Scientific Research Publishing

Applied Mathematics, 2017, 8, 562-589
http://www.scirp.org/journal/am
ISSN Online: 2152-7393
ISSN Print: 2152-7385

# The Role of High Precision Arithmetic in Calculating Numerical Laplace and Inverse Laplace Transforms

**Zinovi Krougly, Matt Davison, Sid Aiyar**

Department of Applied Mathematics, Western University, London, Ontario, Canada
Email: zkrougly@uwo.ca, mdavison@uwo.ca, saiyar.hba2015@ivey.ca

**How to cite this paper:** Krougly, Z., Davison, M. and Aiyar, S. (2017) The Role of High Precision Arithmetic in Calculating Numerical Laplace and Inverse Laplace Transforms. *Applied Mathematics*, **8**, 562-589.
https://doi.org/10.4236/am.2017.84045

**Received:** February 9, 2017
**Accepted:** April 27, 2017
**Published:** April 30, 2017



## Abstract

In order to find stable, accurate, and computationally efficient methods for performing the inverse Laplace transform, a new double transformation approach is proposed. To validate and improve the inversion solution obtained using the Gaver-Stehfest algorithm, direct Laplace transforms are taken of the numerically inverted transforms to compare with the original function. The numerical direct Laplace transform is implemented with a composite Simpson's rule. Challenging numerical examples involving periodic and oscillatory functions, are investigated. The numerical examples illustrate the computational accuracy and efficiency of the direct Laplace transform and its inverse due to increasing the precision level and the number of terms included in the expansion. It is found that the number of expansion terms and the precision level selected must be in a harmonious balance in order for correct and stable results to be obtained.

## Keywords

Numerical Laplace Transform, Numerical Laplace Transform Inversion, Composite Simpson's Rule, Gaver-Stehfest Algorithm, High Precision Computation

## 1. Introduction

Laplace transforms play a key role in many applications of mathematics to the fields of engineering, physics, and finance, whenever probability density functions, or linear differential equations or integral equations are involved. Laplace transform techniques may simplify the task of solving systems of differential equations [1], [2], [3], and can be considered in terms of typical applications [4], [5].

Numerical inversion of Laplace transform is crucial for many applications. Unfortunately, when considering interesting examples, it is often difficult to find an analytical expression for the inverse Laplace transform. Inverting the Laplace transform is a challenging task. This challenge faced in many application areas including the finding of various performance measures in queueing and related probability models [6], [7], [8], [9], in solving partial differential equations [10], and in the pricing and hedging of financial derivatives [11], [12], [13].

This paper investigate the complicated and very interesting relationship between numerical precision and the number of terms in one particular Laplace transform inversion algorithm, the Gaver-Stehfest algorithm, and illustrates this relationship using several carefully chosen numerical examples.

For numerous practical situations the inverse of Laplace transform is complicated and either doesn't have a closed form, or has a solution which cannot be represented by any simple formula, performed even in symbolic software (Maple or Mathematica). An alternative is to use a numerical technique for inversion. One way to choose among various alternative methods is to provide a large set of test problems, and to demonstrate how a specific algorithm works on each of them.

Several algorithms have been proposed for numerical Laplace transforms inversion, see for instance the surveys in [4] and [14]. The Gaver-Stehfest algorithm [15] is one of the most powerful algorithms for this purpose. The convergence of this algorithm has been examined in [16]. Unfortunately despite its theoretical advantages, in many practical applications, numerical inversion often encounters numerical accuracy problems [14] [17] [18] [19] [20]. As such, small rounding errors in computation in standard double arithmetic may significantly corrupt the results, rendering these algorithms impractical to apply. Extended precision allows to add additional significant figures, and produce results that converge to the solution. Laplace and inverse transforms for the test functions used in numerical calculations are presented in **Table 1**. These complicated functions are used to test the accuracy of the numerical Laplace transform and its inverse.

In general, lowercase letters used to denote the function $f(t)$ to be transformed, and the uppercase letter $C(s)$ to denote its Laplace transform, for example, $\mathcal{L}\{f(t)\} = C(s)$. If the closed form of $C(s)$ inversion is unknown, the original $C(s)$ compared with numerical solution $\tilde{\tilde{C}}(s)$ after double transformation. The results are illustrated in the plots and error estimations.

With the help of the *arprec* library [21], C++ and MATLAB numerical class library [22], [23] applied, to investigate the role that extended precision can play in accuracy of Laplace transform and inversions.

The remainder of the paper is organized as follows. In Section 2, a brief description of the underlying theory is given, to introduce numerical Laplace double transformation technique. Sections 3 and 4, apply the Gaver-Stehfest algorithm to the test functions with various degrees numerical accuracy. In

**Table 1.** Laplace and inverse transforms for the test functions used in numerical calculations.

| Function | $C(s)$ | $f(t)$ |
|---|---|---|
| 1 | $\frac{1}{s+2}$ | $e^{-2t}$ |
| 2 | $\left((s+0.2)^2+1\right)^{-1}$ | $e^{-0.2t}\sin(t)$ |
| 3 | $\left(s^2+1\right)^{-1}$ | $\sin(t)$ |
| 4 | $\left(s^2-1\right)\left(s^2+1\right)^{-2}$ | $t\cos(t)$ |
| 5 | $\tan^{-1}\left(s^{-1}\right)$ | $t^{-1}\sin(t)$ |
| 6 | $1/\sqrt{1+s^2}$ | $J_0(t)=\sum_{n=0}^{\infty}\frac{(-1)^n t^{2n}}{2^{2n}(n!)^n}$ |
| 7 | $\frac{e^{-1/s}}{s^{3/2}}$ | $\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}$ |

Sections 5 and 6, the stability and accuracy of the Laplace transform inversion and the role that the number of expansion terms and precision of the arithmetic play in the numerical results is described. Section 7 describes the algorithm and software implementation of the numerical direct Laplace transform. This section gives background material needed to provide the method, described in the next Section 8. In Section 8 the numerical double transformation technique to confirm agreement of the numerical inversion results is presented. In Section 9 compares the execution time for various arbitrary precision calculations. Concluding remarks are given in Section 10. The Appendix introduces C++ code used to implement numerical Laplace and inverse Laplace transform in arbitrary precision, and illustrates the corresponding graphical user interface with the help of several screenshots.

