Numerical inversion of 2-D Laplace transforms applied to fractional diffusion equations

Peter P. Valkó*a,†, Joseph Abateb

a Department of Petroleum Engineering, Texas A&M University, 3116 TAMU, College Station, TX 77843, USA
b 900 Hammond Road, Ridgewood, NJ 07450-2908, USA

Available online 25 November 2004

Abstract

The problem of numerically inverting the double Laplace transform arises, for instance, in solving linear partial differential equations of fractional order on the semi-infinite domain. This work introduces two algorithms for numerical inversion of the double Laplace transform. The algorithms have only one free parameter (the number of terms in the summation) and provide increasing accuracy with increasing value of the parameter. They utilize the public domain one-dimensional routines: FT and GWR. Inherent in the algorithms is multi-precision computing with automatically adjusting the required number of precision digits. By using simple transform pairs and the example of classical diffusion in a force field we show that the procedures can provide extremely high accuracy. Then we derive the operational solution of the fractional diffusion equation subject to reflecting and absorbing boundary conditions at the origin. Finally, we compare the inversion results to the exact solutions given by Metzler and Klafter [Physica A 278 (2000) 107].

© 2004 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Numerical inversion; Double Laplace transform; Fractional diffusion equation

1. Introduction

In a previous paper [3] we presented two algorithms for numerically inverting Laplace transforms, which we called Gaver–Wynn–Rho (GWR) and Fixed–Talbot (FT). Their main advantage is that they...
are easy to use. There is only one free parameter $M$, which is the number of terms used. The algorithms GWR and FT provide increasing accuracy as $M$ increases. In fact, this can only be accomplished by using multi-precision computing.

Our purpose in this paper is to show how GWR and FT can be directly applied to the numerical inversion of two-dimensional Laplace transforms. We build the 2-D algorithm by directly utilizing the routines GWR and FT without any alterations or extensions. We are motivated by the desire to invert the operational solution of fractional diffusion equations. Specifically, let $f(x, t)$ represent the concentration function for a diffusion process defined on the semi-infinite domain $0 \leq x < \infty$ for $0 \leq t < \infty$.

The problem of numerically inverting the double Laplace transform is to determine approximations for $f(x, t)$ when numerical values of the two-dimensional transform function

$$
\hat{f}(p, s) = \int_0^\infty \int_0^\infty f(x, t) e^{-(px+st)} \, dx \, dt
$$

(1)
can be computed.

We are aware of seven numerical procedures that have been published for double Laplace transform inversion [20,11,6,18,12,17,2]. All these procedures are multivariate extensions of well-known inversion methods for one-dimensional Laplace transforms. The procedures [20,11] are based on a numerical inversion scheme which we call the Zakian method. The procedures [6,18,12] are based on the Fourier series method while [17,2] are based on the Laguerre method. In Section 2, we discuss these various methods and their merits. We also illustrate the inherent difficulties encountered in numerical inversion, by considering the Fourier series method as an example.

In Section 3, we review the one-dimensional algorithms GWR and FT. Also in Section 3, we consider some numerical examples to demonstrate the effectiveness of these algorithms. In Section 4, we present two algorithms for numerically inverting two-dimensional Laplace transforms and investigate their performance. In Section 5, we develop operational solutions to fractional diffusion equations, and provide numerical examples in Section 6.

2. Background on LT numerical inversion

There is an enormous literature on the subject of numerically inverting Laplace transforms in one dimension. A bibliography of more than thousand papers is available on the WEB [22]. In fact, there are over 100 algorithms available for numerical inversion of Laplace transforms. However, most of these algorithms are variants of one another. Actually, there are less than 10 basic methods and only four seem worthwhile. That is, only four methods have passed the test of time. The approach in each of the four methods are as follows:

(i) Fourier series expansion;
(ii) Laguerre function expansion;
(iii) Combination of Gaver functionals;
(iv) Deform the Bromwich contour.
A nice review of these methods is given in Davies [8, Chapter 19]. In fact, we recommend reading this chapter because it provides an interesting discussion of the mathematical basis for each approach. Also, Davies gives an overview of various implementation alternatives that have been used in developing algorithms. However, he does not provide the necessary specifics to construct algorithms because of complications associated with parameter selection. Inherent in any algorithm is a number of free parameters. The development of a specific algorithm involves intricate rules for selecting parameter settings. It is worth a short digression here, to understand these inherent difficulties. By way of example, we consider the Fourier series method.

