## Chapter 3

# How Far Away Is Infinity?

**Jörg Waldvogel**

*O God! I could be bounded in a nut-shell,*
*and count myself a king of infinite space,*
*were it not that I have bad dreams.*
— William Shakespeare
(Hamlet, Act 2, Scene 2, Verses 254–256)

### Problem 3

*The infinite matrix $A$ with entries $a_{11} = 1, a_{12} = 1/2, a_{21} = 1/3, a_{13} = 1/4, a_{22} = 1/5, a_{31} = 1/6$, and so on, is a bounded operator on $\ell^2$. What is $\|A\|$?*

In §3.1 we reduce the infinite-dimensional problem to the task of calculating a limit of finite-dimensional matrix norms. A simple estimate will be given that determines two digits. In §3.2 we calculate the matrix norms without the need of much background knowledge by using Matlab's built-in `norm` command and approach the limit by extrapolation. This will bring us 12 digits, but without satisfactory evidence of correctness. A similar but somewhat more efficient algorithm based on the power method will help us produce 21 digits in §3.3. In §3.4 we use second-order perturbation theory to obtain a precise understanding of the error of approximation.

Striving for higher accuracy turns out to be particularly difficult for Problem 3. In fact, Trefethen's first publication of the results on his web page in May 2002 reported only 15 digits, as opposed to 40 digits on the other problems. In a later version, thanks to a method of Rolf Strebel, the missing 25 digits were added. Strebel's method, which is the subject of §3.5, is based on the Euler–Maclaurin sum formula and is the most efficient method to get the full accuracy of 16 digits in IEEE double-precision arithmetic. Though the method performs with remarkable success, the convergence rate slows down for very high accuracies beyond, say, 100 digits. We will overcome this difficulty in §3.6 by capturing infinity via complex analysis and contour integration.

## 3.1 A First Look

A rigorous foundation of the problem makes it necessary to recall some elementary Hilbert space theory [Rud87, Chap. 4].[1] The sequence space $\ell^2$ is the set of all square-summable real-valued sequences $x = (x_1, x_2, \ldots)$ and is endowed with an inner product, that for $x, y \in \ell^2$, by generalizing the Euclidean inner product on $\mathbb{R}^n$, is defined by

$$\langle x, y \rangle = \sum_{k=1}^{\infty} x_k y_k.$$

This induces the norm

$$\|x\| = \sqrt{\langle x, x \rangle},$$

which makes $\ell^2$ a complete space. Complete inner-product spaces are called *Hilbert spaces*; and $\ell^2$ is not only an example of such a space, but is in fact isomorphic to any separable one [Rud87, §4.19]. It is possible to generalize most of the algebraic, geometric and analytic properties of $\mathbb{R}^n$ to Hilbert spaces, and to $\ell^2$ in particular. A linear operator $A : \ell^2 \to \ell^2$ is *bounded*, and therefore continuous, if the operator norm

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} \tag{3.1}$$

is a (finite) real number. This operator norm generalizes the *spectral norm* of a matrix in $\mathbb{R}^{n \times n}$ [Hig96, §6.2], that is, the matrix norm induced by the Euclidean vector norm.

Now, calculating the norm of the particular operator at hand looks like an infinite-dimensional optimization problem. However, as the following lemma shows, one can approximate the infinite-dimensional operator $A$ by its $n$-dimensional principal submatrices $A_n$,

$$A = \begin{pmatrix} \boxed{A_n} & \vdots \\ \cdots & \ddots \end{pmatrix}, \qquad A_n \in \mathbb{R}^{n \times n}.$$

**Lemma 3.1.** *If $1 \leqslant n \leqslant m$, then*

$$\|A_n\| \leqslant \|A_m\| \leqslant \lim_{k \to \infty} \|A_k\| = \|A\| \leqslant \frac{\pi}{\sqrt{6}}.$$

***Proof.*** Let $P_n : \ell^2 \to \operatorname{span}\{e_1, \ldots, e_n\} \subset \ell^2$ be the orthogonal projection from $\ell^2$ onto the $n$-dimensional subspace that is spanned by the first $n$ basis sequences

[1] Rumor has it that Hilbert himself once asked Weyl after a talk [You81, p. 312]: "Weyl, you must just tell me one thing, whatever is a Hilbert space? That I could not understand." A reader who feels like this can skip to the end of Lemma 3.1.

$(e_j)_k = \delta_{jk}$. This subspace can be identified with $\mathbb{R}^n$ and we obtain $A_n = P_n A P_n$. For $n \leqslant m$ we have $P_n = P_n P_m = P_m P_n$ and therefore $A_n = P_n A_m P_n$. Submultiplicativity of the operator norm, and the fact that an orthogonal projection satisfies $\|P_n\| \leqslant 1$, yields

$$\|A_n\| \leqslant \|P_n\|^2 \, \|A_m\| \leqslant \|A_m\| = \|P_m A P_m\| \leqslant \|P_m\|^2 \|A\| \leqslant \|A\|. \tag{3.2}$$

Thus, the sequence $\|A_n\|$ is monotonically increasing. We do not yet know that it is bounded, because we have not given an argument that $\|A\| < \infty$.

However, such a bound follows from the fact that the spectral norm $\|A_n\|$ is known to be bounded by the Frobenius norm $\|A_n\|_F$ (see [Hig96, Table 6.2]) and therefore

$$\|A_n\| \leqslant \|A_n\|_F = \left( \sum_{j,k=1}^{n} a_{jk}^2 \right)^{1/2} \leqslant \left( \sum_{k=1}^{\infty} k^{-2} \right)^{1/2} = \frac{\pi}{\sqrt{6}}. \tag{3.3}$$

Thus, the limit $\lim_{n\to\infty} \|A_n\| = \sup_n \|A_n\| \leqslant \pi/\sqrt{6}$ of the increasing sequence exists. Because of the completeness of the basis sequences we know that $\lim_{n\to\infty} P_n x = x$ for all $x \in \ell^2$ and, hence,

$$\|Ax\| = \lim_{n\to\infty} \|P_n A P_n x\| \leqslant \lim_{n\to\infty} \|A_n\| \, \|x\|, \qquad \text{that is} \quad \|A\| \leqslant \lim_{n\to\infty} \|A_n\|.$$

Combined with (3.2), this finishes the proof. □

Summarizing, Problem 3 in fact asks for

$$\lim_{n\to\infty} \|A_n\|,$$

the limit of the spectral norms of the finite-dimensional principal submatrices $A_n$. We can safely stop talking about infinite-dimensional operators from now on: this limit will be the starting point for our computational enterprise. The precise order of convergence $\|A_n\| \to \|A\|$ will be the subject of §3.4.

Besides setting up the problem in a suitable form, Lemma 3.1 allows us to give, with proof, the first two digits of the answer:

$$1.233 \doteq \|A_3\| \leqslant \|A\| \leqslant \frac{\pi}{\sqrt{6}} \doteq 1.282, \qquad \text{that is} \quad \|A\| \doteq 1.2.$$

In finite dimensions, the spectral norm of a matrix $A_n$ satisfies [GL96, Thm. 2.3.1]

$$\|A_n\| = \sigma_{\max}(A_n) = \sqrt{\lambda_{\max}(G_n)}, \qquad G_n = A_n^T A_n. \tag{3.4}$$

Here $\sigma_{\max}(A_n)$ denotes the largest singular value[2] of $A_n$ and $\lambda_{\max}(G_n)$ the largest eigenvalue of $G_n$.

[2] Generally, the singular values of a matrix $A$ are the square roots of the eigenvalues of the positive semi-definite matrix $G = A^T A$.

**Matrix Generation.** The matrix $\tilde{A} = (1/a_{jk})$ of the reciprocal elements of $A$ is given by the northeast to southwest arrangement of the natural numbers,

$$\tilde{A} = \begin{pmatrix} 1 & 2 & 4 & 7 & \dots \\ 3 & 5 & 8 & 12 & \dots \\ 6 & 9 & 13 & 18 & \dots \\ 10 & 14 & 19 & 25 & \dots \\ & \cdots\cdots\cdots & & & \ddots \end{pmatrix}.$$

We take advantage of the representation of $\tilde{A}$ as the difference of a Hankel matrix and a matrix with constant columns,

$$\tilde{A} = \begin{pmatrix} 2 & 4 & 7 & 11 & \dots \\ 4 & 7 & 11 & 16 & \dots \\ 7 & 11 & 16 & 22 & \dots \\ 11 & 16 & 22 & 29 & \dots \\ & \cdots\cdots\cdots & & & \ddots \end{pmatrix} - \begin{pmatrix} 1 & 2 & 3 & 4 & \dots \\ 1 & 2 & 3 & 4 & \dots \\ 1 & 2 & 3 & 4 & \dots \\ 1 & 2 & 3 & 4 & \dots \\ & \cdots\cdots & & & \ddots \end{pmatrix}.$$

Thus, the entries of $A$ are given as

$$a_{jk} = \frac{1}{b_{j+k} - k}, \qquad j,k = 1,2,\dots, \tag{3.5}$$

where $b_l = 1+(l-1)l/2$. The expression (3.5) is a useful tool for efficiently generating the principal submatrices $A_n$ as well as for coding the mapping $x \mapsto Ax$ without explicitly storing $A$.

**A Matlab session.**

```
>> n = 1024;
>> N = 2*n-1; k = 1:N; b = 1 + cumsum(k);
>> A = 1./(hankel(b(1:n),b(n:N))-ones(n,1)*k(1:n));
```

This is more elegant[3] than the generation of $A_n$ by means of the explicit formula

$$a_{jk} = \frac{1}{(j+k-1)(j+k)/2 - (k-1)}, \qquad j,k = 1,2,\dots,n, \tag{3.6}$$

which is just (3.5) written out in full. Note that the decay is $a_{jk} = O((j+k)^{-2})$ for large $j$ and $k$; this is faster than the decay rate of, say, the famous Hilbert matrix $H = (h_{jk})$ with $h_{jk} = 1/(j+k-1)$.

The entries $a_{jk}$ enjoy a specific feature that will become important in §3.6: they are given by a rational function $a$ evaluated at integer arguments, $a_{jk} = a(j,k)$,

[3] Generating the matrix by the `hankel` command used to be, in Matlab prior to the introduction of the JIT compilation technology in Release 13, substantially faster than by the explicit formula. Now the run time of the two variants is roughly the same.

$j, k \in \mathbb{N}$. That function extends naturally as a meromorphic function to the complex $u$- and $v$-planes,

$$a(u,v) = \frac{1}{b(u+v) - v}, \qquad b(z) = 1 + (z-1)z/2, \tag{3.7}$$

with no poles at the grid points $(u, v) \in \mathbb{N}^2$.

## 3.2 Extrapolation

In this section we present the simple method that was used by many contestants to obtain 10 to 12 digits of $\|A\|$ with some level of confidence, but without going deeper into the mysteries of the matrix $A$.

