AN ALTERNATIVE TO THE EULER–MACLAURIN FORMULA:
APPROXIMATING SUMS BY INTEGRALS ONLY

IOSIF PINELIS

ABSTRACT. The Euler–Maclaurin (EM) summation formula is used in many theoretical studies and numerical calculations. It approximates the sum \( \sum_{k=0}^{n-1} f(k) \) of values of a function \( f \) by a linear combination of a corresponding integral of \( f \) and values of its higher-order derivatives \( f^{(j)} \). An alternative (Alt) summation formula is proposed, which approximates the sum by a linear combination of integrals only, without using high-order derivatives of \( f \). Explicit and rather easy to use bounds on the remainder are given. Possible extensions to multi-index summation are suggested. Applications to summing possibly divergent series are presented. It is shown that the Alt formula will in most cases outperform, or greatly outperform, the EM formula in terms of the execution time and memory use. One of the advantages of the Alt calculations is that, in contrast with the EM ones, they can be almost completely parallelized. Illustrative examples are given. In one of the examples, where an array of values of the Hurwitz generalized zeta function is computed with high accuracy, it is shown that both our implementation of the EM formula and, especially, the Alt formula perform much faster than the built-in Mathematica command \texttt{HurwitzZeta}[].

CONTENTS

1. Introduction 1
2. The EM summation formula 4
3. An alternative (Alt) to the EM formula 5
4. Bounds on the remainders 10
5. Application to summing (possibly divergent) series 11
6. Choosing \( c \) and \( m \) for a desired accuracy 14
7. Parallelization and memory use 18
8. Examples 22
9. Proofs 36
References 42

1. INTRODUCTION

The Euler–Maclaurin (EM) summation formula can be written as follows:

\[
\sum_{k=0}^{n-1} f(k) \approx \int_0^n dx f(x) + \sum_{j=1}^{2m-1} B_j \frac{x^{j-1}}{j!} \left[ f^{(j-1)}(n) - f^{(j-1)}(0) \right],
\]

\( (EM) \)

where \( f \) is a smooth enough function, \( B_j \) is the \( j \)th Bernoulli number, and \( n \) and \( m \) are natural numbers. The remainder/error term of the approximation provided by this formula has a certain explicit integral

\textbf{2010 Mathematics Subject Classification.} Primary 41A35; secondary 26D15, 40A05, 40A25, 41A10, 41A17, 41A25, 41A55, 41A58, 41A60, 65B05, 65B10, 65B15, 65D10, 65D30, 65D32, 68Q17.

\textbf{Key words and phrases.} Euler–Maclaurin summation formula, sums, series, integrals, derivatives, approximation, inequalities, divergent series.
expression, which depends linearly on \( f^{(2m-1)} \); in particular, the EM approximation is exact when \( f \) is a polynomial of degree \(< 2m-1 \). The integral \( \int_0^n dx f(x) \) in \([EM]\) may be considered the main term of the approximation, whereas the summands \( B_j \frac{f^{(j-1)}}{j} (n) - f^{(j-1)}(0) \) may be referred to as the correction terms.

The EM formula has been used in a large number of theoretical studies and numerical calculations, even outside mathematical sciences – see e.g. applications of this formula in problems of renormalization in quantum electrodynamics and quantum field theory \([15\ 2.15.6\text{ and }22\ 3.1.3]\). The EM formula is used, in particular, for numerical calculations of sums in such mathematical software packages as Mathematica (command \$\text{NSum}[]\$ with option \text{Method}`EulerMaclaurin\$), Maple (command \$\text{euler}()\$), PARI/GP (command \$\text{sumsum}()\$), and Matlab (\$\text{numeric::sum}()\$). EM is also used internally in other built-in commands in such software packages requiring calculations of sums.

Clearly, to use the EM formula in a theoretical or computational study, one will usually need to have an antiderivative \( F \) of \( f \) and the derivatives \( f^{(j-1)} \) for \( j = 1, \ldots, 2m-1 \) in tractable or, respectively, computable form.

In this paper, an alternative summation formula (Alt) is offered, which approximates the sum \( \sum_{k=0}^{n-1} f(k) \) by a linear combination of values of an antiderivative \( F \) of \( f \) only, without using values of any derivatives of \( f \):

\[
\sum_{k=0}^{n-1} f(k) \approx \sum_{j=1-m}^{m-1} \tau_{m,1+j} \int_{-1/2-j/2}^{n-1/2-j/2} dx f(x),
\]

(Alt)

where \( f \) is again a smooth enough function, the coefficients \( \tau_{m,r} \) are certain rational numbers not depending on \( f \) and such that \( \sum_{j=1-m}^{m-1} \tau_{m,1+j} = 1 \), and \( n \) and \( m \) are natural numbers. Similarly to the case of the EM formula, the remainder/error term of the approximation provided by the Alt formula has a certain explicit integral expression, which depends linearly on \( f^{(2m)} \); in particular, the Alt approximation is exact when \( f \) is a polynomial of degree \(< 2m \).

Even though the exact expressions of the remainders in the Alt and EM formulas involve higher-order derivatives \( f^{(2m-1)} \) and \( f^{(2m)} \) of the function \( f \), such derivatives need not be computed to compute tight enough bounds on the remainders in typical applications, where the function \( f \) has rather natural analyticity properties; in such cases, in view of Cauchy’s integral formula, it is enough to have appropriate bounds just on the function \( f \) itself.

The coefficients \( \tau_{m,r} \) in the Alt formula are significantly easier to compute than the Bernoulli numbers \( B_j \), which are the coefficients in the EM formula. Moreover, it will be shown (see \( (3.19) \)) that the Bernoulli numbers \( B_j \) can be expressed as certain linear combinations of \( \tau_{m,r} \)’s.

As was noted, to use either the EM formula or the Alt one, one needs to have an antiderivative \( F \) of \( f \) in tractable/computable form anyway. Then \( f \) (being the derivative of \( F \)) will usually be of complexity no less than that of \( F \). On the other hand, the complexity of higher-order derivatives of \( f \) will usually be much greater than that of \( F \) or \( f \). This is the main advantage of the Alt summation formula over the EM one.

Closely related to this is another important advantage of Alt over EM, to be detailed in Subsection 7, the Alt calculations can be almost completely parallelized, whereas the calculation of the derivatives \( f^{(j-1)} \) for \( j = 1, \ldots, 2m-1 \) is non-parallelizable – except for a few special cases when these derivatives are available in simple closed form, rather than only recursively.

Even in such special cases, comparatively least favorable for the Alt formula, it will outperform the EM one when the needed accuracy is high enough. As explained in Subsection 8.4, this will happen because of the comparatively large time needed to compute the Bernoulli numbers.

Summarizing these considerations, we see that the Alt formula should be usually expected to outperform the EM one. Alt’s advantage will usually be the greater, the greater accuracy of the result is required – because then one will need to use a greater number \( \asymp m \) of correction terms \( \frac{B_j}{j} (f^{(j-1)}(n) - f^{(j-1)}(0)) \),
and thus a higher order of the derivatives of \( f \) in the EM formula will be needed. In Section 8, we shall consider specific examples illustrating such general expectations for high-accuracy calculations. (Here and in what follows, for any two positive expressions \( a \) and \( b \), we write \( a \asymp b \) to mean that \( a \leq b \) & \( a \geq b \); in turn, \( a \preceq b \) means \( a \leq C b \) for some constant \( C \), and \( a \succeq b \) means \( b \preceq a \). We also write \( a \sim b \) for \( a/b \to 1 \).)

Concerning very high accuracy, one may ask – as it was done, rhetorically, in the Foreword by D. H. Bailey to [5]: “[...] why should anyone care about finding any answers to 10,000-digit accuracy?” An answer to this question was given in the same Foreword: “[...] recent work in experimental mathematics has provided an important venue where numerical results are needed to very high numerical precision, in some cases to thousands of decimal digits. In particular, precision on this scale is often required when applying integer relation algorithms to discover new mathematical identities. [...] Numerical quadrature (i.e., numerical evaluation of definite integrals), series evaluation, and limit evaluation, each performed to very high precision, are particularly important. These algorithms require multiple-precision arithmetic, of course, but often also involve significant symbolic manipulation and considerable mathematical cleverness as well.” The footnote on the same page in [5] adds this information: “Such algorithms were ranked among The Top 10 [...] of algorithms ‘with the greatest influence on the development and practice of science and engineering in the 20th century.’ ” See also [3] for some specifics on this, as well as [4] for a recent update on [5] – where, in particular, an example is cited, which is now part of Mathematica’s documentation, stating that the largest positive root of Riemann’s prime counting function \( R \) is “shocking[ly]” small, about \( 1.83 \times 10^{-4828} \).

The EM formula was introduced about 282 years ago. Accordingly, a large body of literature has been produced with further developments in the theory and applications of this tool. In contrast, the Alt summation formula appears to have no precedents in the literature. The idea behind the formula and techniques used in its proof also appear to be new. Given the mostly superior performance of the Alt formula in calculations of sums and the already uncovered theoretical connections with the EM formula, new developments in the theory of the Alt formula and its applications do not seem unlikely, and they are certainly welcome.

The rest of this paper is organized as follows.

In Section 2, a rigorous review of the EU formula is given, with an application to the Faulhaber formula for the sums of powers, to be subsequently used in the paper.

In Section 3, the Alt formula is rigorously stated, with discussion. The main idea behind this formula is presented. The mentioned expression of the Bernoulli numbers (which are the coefficients in the EM formula) in terms of the coefficients in the Alt formula is stated as well. Possible extensions of the Alt formula to the case of multi-index sums are suggested.

In Section 4, explicit and rather easy to use bounds on the remainders in the Alt and EM formulas are presented, especially in the case when the function \( f \) has rather natural analyticity properties.

In Section 5 applications of the Alt and EM formulas to summing possibly divergent series are given. It is shown that the corresponding generalized sums, \( \sum_{k \geq 0}^{\text{Alt}} f(k) \) and \( \sum_{k \geq 0}^{\text{EM}} f(k) \) are equal to each other, and that they both differ from the Ramanujan sum \( \sum_{k \geq 0}^{Ra} f(k) \) by the additive constant \( F(0) \), where \( F \) is the chosen antiderivative of \( f \). A simple shift trick is presented, which allows one to make the remainders in the Alt and EM formulas arbitrarily small; this trick is an extension of the method used by Knuth [17] to compute an approximation to the value of the Euler constant.

In Section 6, it is discussed how to choose the number \( m \) “of the correction terms” in the Alt and EM formulas and the value (say \( c \)) of the shift mentioned in the previous paragraph – in order to obtain the desired number \( d \) of digits of accuracy of the (generalized) sum in a nearly optimal time. This is significantly more difficult to do for the EM formula, mainly because it is difficult to assess, especially in general terms, the time needed to compute the values of the higher-order derivatives of \( f \).
A general and rather straightforward way to parallelize the calculations by the Alt formula is detailed in Section 7. The matter of memory use is also considered there. In most cases, the Alt calculations will require substantially less memory than the corresponding EU ones.

To illustrate the above comparisons between the Alt and EM formulas’ performance, four specific examples of a function \( f \) are considered in Section 8, with various levels of the complexity of \( f \), its antiderivative \( F \), and its higher-order derivatives \( f^{(j-1)} \). Measurements of the execution time and memory use for the Alt and EM formulas are presented there for various values of the desired number \( d \) of digits of accuracy. Similar measurements are also presented for the Richardson extrapolation process (REP) in the two of the examples where this kind of extrapolation seems applicable. It appears that the REP cannot compete with either the EM or Alt formula as far as high-accuracy calculations of sums are concerned. That suggests that our code is rather well optimized in that respect.

One should note here that the mentioned comparisons of the performance of the Alt summation with the EM one and the REP concern only problems of summation. Of course, the REP has many other uses in which the Alt formula is not applicable at all. Also, the EM formula may be used to approximate the EM one and the REP concern only problems of summation. Of course, the REP has many other uses in which the Alt formula is not applicable at all. Also, the EM formula may be used to approximate integrals by sums, which cannot be done in general with the Alt formula.

The necessary proofs are deferred to Section 9.

At the end of this introduction, let us fix the notation to be used in the rest of the paper: For any natural number \( \alpha \), let \( C^{\alpha-} \) denote the set of all functions \( f: \mathbb{R} \to \mathbb{R} \) such that \( f \) has continuous derivatives \( f^{(i)} \) of all orders \( i = 0, \ldots, \alpha - 1 \) and the derivative \( f^{(\alpha-1)} \) is absolutely continuous, with a Radon–Nikodym derivative denoted here simply by \( f^{(\alpha)} \). As usual, \( f^{(0)} := f \).

Suppose that \( n \) is a nonnegative integer, \( m \) is a natural number, and \( f \in C^{2m-} \).

2. THE EM SUMMATION FORMULA

Here is an exact form of the formula (EM) stated in the Introduction (see e.g. [10]):

\[
\sum_{k=0}^{n-1} f(k) = A_m^{EM} + R_m^{EM},
\]

where

\[
A_m^{EM} := \int_0^{n-1} dx f(x) + \frac{f(n-1) + f(0)}{2} + \sum_{j=1}^{m-1} \frac{B_{2j}^2}{2j!} [f^{(2j-1)}(n-1) - f^{(2j-1)}(0)],
\]

(2.2)

\( B_j \) is the \( j \)th Bernoulli number, \( R_m^{EM} \) is the remainder given by the formula

\[
R_m^{EM} := \frac{1}{(2m-1)!} \int_0^{n-1} dx f^{(2m-1)}(x) B_{2m-1}(x - \lfloor x \rfloor),
\]

(2.3)

and \( B_j(x) \) is the \( j \)th Bernoulli polynomial, defined recursively by the conditions \( B_0(x) = 1, B'_j(x) = jB_{j-1}(x), \) and \( \int_0^1 dx B_j(x) = 0 \) for \( j = 1, 2, \ldots \) and real \( x \). In particular, for all \( j = 2, 3, \ldots \) the \( j \)th Bernoulli number coincides with the value of the \( j \)th Bernoulli polynomial at 0: \( B_j = B_j(0) \). Here and in what follows, we assume the standard convention, according to which the sum of an empty family is 0. So, the sum in (2.2) equals 0 if \( m = 1 \). It is known that for all real \( x \in [0, 1] \)

\[
|B_{2m-1}(x)| \leq \frac{2(2m-1)!}{(2\pi)^{2m-1}} \zeta(2m-1);
\]
see e.g. [16, page 525]. Therefore,
\[ |R^\text{EM}_m| \leq \frac{2\zeta(2m-1)}{(2\pi)^{2m-1}} \int_0^{n-1} \frac{dx}{f^{(2m-1)}(x)}. \] (2.4)

Here \( \zeta \) is the Riemann zeta function, so that \( \zeta(2m-1) < 1.01 \) for \( m \geq 4 \).

In [7], it is shown that the Abel-Plana summation formula, the Poisson summation formula, and the approximate sampling formula are in a certain sense equivalent to the EM summation formula.

For \( m = 0 \), the Euler–MacLaurin formula takes the form
\[ \sum_{k=0}^{n-1} f(k) = \int_0^{n-1} f(x) \, dx + \frac{f(n-1)+f(0)}{2} + R^\text{EM}_0. \]

Therefore, the general formula (2.1) can be viewed as a higher-order extension of the trapezoidal quadrature formula.

One may note that the Euler–MacLaurin formula implies the Faulhaber formula for the sums of powers. Indeed, take any natural \( p \) and \( n \). Then, using (2.1) with \( f(x) = x^p \), \( m = \lceil p/2 \rceil + 1 \), and \( n \) in place of \( n-1 \), and recalling that \( B_3 = B_5 = \cdots = 0 \), one has \( f^{(2m-1)} = 0 \) and hence \( R^\text{EM}_m = 0 \) and
\[ \sum_{k=0}^{n-1} k^p = -n^p + \sum_{k=0}^{n-1} k^p = \frac{n^{p+1}}{p+1} - \frac{n^p}{2} + \frac{1}{p+1} \sum_{\alpha=2}^{p} B_\alpha \left( \frac{p+1}{\alpha} \right) n^{p+1-\alpha}; \]

here we also use the simple observation that \( f^{(p)}(n) - f^{(p)}(0) = 0 \) if \( f(x) = x^p \). Therefore and because \( B_0 = 1 \) and \( B_1 = -1/2 \), we have the following version of the Faulhaber formula:
\[ \sum_{k=0}^{n-1} k^p = \frac{1}{p+1} \sum_{\alpha=0}^{p} B_\alpha \left( \frac{p+1}{\alpha} \right) n^{p+1-\alpha}, \] (2.5)
for any natural \( p \) and \( n \); this conclusion holds for any nonnegative integers \( p \) and \( n \) as well – assuming the convention \( 0^0 := 1 \), which will be indeed assumed throughout this paper. We shall use (2.5) in the proof of Proposition 3.6.

3. AN ALTERNATIVE (ALT) TO THE EM FORMULA

The following is the main result of this paper:

**Theorem 3.1.** One has
\[ \sum_{k=0}^{n-1} f(k) = A_m - R_m, \] (3.1)

where
\[
A_m := \sum_{j=1}^{m} \sum_{i=0}^{j-1} \int_{i-j/2}^{i-j/2+1} f^{\alpha}, \quad A_m := \sum_{j=1}^{m} \sum_{i=0}^{j-1} \int_{i-j/2}^{i-j/2+1} f^{\alpha},
\] (3.2)

\[
= \sum_{\alpha=1}^{m-1} \tau_{m,1+\alpha} \int_{\alpha/2-1/2}^{\alpha/2+1/2} f^{\alpha} = \sum_{\alpha=1}^{m-1} \tau_{m,1+\alpha} \int_{\alpha/2-1/2}^{\alpha/2+1/2} f^{\alpha},
\] (3.3)

\[
= \tau_{m,1} \int_{i-j/2}^{i-j/2+1} f^{\alpha} + \sum_{\alpha=1}^{m-1} \tau_{m,1+\alpha} \left( \int_{\alpha/2-1/2}^{\alpha/2+1/2} f^{\alpha} \right)
\] (3.4)

\[
= \tau_{m,1} \int_{i-j/2}^{i-j/2+1} f^{\alpha} + \sum_{\alpha=1}^{m-1} \tau_{m,1+\alpha} \left( \int_{\alpha/2-1/2}^{\alpha/2+1/2} f^{\alpha} \right).
\]
is the integral approximation to the sum $\sum_{k=0}^{n-1} f(k)$,

$$\int_a^b := \int_a^b f(x) \, dx := F(b) - F(a),$$  \hspace{1cm} (3.5)

$F$ is any antiderivative of $f$ (so that $\int_a^b = f(x) \, dx$ if $a \leq b$).

$$\gamma_{m,j} := (-1)^{j-1} \frac{2}{m+j} \binom{2m}{m+j},$$  \hspace{1cm} (3.6)

and $R_m$ is the remainder given by the formula

$$R_m := \frac{1}{(2m-1)! 2^{2m+1}} \int_0^1 ds (1-s)^{2m-1} \int_{-1}^1 dv \, v^2m \sum_{j=1}^{m} \sum_{j=1}^{m} f(2m)(k+jsv/2).$$  \hspace{1cm} (3.8)

The sum of all the coefficients of the integrals in each of the expressions in (3.2), (3.3), and (3.4) of $A_m$ is

$$\sum_{j=1}^{m} \gamma_{m,j} j = \sum_{j=1}^{m} \tau_{m,1+|a|} = 1.$$  \hspace{1cm} (3.9)

If $M_{2m}$ is a real number such that

$$\left| \sum_{k=0}^{n-1} f(2m)(k+w) \right| \leq M_{2m} \text{ for all } w \in [m/2, m/2],$$  \hspace{1cm} (3.10)

then the remainder $R_m$ can be bounded as follows:

$$|R_m| \leq \frac{M_{2m}}{(2m+1)! 2^{2m+1}} \sum_{j=1}^{m} |\gamma_{m,j}| j^{2m+1}.$$  \hspace{1cm} (3.11)

Recall the convention that the sum of an empty family is 0. In particular, if $n = 0$, then $\sum_{k=0}^{n-1} f(k) = 0 = A_m = R_m$.

A formal proof of Theorem 3.1 will be given in Section 9. At this point, let us just present the idea leading to representation (3.11). First, one may note that, in accordance with (3.1)–(3.7), the first approximation $A_1 = \int_{1/2}^{1/2}$ of the sum $\sum_{k=0}^{n-1} f(k)$ is obtained by approximating each summand $f(k)$ by $f(k+1/2)$. Next, formally integrating the Taylor expansion

$$f(x) = f(k) + f'(k)(x-k) + f''(k)(x-k)^2/2 + \cdots$$  \hspace{1cm} (3.12)

in $x$ from $k - 1/2$ to $k + 1/2$ and then summing in $k = 0, \ldots, n-1$, one sees that

$$\sum_{k=0}^{n-1} f(k) = \int_{-1/2}^{n-1/2} - \sum_{k=0}^{n-1} f''(k)/24 - \cdots,$$  \hspace{1cm} (3.13)

with the ellipsis $\cdots$ standing for a “negligible” remainder. Integrating now the Taylor expansion (3.12) in $x$ from $k - 1$ to $k + 1$ and then summing again in $k = 0, \ldots, n-1$, one has

$$\sum_{k=0}^{n-1} f(k) = \frac{1}{2} \left( \int_{-1}^{n} + \int_{0}^{n} \right) - \sum_{k=0}^{n-1} f''(k)/6 - \cdots,$$  \hspace{1cm} (3.14)

because

$$\sum_{k=0}^{n-1} \int_{k-1}^{k+1} = \sum_{k=0}^{n-1} \left( \int_{k-1}^{k} + \int_{k}^{k+1} \right) = \int_{-1}^{n} + \int_{0}^{n} = \int_{-1}^{n} + \int_{0}^{n-1}.$$  \hspace{1cm} (3.15)
Multiplying both sides of the identities in (3.13) and (3.14) by 4/3 and −1/3, respectively, and then adding the resulting identities, we eliminate the term containing the second derivatives \( f''(k) \):

\[
\sum_{k=0}^{n-1} f(k) = \frac{4}{3} \int_{-1/2}^{n-1/2} - \frac{1}{6} \left( \int_{-1}^{n} + \int_{0}^{n-1} \right) - \cdots = A_2 - \cdots,
\]

(3.16)

where \( A_m \) is as in (3.2)–(3.4). The last identity, \( \sum_{k=0}^{n-1} f(k) = A_2 - \cdots \), is a non-explicit and non-rigorous version of identity (3.1) for \( m = 2 \).