Over the years, there have been about 40 algorithms developed which are based on the Fourier series method. In other words, there are about 40 variants to the Fourier series procedure for numerical inversion. The developments all start with expressing the Laplace inverse in terms of Fourier integrals

\[ f_s(t) = -\frac{2e^{at}}{\pi} \int_0^\infty \text{Im}\left[ \hat{f}(a + iy) \right] \sin(yt) \, dy, \]  

(2)

\[ f_c(t) = \frac{2e^{at}}{\pi} \int_0^\infty \text{Re}\left[ \hat{f}(a + iy) \right] \cos(yt) \, dy. \]  

(3)

Note, in these integrals “a” is essentially a free parameter. Then the inverse function can be expressed as

\[ f(t) = cf_s(t) + (1 - c)f_c(t). \]  

(4)

Now we have two free parameters, a and c. Next, we provide an approximation to f(t) by using the trapezoidal rule with step size h, in the integrations (2) and (3). Also, we truncate the series in each case to n terms. So, at this point we have four free parameters: a, c, h, n. There is the celebrated Poisson summation formula to help determine the ratio a/h but assigning values to the three parameters is mostly based on heuristic reasoning. However, we are not done! It turns out that for many transforms, the value of n must be made large to achieve a good approximation. Therefore, a convergence acceleration scheme is required to reduce n. With regard to the Fourier series method, most algorithms incorporate the Wynn epsilon algorithm for acceleration of the sequence of partial sums. However, there are a number of other methods used as well, such as: Euler summation, the quotient difference algorithm, the Levin u-transformation and the Barber–Hamer algorithm. Of course, this introduces another parameter m, which is the size of the sequence to accelerate.

It is no wonder that there are about 40 variants of the Fourier series procedure that have been published. Clearly, each algorithm presents a different scheme for finding the optimum parameter values. For a discussion of these procedures and their references see the 1992 survey paper by Abate and Whitt [4]. Since 1992, several Fourier series algorithms have been developed; of notable interest is [7].

Now consider the question of why it is so difficult to select parameter values. The calculations are such, that as we increase the value of one parameter in order to improve the theoretical accuracy, the rounding errors actually decrease the achieved accuracy. That is, there is an intricate interplay between roundoff errors and the values of the parameters when computing with fixed machine precision. The computations are prone to roundoff error propagation and many may become numerically unstable. In other words, with fixed machine precision, their accuracies increase up to a certain point only. Following that, they decrease and are destroyed completely. One answer to numerical instability is to use high pre-
cision arithmetic. That is, use a multi-precision computation environment such as Mathematica, Maple, UBASIC, etc. Unfortunately, this option was not available in the past.

The problems outlined above are not unique to the Fourier series method. Actually, all the methods seem to have inherent computational difficulties. For example, the method based on the Laguerre function expansion has also received a great deal of attention in the literature. There have been about 15 algorithms developed, based on this method. For a discussion of these algorithms and their references see the survey paper [1].

We assert that the four popular methods identified above should be reexamined using multi-precision computing. We have done this for the methods (iii) and (iv) in [3], and have developed the algorithms GWR and FT, which we review in the next section. Then, in Section 4, we show how they can be directly applied to numerically inverting double Laplace transforms.

3. The 1-D algorithms FT and GWR

To demonstrate the simplicity of the FT algorithm, we display here a prototype implementation in Mathematica.

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

Note the source code has ten lines! The user provides the transform function “F[ ]”; the value of \( t \) at which the inverse is desired; and 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. For convenience, the FT program is given as a Mathematica add-on package posted in the MathSource library on the WEB [14].

We now give a sketch of the basis of this procedure, the details are given in [3]. Consider the special representation for the inversion integral given by

\[
f(t) = \frac{1}{2\pi i} \int_{-\pi}^{\pi} \exp(tS(\theta)) \tilde{f}(S(\theta))S'(\theta) d\theta, \tag{5}
\]

where the path \( S(\theta) \) wraps around the negative real axis such that

\[
S(\theta) = r \theta (\cot(\theta) + i) \quad -\pi < \theta < +\pi, \tag{6}
\]

and where \( r \) is a constant to be specified. Note, that \( S'(\theta) = ir(1 + i\sigma(\theta)) \) where

\[
\sigma(\theta) = \theta + (\theta \cot(\theta) - 1) \cot(\theta). \tag{7}
\]
In (5), \( \hat{f}(z) \) is the Laplace transform of \( f(t) \). The integral in (5) can be expressed as

\[
f(t) = \frac{r}{\pi} \int_0^\pi \text{Re} \left[ \exp(tS(\theta)) \hat{f}(S(\theta))(1 + i\sigma(\theta)) \right] d\theta.
\] (8)