The method takes advantage of many packages' capability to directly compute the spectral norm $\|A_n\|$ of the principal submatrices at hand. For instance, if the matrix is named `A` in code, Matlab calculates the norm by invoking the command `norm(A)`. Internally this is done by a singular-value decomposition of $A_n$ (see [GL96, §8.6]). The run time of using `norm(A)` scales as $O(n^3)$.

Alternatively, Matlab offers the command `normest(A,tol)` that iteratively calculates an approximation of $\|A\|$ up to a relative error `tol`. It is based on the power method, which we will discuss in §3.3, and takes $O(n^2)$ time for a fixed accuracy `tol`.

The second column of Table 3.1 shows $\|A_n\|$ for dimensions a power of 2: $n = 1, 2, 4, \ldots, 4096$. The values nicely reflect the monotonicity stated in Lemma 3.1. A close look on the digits that keep agreeing from one row to the next suggests that doubling $n$ gives a little less than one additional correct digit. From this we would infer about 10 correct digits for $n = 2048$. To be on the safe side, we go for $n = 4096$ and get

$$\|A\| \doteq 1.2742\,24152.$$

The memory needed to store the matrix $A_{4096}$ amounts for 120 MB; calculating $\|A_{4096}\|$ by the command `norm(A)` takes 12 minutes and by `normest(A,5e-16)` 12 seconds.[4] Both results agree to 15 digits.

A much more efficient way of getting up to 12 digits in less than a second is based on using the values for small $n$ and extrapolation to the limit $n \to \infty$.

**A Matlab session.** *(cont. of session on p. 50)*

First, let us generate and store the values of the norms for $n = 1, 2, 4, \ldots, 2^{L-1}$. We have chosen $L = 11$, corresponding to the maximum dimension $n = 1024$.

```
>> L = 11;
>> vals = [];
>> for nu = 1:L,
>>    n=2^(nu-1); An=A(1:n,1:n); vals=[vals;normest(An,5e-16)];
>> end;
```

[4] In this chapter all timings are for a 1.6 GHz PC.

**Table 3.1.** *Values of* $\|A_n\|$ *and extrapolation by Wynn's epsilon algorithm.*

| $n$ | `vals` $= \|A_n\|$ | Wynn for $L = 10$ | Wynn for $L = 11$ |
|---|---|---|---|
| 1 | **1.**00000000000000 | **1.**00000000000000 | **1.**00000000000000 |
| 2 | **1.**18335017655166 | **1.29**446756234379 | **1.29**446756234379 |
| 4 | **1.25**253739751680 | **1.274**09858051212 | **1.274**09858051212 |
| 8 | **1.27**004630585408 | **1.27422415**478429 | **1.27422415**478429 |
| 16 | **1.27**352521545013 | **1.27422416**116628 | **1.27422416**116628 |
| 32 | **1.274**11814436915 | **1.27422415**78695 | **1.27422415282**204 |
| 64 | **1.274**20913129766 | **1.27422415285**119 | **1.27422415282**075 |
| 128 | **1.27422**212003778 | **1.27422415**71828 | **1.27422415282**239 |
| 256 | **1.27422**388594855 | **1.27422415**371348 | **1.27422415281**540 |
| 512 | **1.27422411**845808 | **1.27422411**845808 | **1.27422415289**127 |
| 1024 | **1.27422414**844970 | — | **1.27422414**844970 |
| 2048 | **1.27422415226**917 | — | — |
| 4096 | **1.27422415275**182 | — | — |

Next, we use *Wynn's epsilon algorithm* (see Appendix A, p. 247) for approximating, by extrapolation, the limit of the data that are stored in `vals`.

```
>> L2 = 2*L-1; vv = zeros(L2,1); vv(1:2:L2) = vals;
>> for j=2:L
>>   k=j:2:L2+1-j; vv(k)=vv(k)+1./(vv(k+1)-vv(k-1));
>> end;
>> result = vv(1:2:L2);
```

The output, which is stored in the column vector `result`, contains, from top to bottom, the values of the right boundary of Wynn's triangular array. The best approximations for $\|A\|$ are generally located just below the central element. Table 3.1 gives the results of the session just run with $L = 11$, and of another run with $L = 10$. The run time was less than half a second. In the less expensive case $L = 10$ of the extrapolation, the entries for $n = 32$, 64, and 128 agree to 10 digits. In the case $L = 11$ these entries agree to 12 digits; therefore, at this point we can confidently report

$$\|A\| \doteq 1.2742\,24152\,82. \tag{3.8}$$

Thus, extrapolation yields two more digits in a factor of about 40 less run time than the simple approximation by $\|A_{4096}\|$. Similar results are obtained with other extrapolation techniques. In Appendix A, pp. 250 and 256, the reader will find, as yet another example, applications of *Levin's U* and the *rho* algorithm to Problem 3.

This looks like a quick exit from Problem 3. However, to get more evidence for the calculated digits, we need higher accuracy, which this approach cannot easily provide.

## 3.3 The Power Method

One of the bottlenecks of the method in §3.2 is the enormous amount of memory needed to store the matrix $A_n$ for larger $n$: 120 MB are needed for $n = 4096$. This leads us to ask for an iterative method of calculating the norm $\|A_n\|$ that addresses $A_n$ only by the matrix-vector products $x \mapsto Ax$ and $x \mapsto A^T x$. Such a method is readily obtained if we recall (3.4), that is

$$\|A_n\|^2 = \lambda_{\max}(G_n), \qquad G_n = A_n^T A_n.$$

The largest eigenvalue of the symmetric positive semi-definite matrix $G_n$ can efficiently be calculated by the *power method* [GL96, §8.2.1], a basic method that can be found in virtually any textbook on elementary numerical analysis.

**Algorithm 3.1. Power method for calculating $\lambda_{\max}(G_n)$.**

*take an initial vector $x^{(0)}$ of norm one*

**for** $\nu = 1$ **to** $\nu_{\max}$ **do**

$y^{(\nu)} = G_n x^{(\nu-1)}$;

$\lambda^{(\nu-1)} = \left(x^{(\nu-1)}\right)^T y^{(\nu)}$;

**if** $\lambda^{(\nu-1)}$ *is sufficiently accurate* **then exit**;

$x^{(\nu)} = y^{(\nu)}/\|y^{(\nu)}\|$;

**end for**

**Convergence Theory.** This simple method has a correspondingly simple convergence theory.

**Theorem 3.2 ([GL96, Thm. 8.2.1]).** *If the largest eigenvalue $\lambda_1 = \lambda_{\max}(G_n)$ of a symmetric, positive semi-definite $n \times n$ matrix $G_n$ is* simple*, that is, if the $n$ eigenvalues of $G_n$ can be ordered as $\lambda_1 > \lambda_2 \geqslant \lambda_3 \geqslant \ldots \geqslant \lambda_n \geqslant 0$, and if the initial vector $x^{(0)}$ is in general position, that is, not orthogonal to the eigenvector belonging to $\lambda_1$, there is a constant $c > 0$ such that*

$$|\lambda^{(\nu)} - \lambda_{\max}(G_n)| \leqslant c\rho^\nu, \qquad \nu = 1, 2, \ldots,$$

*with the contraction rate $\rho = (\lambda_2/\lambda_1)^2$.*

There are two slick ways of showing that the matrix at hand, $G_n = A_n^T A_n$, does in fact have a largest eigenvalue that is simple. The second method has the advantage of giving a quantitative bound for the contraction rate $\rho$.

**Method 1.** We observe that all entries of $G_n$ are strictly positive. Therefore, the *Perron–Frobenius theory* of nonnegative matrices [HJ85, §8.2] is applicable. In

particular, there is a general theorem [HJ85, Thm. 8.2.5] of Perron stating that for positive matrices the eigenvalue of largest modulus, that is, the *dominant* eigenvalue, is always simple.

**Method 2.** We recall the bound (3.3) on $\|A_n\|_F$, which — given the representation [GL96, form. (2.5.7)] of the Frobenius norm as the sum of the squares of the singular values — can be expressed as

$$\|A_n\|_F^2 = \lambda_1 + \lambda_2 + \ldots + \lambda_n \leqslant \frac{\pi^2}{6}.$$

Because of $\lambda_1 = \|A_n\|^2 \geqslant \|A_3\|^2 \doteq 1.52$ we obtain for $n \geqslant 3$,

$$\lambda_2 \leqslant \frac{\pi^2}{6} - \lambda_1 \leqslant 0.125, \qquad \rho = \left(\frac{\lambda_2}{\lambda_1}\right)^2 \leqslant 6.8 \cdot 10^{-3}.$$

In fact, a numerical calculation shows that for large $n$ the second largest eigenvalue of $G_n$ is $\lambda_2 \doteq 0.020302$, which yields — with (3.8), that is, $\lambda_1 = \|A_n\|^2 \doteq 1.6236$ — the contraction rate

$$\rho \doteq 1.5635 \cdot 10^{-4}. \tag{3.9}$$

This corresponds to a gain of $-\log_{10}\rho \doteq 3.8$ digits per iteration step, which actually has been observed in our numerical experiments.

**Application to Problem 3.** We use (3.5) to efficiently implement the matrix-vector multiplication step in Algorithm 3.1, that is

$$y^{(\nu)} = G_n x^{(\nu-1)}, \qquad G_n = A_n^T A_n.$$

There is no need to explicitly set up the matrix $G_n$. Instead we factor the matrix-vector product into two separate steps, $\tilde{x}^{(\nu)} = A_n x^{(\nu-1)}$ and $y^{(\nu)} = A_n^T \tilde{x}^{(\nu)}$, which are given by

$$\begin{aligned} \tilde{x}_j^{(\nu)} &= \sum_{k=1}^{n} a_{jk} x_k^{(\nu-1)} = \sum_{k=1}^{n} \frac{x_k^{(\nu-1)}}{b_{j+k} - k}, \qquad j = 1, \ldots, n, \\ y_k^{(\nu)} &= \sum_{j=1}^{n} a_{jk} \tilde{x}_j^{(\nu)} = \sum_{j=1}^{n} \frac{\tilde{x}_j^{(\nu)}}{b_{j+k} - k}, \qquad k = 1, \ldots, n. \end{aligned} \tag{3.10}$$

To minimize the influence of roundoff errors it is advisable [Hig96, p. 90] to accumulate all sums[5] from small to large terms, that is, from $j = n$ to $j = 1$ and from $k = n$ to $k = 1$. The price paid for not storing the matrix $A_n$ is minimal: instead of one scalar multiplication in the inner loop of each matrix-vector product two index additions and one division need to be carried out. Since by (3.9) the contraction rate $\rho$ is essentially independent of the dimension $n$, the run time of approximating $\|A_n\|$ to a fixed accuracy scales as $O(n^2)$.

[5] The sums involve only nonnegative terms if the initial vector $x^{(0)}$ was chosen to be nonnegative.