## 2. Numerical Laplace Transforms and Their Inverses

### 2.1. Laplace Transform

Let $f(t)$ be a function defined for $t \geq 0$. Then the integral

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

is said to be the Laplace transform of $f(t)$, provided the integral converges. The symbol $\mathcal{L}$ is the Laplace transformation operator, which act on the function $f(t)$ and generates a new function, $C(s) = \mathcal{L}\{f(t)\}$.

### 2.2. Inverse Laplace Transform

If $C(s)$ represents the Laplace transform of a function $f(t)$, that is, $\mathcal{L}\{f(t)\} = C(s)$, then $f(t)$ is the inverse Laplace transform of $C(s)$ and $f(t) = \mathcal{L}^{-1}\{C(s)\}$. The inverse Laplace transform $\mathcal{L}^{-1}\{C(s)\}$ is uniquely de-

termined in the sense that if $C(s)=G(s)$ and $f(t)$ and $g(t)$ are continuous functions, then $f(t)=g(t)$. This result is known as Lerch's theorem [4].

### 2.3. Numerical Laplace Transform and Inversion

The Laplace transform can be inverted either algebraically or numerically. The notation $\tilde{f}(t)$ used for the numerical approximation to $f(t)$ (numerical inversion of the Laplace transform $C(s)$), and $\tilde{C}(s)$ for the numerical Laplace transform of $f(t)$.

These results, together with the following well known properties which provide very useful numerical checks, may be applied to numerical algorithms and corresponding software.

$$\text{Integral property, } \int_0^\infty e^{-st} f(t)\,\mathrm{d}t = C(0) \tag{2}$$

$$\text{Initial value theorem, } \lim_{t\to 0} f(t) = \lim_{s\to\infty} sC(s) \tag{3}$$

$$\text{Final value theorem, } \lim_{t\to\infty} f(t) = \lim_{s\to 0} sC(s) \tag{4}$$

If $X$ is the random variable with the probability density function $f$ and the cumulative distribution function $F$, this gives

$$C(0) = \int_0^\infty e^{-st}\mathrm{d}F(t) = \int_0^\infty e^{-st} f(t)\,\mathrm{d}t = 1 \tag{5}$$

### 2.4. Numerical Laplace Double Transformation Technique

We define the following double transformation technique for the Laplace transform of the inversion:

$$\tilde{\tilde{C}}(s) = \mathcal{L}\left\{\mathcal{L}^{-1}\left\{C(s)\right\}\right\} \tag{6}$$

This definition will be used to estimate the accuracy of the Laplace transform inversion, when its closed form is unknown.

After applying the Laplace transform, the problem is said to be in the Laplace domain and it is denoted as a function of $s$ not $t$. While calculations might be easier in the Laplace domain, leaving the solution in the Laplace domain is typically not useful. To transform the result back into the time-domain, inverse Laplace transforms are used. When the analytical answer is unknown, it is difficult to know whether or not the numerical inversion results are accurate. Moreover, it is hard to judge whether or not changes to the method improve or degrade the inversion estimate.

The following steps are used:

1) Begin with the Laplace domain function $C(s)$.

2) Compute the numerical inversion using some set of parameters. In this case, we will control the precision level and the number of terms in the approximation. Setting the precision level to $N_1$, we get

$$\hat{f}_{N_1}(t) = \mathcal{L}_{N_1}^{-1}\left\{C(s)\right\} \tag{7}$$

3) Take the Numerical Laplace Transform of $\hat{f}_{N_1}(t)$, resulting in

$$\mathcal{L}\left\{\hat{f}_{N_1}(t)\right\} = \tilde{\tilde{C}}_{N_1}(s) \tag{8}$$

4) Compare the functions $C(s)$ and $\tilde{\tilde{C}}_{N_1}(s)$, and define the error-function as:

$$\varepsilon_{N_1}(s) = \left|C(s) - \tilde{\tilde{C}}_{N_1}(s)\right| \tag{9}$$

5) Repeat the process with some other precision level $N_2$.

6) Compare $\varepsilon_{N_1}(s)$ and $\varepsilon_{N_2}(s)$. The precision level that provides lower errors is superior, and the difference between the error functions can provide a way of quantifying the accuracy improvement gained from increasing the precision level.

## 3. Challenging Examples of Laplace Transform and Their Inverses

This demonstration applies the Gaver-Stehfest algorithm [15] to determine the inverse Laplace transforms of the test functions to various degrees of numerical accuracy. The inverse functions and corresponding test functions are presented in Table 1.

The inverse Laplace transform of $C(s)$ is $f(t)$, defined such that the following must hold:

$$C(s) = \int_0^\infty e^{-st} f(t)\,\mathrm{d}t \tag{10}$$

Table 1 lists seven functions which have been used as tests of the numerical Laplace transform and inverse transform. Test 6 involves the Bessel function of order 0 [24],

$$J_0(t) = \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{2^{2n}(n!)^n}, \text{ and } J_0(s) = \frac{1}{\sqrt{1+s^2}} \tag{11}$$

## 4. Gaver-Stehfest Algorithm of Inverse Laplace Transforms

The Gaver-Stehfest method [15] uses the summation:

$$f(t) \approx \frac{\ln(2)}{t} \sum_{n=1}^{L} K_n F\left(\frac{n\ln(2)}{t}\right), \tag{12}$$

where $F(\cdot)$ is the Laplace transform of $f(t)$. The coefficients $K_n$ depends only on the (necessarily even) number of expansion terms, $L$, given by:

$$K_n = (-1)^{n+L/2} \sum_{k=\left\lfloor \frac{n+1}{2} \right\rfloor}^{\min(n,L/2)} \frac{k^{L/2}(2k)!}{(L/2-k)!k!(k-1)!(n-k)!(2k-n)!} \tag{13}$$

