INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **60**:979–993 (DOI: 10.1002/nme.995)

Multi-precision Laplace transform inversion

J. Abate[1] and P. P. Valkó[2,*,†]

[1]*900 Hammond Road, Ridgewood, NJ 07450-2908, U.S.A.*
[2]*Department of Petroleum Engineering, Texas A&M University, U.S.A.*

SUMMARY

For the numerical inversion of Laplace transforms we suggest to use multi-precision computing with the level of precision determined by the algorithm. We present two such procedures. The Gaver–Wynn–Rho (GWR) algorithm is based on a special sequence acceleration of the Gaver functionals and requires the evaluation of the transform only on the real line. The fixed Talbot (FT) method is based on the deformation of the contour of the Bromwich inversion integral and requires complex arithmetic. Both GWR and FT have only one free parameter: M, which is the number of terms in the summation. Both algorithms provide increasing accuracy as M increases and can be realized in a few lines using current Computer Algebra Systems. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: Laplace transform; numerical inversion; multi-precision

1. INTRODUCTION

Our purpose in this paper is to present two algorithms for the numerical inversion of Laplace transforms using multi-precision computing. The algorithms are variants of two methods widely known as Gaver–Stehfest and Talbot. Although our enhancements may seem modest, they significantly impact the performance of the methods. The key idea is to use multi-precision computing and have the algorithm determine the level of precision based on the user's input of desired accuracy. Further, our algorithms are simple and concise.

The problem of numerically inverting the Laplace transform is to determine approximations for $f(t)$ when numerical values of the transform function

$$\hat{f}(s) = \int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t \tag{1}$$

can be computed. For our purposes, we add the proviso that values of the transform can be computed to any desired precision as a function of the complex variable 's'. This is easily

*Correspondence to: P. P. Valkó, 3116 TAMU, College Station, TX, 77843, U.S.A.
†E-mail: p-valko@tamu.edu

Received 3 February 2003
Revised 30 May 2003
Accepted 25 August 2003

accomplished in a multi-precision computational environment such as *Mathematica*, *Maple*, *UBASIC*, etc.

There are over 100 algorithms available for the numerical inversion of Laplace transforms. Three important comparative studies of methods have been published. Davies and Martin [1] surveyed about 20 different methods and selected 14 specific algorithms to analyse and test. It is a very nice paper. Narayanan and Beskos [2] systematically discussed and tested eight algorithms. The above two studies considered methods that were developed prior to 1980. Duffy [3] tested three software packages based on methods developed after 1980. In addition to these comparative studies, an enormous number of engineering application papers have been written each investigating the merits of a particular procedure. A bibliography of several hundred such papers is available on the WEB [4].

Those algorithms that have passed the test of time fall into four categories. The four groupings of algorithms are according to the basic approach of the method as follows:

(i) Fourier series expansion
(ii) Laguerre function expansion
(iii) Combination of Gaver functionals
(iv) Deform the Bromwich contour

Over the years, there have been about 40 algorithms developed which are based on the Fourier series method. This is so called because it involves approximating the inversion integral with an infinite Fourier series. The first paper to consider such a numerical procedure appeared in 1935 by Koizumi [5]. Some of the more popular computing procedures are: Dubner–Abate (1968), Veillon (1974), Durbin (1974), Crump (1976), Hosono (1981), DeHoog–Knight–Stokes (1982), Honig–Hirdes (1984), Piessens–Huysmans (1984). For a discussion of these procedures and their references see the 1992 survey paper of Abate and Whitt [6]. Since then, several new Fourier series algorithms have been developed. Of notable interest are: D'Amore *et al*. [7] and Sakurai [8]. This last paper seems to have an effective method for handling transforms of functions with discontinuities.

The next most popular approach to numerical inversion is based on the Laguerre function expansion of $f(t)$ in (1). The first paper to consider such a numerical procedure was evidently Ward [9] in 1954. Since then about 15 algorithms have been developed based on the Laguerre approach. Some examples are: Chen (1966), Weeks (1966), Piessens–Branders (1971), Weber (1981), Lyness–Giunta (1986), Garbow–Giunta–Lyness–Murli (1988). For a discussion of these algorithms and their references see the 1996 survey paper of Abate *et al*. [10]. Since then an important contribution to the Laguerre method was developed by Weideman [11].

Another very good approach to numerical Laplace transform inversion is based on the sequence of functionals developed by Gaver [12] in 1966. We shall discuss this method in Section 2.

Finally, one of the best approaches to computing the inverse is to deform the standard contour in the Bromwich inversion integral. The seminal paper on this approach was published in 1979 by Talbot [13]. We shall discuss this method in Section 3.

