INFORMS Journal on Computing
Vol. 18, No. 4, Fall 2006, pp. 408–421
ISSN 1091-9856 | EISSN 1526-5528 | 06 | 1804 | 0408

DOI 10.1287/ijoc.1050.0137
© 2006 INFORMS

A Unified Framework for Numerically Inverting Laplace Transforms

Joseph Abate
900 Hammond Road, Ridgewood, New Jersey 07450-2908, USA

Ward Whitt
Department of Industrial Engineering and Operations Research, Columbia University, New York, New York 10027-6699, USA, ww2040@columbia.edu

We introduce and investigate a framework for constructing algorithms to invert Laplace transforms numerically. Given a Laplace transform $\hat{f}$ of a complex-valued function of a nonnegative real-variable, f, the function f is approximated by a finite linear combination of the transform values; i.e., we use the inversion formula

$$f(t) \approx f_n(t) \equiv \frac{1}{t}\sum_{k=0}^{n} \omega_k \hat{f}\left(\frac{\alpha_k}{t}\right), \quad 0 < t < \infty,$$

where the weights ω_k and nodes α_k are complex numbers, which depend on n, but do not depend on the transform $\hat{f}$ or the time argument t. Many different algorithms can be put into this framework, because it remains to specify the weights and nodes. We examine three one-dimensional inversion routines in this framework: the Gaver-Stehfest algorithm, a version of the Fourier-series method with Euler summation, and a version of the Talbot algorithm, which is based on deforming the contour in the Bromwich inversion integral. We show that these three building blocks can be combined to produce different algorithms for numerically inverting two-dimensional Laplace transforms, again all depending on the single parameter n. We show that it can be advantageous to use different one-dimensional algorithms in the inner and outer loops.

Key words: Laplace transforms; numerical transform inversion; Fourier-series method; Talbot's method; Gaver-Stehfest algorithm; multi-precision computing; multidimensional Laplace transforms; multidimensional transform inversion
History: Accepted by Edward P. C. Kao, Area Editor for Computational Probability and Analysis; received August 2004; revised December 2004; accepted February 2005.

1. Introduction

In recent years, numerical-transform inversion has become recognized as an important technique in operations research, notably for calculating probability distributions in stochastic models. There have now been many applications; e.g., see the survey by Abate et al. (1999) and the textbook treatment by Kao (1997).

Over the years, many different algorithms have been proposed for numerically inverting Laplace transforms; e.g., see the surveys in Abate and Whitt (1992) and Chapter 19 of Davies (2002), the extensive bibliography of Valko and Vojta (2001) and the numerical comparisons by Davies and Martin (1979), Narayanan and Beskos (1982) and Duffy (1993). In contrast to the usual approach, in this paper we do not focus on a particular procedure, but instead introduce and investigate a *framework* that can encompass a wide range of procedures. The flexible framework opens the way to further work, e.g., performing optimization in order to choose the best set of parameters and thus the best procedure, by various criteria, for various classes of functions.

The flexible framework for numerically inverting one-dimensional Laplace transforms is also convenient for developing algorithms to invert multi-dimensional Laplace transforms. We can combine different one-dimensional inversion algorithms to obtain different multi-dimensional inversion algorithms. In particular, given three different one-dimensional inversion algorithms, we directly obtain nine different two-dimensional inversion algorithms, one for each possible combination in the inner and outer loops. These one-dimensional algorithms can be combined with hardly any modifications, in some cases with none at all. Moreover, we show that it can be advantageous from the point of view of algorithm efficiency to combine two different one-dimensional inversion algorithms in the two-dimensional inversion algorithm.

The framework for one-dimensional inversion was originally introduced in several short papers by Zakian (1969, 1970, 1973), but it does not seem to have received much attention, if any. Moreover, in those papers, Zakian did not mention the advantage of having a flexible framework, encompassing several different procedures. Indeed, Zakian proposed a specific procedure, as we will explain in Section 2. The major contribution here, we believe, is suggesting the more general flexible framework, encompassing several different procedures, and showing how that can be exploited in multidimensional inversion algorithms.

To demonstrate the potential of our proposed flexible framework, we look at three specific widely used approaches in this framework, namely, (i) Fourier series expansions with Euler summation, (ii) combinations of Gaver functionals, and (iii) deformation of the contour in the Bromwich integral. These methods are well known as the *Euler algorithm* (the Fourier-series method with Euler summation), e.g., see Dubner and Abate (1968), Abate and Whitt (1992, 1995), Choudhury et al. (1994a), O'Cinneide (1997), Section 19.6 of Davies (2002), and Sakurai (2004); the *Gaver-Stehfest algorithm*, e.g., see Gaver (1966), Stehfest (1970), Section 8 of Abate and Whitt (1992), Section 19.2 of Davies (2002), and Valko and Abate (2004); and *Talbot's method*, e.g., see Talbot (1979), Murli and Rizzardi (1990), Evans (1993), Evans and Chung (2000), Section 19.8 of Davies (2002), and Abate and Valko (2004).

Here is the idea: given a Laplace transform $\hat{f}$ of a complex-valued function f of a nonnegative real-variable,

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

the function f is represented approximately by a finite linear combination of the transform values, via

$$f(t) \approx f_n(t) \equiv \frac{1}{t}\sum_{k=0}^{n} \omega_k \hat{f}\left(\frac{\alpha_k}{t}\right), \quad t > 0, \tag{2}$$

where the *nodes* α_k and *weights* ω_k (i.e., the interior and exterior scaling constants) are complex numbers, which depend on n, but not on the transform $\hat{f}$ or the time argument t. We assume that the Laplace transform $\hat{f}(s)$ in (1) is well defined and analytic for $Re(s) > 0$, where $Re(s)$ is the real part of the complex variable $s \equiv u + iv$; i.e., $Re(s) = u$ for $s = u + iv$ with $i = \sqrt{-1}$ and u and v real numbers. The associated imaginary part of s is $Im(s) = v$. (We have assumed that the "abscissa of convergence" is less than or equal to 0, which is natural for a large class of applications, but that can be generalized. Indeed, it is without loss of generality, because we can make a change of variables.)

Clearly, there are advantages to having the nodes α_k and weights ω_k in the representation of $f_n(t)$ in (2) be independent of both the transform $\hat{f}$ and the time argument t. As a consequence, the procedure can be efficiently applied to multiple transforms and multiple time points. Of course, in general, the accuracy of the approximation will depend on the transform $\hat{f}$ and the time argument t, but there is the potential for great efficiency. Moreover, the same framework applies to multiple procedures as well, so that independent accuracy checks are readily available too.

In most applications, f will be real-valued; then we approximate by the real part, i.e.,

$$\begin{aligned} Re\{f(t)\} \approx Re\{f_n(t)\} &\equiv \frac{1}{t}\sum_{k=0}^{n} Re\left\{\omega_k \hat{f}\left(\frac{\alpha_k}{t}\right)\right\} \\ &= \frac{1}{t}\sum_{k=0}^{n}\left[Re\{\omega_k\} Re\left\{\hat{f}\left(\frac{\alpha_k}{t}\right)\right\} \right. \\ &\qquad \left. - Im\{\omega_k\} Im\left\{\hat{f}\left(\frac{\alpha_k}{t}\right)\right\}\right]. \end{aligned} \tag{3}$$

The case of complex-valued functions arises naturally when we consider inner loops in the inversion of multidimensional transforms; see Section 3.

For any given n, and any given set of approximating parameters $\{\omega_k, \alpha_k, 1 \le k \le n\}$, the function f_n on the right in (2) serves as an approximation of the function f, which we intend to apply for all t that are continuity points of f with $0 < t < \infty$. It is understood that there may be numerical difficulties if t is either very small or very large. In those exceptional cases it is often preferable to apply asymptotics as $t \to 0$ or $t \to \infty$, e.g., from the initial-value and final-value theorems (Doetsch 1974); e.g., as in Abate and Whitt (1997). Alternatively, it may be desirable to scale the function, as in Choudhury and Whitt (1997).

For any specific t, we can increase n in order to obtain better accuracy. However, greater system precision will typically be required to perform the calculation as n increases. As in Abate and Valko (2004), we suggest exploiting multi-precision software, which is now widely available through computer algebra systems such as *Mathematica* and *Maple* in order to obtain the required precision; e.g., see Graf (2004). Such high precision is also an integral part of *UBASIC*, which the first author has been using for more than twenty years.

In fact, in this paper we *assume* that multi-precision software is being used, so we do not consider special measures to control roundoff error within limited precision (e.g., standard double precision). The framework may also be useful with limited precision, provided appropriate measures are taken to control roundoff error, but some procedures, such as the Gaver-Stehfest algorithm, usually require high precision.

For one of the specific inversion routines we consider here (the Euler algorithm), the required precision as a function of n was identified on p. 272 of Abate et al. (1999). For the other two specific inversion routines we consider here, Abate and Valko (2004) identified the required precision as a function of n for a large class of transforms. (The full story is rather complicated; see Abate and Valko 2004 for more details.) Thus in our implementations of the three basic methods we obtain "automatic algorithms": given the accuracy we want for $f(t)$ (by which we mean relative accuracy, measured in significant digits; the number of significant digits is defined as $-\log_{10}$(relative error); see (34)), a formula specifies the appropriate parameter n; then, given n, the program determines the required precision for the computation, using another formula, so that the desired accuracy for $f(t)$ is realized. Of course, if there are difficulties, then the parameter n or the system precision can be increased.

