Appendix A

Convergence Acceleration

Dirk Laurie

Analytical methods seem to become more and more in favor in numerical analysis and applied mathematics and thus one can think (and we do hope) that extrapolation procedures will become more widely used in the future.

—Claude Brezinski and Michela Redivo Zaglia [BZ91, p. v]

The idea of applying suitable transformations for the acceleration of the convergence of a series or for the summation of a divergent series is almost as old as analysis itself.

—Ernst Joachim Weniger [Wen89, p. 196]

A.1 The Numerical Use of Sequences and Series

Almost every practical numerical method can be viewed as the approximation of the limit of a sequence

$$s_1, s_2, s_3, \ldots, \tag{A.1}$$

which sometimes arises via the partial sums

$$s_k = \sum_{i=1}^{k} a_i \tag{A.2}$$

of a series, by computing a finite number of its elements. In this discussion we only consider the case in which the elements of the sequence are real numbers (but most of what we have to say goes for complex-valued sequences too) and leave aside the question of vector-valued, matrix-valued, and function-valued, sequences.

The sequence (A.1) and the series (A.2) are theoretically equivalent if we define $s_0 = 0$, but in practice there is some[84] accuracy to be gained when working with series, assuming of course that the a_i can be found to full machine precision, not by the formula $a_i = s_i - s_{i-1}$.

The question whether a series converges is largely irrelevant when the reason for using a series is to approximate its sum numerically.

[84] Not much—see §A.5.3.

Convergence Is Not Sufficient

In the case of convergent sequences for which $\rho = \lim(s - s_{k+1})/(s - s_k)$ exists, the convergence is said to be *linear* if $-1 \leqslant \rho < 1$, *sublinear* or *logarithmic* if $\rho = 1$, and superlinear if $\rho = 0$. There are important series for which ρ does not exist, for example,

$$\frac{1}{\zeta(s)} = \sum_{k=1}^{\infty} \mu(k)\, k^{-s},$$

where ζ is the Riemann zeta function and $\mu(k)$ is the Möbius function (if k is the product of n distinct primes, then $\mu(k) = -1$ for n odd and $\mu(k) = 1$ for n even; otherwise $\mu(k) = 0$) but we shall not have any more to say about them here.

For practical purposes, one requires *fast* convergence, for example, linear convergence with $\rho \ll 1$, or preferably superlinear convergence. A series such as

$$s_k = \sum_{n=1}^{k} \frac{1}{n^2}$$

is useless unless convergence acceleration techniques are applied, because $O(1/\epsilon)$ terms are needed to obtain an accuracy of ϵ.

But even fast convergence is not enough. A series such as

$$s_k = \sum_{n=0}^{k} \frac{x^n}{n!}$$

is useless in floating-point arithmetic for large negative values of x, because the largest term is orders of magnitude larger than the sum. The round-off error in the largest term swamps the tiny sum.

Convergence Is Not Necessary

There is an important class of divergent series, known as *asymptotic* series, which are highly useful numerically. These series typically have partial sums $s_k(x) = \sum_{n=0}^{k} a_n x^n$, where x is a real or complex variable, with the following properties:

- The power series $\sum_{n=0}^{\infty} a_n x^n$ has convergence radius 0.
- For x sufficiently small, the sequence $a_n x^n$ decreases in magnitude until a smallest term is reached and thereafter increases.
- There exists a function $f(x)$ whose formal power series coincides with the given series.
- If the series alternates, $s_k(x) - f(x)$ is smaller in magnitude than $a_k x^k$.

When all these properties hold, an alternating asymptotic series is often more convenient than a convergent series for the purpose of obtaining a good approximate value of $f(x)$ with rigorous error bound: one stops after the smallest term, and the absolute value of that term is the bound. We call the size of that bound the *terminal accuracy* of the asymptotic series.

Even when an asymptotic series is monotonic, the approximation obtained by truncating it after the smallest term may sometimes be good enough for practical purposes, although in this case there is no error bound. Be aware that it is the rule rather than the exception for an asymptotic series to be valid only in certain sectors of the complex plane.

A well-known example of an asymptotic series is the logarithmic form of Stirling's formula for the factorial (see [Olv74, §8.4]),

$$\log(n!) \sim \left(n+\frac{1}{2}\right)\log n - n + \frac{1}{2}\log(2\pi) + \sum_{j=1}^{\infty} \frac{B_{2j}}{2j(2j-1)n^{2j-1}}$$

$$= \left(n+\frac{1}{2}\right)\log n - n + \frac{1}{2}\log(2\pi) + \frac{1}{12n} - \frac{1}{360n^3} + \frac{1}{1260n^5} - \frac{1}{1680n^7} + \cdots,$$

where B_j is the jth Bernoulli number. Since the numbers B_j grow faster than any power of j, the series is divergent for all n. Yet it is very useful for large n and remains valid as the first step in the calculation of $.(z)$ when n is replaced by $z+1$, where z is a complex number not on the negative real axis. Our solution of Problem 5 to 10,000 digits relies on this formula (see §5.7).

Convergence acceleration algorithms are useful for obtaining improved estimates of the limit s of a sequence s_k when

- *the members of the sequence can be computed to high precision;*
- *the behavior of s_k as a function of k is regular enough.*

A.2 Avoiding Extrapolation

There is an important class of series where extrapolation methods can be avoided: those in which the terms (with or without an alternating sign) are explicitly known as analytic functions evaluated at the integers. In such a case one can express the sum as a contour integral. The formulas are (see Theorem 3.6)

$$\sum_{k=1}^{\infty}(-1)^k f(k) = \frac{1}{2i}\int_C f(z)\csc(\pi z)\,dz, \qquad \sum_{k=1}^{\infty} f(k) = \frac{1}{2i}\int_C f(z)\cot(\pi z)\,dz,$$

where

1. the integration contour C runs from ∞ to ∞, starting in the upper half-plane and crossing the real axis between 0 and 1;
2. f decays suitably as $z \to \infty$ and is analytic in the component of the complex plane that contains the integers $1, 2, 3, \ldots$.

There is considerable scope for skill and ingenuity in the selection of the contour and its parametrization. A simple example is given in §1.7, and a more sophisticated one, with a discussion of the issues involved in choosing the contour, in §3.6.2.

The contour integration method is usually more expensive computationally than a suitable extrapolation method, but has the advantage of in principle being able to obtain any desired precision.

A.3 An Example of Convergence Acceleration

Algorithm A.1. Archimedes' Algorithm for Approximating π.

Purpose: To compute for $k = 1, 2, \ldots, m$, the numbers

$$a_k = n_k \sin(\pi/n_k), \quad b_k = n_k \tan(\pi/n_k), \qquad \text{where } n_k = 3 \cdot 2^k,$$

via the intermediate quantities

$$s_k = \csc(\pi/n_k), \qquad t_k = \cot(\pi/n_k).$$

Procedure:

$$t_1 = \sqrt{3}, \quad s_1 = 2, \quad n_1 = 6, \quad a_1 = n_1/s_1, \quad b_1 = n_1/t_1.$$

$$\text{For } k = 2, 3, \ldots, m: \quad \begin{cases} t_k = s_{k-1} + t_{k-1}, & s_k = \sqrt{t_k^2 + 1}, \\ n_k = 2n_{k-1}, & a_k = n_k/s_k, \quad b_k = n_k/t_k. \end{cases}$$

Archimedes did the whole calculation up to $m = 5$ in interval arithmetic (anticipating its invention by over two thousand years), culminating in the famous inequality

$$3\tfrac{10}{71} < \pi < 3\tfrac{1}{7}.$$

In Table A.1 we give the actual lower bounds $\underline{a}_k$ for a_k and upper bounds $\overline{b}_k$ for b_k that Archimedes got, together with their decimal representation, rounded up or down as appropriate to 6 significant digits, and the machine numbers $\hat{a}_k$, $\hat{b}_k$ that one gets by calculating them in IEEE double precision, rounded to 15 significant digits.

Now one might feel that the 15-digit approximations are not much better than the rational bounds: obtaining $\hat{b}_5 - \hat{a}_5 = 0.00168$ rather than $\overline{b}_k - \underline{a}_k = 0.00193$ seems a poor return for double-precision computation. But the extra precision in the values $\hat{a}_k$ and $\hat{b}_k$ can be put to good use. Since we know that $\sin(\pi x)/x$ has a Taylor series of the form

$$c_0 + c_1 x^2 + c_2 x^4 + \cdots$$

it follows that a_k has a series of the form

$$d_0 + d_1 4^{-k} + d_2 4^{-2k} + \cdots. \tag{A.3}$$

Now the sequence $a'_k = a_k + (a_{k+1} - a_k)/(4^1 - 1)$ has a series starting at 4^{-2k}; the sequence $a''_k = a'_k + (a'_{k+1} - a'_k)/(4^2 - 1)$ has a series starting at 4^{-3k}; etc.

These values are given in Table A.2. It is immediately obvious that the numbers in each column agree to more digits than in the previous column. The final number a_1'''' happens to be correct to all 15 digits, but of course we are not supposed to know that.

Still, by looking at the first row, we are able to assert, on the basis of our knowledge and experience of this extrapolation algorithm, that the limit of the original sequence to 12

Table A.1. *Algorithm A.1 as computed by Archimedes and by a modern computer.*

k	$\underline{a}_k$	$\underline{a}_k$	$\hat{a}_k$	$\overline{b}_k$	$\overline{b}_k$	$\hat{b}_k$
1	3	**3**.00000	**3**.000000000000000	$\frac{918}{265}$	**3**.46416	**3**.46410161513775
2	$\frac{9360}{3013\frac{3}{4}}$	**3**.10576	**3**.105828541230249	$\frac{1836}{571}$	**3**.21542	**3**.21539030917347
3	$\frac{5760}{1838\frac{9}{11}}$	**3**.13244	**3**.132628613281238	$\frac{3672}{1162\frac{1}{8}}$	**3**.15973	**3**.15965994209750
4	$\frac{3168}{1009\frac{1}{6}}$	**3**.13922	**3**.139350203046867	$\frac{7344}{2334\frac{1}{4}}$	**3**.14620	**3**.14608621513143
5	$\frac{6336}{2017\frac{1}{4}}$	**3**.14090	**3**.141031950890510	$\frac{14688}{4673\frac{1}{2}}$	**3**.14283	**3**.14271459964537

digits is 3.14159265359, and feel fairly confident that $\hat{a}_1'''''$ is probably correct to 14 digits (knowing that our machine works to just under 16 digits,[85] we are not absolutely sure of the 15th digit).

Similarly, the Taylor series for $\tan(\pi x)/x$ also contains only even powers of x, and therefore b_k can be expressed as a series of the form (A.3). Extrapolation can be applied to the sequence $\hat{b}_k$ to obtain the values in Table A.3. The improvement in convergence is less spectacular, and only an optimist would claim more than nine digits on this evidence. With hindsight we can explain the difference in behavior: the coefficients in the Taylor series for $\tan(\pi x)/x$ do not decay exponentially, since tan is not an entire function.

The typical challenge solver[86] would have been happy, but the dialectic mathematician is not satisfied. How can you trust a number about which you do not even know whether it is greater or less than the desired quantity?

Table A.2. *Accelerated sequences derived from lower bounds in the double-precision version of Algorithm A.1.*

k	$\hat{a}_k'$	$\hat{a}_k''$	$\hat{a}_k'''$	$\hat{a}_k''''$
1	**3.141**10472164033	**3.14159**245389765	**3.14159265357**789	**3.14159265358979**
2	**3.141**56197063157	**3.14159265**045789	**3.14159265358975**	
3	**3.14159**073296874	**3.14159265354**081		
4	**3.14159**253350506			

Let us be quite clear on this point: the art of confidently asserting the correctness up to a certain accuracy of a computed number, without being able to prove the assertion, *but still being right*, is not mathematics. But it is science. Richard Feynman said:

[85] $\log_{10} 2^{53} \approx 15.95$.

[86] I can, of course, only speak for one such.

Table A.3. *Accelerated sequences derived from upper bounds in the double-precision version of Algorithm A.1.*

k	$\hat{b}'_k$	$\hat{b}''_k$	$\hat{b}'''_k$	$\hat{b}''''_k$
1	**3.1**3248654051871	**3.14**165626057574	**3.14159**254298228	**3.14159265**363782
2	**3.14**108315307218	**3.14159**353856967	**3.14159265**320557	
3	**3.14156**163947608	**3.14159266**703939		
4	**3.14159**072781668			

> Mathematics is not a science from our point of view, in the sense that it is not a *natural* science. The test of its validity is not experiment. [FLS63, p. 3-1]

Scientific computing *is* a science from Feynman's point of view. The test of its validity *is* experiment. Having formulated the theory that the sought-for limit to 12 digits is 3.14159265359, we can test it by computing a_6 and calculating another diagonal of extrapolated values, maybe using higher precision, and by repeating the calculation on a different type of computer.

> *More than any other branch of numerical analysis, convergence acceleration is an experimental science. The researcher applies the algorithm and looks at the results to assess their worth.*

A.4 A Selection of Extrapolation Methods

Strictly speaking, the only difference between interpolation and extrapolation is whether the point at which we wish to approximate a function lies inside or outside the convex hull of the points at which the function is known. In theory and in practice, the difference is profound when (as here) extrapolation is used to estimate a limit to infinity. When doing interpolation, the limiting process is one in which typically a step size h is allowed to approach 0, and the behavior of the function to be interpolated becomes more and more polynomial-like as the step size decreases. When doing extrapolation, the limiting process is one in which more and more terms may be taken into account, but there is no question of the behavior of the partial sum s_n becoming polynomial-like as a function of n. Few functions on $[0, \infty)$, except polynomials themselves, asymptotically behave like polynomials.

In a survey like this one, it is impossible to be exhaustive. I have presented my personal favorite convergence acceleration methods in a unified way. Some of them I like because so often they work so well; others are useful steps on the way to understanding the more sophisticated methods. The emphasis is on computational aspects, not on the theory.

Readers who want a more complete presentation should consult the excellent monograph by Brezinski and Redivo-Zaglia [BZ91], which goes into greater detail than we can, surveys all available theoretical results on acceleration of sequences, gives many more methods, and contains computer software for all of them. Other monographs, each with its own strong points, are those of Wimp [Wim81] and Sidi [Sid03]. The historical development of the field is very ably summarized by Brezinski [Bre00]. A report by Weniger [Wen89] has similar aims to this one, but is much more comprehensive.

Most extrapolation algorithms arise naturally from consideration of a sequence, and little attention has been paid to formulating those as series-to-series transformations.

We shall adopt the following notation for all the methods:

- $s_{k,n}$ is an extrapolated value depending on the elements $s_k, s_{k+1}, \ldots, s_{k+n}$ of the sequence. Thus, $s_{k,0} = s_k$.
- X is a generic extrapolation operator that maps a sequence s_k into a sequence $X(s_k)$.

When we need to display[87] the elements $s_{k,n}$ two-dimensionally, we form the following matrix:

$$S = \begin{pmatrix} s_{1,0} & s_{1,1} & s_{1,2} & \cdots & s_{1,n-2} & s_{1,n-1} \\ s_{2,0} & s_{2,1} & s_{2,2} & \cdots & s_{1,n-2} & \\ \vdots & \vdots & & & & \\ s_{n-1,0} & s_{n-1,1} & & & & \\ s_{n,0} & & & & & \end{pmatrix}.$$

Occasionally, there might be a row number 0, if we decide to treat $s_0 = 0$ as a member in good standing of the given sequence. This is not always a sensible thing to do: for example, in the Archimedes case, we could have started the sequence one term sooner (Archimedes didn't) with the semiperimeter of a triangle, which would require $s_0 = 3\sqrt{3}/2$, not $s_0 = 0$.

The advantage of generating the whole triangular table S is to get more insight into the accuracy and reliability of the extrapolation than just one number, or even one sequence of numbers, would give. This matter is taken up in §A.5.4.

All reasonable extrapolation methods are quasi-linear; that is, they satisfy the relation $X(\lambda s_k + \delta) = \lambda X(s_k) + \delta$, where λ and δ are constants. Most (but not all) of them are based on a model of the form

$$s_k = s + \sum_{j=1}^{m} c_j \phi_{k,j} + \eta_k, \tag{A.4}$$

where the auxiliary columns are ordered so that each $\phi_{k,j+1}$ tends to zero as k increases faster than its predecessor $\phi_{k,j}$, and it is hoped that η_k tends to zero more rapidly than $\phi_{k,m}$. Usually (but not always) the elements of the matrix $\{ = [\phi_{k,j}]$ would be given by a formula of the form

$$\phi_{k,j} = \phi_j(k) \tag{A.5}$$

for certain simple functions ϕ_j.

One can classify extrapolation methods as linear or nonlinear; in the latter case, it is useful to make a further distinction between semilinear and strongly nonlinear methods.

[87] Our notation $s_{k,j}$ corresponds to T_j^k in [Bre00]. There is no universal agreement on how to display the extrapolated values. Some authors put them in a lower triangular matrix so that our rows run along the diagonals. Others emphasize the symmetry of the formula by using a triangular array within which each column is offset downwards by half a step from the previous one.

Linear Extrapolation Methods

These methods are linearly invariant; that is, they satisfy the relation

$$X(\lambda s_k + \mu t_k) = \lambda X(s_k) + \mu X(t_k), \quad \text{where } \lambda \text{ and } \mu \text{ are constants.} \tag{A.6}$$

They typically assume a model of the form (A.4), where the matrix { is known beforehand, independent of the sequence elements, although prior information about the sequence may influence the choice of model.

If in the model equations for $s_k, s_{k+1}, \ldots, s_{k+n}$, we put $m = n$, ignore η_k, and replace the unknown s by $s_{k,n}$, we obtain $n + 1$ equations

$$s_i = s_{k,n} + \sum_{j=1}^{n} c_j \phi_{i,j}, \quad i = k, k+1, \ldots, k+n. \tag{A.7}$$

The various methods differ in the model chosen and the way in which the parameters c_j are eliminated to find $s_{k,n}$.

Linear extrapolation methods can only be expected to work when substantial prior knowledge about the behavior of the sequence is exploited.

In other words: the model must describe the sequence fairly well or the extrapolation method won't deliver.

Semilinear Extrapolation Methods

These typically assume a model of the form (A.4), where the matrix elements have the form $\phi k, j = \alpha k \beta k, j$, where αk is allowed to depend in a quantitative way on the elements of the sequence. The resulting linear systems are then solved by a process similar to (A.7). In particular, extrapolated values are available for all pairs (k, l) that satisfy $k + l \leqslant n$.

Semilinear methods can be extremely effective, even when no a priori information about the asymptotic behavior of the sequence is available.

Strongly Nonlinear Extrapolation Methods

These methods either assume a model of the form (A.4), (A.5) in which the functions ϕ_j depend on unknown parameters that lead to nonlinear equations or in some cases have no explicit model at all, being derived by heuristic reasoning from other methods. Extrapolated values $s_{k,l}$ are only available for some pairs (k, l).

A.4.1 Linear Extrapolation Methods

Having selected the auxiliary functions ϕ_j in (A.5), for each index pair (k, l) one could simply solve the equations (A.7) by brute force—they are, after all, linear. That would require $O(n^5)$ operations to calculate all the entries $s_{k,l}$, $k + l \leqslant n$. But we can do much better than this.

Since it is desirable to have all the $s_{k,j}$ available, linear extrapolation methods should be cast into the form

$$s_{k,j} = s_{k+1,j-1} + f_{k,j}(s_{k+1,j-1} - s_{k,j-1}), \tag{A.8}$$

where the multipliers $f_{k,j}$ do not depend on the original sequence s_k. The geometric picture is

$$\begin{pmatrix} s_{k,j-1} & s_{k,j} \\ s_{k+1,j-1} & \end{pmatrix} \cdot \begin{pmatrix} -f_{k,j} & -1 \\ 1+f_{k,j} & \end{pmatrix} = 0.$$

In most cases, the multipliers are easier to interpret when the model is written as

$$s_{k,j} = s_{k+1,j-1} + \frac{r_{k,j}}{1-r_{k,j}}(s_{k+1,j-1} - s_{k,j-1}). \tag{A.9}$$

It is obvious that if all the multipliers $f_{k,j}$ are chosen with complete freedom, then any extrapolation table S that satisfies $s_{k+1,j-1} \neq s_{k,j-1}$ can be obtained. This would defeat the self-validating property of the extrapolation table. Therefore, it is usual to have a model involving no more than n free parameters, where $n+1$ is the number of available terms of the sequence.

Often it is possible to give a simple formula for the constants $f_{k,j}$, so that the whole extrapolation process takes $O(n^2)$ operations, which is optimal, since there are $O(n^2)$ entries in the triangular table. In the general case, such low complexity is not possible, but there is an ingenious way of organizing Gaussian elimination, known as the E-algorithm, which reduces the operation count from $O(n^5)$ to $O(n^3)$. The general outline can be described in a few sentences. Think of the first step of (A.8) as $s_{:,1} = X_j s_{:,0}$; that is, column 1 of S is obtained by applying an extrapolation operator X_j to column 0. Let the constants $f_{:,0}$ be determined by the property that $X(\phi_{:,1}) = 0$. Now replace all the other columns $\phi_{:,j}$ by $X(\phi_{:,j})$, $j = 2, 3, \ldots, n$. In other words: apply exactly the same extrapolation formula to each column of the matrix { that you have applied to the sequence s_k. Then continue using the new column of extrapolated values instead of s_k and the modified matrix { instead of the original, etc.