A nice review of the above four methods is given in Chapter 19 of Reference [14]. In fact, we recommend reading that chapter because it gives a flavour of numerical inversion considerations in a computing environment with fixed machine precision. In the traditional development of inversion methods, most of the effort was directed at controlling round-off errors. This is because

the process is numerically unstable in a fixed-precision, computing environment. Our approach is to combat this problem with brute computational force, namely, the application of multi-precision. Heretofore, inversion algorithms did not have any means to control the accuracy of the program output. The problem is that as the user tries to increase the accuracy, there comes a point where round-off propagation causes the error to increase dramatically. That is, the procedure is numerically unstable when using fixed machine precision. To overcome this problem, one must have the capability to vary the machine precision at will. This can be achieved by using multi-precision computing with the level of precision determined by the algorithm.

Our approach results in a simple procedure. To demonstrate the simplicity of the algorithm called fixed-Talbot (FT), we display here a prototype implementation in *Mathematica*, for the convenience of the reader. All the FT calculations of this work were done using the shown program.

```
FT[F_, t_, M_] :=
   Module[{np, r, S, theta, sigma},
   np = Max[M, $MachinePrecision];
   r = SetPrecision[2M/(5t), np];
   S = r theta (Cot[theta] + I);
   sigma = theta + (theta Cot[theta] - 1) Cot[theta];
  (r/M) Plus @@ Append[ Table[ Re[ Exp[t S](1 + I sigma) F[S] ],
        {theta, Pi/M, (M-1)Pi/M, Pi/M} ],
           (1/2) Exp[r t] F[r] ]
   ]
```

Note the program has 10 lines! The user provides the transform function ($F[\ ]$); the value of t at which the inverse is desired; and the value of the parameter M, which is the number of terms in the summation. Hence, the algorithm has only one free parameter M, and the accuracy of the result improves as M increases.

In Section 4, we test the effectiveness of the GWR and FT algorithms. We have selected 27 test transforms which are listed in Tables I, VI, and IX. These test transforms fall into two classes which are labelled $\mathbb{F}$ and $\mathbb{G}$. These classes are defined in Section 4. Among the test transforms are six transforms that arise in engineering applications.

2. THE GAVER–WYNN–RHO ALGORITHM

The inverse problem of (1) is given the transform $\hat{f}(s)$, then determine the time function $f(t)$. An analytic solution to the inverse problem is provided by the celebrated Post-Widder formula

$$\phi_k(t) = \frac{(-1)^k}{k!}\left(\frac{k}{t}\right)^{k+1} \hat{f}^{(k)}\left(\frac{k}{t}\right) \tag{2}$$

then $\phi_k(t) \to f(t)$ as $k \to \infty$.

In the 1960s it was difficult to numerically compute high-order derivatives. Therefore, Gaver [12] presented the discrete analog of (2),

$$f_k(t) = \frac{(-1)^k \alpha k}{t} \binom{2k}{k} \Delta^k \hat{f}(k\alpha/t) \tag{3}$$

where $\alpha = \log(2)$ and Δ is the difference operator, $\Delta \hat{f}(nx) = \hat{f}((n+1)x) - \hat{f}(nx)$.

By expanding the difference operator, (3) can be written as

$$f_k(t) = \frac{\alpha k}{t} \binom{2k}{k} \sum_{j=0}^{k} (-1)^j \binom{k}{j} \hat{f}((k+j)\alpha/t) \tag{4}$$

The Gaver functionals can also be computed by a recursive algorithm, as follows:

$$\begin{aligned} G_0^{(n)} &= \tfrac{n\alpha}{t} \hat{f}(n\alpha/t), \quad 1 \leqslant n \leqslant 2M \\ G_k^{(n)} &= (1 + \tfrac{n}{k}) G_{k-1}^{(n)} - (\tfrac{n}{k}) G_{k-1}^{(n+1)}, \quad k \geqslant 1, \ n \geqslant k \\ f_k(t) &= G_k^{(k)} \end{aligned} \tag{5}$$

Unfortunately, the Gaver functionals provide a very poor approximation because $|f(t) - f_k(t)| \sim c/k$ as $k \to \infty$. For example, $f_{1000}(t)$ may yield an estimate to $f(t)$ with only two or three digits of accuracy. To achieve a good approximation, a convergence acceleration algorithm is required for the sequence $f_k(t)$. A good candidate is Saltzer summation, see pp. 35–38 of Reference [15]. In fact, this is the scheme proposed by Stehfest [16]. The Gaver–Stehfest algorithm is widely known.

There are other sequence accelerators that can be used instead; for a study of such alternatives see Valkó and Abate [17]. In that study, we test six summation methods and found that the best acceleration scheme for the Gaver functionals is the Wynn rho algorithm which is given by the recursive algorithm

$$\begin{aligned} \rho_{-1}^{(n)} &= 0, \quad \rho_0^{(n)} = f_n(t), \quad n \geqslant 0 \\ \rho_k^{(n)} &= \rho_{k-2}^{(n+1)} + \frac{k}{\rho_{k-1}^{(n+1)} - \rho_{k-1}^{(n)}}, \quad k \geqslant 1 \end{aligned} \tag{6}$$

see p. 168 of Reference [15]. Then the approximant to $f(t)$ is obtained as

$$f(t, M) = \rho_M^{(0)} \tag{7}$$

Note well, the integer M in (7) must be even.