Similarly eliminating higher-order derivatives in an explicit and rigorous version of the Taylor expansion and generalizing the calculations in (3.15), we shall derive identity (3.1) for all natural \( m \).

The idea described above may remind one of the mentioned earlier Richardson extrapolation process (REP), which results in an elimination of higher-order terms of an asymptotic expansion and thus in a higher rate of convergence; see e.g. 21 [23]. An important difference between the two ideas is that Richardson’s elimination is iterative, in distinction with our representation (3.1), and so, one has to be concerned with the stability of the REP and its generalizations; cf. e.g. 23 [Sections 0.5.2 and 1.6]. Moreover, it will be shown in Section 7 that the computation by formula (3.1) can be almost completely parallelized, whereas the iterative character of Richardson’s method makes that problematic, if at all possible. However, one may note the following.

**Remark 3.2.** Let

\[
\rho_j := \rho_{m,j} := \gamma_{m,j} \text{ for } j = 1, \ldots, m.
\]

(3.17)

Then one has the (downward) recursion

\[
\rho_{m,j-1} = \rho_{m,j} \frac{m+j}{j-m-1} \text{ for } j = m, \ldots, 2, \quad \text{with } \rho_{m,m} = (-1)^{m-1} \frac{m}{2m}.
\]

(3.18)

Of course, this can be rewritten as an “upward” recursion. However, it is the downward recursion that will be used for parallelization, to be described in detail in Section 7.

**Remark 3.3.** In each of the formulas (3.2), (3.3), and (3.4), the first expression is a linear combination of integrals over intervals centered at the point \((n-1)/2\), whereas the endpoints of each of the intervals corresponding to the second expression differ by \(n\).

**Remark 3.4.** The expressions for \( A_m \) in (3.3) are obtained from those in (3.2) by grouping the summands in the double sum with the same integral \( \int_{-j/2}^{n-j/2} / \int_{-j/2}^{n-j/2} \) or \( \int_{-j/2}^{n-j/2} \) and then the expressions for \( A_m \) in the two lines of (3.4) are obtained from the corresponding expressions in (3.3) by grouping the summands with the same value of \( \tau_{m,1+|\alpha|} \). So, each of the expressions in (3.4) requires the calculation of \( 2m-1 \) integrals, which is much fewer for large \( m \) than the \((m+1)m/2\) integrals in (3.2). On the other hand, the coefficients \( \gamma_{m,j} \) in (3.2) are a bit easier to compute than the coefficients \( \tau_{m,j} \) in (3.3) or (3.4).

**Remark 3.5.** Instead of assuming that the function \( f \) is real-valued, one may assume, more generally, that \( f \) takes values in any normed space. In particular, one may allow \( f \) to take values in the \( q \)-dimensional complex space \( \mathbb{C}^q \), for any natural \( q \). Such a situation will be considered in the example in Subsection 8.3. An advantage of dealing with a vector function such as the one defined by (8.17) (rather than separately with each of its components) is that this way one has to compute the coefficients – say \( \tau_{m,1} \) and \( B_{2j} / (2j)! \) in (2.2) – only once, for all the components of the vector function.

We have the following curious and useful identity, whereby the Bernoulli numbers \( B_p \), which are the coefficients in the EM approximation (2.2), are expressed as linear combinations of the coefficients \( \gamma_{m,j} \) and \( \tau_{m,j} \) in the Alt approximation (3.2)–(3.4) to the sum \( \sum_{k=0}^{n-1} f(k) \).
**Proposition 3.6.** Take any \( p = 0, \ldots, 2m - 1 \). Then

\[
B_p = \sum_{j=0}^{m} \gamma_m, j \sum_{i=0}^{j-1} \binom{j-1}{i} p^i = \frac{1}{2^p} \left( \tau_m, 1, (-1)^p + \sum_{\beta=2}^{m} \tau_m, \beta \left[ (\beta - 2)^p + (-\beta)^p \right] \right).
\] (3.19)

The last expression in (3.19) is obtained from the preceding one by grouping the summands in the double sum with the same value of \( \frac{j}{2} - i - 1 \); cf. Remark 3.4.

Identity (3.19) will be used in the proofs of Propositions 5.1 and 5.5. In fact, this identity was discovered in numerical experiments suggesting the limit relation (5.12) in Proposition 5.5. The special case of (3.19) for \( p = 0 \) is (3.9).

Identities of a kind somewhat similar to (3.19) have been known. A survey of them was given in [10]. Identity (3.19) will be used in the proofs of Propositions 5.1 and 5.5.

A notable distinction between identities (3.19) and (3.20) is that, in view of the definition (3.6) of \( \gamma_m, j \), the binomial coefficients involved in (3.19) are of the form \( \binom{2m}{i} \), whereas (3.20) involves all the first \( p + 1 \) rows of the Pascal binomial triangle.

Recall the notation introduced in (3.5). The first three approximations \( A_m \) of the sum \( \sum_{k=0}^{n-1} f(k) \) are as follows:

\[
A_1 = \int_{-1/2}^{n-1/2} f(x) \, dx,
\]
\[
A_2 = \frac{4}{3} \int_{-1/2}^{n-1/2} \left( \int_{-1}^{0} f(x) \, dx + \int_{0}^{n-1} f(x) \, dx \right) \quad \text{(cf. (3.16))},
\]
\[
A_3 = \frac{3}{2} \int_{-1/2}^{n-1/2} \frac{3}{10} \left( \int_{-1}^{0} f(x) \, dx + \int_{0}^{n-1} f(x) \, dx \right) + \frac{1}{30} \left( \int_{-1/2}^{n+1/2} f(x) \, dx + \int_{-1/2}^{n-1/2} f(x) \, dx \right) \quad \text{(3.21)}
\]
\[
= \frac{23}{15} \int_{-1/2}^{n-1/2} \left( \frac{3}{10} \left( \int_{-1}^{0} f(x) \, dx + \int_{0}^{n-1} f(x) \, dx \right) + \frac{1}{30} \left( \int_{-1/2}^{n+1/2} f(x) \, dx + \int_{-1/2}^{n-1/2} f(x) \, dx \right) \right), \quad \text{(3.22)}
\]

using the first expression for \( A_m \) in (3.2); cf. Remark 3.3.

If (as usually will be the case) \( n \geq m - 1 \), then the integral approximation \( A_m \) can be written as just one integral, as follows:

\[
A_m = \int_{-m/2}^{n-1+m/2} f(x) h_m(x), \quad \text{(3.23)}
\]

where

\[
h_m := \sum_{j=1}^{m} \gamma_m, j \sum_{i=0}^{j-1} I_{i-j/2, n-1-j/2-i} = \sum_{\alpha=1-m}^{m-1} \tau_{m, 1+\alpha} I_{(\alpha/2-1/2, n-1/2-\alpha/2)} \quad \text{(3.24)}
\]

and \( I_A \) denotes the indicator function of a set \( A \).

The integral approximation of the sum \( \sum_{k=0}^{n-1} f(k) \) is illustrated in Figure 1 for \( n = 10 \) and \( m = 3 \). In the left panel of the figure, each of the six integrals \( \int_{a}^{b} \) in the expression (3.21) for \( A_3 \) is represented by a rectangle whose projection onto the horizontal axis is the interval \( (a, b) \) and whose height equals the absolute value of the coefficient of the integral in that expression for \( A_3 \). The rectangle is placed above or below the horizontal axis depending on whether the respective coefficient is positive or negative. Thus, each such rectangle also represents a summand of the form \( \gamma_m, j I_{i-j/2, n-1-j/2-i} \) in the expression (3.24) of \( h_m \). The rectangles of the same height are shown in the same color. E.g., the two green rectangles
represent the integrals \( \int_{-1}^{10} f \) and \( \int_{0}^{10} f \); the height of each of these green rectangles is \( \frac{3}{10} \), the absolute value of the coefficient \( -\frac{3}{10} \) of these integrals, and these rectangles are “negative” (that is, below the horizontal axis), since the coefficient \( -\frac{3}{10} \) is negative.

The resulting function \( h_3 \), which is a sort of sum of all the “positive” and “negative” rectangles or, more precisely, the sum of the corresponding functions \( \sum_{k=0}^{n-1} \frac{f(k)}{2} \) (for \( n = 10 \)), is shown in the right panel of Figure 1. In accordance with (3.21), (3.22), the middle blue rectangle has the same base as, and hence can be “absorbed into”, the red rectangle.

One can see that the proposed integral approximation of the sum \( \sum_{k=0}^{n-1} f(k) \) works by (i) “borrowing” information about how the function \( f \) integrates in left and right neighborhoods of, respectively, the left and right endpoints of the interval \([0, n-1]\) and (ii) taking into account boundary effects near the endpoints both inside and outside the interval \([0, n-1]\).

**Remark 3.7.** For real \( a \), let \( \int_{a}^{\infty} \) exist if \( \int_{a}^{r} \) dx f(x) := \( \lim_{a \rightarrow \infty} \int_{a}^{r} \) dx f(x) \( r \in \mathbb{N} \), if this limit exists and is finite. If such an “improper” integral \( \int_{a}^{\infty} \) dx f(x) exists and is finite if the series \( \sum_{k=0}^{n} f^{(k)}(k + v) \) converges uniformly in \( v \in [-m/2, m/2] \), then (3.1) will hold if the instances of \( \sum_{k=0}^{n} f^{(k)}(k + v) \) are replaced respectively by \( \sum_{k=0}^{n} f^{(k)}(k + v) \) in (3.1), (3.2), (3.3), (3.4), and (3.8).

**Remark 3.8.** Suppose that the function \( f \) in Theorem 3.1 is given by the formula \( f(x) = g(x) \epsilon \) for some function \( g \), some real \( \epsilon > 0 \), and all real \( x \). So, if \( \epsilon \) is a small number, the sum \( \sum_{k=0}^{n-1} f(k) = \sum_{k=0}^{n-1} g(k) \epsilon \) may be thought of as an integral sum for the function \( g \) over a fine partition of an interval. Suppose now that, for instance, the function \( |g^{(2m)}| \) is nondecreasing on the interval \([-\epsilon \leq \epsilon m/2, \infty)\) and let \( M_{2m} := \int_{-\epsilon \leq \epsilon m/2}^{\epsilon m/2} \) dy \( |g^{(2m)}(y)| \). Then for all \( v \in [-m/2, m/2] \)

\[
\sum_{k=0}^{n-1} f^{(k)}(k + v) = \epsilon^{2m} \sum_{k=0}^{n-1} |g^{(2m)}|(k + v) \epsilon \leq \epsilon^{2m} M_{2m},
\]

so that, by (3.11),

\[
|R_{2m}| \leq \epsilon^{2m} \frac{M_{2m}}{(2m + 1)!} 2^{2m} \sum_{j=1}^{m} |Y_{m,j}| f^{2m+1},
\]
which provides a justification for referring to $R_m$ as the remainder. Another, less trivial but hopefully more convincing justification will be provided in Sections 4, 5, and 6.

Also, it is clear that $R_m = 0$ if the function $f$ is any polynomial of degree at most $2m - 1$. □

**Remark 3.9.** The alternative summation formula given in Theorem 3.1 can be generalized to multi-index sums. Indeed, one can similarly approximate multi-index sums

\[
\sum_{k_1=0}^{n_1-1} \cdots \sum_{k_r=0}^{n_r-1} f(k_1, \ldots, k_r)
\]

of values of functions $f$ of several variables by linear combinations of corresponding integrals of $f$ over rectangles in $\mathbb{R}^r$, using the multivariable Taylor expansions $f(\mathbf{k} + \mathbf{u}) = f(\mathbf{k}) + f'(\mathbf{k}) \cdot \mathbf{u} + \cdots$, where $\mathbf{k} := (k_1, \ldots, k_r)$ and $\mathbf{u} := (u_1, \ldots, u_r)$. See [20], where an extension to sums over lattice polytopes is also given. □

## 4. Bounds on the Remains

**Remark 4.1.** If on the interval $(-m/2 - 1/2, n - 1/2 + m/2)$ one has $|f^{(2m)}| \leq g_{2m}$ for some nonnegative convex function $g_{2m} : \mathbb{R} \to \mathbb{R}$, then (3.10) will hold with

\[
M_{2m} \leq \int_{-m/2 - 1/2}^{n - 1/2 + m/2} dx g_{2m}(x) \leq \int_{-m/2 - 1/2}^{\infty} dx g_{2m}(x);
\]

(4.1)

here we used the simple observation that $g(a) \leq \int_{a - 1/2}^{a + 1/2} dx g(x)$ for any real $a$ and any function $g$ that is convex on the interval $(a - 1/2, a + 1/2)$. □

In typical applications, the function $f$ will admit an analytic extension of at most a polynomial growth into a large enough region of the complex plane $\mathbb{C}$ containing $[0, \infty)$. Then the higher-order derivatives of $f$ can be nicely bounded without an explicit evaluation of them, and then one can use Remark 4.1 and (3.11) to easily bound the remainder $R_m$. Such a bounding tool is provided by

**Proposition 4.2.** For real $a \geq (m + 3)/2$ and $\theta_0 \in (0, \pi/2)$, consider the sector

\[
S := \{-a + re^{i\theta} : r \geq 0, |\theta| \leq \theta_0\}
\]

of the complex plane $\mathbb{C}$ with vertex at the point $-a$, so that $S \supseteq [0, \infty)$; of course, here $i$ denotes the imaginary unit. Suppose that the function $f : \mathbb{R} \to \mathbb{R}$ can be extended to a function (which we shall still denote by $f$) mapping $\mathbb{R} \cup S$ into $\mathbb{C}$ such that $f$ is continuous on the set $S$, holomorphic in the interior of $S$, and

\[
|f(z)| \leq \mu |z + a + 1|^{\lambda}
\]

(4.2)

for some real $\mu \in [0, \infty)$ and $\lambda \in [0, 2m - 1)$ and for all $z \in S$. Then

\[
|R_m| \leq \frac{C(\mu, \lambda, \theta_0, m)}{(a - m/2 - 1/2)^{2m - 1 - \lambda}},
\]

(4.3)

where

\[
C(\mu, \lambda, \theta_0, m) := \frac{\mu (2 + \sin \theta_0)^\lambda}{(2m + 1)(2m - 1 - \lambda)(2 \sin \theta_0)^{2m}} \sum_{j=1}^{m} |\gamma_{m, j}| f^{2m+1}.
\]

(4.4)

Similarly, for $m \geq 4$, $a > 0$, and $\lambda \in [0, 2m - 2)$,

\[
|R_{m_E}| \leq \frac{C_{m_E}(\mu, \lambda, \theta_0, m)}{a^{2m - 2 - \lambda}};
\]

(4.5)

where

\[
C(\mu, \lambda, \theta_0, m) := \frac{2.02 \mu (2 + \sin \theta_0)^\lambda}{(2m - 2 - \lambda)(2 \sin \theta_0)^{2m - 1}} (2m - 1)! (2\pi)^{2m - 1}.
\]

(4.6)
In applications, $m$ will be rather large, and $a$ will be substantially greater than $m$, to make the upper bounds on $|R_m|$ and $|R_m^\text{EM}|$ in (4.3) and (4.5) very small.

**Remark 4.3.** As in Proposition 4.2, suppose that $f$ is continuous on the set $S$ and holomorphic in the interior of $S$. Consider the function $h$ defined by the formula $h(z) := f(z)/(z + a + 1)^\lambda$ for $z \in S$. In view of the Phragmén–Lindelöf extension of the maximum modulus principle (applied to $h$), for the condition (4.2) to hold for all $z \in S$, it is enough that (4.2) hold for all $z$ on the boundary of $S$—provided that $f$ does not grow too fast on $S$, that is, provided that $|f(z)| \leq Ce^{\left|z+a\right|^\alpha}$ for some $\alpha \in [0, \pi/2]$, some real $C$, and all $z \in S$; see e.g. [24, §5.6.1].

Bounds (3.11) and (4.3)–(4.4) are complemented by Proposition 4.4.

If $m \geq 2$ then

$$\sum_{j=1}^{m} \left| r_{m,j} \right|^j 2^{m+1} \leq 1.001 \pi \lambda^m 2^{2m+1},$$

(4.7)

where

$$\lambda := \max_{0 < t < 1} \lambda(t) = 0.3081 \ldots, \quad \Lambda(t) := (1-t)^{r-1}(1+t)^{-1-t}.$$

(4.8)

For $m = 1$, (4.7) holds with 1.0331 in place of 1.001.

It appears that in most practical situations it will be possible to use Proposition 4.2 with $\theta_0 = \pi/2$. In such a case, the expression for the constant in (4.4) can be simplified, and we immediately obtain the following corollary of Propositions 4.2 and 4.4.

**Corollary 4.5.** Suppose that $m \geq 2$ and the conditions in Proposition 4.2 hold with $\theta_0 = \pi/2$. Then

$$|R_m| \leq \frac{1.001 \pi \mu 3^\lambda}{(2m+1)(2m-1-\lambda)} \left( \frac{\lambda}{4} \right)^m \frac{m^{2m+1}}{(a-m/2-1/2)^{2m-1-\lambda}}.$$ 

(4.9)

For $m = 1$, (4.9) holds with 1.0331 in place of 1.001.

5. **Application to Summing (Possibly Divergent) Series**

The alternative summation formula presented in Theorem 3.1 can be used for summing (possibly divergent) series, as follows.

**Proposition 5.1.** Let $m_0$ be a natural number, and suppose that $m \geq m_0$. Suppose that

$$f^{(2m_0-1)}(x) \xrightarrow{x \to \infty} 0$$

(5.1)

and the series

$$\sum_{k=0}^{\infty} f^{(2m)}(k + w) \text{ converges uniformly in } w \in [-m/2,m/2].$$

(5.2)

Let $F$ be any antiderivative of $f$, so that $F' = f$. Then

$$\sum_{k=0}^{\infty} f(k) := \lim_{n \to \infty} \left( \sum_{k=0}^{n-1} f(k) - G_{m_0,F}(n) \right) = -G_{m,F}(0) - R_{m,F}(\infty),$$

(5.3)
where (cf. (3.2), (3.3), and (3.4))

\[
G_{m,F}(n) := \sum_{j=0}^{m-1} \gamma_{m,j} \sum_{i=0}^{j-1} F(n-1+j/2-i) = \sum_{\alpha=1}^{m} \tau_{m,1+\alpha} F(n-1/2-\alpha/2) = \tau_{m,1} F(n-1/2) + \sum_{\alpha=1}^{m-1} \tau_{m,1+\alpha} \left[F(n-1/2-\alpha/2) + F(n-1/2+\alpha/2)\right]
\]  

(5.4)  

(5.5)

(5.6)

and (cf. (3.8))

\[
R_{m,f}(\infty) := \frac{1}{(2m-1)!} \int_{0}^{1} dx \left(1-s\right)^{2m-1} \int_{-1}^{1} dy \sqrt[2m]{\sum_{j=1}^{m} \gamma_{m,j} f^{2m+1}(k)} \sum_{k=0}^{\infty} f^{2m}(k+js). \]

(5.7)

The limit \( \sum_{k=0}^{\infty} f(k) \) in (5.3) may be referred to as the (generalized) sum of the possibly divergent series \( \sum_{k=0}^{\infty} f(k) \) by means of the Alt formula (5.1).

**Remark 5.2.** The centering/stabilizing term \( G_{m,F}(n) \) in (5.3) equals \( F(n-1/2) \) for all natural \( n \) if one can take \( m_0 = 1 \). Thus, if \( F(\infty) := \lim_{n \to \infty} F(n) \) exists and is finite, and if \( F \) is chosen so that \( F(\infty) = 0 \), then we will have \( \sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{\infty} f(k) \).

The key point in the proof of Proposition 5.1 is that, under the conditions of Proposition 5.1, \( G_{m,F}(n) \to G_{m_0,F}(n) \) as \( n \to \infty \).

The EM summation formula, too, can be used for summing possibly divergent series. The following proposition is rather similar to Proposition 5.1.

**Proposition 5.3.** Let \( m_0 \) be a natural number, and suppose that \( m \geq m_0 \). Suppose that

\[
f^{(2j-1)}(x) \to 0 \quad \text{for } j = m_0, \ldots, m-1
\]

(5.8)

and

\[
\int_{0}^{\infty} dx |f^{(2m-1)}(x)| < \infty.
\]

(5.9)

Let \( F \) be any antiderivative of \( f \), so that \( F' = f \). Then

\[
\sum_{k=0}^{\infty} f(k) := \lim_{n \to \infty} \left[ \sum_{k=0}^{n-1} f(k) - G_{m,F}(n) \right] = f(0) - G_{m,F}(1) + R_{m,F}(\infty),
\]

(5.10)

where (cf. (2.2))

\[
G_{m,F}(n) := F(n-1) + \frac{F'(n-1)}{2} + \sum_{j=1}^{m-1} B_{2j} \left[ F^{(2j)}(n-1) \right]
\]

(5.11)

and (cf. (2.3))

\[
R_{m,F}(\infty) := \frac{1}{(2m-1)!} \int_{0}^{\infty} dx f^{(2m-1)}(x) B_{2m-1}(x - [x]).
\]

(5.12)

The limit \( \sum_{k=0}^{\infty} f(k) \) in (5.10) may be referred to as the (generalized) sum of the possibly divergent series \( \sum_{k=0}^{\infty} f(k) \) by means of the EM formula (2.1).