In the table S, each column $s_{k,j}$, $k = 1, 2, 3, \ldots$, should converge faster than the previous one if the model is appropriate. The rows of the array $s_{k,j}$, $j = 0, 1, 2, \ldots$, should in that case converge even faster. This behavior, of rows converging faster than columns, is conspicuous; its absence indicates that the model is bad.

Euler's Transformation

Euler's method is one of the rare instances of a genuine series-to-series transformation. The derivation of this grandfather of all extrapolation methods is a typical piece of Eulery. Let I be the identity ($Ia_k = a_k$) and Δ the forward difference ($\Delta a_k = a_{k+1} - a_k$) operator on a sequence. Note that $(I+\Delta)a_k = a_{k+1}$. Let $b_k = a_k / r^{k-1}$, with $r \neq 1$. Then

$$\sum_{k=1}^{\infty} a_k = \sum_{k=0}^{\infty} r^k b_{k+1} = \sum_{k=0}^{\infty} (r(I+\Delta))^k b_1$$

$$= (I - r(I+\Delta))^{-1} b_1 = \frac{1}{1-r}\left(I - \frac{r\Delta}{1-r}\right)^{-1} b_1$$

$$= \frac{1}{1-r}\sum_{j=0}^{\infty}\left(\frac{r\Delta}{1-r}\right)^j b_1 = \frac{1}{1-r}\sum_{j=0}^{\infty}\left(\frac{r}{1-r}\right)^j \Delta^j b_1.$$

We get our two-dimensional table by not starting at the first term, but using the transformation only to estimate the tail. This gives

$$s_{k,n} = \sum_{i=1}^{k-1} a_i + \frac{1}{1-r} \sum_{j=0}^{n} \left(\frac{r}{1-r}\right)^j \Delta^j b_k. \tag{A.10}$$

A little algebra shows that we can generate these values recursively columnwise by

$$s_{k,j+1} = s_{k+1,j} + \frac{r}{1-r}(s_{k+1,j} - s_{k,j}). \tag{A.11}$$

So Euler's transformation is seen to be the simplest possible case of (A.8), since $f_{k,l}$ is constant.

Although we have not used an explicit model in the derivation, the form of the final equation (A.10) reveals the underlying model: $s_{k,n}$ is constant in n for $n \geqslant m$ if b_k is a polynomial of degree no more than $m-1$ in k. In other words, the model is

$$a_k = r^k(c_0 + c_1 k + c_2 k^2 + \cdots + c_{n-1} k^{n-1}).$$

It is obvious that a_k has this form when

$$s_k = s + r^k(c_0 + c_1 k + c_2 k^2 + \cdots + c_n k^n).$$

While no doubt such sequences do occur, they are not very typical. More usually, in cases where we do happen to know the correct r, the first column $s_{k,1}$ converges substantially faster than s_k, but the first row $s_{0,k}$, which should be the really fast-converging one, at best has only approximately geometric convergence with factor $r/(1-r)$.

The most common application of Euler's transformation is to an alternating series converging slower than geometrically, when $r = -1$. It is fairly efficient in that case, not because the model is appropriate (it seldom is), but because $r/(1-r) = -1/2$, which is not too bad. For example, the series

$$\sum_{k=1}^{\infty} (-1)^{k-1} k^{-1} = \log 2 \doteq 0.693147180559945$$

is accelerated reasonably well by Euler's transformation, despite the fact that the high-order differences $\Delta^k c_1$ fail to approach zero any faster than a_k does. (It is a nice little exercise to prove that for this series, $\Delta^k c_1 = a_{k+1}$.) In Table A.4 and later in the chapter, we have made a column out of row 0 to facilitate comparison.

Table A.4 shows that, as expected, $s_{0,k}$ converges approximately geometrically with ratio $\frac{1}{2}$: after 10 steps we have three digits, and $2^{-10} \approx 0.001$. It is tempting to exploit that behavior through another application of Euler's transformation with $r = 1/2$, but only the first column is useful. (Guess what the first row will look like if you go all the way. Try it out. Shouldn't you have expected it?) The entries in the first column of this repeated transformation are given in Table A.4 by t_k: one extra digit—not worth the effort.

We have paid a good deal of attention to Euler's transformation, only to find that it is not spectacularly effective. But we now have the shoulders of a giant to stand on.

Table A.4. *Euler's transformation applied to* $\sum_{k=1}^{\infty}(-1)^{k-1}k^{-1}$.

k	s_k	$s_{0,k}$	t_k
1	1.000000000000000	**0**.500000000000000	1.000000000000000
2	**0**.500000000000000	**0.6**25000000000000	**0**.750000000000000
3	**0**.833333333333333	**0.666**666666666667	**0**.708333333333333
4	**0**.583333333333333	**0.682**291666666667	**0.697**916666666667
5	**0**.783333333333333	**0.688**541666666667	**0.694**791666666667
6	**0.6**16666666666667	**0.691**145833333333	**0.693**750000000000
7	**0**.759523809523809	**0.692**261904761905	**0.693**377976190476
8	**0.6**34523809523809	**0.692**750186011905	**0.693**238467261905
9	**0**.745634920634921	**0.692**967199900793	**0.693**184213789682
10	**0.6**45634920634921	**0.693**064856150793	**0.693**162512400794

Modified Euler Extrapolation

An obvious modification of (A.11) is to allow r to have a different value for each column, giving

$$s_{k,j+1} = s_{k+1,j} + \frac{r_j}{1 - r_j}(s_{k+1,j} - s_{k,j}). \tag{A.12}$$

It is not hard to show that this formula gives $s_{k,n} = s$ when

$$s_k = s + \sum_{j=1}^{n} c_j r_j^k. \tag{A.13}$$

It can be used, therefore, when the sequence is well approximated by a sum of geometric sequences with known decay rates. Repeated values of r_j have the same effect as in the case of the ordinary Euler transformation.

Richardson Extrapolation

Richardson extrapolation is appropriate when the sequence behaves like a polynomial in some sequence h_k.

In the original application, the model is (A.4) and (A.5) with

$$\phi_j(k) = h_k^{p_j}, \tag{A.14}$$

where the numbers h_k are step sizes in a finite-difference method and the exponents p_j are known in advance from an analysis involving Taylor series. In the most general case, we need the E-algorithm, but in the two most commonly encountered cases, the parameters $r_{k,j}$ in (A.9) can be found more simply.

1. If the step sizes are in geometric progression, that is, $h_{k+1}/h_k = r$ (the most common case being $r = 1/2$), then we put $r_{k,j} = r^{p_j}$.

2. If the exponents are a constant multiple of $1, 2, 3, \ldots$, that is, they are given by $p_j = cj$ for some constant c, then we put $r_{k,j} = (h_{k+1}/h_k)^c$.

Case 1 can be recognized as equivalent to modified Euler extrapolation, whereas case 2 is an application of Neville–Aitken interpolation.

We have already, in §A.3, seen a spectacular application of Richardson extrapolation, so we need no further example of a case where it is successful. Instead, we show what can happen when it is inappropriately used.

Richardson extrapolation is the engine inside Romberg integration. In that application, s_k is formed from trapezoidal or midpoint rules with the step size being continually halved, for example, using midpoint sums,

$$\int_0^1 f(x)\,dx \approx s_k = h \sum_{j=1}^{2^k} f((j - \tfrac{1}{2})h), \quad \text{where } h = 2^{-k}.$$

For smooth functions, the same model with even powers of h is applicable as in the case of the calculation of π, but Table A.5 shows what happens for

$$\int_0^1 -\log(x)\,dx = 1.$$

The first row converges no better than the first column, which is a sure sign that the model is bad.

A careful analysis of the error expansion will reveal which powers of h are introduced by the logarithmic singularity, but it is also quite easy to diagnose it numerically; see §A.5.2. Since the step sizes are in geometric progression, we can choose p to knock out precisely those powers. If we take the p_j from the sequence $\{1, 2, 4, 6, \ldots\}$, we obtain the results of Table A.6. Note that $s_{k,1}$ converges much faster than before, but that is not the real point. The first row converges much more quickly still, and that is what we are looking for to reassure us that the proper model has been used.

Table A.5. *Romberg integration applied to* $\int_0^1 -\log(x)\,dx = 1$, *assuming an expansion in even powers.*

h_k^{-1}	$s_{k,1}$	$s_{0,k}$
2	**0.9**42272533258662	**0.9**42272533258662
4	**0.97**1121185130247	**0.97**3044428588352
8	**0.985**559581182433	**0.986**736072861005
16	**0.992**779726126725	**0.993**394043936872
32	**0.996**389859013850	**0.996**700250685739
64	**0.998**194929253506	**0.998**350528242663

Table A.6. *Romberg integration applied to* $\int_0^1 -\log(x)\,dx = 1$, *assuming an expansion in the powers* $1, 2, 4, 6, \ldots$.

h_k^{-1}	$s_{k,1}$	$s_{0,k}$
2	**0.99**4914691495074	**0.99**4914691495074
4	**0.998**706050625142	**0.9999**69837001832
8	**0.9996**74995582249	**0.999999**853250138
16	**0.99999**18652198826	**0.999999999**613708
32	**0.999979**656975437	**0.999999999999**550
64	**0.999994**913863731	**0.999999999999999**

Salzer's Extrapolation

Salzer's extrapolation [Sal55] is appropriate when s_k is well approximated by a rational function in k.

The model is (A.4), (A.5) with $\phi_j(k) = (k+k_0)^{-j}$, where k_0 is a fixed number chosen in advance (usually $k_0 = 0$). This is a special case of Richardson extrapolation, obtained by putting $h_k = (k + k_0)^{-1}$ and $p_j = j$ in (A.14). So the extrapolated values can be obtained by case 2 of the Richardson extrapolation.

There are two other ways of solving the equations. Multiplying (A.7) by $(i + k_0)^n$, we obtain

$$(i + k_0)^n s_i = (i + k_0)^n s_{k,n} + \sum_{j=1}^{n} c_j (i + k_0)^{n-j}, \quad i = k, k + 1, \ldots, k + n.$$

Taking the nth difference at $i = k$, everything under the sum vanishes and we are left with

$$s_{k,n} = \frac{\Delta^n((k + k_0)^n s_k)}{\Delta^n (k + k_0)^n}. \tag{A.15}$$

This formula is not recursive (which is close to Salzer's own [Sal55] formulation), and the preliminary multiplication by a different power of $(k + k_0)$ for each column $s_{:,l}$ is undesirable. Still, it gives an interesting view of the extrapolation: it is a discrete analogue of n applications of L'Hospital's rule. Suppose we wanted to find $\lim_{x\to\infty} f(x)$. One way of doing it is to use an auxiliary function g such that $g(x) \to_{x\to\infty} 0$ and to calculate

$$\lim_{x\to\infty} \frac{\frac{d^n}{dx^n}(f(x)/g(x))}{\frac{d^n}{dx^n}(1/g(x))}.$$

The third way is discussed in the next section.

For the series $s = \sum_{k=1}^{\infty} k^{-2} = \pi^2/6 \doteq 1.6449340668482$ we get the result of Table A.7 in IEEE double precision. The final result has about 11 correct digits, which is the best we can hope for because of round-off (see §A.5.3).

Table A.7. *Salzer's extrapolation applied to* $\sum_{k=1}^{\infty} k^{-2}$.

k	$s_{k,1}$	$s_{0,k}$
1	**1.**50000000000000	**1.**50000000000000
2	**1.**58333333333333	**1.**62500000000000
3	**1.**61111111111111	**1.**64351851851852
4	**1.**62361111111111	**1.**64496527777778
5	**1.**63027777777778	**1.**64495138888888
6	**1.**63424603174603	**1.**64493518518521
7	**1.**63679705215420	**1.**64493394341858
8	**1.**63853316326531	**1.**64493404116995
9	**1.**63976773116654	**1.**64493406624713
10	**1.**64067682207563	**1.**64493406714932
11	**1.**64136552730979	**1.**64493406688417
12	**1.**64189971534398	**1.**64493406684468
13	**1.**64232236961279	**1.**64493406683127

Modified Salzer's Extrapolation

Like all linear methods, Salzer extrapolation fails dismally on sequences that fail to conform to the underlying model. A simple modification goes a long way to remedy this.

The modified model is (A.4), (A.5) with $\phi_j(k) = \psi(k)(k+k_0)^{1-j}$ for some nonzero auxiliary function $\psi(k)$. Obviously, best results are obtained when $\psi(k)$ approximates $(s - s_k)$. This model no longer falls in the Richardson framework, and we have to find another computational procedure. The E-algorithm can of course be used, but there is a more economical alternative.

Dividing (A.7) by $\psi(i)$ we obtain

$$\frac{s_i}{\psi(i)} = \frac{s_{k,n}}{\psi(i)} + \sum_{j=0}^{n-1} c_{j+1}(i+k_0)^{-j}, \quad i = k, k+1, \ldots, k+n.$$

Now apply not ordinary differences as before, but divided differences, thinking of s_k as a function of t evaluated at $t_k = (k+k_0)^{-1}$. The divided differences of a sequence f_k given at those abscissas are defined recursively by

$$\delta^0 f_k = f_k;$$

$$\delta^n f_k = \frac{\delta^{n-1} f_{k+1} - \delta^{n-1} f_k}{t_{k+n} - t_k}. \tag{A.16}$$

Table A.8. *Salzer's and modified Salzer's extrapolation applied to* $\sum_{k=1}^{\infty} k^{-3/2}$.

k	$s_{0,k}$ with $\psi = 1$	$s_{0,k}$ with $\psi(k) = k^{-1/2}$
1	1.70710678118655	**2**.20710678118655
2	**2**.04280209908108	**2**.55223464247637
3	**2**.20053703479084	**2.60**809008399373
4	**2**.28756605115163	**2.612**55796998662
5	**2**.34336074156378	**2.612**44505799459
6	**2**.38255729176400	**2.61237**916796721
7	**2**.41169662004271	**2.61237**468295396
8	**2**.43423616494222	**2.61237**522998939
9	**2**.45220141289729	**2.6123753**4769965
10	**2**.46686223192588	**2.6123753**5041131
11	**2**.47905652271137	**2.61237534**876941
12	**2**.48936011369519	**2.61237534**899252
13	**2**.49818208203943	**2.61237534**782506

We obtain

$$s_{k,n} = \frac{\delta^n \left(\frac{s_k}{\psi(k)} \right)}{\delta^n \left(\frac{1}{\psi(k)} \right)}. \tag{A.17}$$

This can be implemented economically by forming two tables of divided differences, one for the denominator and one for the numerator.[88]

One can often guess a good ψ by integration; for example, if ϕ can be extended to a function defined on $(0, \infty)$, then it is plausible to choose

$$\psi(x) = \int_x^{\infty} \phi(x)\, dx. \tag{A.18}$$

As an example we take $s = \sum_{k=1}^{\infty} k^{-3/2} = \zeta(3/2) \doteq 2.61237534868549$. We show in Table A.8 the first row of the Salzer algorithm and of the modified Salzer algorithm with $\psi(k) = k^{-1/2}$, obtained by the method of integration.

Operator Polynomial Extrapolation

A rich family of linear extrapolation methods arises from yet another generalization of Euler's transformation. Express (A.11) in terms of the shift operator E (defined by

[88]This procedure is in fact equivalent to the E-algorithm if one takes advantage of the specially simple form of the dependence of the functions ϕ_j on j.

$Es_k = s_{k+1}$) as

$$s_{k,j+1} = \frac{(E - rI)s_{k,j}}{1 - r} = \frac{p_1(E)s_{k,j}}{p_1(1)},$$

where $p_1(t) = (t - r)$. From this we deduce that

$$s_{k,n} = \frac{p_n(E)s_k}{p_n(1),} \tag{A.19}$$

where $p_n(t) = (t - r)^n$. We get the modified Euler formula when $p_n(t) = \prod_{j=1}^{n}(t - r_j)$. In general, when p_n is a polynomial of degree n, we call (A.19) an *operator polynomial extrapolation* formula. Note that the modified Salzer extrapolation (A.17) can also be viewed as a special case of (A.19).

To understand how to stand on Euler's shoulders, we need the following concept:

> A sequence a_k is *totally monotonic* if every difference $\Delta^j a_k$, $j = 0, 1, 2, \ldots$ (including a_k itself, $j = 0$), has the constant sign $(-1)^j$. A sequence a_k is *totally alternating* if $(-1)^k a_k$ is totally monotonic.

Totally monotonic sequences have many pleasant properties, the most important of which is Hausdorff's theorem:

> The null sequence a_k is totally monotonic if and only if there exists a weight function w, nonnegative over $(0, 1)$, such that
>
> $$a_k = \int_0^1 t^{k-1} w(t)\, dt, \quad k = 1, 2, \ldots .$$

Two corollaries are:

1. The sequence a_k is totally alternating if and only if there exists a weight function w, nonnegative over $(-1, 0)$, such that
$$a_k = \int_0^{-1} t^{k-1} w(t)\, dt, \quad k = 1, 2, \ldots .$$
2. If the support of w is the interval $[0, r]$ for some $0 < r < 1$, or in the alternating case $[r, 0]$ for some $-1 < r < 0$, then a_k converges geometrically with ratio r.

In view of these corollaries, we will henceforth use the notation $[0, r]$ for the closed interval with endpoints 0 and r even when $r < 0$.

Inspired by Hausdorff's theorem, let us suppose that $a_k = \int_0^r t^{k-1} w(t)\, dt$ with w nonnegative over the integration interval. Then

$$s_k = \sum_{j=1}^{k} \int_0^r t^{j-1} w(t)\, dt = \int_0^r \frac{(1 - t^k)w(t)\, dt}{1 - t}.$$

This allows us to represent (A.19) as

$$s_{k,n} = \frac{1}{p_n(1)} \int_0^r \frac{(p_n(1) - t^k p_n(t))w(t)\, dt}{1 - t}.$$

Since $s = \lim s_k = \int_0^r \frac{w(t)\,dt}{1-t}$, we get the error formula

$$s - s_{k,n} = \frac{1}{p_n(1)} \int_0^r \frac{t^k p_n(t) w(t)\,dt}{1-t}. \tag{A.20}$$

The error formula suggests several strategies.

1. If we know nothing about the sequence a_k except that it is totally monotonic or totally alternating, there is the (fairly crude) error bound

$$\left|1 - \frac{s_{0,n}}{s}\right| \leqslant \frac{\max_{t\in[0,r]} |p_n(t)|}{p_n(1)}.$$

In the case of Euler's method for an alternating series with $r = -1$, we have $p_n(t) = (t+1)^n$, so that the crude bound is $O(2^{-n})$. A very good polynomial to use from the point of view of this bound is the Chebyshev polynomial of degree n, shifted to the interval $[-1, 0]$, which gives the bound $O(\lambda^{-n})$ with $\lambda = 3+2\sqrt{2} \doteq 5.828$ [CRZ00]. This formula is Algorithm 1 in [CRZ00].

2. If we know something about the weight function w, it is possible to optimize the polynomials to take advantage of the fact. For example, Cohen, Rodriguez Villegas, and Zagier [CRZ00] (hereafter referred to as CRVZ) derived polynomials that are good to use when w is analytic in certain regions of the complex plane that include the interval $[0, r]$. The resulting method can in the most favorable case be guaranteed to achieve $O(\lambda^{-n})$ with $\lambda \doteq 17.9$. The simplest family of polynomials in this class is defined by the identity

$$A_n(\sin^2\theta) = \frac{d^n(\sin^n\theta\cos^n\theta)}{d\theta^n}. \tag{A.21}$$

The polynomials A_n do not satisfy a three-term recursion and it seems to require $O(n^3)$ operations to generate all the coefficients of $A_0, A_1, \ldots, A_n$. Of course, one could precompute and store those.

3. If we assume that $w(t)/(1-t)$ is $(2n)$ times continuously differentiable, then a promising choice is to take the Legendre polynomials.

4. If $w(t)$ has an $O(t^\beta)$ singularity at $t = 0$, then the Jacobi polynomials suggest themselves. For example, for series of the form

$$\eta(\beta, r) = \frac{1}{\beta} - \frac{r}{1+\beta} + \frac{r^2}{2+\beta} - \frac{r^3}{3+\beta} + \cdots, \tag{A.22}$$

use the Jacobi polynomials $J^{(0,\beta-1)}$, shifting from the interval $x \in [-1, 1]$ to $t \in [0, r]$ by the transformation $t = r(x+1)/2$.

5. For monotonic series with $r = 1$, this approach has not so far been spectacularly successful. One reason is that, since $1 \in [0, r]$, the crucial factor $\max_{t\in[0,r]} |p_n(t)|/|p_n(1)|$ cannot be made smaller than 1.