However, we are not done with the GWR approximation. It is known that the computation of (4)–(6) are prone to round-off error propagation and hence are unstable. In other words, given a fixed precision, then as M increases the accuracy of the approximant $f(t, M)$ also increases but only to a point, thereafter the accuracy quickly decreases. To overcome this problem, the computing precision must be made larger as M increases. This can be achieved using a multi-precision computational environment. Hence, to complete the algorithm specification, we need the computational precision requirement,

$$\text{number of precision decimal digits} = (2.1)M \tag{8}$$

The GWR algorithm may therefore be summarized as follows:

> GWR Algorithm. Specify the transform $\hat{f}(s)$ and furnish values for t and M, where M is an even integer. First set the precision to $(2.1)M$, then compute the functionals, $f_1(t), f_2(t), \ldots, f_M(t)$ using either (4) or (5). Finally, compute the approximant $f(t, M)$ from (6) and (7).

A prototype implementation of this algorithm is given as a *Mathematica* add-on package posted in the MathSource library on the WEB [18].

In Section 4, we examine the performance of the GWR algorithm. For a large class of transforms, we find the relative error estimate

$$\left| \frac{f(t) - f(t, M)}{f(t)} \right| \approx 10^{-0.8\,M} \tag{9}$$

that is, the number of significant digits in the approximant $f(t, M)$ is about equal to M.

We conclude this section with computational considerations concerning the Gaver functionals. Note in (4) and (5) the computation of $f_k(t)$ involves real arithmetic. However, we must remember that the Laplace transform is inherently a function of the complex variable s. For example, consider the transform $\hat{f}(s) = 1/\sqrt{s-1}$ which has the inverse $f(t) = \exp(t)/\sqrt{\pi t}$. Note in (4) and (5) the transform is calculated at values of $s = k\log(2)/t$ for $k = 1, 2, \ldots$. Hence, the sampling interval may contain the singular point $s = 1$. This situation should be avoided (e.g. see Table VII). Therefore, the prudent procedure is to apply the transformation rule $s = s + 1$ to $\hat{f}(s)$ that is invert $1/\sqrt{s}$ and multiply the result by $\exp(t)$.

Although the test transform F09 in Table I seems to have a singularity at $s = 2$, analysis reveals that it is indeed a real function for all $s > 0$.

3. THE FIXED TALBOT ALGORITHM

In 1979 Talbot [13] pioneered the approach, to numerical Laplace inversion, of deforming the standard contour in the Bromwich integral

$$f(t) = \frac{1}{2\pi \mathrm{i}} \int_B \exp(ts)\hat{f}(s)\,\mathrm{d}s \tag{10}$$

There have been only a few such algorithms developed along these line, see References [19–21]. It seems surprising how little attention this approach has received in the literature relative to other methods.

In (10) the contour B, is a vertical line defined by $s = r + \mathrm{i}y$, where $-\infty < y < +\infty$ and r has a fixed value chosen such that all the singularities of the transform are to the left of it. The convergence of the integral (10) would be greatly improved if s could take on values with a large, negative, real component. We can deform the contour B into any open path that wraps around the negative real axis provided no singularity of $\hat{f}(s)$ is crossed in the deformation of B. In other words, the transform is analytic in the region of the complex plain to the right of the path B. Therefore, by Cauchy's theorem the deformed contour is valid. Talbot's brilliant contribution is the carefully chosen path of the form

$$s(\theta) = r\theta(\cot\theta + \mathrm{i}), \quad -\pi < \theta < +\pi \tag{11}$$

Table I. Some test transforms in class $\mathbb{F}$ and their inverses.

ID	$\hat{f}(s)$	$f(t)$
F01	$\dfrac{1}{\sqrt{s}+\sqrt{s+1}}$	$\dfrac{1-\mathrm{e}^{-t}}{2\sqrt{\pi t^3}}$
F02	$\dfrac{1}{\sqrt{s}(1+\sqrt{s})}$	$e^t \operatorname{erfc}(\sqrt{t})$
F03	$\dfrac{s-\sqrt{s^2-1}}{\sqrt{s^2-1}}$	$I_1(t)$
F04	$\exp(-2\sqrt{s})$	$\dfrac{\mathrm{e}^{-1/t}}{\sqrt{\pi t^3}}$
F05	$\dfrac{\exp(-\dfrac{1}{4s})}{\sqrt{s^3}}$	$\dfrac{2\sin(\sqrt{t})}{\sqrt{\pi}}$
F06	$\dfrac{-\log(s)}{s}$	$\log(t)+\gamma$
F07	$\log(1+\dfrac{1}{s})$	$\dfrac{1-e^{-t}}{t}$
F08	$\dfrac{s-1}{s\log(s)}$	$v(t,-1)-v(t,0)$
F09	$\dfrac{\log(s-1+\sqrt{s^2-2s})}{\sqrt{s^2-2s}}$	$e^t K_0(t)$
F10	$\dfrac{e^s K_1(s)}{s}$	$\sqrt{t(t+2)}$
F11*	$-\sqrt{s}$	$\dfrac{1}{2\sqrt{\pi t^3}}$
F12*	$s\log(s)$	$\dfrac{1}{t^2}$