For each function an error $E$ is calculated as the measure for the accuracy of the numerical solution. Let $f(t)$ be the analytical solution defined for $t = t_1, t_2, \cdots, t_m$. We define by $\tilde{f}(t) = \mathcal{L}^{-1}\{C(s)\}$ the numerical solution. Then $E$ gives the root-mean-square deviation between the analytical and numerical solutions for the $t$ values [19]:

$$E=\left(\sum_{i=1}^{m}\left(f\left(i/2\right)-\tilde{f}\left(i/2\right)\right)^{2}/m\right)^{1/2},\ i=1,2,\cdots m \tag{14}$$

The sum (12) doesn't provide convergence due to roundoff errors for the large number of terms $L$, usually if $L$ exceeds the number of decimal digits of precision $N$ (e.g. $L$ greater than 16 for standard double precision arithmetics). The software implementation of the numerical Laplace transform and the Laplace transform inversion are given in the Appendix.

## 5. Accuracy of the Numerical Laplace Transform Inversion as a Function of the Number of Expansion Terms and Precision

Numerical inversion of the Laplace transform is an unstable process, so all algorithms are applied in arbitrary precision. In Table 2 and in Figures 2-8 numerical results related to the test functions presented in Table 1 are reported. The numerical solutions are given for different precisions: $N=16$ (double precision), $N=32$, $N=64$, $N=128$ and $N=256$. First, as presented in Table 2, assigned $L=N$, and the number of terms $L$ in the approximation equals to the digits of precision $N$. The measures (14) are used to calculate the errors $E$ to the $t$ values 0.5, 1, 1.5, ..., 35 for the functions 1 - 6, and to the $t$ values 0.5, 1, 1.5, ..., 140 for the function 7. The reason to use this presentation is because $t$ values require a large amount of space to realize the correct accuracy results.

**Example 1.** The first function explored is:

$$C\left(s\right)=\frac{1}{s+2},\ \text{with}\ \ f\left(t\right)=e^{-2t}, \tag{15}$$

where $f\left(t\right)=e^{-2t}$ is the exact solution of the inversion. The results correspond to finding numerical inverse Laplace transform are plotted in Figure 1. For this simple function standard double arithmetic algorithms work well. In Table 2 the

Table 2. Calculation errors of the inverse Laplace transform in arbitrary precision, Values 5.66e−6 $\equiv 5.66\times10^{-6}$.

| Function | $C(s)$ | $N=16$ | $N=32$ | $N=64$ | $N=128$ | $N=256$ |
|---|---|---|---|---|---|---|
| 1 | $1/(s+2)$ | 5.66e−6 | 1.22e−9 | 6.52e−17 | 2.25e−32 | 7.02e−69 |
| 2 | $\left((s+0.2)^{2}+1\right)^{-1}$ | 3.44e−2 | 7.33e−3 | 2.46e−4 | 1.54e−10 | 5.38e−36 |
| 3 | $\left(s^{2}+1\right)^{-1}$ | 6.13e−1 | 5.04e−1 | 1.68e−1 | 2.44e−5 | 1.01e−27 |
| 4 | $\left(s^{2}-1\right)\left(s^{2}+1\right)^{-2}$ | 1.43e+1 | 1.35e+1 | 6.34e0 | 1.33e−3 | 9.88e−26 |
| 5 | $\tan^{-1}\left(s^{-1}\right)$ | 3.51e−2 | 2.09e−2 | 4.80e−3 | 3.66e−7 | 3.84e−30 |
| 6 | $1/\sqrt{1+s^{2}}$ | 1.12e−1 | 8.04e−2 | 2.26e−2 | 2.12e−6 | 2.78e−16 |
| 7 | $e^{-1/s}/s^{3/2}$ | 2.59e−1 | 6.16e−3 | 3.83e−13 | 2.14e−41 | 2.84e−88 |