We can approximate the value of the integral in (8) by using the trapezoidal rule with step size \( \pi/M \), and \( \theta_k = k\pi/M \)

\[
f(t, M) = \frac{r}{M} \left\{ \frac{1}{2} \hat{f}(r) \exp(rt) + \sum_{k=1}^{M-1} \text{Re} \left[ \exp(tS(\theta_k)) \hat{f}(S(\theta_k))(1 + i\sigma(\theta_k)) \right] \right\}.
\] (9)

Based on numerical experiments we fix the parameter \( r \) to the value \( r = 2M/5t \).

\[
r = 2M/5t.
\] (10)

Then the approximant \( f(t, M) \) given by (9) has only one free parameter, \( M \), the number of terms to be summed. To control the roundoff error in the computation of (9), we specify the precision requirement for the Laplace transform values involved:

\[
\text{number of precision decimal digits} = M.
\] (11)

We stress that (11) is a vital part of the FT algorithm. Actually, it is the keynote of our approach.

At the end of this section, we consider some numerical examples which demonstrate the effectiveness of the FT algorithm. For a large class of transforms, we find the relative error estimate

\[
\left| \frac{f(t) - f(t, M)}{f(t)} \right| \approx 10^{-0.6M}.
\] (12)

That is, the number of significant digits achieved in the approximant \( f(t, M) \) given by (9) is about equal to \( 0.6M \), for a particular class of transforms.

We now sketch the GWR algorithm. A discrete inversion formula for the Laplace transform is given in terms of the so-called Gaver functionals,

\[
f_k(t) = \alpha k \left( \frac{2k}{t} \right)^k \sum_{j=0}^{k} (-1)^j \binom{k}{j} \hat{f}((k+j)\alpha/t),
\] (13)

where \( \alpha = \log(2) \). Then the inversion formula is given by

\[
f(t) = \lim_{k \to \infty} f_k(t).
\] (14)

Unfortunately, the Gaver functionals as given by (13) provide a very poor approximation because \( |f(t) - f_k(t)| \sim c/k \) as \( k \to \infty \). Hence, to achieve a good approximation, a convergence acceleration scheme is required for the sequence \( f_k(t) \). A very good candidate is the Wynn rho algorithm, which is given by the recursive scheme

\[
\rho_{-1}^{(n)} = 0, \quad \rho_0^{(n)} = f_n(t), \quad n \geq 0,
\]

\[
\rho_k^{(n)} = \rho_{k-1}^{(n+1)} + \frac{k}{\rho_{k-1}^{(n+1)} - \rho_k^{(n+1)}}, \quad k \geq 1,
\] (15)
then the approximant to \( f(t) \) is obtained as
\[
f(t, M) = \rho_0^M.
\]  
(16)

Note well, the integer \( M \) in (16) must be even.

To complete the algorithm specifications we need the computational precision requirement.

Number of precision decimal digits = \( 2.1M \),
(17)

which is a vital part of the GWR algorithm.

An implementation of this algorithm is given as a Mathematica add-on package posted in the MathSource library on the WEB [15]. Below we consider some numerical examples and find the relative error estimate
\[
\left| \frac{f(t) - f(t, M)}{f(t)} \right| \approx 10^{-0.8M},
\]
(18)
for a large class of transforms.

It should be remarked that the approximation (13) has the same general representation as one proposed by Zakian [23], which may be written as
\[
f_n(t) = \frac{1}{t} \sum_{k=1}^{n} c_k \hat{f}(a_k/t).
\]
(19)

The approach of Zakian for determining the coefficients \( c_k \) and \( a_k \) is equivalent to the application of a Gaussian quadrature rule to the Bromwich inversion integral. A two-dimensional extension of the Zakian algorithm is used in [20,11].

We now consider the performance of the algorithms FT and GWR. 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 \( F \). In fact, these are the error estimates given by (12) and (18). They seem to be valid even though the theoretical basis for (12) and (18) is lacking.