*Pseudotransforms.

where r is a parameter. Our path (11) has one parameter whereas Talbot's path included two parameters. Note some points on the path: $s(0)=r$, $s(\pi/2)=\mathrm{i}r\pi/2$ and $s(3\pi/4)=3\pi(-1+\mathrm{i})r/4$.

It is of interest to examine the genesis of contour (11). Consider the inversion integral (10) with the transform $\hat{f}(s)=1/s^\alpha$ for $\alpha>0$; it can be expressed as

$$f(t)=\frac{1}{2\pi\mathrm{i}}\int_B \exp(t(s-a\log s))\,\mathrm{d}s \tag{12}$$

where $a=\alpha/t$. In general, the integral (12) is difficult to evaluate numerically because of the oscillatory behaviour of the integrand. We can circumvent this problem by choosing the

so-called steepest descent path which has the property

$$\mathrm{Im}(s - a\log s) = 0 \tag{13}$$

Let $s = x + \mathrm{i}y$, then (13) yields the equation of the steepest descent path

$$x = y\cot(y/a) \tag{14}$$

which is equivalent to (11) for $r = a$. However, note that (14) has the property (13) only for the transform $\hat{f}(s) = 1/s^{\alpha}$. Whereas, we propose to use (11) for all transforms. For more details of the above argument, see Reference [21].

To continue our development of the FT algorithm, we replace the contour B in (10) with (11), then

$$f(t) = \frac{1}{2\pi\mathrm{i}}\int_{-\pi}^{\pi} \exp(ts(\theta))\,\hat{f}(s(\theta))s'(\theta)\,\mathrm{d}\theta \tag{15}$$

Note that $s'(\theta) = \mathrm{i}r(1 + \mathrm{i}\sigma(\theta))$ where

$$\sigma(\theta) = \theta + (\theta\cot\theta - 1)\cot\theta \tag{16}$$

Then we find

$$f(t) = \frac{r}{\pi}\int_{0}^{\pi} \mathrm{Re}[\exp(ts(\theta))\,\hat{f}(s(\theta))(1 + \mathrm{i}\sigma(\theta))]\,\mathrm{d}\theta \tag{17}$$

We can approximate the value of the integral in (17) by using the trapesoidal rule with step size π/M, and $\theta_k = k\pi/M$

$$f(t, M) = \frac{r}{M}\left\{\frac{1}{2}\,\hat{f}(r)\exp(rt) + \sum_{k=1}^{M-1} \mathrm{Re}[\exp(ts(\theta_k))\,\hat{f}(s(\theta_k))(1 + \mathrm{i}\sigma(\theta_k))]\right\} \tag{18}$$

Based on numerical experiments we fix the parameter r to the value

$$r = 2M/(5t) \tag{19}$$

Then the approximant $f(t, M)$ given by (18) has only one free parameter, M, the number of terms to be summed. To control the round-off error in the computation of (18), we specify the precision requirement

$$\text{number of precision decimal digits} = M \tag{20}$$

Hence, the FT algorithm may be summarized as follows:

> FT Algorithm. Specify the transform $\hat{f}(s)$ and furnish values for t and M. First set the precision to M, then compute the approximant $f(t, M)$ from (18) and (19).

The word 'fixed' in the name of the FT algorithm points to the fact that the path (11) is fixed because the parameter 'r' has the constant value given by (19); whereas, in Reference [13] the path is not fixed for all transforms. A prototype implementation of this algorithm in *Mathematica* was given in Section 1.

The FT algorithm is a substantial simplification of the original Talbot procedure. Of course, when Talbot developed his procedure, he did not have access to multi-precision computing software. And neither did Murli and Rizzardi [19] who implemented a FORTRAN version of Talbot's procedure. Their program has a line count of 3090. Also, the FT algorithm is different from the procedure proposed by Evans and Chung [21]. The functional form of their contour coincides with our (11). However, their approach to the parameter r is very different from our value (19), which is independent of the transform to be inverted. Whereas, Evans and Chung [21] use $r = \alpha/t$ where α is determined from the asymptotic behaviour of the transform. That is $\hat{f}(s) \sim c/s^{\alpha}$ as $s \to \infty$. Indeed, this choice of r furnishes the optimal contour but only when $\hat{f}(s) = c/s^{\alpha}$ as shown in (14). However, it is not effective in general, as we shall see in Section 4 with regard to the transforms F04–F10 of Table I.