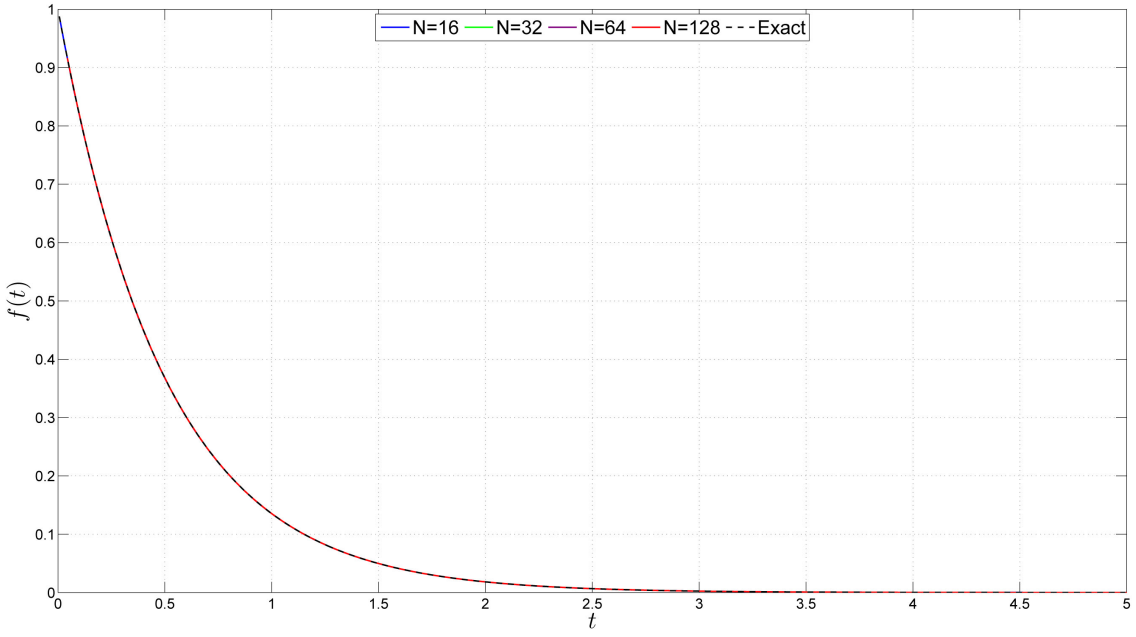



**Figure 1.** Given $C(s)=\frac{1}{s+2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=e^{-2t}$.

calculations in double precision are at the order of $10^{-6}$. In the plot, the inversion $\tilde{f}(t)=\mathcal{L}^{-1}\left\{\frac{1}{s+2}\right\}$ and the exact solution $f(t)=e^{-2t}$ visibly overlapping even for $N=16$. The precision from $N=32$ to $N=256$ gives the sequence of improvements. High accuracy is borne out by the errors at the order $10^{-32}$ and $10^{-69}$ respectively to the precision $N=128$ and $N=256$.

**Example 2.** The next test function inverted is

$$C(s)=\frac{1}{(s+0.2)^2+1}, \quad \text{with} \quad f(t)=e^{-0.2t}\sin(t), \tag{16}$$

and the results correspond to numerical inverse Laplace transform are shown graphically in Figure 2. The accuracy, at double precision, in comparison with the exact solution is very poor. As can be seen from Table 2 the calculations in double precision are at the order of $10^{-2}$. The precision level $N=32$ and $N=64$ show improvements in accuracy over double precision. In the plot, the inversion for $N=128$ and the exact solution are visibly overlapping. High accuracy is borne out in Table 2 by the errors at the order $10^{-10}$ and $10^{-35}$ respectively to the precision level $N=128$ and $N=256$.

**Example 3.** The Figure 3 provides the inverse results for the function

$$C(s)=\frac{1}{s^2+1}, \quad \text{with} \quad f(t)=\sin(t) \tag{17}$$

Steady improvement of the answer is observed through $N=128$. By $N=128$ the inversion is indistinguishable from the exact solution on the interval shown, but will eventually diverge from the exact solution for some higher values of $t$.

