INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Power Algorithms for Inverting Laplace Transforms

Efstathios Avdis, Ward Whitt,

To cite this article:
Efstathios Avdis, Ward Whitt, (2007) Power Algorithms for Inverting Laplace Transforms. INFORMS Journal on Computing 19(3):341-355. https://doi.org/10.1287/ijoc.1060.0217

INFORMS Journal on Computing
Vol. 19, No. 3, Summer 2007, pp. 341–355
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1903 | 0341

DOI 10.1287/ijoc.1060.0217

Power Algorithms for Inverting Laplace Transforms

Efstathios Avdis, Ward Whitt
Department of Industrial Engineering and Operations Research, Columbia University,
New York, New York 10027 {stathi.avdis@gmail.com, ww2040@columbia.edu}

This paper investigates ways to create algorithms to invert Laplace transforms numerically within a unified framework proposed by Abate and Whitt (2006). That framework approximates the desired function value by a finite linear combination of transform values, depending on parameters called weights and nodes, which are initially left unspecified. Alternative parameter sets, and thus algorithms, are generated and evaluated here by considering power test functions. Real weights for a real-variable power algorithm are found for specified real powers and positive real nodes by solving a system of linear equations involving a generalized Vandermonde matrix, using *Mathematica*. The resulting power algorithms are shown to be effective, with the parameter choice being tunable to the transform being inverted. The powers can be advantageously chosen from series expansions of the transform. Experiments show that the power algorithms are robust in the nodes; it suffices to use the first n positive integers. The power test functions also provide a useful way to evaluate the performance of other algorithms.

Key words: Laplace transforms; numerical transform inversion; power test functions; power algorithms; Fourier-series method; Talbot's method; Gaver-Stehfest algorithm; Zakian's algorithm; multiprecision computing; generalized Vandermonde matrix; *Mathematica* programming language
History: Accepted by Edward P. C. Kao, Area Editor for Computational Probability and Analysis; received June 2006; revised December 2006; accepted December 2006. Published online in *Articles in Advance* July 20, 2007.

1. Introduction

1.1. The Unified Framework

We propose a new class of algorithms for inverting Laplace transforms numerically, called *power algorithms*, with parameters that are tunable to the transform being inverted. Our power algorithms are constructed using *power test functions* within a *unified framework* for constructing algorithms to invert Laplace transforms numerically, proposed by Abate and Whitt (2006). Many pointers to the literature appear in Abate and Whitt (2006), including Zakian (1969, 1970, 1973) and Wellekens (1970), which provided a basis for the framework, even though they were only concerned with developing a single algorithm.

The goal of the inversion is to calculate values of a real-valued function f of a nonnegative real variable from its Laplace transform

$$\hat{f}(s) \equiv \mathscr{L}(f)(s) \equiv \int_0^\infty e^{-st} f(t)\,dt. \tag{1}$$

There are many applications of Laplace transforms and their numerical inversion in operations research, e.g., in queueing and financial engineering; see Abate et al. (1999), Petrella and Kou (2004), and references therein.

The classical *Bromwich inversion integral* expresses $f(t)$ exactly via the *contour integral*

$$f(t) = \frac{1}{2\pi i}\int_C \hat{f}(s)e^{st}\,ds, \quad t > 0, \tag{2}$$

where s is a complex variable and C is a contour extending from $c - i\infty$ to $c + i\infty$, falling to the right of all singularities of $\hat{f}$; see Theorem 24.4 of Doetsch (1974).

For numerical calculation, the unified framework approximates the function f by a finite linear combination of transform values; specifically,

$$f(t) \approx f_n(t) \equiv f_{n,\alpha,\omega}(t) \equiv \frac{1}{t}\sum_{k=1}^{n} \omega_k \hat{f}\left(\frac{\alpha_k}{t}\right), \quad t > 0, \tag{3}$$

where $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$ and $\omega \equiv (\omega_1, \ldots, \omega_n)$ are vectors of complex numbers, called *nodes* and *weights*, respectively. The nodes and weights do not necessarily depend on the transform $\hat{f}$ or the function argument t, but typically depend upon n. This is a framework rather than a single algorithm because the nodes and weights are initially left unspecified.

One way to gain insight into the framework (3) is to make the change of variables $z = st$, allowing us to rewrite the contour integral (2) as

$$f(t) = \frac{1}{2\pi i t}\int_{C'} \hat{f}(z/t)e^{z}\,dz, \quad t > 0, \tag{4}$$

where C' is the same contour as a function of z. From (4), we see that t^{-1} appears both in the multiplicative constant and the argument of the transform $\hat{f}$. From (4), we anticipate that appropriate numerical integration applied to the integral in (4) will produce the representation (3). Since the contour can be transformed without altering the integral (under regularity conditions), there should be freedom in choosing the nodes.

Abate and Whitt (2006) showed that three popular numerical inversion algorithms can be represented in this framework: (i) the Gaver (1966)–Stehfest (1970) algorithm, which we denote by $\mathcal{G}$, for which the weights and nodes (and thus the transform arguments) are real, (ii) the Fourier-series method with Euler summation, from Abate and Whitt (1992, 1995), which we denote by $\mathcal{E}$ because they refer to the algorithm as "Euler," and (iii) Talbot's (1979) algorithm, as modified by Abate and Valko (2004), which we denote by $\mathcal{T}$, which is based on advantageously deforming the contour in the Bromwich inversion integral.

Abate and Whitt (2006) showed that the three algorithms $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{T}$ can be combined in any combination to produce nine different two-dimensional algorithms, and examined the behavior of each. They showed that it can be advantageous to use different one-dimensional algorithms in the inner and outer loops.

A key structural property in the framework (3) is the *linearity*. Given that sequence acceleration is often applied in constructing inversion algorithms, see Valko and Abate (2004) and Wimp (1981), the linearity in (3) implies that a *linear acceleration method* is being used instead of a nonlinear one. We will strongly exploit the linearity in this paper. Despite the established power of nonlinear acceleration techniques, the linear methods seem to be remarkably effective in this inversion context. In support, Valko and Abate (2004) showed that the linear Salzer scheme exploited by Stehfest (1970) for accelerating convergence of the sequence of Gaver (1966) functions is remarkably effective compared to several nonlinear methods. Euler summation has proven to be very effective with the Fourier-series method as well; see O'Cinneide (1997). We will not directly exploit sequence acceleration here, though.

An attractive feature of the power algorithms developed in this paper, like the Gaver-Stehfest algorithm, is that complex variables need not be considered at all, because our proposed power algorithms are real-variable algorithms. Real-variable algorithms are sometimes preferred because mathematical software can have difficulties properly evaluating functions of one or more complex variables. We will be considering the case in which the function f is real-valued and the weights and nodes are real, so that (3) applies directly in the real domain. We briefly discuss extensions to complex variables in Section 9.

1.2. Creating New Algorithms

Abate and Whitt (2006) suggested that the unified framework could be used to create new one-dimensional inversion algorithms. They suggested choosing a family of test functions and performing optimization to select nodes and weights that minimize the error for those test functions. We start here to investigate that idea seriously. In doing so, we have two main goals: First, we aim to understand better the process of numerical inversion; and second, we aim to develop a method to construct algorithms efficiently in the framework with nodes and weights that are ideally suited for the specific transform to be inverted or the function arguments of interest. We focus on real power algorithms, so we are especially interested in making comparisons to the Gaver-Stehfest algorithm.

For the optimization, it is evident that there are many ways to proceed. For example, let F be a set of test functions and let T be a set of time points (arguments for f). For n given, we can minimize, over the vectors α and ω, a weighted sum of the rth powers of the errors for $f \in F$ and $t \in T$, defined by

$$e(\alpha, \omega; F, T) \equiv \sum_{f \in F} \sum_{t \in T} c(f, t) |f(t) - f_{n,\alpha,\omega}(t)|^r, \qquad (5)$$

where $c(f, t) > 0$ are weights to place more emphasis on certain functions and certain times; we emphasize the simple special case in which $r = c(f, t) = 1$.

In general, the optimization problem is quite complicated, because the nodes α_k in (3) appear inside the argument of the transform $\hat{f}$. For fixed nodes, optimization over the weights is much more tractable, because $f_{n,\alpha,\omega}$ in (3) is a *linear function of the weights*. Thus, we consider only systematic optimization over the weights, for specified nodes. We experimentally investigate the consequence of different node sets.

We consider one very natural family of test functions: *powers*. For real p, the pth power is the function $f(t) \equiv t^p$, which has well-defined Laplace transform $\hat{f}(s) \equiv \Gamma(p+1)/s^{p+1}$ for all $p > -1$, where $\Gamma(p)$ is the gamma function, for which $\Gamma(p+1) = p!$ when p is an integer. (Powers with $p \leq -1$ may also be of interest. They correspond to pseudotransforms, as discussed in Sections 12–14 and the appendix of Doetsch 1974, and they may capture asymptotic behavior for large t (and small s) via Heaviside's theorem, p. 254 of Doetsch 1974, but we emphasize $p > -1$ here.)

For any integer $n > 1$, any n positive real nodes, and any n powers, we are able to find n real weights that make the inversion *exact* for those powers for all $t > 0$; i.e., we are able to find a weight vector ω such that $f_{n,\alpha,\omega}(t) = f(t)$ for all $f \in F$ and all $t > 0$. Moreover, we are able to find these weights by solving a

system of linear equations, which is easily done to high precision with *Mathematica*. (A useful reference about transforms related to *Mathematica* is Graf 2004.) We call the new inversion algorithms created in this way *power algorithms*.

We show that the power algorithms are effective. They are appealing because they are real-variable algorithms, like Gaver-Stehfest, but conceptually simple. Both the derivation, via power test functions, and the creation, via the solution of a system of linear equations, are easy to understand. As with the Gaver-Stehfest algorithm, the biggest disadvantage is the high precision that is required, but that is routinely available with mathematical software such as *Mathematica*. As in Abate and Valko (2004), it is possible to apply multiprecision Laplace inversion, working with the precision needed to obtain specified accuracy in the calculated value (see Section 5).