In Section 4, we examine the performance of the FT algorithm. For a large class of transforms, we find the relative error estimate

$$\left| \frac{f(t) - f(t, M)}{f(t)} \right| \approx 10^{-0.6M} \tag{21}$$

That is, the number of significant digits in the approximant $F(t, M)$ is about equal to $0.6M$.

There is a potential computing problem that one may encounter with the software implementation of FT. It has to do with computing the values of certain transforms. One problem concerns the location of the branch cut for the square-root function in various software systems. The problem, and its solution are discussed in detail by Murli and Rizzardi [19]. For example, consider the transform $\hat{f}(s) = 1/\sqrt{s^2 - 1}$ which has the inverse $I_0(t)$. Both *Mathematica* and *UBASIC* have a problem with the transform. The simple solution is to express $\hat{f}(s)$ as $1/(\sqrt{s-1}\sqrt{s+1})$. Also a problem may arise with the function $\log(s^2+1)$. Again, the solution is to use the form $\log(s - \mathrm{i}) + \log(s + \mathrm{i})$.

4. PERFORMANCE OF THE ALGORITHMS

It would be nice to provide simple general error bounds that are independent of the transform under consideration. Unfortunately, this is not possible. Even for a restricted class of transforms, we are not able to provide rigorous error criteria. However, we have found a simple empirical error estimate for a large class of transforms which we label $\mathbb{F}$. In fact, these are the error estimates given by (9) and (21). They seem to be valid even though the theoretical basis for (9) and (21) is lacking.

A transform is a member of class $\mathbb{F}$ if it meets two conditions: (1) the transform has all its singularities on the real axis to the left of $s = a$, $a \geqslant 0$; and (2) the inverse $f(t)$ is infinitely differentiable for $t > 0$. Some examples of transforms in class $\mathbb{F}$ are given in Table I. This is the first set of test transforms that will be used to demonstrate the effectiveness of the GWR and FT algorithms. Note, we did not include any rational transforms in Table I; that is, transforms which are the ratio of two polynomials. It is well known that rational transforms are easy to invert. However, we do include two pseudotransforms in Table I, namely F11 and F12. These are not bona fide transform pairs. For example, the forward transform (1) of $f(t) = 1/t$, does not exist. However, there is a sense in which the inverses of F11 and F12 are valid, see p. 62 of Reference [22]. Pseudotransforms are useful in the fractional calculus. For example,

given the unit function $h(t) = 1$, then the inverse of $\sqrt{s}$ represents the three-halves derivative of $h(t)$. It is reasonable to expect that a good numerical inversion algorithm should be able to handle these so-called pseudotransforms.

For a class $\mathbb{F}$ transform, the number of significant digits of accuracy achieved by the GWR algorithm is about M whereas for FT it is about $(0.6)M$, see (9) and (21). Further, the accuracy is almost constant over a very large time interval. Indeed, Tables II–IV show this to be the case for the test transforms F06, F04 and F05, respectively. Actually, we generated tables for all the transforms F01–F12, but to save space we display the results for only three of the transforms. The results for test transform F06 represent the best case, while those of F05 show the worst case, and those of F04 are average. Note in Table IV, the error for F05 at $M = 20$ is not constant over the entire t-interval. Table V shows these results for the fixed value $M = 30$.

All the transforms in Table I have inverses in terms of familiar special functions. In fact, all these inverses can be computed by a function call in *Mathematica*. Therefore, it is easy to check the results of the inversions.

In order to further test the algorithms, we consider the more complicated transforms in Table VI, which do not have inverses in terms of familiar special functions. Most of the test transforms in Table VI are taken from Duffy [3]. His paper provides integral representations

Table II. The number of significant digits obtained for GWR and FT as a function of M and t for the transform F06.

	GWR Algorithm: M				FT Algorithm: M			
t	20	40	100	200	20	40	100	200
0.1	17	33	81	161	12	24	59	118
1	16	33	81	161	12	24	59	118
7	18	34	81	161	13	24	60	119
20	19	34	81	161	13	24	60	119
50	18	34	82	162	14	25	60	120
100	18	34	83	162	14	25	61	120
300	18	34	82	162	14	26	61	120
800	17	34	82	162	14	26	61	120

Table III. The number of significant digits obtained for GWR and FT as a function of M and t for the transform F04.

	GWR Algorithm: M				FT Algorithm: M			
t	20	40	100	200	20	40	100	200
0.1	9	20	51	106	11	25	71	129
1	10	23	60	126	14	28	67	129
7	12	26	72	158	12	24	62	121
20	13	29	79	161	11	23	60	119
50	14	32	80	159	11	23	58	118
100	16	33	79	160	10	22	58	117
300	16	30	80	160	10	22	57	116
800	18	32	80	159	10	21	56	116

Table IV. The number of significant digits obtained for GWR and FT as a function of M and t for the transform F05.