**Remark 5.4.** Suppose that the function \( f \) satisfies conditions of Proposition 4.2. Then, in view of the bound (5.9) in the proof of Proposition 4.2, conditions (5.1), (5.2), (5.8), and (5.9) will all hold if \( 2m_0 > 1 + \lambda \) and \( m > m_0 \).
Proposition 5.5. Let the conditions of Propositions 5.1 and 5.3 hold. Then
\[ G_{m,F}(n) - G^{EM}_{m,F}(n) \xrightarrow[n\to\infty]{} 0, \]
and hence the generalized sums in (5.3) and (5.10) are equal to each other:
\[ \sum_{k\geq 0} f(k) = \sum_{k\geq 0} f(k). \]
Moreover, \( G_{m,p}(n) = G^{EM}_{m,p}(n) \) for any polynomial \( P \) of degree \( \leq 2m - 1 \) and any \( n \).

Proposition 5.5 suggests some curious “objectivity” in summing possibly divergent series, when the two seemingly quite different methods of summation yield the same result. Note that each of the generalized sums \( \sum_{k=0}^{\infty} f(k) \) and \( \sum_{k=0}^{\infty} f(k) \) in (5.3) and (5.10) depends on the choice of the additive constant in the expression of an antiderivative \( F \) of \( f \) and is thus similar to the indefinite integral \( \int f(x) \, dx \); yet, these two generalized sums are equal to each other for any choice of an antiderivative \( F \) of \( f \), provided that the conditions of Propositions 5.1 and 5.3 hold.

Further, one may note that
\[ \sum_{k\geq 0} f(k) = \sum_{k\geq 0} f(k) = F(0) + \sum_{k\geq 0} f(k), \]
again provided that the conditions of Propositions 5.1 and 5.3 hold, where \( \sum_{k=0}^{\infty} f(k) \) denotes the Ramanujan constant of the series \( \sum_{k=0}^{\infty} f(k) \). Cf. e.g. the expression for the Ramanujan constant \( \sum_{k=1}^{\infty} f(k) \) of the series \( \sum_{k=1}^{\infty} f(k) \) in formula (23) in [8]; to match the expression on the right-hand side of formula (5.10) in the present paper with that in [8], one should accordingly replace there \( N, \delta^{k-1} f(1) \), and \( f_0 \) by \( 2m - 1, \delta^{k-1} f(0) \) and \( f_0 \), respectively, and also recall that \( B_1 = -1/2 \) and \( B_3 = B_5 = \cdots = 0 \). So, choosing the antiderivative \( F \) of \( f \) determined by the condition \( F(0) = 0 \), we would have \( \sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{\infty} f(k) \).

However, such a choice of \( F \) may not always be the most natural one. For instance, by (5.13), (5.10), and (8.19), \( \sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{\infty} f(k) = \zeta(p, \delta) \) for \( f = f_{p, \delta} \) as in (8.18), \( F = F_{p, \delta} \) as in (8.20), \( p \in \mathbb{C} \setminus \{1\} \), and \( \delta \in \mathbb{C} \setminus (-\infty, 0] \), whereas \( \sum_{k=0}^{\infty} f_{p, \delta}(k) = \zeta(p, \delta) + F_{p, \delta}(0) = \zeta(p, \delta) + \frac{1}{1-p} \delta^{-p} \neq \zeta(p, \delta) \) (cf. [8], formula (52)). Cf. also Remark 5.2 which suggests that, in the case when the series \( \sum_{k=0}^{\infty} f(k) \) converges, the natural choice of \( F \) will usually be given by the condition \( F(\infty) = 0 \), rather than \( F(0) = 0 \).

To compute the generalized sums \( \sum_{k=0}^{\infty} f(k) \) and \( \sum_{k=0}^{\infty} f(k) \) effectively, one has to make sure that the remainders \( R_{m,F}(\infty) \) and \( R^{EM}_{m,F}(\infty) \) can be made arbitrarily small. It can be seen that the bounds in (3.11)–(3.10) and (2.4) are rather tight. Therefore, usually the only way to ensure that the remainders \( R_{m,F}(\infty) \) and \( R^{EM}_{m,F}(\infty) \) be small will be to make the high-order derivatives \( f^{2m}(n) \) and \( f^{(2m-1)}(n) \) small. This can be achieved by the following simple trick.

For any function \( h: \mathbb{R} \to \mathbb{R} \) and any real \( c \), let \( h_c \) denote the \( c \)-shift of \( h \) defined by the formula
\[ h_c(x) := h(x + c) \]
for all real \( x \). Let now \( c \) be any natural number, and suppose the conditions in Proposition 5.1 hold. Note that
\[ G_{m,F}(n + c) = G_{m,F}(n) \]
for all natural $n$. So,

$$\sum_{k=0}^{n+c-1} f(k) - G_{m,f}(n+c) = \sum_{k=0}^{c-1} f(k) + \left( \sum_{k=0}^{n-1} f_c(k) - G_{m,f}(c) \right).$$

Letting now $n \to \infty$ and using (5.14) again (now with 0 in place of $n$), we see that Proposition 5.1 immediately yields

**Corollary 5.6.** Under the conditions in Proposition [5.1] for any natural $c$

$$\sum_{k \geq 0} f(k) = \sum_{k=0}^{c-1} f(k) - G_{m,f}(c) - R_{m,f}(\infty). \quad (5.15)$$

That is, (5.3) holds if $G_{m,F}(0)$ and $R_{m,f}(\infty)$ are replaced there by $-\sum_{k=0}^{c-1} f(k) + G_{m,f}(c)$ and $R_{m,f}(\infty)$, respectively. Under the condition (5.2), one can indeed make the remainder $R_{m,f}(\infty)$ arbitrarily small by taking a large enough $c$. The extra price to pay for this is the need to compute the additional term $\sum_{k=0}^{c-1} f(k)$.

Similarly, Proposition 5.3 immediately yields

**Corollary 5.7.** Under the conditions in Proposition 5.3 for any natural $c$

$$\sum_{k \geq 0}^{EM} f(k) = \sum_{k=0}^{c-1} f(k) - G_{m,f}(c+1) + R_{m,f}(\infty). \quad (5.16)$$

Under the condition (5.9), one can make the remainder $R_{m,f}(\infty)$ arbitrarily small by taking a large enough $c$.

One can view (5.15) and (5.16) as extrapolation formulas (in a broad enough sense), with the correction terms $-G_{m,f}(c)$ and $-G_{m,f}(c+1)$ added to the partial sums $\sum_{k=0}^{c-1} f(k)$ and $\sum_{k=0}^{c-1} f(k)$ of the series $\sum_{k=0}^{\infty} f(k)$ to obtain better approximations to its generalized sum $\sum_{k \geq 0}^{Alt} f(k) = \sum_{k \geq 0}^{EM} f(k)$.

The special case of Corollary 5.7 with $f(x) = \frac{1}{1+x}$ for $x \geq 0$ was, essentially, the basis of Knuth’s method in [17] to compute a decimal approximation to Euler’s constant; cf. [17, formula (7)]. In that case, one can take $m_0 = 1$ and $F(x) = \ln(x+1)$ for $x \geq 0$. Then the “stabilizers” in (5.15) and (5.16) will be $G_{m_0,f}(n) = \ln(n+1/2)$ and $G_{m_0,f}(n) = \ln(n+\frac{1}{2\pi})$, respectively, which in particular illustrates (5.12).

6. CHOOSING C AND M FOR A DESIRED ACCURACY

As was noted, the integer $c$ should be taken to be large enough to ensure that the remainders $R_{m,f}(\infty)$ and $R_{m,f}(\infty)$ in (5.15) and (5.16) be small. How large $c$ must be depends on the choice of $m$. In turn, one can see that the order $m$ of the approximation formulas (3.1) and (2.1) should also be large enough for the remainders to be small.

In the following two subsections, we shall consider these theses in some detail. As we shall see, there is a certain balance between the required values of $m$ and $c$. If the value of $m$ is too small, then the required value of $c$ must be too large to ensure the desired accuracy. Thus, trying to decrease $m$ to reduce the volume of calculations to compute $G_{m,f}(c)$ or $G_{m,f}(c+1)$ in (5.15) and (5.16) according to (5.4), (5.6) and (5.11) may result in an increased volume of calculations to compute the sums $\sum_{k=0}^{c-1} f(k)$ and $\sum_{k=0}^{c-1} f(k)$ in (5.15) and (5.16).

Suppose that $m \geq 2$ and all the conditions in Proposition 4.2 hold with $\theta_0 = \pi/2$ – except possibly the condition $a \geq (m+3)/2$, so that here

$$S = \Pi^{-a}_+ := \{z \in \mathbb{C} : \Re z \geq -a\}. \quad (6.1)$$
Let \( c \) be any natural number such that \( a + c \geq (m + 3)/2 \) – this latter condition replacing the condition \( a \geq (m + 3)/2 \). Condition (4.2) will hold (now for all \( z \in \Pi_{c-a}^1 \)) with \( f_c \) and \( a_c := a + c \) in place of \( f \) and \( a \), respectively.

6.1. **Choosing \( c \) and \( m \) nearly optimally in (5.15).** In view of Corollary 4.5

\[
|R_{m,f}(\infty)| \leq R_{m,c}^* := R_{m,c,\mu,\lambda,a}^* := \frac{1001\pi^{3\lambda}}{(2m+1)(2m-1-\lambda)} \left( \frac{\Lambda_c}{4} \right)^{m} \frac{m^{2m+1}}{(c + a - m/2 - 1/2)^{2m-1-\lambda}} \quad (6.2)
\]

for \( m \to \infty \) and \( c - m/2 \geq (\kappa + \varepsilon)m \) for some fixed real \( \varepsilon > 0 \), where

\[
\kappa := \sqrt{\frac{\Lambda_c}{4}} = 0.27754 \ldots
\]

So, to ensure that

\[
|R_{m,f}(\infty)| \leq \frac{1}{2} 10^{-d}
\]

for a large enough natural \( d \), an appropriate choice of \( c \) will be as follows:

\[
c = \lceil c_{d,m} \rceil \approx m/2 + \kappa m 10^{d/((2m))}, \quad (6.4)
\]

where \( c_{d,m} = c_{d,m,\mu,\lambda,a} \) is the root of the equation \( R_{m,c}^* = \frac{1}{2} 10^{-d} \).

Let now \( T_f = T_f(d) \) denote the time needed to compute one value of the function \( f \). We suppose here that this time does not depend significantly on the value of the argument of \( f \) in the range \( \{0, \ldots, c-1\} \); otherwise, take the average over the range.

However, \( T_f = T_f(d) \) will depend on the working accuracy needed to attain the desired accuracy of \( \frac{1}{2} 10^{-d} \) of the ultimate result. We shall see at the end of this subsection that, for large \( d \), this working accuracy will have to be \( 10^{-d} \), where \( d \) exceeds \( d \) only by a summand \( \asymp \ln d \).

Similarly introduce \( T_F = T_F(d) \), the time “cost” per value of the antiderivative \( F \) of \( f \), and \( T_T = T_T(d) \), the time “cost” per value of \( T \). Then the total time needed to compute the approximate value \( \sum_{k=0}^{c-1} f(k) - G_{m,F}(c) \) of the generalized sum \( \sum_{k=0}^{c-1} f(k) \) in (5.15) is

\[
\mathcal{T} \approx T_f c + T_m m + T_F \times 2m \approx \mathcal{T}(m) := Km \left( 1 + \omega 10^{d/(2m)} \right) \quad (6.5)
\]

in view of (6.4), where

\[
K := T_f/2 + T_c + 2T_F \quad \text{and} \quad \omega := \frac{K T_f}{K}. \quad (6.6)
\]

We can now find an approximately optimal value of \( m \) by minimizing \( \mathcal{T}(m) \) in \( m \), and then choose \( c \) in accordance with (6.4). Assuming that \( T_F \geq T_f \) and hence \( K > 2.5T_f \), we will have

\[
\omega < \frac{K}{2.5} \approx 0.1. \quad (6.7)
\]

It is easy to see that \( \mathcal{T}(m) \) is strictly convex in \( m \geq 1 \), \( \mathcal{T}(m) \to \infty \) as \( m \to \infty \), and

\[
\mathcal{T}'(1)/K = 1 + \omega 10^{d/(2m)} (1 - \frac{d}{2m} \ln 10) < 1 - 10^{2/3} \omega < 0
\]

provided that \( d \geq 500 \) (which will be the case in the examples to be considered in this paper) and \( \omega > 10^{-252} \). The latter condition will hold in all realistic situations – when the time “cost” per value of the antiderivative \( F \) is not quite prohibitively high. Therefore, \( \mathcal{T}(m) \) attains its minimum in \( m \geq 1 \) at the unique root \( m = m_\omega \in (1, \infty) \) of the equation \( \mathcal{T}'(m) = 0 \), which can be rewritten as

\[
1 + \omega 10^{d/(2m)} (1 - \frac{d}{2m} \ln 10) = 0. \quad (6.8)
\]
This root is given by the formula
\[
m_{\omega} = \frac{\ln 10}{1 + L\left(\frac{1}{e^\omega}\right)} \frac{d}{2} \approx \frac{\ln 10}{2\ln \frac{1}{\omega}} d, \tag{6.9}
\]
where \(L\) is the Lambert product-log function, so that for all positive real \(z\) and \(u\) one has \(z = L(u) \iff u = ze^z\); the approximate equalities here involving \(\omega\) hold under the natural assumption that \(\omega\) is small; recall (6.7) and the corresponding discussion. For \(m = m_{\omega}\), it follows from (6.8) that
\[
\omega 10^{d/(2m)} = \frac{1}{d m \ln 10 - 1} = \frac{1}{L\left(\frac{1}{e^\omega}\right)} \approx \frac{1}{\ln \frac{1}{\omega}}.
\]
So, by (6.4), (6.9), and (6.5), an appropriate choice of \(c\) will be
\[
c \approx m_\omega \left(\frac{1}{2} + \frac{\kappa}{\omega L\left(\frac{1}{e^\omega}\right)}\right) \approx \frac{\ln 10}{1 + L\left(\frac{1}{e^\omega}\right)} \left(\frac{1}{2} + \frac{\kappa}{\omega L\left(\frac{1}{e^\omega}\right)}\right) \frac{d}{2} \approx \frac{\kappa \ln 10}{2\omega \ln \frac{1}{\omega}} \frac{d}{2} \approx \frac{\kappa \ln 10 \ln 10}{2\omega \ln \frac{1}{\omega}} \frac{d}{2}, \tag{6.10}
\]
whence
\[
\mathcal{T}(m) = \frac{K \ln 10}{L\left(\frac{1}{e^\omega}\right)} \frac{d}{2} \approx \frac{K \ln 10}{2 \ln \frac{1}{\omega}} d. \tag{6.11}
\]

One can see that it is rather straightforward to find nearly optimal values of \(m\) and \(c\) for (5.15). Formulas (6.9) and (6.10) show that such values of \(m\) and \(c\) are both approximately proportional to the desired number \(d\) of digits of accuracy, and the proportionality coefficients are rather moderate in size unless \(\omega\) is very small.

Usually, \(T_r\) will be small compared to \(T_f\) and \(T_T\); then, \(\omega\) will be very small only if \(T_f\) is much greater than \(T_r\). In particular, in the rather typical case when \(T_f \approx T_f \gg T_T\), nearly optimal values of \(m\) and \(c\) are as follows: \(m \approx 0.55d\) and \(c \approx 1.5d\).

Also, according to (6.11) and (6.9), the needed time-per-digit \(\mathcal{T}(m)/d\) when using formula (5.15) will be approximately proportional to the “aggregated” time “cost-per-value” \(K = T_f/2 + T_r + 2T_T\) for values of \(f, \tau_r\), and \(F\) – again with a proportionality coefficient \(\left(\frac{\ln 10}{2 \ln \frac{1}{\omega}}\right)\) that is moderate in size unless \(\omega\) is very small.

Again, by (6.9) and (6.10), for optimal values of \(m\) and \(c\) we have \(m \approx d\) and \(c \approx d\). Hence, the needed working accuracy \(d_r\) is \(d + O(\log_{10} d)\), which is \(\sim d\) for large \(d\). Mathematica keeps a rigorous record of the accuracy in its calculations with arbitrary-precision numbers. From the Mathematica tutorial/ArbitraryPrecisionNumbers:

[...] the Wolfram Language keeps track of which digits in your result could be affected by unknown digits in your input. It sets the precision of your result so that no affected digits are ever included. This procedure ensures that all digits returned by the Wolfram Language are correct, whatever the values of the unknown digits may be.

6.2. Choosing \(c\) and \(m\) in (5.16) for a desired accuracy. In this subsection, we still be assuming the conditions stated in the paragraph containing (6.1). First here, instead of Corollary 4.5, let us use (4.5) to get
\[
|R_{em, f, \varepsilon}^m(\omega)| \leq R_{m, c}^{EM, *} := R_{m, c, \mu, \lambda, \alpha}^{EM, *} := \frac{2.02 \mu^3 \lambda (2m - 1)!}{2m - 2 - \lambda} \frac{1}{(2\pi)^{2m-1}} \frac{1}{(c + \alpha)^{2m-2-\lambda}} \tag{6.12}
\]
for \(m \to \infty\) and \(c \geq (\kappa^{EM} + \varepsilon)m\) for some fixed real \(\varepsilon > 0\), where
\[
\kappa^{EM} := \frac{1}{\pi e} \approx 0.12.
\]
So, to ensure that 
\[ |\epsilon^EM_m(\infty)| \leq \frac{1}{2} 10^{-d} \]
for a large enough natural \( d \), an appropriate choice of \( c \) will be as follows:
\[ c = \left[ c^EM_d \right] \approx k^EM m 10^{d/(2m)} , \quad (6.13) \]
where \( c^EM_{d,m} = c^EM_{d,m; \lambda, \lambda, \lambda} \) is the root \( c \) of the equation \( R^EM_{m,c} = \frac{1}{2} 10^{-d} \).

Let \( T_f = T_f(d) \) still denote the time needed to compute one value of the function \( f \).

In comparison with dealing with formula (5.15), it will usually be much more difficult to find nearly optimal choices of \( m \) and \( c \) for the EM calculations according to (5.16) – mainly because it is difficult to assess, especially in general terms, the time needed to compute the values of the derivatives
\[ F^{(2j)}(c) = f^{(2j-1)}(c) \quad \text{for} \quad j = 1, \ldots, m - 1, \quad (6.14) \]
needed in (5.15).

For instance, suppose that the expression for the function \( f \) contains the product \( gh \) of two non-polynomial functions \( g \) and \( h \). Even if the time to compute the values \( g^{(i)}(c) \) and \( h^{(i)}(c) \) is not growing with \( i \), still \( \sim j \) multiplications will be needed to compute the value \( (gh)^{(j)}(c) = \sum_{i=0}^{j} \binom{j}{i} g^{(i)}(c) h^{j-i}(c) \) by the Leibniz formula. Here we are not taking into account the efforts to compute the \( g^{(i)}(c) \)'s, \( h^{(i)}(c) \)'s, and the binomial coefficients \( \binom{j}{i} \). So, the time to compute all the derivatives (6.14) will be \( \geq m^2 \) – which may be compared with the time \( Km \) to compute the values of one function, \( F \), needed in (5.3).

If the expression for the function \( f \) contains the composition \( g \circ h \) of two functions \( g \) and \( h \), then the derivatives of \( g \circ h \) of required orders can be computed by a recursive version of the Faà di Bruno formula: \( (g \circ h)^{(j)}(c) = P_j \), where \( P_j := P_j(g_0, \ldots, g_j, h_1, \ldots, h_j) \) is a polynomial in \( g_0, \ldots, g_j, h_1, \ldots, h_j \) of degree \( j + 1 \) given recursively by the formulas \( P_0(g_0) := g_0 \) and
\[ P_{j+1} := \left( \sum_{i=0}^{j} \frac{\partial P_j}{\partial g_i} g_{i+1} \right) h_1 + \sum_{i=1}^{j} \frac{\partial P_j}{\partial h_i} h_{i+1} \]
for \( j = 0, 1, \ldots, \) and \( g_i := g^{(i)}(h(c)) \) and \( h_i := h^{(i)}(c) \) for all \( i \). So, if the functions \( g \) and \( h \) are non-polynomial, here as well the time to compute all the derivatives (6.14) will be \( \geq m^2 \) – counting neither the time to compute the \( g^{(i)}(h(c)) \)'s and \( h^{(i)}(c) \)'s, nor the time to compute the partial derivatives of the polynomials \( P_j \). Note that the number of monomials in the polynomial \( P_j \) equals the number of partitions of \( j \), which is known [11] to be asymptotic to \( \frac{1}{4\sqrt{\pi}} e^{c_0\sqrt{j}} \) as \( j \to \infty \), where \( c_0 := \pi \sqrt{2/3} \approx 2.56 \ldots \). So, here the time to compute \( f^{(j)}(c) \) will grow for large \( j \) much faster than any power of \( j \). Of course, such a difficult situation can quickly become much worse when the expression for \( f \) is more complicated than just the product or the composition of two non-polynomial functions.