1.3. Organization of the Paper

We start in Section 2 by establishing properties of the power test functions and developing the power inversion algorithms. In Section 3 we describe experiments conducted to understand how to choose the set of powers. In Section 4 we describe experiments conducted to understand how to choose the set of nodes. These experiments show that (i) the powers can be chosen advantageously depending on the function, and (ii) there is considerable freedom in the choice of the nodes among positive real nodes. In Section 5 we discuss accuracy and precision.

In Section 6 we discuss Zakian's algorithm, denoted by $\mathcal{Z}$, which can be regarded as a variant of the power algorithm, having only integer powers, but complex nodes and weights. Zakian's algorithm is designed to perform especially well for smooth (analytic, with derivatives of all orders) functions, but as a consequence it can perform poorly for nonsmooth functions, as we will show.

In Section 7 we indicate how the power test functions can be used to evaluate other algorithms. We "score" the Gaver-Stehfest, Euler, Talbot, and Zakian algorithms, obtaining revealing results consistent with experience. In Section 8 we discuss how to use multiple algorithms or multiple instances of one algorithm to estimate the inversion error while performing the inversion (of a transform of an unknown function). We mention some extensions in Section 9, including gamma test functions and complex variables. We draw conclusions in Section 10. A substantial amount of additional material is contained in the Online Supplement to this paper on the journal's website (and the authors' websites), which provides an important expansion of the story.

2. Creating Power Algorithms

2.1. Power Test Functions

It is natural to consider the first n nonnegative integer powers as test functions, because accuracy for them implies accuracy for polynomials of degree $n-1$ by linearity. By the Weierstrass approximation theorem, all continuous functions on the positive halfline $[0,\infty)$ can be approximated uniformly over bounded subintervals by polynomials. For continuous functions $f(t)$ that converge to 0 as $t \to \infty$, the uniformity can extend over the entire positive halfline. However, we do not limit attention to nonnegative integer powers. Through experiments, we found that it can be desirable to include positive fractional powers and negative powers in the interval $(-1,0)$. We discuss the choice of powers in Section 3.

The polynomial perspective also indicates that, in general, the specific method and the accuracy should depend on the function argument t, with the inversion difficulty increasing in t. That can also be seen from the damping by multiplying $f(t)$ by e^{-at} in the Fourier-series method; e.g., see Abate and Whitt (1995). Abate and Valko (2004) found that for some "good" transforms (their set **F**) the $\mathcal{G}$ and $\mathcal{T}$ inversion accuracy is largely independent of t, but for other transforms the accuracy decreases in t (or, equivalently, the computational complexity increases in t). Our results are consistent with that conclusion. As in Abate and Valko (2004), we achieve inversion accuracy independent of t for some functions, but inversion accuracy decreasing in t for other functions. Here we plot inversion numerical results for $0.5 \le t \le 10$. In the Online Supplement, we show results for a wide range of t – small ($0.5 \le t \le 10$), medium ($10 \le t \le 100$), and large ($100 \le t \le 1{,}000$). The difficulties with exceptionally large or small arguments can often be addressed by scaling; e.g., see Choudhury and Whitt (1997).

Since the pth power $f(t) \equiv t^p$ has Laplace transform $\Gamma(p+1)/s^{p+1}$ for $p > -1$, we see that the framework (3) is exact, using real weights and nodes, for the pth power for $p \in \mathcal{P} \equiv \{p \in \mathbb{R}\colon p > -1\}$ if

$$\frac{1}{t}\sum_{k=1}^{n}\frac{\Gamma(p+1)}{(\alpha_k/t)^{p+1}}\omega_k = t^p, \tag{6}$$

which, by eliminating $t > 0$, is equivalent to the *power relation*

$$\sum_{k=1}^{n}\frac{\Gamma(p+1)}{\alpha_k^{p+1}}\omega_k = 1. \tag{7}$$

Of course, we cannot expect to achieve equality in the power relation (7) when we are considering a large number of powers. In general, we can select n real weights and n positive real nodes by minimizing the error in the power relations, over α and ω, for m

powers p_j with $-1 < p_1 < \cdots < p_m$, by considering the following *mathematical program*:

$$\begin{aligned} \min_{\omega,\alpha,\varepsilon} \quad & \sum_{j=1}^{m} c_j \varepsilon_j \\ \text{s.t.} \quad & 1-\varepsilon_j \le \sum_{k=1}^{n} \frac{\Gamma(p_j+1)}{\alpha_k^{p_j+1}} \omega_k \le 1+\varepsilon_j, \quad 1 \le j \le m, \\ & \varepsilon_j \ge 0, \quad 1 \le j \le m, \\ & \alpha_1 \ge \delta, \quad \alpha_k - \alpha_{k-1} \ge \delta, \quad 2 \le k \le n, \end{aligned} \tag{8}$$

where the parameter δ is a small positive quantity to maintain minimum separation between successive nodes. The positive numbers c_j weigh the violation of the jth *power constraint*, $1 \le j \le m$.

The mathematical program is a complicated *nonconvex nonlinear program*, because of the node variables α_k appearing in the denominator of the power constraints in (8). With an additonal assumption, this particular nonlinear program can be regarded as a *signomial program*; see Section 3 of Ecker (1980). To obtain a signomial program, we make the additional assumption that the weights alternate in sign (which experience indicates is appropriate). Then the sum in the power relation (7) becomes the difference of two posynomials (positive sums of powers of ratios). That is a way to attack the problem, but it is not elementary because of the nonconvexity.

We obtain great simplification if we fix the nodes. Then the mathematical program (8) becomes a *linear program* (LP). That suggests an iterative algorithm, searching over the node vectors, using an LP for each. (That is one approach for signomials.) For fixed nodes, the resulting LP is not large by LP standards, but nevertheless it is challenging because we need to work with high precision; standard double precision will not suffice. However, we do not consider the mathematical programs further here. Instead we make a further simplification.

2.2. A System of Linear Equations for Given Nodes

In addition to fixing the nodes, we go further by considering n given (distinct) positive real nodes and n distinct powers when we look for the n weights. Then the power relations in (7) for the n values of p become a system of n linear equations in n unknowns, where the weights are the unknowns. We discuss how to select the powers p and the nodes α_k in Sections 3 and 4, respectively.

We can write the linear system in matrix form as

$$A\omega = b \equiv \left[\frac{1}{\Gamma(p_1+1)}, \ldots, \frac{1}{\Gamma(p_n+1)} \right]^T, \tag{9}$$

where T denotes transpose and $A \equiv A_{n,\mathscr{P}}$ is the *node matrix*

$$A \equiv A_{n,\mathscr{P}} \equiv \begin{bmatrix} \left(\frac{1}{\alpha_1}\right)^{p_1+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_1+1} \\ \left(\frac{1}{\alpha_1}\right)^{p_2+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_2+1} \\ \vdots & & \vdots \\ \left(\frac{1}{\alpha_1}\right)^{p_n+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_n+1} \end{bmatrix}. \tag{10}$$

Fortunately, it is not difficult to solve the linear system $A\omega = b$ in (9) and (10) with *Mathematica* because the node matrix A in (10) is a *generalized Vandermonde matrix* with positive real "points" $1/\alpha_k$ and "exponents" p_k+1, $k = 1, \ldots, n$, with $p_k > -1$. Without loss of generality, we assume that

$$0 < 1/\alpha_1 < \cdots < 1/\alpha_n \quad \text{and} \quad 0 < p_1+1 < \cdots < p_n+1.$$

The generalized Vandermonde matrix is nonsingular, even totally positive; see p. 99 of Gantmacher (1959) or p. 76 of Gantmacher and Krein (2002). Hence, the linear system of equations in (9) always has a unique solution.

Even though the generalized Vandermonde matrices are nonsingular, they are notoriously *ill-conditioned* (have high matrix condition number). That tends to make the linear system (9) unsolvable in practice with standard double precision. We circumvent that difficulty by using *Mathematica*, which supports high precision. We discuss the required precision for specified accuracy in the calculation in Section 5.

It is also significant that new efficient methods for solving generalized Vandermonde linear systems have been developed; see Demmel and Koev (2005). They achieve greater accuracy with less precision by avoiding subtractive cancellation. In Section 8, they show, by comparing to a *Mathematica* 100-digit-precision calculation, that they are able to solve generalized Vandermonde linear systems accurately with standard double precision. Thus there is the potential to achieve much greater efficiency with our power algorithms. We do not consider that approach here, leaving it as an interesting direction for future research. We emphasize simplicity by showing that our power algorithms are effective by a straightforward application of *Mathematica*.

We summarize our algorithm to construct power algorithms in Figure 1. In the next two sections we examine how to choose the set of powers $\mathscr{P}$ and the set of nodes $\mathscr{N}$. We will justify the default choices $\mathscr{P}^*$ and $\mathscr{N}^*$. In Section 5 we discuss accuracy and precision. From n terms in the sum (3), we expect to get

Constructing a Real-Variable Power Inversion Algorithm

1. Let the number of terms be n; e.g., $n = 30$.
2. Let the computer precision be $1.5n$; e.g., 45 significant digits.
3. Choose n distinct real numbers $p > -1$ to form the power set $\mathcal{P}$; e.g., with an integer n divisible by 5, let
$$\mathcal{P} = \mathcal{P}^* \equiv \mathcal{P}\left(-\frac{5j}{n}; \frac{k}{2}\right) \equiv \left\{-\frac{5j}{n}: 1 \le j \le \frac{n}{5} - 1\right\} \cup \left\{\frac{k}{2}: 0 \le k \le \frac{4n}{5}\right\}.$$
4. Choose n distinct positive real numbers to serve as the node set $\mathcal{N}$; e.g.,
$$\mathcal{N} \equiv \{\alpha_k: 1 \le k \le n\} = \mathcal{N}^* \equiv \{1, \ldots, n\}.$$
5. Using the specified computer precision, solve the system of n linear equations
$$\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}} \omega_k = 1, \quad p \in \mathcal{P},$$
to obtain the n real weights $\omega_1, \ldots, \omega_n$.
6. **Apply the created power inversion algorithm.** Using the computer precision, nodes and weights specified above, calculate the required transform values $\hat{f}(\alpha_k/t)$ and the weighted sum to obtain the real-variable numerical inversion
$$f_n(t) \equiv f_{n,\alpha,\omega}(t) = \frac{1}{t} \sum_{k=1}^{n} \omega_k \hat{f}\left(\frac{\alpha_k}{t}\right).$$

Figure 1 Algorithm Summary

from $n/6$ to $n/2$ significant digits in our computation of $f(t)$, with higher accuracy for smaller values of t.