**Table 3.2.** *Values of $\|A_n\|$ and extrapolation by Wynn's epsilon algorithm.*

| $n$ | vals $= \|A_n\|$ | Wynn for $L = 16$ |
|---|---|---|
| .. | ..... | ..... |
| 64 | **1.274209131297662** | **1.27422415282122**8343690360386 |
| 128 | **1.27422**2120037776 | **1.27422415282122818**8076956823 |
| 256 | **1.27422**3885948554 | **1.27422415282122818821**3398344 |
| 512 | **1.274224**118458080 | **1.27422415282122818821**2253584 |
| 1024 | **1.274224**148449695 | **1.27422415282122818821**3171143 |
| 2048 | **1.27422415**2269170 | **1.27422415282122818820**5973369 |
| 4096 | **1.27422415**2751815 | **1.27422415282122818829**9867942 |
| 8192 | **1.27422415281**2522 | **1.2742241528212281**85212556813 |
| 16384 | **1.27422415282**0138 | **1.27422415282122**8335619616358 |
| 32768 | **1.27422415282**1091 | **1.27422415282**1091776178588977 |

The reader will find an implementation `OperatorNorm(x,tol)` of these ideas coded in PARI/GP in Appendix C.2.1 and coded in Matlab in Appendix C.3.1. It takes as input an initial vector `x` and an absolute tolerance `tol` for $\|A_n\|$ and outputs the approximation to $\|A_n\|$ as well as the final vector $x^{(k)}$ of the power method. This final vector allows for a *hierarchical* version of the iterative procedure: since the eigenvector of $G_n$ belonging to the dominant eigenvalue is an approximation of a fixed vector in the sequence space $\ell^2$, we can profitably use this final vector of dimension $n$, padded by zeros, to start the power method for dimension $2n$. Compared to a fixed initial vector, such as the first basis vector, this reduces the run time by about a factor of two.

We now extend Table 3.1 up to the dimension $n = 32768$. To increase the confidence in the correctness of the solution we go to extended precision, using PARI/GP's default precision of 28 decimal digits.

**A PARI/GP session.** *(compare with Matlab session on p. 51)*

```
? dec=28;  default(realprecision,dec); tol=10^(2.0-dec);
? L=16; vals=vector(L); res=[1.0,1.0]; vals[1]=res[1];
? for(nu=2,L, x0=concat(res[2],0*res[2]); res=OperatorNorm(x0,tol);\
        vals[nu]=res[1]);
? L2=2*L-1; vv=vector(L2,j, if(j%2==1, vals[(j+1)/2], 0));
? for(j=2,L, forstep(k=j, L2+1-j, 2,\
        vv[k]=vv[k]+1/(vv[k+1]-vv[k-1])));
? result=vector(L,j, vv[2*j-1])
```

The entries of the vectors `vals` and `result` are shown in the second and third columns of Table 3.2.[6] The run time was less than 8 hours.

[6] It is comforting to observe that Table 3.2 and Table 3.1 agree to all the digits given.

The entries for $n = 256$, 512, and 1024 agree to 21 digits. Therefore we can confidently report

$$\|A\| \doteq 1.2742\,24152\,82122\,81882\,1.$$

This is about as far as we can get by the straightforward technique of handling the principal submatrices $A_n$ and extrapolating to the limit as $n \to \infty$ by a general-purpose method. In §3.5 we will start using conclusive information about the *sequence* of submatrices, or equivalently, about the infinite matrix $A$.

**A Dead End: Calculating a Closed Form for $A^TA$.** We conclude this section with a digression to study a rather surprising closed form for the elements

$$g_{jl} = \sum_{k=1}^{\infty} a_{kj} a_{kl}$$

of the infinite matrix $G = A^TA$. Such a closed form suggests using the principal submatrices $\hat{G}_n$ of $G$ instead of the matrices $G_n = A_n^T A_n$ to run the power method. However, as we shall see, this does not seem to be of any use in the actual calculation of $\|A\|$. A reader with no further interest in higher transcendental functions and symbolic computation might wish to skip the rest of this section and go to §3.4.

By representing the coefficient $a_{jk}$ as in (3.6) we obtain $g_{jl} = \sum_{k=1}^{\infty} R_{jl}(k)$, where

$$R_{jl}(k) = \frac{4}{\big(k^2 + (2j-1)k + (j-1)(j-2)\big)\big(k^2 + (2l-1)k + (l-1)(l-2)\big)},$$

a rational function with denominator of degree 4 in $k$. An elegant technique for evaluating sums of rational functions over equally spaced points is described in [AS84, §6.8]:[7] the expansion of $R_{jl}$ in partial fractions directly yields the value of the sum in terms of the psi (or digamma) function, $\psi(z) = \Gamma'(z)/\Gamma(z)$, and its derivatives, called polygamma functions.

The partial-fraction expansion of $R_{jl}$ is determined by its poles, located at

$$k_{1,2} = \frac{1}{2}\left(1 - 2j \pm \sqrt{8j-7}\right), \qquad k_{3,4} = \frac{1}{2}\left(1 - 2l \pm \sqrt{8l-7}\right).$$

Now, the term $g_{jl}$ will be a linear combination of the values $\psi(1-k_\nu)$, $\psi'(1-k_\nu)$, $\psi''(1-k_\nu)$,..., $\psi^{(\mu_\nu - 1)}(1-k_\nu)$, where $\mu_\nu$ is the multiplicity of the pole $k_\nu$. Some computer algebra systems, such as *Mathematica* and Maple, offer implementations of this algorithm.

**A Maple session.**

```
> j:=1: l:=1:
> j1:=2*j-1: j2:=(j-1)*(j-2): l1:=2*l-1: l2:=(l-1)*(l-2):
> gjl:=sum(4/(k^2+j1*k+j2)/(k^2+l1*k+l2),k=1..infinity);
```

$$g_{11} = \frac{4}{3}\pi^2 - 12$$

[7] According to [Nie06, §24] the technique dates back to a short note of Appell from 1878.

A few more results are, after some beautification:

$$g_{12} = \frac{2}{9}\,\pi^2 - \frac{43}{27}, \quad g_{22} = \frac{4}{27}\,\pi^2 - \frac{31}{27}, \quad g_{24} = \frac{137}{1350}, \quad g_{17} = \frac{26281}{396900},$$

$$g_{13} = 4\,\gamma - 2 + \left(2 + \frac{8}{\sqrt{17}}\right)\psi\left(\frac{7-\sqrt{17}}{2}\right) + \left(2 - \frac{8}{\sqrt{17}}\right)\psi\left(\frac{7+\sqrt{17}}{2}\right),$$

where $\gamma$ is Euler's constant,

$$g_{35} = -\frac{2}{17}\left(\psi\left(\frac{7-\sqrt{17}}{2}\right) + \psi\left(\frac{7+\sqrt{17}}{2}\right)\right) + \frac{2}{17}\left(\left(1 + \frac{4}{\sqrt{33}}\right)\psi\left(\frac{11-\sqrt{33}}{2}\right) + \left(1 - \frac{4}{\sqrt{33}}\right)\psi\left(\frac{11+\sqrt{33}}{2}\right)\right).$$

From the point of view of closed forms the case $g_{33}$ is of particular interest. Maple yields

$$g_{33} = \frac{4}{17}\left(\psi'\left(\frac{7-\sqrt{17}}{2}\right) + \psi'\left(\frac{7+\sqrt{17}}{2}\right)\right) + \frac{8\sqrt{17}}{289}\left(\psi\left(\frac{7-\sqrt{17}}{2}\right) - \psi\left(\frac{7+\sqrt{17}}{2}\right)\right),$$

whereas *Mathematica* gives:

**A Mathematica session.**

```
g[j_, l_] :=
 Sum[4/((k^2 + (2 j - 1) k + (j - 1) (j - 2)) (k^2 + (2 l - 1) k + (l - 1) (l - 2))),
  {k, 1, Infinity}] //
  FullSimplify

g[3, 3]
```

$$-\frac{9}{4} - \frac{4}{289}\,\pi\,\mathtt{Sec}\left[\frac{\sqrt{17}\,\pi}{2}\right]^2 \left(-17\,\pi + \sqrt{17}\;\mathtt{Sin}\left[\sqrt{17}\,\pi\right]\right)$$

John Boersma communicated to us a proof that the two expressions have the same value indeed. Moreover, he proved the remarkable fact that $g_{jj}$ is generally expressible in terms of elementary functions:[8]

$$g_{jj} = \frac{4\pi^2}{3(2q+1)^2} - \frac{4}{(2q+1)^2}\left(\sum_{m=1}^{j-q-1}\frac{1}{m^2} + \sum_{m=1}^{j+q}\frac{1}{m^2}\right) - \frac{8}{(2q+1)^3}\sum_{m=j-q}^{j+q}\frac{1}{m},$$

if $j = (q^2+q+2)/2$ for some $q = 0, 1, 2, \ldots$, and otherwise

$$g_{jj} = -4\sum_{m=0}^{j-1}\frac{1}{(m^2+m+2-2j)^2} + \frac{4\pi}{(8j-7)^2}\,\sec^2(\sqrt{8j-7}\,\pi/2)\left((8j-7)\pi - \sqrt{8j-7}\,\sin(\sqrt{8j-7}\,\pi)\right).$$

[8] Courtesy of J. Boersma, his proof can be found at the web page for this book.

The partial-fraction algorithm shows that any entry $g_{jl}$ of the infinite matrix $G$ can be computed as a closed-form expression in terms of the functions $\psi$ and $\psi'$. As elegant as this might be, this approach turns out to be a dead end:

- In contrast to the entries $a_{jl}$ of the matrix $A$ the entries $g_{jl}$ of $G$ take much more effort to calculate for given indices $j$ and $l$. This is, however, not compensated by a faster rate of decay.
- Recall from Table 3.1 that

$$\|A\| - \|A_{128}\| \doteq 2.0328 \cdot 10^{-6}.$$

Consider, on the other hand, the principal submatrix $\hat{G}_{128}$ of $G$ of dimension $n = 128$, whose generation involves substantial symbolic and numerical effort. Its dominant eigenvalue, as computed by Matlab is found to be

$$\lambda_{\max}(\hat{G}_{128}) \doteq 1.6236447743.$$

Therefore,

$$\|A\| - \sqrt{\lambda_{\max}(\hat{G}_{128})} \doteq 0.9485 \cdot 10^{-6},$$

a rather modest gain of accuracy.

## 3.4 Second-Order Perturbation Theory

A closer look at Table 3.2 suggests that the norms of the principal submatrices $A_n$ of dimension $n$ approximate the norm of the operator $A$ up to an error of order $O(n^{-3})$. We will use second-order perturbation theory to explain this behavior. As a by-product we obtain a correction term that can be used to solve Problem 3 to 12 correct digits in IEEE double precision.

The following result is a special case of a second-order perturbation expansion that Stewart originally established for the *smallest* singular value of a matrix. It holds true, in fact, for the largest singular value with exactly the same proof.

