 \le r_2 < \infty \quad \text{if} \quad n = 2, \qquad 0 < r_1 \le r_2 \le \frac{4}{n-2} \quad \text{if} \quad n \ge 3. \tag{16.60}$$

(A 'small' r_1 will make condition (16.59) weak near $y = 0$, and a 'large' r_2 will make it weak for $|y| \to \infty$.)

Lemma 16.2. Let f satisfy (16.57)–(16.59), and assume the continuity of $\partial f/\partial y$. Then $\mathcal{G}$ given by (16.56) is well-defined and Fréchet-differentiable, with derivative $\mathcal{G}'(u) \in \mathcal{B}(H_0^1(\Omega), H^{-1}(\Omega))$ (for $u \in H_0^1(\Omega)$) given by

$$(\mathcal{G}'(u)[v])[\varphi] = \int_\Omega \frac{\partial f}{\partial y}(\cdot, u) v \varphi \, \mathrm{d}x \quad (v, \varphi \in H_0^1(\Omega)). \tag{16.61}$$

The proof of Lemma 16.2 is rather technical, and therefore omitted here.
According to (16.45) and (16.61), we have

$$(\mathcal{F}'(u)[\varphi])[\psi] = \int_\Omega \left[\nabla\varphi \cdot \nabla\psi + \frac{\partial f}{\partial y}(\cdot, u)\varphi\psi \right] \mathrm{d}x \qquad (16.62)$$
$$= (\mathcal{F}'(u)[\psi])[\varphi] \quad (u, \varphi, \psi \in H_0^1(\Omega))$$

for the operator $\mathcal{F}$ defined in (16.55), which in particular implies condition (16.13) (for *any* $\tilde{u} \in H_0^1(\Omega)$; note that (16.5)), in the setting (16.43) and (16.55). Thus, the *dual and symmetric case* (see Section 16.2) applies.

We mention that several simplifications and extensions are possible if the domain Ω is *bounded*.

Again, we now comment on the computation of an approximate solution $\tilde{u}$, and of the terms δ, K, and g satisfying (16.6)–(16.9), needed for the application of Theorem 16.1, here in the setting (16.43) and (16.55).

16.3.1. Computation of $\tilde{u}$

By (16.43), $\tilde{u}$ needs to be in $X = H_0^1(\Omega)$ only (and no longer in $H^2(\Omega)$, as in the 'strong solutions' approach of the previous subsection). In the finite element context, this significantly increases the class of permitted elements; for example, the 'usual' linear (or quadratic) triangular elements can be used. In the case of an unbounded domain Ω, we are, furthermore, allowed to use approximations $\tilde{u}$ of the form

$$\tilde{u} = \begin{cases} \tilde{u}_0 & \text{on } \Omega_0, \\ 0 & \text{on } \Omega \setminus \Omega_0, \end{cases} \qquad (16.63)$$

with $\Omega_0 \subset \Omega$ denoting some bounded subdomain (the 'computational' domain), and $\tilde{u}_0 \in H_0^1(\Omega_0)$ some approximate solution of the differential equation (16.1) on Ω_0, subject to Dirichlet boundary conditions on $\partial\Omega_0$.

We pose the additional condition of $\tilde{u}$ being *bounded*, which on one hand is satisfied anyway for all practical numerical schemes, and on the other hand turns out to be very useful in the following.

16.3.2. Defect bound δ

By (16.43) and (16.55), condition (16.6) for the defect bound δ now amounts to

$$\| -\Delta\tilde{u} + f(\cdot, \tilde{u}) \|_{H^{-1}} \le \delta, \qquad (16.64)$$

which is a slightly more complicated task than computing an upper bound for an integral (as was needed in Section 16.2). The best general way seems to be the following. First compute an additional approximation $\rho \in H(\mathrm{div}, \Omega)$ to $\nabla\tilde{u}$. (Here, $H(\mathrm{div}, \Omega)$ denotes the space of all vector-valued functions $\tau \in L^2(\Omega)^n$ with weak derivative $\mathrm{div}\,\tau$ in $L^2(\Omega)$. Hence,

obviously $H(\mathrm{div},\Omega) \supset H^1(\Omega)^n$, and ρ can be computed *e.g.*, by interpolation (or some more general projection) of $\nabla\tilde{u}$ in $H(\mathrm{div},\Omega)$, or in $H^1(\Omega)^n$. It should be noted that ρ comes 'for free' as a part of the approximation, if *mixed* finite elements are used to compute $\tilde{u}$.

Furthermore, according to the arguments before and after (16.52), applied with $p = p' = 2$,

$$\|w\|_{H^{-1}} \le C_2\|w\|_{L^2} \quad \text{for all} \quad w \in L^2(\Omega). \tag{16.65}$$

For *explicit* calculation of C_2, we refer to the appendix in Plum (2008). By (16.46) and (16.65),

$$\begin{aligned}\|-\Delta\tilde{u} + f(\cdot,\tilde{u})\|_{H^{-1}} &\le \|\,\mathrm{div}(-\nabla\tilde{u}+\rho)\|_{H^{-1}} + \|-\mathrm{div}\,\rho + f(\cdot,\tilde{u})\|_{H^{-1}} \\ &\le \|\nabla\tilde{u}-\rho\|_{L^2} + C_2\|-\mathrm{div}\,\rho + f(\cdot,\tilde{u})\|_{L^2}, \end{aligned}\tag{16.66}$$

which reduces the computation of a defect bound δ (satisfying (16.64)) to computing bounds for two *integrals*, *i.e.*, we are back to the situation already discussed in Section 16.2.2.

There is an alternative way to compute δ if $\tilde{u}$ is of the form (16.63), with $\tilde{u}_0 \in H^2(\Omega_0) \cap H_0^1(\Omega_0)$, and with Ω_0 having a Lipschitz boundary. This situation can arise, *e.g.*, if Ω is the whole of $\mathbb{R}^n$, and the 'computational' domain Ω_0 is chosen as a 'large' rectangle, whence $\tilde{u}_0$ can be given, for instance, in the form (16.24).

Using partial integration on Ω_0, it follows that

$$\begin{aligned}&\|-\Delta\tilde{u} + f(\cdot,\tilde{u})\|_{H^{-1}} \le \\ &\quad C_2\Big[\|-\Delta\tilde{u}_0 + f(\cdot,\tilde{u}_0)\|^2_{L^2(\Omega_0)} + \|f(\cdot,0)\|^2_{L^2(\Omega\setminus\Omega_0)}\Big]^{\frac{1}{2}} + C_{tr}\left\|\frac{\partial\tilde{u}_0}{\partial\nu_0}\right\|_{L^2(\partial\Omega_0)},\end{aligned}\tag{16.67}$$

with C_{tr} denoting a constant for the trace embedding $H^1(\Omega_0) \hookrightarrow L^2(\partial\Omega_0)$, the explicit computation of which is addressed in the appendix of Plum (2008), and $\partial\tilde{u}_0/\partial\nu_0$, the normal derivative on $\partial\Omega_0$.