The approach we have taken to construct power inversion algorithms within the unified framework via the linear system with the Vandermonde matrix relates to classic methods; e.g., see Chap. 19 of Bellman (1970), especially Sections 7–11. To a large extent, these old methods are made effective today by the evolution of computer technology and the accompanying mathematical software, allowing us to compute rapidly with high precision. Such technology changes invite reconsidering old ideas.

3. Choosing the Powers

A natural initial candidate for the set of n powers is the first n nonnegative integers, because that yields an exact inversion for all polynomials of degree $n-1$. That is also appropriate for smooth functions (with continuous derivatives of high orders) because matching $(n-1)$-degree polynomials matches the first n coefficients (derivatives at 0) in the Maclaurin series expansion (Taylor series expansion about 0) for the function f: With $f^{(n)}(x)$ being the nth derivative of f evaluated at x,

$$f(t) = f(0) + f^{(1)}(0)t + f^{(2)}(0)\frac{t^2}{2!} + \cdots + f^{(n-1)}(0)\frac{t^{n-1}}{(n-1)!} + O(t^n) \quad \text{as } t \to 0. \tag{11}$$

However, there are situations in which we might want to do something different. The first involves a smooth function that has some of its derivatives equal to 0. For example, an odd function like $\sin(t)$ has an expansion in odd powers,

$$\sin(t) = \sum_{j=0}^{\infty} \frac{(-1)^j}{(2j+1)!} t^{2j+1}, \tag{12}$$

while an even function like $\cos(t)$ has an expansion in even powers. Thus, to approximate $\sin(t)$, it would be better to fit a degree-$(2n-1)$ polynomial with only odd powers than to fit a full degree-$(n-1)$ polynomial (both using n terms).

The second involves functions that are not smooth. In particular, we often encounter functions that have series expansions in fractional powers of t, and thus fail to have derivatives of all orders at the origin. A simple example is $e^{-\sqrt{t}}$. Since

$$e^{-t} = \sum_{j=0}^{\infty} \frac{(-1)^j}{j!} t^j \tag{13}$$

the function $e^{-\sqrt{t}}$ has an expansion in half powers. Similarly, by (12), $\sin(t^{1/m})$ for positive integer m has an expansion in powers of the form $(2k-1)/m$ for $k \ge 1$.

Of course, here we are considering numerical transform inversion, which is usually considered only when we know the transform $\hat{f}$ but not the function f. Thus we want to start with the transform $\hat{f}$. Fortunately, starting from the transform $\hat{f}$, we are often able to establish a series expansion for it, and then obtain a corresponding series expansion of the function f by doing a term-by-term inversion. Indeed, such concepts are a fundamental part of Laplace transform theory; see Sections 30–37 of Doetsch (1974).

We suggest exploiting this basic theory for constructing the power set. Starting from the transform $\hat{f}$, we suggest identifying the series representation (expressed as a function of a real variable s)

$$\hat{f}(s) = \sum_{k=1}^{\infty} \frac{a_k}{s^{p_k+1}}, \quad -1 < p_1 < p_2 < \cdots \tag{14}$$

or an initial portion for all suitably large real s, thinking of s as large. In establishing (14) we aim only to identify the powers p_k; we do not use the coefficients a_k. Laplace transform theory tells us that, under regularity conditions, we will have an associated series expansion for f, namely,

$$f(t) = \sum_{k=1}^{\infty} \frac{a_k t^{p_k}}{\Gamma(p_k+1)} \tag{15}$$

for the same powers, thinking of t as small. Theorem 30.2 of Doetsch (1974) provides theoretical support.

Thinking of t as small, we anticipate that the initial powers in (14) and (15) will be most important to include in our power set.

Such analysis is often not difficult to perform directly, as we will illustrate. To cover a broad range of cases approximately, without any analysis, we propose a standard default power set of size n. Assuming n to be an integer multiple of 5, we use $(n/5)-1$ negative powers, evenly distributed in the interval $(-1, 0)$ and $(4n/5)+1$ half powers, extending from 0 up to $2n/5$. We call that the *default power set* and denote it by

$$\mathcal{P}^* \equiv \mathcal{P}\left(-\frac{5j}{n}; \frac{k}{2}\right) \equiv \left\{-\frac{5j}{n}: 1 \le j \le \frac{n}{5}-1\right\} \cup \left\{\frac{k}{2}: 0 \le k \le \frac{4n}{5}\right\}, \quad (16)$$

as in Step 3 in Figure 1.

We will illustrate by describing results from numerical inversion experiments. We compare the performance of these power algorithms to the Gaver-Stehfest algorithm. We chose Gaver-Stehfest because it also uses real nodes and weights, and because it has been well studied and is known to work very well in practice, e.g., see Abate and Valko (2004), Valko and Abate (2004), and Abate and Whitt (2006). We let $n = 30$. For the power algorithms, we use the default node set $\mathcal{N}^* = \{1, 2, \ldots, 30\}$. In Gaver-Stehfest, the nodes are also evenly spaced, at $k\ln(2)$, $1 \le k \le n = 30$. (The results are essentially the same when we use the Gaver-Stehfest node set instead of $\mathcal{N}^*$.)

A fundamental property of the Gaver-Stehfest algorithm is that its weights alternate in sign; see (32) of Abate and Whitt (2006). Figure 2 shows that the default $(\mathcal{P}^*, \mathcal{N}^*)$ power algorithm closely mimics this pattern, but does not follow it exactly. Figure 2 depicts the base-10 logarithm of the absolute value of the weights ω_k from $\mathcal{G}$ and $(\mathcal{P}^*, \mathcal{N}^*)$ for $n = 20$; plots for other n appear in the Online Supplement. The $\mathcal{G}$ weights appear as circles, filled in if $\omega_k > 0$ and empty if $\omega_k < 0$, whereas the $(\mathcal{P}^*, \mathcal{N}^*)$ weights appear as squares, filled in if $\omega_k > 0$ and empty if $\omega_k < 0$.

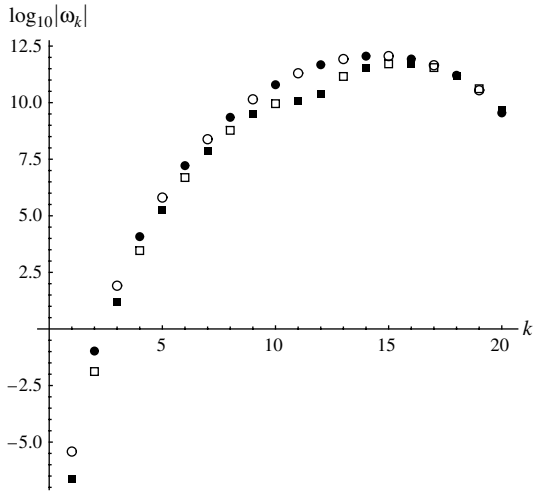

Figure 2 **The Successive Weights of $\mathcal{G}$ (Circles) and $\mathcal{P}^*$ (Squares) for $n = 20$ in a Logarithmic Scale. Absolute Values Are Shown, with Positive Weights Filled in (Dark) and Negative Weights Empty**

In our experiments, we consider the following 6 alternatives for the power set, each containing $n = 30$ powers:

1. Nonnegative integers: $\mathcal{P}(k) \equiv \{k: 0 \le k \le n-1\}$.
2. Nonnegative even integers: $\mathcal{P}(2k) \equiv \{2k: 0 \le k \le n-1\}$.
3. Nonnegative odd integers: $\mathcal{P}(2k+1) \equiv \{2k+1: 0 \le k \le n-1\}$.
4. Nonnegative integer multiples of 1/2: $\mathcal{P}(k/2) \equiv \{k/2: 0 \le k \le n-1\}$.
5. All integer multiples of 1/2: $\mathcal{P}((k-1)/2) \equiv \{k/2: -1 \le k \le n-2\}$.
6. The default power set with $(n/5)-1$ negative fractional powers and $(4n/5)+1$ nonnegative integer multiples of 1/2, i.e., $\mathcal{P}^*$ in (16).

Figure 3 shows inversion results for six different transforms for some of these power sets. In each case we compare to the Gaver-Stehfest algorithm. The six transforms considered in Figure 3 are chosen to contain cases in which each of the first five power sets performs best. A list of the transforms we have used appears in the Online Supplement; the indices correspond to their listing there. The functions f_1, f_3, f_4, and f_5 are the functions with those same indices from Table 1 of Valko and Abate (2004). (The third function is $f_3(t) \equiv e^{-t}I_0(t)$, where $I_0(t)$ is the modified Bessel function, as in Section 9.6 of Abramowitz and Stegun 1972.) The function f_9 is the exponential function e^{-t}; the other functions are $f_{12} \equiv \cos(t)$ and $f_{13} \equiv t^{-1/2} + f_4$.