Different specific algorithms are obtained by introducing different families of approximating parameters $\{\omega_k, \alpha_k, 0 \le k \le n\}$. Indeed, Stehfest (1970) originally expressed the Gaver-Stehfest method in this form. We will cast the Fourier-series method with Euler summation and Talbot's method in this form as well. It turns out that other methods also are already in this form. In particular, the Padé method used by Zakian (1969, 1970) and Vlach (1969) and the Gaussian quadrature method of Piessens (1969, 1971) and Wellekens (1970) are in this form, but we will not examine them in detail here.

On the other hand, there also are methods that are not, at least directly, in the form of (2). Among these is the popular Laguerre or Weeks method; e.g., see Abate et al. (1998) and Section 19.5 of Davies (2002). There we have

$$f(t) \approx \sum_{i=0}^{n} c_i l_i(bt), \tag{4}$$

where $l_i(t)$, $i \ge 0$, are the standard Laguerre functions, which form an orthonormal basis for the function space $L_2([0,\infty), \mathbf{R})$, while the weights c_i and the scale factor b depend on the transform $\hat{f}$.

A main purpose for the unified framework is to develop new algorithms for multidimensional transform inversion. So far, relatively little attention has been given to inversion of multidimensional Laplace transforms; e.g., see Choudhury et al. (1994a), Abate et al. (1998), Hwang and Lu (1999), and references therein. The Fourier-series multidimensional inversion routines in Choudhury et al. (1994a) have been applied to tackle challenging performance-analysis problems in the steady-state analysis of stochastic networks and the time-dependent analysis of queues with time-varying rates in Choudhury et al. (1994b, 1995a, b, 1997). Choudhury et al. (1997) includes an example of a 21-dimensional inversion.

It is significant that the simple framework in (2) extends easily to multidimensional transforms. For any two-dimensional transform $\tilde{f}(s_1, s_2)$ of a two-dimensional complex-valued function of nonnegative real variables, $f(t_1, t_2)$, the corresponding two-dimensional inversion formula is

$$f(t_1, t_2) \approx f_{n_1, n_2}(t_1, t_2) \equiv \frac{1}{t_1 t_2} \sum_{k_1=0}^{n_1} \omega_{k_1} \sum_{k_2=0}^{n_2} \omega'_{k_2} \tilde{f}\left(\frac{\alpha_{k_1}}{t_1}, \frac{\alpha'_{k_2}}{t_2}\right), \tag{5}$$

where n_1 and n_2 are the parameters, k_1 and k_2 are the indices, ω_{k_1} and ω'_{k_2} are the weights, and α_{k_1} and α'_{k_2} are the nodes for the outer and inner loops, respectively. We can calculate the approximant $f_{n_1, n_2}(t_1, t_2)$ at any argument (t_1, t_2) provided that we can evaluate the double transform $\tilde{f}(s_1, s_2)$ at any argument (s_1, s_2). As in (2) and (3), we use the real part if $f(t_1, t_2)$ is real-valued.

Given the three specific one-dimensional algorithms we consider in the framework (2), namely, Gaver ($\mathcal{G}$), Euler ($\mathcal{E}$), and Talbot ($\mathcal{T}$), we can combine them in every possible combination to obtain *nine* two-dimensional algorithms. In particular, it is not necessary to use the same routine in both the inner and outer loop. Indeed, we will show that *it can be advantageous to combine two different one-dimensional routines*. We are unaware of this ever having been done before.

In this paper, we examine these nine two-dimensional algorithms, using the notation $\mathcal{G}\mathcal{T}$, where the first operator $\mathcal{G}$ applies to the outer loop, while the second operator $\mathcal{T}$ applies to the inner loop. We discuss the performance of these two-dimensional routines, illustrating with numerical examples.

Here is how the rest of this paper is organized: we start in Section 2 by explaining the unified framework. Then in Section 3 we elaborate on the extension of the framework to two dimensions. In the next three sections we introduce specific algorithms in the framework. In Sections 4, 5, and 6 we briefly specify our versions of the one-dimensional Gaver-Stehfest, Fourier-series (with Euler summation), and Talbot algorithms in the unified framework. Then in Section 7 we discuss the performance of these one-dimensional algorithms. In Section 8 we show how the three one-dimensional inversion algorithms can be combined to produce specific two-dimensional algorithms. In Section 9 we present numerical examples evaluating the performance of the two-dimensional inversion algorithms. In Section 10 we draw conclusions.

2. The Unified Framework

In this section we provide motivation for the general inversion formula (2), explaining how it was originally derived. We present two derivations: first, one based on the Bromwich inversion integral due to Wellekens (1970) and, second, the original one based on approximating delta functions due to Zakian (1969, 1970, 1973). However, we will not use those specific inversion routines later in the paper. We start by observing that it is natural to have t^{-1} appear both inside and outside the transform $\hat{f}$ in the inversion formula for $f(t)$ in (2).

Initial-Value and Final-Value Theorems

One way to see the role of t^{-1} is to consider appropriate Abelian and Tauberian theorems, in particular, the familiar initial-value and final-value theorems; e.g., see Bingham et al. (1987), Doetsch (1974), or Feller (1971). In simple form, the *initial-value theorem* states that, when the two limits both exist,

$$\lim_{t\downarrow 0} f(t) = \lim_{t\downarrow 0} \frac{1}{t}\hat{f}\left(\frac{1}{t}\right), \tag{6}$$

while the *final-value theorem* states that, when the two limits both exist,

$$\lim_{t\uparrow \infty} f(t) = \lim_{t\uparrow \infty} \frac{1}{t}\hat{f}\left(\frac{1}{t}\right). \tag{7}$$

Moreover, other refined asymptotics agree when the transform is scaled in that way, e.g., see Heavyside's theorem, Theorem 37.1 of Doetsch (1974).

Starting from the Bromwich Inversion Integral

We now start to review Wellekens' (1970) argument, which begins with the Bromwich inversion integral. Since the Euler and Talbot algorithms also can be derived from the Bromwich inversion integral, this argument explains why they too fit in the framework (2).

Suppose that f is a real-valued function of a nonnegative real variable, which is continuous at t. The Bromwich inversion integral expresses $f(t)$ as the *contour integral*

$$f(t) = \frac{1}{2\pi i}\int_C \hat{f}(s)e^{st}\,ds, \quad t > 0, \tag{8}$$

where the contour C extends from $c - i\infty$ to $c + i\infty$ for $c > 0$, falling to the right of all singularities of $\hat{f}$, under the usual regularity conditions; see Theorem 24.4 of Doetsch (1974).

Following Wellekens (1970), we make the change of variables $z = st$ and rewrite the contour integral as

$$f(t) = \frac{1}{2\pi i t}\int_{C'} \hat{f}(z/t)e^{z}\,dz, \quad t > 0, \tag{9}$$

where C' is the same contour as a function of z. By that step alone, we see how t^{-1} appears both in the multiplicative constant and the argument of the transform $\hat{f}$. Indeed, almost any method of numerical integration (approximate quadrature) applied to the integral in (9) produces the representation (2); see Davis and Rabinowitz (1984).

Rational Approximations for the Exponential Function

To achieve the unified framework in (2) from (9), Wellekens (1970) approximated the exponential function e^z in the integrand of (9) by a rational function. Indeed, both of the earlier approaches exploit rational approximations for the exponential function, so we digress to discuss that intermediate step now.

Writing the rational function in a *partial-fraction* representation (e.g., see p. 76 of Doetsch 1974), we have

$$e^z \approx \sum_{k=0}^{n} \frac{\omega_k}{\alpha_k - z}, \tag{10}$$

where z is a complex variable and α_k and ω_k are complex numbers.

We will impose regularity conditions on the complex numbers α_k and ω_k appearing in (10): First, we will assume that the parameters α_k are distinct, implying that the rational approximant in (10) has $n+1$ poles. We will require that the approximant be analytic for $Re(z) > 0$. Consequently, we assume that $Re(\alpha_k) < 0$ for all k. We also assume that the pairs of complex numbers (α_k, ω_k) occur in *complex-conjugate pairs*. (If $s \equiv u + iv$, then its complex conjugate is $\bar{s} \equiv u - iv$.) That is, if either α_k or ω_k has a non-zero imaginary part, then we also have the pair $(\bar{\alpha}_k, \bar{\omega}_k)$ among the n terms in the sum (10). Since

$$\frac{\omega_k}{\alpha_k - z} + \frac{\bar{\omega}_k}{\bar{\alpha}_k - z}$$

is real for all real z, that condition guarantees that the approximating rational function is real-valued for all real z. Consequently, we can alternatively express (10) as

$$e^z \approx \frac{1}{2}\sum_{k=0}^{n}\left[\frac{\omega_k}{\alpha_k - z} + \frac{\bar{\omega}_k}{\bar{\alpha}_k - z}\right], \tag{11}$$

where $Re(\alpha_k) < 0$ for all k.