16.3.3. Bound K for L^{-1}

According to (16.43), condition (16.7) now reads

$$\|u\|_{H_0^1} \le K\|L[u]\|_{H^{-1}} \quad \text{for all} \quad u \in H_0^1(\Omega), \tag{16.68}$$

with L defined in (16.5), now given by (note that (16.55), (16.56))

$$L = -\Delta + \mathcal{G}'(\tilde{u}) : H_0^1(\Omega) \to H^{-1}(\Omega).$$

Under the growth conditions (16.57)–(16.60), Lemma 16.2 (or (16.61)) shows that, more concretely,

$$(L[\varphi])[\psi] = \int_\Omega \left[\nabla\varphi\cdot\nabla\psi + \frac{\partial f}{\partial y}(\cdot,\tilde{u})\varphi\psi\right]\mathrm{d}x \quad (\varphi,\psi \in H_0^1(\Omega)). \tag{16.69}$$

Making use of the isomorphism $\Phi : H_0^1(\Omega) \to H^{-1}(\Omega)$ given by (16.12) or (16.47), we obtain

$$\|L[u]\|_{H^{-1}} = \|\Phi^{-1}L[u]\|_{H_0^1} \quad (u \in H_0^1(\Omega)).$$

Since, moreover, $\Phi^{-1}L$ is $\langle\cdot,\cdot\rangle_{H_0^1}$-symmetric by (16.69) and (16.13), and defined on the whole Hilbert space $H_0^1(\Omega)$, and hence *self-adjoint*, we find that (16.68) holds for any

$$K \geq [\min\{|\lambda| : \lambda \text{ is in the spectrum of } \Phi^{-1}L\}]^{-1}, \tag{16.70}$$

provided that the min is positive (which is clearly an unavoidable condition for $\Phi^{-1}L$ being invertible with bounded inverse). Thus, in order to compute K, bounds are needed for:

(I) the essential spectrum of $\Phi^{-1}L$ (*i.e.*, accumulation points of the spectrum, and eigenvalues of infinite multiplicity), and

(II) the isolated eigenvalues of $\Phi^{-1}L$ of finite multiplicity, more precisely those neighbouring 0.

With regard to (I), if Ω is unbounded, we suppose again that $\tilde{u}$ is given in the form (16.63), with some bounded Lipschitz domain $\Omega_0 \subset \Omega$. If Ω is bounded, we may assume the same, simply choosing $\Omega_0 := \Omega$ (and $\tilde{u}_0 := \tilde{u}$).

Now define $L_0 : H_0^1(\Omega) \to H^{-1}(\Omega)$ by (16.69), but with $(\partial f/\partial y)(x, \tilde{u}(x))$ replaced by $(\partial f/\partial y)(x, 0)$. Using the Sobolev–Kondrachov–Rellich Embedding Theorem (Adams 1975), implying the compactness of the embedding $H^1(\Omega_0) \hookrightarrow L^2(\Omega_0)$, we find that $\Phi^{-1}L - \Phi^{-1}L_0 : H_0^1(\Omega) \to H_0^1(\Omega)$ is compact. Therefore, the perturbation result given in Kato (1966, IV, Theorem 5.35) shows that the essential spectra of $\Phi^{-1}L$ and $\Phi^{-1}L_0$ coincide. Thus, being left with the computation of bounds for the essential spectrum of $\Phi^{-1}L_0$, one can use Fourier transform methods, for instance, if $\Omega = \mathbb{R}^n$ and $(\partial f/\partial y)(\cdot, 0)$ is constant, or Floquet theory if $(\partial f/\partial y)(\cdot, 0)$ is periodic. Alternatively, if

$$\frac{\partial f}{\partial y}(x, 0) \geq c_0 > -\rho^* \quad (x \in \Omega), \tag{16.71}$$

with $\rho^* \in [0, \infty)$ denoting the minimal point of the spectrum of $-\Delta$ on $H_0^1(\Omega)$, we obtain by straightforward estimates of the Rayleigh quotient that the (full) spectrum of $\Phi^{-1}L_0$, and thus in particular the essential spectrum, is bounded from below by $\min\{1, (c_0 + \rho^*)/(\sigma + \rho^*)\}$.

With regard to (II), for computing bounds to eigenvalues of $\Phi^{-1}L$, the parameter σ in the H_0^1-product (16.44) is chosen such that

$$\sigma > \frac{\partial f}{\partial y}(x, \tilde{u}(x)) \quad (x \in \Omega). \tag{16.72}$$

Thus, the right-hand side of (16.72) is assumed to be bounded above. Furthermore, assume that the infimum s_0 of the essential spectrum of $\Phi^{-1}L$ is *positive*, which is true, *e.g.*, if (16.71) holds. As a particular consequence of (16.72) (and (16.47)) we obtain that $s_0 \le 1$ and all eigenvalues of $\Phi^{-1}L$ are less than 1, and that, via the transformation $\kappa = 1/(1-\lambda)$, the eigenvalue problem $\Phi^{-1}L[u] = \lambda u$ is equivalent to

$$-\Delta u + \sigma u = \kappa\left(\sigma - \frac{\partial f}{\partial y}(\cdot, \tilde{u})\right)u \tag{16.73}$$

(to be understood as an equation in $H^{-1}(\Omega)$), which is furthermore equivalent to the eigenvalue problem for the self-adjoint operator

$$R := (I_{H_0^1(\Omega)} - \Phi^{-1}L)^{-1}.$$

Thus, *defining* the essential spectrum of problem (16.73) to be that of R, we find that it is bounded from below by $1/(1-s_0)$ if $s_0 < 1$, and is empty if $s_0 = 1$. In particular, its infimum is larger than 1, since $s_0 > 0$ by assumption.

Therefore, the computer-assisted eigenvalue enclosure methods mentioned in Section 16.2.3 (which are also applicable to eigenvalues *below* the essential spectrum of a problem like (16.73); see Zimmermann and Mertins (1995)) can be used to enclose the eigenvalue(s) of problem (16.73) neighbouring 1 (if they exist), whence by the transformation $\kappa = 1/(1-\lambda)$ enclosures for the eigenvalue(s) of $\Phi^{-1}L$ neighbouring 0 are obtained (if they exist). Also taking s_0 into account, the desired constant K can now easily be computed via (16.70). (Note that $K = s_0^{-1}$ can be chosen if no eigenvalues below the essential spectrum exist.)

16.3.4. Local Lipschitz bound g for $\mathcal{F}'$

In the setting (16.43), (16.55), condition (16.8) now reads

$$\left|\int_\Omega \left[\frac{\partial f}{\partial y}(x, \tilde{u}(x)+u(x)) - \frac{\partial f}{\partial y}(x, \tilde{u}(x))\right] v(x)\varphi(x)\,\mathrm{d}x\right| \le g(\|u\|_{H_0^1})\|v\|_{H_0^1}\|\varphi\|_{H_0^1} \tag{16.74}$$

for all $u, v, \varphi \in H_0^1(\Omega)$. Here, we assumed that the Fréchet derivative of $\mathcal{G}$ (defined in (16.56)) is given by (16.61), which is true, under the growth conditions (16.57)–(16.60), for example, and which we assume in the following.

As in the strong solution approach treated in Section 16.2, we start with a monotonically non-decreasing function $\tilde{g} : [0,\infty) \to [0,\infty)$ satisfying

$$\left|\frac{\partial f}{\partial y}(x, \tilde{u}(x)+y) - \frac{\partial f}{\partial y}(x, \tilde{u}(x))\right| \le \tilde{g}(|y|) \quad \text{for all } \ x \in \Omega,\ y \in \mathbb{R}, \tag{16.75}$$

and $\tilde{g}(t) \to 0$ as $t \to 0+$, but now we require in addition that $\tilde{g}(t^{1/r})$ is a *concave* function of t. Here, $r := r_2$ is the (larger) exponent in (16.59).