To use the EM formula (2.1) – (2.2), one also needs to compute the Bernoulli numbers \( B_2, \ldots, B_{2m-2} \). In Mathematica, for large enough \( m \) this is done using the Fillbrown algorithm [9], which requires \( \asymp m^2 / \ln m \) multiplications of integers represented by \( \leq m \ln m \) bits, according to the analysis in [9]; note the typo in the abstract in [9], where it should be \( m^2 / \log m \) in place of \( m^2 \log m \).

We can now see that calculations by the EM formula will in most cases involve different computational strands of different degrees of growth in complexity, and those strands can be mixed in different proportions. The rates of growth within the different strands and the proportions between the strands will of course strongly depend on the function \( f \), as well as on the desired number \( d \) of digits of accuracy and the choice of \( m \) and \( c \). These rates and proportions may also significantly vary with \( j \), the index in \( f^{(j)}(c) \) and \( B_{2j} \). However, in most case the time needed to compute the Bernoulli numbers \( B_2, \ldots, B_{2m-2} \) will be much less than that for the derivatives (6.14).
It should be clear from the above discussion that we can only offer a rough, tentative analysis pertaining to the choice of nearly optimal values of \( m \) and \( c \) in (5.16) in general — assuming that the time to compute the derivatives (6.14) and the Bernoulli numbers \( d \) in (5.13)

somewhat informal analysis, we assume that \( d \) is a positive real “constant” \( \varepsilon \) and \( T_{\text{der}} \); \( T_{\text{der}} \) usually will, and \( \varepsilon \) may, depend on the desired accuracy \( \frac{1}{2} 10^{-d} \).

So, the total time needed to compute the approximate value \( \sum_{k=0}^{c-1} f(k) \) of the generalized sum \( \sum f(k) \) in (5.16) will be as follows:

\[
\mathcal{J}^{\text{EM}} \approx \mathcal{J}^{\text{EM}}(m) := T_f m 10^{d/(2m)} + T_{\text{der}} m^{2+\varepsilon},
\]

in view of (6.13), where \( T_B \) is some positive constant and

\[
\tilde{T}_f := \kappa^{\text{EM}} T_f.
\]

We can now find an approximately optimal value of \( m \) by minimizing \( \mathcal{J}^{\text{EM}}(m) \) in \( m \), and then choose \( c \) in accordance with (6.13). It is easy to see that \( \mathcal{J}^{\text{EM}}(m) \) is strictly convex in \( m \geq 1 \), \( \mathcal{J}^{\text{EM}}(m) \to \infty \) as \( m \to \infty \), and

\[
(\mathcal{J}^{\text{EM}})'(m) = T_f 10^{d/(2m)} \left( 1 - d \ln 10 / (2m) \right) + (2 + \varepsilon) T_{\text{der}} m^{1+\varepsilon}
\]

tends to \(-\infty \) if \( d \to \infty \) but \( m \) stays bounded. So, \( \mathcal{J}^{\text{EM}}(m) \) is minimized in \( m \) only when \( m \) is the root of the equation \((\mathcal{J}^{\text{EM}})'(m) = 0\), and the minimizer \( m \) tends to \( \infty \); here and in the rest of this somewhat informal analysis, we assume that \( d \to \infty \). If \( m \to \infty \) but \( d/m \) stays bounded, then, by (6.16), \((\mathcal{J}^{\text{EM}})'(m) \to \infty \). So, for the minimizer \( m \) of \( \mathcal{J}^{\text{EM}}(m) \), we have \( m \to \infty \) and \( d/m \to \infty \); hence, again by (6.16),

\[
\tilde{T}_f 10^{d/(2m)} d \ln 10 / (2m) \sim (2 + \varepsilon) T_{\text{der}} m^{1+\varepsilon},
\]

whence \( d/(2m) \sim (1 + \varepsilon) \log_{10} m \) and

\[
m \sim \frac{2d}{(1 + \varepsilon) \log_{10} d}.
\]

It also follows from (6.15), (6.17), and (6.18) that

\[
\mathcal{J}^{\text{EM}}(m) \sim T_{\text{der}} m^{2+\varepsilon} \sim T_{\text{der}} \left( \frac{d}{2(1 + \varepsilon) \log_{10} d} \right)^{2+\varepsilon} \sim \left( \frac{d}{\log_{10} d} \right)^{2+\varepsilon},
\]

which may be compared with \( \mathcal{J}(m) \approx d \) according to (6.11).

7. Parallelization and Memory Use

It was made clear in Section 6 that both \( m \) and \( c \) should be large enough in order for the remainders \( R_{\text{der}}(\infty) \) and \( R_{\text{der}}^{\text{EM}}(\infty) \) in (5.15) and (5.16) to be small. It is therefore important to consider whether calculations of the terms \( \sum_{k=0}^{c-1} f(k) \) and \( G_{\text{der}}(c) \) in (5.15) and \( \sum_{k=0}^{c-1} f(k) \) and \( G_{\text{der}}^{\text{EM}}(c + 1) \) in (5.16) can be parallelized, for such large values of \( m \) and \( c \).

Let \( k_s \) denote the number of computer cores available for parallel computation. The terms \( \sum_{i=0}^{c-1} f(i) \) in (5.15) and \( \sum_{i=0}^{c-1} f(i) \) in (5.16) can be easily computed about \( k_s \), times as fast in parallel on the \( k_s \) cores as on one such core or on a single-core CPU. To do such a parallel calculation, one can partition the index range, say \( R := \{0, \ldots, c - 1\} \), into \( k_s \) sub-ranges \( R_1, \ldots, R_{k_s} \), each approximately of same size \( \approx c/k_s \), compute each of the corresponding parts \( S_k := \sum_{i \in R_k} f(i) \) of the sum \( \sum_{i=0}^{c-1} f(i) \) on (say) core \( k \) of the \( k_s \) parallelly engaged cores, and then quickly add the partial sums \( S_1, \ldots, S_{k_s} \). In Mathematica, such a calculation can be done by issuing the command

\[
\text{ParallelSum}[\text{SetAccuracy}[f[i], \text{d1}], \{i, 0, c-1\}, \text{Method} \to \text{"CoarsestGrained"}],
\]

where \( \text{d1} \) is the accuracy set for each summand \( f(i) \). In distinction with "FinestGrained", the
"CoarsestGrained" method minimizes the overhead caused by data interchange between the controlling process (master kernel) and the subordinate processes (subkernels, running on available parallel cores).

However, there seems to be no way in general to parallelize the calculation of the consecutive derivatives \( F^{(j)}(c) = f^{(j-1)}(c) \) for \( j = 1, \ldots, m-1 \) in the expression of the term \( G_{m,k}^{EM}(c+1) \) in (5.16), defined by (5.11). Usually, this will be the bottleneck in using the EM summation formula for high-precision calculations.

It is possible to compute the Bernoulli numbers in parallel [12]. However, such parallelization is not done in Mathematica, and it has not been done in the computer experiments to be described in the examples in Section 8 – mainly because, as was noted, except for a very narrow set of functions \( f \), the time needed to compute the Bernoulli numbers \( B_2, \ldots, B_{2m-2} \) will be much less than that for the derivatives (6.14). The usually very large amount of time needed to compute those derivatives certainly precludes values of \( m > \frac{1}{2} \times 10^4 \). On the other hand, for values of \( m \leq \frac{1}{2} \times 10^4 \), the only execution time reported in [12] is for \( B_{2m} = B_{10^4} \), with \( m = \frac{1}{2} \times 10^4 \) – only for a one-core calculation, which took 0.25 sec (on a 16-core 2.6 GHz AMD Opteron (64-bit) machine with 96 GB RAM, running Ubuntu Linux). In comparison, it took Mathematica just about 0.05 sec to compute the same number, \( B_{10^4} \) (on a roughly comparable 12-core 2.30 GHz Intel Xeon (64-bit) machine with 128 GB RAM, running Windows 7). The programming in [12] was done in C++. Relevant here may be the following quote from [2]: “Mathematica is only about three times slower than C++, but only after a considerable rewriting of the code to take advantage of the peculiarities of the language. The baseline version of our algorithm in Mathematica is considerably slower.” The code in [12] may be significantly more complicated than the code for the Bernoulli numbers used in Mathematica, so that the overhead caused by the complexity of the code used in [12] may be relatively too large for not too large values of \( m \). Note also that it usually takes about 2 to 4 sec to launch several kernels in Mathematica. For all these reasons, it seems to make little (if any) sense to parallelize the calculation of the Bernoulli numbers \( B_2, \ldots, B_{2m-2} \) when \( m \) is not very large.

In contrast with the term \( G_{m,k}^{EM}(c+1) \) in (5.16), the calculation of the term \( G_{m,r}(c) \) in (5.15) can be almost fully parallelized. For large \( m \), of the three expressions (5.4)–(5.6) for \( G_{m,r}(c) \), one should use the one in (5.6) as containing the fewest number \( m \) of summands, versus \( \sim 2m \) summands in (5.5) and \( \sim m^2/2 \) summands in (5.4), cf. Remark 5.4. Next, the values of the function \( F \) in (5.6) (with \( n = c \)) can of course be easily computed in parallel, on several cores.

Also, with a little trick, the calculations of the coefficients

\[ \tau_j := \tau_{m,j} \]

in (5.6) can be almost entirely parallelized, and this can be done so that very little memory space is needed. Indeed, assume for simplicity that the large natural number \( m \) is even. Let \( 0 = m_0, m_1, \ldots, m_k = m \) be even numbers such that

\[ \ell_k := m_k - m_{k-1} \approx m/k_+ \quad \text{(7.1)} \]

so that \( \ell_k \) is even; here and in what follows, \( k \) is an arbitrary number in the set \( \{1, \ldots, k_+\} \). (Note that in this section the meaning of \( m_0 \) is quite different from that in other parts of this paper – such as formula (5.15), for example.)

A particular way to specify values of the \( \ell_k \)'s and \( m_k \)'s is as follows. Let \( q \) and \( r \) denote the nonnegative integers that are, respectively, the quotient and the remainder of the division of \( m/2 \) by \( k_+ \), so that \( m/2 = k_+q + r \) and \( r < k_+ \). Let

\[ \ell_k := 2(q+1\{k \leq r\}) \quad \text{and} \quad m_k := 2(kq + k \land r), \quad \text{(7.2)} \]
where \( I\{A\} \) denotes the indicator of an assertion \( A \). Then indeed the \( m_k \)'s are even numbers, \( m_k = m \) and condition \([7.1]\) holds. Let us also assume the quite natural condition that \( m \) is large enough so as
\[
m \geq 4k, \quad \text{and hence} \quad \ell_k \geq 4 \quad \text{and} \quad \ell_k/2 - 1 \geq 1. \tag{7.3}
\]

By \((5.6)\) with \( \tau_j := \tau_{m,j} \),
\[
G_{m,F}(e) = \sum_{k=1}^{m} \sum_{j=0}^{\ell_k/2-1} \tau_{j,j} H_{j,j} = \Sigma^\text{ev} + \Sigma^\text{od}, \tag{7.4}
\]
\[
H_{j,j} := F(e-j/2) + F(e+j/2-1) I\{j \geq 2\}, \tag{7.5}
\]
\[
\Sigma^\text{ev} := \sum_{j=0}^{\ell_k/2-1} \tau_{j,j} H_{j,j}, \quad \Sigma^\text{od} := \sum_{j=0}^{\ell_k/2-1} \tau_{j,j} H_{j,j},
\]

\( j_k,j := m_k - 2i \); \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm}
\[
\text{here, the superscripts } \text{ev} \text{ and } \text{od} \text{ allude to “even” and “odd”, respectively.}
\]

The key in computing \( \Sigma^\text{ev} \) and \( \Sigma^\text{od} \) is the simple identities
\[
\Sigma^\text{ev} = \tau_{m+2} \eta_{k+2/2-1} + \sigma_{k+2/2-1} \quad \text{and} \quad \Sigma^\text{od} = \tau_{m+1} \eta_{k+2/2-1} + \sigma_{k+2/2-1}, \tag{7.7}
\]
where
\[
\eta_{k,j} := \sum_{i=0}^{\ell_k/2-1} H_{j,i}, \quad \sigma_{k,j} := \sum_{i=0}^{\ell_k/2-1} \theta_{j,i} H_{j,j}, \tag{7.8}
\]
\[
\eta_{k,j}^\text{od} := \sum_{i=0}^{\ell_k/2-1} H_{j+1,i}, \quad \sigma_{k,j}^\text{od} := \sum_{i=0}^{\ell_k/2-1} \theta_{j+1,i} H_{j+1,i}, \tag{7.9}
\]
\[
\theta_{j,0}^\text{ev} := \tau_{j,0} \eta_{k+2} + \gamma_{j,0} \eta_{k+2/2} + \cdots + \gamma_{m_k} \tag{7.10}
\]
\[
\theta_{j,0}^\text{od} := \tau_{j,0} \eta_{k+2} + \gamma_{j,0} \eta_{k+2/2} + \cdots + \gamma_{m_k} \tag{7.11}
\]
and \( \gamma_j := \gamma_{m,j} \); the last equalities in the last two lines of the above display follow by \((3.7)\), which in particular implies \( \tau_{m+2} = 0 = \tau_{m+1} \) for \( k = k_0 \). So,
\[
\theta_{k,0}^\text{ev} = \tau_{k,0} \eta_{k+2} + \gamma_{k,0} \eta_{k+2/2} + \cdots + \gamma_{m_k} \tag{7.12}
\]
\[
\theta_{k,0}^\text{od} = \tau_{k,0} \eta_{k+2} + \gamma_{k,0} \eta_{k+2/2} + \cdots + \gamma_{m_k} \tag{7.13}
\]
where
\[
\bar{\rho}_{k,0}^\text{ev} := \rho_{k,0}^\text{ev} = \frac{\tilde{\rho}_{k,0}^\text{ev}}{\eta_{k,0}} \quad \bar{\rho}_{k,0}^\text{od} := \rho_{k,0}^\text{od} = \frac{\tilde{\rho}_{k,0}^\text{od}}{\eta_{k,0}}. \tag{7.14}
\]
with \( \rho \) defined by \((3.17)\). Recalling that \( m_k \) is even and using also \((3.6), (7.6), \) and \((3.18)\), we have
\[
\tilde{\rho}_{k,0}^\text{ev} = -2 \left( \frac{2m}{m+m_k} \right) / \left( \frac{2m}{m} \right) \quad \text{and} \quad \tilde{\rho}_{k,0}^\text{od} = \frac{m+m_k}{m_k-m-1}, \tag{7.15}
\]
\[
\bar{\rho}_{k,0}^\text{ev} = \frac{m+m_k}{j_{k,0}+1} \quad \text{and} \quad \bar{\rho}_{k,0}^\text{od} = \frac{m+m_k}{j_{k,0}+1} \quad \forall i = 0, \ldots, \ell_k/2 - 1. \tag{7.16}
\]

For each \( k = 1, \ldots, k_0 \), one can compute \( \theta_{k,j}^\text{ev}, \eta_{k,j}^\text{ev}, \sigma_{k,j}^\text{ev}, \theta_{k,j}^\text{od}, \eta_{k,j}^\text{od}, \sigma_{k,j}^\text{od} \) on core \( k \). This is done in \( \ell_k/2 \) steps, indexed by \( i = 0, \ldots, \ell_k/2 - 1 \). At the initial step \( i = 0 \), one can
compute \( \tilde{\rho}_{k,0}^{ev} \) and \( \tilde{\rho}_{k,0}^{od} \) by formula (7.15), then \( \sigma_{k,0}^{ev} = \rho_{k,0}^{ev}/m_k \) and \( \sigma_{k,0}^{od} = \rho_{k,0}^{od}/(m_k - 1) \) by (7.14) and (7.6), then \( \sigma_{k,0}^{ev} \) and \( \sigma_{k,0}^{od} \) by (7.12) and (7.13), and finally
\[
\eta_{k,0}^{ev} = H_{m_k}, \quad \eta_{k,0}^{od} = H_{m_k - 1}, \quad \sigma_{k,0}^{ev} = \eta_{k,0}^{ev}H_{m_k}, \quad \sigma_{k,0}^{od} = \eta_{k,0}^{od}H_{m_k - 1}
\] (7.17)
in accordance with (7.8), (7.9), (7.12), (7.13), and (7.6). After that, at each step \( i = 1, \ldots, \ell_k/2 - 1 \), one computes \( \tilde{\rho}_{k,i}^{ev} \) and \( \tilde{\rho}_{k,i}^{od} \) (in this order) by formula (7.16), then \( \tilde{\sigma}_{k,i}^{ev} = \tilde{\rho}_{k,i}^{ev}/j_{k,i} \) and \( \tilde{\sigma}_{k,i}^{od} = \tilde{\rho}_{k,i}^{od}/(j_{k,i} - 1) \) by (7.14), then \( \theta_{k,i}^{ev} \) and \( \theta_{k,i}^{od} \) by the recursions in (7.12) and (7.13), and finally, in accordance with (7.8) and (7.9), with \( j := j_{k,i} \),
\[
\eta_{k,i}^{ev} = \eta_{k,i-1}^{ev} + H_j, \quad \sigma_{k,i}^{ev} = \eta_{k,i-1}^{ev} + \theta_{k,i}^{ev}H_j, \\
\eta_{k,i}^{od} = \eta_{k,i-1}^{od} + H_j - 1, \quad \sigma_{k,i}^{od} = \eta_{k,i-1}^{od} + \theta_{k,i}^{od}H_j - 1
\] (7.18)
Then, for each \( k = 1, \ldots, k_* \), the six computed values of
\[
\theta_{k, \ell_k/2 - 1}^{ev}, \quad \theta_{k, \ell_k/2 - 1}^{od}, \quad \eta_{k, \ell_k/2 - 1}^{ev}, \quad \eta_{k, \ell_k/2 - 1}^{od}, \quad \sigma_{k, \ell_k/2 - 1}^{ev}, \quad \sigma_{k, \ell_k/2 - 1}^{od}
\] (7.19)
are transmitted from kernel \( k \) (running on core \( k \)) to the master kernel. By (7.10), (7.11), (7.6), and (7.1), \( \theta_{k, \ell_k/2 - 1}^{ev} = \tau_{m_k - 1}^2 - \tau_{m_k - 2} \) and \( \theta_{k, \ell_k/2 - 1}^{od} = \tau_{m_k - 1} - \tau_{m_k - 2} \). So, the values of
\[
\tau_{m_k - 2} = \theta_{k, \ell_k/2 - 1}^{ev} + \cdots + \theta_{k, \ell_k/2 - 1}^{ev} \quad \text{and} \quad \tau_{m_k + 1} = \theta_{k, \ell_k/2 - 1}^{od} + \cdots + \theta_{k, \ell_k/2 - 1}^{od}
\] (7.20)
for all \( k = 1, \ldots, k_* \) can be very quickly computed in the master kernel. Now the calculation of \( G_m,\ell(c) \) can be very quickly completed by (7.4) and (7.7).

We see that the bulk of this calculation is to compute – for each \( k = 1, \ldots, k_* \), on core \( k \) – the values (7.19), which takes \( \ell_k/2 \approx m/(2k_*) \) steps, in view of (7.1).

Each of these steps – except for the initial step, corresponding to \( i = 0 \) – involves just 6 operations of multiplication/division and 8 operations of addition/subtraction of real numbers of a given accuracy (say \( d_1 \), not counting the additions/subtractions of 1 in (7.13) and (7.16). Moreover, at each step \( i = 1, \ldots, \ell_k/2 - 1 \), there is no need to keep in memory the values \( \theta_{k,i-1}^{ev}, \theta_{k,i-1}^{od}, \rho_{k,i-1}^{ev}, \rho_{k,i-1}^{od}, \sigma_{k,i-1}^{ev}, \sigma_{k,i-1}^{od} \) computed at the previous step \( i - 1 \). For instance, the recursion in (7.12) can be realized in programming code as \( \theta_{k,i}^{ev} \leftarrow \theta_{k,i}^{ev} + \rho_{k,i}^{ev} \), with the single value of \( \theta_{k,i}^{ev} \) updated in the memory for each \( i = 1, \ldots, \ell_k/2 - 1 \).

Therefore, all the steps \( i = 1, \ldots, \ell_k/2 - 1 \) together require memory storage of only \( \approx k_* \) real numbers of the accuracy \( d_1 \) – in addition to the memory needed to compute and store the current value of \( H \), defined in (7.5).

The most significant computational difference of the initial step \( i = 0 \) from steps \( i = 1, \ldots, \ell_k/2 - 1 \) is the calculation of the binomial coefficients in (7.15). This can be done very quickly using a version of the divide-and-conquer algorithm, as it is done e.g. by Mathematica. However, then the needed memory storage will be much greater than \( \approx k_* \) of real numbers of the accuracy \( d_1 \).

Yet, it is not hard to figure out how to compute the values of \( \tilde{\rho}_{k,0}^{ev} \) in (7.15) very fast, in an amount of time negligible as compared with the total time to compute \( G_m,\ell(c) \), and still requiring memory storage of only \( \approx k_* \), real numbers of the accuracy \( d_1 \). Indeed, letting \( \tilde{\rho}_{k,0}^{ev} := -2 \), it is easy to see that for each \( k = 1, \ldots, k_* \)
\[
\tilde{\rho}_{k,0}^{ev} = \tilde{\rho}_{k,0}^{ev} N_k,
\]
where
\[
N_k := N_k,\ell_k := \prod_{j=m_k-1+1}^{m_k} v_j = \prod_{i=1}^{m_k-1} v_{m_k-1+i} \quad \text{with} \quad v_j := \frac{j - m - 1}{m + j},
\] (7.21)
can be computed recursively in kernel \( k \) in \( \ell_k \approx m/k_* \) steps, as follows:
\[
N_{k,1} = v_{m_k-1+1}; \quad N_{k,i} = N_{k,i-1} v_{m_k-1+i} \quad \forall i = 2, \ldots, \ell_k.
\]
Similarly to a previous comment, note here that there is no need to keep in memory the value $N_k, i − 1$ computed at the previous step $i − 1$.