Of course, we want to choose the rational function in (10) or (11) so that we obtain a good approximation. A classic way to do that is to choose the complex numbers α_k and ω_k so as to match the MacLaurin series of the two functions as far as possible. That procedure is effective; indeed, that procedure is the classical method of *Padé approximation*; moreover, the exponential function was one of the first functions to be treated; see the Preface and Section 1.2 of Baker and Graves-Morris (1996). It should thus not be surprising that Zakian (1969, 1970, 1973) and Wellekens (1970) exploited Padé approximation when they approximated the exponential function, but of course there are many other possible rational approximations for the exponential function, with merits depending on the context.

The Rest of Wellekens' Argument
Continuing to follow Wellekens (1970), we achieve the unified framework in (2) from (9) by approximating the exponential function e^z in the integrand of (9) by a rational function, using (10) or (11). For example, using (10), we get

$$\begin{aligned} f_n(t) &\approx \frac{1}{2\pi i t}\int_{C'}\hat{f}(z/t)\sum_{k=0}^{n}\frac{\omega_k}{\alpha_k - z}\,dz \\ &= \frac{1}{t}\sum_{k=0}^{n}\frac{1}{2\pi i}\int_{C'}\hat{f}(z/t)\frac{\omega_k}{\alpha_k - z}\,dz \\ &= \frac{1}{t}\sum_{k=0}^{n}\omega_k\hat{f}\left(\frac{\alpha_k}{t}\right), \quad t>0, \end{aligned} \tag{12}$$

using the Cauchy integral formula in the last step, e.g., p. 9 of Davies (2002).

It still remains to choose the specific rational approximation for the exponential function. Wellekens (1970) shows that the classic Padé approximation arises when we perform Gaussian quadrature to the integral (9). Thus there is additional justification for Padé approximation.

Zakian's Original Argument
Zakian originally obtained (2) in a very different way, in particular, by approximating a delta function by a finite linear combination of exponential functions. Let $\delta((x/t)-1)$ be the scaled delta function that attaches mass t to the point t, for each $t>0$; i.e., it satisfies the properties

$$\int_0^T \delta((x/t)-1)\,dx = t \quad \text{and} \quad \delta((x/t)-1)=0 \quad \text{for } x\neq t,\ 0<t<T. \tag{13}$$

More precisely, if f is an integrable complex-valued function of a nonnegative real variable that is continuous at t, then

$$f(t)=\frac{1}{t}\int_0^T f(x)\delta((x/t)-1)\,dx, \quad 0<t<T. \tag{14}$$

Note that δ actually is a function of *two* variables, x and t.

The idea, then, is to approximate the scaled delta function $\delta((x/t)-1)$ by a finite linear combination of exponential functions; i.e.,

$$\delta((x/t)-1)\approx\delta_n((x/t)-1)\equiv\sum_{k=0}^{n}\omega_k e^{-\alpha_k x/t}, \tag{15}$$

where α_k and ω_k are complex numbers. It may not be immediately clear that this is a good thing to do, but we explain later.

Now we consider an integrable function f that we want to evaluate, again assumed to be a complex-valued function of a nonnegative real variable. For each continuity point t of f, we approximate $f(t)$ by the associated integrals with respect to $\delta_n(x/t-1)$; i.e., combining (14) and (15), we use the approximation

$$\begin{aligned} f(t) \;\approx\; f_n(t) &\equiv \frac{1}{t}\int_0^T f(x)\delta_n((x/t)-1)\,dx, \quad 0<t<T, \\ &= \frac{1}{t}\int_0^T f(x)\sum_{k=0}^{n}\omega_k e^{-\alpha_k x/t}\,dx, \\ &= \frac{1}{t}\sum_{k=0}^{n}\omega_k\int_0^T f(x)e^{-\alpha_k x/t}\,dx, \\ &\to \frac{1}{t}\sum_{k=0}^{n}\omega_k\int_0^\infty f(x)e^{-\alpha_k x/t}\,dx, \\ &= \frac{1}{t}\sum_{k=0}^{n}\omega_k\hat{f}\left(\frac{\alpha_k}{t}\right), \end{aligned} \tag{16}$$

where we let $T\to\infty$ in the last step, with the limit being justified by standard assumptions on f guaranteeing that the Laplace transform is well defined. We see that (16) is of the same form as (2).

The linearity in formula (2) implies that the inversion can be applied either to the complex-valued function f or to its real and imaginary parts: Suppose that f is a complex-valued function and let $f^{(1)}\equiv Re(f)$ and $f^{(2)}\equiv Im(f)$ be the real and imaginary parts of f. Then

$$f(t)=f^{(1)}(t)+if^{(2)}(t), \qquad \hat{f}(s)=\hat{f}^{(1)}(s)+i\hat{f}^{(2)}(s) \tag{17}$$

and

$$\begin{aligned} f_n(t) &\equiv \frac{1}{t}\sum_{k=0}^{n}\omega_k\hat{f}\left(\frac{\alpha_k}{t}\right) \\ &= \frac{1}{t}\sum_{k=0}^{n}\omega_k\left[\hat{f}^{(1)}\left(\frac{\alpha_k}{t}\right)+i\hat{f}^{(2)}\left(\frac{\alpha_k}{t}\right)\right] \\ &= \frac{1}{t}\sum_{k=0}^{n}\omega_k\hat{f}^{(1)}\left(\frac{\alpha_k}{t}\right)+i\frac{1}{t}\sum_{k=0}^{n}\omega_k\hat{f}^{(2)}\left(\frac{\alpha_k}{t}\right) \\ &= \frac{1}{t}\sum_{k=0}^{n}Re\left\{\omega_k\hat{f}^{(1)}\left(\frac{\alpha_k}{t}\right)\right\}+i\frac{1}{t}\sum_{k=0}^{n}Re\left\{\omega_k\hat{f}^{(2)}\left(\frac{\alpha_k}{t}\right)\right\} \\ &\equiv f_n^{(1)}(t)+if_n^{(2)}(t). \end{aligned} \tag{18}$$

It remains to show that the approximants δ_n in (15) can indeed serve as good approximants for the scaled delta function δ. At first, it may not be obvious that the delta function admits a reasonable approximation of the form (15). Indeed, it is obvious that the approximation δ_n cannot be a good approximation for the delta function δ uniformly in the neighborhood of its discontinuity, since the approximant is a continuous function, but nevertheless we can hope to approximate $f(t)$ at points t that are continuity points of f.

As Zakian (1970) observed, the possibility of good approximations is evident if we take Laplace transforms: The Laplace transform of the scaled delta function is

$$\hat{\delta}(s,t) \equiv \int_0^\infty e^{-sx}\delta((x/t)-1)\,dx = te^{-ts}, \tag{19}$$

while the Laplace transform of the general nth approximant is

$$\hat{\delta}_n(s,t) \equiv \int_0^\infty e^{-sx}\delta_n((x/t)-1)\,dx = \sum_{k=0}^{n} \frac{\omega_k}{s+(\alpha_k/t)}. \tag{20}$$

Assuming that the nodes α_k are distinct for $0 \le k \le n$, we see that the Laplace transform of the approximant is a proper rational function with poles at α_k/t, expressed in a partial-fraction representation with partial fraction coefficients ω_k.

At this point we exploit uniqueness and continuity theorems for Laplace transforms, e.g., see Doetsch (1974) and Chapter 13 of Feller (1971), to justify characterizing functions and measures in terms of Laplace transforms. Thus, assuming that we can work with Laplace transforms, in order to approximate the scaled delta function $\delta((x/t)-1)$ by its approximant $\delta_n((x/t)-1)$ in (15), it suffices to approximate the exponential function te^{-ts} in (19) by the rational functions in (20). As observed by Zakian (1970), that problem is the well-studied problem of Padé approximation.

Summary
We regard the discussion so far in this section as providing support for considering the unified framework in (2). Seeing those derivations helps us understand why the framework (2) is natural. Zakian (1969, 1970, 1973) and Wellekens (1970) went further, exploiting Padé approximation, to obtain specific algorithms, considering them as "the solution." In contrast, we suggest considering alternative approximants within this framework. In Sections 4, 5, and 6 we discuss three very different ways to arrive at approximants, using well-established inversion algorithms. In that way, we demonstrate that the framework may be useful without necessarily using Padé approximation.