A transform is a member of class \( F \) if it meets two conditions: (1) the transform has all its singularities on the real axis to the left of \( s = a, \ a \geq 0 \); and (2) the inverse \( f(t) \) is infinitely differentiable for \( t > 0 \). Some examples of transforms in class \( F \) are
\[
\hat{f}(s) = -\log(s)/s, \quad f(t) = \log(t) + \gamma, \quad (20)
\]
\[
\hat{f}(s) = \exp\left(-\frac{1}{4s}\right)/s \sqrt{s}, \quad f(t) = 2 \sin(\sqrt{t})/\sqrt{\pi}, \quad (21)
\]
\[
\hat{f}(s) = +\sqrt{s}, \quad f(t) = -1/\sqrt{4\pi t^3}. \quad (22)
\]

Note that \( f(t) \) in (22) is the three-halves derivative of 1, see p. 105 of Podlubny [19]. In fact, (22) is not a bona fide Laplace transform pair, because the forward transform integral does not exist. However, this pseudotransform is a useful concept and they have been around for a long time, see p. 62 of Doetsch [9].

Numerical results for the transforms (20)–(22) are given in Tables 1–3, and the number of significant digits are as predicted, except for a few points.
Next we consider another large class of transforms that we label $\mathcal{G}$. The definition of class $\mathcal{G}$ is the same as $\mathcal{F}$ except we remove the restriction of only real singularities, and allow singular points in the complex plane. An example of such a transform is

$$\hat{f}(s) = \frac{1}{\sqrt{s^2 + 1}}, \quad f(t) = J_0(t).$$

\[ (23) \]
Table 4  
The number of significant digits obtained for GWR and FT as a function of $M$ and $t$ for the transform $1/\sqrt{s^2 + 1}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$M =$</th>
<th>GWR algorithm</th>
<th>FT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>0.1</td>
<td>18</td>
<td>32</td>
<td>80</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>31</td>
<td>79</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>21</td>
<td>75</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>6</td>
<td>47</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>300</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5  
The number of significant digits obtained for GWR and FT as a function of $M$ and $t$ for the transform (24)

<table>
<thead>
<tr>
<th>$t$</th>
<th>$M =$</th>
<th>GWR algorithm</th>
<th>FT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>0.1</td>
<td>&gt; 20</td>
<td>&gt; 40</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>15</td>
<td>34</td>
</tr>
<tr>
<td>50</td>
<td>9</td>
<td>17</td>
<td>43</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>19</td>
<td>46</td>
</tr>
<tr>
<td>300</td>
<td>13</td>
<td>23</td>
<td>52</td>
</tr>
<tr>
<td>800</td>
<td>14</td>
<td>28</td>
<td>61</td>
</tr>
</tbody>
</table>

Table 4 shows the accuracy of the inversion of transform (23). Note, for a fixed value of $M$, the accuracy decreases as $t$ increases. That is, the error estimates (12) and (18) do not apply for transforms in class $G$.

We do not have a simple empirical error estimate for class $G$ 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.

Finally, we consider transforms for which the inverse function has discontinuities in the $n$-derivative for $n \geq 0$. An example of such a transform is

$$\hat{f}(s) = \frac{1 - e^{-2s}}{s} + e^{-2s} \log \left(1 + \frac{1}{s}\right), \quad f(t) = \frac{3 - t - e^{2-t}}{t - 2}H(t - 2) + H(t),$$

where $H(t)$ is the Heaviside step function.

Table 5 gives the numerical results for the transform (24). Note, that the results are not very good. All inversion methods have problems with this class of transform.
Let us remark that some software implementation of FT may encounter a computing problem with certain transform functions that have a branch cut. For example, there may be a problem with the evaluation of $\sqrt{s^2 + 1}$. The simple solution is to express it as $\sqrt{s + i}\sqrt{s - i}$.

4. Two algorithms for 2-D numerical inversion

Our goal is to provide a simple procedure for numerically inverting two-dimensional Laplace transforms. The algorithm should have only one free parameter and should utilize the routines FT and GWR.

Consider double inversion as a two step process. In the first step we invert, say, on the “$s$”-transform variable

$$\hat{f}(p,t) = L^{-1}\{\hat{\hat{f}}(p,s)\}. \quad (25)$$

Here, we see the motivation for our notation in (1), that is the double hat represents the double transform. If in the transform function $\hat{\hat{f}}(p,s)$, we consider $p$ as a constant and invert on $s$, then we are left with a one-dimensional transform $\hat{f}(p,t)$.