**Theorem 3.3 ([Ste84]).** *Let*

$$\tilde{A} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} + E$$

*be a block matrix of dimension $m \times m$. Let $u$ and $v$ be the left and right singular vectors belonging to the largest singular value of the $n \times n$ principal submatrix $A$,*[9] *$n \leqslant m$. For $\|E\| \to 0$, there holds the asymptotic expansion*

$$\|\tilde{A}\|^2 = \|A\|^2 + \|u^T B\|^2 + \|Cv\|^2 + O(\|E\|^3).$$

For the problem at hand we apply this perturbation theorem to the block partitioning

$$A_m = \begin{pmatrix} A_n & B_{n,m} \\ C_{n,m} & D_{n,m} \end{pmatrix} = \begin{pmatrix} A_n & 0 \\ 0 & 0 \end{pmatrix} + E_{n,m}, \qquad n \leqslant m.$$

[9] That is, $u$ and $v$ are vectors of norm one that satisfy $AA^T u = \sigma_{\max}^2(A)u$ and $A^T Av = \sigma_{\max}^2(A)v$. Algorithm 3.1 actually allows us to calculate them: $x^{(\nu)}$ and $Ax^{(\nu)}$, normalized to have norm one, approximate $u$ and $v$ up to an error of order $O(\rho^\nu)$.

Using once again the bound of the spectral norm by the Frobenius norm, we get

$$\|E_{n,m}\|^2 \leqslant \|E_{n,m}\|_F^2 = \|A_m\|_F^2 - \|A_n\|_F^2 \leqslant \sum_{k>n^2/2} k^{-2} = O(n^{-2}).$$

In the same way we obtain $\|B_{n,m}\| = O(n^{-1})$ and $\|C_{n,m}\| = O(n^{-1})$. Thus, if we denote by $u_n$ and $v_n$ the left and right singular vectors of $A_n$ that belong to the largest singular value, Theorem 3.3 results in the error estimate

$$\begin{aligned}\|A_n\|^2 \leqslant \|A_m\|^2 &= \|A_n\|^2 + \|u_n^T B_{n,m}\|^2 + \|C_{n,m} v_n\|^2 + O(n^{-3}) \\ &\leqslant \|A_n\|^2 + \|B_{n,m}\|^2 + \|C_{n,m}\|^2 + O(n^{-3}) = \|A_n\|^2 + O(n^{-2}). \quad (3.11)\end{aligned}$$

Thus, the argument so far proves an error estimate of order $O(n^{-2})$ only — instead of the numerically observed order $O(n^{-3})$. We must have given away too much in using the submultiplicativity bounds

$$\|u_n^T B_{n,m}\| \leqslant \|B_{n,m}\| = O(n^{-1}), \qquad \|C_{n,m} v_n\| \leqslant \|C_{n,m}\| = O(n^{-1}).$$

In fact, a closer look at the singular vectors $u_n$ and $v_n$ reveals, experimentally, that their $k$th component decays of order $O(k^{-2})$ as $k \to \infty$.[10] *Assuming* this to be true, we obtain after some calculations the refined estimates

$$\|u_n^T B_{n,m}\| = O(n^{-3/2}), \qquad \|C_{n,m} v_n\| = O(n^{-3/2});$$

hence $\|A_m\|^2 = \|A_n\|^2 + O(n^{-3})$. Since all these estimates are uniform in $m$ we can pass to the limit and obtain the desired error estimate

$$\|A\| = \|A_n\| + O(n^{-3}).$$

We *conjecture* that the actual decay rate of the components of the singular vectors improves the error term $O(n^{-3})$ in (3.11) in the same way to $O(n^{-4})$, that is

$$\|A_m\|^2 = \|A_n\|^2 + \|u_n^T B_{n,m}\|^2 + \|C_{n,m} v_n\|^2 + O(n^{-4}).$$

If we combine with $\|A\|^2 = \|A_m\|^2 + O(m^{-3})$ and take

$$m_n = \lceil n^{4/3} \rceil,$$

we finally get an improved approximation of the conjectured order $O(n^{-4})$:

$$\|A\| = \left(\|A_n\|^2 + \|u_n^T B_{n,m_n}\|^2 + \|C_{n,m_n} v_n\|^2\right)^{1/2} + O(n^{-4}). \quad (3.12)$$

The improved order of convergence, as compared to $\|A_n\|$, is paid for with a moderate increase in the computational cost from $O(n^2)$ to $O(m_n \cdot n) = O(n^{7/3})$. Therefore, the run time needed to achieve a given error $\epsilon$ of approximation improves from $O(\epsilon^{-2/3})$ to $O(\epsilon^{-7/12})$.

Table 3.3 shows some numerical results[11] obtained with Matlab, which are consistent with the conjectured order $O(n^{-4})$. The run time was about 10 seconds; 12 digits are correct for $n = 2048$. Because of roundoff errors in the last two digits Wynn's epsilon algorithm cannot improve upon these 12 digits.

[10] This reflects most probably the proper decay rate of the corresponding left and right singular vectors of the *infinite* matrix $A$.

[11] The code can be found at the web page for this book.

**Table 3.3.** *Values of $\|A_n\|$ and the improved approximation (3.12).*

| $n$ | $\|A_n\|$ | $\left(\|A_n\|^2 + \|u_n^T B_{n,m_n}\|^2 + \|C_{n,m_n} v_n\|^2\right)^{1/2}$ |
|---|---|---|
| 128 | **1.27422**212003778 | **1.27422413**149024 |
| 256 | **1.27422**388594855 | **1.27422415**144773 |
| 512 | **1.27422411**845808 | **1.27422415**73407 |
| 1024 | **1.27422414**844970 | **1.27422415281**574 |
| 2048 | **1.27422415226**917 | **1.27422415282**070 |

## 3.5 Summation Formulas: Real Analysis

As an alternative to the acceleration of convergence of the sequence $\|A_n\|$, we now pursue the idea of passing to the limit $n \to \infty$ earlier, at each iteration step of the power method (Algorithm 3.1). Equivalently, this means to apply the power method to the infinite matrix $A$. In doing so we have to evaluate infinite-dimensional matrix-vector products such as $Ax$ and $A^T x$, that is, series such as

$$\sum_{k=1}^{\infty} a_{jk} x_k \qquad \text{and} \qquad \sum_{k=1}^{\infty} a_{kj} x_k. \tag{3.13}$$

A simple truncation of the sum at index $n$ would lead us back to the evaluation of $\|A_n\|$. Thus, we need a more sophisticated method to approximate these sums. Essentially, such methods can be justified on the ground of the actual rate of decay of the terms of the sum.

In the course of the power method, the vectors $x$ at hand are approximations of the left and right singular vectors belonging to the largest singular value of $A$. From the discussion of §3.4 we have good reasons to assume that $x_k = O(k^{-2})$ as $k \to \infty$. The explicit formula (3.6) implies that, for $j$ fixed, the matrix entries $a_{jk}$ and $a_{kj}$ decay as $O(k^{-2})$, too. Further, we observe that all terms of the sum will be nonnegative, if the initial vector for the power method was chosen to be nonnegative. Hence, a good *model* of the sums (3.13) is

$$\zeta(4) = \sum_{k=1}^{\infty} k^{-4} = \frac{\pi^4}{90}, \tag{3.14}$$

for which we will test our ideas later on. For sums of this type, we will derive some *summation formulas*

$$\sum_{k=1}^{\infty} f(k) \approx \sum_{k=1}^{n} w_k \cdot f(c_k), \tag{3.15}$$

where — in analogy to quadrature formulas — the nonnegative quantities $w_k$ are called *weights* and the $c_k$ are the *sampling points*. We will take the freedom to view $f(k)$ as a function of $k$ that naturally extends to non-integer arguments $c_k \notin \mathbb{N}$. This point of view is natural indeed for the coefficients $a_{jk} = a(j,k)$ at hand, with

$a(j,k)$ the rational function given in (3.7). Now, if we approximate the matrix-vector products that arise in the power method by the summation formula (3.15), the components $x_k$ will inherit this property. This way, the main step $y^{(\nu)} = A^T A x^{(\nu-1)}$ of the power method (see (3.10)) transforms to

$$\begin{aligned} \tilde{x}^{(\nu)}(c_j) &= \sum_{k=1}^{n} w_k \cdot a(c_j, c_k) \cdot x^{(\nu-1)}(c_k), \qquad j = 1, \ldots, n, \\ y^{(\nu)}(c_k) &= \sum_{j=1}^{n} w_j \cdot a(c_j, c_k) \cdot \tilde{x}^{(\nu)}(c_j), \qquad j = 1, \ldots, n. \end{aligned} \tag{3.16}$$

Upon introducing the diagonal matrix $W_n = \mathrm{diag}(w_1, \ldots, w_n)$ and the matrix $T_n = (a(c_j, c_k))_{jk}$, we observe that the transformed power iteration (3.16) calculates, in fact, the dominant eigenvalue of the matrix

$$\tilde{G}_n = T_n^T W_n T_n W_n. \tag{3.17}$$

By means of the similarity transformation $B \mapsto W_n^{1/2} B W_n^{-1/2}$ we conclude that this eigenvalue is, also, the dominant eigenvalue of the matrix

$$W_n^{1/2} \tilde{G}_n W_n^{-1/2} = (W_n^{1/2} T_n W_n^{1/2})^T (W_n^{1/2} T_n W_n^{1/2}) = \tilde{A}_n^T \tilde{A}_n.$$

Hence, we may actually calculate the *norm* $\|\tilde{A}_n\|$ of the transformed matrix

$$\tilde{A}_n = W_n^{1/2} T_n W_n^{1/2} = \left( \frac{\sqrt{w_j w_k}}{(c_j + c_k - 1)(c_j + c_k)/2 - (c_k - 1)} \right)_{j,k=1,\ldots,n}. \tag{3.18}$$

We conjecture, and the results of §3.4 as well as our numerical experiments confirm it, that

$$\lim_{n \to \infty} \|\tilde{A}_n\| = \|A\|,$$

with the same order of approximation as the underlying summation formula (3.15). A proof of this fact is an open problem that we leave as a challenge to the reader.

**Strebel's Summation Formula.** To get a working algorithm we need a summation formula that is of higher order than simple truncation at $n$, which, for the problem at hand, is of order $O(n^{-3})$. We will follow the ideas that were communicated to us by Rolf Strebel and construct a method that is, at least for the evaluation of $\zeta(4)$, *provably* of order $O(n^{-7})$.

A summation formula of the type (3.15) should shorten the range of indices from $1, \ldots, \infty$ to $1, \ldots, n$ without introducing too much error. In the case of an *integral*, a substitution by means of a one-to-one mapping $\phi : [1, n) \to [1, \infty)$ would do the job *exactly*,

$$\int_1^\infty f(x)\, dx = \int_1^n \phi'(\xi)\, f(\phi(\xi))\, d\xi.$$

If we view the sum as a kind of "approximation" to the integral, we might try

$$\sum_{k=1}^{\infty} f(k) \approx \sum_{k=1}^{n-1} \phi'(k)\, f(\phi(k)).$$

Now, the precise relation between equally spaced sums and integrals is described by the following theorem. For the convenience of presentation we have shifted the lower index of summation to $k = 0$.