Since so many of the above possibilities involve orthogonal polynomials, it is worthwhile to show in detail how these would be applied. Let the required orthogonal polynomials, shifted to the interval $[0, r]$, satisfy the recursion

$$p_0(t) = 1;$$

$$p_1(t) = (t - a_0);$$

$$p_{n+1}(t) = (t - a_n)p_n(t) - b_n p_{n-1}(t), \qquad n = 1, 2, \ldots .$$

Then the extrapolation algorithm is given by

$$\hat{s}_{k,0} = s_{k,0};$$

$$\hat{s}_{k,1} = s_{k+1,0} - a_0 s_{k,0};$$

$$\hat{s}_{k,n+1} = \hat{s}_{k+1,n} - a_n \hat{s}_{k,n} - b_n \hat{s}_{k,n-1}, \quad n = 1, 2, \ldots;$$

$$s_{k,n} = \frac{\hat{s}_{k,n}}{p_n(1)}.$$

One example must suffice here. The series $\eta(\beta, r)$ defined in (A.22) is to be evaluated for $r = 0.94$, $\beta = 0.125$. We do it four times, the last two times using more and more information. In Table A.9, each column is the transposed first row of the extrapolation table. First, the polynomials A_n of CRVZ (A.21), taken over $(0, -1)$; second, the Legendre polynomials shifted to $[0, -1]$; third, the CRVZ polynomials shifted to $[0, -0.94]$; last,

Table A.9. *The CRVZ method applied to $\eta(\beta, r)$ with $r = 0.94$, $\beta = 0.125$.*

k	CRVZ(−1)	Legendre(−1)	CRVZ(−0.94)	Jacobi(−0.94)
1	8.00000000000000	8.00000000000000	8.00000000000000	8.00000000000000
2	7.44296296296296	7.44296296296296	7.43159486016629	7.24346076458753
3	7.40927089580931	7.42063107088989	7.41224315975913	7.41890823284965
4	7.42291549578755	7.42333978080714	7.42286010429028	7.42300312779417
5	7.42312861386679	7.42305165138502	7.42312700068607	7.42310843651716
6	7.42311090693716	7.42311564776528	7.42311289945816	7.42311121107761
7	7.42311207370828	7.42311055990758	7.42311135754321	7.42311128485659
8	7.42311117324569	7.42311127821820	7.42311128787958	7.42311128682738
9	7.42311130057045	7.42311129133954	7.42311128684365	7.42311128688016
10	7.42311128604961	7.42311128388467	7.42311128688337	7.42311128688157
11	7.42311128677714	7.42311128731700	7.42311128688239	7.42311128688161
12	7.42311128692742	7.42311128679323	7.42311128688171	7.42311128688161
13	7.42311128687268	7.42311128688948	7.42311128688162	7.42311128688161
14	7.42311128688278	7.42311128688073	7.42311128688161	7.42311128688161
15	7.42311128688154	7.42311128688153	7.42311128688161	7.42311128688161
16	7.42311128688161	7.42311128688164	7.42311128688161	7.42311128688161

the Jacobi polynomials $J^{(0,-0.875)}$ shifted to $[0, -0.94]$. There is no meaningful difference between the first two methods (apart from the greater convenience of the three-term recursion in the Legendre case), but the third method is distinctly better. Not shown here is Legendre (-0.94), which also shows improvement, but slightly less so than CRVZ. The last column shows very impressive convergence, which even the various nonlinear algorithms to be discussed later cannot match. This example demonstrates the great value of having analytical information about the sequence.

A.4.2 Semilinear Algorithms

This family of algorithms is due to Levin [Lev73]. They are modified Salzer's algorithms with the following auxiliary functions (the labels come from Levin's own notation):

T $\psi(k) = a_k$.

U $\psi(k) = (k + k_0)a_k$.

W $\psi(k) = a_k^2/(a_{k+1} - a_k)$.

Note that we do not know continuous functions $\psi(x)$ for all x. The implementation goes exactly as described for the modified Salzer's algorithm.

The U-algorithm, in particular, is amazingly effective over a large class of sequences. When the T-algorithm works, so does the U-algorithm, although it may need one term more to achieve the same accuracy. Since round-off error cannot be ignored (see §A.5.3), it is better to use the T-algorithm in the cases where both work. The W-algorithm has the disadvantage of always needing one term more than the U-algorithm to form a particular $s_{k,l}$, and it is hard to find cases where the W-algorithm works but the U-algorithm does not.

Unfortunately it is also quite difficult to characterize the sequences for which any of these algorithms is exact. One reason for that is that semilinear algorithms do not possess the linear invariance property (A.6). In particular, we cannot expect that it will be exact for a sum of two functions if it happens to be exact for either separately.

To understand the startling effectiveness of the algorithms, it is useful to think of the analogue with integration: by (A.18), ψ is a plausible choice of auxiliary function when $\psi' = \phi$. Any nonzero multiple of ψ will do just as well, so the three transformations correspond, respectively, to

T $c\psi(x) = \psi'(x)$.

U $c\psi(x) = (x + x_0)\psi'(x)$.

W $c\psi(k) = (\psi'(x))^2/\psi''(x)$.

These differential equations have solutions

T $\psi(x) = e^{cx}$.

U $\psi(x) = (x + x_0)^c$.

W The general solution includes the other two as special cases.

Therefore, we expect the T-algorithm to be effective when $s_k \sim r^k$, the U-algorithm when $s_k \sim k^{-j}$, and the W-algorithm in either case. Actually the U-algorithm is nearly as effective as the T-algorithm when $s_k \sim r^k$, since its model has as its $(j+1)$st auxiliary function what the T-algorithm has as its jth. In practice, it is usually enough to use the U-algorithm all the time and ignore the others, since it is quite hard to find a function that suits the W-algorithm but not the U-algorithm, and in any case the W-algorithm is more susceptible to round-off error.

A.4.3 Strongly Nonlinear Methods

A typical feature of strongly nonlinear methods is that an approximation $s_{k,l}$ that uses precisely the values $s_{k:k+l}$ and no others is not available for all possible combinations of k and l.

Aitken's Method

Although more recent than Euler's transformation, "Aitken's Δ^2-method" is the best-known of all convergence acceleration procedures. It takes three members of the sequence and returns one number. Since it uses the elements s_k, s_{k+1}, s_{k+2} we call that number $s_{k,2}$. The basic formula resembles (A.9):

$$s_{k,2} = s_{k+2} + \frac{r_k}{1 - r_k}(s_{k+2} - s_{k+1}), \tag{A.23}$$

$$\text{where } r_k = \frac{s_{k+2} - s_{k+1}}{s_{k+1} - s_k}. \tag{A.24}$$

The formula for r_k is motivated by the model

$$a_k = cr^k,$$

the same model as for the first column of Euler's transformation but with r regarded as unknown. The nickname derives from another way of writing (A.23):

$$s_{k,2} = s_{k+2} - \frac{(\Delta s_{k+1})^2}{\Delta^2 s_k}. \tag{A.25}$$

Aitken's method is contained in two other methods:

1. Aitken's formula applied to $s_0, s_1, s_2, \ldots, s_n$ gives the same values for $s_{k,2}$ that Levin's T-transform gives for $s_{k,1}$ when applied to $s_1, s_2, \ldots, s_n$.

2. The second colum $s_{k,2}$ of the epsilon algorithm (see the next section) is identical to that of Aitken's method.

It is often effective to apply Aitken's method again on the transformed sequence, etc., but there is then no longer a simple model to tell us when the transformation is exact.

The Epsilon Algorithm

The model for Wynn's epsilon algorithm [Wyn56a] is the same (A.13) as for the modified Euler method, except that the ratios r_j are unknown rather than known. In other words, the epsilon algorithm can deliver the exact limit, using $2n+1$ members of a sequence, whenever the sequence can be written as a sum of n geometric sequences.

It is without doubt the most elegant of all convergence acceleration methods, with a marvellously simple recursion formula,

$$s_{k,j} = s_{k+1,j-2} + \frac{1}{s_{k+1,j-1} - s_{k,j-1}},$$

which contains no extraneous multipliers at all. To start off the recursion, we need $s_{k,-1} = 0$ as well as the usual $s_{k,0} = s_k$. The subscripts hide the pretty geometric picture. If we denote the four entries in the table by geographic letters of the alphabet and use a table in which each column is offset by half a step, we get

$$\begin{pmatrix} & s_{k,j-1} & \\ s_{k+1,j-2} & & s_{k,j} \\ & s_{k+1,j-1} & \end{pmatrix} = \begin{pmatrix} & N & \\ W & & E \\ & S & \end{pmatrix}, \qquad (N-S)(W-E) = 1.$$

The extrapolated values are found in the columns $s_{:,j}$ when j is even. In the case when the even columns converge (the usual case), the odd columns diverge to infinity.

Since only the even-numbered columns matter, a useful alternative formulation [Wyn66] eliminates the odd-numbered columns. Once again we get a pretty picture:

$$\begin{pmatrix} & s_{k,j-2} & \\ s_{k+2,j-4} & s_{k+1,j-2} & s_{k,j} \\ & s_{k+2,j-2} & \end{pmatrix} = \begin{pmatrix} & N & \\ W & C & E \\ & S & \end{pmatrix},$$

$$\frac{1}{C-N} + \frac{1}{C-S} = \frac{1}{C-W} + \frac{1}{C-E}.$$

The column $s_{:,2m}$ of the epsilon algorithm is exact when the sequence $s - s_k$ satisfies a linear difference equation of order m, that is, when constants $c_0, c_1, \ldots, c_m$ exist such that

$$c_0 s_k + c_1 s_{k+1} + \cdots + c_m s_{k+m} \text{ is constant for all } k.$$

It is therefore suitable for the same sequences as the modified Euler transformation, but with two important differences: on the plus side, it is not necessary to know the factors r_j in (A.12), and on the minus side, twice as many terms are required.

The Rho Algorithm

The rho algorithm [Wyn56b] is nearly as simple as the epsilon algorithm, having the model

$$s_{k,j} = s_{k+1,j-2} + \frac{j}{s_{k+1,j-1} - s_{k,j-1}}. \tag{A.26}$$

The extrapolated values appear in the same columns as in the case of the epsilon algorithm.

The column $s_{:,2m}$ of the rho algorithm is exact when

$$s_k = p(k)/q(k), \tag{A.27}$$

where p and q are polynomials of degree not more than m. It is therefore suitable for the same sequences as Salzer's transformation.

The simplicity of (A.26) arises from the hypothesis (A.27). It may sometimes be the case that a better model is

$$s_k = p(x_k)/q(x_k),$$

where the sequence x_k is known. The rho algorithm then becomes

$$s_{k,j} = s_{k+1,j-2} + \frac{x_k - x_{k-j}}{s_{k+1,j-1} - s_{k,j-1}}. \tag{A.28}$$

The Modified Rho Algorithm

Like Salzer's algorithm, the rho algorithm is not very effective when the sequence does not conform to the model of a rational function, but can be modified to certain other functions. The idea is to view the epsilon algorithm as the case $\theta = 0$, and the rho algorithm as the case $\theta = 1$, of the following algorithm:

$$s_{k,j} = s_{k+1,j-2} + \frac{1 + \theta(j-1)}{s_{k+1,j-1} - s_{k,j-1}}. \tag{A.29}$$

If s_k is well modelled by $s - s_k \approx k^{-1/\theta}\psi(k)$, where $\psi(k)$ is a rational function, the modified rho algorithm can be very effective.

The Theta Algorithm

The theta algorithm [Bre71] takes the guesswork out of choosing θ in the modified rho algorithm. To derive it, we write the first two stages of (A.29) as

$$\begin{aligned} s_{k,1} &= \frac{1}{s_{k+1,0} - s_{k,0}}, \\ s_{k,2} &= s_{k+1,0} + \frac{t}{s_{k+1,1} - s_{k,1}}, \end{aligned} \tag{A.30}$$

where $t = 1 + \theta$, and then ask ourselves, Why can't we allow t to depend on k? Since the ultimate in convergence acceleration is to reach a constant sequence, we throw caution overboard and choose t_k such that $s_{k+1,2} = s_{k,2}$ in (A.30), that is,

$$s_{k+1,0} + \frac{t_k}{s_{k+1,1} - s_{k,1}} = s_{k+2,0} + \frac{t_k}{s_{k+2,1} - s_{k+1,1}}.$$

This can be written as

$$t_k = -\frac{\Delta s_{k+1,0}}{\Delta^2 s_{k,1}}.$$

The theta algorithm is then continued to further columns by analogy to the epsilon and rho algorithms as

$$s_{k,2j+1} = s_{k+1,2j-1} + \frac{1}{\Delta s_{k,2j}},$$

$$s_{k,2j+2} = s_{k+1,2j} + \frac{\Delta s_{k+1,2j}\Delta s_{k,2j+1}}{\Delta^2 s_{k,2j}}.$$

The theta algorithm is extremely versatile in the sense that it can accelerate the convergence of a large class of sequences, which is, however, difficult to characterize. In this sense it is reminiscent of the Levin W-algorithm. On the negative side, it uses up $3n + 1$ terms of the original sequence to produce a row of n accelerated values against, respectively, $2n + 1$ terms for the other strongly linear algorithms discussed here, and only $n + 1$ terms for the linear and semilinear methods. It is also more prone to the effects of round-off error.

A.5 Practical Issues

A.5.1 Using a Subsequence

A trivial way of accelerating a sequence is to form a subsequence $t_k = s_{n_k}$, with $1 \leqslant n_1 < n_2 < n_3 < \cdots$. This technique is sometimes a useful preconditioning step before applying a convergence acceleration algorithm.

For example, suppose that in Problem 3 $s_k = \|A_k\|$ is obtained by using MATLAB to find the norm of the $k \times k$ submatrix A_k of the infinite matrix A (see §3.1). The best we can do with Levin's U-algorithm before numerical instability (see §A.5.3) dominates is about seven correct digits, using 16 terms. Using more terms makes matters worse, not better.

Now suppose we still use 16 terms, but start later: $t_k = s_{k+16}$. One would think that since these terms are closer to the limit, a better extrapolated value could be obtained. In fact, we still get about seven digits, but we already get them using 10 terms. Numerical instability sets in earlier. If we still use no term further than s_{32}, but take a larger stride, we do a little better. For example, stride 2 involves working with the subsequence s_{2k}; we obtain nine digits. However, taking strides 3, 4, etc., does not significantly improve the accuracy any further.

The real advantage of a subsequence is achieved, though, when the index sequence n_k grows faster than linearly. For example, if $s - s_k \equiv k^{-1}$ and $n_k = 2^k$, then $s - t_k \equiv 2^{-k}$: sublinear convergence has been turned into linear convergence. Similarly, linear convergence is turned into quadratic convergence. In Problem 3, one can get about 12 digits in floating-point arithmetic using $n_k = 2^{k-1}$, $k = 1, 2, \ldots, 10$. Since the subsequence already converges linearly with $\rho \approx \frac{1}{2}$, there is very little build-up of round-off error, and it is in principle possible to get fairly close to machine accuracy using this technique.

The extrapolated values for this example are given in Table A.10. Since we have been gaining a steady one to two digits per extrapolation step, and round-off error is insignificant, it requires no great leap of faith to accept the value $s = \|A\| \doteq 1.27422415282$, which is correct to 12 digits.

The epsilon algorithm also delivers contest accuracy (10 digits, almost 11) when applied to this sequence t_k (see Table 3.1 in Chapter 3).

Table A.10. *Levin's U-algorithm applied to the sequence* $\|A_{n_k}\|$ *of Problem 3.*

n_k	$t_k = s_{n_k}$	$t_{1,k}$
1	**1.**00000000000000	**1.**00000000000000
2	**1.**18335017655166	**1.**28951567784715
4	**1.**25253739751680	**1.**36301342016060
8	**1.**27004630585408	**1.**26782158984849
16	**1.**27352521545013	**1.**27445564643953
32	**1.**27411814436915	**1.**27422101365494
64	**1.**27420913129766	**1.**27422405834917
128	**1.**27422212003778	**1.**27422416013221
256	**1.**27422388594855	**1.**27422415291307
512	**1.**27422411845808	**1.**27422415282063

A.5.2 Diagnosing the Nature of a Sequence

To estimate ρ, one can form the auxiliary sequence $\rho_k = \lim(s_{k+2} - s_{k+1})/(s_{k+1} - s_k)$, whose limit, if it exists, must equal ρ. It is in general very risky to base opinions on the nature of a slowly convergent sequence by examining a finite number of terms. The one exception arises when ρ is known to have one of a finite set of values, and the only uncertainty is which to take.

In the example of §A.4.1, we get the following values of ρ_k:

0.500487721799137
0.500065583337346
0.500008367557636
0.500001051510722

If we know that the asymptotic expansion of the error contains negative powers of 2, it is easy to decide that $\rho = 1/2$ and not $1/4$ or 1.

It must be stressed that this kind of test is of limited use.

A feature of convergence acceleration methods of which users must be aware is that in the case of a divergent alternating sequence, the algorithm usually delivers an approximation to a so-called *antilimit*. There are situations where the antilimit can be rigorously defined by analytical continuation; for example,

$$1 + r + r^2 + r^3 + \cdots = \frac{1}{1 - r}$$

has a right-hand side that is defined for all $r \neq 1$ and that is the antilimit of the series on the left when $r \leqslant -1$ or $r > 1$.

This behavior can be very useful—an example follows below—but it does mean that divergence cannot be diagnosed. For example, the following sequence was shown to me by

a researcher who wished to know whether it converges to 0:

$$\begin{array}{r}
-1.000000000000000 \\
0.732050807568878 \\
-0.630414938191809 \\
0.576771533743575 \\
-0.543599590009618 \\
0.521053669642427 \\
-0.504731494506494 \\
0.492368058352063 \\
-0.482678712072860 \\
0.474880464055156 \\
-0.468468857699484 \\
0.463104220153723 \\
-0.458549452588645 \\
0.454634026719303
\end{array}$$

On these data, Euler's transformation gives -0.0001244, Levin's T-algorithm -3.144×10^{-17}, the epsilon algorithm 1.116×10^{-10}, and the theta algorithm -4.737×10^{-12}. But 0 is not the the limit of the sequence; it is an antilimit. The sequence is divergent. However, $|s_k|$ is a convergent sequence. Salzer's transformation gives 0.402759395, Levin's U-algorithm 0.4027594, the rho algorithm 0.4027593957, and the theta algorithm 0.4027594. The evidence is overwhelming that the limit is not 0.

As an example of the usefulness of an antilimit, take, for example, $s_k = \sum_{n=0}^{k}(-1)^n n!$. Using 16 terms, Levin's T-algorithm gives 0.59634736, the epsilon algorithm 0.596, and the theta algorithm 0.596347. The series is, in fact, the asymptotic series for $f(z) = \int_z^\infty e^{z-x}x^{-1}\,dx$ in negative powers of x, evaluated at $x = 1$. By other methods we find $f(1) \doteq 0.596347362323194$. This shows that some violently divergent series can be summed, if not to great accuracy, by nonlinear acceleration methods. In fact, inputting a sequence of random deviates uniformly distributed over (0, 1) to the epsilon algorithm is quite likely to produce "accelerated" values clustering around 0.5.

> *Strongly nonlinear acceleration methods are so powerful that they require great care in their handling.*

Here is a cautionary example. We define the two sequences

$$s_k = \sum_{n=1}^{k}(0.95)^n/n,$$

$$t_k = s_k + 19(0.95)^k/k.$$

Since $t_k - s_k$ is a null sequence, both sequences converge to the same limit, namely, $\log 20 \doteq 2.99573227355399$. And indeed, the Levin U-transformation in IEEE double precision approximates that limit to a terminal accuracy of five and seven digits, respectively. We now form $u_k = \sqrt{s_k t_k}$. This sequence obviously has the same limit as the other two. Yet Levin's algorithm, using 30 terms, insists that the limit is 2.9589224422146, with entries 25 to 30 of the first row all within half a digit in the last place of each other. The theta algorithm insists

that the limit is 2.958919941, also in agreement with all these decimals between entries 21, 24, and 27 of the first row.

The sequence u_k is not sufficiently well behaved for extrapolation to work. It is neither monotonic nor alternating. The sequence decreases from over 4 to 2.95887231295868, then increases again. Both algorithms seem to treat u_k as an asymptotic sequence and to produce a spurious pseudolimit close to the place where u_k changes least. What saves us is that although both algorithms seem to produce a limit accurate to 10 digits, the two limits only agree to 6 digits. Linear algorithms like Salzer's do not produce any spurious limits at all.

Whenever possible, use more than one extrapolation method.

In the case of linear extrapolation methods, such as Richardson extrapolation, it is usually possible to prove that the accelerated sequence converges to the same limit as the original sequence. In the case of nonlinear methods, such a proof is seldom available, and when available, requires hypotheses that are unverifiable in practice. And, as we have seen, internal consistency among the entries of one nonlinear acceleration method does not guarantee anything.

In finite-precision arithmetic, "convergence" does not mean that one can in principle get arbitrarily close to the desired limit. It means that sooner or later (in the case of a good extrapolation method, sooner rather than later) a stage is reached where taking into account more terms does not further improve the approximation. Like an asymptotic series, a numerically accelerated sequence in finite precision has a certain terminal accuracy. The factors that determine terminal accuracy are discussed in the following section.

A.5.3 Condition and Stability

We have left this question to the last, but it is in fact an extremely important one. In any numerical computation, there are two main sources of error: truncation error, which arises because only a finite number of terms of an infinite sequence are taken into account; and round-off error, which arises because the computer cannot perform calculations exactly. Usually round-off error is only an important issue when the original sequence is monotonic.

In the case of acceleration methods, we typically start from a sequence in which truncation error is much larger than round-off error. The effect of the extrapolation is to reduce the truncation error, but (particularly in the case of monotonic sequences) to increase the round-off error.

A useful visual tool is to plot the absolute value of the differences in the extrapolated sequence $s_{1,k}$ on a logarithmic scale. For the example of §A.4.1, this graph is shown in Figure A.1. Also on the graph is shown an estimate of the accumulated round-off error (we will explain in a moment how to obtain that). It is conspicuous that the differences decrease steadily, and then start to rise, thereafter closely following the estimated round-off error.

The rule of thumb here is that when a monotonic sequence is accelerated numerically, the sequence thus obtained should be thought of in much the same way as an asymptotic series: the best place to stop is just before the first difference starts to increase. That is the point at which the truncation and round-off errors are both approximately equal to terminal accuracy.