Thus – in addition to the calculation of the values of $H_1, …, H_m$ in accordance with (7.5) – the calculation of $G_{m,F}(c)$ requires only $≈ m/k_s$ arithmetical operations (a.o.’s) in each of the parallel $k_s$ kernels plus only $≈ k_s$ a.o.’s in the master kernel, and the required memory storage is only for $≈ k_s$ real numbers of the accuracy $d_1$.

8. Examples

Four examples will be considered in this section, to illustrate applications of the Alt formula to the summation of possibly divergent series and measure its performance in terms of the execution time and memory use, in comparison with the performance of the EM formula – and also with the performance of Mathematica and the Richardson extrapolation process (REP) where the latter tools are applicable. Applications of the REP to summation rely on asymptotic expansions, which appear to have been designed on an ad hoc basis, depending on the function $f$, as was e.g. done in [23, Appendix E.2].

In the first of these examples, considered in Subsection 8.1 both the function $f$ and its antiderivative $F$ are given by rather simple expressions, but high-order derivatives of $f$ are significantly more complicated. As expected, here the Alt formula greatly outperforms the EM one, both in the execution time and memory use.

In the example considered in Subsection 8.2 both the function $f$ and its antiderivative $F$ are complicated, and high-order derivatives of $f$ are significantly more complicated yet. Here the advantages of the Alt formula over the EM one, especially in the execution time, are even more pronounced. In particular, Alt calculations yield 500 (decimal) digits of the results in about 0.8 sec, whereas it takes the EM formula over half an hour to produce just 150 digits.

In the example considered in Subsection 8.3 – where an array of values of the Hurwitz generalized zeta function is computed with various degrees of accuracy – the function $f$, its antiderivative $F$, and high-order derivatives of $f$ are more or less equally complicated. Here the execution time and memory use numbers of the Alt and EM formulas are within a factor of 2 from each other for up to $d = 16 \times 10^3$ digits of accuracy of the result of the calculations, with the EM formula being better for the calculations without parallelization, and the comparison reversed when the calculations are parallelized. In this example, calculations of the coefficients $\tau_{m,r}$ and $B_{2j}$ in the Alt and EM formulas take relatively small fractions of the execution time and memory use. However, for very large values of $d$ it is expected that here too the Alt formula will significantly outperform the EM one even without parallelization, because of the comparatively fast growth in $d$ of the execution time and memory use needed to compute the Bernoulli numbers $B_{2j}$. As for the comparison with Mathematica, both the EM formula and, especially, the Alt one in the parallelized version perform much faster than the (non-parallelizable) built-in Mathematica command `HurwitzZeta[]`; however, `HurwitzZeta[]` is better in terms of memory use than our implementations of the EM and Alt formulas. The Richardson extrapolation scheme (REP) is also applicable here, but its performance in this situation is much inferior even to that of Mathematica’s `HurwitzZeta[]`, let alone the EM and Alt formulas.

Finally, in the example considered in Subsection 8.4 the function $f$ and its high-order derivatives of $f$ are simple, but the antiderivative $F$ of $f$ is significantly more complicated. Here the comparison between the EM and Alt formulas is somewhat similar to that in the previous example; see Subsection 8.4 for details. Again, for very, very large values of the number $d$ of the required digits of accuracy of the result it is expected that in this example too the Alt formula will significantly outperform the EM one even without parallelization. In this situation as well, the REP’s performance is much inferior to that of the EM and Alt formulas.
Recall now again that, to use the EM formula, one will usually need to have an antiderivative \( F \) of \( f \) in tractable/computable form. Then \( f \) (being the derivative of \( F \)) will usually be of complexity no less than that of \( F \). So, the latter two of the four mentioned examples are among the relatively few settings least favorable to the Alt formula, in comparison with the EM one. It may therefore be noted that such unfavorable examples are rather disproportionately represented among the four examples considered in Section 8. Yet, even then, the Alt formula can be expected to significantly outperform the EM one when very high accuracy is needed. In other cases, Alt will usually be significantly better than EM even for moderately high accuracy.

The annotated code and details of the calculations in the mentioned examples are given in the Mathematica notebooks and their pdf images in the zip file AltSum.zip at the Selected Works site

\texttt{https://works.bepress.com/iosif-pinelis/}

Our calculations were verified by comparing with each other the computed decimal approximations to the values of the corresponding generalized sums, \( \sum_{k=0}^{\infty} f(k) \) and \( \sum_{k=0}^{\infty} \text{Alt} f(k) \), and also, in the examples in Subsections 8.3 and 8.4 by comparing those decimal approximations with the ones computed using the built-in Mathematica commands 

\texttt{HurwitzZeta[]}

and 

\texttt{EulerGamma}.

8.1. Example: simple \( f \) and \( F \), complicated derivatives \( f^{(2j)} \). Here we consider summing the divergent series \( \sum_{k=0}^{\infty} f(k) \) according to Corollaries 5.6 and 5.7 where

\[
 f(x) := \frac{3x^3}{\sqrt{x^2 + 1}}. \tag{8.1}
\]

Then for an antiderivative \( F \) of \( f \) one can take the function given by the formula

\[
 F(x) = (x^2 - 2) \sqrt{x^2 + 1}. \tag{8.2}
\]

The remainder \( R_{m,f_c}(\infty) \) in (5.15) will be bounded according to (6.2). Let

\[
 a = -2 \quad \text{and} \quad \lambda = 2. \tag{8.3}
\]

Then for all \( z \in \Pi_a^+ \) (recall here the definition in (6.1)) we have \( |z| \geq \Re z \geq -a = 2 \) and \( |z - 1| \geq |z| - 1 \geq 2 - 1 = 1 \), whence

\[
 |z^2 + 1| \geq |z|^2 - 1 = |z|^2(1 - 1/|z|^2) \geq |z|^2(1 - 1/2^2) = \frac{3}{4} |z|^2,
\]

\[
 |z| \leq |z - 1| + 1 \leq 2|z - 1| \quad \text{and} \quad |f(z)| \leq \frac{3|z|^3}{\sqrt{\frac{3}{4} |z|}} = 2\sqrt{3}|z|^2 \leq 8\sqrt{3}|z - 1|^2 = 8\sqrt{3}|z + a + 1|^2,
\]

so that condition (4.2) holds with \( \mu = 8\sqrt{3} = 13.85\ldots \). Using Remark 4.3 this possible value for \( \mu \) can be a bit improved, to the optimal value

\[
 \mu = \frac{24}{\sqrt{3}} = 10.73\ldots; \tag{8.4}
\]

for details of this calculation, see Mathematica notebook \texttt{sqrt\_mu.nb} and its pdf image \texttt{sqrt\_mu.pdf};

\begin{itemize}
  \item all the files and folders mentioned here and in what follows are contained in the mentioned earlier zip file AltSum.zip at the Selected Works site
  \texttt{https://works.bepress.com/iosif-pinelis/}
\end{itemize}

The value \( \mu \) in (8.4) will be assumed in the rest of the consideration of this example. One may note that with this value of \( \mu \) the inequality in (4.2) turns into the equality for \( z = -a = 2 \).

So, (6.2) will hold if

\[
 m \geq 2 \quad \text{and} \quad c \geq 2 + (m + 3)/2 = (m + 7)/2. \tag{8.5}
\]
By Remark 5.4, Corollaries 5.6 and 5.7 will hold with
\[ m > m_0 = 2, \]
which will also be assumed in this example. Then, by (5.4) and (5.11), one has the following, rather complicated expressions for \( G_{m_0,F}(n) \) and \( G_{m_0,F}^{EM}(n) \):
\[
G_{m_0,F}(n) = \frac{(8n^2 - 8n - 14)\sqrt{n^2 - n + 5/4 - (n^2 - 2n - 1)\sqrt{n^2 - 2n + 2} - (n^2 - 2)\sqrt{n^2 + 1}}}{6}
\]
and
\[
G_{m_0,F}^{EM}(n) = \frac{4n^6 - 18n^5 + 32n^4 - 22n^3 - 15n^2 + 34n - 23}{4(n^2 - 2n + 2)^{3/2}}.
\]
Writing the terms of the form \((n^2 + Bn + C)^p\) in these expressions for \( G_{m_0,F}(n) \) and \( n!G_{m_0,F}^{EM}(n) \) as \( n^2(1 + Be + Ce^2)^p \) with \( e := \frac{1}{n} \), and then expanding \( (1 + Be + Ce^2)^p \) into the powers of \( e \), we have
\[
G_{m_0,F}(n) = n^3 - \frac{3n^2}{2} - n + \frac{3}{4} - \frac{9}{8n} - \frac{9}{16n^2} + \frac{1}{8n^3} + \frac{15}{32n^4} + \frac{31}{128n^5} + O\left(\frac{1}{n^6}\right)
\]
and
\[
G_{m_0,F}^{EM}(n) = n^3 - \frac{3n^2}{2} - n + \frac{3}{4} - \frac{9}{8n} - \frac{9}{16n^2} + \frac{1}{8n^3} + \frac{15}{32n^4} + \frac{19}{128n^5} + O\left(\frac{1}{n^6}\right).
\]
So, for this example, we confirm the asymptotic relation (5.12) and see that here one can replace \( G_{m,F}(n) \) and \( G_{m,F}^{EM}(n) \) in (5.15) and (5.16) by
\[
n^3 - \frac{3n^2}{2} - n + \frac{3}{4}.
\]
We can also see that here in fact \( G_{m_0,F}(n) - G_{m_0,F}^{EM}(n) = O\left(\frac{1}{n^6}\right) \).

In this example, concerning the quantities \( T_f \), \( T_F \), and \( T_\omega \) introduced in Subsection 6.1, we have
\[ T_f \approx T_F >> T_\omega \]
so that, by (6.6), (6.7), (6.9), and (6.10),
\[
\omega \approx \frac{\kappa}{2.5} \approx 0.1, \quad m = m_\omega \approx 0.53d, \quad \text{and} \quad C = c_{d,m_\omega} \approx m_\omega \left( \frac{1}{2} + \frac{\kappa}{\omega L\left(\frac{1}{c_0}\right)} \right) \approx 1.55d.
\]
Accordingly, we will take here
\[ m = 2 \left[ \frac{1}{2} 0.53d \right], \]
for \( m \) to be an even integer, as was assumed in Section 7. However, \( c \) will be taken to be, more accurately, the least integer majorizing the solution for \( c \) of the equation \( R^\ast_{m,c} = \frac{1}{3} 10^{-d} \), where \( R^\ast_{m,c} \) is as in (6.2) and \( d \) is the desired number of known digits of the generalized sum \( \sum_{k \geq 0} f(k) \) in (5.15) after the decimal point:
\[
c = \left\lfloor \frac{m + 1}{2} - a + \left( 2 \times 10^d \frac{1.001 \pi \mu 3^3 \kappa 2^{m-1}}{(2m+1)(2m-1-\lambda)} \right)^{1/(2m-1-\lambda)} \right\rfloor,
\]
with \( \kappa \) as defined in (6.3). Then conditions (8.5) and (8.6) will hold if \( d \geq 6 \) (say), which will of course be the case.

Table 1 presents the wall-clock time (in sec) and the required memory use (in bytes) to compute \( d \) digits of the generalized sum \( \sum_{k \geq 0} f(k) \) in (5.15), based on the Alt summation formula (5.1), for each of the selected values of \( d \), ranging from 2 \times 10^3 to 32 \times 10^4. It appears sensible enough to present the memory use per digit of accuracy. For instance, the recorded memory use \( 20d \) in Table 1 for \( d = 2 \times 10^3 \) and a one-core calculation means that only about 20 bytes of memory were used per one decimal digit of the \( 2 \times 10^3 \) correct computed digits of \( \sum_{k \geq 0} f(k) \). One may recall here that one byte can represent about \( \log_{10}(2^8) \approx 2.41 \) decimal digits. The calculations were done on one core and also on 12 parallel cores of a 12-core 2.30GHz Intel Xeon (64-bit) machine with 128 GB RAM, running Windows 7. The values of \( m \) and \( c \) in (5.15) were chosen, for each selected value of \( d \), in accordance with the above
Discussion. Deployment of multiple parallel cores may take a few seconds; so, it is justified only if the execution time is at least a couple of seconds. Therefore, the very small amount of time of the calculation for \( d = 2 \times 10^3 \) on parallel cores recorded in Table 1 shows that, for \( f \) as in this example, it may make sense to use parallelization only for \( d > 2 \times 10^3 \). One can see that the ratios of the one-core times to the corresponding 12-core times in Table 1 tend to get close to 12, the number of parallel cores, as \( d \) increases from \( 2 \times 10^3 \) to \( 16 \times 10^4 \). The calculation on one core for \( d = 32 \times 10^4 \) has not been done, since it is expected to take a very long time, about 1600 sec \( \times \) 12 \( \approx \) 5 hrs. The annotated code and details of these calculations are given in Mathematica notebooks \sqrt{ASqrt.nb} and \sqrt{AParSqrt.nb} (for the one-core and 12-core Alt calculations, respectively) and their pdf images \sqrt{ASqrt.pdf} and \sqrt{AParSqrt.pdf}.

| Alt | # cores | \( d \)            |
|-----|---------|--------------------|
|     |         | \( 2 \times 10^3 \) | \( 10^4 \) | \( 2 \times 10^4 \) | \( 4 \times 10^4 \) | \( 8 \times 10^4 \) | \( 16 \times 10^4 \) | \( 32 \times 10^4 \) |
|     |         | 0.24              | 5.5     | 150               | 730                | 3600               | –                  |
| time| 1       |                   |         |                   |                    |                    |                    |
| memory | 1     | 20\( d \)         | 12\( d \) | 11\( d \)         | 11\( d \)         | 11\( d \)         | –                  |

Table 1. The wall-clock time (in sec) and memory use (in bytes) to compute \( d \) digits of the generalized sum \( \sum_{k \geq 0}^{\text{Alt}} f(k) \) in (5.15), on one core and on 12 parallel cores, for \( f \) as in (8.1) and each of the listed values of \( d \) – by using (5.15).

| EM  | # cores | \( d \)            |
|-----|---------|--------------------|
|     |         | \( \frac{1}{2} \times 10^3 \) | \( 10^3 \) | \( 2 \times 10^3 \) | \( 4 \times 10^3 \) |
|     |         | 0.69              | 3.0     | 81                | 1600              |
| time | 1       |                   |         |                   |                    |                    |                    |
| memory | 1     | 13 \( \times 10^3 d \) | 21 \( \times 10^3 d \) | 29 \( \times 10^3 d \) | 24 \( \times 10^3 d \) |

Table 2. The wall-clock time (in sec) and memory use (in bytes) to compute \( d \) digits of the generalized sum \( \sum_{k \geq 0}^{\text{EM}} f(k) \) in (5.16), on one core and on 12 parallel cores, for \( f \) as in (8.1) and each of the listed values of \( d \) – by using (5.16).

Table 2 presents the wall-clock times (in sec) and the required memory use (in bytes) to compute \( d \) digits of the generalized sum \( \sum_{k \geq 0}^{\text{EM}} f(k) \) in (5.16), based on the EM summation formula (2.1), for each of the selected values of \( d \), ranging from \( \frac{1}{2} \times 10^3 \) to \( 4 \times 10^3 \). The calculations were done on the same 12-core machine. The values of \( m \) and \( c \) in (5.16) were chosen, for each selected value of \( d \), in accordance with the discussion in Subsection 6.2. However, the value of the exponent \( 2 + \epsilon \) in (6.15) is not so easy to measure, and it may depend on \( d \). Anyhow, reasonably strong efforts were exerted to bracket
the optimal value of $2 + \varepsilon$ and then to choose values of $m$ and $c$ accordingly. The annotated code and
calculations are given in Mathematica notebooks \sqrt\AEMSqrt.nb and \sqrt\AEMParsqrt.nb (for the one-core and 12-core Alt calculations, respectively) and their pdf images \sqrt\AEMSqrt.pdf and \sqrt\AEMParsqrt.pdf. The ratios of the one-core times to the corresponding 12-core times in Table 2 are at best about 2, which reflects the fact that only the calculation of the term $\sum_{k=0}^{4} f(k)$ in the approximation $\sum_{k=0}^{4} f(k) - G_{m,F}^{\text{EM}}(c+1)$ to the generalized sum $\sum_{k=0}^{4} f(k)$ in (5.16) can be easily parallelized.

One can see that, for $f$ as in (8.1), the calculations based on the Alt summation formula (3.1) are
much faster than those based on the EM formula. For instance, the one-core times for $d = 2 \times 10^3$ in Tables 1 and 2 are about 0.24 sec and 81 sec, respectively, with the ratio 81/0.24 ≈ 340. Such ratios seem to grow fast with $d$. For instance, the Alt summation formula (3.1) allows one to compute 2.5 times as many digits ($10^4$ vs. $4 \times 10^3$) in a 0.72/850 ≈ 1/1200 fraction of the time needed by the EM formula.

As for the memory use, one can see from Table 1 that the corresponding numbers for the calculations
according to Corollaries 5.6 and 5.7, where

$$\sum_{k=0}^{4} f(k)$$

according to Corollaries 5.6 and 5.7, where

and the optimal value of $2 + \varepsilon$ and then to choose values of $m$ and $c$ accordingly. The annotated code and
calculations are given in Mathematica notebooks \sqrt\AEMSqrt.nb and \sqrt\AEMParsqrt.nb (for the one-core and 12-core Alt calculations, respectively) and their pdf images \sqrt\AEMSqrt.pdf and \sqrt\AEMParsqrt.pdf. The ratios of the one-core times to the corresponding 12-core times in Table 2 are at best about 2, which reflects the fact that only the calculation of the term $\sum_{k=0}^{4} f(k)$ in the approximation $\sum_{k=0}^{4} f(k) - G_{m,F}^{\text{EM}}(c+1)$ to the generalized sum $\sum_{k=0}^{4} f(k)$ in (5.16) can be easily parallelized.

One can see that, for $f$ as in (8.1), the calculations based on the Alt summation formula (3.1) are
much faster than those based on the EM formula. For instance, the one-core times for $d = 2 \times 10^3$ in Tables 1 and 2 are about 0.24 sec and 81 sec, respectively, with the ratio 81/0.24 ≈ 340. Such ratios seem to grow fast with $d$. For instance, the Alt summation formula (3.1) allows one to compute 2.5 times as many digits ($10^4$ vs. $4 \times 10^3$) in a 0.72/850 ≈ 1/1200 fraction of the time needed by the EM formula.

As for the memory use, one can see from Table 1 that the corresponding numbers for the calculations
according to Corollaries 5.6 and 5.7, where

and the optimal value of $2 + \varepsilon$ and then to choose values of $m$ and $c$ accordingly. The annotated code and
calculations are given in Mathematica notebooks \sqrt\AEMSqrt.nb and \sqrt\AEMParsqrt.nb (for the one-core and 12-core Alt calculations, respectively) and their pdf images \sqrt\AEMSqrt.pdf and \sqrt\AEMParsqrt.pdf. The ratios of the one-core times to the corresponding 12-core times in Table 2 are at best about 2, which reflects the fact that only the calculation of the term $\sum_{k=0}^{4} f(k)$ in the approximation $\sum_{k=0}^{4} f(k) - G_{m,F}^{\text{EM}}(c+1)$ to the generalized sum $\sum_{k=0}^{4} f(k)$ in (5.16) can be easily parallelized.

One can see that, for $f$ as in (8.1), the calculations based on the Alt summation formula (3.1) are
much faster than those based on the EM formula. For instance, the one-core times for $d = 2 \times 10^3$ in Tables 1 and 2 are about 0.24 sec and 81 sec, respectively, with the ratio 81/0.24 ≈ 340. Such ratios seem to grow fast with $d$. For instance, the Alt summation formula (3.1) allows one to compute 2.5 times as many digits ($10^4$ vs. $4 \times 10^3$) in a 0.72/850 ≈ 1/1200 fraction of the time needed by the EM formula.

As for the memory use, one can see from Table 1 that the corresponding numbers for the calculations
according to Corollaries 5.6 and 5.7, where

and the optimal value of $2 + \varepsilon$ and then to choose values of $m$ and $c$ accordingly. The annotated code and
calculations are given in Mathematica notebooks \sqrt\AEMSqrt.nb and \sqrt\AEMParsqrt.nb (for the one-core and 12-core Alt calculations, respectively) and their pdf images \sqrt\AEMSqrt.pdf and \sqrt\AEMParsqrt.pdf. The ratios of the one-core times to the corresponding 12-core times in Table 2 are at best about 2, which reflects the fact that only the calculation of the term $\sum_{k=0}^{4} f(k)$ in the approximation $\sum_{k=0}^{4} f(k) - G_{m,F}^{\text{EM}}(c+1)$ to the generalized sum $\sum_{k=0}^{4} f(k)$ in (5.16) can be easily parallelized.

One can see that, for $f$ as in (8.1), the calculations based on the Alt summation formula (3.1) are
much faster than those based on the EM formula. For instance, the one-core times for $d = 2 \times 10^3$ in Tables 1 and 2 are about 0.24 sec and 81 sec, respectively, with the ratio 81/0.24 ≈ 340. Such ratios seem to grow fast with $d$. For instance, the Alt summation formula (3.1) allows one to compute 2.5 times as many digits ($10^4$ vs. $4 \times 10^3$) in a 0.72/850 ≈ 1/1200 fraction of the time needed by the EM formula.