	GWR Algorithm: M				FT Algorithm: M			
t	20	40	100	200	20	40	100	200
0.1	18	34	83	160	14	27	62	120
1	18	34	82	162	14	27	62	120
7	17	33	81	161	13	26	61	120
20	16	33	80	161	13	26	61	120
50	15	32	80	160	13	25	61	119
100	12	31	79	161	12	25	60	119
300	6	26	78	157	12	25	60	119
800	0	15	74	152	3	23	58	117

Table V. The number of significant digits obtained for GWR and FT as a function of t for each transform and using the constant value $M = 30$.

	GWR Algorithm: t				FT Algorithm: t			
ID	1	20	100	800	1	20	100	800
F01	25	21	16	20	17	16	16	15
F02	22	25	25	24	18	18	18	18
F07	25	23	16	24	16	17	17	16
F08	24	25	25	25	17	18	18	18
F09	25	25	24	24	17	18	18	18
F10	25	25	26	26	17	17	17	17
F11	23	24	24	24	17	16	16	16
F12	24	24	24	24	17	15	15	15

of the inverses, which we used to calculate the reference solution $f(t)$. Duffy did a nice job researching the literature to find interesting examples of transforms used in engineering applications. Transform F14 arises from a problem in the theory of beams. Transform F15 is found in a study of the longitudinal impact on viscoplastic rods. Transform F16 arises in modeling the transients in an electric transmission line. Transforms F18 and F20 are found in analytical models of the impulsive displacement of viscoelastic fluids. Transform F21 arises in a study of telephone traffic congestion. Note that the transform F21 is given implicitly by a non-linear equation. In this case the transform values are obtained by numerically solving the non-linear equation using, for example, Newton's method.

Note in Table VI that the transform F14 is defined as the product of the transforms F03 and F13. Table VII shows the accuracy results for the inversion of F03, F13 and F14. These transforms all have a singularity at $s = +1$. The inversion computations are performed directly and also using the operational rule $s = s + 1$. Table VII shows that the results are better using the operational rule. Note the accuracy achieved with transform F14 using $s = s + 1$ and for the value $M = 30$, in Table VII. We obtain about 20 significant digits using the FT algorithm. Now F14 is test transform #5 of Duffy [3]. To test Talbot's method, Duffy uses the software implementation by Murli and Rizzardi [19], which is ACM/TOMS Algorithm

Table VI. Some class $\mathbb{F}$ transforms taken from engineering applications.

ID	$\hat{f}(s)$
F13	$\dfrac{1}{\sqrt{s^2 - \dfrac{s}{2}\sqrt{s^2-1}}}$
F14	F03 $\cdot$ F13
F15	$\dfrac{(100s-1)\,\sinh(\sqrt{s}/2)}{s(s\,\sinh(\sqrt{s}) + \sqrt{s}\cosh(\sqrt{s}))}$
F16	$\dfrac{1}{s}\exp\left(\dfrac{1}{2}(s+1-\sqrt{s^2+6s+1})\right)$
F17	$\dfrac{\sqrt{s+1}}{2\sqrt{s}\sqrt{1+2s/5}}$
F18	$\dfrac{1}{s}\exp(-s\cdot \text{F17})$
F19	$\dfrac{\sqrt{1+\log(100)}}{\sqrt{1+\log((100s+1)/(s+1))}} - 1$
F20	$\dfrac{1}{s}\exp(-s\cdot \text{F19})$
F21	$(2s+1-\hat{f}(s))\hat{f}(s) = \log(2s+2-\hat{f}(s)))$

Table VII. The number of significant digits obtained for GWR and FT as a function of t and using the constant value $M = 30$. Note, F03(s+1) is the transform F03 with the substitution $s = s + 1$.

	GWR Algorithm: t				FT Algorithm: t			
ID	0.3	2	4	9	0.3	2	4	9
F03	27	26	20	10	21	20	20	8
F03(s+1)	26	26	25	23	21	20	20	20
F13	26	24	17	6	18	18	18	8
F13(s+1)	25	24	21	20	18	18	18	16
F14	27	24	18	7	21	21	21	8
F14(s+1)	27	25	25	21	21	21	21	20

#682. Surprisingly, Duffy's results achieved only a few significant digits of accuracy for this transform, see his Figure 14. In fact, he tried the program for values of M (which he calls n) up to 100. Duffy concludes from Figure 14: 'For all times, the method does rather poorly'. We find this to be a curious statement. It seems to indicate that Talbot's method failed because F14 is a problematic transform. Indeed, it was the program that failed and not Talbot's method! Clearly, one should always be cautious with software programs. Recall the potential computing

Table VIII. The number of significant digits obtained for GWR and FT as a function of t for each transform and using the constant value $M = 30$.