There are also two main sources of round-off error itself. The first arises from the original data, which are obtained by an inexact procedure, such as the rounding of an exact

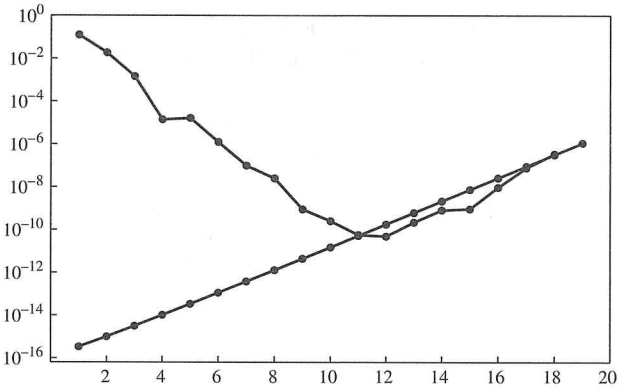

Figure A.1. *The V-shaped line shows $|s_{1,j+1} - s_{1,j}|$ for Salzer's extrapolation applied to the sequence $s_k = \sum_{n=1}^{k} n^{-2}$. The rising line shows a bound for the accumulated round-off error in $|s_{1,j}|$. This graph shows the typical behavior of a linear or semilinear method applied to a monotonic sequence. The two lines meet near the cusp of the V, allowing one to assess confidently the terminal accuracy of the extrapolation.*

value to floating point. The second arises from the further computation in finite-precision arithmetic. In a good implementation, convergence acceleration algorithms are backward stable, which means that the numerics do not introduce any error significantly larger than what would have been caused by perturbing the data at round-off error level. It is therefore sufficient in practice to consider only the propagation of initial round-off error.

All the acceleration algorithms we have considered can be thought of as having the form

$$s_{k,j} = f(s_k, s_{k+1}, \ldots, s_j),$$

although the function is never written out explicitly, but built up by recursion. We define the condition number of the formula by

$$\kappa[f] = \sum_{i=k}^{j} \left| \frac{\partial}{\partial s_i} f(s_k, s_{k+1}, \ldots, s_j) \right|.$$

Roughly speaking, if each s_i is perturbed by round-off error of size μ, then we can expect $s_{k,j}$ to be contaminated by round-off error of size $\mu\kappa[f]$.

In the case of linear transformations, it is easy to estimate the condition number. Let $\kappa_{k,j}$ be the value of $\kappa[f]$ for $s_{k,j}$. Clearly $\kappa_{k,0} = 1$ for all k. From (A.8), we find that

$$\kappa_{k,j} \leqslant \kappa_{k+1,j-1} + |f_{k,j}|(\kappa_{k+1,j-1} + \kappa_{k,j-1}).$$

This estimate is only an inequality, but it is a sharp inequality, and in practice the round-off errors do not tend to cancel out. The estimated round-off error in Figure A.1 was obtained as $2^{-53}\eta_{k,j}$, where

$$\eta_{k,0} = 1,$$

$$\eta_{k,j} = \eta_{k+1,j-1} + |f_{k,j}|(\eta_{k+1,j-1} + \eta_{k,j-1}).$$

We earlier made the remark that from the point of view of accuracy, there is not much to be gained by working with a series rather than a sequence. Even if we had the terms of the series available exactly but the partial sums were still in floating point arithmetic, we would get $\eta_{k,1} = 1$ instead of $\eta_{k,1} = 1 + |f_{k,1}|$. The recursion thereafter proceeds as usual. It is easy to show that the best that can happen is that $\eta_{k,j}$ is reduced by a factor of $(1 + c)$, where $c = \max\{|f_{k,1}|, |f_{k+1,1}|, \ldots, |f_{j-1,1}|\}$.

In the case of the modified Salzer transformation, it is a nuisance to obtain the form (A.8), so we derive a similar procedure for the divided difference formulation. To go with formula (A.16), define

$$\alpha^0 f_k = f_k;$$

$$\alpha^n f_k = \frac{\alpha^{n-1} f_{k+1} + \alpha^{n-1} f_k}{t_{k+n} - t_k};$$

$$\eta_{k,n} = \frac{\alpha^n \left(\frac{1}{\psi(k)}\right)}{\delta^n \left(\frac{1}{\psi(k)}\right)}.$$

For semilinear transformations, one can to a first-order approximation ignore the dependence of the auxiliary sequence on the data. The same procedure as for the modified Salzer transformation remains valid.

The strongly nonlinear transformations are not so easy to analyze. We can no longer, as in the case of the semilinear methods, ignore the dependence of the multipliers $f_{k,j}$ on the data when the methods are written in the form (A.8). This is because the multipliers themselves depend on differences and even second differences of computed quantities, and it is not reasonable to assume that they are accurate enough to be dropped from the analysis.

But in the case of strongly nonlinear methods, this substantial analytical effort is useless. Unlike linear and semilinear methods, bounds on the round-off error tend to be highly pessimistic. The reason for this is that as round-off error starts playing a role, extrapolated values that should in exact arithmetic be very close to each other (since both are close to the limit) turn out to be not that close. The large round-off–contaminated multipliers that might arise because of division by the tiny difference between supposedly close quantities do not occur. The observed behavior for a strongly nonlinear method is that when round-off error and truncation error become comparable, the extrapolated values do not get steadily worse, but tend to vary in a random-looking way around the limit, with amplitude at the accumulated round-off level.

An example will make this clear. Table A.11 shows the rho algorithm in action on the $s_k = \|A_k\|$ values from Problem 3 (see §3.2). The differences of these numbers are graphed in Figure A.2. Note that in both cases at about $j = 10$ the algorithm stops delivering further correct digits, but does not immediately start deteriorating as do the linear and semilinear algorithms; it just mills around without getting much better or much worse. Be warned: the slight tendency for the differences to get smaller does not mean that another digit of accuracy is gained once the initial stage of fast convergence is past. It merely shows that a nonlinear method can produce antilimits even out of random sequences.

Finally, bear in mind that mathematically equivalent formulae are not numerically equivalent. A couple of examples should make this point clear. The formula (A.8) can also

Table A.11. *The rho algorithm applied to the sequence* $\|A_{k_j}\|$ *of Problem 3.*

j	$k_j = j$	$k_j = 5j$
1	**1.**00000000000000	**1.26**121761618336
2	**1.**32196073923600	**1.27**547538244135
3	**1.27**123453514403	**1.274**14260105494
4	**1.27**393314066972	**1.2742**2328312872
5	**1.2742**2017029519	**1.27422**410947073
6	**1.2742**1718653107	**1.27422**414923215
7	**1.2742**2358265440	**1.27422**414886405
8	**1.2742**2398509675	**1.27422415**124899
9	**1.27422**414504466	**1.2742241528**1792
10	**1.27422**416342250	**1.2742241528**1284
11	**1.27422**409923149	**1.2742241526**7127
12	**1.27422**410946623	**1.2742241526**6073
13	**1.27422**412252998	**1.2742241525**8201
14	**1.27422**411949585	**1.2742241526**5168
15	**1.27422**415828802	**1.2742241526**7954
16	**1.27422**413800383	**1.2742241526**6834
17	**1.27422**414441947	**1.2742241526**7514

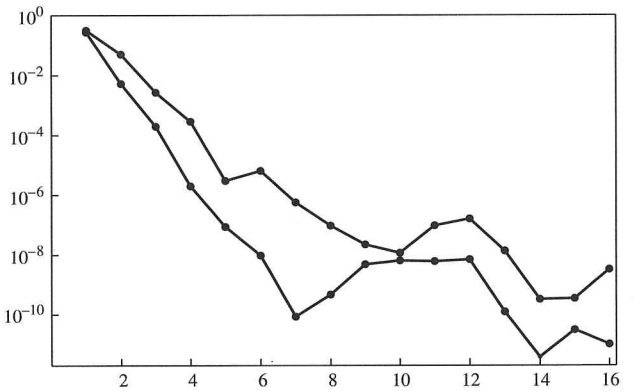

Figure A.2. *The upper line shows* $|s_{1,2j+2} - s_{1,2j}|$ *for the rho algorithm applied to the sequence* $s_k = \|A_k\|$ *arising from Problem 3. The lower line shows the same, but applied to* s_{5k} *instead of* s_k*. This graph exhibits the typical behavior of a strongly nonlinear method applied to a monotonic sequence. There is no V-shape as in the case of a linear or semilinear method, but at the stage where round-off error becomes significant, the magnitude of the differences settles down at the terminal accuracy level.*

be written as

$$s_{k,j} = (1 + f_{k,j})s_{k+1,j-1} - f_{k,j}s_{k,j-1}. \tag{A.31}$$

Suppose that $s_{k,j} \approx 1$, $s_{k+1,j-1} - s_{k,j-1} \approx 0.01$, $f_k \approx 0.1$. Then the correction term in (A.8) is approximately 0.001, so we can afford the loss of three digits in the calculation of f_k. But the correction term in (A.31) is approximately 0.1, so we can afford only the loss of one digit in the calculation of f_k. A more extreme case comes from Aitken's formula: instead of (A.23), (A.24), or (A.25), one might be tempted to use the "more elegant" formula

$$s_{k,2} = \frac{s_k s_{k+2} - s_{k+1}^2}{s_k - 2s_{k+1} + s_{k+2}}.$$

But this would produce a value of $s_{k,2}$ with the same number of correct digits as the second difference of s_k, which clearly is a quantity subject to a lot of cancellation.

A.5.4 Stopping Rules

Let me warn you, right at the outset, that any pure mathematician who feels squeamish about any of the preceding material will recoil in horror at what is about to follow. In fact, I myself feel uncomfortable with some of it.

Consider the following situation: the sequence s_k is so expensive to compute that, at each stage, before computing another term, a decision must be taken as to the necessity or indeed the utility of doing so. For example, one may wish to know s to 10 digits only, and one would like to exit (reporting success) if that accuracy has already been achieved, or to abort (reporting failure) if no further increase of accuracy can be expected from the current algorithm at the current precision.

A stopping rule is an algorithmic procedure by which that decision is automated. We give some examples of such rules, with a strong injunction that it is preferable by far to use the competent human eye.

When accelerating an alternating sequence by any of the methods discussed above, a plausible stopping rule is:

- Check whether $|s_{1,n-1} - s_{1,n-2}| \geqslant |s_{1,n-2} - s_{1,n-3}|$. If so, exit with best extrapolated value $s_{1,n-2}$ and error estimate $c|s_{1,n-1} - s_{1,n-2}|/(1 - c)$, where $c = |s_{1,n-1} - s_{1,n-2}|/|s_{1,n-2} - s_{1,n-3}|$. (This is in fact often a fairly good error estimate even when the terminating condition is not yet satisfied.)

For the linear and semilinear methods, a plausible stopping rule for monotonic sequences is (having just calculated $s_{1,n-1}$ and $\eta_{1,n-1}$):

- Check whether $|\eta_{1,n-1}|\mu \geqslant |s_{1,n-1} - |s_{1,n-2}|$, where μ is the typical error in the s_k values that were used. If so, exit with best extrapolated value $s_{1,n-1}$ and error estimate $|\eta_{1,n-1}|\mu$. If not, apply the same test as for alternating sequences.

For the strongly nonlinear methods, there is no very satisfactory stopping rule for monotonic sequences. My suggestion is to use the same method as for alternating sequences and to multiply the error estimate by 10.

For the human eye, there is a very valuable visual aid. It is this: print a table of the number of digits to which $s_{k,j}$ agrees with $s_{k-1,j}$, that is, the rounded value of $-\log_{10}|1 - s_{k-1}/s_k|$. Such small integers fit nicely in one line and the behavior of the acceleration table is conspicuous.

For example, here is the digit agreement table for the modified Salzer extrapolation example that we have met before:

```
1   2   3   4   5   6   7   7   9   9  10   9   9   9   8   8   7
1   2   4   5   5   7   7   8   9  11   9  10   9   8   8   7   6
2   3   4   5   6   8   8   9  10  10  10   9   8   8   7   6
2   3   5   5   7   8   8  10  10  10   9   8   8   7   6
2   3   5   6   7   8   9  10  11   9   8   8   7   6
2   4   5   6   7   8   9  10   9   9   9   7   7
3   4   6   6   8   9  10   9   9   9   8   7
3   4   6   7   8   9  10   9  10   8   7
3   4   6   7   8   9  10  10   8   7
3   4   6   7   9  10  10   9   8
3   4   7   7   9  10   9   8
3   4   7   7   9   9   9
3   5   7   7   9   9
3   5   7   8   9
3   5   7   8
3   5   7
3   5
3
```

It is obvious that 10 digits, maybe 11, is terminal accuracy here. The ridge of terminal accuracy, flanked on both sides by less accurate values, is typical of linear and semilinear methods. For the modified rho algorithm with $\theta = 2$, on the same s_k values, we get:

```
3   5   7   9  11  11  11  11  12
3   5   8  10  10  10  11  12
4   6   8  10  10  11  10  10
4   6   9  10  11  11  11
4   7   9  10  10  10  11
4   7   9  10  10  10
4   7  10  10  10  11
5   7  10  10  10
5   8  10  10  10
5   8  10  10
5   8  10  11
5   8  10
5   8  10
5   8
5   8
5
6
```

As is typical of nonlinear methods, we observe a plateau at terminal accuracy. Terminal accuracy is also 10, maybe 11 digits; it would be unwise to trust the values that agree to 12 digits.

A.5.5 Conclusion

We have presented many convergence acceleration algorithms, so here is a short guide to which one to use.

1. If you know very precisely what the behavior of the sequence is, use an appropriate linear transformation; for example, if the sequence arises from a finite difference method for differential equations, use Richardson extrapolation or, when that is not applicable, the E-algorithm. One of the ways of solving Problem 10 does just this (see §10.3.2). Problem 8 can also be solved with the aid of a finite difference method, accelerated by Richardson extrapolation (see §8.2).

2. If the sequence is alternating, try the Levin T-transformation; if it is monotonic, try the Levin U-transformation. The T-transformation works very well for the alternating series arising from Problems 1 and 9.

3. If the sequence is alternating, confirm the Levin T result by also trying the epsilon algorithm; if it is monotonic and you know a little about its behavior, confirm the Levin U result by also trying the modified rho algorithm with a suitable value of θ.

4. If all else fails, try the Levin W-transformation and the theta algorithm.

5. If none of the previous is particularly satisfactory, try the effect of applying the transformations to a suitably chosen subsequence, with $k_j = 2^{j-1}$ being the obvious first choice.

6. Irregular sequences and series are not suitable to convergence acceleration. For example, the dependence on the sequence of primes makes the approach of extrapolation from small matrices that works so well in Problem 3 totally unsuitable for Problem 7 (see §7.2.2). Another example with even more erratic behavior is the series

$$\frac{1}{\zeta(s)} = 1 - 2^{-s} - 3^{-s} - 5^{-s} + 6^{-s} - 7^{-s} + \cdots$$

   (the general factor of n^{-s} is $\mu(n)$, the *Möbius function*. This is 0 if n has a multiple prime factor; otherwise it is $(-1)^k$, where k is the number of prime factors of n.)

7. Some sequences and series can be accelerated only with the aid of detailed information which would not be available for a series obtained numerically. For example, series in which a_k can be represented as a product of two slowly convergent null sequences, each by itself well modelled by a simple function of n, but with different asymptotic behavior, can be surprisingly intractable. One such series is $g(x, k_0) = \sum_{k=0}^{\infty} x^k (k + k_0)^{-1}$, which for $k_0 = 0.125$ and $x \approx 0.94$ behaves similarly to a series that arises in the solution to Problem 6 (see §6.3). None of the general methods we have discussed do better than six digits. It is fortunate that this series does not really require convergence acceleration, since for $(k + k_0)(1 - x) > 1$, we have the bound $|s - s_k| < x^k$; thus 600 unaccelerated terms suffice for 16 digits. In this context, $0.94 \ll 1$.

A.6 Notes and References

Although convergence acceleration is mainly an experimental science, there are some theorems (for a selection of those, see [BZ91]) saying that a given class of sequences will be accelerated by a given algorithm (for example, linear sequences are accelerated by the Aitken process).

The use of extrapolation to accelerate the Archimedes sequence is mentioned by Bauer, Rutishauser, and Stiefel [BRS63], who point out that Huygens found a_3' in 1654 and Kommerell found a_2'' in 1936.

The term "quasi-linear" is due to Brezinski [Bre88].

The E-algorithm was independently discovered by no fewer than five authors in the 1970s and has also been called by grander names, such as generalized Mühlbach–Neville–Aitken extrapolation, or the Brezinski–Håvie protocol. For its history, see [Bre00].

The literature on convergence acceleration abounds with names like E-algorithm, g-algorithm, ϵ-algorithm, etc. They usually reflect notation used in an early paper describing the algorithm.

Equation (A.12) can be generalized: if any factor r_j is repeated m times, terms of the form $s_k = r_j^k p(k)$, where p is a polynomial of degree not more than $m-1$, are annihilated.

Although the retrieval of the exact limit when the model (A.14) is satisfied is in general an $O(n^3)$ process, there exist $O(n^2)$ versions of the Richardson process that will transform the sequence to one that, though not constant, does converge faster than any of the modelling functions ϕ_j. Over half of a recent book [Sid03] is devoted to the detailed analysis of many variations of the Richardson process.

In books on difference calculus, the notation δ is usually reserved for centered differences, not for divided differences.

Our description of operator polynomial methods is based on the article by Cohen, Rodriguez, Villegas, and Zagier [CRZ00]. However, the suggestion to use Legendre and Jacobi polynomials is not mentioned there.

The epsilon algorithm has some other marvellous properties: for example, it can accelerate all totally monotonic and totally accelerating sequences, and when applied to a power series, it produces a certain subset of the Padé approximants of the function defined by that power series.

The term "stride" was coined by Carl DeVore in his write-up of Problem 3 [DeV02]. The method he uses for convergence acceleration is not covered here: it is a special-purpose nonlinear extrapolation based on the model

$$s_{k+1} - s_k \approx c_k (k + b_k)^{-d_k}.$$

The parameters c_k, b_k, and d_k are defined by assuming exactness at four consecutive k values. It is easy to eliminate c_k, leading to two nonlinear equations for b_k and d_k. Having obtained the parameters, one exploits the fact that

$$\zeta(d, b) = \sum_{n=1}^{\infty} (n + b)^{-d}$$

is a fairly well-known function (in the Maple implementation called the Hurwitz zeta function). This gives the acceleration formula

$$s_{k,3} = s_k + c_k \left(\zeta(d_k, k + b_k) - \zeta(d_k, b_k) \right).$$

In cases where the two nonlinear equations prove troublesome to solve, DeVore suggested taking $b_k = 0$, in which case three terms suffice to determine c_k and d_k. The acceleration formula becomes

$$s_{k,2} = s_k + c_k \left(\zeta(d_k, k) - \zeta(d_k, 0)\right).$$

DeVore obtained 12 digits of $\|A\|$ in 20-digit arithmetic, using the latter formula on $t_k = s_{3k}$, $k = 1, 2, \ldots, 14$.

For the numerical sequence in §A.5.2, s_k is given by the value of the $(k+1)$-st Laguerre polynomial at its first extremum in $[0, \infty)$. It was shown to me in 1973 by Syvert Nørsett. We both worked on it over a weekend, in my case to find a numerical value for the limit on a mechanical desk calculator, in his to show that the analytical limit of $|s_k|$ equals $|J_0(x_0)|$, where x_0 is the first extremum in $[0, \infty)$ of the Bessel function J_0. Thus the correct value in this case is $s \doteq 0.402759395702553$.

Appendix B

Extreme Digit-Hunting

If you add together our heroic numbers the result is $\tau = 1.497258836\ldots$. *I wonder if anyone will ever compute the ten thousandth digit of this fundamental constant?*
—Lloyd N. Trefethen [Tre02, p. 3]

I am ashamed to tell you to how many figures I carried these computations, having no other business at the time.
—Isaac Newton, after computing 15 digits of π in 1666 [BB87, p. 339]

Since we understood all the relevant algorithms, we tried our best to find 10,000 digits for each of the 10 Challenge problems. We succeeded with 9 of them and summarize our efforts here. In all the even-numbered cases, the task was quite simple. But each odd-numbered problem provided new challenges. As one might expect, a technique that works beautifully to get 10 or 20 digits might well fail badly when trying for hundreds or thousands of digits.

All timings are on a Macintosh G4 laptop with 1 GHz CPU speed, and were done in *Mathematica* unless specified otherwise. The high-precision arithmetic in version 5 of *Mathematica* is especially fast, as it is in PARI/GP, which was used in some cases. Code for each of these is on the web page for this book. The problems divide naturally into two groups, according to whether their solution requires a lot of arithmetic at 10,000-digit precision, or not. Problems 1, 3, and 7 do require many high-precision operations. The others do not. Thus, even though Problem 5 requires much more time than Problem 9, most of the time is spent on initializing the gamma-function computations; once that is done, the rest can be done quickly.

While such an exercise might seem frivolous, the fact is that we learned a lot from the continual refinement of our algorithms to work efficiently at ultrahigh precision. The reward is a deeper understanding of the theory, and often a better algorithm for low-precision cases. A noteworthy example is Problem 9, which first required over 20 hours using a complicated series for Meijer's G function that involved the gamma function. But we were able to eliminate the very expensive gamma function from the series, and then differentiate the result symbolically, with the result that our current best time is now 34 minutes.

As always, the question of correctness arises. We have interval validations of correctness of 10,000 digits for Problems 2, 4, 6, 7, 8, and 10. For the others, we rely on the traditional techniques: comparing different algorithms, machines, software, programmers, and programming languages. And we work up, always comparing digits with past computations. Thus we have ample evidence to believe in the correctness of the 10,002 digits given here for each of nine of the challenge problems.