Figure 3 shows the base-10 logarithm of the absolute error $|f(t) - f_n(t)|$ between the function f and the numerical inversion f_n as a function of t, $0.5 \le t \le 10.0$, for the six different transforms $\hat{f}$ with known inverses f. For clarity, we do not show the performance of all power sets in each plot.

Since the exponential function f_9 has a series expansion in terms of nonnegative integer powers with all coefficients nonzero, as shown in (13), we should expect the integer power set $\mathcal{P}(k)$ to perform best for $f_9(t) \equiv e^{-t}$, and it does. That can be determined directly from the series expansion of its transform

$$\hat{f}_9(s) \equiv \frac{1}{1+s} = \sum_{j=0}^{\infty} \frac{(-1)^j}{s^j}, \quad (17)$$

so that we could apply (14), (15), and (17) to deduce (13).

For f_9, the half-power set $\mathcal{P}(k/2)$ performs next best; it does not omit any initial integer powers

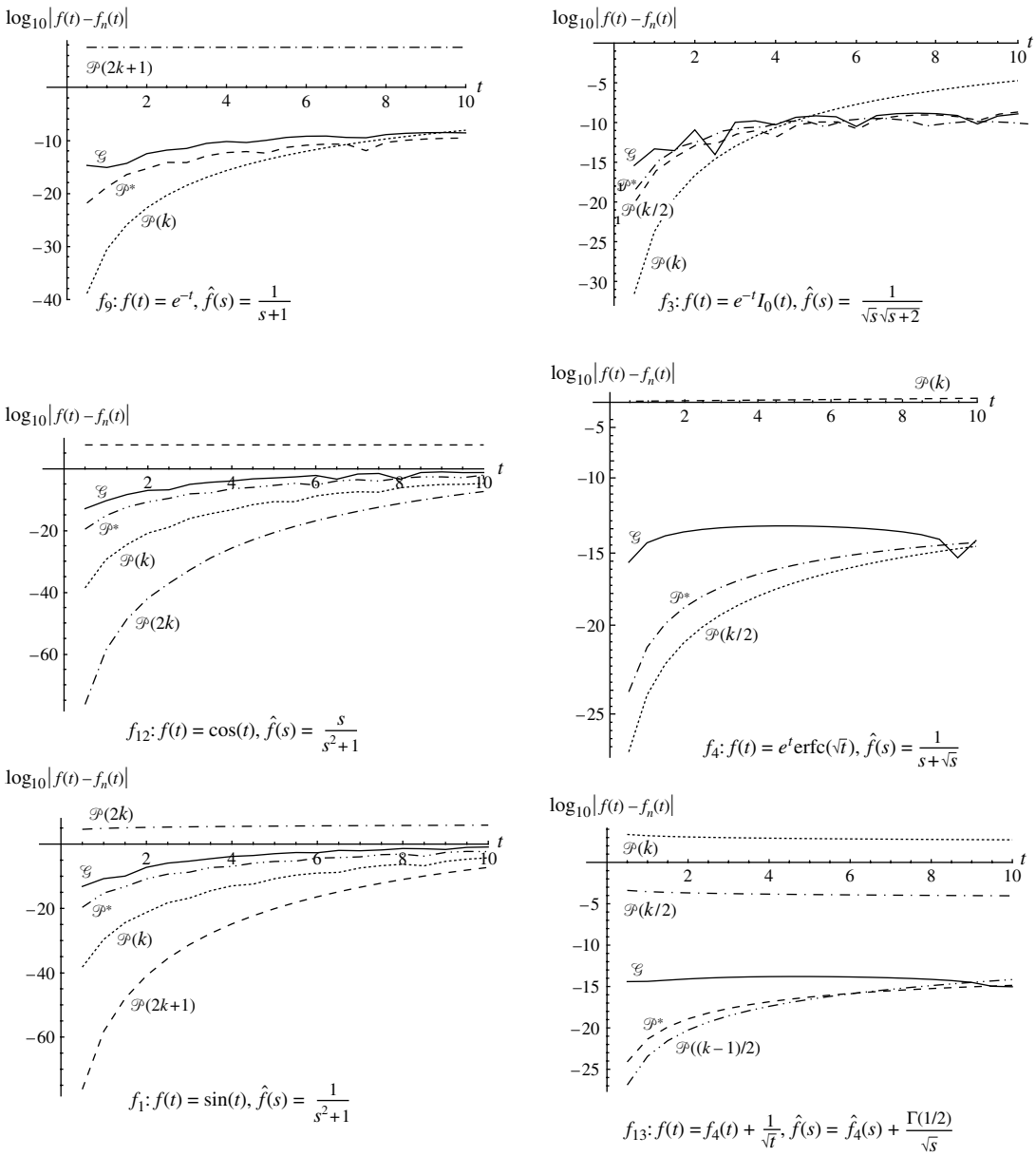

Figure 3 **Comparison of Inversion Performance for Different $\mathcal{P}$ with the Gaver-Stehfest Algorithm. The Base-10 Logarithm of the Absolute Error Is Plotted as a Function of t for $0.5 \le t \le 10.0$**

but the noninteger half powers are largely wasted, because they appear at the sacrifice of higher integer powers. Similarly, the default power set $\mathcal{P}^*$ performs reasonably well, but the negative powers and noninteger powers are largely wasted because they appear at the sacrifice of higher integer powers. In contrast, the even and odd power sets $\mathcal{P}(2k)$ and $\mathcal{P}(2k+1)$ perform quite poorly, because they are missing important low-order integer powers.

Indeed, the performance of the different power sets for all the examples can be explained, to a large extent, by the series expansions, which hold for both the function f and the transform $\hat{f}$, in the relation described in (14) and (15) above. First, the odd power set $\mathcal{P}(2k+1)$ performs best for $f_1(t) \equiv \sin(t)$, as expected from (12). We can deduce (12) from the relation between the series expansions in (14) and (15), starting from the series expansion of the transform $\hat{f}_1$, namely

$$\hat{f}_1(s) \equiv \frac{1}{1+s^2} = \sum_{j=0}^{\infty} \frac{(-1)^j}{s^{2j}}. \tag{18}$$

Similarly, the performance for $f_{12}(t) \equiv \cos(t)$ can be explained by the series expansion

$$\hat{f}_{12}(s) \equiv \frac{s}{1+s^2} = \sum_{j=0}^{\infty} \frac{(-1)^j}{s^{2j+1}}. \tag{19}$$

We deduce that the even power set $\mathcal{P}(2k)$ performs best.

Now consider

$$\hat{f}_3(s) \equiv \frac{1}{\sqrt{s(s+2)}}. \tag{20}$$

We first ignore the square root in the transform $\hat{f}_3(s)$, to obtain

$$\hat{f}_3(s)^2 \equiv \frac{1}{s(s+2)} = \frac{(1/2s)}{1+(s/2)} = (1/2s)\sum_{j=0}^{\infty}\frac{(-2)^j}{s^j} = (1/2)\sum_{j=0}^{\infty}\frac{(-2)^j}{s^{j-1}}. \tag{21}$$

Next, by exploiting fundamental operations on series, as in display 3.6.18 on p. 15 of Abramowitz and Stegun (1972), we deduce that the form (the powers, but not the coefficients) is preserved by taking the square root of the function. Hence we deduce that $\hat{f}_3(s)$ has an asymptotic expansion of the same form, as a function of s, as does $\hat{f}_3(s)^2$. Thus we expect that the integer power set $\mathcal{P}(k)$ should perform best for f_3, and indeed that is true for all sufficiently small t, even though there is a crossover for larger t.

Finally, the two functions f_4 and f_{13} are examples for which it is best to have noninteger values in the power set $\mathcal{P}$. First consider f_4. It is easy to see that $\hat{f}(s) = 1/(s+\sqrt{s})$ has a series expansion in half powers starting at s, so that we want exactly the power set $\mathcal{P}(k/2)$. And, indeed, that half power set performs best. Next consider $\hat{f}_{13}$. From the analysis of $\hat{f}_4$, we see that f_{13} will again have an expansion in half powers, but now starting at $p_1 = -1/2$, so that $\mathcal{P}((k-1)/2)$ should be best, and it is.

If we cannot construct the series expansion of $\hat{f}$ analytically, then we may be able to proceed numerically. Anticipating the series expansion (14), we can find p_1 and a_1 by searching over p to find when $s^{p+1}\hat{f}(s)$ is approximately constant as a function of positive real s for large s. We then let that constant limiting value be a_1. Given p_1 and a_1, we can find p_2 and a_2 by repeating that analysis for $\hat{f}(s) - a_1 s^{-(p_1+1)}$, and so forth. We make p_1 and p_2 our first two powers. From the initial powers, we may be able to deduce the entire power set. For example, we may decide to use only positive multiples of p_1 if $p_1 > 0$ and $p_2 = 2p_1$. However, we do not carefully explore this approach here, leaving it as an important direction for future research.

As a general observation from Figure 3, we see that in each case the best power algorithm is able to improve upon Gaver-Stehfest for all sufficiently small t and performs similarly for $5 \le t \le 10$. Moreover, we see that the default power set $\mathcal{P}^*$ consistently performs quite well. We also see that the power set can make a big difference, and we have shown how to choose it. However, we find that the default power algorithm tends to perform slightly worse than Gaver-Stehfest for very large t, of the order $100 \le t \le 1{,}000$; see the Online Supplement. Overall, the default power algorithm performs about the same as the Gaver-Stehfest algorithm, which in turn performs much better than the original Gaver (1966) algorithm (without the Salzer acceleration added by Stehfest 1970).