In the second step, we invert on the “$p$” transform variable and obtain

$$f(x,t) = L^{-1}\{\hat{f}(p,t)\}. \quad (26)$$

Consider the numerical inversion calculation for (26). We need to compute the single-transform function $\hat{f}(p,t)$ for a sequence of $p$-values. The easy way to do this, is to use the real inversion algorithm GWR in this step. Note, for each $p$-value to be computed, we need to do numerical inversion on the “$s$”-variable as represented in (25). For this step, which we call the inner loop, we should use the FT algorithm because it is fast.

Hence, the conceptual construction of the two-dimensional algorithm is complete. Now with the power of Mathematica we can implement it using the conceptual construct as follows:

```
L2DGWRFT[F_, x_, t_, M_] := Module[
  {Fp},
  Fp[p_] := FT[F[p], t, 3 M];
  GWR[Fp, x, M]
]
```

The user provides the double transform function as $F[p_,s_]$ the values of $x$ and $t$ at which the inverse is desired; and the value of the parameter $M$. In the first step of the procedure, the inversion (25) is performed using the FT algorithm; and the second step is (26) using GWR. Note, the FT routine in the module requires a precision of $3M$. Recall the output of FT will have precision of about $(0.6)(3M)$, as given by (12), which is the required input precision for the routine GWR.

Hence, we have constructed the two-dimensional algorithm which we call L2DGWRFT.

We can also construct a second two-dimensional algorithm by replacing FT in the inner loop with GWR, as follows:
Table 6
The number of significant digits obtained for the transform \(\log(ps)/ps\) with \(M = 20\)

<table>
<thead>
<tr>
<th>(t)</th>
<th>(x)</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>0.3</td>
<td>16</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 7
The number of significant digits obtained for the transform \(\log(ps)/ps\) with \(M = 30\)

<table>
<thead>
<tr>
<th>(t)</th>
<th>(x)</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>0.3</td>
<td>23</td>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>26</td>
<td>26</td>
<td>24</td>
</tr>
</tbody>
</table>

L2DGWRGWR\[F_, x_, t_, M_\]:=Module[  
  \{Fp\},  
  Fp[p_] := GWR[F[p], t, 4 M];  
  GWR[Fp, x, M]  
]

Notice, in this module, the inner loop GWR needs to be called with an \(M\)-value given by \(4M\). Also, we point out that the algorithm L2DGWRFT is faster than L2DGWRGWR by about a factor of six when \(M = 30\).

We now consider some simple transforms with which to test the effectiveness of the two-dimensional algorithms. As a first example, consider the transform

\[
\hat{f}(s, p) = \frac{-\log(ps)}{ps}, \quad f(x, t) = \log(xt) + 2\gamma .
\]  (27)

The results shown in Tables 6 and 7 were generated defining \(F[p\_][s\_] = -\log(ps)/(ps)\). If we consider \(p\) as a constant, then the \(s\)-transform belongs to class \(F\). Likewise, the \(p\)-transform, for \(s\) constant, also belongs to class \(F\). Hence we may expect the double inversion to need moderate \(M\). Indeed, this is the case as shown in Tables 6 and 7. Note, in each case the number of significant digits is about equal to \(0.8M\).

As a second example, consider the transform

\[
\hat{f}(s, p) = \frac{1}{\sqrt{ps}} \exp(-2\sqrt{ps}), \quad f(x, t) = \frac{1}{\pi \sqrt{xt}} \exp\left(-\frac{1}{4x^2t}\right).
\]  (28)

The inversion results are shown in Table 8 and we see that they are comparable to those of Table 7.
Table 8
The number of significant digits obtained for the transform $\frac{1}{\sqrt{p}} \exp(-2\sqrt{s/p})$ with $M = 30$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9
The number of significant digits obtained for the transform of the Smoluchowski equation (31) for $x_0 = 0$ with $M = 30$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

* Absolute error is less than $10^{-10}$.