Framework for Optimization
As mentioned earlier, the flexible framework presents the opportunity to perform optimization to select the "best" procedure, by some criterion. Indeed, one such optimization is the Padé approximation performed by Zakian (1970). As an example of a different approach, we could consider a given value of n and choose the resulting $2n+2$ parameters by doing optimal fitting, using families of test functions and mathematical programming. To be concrete, suppose that we consider m_1 test functions f_i and m_2 arguments of interest t_j. Let $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ be the vectors of weights and nodes, i.e., $\boldsymbol{\omega} \equiv (\omega_0, \ldots, \omega_n)$ and $\boldsymbol{\alpha} \equiv (\alpha_0, \ldots, \alpha_n)$. Then our object might be to choose the vectors $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ in order to minimize the squared error. That is, we might perform the minimization

$$\min_{\boldsymbol{\omega},\,\boldsymbol{\alpha}} \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\left(f_i(t_j) - \frac{1}{t_j}\sum_{k=0}^{n}\omega_k \hat{f}_i\left(\frac{\alpha_k}{t_j}\right)\right)^2, \tag{21}$$

where the decision variables $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ are vectors of complex numbers. In the optimization (21), the test functions f_i and their transforms $\hat{f}_i$ are presumed to be known. We might also put further constraints upon the variables $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$. For example, paralleling the Gaver-Stehfest procedure, we could require that they be real numbers. Finally, given the vectors $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$, we have a new algorithm in the unified framework (2) to apply to new transforms of interest. This seems to be an interesting direction of research, but we do not pursue it here; see Audis and Whitt (2006).

3. Two-Dimensional Inversion Algorithms

Starting from a function of two variables, $f(t_1, t_2)$, we can construct the Laplace transform in a two-step process, applying the one-dimensional construction twice: first, consider t_2 a constant, and apply (1) with the complex variable s_1 to f to get the one-dimensional Laplace transform

$$\hat{f}(s_1, t_2) \equiv \int_0^\infty e^{-s_1 t_1} f(t_1, t_2)\,dt_1, \tag{22}$$

which we assume is well defined for $Re(s_1) > 0$. Then consider the complex variable s_1 a constant, and apply (1) a second time with the complex variable s_2 to get the two-dimensional Laplace transform

$$\tilde{f}(s_1, s_2) \equiv \int_0^\infty e^{-s_2 t_2} \hat{f}(s_1, t_2)\,dt_2, \tag{23}$$

which we assume is well defined for $Re(s_1) > 0$ and $Re(s_2) > 0$.

Likewise, two-dimensional transform inversion can be considered a two-step process, applying the one-dimensional inversion formula (2) twice: For specified argument pair (t_1, t_2), first invert $\tilde{f}(s_1, s_2)$ with respect to s_2 (inner loop), regarding s_1 as constant, to get an approximation for $\hat{f}(s_1, t_2)$. Then invert the approximation for $\hat{f}(s_1, t_2)$ with respect to s_1 (outer loop), regarding t_2 as constant, getting the desired approximation for $f(t_1, t_2)$.

It is straightforward to extend the unified framework to two dimensions: Following the two-step procedure just described, we have

$$\hat{f}(s_1, t_2) \approx \hat{f}_{n_2}(s_1, t_2) \equiv \frac{1}{t_2}\sum_{k_2=0}^{n_2} \omega'_{k_2} \tilde{f}\left(s_1, \frac{\alpha'_{k_2}}{t_2}\right) \tag{24}$$

and

$$f(t_1, t_2) \approx f_{n_1}(t_1, t_2) \equiv \frac{1}{t_1} \sum_{k_1=0}^{n_1} \omega_{k_1} \hat{f}_{n_2}\left(\frac{\alpha_{k_1}}{t_1}, t_2\right). \tag{25}$$

Combining (24) and (27), we obtain the general form for the inversion

$$f(t_1, t_2) \approx f_{n_1, n_2}(t_1, t_2) \equiv \frac{1}{t_1 t_2} \sum_{k_1=0}^{n_1} \omega_{k_1} \sum_{k_2=0}^{n_2} \omega'_{k_2} \tilde{f}\left(\frac{\alpha_{k_1}}{t_1}, \frac{\alpha'_{k_2}}{t_2}\right), \tag{26}$$

as in (5).

We can justify the two-dimensional inversion formula the same way we did for the one-dimensional case in Section 2. When we do, we see that the two-dimensional objects are just the product of the two corresponding one-dimensional objects. For example, the two-dimensional scaled delta function is just the product of the two one-dimensional delta functions. Moreover, the two-dimensional linear combination of exponentials is the product of the two one-dimensional linear combinations of exponentials. Thus the Laplace transforms are the products of the two one-dimensional Laplace transforms. Hence the one-dimensional Padé approximation can be applied to each dimension in the two-dimensional case.

As a consequence of the general two-dimensional framework in (26), we can apply the three one-dimensional routines in Sections 4, 5, and 6 in *any* combination to obtain *nine* different candidate two-dimensional routines. Even though the Gaver-Stehfest routine works only with real weights and nodes, we can apply the Gaver-Stehfest routine in the inner loop with any of these other procedures, because the Gaver-Stehfest routine extends to complex-valued functions via (17) and (18).

From the experience in Choudhury et al. (1997), we know that it may be necessary to do extra work for the inner loop to ensure that the intermediate function $\hat{f}(s_1, t_2)$ has sufficient accuracy to perform the calculation in the outer loop accurately. Thus, we anticipate that we may need $n_2 > n_1$. However, for ease of use, it is desirable to have only one parameter n, as in the one-dimensional case. We can achieve that by letting

$$n_1 = n \quad \text{and} \quad n_2 = cn \tag{27}$$

for a fixed parameter c, which depends on the specific routines used in the inner and outer loops. That is assuming that the system precision required in the outer loop and the accuracy produced in the inner loop both grow proportionally to n, which experience shows to be approximately so with the algorithms considered here.

What is surprising is that it often suffices to have $c = 1$. For the specific two-dimensional algorithms we consider, built from the three one-dimensional algorithms $\mathscr{G}$, $\mathscr{E}$, and $\mathscr{T}$, we find that $c = 1$ often suffices; see Section 8.

4. The Gaver-Stehfest Algorithm

Now we start specifying specific one-dimensional inversion algorithms in the unified framework (2). We start with the Gaver-Stehfest procedure, because it directly appears in the framework. The Gaver-Stehfest algorithm is distinguished by the fact that it does not use complex numbers; the weights and nodes are real numbers.

The Gaver-Stehfest procedure is based on the sequence of Gaver approximants, $\{f_n(t)\colon n \geq 1\}$, derived by Gaver (1966), which can be written as

$$f_n(t) \equiv \frac{n \ln(2)}{t} \binom{2n}{n} \sum_{k=0}^{n} (-1)^k \binom{n}{k} \hat{f}\left(\frac{(n+k)\ln(2)}{t}\right). \tag{28}$$

There is a simple probabilistic derivation, which is reviewed in Section 8 of Abate and Whitt (1992). Note that the Gaver approximants are directly in the framework (2).

The Gaver approximants can be computed by a recursive algorithm, namely,

$$\begin{aligned} G_0^{(k)} &= \tfrac{\gamma k}{t} \hat{f}(k\gamma/t), \quad 1 \leq k \leq 2n, \\ G_j^{(k)} &= (1 + (k/j)) G_{j-1}^{k} - (k/j) G_{j-1}^{(k+1)}, \\ &\qquad\qquad 1 \leq j \leq n,\ j \leq k \leq 2n - j, \\ f_n(t) &= G_n^{(n)}, \end{aligned} \tag{29}$$

where $\gamma \equiv \ln(2)$.

Unfortunately, however, the convergence $f_n(t) \to f(t)$ as $n \to \infty$ is slow (logarithmic), so acceleration is needed. Accordingly, Stehfest (1970) proposed the *linear* Salzer acceleration scheme. Since the acceleration method is linear, the Gaver-Stehfest algorithm fits in the framework (2), just like the Gaver approximants in (28).

This linear acceleration method is reviewed by Valko and Abate (2004), where the Salzer method is compared to popular *nonlinear* sequence accelerators. Valko and Abate found that the Salzer scheme performed remarkably well, being outperformed only by the Wynn rho algorithm. As an aside, we point out that the framework (2) is distinguished by encompassing linear acceleration techniques, but not nonlinear acceleration techniques, when acceleration is used.

Here we focus on the original Salzer scheme proposed by Stehfest, because it performs well and because it is of the right (linear) form. For any $t > 0$ and positive integer M, Salzer summation yields the Gaver-Stehfest inversion formula

$$f_g(t, M) = \sum_{n=1}^{M} (-1)^{n+M} \left(\frac{n^M}{M!}\right) \binom{M}{n} f_n(t), \tag{30}$$

where $f_n(t)$ is given in (28).

Putting (28) into (30) and rearranging the double summation, we get

$$f_g(t, M) = \frac{\ln(2)}{t} \sum_{k=1}^{2M} \zeta_k \hat{f}\left(\frac{k\ln(2)}{t}\right), \tag{31}$$

where

$$\zeta_k = (-1)^{M+k} \sum_{j=\lfloor (k+1)/2 \rfloor}^{k \wedge M} \frac{j^{M+1}}{M!} \binom{M}{j} \binom{2j}{j} \binom{j}{k-j}, \tag{32}$$

with $\lfloor x \rfloor$ being the greatest integer less than or equal to x and $k \wedge M \equiv \min\{k, M\}$.

Inversion formula (31) is in the form (2) with $n = 2M$, $\alpha_k = k\ln(2)$, and $\omega_k = \ln(2)\zeta_k$ for ζ_k in (32). We shift the notation from n to M primarily because we would have to require that n be an even integer. The parameter M is natural too, because the algorithm uses M Gaver functionals; the extra M terms in (31) are used to perform the Salzer summation. We consistently use the parameter M for the rest of this paper.