The power algorithms (with appropriate power sets) consistently produce good performance for small values of t, as should be expected, but they yield inconsistent performance for larger values of t. The accuracy as measured by number of significant digits holds for larger values of t, with $t \ge 10$, for the functions f_3, f_4, and f_{13}, but not for f_1, f_9, or f_{12}. The problem with f_9 can be remedied by scaling; see Example 4.1 of Choudhury and Whitt (1997). The other cases can be understood from the analysis of the Gaver-Stehfest algorithm in Valko and Abate (2004) and Abate and Valko (2004). The good transforms f_3, f_4, and f_{13} yielding good performance are in their class **F**, with all singularities of the transform lying on the negative real axis and the function being smooth, while the last two transforms, f_1 and f_{12}, are not in their class **F**, because they have singularities off the negative real axis.

All the transforms in Figure 3 have known closed-form inverses, so inversion is not actually needed for these examples. We note that it is easy to obtain transforms without closed-form inverses by considering operations on these transforms and others. For example, we can consider products of transforms. An m-fold product of Laplace transforms is the transform of the $(m-1)$-dimensional convolution integral of the corresponding time-domain functions, which is somewhat difficult to compute directly. Similarly, in queueing, the steady-state waiting-time distribution in the $M/G/1$ model can be expressed, as a function of the Laplace transform of the service-time distribution, via the Pollaczek-Khintchine Laplace transform; see (1.1) of Abate and Whitt (1992). For service-time distributions with nonrational Laplace transforms, the associated waiting-time distribution is quite complicated, even though the Pollaczek-Khintchine Laplace transform is relatively simple. Many other probability operators that are advantageously approached via Laplace transforms are given in Abate and Whitt (1996).

4. Choosing the Nodes

We have just seen that the choice of the power set can make a big difference in the performance of the inversion. In contrast, surprisingly, the node set makes little difference, provided that the nodes are distinct positive real numbers, specified to high precision, and we solve the linear system with high precision to get

the weights. In particular, we find that the default node set $\mathcal{N}^* \equiv \{1, 2, \ldots, n\}$ used in Step 3 of the algorithm summary in Figure 1 is an excellent choice. In this section we provide justification.

Our starting point for considering possible real node sets was the Gaver-Stehfest algorithm, which has n evenly-space real nodes at $k\ln(2) \approx 0.79k$. We first did many experiments with power algorithms based on the Gaver-Stehfest nodes, but then considered modifying the node set. We found that our *default node set* $\mathcal{N}^* = \{1, 2, \ldots, n\}$ produces essentially the same performance as the Gaver-Stehfest node set.

We conducted several experiments to evaluate the impact of the node set, varying the nodes in many ways, while using the same default power set $\mathcal{P}^*$. We will describe the most revealing experiment in detail, and summarize the rest, leaving the details for the Online Supplement. We consider node sets of the form

$$\alpha_k = \theta + k\delta, \quad 1 \le k \le n = 30, \tag{22}$$

as a function of a positive real *shift* θ and a positive real *spacing* δ. For each candidate node set over a wide range of parameters θ and δ, we solve the system of linear equations to obtain the corresponding weights and examine the resulting inversion error. Figure 4 shows the surface of the average error (measured in the logarithm to base 10) in the inversion of the standard exponential transform $\hat{f}_0$, for $\theta \in \{0, 0.25, \ldots, 4.75\}$ and $\delta \in \{0.1, 0.25, \ldots, 2.95\}$. Surfaces of the average error for other transforms and other values of n ($n = 20, 30, 40$) appear in the Online Supplement.

On the one hand, we find that small spacing, such as $\delta < 0.7$, is bad, especially when combined with zero shift. The performance degradation seems to be due to the largest node being too small; e.g., having all 30 nodes in the interval $[1, 3]$ produces poor results. On the other hand, there is little performance difference across the node sets for $\delta > 0.7$, where the largest node is at least 21. Thus we conclude that the default values $\theta = 0$ and $\delta = 1.0$ associated with $\mathcal{N}^*$ are fine.

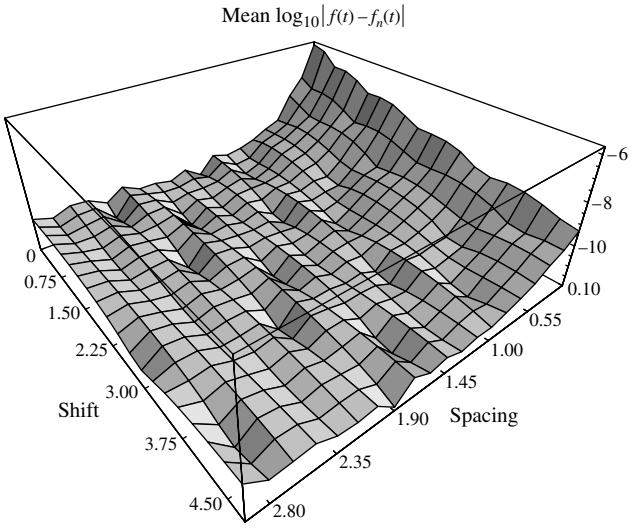

Figure 4 **The Average of the Logarithm of the Absolute Inversion Error as a Function of the Shift θ and the Spacing δ of the Nodes in (22) for f_0: $f(t) = e^{-t}$, $\hat{f}(s) = 1/(s+1)$**

We also carried out several more experiments to examine how the structure of the node sets affects performance. Since the location of the largest node is important, we explored that in more depth, by moving the largest node of the set $\mathcal{N}^* = \{1, 2, \ldots, n\}$ to the right to $n + k$ for different values of k, $k = 0, \ldots, n$ while leaving the rest of the nodes in $\mathcal{N}^*$ intact. Moving the largest node further out does not yield significant improvement. In addition, we compared the performance of node sets with nodes linearly and evenly spaced against the performance of node sets with nodes geometrically spaced in the same interval. The experiments on a few transforms showed that linear spacing is superior to geometric spacing.

Furthermore, we performed experiments with randomly generated sets of nodes. We perturbed one or more nodes by multiplying each by a random number uniformly distributed in the interval $(1 - \varepsilon, 1 + \varepsilon)$, for various small positive ε. The resulting inversions differ little from the unperturbed version, provided that we re-solve the system of linear equations to get new weights. We even used entirely random node sets. We generated 100 independent replications of node sets with $n = 30$ i.i.d. nodes, uniformly distributed in the closed interval $[1, 30]$. We then applied the solution of the system of linear equations to several transforms. In each case, we observed a relatively narrow band of performance results across all the replications.

We conclude that, given a high-precision solution to the linear system, the power algorithm is robust to the choice of the real node set $\mathcal{N}$, confirming that the default node set $\mathcal{N}^*$ is satisfactory.

We also performed a sensitivity analysis to small changes in the weights and nodes. In contrast to the robustness described above, the power algorithm is not robust to small changes in the nodes, for given weights. The algorithm breaks down if we perturb any of the nodes, but do not re-solve the linear system of equations for new weights. The algorithm also breaks down if we perturb the weights, after having solved the linear system, for any given node set.

5. Accuracy and Precision

In this section, we investigate accuracy and precision, measured in the number of digits. We first investigate how the number n of terms in the sum (3) and the computer precision affect the precision of the weights obtained by solving the linear system (9) using the

Table 1 **Minimum Precision of the Computed Weights (Number of Significant Digits, Specifically $|\log_{10}|w_{p+1}-w_p|/|w_p||$, Where w_p Stands for the Weight Computed with Precision p) as a Function of the Computer Precision (Number of Significant Digits) and the Number of Terms, n**

	n				
Precision	20	30	40	50	60
10	0.5	×	×	×	×
20	13	1.7	×	×	×
30	26	14	×	×	×
40	35	23	10	×	×
50	44	32	19	7	×
60	55	42	29	19	2
70	64	52	39	29	19
100	95	81	69	60	47

default node set $\mathcal{N}^*$ and power set $\mathcal{P}^*$. Table 1 shows the minimum precision of the n computed weights as a function of the computer precision and the number of terms, with each ranging over several multiples of 10. From Table 1, we see that the required computer precision to solve the linear system as a function of n is approximately $1.2n$. We have used $1.5n$ as the precision requirement in Figure 1 to be safe. For any given n, the precision of the weights increases with the computer precision, as shown in Table 1.

We find that the precision of the inversion, as measured by the base-10 logarithm of the absolute error $|f_n(t) - f(t)|$ primarily depends on n provided that the computer precision is above the threshold $1.2n$. The performance of the default power algorithm with node set $\mathcal{N}^*$ and power set $\mathcal{P}^*$ tends to be similar to Gaver-Stehfest, as described in Section 7 of Abate and Whitt (2006), but the power algorithm performs slightly better for $0 < t \leq 10$, significantly so for smaller values of t. The ordering is reversed for larger t, though; see the Online Supplement. A specific performance comparison for the transform $\hat{f}_4$ is shown for five values of n in Figure 5. The precision was set high here, so as not to be a factor. Reduction to $1.2n$ produces no significant change.

6. Zakian's Algorithm

Zakian's (1969, 1970, 1973) algorithm is described in Section 2 of Abate and Whitt (2006); also see Wellekens (1970), Singhal and Vlach (1975), Section 3.3 of Davies and Martin (1979), and references therein. It can be derived by introducing a rational approximation for the exponential function in the Bromwich inversion integral (2), i.e.,

$$e^z \approx \sum_{k=1}^{n} \frac{\omega_k}{\alpha_k - z}, \tag{23}$$

where z, α_k, and ω_k are all complex numbers. Zakian chose the complex numbers α_k and ω_k to match the first $2n$ coefficients in the MacLaurin series expansions of the functions in (23). That is accomplished through a *Padé approximation*; see Baker and Groves-Morris (1996) and Saff and Varga (1978). The Padé approximation arises when we apply Gaussian quadrature to the integral (4). We implemented Zakian's algorithm based on the $(n-1)/n$ Padé approximant of e^{-z}, following pp. 522 and 523 of Zakian and Edwards (1978).