8.2. **Example: complicated $f$ and $F$, very complicated derivatives $f^{(2)}$.** Here we consider summing
the convergent series $\sum_{k=0}^{\infty} f(k)$ according to Corollaries 5.6 and 5.7, where

$$f(x) := \frac{x}{(x^2 + 2)\sqrt{x^2 + 1}} \text{erf}^{-1} \left( \frac{\arctan \frac{1}{\sqrt{x^2 + 1}}}{\sqrt{x^2 + 1}} \right)$$

(8.9)

for real $x \geq 0$ and $\text{erf}^{-1}$ is the function inverse to the error function erf, given by the formula $\text{erf} w := \frac{2}{\sqrt{\pi}} \int_0^w e^{-v^2} dv$ for $w \in \mathbb{C}$. Then for an antiderivative $F$ of $f$ one can take the function given by the formula

$$F(x) = \frac{1}{\sqrt{\pi}} \exp \left\{ \text{erf}^{-1} \left( \frac{\arctan \frac{1}{\sqrt{x^2 + 1}}}{\sqrt{x^2 + 1}} \right)^2 \right\}.$$  

(8.10)

One can see that Corollaries 5.6 and 5.7 may be applicable, at least in principle, even when the function $f$ is rather far from elementary.

Let

$$a := -3, \quad \varepsilon_1 := \frac{1}{3}, \quad \varepsilon_2 := \frac{38}{100}, \quad \delta := \frac{48}{100}, \quad \kappa_1 := \frac{2}{\sqrt{\pi}} (1 - \delta^2 e^{\delta^2}) \approx 0.80.$$  

For any real $r > 0$, let $D_r := \{ z \in \mathbb{C} : |z| \leq r \}$. For all $v \in D_5$ we have $|1 - e^{-r^2}| = \left| \int_0^{r^2} e^v dv \right| \leq |v|^2 e^{|v|^2} \leq \delta^2 e^{\delta^2}$. So, for any $w_1$ and $w_2$ in $D_5$

$$\frac{\sqrt{\pi}}{2} |\text{erf} w_1 - \text{erf} w_2| = \left| \int_{w_1}^{w_2} e^{-v^2} dv \right| \geq |w_1 - w_2| - \left| \int_{w_1}^{w_2} (1 - e^{-v^2}) dv \right| \geq |w_1 - w_2|(1 - \delta^2 e^{\delta^2}) = \frac{\sqrt{\pi}}{2} \kappa_1 |w_1 - w_2|,$$
so that the function erf is one-to-one on $D_{\delta}$. It also follows that $|\text{erf} w| \geq \kappa_1 |w|$ for all $w \in D_{\delta}$, so that all the points in the image of the boundary of the disk $D_{\delta}$ under the map erf are at distance $\geq \kappa_1 \delta > \varepsilon_2$ from 0. It now follows by Darboux–Picard theorem (see e.g. Subsection 2.1 in [19] or the more general Jordan Filling principle there) that $\text{erf}(D_{\delta}) \supseteq D_{\varepsilon_2}$. Hence, the inverse function $\text{erf}^{-1}$ can be extended to $D_{\varepsilon_2}$, and

$$\text{erf}^{-1}(D_{\varepsilon_2}) \subseteq D_{\delta}. \quad (8.11)$$

Next, for any $v \in \mathbb{C}$ we have

$$|v| \leq \varepsilon_1 \implies |\text{arctan} v| = \left| \int_0^v \frac{du}{1 + u^2} \right| \leq \frac{|v|}{1 - |v|^2} \leq \frac{\varepsilon_1}{1 - \varepsilon_1^2} < \varepsilon_2. \quad (8.12)$$

Further, recall here (6.1) and take any $z \in \Pi_a^+ = \Pi^+_{\frac{1}{2}}$, so that $\Re z \geq 3$. Let $s := (\Re z)^2 \geq a^2 = 9$ and $t := (3z)^2 \geq 0$. Then $|z^2 + 1|^2 = (1 + s - t)^2 + 4st =: h(t)$. Clearly, $h(t)$ is convex in $t$ and $h'(0) = 2(s - 1) \geq 16 > 0$ for $t \geq 0$. So, $|z^2 + 1|^2 \geq h(0) = (1 + s)^2 \geq (1 + a^2)^2$. Thus,

$$z \in \Pi_a^+ \implies \frac{1}{\sqrt{s^2 + 1}} \leq \frac{1}{\sqrt{a^2 + 1}} < \frac{1}{2} = \varepsilon_1. \quad (8.13)$$

Similarly, $\left| \frac{z}{s + 2} \right|^2$ is a simple rational function of $s$ and $t$, which is easy to maximize over $s \geq a^2$ and $t \geq 0$. Actually, in view of the maximum modulus principle, here one may assume that $s = a^2 = 9$. It is then easy to see that the derivative of the mentioned rational function in $t$ is $\leq 0$. So, the maximum of $\left| \frac{z}{s + 2} \right|^2$ in $z \in \Pi_a^+$ is attained at $z = -a = 3$. Hence,

$$z \in \Pi_a^+ \implies \left| \frac{z}{s + 2} \right| \leq \frac{|a|}{s + 2} = \frac{3}{11}. \quad (8.14)$$

Combining this with (8.9), (8.13), (8.12), and (8.11), we have $|f(z)| \leq \frac{3}{11} \frac{1}{2} \delta = \frac{48}{1100}$ for all $z \in \Pi_{-a}$. That is, condition (4.2) will hold with

$$a = -3, \quad \lambda = 0, \quad \text{and} \quad \mu = \frac{48}{1100}, \quad (8.14)$$

which will be assumed in the rest of the consideration of this example. The remainder $R_{m,f}(\infty)$ in (5.15) can now be bounded according to (6.2), which will hold if

$$m \geq 2 \quad \text{and} \quad c \geq 3 + (m + 3)/2 = (m + 9)/2. \quad (8.15)$$

By Remark 5.3 Corollaries 5.6 and 5.7 will hold with

$$m > m_0 = 1, \quad (8.16)$$

which will also be assumed in this example. Then, by (5.4) and (5.11), one has

$$G_{m_0,F}(n) = F(n - 1) = \frac{1}{\sqrt{\pi}} \exp \left\{ \text{erf}^{-1} \left( \frac{1}{\sqrt{(n - 1)^2 + 1}} \right)^2 \right\} = \frac{1}{\sqrt{\pi}} + o(1)$$

and

$$G_{m_0,F}^\text{EM}(n) = F(n - 1) + \frac{1}{2} f(n - 1) = \frac{1}{\sqrt{\pi}} + o(1).$$

So, for this example as well, we confirm the asymptotic relation (5.12).

In this example, as in the example in Subsection 8.1 we have $T_f \approx T_F >> T_\varepsilon$. (In fact, here the values of $T_f$ and $T_F$ are much greater than those in the example in Subsection 8.1 so that here the comparison $>> T_\varepsilon$ is even more pronounced.) So, here we still take $m$ and $c$ according to (8.7) and (8.8).

Tables 3 and 4 are similar to Tables 1 and 2 respectively. The main difference is that here the values of $d$ are significantly smaller than the corresponding values in the example in Subsection 8.1 because the execution times in this setting are much greater. The annotated code and details of these calculations are given in Mathematica notebooks \texttt{\textbackslash erfInvAtan\textbackslash AErFInvAtan.nb}, \texttt{\textbackslash erfInvAtan\textbackslash APArErfInvAtan.nb}, \texttt{\textbackslash erfInvAtan\textbackslash AParErfInvAtan.nb}, \texttt{\textbackslash erfInvAtan\textbackslash AParErfInvAtan.nb}, ...
Table 3. The wall-clock time (in sec) and memory use (in bytes) to compute $d$ digits of the generalized sum $\sum_{k \geq 0} f(k)$ in (5.15), on one core and on 12 parallel cores, for $f$ as in (8.9) and each of the listed values of $d$ – by using (5.15).

| # cores | $d$       |
|--------|----------|
|        | $\frac{1}{2} \times 10^3$ | $10^3$ | $2 \times 10^3$ | $4 \times 10^3$ | $8 \times 10^3$ |
| time   | 1        | 5.4    | 30             | 240             | 1800            | –             |
| memory | 1        | 8.2 $\times 10^3 d$ | $11 \times 10^3 d$ | $14 \times 10^3 d$ | $16 \times 10^3 d$ | –             |
| time   | 12       | 0.78   | 4.0            | 29              | 210             | 1900          |
| memory | 12       | $21 \times 10^3 d$ | $16 \times 10^3 d$ | $20 \times 10^3 d$ | $30 \times 10^3 d$ | $50 \times 10^3 d$ |

Table 4. The wall-clock time (in sec) and memory use (in bytes) to compute $d$ digits of the generalized sum $\sum_{k \geq 0} f(k)$ in (5.16) on one core and on 12 parallel cores, for $f$ as in (8.9) and each of the listed values of $d$ – by using (5.16).

| # cores | $d$       |
|--------|----------|
|        | 20       | 50      | 100        | 150        |
| time   | 1        | 1.1     | 33         | 670        | –           |
| memory | 1        | $61 \times 10^3 d$ | $560 \times 10^3 d$ | $1100 \times 10^3 d$ | – |
| time   | 12       | 0.62    | 14         | 300        | 1900        |
| memory | 12       | $190 \times 10^3 d$ | $1600 \times 10^3 d$ | $1600 \times 10^3 d$ | $2700 \times 10^3 d$ |

We see that here the Alt formula operates on a much higher scale than the EM one. Based on Table 4, a rather conservative prediction of the execution times on the 12 cores using the EM formula for $d = \frac{1}{2} \times 10^3, 10^3, 2 \times 10^3, 4 \times 10^3,$ and $8 \times 10^3$ would be about 2.2 days, 35 days, 1.5 years, 25 years, and 390 years – versus the corresponding execution times 0.78 sec, 4.0 sec, 29 sec, 210 sec, and 1900 sec in Table 3. The predicted memory use for the same values of $d$ would be about 7 GB, 33 GB, 150 GB, 730 GB, and 3400 GB – versus the corresponding memory use numbers $21 \times 10^3 \times \frac{1}{2} \times 10^3 B = 0.011 GB, 16 \times 10^3 \times 10^3 B \approx 0.016 GB, 20 \times 10^3 \times 2 \times 10^3 B \approx 0.04 GB, 30 \times 10^3 \times 4 \times 10^3 B \approx 0.12 GB,$ and $50 \times 10^3 \times 8 \times 10^3 B \approx 0.4 GB$ in Table 3. Details on these predictions can be found in Mathematica notebook \erfInvAtan\prediction.nb and its pdf image \erfInvAtan\prediction.pdf.

8.3. Example (Calculation of an array of values of the Hurwitz generalized zeta function): complicated $f$ and $F$, comparatively simple derivatives $f^{(2j)}$, Both the EM formula and the Alt summation formula are applicable when the function $f$ takes values in any normed space, rather than just real values. In particular, $f$ may be complex-valued. For instance, let us consider in this example the vector function

$$f := (f_{-1+i}, f_{i}, f_{1+i}, f_{2+i}),$$

(8.17)
with values in $\mathbb{C}^4$, where

\[ f_{p, \delta}(x) := (x + \delta)^{-p} \quad \text{(8.18)} \]

for $x \geq 0$, with $\delta \in \mathbb{C} \setminus \{0, -1, -2, \ldots\}$ and $p \in \mathbb{C}$; in this context, of course $i$ denotes the imaginary unit. As usual, for any $z \in \mathbb{C} \setminus \{0\}$ with $\theta := \arg z \in (-\pi, \pi]$, we let $z^p := |z|^p e^{i\theta p}$. So, the vector series corresponding to the vector function $f$ in (8.17) is

\[ \sum_{k=0}^{\infty} f(k) = \left( \sum_{k=0}^{\infty} (k+i)^{1-i} \right) \sum_{k=0}^{\infty} (k+i)^{-i} \sum_{k=0}^{\infty} (k+i)^{-1-i} \sum_{k=0}^{\infty} (k+i)^{-2-i} \).

Each of the first three components of this vector series is a divergent series in $\mathbb{C}$, whereas the fourth component is a convergent series in $\mathbb{C}$ — whose value is $\zeta(2+i,i)$, where $\zeta$ is the Hurwitz generalized zeta function. The expression $\zeta(p, 1)$ defines the Riemann zeta function.

Usually (see e.g. [1, page 15]), $\zeta(p, \delta)$ is defined only for real $\delta > 0$ — initially, as $\sum_{k=0}^{\infty} f_{p, \delta}(k)$ for $p \in \Pi^+_1$ (where $\Pi^+_1 := \{\zeta \in \mathbb{C} : \Re \delta > a\}$ for real $a$; cf. the definition of $\Pi^+_1$ in (6.11)), and then extended by analytic continuation to all $p \in \mathbb{C} \setminus \{1\}$. However, $\zeta(p, \delta)$ can actually be defined for all $\delta \in \mathbb{C} \setminus (-\infty, 0]$ and all $p \in \mathbb{C} \setminus \{1\}$. One way to do this is as follows.

Take any $\delta \in \mathbb{C} \setminus (-\infty, 0]$. Again, consider first the case when $p \in \Pi^+_1$. Then, by Proposition 5.3 with $m \geq m_0 = 1$,

\[ \zeta(p, \delta) = Z_m(p, \delta) := f_{p, \delta}(0) - G_{m,f_{p, \delta}}^{EM}(1) + R_{m,f_{p, \delta}}^{EM}(\infty), \quad \text{(8.19)} \]

where

\[ F_{p, \delta}(x) := \frac{1}{x^p} (x + \delta)^{1-p}. \quad \text{(8.20)} \]

It is easy to see that $Z_m(p, \delta)$ is analytic in $p \in \Pi^+_1 \setminus \{1\}$. Note also that

\[ p \in \Pi^+_1 \setminus \{1\} \iff m > 1 - \frac{1}{2} \Re p. \quad \text{(8.21)} \]

So, the formula $\zeta(p, \delta) := Z_m(p, \delta)$ for any $p \in \mathbb{C} \setminus \{1\}$ and any natural $m > 1 - \frac{1}{2} \Re p$ provides a well-defined analytic continuation $\mathbb{C} \setminus \{1\} \ni p \mapsto \zeta(p, \delta)$, for each $\delta \in \mathbb{C} \setminus (-\infty, 0]$. In fact, this extension is analytic in $\delta \in \mathbb{C} \setminus (-\infty, 0]$ as well.

Moreover, again by Proposition 5.3 but now with any

\[ m_0 > 1 - \frac{1}{2} \Re p \quad \text{and} \quad m \geq m_0 \quad \text{(8.22)} \]

(cf. (8.21)), we deduce from (8.19), Corollary 5.7, Proposition 5.5 and Corollary 5.6 that

\[ \zeta(p, \delta) = \sum_{k=0}^{\infty} f_{p, \delta}(k) = \sum_{k=0}^{c} f_{p, \delta}(k) - G_{m,f_{p, \delta}}^{EM}(c + 1) + R_{m,f_{p, \delta}}^{EM}(\infty) \]

or

\[ \sum_{k=0}^{\infty} f_{p, \delta}(k) = \sum_{k=0}^{c-1} f_{p, \delta}(k) - G_{m,f_{p, \delta}}^{EM}(c) + R_{m,f_{p, \delta}}^{EM}(\infty) \]

for any $p \in \mathbb{C} \setminus \{1\}$, any $\delta \in \mathbb{C} \setminus (-\infty, 0]$, any natural $c$, and any $m$ and $m_0$ as in (8.22).

So,

\[ \left( \zeta(-1+i,i), \zeta(i,i), \zeta(1+i,i), \zeta(2+i,i) \right) = \sum_{k=0}^{c} f(k) - G_{m,f(c+1)}^{EM} + R_{m,f(c)}^{EM}(\infty) \quad \text{(8.23)} \]

\[ = \sum_{k=0}^{c-1} f(k) - G_{m,f(c)} + R_{m,f(c)}^{EM}. \quad \text{(8.24)} \]

where $f$ is the vector function as in (8.17),

\[ F := (F_{-1+i,i}, F_{i,i}, F_{1+i,i}, F_{2+i,i}) \]
is a vector function with values in $\mathbb{C}^d$ which is an antiderivative of $f$, $F_{p, \delta}$ as in (8.20), $c$ is any natural number, and $m$ and $m_0$ are as in (8.6). Thus, formulas (8.23) and (8.24) present as a way to compute an array of values of the Hurwitz generalized zeta function.

Suppose that a real number $a$ is such that $a \leq \Re \delta - 1$. Recall then (6.1) and take any $z \in \Pi_{\delta}^+$, so that $\Re z \geq -a$, whence $|z + a + 1| \geq \Re z + a + 1 \geq 1$,

$$|z + \delta| \leq |z + a + 1| + |\delta - a - 1| \leq (1 + |\delta - a - 1|)z + a + 1,$$

$$|z + \delta| \geq \Re (z + \delta) \geq -a + \Re \delta \geq 1.\text{ Therefore, if } \Re p < 0, \text{ then }$$

$$|f_{p, \delta}(z)| \leq |z + \delta|^{-\Re p} e^{|\Im p|/2} \leq e^{\pi|\Im p|/2} (1 + |\delta - a - 1|)^{-\Re p} |z + a + 1|^{-\Re p},$$

so that condition (4.2) holds with $\mu = e^{\pi|\Im p|/2} (1 + |\delta - a - 1|)^{-\Re p}$, $\lambda = -\Re p$, and $f_{p, \delta}$ in place of $f$. If $\Re p \geq 0$, then

$$|f_{p, \delta}(z)| \leq e^{\pi|\Im p|/2},$$

so that condition (4.2) holds with $\mu = e^{\pi|\Im p|/2}$, $\lambda = 0$, and $f_{p, \delta}$ in place of $f$. Thus, for each coordinate of the function $f$ in (8.17), condition (4.2) holds with

$$a = -1, \quad \lambda = 1, \quad \mu = 2e^{\pi/2}, \quad (8.25)$$

which will be assumed in the rest of the consideration of this example. The remainder $R_{m, f}(\infty)$ in (5.15) can now be bounded according to (6.2), which will hold if (8.15) holds. By Remark 5.4, Corollaries 5.6 and 5.7 will hold if (8.6) holds, which will also be assumed in this example. Writing the terms of the form $(n + \theta)^{1-p}$ in the expressions in (5.34) and (5.11) for $G_{m_0, F}(n)$ and $G_{m_0}^{EM}(n)$ as $n^{1-p}(1 + \theta \epsilon)^{1-p}$ with $\epsilon := \frac{1}{n}$, and then expanding $(1 + \theta \epsilon)^{1-p}$ into the powers of $\epsilon$, we have

$$G_{m_0, F}(n) = \frac{1}{2}F(n - \frac{1}{2}) - \frac{1}{6}F(n - 1) - \frac{1}{6}F(n) - G(n) + O\left(\frac{1}{n}\right)$$

and

$$G_{m_0}^{EM}(n) = F(n - 1) + \frac{1}{2}f(n - 1) + \frac{B_2}{2}f(n - 1) - \tilde{G}(n) + O\left(\frac{1}{n}\right),$$

where

$$\tilde{G}(n) := \frac{1}{n} \left( \sum_{n}^{30} \frac{1}{12(2i)}, 2^{\pi+1+3\pi} \frac{1}{2(1-i)}, \frac{1}{1} - \epsilon, 0 \right)$$

and $O\left(\frac{1}{n}\right)$ stands for a vector of the form $(O\left(\frac{1}{n}\right), O\left(\frac{1}{n}\right), O\left(\frac{1}{n}\right), O\left(\frac{1}{n}\right))$. So, for this example as well, we confirm the asymptotic relation (5.12). One may also note here that the factor $\frac{1}{n} = e^{-\ln n}$ in the above expression for $\tilde{G}(n)$ is of modulus 1, and it oscillates in $n$ with frequency $\frac{\ln n}{2\pi e} \to 0$.

In this example, as in the example in Subsection 8.1, we have $T_f \approx T_F > T_\epsilon$. (In fact, here the values of $T_f$ and $T_F$ are much greater than those in the example in Subsection 8.1, so that here the comparison $T_\epsilon$ is even more pronounced.) Therefore, here we still take $m$ and $c$ according to (8.7) and (8.8).

Tables 3 and 6 are similar to Tables 1 and 2 respectively. As in the example in Subsection 8.2, here the values of $d$ are significantly smaller than the corresponding values in the example in Subsection 8.1 because the execution times in this setting are much greater. The annotated code and details of these calculations are given in Mathematica notebooks \vector\AVect.nb, \vector\AParVect.nb, \vector\EMVect.nb, \vector\EMPParVect.nb, and their pdf images, with file name extension .pdf in place of .nb.

The main difference between the present example and the examples in Subsections 8.1 and 8.2 is that in this exceptional case the values of the higher-order derivatives of $f$ are about as easy to compute as the values of the function $f$ itself, since

$$f_{p, \delta}^{(2j-1)}(x) = -p(p + 1) \cdots (p + 2j - 2)(x + \delta)^{-p - 2j + 1} \quad (8.26)$$
for \( j = 1, 2, \ldots \) and \( x \geq 0 \). Accordingly, we see that the numbers in Table 5 are of the same order of magnitude as the corresponding numbers in Table 6; the latter numbers are somewhat better for the one-core calculations and somewhat worse for the 12-core ones.

The execution time and memory use measurements reported in Tables 5 and 6 may be compared with the corresponding measurements for the Mathematica built-in command \texttt{HurwitzZeta[]} , which is one of the Mathematica commands that cannot be parallelized (in Mathematica). The execution time and memory use for \texttt{HurwitzZeta[]} are reported in Table 7 for details, see again Mathematica notebook \texttt{\vector\AEMVect.nb} and its pdf image \texttt{\vector\AEMVect.pdf}. The label \( \text{Ma} \) in the upper left cell in Table 7 indicates the use of Mathematica.