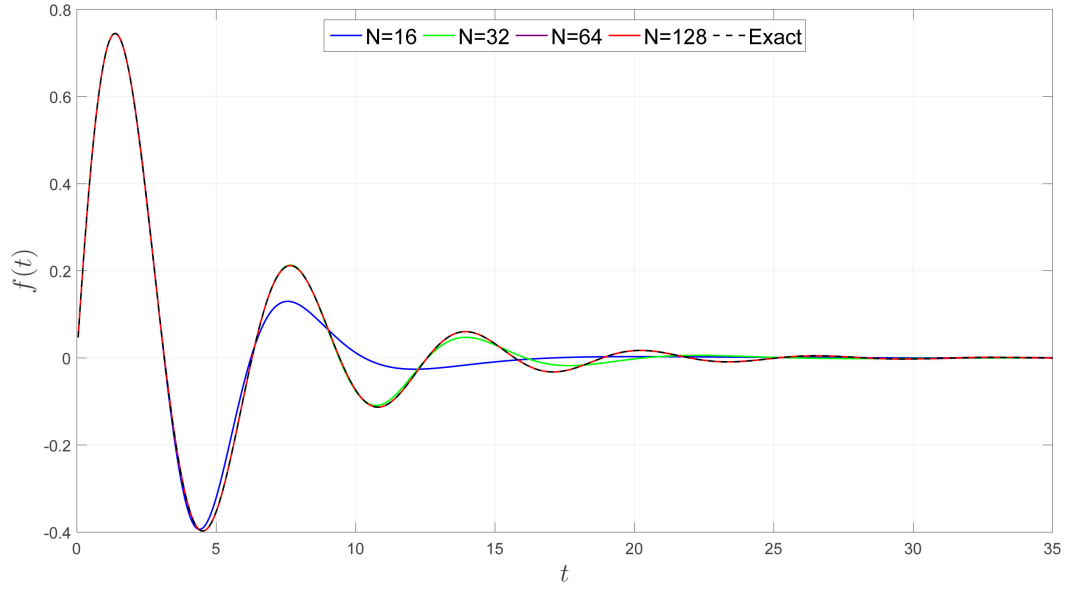



**Figure 2.** Given $C(s)=\dfrac{1}{(s+0.2)^2+1}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=e^{-0.2t}\sin(t)$.

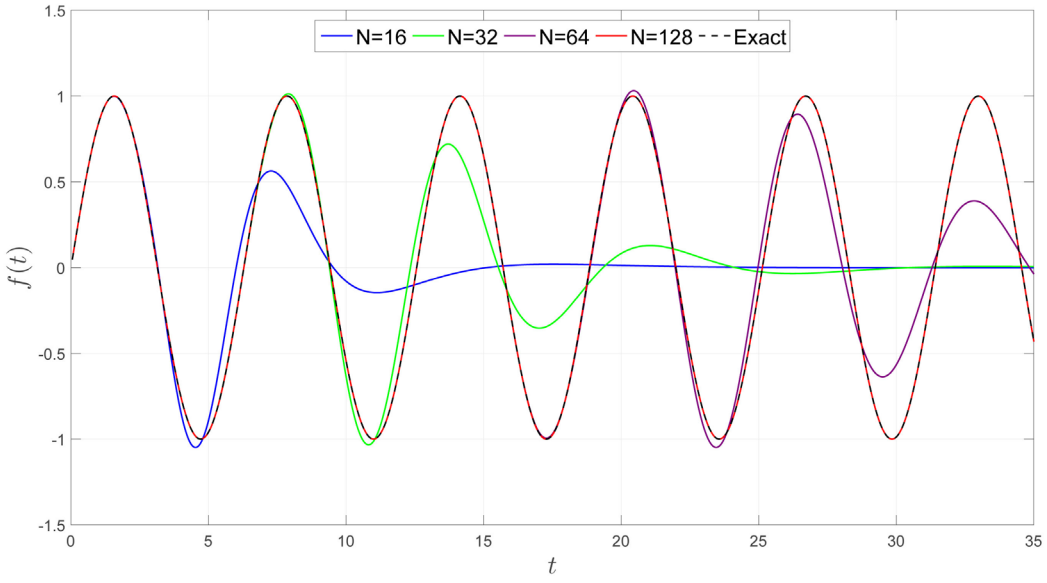



**Figure 3.** Given $C(s)=\dfrac{1}{s^2+1}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=\sin(t)$.

**Example 4.** Figure 4 shows the inverse results to the function

$$C(s)=\frac{s^2-1}{(s^2+1)^2},\quad \text{with}\quad f(t)=t\cos(t) \tag{18}$$

Similar to the previous two test functions, as $N$ increases, the interval on which the inversion is more accurate gets longer. Since the function is diverging and oscillating, the inaccuracies are more visible than in the previous two figures.

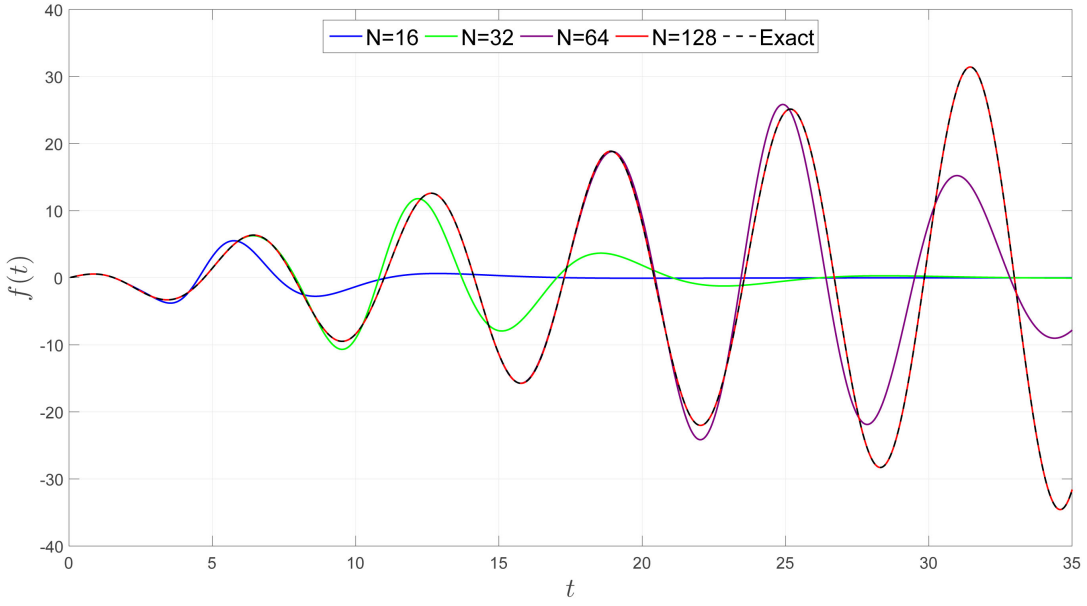



**Figure 4.** Given $C(s)=\frac{s^2-1}{\left(s^2+1\right)^2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=t\cos(t)$.

**Example 5.** Consider the inverse of the function

$$C(s)=\tan^{-1}\left(\frac{1}{s}\right), \quad \text{with} \quad f(t)=\frac{\sin(t)}{t} \tag{19}$$

In Figure 5 the precision level $N=32$ and $N=64$ show improvements in accuracy over double precision, but higher $N$ do not result in a visibly better inversion. In the plot, the numerical inversion for $N\geq 64$ and the exact solution are visibly overlapping.

**Example 6.** The inverse results to the function

$$C(s)=\frac{1}{\sqrt{1+s^2}}, \quad \text{with} \quad f(t)=J_0(t) \tag{20}$$

is plotted in Figure 6. Varying precision levels $N=32$, $N=64$, $N=64$ and $N=128$ show successive improvements in accuracy over double precision. The numerical inversion for $N\geq 128$ and the exact solution are visibly overlapping.

Naively it would seem that continuing to increase the number of terms $L$ would improve the accuracy of the approximation, since more terms seem to produce results closer to the exact solution. Consider the Bessel example with the precision level $N=32$, increasing the number of terms to $L=64$. The result is slightly better, close to $2.26\times 10^{-2}$, that for $N=64$ and $L=64$. But using the precision level $N=32$ and the number of terms in the approximation $L=128$ (Figure 7), the error will be at the order of $10^{39}$ due to numerical error dominating the solutions that are obtained this way. The algorithm was unable to provide even an order of magnitude estimate.

**Example 7.** Consider the inverse of the function

$$C(s)=\frac{e^{-1/s}}{s^{3/2}}, \quad \text{with} \quad f(t)=\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}} \tag{21}$$