Problem 1. Using the method discussed in §1.4, this can be done using the trapezoidal rule with 91,356 terms to get the real part of $\int_C z^{i/z-1}\,dz$, where C is the contour given by

$$z = \frac{\pi e^t}{\pi e^t + 2 - 2t} + \frac{2i}{\cosh t},$$

and with t running from -8.031 to 10.22. The reason this simple method and small interval of integration work to such high precision is the double exponential decay of the integrand in combination with an exponential convergence rate of the trapezoidal rule. Still, it takes 22.6 hours and involves over 2 million operations at the full 10,000-digit precision. This computation was done independently by two of us, using different languages (*Mathematica*, Maple, and C++), different operating systems and computers, and different algorithms (the Ooura–Mori method).

```
0.32336 74316 77778 76139 93700 <<9950 digits>> 42382 81998 70848 26513 96587 27
```

Problem 2. Simply using high-enough precision on the initial conditions is adequate. In a fixed-precision environment one would do various experiments to estimate k so that $d + k$ digits of precision in the starting values will give d digits of the answer. But it is elegant, fast, and more rigorous to use the interval algorithm of §2.3, which will provide validated digits. It takes only two seconds to get the answer, starting with an interval of diameter 10^{-10038} around the initial conditions. This time excludes the experiments necessary to learn that 38 extra digits will be adequate.

```
0.99526 29194 43354 16089 03118 <<9950 digits>> 39629 06470 50526 05910 39115 30
```

Problem 3. For this problem we have only 273 digits, obtained in a month-long run by the method of §3.6.2. We chose the contour (3.29) with parameter $\sigma = 3/4$, the parametrization

$$t = \Phi(\tau) = \sinh\left(2\sinh\left(\frac{1}{2}\operatorname{arcsinh}(\tau)\right)\right) = \sinh\left(\frac{\tau}{\sqrt{(1+\sqrt{1+\tau^2})/2}}\right),$$

the truncation point

$$T = \frac{1}{2}\left(\log\log\left(\frac{8}{\epsilon}\right)\right)^2, \quad \epsilon = 10^{-275},$$

and the reciprocal step size $1/h = 516.2747$. In analogy to (3.32) an empirical model of the accuracy was fitted to runs in lower precisions. Here, with d denoting the number of correct

digits, such a model takes the form

$$h^{-1} \approx 0.2365 \left(\frac{d}{\log d} - 2.014 \right)^2 \quad \text{for } d > 120$$

and predicts the correctness of 273 digits.

```
1.2742 24152 82122 81882 12340 <<220 digits>> 75880 55894 38735 33138 75269 029
```

Problem 4. Once the location of the minimum is obtained to a few digits, one can use Newton's method on the gradient to zero in on the corresponding critical point of f. It is only necessary to get the critical point to 5000 digits. Then use 10,000 digits of precision to evaluate the function at the critical point. This takes 8 seconds, where Newton's method is used in a way that increases the precision at each step: start with machine precision and then repeatedly double the working precision, using each result as the seed for the next. One can validate the answer in 44 seconds using the ϵ-inflation method of §4.6. Just inflate the high-precision critical point to a square of side 10^{-3} and verify Krawczyk's condition.

```
-3.3068 68647 47523 72800 76113 <<9950 digits>> 46888 47570 73423 31049 31628 61
```

Problem 5. Using the six nonlinear equations (5.2) to get the values of θ_1 and θ_2 to 5000 digits, and then maximizing $\epsilon(\theta_1, \theta_2)$, as discussed in §5.6, to 10,000 digits works nicely. It takes six hours for an initial gamma function computation to 10,000 digits. Once that is done, subsequent gamma values are very fast. Then it takes just an hour to solve the six-equation system, and only four minutes for the ϵ evaluation. Using only the six-dimensional system takes much longer, but yields the same result. And a completely independent computation using only the approach of finding a maximum to ϵ also yielded the same 10,000 digits.

```
0.21433 52345 90459 63946 15264 <<9950 digits>> 68023 90106 82332 94081 32745 91
```

Problem 6. Because of the simple form (6.1) of the answer, namely,

$$\epsilon = \frac{1}{4}\sqrt{1-\eta^2}, \quad \text{where } M\left(\sqrt{4-(\eta-1)^2}, \sqrt{4-(\eta+1)^2}\right) = 1,$$

with M being the arithmetic-geometric mean, 10,000 digits can be obtained very quickly, using fast algorithms for the arithmetic-geometric mean and the secant method to find η. The total time needed is about half a second. The result can be quickly validated using interval methods as explained in §6.5.1.

```
0.061913 95447 39909 42848 17521 <<9950 digits>> 92584 84417 28628 87590 08473 83
```

Problem 7. Using the rational form of the answer obtained by Dumas, Turner, and Wan yields 10,000 digits in no time, but sidesteps the many days needed to obtain that exact rational (see §7.5). Thus we carried out a purely numerical approach, using the conjugate gradient method with preconditioning to compute the entire first column of the inverse. This required maintaining a list of 20,000 10,010-digit reals, and took 129 hours and about $4 \cdot 10^9$

arithmetic operations at 10,010-digit precision. The result can be validated by combining knowledge of the relationship of the error to the size of the residual, which can be computed by interval methods as discussed in §7.4.2.

```
0.72507 83462 68401 16746 86877 <<9950 digits>> 52341 88088 44659 32425 66583 88
```

Problem 8. The method of equation (8.8) yields the answer by simple root-finding. The answer, t, satisfies $\theta(e^{-\pi^2 t}) = \pi/2\sqrt{5}$. The series defining θ can safely be truncated after 73 terms. Hence, the derivative of this is easy and Newton's method can be used yielding 10,000 digits in eight seconds. *Mathematica*'s `FindRoot` is efficient in the sense that, when started with a machine precision seed, it will start its work at machine precision and work up, doubling the precision at each iteration. The result can be quickly validated by the ϵ-inflation method of interval arithmetic as explained in §8.3.2.

```
0.42401 13870 33688 36379 74336 <<9950 digits>> 34539 79377 25453 79848 39522 53
```

Problem 9. The integral $I(\alpha)$ can be represented in terms of the gamma function and Meijer's G function (see §9.6), and this leads to a representation using an infinite series. Moreover, the series can be differentiated term by term, allowing us to approximate $I'(\alpha)$ as a partial sum of an infinite series. Thus we need only find the correct solution to $I'(\alpha) = 0$, which can be done by the secant method. Formula (9.13) shows how the gamma function can be eliminated, which saves a lot of time (see §5.7). That formula also yields to term-by-term differentiation, with the result that 10,000 digits can be obtained in 34 minutes. The differentiation can also be done numerically, but then one must work to 12,500-digit precision, which slows things down. The two methods, run on different machines and with different software (*Mathematica* and PARI/GP), lead to the same result. Using interval methods (bisection on an interval version of $I'(\alpha)$) validated the first 1000 digits.

```
0.78593 36743 50371 45456 52439 <<9950 digits>> 63138 27146 32604 77167 80805 93
```

Problem 10. Given the symbolic work in §10.7 that presents the answer as

$$\frac{2}{\pi}\arcsin\left((3-2\sqrt{2})^2(2+\sqrt{5})^2(\sqrt{10}-3)^2(5^{1/4}-\sqrt{2})^4\right),$$

this is the easiest of all: we simply compute π, three square roots, and an arcsine to 10,010 digits. This takes 0.4 seconds. An interval computation then validates correctness.

```
0.00000038375 87979 25122 61034 07133 <<9950 digits>> 65284 03815 91694 68620 19870 94
```

Appendix C
Code

We may say most aptly, that the Analytical Engine weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves.

— Augusta Ada Byron King, Countess of Lovelace
[Men43, Note A, p. 696]

Here we collect, for the convenience of the reader, all the small code that was used in the chapters but not displayed there in order not to distract from the flow of the arguments. This code, and more elaborate versions with all the bells and whistles, can be downloaded from the web page for this book:

`www.siam.org/books/100digitchallenge`

C.1 C

Monte Carlo Simulation of Brownian Motion

This small C program was used to obtain Table 10.1 in §10.1.

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

void main()                          /* walk on spheres */
{
    const float pi2 = 8.*atan(1.)/RAND_MAX;
    const float a = 10., b = 1., h = 1.e-2; /* geometry */
    const int n = 1e8;              /* number of samples */
    float x,y,r,phi,p;
    int k,count,hit;

    count = 0;
    for (k=0; k<n; k++) {            /* statistics loop */
        x = 0.; y = 0.;
```

```
        do {                                  /* a single run */
            r = min(min(a-x,a+x),min(b-y,b+y));
            phi = pi2*rand();
            x += r*cos(phi); y += r*sin(phi);
        } while ((fabs(x)<a-h) & (hit=(fabs(y)<b-h)));
        count += hit;
    }
    p = ((float)count)/n;
    printf("%e \n",p);
}
```

C.2 PARI/GP

PARI/GP is an interactive programming environment that was originally developed by Karim Belabas and Henri Cohen for algorithmic number theory. It has many powerful features of symbolic algebra and of high-precision arithmetic, including basic linear algebra. PARI/GP (we used version 2.1.3) is free for private or academic purposes and can be downloaded from

`http://pari.math.u-bordeaux.fr/download.html`

C.2.1 Problem-Dependent Functions and Routines

Operator Norm with Power Method (§3.3)

Used in session on p. 55.

```
{OperatorNorm(x,tol) =
    n=length(x); b=vector(2*n,l, 1+(l^2-l)/2);
    lambda=0; lambda0=1;
    while(abs(lambda-lambda0)>tol,
        xhat=x/sqrt(sum(k=-n,-1, x[-k]^2));
        y=vector(n,k, sum(l=-n,-1, xhat[-l]/(b[k-l]+l)));
        x=vector(n,j, sum(k=-n,-1,  y[-k]/(b[j-k]-j)));
        lambda0=lambda; lambda=sum(k=-n,-1,xhat[-k]*x[-k]);
    );
    [sqrt(lambda),x]
}
```

C.3 MATLAB

All the MATLAB code was tested to run under version 6.5 (R13).

C.3.1 Problem-Dependent Functions and Routines

Operator Norm with Power Method (§3.3)

MATLAB version of the PARI/GP procedure used in session on p. 55.

```
function [s,v,u] = OperatorNorm(x,tol)
```

```
% Input:    x       initial right singular vector
%           tol     desired absolute tolerance
% Output:   s       maximal singular value (operator norm)
%           v       right singular vector
%           u       left  singular vector

n = length(x); y = zeros(n,1);
k = 1:n;
lambda = 0; lambda0 = 1;
while abs(lambda-lambda0) > tol
    xhat = x/norm(x);
    for j=1:n, y(j)=(1./((j+k-1).*(j+k)/2-(k-1)))*xhat; end
    for j=1:n, x(j)=(1./((k+j-1).*(k+j)/2-(j-1)))*y;    end
    lambda0 = lambda; lambda   = x'*xhat;
end
s = sqrt(lambda);
v = x/norm(x);
u = y/norm(y);

return
```

Return Probability (§6.2)

Used in session on p. 124.

```
function p = ReturnProbability(epsilon,n)

pE = 1/4 + epsilon; pW = 1/4 - epsilon; pN = 1/4; pS = 1/4;

m = 2*n+1; ctr = sub2ind([m,m],n+1,n+1);
A_EW = spdiags(ones(m,1)*[pW pE],[-1 1],m,m);
A_NS = spdiags(ones(m,1)*[pS pN],[-1 1],m,m);
A = kron(A_EW,speye(m)) + kron(speye(m),A_NS);
r = A(:,ctr); A(:,ctr) = 0; q = (speye(m^2)-A)\r;
p = q(ctr);

return
```

Occupation Probabilities (§6.3.1)

Used in session on p. 128 and the function `ExpectedVisitsExtrapolated`.

```
function p = OccupationProbability(epsilon,K)

global pE pW pN pS;
pE = 1/4 + epsilon; pW = 1/4 - epsilon; pN = 1/4; pS = 1/4;

p = zeros(K,1); p(1) = 1; Pi = 1;
for k=1:K-1
    Pi = step(step(Pi)); p(k+1) = Pi(k+1,k+1);
end

return
```

```
function PiNew = step(Pi)

global pE pW pN pS;
[k,k] = size(Pi); PiNew = zeros(k+1,k+1); i=1:k;
PiNew(i+1,i+1) =                    pE*Pi;
PiNew(i  ,i  ) = PiNew(i  ,i  ) + pW*Pi;
PiNew(i  ,i+1) = PiNew(i  ,i+1) + pN*Pi;
PiNew(i+1,i  ) = PiNew(i+1,i  ) + pS*Pi;

return
```

Extrapolation of the Expected Number of Visits (§6.3.2)

Used in session on p. 129.

```
function val = ExpectedVisistsExtrapolated(epsilon,K,extraTerms)

p = OccupationProbability(epsilon,K+extraTerms);
steps = (extraTerms-1)/2;
val = sum(p(1:K))+WynnEpsilon(p(end-2*steps:end),steps,'series','off');

return
```

C.3.2 General Functions and Routines

Strebel's Summation Formula

Used in session on p. 64.

```
function [w,c] = SummationFormula(n,alpha)

% Calculates weights w and nodes c for a summation formula that
% approximates sum(f(k),k=1..infinity) by sum(w(k)*f(c(k)),k=1..n).
% Works well if f(k) is asymptotic to k^(-alpha) for large k.
% Put alpha = 'exp' to use an exponential formula.

switch alpha
    case 'exp'
        k=n:-1:2;
        u=(k-1)/n;
        c1 = exp(2./(1-u).^2-1./u.^2/2); c = k + c1;
        w = 1 + c1.*(4./(1-u).^3+1./u.^3)/n;
        kinf = find(c==Inf);
        c(kinf)=[]; w(kinf)=[];
        c = [c 1]; w = [w 1];
    otherwise
        n = ceil(n/2); a1 = (alpha-1)/6; a6 = 1+1/a1;
        k2 = n-1:-1:0;
        w2 = n^a6./(n-k2).^a6-a6*k2/n;
        c2 = n+a1*(-n+n^a6./(n-k2).^(1/a1))-a6*k2.^2/2/n;
        k1 = n-1:-1:1;
        w = [w2 ones(size(k1))];
        c = [c2          k1  ];
end

return
```

Trapezoidal Sums with sinh-Transformations

Used in session on p. 71.

```
function [s,steps] = TrapezoidalSum(f,h,tol,level,even,varargin)

% [s,steps] = TrapezoidalSum(f,h,tol,level,even,varargin)
%
% applies the truncated trapezoidal rule with sinh-transformation
%
%  f            integrand, evaluates as f(t,varargin)
%  h            step size
%  tol          truncation tolerance
%  level        number of recursive applications of sinh-transformation
%  even         put 'yes' if integrand is even
%  varargin     additional arguments for f
%
%  s            value of the integral
%  steps        number of terms in the truncation trapezoidal rule

[sr,kr] = TrapezoidalSum_(f,h,tol,level,varargin{:});
if isequal(even,'yes')
    sl = sr; kl = kr;
else
    [sl, kl] = TrapezoidalSum_(f,-h,tol,level,varargin{:});
end
s = sl+sr; steps = kl+kr+1;

return

function [s,k] = TrapezoidalSum_(f,h,tol,level,varargin)

t = 0; F0 = TransformedF(f,t,level,varargin{:})/2;
val = [F0]; F = 1; k = 0;
while abs(F) >= tol
    t = t+h; F = TransformedF(f,t,level,varargin{:});
    k = k+1; val = [F val];
end
s = abs(h)*sum(val);

return

function val = TransformedF(f,t,level,varargin)

dt = 1;
for j=1:level
    dt = cosh(t)*dt; t = sinh(t);
end
val = feval(f,t,varargin{:})*dt;

return
```

Wynn's Epsilon Algorithm

Used in function `ExpectedVisistsExtrapolated`.

```
function val = WynnEpsilon(a,steps,object,display)

% function S = WynnEpsilon(a,steps,object,display)
%
% extrapolates a sequence or series by using Wynn's epsilon algorithm
%
%   a          vector of terms
%   steps      number of extrapolation steps
%   object     'sequence': extrapolation to the limit of a
%              'series'  : extrapolation to the limit of cumsum(a)
%   display    'on':   plot the absolute value of differences in the
%              first row of the extrapolation table for diagnosis
%
%   val        extrapolated value

S = zeros(length(a),2*steps+1);
switch object
    case 'sequence'
        S(:,1) = a;
    case 'series'
        S(:,1) = cumsum(a);
    otherwise
        error('MATLAB:badopt','%s: no such object known',object);
end
for j=2:2*steps+1
    if j==2
        switch object
            case 'sequence'
                S(1:end-j+1,j) = 1./diff(a);
            case 'series'
                S(1:end-j+1,j) = 1./a(2:end);
        end
    else
        S(1:end-j+1,j) = S(2:end-j+2,j-2)+1./diff(S(1:end-j+2,j-1));
    end
end
S = S(:,1:2:end);
if isequal(display,'on')
    h=semilogy(0:length(S(1,:))-2,abs(diff(S(1,:))));
    set(h,'LineWidth',3);
end
val = S(1,end);

return
```

Richardson Extrapolation with Error Control

Used in sessions on pp. 173, 174, and 208.

```
function [val,err,ampl] = richardson(tol,p,nmin,f,varargin)
```

```
% [val,err,ampl] = richardson(tol,p,nmin,f,varargin)
%
% richardson extrapolation with error control
%
%   tol     requested relative error
%   p       f(...,h) has an asymptotic expansion in h^p
%   nmin    start double harmonic sequence at h=1/2/nmin
%   f       function, evaluates as f(varargin,h)
%
%   val         extrapolated value (h -> 0)
%   err         estimated relative error
%   ampl        amplification of relative error in the
%                   evaluation of f

% initialize tableaux
j_max = 9; T = zeros(j_max,j_max);
n = 2*(nmin:(j_max+nmin));     % double harmonic sequence
j=1; h=1/n(1);
T(1,1) = feval(f,varargin{:},h);
val=T(1,1); err=abs(val);

% do Richardson until convergence
while err(end)/abs(val) > tol & j<j_max
    j=j+1; h=1/n(j);
    T(j,1) = feval(f,varargin{:},h);
    A(j,1) = abs(T(j,1));
    for k=2:j
       T(j,k)=T(j,k-1)+(T(j,k-1)-T(j-1,k-1))/((n(j)/n(j-k+1))^p-1);
       A(j,k)=A(j,k-1)+(A(j,k-1)+A(j-1,k-1))/((n(j)/n(j-k+1))^p-1);
    end
    val = T(j,j);
    err = abs(diff(T(j,1:j)));   % subdiagonal error estimates
    if j > 2            % extrapolate error estimate once more
        err(end+1) = err(end)^2/err(end-1);
    end
end
ampl = A(end,end)/abs(val);
err = max(err(end)/abs(val),ampl*eps);

return
```

Approximation of the Heat Equation on Rectangles

Used in session on p. 173.

```
function u_val = heat(pos,rectangle,T,u0,f,bdry,h);

% u_val = heat(pos,rectangle,T,u0,f,bdry,h)
%
% solves the heat equation u_t - \Delta u(x) = f with dirichlet
% boundary conditions on rectangle [-a,a] x [-b,b] using a five-point
% stencil and explicit Euler time-stepping
%
```

```
%   pos          [x,y] a point of the rectangle
%   rectangle    [a,b]
%                2a   length in x-direction
%                2b   length in y-direction
%   T            final time
%   u0           intial value                          (constant)
%   f            right-hand side                       (constant)
%   bdry         Dirichlet data [xl,xr,yl,yu]
%                xl  boundary value on {-a} x [-b,b]   (constant)
%                xr  boundary value on { a} x [-b,b]   (constant)
%                yl  boundary value on [-a,a] x {-b}   (constant)
%                yu  boundary value on [-a,a] x { b}   (constant)
%   h   h = 1/n suggested approximate grid-size
%   u_val        solution at pos = [x,y]

% the grid
a = rectangle(1); b = rectangle(2);
n = 2*ceil(1/2/h); % make n even
nx = ceil(2*a)*n+1; ny = ceil(2*b)*n+1;
dx = 2*a/(nx-1); dy = 2*b/(ny-1);
nt = ceil(4*T)*ceil(1/h^2); dt = T/nt;
x = 1:nx; y=1:ny;

% initial and boundary values
u = u0*ones(nx,ny);
u(x(1),y) = bdry(1); u(x(end),y) = bdry(2);
u(x,y(1)) = bdry(3); u(x,y(end)) = bdry(4);

% the five-point stencil
stencil = [-1/dy^2 -1/dx^2 2*(1/dx^2+1/dy^2) -1/dx^2 -1/dy^2];

% the time-stepping
x = x(2:end-1); y = y(2:end-1);
for k=1:nt
    u(x,y) = u(x,y) + dt*(f - stencil(1)*u(x,y-1)...
                           - stencil(2)*u(x-1,y)...
                           - stencil(3)*u(x,y)...
                           - stencil(4)*u(x+1,y)...
                           - stencil(5)*u(x,y+1));
end

% the solution
u_val = u(1+round((pos(1)+a)/dx),1+round((pos(2)+b)/dy));

return
```

Poisson Solver on Rectangle with Constant Data

Used in sessions on pp. 205 and 206.