The weights ζ_k in (32) (and thus ω_k in (2)) depend on both n and k. Since the nodes $\alpha_k = k\ln(2)$ are constant multiples of k, there is even spacing of the nodes along the real line. We remark that

$$\sum_{k=0}^{2M} \zeta_k = 0 \quad \text{for all } M \geq 1. \tag{33}$$

Because of the binomial coefficients in the weights, the Gaver-Stehfest algorithm tends to require high system precision in order to yield good accuracy in the calculations. From Abate and Valko (2004), we conclude that the required system precision is about $2.2M$ when the parameter is M. The precision requirement is driven by the coefficients ζ_k in (32). Such a high level of precision is *not* required for the computation of the transform $\hat{f}(s)$.

Abate and Valko (2004) investigated, experimentally, the precision produced as a function of the parameter M. They found that the answer depends on the transform. From extensive experimentation, Abate and Valko (2004) conclude that about $0.90M$ significant digits are produced for $f(t)$ with good transforms. By $0.90M$ significant digits, we mean that

$$\text{relative error} = \left| \frac{f(t) - f_g(t, M)}{f(t)} \right| \approx 10^{-0.90M}. \tag{34}$$

In other words, we are considering the relative error. We also need to explain what we mean by *good transforms*. Transforms are said to be "good" (of their class **F**) if the transforms have all their singularities on the negative real axis and the functions f are infinitely differentiable for all $t > 0$. If the tranforms are not good, then the number of significant digits may not be so great and may not be proportional to M.

Thus the efficiency of the Gaver-Stehfest algorithm, measured by the ratio of the significant digits produced to the precision required, is

$$\begin{aligned} eff(\mathcal{G}) &\equiv \frac{\text{significant digits produced}}{\text{precision required}} \\ &\approx \frac{0.90M}{2.2M} \approx 0.4. \end{aligned} \tag{35}$$

Let $\lceil x \rceil$ be the least integer greater than or equal to x.

Gaver-Stehfest Algorithm Summary
If j significant digits are desired, then let M be the positive integer $\lceil 1.1j \rceil$. Given M, set the system precision at $\lceil 2.2M \rceil$. Given M and the system precision, calculate the weights ζ_k, $1 \leq k \leq 2M$, using (32). Then, given the transform $\hat{f}$ and the argument t, calculate the sum $f_g(t, M)$ in (31). □

5. The Euler Algorithm

The Euler algorithm is an implementation of the Fourier-series method, using Euler summation to accelerate convergence of the final infinite series; e.g., see Abate et al. (1999). Since the Fourier-series method can also be derived from the Bromwich inversion integral in (8), the derivation of the framework (2) from the Bromwich inversion integral in Section 2 shows that the Fourier-series method fits directly in the framework (2). Since Euler summation is a linear acceleration algorithm, the full algorithm Euler also fits in the framework (2).

We can also start from the final Euler algorithm itself: we obtain the desired Euler algorithm in the framework (2) by fixing the parameters in the Euler algorithm in our previous papers. Referring to the algorithm summary on p. 272 of Abate et al. (1999), we let old parameter vector (l, m, n, A) be assigned values $(1, M, M, 2\ln(10)M/3)$, where M is a positive integer. Here the original l (lower case L) has been replaced by 1 (one). We omit that roundoff-error-control parameter, because we are thinking of employing multi-precision software, as in Abate and Valko (2004). The value of A is taken from equation (40) on p. 271 of Abate et al. (1999).

With this parameter assignment, the Euler inversion formula to calculate $f(t)$ numerically for real-valued f is

$$f_e(t, M) = \frac{10^{M/3}}{t} \sum_{k=0}^{2M} \eta_k Re\left(\hat{f}\left(\frac{\beta_k}{t}\right)\right), \tag{36}$$

where

$$\beta_k = \frac{M\ln(10)}{3} + \pi i k, \qquad \eta_k \equiv (-1)^k \xi_k, \tag{37}$$

with $i = \sqrt{-1}$ and

$$\xi_0 = \frac{1}{2}, \qquad \xi_k = 1, \quad 1 \le k \le M, \qquad \xi_{2M} = \frac{1}{2^M},$$
$$\xi_{2M-k} = \xi_{2M-k+1} + 2^{-M}\binom{M}{k}, \quad 0 < k < M. \tag{38}$$

Inversion formula (36) is in the form (2) with $n = 2M$, $\omega_k = 10^{M/3}\eta_k$ and $\alpha_k = \beta_k$ for η_k and β_k in (37) and (38). Again we let the single parameter be M instead of n. As with the Gaver-Stehfest algorithm, about half of the $2M+1$ terms appearing in the sum in (36) are to accelerate convergence using Euler summation.

Like the Gaver-Stehfest algorithm, there is even spacing of the nodes β_k, but now they are evenly spaced on the *vertical* line $s = M\ln(10)/3t$ instead of on the real axis. As with the Gaver-Stehfest algorithm, the weights are real with

$$\sum_{k=0}^{n} \eta_k = 0 \quad \text{for all even } n \ge 2. \tag{39}$$

However, now the nodes are complex, so when f is real valued, we work with

$$Re\left\{\eta_k \hat{f}\left(\frac{\beta_k}{t}\right)\right\} = \eta_k Re\left\{\hat{f}\left(\frac{\beta_k}{t}\right)\right\}. \tag{40}$$

The Euler algorithm tends to be more efficient than the Gaver-Stehfest algorithm. Given M, the required system precision is only about M, but it produces about $0.6M$ significant digits for good transforms. Thus the efficiency of the Euler algorithm, again measured by the ratio of the significant digits produced to the precision required, is

$$eff(\mathscr{E}) \equiv \frac{\text{significant digits produced}}{\text{precision required}} \approx \frac{0.60M}{1.0M} \approx 0.6. \tag{41}$$

which is about 3/2 times that of the Gaver-Stehfest algorithm.

Euler Algorithm Summary
If j significant digits are desired, then let M be the positive integer $\lceil 1.7j \rceil$. Given M, set the system precision at M. Given M and the system precision, calculate the weights η_k and nodes β_k using (37) and (38). Then, given the transform $\hat{f}$ and the argument t, calculate the sum $f_e(t, M)$ in (36). □

6. The Talbot Algorithm

The Talbot algorithm also starts from the Bromwich integral (8), so it too is destined to fit in the framework (2). The Talbot algorithm is based on cleverly deforming the contour in the Bromwich inversion integral. We obtain the desired version of Talbot's method in the framework (2) by fixing parameters in the version of Talbot's method in Section 3 of Abate and Valko (2004). It should be noted that the version of Talbot's algorithm there is a considerable simplification of previous versions in the literature.

The Talbot inversion formula to calculate $f(t)$ numerically for real-valued f is

$$f_b(t, M) = \frac{2}{5t}\sum_{k=0}^{M-1} Re\left(\gamma_k \hat{f}\left(\frac{\delta_k}{t}\right)\right) \tag{42}$$

(subscript b for TalBot), where

$$\delta_0 = \frac{2M}{5}, \quad \delta_k = \frac{2k\pi}{5}(\cot(k\pi/M) + i), \quad 0 < k < M,$$
$$\gamma_0 = \frac{1}{2}e^{\delta_0}, \quad \gamma_k = [1 + i(k\pi/M)(1 + [\cot(k\pi/M)]^2) - i\cot(k\pi/M)]e^{\delta_k}, 0 < k < M, \tag{43}$$

where again $i \equiv \sqrt{-1}$.

The parameters in the framework (2) are $n = M$, $\alpha_k = \delta_k$, and $\omega_k \equiv (2/5)\gamma_k$ for δ_k and γ_k in (43). For this fixed-Talbot algorithm, both the weights and the nodes are complex. Now there is uneven spacing of the nodes. Now the weights do not sum to zero, but the sum is small. In particular,

$$Re\left\{\sum_{k=0}^{M-1} \gamma_k\right\} \approx 10^{-0.6M}. \tag{44}$$

As discussed in Abate and Valko (2004), for the implementation of the Talbot algorithm there may be complications with a few complex-function routines. Those difficulties involve calculating the required values of the Laplace transform. Without those difficulties, the Talbot algorithm tends to be about as efficient as the Euler algorithm. When the parameter is M, about $0.6M$ significant digits are produced for a good transform, while about M digits of precision are required. Thus the efficiency of the Talbot algorithm is

$$eff(\mathscr{T}) \equiv \frac{\text{significant digits produced}}{\text{precision required}} \approx \frac{0.60M}{1.0M} \approx 0.6, \tag{45}$$

just as for the Euler algorithm.

Talbot Algorithm Summary
If j significant digits are desired, then let M be the positive integer $\lceil 1.7j \rceil$. Given M, let the system precision also be M. Given M and the system precision, calculate the weights γ_k and nodes δ_k using (43). Then, given the transform $\hat{f}$ and the argument t, calculate the sum $f_b(t, M)$ in (42). □

7. Performance of the One-Dimensional Algorithms

In this section we briefly discuss the performance of the three one-dimensional algorithms. We have extensive experience with the Euler algorithm based on our previous work. For the Gaver and Talbot algorithms, we draw extensively upon Abate and Valko (2004); see Section 4 for supporting details. The story is quite complicated, so that reference to Abate and Valko (2004) is important for serious study.