Zakian's approach yields the same system of equations as in (7) for the integer power set $\mathcal{P}(k)$, namely,

$$\sum_{k=1}^{n} \frac{\omega_k j!}{\alpha_k^{j+1}} = 1, \quad j = 0, 1, 2, \ldots, 2n-1. \tag{24}$$

Unlike our real power algorithm with n nodes in $\mathcal{N}^*$, here the nodes and weights are both variables, so that there are $2n$ (nonlinear) equations in $2n$ unknowns. The Padé theory guarantees that there is a unique solution and provides an efficient algorithm. It is significant that the nodes and weights from Zakian's algorithm are *complex numbers*, not real numbers.

By considering only integer powers, the Zakian algorithm is especially tuned to polynomials and other smooth functions with derivatives of high orders. Since the nodes and weights are complex numbers, the Zakian algorithm can be quite different from the real power algorithms we consider, even for integer

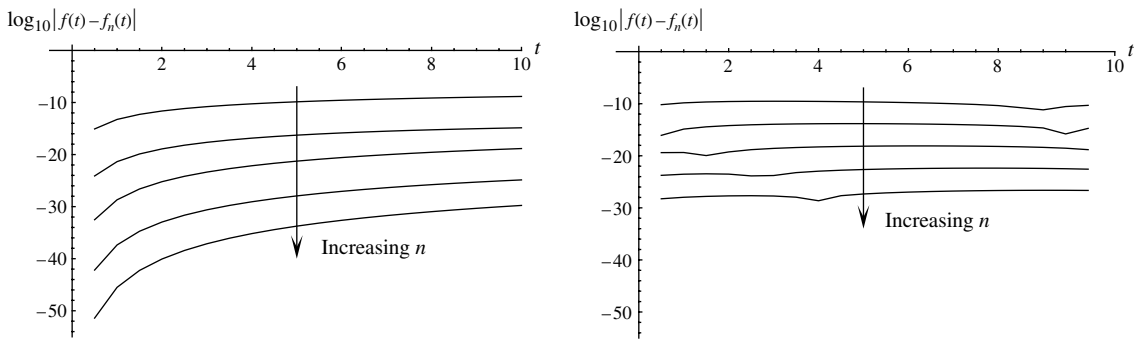

Figure 5 **Performance of the Inversion for $f_4(t) = e^t \operatorname{erfc}(\sqrt{t})$, $\hat{f}_4(s) = 1/(s+\sqrt{s})$ with $\mathcal{P}^*$, $\mathcal{N}^*$ (Left) vs. $\mathcal{G}$ (Right) for $n = 20, 30, 40, 50, 60$**

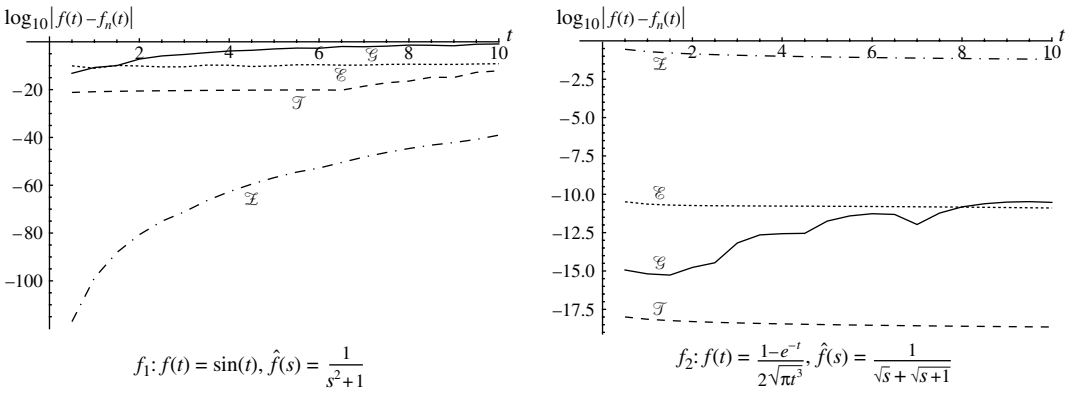

Figure 6 Good and Bad Performance of Zakian's Algorithm

powers. The special structure of the Zakian algorithm ($\mathcal{Z}$) makes it especially effective for such smooth functions. Unfortunately, that special focus comes at a penalty, because the Zakian algorithm turns out to perform poorly for nonsmooth functions. We illustrate this sharply diverging performance in Figure 6. For comparison, we show results for the algorithms $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{T}$, together with $\mathcal{Z}$ in Figure 6. For the smooth function $f_1 \equiv \sin(t)$, Zakian produces far better accuracy than any of the other algorithms, but for the nonsmooth function f_4, Zakian performs worse than the other algorithms. All the algorithms perform poorly for f_1 for large t; see the Online Supplement.

We also exhibit this diverging performance by considering inversions of different functions only for $\mathcal{Z}$ in Figure 7.

7. Evaluating Established Algorithms

We now show how power test functions can be used to evaluate the performance of the established algorithms: Talbot ($\mathcal{T}$), Euler ($\mathcal{E}$), Gaver-Stehfest ($\mathcal{G}$), and Zakian ($\mathcal{Z}$). As shown by Abate and Whitt (2006), these other algorithms all can be expressed in the unified framework (3) by appropriate node and weight vectors α and ω. Thus the algorithms are characterized by these vectors α and ω, which are specified in Abate and Whitt (2006). The Zakian weights and nodes for $n = 30$ are given in the Online Supplement. We display all the node sets in Figure 8.

In order to see how well these inversion algorithms perform in the inversion of power functions, we numerically identify the set of powers p for which the power relation (7) holds to within a small specified error ε, and the complementary set where it is violated. For that purpose, let $p_\varepsilon^+(n)$ to be the smallest nonnegative number p for which the pth power condition is violated by at least a small positive number ε,

$$p_\varepsilon^+(n) \equiv \min\left\{p \geq 0: \left|\Re\left\{\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}}\omega_k\right\} - 1\right| > \varepsilon\right\}. \quad (25)$$

Similarly, let

$$p_\varepsilon^-(n) \equiv \max\left\{p \leq 0: \left|\Re\left\{\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}}\omega_k\right\} - 1\right| > \varepsilon\right\}. \quad (26)$$

We call these functions of n, $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$, the *positive power threshold* and the *negative power threshold*, respectively. As candidate powers p, we consider all numbers $k/2$, $-200 \leq k \leq 200$. As mentioned in Section 1, for $p \leq -1$, the powers correspond to pseudofunctions, as discussed in Sections 12–14 of Doetsch (1974) and the appendix there, and possibly to asymptotic behavior for large t, as characterized by Heaviside's theorem on p. 254 of Doetsch (1974). We will

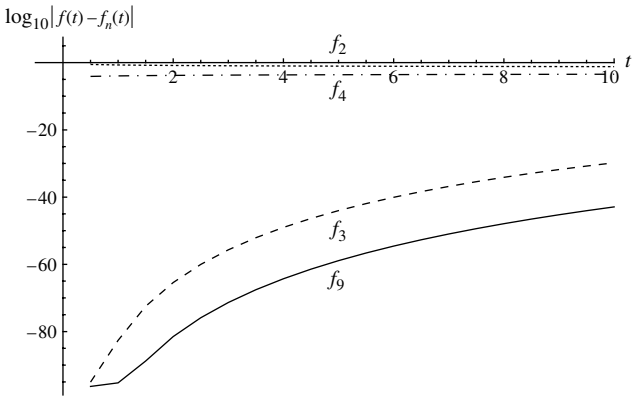

Figure 7 The Extremes of Zakian's Algorithm

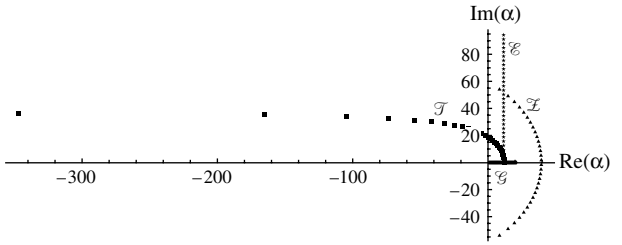

Figure 8 All the Nodes of Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$), and Zakian ($\mathcal{Z}$) for $n = 30$

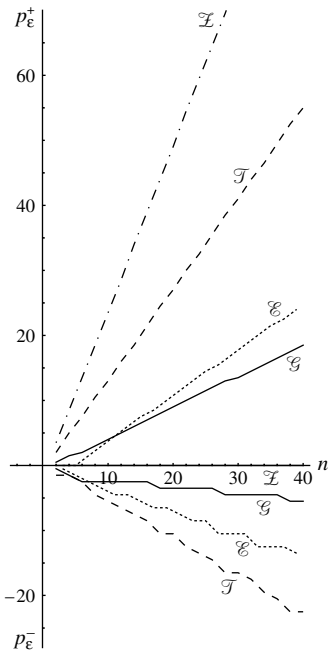

Figure 9 **Scoring the Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$), and Zakian ($\mathcal{X}$) Algorithms Using Power Test Functions in Half Powers. The Error Tolerance Is $\varepsilon = 0.05$**

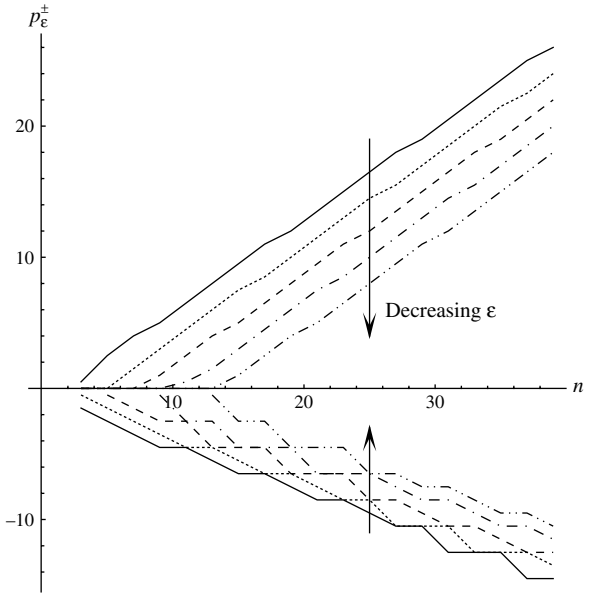

Figure 10 **$p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for $\varepsilon = 5 \times 10^{-j}$, $j = 1, 2, 3, 4, 5$ for the Euler Algorithm**

show that these algorithms, with the exception of Zakian, tend to satisfy the power relations for negative powers.