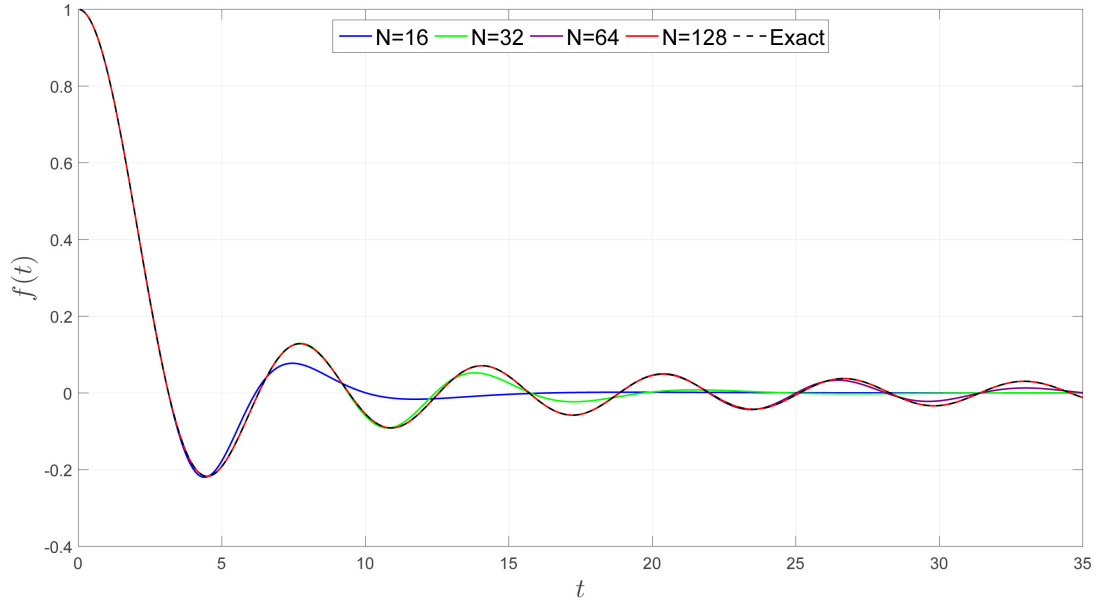



**Figure 5.** Given $C(s)=\tan^{-1}\left(\frac{1}{s}\right)$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=\frac{\sin(t)}{t}$.

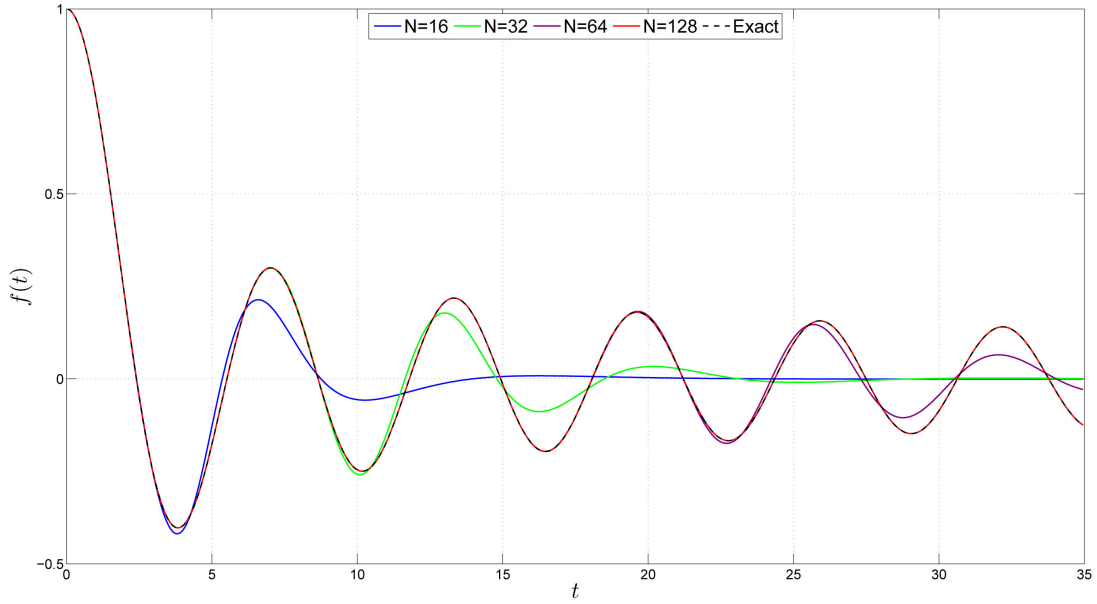



**Figure 6.** Given $C(s)=\frac{1}{\sqrt{1+s^2}}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=J_0(t)$, Bessel function of order 0.

In **Figure 8** steady improvement is observed through $N=64$ with the error at the order of $10^{-13}$ (13 decimal places accuracy, **Table 2**).

As can be seen from **Table 2**, all test functions have low level of accuracy for $N=16$, $N=32$ and for $N=64$ (except the last example). For all functions the algorithm gave reasonable answer (at least 3 decimal places of accuracy), increasing the precision level up to 128. Finally all functions have high level of

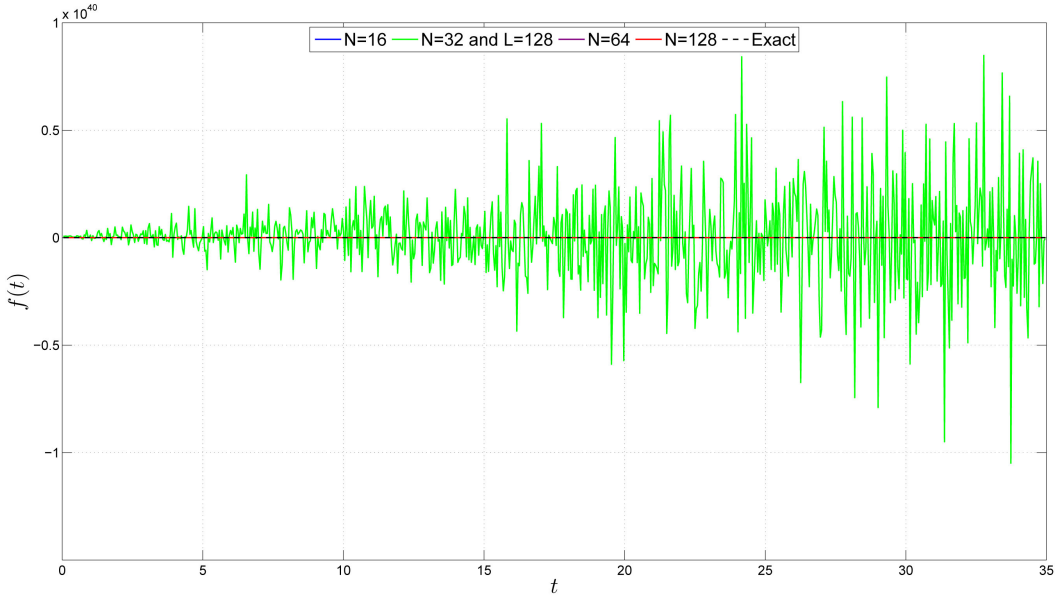



**Figure 7.** Given $C(s)=\frac{1}{\sqrt{1+s^2}}$. The precision level is $N=32$ and the number of expansion terms is $L=128$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=J_0(t)$. The algorithm was unable to provide even an order of magnitude estimate.