Table 10
The number of significant digits obtained for the transform of the Smoluchowski equation (31) for $x_0 = 2$ with $M = 30$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

In order to further test the algorithms L2DGWRFT and L2DGWRGWR, we consider a more complicated transform which is found in an analytical model of sedimentation. The equation of motion (in dimensionless form) pertains to diffusion in a force field; let $f(x, t)$ be the concentration function, then we have

$$\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial f}{\partial x},$$

(29)

with the initial and boundary conditions: $f(x, t)|_{t=0} = \delta(x - x_0)$ and $\frac{\partial f}{\partial x}|_{x=0} = 0$. The solution of (29) was first given by Smoluchowski [21] as
\[ f(x, t) = \frac{1}{2\sqrt{\pi t}} \left[ \exp \left( -\frac{(x - x_0)^2}{4t} \right) + \exp \left( -\frac{(x + x_0)^2}{4t} \right) \right] \times \exp \left( -\frac{(x - x_0)^2}{2} - \frac{t}{4} \right) + \frac{1}{2} e^{-t} \text{erfc} \left( \frac{x + x_0 - t}{2\sqrt{t}} \right). \] (30)

Our test will be to recover (30) from the Laplace transform solution by inversion. A nice presentation of the operational solution is given by Gaver [10], which he found,

\[ \hat{f}(p, s) = \frac{1}{s} \left( \frac{\psi_1}{p + \psi_1} \right) \left[ p e^{-x_0 \psi_2} - \psi_2 e^{-x_0 p} \right], \] (31)

where \( \psi_{1,2} = \frac{1}{2} (\sqrt{1 + 4s} \pm 1) \). Note the transform factorization of (31) and that the last factor is 1 for \( x_0 = 0 \).

Table 9 displays the inversion results of (31) for \( x_0 = 0 \) and parameter value \( M = 30 \). Whereas, Table 10 is for the initial value of \( x_0 = 2 \). Notice, that in both cases the results are very good.

5. Operational solution of the fractional diffusion equation

Consider the fractional diffusion equation

\[ f(x, t) - f(x, 0) = D_t^{-\gamma} K_{\gamma} \frac{\partial^2}{\partial x^2} f(x, t), \] (32)

with \( 0 < \gamma < 1 \), \( K_{\gamma} \) a constant and \( D_t^{-\gamma} \) the fractional integration operator, see Metzler and Klafter [16]. We impose the following initial and boundary conditions

\[ f(x, t)|_{t=0} = \delta(x - x_0), \] (33)

\[ \frac{\partial}{\partial x} f(x, t)|_{x=0} = 0. \] (34)

The condition (34) is that of a reflecting barrier placed at the origin of the \( x \)-axis on which the diffusing particle moves.

To find the operational solution of (32), first take the Laplace transform with respect to the spatial variable \( x \), and apply the boundary conditions, then

\[ \hat{f}(p, t) - e^{-x_0 p} = D_t^{-\gamma} K_{\gamma} \left( p^2 \hat{f}(p, t) - pf(0, t) \right). \] (35)

Next, take the Laplace transform with respect to the time variable, \( t \), then

\[ \hat{f}(p, s) - s^{-1} e^{-x_0 p} = K_{\gamma} \left( p^2 \hat{f}(p, s) - s^{-\gamma} p \hat{f}(0, s) \right), \] (36)

where we have used the fractional integration rule

\[ \mathcal{L}^{-1} \{ D_t^{-\gamma} g(t) \} = s^{-\gamma} \hat{g}(s). \] (37)

Rearranging (36), we have

\[ \hat{f}(p, s) = \frac{p \hat{f}(0, s) - \psi^2 s^{-1} e^{-x_0 p}}{p^2 - \psi^2}. \] (38)
where $\psi^2 = s^{\gamma} / K_{\gamma}$. Note, that the denominator of (38) has two roots, $p = \pm \psi$. If the positive root were to remain then the solution would have exponential growth. Therefore, we impose the condition that the transform must be analytic in right-half $p$-plane. Hence, the numerator in (38) must have a zero at $p = \psi$, then

$$\hat{f}(0, s) = \psi s^{-1} e^{-x_0 p}. \quad (39)$$

Finally, the operational solution is given by

$$\hat{f}(p, s) = \frac{1}{s} \left( \frac{\psi}{p + \psi} \right) \left[ \frac{pe^{-x_0 \psi} - \psi e^{-x_0 p}}{p - \psi} \right]. \quad (40)$$

Note that the transform (40) is composed of two main factors, and that the second factor is 1 when $x_0 = 0$. Recall, we found a similar transform factorization for the operational solution to the Smoluchowski equation.