In practice, $\tilde{g}$ can often be taken to have the form

$$\tilde{g}(t) = \sum_{j=1}^{N} a_j t^{\mu_j} \quad (0 \le t < \infty),$$

where $a_1, \ldots, a_N > 0$ and $\mu_1, \ldots, \mu_N \in (0, r]$ are arranged in order to satisfy (16.75).

According to (16.60), one can find some

$$q \in (1, \infty) \quad \text{if} \quad n = 2, \qquad q \in \left[\frac{n}{2}, \infty\right) \quad \text{if} \quad n \ge 3, \tag{16.76}$$

such that qr is in the range (16.50). Since (16.76) implies that $p := 2q/(q-1)$ is also in the range (16.50), both the embeddings $H_0^1(\Omega) \hookrightarrow L^{qr}(\Omega)$ and $H_0^1(\Omega) \hookrightarrow L^p(\Omega)$ are bounded.

Using in addition the concavity of $\psi(t) := \tilde{g}(t^{1/r})$ and Jensen's inequality (Bauer 1978), one can now prove that (16.74) holds for

$$g(t) := C_2^2 \cdot \tilde{g}\big(C_{qr}(C_p/C_2)^{\frac{2}{r}} t\big) \quad (0 \le t < \infty) \tag{16.77}$$

(Plum 2008), which also satisfies (16.9) and is non-decreasing.

16.3.5. A numerical example

We consider the problem of finding non-trivial solutions to the nonlinear Schrödinger equation

$$-\Delta u + V(x)u - u^2 = 0 \quad \text{on} \quad \Omega := \mathbb{R}^2, \tag{16.78}$$

where $V(x) = A + B\sin(\pi(x_1 + x_2))\sin(\pi(x_1 - x_2))$, with real parameters A and B. The results presented here have been obtained by B. Breuer, P. J. McKenna and M. Plum (unpublished).

We are interested only in solutions which are symmetric with respect to reflection about both coordinate axes. Thus, we include these symmetries in all function spaces used, and in the numerical approximation spaces.

The particular case $A = 6$, $B = 2$ is treated. On a 'computational' domain $\Omega_0 := (-\ell, \ell) \times (-\ell, \ell)$, an approximation $\tilde{u}_0 \in H^2(\Omega_0) \cap H_0^1(\Omega_0)$ of the differential equation in (16.78) was computed, with Dirichlet boundary conditions on $\partial\Omega_0$, in a finite Fourier series form like (16.24) (with $N = M = 80$).

To find $\tilde{u}_0$, we start with a non-trivial approximate solution for Emden's equation (which is (16.78) with $A = B = 0$) on Ω_0, and perform a path-following Newton method, deforming (A, B) from $(0, 0)$ into $(6, 2)$.

In the single Newton steps, a collocation method with equidistant collocation points is used. By increasing the side length of Ω_0 in an additional path following, the approximation $\tilde{u}_0$ remains 'stable', with rapidly decreasing normal derivative $\partial\tilde{u}_0/\partial\nu_0$ (on $\partial\Omega_0$), as ℓ increases; this gives rise to

some hope that a 'good' approximation $\tilde{u}$ for problem (16.78) is obtained in the form (16.63).

For $\ell = 8$, $\|\partial\tilde{u}_0/\partial\nu_0\|_{L^2(\partial\Omega_0)}$ turned out to be small enough compared with $\| - \Delta\tilde{u}_0 + V\tilde{u}_0 - \tilde{u}_0^2\|_{L^2(\Omega_0)}$, and a defect bound δ (satisfying (16.64)) is computed via (16.67) as

$$\delta = 0.7102 \cdot 10^{-2}; \tag{16.79}$$

note that, by the results mentioned in the appendix of Plum (2008),

$$C_2 = \sigma^{-\frac{1}{2}}, \quad \text{and} \quad C_{tr} = \sigma^{-\frac{1}{2}}\left[\ell^{-1} + +\sqrt{\ell^{-2} + 2\sigma}\right]^{\frac{1}{2}}.$$

Moreover, (16.72) requires $\sigma > A + B = 8$ (since $\tilde{u}$ turns out to be non-negative). Choosing $\sigma := 9$, we obtain $C_2 \leq 0.3334$ and $C_{\mathrm{tr}} \leq 0.6968$.

Since condition (16.71) holds for $c_0 = A - B = 4$ (and $\rho^* = 0$), the arguments following (16.71) give the lower bound $s_0 := 4/9 \geq 0.4444$ for the essential spectrum of $\Phi^{-1}L$, and hence the lower bound $1/(1 - s_0) = 1.8$ for the essential spectrum of problem (16.73).

By the eigenvalue enclosure methods mentioned in Section 16.2.3, the bounds

$$\kappa_1 \leq 0.5293, \quad \kappa_2 \geq 1.1769$$

for the first two eigenvalues of problem (16.73) could be computed, which by (16.70) leads to the constant

$$K = 6.653 \tag{16.80}$$

satisfying (16.68). To compute g satisfying (16.8) or (16.74), we first note that (16.75) holds for

$$\tilde{g}(t) := 2t,$$

and (16.59) for $r_1 = r_2 = 1$, whence the additional concavity condition is satisfied. Choosing $q := 2$ yields $qr = 2$ and $p = 4$ in the arguments following (16.76), whence (16.77) gives

$$g(t) = 2C_2C_4^2t = \frac{1}{9}t \tag{16.81}$$

since $2C_2C_4^2 = \sigma^{-1}$ by Lemma 2a) in the appendix of Plum (2008).

Using (16.79)–(16.81), it follows that (16.14) and (16.15) hold for $\alpha = 0.04811$, whence Theorem 16.1 implies the existence of a solution $u^* \in H_0^1(\mathbb{R}^2)$ to problem (16.78) with

$$\|u^* - \tilde{u}\|_{H_0^1} \leq 0.04811. \tag{16.82}$$

It is easy to check on the basis of the numerical data that $\|\tilde{u}\|_{H_0^1} > 0.04811$, whence (16.82) shows in particular that u^* is non-trivial.

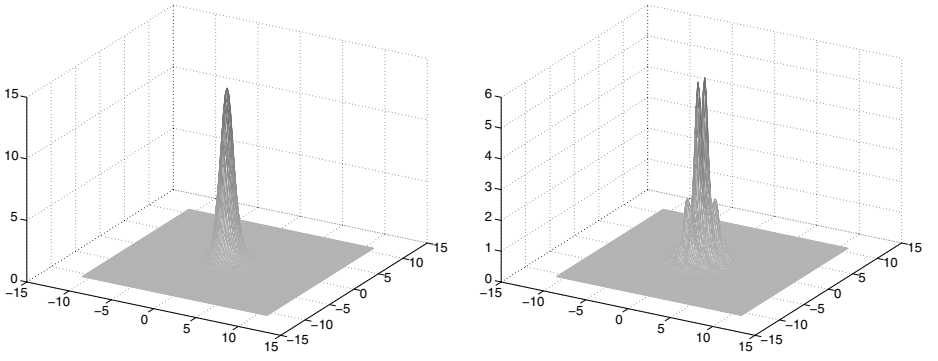

Figure 16.2. Example (16.78); $A = 6$, $B = 2$ (*left*) and $A = 6$, $B = 26$ (*right*).

We wish to remark that it would be of great interest to achieve such results also for cases where $0 < A < B$ in the potential V, because in this case V is no longer non-negative, which excludes an important class of purely analytical approaches to prove existence of a non-trivial solution.