**Theorem 3.4 (Euler–Maclaurin sum formula [Hen77, §11.11]).** *Let $n, m$ be positive integers and $f$ a function that is $2m$ times continuously differentiable on the interval $[0, n]$. Then*

$$\sum_{k=0}^{n} f(k) = \frac{1}{2}\big(f(0)+f(n)\big)+\int_0^n f(x)\,dx+\sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}\big(f^{(2k-1)}(n)-f^{(2k-1)}(0)\big)+R_{2m}$$

*with the remainder bounded by*

$$|R_{2m}| \leqslant \frac{|B_{2m}|}{(2m)!}\int_0^n |f^{(2m)}(x)|\,dx.$$

*Here, the quantities $B_{2k}$ denote the Bernoulli numbers.*

The terms in this formula that appear in addition to the sum $\sum_{k=0}^{n} f(k)$ and the desired integral $\int_0^n f(x)\,dx$ will restrict the possible choices of a suitable mapping $\phi$. One such possibility is given in the next lemma.

**Lemma 3.5 (R. Strebel).** *Let $n$ be a positive integer and $f_n(x) = (x+n)^{-\alpha}$, $\alpha > 1$. The function*

$$\phi_n(\xi) = \frac{n^{1+\beta}(n-\xi)^{-\beta}}{\beta} - \frac{n}{\beta} - \frac{(1+\beta)\xi^2}{2n}, \qquad \beta = \frac{6}{\alpha-1},$$

*is strictly increasing and maps $[0, n)$ onto $[0, \infty)$. Then*

$$\sum_{k=0}^{\infty} f_n(k) = \sum_{k=0}^{n-1} \phi_n'(k)\cdot f_n(\phi_n(k)) + O(n^{-3-\alpha}).$$

***Proof.*** We write $\tilde{f}_n(\xi) = \phi_n'(\xi)\cdot f_n(\phi_n(\xi))$ for short. The mapping $\phi_n$ was chosen such that, as $\xi \to 0$,

$$\phi_n(\xi) = \xi + O(\xi^3).$$

Therefore

$$\tilde{f}_n(0) = f_n(0), \qquad \tilde{f}_n'(0) = f_n'(0).$$

A short calculation shows that

$$\tilde{f}_n'''(0) = c_{1,\alpha}\, n^{-3-\alpha}, \qquad f_n'''(0) = c_{2,\alpha}\, n^{-3-\alpha},$$

with some constants $c_{1,\alpha}$ and $c_{2,\alpha}$ that depend on $\alpha$ only. Moreover, $\phi_n$ was also chosen such that, as $\xi \to n$,

$$\tilde{f}_n(\xi) = c_{3,\alpha}(n-\xi)^5 + o((n-\xi)^5),$$

with a constant $c_{3,\alpha}$. Hence

$$\tilde{f}_n(n) = \tilde{f}_n'(n) = \tilde{f}_n''(n) = \tilde{f}_n'''(n) = \tilde{f}_n^{(4)}(n) = 0.$$

Finally, we observe that

$$\int_0^\infty |f_n^{(4)}(x)|\,dx = O\left(\int_0^\infty (x+n)^{-4-\alpha}\,dx\right) = O(n^{-3-\alpha})$$

and

$$\int_0^n |\tilde{f}_n^{(4)}(\xi)|\,d\xi \leqslant n \cdot \max_{\xi\in[0,n]} |\tilde{f}_n^{(4)}(\xi)| = n \cdot O(n^{-4-\alpha}) = O(n^{-3-\alpha}).$$

If we apply the Euler–Maclaurin sum formula twice, we thus obtain

$$\sum_{k=0}^{\infty} f_n(k) = \frac{1}{2}f_n(0) + \int_0^\infty f_n(x)\,dx - \frac{B_2}{2!}f_n'(0) + O(n^{-3-\alpha})$$

$$= \frac{1}{2}\tilde{f}_n(0) + \int_0^n \tilde{f}_n(\xi)\,d\xi - \frac{B_2}{2!}\tilde{f}_n'(0) + O(n^{-3-\alpha}) = \sum_{k=0}^{n-1} \tilde{f}_n(k) + O(n^{-3-\alpha}),$$

which is the assertion. □

We arrive at *Strebel's summation formula* for $f(x) = x^{-\alpha}$, $\alpha > 1$, which is of the desired form

$$\sum_{k=1}^{\infty} f(k) = \sum_{k=1}^{n} w_k \cdot f(c_k) + O(n^{-3-\alpha}), \tag{3.19}$$

by splitting, with $m = \lceil n/2 \rceil$, $\sum_{k=1}^{\infty} f(k) = \sum_{k=1}^{m-1} f(k) + \sum_{k=0}^{\infty} f_m(k)$ and applying Lemma 3.5 to the second term. Then, the weights are

$$w_k = \begin{cases} 1, & 1 \leqslant k \leqslant m-1, \\ \phi_m'(k-m), & m \leqslant k \leqslant 2m-1, \end{cases} \tag{3.19-1}$$

and the sampling points are

$$c_k = \begin{cases} k, & 1 \leqslant k \leqslant m-1, \\ m + \phi_m(k-m), & m \leqslant k \leqslant 2m-1. \end{cases} \tag{3.19-2}$$

If $n = 2m$, we additionally define $w_n = 0$. In Appendix C.3.2 the reader will find an implementation as a Matlab procedure that is called by

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

**Table 3.4.** *Values of $\|\tilde{A}_n\|$ for two different summation formulas.*

| $n$ | $\|\tilde{A}_n\|$ with (3.19) | $\|\tilde{A}_n\|$ with (3.20) |
|---|---|---|
| 4 | **1.2**84206027130483 | **1.2**19615351739390 |
| 8 | **1.274**196943864618 | **1.2**63116205917547 |
| 16 | **1.2742**23642340573 | **1.2742**07431536352 |
| 32 | **1.274224**147506024 | **1.27422415**2001268 |
| 64 | **1.274224152**770219 | **1.274224152821228** |
| 128 | **1.274224152820**767 | **1.274224152821228** |
| 256 | **1.2742241528212**24 | **1.274224152821228** |
| 512 | **1.274224152821228** | **1.274224152821228** |
| 1024 | **1.274224152821228** | **1.274224152821228** |

We test it on the evaluation of $\zeta(4)$, using $n = 2$, 20, and 200:

**A Matlab session.**

```
>> zeta4 = pi^4/90;
>> alpha = 4; f = inline('1./x.^alpha','x','alpha');
>> error = [];
>> for n = [2,20,200]
>>     [w,c] = SummationFormula(n,alpha);
>>     error = [error; abs(zeta4 - w*f(c,alpha)')];
>> end
>> error

error = 8.232323371113792e-002
        1.767847579436932e-008
        1.554312234475219e-015
```

The result nicely reflects that the error is, for $\alpha = 4$, of order $O(n^{-7})$.

**Application to Problem 3.** Now that we have a good summation formula in hand, we take the transformed matrix $\tilde{A}_n$, as defined in (3.18), and calculate its norm for various $n$ using the power method of §3.3, applied here to $\tilde{A}_n^T \tilde{A}_n$ instead of $A_n^T A_n$. The results of a run[12] in Matlab up to dimension $n = 1024$ are shown in the second column of Table 3.4; the run time was less than a second. The data are consistent with the conjectured $O(n^{-7})$ order of approximation. The reduction of the dimension is remarkable: whereas $\|A_{32768}\|$ gives 13 correct digits only, $\|\tilde{A}_{512}\|$ is correct to 16 digits — even if evaluated in the realm of IEEE double-precision arithmetic.

[12]The code can be found at the web page for this book.

**An exponential summation formula.** For much higher accuracy, the $O(n^{-7})$ order of approximation is yet not good enough. Following the ideas of the proof of Lemma 3.5 one might try to eliminate all terms $B_{2k} f^{(2k-1)}(0)/(2k)!$ in the Euler–Maclaurin sum formula for $\sum_{k=0}^{\infty} f(k)$ by using a transformation

$$\tilde{f}_n(\xi) = (1 + n^{-1}\phi'_{\exp}(\xi/n))\, f(\xi + \phi_{\exp}(\xi/n)),$$

where $\phi_{\exp}(\xi)$ is a function that vanishes with all its derivatives at $\xi = 0$ and which grows fast to infinity for $\xi \to 1$. In fact, we can then show that

$$\sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{n-1} (1 + n^{-1}\phi'_{\exp}(k/n))\, f(k + \phi_{\exp}(k/n)) + R_{2m} - \tilde{R}_{2m}, \tag{3.20}$$

where $R_{2m}$ and $\tilde{R}_{2m}$ denote the remainder terms of the Euler–Maclaurin formulas for the two sums. The point is, that the summation formula (3.20) holds for *all* $m$. An analysis of the error term $R_{2m} - \tilde{R}_{2m}$ would try to find a particular index $m_n$, depending on $n$, that minimizes the error. However, such an analysis turns out to be very involved, even for the simple function $f(x) = (x+1)^{-\alpha}$, $\alpha > 1$. Strebel did some numerical experiments and obtained very promising results for the particular choice

$$\phi_{\exp}(u) = \exp\left(\frac{2}{(1-u)^2} - \frac{1}{2u^2}\right). \tag{3.20-1}$$

In Appendix C.3.2 the reader will find an implementation as a Matlab procedure that is called by

```
[w,c] = SummationFormula(n,'exp').
```

Indeed, for Problem 3 this formula is a further improvement, even in IEEE double-precision arithmetic. The results of a run[13] in Matlab up to dimension $n = 1024$ are shown in the third column of Table 3.4; the run time was once more less than one second. To get 16 correct digits, we need go only to dimension $n = 64$, which takes under 0.1 second.

We have implemented the summation formula in PARI/GP and applied it to Problem 3 using high-precision arithmetic.[13] Table 3.5 shows the number of correct digits for various $n$; "correctness" was assessed by comparing with the result obtained for $n = 1200$. Though the formula performs with remarkable success — we get 25 digits in less than 2 seconds as compared to 8 hours for 21 digits in §3.3 — the convergence rate slows down for larger $n$. In the next section we will turn to summation formulas derived from complex analysis that enjoy *reliable* exponential convergence rates.

## 3.6 Summation Formulas: Complex Analysis

In this section we describe a general technique for evaluating sums as contour integrals. Because there are excellent methods known for numerical quadrature, this

[13] The code can be found at the web page for this book.