Now we consider the case of an absorbing barrier at the origin. That is, we find the operational solution of (32) subject to the initial and boundary conditions

$$f(x, t) \Big|_{t=0} = \delta(x - x_0), \quad (41)$$
$$f(x, t) \Big|_{x=0} = 0. \quad (42)$$

In this case, (35) is replaced by

$$\hat{f}(p, t) - e^{-x_0 p} = D_{1-\gamma} K_{\gamma} \left( p^2 \hat{f}(p, t) - \frac{\partial}{\partial x} f(0, t) \right). \quad (43)$$

Then proceeding as before, we find

$$\hat{f}(p, s) = \frac{\psi^2 (e^{-x_0 \psi} - e^{-x_0 p})}{s(p^2 - \psi^2)}, \quad (44)$$

where as before, $\psi^2 = s^{\gamma} / K_{\gamma}$.

### 6. Numerical examples of fractional diffusion

In this section we examine the effectiveness of the numerical inversion algorithms on the double transforms (40) and (44). We compare the inversion results to the exact solutions which are given in [16]. For the reflecting barrier, the concentration function is the analytical inverse of (40), given by

$$f(x, t) = W(x - x_0, t) + W(-x - x_0, t). \quad (45)$$

where $W(x, t)$ denotes the solution to the unrestricted process. For the special case $\gamma = 1/2$, it may be represented in terms of the Meijer G-function as

$$W(x, t) = \frac{1}{\sqrt{2\pi} \tau^{3/2}} G^{3,0}_{0,3} \left[ \begin{array}{c} x^4 \\ 16\tau^2 \end{array} \middle| \begin{array}{c} 0, 1, 1 \\ 4, 2 \end{array} \right]. \quad (46)$$

where $\tau = 4K_{1/2}\sqrt{t}$. The Meijer G-function can be evaluated with any required precision using the built in function of Mathematica.
Table 11
The number of significant digits obtained for the transform of the fractional diffusion equation (40) for $x_0 = 0$ with $M = 30$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>L2DGWRGWR algorithm</th>
<th>L2DGWRFT algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.30</td>
<td>24 24 22 22 18</td>
<td>24 24 22 20 18</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>24 24 22 22 21</td>
<td>24 24 23 23 21</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>24 24 22 22 21</td>
<td>24 24 23 23 21</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>24 24 22 22 22</td>
<td>24 24 23 23 22</td>
</tr>
<tr>
<td></td>
<td>10.00</td>
<td>24 24 22 22 22</td>
<td>24 24 23 23 22</td>
</tr>
<tr>
<td></td>
<td>30.00</td>
<td>24 24 22 22 22</td>
<td>24 24 23 23 22</td>
</tr>
</tbody>
</table>

Table 12
The number of significant digits obtained for the transform of the fractional diffusion equation (40) for $x_0 = 2$ with the L2DGWRFT algorithm

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>$M = 30$</th>
<th>$M = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.30</td>
<td>18 11 4 2 3</td>
<td>61 34 12 7 11</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>19 13 5 3 4</td>
<td>62 34 12 8 12</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>19 11 5 3 4</td>
<td>62 34 12 8 12</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>19 13 5 3 5</td>
<td>62 35 12 8 13</td>
</tr>
<tr>
<td></td>
<td>10.00</td>
<td>19 12 5 3 5</td>
<td>62 35 13 8 13</td>
</tr>
<tr>
<td></td>
<td>30.00</td>
<td>19 12 5 3 5</td>
<td>62 35 13 8 13</td>
</tr>
</tbody>
</table>

Table 13
The number of significant digits obtained for the transform of the fractional diffusion equation (44) for $x_0 = 2$ calculated with $M = 30$ and $M = 100$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>$M = 30$</th>
<th>$M = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.30</td>
<td>17 10 4 2 3</td>
<td>62 34 12 7 11</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>17 11 4 3 4</td>
<td>61 34 12 8 12</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>18 11 4 3 4</td>
<td>61 34 12 7 12</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>18 11 4 3 5</td>
<td>60 34 12 8 13</td>
</tr>
<tr>
<td></td>
<td>10.00</td>
<td>18 12 4 3 5</td>
<td>63 34 12 8 13</td>
</tr>
<tr>
<td></td>
<td>30.00</td>
<td>18 12 4 3 5</td>
<td>63 34 12 8 13</td>
</tr>
</tbody>
</table>

Table 11 shows the inversion results for the transform (40) for the case $\gamma = 1/2$, $K_{1/2} = 1$ and $x_0 = 0$. We use the parameter value $M = 30$. Notice that the number of significant digits is high.