```
function u_val = poisson(pos,rectangle,f,bdry,solver,h);

% u_val = poisson(pos,rectangle,f,bdry,solver,h)
%
```

```
% solves poisson equation with dirichlet boundary conditions
% on rectangle [-a,a] x [-b,b] using a five-point stencil
%
%   pos         [x,y] a point of the rectangle
%   rectangle   [a,b]
%               2a   length in x-direction
%               2b   length in y-direction
%   f           right-hand side of -\Delta u(x) = f   (constant)
%   bdry        Dirichlet data [xl,xr,yl,yu]
%               xl  boundary value on {-a} x [-b,b]   (constant)
%               xr  boundary value on { a} x [-b,b]   (constant)
%               yl  boundary value on [-a,a] x {-b}   (constant)
%               yu  boundary value on [-a,a] x { b}   (constant)
%   solver      'Cholesky'  sparse Cholesky solver
%               'FFT'       FFT based fast solver
%   h           h discretization parameter
%
%   u_val       solution at pos = [x,y]

% the grid
n = ceil(1/h);
a = rectangle(1); b = rectangle(2);
nx = 2*ceil(a)*n-1; ny = 2*ceil(b)*n-1; x=1:nx; y=1:ny;
dx = (2*a)/(nx+1); dy = (2*b)/(ny+1);

% the right hand side
r = f*ones(nx,ny);
r(1,y) = r(1,y)+bdry(1)/dx^2; r(nx,y) = r(nx,y)+bdry(2)/dx^2;
r(x,1) = r(x,1)+bdry(3)/dy^2; r(x,ny) = r(x,ny)+bdry(4)/dy^2;

% solve it
switch solver
    case 'Cholesky' % [Dem87,Sect. 6.3.3]
        Ax = spdiags(ones(nx,1)*[-1 2 -1]/dx^2,-1:1,nx,nx);
        Ay = spdiags(ones(ny,1)*[-1 2 -1]/dy^2,-1:1,ny,ny);
        A  = kron(Ay,speye(nx)) + kron(speye(ny),Ax);
        u = A\r(:); u = reshape(u,nx,ny);
    case 'FFT'      % [Dem87,Sect. 6.7]
        u = dst2(r);
        d = 4*(sin(x'/2*pi/(nx+1)).^2*ones(1,ny)/dx^2+ ...
                        ones(nx,1)*sin(y/2*pi/(ny+1)).^2/dy^2);
        u = u./d;
        u = dst2(u);
    otherwise
        error('MATLAB:badopt','%s: no such solver known',solver);
end

% the solution
u_val = u(round((pos(1)+a)/dx),round((pos(2)+b)/dy));

return

% subroutines for 1D and 2D fast sine transform [Dem97,p.324]

function y = dst(x)
```

```
n = size(x,1); m = size(x,2);
y = [zeros(1,m);x]; y = imag(fft(y,2*n+2));
y = sqrt(2/(n+1))*y(2:n+1,:);
return

function y = dst2(x)
y = dst(dst(x)')';
return
```

C.4 INTLAB

INTLAB is a MATLAB toolbox for self-validating algorithms written by Siegfried Rump [Rum99a, Rum99b]. For portability the toolbox is written entirely in MATLAB, making heavy use of BLAS routines. INTLAB (we used version 4.1.2) is free for private or academic purposes and can be downloaded from

`www.ti3.tu-harburg.de/~rump/intlab/.`

A tutorial can be found in the master's thesis of Hargreaves [Har02].

C.4.1 Utilities

Gradient of `hull` Command

Needed for applying `IntervalNewton` *to the function* `theta`.

Automatic differentiation of the command `'hull'` needs some short code to be put in a file named `'hull.m'` to a subdirectory `'@gradient'` of the working directory:

```
function a = hull(a,b)

a.x  = hull(a.x,b.x);
a.dx = hull(a.dx,b.dx);

return
```

Subdivision of Intervals and Rectangles

Used in the commands `IntervalBisection` *and* `IntervalNewton`.

```
function x = subdivide1D(x)

% subdivides the intervals of a list (row vector) by bisection

x1 = infsup(inf(x),mid(x));
x2 = infsup(mid(x),sup(x));
x = [x1 x2];

return

function x = subdivide2D(x)
```

```
% subdivides the rectangles of a list of
% rectangles (2 x k matrix of intervals)

x1_ = x(1,:); x2_ = x(2,:);
l1 = infsup(inf(x1_),mid(x1_));
r1 = infsup(mid(x1_),sup(x1_));
l2 = infsup(inf(x2_),mid(x2_));
r2 = infsup(mid(x2_),sup(x2_));
x = [l1 l1 r1 r1; l2 r2 l2 r2];

return
```

C.4.2 Problem-Dependent Functions and Routines

Objective Function (Chapter 4)

```
function f = fun(x)

f.x       = 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;

f.dx(1,:) = 50*cos(50*x(1,:)).*exp(sin(50*x(1,:)))+ ...
                70*cos(70*sin(x(1,:))).*cos(x(1,:))- ...
                10*cos(10*x(1,:)+10*x(2,:))+1/2*x(1,:);

f.dx(2,:) = 60*cos(60*exp(x(2,:))).*exp(x(2,:))+ ...
                80*cos(sin(80*x(2,:))).*cos(80*x(2,:))- ...
                10*cos(10*x(1,:)+10*x(2,:))+1/2*x(2,:);

return
```

Proof of Rigorous Bound for $\lambda_{\min}(A_{1142})$ (§7.4.2)

The results of the following session are used in the proof of Lemma 7.1.

```
>> n = 1142; p_n = 9209;
>> A = spdiags(primes(p_n)',0,n,n); e = ones(n,2);
>> for k=2.^(0:floor(log2(n))), A = A + spdiags(e,[-k k],n,n); end
>> [V,D] = eig(full(A)); V = intval(V);
>> R = V*D-A*V;
>> for i=1:n, lambda(i) = midrad(D(i,i),norm(R(:,i))/norm(V(:,i))); end
>> [lambda0,j] = min(inf(lambda));
>> lambda_min = infsup(lambda(j))

lambda_min = [  1.120651470854673e+000,  1.120651470922646e+000]

>> lambda0 = intval(1.120651470854673);
>> alpha2 = 11*100; lambda1 = 9221-100;
>> lambdaF = infsup(2*(lambda0*lambda1-alpha2)/...
                      (lambda0+lambda1+sqrt(4*alpha2+(lambda1-lambda0)^2)))

lambdaF = [  1.000037435271862e+000,  1.000037435271866e+000]
```

Interval Theta-Function (§8.3.2)

Used in session on p. 179.

```
function val = theta(q,k)

val = hull(theta_(q,k-1),theta_(q,k));

return

function val = theta_(q,k)

j=0:k; a=(-1).^j./(2*j+1).*q.^(j.*(j+1));
val=2*q^(1/4).*sum(a);

return
```

C.4.3 General Functions and Routines

Two-Dimensional Minimization

As an alternative to using Mathematica in the session on p. 85 one can call:

```
>> [minval,x]=LowestCriticalPoint(@fun,infsup([-1;-1],[1;1]),...
                    infsup(-inf,-3.24),5e-11)

intval minval = -3.306868647_____
x =  -0.02440307969482
      0.21061242715405
```

Implementation of Algorithm 4.2

```
function [minval,x] = LowestCriticalPoint(fun,x,minval,tol)

% [minval,x] = LowestCriticalPoint(fun,x,minval,tol)
%
% solves (interior) global minimization problem on a list
% of rectangles using interval arithmetic.
%
%   fun         objective function. f = fun(x) should give
%               for a (2 x n)-vector of input intervals x
%               a cell structure f.x, f.dx containing the
%               (1 x n)-vector of f-intervals f.x and the
%               (2 x n)-vector of df-intervals f.dx
%
%   x           input: list of rectangles, i.e. (2 x n)-
%               vector of intervals, specifying search region
%               output: midpoint of final enclosing rectangle
%
%   minval      interval enclosing global minimum
%
%   tol         relative tolerance (for minima below 1e-20,
%               absolute tolerance)
```

```
while relerr(minval) > tol
    x = subdivide2D(x);
    f = feval(fun,x);
    upper = min(minval.sup,min(f.x.sup));
    rem = (f.x > upper) | any(not(in(zeros(size(f.dx)),f.dx)));
    f.x(rem) = []; x(:,rem) = [];
    minval = infsup(min(f.x.inf),upper);
end
x = mid(x);

return
```

Interval Bisection

Used in session on p. 138.

```
function x = IntervalBisection(fun,x,tol,varargin)

% x = IntervalBisection(f,x,tol,varargin)
%
% applies the interval bisection method for root-finding
%
%   f           interval function
%   x           at input:  search interval
%               at output: interval enclosing the roots
%   tol         relative error
%   varargin    additional arguments for f

while max(relerr(x))>tol
    x = subdivide1D(x);
    f = x;
    for k=1:length(x)
        f(k) = feval(fun,x(k),varargin{:});
    end
    rem = not(in(zeros(size(f)),f));
    f(rem) = []; x(rem) = [];
end
x=infsup(min(inf(x)),max(sup(x)));

return
```

Interval Newton Iteration

Used in session on p. 179.

```
function X = IntervalNewton(f,X1,varargin)

% X = IntervalNewton(f,X1,varargin)
%
% applies interval Newton method until convergence
%
%   f           interval function, must be enabled for automatic
%                   differentiation, call f(x,varargin)
%   X1          initial interval
```

```
%   varargin    additional arguments for f
%
%   X           converged interval

X = intval(inf(X1));
while X ~= X1
    X = X1;
    x = intval(mid(X));
    F = feval(f,gradientinit(X),varargin{:});
    fx = feval(f,x,varargin{:});
    X1 = intersect(x-fx/F.dx,X);
end

return
```

Interval Arithmetic-Geometric Mean

Used in session on p. 138.

```
function m = AGM(a,b)

rnd = getround;
if isa(a,'double'), a = intval(a); end
if isa(b,'double'), b = intval(b); end
minf = inf(EnclosingAGM(a.inf,b.inf));
msup = sup(EnclosingAGM(a.sup,b.sup));
m = infsup(minf,msup);
setround(rnd);

return

function m = EnclosingAGM(a,b)

a1 = -inf; b1 = inf;
while (a > a1) | (b < b1)
    a1 = a; b1 = b;
    setround(-1); a = sqrt(a1*b1);
    setround( 1); b = (a1+b1)/2;
end
m = infsup(a,b);

return
```

C.5 *Mathematica*

All the *Mathematica* code is for version 5.0.

C.5.1 Utilities

Kronecker Tensor Product of Matrices

Used in the function `ReturnProbability` *on p. 278.*

```
SetAttributes[KroneckerTensor, OneIdentity];
KroneckerTensor[u_?MatrixQ, v_?MatrixQ] :=
 Module[{w = Outer[Times, u, v]}, Partition[
   Flatten[Transpose[w, {1, 3, 2, 4}]], Dimensions[w][[2]] Dimensions[w][[4]]]];
KroneckerTensor[u_, v_, w__] := Fold[KroneckerTensor, u, {v, w}];
CircleTimes = KroneckerTensor;
```

Supporting Interval Functions

Used in various interval functions such as `ReliableTrajectory`, `LowestCriticalPoint`, `IntervalBisection`, *and* `IntervalNewton`.

```
mid[X_] := (Min[X] + Max[X]) / 2;
diam[X_] := Max[X] - Min[X];
diam[{X__Interval}] := Max[diam /@ {X}];
extremes[X_] := {Min[X], Max[X]};
hull[X_] := Interval[extremes[X]];
MidRad[x_, r_] := x + Interval[{-1, 1}] r;
IntervalMin[{Interval[{a_, b_}], Interval[{c_, d_}]}] :=
  Interval[{Min[a, c], Min[b, d]}];
IntervalMin[{}] := ∞;
```

Subdivision of Intervals and Rectangles

Used in various interval functions such as `LowestCriticalPoint`, `IntervalBisection`, *and* `IntervalNewton`.

```
subdivide1D[X_] := Interval /@ {{Min[X], mid[X]}, {mid[X], Max[X]}};
subdivide2D[{X_, Y_}] := Distribute[{subdivide1D[X], subdivide1D[Y]}, List];
```

Count of Digit Agreement and Form for Pretty Interval Output

Used in sessions on pp. 42, 139, 161, 179, and 279.

```
DigitsAgreeCount[a_, b_] := (prec = Ceiling@Min[Precision /@ {a, b}];
   {{ad, ae}, {bd, be}} = RealDigits[#, 10, prec] & /@ {a, b};
   If[ae ≠ be ∨ a b ≤ 0, Return[0]]; If[ad == bd, Return@Length[ad]];
   {{com}} = Position[MapThread[Equal, {ad, bd}], False, 1, 1] - 1; com);
DigitsAgreeCount[Interval[{a_, b_}]] := DigitsAgreeCount[a, b];
IntervalForm[Interval[{a_, b_}]] :=
 (If[(com = DigitsAgreeCount[a, b]) == 0, Return@Interval[{a, b}]];
  start = Sign[a] N[FromDigits@{ad[[Range@com]], 1}, com];
  {low, up} = SequenceForm@@ Take[#, {com + 1, prec}] & /@ {ad, bd};
  If[ae == 0, start /= 10; ae++];
  SequenceForm[DisplayForm@SubsuperscriptBox[NumberForm@start, low, up],
   If[ae ≠ 1, Sequence @@ {" × ", DisplayForm@SuperscriptBox[10, ae - 1]}, ""]])
```

C.5.2 Problem-Dependent Functions and Routines

Reliable Photon Trajectory (§2.3)

Used in session on p. 42.

```
Options[ReliableTrajectory] :=
  {StartIntervalPrecision → Automatic, AccuracyGoal → 12};
ReliableTrajectory[p_, v_, tMax_, opts___Rule] :=
  Module[{error = ∞, ϵ, s0, s, P, V, t, Trem, path, S, T},
   {ϵ, s, s0} = {10.^(-AccuracyGoal), AccuracyGoal, StartIntervalPrecision} /.
      {opts} /. Options[ReliableTrajectory];
   If[s0 === Automatic, s0 = s];
   s = s0; wp = Max[17, s + 2];
   While[error > ϵ,
    P = N[(MidRad[#, 10^-s] &) /@ p, wp];
    V = N[(MidRad[#, 10^-s] &) /@ v, wp];
    path = {P};
    Trem = Interval[tMax];
    While[Trem > 0, M = Round[P + 2 V / 3];
     S = t /. (Solve[(P + t V - M).(P + t V - M) == 1 / 9, t]);
     If[FreeQ[S, Power[Interval[{_?Negative, _?Positive}], _]],
          T = IntervalMin[Cases[S, _?Positive]], Break[]];
     Which[
      T ≤ Trem, P += T V; V = H[P - M].V; Trem -= T,
      T > Trem && Trem ≥ 2 / 3, P += 2 V / 3; Trem -= 2 / 3,
      T > Trem && Trem < 2 / 3, P += Trem V; Trem = 0,
      True, Break[]];
     AppendTo[path, P];
     If[Precision[{Trem, P, V, T}] < ag, Break[]]];
    wp = Max[17, (++s) + 2];
    error = diam[P + Table[MidRad[-Max[Abs[Trem]], Max[Abs[Trem]]], {2}]]];
   Print[StringForm["Initial condition interval radius is 10^^`.", s]];
   path];
```

Return Probability (§6.2)

Mathematica version of the MATLAB function on p. 265.

```
Matrices[n_] := Matrices[n] = (m = 2 n + 1; pE = 1/4 + ϵ0;
    pW = 1/4 - ϵ0; pN = 1/4; pS = 1/4; Id = SparseArray[{i_, i_} → 1, {m, m}];
    PEW = SparseArray[{{i_, j_} /; j == i + 1 → pE, {i_, j_} /; j == i - 1 → pW}, {m, m}];
    PNS = SparseArray[{{i_, j_} /; j == i + 1 → pN, {i_, j_} /; j == i - 1 → pS}, {m, m}];
    {PEW⊗Id + Id⊗PNS, Id⊗Id});
ReturnProbability[ϵ_Real, n_Integer] := ({A, Id} = Matrices[n];
   Block[{ϵ0 = ϵ}, m = 2 n + 1; ctr = n m + n + 1; r = A[[All, ctr]];
    A[[All, ctr]] = 0; q = LinearSolve[Id - A, r]; q[[ctr]]]);
```

Multiplication by the Matrix A_n (§7.3)

Black-box definition of the sparse matrix A_n on p. 149.

```
n = 20000;
BitLength := Developer`BitLength;
indices[i_] := indices[i] =
  i + Join[2^(Range@BitLength[n - i] - 1), -2^(Range@BitLength[i - 1] - 1)]
diagonal = Prime[Range@n];
A[x_] := diagonal * x + MapIndexed[Plus @@ x[[indices[#2[[1]]]]] &, x];
```

Proof of the Rigorous Bound for $\lambda_{\min}(A_{1142})$ (§7.4.2)

The proof of Lemma 7.1 can be based on the following session.

```
n = 1142;
A = SparseArray[{{i_, i_} → Prime[i]}, n] + (# + Transpose[#]) &@
    SparseArray[Flatten@Table[{i, i + 2^j} → 1, {i, n - 1}, {j, 0, Log[2., n - i]}], n];
{λ, V} = Eigensystem[Normal@N[A]];
r = (λMin = λ[[n]]) (x = Interval /@ V[[n, All]]) - x.A;
(λMin += Interval[{-1, 1}] Norm[r] / Norm[x]) // IntervalForm
```

$$1.1206514708^{96148}_{84156}$$

```
λ1 = Prime[n + 1] - 100; α2 = 11 × 100;
(λFMin = 2 ((λ0 = Interval[Min[λMin]]) λ1 - α2) / (λ0 + λ1 + √(4 × α2 + (λ1 - λ0)^2))) // IntervalForm
```

$$1.00003743530134^{8}_{5}$$

Interval Theta Function (§8.3.2)

Used in session on p. 179.

```
SetAttributes[{θ0, θ1}, Listable];
θ0[q_, K_] := 2 q^(1/4) Sum[(-1)^k / (2 k + 1) q^(k (k+1)), {k, 0, K}];
θ1[q_, K_] := q^(-3/4) / 2 Sum[(-1)^k (2 k + 1) q^(k (k+1)), {k, 0, K}];
θ[q_Interval, K_Integer] := hull[θ0[q, {K - 1, K}]];
θ^(1,0)[q_Interval, K_Integer] := hull[θ1[q, {K - 1, K}]];
```

C.5.3 General Functions and Routines

Two-Dimensional Minimization

Used in session on p. 85.

```
LowestCriticalPoint[f_, {x_, a_, b_}, {y_, c_, d_}, upperbound_, tol_] :=
  (rects = N[{ {Interval[{a, b}], Interval[{c, d}]}}];
   fcn[{xx_, yy_}] := f /. {x → xx, y → yy};
   gradf[{xx_, yy_}] := Evaluate[{D[f, x], D[f, y]} /. {x → xx, y → yy}];
   {low, upp} = {-∞, upperbound};
   While[(upp - low) > tol,
    rects = Join @@ subdivide2D /@ rects;
    fvals = fcn /@ rects;
    upp = Min[upp, Min[Max /@ fvals]];
    pos = Flatten[Position[Min /@ fvals, _?(# ≤ upp &)]];
    rects = rects[[pos]]; fvals = fvals[[pos]];
    pos = Flatten[
      Position[Apply[And, IntervalMemberQ[gradf /@ rects, 0], {1}], True]];
    rects = rects[[pos]]; low = Min[fvals[[pos]]]];
   {{low, upp}, Map[mid, rects, {2}]});
```

Interval Arithmetic-Geometric Mean

Used in session on p. 139.

```
AGMStep[{a_, b_}] := {√(a b), 1/2 (a + b)};
EnclosingAGM[{a_Real, b_Real}] :=
  Interval@FixedPoint[extremes@AGMStep[Interval /@ #] &, {a, b}];
Unprotect[ArithmeticGeometricMean];
ArithmeticGeometricMean[A_Interval, B_Interval] :=
  Block[{Experimental`$EqualTolerance = 0, Experimental`$SameQTolerance = 0},
   Interval@extremes[EnclosingAGM /@ {Min /@ #, Max /@ #} &@{A, B}]];
Protect[ArithmeticGeometricMean];
```

Interval Bisection

Used in session on p. 139.

```
IntervalBisection[f_, {a_, b_}, tol_] :=
 (X = Interval[{a, b}]; pos = {1}; While[Max[diam /@ X] > tol,
   pos = Flatten@Position[f /@ (X = Join @@ subdivide1D /@ X),
     _?(IntervalMemberQ[#, 0] &)]; X = X[[pos]]; IntervalUnion @@ X)
```

Interval Newton Iteration

Used in session on p. 179.