So far, verification has not been successful for such cases, due to difficulties in the homotopy method that has to be used for our computer-assisted eigenvalue enclosures (see the brief remarks in Section 16.2.3); note that these difficulties occur on a rather 'technical' level. However, an apparently 'good' approximation $\tilde{u}$, *e.g.*, in the case $A = 6$, $B = 26$, could be computed.

Figure 16.2 shows plots of $\tilde{u}$ for the successful case $A = 6$, $B = 2$, and for the unsuccessful case $A = 6$, $B = 26$.

Acknowledgements

Many colleagues helped to improve earlier versions of this manuscript. In particular I want to express my sincere thanks to Florian Bünger, Luiz Henrique de Figueiredo, Viktor Härter, Christian Jansson, Raymond Moore, Arnold Neumaier, and Nick Trefethen for their thorough reading and for many valuable comments. Furthermore my thanks to John Pryce, Jiri Rohn, Hermann Schichl, Sergey Shary, and many others.

Moreover, my special thanks to Florian Bünger and my friend Michael Plum from Karlsruhe for their detailed advice on the penultimate section. In particular, I am indebted to Michael Plum for providing the last section of this paper. Finally, my thanks to my dear colleague and friend Professor Shin'ichi Oishi from Waseda university, Tokyo, for the opportunity to write this article in the pleasant atmosphere of his institute.

Last but not least, my thanks to the staff of *Acta Numerica*, chief amongst them Glennis Starling, for an excellent copy-editing job.

REFERENCES

J. P. Abbott and R. P. Brent (1975), 'Fast local convergence with single and multistep methods for nonlinear equations', *Austr. Math. Soc. B* **19**, 173–199.

ACRITH (1984), *IBM High-Accuracy Arithmetic Subroutine Library* (ACRITH), Release 1, IBM Deutschland GmbH, Böblingen.

R. A. Adams (1975), *Sobolev Spaces*, Academic Press, New York.

G. Alefeld, private communication.

G. Alefeld (1994), Inclusion methods for systems of nonlinear equations. In *Topics in Validated Computations* (J. Herzberger, ed.), Studies in Computational Mathematics, Elsevier, Amsterdam, pp. 7–26.

G. Alefeld and J. Herzberger (1974), *Einführung in die Intervallrechnung*, BI Wissenschaftsverlag.

G. Alefeld, V. Kreinovich and G. Mayer (1997), 'On the shape of the symmetric, persymmetric, and skew-symmetric solution set', *SIAM J. Matrix Anal. Appl.* **18**, 693–705.

G. Alefeld, V. Kreinovich and G. Mayer (2003), 'On the solution sets of particular classes of linear interval systems', *J. Comput. Appl. Math.* **152**, 1–15.

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. C. Sorensen (1995), *LAPACK User's Guide, Release* 2.0, 2nd edn, SIAM, Philadelphia.

M. V. A. Andrade, J. L. D. Comba and J. Stolfi (1994), Affine arithmetic. Extended abstract, presented at *INTERVAL'94*, St. Petersburg.

ARITHMOS (1986), *ARITHMOS: Benutzerhandbuch*, Siemens AG, Bibl.-Nr. U 2900-I-Z87-1 edition.

M. Aschbacher (1994), *Sporadic Groups*, Cambridge University Press.

A. Avizienis (1961), 'Signed-digit number representations for fast parallel arithmetic', *Ire Trans. Electron. Comp.* **EC-10**, 389–400.

H. Bauer (1978), *Wahrscheinlichkeitstheorie und Grundzüge der Maßtheorie*, 3rd edn, de Gruyter, Berlin.

O. Beaumont (2000), Solving interval linear systems with oblique boxes. Research report PI 1315, INRIA.

H. Behnke (1989), Die Bestimmung von Eigenwertschranken mit Hilfe von Variationsmethoden und Intervallarithmetik. Dissertation, Institut für Mathematik, TU Clausthal.

H. Behnke and F. Goerisch (1994), Inclusions for eigenvalues of selfadjoint problems. In *Topics in Validated Computations* (J. Herzberger, ed.), Studies in Computational Mathematics, Elsevier, Amsterdam, pp. 277–322.

A. Ben-Tal and A. Nemirovskii (2001), *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, SIAM, Philadelphia.

F. Bernelli Zazzera, M. Vasile, M. Massari and P. Di Lizia (2004), Assessing the accuracy of interval arithmetic estimates in space flight mechanics. Final report, Ariadna id: 04/4105, Contract Number: 18851/05/NL/MV.

Y. Bertot and P. Castéran (2004), *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*, Texts in Theoretical Computer Science, Springer.

M. Berz and K. Makino (1999), 'New methods for high-dimensional verified quadrature', *Reliable Computing* **5**, 13–22.

C. H. Bischof, A. Carle, G. Corliss and A. Griewank (1991), ADIFOR: Generating derivative codes from Fortran programs. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory.

B. Borchers (1999), 'SDPLIB 1.2: A library of semidefinite programming test problems', *Optim. Methods Software* **11**, 683–690.

F. Bornemann, D. Laurie, S. Wagon and J. Waldvogel (2004), *The SIAM 100-Digit Challenge: A Study in High-Accuracy Numerical Computing*, SIAM, Philadelphia.

N. C. Börsken (1978), Komplexe Kreis-Standardfunktionen. Diplomarbeit, Freiburger Intervall-Ber. 78/2, Institut für Angewandte Mathematik, Universität Freiburg.

K. D. Braune (1987), Hochgenaue Standardfunktionen für reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern. Dissertation, Universität Karlsruhe.

B. Breuer, J. Horák, P. J. McKenna and M. Plum (2006), 'A computer-assisted existence and multiplicity proof for travelling waves in a nonlinearly supported beam', *J. Diff. Equations* **224**, 60–97.

B. Breuer, P. J. McKenna and M. Plum (2003), 'Multiple solutions for a semilinear boundary value problem: A computational multiplicity proof', *J. Diff. Equations* **195**, 243–269.

B. M. Brown, D. K. R. McCormack and A. Zettl (2000), 'On a computer assisted proof of the existence of eigenvalues below the essential spectrum of the Sturm–Liouville problem', *J. Comput. Appl. Math.* **125**, 385–393.

M. W. Browne (1988), Is a math proof a proof if no one can check it? *The New York Times*, December 1988, p. 1.

J. R. Bunch, J. W. Demmel and C. F. Van Loan (1989), 'The strong stability of algorithms for solving symmetric linear systems', *SIAM J. Matrix Anal. Appl.* **10**, 494–499.

F. Bünger (2008), private communication.

O. Caprani and K. Madsen (1978), 'Iterative methods for interval inclusion of fixed points', *BIT Numer. Math.* **18**, 42–51.

F. Chatelin (1988), Analyse statistique de la qualité numérique et arithmétique de la résolution approchée d'équations par calcul sur ordinateur. Technical Report F.133, Centre Scientifique IBM–France.

Y. S. Choi and P. J. McKenna (1993), 'A mountain pass method for the numerical solutions of semilinear elliptic problems', *Nonlinear Anal. Theory Methods Appl.* **20**, 417–437.

L. Collatz (1942), 'Einschließungssatz für die charakteristischen Zahlen von Matrizen', *Math. Z.* **48**, 221–226.

G. F. Corliss and L. B. Rall (1987), 'Adaptive, self-validating numerical quadrature', *SIAM J. Sci. Statist. Comput.* **8**, 831–847.

G. Corliss, C. Faure, A. Griewank, L. Hascoët and U. Nauman (2002), *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Springer.