Thus the range $(p_\varepsilon^-(n), p_\varepsilon^+(n))$ is the smallest contiguous interval of p among half powers for which the pth power relation is satisfied within ε. In Figure 9 we show $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for the four algorithms $\mathcal{G}$, $\mathcal{E}$, $\mathcal{T}$, and $\mathcal{X}$ for $2 \le n \le 40$ and $\varepsilon = 0.05$. We obtain the points shown by fixing n and evaluating the power conditions numerically. For $p \ge 0$, the points to the right and under each curve meet the power conditions within ε, while the points to the left and over do not. Similarly, for $p \le 0$, the points to the right and over each curve meet the power conditions within ε, while the points to the left and over do not.

The single most striking observation from Figure 9 is that the positive and negative power thresholds $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ are *linear* in n, demonstrating consistent regular behavior as n changes. Moreover, for $p \ge 0$, all four existing algorithms meet the first few power conditions within ε, although they vary widely in how many they meet. Figure 9 shows that Euler performs slightly better than Gaver-Stehfest and that Talbot performs much better than either of them, which is consistent with extensive experience, including numerical inversion examples for ten transforms by all these methods in the Online Supplement.

Zakian's algorithm is the best for $p \ge 0$, which translates into the spectacular inversion for f_1 shown in Section 6. On the other hand, for $p < 0$, Zakian's algorithm does not satisfy any power conditions; $p_\varepsilon^-(n) = 0$ for all n. In contrast, the Gaver-Stehfest, Euler, and Talbot algorithms do meet some power conditions within ε for $p < 0$ with the same ranking as for $p \ge 0$. Figure 9 helps explain why the Zakian algorithm is fundamentally different from the other algorithms. The Zakian algorithm evidently is highly tuned for smooth functions (i.e., analytic functions, with derivatives of all orders) at the expense of nonsmooth functions.

We also observed that when $p \notin (p_\varepsilon^-(n), p_\varepsilon^+(n))$, the pth power conditions rapidly diverge from 1. Further analysis indicates that the lines in Figure 9 have additional regularity as we change the error tolerance ε. That is illustrated by Figure 10, which plots $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ as functions of ε for the Euler algorithm $\mathcal{E}$. Our experiments lead us to conclude that the power thresholds are approximately linear in the two variables n and $\log_{10}(\varepsilon)$. The positive power threshold $p_\varepsilon^+(n)$ has approximately the linear formula

$$p_\varepsilon^+(n) \approx c_n n + c_\varepsilon \log_{10}(\varepsilon) + c, \tag{27}$$

where the three parameters c_n, c_ε, and c depend on the algorithm, as indicated in Table 2.

In closing this section, we emphasize that the scoring we have done is based on the power functions,

Table 2 **The Positive Power Threshold $p_\varepsilon^+(n) = c_n n + c_\varepsilon \log_{10}(\varepsilon) + c$ for Different Algorithms**

Algorithm	c_n	c_ε	c
Gaver-Stehfest	0.47	0.79	0.24
Euler	0.70	2.08	−0.52
Talbot	1.39	0.47	−0.17

specifically, the power functions with half powers. We have done experiments indicating that our results extend to quarter powers, but it remains to consider other test functions. Thus one should be careful in extrapolating these results to all test functions.

8. Error Estimates

In this section, we briefly discuss a standard practical method to obtain error estimates (but not bounds) while performing the numerical inversion. We rely on multiple calculations, using different methods or different parameter settings. In doing so, we want to be sure that we produce a genuinely different calculation. That is easily achieved by using a different algorithm. That can easily be achieved with power algorithms by changing the node set, provided that we re-solve the system of linear equations to get new weights in each case.

Suppose that calculation i produces estimate x_i for the desired function value $f(t)$, $1 \le i \le m$. We then estimate the absolute error, say $e(f(t))$, by

$$e(f(t)) \approx \min\{|x_i - x_j|: 1 \le i, j, \le m,\ i \ne j\}. \tag{28}$$

If the pair (i, j) yields the minimum in (28), then either x_i or x_j can serve as the final estimate of $f(t)$. We do this calculation for a low value of m, such as $m = 10$, to ensure that small values are prohibitively unlikely to occur by chance.

In many cases we will have additional information about which estimates are likely to be more accurate. For example, if we use a power algorithm with increasing nodes sets, then we anticipate that the error will decrease with i. In support, we should thus see that the minimum is attained for the pair $(m-1, m)$ and that $|x_i - x_m|$ decreases in i. We would then use x_m as our estimate of $f(t)$ and $|x_{m-1} - x_m|$ as our estimate of the error. Since that should actually estimate the error for x_{m-1}, the estimate should be conservative. In this well-ordered setting we would be suspicious of a minimum obtained for an alternate pair $(i, i+1)$.

We performed experiments for several transforms with known inverses in order to verify that this heuristic procedure is effective, and it is.

9. Extensions

9.1. Gamma Test Functions

With probability applications in mind, as in Abate and Whitt (1995), it is natural to consider gamma probability density functions (pdfs), because arbitrary probability distributions on the positive halfline $[0, \infty)$ can be approximated arbitrarily closely by finite mixtures of gamma distributions. (Gamma distributions can be made arbitrarily close to point masses on the positive halfline, and so finite mixtures of gamma distributions can be made arbitrarily close to finite mixtures of point masses on the positive halfline, which in turn are dense in the family of all probability distributions on the positive halfline, using standard metrics, such as the Prohorov metric; e.g., see p. 77 of Whitt 2002.)

Gamma test functions are also of interest because they are natural generalizations of the power test functions we have considered, but for which the function argument t remains in the picture. Let $g(t; \mu, \lambda)$ be a gamma pdf with rate λ and order μ, and let $\hat{g}(s; \mu, \lambda)$ be its Laplace transform, i.e.,

$$g(t; \mu, \lambda) \equiv \frac{\lambda e^{-\lambda t}(\lambda t)^{\mu-1}}{\Gamma(\mu)} \quad \text{and} \quad \hat{g}(s; \mu, \lambda) \equiv \left(\frac{\lambda}{s+\lambda}\right)^{\mu}. \tag{29}$$

In the unified framework, exact inversion of a gamma pdf at t means

$$g(t; \mu, \lambda) = \frac{1}{t}\sum_{k=1}^{n} \omega_k \hat{g}\left(\frac{\alpha_k}{t}; \mu, \lambda\right), \tag{30}$$

which, upon substitution and simplification, becomes

$$\sum_{k=1}^{n} \frac{\Gamma(\mu)}{(\lambda t + \alpha_k)^{\mu}} \omega_k = e^{-\lambda t}. \tag{31}$$

Note that the power relation (7) is obtained from (31) in the special case $\lambda = 0$. Moreover, for the special case $\mu = 1$, we get the partial fraction representation of the exponential function. From (31), it is apparent that we can develop a *real-variable gamma inversion algorithm* just like the real-variable power inversion algorithm that we have considered.

9.2. Complex Variables

An attractive feature of the power algorithms we have considered is that they only involve real variables, but we can consider extensions to complex variables. The framework (3) also applies if f is a complex-valued function of a nonnegative real variable. Moreover, whether f is real-valued or complex-valued, the weights and nodes in (3) can be real or complex. For example, both are complex in Talbot's algorithm.

If f is complex-valued, then we approximate the real part by

$$\begin{aligned}\Re\{f(t)\} \approx \Re\{f_n(t)\} &\equiv \frac{1}{t}\sum_{k=1}^{n} \Re\left\{\omega_k \hat{f}\left(\frac{\alpha_k}{t}\right)\right\} \\ &= \frac{1}{t}\sum_{k=1}^{n}\Big[\Re\{\omega_k\}\Re\left\{\hat{f}\left(\frac{\alpha_k}{t}\right)\right\} \\ &\qquad - \Im\{\omega_k\}\Im\left\{\hat{f}\left(\frac{\alpha_k}{t}\right)\right\}\Big].\end{aligned} \tag{32}$$

For the general case of complex weights and nodes, the power conditions are

$$\Re\left\{\sum_{k=1}^{n}\frac{\Gamma(p+1)}{\alpha_k^{p+1}}\,\omega_k\right\}=1,\quad p\in\mathcal{P}. \tag{33}$$

In the case of complex nodes and weights, we can again obtain the system of linear equations in (9). We fix the nodes α_k to complex numbers with positive real parts. We can again solve the linear system by *Mathematica*.

We can still formulate an LP that will produce weights that minimize the violation of each equation. For two complex numbers s_1 and s_2, $s_2\neq 0$, we have

$$\Re\left\{\frac{s_1}{s_2}\right\}=\frac{1}{|s_2|^2}(\Re\{s_1\}\Re\{s_2\}+\Im\{s_1\}\Im\{s_2\}). \tag{34}$$

Let $\omega_k^{\Re}$ and $\omega_k^{\Im}$ be the decision variables for the real and imaginary part of ω_k. The real LP that relaxes the constraints and minimizes the violation of each constraint in (8) here becomes the LP with variables $\omega_k^{\Re}$, $\omega_k^{\Im}$, and ε_j:

$$\begin{aligned}\min_{\omega^{\Re},\,\omega^{\Im},\,\varepsilon}\ & \sum_{j=1}^{m}c_j\varepsilon_j\\ \text{s.t. }\ & 1-\varepsilon_j\le\sum_{k=1}^{n}\frac{\Gamma(p_j+1)}{\left|\alpha_k^{p_j+1}\right|^2}\left(\Re\left\{\alpha_k^{p_j+1}\right\}\omega_k^{\Re}+\Im\left\{\alpha_k^{p_j+1}\right\}\omega_k^{\Im}\right)\\ & \le 1+\varepsilon_j,\quad 1\le j\le m,\\ & \varepsilon_j\ge 0,\quad 1\le j\le m,\\ & \Re\{\alpha_1\}\ge\delta,\quad \Re\{\alpha_k\}-\Re\{\alpha_{k-1}\}\ge\delta,\ 2\le k\le n,\end{aligned} \tag{35}$$

where, as before, the positive numbers c_j weigh the violation of the jth row of system (35).