We first summarize the analysis of efficiency of the three one-dimensional inversion routines specified above. Unfortunately, there is no systematic error analysis. Moreover, the observed performance depends on the transform. We describe the performance for good transforms, i.e., those with all singularities falling on the negative real axis and for which the function f is infinitely differentiable (class **F** in Abate and Valko 2004). For good transforms, the error is almost independent of t and the transform. For other transforms, that is not true.

Twelve examples of transforms in **F** are listed in Table 1 of Abate and Valko (2004). A summary of the algorithm efficiency described in the last three sections appears in Table 1. From our experience, and from Table 1, we conclude that the Euler and Talbot algorithms are more efficient than the Gaver-Stehfest algorithm. However, the Gaver-Stehfest algorithm has the advantage that only real numbers are used.

We next compare the three algorithms for a specific example, namely the transform

$$\hat{f}(s) = \frac{1}{\sqrt{s}+s}, \tag{46}$$

which has known inverse

$$f(t) = e^t erfc(\sqrt{t}), \quad t \geq 0, \tag{47}$$

where *erfc* is the complementary error function. The performance of the three algorithms as a function of the single parameter M is summarized in Table 2. The calculations were performed using the multi-precision language UBASIC on a 0.2 GHz CPU.

Since the transform in (46) is a good transform (in class **F**, Abate and Valko 2004), the error tends to be independent of t. However, the results in Table 2 are quite complicated. In particular, they are nonlinear. For example, as M goes from 50 to 100, $\mathcal{G}$ takes six times as long, whereas $\mathcal{E}$ takes four times as long. This has to do with computing costs at different levels of precision, even though the same number of terms are being computed. Now compare $\mathcal{E}$ and $\mathcal{T}$ at the same value of M: they both have the same number of terms, so we might expect them to take the same time; however, $\mathcal{T}$ has a complex multiplication for each term, whereas $\mathcal{E}$ is using only real multiplication.

Only a few significant digits are required for most engineering applications. The very high accuracy we are able to obtain with high system precision for this example, shown in Table 2, may be important when we consider high-dimensional multidimensional inversions, because then we may need greater accuracy in the inner loops in order to obtain required precision in the outer loops.

There may not need to be great concern about the computational effort required to compute the weights and nodes, because for multiple applications the weights and nodes can be computed in advance and stored for easy access (even for multiple values of M), but they might also be computed on the fly. Hence we have investigated the computational effort required to compute the weights and nodes for the different algorithms. Table 3 shows the time in milliseconds (with a 0.2 GHz CPU) required to compute the weights and nodes for the three methods $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{T}$ specified above.

Since the new version of the Talbot algorithm is relatively less tested, we present an additional example

Table 2 A Comparison of the Performance of the Three One-Dimensional Inversion Routines as a Function of M for the Transform in (46) with Inverse Function in (47)

	Significant digits			Execution time		
M	$\mathcal{G}$	$\mathcal{E}$	$\mathcal{T}$	$\mathcal{G}$	$\mathcal{E}$	$\mathcal{T}$
20	18	13	12	2	4	2
30	27	19	18	5	7	4
50	45	30	30	15	17	10
100	91	59	60	90	68	47

Note. The execution time is in milliseconds on a 0.2 GHz CPU.

Table 1 A Rough Analysis of Cost and Efficiency for the Three Specified One-Dimensional Inversion Routines

Inversion routine	No. of transform evaluations	System precision required decimal digits	Significant digits produced for good transform	Efficiency (SD/prec.)
Gaver-Stehfest	$2M$	$2.2M$	$0.90M$	0.4
Euler	$2M+1$	$1.0M$	$0.60M$	0.6
Talbot	M	$1.0M$	$0.60M$	0.6

Table 3 A Comparison of the Execution Times for the Weights and Nodes (in Milliseconds with a 0.2 GHz CPU) for the Three Specified One-Dimensional Inversion Routines

M	Euler	Talbot	Gaver-Stehfest
25	≈ 0	6	12
50	1	25	100
100	4	160	1,200
200	20	1,600	19,000

Table 4 Significant Digits Produced by the One-Dimensional Talbot Algorithm as a Function of t and M for the Example in (48) and (49)

	M				
t	10	20	40	100	200
10^{-8}	1	10	23	59	119
10^{-6}	6	12	23	59	119
10^{-2}	6	12	23	59	119
10^{-1}	6	12	23	59	119
10^{0}	6	11	23	59	119
10^{1}	5	11	22	58	118
10^{2}	5	10	21	57	118
10^{4}	3	9	20	55	114
10^{6}	2	8	19	54	113
10^{8}	1	7	18	53	112

examining its performance. We apply the Talbot algorithm to the transform

$$\hat{f}(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}, \tag{48}$$

which has known inverse

$$f(t) = \frac{1-e^{-t}}{\sqrt{4\pi t^3}}, \quad t \geq 0. \tag{49}$$

Numerical results for a wide range of t and M are given in Table 4. Again we see excellent performance.

8. Candidate Two-Dimensional Algorithms

In this section we combine the three one-dimensional inversion algorithms $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{T}$ specified in Sections 4, 5, and 6 to construct nine two-dimensional inversion algorithms. In each case, the two-dimensional inversion routine can be expressed in the common form (26), but in some cases further simplification can be obtained by exploiting properties of complex conjugates. We display the possible algorithms below in Table 5.

In Table 5 we use the notation in Sections 4–6. In particular, we use the single parameter M instead of n. The outer loop has parameter M and the inner loop has parameter cM, so there remains only the single parameter M.

We also use the special notation for the nodes and weights introduced in Sections 4–6. Thus, ζ_k are the Gaver weights in (32), η_k and β_k are the Euler weights and nodes in (37) and (38), and γ_k and δ_k are the Talbot weights and nodes in (43).

It is understood that the weights and nodes in the outer loop depend on the parameter M, while the weights and nodes in the inner loop depend on the parameter cM, where c is specified at the left, based on our experience with a collection of good transforms (the class **F** from Abate and Valko 2004).

Table 5 The Nine Two-Dimensional Inversion Routines Based on the Three One-Dimensional Routines: Gaver-Stehfest $\mathcal{G}$, Euler $\mathcal{E}$, and Talbot $\mathcal{T}$

$\mathcal{T}(M)\mathcal{G}(M)$:	$\frac{2\ln(2)}{5t_1t_2}\sum_{k_1=0}^{M-1} Re\left\{\gamma_{k_1}\sum_{k_2=1}^{2M}\zeta_{k_2}\tilde{f}\left(\frac{\delta_{k_1}}{t_1}, \frac{k_2\ln(2)}{t_2}\right)\right\}$
$\mathcal{T}(M)\mathcal{T}(M)$:	$\frac{2}{25t_1t_2}\sum_{k_1=0}^{M-1} Re\left\{\gamma_{k_1}\sum_{k_2=0}^{M-1}\left[\gamma_{k_2}\tilde{f}\left(\frac{\delta_{k_1}}{t_1}, \frac{\delta_{k_2}}{t_2}\right)+\bar{\gamma}_{k_2}\tilde{f}\left(\frac{\delta_{k_1}}{t_1}, \frac{\bar{\delta}_{k_2}}{t_2}\right)\right]\right\}$,
$\mathcal{E}(M)\mathcal{G}(M)$:	$\frac{10^{M/3}\ln(2)}{t_1t_2}\sum_{k_1=0}^{2M}\eta_{k_1}\sum_{k_2=1}^{2M}\zeta_{k_2} Re\left\{\tilde{f}\left(\frac{\beta_{k_1}}{t_1}, \frac{k_2\ln(2)}{t_2}\right)\right\}$
$\mathcal{E}(M)\mathcal{T}(M)$:	$\frac{10^{M/3}}{5t_1t_2}\sum_{k_1=0}^{2M}\eta_{k_1}\sum_{k_2=0}^{M-1} Re\left\{\gamma_{k_2}\tilde{f}\left(\frac{\beta_{k_1}}{t_1}, \frac{\delta_{k_2}}{t_2}\right)+\bar{\gamma}_{k_2}\tilde{f}\left(\frac{\beta_{k_1}}{t_1}, \frac{\bar{\delta}_{k_2}}{t_2}\right)\right\}$
$\mathcal{T}(M)\mathcal{E}(M)$:	$\frac{10^{M/3}}{5t_1t_2}\sum_{k_1=0}^{M-1} Re\left\{\gamma_{k_1}\sum_{k_2=0}^{2M}\eta_{k_2}\left[\tilde{f}\left(\frac{\delta_{k_1}}{t_1}, \frac{\beta_{k_2}}{t_2}\right)+\tilde{f}\left(\frac{\delta_{k_1}}{t_1}, \frac{\bar{\beta}_{k_2}}{t_2}\right)\right]\right\}$
$\mathcal{G}(M)\mathcal{T}(3M)$:	$\frac{2\ln(2)}{5t_1t_2}\sum_{k_1=1}^{2M}\zeta_{k_1}\sum_{k_2=0}^{3M-1} Re\left\{\gamma'_{k_2}\tilde{f}\left(\frac{k_1\ln(2)}{t_1}, \frac{\delta'_{k_2}}{t_2}\right)\right\}$
$\mathcal{G}(M)\mathcal{G}(2M)$:	$\frac{(\ln(2))^2}{t_1t_2}\sum_{k_1=1}^{2M}\zeta_{k_1}\sum_{k_2=1}^{4M}\zeta'_{k_2}\tilde{f}\left(\frac{k_1\ln(2)}{t_1}, \frac{k_2\ln(2)}{t_2}\right)$
$\mathcal{E}(M)\mathcal{E}(M)$:	$\frac{10^{2M/3}}{2t_1t_2}\sum_{k_1=0}^{2M}\eta_{k_1}\sum_{k_2=0}^{2M}\eta_{k_2} Re\left\{\tilde{f}\left(\frac{\beta_{k_1}}{t_1}, \frac{\beta_{k_2}}{t_2}\right)+\tilde{f}\left(\frac{\beta_{k_1}}{t_1}, \frac{\bar{\beta}_{k_2}}{t_2}\right)\right\}$
$\mathcal{G}(M)\mathcal{E}(3M)$:	$\frac{10^{M}\ln(2)}{t_1t_2}\sum_{k_1=1}^{2M}\zeta_{k_1}\sum_{k_2=0}^{6M}\eta'_{k_2} Re\left\{\tilde{f}\left(\frac{k_1\ln(2)}{t_1}, \frac{\beta'_{k_2}}{t_2}\right)\right\}$