One can see that, in terms of the execution time, our implementation of the EM formula performs significantly better than the built-in command \texttt{HurwitzZeta[]} , being about 6 times as fast for \( d = 1000 \) digits of accuracy and about 14 times as fast for \( d = 8000 \). In particular, this suggests that our code of the calculations by the EM formula is rather efficiently optimized for the execution time.

The Alt summation formula performs even better, especially in the parallelized version, being about 10 times as fast as \texttt{HurwitzZeta[]} for \( d = 1000 \) and about 23 times as fast for \( d = 8000 \).
Ma | # cores | \( d \)
---|---|---
| | \( \times 10^3 \) | \( \times 10^3 \) | \( \times 10^3 \) | \( \times 10^3 \)
| time | 1 | 6.9 | 44 | 310 | 2300
| memory | 1 | 730d | 310d | 410d | 710d

Table 7. The wall-clock time (in sec) and memory use (in bytes) to compute \( d \) digits of \( \sum_{k \geq 0} f(k) = \sum_{k \geq 0} \sum_{i=0}^{\infty} f(k) = (\zeta(-1+i,i), \zeta(i,i), \zeta(1+i,i), \zeta(2+i,i)) \) on one core, for \( f \) as in (8.17) and each of the listed values of \( d \) – by using the (non-parallelizable) built-in Mathematica command `HurwitzZeta[]`.

On the other hand, `HurwitzZeta[]` is substantially better in terms of memory use than our implementations of the EM and Alt summation formulas.

It was also suggested to the author of this paper to consider the performance of the Richardson extrapolation process (REP), described e.g. in [23]. Results of the corresponding numerical experiment are reported in Table 8. Details are given in Mathematica notebook `\vector\richardsonVect.nb` and its pdf image `\vector\richardsonVect.pdf`. The label REP in the upper left cell in Table 8 (as well as in Table 11 to follow) indicates the use of the REP.

REP | # cores | \( d \)
---|---|---
| | 23 | 31 | 40 | 49
| time | 1 | 9.3 | 36 | 160 | 670
| memory | 1 | \( 2.0 \times 10^6 d \) | \( 5.7 \times 10^6 d \) | \( 19 \times 10^6 d \) | \( 62 \times 10^6 d \)

Table 8. The wall-clock time (in sec) and memory use (in bytes) to compute \( d \) digits of \( (\zeta(-1+i,i), \zeta(i,i), \zeta(1+i,i), \zeta(2+i,i)) \) (on one core), for \( f \) as in (8.17) and each of the listed values of \( d \) – by using the (non-parallelizable) REP.

One can see that that the convergence of the REP is very slow, in comparison: to gain just 8 or 9 digits of accuracy, one needs to quadruple the execution time. At this rate, it would take the REP about \( \frac{670}{160} \times 4^{95} \approx 3 \times 10^{32} \) years to compute 1000 digits of the generalized sum – whereas the corresponding calculation using the Alt summation formula takes only 0.70 sec, as shown in Table 5.

The main underlying reason for this stark contrast seems to be the fact that the REP takes into account only the exponents – but not the coefficients – in the asymptotic expansions on which that method is based; see lines 2–3 after formula (1.1.2) on page 21 in [23]. Thus, much of the available information is neglected in the REP.

Also, because of its recursive nature, it appears that calculations by the REP cannot be parallelized. Approximations to the value \( \zeta(2) = \zeta(2,1) = \frac{\pi^2}{6} \) of the Riemann zeta function were also computed in [23] by the GREP (Generalization of the Richardson Extrapolation Process). However, as stated on page 138 in [23], with one choice of the relevant parameters of the GREP, “[a]dding more terms to the process does not improve the accuracy; to the contrary, the accuracy dwindles quite quickly. With the [other choice of the parameters], we are able to improve the accuracy to almost machine precision.” So, at least in this case, GREP provides much less accuracy than even the original REP – cf. Table 8.

As noted on page 57 in [23] concerning generalizations of the REP, “[the] problems of convergence and stability [...] turn out to be very difficult mathematically, especially because of this generality. As a
result, the number of the meaningful theorems that have been obtained and that pertain to convergence
and stability has remained small.”

In contrast with the explicit and rather easy to use upper bounds on the remainders for EM formula
and the Alt summation formula – such as the ones given by (4.5) and (4.9) – only error bounds of generic
form $O(\cdot)$ seem to be available for the REP, without specification of the corresponding constant factors. 
This makes it more difficult to design calculations by the REP or its generalizations.

No specific execution time or memory use data on the REP or its generalizations seem to be given in
or elsewhere.

8.4. Example (Calculation of the Euler constant): simple $f$, comparatively complicated $F$, and
simple derivatives $f^{(2j)}$. The Euler constant is defined by the formula
\[
\gamma_{Eu} := \lim_{n \to \infty} (\mathcal{H}_n - \ln n),
\]
where
\[
\mathcal{H}_n := \sum_{\alpha=1}^{n} \frac{1}{\alpha},
\]
the $n$th harmonic number. Let here
\[
f(x) := \frac{1}{x + 1},
\]
with the antiderivative
\[
F(x) := \ln(x + 1)
\]
for real $x \geq 0$. Note that here values of $F$ are significantly harder to compute with high accuracy than
values of $f$.

Take any natural $c \geq (m + 3)/2$. Since $\Re z \geq 0$ implies $|z + 1| \geq \Re z + 1 \geq 1$, all the conditions of
Proposition $4.2$ will hold for $f_c$ and $a_c := c + a$ in place of $f$ and $a$ (respectively) if $a = 0$, $\lambda = 0$, and
$\mu = 1$. So, by Remark $5.4$, Corollaries $5.6$ and $5.7$ will hold with $m > m_0 = 1$, which will be assumed
in this example. In view of Corollary $5.6$, (5.6), (8.30), and (3.9), here the generalized sum $\sum_{k \geq 0} f(k)$ in
(8.27) equals
\[
\gamma_{Eu} = \mathcal{H}_c + \ln 2 - \tau_{m,1} \ln(2c + 1) - \sum_{j=2}^{m} \tau_{m,j} \ln((2c + 1)^2 - (j - 1)^2) - R_{m,f,(\infty)}.
\]
To obtain the latter expression, we rewrote the term $F(c - 1/2 - \alpha/2) + F(c - 1/2 + \alpha/2) = \ln(c + 1/2 - \alpha/2) + \ln(c + 1/2 + \alpha/2)$ in the expression of $G_{m,F}(c)$ in (8.27) (cf. (5.6)) as $-2 \ln 2 + \ln((2c + 1)^2 - (j - 1)^2)$ for $j := 1 + \alpha$, which allows one to almost halve the number of harder-to-
compute values of the logarithm function.

By Remark $4.1$, one may take $M_{2m} = \frac{(2m - 1)!}{(c - m/2 + 1/2)2m}$, whence, by (3.11) and (4.7), here we have
\[
|R_{m,f,(\infty)}| \leq R_{m,c}^{\text{alt}} \cdot \frac{0.001 \pi}{(2m + 1)2m} \left( \frac{\Lambda}{4} \right)^m \frac{1}{(c - m/2 + 1/2)2m},
\]
which in this particular case yields a slight improvement on the general bound $R_{m,c}^{\text{alt}}$ in (6.2) on $|R_{m,f,(\infty)}|$, and
the previously assumed condition $c \geq (m + 3)/2$ can now be relaxed to $c > (m - 1)/2$.

Similarly, in view of Corollary $5.7$, (5.11), and (8.30), the generalized sum $\sum_{k \geq 0} f(k)$ in (5.11) equals
\[
\gamma_{Eu} = \mathcal{H}_{c+1} - \ln(c + 1) - \sum_{j=1}^{m-1} \frac{B_{2j}}{2j(c + 1)^{2j}} + R_{m,f,(\infty)}^{\text{alt}}.
\]
By (2.4), assuming $m \geq 4$, here we have

$$|R_{m,f_c}(\infty)| \leq R_{m,c}^{EM,*} := \frac{2.02(2m - 2)!}{(2\pi)^{2m-1}(c + 1)^{2m-1}},$$

which in this particular case yields a slight improvement on the general bound $R_{m,c}^{EM,*}$ in (6.12) on $|R_{m,f_c}(\infty)|$.

In this example, in contrast with the previous ones, time measurements show that in the considered range of values of $d$ one has $T_F \approx 10d^{1/2}T_f$; see Mathematica notebooks `\euler/1-over-k-timing.nb` and `\euler/log-timing.nb` and the corresponding pdf images for details. So, here $T_F \gg T_f \lor T_t$.

Also, by the mentioned “halving” the number of “costly” values of $F$ to compute, here we should replace the term $2T_F$ in the expression of $K$ in (6.6) by $T_F$. So,

$$\omega \approx \omega_d := \frac{\kappa}{10d^{1/2}} \text{ and } m = m_\omega,$$

in accordance with (6.9).

Accordingly, we will take here

$$m = 2 \left\lfloor \frac{1}{2} m_\omega \right\rfloor,$$

for $m$ to be an even integer, as was assumed to be in Section 7. Formula (8.32) is replaced here by

$$c = \left[ \frac{m - 1}{2} + \left( 2 \times 10^d \cdot \frac{1.001 \kappa^2 m^{2m+1}}{(m + 1)2m} \right)^{1/(2m)} \right],$$

since the bound $R_{m,c}^{*}$ in (8.31) here replaces the bound $R_{m,c}^{*}$ in (6.2) on $|R_{m,f_c}(\infty)|$.

Then conditions $c > (m - 1)/2$ and $m > m_0 = 1$ will hold.

Tables 9 and 10 are similar to Tables 1 and 2, respectively. The annotated code and details of these calculations are given in Mathematica notebooks `\euler/AEuler.nb`, `\euler/AParEuler.nb`, `\euler/AEMEuler.nb`, `\euler/AEMParEuler.nb`, and their pdf images, with file name extension .pdf in place of .nb.

| Alt | # cores | $d$ |
|-----|---------|-----|
|     |         | 10^3 | $2 \times 10^3$ | $4 \times 10^3$ | $8 \times 10^3$ | $16 \times 10^3$ | $32 \times 10^3$ | $64 \times 10^3$ | $128 \times 10^3$ |
| time | 1       | 0.09 | 0.29 | 1.1 | 6.0 | 31 | 150 | 820 | – |
| memory | 1 | 270d | 270d | 320d | 420d | 620d | 1500d | 680d | – |
| time | 12 | 0.10 | 0.13 | 0.26 | 0.87 | 3.9 | 21 | 100 | 510 |
| memory | 12 | 1900d | 1700d | 2200d | 2200d | 6300d | 13000d | 240d | 230d |

Table 9. The wall-clock time (in sec) and memory use (in bytes) to compute $d$ digits of the Euler constant, on one core and on 12 parallel cores, for $f$ as in (8.29) and each of the listed values of $d$ – by using (5.15).

We have also considered the performance of the Richardson extrapolation process (REP) in the present example. Results of the corresponding numerical experiment are reported in Table 11. Details are given in Mathematica notebook `\euler/richardsonEuler.nb` and its pdf image `\euler/richardsonEuler.pdf`.

Projections based on the data presented in Table 11 suggest that it would take the REP about $18 \times 10^4$ years to compute 1000 digits of $\gamma_{Eu} = \sum_{k=0}^{\infty} f(k) = \sum_{k=0}^{EM} f(k)$ (see again Mathematica notebook
APPROXIMATING SUMS BY INTEGRALS ONLY

EM

| # cores | \(d\) | \(10^3\) | \(2 \times 10^3\) | \(4 \times 10^3\) | \(8 \times 10^3\) | \(16 \times 10^3\) | \(32 \times 10^3\) | \(64 \times 10^3\) | \(128 \times 10^3\) |
|---------|------|--------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| time | 1 | 0.06 | 0.26 | 1.0 | 5.7 | 14 | 70 | 390 | – |
| memory | 1 | 130\(d\) | 110\(d\) | 140\(d\) | 240\(d\) | 1700\(d\) | 2000\(d\) | 3100\(d\) | – |
| time | 12 | 0.07 | 0.17 | 0.66 | 3.0 | 15 | 71 | 470 | 2600 |
| memory | 12 | 150\(d\) | 170\(d\) | 160\(d\) | 240\(d\) | 500\(d\) | 1000\(d\) | 25000\(d\) | 39000\(d\) |

Table 10. The wall-clock time (in sec) and memory use (in bytes) to compute \(d\) digits of the Euler constant on one core and on 12 parallel cores, for \(f\) as in (8.29) and each of the listed values of \(d\) – by using (5.16).

REP

| # cores | \(d\) | 74 | 92 | 113 | 136 | 161 | 188 | 218 |
|---------|------|----|----|----|----|----|----|----|
| time | 1 | 0.36 | 1.4 | 2.6 | 19 | 75 | 320 | 1300 |
| memory | 1 | 590\(d\) | 520\(d\) | 510\(d\) | 420\(d\) | 400\(d\) | 380\(d\) | 360\(d\) |

Table 11. The wall-clock time (in sec) and memory use (in bytes) to compute \(d\) digits of the Euler constant on one core, for \(f\) as in (8.29) and each of the listed values of \(d\) – by using the (non-parallelizable) REP.

\texttt{euler/richardsonEuler.nb} and its pdf image \texttt{euler/richardsonEuler.pdf} for details - compared with 0.06 sec by the EM formula itself, without the Richardson extrapolation.

The present example is to an extent similar to the example in Subsection 8.3, the main difference between these two examples being that here the time \(T_F\) to compute a value of \(F\) is much greater for large \(d\) than the time \(T_f\) to compute a value of \(f\). On the other hand, the main difference between the present example and the examples in Subsections 8.1 and 8.2 is that in this exceptional case as well the values of the higher-order derivatives of \(f\) are about as easy to compute as the values of the function \(f\) itself; see (8.26) with \(p = 1\) and \(\delta = 1\). Thus, the present example represents one of the few least favorable situations for the Alt summation formula in comparison with the EM one.

Yet, we see that, as in the example in Subsection 8.3, the execution time numbers in Table 9 are of the same order of magnitude as the corresponding numbers in Table 10. The latter numbers are somewhat better for the one-core calculations and somewhat worse for the 12-core ones. However, according to [6], \(d\) values of the logarithmic function can be computed with \(\approx d\) digits of precision in time \(T_{\text{log}} \approx M(d) d \log d \approx d^2 \log d \log \log d\), where \(M(d) \approx d \log d \log \log d\) is the time needed to perform precision \(d\) multiplication. On the other hand, the best known algorithms for Bernoulli numbers [9, 12] will compute the first \(d\) Bernoulli numbers with with \(\approx d\) digits of precision in time \(\approx (d^3/\log d) M(d \log d) \approx d^3 \log d \log \log d \gg T_{\text{log}}\). So, for very large \(d\), it should be expected that even on one core the Alt summation formula will perform faster than the EM one.

The memory use numbers in Table 9 are more or less similar to the corresponding numbers in Table 10 for the one-core calculations, but about 10 times as large for the 12-core ones. One may note the big drop in the reported memory use from 13000\(d\) to 240\(d\) in the last row of Table 9 (and also the smaller...
drop 1500$d$ to 680$d$ in the one-core memory use row of the same table) when $d$ increases from $32 \times 10^3$ to $64 \times 10^3$.

Quite a similar drop occurs in a much simpler, distilled version of this situation, when just the sum of a large number of values of the logarithmic function is computed with high accuracy; see details in Mathematica notebook \euler\memoryMatter.nb and its pdf image \euler\memoryMatter.pdf. Such a drop has not been observed in Maple [13]. According to a representative of Wolfram Research (the developer of Mathematica), the drop occurs because Mathematica switches from one method to another depending on various parameters of the computational process. Somewhat similar drops (in that case not only in the memory use but also in the execution time) occur in Mathematica calculations of the Bernoulli distribution.

9. PROOFS

Proof of Theorem 3.1 Take any $k = 0, \ldots, n - 1$ and consider the Taylor expansion

$$f(x) = \sum_{i=0}^{2m-1} \frac{f^{(i)}(k)}{i!} u^i + \frac{u^{2m}}{(2m-1)!} \int_0^1 ds (1 - s)^{2m-1} f(2m)(k + su)$$

for all $x \in (k - m/2, k + m/2]$, where $u := x - k$. Integrating both sides of this identity in $x \in (k - j/2, k + j/2]$ (or, equivalently, in $u \in (-j/2, +j/2]$) for each $j = 1, \ldots, m$, then multiplying by $\gamma_{m,j}$, and then summing in $j$, one has

$$A_{m,k} = S_{m,k} + R_{m,k},$$

where

$$A_{m,k} := \sum_{j=1}^{m} \gamma_{m,j} \int_{k-j/2}^{k+j/2} dx f(x),$$

$$S_{m,k} := \sum_{\alpha=0}^{m-1} f(2\alpha)(k) \sum_{j=1}^{m} \gamma_{m,j} j^{2\alpha+1},$$

$$R_{m,k} := \frac{1}{(2m-1)!} \int_0^1 ds (1 - s)^{2m-1} \sum_{j=1}^{m} \gamma_{m,j} \int_{-j/2}^{j/2} du u^{2m} f(2m)(k + su)$$

$$= \frac{1}{(2m-1)!} \int_0^1 ds (1 - s)^{2m-1} \int_{-1}^{1} dv v^{2m} \sum_{j=1}^{m} \gamma_{m,j} j^{2m+1} f(2m)(k + jsv/2).$$

Clearly, by (3.6),

$$\sum_{k=0}^{n-1} R_{m,k} = R_m.$$ (9.3)

Next, take any $\alpha = 0, \ldots, m - 1$. Then, by (3.6),

$$-\binom{2m}{m} \sum_{j=1}^{m} \gamma_{m,j} j^{2\alpha+1} = 2 \sum_{j=1}^{m} (-1)^j \binom{2m}{m+j} j^{2\alpha} = 2 \sum_{j=1}^{m} (-1)^j \binom{2m}{m+j} j^{2\alpha}$$

$$= \sum_{j=1}^{m} (-1)^j \binom{2m}{m+j} j^{2\alpha} - \binom{2m}{m} I\{\alpha = 0\}. (9.4)$$

Here and elsewhere, $I\{\cdot\}$ denotes the indicator function. The power function $\psi_\alpha$ defined by the formula $\psi_\alpha(z) := z^{2\alpha}$ for real $z$ is obviously a polynomial of degree $2\alpha < 2m$. Hence, $\Delta^{2m} \psi_\alpha = \psi^{(2m)}_\alpha = 0$, where
(for any natural $p$) $\Delta^p$ is the $p$th power of the symmetric difference operator $\Delta$ defined by the formula

$$(\Delta^p \phi)(z) := \phi(z + 1/2) - \phi(z - 1/2)$$

for all functions $\phi : \mathbb{R} \to \mathbb{R}$ and all real $z$, so that

$$(\Delta^p \phi)(z) = \sum_{\beta=0}^{n} (-1)^{\beta} \binom{p}{\beta} \phi(z + p/2 - \beta).$$

(9.5)

Therefore,

$$(-1)^m \sum_{j=-m}^{m} (-1)^j \left( \frac{2m}{m+j} \right) j^{2\alpha} = \sum_{\beta=0}^{2m} (-1)^{\beta} \binom{2m}{\beta} (m - \beta)^{2\alpha} = (\Delta^{2m} \psi\alpha)(0) = 0.$$ 

It follows from (9.4) that

$$\sum_{j=1}^{m} \gamma_{m,j} f^j I\{ q = 1 \} \quad \text{for } q = 1, 3, \ldots, 2m - 1,$$

(9.6)

which in particular confirms the equality of the first two sums in (3.9), involving the $\gamma_{m,j}$'s, to 1. The second equality in (3.9) now follows from, say, yet to be proved equality (3.3) by taking there any natural $n \geq m$ and letting $f = 1_{[m/2 - 1, m/2]}$; then each of the integrals in (3.2)–(3.3) equals $n - m - 1 \neq 0$.

In view of (9.2), it also follows from (9.6) that

$$S_{m,k} = f(k).$$

(9.7)

Take any $j = 1, \ldots, m$ and let $G(y) := G_j(y) := F(y - j/2)$ for all real $y$ – where, recall, $F$ is any antiderivative of the function $f$. Then

$$\sum_{k=0}^{n-1} \int_{k-j/2}^{k+j/2} = \sum_{k=0}^{n-1} \left[ F(k + j/2) - F(k - j/2) \right]$$

$$= \sum_{k=0}^{n-1} G(k + j) - \sum_{k=0}^{n-1} G(k)$$

$$= \sum_{k=j}^{n-1+j} G(k) - \sum_{k=0}^{n-1} G(k)$$

$$= \sum_{k=n}^{n-1+j} G(k) - \sum_{k=0}^{j-1} G(i)$$

$$= \sum_{i=0}^{j-1} G(n - 1 + j - i) - \sum_{i=0}^{j-1} G(i)$$

since $\{i - j/2; i = 0, \ldots, j - 1\} = \{-1 + j/2 - i; i = 0, \ldots, j - 1\}$. So,

$$\sum_{k=0}^{n-1} A_{m,k} = \sum_{k=0}^{n-1} \sum_{j=1}^{m} \gamma_{m,j} \int_{k-j/2}^{k+j/2} = \sum_{j=1}^{m} \gamma_{m,j} \sum_{k=0}^{n-1} \int_{k-j/2}^{k+j/2} = \sum_{j=1}^{m} \gamma_{m,j} \sum_{i=0}^{j-1} \int_{i-j/2}^{n-1+j/2-i} = A_m$$

(9.8)

by the definition of $A_m$ in (3.2), and the second equality in (3.2) also follows. Now (3.1) follows from (9.1), (9.3), (9.7), and (9.8).