10. Conclusions

10.1. Summary

In Section 2 we developed new real-variable power inversion algorithms within the unified framework (3) proposed by Abate and Whitt (2006). The algorithm for constructing power inversion algorithms is summarized in Figure 1. The algorithm is created by solving the linear system (9), involving the generalized Vandermonde matrix A in (10) to obtain the weights. For given nodes and weights, the algorithm itself is simply the sum (3), typically involving only $n=30$ terms, but high computer precision is required, e.g., $1.5n$.

In Section 3 we showed how the powers can be advantageously chosen starting from a series expansion of the transform, as in (14). In Sections 3 and 4 we developed an effective default power set $\mathcal{P}^*$, displayed in (16), and an effective default node set $\mathcal{N}^*\equiv\{1,2,\ldots,n\}$. The default power algorithm behaves about the same as the Gaver-Stehfest algorithm, providing significant improvement for small t, but performing slightly worse for large t.

In Section 4 we showed that the power algorithm is robust to changes in the positive real node set, provided that the linear system is re-solved to get new weights. In Section 5 we discussed accuracy and precision requirements of the power algorithms.

In Section 6 we discussed the original Zakian ($\mathcal{Z}$) algorithm, which served as motivation for the unified framework. We observed that $\mathcal{Z}$ can be regarded as a special power algorithm, using only integer powers and complex nodes and weights. The nonlinear system of $2n$ equations in $2n$ unknowns (the n nodes and n weights) can be efficiently solved by Padé approximation, which is easily carried out via *Mathematica*. We showed that $\mathcal{Z}$ has inconsistent performance, performing spectacularly well for smooth functions, but poorly for nonsmooth functions.

In Section 7 we showed that the power test functions can be used to evaluate the performance of other algorithms within the unified framework (3). Consistent with experience, this scoring shows the ranking: Talbot ($\mathcal{T}$) > Euler ($\mathcal{E}$) > Gaver-Stehfest ($\mathcal{G}$). That is confirmed by detailed inversions of several transforms by all the algorithms in the Online Supplement. In Section 8 we discussed how to estimate the error in numerical inversion calculations. In Section 9 we mentioned two possible extensions: (i) gamma test functions and (ii) complex-variable power algorithms, where the function f, the node vector α or the weight vector ω can be complex-valued.

10.2. Directions for Future Research

There are many promising directions for future research, which we hope to pursue. First, we still seek theoretical explanation for many of the experimental results reported here: How can we explain the linear structure in Equation (27) with the algorithm-dependent coefficients, as illustrated in Figure 10? How can we better explain the poor performance of the Zakian algorithm for nonsmooth functions, as illustrated in Figures 6 and 7? Can we better understand the performance of the power algorithm as a function of t, gaining insight into when there is degradation in performance for large t, extending the initial work of Abate and Valko (2004)? And what special methods can we develop to treat the difficult functions for large t?

It would also be desirable to make the power algorithms more effective. Toward that end, we need to develop a systematic algorithm to determine the series expansions in (14) and (15). The proposed method in the fourth paragraph from the end of Section 3 is being explored. It would also be of interest to apply the new methods by Demmel and Koev (2005)

for solving linear systems with generalized Vandermonde matrices without subtractive cancellation, to solve the linear systems in the power algorithms more efficiently. It would be of interest to consider further the application of mathematical programming to create power algorithms, e.g., via (8). For fixed nodes, that means linear programming, but with high precision. For variable nodes with weights of alternating signs, that means signomial programming.

Here we have restricted attention to real-valued power algorithms. We have begun to consider analogous complex-valued power algorithms, starting with complex nodes, such as the nodes in $\mathcal{T}$, $\mathcal{E}$, and $\mathcal{Z}$. The general approach is outlined in Section 9. Preliminary experimental results show, first, that it is possible to improve the power algorithms by using complex nodes instead of real nodes and, second, it is possible to improve the other algorithms for some transforms by using their complex-valued nodes but new weights obtained from a power algorithm. In particular, it is possible to improve the performance of Zakian's algorithm dramatically for nonsmooth functions by using the Zakian nodes but replacing the Zakian weights by those obtained from the default power algorithm, based on the Zakian nodes and the default power set $\mathcal{P}^*$.

Finally, it would also be of interest to consider other families of test functions, such as the gamma test functions mentioned in Section 9, and to apply similar methods to other transforms.

Acknowledgments

The authors thank Joseph Abate for his many contributions to this line of research and for helpful discussions related to this paper. The authors also thank the referees for helpful suggestions. The second author was partially supported by National Science Foundation Grant DMI-0457095.

References

Abate, J., P. P. Valko. 2004. Multi-precision Laplace inversion. *Internat. J. Numer. Methods Engrg.* **60** 979–993.

Abate, J., W. Whitt. 1992. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems* **10** 5–88.

Abate, J., W. Whitt. 1995. Numerical inversion of Laplace transforms of probability distributions. *ORSA J. Comput.* **7** 36–43.

Abate, J., W. Whitt. 1996. An operational calculus for probability distributions via Laplace transforms. *Adv. Appl. Probab.* **28** 75–113.

Abate, J., W. Whitt. 2006. A unified framework for numerically inverting Laplace transforms. *INFORMS J. Comput.* **18** 408–421.

Abate, J., G. L. Choudhury, W. Whitt. 1999. An introduction to numerical inversion and its application to probability models. W. Grassman, ed. *Computational Probability*. Kluwer Academic Publishers, Boston, MA, 257–323.

Abramowitz, M., I. A. Stegun. 1972. *Handbook of Mathematical Functions*. National Bureau of Standards, Washington, D.C.

Baker, G. A., P. Graves-Morris. 1996. *Padé Approximants*, 2nd ed., *Encyclopedia of Mathematics and Its Applications*, Vol. 59. Cambridge University Press, Cambridge, UK.

Bellman, R. 1970. *Introduction to Matrix Analysis*, 2nd ed. McGraw-Hill, New York.

Choudhury, G. L., W. Whitt. 1997. Probabilistic scaling for the numerical inversion of nonprobability transforms. *INFORMS J. Comput.* **9** 175–184.

Davies, B., B. L. Martin. 1979. Numerical inversion of Laplace transforms: A critical evaluation and review of methods. *J. Computational Phys.* **33** 1–32.

Demmel, J., P. Koev. 2005. The accurate and efficient solution of a totally positive generalized Vandermonde linear system. *SIAM J. Matrix Appl.* **27** 142–152.

Doetsch, G. 1974. *Introduction to the Theory and Application of the Laplace Transformation*. Springer, New York.

Ecker, J. G. 1980. Geometric programming: Methods, computations and applications. *SIAM Rev.* **22** 338–362.

Gantmacher, F. P. 1959. *Matrix Theory*, Vol. II. Chelsea, New York.

Gantmacher, F. P., M. G. Krein. 2002. *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems*. AMS Chelsea Publishing, Providence, RI.

Gaver, D. P. 1966. Observing stochastic processes and approximate transform inversion. *Oper. Res.* **14** 444–459.

Graf, U. 2004. *Applied Laplace Transforms and z-Transforms for Scientists and Engineers: A Computational Approach Using a Mathematica Package*. Birkhauser-Verlag, Boston, MA.

O'Cinneide, C. A. 1997. Euler summation for Fourier series and Laplace transform inversion. *Stochastic Models* **13** 315–337.

Petrella, G., S. G. Kou. 2004. Numerical pricing of discrete barrier and lookback options via Laplace transforms. *J. Computational Finance* **8** 1–37.

Saff, E. B., R. S. Varga. 1978. On the zeros and poles of Padé approximants to e^z, III. *Numerische Mathematik* **30** 241–266.

Singhal, K., J. Vlach. 1975. Computation of time domain response by numerical inversion of the Laplace transform. *J. Franklin Inst.* **299** 109–126.

Stehfest, H. 1970. Algorithm 368: Numerical inversion of Laplace transforms. *Comm. ACM* **13** 47–49, 624.

Talbot, A. 1979. The accurate inversion of Laplace transforms. *J. Inst. Math. Appl.* **23** 97–120.

Valko, P. P., J. Abate. 2004. Comparison of sequence accelerators for the Gaver method of numerical Laplace transform inversion. *Comput. Math. Applications.* **48** 629–636.

Wellekens, C. J. 1970. Generalization of Vlach's method for the numerical inversion of the Laplace transform. *Electronic Lett.* **6** 742–744.

Whitt, W. 2002. *Stochastic-Process Limits*. Springer, New York.

Wimp, J. 1981. *Sequence Transformations and Their Applications*. Academic Press, New York.

Zakian, V. 1969. Numerical inversion of Laplace transform. *Electronic Lett.* **5** 120–121.

Zakian, V. 1970. Optimisation of numerical inversion of Laplace transforms. *Electronic Lett.* **6** 677–679.

Zakian, V. 1973. Properties of I_{MN} approximants. P. R. Graves-Morris, ed. *Padé Approximants and Their Applications*. Academic Press, New York, 141–144.

Zakian, V., M. J. Edwards. 1978. Tabulation of constants for full grade I_{MN} approximants. *Math. Comput.* **32** 519–531.