```
IntervalNewton[f_, {a_, b_}] := Block[{Experimental`$EqualTolerance = 0},
  X1 = Interval[{a, b}]; X = Interval[a]; While[X =!= X1, X = X1;
   x = Interval[mid[X]]; X1 = IntervalIntersection[x - f[x]/f'[X], X]]; X]
```

Appendix D

More Problems

Whatever the details of the matter, it finds me too absorbed by numerous occupations for me to be able to devote my attention to it immediately.

—John Wallis, upon hearing about a problem posed by Fermat in 1657 [Hav03, p. 92]

While realizing that the solution of problems is one of the lowest forms of Mathematical research, and that, in general, it has no scientific value, yet its educational value can not be over estimated. It is the ladder by which the mind ascends into the higher fields of original research and investigation. Many dormant minds have been aroused into activity through the mastery of a single problem.

—Benjamin Finkel and John Colaw on the first page of the first issue of the *American Mathematical Monthly*, 1894

To help readers experience first-hand the excitement, frustration, and joy of working on a challenging numerical problem, we include here a selection in the same style as Trefethen's 10. Of these 22, the two at the end can be considered research problems in the sense that the proposer does not know even a single digit of the answer.

If you solve one of these and wish to share your solution, we will be happy to receive it. We will post, on the web page of this book, solutions that are submitted to us.

1. What is $\sum_n 1/n$, where n is restricted to those positive integers whose decimal representation does not contain the substring 42? *(Folkmar Bornemann)*

2. What is the sum of the series $\sum_{n=1}^{\infty} 1/f(n)$, where $f(1) = 1$, $f(2) = 2$, and if $n > 2$, $f(n) = nf(d)$, with d the number of base-2 digits of n? *(David Smith)*

Remark. Problem A6 of the 2002 Putnam Competition asked for the integers $b \geqslant 2$ such that the series, when generalized to base-b digits, converges.

3. Let $m(k) = k - k/d(k)$ where $d(k)$ is the smallest prime factor of k. What is

$$\lim_{x\uparrow 1} \frac{1}{1-x} \prod_{k=2}^{\infty} \left(1 - \frac{x^{m(k)}}{k+1}\right) ?$$

(Arnold Knopfmacher)

Remark. This problem arose from the work of Knopfmacher and Warlimont [KW95]. It is a variation of functions that arise in the study of probabilities related to the irreducible factors in polynomials over Galois fields.

4. If N point charges are distributed on the unit sphere, the potential energy is

$$E = \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} |x_j - x_k|^{-1},$$

where $|x_j - x_k|$ is the Euclidean distance between x_j and x_k. Let E_N denote the minimal value of E over all possible configurations of N charges. What is E_{100}? *(Lloyd N. Trefethen)*

5. Riemann's prime counting function is defined as

$$R(x) = \sum_{k=1}^{\infty} \frac{\mu(k)}{k} \operatorname{li}(x^{1/k}),$$

where $\mu(k)$ is the Möbius function, which is $(-1)^{\rho}$ when k is a product of ρ different primes and zero otherwise, and $\operatorname{li}(x) = \int_0^x dt/\log t$ is the logarithmic integral, taken as a principal value in Cauchy's sense. What is the largest positive zero of R? *(Jörg Waldvogel)*

Remark. The answer to this problem is truly shocking.

6. Let A be the 48×48 Toeplitz matrix with -1 on the first subdiagonal, $+1$ on the main diagonal and the first three superdiagonals, and 0 elsewhere, and let $\|\cdot\|$ be the matrix 2-norm. What is $\min_p \|p(A)\|$, where p ranges over all monic polynomials of degree 8? *(Lloyd N. Trefethen)*

7. What is the value of

$$\int_{-1}^{1} \exp\left(x + \sin e^{e^{ex+1/3}}\right) dx \ ?$$

(Lloyd N. Trefethen)

8. What is the value of

$$\int_0^{\infty} x\, J_0(x\sqrt{2}) J_0(x\sqrt{3}) J_0(x\sqrt{5}) J_0(x\sqrt{7}) J_0(x\sqrt{11})\, dx,$$

where J_0 denotes the Bessel function of the first kind of order zero?

(Folkmar Bornemann)

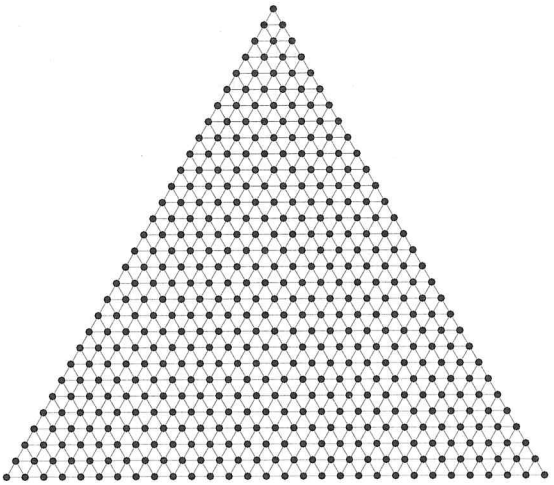

Figure D.1. *A triangular lattice.*

9. If $f(x, y) = e^{-(y+x^3)^2}$ and $g(x, y) = \frac{1}{32}y^2 + e^{\sin y}$, what is the area of the region of the x-y plane in which $f > g$? *(Lloyd N. Trefethen)*

10. The square $c_m(\mathbb{R}^N)^2$ of the least constant in the Sobolev inequality for the domain $\mathbb{R}^N$ is given by the multidimensional integral

$$c_m(\mathbb{R}^N)^2 := (2\pi)^{-N} \int_{\mathbb{R}^N} \left(\sum_{|k| \leqslant m} x^{2k} \right)^{-1} dx,$$

where $k = (k_1, k_2, \ldots, k_N)$ is a multi-index with nonnegative integer elements, and

$$|k| := \sum_{j=1}^{N} k_j, \quad x^k := \prod_{j=1}^{N} x_j^{k_j}.$$

For example, we have $c_3(\mathbb{R}^1) = 0.5$. What is $c_{10}(\mathbb{R}^{10})$? *(Jörg Waldvogel)*

11. A particle starts at the top vertex of the array shown in Figure D.1 with 30 points on each side, and then takes 60 random steps. What is the probability that it ends up in the bottom row? *(Lloyd N. Trefethen)*

12. The random sequence x_n satisfies $x_0 = x_1 = 1$ and the recursion $x_{n+1} = 2x_n \pm x_{n-1}$, where each $\pm$ sign is chosen independently with equal probability. To what value does $|x_n|^{1/n}$ converge for $n \to \infty$ almost surely? *(Folkmar Bornemann)*

13. Six masses of mass 1 are fixed at positions $(2, -1)$, $(2, 0)$, $(2, 1)$, $(3, -1)$, $(3, 0)$, and $(3, 1)$. Another particle of mass 1 starts at $(0, 0)$ with velocity 1 in a direction θ (counterclockwise from the x-axis). It then travels according to Newton's laws, feeling an inverse-square force of magnitude r^{-2} from each of the six fixed masses. What is the shortest time in which the moving particle can be made to reach position $(4, 1)$? *(Lloyd N. Trefethen)*

14. Suppose a particle's movement in the x–y plane is governed by the kinetic energy $T = \frac{1}{2}(\dot{x}^2 + \dot{y}^2)$ and the potential energy

$$U = y + \frac{\epsilon^{-2}}{2}(1 + \alpha x^2)(x^2 + y^2 - 1)^2.$$

The particle starts at the position $(0, 1)$ with the velocity $(1, 1)$. For which parameter α does the particle hit $y = 0$ first at time 10 in the limit $\epsilon \to 0$? *(Folkmar Bornemann)*

15. Let $u = (x, y, z)$ start at $(0, 0, z_0)$ at $t = 0$ with $z_0 \geqslant 0$ and evolve according to the equations

$$\begin{aligned}
\dot{x} &= -x + 10y + \|u\|(-0.7y - 0.03z),\\
\dot{y} &= -y + 10z + \|u\|(0.7y - 0.1z),\\
\dot{z} &= -z + \|u\|(0.03x + 0.1y),
\end{aligned}$$

where $\|u\|^2 = x^2 + y^2 + z^2$. If $\|u(50)\| = 1$, what is z_0? *(Lloyd N. Trefethen)*

16. Consider the Poisson equation $-\Delta u(x) = \exp(\alpha\|x\|^2)$ on a regular pentagon inscribed to the unit circle. On four sides of the pentagon there is the Dirichlet condition $u = 0$, while on one side u satisfies a Neumann condition; that is, the normal derivative of u is zero. For which α does the integral of u along the Neumann side equal e^{α}? *(Folkmar Bornemann)*

17. At what time t_∞ does the solution of the equation $u_t = \Delta u + e^u$ on a 3×3 square with zero boundary and initial data blow up to ∞? *(Lloyd N. Trefethen)*

18. Figure D.2 shows the Daubechies scaling function $\phi_2(x)$ (see [Dau92]) drawn as a curve in the x–y plane. Suppose the heat equation $u_t = u_{xx}$ on the interval $[0, 3]$ is solved with initial data $u(x, 0) = \phi_2(x)$ and boundary conditions $u(0) = u(3) = 0$. At what time does the length of this curve in the x–y plane become 5.4? *(Lloyd N. Trefethen)*

19. Let u be an eigenfunction belonging to the third eigenvalue of the Laplacian with Dirichlet boundary conditions on an L-shaped domain that is made up from three unit squares. What is the length of the zero-level set of u? *(Folkmar Bornemann)*

20. The Koch snowflake is a fractal domain defined as follows. Start with an equilateral triangle with side length 1, and replace the middle third of each side by two sides of an outward-pointing equilateral triangle of side length $1/3$. Now replace the middle third of each of the 12 new sides by two sides of an outward-pointing equilateral triangular of side

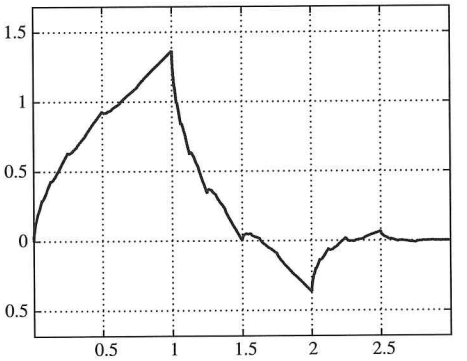

Figure D.2. *Daubechies scaling function* $\phi_2(x)$.

length $1/9$; and so on ad infinitum. What is the smallest eigenvalue of the negative of the Laplacian on the Koch snowflake with zero boundary conditions? *(Lloyd N. Trefethen)*

21. Consider the Poisson equation $-\mathrm{div}(c(x)\mathrm{grad}\, u(x)) = 1$ on the unit square with homogeneous Dirichlet boundary conditions. What is the supremum of the integral of u over the square if $a(x)$ is allowed to vary over all measurable functions that are 1 on half of the area of the square, and 100 on the rest? *(Folkmar Bornemann)*

22. Let $h(z)$ be that solution to the functional equation $\exp(z) = h(h(z))$ which is analytic in a neighborhood of the origin and increasing for real z. What is $h(0)$? What is the radius of convergence of the Maclaurin series of h? *(Dirk Laurie)*

Remark. There are additional properties needed to make h unique. One such simple property has to be found before solving this problem; none is known in the literature right now.

References

[Apo74] Tom M. Apostol, *Mathematical analysis*, second ed., Addison-Wesley, Reading, MA, 1974. (Cited on p. 185.)

[AS84] Milton Abramowitz and Irene A. Stegun (eds.), *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Wiley, New York, 1984. (Cited on pp. 23, 56, 102, 103, 116, 117, 135.)

[AW97] Victor Adamchik and Stan Wagon, *A simple formula for* π, Amer. Math. Monthly **104** (1997), no. 9, 852–855. (Cited on p. viii.)

[Bai00] David H. Bailey, *A compendium of BPP-type formulas for mathematical constants*, manuscript, November 2000, `http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-formulas.pdf`. (Cited on p. viii.)

[Bak90] Alan Baker, *Transcendental number theory*, second ed., Cambridge University Press, Cambridge, UK, 1990. (Cited on p. 219.)

[Bar63] Vic D. Barnett, *Some explicit results for an asymmetric two-dimensional random walk*, Proc. Cambridge Philos. Soc. **59** (1963), 451–462. (Cited on pp. 13, 130, 137.)

[BB87] Jonathan M. Borwein and Peter B. Borwein, *Pi and the AGM. A study in analytic number theory and computational complexity*, Wiley, New York, 1987. (Cited on pp. 137, 138, 176, 216, 217, 259.)

[BB04] Jonathan M. Borwein and David H. Bailey, *Mathematics by experiment: Plausible reasoning in the 21st century*, A. K. Peters, Wellesley, 2004. (Cited on p. viii.)

[BBG04] Jonathan M. Borwein, David H. Bailey, and Roland Girgensohn, *Experimentation in mathematics: Computational paths to discovery*, A. K. Peters, Wellesley, 2004. (Cited on p. viii.)

[BBP97] David Bailey, Peter Borwein, and Simon Plouffe, *On the rapid computation of various polylogarithmic constants*, Math. Comp. **66** (1997), no. 218, 903–913. (Cited on p. viii.)

[Ber89] Bruce C. Berndt, *Ramanujan's notebooks. Part II*, Springer-Verlag, New York, 1989. (Cited on p. 211.)

[Ber98] ———, *Ramanujan's notebooks. Part V*, Springer-Verlag, New York, 1998. (Cited on pp. 221.)

[Ber01] Michael Berry, *Why are special functions special?*, Physics Today **54** (2001), no. 4, 11–12. (Cited on p. 121.)

[BF71] Paul F. Byrd and Morris D. Friedman, *Handbook of elliptic integrals for engineers and scientists*, 2nd rev. ed., Springer-Verlag, New York, 1971. (Cited on p. 142.)

[BR95] Bruce C. Berndt and Robert A. Rankin, *Ramanujan, Letters and commentary*, American Mathematical Society, Providence, 1995. (Cited on p. 220.)

[Bre71] Claude Brezinski, *Accélération de suites à convergence logarithmique*, C. R. Acad. Sci. Paris Sér. A-B **273** (1971), A727–A730. (Cited on p. 246.)

[Bre76] Richard P. Brent, *Fast multiple-precision evaluation of elementary functions*, J. Assoc. Comput. Mach. **23** (1976), no. 2, 242–251. (Cited on p. 30.)

[Bre88] Claude Brezinski, *Quasi-linear extrapolation processes*, Numerical Mathematics, Singapore 1988, Birkhäuser, Basel, 1988, pp. 61–78. (Cited on p. 257.)

[Bre00] ———, *Convergence acceleration during the 20th century*, J. Comput. Appl. Math. **122** (2000), no. 1–2, 1–21. (Cited on pp. 230, 231, 257.)

[BRS63] Friedrich L. Bauer, Heinz Rutishauser, and Eduard Stiefel, *New aspects in numerical quadrature*, Proc. Sympos. Appl. Math., Vol. XV, Amer. Math. Soc., Providence, R.I., 1963, pp. 199–218. (Cited on pp. 68, 257.)

[BT99] B. Le Bailly and J.-P. Thiran, *Computing complex polynomial Chebyshev approximants on the unit circle by the real Remez algorithm*, SIAM J. Numer. Anal. **36** (1999), no. 6, 1858–1877. (Cited on p. 119.)

[BY93] Folkmar Bornemann and Harry Yserentant, *A basic norm equivalence for the theory of multilevel methods*, Numer. Math. **64** (1993), no. 4, 455–476. (Cited on p. 157.)

[BZ91] Claude Brezinski and Michela Redivo Zaglia, *Extrapolation methods*, North-Holland, Amsterdam, 1991. (Cited on pp. 225, 230, 257.)

[BZ92] Jonathan M. Borwein and I. John Zucker, *Fast evaluation of the gamma function for small rational fractions using complete elliptic integrals of the first kind*, IMA J. Numer. Anal. **12** (1992), no. 4, 519–526. (Cited on p. 143.)

[Cau27] Augustin-Louis Cauchy, *Sur quelques propositions fondamentales du calcul des résidus*, Exerc. Math. **2** (1827), 245–276. (Cited on pp. 211, 212.)

[CDG99] David W. Corne, Marco Dorigo, and Fred Glover (eds.), *New ideas in optimization*, McGraw-Hill, Berkshire, 1999. (Cited on p. 106.)

[CGH+96] Robert M. Corless, Gaston H. Gonnet, David E. G. Hare, David J. Jeffrey, and Donald E. Knuth, *On the Lambert W function*, Adv. Comput. Math. **5** (1996), no. 4, 329–359. (Cited on pp. 23, 24.)

[CM01] Nikolai Chernov and Roberto Markarian, *Introduction to the ergodic theory of chaotic billiards*, Instituto de Matemática y Ciencías Afines (IMCA), Lima, 2001. (Cited on p. 44.)

[Col95] Courtney S. Coleman, CODEE Newsletter (spring 1995), cover. (Cited on p. 91.)

[Cox84] David A. Cox, *The arithmetic-geometric mean of Gauss*, Enseign. Math. (2) **30** (1984), no. 3–4, 275–330. (Cited on p. 216.)

[Cox89] ———, *Primes of the form $x^2 + ny^2$. Fermat, class field theory and complex multiplication*, Wiley, New York, 1989. (Cited on p. 220.)

[CP01] Richard Crandall and Carl Pomerance, *Prime numbers, A computational perspective*, Springer-Verlag, New York, 2001. (Cited on pp. 156, 202.)

[CRZ00] Henri Cohen, Fernando Rodriguez Villegas, and Don Zagier, *Convergence acceleration of alternating series*, Experiment. Math. **9** (2000), no. 1, 3–12. (Cited on pp. 241, 257.)

[Dau92] Ingrid Daubechies, *Ten lectures on wavelets*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992. (Cited on p. 284.)

[DB02] Peter Deuflhard and Folkmar Bornemann, *Scientific computing with ordinary differential equations*, Springer-Verlag, New York, 2002, Translated by Werner C. Rheinboldt. (Cited on pp. 170, 172, 207.)

[Dem97] James W. Demmel, *Applied numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1997. (Cited on pp. 124, 205.)

[DeV02] Carl DeVore, 2002, A Maple worksheet on Trefethen's Problem 3, `http://groups.yahoo.com/group/100digits/files/Tref3.mws`. (Cited on p. 257.)

[DH03] Peter Deuflhard and Andreas Hohmann, *Numerical analysis in modern scientific computing. An introduction*, second ed., Springer-Verlag, New York, 2003. (Cited on pp. 148, 154.)

[Dix82] John D. Dixon, *Exact solution of linear equations using p-adic expansions*, Numer. Math. **40** (1982), no. 1, 137–141. (Cited on p. 167.)

[DJ01] Richard T. Delves and Geoff S. Joyce, *On the Green function for the anisotropic simple cubic lattice*, Ann. Phys. **291** (2001), 71–133. (Cited on p. 144.)

[DKK91] Eusebius Doedel, Herbert B. Keller, and Jean-Pierre Kernévez, *Numerical analysis and control of bifurcation problems. I. Bifurcation in finite dimensions*, Internat. J. Bifur. Chaos Appl. Sci. Engrg. **1** (1991), no. 3, 493–520. (Cited on p. 88.)

[DR84] Philip J. Davis and Philip Rabinowitz, *Methods of numerical integration*, second ed., Academic Press, Orlando, FL, 1984. (Cited on pp. 68, 70.)

[DR90] John M. DeLaurentis and Louis A. Romero, *A Monte Carlo method for Poisson's equation*, J. Comput. Phys. **90** (1990), no. 1, 123–140. (Cited on pp. 200, 201.)

[DS00] Jack Dongarra and Francis Sullivan, *The top 10 algorithms*, IEEE Computing in Science and Engineering **2** (2000), no. 1, 22–23. (Cited on p. viii.)

[DT02] Tobin A. Driscoll and Lloyd N. Trefethen, *Schwarz–Christoffel mapping*, Cambridge University Press, Cambridge, UK, 2002. (Cited on p. 223.)

[DTW02] Jean-Guillaume Dumas, William Turner, and Zhendong Wan, *Exact solution to large sparse integer linear systems*, Abstract for ECCAD'2002, May 2002. (Cited on p. 164.)

[Dys96] Freeman J. Dyson, *Review of "Nature's Numbers" by Ian Stewart*, Amer. Math. Monthly **103** (1996), 610–612. (Cited on p. 147.)

[EMOT53] Arthur Erdélyi, Wilhelm Magnus, Fritz Oberhettinger, and Francesco G. Tricomi, *Higher transcendental functions. Vols. I, II*, McGraw-Hill, New York, 1953. (Cited on p. 135.)

[Erd56] Arthur Erdélyi, *Asymptotic expansions*, Dover, New York, 1956. (Cited on p. 70.)

[Eva93] Gwynne Evans, *Practical numerical integration*, Wiley, Chichester, UK, 1993. (Cited on p. 18.)

[Fel50] William Feller, *An introduction to probability theory and its applications. Vol. I*, Wiley, New York, 1950. (Cited on pp. 125, 126.)

[FH98] Samuel P. Ferguson and Thomas C. Hales, *A formulation of the Kepler conjecture*, Tech. report, ArXiv Math MG, 1998, `http://arxiv.org/abs/math.MG/9811072`. (Cited on p. 95.)

[FLS63] Richard P. Feynman, Robert B. Leighton, and Matthew Sands, *The Feynman lectures on physics. Vol. 1: Mainly mechanics, radiation, and heat*, Addison-Wesley, Reading, MA, 1963. (Cited on p. 230.)

[Fou78] Joseph Fourier, *The analytical theory of heat*, Cambridge University Press, Cambridge, UK, 1878, Translated by Alexander Freeman. Reprinted by Dover Publications, New York, 1955. French original: "Théorie analytique de la chaleur," Didot, Paris, 1822. (Cited on pp. 169, 174, 175, 176.)

[GH83] John Guckenheimer and Philip Holmes, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, Springer-Verlag, New York, 1983. (Cited on p. 40.)

[GL81] Alan George and Joseph W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981. (Cited on pp. 149, 150, 151.)

[GL96] Gene H. Golub and Charles F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, 1996. (Cited on pp. 49, 51, 53.)

[Goo83] Nicolas D. Goodman, *Reflections on Bishop's philosophy of mathematics*, Math. Intelligencer **5** (1983), no. 3, 61–68. (Cited on p. 147.)