**Table 3.5.** *Number of correct digits of $\|\tilde{A}_n\|$ with summation formula (3.20);* `dec` *is the mantissa length of the high-precision arithmetic.*

| $n$ | `dec` | no. of correct digits | run time |
|---|---|---|---|
| 100 | 30 | 25 | 1.9 sec |
| 200 | 50 | 40 | 13 sec |
| 400 | 75 | 55 | 92 sec |
| 600 | 75 | 66 | 3.5 min |
| 800 | 100 | 75 | 10 min |
| 1000 | 100 | 82 | 27 min |

will result in a very efficient algorithm to deal with the infinite sums (3.13) that have emerged in the application of the power method to the infinite matrix $A$.

The technique is a consequence of the residue theorem of complex analysis and is therefore restricted to sums of terms that depend analytically on the index. The process is better known in the opposite direction: the evaluation of a contour integral by a sum of residues. Summation by contour integration has earlier been used by Milovanović [Mil94], among others. A particularly useful result of this approach to summation, general enough to serve our purposes, is given by the following theorem.

**Theorem 3.6.** *Let $f(z)$ be a function that is analytic in a domain of the complex plane. Further, suppose that for some $\alpha > 1$, $f(z) = O(z^{-\alpha})$ as $z \to \infty$. Let $\mathcal{C}$ be a contour in the domain of analyticity of $f$ that is symmetric with respect to the real axis, passes from infinity in the first quadrant into the fourth quadrant, and has, on its left (with respect to its orientation), all the positive integers, no other integers, and no boundary points of the domain of analyticity of $f$. Then*

$$\sum_{k=1}^{\infty} f(k) = \frac{1}{2\pi i} \int_{\mathcal{C}} f(z) \cdot \pi \cot(\pi z)\, dz, \tag{3.21}$$

$$\sum_{k=1}^{\infty} (-1)^k f(k) = \frac{1}{2\pi i} \int_{\mathcal{C}} f(z) \cdot \pi \csc(\pi z)\, dz. \tag{3.22}$$

***Proof.*** [14] Consider the 1-periodic meromorphic function $\pi \cot(\pi z)$, whose poles are all simple and located at the integers $z = n$, $n \in \mathbb{Z}$. From the Laurent series at $z = 0$, namely

$$\pi \cot(\pi z) = \frac{1}{z} - \frac{\pi^2}{3} z + O(z^3),$$

we conclude that the residues are all 1.

[14] We restrict ourselves to (3.21). The formula (3.22) for an alternating series can be proved in analogy, if one notes that at the pole $z = n$, $n \in \mathbb{Z}$, the residue of the function $\pi \csc(\pi z)$ is $(-1)^n$.

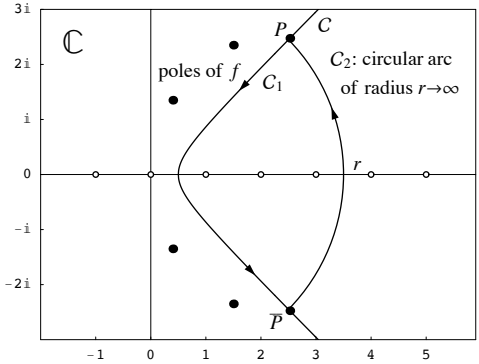


**Figure 3.1.** *Contours in the proof of Theorem 3.6 ($n = 3$).*

Let $\mathcal{C}_1$ be the arc of the contour $\mathcal{C}$ between a point $P$ and its complex conjugate $\bar{P}$; let $\mathcal{C}_2$ be the circular arc of radius $r$ joining $\bar{P}$ and $P$ counterclockwise (see Figure 3.1). If we restrict ourselves to the radii $r_n = n + 1/2$, $n \in \mathbb{N}$, the function $\pi \cot(\pi z)$ remains uniformly bounded when $|z| = r_n$, $z \in \mathbb{C}$, and $n \to \infty$. Now, the residue theorem [Hen74, Thm. 4.7a] gives

$$\sum_{k=1}^{n} f(k) = \frac{1}{2\pi i} \int_{\mathcal{C}_1} f(z) \cdot \pi \cot(\pi z)\, dz + \frac{1}{2\pi i} \int_{\mathcal{C}_2} f(z) \cdot \pi \cot(\pi z)\, dz.$$

Passing to the limit $n \to \infty$ yields the conclusion, because

$$\left| \int_{\mathcal{C}_2} f(z) \cdot \pi \cot(\pi z)\, dz \right| = \text{length}(\mathcal{C}_2) \cdot O(r_n^{-\alpha}) = O(r_n^{1-\alpha}) \to 0.$$

Here we have used the asymptotic decay $f(z) = O(z^{-\alpha})$, $\alpha > -1$. □

Theorem 3.6 enables us to transform infinite sums into integrals. We first choose an appropriate contour $\mathcal{C}$, parametrized by the real variable $t$ through the complex-valued function $Z(t)$. Since we are working with tools of complex analysis, it is best to use contours $\mathcal{C}$ that are *analytic curves*, that is, the function $Z$ must itself be an analytic function of the real variable $t$. For the ease of numerical evaluation we prefer curves that can be expressed in terms of elementary functions. Using the parametrization, the integral in (3.21) transforms to

$$\sum_{k=1}^{\infty} f(k) = \int_{-\infty}^{\infty} F(t)\, dt, \qquad F(t) = \frac{1}{2\,i} f(Z(t)) \cdot \cot(\pi Z(t)) \cdot Z'(t). \tag{3.23}$$

**Example.** Let us transform the sum (3.14), which defines $\zeta(4)$, into an integral. We choose the contour $\mathcal{C}$ parametrized through the function $Z(t) = (1 - it)/2$, apply Theorem 3.6 to the function $f(z) = z^{-4}$, and obtain

$$\zeta(4) = \frac{1}{2\pi i} \int_{\mathcal{C}} \frac{\pi \cot(\pi z)}{z^4}\, dz = 16 \int_{-\infty}^{\infty} \frac{t(1 - t^2) \tanh(\pi t/2)}{(1 + t^2)^4}\, dt. \tag{3.24}$$

Any numerical quadrature method used to approximate the integral (3.23) will result in a summation formula for $f$ of the form (3.15). However, because of the intermediate function $Z$, the weights and sampling points will now be complex.

Note that once the contour $\mathcal{C}$ has been chosen, the parametrization of $\mathcal{C}$ can still be modified by using a new parameter $\tau$ that is related to $t$ by an analytic transformation $t = \Phi(\tau)$ with $\Phi'(\tau) > 0$. This additional flexibility is the main advantage of the summation by contour integration over the real-analysis method of §3.5. A proper choice of the parametrization will help us reduce the number of terms in the resulting summation formula considerably.

### 3.6.1 Approximation of Contour Integrals by Trapezoidal Sums

We approximate the integral $S = \int_{-\infty}^{\infty} F(t)\,dt$ in (3.23) by its trapezoidal sum $T(h)$ of step size $h > 0$,

$$T(h) = h \sum_{j=-\infty}^{\infty} F(j \cdot h).$$

In many texts on numerical analysis the trapezoidal rule is treated as the "ugly duckling" among algorithms for approximating definite integrals. It turns out, however, that the trapezoidal rule, simple as it is, is among the *most powerful* algorithms for the numerical quadrature of analytic functions. Among the first who pointed out that exceptional behavior of the trapezoidal rule for infinite intervals were Milne (in an unpublished 1953 note, see [DR84, p. 212]) and Bauer, Rutishauser, and Stiefel [BRS63, pp. 213–214]. Later Schwartz [Sch69] and Stenger [Ste73] applied the trapezoidal rule to more general analytic integrals; see also the book by Davis and Rabinowitz [DR84, §3.4]. We also mention the Japanese school starting with the work of Iri, Moriguti, and Takasawa [IMT70], which is also based on the trapezoidal rule and is now known as the IMT method. Their ideas were further developed into the *double-exponential formulas* by Takahasi and Mori [TM74] and by Mori [Mor78]. Full generality for handling analytic integrals is achieved by combining the trapezoidal rule (applied to integrals over $\mathbb{R}$) with analytic transformations of the integration parameter (see [Sch89, Chap. 8]). Applications to multidimensional integrals over Cartesian-product domains (rectangles, strips, quadrants, slabs, etc.) are discussed in [Wal88].

**Truncation of trapezoidal sums.** The *infinite* sums $T(h)$ are inherently difficult to compute if the integrand decays too slowly, such as $O(|t|^{-\alpha})$ as $t \to \pm\infty$. An obvious problem is the large number of terms that would have to be included in the sum; more serious, however, is the estimation of the remainder of a truncated sum.

Typically such a sum is truncated by disregarding all terms that are considered "too small". Let us formalize this idea by specifying a threshold or truncation tolerance $\epsilon > 0$ and define the truncated trapezoidal sum

$$T_\epsilon(h) = h \sum_{j \in \mathbb{Z}:\, |F(j \cdot h)| \geqslant \epsilon} F(j \cdot h). \tag{3.25}$$

Then an approximation of or a useful bound on the remainder $R_\epsilon(h) = T(h) - T_\epsilon(h)$ is needed. To get a rough idea of the principal problems involved, we consider, as a model, the truncation of the integral

$$\int_1^\infty t^{-\alpha}\,dt, \quad \alpha > 1,$$

at the threshold $\epsilon$, that is, at the point of integration $t_\epsilon = \epsilon^{-1/\alpha}$. The remainder is

$$R_\epsilon = \int_{t_\epsilon}^\infty t^{-\alpha}\,dt = \frac{\epsilon^{(\alpha-1)/\alpha}}{\alpha - 1},$$

which can be much larger than the truncation tolerance $\epsilon$. For instance, we obtain $R_\epsilon = \sqrt{\epsilon}$ for $\alpha = 2$. Therefore, in the case of slowly decaying integrands the truncation by a threshold does not lead to accurate results. Instead, we propose to enhance the decay of the integrand by introducing a new integration variable. For integrals along the real axis the transformation

$$t = \sinh(\tau), \qquad dt = \cosh(\tau)\,d\tau, \tag{3.26}$$

proves to be appropriate. The integral becomes

$$S = \int_{-\infty}^\infty G(\tau)\,d\tau, \qquad G(\tau) = F(\sinh(\tau))\cosh(\tau).$$

If $F(t)$ decays as a power of $t$, $G(\tau)$ decays *exponentially*. Consider, again as a model, the truncation by a threshold $\epsilon$ of a typical integral with an exponentially decaying integrand,

$$\int_0^\infty ae^{-\alpha\tau}\,d\tau, \quad \alpha > 0,\ a > 0.$$

The point of truncation is $t_\epsilon = \log(a/\epsilon)/\alpha$, and the remainder is

$$R_\epsilon = \int_{t_\epsilon}^\infty ae^{-\alpha\tau}\,d\tau = \epsilon/\alpha.$$

Thus, we see that a truncation tolerance $\epsilon = \alpha \cdot \texttt{tol}$ suffices to accumulate the trapezoidal sum of an exponentially decaying integrand to the accuracy `tol`.