A. Cuyt, B. Verdonk, S. Becuwe and P. Kuterna (2001), 'A remarkable example of catastrophic cancellation unraveled', *Computing* **66**, 309–320.

E. N. Dancer and S. S. Yan (2005), 'On the superlinear Lazer–McKenna conjecture', *J. Diff. Equations* **210**, 317–351.

G. Darboux (1876), 'Sur les développements en série des fonctions d'une seule variable', *J. des Mathématiques Pures et Appl.* **3**, 291–312.

M. Daumas, G. Melquiond and C. Muñoz (2005), Guaranteed proofs using interval arithmetic. In *Proc. 17th IEEE Symposium on Computer Arithmetic* (ARITH'05).

T. J. Dekker (1971), 'A floating-point technique for extending the available precision', *Numer. Math.* **18**, 224–242.

J. B. Demmel (1989), On floating point errors in Cholesky. LAPACK Working Note 14 CS-89-87, Department of Computer Science, University of Tennessee, Knoxville, TN, USA.

J. B. Demmel, B. Diament and G. Malajovich (2001), 'On the complexity of computing error bounds', *Found. Comput. Math.* **1**, 101–125.

J. B. Demmel, I. Dumitriu, O. Holtz and P. Koev (2008), Accurate and efficient expression evaluation and linear algebra. In *Acta Numerica*, Vol. 17, Cambridge University Press, pp. 87–145.

J. B. Demmel, Y. Hida, W. Kahan, X. S. Li, S. Mukherjee and E. J. Riedy (2004), Error bounds from extra precise iterative refinement. Report no. ucb/csd-04-1344, Computer Science Devision (EECS), University of California, Berkeley.

P. S. Dwyer (1951), *Linear Computations*, Wiley, New York/London.

C. Eckart and G. Young (1936), 'The approximation of one matrix by another of lower rank', *Psychometrika* **1**, 211–218.

J.-P. Eckmann, H. Koch and P. Wittwer (1984), 'A computer-assisted proof of universality for area-preserving maps', *Mem. Amer. Math. Soc.* **47**, 289.

P. Eijgenraam (1981), The solution of initial value problems using interval arithmetic.

B. Fazekas, M. Plum and C. Wieners (2005), Enclosure for biharmonic equation. In *Dagstuhl Online Seminar Proceedings 05391*. http://drops.dagstuhl.de/portal/05391/.

L. H. de Figueiredo and J. Stolfi (2004), 'Affine arithmetic: Concepts and applications', *Numer. Algorithms* **37**, 147–158.

L. V. Foster (1994), 'Gaussian elimination with partial pivoting can fail in practice', *SIAM J. Matrix Anal. Appl.* **14**, 1354–1362.

L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier and P. Zimmermann (2005), MPFR: A multiple-precision binary floating-point library with correct rounding. Research Report RR-5753, INRIA. Code and documentation available at: http://hal.inria.fr/inria-00000818.

A. Frommer (2001), Proving conjectures by use of interval arithmetic. In *Perspectives on Enclosure Methods: SCAN 2000* (U. Kulisch *et al.*, ed.), Springer.

Z. Galias and P. Zgliczynski (1998), 'Computer assisted proof of chaos in the Lorenz equations', *Physica* D **115**, 165–188.

I. Gargantini and P. Henrici (1972), 'Circular arithmetic and the determination of polynomial zeros', *Numer. Math.* **18**, 305–320.

B. Gidas, W. Ni and L. Nirenberg (1979), 'Symmetry and related problems via the maximum principle', *Comm. Math. Phys.* **68**, 209–243.

D. Gilbarg and N. S. Trudinger (1983), *Elliptic Partial Differential Equations of Second Order*, 2nd edn, Springer.

D. Goldberg (1991), 'What every computer scientist should know about floating-point arithmetic', *ACM Comput. Surv.* **23**, 5–48.

M. J. C. Gordon (2000), From LCF to HOL: A short history. In *Proof, Language, and Interaction: Essays in Honour of Robin Milner* (G. Plotkin, C. Stirling and M. Tofte, eds), MIT Press. http://www.cl.cam.ac.uk/~mjcg/papers/HolHistory.html.

D. Gorenstein, R. Lyons and R. Solomon (1994), *The Classification of the Finite Simple Groups*, Vol. 40 of *Math. Surveys Monographs*, AMS, Providence, RI.

A. Griewank (2003), A mathematical view of automatic differentiation. In *Acta Numerica*, Vol. 12, Cambridge University Press, pp. 321–398.

P. Grisvard (1985), *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston.

E. Hansen (1969), The centered form. In *Topics in Interval Analysis* (E. Hansen, ed.), Oxford University Press, pp. 102–106.

E. R. Hansen and R. Smith (1967), 'Interval arithmetic in matrix computations II', *SIAM J. Numer. Anal.* **4**, 1–9.

G. Hargreaves (2002), Interval analysis in MATLAB. Master's thesis, University of Manchester. http://www.manchester.ac.uk/mims/eprints.

J. Hass, M. Hutchings and R. Schlafly (1995), 'The double bubble conjecture', *Electron. Res. Announc. Amer. Math. Soc.* **1**, 98–102.

D. J. Higham and N. J. Higham (1992*a*), 'Componentwise perturbation theory for linear systems with multiple right-hand sides', *Linear Algebra Appl.* **174**, 111–129.

D. J. Higham and N. J. Higham (1992*b*), 'Backward error and condition of structured linear systems', *SIAM J. Matrix Anal. Appl.* **13**, 162–175.

N. J. Higham (2002), *Accuracy and Stability of Numerical Algorithms*, 2nd edn, SIAM, Philadelphia.

J. Hölzl (2009), Proving real-valued inequalities by computation in Isabelle/HOL. Diplomarbeit, Fakultät für Informatik der Technischen Universität München.

IEEE 754 (2008), *ANSI/IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic*, New York.

C. Jansson (1991), 'Interval linear systems with symmetric matrices, skew-symmetric matrices, and dependencies in the right hand side', *Computing* **46**, 265–274.

C. Jansson (1994), On self-validating methods for optimization problems. In *Topics in Validated Computations* (J. Herzberger, ed.), Studies in Computational Mathematics, Elsevier, Amsterdam, pp. 381–438.

C. Jansson (1997), 'Calculation of exact bounds for the solution set of linear interval systems', *Linear Algebra Appl.* **251**, 321–340.

C. Jansson (2004*a*), 'A rigorous lower bound for the optimal value of convex optimization problems', *J. Global Optim.* **28**, 121–137.

C. Jansson (2004*b*), 'Rigorous lower and upper bounds in linear programming', *SIAM J. Optim.* **14**, 914–935.

C. Jansson (2006), VSDP: A MATLAB software package for verified semidefinite programming. In *NOLTA 2006*, pp. 327–330.

C. Jansson (2009), 'On verified numerical computations in convex programming', *Japan J. Indust. Appl. Math.* **26**, 337–363.

C. Jansson and J. Rohn (1999), 'An algorithm for checking regularity of interval matrices', *SIAM J. Matrix Anal. Appl.* **20**, 756–776.

C. Jansson, D. Chaykin and C. Keil (2007), 'Rigorous error bounds for the optimal value in semidefinite programming', *SIAM J. Numer. Anal.* **46**, 180–200. http://link.aip.org/link/?SNA/46/180/1.

W. M. Kahan (1968), A more complete interval arithmetic. Lecture notes for a summer course at the University of Michigan.