[Grif90] Peter Griffin, *Accelerating beyond the third dimension: Returning to the origin in simple random walk*, Math. Sci., **15** (1990), 24–35. (Cited on p. 146.)

[GW01] Walter Gautschi and Jörg Waldvogel, *Computing the Hilbert transform of the generalized Laguerre and Hermite weight functions*, BIT **41** (2001), no. 3, 490–503. (Cited on p. 70.)

[GZ77] M. Lawrence Glasser and I. John Zucker, *Extended Watson integrals for the cubic lattices*, Proc. Nat. Acad. Sci. U.S.A. **74** (1977), no. 5, 1800–1801. (Cited on p. 143.)

[Hac92] Wolfgang Hackbusch, *Elliptic differential equations. Theory and numerical treatment*, Springer-Verlag, Berlin, 1992. (Cited on p. 205.)

[Had45] Jacques Hadamard, *The psychology of invention in the mathematical field*, Princeton University Press, Princeton, NJ, 1945. (Cited on p. 17.)

[Han92] Eldon Hansen, *Global optimization using interval analysis*, Monographs and Textbooks in Pure and Applied Mathematics, Vol. 165, Marcel Dekker, New York, 1992. (Cited on pp. 84, 94, 95, 98.)

[Har40] Godfrey H. Hardy, *Ramanujan. Twelve lectures on subjects suggested by his life and work*, Cambridge University Press, Cambridge, UK, 1940. (Cited on p. 221.)

[Har02] Gareth I. Hargreaves, *Interval analysis in MATLAB*, Master's thesis, University of Manchester, December 2002, Numerical Analysis Report No. 416. (Cited on p. 272.)

[Hav03] Julian Havil, *Gamma*, Princeton University Press, Princeton, NJ, 2003. (Cited on p. 281.)

[Hen61] Ernst Henze, *Zur Theorie der diskreten unsymmetrischen Irrfahrt*, ZAMM **41** (1961), 1–9. (Cited on pp. 13, 137.)

[Hen64] Peter Henrici, *Elements of numerical analysis*, Wiley, New York, 1964. (Cited on p. 19.)

[Hen74] ———, *Applied and computational complex analysis. Vol. 1: Power series—integration—conformal mapping—location of zeros*, Wiley, New York, 1974. (Cited on pp. 67, 211, 212, 213.)

[Hen77] ———, *Applied and computational complex analysis. Vol. 2: Special functions—integral transforms—asymptotics—continued fractions*, Wiley, New York, 1977. (Cited on pp. 62, 70.)

[Hen86] ———, *Applied and computational complex analysis. Vol. 3: Discrete Fourier analysis—Cauchy integrals—construction of conformal maps—univalent functions*, Wiley, New York, 1986. (Cited on pp. 203, 204, 213, 214, 223.)

[Her83] Joseph Hersch, *On harmonic measures, conformal moduli and some elementary symmetry methods*, J. Analyse Math. **42** (1982/83), 211–228. (Cited on p. 213.)

[Hig96] Nicholas J. Higham, *Accuracy and stability of numerical algorithms,* 2nd ed., Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2002. `http://www.ma.man.ac.uk/~higham/asna` (Cited on pp. 5, 40, 48, 49, 54, 151, 152, 159, 163, 164, 173, 192, 205, 208.)

[HJ85] Roger A. Horn and Charles R. Johnson, *Matrix analysis*, Cambridge University Press, Cambridge, UK, 1985. (Cited on pp. 53, 155, 162, 163, 165.)

[Hof67] Peter Hofmann, *Asymptotic expansions of the discretization error of boundary value problems of the Laplace equation in rectangular domains*, Numer. Math. **9** (1966/1967), 302–322. (Cited on p. 206.)

[Hug95] Barry D. Hughes, *Random walks and random environments. Vol. 1: Random walks*, Oxford University Press, New York, 1995. (Cited on pp. 121, 122, 130, 139, 140, 143.)

[IMT70] Masao Iri, Sigeiti Moriguti, and Yoshimitsu Takasawa, *On a certain quadrature formula (Japanese)*, Kokyuroku Ser. Res. Inst. for Math. Sci. Kyoto Univ. **91** (1970), 82–118; English translation: J. Comput. Appl. Math. 17, 3–20 (1987). (Cited on p. 68.)

[Jac29] Carl Gustav Jacob Jacobi, *Fundamenta nova theoriae functionum ellipticarum*, Bornträger, Regiomontum (Königsbērg), 1829. (Cited on p. 218.)

[JDZ03] Geoff S. Joyce, Richard T. Delves, and I. John Zucker, *Exact evaluation of the Green functions for the anisotropic face-centred and simple cubic lattices*, J. Phys. A: Math. Gen. **36** (2003), 8661–8672. (Cited on p. 144.)

[Joh82] Fritz John, *Partial differential equations*, fourth ed., Springer-Verlag, New York, 1982. (Cited on p. 176.)

[Kea96] R. Baker Kearfott, *Rigorous global search: Continuous problems*, Nonconvex Optimization and Its Applications, Vol. 13, Kluwer Academic Publishers, Dordrecht, 1996. (Cited on pp. 84, 93, 94, 95, 97.)

[Kno56] Konrad Knopp, *Infinite sequences and series*, Dover, New York, 1956. (Cited on p. 135.)

[Knu81] Donald E. Knuth, *The art of computer programming. Vol. 2: Seminumerical algorithms*, second ed., Addison-Wesley, Reading, 1981. (Cited on p. 29.)

[Koe98] Wolfram Koepf, *Hypergeometric summation. An algorithmic approach to summation and special function identities*, Vieweg, Braunschweig, 1998. (Cited on pp. 131, 132.)

[Kol48] Andrey N. Kolmogorov, *A remark on the polynomials of P. L. Čebyšev deviating the least from a given function*, Uspehi Matem. Nauk (N.S.) **3** (1948), no. 1(23), 216–221. (Cited on p. 118.)

[KS91] Erich Kaltofen and B. David Saunders, *On Wiedemann's method of solving sparse linear systems*, Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991), Lecture Notes in Comput. Sci., Vol. 539, Springer, Berlin, 1991, pp. 29–38. (Cited on p. 166.)

[Küh82] Wilhelm O. Kühne, *Huppel en sy maats*, Tafelberg, Kaapstad, 1982. (Cited on p. 101.)

[KW95] Arnold Knopfmacher and Richard Warlimont, *Distinct degree factorizations for polynomials over a finite field*, Trans. Amer. Math. Soc. **347** (1995), no. 6, 2235–2243. (Cited on p. 282.)

[Lan82] Oscar E. Lanford, III, *A computer-assisted proof of the Feigenbaum conjectures*, Bull. Amer. Math. Soc. (N.S.) **6** (1982), no. 3, 427–434. (Cited on p. 95.)

[LB92] John Lund and Kenneth L. Bowers, *Sinc methods for quadrature and differential equations*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992. (Cited on p. 188.)

[Lev73] David Levin, *Development of non-linear transformations of improving convergence of sequences*, Internat. J. Comput. Math. **3** (1973), 371–388. (Cited on p. 243.)

[Lon56] Ivor M. Longman, *Note on a method for computing infinite integrals of oscillatory functions*, Proc. Cambridge Philos. Soc. **52** (1956), 764–768. (Cited on p. 18.)

[Luk75] Yudell L. Luke, *Mathematical functions and their approximations*, Academic Press, New York, 1975. (Cited on pp. 117, 195.)

[LV94] Dirk P. Laurie and Lucas M. Venter, *A two-phase algorithm for the Chebyshev solution of complex linear equations*, SIAM J. Sci. Comput. **15** (1994), no. 6, 1440–1451. (Cited on pp. 111, 115.)

[Lyn85] James N. Lyness, *Integrating some infinite oscillating tails*, Proceedings of the International Conference on Computational and Applied Mathematics (Leuven, 1984), Vol. 12/13, 1985, pp. 109–117. (Cited on p. 19.)

[Men43] Luigi Frederico Menabrea, *Sketch of the analytical engine invented by Charles Babbage, Esq. With notes by the translator (A.A.L.)*, Taylor's Scientific Memoirs **3** (1843), no. 29, 666–731. (Cited on p. 263.)

[Mil63] John Milnor, *Morse theory*, Based on lecture notes by M. Spivak and R. Wells. Annals of Mathematics Studies, No. 51, Princeton University Press, Princeton, NJ, 1963. (Cited on p. 89.)

[Mil94] Gradimir V. Milovanović, *Summation of series and Gaussian quadratures*, Approximation and Computation (West Lafayette, IN, 1993), Birkhäuser, Boston, 1994, pp. 459–475. (Cited on p. 66.)

[Mon56] Elliot W. Montroll, *Random walks in multidimensional spaces, especially on periodic lattices*, J. Soc. Indust. Appl. Math. **4** (1956), 241–260. (Cited on p. 145.)

[Moo66] Ramon E. Moore, *Interval analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1966. (Cited on p. 92.)

[Mor78] Masatake Mori, *An IMT-type double exponential formula for numerical integration*, Publ. Res. Inst. Math. Sci. Kyoto Univ. **14** (1978), no. 3, 713–729. (Cited on p. 68.)

[MS83] Gurii I. Marchuk and Vladimir V. Shaĭdurov, *Difference methods and their extrapolations*, Springer-Verlag, New York, 1983. (Cited on p. 206.)

[MS01] Masatake Mori and Masaaki Sugihara, *The double-exponential transformation in numerical analysis*, J. Comput. Appl. Math. **127** (2001), no. 1–2, 287–296. (Cited on p. 188.)

[MT03] Oleg Marichev and Michael Trott, *Meijer G function*, The Wolfram Functions Site, Wolfram Research, 2003, `http://functions.wolfram.com`. (Cited on p. 194.)

[Neh52] Zeev Nehari, *Conformal mapping*, McGraw-Hill, New York, 1952. (Cited on p. 203.)

[Neu90] Arnold Neumaier, *Interval methods for systems of equations*, Encyclopedia of Mathematics and Its Applications, vol. 37, Cambridge University Press, Cambridge, U.K., 1990. (Cited on pp. 93, 94.)

[Nie06] Niels Nielsen, *Handbuch der Theorie der Gammafunktion.*, Teubner, Leipzig, 1906. (Cited on p. 56.)

[NPWZ97] István Nemes, Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger, *How to do Monthly problems with your computer*, Amer. Math. Monthly **104** (1997), no. 6, 505–519. (Cited on p. 132.)

[Olv74] Frank W. J. Olver, *Asymptotics and special functions*, Academic Press, New York, 1974. (Cited on pp. 70, 227.)

[OM99] Takuya Ooura and Masatake Mori, *A robust double exponential formula for Fourier-type integrals*, J. Comput. Appl. Math. **112** (1999), no. 1–2, 229–241. (Cited on pp. 28, 190, 191.)

[PBM86] Anatoliĭ P. Prudnikov, Yury A. Brychkov, and Oleg I. Marichev, *Integrals and series. Vol. 1: Elementary functions*, Gordon & Breach, New York, 1986. (Cited on p. 142.)

[PdDKÜK83] Robert Piessens, Elise de Doncker-Kapenga, Christoph W. Überhuber, and David K. Kahaner, *QUADPACK: A subroutine package for automatic integration*, Springer-Verlag, Berlin, 1983. (Cited on pp. 18, 183.)

[Pól21] Georg Pólya, *Über eine Aufgabe der Wahrscheinlichkeitsrechnung betreffend die Irrfahrt im Straßennetz*, Math. Ann. **83** (1921), 149–160. (Cited on pp. 122, 135, 143.)

[Pow64] Michael J. D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comput. J. **7** (1964), 155–162. (Cited on p. 184.)

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical recipes in C*, second ed., Cambridge University Press, Cambridge, UK, 1992. (Cited on p. 77.)

[PW34] Raymond E. A. C. Paley and Norbert Wiener, *Fourier transforms in the complex domain*, American Mathematical Society, New York, 1934. (Cited on p. 70.)

[PWZ96] Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger, $A = B$, A. K. Peters, Wellesley, 1996. (Cited on pp. 131, 132.)

[Rai60] Earl D. Rainville, *Special functions*, Macmillan, New York, 1960. (Cited on p. 135.)

[Rau91] Jeffrey Rauch, *Partial differential equations*, Springer-Verlag, New York, 1991. (Cited on pp. 199, 209.)

[Rem34a] Eugene J. Remes (Evgeny Ya. Remez), *Sur le calcul effectif des polynômes d'approximation de Tchebichef*, C. R. Acad. Sci. Paris **199** (1934), 337–340. (Cited on p. 107.)

[Rem34b] ———, *Sur un procédé convergent d'approximation successives pour déterminer les polynômes d'approximation*, C. R. Acad. Sci. Paris **198** (1934), 2063–2065. (Cited on p. 107.)

[RS75] Michael Reed and Barry Simon, *Methods of modern mathematical physics. II. Fourier analysis, self-adjointness*, Academic Press, New York, 1975. (Cited on p. 70.)

[Rud87] Walter Rudin, *Real and complex analysis*, third ed., McGraw-Hill, New York, 1987. (Cited on p. 48.)

[Rum98] Siegfried M. Rump, *A note on epsilon-inflation*, Reliab. Comput. **4** (1998), no. 4, 371–375. (Cited on p. 97.)

[Rum99a] ———, *Fast and parallel interval arithmetic*, BIT **39** (1999), no. 3, 534–554. (Cited on p. 272.)

[Rum99b] ———, *INTLAB—interval laboratory*, Developments in Reliable Computing (Tibor Csendes, ed.), Kluwer, Dordrecht, 1999, pp. 77–104. (Cited on p. 272.)

[Rut90] Heinz Rutishauser, *Lectures on numerical mathematics*, Birkhäuser, Boston, 1990. (Cited on p. 107.)

[Sal55] Herbert E. Salzer, *A simple method for summing certain slowly convergent series*, J. Math. Phys. **33** (1955), 356–359. (Cited on p. 237.)

[Sch69] Charles Schwartz, *Numerical integration of analytic functions*, J. Computational Phys. **4** (1969), 19–29. (Cited on p. 68.)

[Sch89] Hans R. Schwarz, *Numerical analysis: A comprehensive introduction*, Wiley, Chichester, UK, 1989, With a contribution by Jörg Waldvogel, Translated from the German. (Cited on p. 68.)

[Sid03] Avram Sidi, *Practical extrapolation methods: Theory and applications*, Cambridge University Press, Cambridge, UK, 2003. (Cited on pp. 230, 257.)

[Sin70a] Yakov G. Sinaĭ, *Dynamical systems with elastic reflections. Ergodic properties of dispersing billiards*, Uspehi Mat. Nauk **25** (1970), no. 2 (152), 141–192. (Cited on p. 44.)

[Sin70b] Ivan Singer, *Best approximation in normed linear spaces by elements of linear subspaces*, Springer-Verlag, Berlin, 1970. (Cited on pp. 113, 119.)

[SL68] Vladimir I. Smirnov and N. A. Lebedev, *Functions of a complex variable: Constructive theory*, The MIT Press, Cambridge, MA, 1968. (Cited on pp. 113, 119.)

[Smi97] Frank Smithies, *Cauchy and the creation of complex function theory*, Cambridge University Press, Cambridge, UK, 1997. (Cited on p. 211.)

[Sok97] Alan D. Sokal, *Monte Carlo methods in statistical mechanics: Foundations and new algorithms*, Functional integration (Cargèse, 1996), NATO Adv. Sci. Inst. Ser. B Phys., Vol. 361, Plenum, New York, 1997, pp. 131–192. (Cited on pp. 199, 201.)

[SR97] Lawrence F. Shampine and Mark W. Reichelt, *The MATLAB ODE suite*, SIAM J. Sci. Comput. **18** (1997), no. 1, 1–22. (Cited on p. 171.)

[Ste65] Hans J. Stetter, *Asymptotic expansions for the error of discretization algorithms for non-linear functional equations*, Numer. Math. **7** (1965), 18–31. (Cited on p. 172.)

[Ste73] Frank Stenger, *Integration formulae based on the trapezoidal formula*, J. Inst. Math. Appl. **12** (1973), 103–114. (Cited on p. 68.)

[Ste84] Gilbert W. Stewart, *A second order perturbation expansion for small singular values*, Linear Algebra Appl. **56** (1984), 231–235. (Cited on p. 58.)

[Ste01] ———, *Matrix algorithms. Vol. II, Eigensystems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2001. (Cited on p. 162.)

[SW97] Dan Schwalbe and Stan Wagon, *VisualDSolve, Visualizing differential equations with Mathematica*, TELOS/Springer-Verlag, New York, 1997. (Cited on p. 87.)

[Syl60] James Joseph Sylvester, *Notes on the meditation of Poncelet's theorem*, Philosophical Magazine **20** (1860), 533. (Cited on p. 181.)

[Sze75] Gábor Szegő, *Orthogonal polynomials*, fourth ed., American Mathematical Society, Providence, 1975. (Cited on p. 135.)

[Tab95] Serge Tabachnikov, *Billiards*, Société Mathématique de France, Marseille, 1995. (Cited on pp. 40, 44.)

[Tan88] Ping Tak Peter Tang, *A fast algorithm for linear complex Chebyshev approximations*, Math. Comp. **51** (1988), no. 184, 721–739. (Cited on p. 119.)

[TB97] Lloyd N. Trefethen and David Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1997. (Cited on pp. 4, 10, 154, 156.)

[Tho95] James W. Thomas, *Numerical partial differential equations: Finite difference methods*, Springer-Verlag, New York, 1995. (Cited on p. 172.)

[TM74] Hidetosi Takahasi and Masatake Mori, *Double exponential formulas for numerical integration*, Publ. Res. Inst. Math. Sci. Kyoto Univ. **9** (1973/74), 721–741. (Cited on p. 68.)

[Tre81] Lloyd N. Trefethen, *Near-circularity of the error curve in complex Chebyshev approximation*, J. Approx. Theory **31** (1981), no. 4, 344–367. (Cited on p. 115.)

[Tre98] ———, *Maxims about numerical mathematics, computers, science, and life*, SIAM News **31** (1998), no. 1, 4. `http://www.siam.org/siamnews/01-98/maxims.htm`. (Cited on pp. vii, 4, 34.)

[Tre00] ———, *Predictions for scientific computing 50 years from now*, Mathematics Today (April 2000), 53–57. (Cited on p. 4.)

[Tre02] ———, *The $100, 100-Digit Challenge*, SIAM News **35** (2002), no. 6, 1–3. (Cited on pp. 2, 169, 259.)

[Tse96] Ching-Yih Tseng, *A multiple-exchange algorithm for complex Chebyshev approximation by polynomials on the unit circle*, SIAM J. Numer. Anal. **33** (1996), no. 5, 2017–2049. (Cited on p. 119.)

[Tuc02] Warwick Tucker, *A rigorous ODE solver and Smale's 14th problem*, Found. Comput. Math. **2** (2002), no. 1, 53–117. (Cited on p. 95.)

[vzGG99] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, Cambridge University Press, New York, 1999. (Cited on pp. 164, 166, 167.)

[Wal88] Jörg Waldvogel, *Numerical quadrature in several dimensions*, Numerical Integration, III (Oberwolfach, 1987), Birkhäuser, Basel, 1988, pp. 295–309. (Cited on p. 68.)

[Wat39] George N. Watson, *Three triple integrals*, Quart. J. Math., Oxford Ser. **10** (1939), 266–276. (Cited on p. 143.)

[Wat88] G. Alistair Watson, *A method for the Chebyshev solution of an overdetermined system of complex linear equations*, IMA J. Numer. Anal. **8** (1988), no. 4, 461–471. (Cited on p. 111.)

[Wat00] ———, *Approximation in normed linear spaces*, J. Comput. Appl. Math. **121** (2000), no. 1–2, 1–36. (Cited on p. 119.)

[Web91] Heinrich Weber, *Elliptische Functionen und algebraische Zahlen*, Vieweg, Braunschweig, 1891. (Cited on pp. 220, 221.)

[Wen89] Ernst Joachim Weniger, *Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series*, Computer Physics Reports **10** (1989), 189–371. (Cited on pp. 225, 230.)

[Wie86] Douglas H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inform. Theory **32** (1986), no. 1, 54–62. (Cited on p. 165.)

[Wim81] Jet Wimp, *Sequence transformations and their applications*, Mathematics in Science and Engineering, Vol. 154, Academic Press, New York, 1981. (Cited on p. 230.)

[WW96] Edmund T. Whittaker and George N. Watson, *A course of modern analysis*, Cambridge University Press, Cambridge, UK, 1996, Reprint of the fourth (1927) edition. (Cited on pp. 101, 135, 176.)

[Wyn56a] Peter Wynn, *On a device for computing the $e_m(S_n)$ tranformation*, Math. Tables Aids Comput. **10** (1956), 91–96. (Cited on p. 245.)

[Wyn56b] ———, *On a procrustean technique for the numerical transformation of slowly convergent sequences and series*, Proc. Cambridge Philos. Soc. **52** (1956), 663–671. (Cited on p. 245.)

[Wyn66] ———, *Upon systems of recursions which obtain among the quotients of the Padé table*, Numer. Math. **8** (1966), 264–269. (Cited on p. 245.)

[You81] Laurence Chisholm Young, *Mathematicians and their times*, North-Holland, Amsterdam, 1981. (Cited on p. 48.)

[Zau89] Erich Zauderer, *Partial differential equations of applied mathematics*, second ed., Wiley, New York, 1989. (Cited on pp. 202, 209.)

[Zuc79] I. J. Zucker, *The summation of series of hyperbolic functions*, SIAM J. Math. Anal. **10** (1979), no. 1, 192–206. (Cited on p. 220.)