The reader will find a Matlab implementation of these ideas as the routine

`TrapezoidalSum(f,h,tol,level,even)`

in Appendix C.3.2. The routine assumes that the integrand `f` decays monotonically in the tail where the threshold `tol` applies. The nonnegative integer `level` tells the routine how often the sinh transformation (3.26) has to be applied recursively. We will call the method used with `level` = 1 a *single-exponential formula*, with `level` = 2 a *double-exponential formula*. If the switch `even` is set to `'yes'`, only half of the sum is accumulated for symmetry reasons. For numerical accuracy, the sum is accumulated from the smaller to the larger values (assuming monotonicity). Therefore the terms have to be stored until the smallest one has been computed.

**Discretization Error.** The error theory of the trapezoidal rule is ultimately connected to the Fourier transform of $F$:

$$\hat{F}(\omega) = \int_{-\infty}^{\infty} F(t) e^{-i\omega t}\, dt.$$

In fact, the *Poisson summation formula* [Hen77, Thm. 10.6e] expresses $T(h)$ as a corresponding trapezoidal sum, with step size $2\pi/h$, of the Fourier transform $\hat{F}$:

$$T(h) = h \sum_{j=-\infty}^{\infty} F(j \cdot h) = \sum_{k=-\infty}^{\infty} \hat{F}\left(k \cdot \frac{2\pi}{h}\right).$$

Note that the sum over $k$ has to be interpreted as a *principal value*, that is, as the limit of the sum from $-N$ to $N$ as $N \to \infty$. This, however, becomes relevant only if $\hat{F}$ decays slowly. Now, we observe that the term $k = 0$ of the trapezoidal sum of the Fourier transform is the integral at hand

$$\hat{F}(0) = \int_{-\infty}^{\infty} F(t)\, dt = S.$$

Therefore, the Poisson summation formula yields the error formula

$$E(h) = T(h) - S = \hat{F}(2\pi/h) + \hat{F}(-2\pi/h) + \hat{F}(4\pi/h) + \hat{F}(-4\pi/h) + \dots . \quad (3.27)$$

Hence, the decay rate of $E(h)$ for $h \to 0$ is determined by the asymptotic behavior of the Fourier transform $\hat{F}(\omega)$ as $\omega \to \pm\infty$. In many specific cases this asymptotics can be found by the saddle point method (method of steepest descents), see, e.g., [Erd56] or [Olv74] for the theory, or [GW01, pp. 495ff] for a worked example.

The error formula (3.27) yields an especially nice and simple result if we assume that $F$ is analytic in the strip $|\mathrm{Im}(t)| < \gamma_*$, $\gamma_* > 0$, and that $F(x + iy)$ is integrable with respect to $x$, uniformly in $|y| \leqslant \gamma$ for any $\gamma < \gamma_*$. Then the modulus of the Fourier transform is known to decay exponentially [RS75, Thm. IX.14]:[15]

$$|\hat{F}(\omega)| = O\left(e^{-\gamma|\omega|}\right) \qquad \text{as} \quad \omega \to \pm\infty,$$

where $0 < \gamma < \gamma_*$ can be chosen arbitrarily. If we plug this into (3.27) we obtain the error estimate[16]

$$E(h) = O\left(e^{-2\pi\gamma/h}\right) \qquad \text{as} \quad h \to 0, \quad (3.28)$$

that is, we get *exponential* convergence: halving the step size results in doubling the number of correct digits. Note that exponential convergence is much better than the higher-order convergence $O(h^{2m})$ achieved in Romberg integration with $m - 1$ steps of Richardson extrapolation applied to the trapezoidal rule (see Appendix A, p. 238).

[15] The more difficult $L^2$-version of this result is one of the classic Paley–Wiener theorems [PW34, §3, Thm. IV], stated in their seminal monograph on Fourier transforms in the complex domain.

[16] For a different proof see [DR84, p. 211].

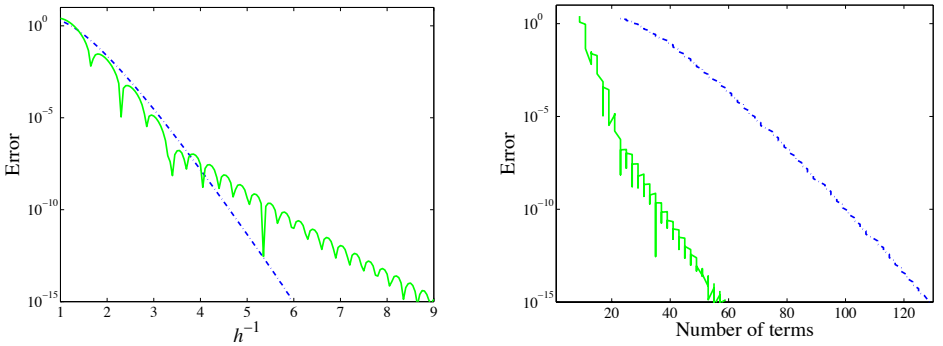


**Figure 3.2.** *Error of trapezoidal sum vs.* $1/h$ *(left), and vs. the number of terms (right). Results are shown for a single-exponential (dashed line) and a double-exponential (solid line) formula.*

**Example.** Let us illustrate the use of the truncated trapezoidal sum (3.25) and the sinh transformation (3.26) for the integral (3.24), which resulted from the application of the method of contour integration to the sum defining $\zeta(4)$. Since the integrand decays slowly as $O(|t|^{-5})$ as $t \to \pm\infty$, we have to apply the sinh transformation (3.26) at least once. We start with the single-exponential formula (`level` $= 1$), using the step sizes $h = 0.3$ and $h = 0.15$.

**A Matlab session.**

```
>> f = inline('z^(-4)','z');
>> Z = inline('1/2-i*t/2','t');
>> dZ = inline('-i/2','t');
>> F_Summation = inline('real(f(Z(t))*cot(pi*Z(t))*dZ(t)/2/i)',...
>>     't','f','Z','dZ');
>> tol = 1e-16; level = 1; even = 'yes'; s = [];
>> for h = [0.3 0.15]
>>     s = [s;
>>      TrapezoidalSum(F_Summation,h,tol,level,even,f,Z,dZ)];
>> end
>> s

s = 1.082320736262448e+000
    1.082323233711138e+000
```

For $h = 0.3$, using 73 terms of the trapezoidal sum, 6 digits of the result are correct; whereas all 16 digits are correct for $h = 0.15$ using 143 terms of the trapezoidal sum. The errors of several runs with $h$ in the range from 1 to $1/6$ are shown (dashed line) in the left part of Figure 3.2; they nicely reflect the exponential convergence. The corresponding number of terms used in the truncated trapezoidal sums are shown in the right part of Figure 3.2.

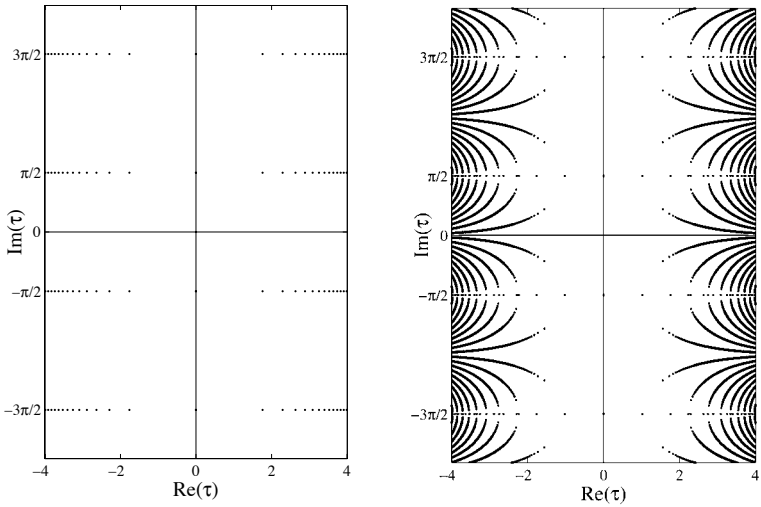


**Figure 3.3.** *Proliferation of singularities in the parameter plane for a single-exponential (left) and a double-exponential (right) formula. There are many more singularities outside the rectangles shown.*

A look at the right integral in (3.24) shows that the singularities of the integrand are, *before* the sinh transformation, located at $(2k+1)i$, $k \in \mathbb{Z}$. *After* the transformation the singularities $\tau_s$ of the integrand satisfy

$$\sinh(\tau_s) = (2k+1)i, \qquad \text{which implies} \qquad \operatorname{Im}(\tau_s) = \frac{(2m+1)\pi\, i}{2}$$

for some $m \in \mathbb{Z}$; see the left part of Figure 3.3. Hence, the transformed integrand is analytic in the strip $|\operatorname{Im}(\tau)| < \gamma_*$ with $\gamma_* = \pi/2$ and the error estimate (3.28) specializes to $E(h) = O(e^{-\beta/h})$, for all $\beta < \pi^2$. Thus, an increase of $1/h$ by, say, 1 yields, in the asymptotic regime, a gain in accuracy of approximately $\pi^2/\log(10) \doteq 4$ digits. Indeed, the dashed line in the left part of Figure 3.2 has a slope of approximately $-4$ in its linear asymptotic regime.

It is tempting to enhance the decay of the integrand by a double-exponential parametrization, that is, by applying the sinh transformation (3.26) once more.[17] This would simplify the computation of the trapezoidal sums by making the number of terms in the truncated sum $T_\epsilon(h)$ essentially independent of the threshold $\epsilon$. Such a repeated application of the sinh transformation has to be used with care, however.

The right part of Figure 3.3 shows the singularities of the integrand of the example at hand after such a second sinh transformation. We observe that the integrand is no longer analytic in a strip containing the real axis. In contrast, the domain of analyticity has now the shape of a "funnel" of exponentially decaying width. Although the formula (3.28) for the discretization error is not applicable, integrands with such funnel domains of analyticity can still be useful: for the integral

[17] To do so, one puts `level` $= 2$ in the Matlab session of p. 71.

at hand, the solid line in the left part of Figure 3.2 shows that the convergence is apparently still exponential, even if the decay has slowed down. Therefore, smaller step sizes have to be used to get the same accuracy as with the singly transformed integrand. This potential increase of the computational effort is compensated by the much faster decay of the integrand, which allows much earlier truncation of the trapezoidal sum. So let us compare the two methods with respect to the number of terms that were actually used: the right part of Figure 3.2 shows that, at the same level of accuracy, the double application of the sinh transformation requires fewer terms by a factor of 2.5. Thus the computational cost of the sinh transformations is about the same for both the single-exponential and the double-exponential formula. However, the double-exponential formula saves the factor of 2.5 in the evaluations of the original integrand. This will become a major advantage in the final solution of Problem 3, for which the transformations (and the transcendentals of the summation formula) are applied only once, but a large number of integrals has to be calculated during the power iteration.