Y. Kanzawa and S. Oishi (1999*a*), 'Imperfect singular solutions of nonlinear equations and a numerical method of proving their existence', *IEICE Trans. Fundamentals* **E82-A**, 1062–1069.

Y. Kanzawa and S. Oishi (1999*b*), 'Calculating bifurcation points with guaranteed accuracy', *IEICE Trans. Fundamentals* **E82-A**, 1055–1061.

T. Kato (1966), *Perturbation Theory for Linear Operators*, Springer, New York.

R. B. Kearfott (1997), 'Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems', *SIAM J. Sci. Comput.* **18**, 574–594.

R. B. Kearfott, M. Dawande, K. Du and C. Hu (1992), 'INTLIB: A portable Fortran-77 elementary function library', *Interval Comput.* **3**, 96–105.

R. B. Kearfott, J. Dian and A. Neumaier (2000), 'Existence verification for singular zeros of complex nonlinear systems', *SIAM J. Numer. Anal.* **38**, 360–379.

R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary and P. van Hentenfyck (2005) Standardized notation in interval analysis. In *Proc. XIII Baikal International School–Seminar: Optimization Methods and their Applications*, Vol. 4, Melentiev Energy Systems Institute SB RAS, Irkutsk.

C. Keil (2006), Lurupa: Rigorous error bounds in linear programming. In *Algebraic and Numerical Algorithms and Computer-assisted Proofs* (B. Buchberger, S. Oishi, M. Plum and S. M. Rump, eds), Vol. 05391 of *Dagstuhl Seminar Proceedings*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. http://drops.dagstuhl.de/opus/volltexte/2006/445.

C. Keil and C. Jansson (2006), 'Computational experience with rigorous error bounds for the Netlib linear programming library', *Reliable Computing* **12**, 303–321. http://www.optimization-online.org/DB_HTML/2004/12/1018.html.

R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo and M. Rauch (1993), *C-XSC A C++ Class Library for Extended Scientific Computing*, Springer, Berlin.

O. Knüppel (1994), 'PROFIL/BIAS: A fast interval library', *Computing* **53**, 277–287.

O. Knüppel (1998), PROFIL/BIAS and extensions, Version 2.0. Technical report, Institut für Informatik III, Technische Universität Hamburg–Harburg.

D. E. Knuth (1969), *The Art of Computer Programming: Seminumerical Algorithms*, Vol. 2, Addison-Wesley, Reading, MA.

L. V. Kolev and V. Mladenov (1997), Use of interval slopes in implementing an interval method for global nonlinear DC circuit analysis. *Internat. J. Circuit Theory Appl.* **12**, 37–42.

W. Krämer (1987), Inverse Standardfunktionen für reelle und komplexe Intervallargumente mit *a priori* Fehlerabschätzung für beliebige Datenformate. Dissertation, Universität Karlsruhe.

W. Krämer (1991), Verified solution of eigenvalue problems with sparse matrices. In *Proc. 13th World Congress on Computation and Applied Mathematics*, pp. 32–33.
R. Krawczyk (1969*a*), 'Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken', *Computing* **4**, 187–201.
R. Krawczyk (1969*b*), 'Fehlerabschätzung reeller Eigenwerte und Eigenvektoren von Matrizen', *Computing* **4**, 281–293.
R. Krawczyk and A. Neumaier (1985), 'Interval slopes for rational functions and associated centered forms', *SIAM J. Numer. Anal.* **22**, 604–616.
V. Kreinovich, A. V. Lakeyev and S. I. Noskov (1993), 'Optimal solution of interval linear systems is intractable (NP-hard)', *Interval Comput.* **1**, 6–14.
V. Kreinovich, A. Neumaier and G. Xiang (2008), 'Towards a combination of interval and ellipsoid uncertainty', *Vych. Techn.* (*Computational Technologies*) **13**, 5–16.
U. Kulisch (1981), *Computer Arithmetic in Theory and Practice*, Academic Press.
M. La Porte and J. Vignes (1974), 'Etude statistique des erreurs dans l'arithmétique des ordinateurs: Application au contrôle des résultats d'algorithmes númeriques', *Numer. Math.* **23**, 63–72.
O. A. Ladyzhenskaya and N. N. Uralt́seva (1968), *Linear and Quasilinear Elliptic Equations*, Academic Press, New York.
J. Lahmann and M. Plum (2004), 'A computer-assisted instability proof for the Orr–Sommerfeld equation with Blasius profile', *Z. Angew. Math. Mech.* **84**, 188–204.
C. W. H. Lam, L. Thiel and S. Swiercz (1989), 'The nonexistence of finite projective planes of order 10', *Canad. J. Math.* **41**, 1117–1123.
N. J. Lehmann (1963), 'Optimale Eigenwerteinschließung', *Numer. Math.* **5**, 246–272.
X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, B. J. Thompson, T. Tung and D. J. Yoo (2002), 'Design, implementation and testing of extended and mixed precision BLAS', *ACM Trans. Math. Software* **28**, 152–205.
E. Loh and W. Walster (2002), 'Rump's example revisited', *Reliable Computing* **8**, 245–248.
R. Lohner (1988), Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anordnungen. PhD thesis, University of Karlsruhe.
Maple (2009), *Release 13, Reference Manual*.
S. Markov and K. Okumura (1999), The contribution of T. Sunaga to interval analysis and reliable computing. In *Developments in Reliable Computing* (T. Csendes, ed.), Kluwer, pp. 167–188.
Mathematica (2009), *Release 7.0, Reference Manual*.
MATLAB (2004), *User's Guide, Version 7*, The MathWorks.
R. E. Moore (1962), Interval arithmetic and automatic error analysis in digital computing. Dissertation, Stanford University.
R. E. Moore (1966), *Interval Analysis*, Prentice-Hall, Englewood Cliffs.
R. E. Moore (1977), 'A test for existence of solutions for non-linear systems', *SIAM J. Numer. Anal.* **4**, 611–615.
R. E. Moore (1999), 'The dawning', *Reliable Computing* **5**, 423–424.

R. E. Moore, R. B. Kearfott and M. J. Cloud (2009), *Introduction To Interval Analysis*, Cambridge University Press.

J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, R. Revol, D. Stehlé and S. Torres (2009), *Handbook of Floating-Point Arithmetic*, Birkhäuser, Boston.

K. Nagatou, M. T. Nakao and N. Yamamoto (1999), 'An approach to the numerical verification of solutions for nonlinear elliptic problems with local uniqueness', *Numer. Funct. Anal. Optim.* **20**, 543–565.

M. T. Nakao (1988), 'A numerical approach to the proof of existence of solutions for elliptic problems', *Japan J. Appl. Math.* **5**, 313–332.

M. T. Nakao (1993), Solving nonlinear elliptic problems with result verification using an H^{-1}type residual iteration. *Computing* (Suppl.) **9**, 161–173.

M. T. Nakao and N. Yamamoto (1995), 'Numerical verifications for solutions to elliptic equations using residual iterations with higher order finite elements', *J. Comput. Appl. Math.* **60**, 271–279.

M. T. Nakao, K. Hashimoto and Y. Watanabe (2005), 'A numerical method to verify the invertibility of linear elliptic operators with applications to nonlinear problems', *Computing* **75**, 1–14.

N. S. Nedialkov (1999), Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. PhD dissertation, University of Toronto, Canada.

N. S. Nedialkov and K. R. Jackson (2000), ODE software that computes guaranteed bounds on the solution. In *Advances in Software Tools for Scientic Computing* (H. P. Langtangen, A. M. Bruaset and E. Quak, eds), Springer, pp. 197–224.