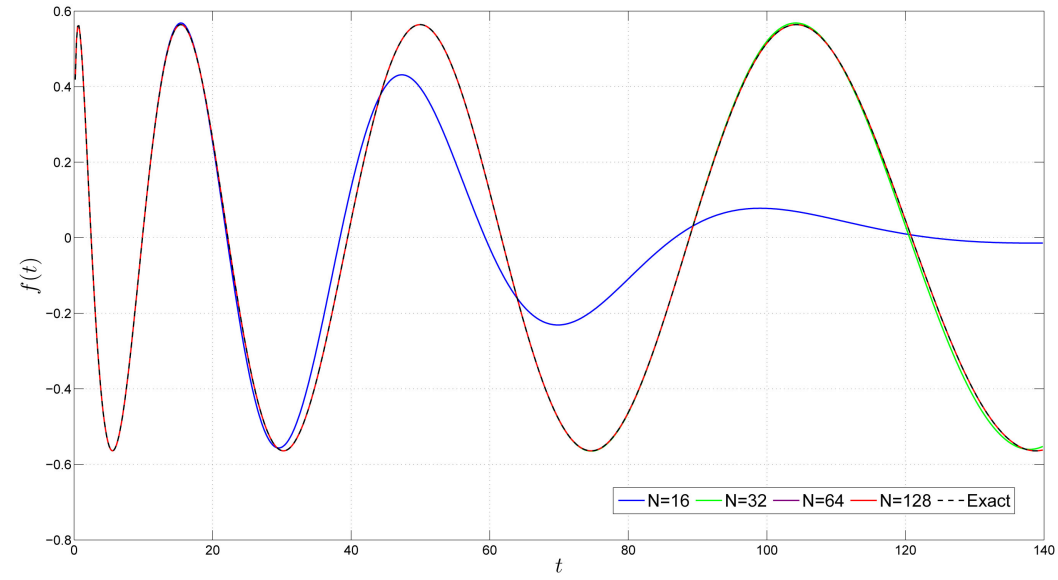



**Figure 8.** Given $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}$.

accuracy (at least 16 decimal place accuracy) increasing the precision level up to 256 digits.

## 6. The Role of the Number of Expansion Terms and Precision in the Numerical Accuracy

In Table 2 the numerical solutions are given for the number of expansion terms

equal to the precision, $L = N$. Now the accuracy of the numerical inversion is investigated, varying the number of expansion terms and precision. The Table 3 and Figure 9 give the error estimates of the numerical inverse using the Gaver-Stehfest implementation for the function $f(t) = e^{-0.2t}\sin(t)$ having the Laplace transform $C(s) = \left((s+0.2)^2 + 1\right)^{-1}$. The solutions are observed while increasing the number of expansion terms $L$. However, there is a limitation to adding additional terms.

Let $N = 16$. Increasing the number of terms in the computation to $L \geq 64$ it appears that the numerical inversion becomes unstable and functions are dominated by numerical errors. If $N = 32$ and $N = 64$, the numerical inver-

Table 3. Numerical errors of the inverse Laplace transform $\tilde{f}(t) = \mathcal{L}^{-1}\left\{\left((s+0.2)^2+1\right)^{-1}\right\}$ as a function of the number of expansion terms $L$ and precision $N$, Values 3.44e−2 $\equiv 3.44\times10^{-2}$.

| $C(s) = \left((s+0.2)^2+1\right)^{-1}$ | $N = 16$ | $N = 32$ | $N = 64$ | $N = 128$ | $N = 256$ |
|---|---|---|---|---|---|
| $L = 16$ | 3.44e−2 | 3.44e−2 | 3.44e−2 | 3.44e−2 | 3.44e−2 |
| $L = 32$ | 7.33e−3 | 7.33e−3 | 7.33e−3 | 7.33e−3 | 7.33e−3 |
| $L = 64$ | 4.66e+10 | 2.66e−4 | 2.46e−4 | 2.46e−4 | 2.46e−4 |
| $L = 128$ | 2.78e+53 | 6.86e+38 | 1.01e+10 | 1.54e−10 | 1.54e−10 |
| $L = 256$ | 2.99e+139 | 5.47e+124 | 9.72e+95 | 4.57e+23 | 5.38e-36 |