Notes. The value of c has been specified based on experience with good transforms. The routines are ordered according to the number of transform evaluations required. A prime appears on the nodes and weights in the inner loop in the three cases in which $c \neq 1$, indicating that these parameters depend on cM, not M.

When $c \neq 1$, which occurs in only three cases, the corresponding parameters in the inner loop are designated by the addition of a $'$ (prime). For example, with the all-Gaver routine $\mathscr{G}(M)\mathscr{G}(2M)$, ζ_k depends on M, while ζ_k' depends on $2M$. Similarly, in $\mathscr{G}(M)\mathscr{E}(3M)$, ζ_k again depends on M, but η_k' and β_k' depend on $3M$. However, much to our surprise, in six of the nine two-dimensional algorithms we found that it suffices to let $c = 1$, making the number of computations in the inner loop no greater than in the outer loop.

The four algorithms combining $\mathscr{E}$ and $\mathscr{T}$ exploit relations involving complex conjugates, so there is an extra term in each sum, with the entire quantity divided by 2.

9. Performance of the Two-Dimensional Algorithms

In this section we describe the performance of the two-dimensional inversion routines. Paralleling Table 1, in Table 6 we analyze the efficiency of the nine routines specified in Section 8. We order the algorithms according to the approximate number of transform evaluations required, because that seems to be the decisive factor. The algorithms at the top, requiring fewer transform evaluations, tend to be most efficient.

Now we consider two concrete examples in detail, both taken from Ditken and Prudnikov (1962). Both are good transforms (in the class **F** in Abate and Valko 2004). (We have examined about ten examples from Ditken and Prudnikov (1962); these two are representative. We also have conducted experiments on queueing transforms. Applications to queues will be discussed in Abate and Whitt 2005.) The first example here is the two-dimensional Laplace transform

$$\hat{f}(s_1, s_2) = \frac{1}{s_1 s_2 \sqrt{s_1}}\left(1 - \frac{s_1}{s_1 + s_2 + \sqrt{2 s_1 s_2}}\right) \tag{50}$$

Table 6 A Rough Analysis of Cost and Efficiency for the Nine Two-Dimensional Inversion Routines

Inversion routine	Approx. no. of transform evaluations	Sig. digits produced for good transform	Efficiency (sd's/evals.)
$\mathcal{T}(M)\mathcal{G}(M)$	$2M^2$	$0.6M$	$0.300/M$
$\mathcal{T}(M)\mathcal{T}(M)$	$*2M^2$	$0.6M$	$0.300/M$
$\mathcal{E}(M)\mathcal{G}(M)$	$4M^2$	$0.6M$	$0.150/M$
$\mathcal{E}(M)\mathcal{T}(M)$	$*4M^2$	$0.6M$	$0.150/M$
$\mathcal{T}(M)\mathcal{E}(M)$	$*4M^2$	$0.6M$	$0.150/M$
$\mathcal{G}(M)\mathcal{T}(3M)$	$6M^2$	$0.8M$	$0.133/M$
$\mathcal{G}(M)\mathcal{G}(2M)$	$8M^2$	$0.6M$	$0.075/M$
$\mathcal{E}(M)\mathcal{E}(M)$	$*8M^2$	$0.6M$	$0.075/M$
$\mathcal{G}(M)\mathcal{E}(3M)$	$12M^2$	$0.8M$	$0.067/M$

Note. The routines have been ordered according to the approximate number of required transform evaluations. The asterisks indicate values that have been multiplied by two because there is a second evaluation involving the complex conjugate.

Table 7 A Comparison of the Performance of the Nine Two-Dimensional Inversion Routines as a Function of M for the Transform in (50) with Inverse Function in (51)

	Significant digits M				Execution time M			
Routine	10	20	30	50	10	20	30	50
$\mathcal{T}(M)\mathcal{G}(M)$	6	12	18	30	0.05	0.20	0.6	3
$\mathcal{T}(M)\mathcal{T}(M)$	6	12	18	30	0.04	0.15	0.4	2
$\mathcal{E}(M)\mathcal{G}(M)$	6	12	17	27	0.05	0.41	1.3	7
$\mathcal{E}(M)\mathcal{T}(M)$	7	13	19	30	0.07	0.29	0.8	4
$\mathcal{T}(M)\mathcal{E}(M)$	7	13	19	30	0.07	0.28	0.8	4
$\mathcal{G}(M)\mathcal{T}(3M)$	8	16	24	40	0.13	1.00	3.6	21
$\mathcal{G}(M)\mathcal{G}(2M)$	9	13	17	28	0.19	0.80	2.8	19
$\mathcal{E}(M)\mathcal{E}(M)$	6	14	18	30	0.14	0.71	2.0	9
$\mathcal{G}(M)\mathcal{E}(3M)$	8	16	24	39	0.27	1.90	7.0	40

Note. The execution time is in seconds on an 0.2 GHz CPU.

with inverse function

$$f(t_1, t_2) = \frac{2}{\sqrt{\pi}}\left[\sqrt{t_1^2 + t_2^2} - t_2\right]^{1/2}. \tag{51}$$

Numerical results for this first example are given in Table 7. We consider all nine methods in Table 5.

The execution time reflects the number of transform evaluations, but it also reflects the required precision setting. For example, $\mathscr{T}\mathscr{G}$ and $\mathscr{T}\mathscr{T}$ require about the same number of transform evaluations, $2M^2$, but the execution time for $\mathscr{T}\mathscr{G}$ is higher because the precision setting is higher. The precision setting is $2.2M$ for $\mathscr{T}\mathscr{G}$, but only M for $\mathscr{T}\mathscr{T}$. In contrast, $\mathscr{E}\mathscr{T}$ and $\mathscr{T}\mathscr{E}$ have both the same number of evaluations, $4M^2$, and the same precision setting, so that the overall execution time is consistently about the same. On the other hand, $\mathscr{T}\mathscr{T}$ and $\mathscr{E}\mathscr{T}$ have the same precision setting, but $\mathscr{T}\mathscr{T}$ evaluates half as many terms as $\mathscr{E}\mathscr{T}$; accordingly, the execution time for $\mathscr{T}\mathscr{T}$ is about half that for $\mathscr{E}\mathscr{T}$.

Our second example is the two-dimensional Laplace transform

$$\hat{f}(s_1, s_2) = \frac{exp\left(1/\sqrt{s_2(s_1+1)}\right)}{s_2\sqrt{s_1+1}} \tag{52}$$

with inverse function

$$f(t_1, t_2) = \frac{e^{-t_1}}{\sqrt{\pi t_1}} I_0\left(\left[8\sqrt{t_1 t_2}\right]^{1/2}\right), \tag{53}$$

where I_0 is the modified Bessel function; see p. 66 of Magnus et al. (1966). Numerical results for this second example are given in Table 8.

From Tables 7 and 8, we conclude that the algorithms in the upper part of the tables, requiring fewer transform evaluations, are most efficient. Rough generalizations would be: (1) it is not so good to use the Gaver-Stehfest routine in the outer loop and (2) the Talbot routine tends to be efficient wherever it is used.