	GWR Algorithm: t				FT Algorithm: t			
ID	0.3	2	4	9	0.3	2	4	9
F15	17	16	15	15	21	19	18	17
F16	25	25	23	20	15	19	19	19
F17	24	24	23	20	14	14	17	18
F18	19	24	21	19	13	14	18	19
F19	26	25	24	24	19	18	18	18
F20	26	25	25	25	19	19	19	19
F21	25	23	21	12	16	17	17	18

Table IX. Some test transforms in class $\mathbb{G}$ and their inverses.

ID	$\hat{g}(s)$	$g(t)$
G1	$\frac{1}{s^2+1}$	$\sin(t)$
G2	$\frac{1}{\sqrt{s^2+1}}$	$J_0(t)$
G3	$\frac{1}{\sqrt{s+\sqrt{s^2+1}}}$	$\frac{\sin(t)}{\sqrt{2\pi t^3}}$
G4	$\arctan(\frac{1}{s})$	$\frac{\sin(t)}{t}$
G5	$1-\exp(s-\sqrt{s^2+1})$	$\frac{J_1(\sqrt{t(t+2)})}{\sqrt{t(t+2)}}$
G6*	$-\log(s^2+1)$	$\frac{2\cos(t)}{t}$

*Pseudotransform.

problems cited at the end of Section 3. Because all methods have limitations, we emphasize the utilization of more than one algorithm to invert a transform.

Table VIII shows the accuracy results for the inversion of test transforms F15–F21. Note the results in Table VIII are comparable to the results in Table V. We conclude that the effectiveness of the algorithms GWR and FT seem to be rather constant across all class $\mathbb{F}$ transforms.

We next consider another large class of transforms that we label $\mathbb{G}$. The definition of class $\mathbb{G}$ is the same as $\mathbb{F}$ except we remove the restriction of only real singularities. That is, transforms in class $\mathbb{G}$ have singular points lying off the real axis. Examples of such transforms are given in Table IX.

Table X shows the accuracy of the inversion of transform G2 for the algorithms GWR and FT. There is a striking difference between Table X and say Table II. For a class $\mathbb{G}$ inversion, if we fix the value of M then the accuracy decreases as t increases. On the other hand, if we

Table X. The number of significant digits obtained for GWR and FT as a function of M and t for the transform G2.

	GWR Algorithm: M				FT Algorithm: M			
t	50	100	200	400	50	100	200	400
0.1	42	82	162	321	30	60	119	237
1	41	81	161	320	30	60	119	237
7	29	75	155	313	30	60	118	236
20	11	47	137	295	14	59	118	236
50	0	15	77	248	0	14	114	236
100	0	0	29	154	0	0	28	228
200	0	0	2	60	0	0	0	57
300	0	0	0	21	0	0	0	0

Table XI. The number of significant digits obtained for GWR and FT as a function of M and t for the transform G4.

	GWR Algorithm: M				FT Algorithm: M			
t	50	100	200	400	50	100	200	400
0.1	42	83	162	321	31	61	119	237
1	41	81	161	318	31	61	119	237
7	29	75	154	315	30	60	118	236
20	11	47	138	301	15	59	118	236
50	0	15	77	248	0	14	115	235
100	0	1	30	154	0	0	30	229
200	0	0	3	61	0	0	0	57
300	0	0	0	21	0	0	0	0

fix the value of t, then the accuracy improves as M increases. This is demonstrated in Tables X and XI. We do not have a simple empirical error estimate for class $\mathbb{G}$ transforms. However, this situation is not as bad as it seems. With the aid of multi-precision, we can compute the inversion by using brute force, that is, continue to increase M until the result converges. A simple strategy for GWR and FT is to run the program with, for example, $M = 20, 40, 80, \ldots$, until two successive values of $f(t)$ agree to, say, 15 significant digits. This brute force approach may seem to be lacking elegance but it is simple and effective. However, on occasion it may be time consuming.

The above problem is well known with regard to the Talbot method. Talbot's approach was to augment the contour in (11) with one additional parameter that was dependent on the distances of the singularity from the real axis. His approach was moderately effective but complicated. Evans [20] introduced an alternative contour which is more effective but the procedure seems complicated. Evans and Chung [21] have a relatively simple procedure, but it is restricted to transforms with off-axis singularities that are poles. Hence, their algorithm cannot handle the test transforms G2–G6 in Table IX.

It can be shown that with our strategy, the required value of M is proportional to t, that is

$$M = M_0 + \mu t \tag{22}$$

Table XII. The number of significant digits obtained for GWR and FT as a function of t for each transform and using the value $M = 30 + (1.6)t$.