To show that the first expression in (3.3) equals that in (3.2), note that the conjunction of the conditions $j \in \{1, \ldots, m\}$ and $i \in \{0, \ldots, j - 1\}$ is equivalent to the conjunction of the conditions $\alpha \in \{1 - m, \ldots, m -
Lemma 9.1. The key to this proof is $\zeta$-functions. Inequality (4.5) is obtained similarly, using (2.4) and the inequality integral formula.

Proof of Proposition 4.2. Let $f(x) = x^p$. Then, in view of the condition $p = 0, \ldots, 2m - 1$, we have $f^{(2m)} = 0$ and hence $R_m = 0$. So, by (3.1), with $C := -\frac{1}{p+1} \sum_{j=1}^{m} \gamma_{m,j} \left( i - j \right)^{p+1}$,

$$\sum_{k=0}^{n-1} k^p = C + \frac{1}{p+1} \sum_{j=1}^{m} \gamma_{m,j} \sum_{i=0}^{j-1} \left( n - 1 + \frac{j}{2} - i \right)^{p+1} \quad \alpha$$

is a polynomial in $n$. Comparing the coefficient of $n^{p+1-\alpha}$ for $\alpha = p$ in this polynomial with the corresponding coefficient in the Faulhaber formula (2.5), we obtain the first equality in (3.19). The second equality there is obtained quite similarly to the equalities in (3.3) and (3.4).

(A different, longer proof of Proposition 3.6, which does not use (3.1), was given by Amdeberhan [18].)

Proof of Proposition 4.2. Take any nonnegative integer $\alpha$ and any real $x \geq -m/2 - 1/2$. Then the condition $a \geq (m+3)/2$ yields $x+a \geq 1$. Let $C_{a}(r_{a})$ denote the circle $\{ z \in \mathbb{C}: |z-x| = r_{a} \}$, where $r_{a} := (x+a) \sin \theta_{a}$. Then it is easy to see that $C_{a}(r_{a})$ is contained in the convex set $S$. So, by the Cauchy integral formula,

$$|f^{(\alpha)}(x)| = \frac{\alpha!}{2\pi i} \int_{C_{a}(r_{a})} \frac{dz}{(z-x)^{\alpha+1}} \leq \frac{\alpha!}{2\pi r_{a}^{\alpha}} \int_{0}^{2\pi} dr e^{-i\alpha} f(x+r_{a}e^{i\theta}) \quad |f^{(\alpha)}(x)| \leq \mu |x+r_{a}e^{i\theta}|^{\alpha} \leq \mu (x+a)^{\alpha} (2 + \sin \theta_{a})^{\alpha}.$$

Thus, for all real $x \geq -m/2 - 1/2$

$$|f^{(\alpha)}(x)| \leq \mu \alpha! \frac{(2 + \sin \theta_{a})^{\alpha}}{(x+a)^{\alpha-\lambda}} \quad g_a(x). \quad (9.9)$$

To complete the proof of (4.3), it remains to refer to (3.11) and (4.1), and to do straightforward calculations. Inequality (4.5) is obtained similarly, using (2.4) and the inequality $\zeta(2m-1) < 1.01$ for $m \geq 4$ instead of (3.11) and (4.1). Proposition 4.2 is now proved.

Proof of Proposition 4.3. The key to this proof is

Lemma 9.1. For $j = 1, \ldots, m - 1$

$$|\gamma_{m,j}| \leq \rho_{m,j} := 2m^{2m+1} (m-j)^{j-m-1/2} (m+j)^{m-j-1/2}.$$
The bound \( \hat{\rho}_{m,j} \) on \( |\gamma_{m,j}|/j \) was obtained by using the definition (3.6) of \( \gamma_{m,j} \) together with the Stirling approximation formula. Therefore, this bound is asymptotically tight when \( m - j \) is large.

On the other hand, the values of \( |\gamma_{m,j}|/j \) are comparatively small when \( m \) is large but \( m - j \) is not, and so, the contribution of these terms into the sum \( \sum_{j=1}^{m} |\gamma_{m,j}|/j^{2m+1} \) in (4.7) is relatively small.

Proof of Lemma 9.1 Consider the ratio
\[
\frac{r_{m,j} := \hat{\rho}_{m,j}}{|\gamma_{m,j}|/j} \tag{9.12}
\]
and then the ratio
\[
q_{m,j} := \frac{r_{m,j+1}}{r_{m,j}} = \left( \frac{m - j}{m - 1 - j} \right)^{m - j - 1/2} \left( \frac{m + j}{m + j + 1} \right)^{m + j + 1/2}.
\]
for \( j = 1, \ldots, m - 2 \). Let us take here the liberty to deal with \( q_{m,j} \) as a function of real arguments \( m \in (1, \infty) \) and \( j \in [0, m - 1) \), with the value of the function at the point \( (m, j) \) defined as the value of the latter displayed expression. Then we have
\[
\frac{\partial^2}{\partial j^2} \ln q_{m,j} = \frac{2m(2j + 1)(m^2 + j^2 + j)}{(m - 1 - j)^2(m - j)^2(m + j + 1)^2} > 0,
\]
so that \( \ln q_{m,j} \) is strictly convex in \( j \).

Next, \( \frac{\partial^2}{\partial m^2} \ln q_{m,0} = \frac{2}{m(m^2 - 1)^2} > 0 \), so that \( \ln q_{m,0} \) is strictly convex in \( m \). Moreover, \( \ln q_{m,0} \to 0 \) as \( m \to \infty \). So, \( \ln q_{m,0} > 0 \).

Further, \( \frac{d}{dm} \left[ \left( \frac{\partial}{\partial j} \ln q_{m,j} \right) \right]_{j=0} = -\frac{m^2 + 1}{m^2(m^2 - 1)^2} < 0 \), so that \( \left( \frac{\partial}{\partial j} \ln q_{m,j} \right) \) \( j=0 \) is decreasing in \( m \), with \( \left( \frac{\partial}{\partial j} \ln q_{m,j} \right) \) \( j=0 \to 0 \) as \( m \to \infty \). So, \( \left( \frac{\partial}{\partial j} \ln q_{m,j} \right) \) \( j=0 > 0 \).

Recalling now that \( \ln q_{m,0} > 0 \) and \( \ln q_{m,j} \) is convex in \( j \), we conclude that \( \ln q_{m,j} > 0 \) and \( q_{m,j} > 1 \).

So, in view of the definition of \( q_{m,j} \), we see that \( r_{m,j} \) is increasing in \( j = 1, \ldots, m - 1 \).

We also have \( \frac{d^2}{d m^2} \ln r_{m,1} = \frac{2}{m(m^2 - 1)^2} > 0 \), so that \( \ln r_{m,1} \) is strictly convex in \( m \). Moreover, \( \ln r_{m,1} \to 0 \) as \( m \to \infty \). So, \( \ln r_{m,1} > 0 \) and \( r_{m,1} > 1 \). Since \( r_{m,j} \) is increasing in \( j = 1, \ldots, m - 1 \), we now have \( r_{m,j} > 1 \) for all \( j = 1, \ldots, m - 1 \). Thus, in view of the definition of \( r_{m,j} \), the proof of Lemma 9.1 is complete.

Let us now turn back to the the proof of Proposition 4.4. The last sentence of Proposition 4.4 is trivial. So, assume that \( m \geq 2 \). By Lemma 9.1 and (4.8),
\[
|\gamma_{m,j}|/j^{2m+1} < \hat{\rho}_{m,j} j^{2m} = 2m^{2m} \Lambda(j/m)m(1 - j^2/m^2)^{-1/2} \leq 2m^{2m} \Lambda_m(1 - j^2/m^2)^{-1/2}
\]
for \( j = 1, \ldots, m - 1 \), whence
\[
\sum_{j=1}^{m} |\gamma_{m,j}|/j^{2m+1} < 2m^{2m} \Lambda_m \sum_{j=1}^{m-1} \frac{1}{\sqrt{1 - j^2/m^2}} + |\gamma_{m,m}|m^{2m+1}. \tag{9.10}
\]
Since \( \frac{1}{\sqrt{1 - x^2/m^2}} \) is convex in \( x \in [0, 1] \), we have
\[
\sum_{j=1}^{m-1} \frac{1}{\sqrt{1 - j^2/m^2}} \leq \sum_{j=1}^{m-1} \int_{j-1/2}^{j+1/2} \frac{1}{\sqrt{1 - x^2/m^2}} \, dx < \int_{0}^{m} \frac{1}{\sqrt{1 - x^2/m^2}} \, dx = \pi \frac{2}{m}. \tag{9.11}
\]
Next, let us show that for natural \( m \)
\[
\kappa_m := \left( \frac{2m}{m} \right) / \left( 2m^2 e^{-1/(8m)} / \sqrt{\pi m} \right) > 1. \tag{9.12}
\]
Then \( \frac{d}{dm} \ln K_m = \frac{1}{(8m^2(m+1)^2(2m+1))} > 0 \), so that \( K_m \) is increasing in \( m \), with \( K_m \to 1 \) as \( m \to \infty \). Hence, \( K_m < 1 \) for all \( m \). That is, \( \kappa_m \) is decreasing in \( m \), with \( \kappa_m \to 1 \) as \( m \to \infty \). Thus, the inequality in (9.12) is checked. It now follows from (9.10), (9.11), and (3.6) that

\[
\sum_{j=1}^{m} |\gamma_{m,j}| j^{2m+1} < 2m^{2m+1} \left( \frac{\pi}{2} \Lambda_m + 2^{-2m} e^{1/(8m)} \sqrt{\pi/m} \right) = 2m^{2m+1} \Lambda_m (1 + \varepsilon_m),
\]

where \( \varepsilon_m := 2(4\Lambda_m)^{-1/8m}/\sqrt{\pi m} \). Clearly, \( \varepsilon_m \) is decreasing in \( m \geq 1 \). Also, \( \varepsilon_{26} < 0.001 \). So, \( \varepsilon_m < 0.001 \) for \( m \geq 26 \). To complete the proof of Proposition 4.4 it remains to note that, by direct calculations, inequality (3.7) holds for all \( m = 2, \ldots, 25 \).

**Proof of Proposition 5.7**

Take any natural \( n \). Let

\[
R_{m,f}(n) := R_m,
\]

with \( R_m \) as defined in (3.8). Then, by (5.2),

\[
R_{m,f}(n) \to R_{m,f}(\infty)
\]

and, by (3.1)–(3.2),

\[
\sum_{k=0}^{n-1} f(k) = G_{m,F}(n) - G_{m,F}(0) - R_{m,f}(n).
\]

So,

\[
\sum_{k=0}^{n-1} f(k) - G_{m_0,F}(n) = [G_{m,F}(n) - G_{m,F}(0)] - G_{m,F}(0) - R_{m,f}(n).
\]

Next, by the linearity of \( G_{m,F} \) in \( F \),

\[
G_{m,F}(n) - G_{m_0,F}(n) = [G_{m,T}(n) - G_{m_0,T}(n)] + [G_{m,F-T}(n) - G_{m_0,F-T}(n)],
\]

where \( T = T_{n,m_0,F} \) is the Taylor polynomial of order \( p := 2m_0 - 1 \) for the function \( F \) at the point \( n - 1 \), so that

\[
T(x) = \sum_{\alpha=0}^{p} \frac{F^{(\alpha)}(n-1)}{\alpha!} (x - n + 1)^{\alpha}
\]

for real \( x \).

By Proposition 5.6, \( G_{m_0,F} = B_{\beta} = G_{m_0,F_0} \) for all \( \beta = 0, \ldots, p \), where \( P_{\beta}(x) := x^\beta \) for real \( x \). Since \( T \) is a polynomial of degree \( \leq p \), it is a linear combination of the polynomials \( P_0, \ldots, P_p \). Therefore and because \( G_{m,F} \) is linear in \( F \), one has \( G_{m,T}(0) = G_{m_0,T}(0) \). Since \( p = 2m_0 - 1 \) and \( m \geq m_0 \), it also follows that \( T^{(2m)} = 0 \) and hence, by (9.14), \( R_{m,T}(n) = 0 \). Thus, by (9.16),

\[
\sum_{k=0}^{n-1} T'(k) = G_{m,T}(n) - G_{m,T}(0) = G_{m_0,T}(n) - G_{m_0,T}(0).
\]

Therefore and because \( G_{m,T}(0) = G_{m_0,T}(0) \), we have

\[
G_{m,T}(n) = G_{m_0,T}(n), \quad \text{for all natural } n.
\]

Further, the remainder \( (F - T)(n - 1 + w) \) at point \( n - 1 + w \) of the Taylor approximation \( T \) of \( F \) at \( n \) equals

\[
F^{(p+1)}(n - 1 + \theta w)w^{p+1}/(p+1)! = f^{(2m_0-1)}(n - 1 + \theta w)w^{p+1}/(p+1)!
\]
for all real \( w \) and some \( \theta = \theta_{f,m,n,m_0} \in (0,1) \). So, by (5.1),
\[
(F - T)(n - 1 + w) \xrightarrow{n \to \infty} 0
\]
for each real \( w \), and so, by (5.4),
\[
G_{m,F,T}(n) \xrightarrow{n \to \infty} 0.
\]
(9.20)

It follows now from (9.18) and (9.19) that \( G_{m,F}(n) - G_{m_0,F}(n) \xrightarrow{n \to \infty} 0 \). To complete the proof of Proposition 5.3, it remains to refer to (9.17) and (9.15). As for the equalities (5.5) and (5.6), their proof is quite similar to that of (5.3) and (5.4). Proposition 5.1 is now completely proved. \( \square \)

**Proof of Proposition 5.3** Take any natural \( n \). Let
\[
R_{EM}(n) := R_{EM,n},
\]
with \( R_{EM} \) as defined in (2.3). Then, by (5.9),
\[
R_{EM,n}(n) \xrightarrow{n \to \infty} R_{EM}(\infty)
\]
and, by (2.1)–(2.2), \( \sum_{k=0}^{n-1} f(k) = f(0) + G_{EM}(n) - G_{m,F}(1) + R_{EM,n}(n) \). So,
\[
\sum_{k=0}^{n-1} f(k) - G_{m,F}(n) = f(0) + [G_{EM}(n) - G_{m,F}(n)] - G_{m,F}(1) + R_{EM,n}(n).
\]
But
\[
G_{m,F}(n) - G_{m,F}(n) = \sum_{j=m_0}^{m-1} B_{2j} \frac{(2j)!}{(2j)!} f(2j)(n-1) = \sum_{j=m_0}^{m-1} B_{2j} \frac{(2j)!}{(2j)!} f(2j-1)(n-1) \xrightarrow{n \to \infty} 0
\]
by (5.8). To complete the proof of Proposition 5.3 it now remains to refer to (9.23) and (9.22). \( \square \)

**Proof of Proposition 5.5** Let us first verify the last sentence of this proposition. Here it is enough to assume that \( P(x) = P^m(x) := (x - n)^\beta \) for an arbitrary \( \beta = 0, \ldots, 2m_0 - 1 \). Then, by (5.4) and Proposition 3.6
\[
G_{m,P}(n) = \sum_{j=1}^{m} \sum_{i=0}^{j-1} (-1 + j/2 - i)^\beta = B^\beta,
\]
whereas, in view of (5.11) and because \( B_0 = 1, B_1 = 1/2, \) and \( B_3 = B_5 = \cdots = 0, \)
\[
G_{m,P}(n) = (-1)^\beta + \frac{\beta}{2} \sum_{j=1}^{m} B_{2j} \frac{(2j)!}{(2j)!} (-1)^\beta = \sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha} (-1)^\beta
\]
\[
\sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha} (-1)^\beta = (B^\beta + \sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha}) (-1)^\beta.
\]
So, if \( \beta \neq 1 \), then
\[
(-1)^\beta [G_{m,P}(n) - G_{m,P}(n)] = 1 - (-1)^\beta |B^\beta| = \sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha} (-1)^\beta = \sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha} = \sum_{\alpha=0}^{\beta-1} B_{2\alpha} \frac{(\beta)}{\alpha} = |1 - (-1)^\beta| B^\beta,
\]
by a well-known identity for the Bernoulli numbers – see e.g. [14 formula (2), page 229]. If \( \beta \) is even, then \( 1 - (-1)^\beta |B^\beta| = 0 \). If \( \beta = 3, 5, \ldots \), then again \( 1 - (-1)^\beta |B^\beta| = 0 \). So, \( G_{m,P}(n) = G_{m,P}(n) \) for all \( \beta \in \{0, \ldots, 2m_0 - 1\} \setminus \{1\} \). Also, \( G_{m,P}(n) = -1/2 = B_1 = G_{m,P}(n) \), by (9.24). This completes the verification of the last sentence of Proposition 5.5.

Let now \( T_{n,m,F} \) be the Taylor polynomial of order \( p = 2m_0 - 1 \) for the function \( F \) at the point \( n - 1 \), as in the proof of Proposition 5.1. Then, by (9.20), \( G_{m,F,T}(n) \xrightarrow{n \to \infty} 0 \). Also, by (5.11),
\( G_{m,F,T}(n) = 0 \). On the other hand, \( T \) is a polynomial of degree \( \leq 2m_0 - 1 \), and so, by the already proved last sentence of Proposition 5.5, it remains to use the linearity of \( G_{m,F} \) and \( G_{m,F}^{\text{EM}} \) in \( F \), which yields

\[
G_{m_0,F}(n) - G_{m_0,F}^{\text{EM}}(n) = [G_{m_0,T}(n) - G_{m_0,F}(n)] + [G_{m_0,F-T}(n) - G_{m_0,F}(n)] = G_{m_0,F-T}(n) \xrightarrow{n\to\infty} 0.
\]

\[\square\]

**References**

[1] G. E. Andrews, R. Askey, and R. Roy. *Special functions*, volume 71 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1999.

[2] S. B. Aruoba and J. Fernández-Villaverde. A comparison of programming languages in economics. Working Paper 20263, National Bureau Of Economic Research, June 2014.

[3] D. H. Bailey and J. M. Borwein. Experimental mathematics: recent developments and future outlook. In *Mathematics unlimited—2000 and beyond*, pages 51–66. Springer, Berlin, 2001.

[4] F. Bornemann. The SIAM 100-digit challenge: a decade later. Inspirations, ramifications, and other eddies left in its wake. *Jahresber. Dtsch. Math.-Ver.*, 118(2):87–123, 2016.

[5] F. Bornemann, D. Laurie, S. Wagon, and J. Waldvogel. The SIAM 100-digit challenge: a decade later. Inspirations, ramifications, and other eddies left in its wake. *Mathematics unlimited—2000 and beyond*, pages 51–66. Springer, Berlin, 2001.

[6] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. [arXiv:1004.3412](https://arxiv.org/abs/1004.3412) see also [http://maths-people.anu.edu.au/~brent/pub/pub028.html](http://maths-people.anu.edu.au/~brent/pub/pub028.html)

[7] P. L. Butzer, P. J. S. G. Ferreira, G. Schmeisser, and R. L. Stens. The summation formulae of Euler-Maclaurin, Abel-Plana, Poisson, and their interconnections with the approximate sampling formula of signal analysis. *Results Math.*, 59(3-4):359–400, 2011.

[8] B. Candelisergher, H. G. Gadiyar, and R. Padma. Ramanujan summation and the exponential generating function \( \sum_{k=0}^{\infty} \frac{x^k}{k!} (-k) \). *Ramanujan J.*, 21(1):99–122, 2010.

[9] S. Fillebrown. Faster computation of Bernoulli numbers. *J. Algorithms*, 13(3):431–445, 1992.

[10] H. W. Gould. Explicit formulas for Bernoulli numbers. *Amer. Math. Monthly*, 79:44–51, 1972.

[11] G. H. Hardy and S. Ramanujan. Asymptotic formulation in combinatorial analysis [Proc. London Math. Soc. (2) 17 (1918), 75–115]. In *Collected papers of Srinivasa Ramanujan*, pages 276–309. AMS Chelsea Publ., Providence, RI, 2000.

[12] D. Harvey. A multithreaded algorithm for computing Bernoulli numbers. *Math. Comp.*, 79(272):2361–2370, 2010.

[13] G. Ierley. Private communication, 2017.

[14] K. Ireland and M. Rosen. *A classical introduction to modern number theory*, volume 84 of Graduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1990.

[15] H. Kleinert. *Path integrals in quantum mechanics, statistics, polymer physics, and financial markets*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, fifth edition, 2009.

[16] K. Knopp. *Theory and application of infinite series*. Blackie, London, 1951.

[17] D. E. Knuth. Euler’s constant to 1271 places. *Math. Comp.*, 16:275–281, 1962.

[18] MathOverflow. A representation of the Bernoulli numbers. [http://mathoverflow.net/questions/252404/a-representation-of-the-bernoulli-numbers#252410](http://mathoverflow.net/questions/252404/a-representation-of-the-bernoulli-numbers#252410) 2016.

[19] I. Pinelis. Approximating sums by integrals only: multiple sums and sums over lattice polytopes. [https://arxiv.org/abs/1705.09159](https://arxiv.org/abs/1705.09159) 2017.

[20] L. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stress in a masonry dam. *Phil. Trans. Roy. Soc. London, Series A*, 210:307–357, 1910.

[21] M. V. Sadovskii. Quantum field theory, volume 17 of De Gruyter Studies in Mathematical Physics. De Gruyter, Berlin, 2013.

[22] A. Sidi. Practical extrapolation methods, volume 10 of Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2003. Theory and applications.

[23] E. C. Titchmarsh. *The theory of functions*. Oxford University Press, Oxford, 1958. Reprint of the second (1939) edition.