**Summary.** For the convenience of the reader we finally write down the summation formula of the type (3.15) that is obtained by applying the trapezoidal rule to the contour integral (3.23). If we choose a contour parametrized through $Z(t)$, a reparametrization $\Phi(\tau)$, a step size $h$, and a truncation point $T$ we get

$$\sum_{k=1}^{\infty} f(k) \approx \sum_{k=-m}^{m} w_k f(c_k), \qquad m = \lfloor T/h \rfloor,$$

with the (complex-valued) sampling points and weights

$$c_k = Z(\Phi(kh)), \qquad w_k = \frac{h}{2i} \cot(\pi c_k) \cdot Z'(\Phi(kh)) \cdot \Phi'(kh).$$

The discussion so far has shown that for certain $f$ there are suitable choices that make the error exponentially small in the number of terms, that is $n = 2m + 1$.

### 3.6.2 Application to Problem 3

As in §3.5, having a good summation formula in hand, we can use the power iteration in the form (3.16) to approximate $\|A\|^2$. That is, we calculate the dominant eigenvalue of the transformed matrix $\tilde{G}_n$, which we defined in (3.17).[18] To make the method work we need only choose a contour, a reparametrization, and a truncation point. Then we apply the procedure for various step sizes.

**Choosing the contour.** The sample points enter the transformed power iteration (3.16) via the expression $a(c_j, c_k)$ in a twofold operational sense: as a means to

[18] As in §3.5 we can argue that this value is also the dominant eigenvalue of the matrix $\tilde{A}_n^T \tilde{A}_n$, where the transformed matrix $\tilde{A}_n$ is defined in (3.18). However, because now $\tilde{A}_n$ has *complex* entries, it is *not* the case that this eigenvalue is equivalently given by $\|\tilde{A}_n\|^2$ — a value that in the complex case is equal to the dominant eigenvalue of $\tilde{A}_n^H \tilde{A}_n$ instead.

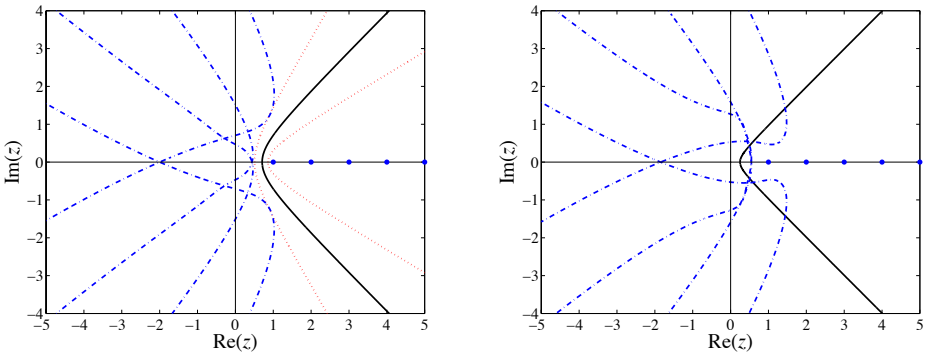


**Figure 3.4.** *Loci of poles (dashed lines) for the contour (3.29) (solid line) with a good choice of the parameter $\sigma = 1/\sqrt{2}$ (left) and a bad choice $\sigma = 1/4$ (right). The dotted lines on the left show the boundaries of the image $z = Z(t)$ (with $\sigma = 1/\sqrt{2}$) of the strip $|\mathrm{Im}(t)| \leqslant 0.25$ in the parameter plane.*

efficient summation as well as a new index for the resulting vectors. Therefore the poles of the underlying analytic functions depend on the contour itself. In fact, these poles lie on the loci given by the zeros $v(t)$ and $u(t)$ of the denominator $b(u+v)-v$ of $a(u,v)$, as given by (3.7), while one of the variables is running through $Z(t)$:

$$b(u(t) + Z(t)) - Z(t) = 0, \qquad b(Z(t) + v(t)) - v(t) = 0.$$

These are quadratic equations and we obtain four connected branches for the loci of the poles (see Figure 3.4).

Because of the dependence of the poles on the chosen contour, the simplest such choice, which was good for our model $\zeta(4) = \sum_{k=1}^{\infty} k^{-4}$, namely $Z(t) = \sigma - it$ with $0 < \sigma < 1$, does not satisfy the conditions of Theorem 3.6. Instead we use the right branch of a rectangular hyperbola $\mathrm{Re}(z^2) = \sigma^2$, suitably parametrized to yield exponentially decaying integrands for the trapezoidal rule:

$$Z(t) = \sigma(\cosh(t) - i\sinh(t)), \qquad Z'(t) = -i \cdot \overline{Z(t)}. \tag{3.29}$$

The particular choice $\sigma = 1/\sqrt{2}$ yields the equivalent representation $Z(t) = \cosh(t - i\pi/4)$; the loci of poles are shown on the left part of Figure 3.4 and satisfy the conditions of Theorem 3.6. As shown on the right part of Figure 3.4, the choice of the parameter $\sigma = 1/4$ leads to intersections of the loci of poles with the contour. Such a choice is not admissible.

**Choosing the parametrization.** With $\sigma = 1/\sqrt{2}$, and without further reparametrization, the domain of analyticity extends to the strip $|\mathrm{Im}(t)| < \gamma_*$, where $\gamma_* \approx 0.25$ as can be read off from the dotted lines on the left of Figure 3.4. Therefore, we expect errors that are exponentially small in the reciprocal step size $h^{-1}$. Although practical for low accuracies, such as calculations in IEEE double precision, this

**Table 3.6.** *Values of* $\lambda_{\max}^{1/2}(\tilde{G}_n)$*, truncation tolerance* $\epsilon = 10^{-16}$*.*

| $h$ | $n$ | $\sigma = 1/\sqrt{2}$ | $\sigma = 1/4$ |
|---|---|---|---|
| 0.64 | 11 | **1.**165410725532445 | $9.202438226420232 \cdot 10^{-17}$ |
| 0.32 | 21 | **1.2**52235875316941 | $5.344162844425345 \cdot 10^{-17}$ |
| 0.16 | 41 | **1.27**3805685815999 | 2.445781134567926 |
| 0.08 | 83 | **1.274224**137562002 | 1.722079300161066 |
| 0.04 | 167 | **1.274224152821228** | 4.210984571814455 |

parametrization turns out to be rather slow for higher accuracies. Some experimentation led us to the particular choice

$$t = \Phi(\tau) = \tau + \frac{\tau^3}{3}. \tag{3.30}$$

**Choosing the truncation point.** From §3.5 we know that the decay rate of the terms of the sums in the power iteration is the same as for $\zeta(4) = \sum_{k=1}^{\infty} k^{-4}$. Thus, with the parametrization (3.30) a good point of truncation for the truncation tolerance $\epsilon$ is at

$$T = \log^{1/3} \epsilon^{-1}. \tag{3.31}$$

**Results in IEEE double-precision arithmetic.** The results of a Matlab run[19] with the choices (3.29) for the contour, (3.30) for the reparametrization, and (3.31) for the truncation point are shown in Table 3.6. The run with the suitable contour parameter $\sigma = 1/\sqrt{2}$, shown in the third column, nicely exhibits the exponential convergence: the number of correct digits doubles if we double the dimension $n$; 16 correct digits were obtained in less than a second. The run with the bad contour parameter $\sigma = 1/4$, shown in the fourth column, proves that it is important to satisfy the conditions of Theorem 3.6: all the digits are garbage. Thus a careful theoretical study turns out to be indispensable for the method at hand.

**Results in high-precision arithmetic.** For experiments with higher accuracies we have implemented this approach in PARI/GP with the above choices (3.29), (3.30), (3.31), and $\sigma = 1/\sqrt{2}$. Because the contour is symmetric with respect to the real axis, the sums (3.16) of the transformed power method also enjoy a symmetry with respect to complex conjugation. Taking advantage of this symmetry allowed us to cut in half the computational effort.[19]

Table 3.7 shows the number of correct digits for various runs; "correctness" was assessed by comparing with a result that was obtained in a month-long computation with a predicted accuracy of 273 digits. Such a prediction can be obtained as follows.

At the end of §3.6.1 we considered reparametrizations that enforce a stronger decay of the integrand and observed that they certainly yield shorter trapezoidal

[19] The code can be found at the web page for this book.

**Table 3.7.** *Number of correct digits of* $\lambda_{\max}^{1/2}(\tilde{G}_n)$ *with* $\epsilon = 10^{2-\mathtt{dec}}$*;* `dec` *is the mantissa length of the high-precision arithmetic.*

| $1/h$ | $n$ | `dec` | no. of correct digits | run time |
|---|---|---|---|---|
| 40 | 321 | 28 | 25 | 9.8 sec |
| 65 | 577 | 38 | 36 | 43 sec |
| 90 | 863 | 48 | 47 | 2.3 min |
| 110 | 1119 | 57 | 56 | 5.0 min |
| 160 | 1797 | 86 | 76 | 32 min |
| 240 | 2991 | 105 | 98 | 2.0 h |
| 384 | 5315 | 144 | 131 | 13 h |
| 640 | 9597 | 183 | 180 | 71 h |

sums, but at the expense of a slower convergence rate caused by the proliferation of singularities. Experiments show that for Problem 3 too, these two effects tend to balance each other in the sense that different parametrizations differ only by a constant factor in the overall computational effort. Now, for the case of no reparametrization the computational effort to obtain $d$ correct digits is easily estimated: the exponential convergence yields the reciprocal step size $h^{-1} = O(d)$, the exponential decay results in the truncation point $T = O(d)$, and the power iteration needs $O(d)$ iterations. Thus, we get the dimension $n = O(T/h) = O(d^2)$ and[20]

$$\text{no. of } d\text{-digit operations} = O\big(\text{no. of power iterations} \cdot n^2\big) = O\big(d^5\big).$$

On the other hand, for the parametrization (3.30) we have the truncation point given in (3.31), namely $T = O(d^{1/3})$. If we assume the same asymptotic operation count $O(d^5)$ as above, we get the reciprocal step size $h^{-1} = O(d^{5/3})$. In fact, the data of Table 3.7 are consistent with this asymptotic behavior; a simple fit yields

$$d \approx 51.4 \sinh\left(\frac{3}{5}\operatorname{arcsinh}\left(\frac{h^{-1}}{47.7}\right)\right). \tag{3.32}$$

This empirical law allows us to predict the accuracy for a given step size $h$.

As efficient as this reliably exponentially convergent method is, the complexity of the problem is still too large to allow us to calculate 10 000 digits, as we have done for each of the other nine problems: the dimension of the approximating problem would be $n \approx 3 \cdot 10^7$. Calculating the dominant eigenvalue of a full matrix of that dimension to 10 000 digits is beyond imagination: infinity is too far away.

[20] Since these operations are basically multiplications, the run time scales, using Karatsuba multiplication, as $O(d^{6.58\cdots})$ and, with an FFT-based fast multiplication, as $O(d^{6+\kappa})$, where $\kappa > 0$ is arbitrary.