	GWR Algorithm: t				FT Algorithm: t			
ID	20	50	100	200	20	50	100	200
G1	18	19	25	39	23	22	21	20
G2	18	19	25	39	24	22	21	20
G3	18	19	25	39	22	22	22	22
G4	18	19	25	40	23	23	22	21
G5	18	19	25	39	22	22	21	21
G6	18	19	25	40	23	23	21	21

Table XII demonstrates this fact and the FT results are about constant for the value $\mu = 5/\pi \approx 1.6$ in (22). Note, the transforms G1–G6 all have the same singular point $s = +\mathrm{i}$. Suppose the transform $g(s)$ has one singularity in the second quadrant at $s = -\alpha + \mathrm{i}\beta$, where α, $\beta \geqslant 0$. From the geometry of the contour (11) and the value of the parameter r given by (19), then we determine μ in (22) to be

$$\mu = \frac{2.5\beta}{\pi - \arctan(\beta/\alpha)} \tag{23}$$

For example, if $\alpha = 0$ and $\beta = 1$ then $\mu = 5/\pi$.

5. CONCLUSIONS

In this paper, we have investigated two simple and effective algorithms for numerically inverting Laplace transforms. The results of Section 4 demonstrate that the algorithms GWR and FT perform well for two large classes of transforms labelled $\mathbb{F}$ and $\mathbb{G}$.

From our experience, GWR and FT are the best all-round practical algorithms for inversion when using multi-precision computing. Further, we consider it important to have two 'best' algorithms. It does not seem possible to find a simple algorithm with rigorous error bounds which are independent of the transform. Therefore, we recommend the use of two algorithms, each with empirical error estimates to numerically invert a formidable transform.

The FT algorithm is a substantial simplification of the original Talbot procedure. Essentially, the FT algorithm is defined by the seven equations (11), (15)–(20). In contrast, Talbot [13] has a rather lengthy algorithm which is described by 48 equations. In fact, the equations in Reference [13] needed to implement the procedure are: (7)–(9), (24)–(31) and (58)–(94). The major portion of his algorithm is devoted to estimating values for the parameters used in the method.

REFERENCES

1. Davies B, Martin B. Numerical inversion of the Laplace transform: a survey and comparison of methods. *Journal of Computational Physics* 1979; **33**:1–32.
2. Narayanan GV, Beskos DE. Numerical operational methods for time-dependent linear problems. *International Journal for Numerical Methods in Engineering* 1982; **18**:1829–1854.

3. Duffy DG. On the numerical inversion of Laplace Transform: comparison of three new methods on characteristic problems from applications. *ACM Transactions on Mathematical Software* 1993; **19**:333–359.
4. Valkó PP, Vojta BL. *The List*. 2001; `http://pumpjack.tamu.edu/~valko`
5. Koizumi S. A new method of evaluation of the Heaviside operation expression by Fourier series. *Philosophical Magazine* 1935; **10**:1061–1076.
6. Abate J, Whitt W. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems* 1992; **10**:5–88.
7. D'Amore L, Lacetti G, Murli A. An implementation of a Fourier-series method for the numerical inversion of the Laplace transform. *ACM Transactions on Mathematical Software* 1999; **25**:279–305.
8. Sakurai T. Numerical inversion of the Laplace transform of functions with discontinuities. *Queueing Systems*, 2004; to appear.
9. Ward EE. The calculation of transients in dynamical systems. *Proceedings of the Cambridge Philosophical Society* 1954; **50**:49–59.
10. Abate J, Choudhury G, Whitt W. On the Laguerre-method for numerically inverting Laplace transforms. *INFORMS Journal of Computing* 1996; **8**:413–427.
11. Weideman JAC. Algorithms for parameter selection in the Weeks method for inverting the Laplace transform. *SIAM Journal on Scientific Computing* 1999; **21**:111–128.
12. Gaver DP Jr. Observing stochastic processes and approximate transform inversion. *Operations Research* 1966; **14**:444–459.
13. Talbot A. The accurate numerical inversion of Laplace transforms. *Journal of the Institute of Mathematics and Its Applications* 1979; **23**:97–120.
14. Davies B. *Integral Transforms and Their Applications* (3rd edn). Springer: New York, 2002.
15. Wimp J. *Sequence Transformations and Their Applications.* Academic Press: New York, 1981.
16. Stehfest H. Algorithm 368: numerical inversion of Laplace transforms. *Communications of the ACM* 1970; **13**:47–49 and 624.
17. Valkó PP, Abate J. Comparison of sequence accelerators for the Gaver method of numerical Laplace transform inversion. *Computational Mathematics and Applications* 2004, to appear.
18. *Mathematica* Information Center, http://library.wolfram.com/database/MathSource/4738/
19. Murli A, Rizzardi M. Algorithm 682. Talbot's method for the Laplace inversion problem. *ACM Transactions on Mathematical Software* 1990; **16**:158–168.
20. Evans GA. Numerical inversion of Laplace transforms using optimal contours in the complex plane. *International Journal of Computer Mathematics* 1993; **49**:93–105.
21. Evans GA, Chung KC. Laplace transforms inversion using optimal contours in the complex plane. *International Journal of Computer Mathematics* 2000; **73**:531–543.
22. Doetsch G. *Introduction to the Theory and Application of Laplace Transformation*. Springer: New York, 1974.