NETLIB (2009), Linear Programming Library. http://www.netlib.org/lp.

A. Neumaier (1974), 'Rundungsfehleranalyse einiger Verfahren zur Summation endlicher Summen', *Z. Angew. Math. Mech.* **54**, 39–51.

A. Neumaier (1984), 'New techniques for the analysis of linear interval equations', *Linear Algebra Appl.* **58**, 273–325.

A. Neumaier (1987), 'Overestimation in linear interval equations', *SIAM J. Numer. Anal.* **24**, 207–214.

A. Neumaier (1988), 'An existence test for root clusters and multiple roots', *Z. Angew. Math. Mech.* **68**, 256–257.

A. Neumaier (1989), 'Rigorous sensitivity analysis for parameter-dependent systems of equations', *J. Math. Anal. Appl.* **144**, 16–25.

A. Neumaier (1990), *Interval Methods for Systems of Equations*, Encyclopedia of Mathematics and its Applications, Cambridge University Press.

A. Neumaier (1993), 'The wrapping effect, ellipsoid arithmetic, stability and confidence regions', *Computing Supplementum* **9**, 175–190.

A. Neumaier (2001), *Introduction to Numerical Analysis*, Cambridge University Press.

A. Neumaier (2002), 'Grand challenges and scientific standards in interval analysis', *Reliable Computing* **8**, 313–320.

A. Neumaier (2003), 'Enclosing clusters of zeros of polynomials', *J. Comput. Appl. Math.* **156**, 389–401.

A. Neumaier (2004), Complete search in continuous global optimization and constraint satisfaction. In *Acta Numerica*, Vol. 13, Cambridge University Press, pp. 271–369.

A. Neumaier (2009), FMathL: Formal mathematical language. http://www.mat.univie.ac.at/~neum/FMathL.html.
A. Neumaier (2010), 'Improving interval enclosures', *Reliable Computing*. To appear.
A. Neumaier and T. Rage (1993), 'Rigorous chaos verification in discrete dynamical systems', *Physica* D **67**, 327–346.
A. Neumaier and O. Shcherbina (2004), 'Safe bounds in linear and mixed-integer programming', *Math. Program.* A **99**, 283–296.
W. Oettli and W. Prager (1964.), 'Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides', *Numer. Math.* **6**, 405–409
T. Ogita, S. Oishi and Y. Ushiro (2001), 'Fast verification of solutions for sparse monotone matrix equations', *Comput. Suppl.* **15**, 175–187.
S. Oishi (1998), private communication.
S. Oishi (2000), *Numerical Methods with Guaranteed Accuracy* (in Japanese), Corona-sya.
S. Oishi and S. M. Rump (2002), 'Fast verification of solutions of matrix equations', *Numer. Math.* **90**, 755–773.
T. Okayama, T. Matsuo and M. Sugihara (2009), Error estimates with explicit constants for sinc approximation, sinc quadrature and sinc indefinite integration. Technical Report METR2009-01, The University of Tokyo.
J. B. Oliveira and L. H. de Figueiredo (2002) 'Interval computation of Viswanath's constant', *Reliable Computing* **8**, 131–138.
F. Ordóñez and R. M. Freund (2003), 'Computational experience and the explanatory value of condition measures for linear optimization', *SIAM J. Optim.* **14**, 307–333.
M. Overton (2001), *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia.
A. Ovseevich and F. Chernousko (1987), 'On optimal ellipsoids approximating reachable sets', *Problems of Control and Information Theory* **16**, 125–134.
M. Payne and R. Hanek (1983), 'Radian reduction for trigonometric functions', *SIGNUM Newsletter* **18**, 19–24.
K. Petras (2002), 'Self-validating integration and approximation of piecewise analytic functions', *J. Comput. Appl. Math.* **145**, 345–359.
M. Plum (1992), 'Numerical existence proofs and explicit bounds for solutions of nonlinear elliptic boundary value problems', *Computing* **49**, 25–44.
M. Plum (1994), 'Enclosures for solutions of parameter-dependent nonlinear elliptic boundary value problems: Theory and implementation on a parallel computer', *Interval Comput.* **3**, 106–121.
M. Plum (1995), 'Existence and enclosure results for continua of solutions of parameter-dependent nonlinear boundary value problems', *J. Comput. Appl. Math.* **60**, 187–200.
M. Plum (1996), Enclosures for two-point boundary value problems near bifurcation points. In *Scientific Computing and Validated Numerics: Proc. International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, SCAN-95* (G. Alefeld *et al.*, eds), Vol. 90 of *Math. Res.*, Akademie Verlag, Berlin, pp. 265–279.

M. Plum (1997), 'Guaranteed numerical bounds for eigenvalues', In *Spectral Theory and Computational Methods of Sturm–Liouville Problems: Proc. 1996 Conference, Knoxville, TN, USA* (D. Hinton *et al.*, eds), Vol. 191 of *Lect. Notes Pure Appl. Math.*, Marcel Dekker, New York, pp. 313–332.

M. Plum (2008), 'Existence and multiplicity proofs for semilinear elliptic boundary value problems by computer assistance', *DMV Jahresbericht* **110**, 19–54.

M. Plum and C. Wieners (2002), 'New solutions of the Gelfand problem', *J. Math. Anal. Appl.* **269**, 588–606.

S. Poljak and J. Rohn (1993), 'Checking robust nonsingularity is NP-hard', *Math. Control, Signals, and Systems* **6**, 1–9.

L. B. Rall (1981), *Automatic Differentiation: Techniques and Applications*, Vol. 120 of *Lecture Notes in Computer Science*, Springer.

H. Ratschek and J. Rokne (1984), *Computer Methods for the Range of Functions*, Halsted Press, New York.

A. Rauh, E. Auer and E. P. Hofer (2006), ValEncIA-IVP: A comparison with other initial value problem solvers. In *Proc. 12th GAMM–IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics*, SCAN, Duisburg.

K. Rektorys (1980), Variational methods in mathematics. In *Science and Engineering*, 2nd edn, Reidel, Dordrecht.

F. N. Ris (1972), Interval analysis and applications to linear algebra. PhD dissertation, Oxford University.

R. H. Risch (1969), 'The problem of integration in finite terms', *Trans. Amer. Math. Soc.* **139**, 167–189.

J. Rohn (1994), NP-hardness results for linear algebraic problems with interval data. In *Topics in Validated Computations* (J. Herzberger, ed.), Studies in Computational Mathematics, Elsevier, Amsterdam, pp. 463–471.

J. Rohn (2005), A handbook of results on interval linear problems. http://www.cs.cas.cz/rohn/handbook.

J. Rohn (2009*a*), 'Forty necessary and sufficient conditions for regularity of interval matrices: A survey', *Electron. J. Linear Algebra* **18**, 500–512.

J. Rohn (2009*b*), VERSOFT: Verification software in MATLAB/INTLAB. http://uivtx.cs.cas.cz/~rohn/matlab.

J. Rohn and V. Kreinovich (1995), 'Computing exact componentwise bounds on solutions of linear system is NP-hard', *SIAM J. Matrix Anal. Appl.* **16**, 415–420.

S. M. Rump (1980), Kleine Fehlerschranken bei Matrixproblemen. PhD thesis, Universität Karlsruhe.

S. M. Rump (1983), Solving algebraic problems with high accuracy. Habilitationsschrift, published in *A New Approach to Scientific Computation* (U. W. Kulisch and W. L. Miranker, eds), Academic Press, pp. 51–120.