Table 12 displays results for the same problem as in Table 11, except the initial position is $x_0 = 2$. We use a relatively low and large parameter value, $M = 30$ and $M = 100$.

The exact result for the absorbing barrier (44) is

$$f(x, t) = W(x - x_0, t) - W(-x - x_0, t),$$

(47)

see Ref. [16]. Table 13 shows the inversion results of the transform (44) for the case $\gamma = 1/2$, $K_{1/2} = 1$ and $x_0 = 2$ calculated with $M = 30$ and $M = 100$.

As seen in Tables 12 and 13, the case of $x_0$ not equal to zero is more difficult to invert. The reason is that in this case the concentration function (at fixed $t$) has a discontinuity in the derivative at $x = x_0$. 


Table 14
The number of significant digits obtained for the fractional diffusion equation with reflecting boundary for \( x_0 = 2 \); L2DGWRFT algorithm with \( M = 30 \) applied to (49) and (50) independently, as specified by (51)

<table>
<thead>
<tr>
<th>( t )</th>
<th>FT inverts on ( s )</th>
<th>FT inverts on ( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 0.3 )</td>
<td>( 23 )</td>
<td>( 24 )</td>
</tr>
<tr>
<td>( 1 )</td>
<td>( 24 )</td>
<td>( 23 )</td>
</tr>
<tr>
<td>( 3 )</td>
<td>( 23 )</td>
<td>( 23 )</td>
</tr>
<tr>
<td>( 10 )</td>
<td>( 24 )</td>
<td>( 23 )</td>
</tr>
<tr>
<td>( 30 )</td>
<td>( 26 )</td>
<td>( 23 )</td>
</tr>
</tbody>
</table>

Therefore, the performance of the inversion algorithm is similar to that given in Table 5 for the single transform (24), namely near the discontinuity the accuracy is limited even at large \( M \). Usually, there is a work-around to this problem. For example, consider the transform (40). Here the factor \( \exp(-x_0p) \) can be isolated by expressing the transform as

\[
\hat{f}(p, s) = \hat{f}_1(p, s) - e^{-x_0p} \hat{f}_2(p, s),
\]  
(48)

where

\[
\hat{f}_1(p, s) = \frac{\psi pe^{-x_0\psi}}{s(p^2 - \psi^2)}
\]  
(49)

and

\[
\hat{f}_2(p, s) = \frac{\psi^2}{s(p^2 - \psi^2)}.
\]  
(50)

Now, we can invert the transforms (49) and (50) independently, because we can apply the operational shift-rule:

\[
\hat{f}(p, s) = f_1(x, t) - H(x - x_0) f_2(x - x_0, t).
\]  
(51)

In fact, for \( x \leq x_0 \), only (49) needs to be inverted. As it turns out, that is an easy inversion and \( M = 30 \) provides high accuracy, using L2DGWRFT. However, for \( x \geq x_0 \), the results are not very good as shown in the first part of Table 14.

With regard to the transform variables, \( s \) and \( p \), our convention has been to use FT to invert the \( s \) variable. However, the assignment of \( s \) to FT is not necessary. Theoretically, the results should be independent of which transform variable FT inverts. From a practical standpoint, it may sometimes make a difference if we assign the \( p \) variable to FT; and it is the case in this problem. Note the results shown in the second part of Table 14.

7. Conclusions

We have presented two algorithms for numerically inverting two-dimensional Laplace transforms. We call them L2DGWRFT and L2DGWRGWR. The numerical examples given in Section 4 show that the procedures can provide extremely high accuracy. Further, the algorithms are simple and concise.
In Section 6, we show that these algorithms provide an effective method for the numerical solution of fractional diffusion equations which can be represented by linear partial differential equations defined on the semi-infinite domain, \(0 \leq x < \infty\) and \(0 \leq t < \infty\).

Let us remark that other methods have been developed for the numerical solution of finite domain problems, see, for example, [5,13]. In fact, numerical schemes presented in [13] can also handle nonlinear partial differential equations of fractional order.

References