Table 8 A Comparison of the Performance of Nine Two-Dimensional Inversion Routines as a Function of M for the Transform in (52) with Inverse Function in (53)

	Significant digits M				Execution time M			
Routine	10	20	30	50	10	20	30	50
$\mathcal{T}(M)\mathcal{G}(M)$	7	13	19	31	0.08	0.31	1.1	7
$\mathcal{T}(M)\mathcal{T}(M)$	6	12	18	30	0.05	0.19	0.6	3
$\mathcal{E}(M)\mathcal{G}(M)$	6	13	19	28	0.08	0.63	2.2	13
$\mathcal{E}(M)\mathcal{T}(M)$	7	12	18	30	0.10	0.37	1.1	5
$\mathcal{T}(M)\mathcal{E}(M)$	7	13	19	30	0.09	0.35	1.1	5
$\mathcal{G}(M)\mathcal{T}(3M)$	9	18	28	46	0.22	1.86	7.2	48
$\mathcal{G}(M)\mathcal{G}(2M)$	9	13	17	26	0.27	2.04	8.4	72
$\mathcal{E}(M)\mathcal{E}(M)$	6	13	18	30	0.24	1.12	3.6	17
$\mathcal{G}(M)\mathcal{E}(3M)$	9	17	22	37	0.35	2.73	11.2	81

Note. The execution time is in seconds on a 0.2 GHz CPU.

However, there is a difficulty with the Talbot algorithm, which does not have to do with efficiency. It is caused by the fact that the Talbot routine involves both complex weights and complex nodes. We found that we sometimes encountered difficulties evaluating transform values when the Talbot algorithm was used in the inner loop. That is because of difficulties in the software (e.g., *Mathematica*) properly evaluating complex-valued functions of two complex variables, e.g., involving the square-root function.

Thus, at this time, we would not recommend using the Talbot algorithm in the inner loop, or at least being aware that there might be problems involving computation with complex numbers and functions. That leaves $\mathcal{TG}$, $\mathcal{EG}$, and $\mathcal{TE}$ as the top-ranked two-dimensional routines. *None of these uses the same one-dimensional routine in both loops*. However, all nine methods are effective. Overall, our study indicates the unified framework has promise.

10. Conclusions

In this paper we have proposed the general unified framework in (2) for numerically inverting Laplace transforms. We pointed out that the flexible framework makes it possible to perform optimization in order to select specific inversion routines for special classes of functions, but it remains to pursue that approach. In Sections 4, 5, and 6 we showed that the Gaver-Stehfest method, $\mathcal{G}$, the Fourier-series method with Euler summation, $\mathcal{E}$, and a version of Talbot's method, $\mathcal{T}$, all fit into this common framework.

In Section 3 we observed that the unified framework extends directly to yield a corresponding framework for numerically inverting multidimensional Laplace transforms. In Section 8 we observed that the three specific one-dimensional algorithms we introduced in the framework (2) produce nine candidate two-dimensional algorithms, all in the two-dimensional framework (5).

In Sections 7 and 9 we reported results on the performance of the algorithms. In Section 7 we concluded that the Euler and Talbot one-dimensional algorithms are about equally efficient, as measured by the ratio of the number of significant digits produced to the number of digits of machine precision required, while these two are about 1.5 times as efficient as the Gaver-Stehfest one-dimensional algorithm. However, the Gaver-Stehfest algorithm has the advantage of working with only real numbers.

Investigations of the two-dimensional algorithms in Section 9 indicate that all nine combinations of the three one-dimensional routines can be effective two-dimensional algorithms. However, we encountered difficulties calculating values of two-dimensional transforms when the Talbot method $\mathcal{T}$ is used in the inner loop, because of problems evaluating transforms for conjugate s. Initial results indicate that the best two-dimensional algorithms are the combinations $\mathcal{TG}$, $\mathcal{EG}$ and $\mathcal{TE}$. Evidently it can be advantageous to use different one-dimensional routines in the inner and outer loops.

Much remains to be done. First, as is often the case with numerical transform inversion (and many other numerical methods), a systematic error analysis is lacking. The bounds for the Fourier-series method with Euler summation in Abate and Whitt (1995), O'Cinneide (1997), and Sakurai (2004) are the exception. There is a need for much more testing. It remains to understand better how the algorithms perform on different classes of transforms. More work is needed, extending Abate and Valko (2004), to understand how the algorithm performance depends on the structure of the transform. It remains to see what combinations of the three one-dimensional routines will be especially effective for high-dimensional inversion problems.

It also remains to consider other kinds of transforms. We have found that essentially the same unified framework applies to generating functions; we plan to elaborate on that elsewhere. Thus, just as in Choudhury et al. (1994a), the unified framework will apply to multidimensional transforms, where each dimension may involve either a generating function or a Laplace transform.

References

Abate, J., P. P. Valko. 2004. Multi-precision Laplace inversion. *Internat. J. Numer. Meth. 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. 1997. Asymptotics for M/G/1 low-priority waiting-time tail probabilities. *Queueing Systems* **25** 173–223.

Abate, J., G. L. Choudhury, W. Whitt. 1998. Numerical inversion of multidimensional Laplace transforms by the Laguerre method. *Performance Eval.* **31** 229–243.

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, Boston, MA, 257–323.

Audis, E., W. Whitt. 2006. Power algorithms for inverting Laplace transforms. Columbia University, New York. Available at http://www.columbia.edu/~ ww2040/recent.html.

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

Bingham, N. H., C. M. Goldie, J. L. Teugels. 1987. *Regular Variation*. *Encyclopedia of Mathematics and Its Applications*, Vol. 27. Cambridge University Press, Cambridge, UK.

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

Choudhury, G. L., K. K. Leung, W. Whitt. 1995a. Calculating normalization constants of closed queueing networks by numerically inverting their generating functions. *J. ACM* **42** 935–970.

Choudhury, G. L., K. K. Leung, W. Whitt. 1995b. An inversion algorithm to compute blocking probabilities in loss networks with state-dependent rates. *IEEE/ACM Trans. Networking* **3** 585–601.

Choudhury, G. L., D. M. Lucantoni, W. Whitt. 1994a. Multidimensional transform inversion with applications to the transient $M/G/1$ queue. *Ann. Appl. Probab.* **4** 719–740.

Choudhury, G. L., D. M. Lucantoni, W. Whitt. 1994b. The transient BMAP/G/1 queue. *Stochastic Models* **10** 145–182.

Choudhury, G. L., D. M. Lucantoni, W. Whitt. 1997. Numerical solution of $M_t/G_t/1$ queues. *Oper. Res.* **45** 451–463.

Davies, B. 2002. *Integral Transforms and Their Applications*, 3rd ed. Springer, New York.

Davies, B., B. Martin. 1979. Numerical inversion of the Laplace transform: A survey and comparison of methods. *J. Computational Phys.* **33** 1–32.

Davis, P. J., P. Rabinowitz. 1984. *Methods of Numerical Integration*. Academic Press, New York.

Ditken, V. A., A. P. Prudnikov. 1962. *Operational Calculus in Two Variables and Applications*. Pergamon, New York. (English translation of 1958 Russian edition.)

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

Dubner, H., J. Abate. 1968. Numerical inversion of Laplace transforms by relating them to the finite Fourier cosine transform. *J. ACM* **15** 115–123.

Duffy, D. G. 1993. On the numerical inversion of Laplace transforms: Comparison of three new methods on characteristic problems from applications. *ACM Trans. Math. Software* **19** 333–359.

Evans, G. A. 1993. Numerical inversion of Laplace transforms using optimal contours in the complex plane. *Internat. J. Comput. Math.* **49** 93–105.

Evans, G. A., K. C. Chung. 2000. Laplace transform inversion using optimal contours in the complex plane. *Internat. J. Comput. Math.* **73** 531–543.

Feller, W. 1971. *An Introduction to Probability Theory and Its Applications*, 2nd ed. Wiley, New York.

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, Berlin, Germany.

Hwang, C., M. J. Lu. 1999. Numerical inversion of 2-D Laplace transforms by fast Hartley transform computations. *J. Franklin Inst.* **336** 955–972.

Kao, E. P. C. 1997. *An Introduction to Stochastic Processes*. Duxbury Press, New York.

Magnus, W., F. Oberhettinger, R. P. Soni. 1966. *Formulas and Theorems for the Special Functions of Mathematical Physics*. Springer, New York.

Murli, A., M. Rizzardi. 1990. Algorithm 682. Talbot's method for the Laplace inversion problem. *ACM Trans. Math. Software* **16** 158–168.

Narayanan, G. V., D. E. Beskos. 1982. Numerical operational methods for time-dependent linear problems. *Internat. J. Numer. Methods Engrg.* **18** 1829–1854.

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

Piessens, R. 1969. New quadrature formulas for the numerical inversion of the Laplace transform. *BIT* **9** 351–361.

Piessens, R. 1971. Gaussian quadrature formulas for the numerical inversion of the Laplace transform. *J. Engrg. Math.* **5** 1–9.

Sakurai, T. 2004. Numerical inversion of the Laplace transform of functions with discontinuities. *Adv. Appl. Probab.* **36** 616–642.

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. Appl.* **48** 629–636.

Valko, P. P., B. L. Vojta. 2001. *The List*. Bibliography of papers on numerical transform inversion and its application. http://pumpjack.tamu.edu/~valko.

Vlach, J. 1969. Numerical method for transient responses. *J. Franklin Inst.* **288** 91–113.

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

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.