S. M. Rump (1990), 'Rigorous sensitivity analysis for systems of linear and nonlinear equations', *Math. Comput.* **54**, 721–736.

S. M. Rump (1994), Verification methods for dense and sparse systems of equations. In *Topics in Validated Computations* (J. Herzberger, ed.), Elsevier, Studies in Computational Mathematics, Amsterdam, pp. 63–136.

S. M. Rump (1999*a*), INTLAB: INTerval LABoratory. In *Developments in Reliable Computing* (T. Csendes, ed.), Kluwer, Dordrecht, pp. 77–104. http://www.ti3.tu-harburg.de/rump/intlab/index.html.

S. M. Rump (1999*b*), 'Fast and parallel interval arithmetic', *BIT Numer. Math.* **39**, 539–560.

S. M. Rump (2001*a*), Rigorous and portable standard functions. *BIT Numer. Math.* **41**, 540–562.

S. M. Rump (2001*b*), 'Computational error bounds for multiple or nearly multiple eigenvalues', *Linear Algebra Appl.* **324**, 209–226.

S. M. Rump (2003*a*), 'Ten methods to bound multiple roots of polynomials', *J. Comput. Appl. Math.* **156**, 403–432.

S. M. Rump (2003*b*), 'Structured perturbations I: Normwise distances', *SIAM J. Matrix Anal. Appl.* **25**, 1–30.

S. M. Rump (2003*c*), 'Structured perturbations II: Componentwise distances', *SIAM J. Matrix Anal. Appl.* **25**, 31–56.

S. M. Rump (2006), 'Eigenvalues, pseudospectrum and structured perturbations', *Linear Algebra Appl.* **413**, 567–593.

S. M. Rump (2009), 'Ultimately fast accurate summation', *SIAM J. Sci. Comput.* **31**, 3466–3502.

S. M. Rump and S. Graillat (2009), Verified error bounds for multiple roots of systems of nonlinear equations. To appear in *Numer. Algorithms*; published online at Numer Algor DOI 10.1007/s11075-009-9339-3.

S. M. Rump and S. Oishi (2009), Verified error bounds for multiple roots of nonlinear equations. In *Proc. International Symposium on Nonlinear Theory and its Applications: NOLTA'09*.

S. M. Rump and H. Sekigawa (2009), 'The ratio between the Toeplitz and the unstructured condition number', *Operator Theory: Advances and Applications* **199**, 397–419.

S. M. Rump and J. Zemke (2004), 'On eigenvector bounds', *BIT Numer. Math.* **43**, 823–837.

S. M. Rump, T. Ogita and S. Oishi (2008), 'Accurate floating-point summation I: Faithful rounding', *SIAM J. Sci. Comput.* **31**, 189–224.

N. V. Sahinidis and M. Tawaralani (2005), 'A polyhedral branch-and-cut approach to global optimization', *Math. Program.* B **103**, 225–249.

H. Schichl and A. Neumaier (2004), 'Exclusion regions for systems of equations', *SIAM J. Numer. Anal.* **42**, 383–408.

H. Schichl and A. Neumaier (2005), 'Interval analysis on directed acyclic graphs for global optimization', *J. Global Optim.* **33**, 541–562.

S. P. Shary (2002), 'A new technique in systems analysis under interval uncertainty and ambiguity', *Reliable Computing* **8**, 321–419.

G. W. Stewart (1990), 'Stochastic perturbation theory', *SIAM Rev.* **32**, 579–610.

T. Sunaga (1956), Geometry of numerals. Master's thesis, University of Tokyo.

T. Sunaga (1958), 'Theory of an interval algebra and its application to numerical analysis', *RAAG Memoirs* **2**, 29–46.

A. Takayasu, S. Oishi and T. Kubo (2009*a*), Guaranteed error estimate for solutions to two-point boundary value problem. In *Proc. International Symposium on Nonlinear Theory and its Applications: NOLTA'09*, pp. 214–217.

A. Takayasu, S. Oishi and T. Kubo (2009*b*), Guaranteed error estimate for solutions to linear two-point boundary value problems with FEM. In *Proc. Asia Simulation Conference 2009 (JSST 2009), Shiga, Japan*, pp. 1–8.

M. Tawaralani and N. V. Sahinidis (2004), 'Global optimization of mixed-integer nonlinear programs: A theoretical and computational study', *Math. Program.* **99**, 563–591.

M. J. Todd (2001), Semidefinite programming. In *Acta Numerica*, Vol. 10, Cambridge University Press, pp. 515–560.

L. N. Trefethen (2002), 'The SIAM 100-dollar, 100-digit challenge', *SIAM-NEWS* **35**, 2. http://www.siam.org/siamnews/06-02/challengedigits.pdf.

L. N. Trefethen and R. Schreiber (1990), 'Average-case stability of Gaussian elimination', *SIAM J. Matrix Anal. Appl.* **11**, 335–360.

W. Tucker (1999), 'The Lorenz attractor exists', *CR Acad. Sci., Paris, Sér. I, Math.* **328**, 1197–1202.

R. H. Tütüncü, K. C. Toh and M. J. Todd (2003), 'Solving semidefinite-quadratic-linear programs using SDPT3', *Math. Program.* B **95**, 189–217.

L. Vandenberghe and S. Boyd (1996), 'Semidefinite programming', *SIAM Review* **38**, 49–95.

J. Vignes (1978), 'New methods for evaluating the validity of the results of mathematical computations', *Math. Comp. Simul.* **XX**, 227–249.

J. Vignes (1980), *Algorithmes Numériques: Analyse et Mise en Oeuvre 2: Equations et Systèmes Non Linéaires*, Collection Langages et Algorithmes de l'Informatique, Editions Technip, Paris.

D. Viswanath (1999) 'Random Fibonacci sequences and the number 1.13198824 . . .', *Math. Comp.* **69**, 1131–1155.

D. Viswanath and L. N. Trefethen (1998), 'Condition numbers of random triangular matrices', *SIAM J. Matrix Anal. Appl.* **19**, 564–581.

M. Warmus (1956), 'Calculus of approximations', *Bulletin de l'Academie Polonaise des Sciences* **4**, 253–259.

B. Werner and A. Spence (1984), 'The computation of symmetry-breaking bifurcation points', *SIAM J. Numer. Anal.* **21**, 388–399.

J. H. Wilkinson (1965), *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford.

S. J. Wright (1993), 'A collection of problems for which Gaussian elimination with partial pivoting is unstable', *SIAM J. Sci. Comput.* **14**, 231–238.

N. Yamanaka, T. Okayama, S. Oishi and T. Ogita (2009), 'A fast verified automatic integration algorithm using double exponential formula', *RIMS Kokyuroku* **1638**, 146–158.

R. C. Young (1931), 'The algebra of many-valued quantities', *Mathematische Annalen* **104**, 260–290.

Y.-K. Zhu, J.-H. Yong and G.-Q. Zheng (2005), 'A new distillation algorithm for floating-point summation', *SIAM J. Sci. Comput.* **26**, 2066–2078.

G. Zielke and V. Drygalla (2003), 'Genaue Lösung linearer Gleichungssysteme', *GAMM Mitt. Ges. Angew. Math. Mech.* **26**, 7–108.

S. Zimmermann and U. Mertins (1995), 'Variational bounds to eigenvalues of self-adjoint eigenvalue problems with arbitrary spectrum', *Z. Anal. Anwendungen* **14**, 327–345.