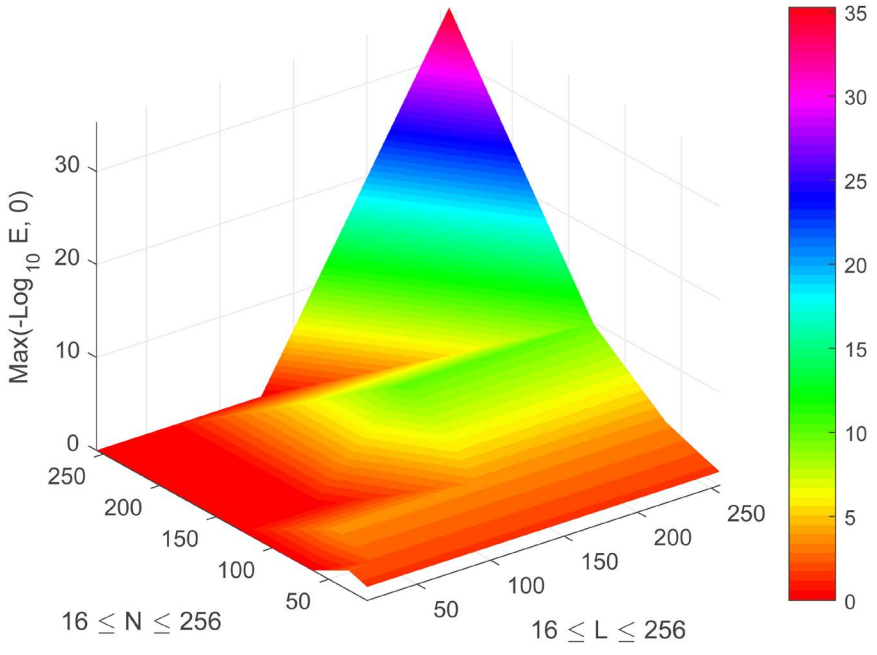



Figure 9. Numerical errors $E$ of the Laplace transform inversion $\tilde{f}(t) = \mathcal{L}^{-1}\left\{\left((s+0.2)^2+1\right)^{-1}\right\}$ as a function of the number of expansion terms $L$ and precision $N$. Estimated data presented in terms of $\mathrm{MAX}(-\mathrm{Log}_{10}E, 0)$, here 0 denotes the algorithm has failed.

sion becomes unstable for $L \geq 128$. If $N = 128$, the numerical inversion becomes unstable for $L \geq 256$. On the other hand there is only slight improvement if the precision exceeds the number of terms, $N > L$.

Table 3 clearly shows that there is no point in increasing the precision beyond a point warranted by the accuracy of the number of terms in Gaver-Stehfest algorithm. On the other hand, using a large number of terms will not be of benefit if the precision with which each term is calculated, is insufficient. For example, $L = 256$, $N = 128$ is a useless result while $N = 256$ gives high accuracy. The example, $N = 16$, $L = 16$ gives the same accuracy as $N = 256$, $L = 16$.

Similar conclusions may be drawn from the results reported in Table 4 and in Figure 10.

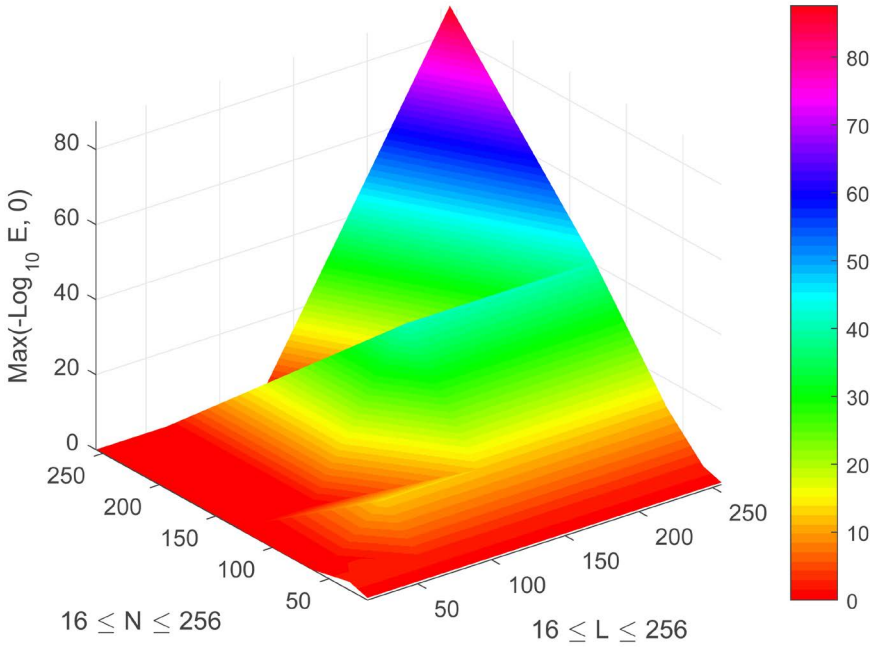



**Figure 10.** Numerical errors $E$ of the Laplace transform inversion $\tilde{f}(t) = \mathcal{L}^{-1}\left\{\left(e^{-1/s}/s^{3/2}\right\}\right.$ as a function of the number of expansion terms $L$ and precision $N$ Estimated data presented in terms of $\mathrm{MAX}\left(-\mathrm{Log}_{10}E, 0\right)$, here 0 denotes the algorithm has failed.

**Table 4.** Numerical errors of the inverse Laplace transform $\tilde{f}(t) = \mathcal{L}^{-1}\left\{e^{-1/s}/s^{3/2}\right\}$.

| $C(s) = e^{-1/s}/s^{3/2}$ | $N = 16$ | $N = 32$ | $N = 64$ | $N = 128$ | $N = 256$ |
|---|---|---|---|---|---|
| $L = 16$ | 2.59e−1 | 2.59e−1 | 2.59e−1 | 2.59e−1 | 2.59e−1 |
| $L = 32$ | 6.16e−3 | 6.16e-3 | 6.16e−3 | 6.16e−3 | 6.16e−3 |
| $L = 64$ | 3.33e+13 | 1.19e−1 | 3.83e−13 | 3.83e−13 | 3.83e−13 |
| $L = 128$ | 4.64e+56 | 2.00e+42 | 3.08e+13 | 2.14e−41 | 2.14e−41 |
| $L = 256$ | 1.60e+143 | 6.17e+128 | 1.14e+100 | 7.94e+27 | 2.84e−88 |

The error estimates are given for the example

$$f(t)=\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}},\quad \text{with}\quad C(s)=\frac{e^{-1/s}}{s^{3/2}} \tag{22}$$

Let $N=16$. Increasing the number of terms in the computation to $L\geq 64$ we can see that the numerical inversion becomes unstable. If $N=32$ and $N=64$, the numerical inversion failed for $L\geq 128$. In case of $N=128$, the numerical inversion becomes unstable for $L\geq 256$. There is only a slight improvement if the precision exceeds the number of terms $N>L$. Again as can be seen from **Table 3**, the number of terms and precision must be in a harmonious balance for good results to be obtained.

Evidently the two plots in **Figure 9** and **Figure 10**, are very much alike. They show similar tracking boundary movements and illustrate whether the algorithm has succeeded in obtaining high-order accuracy or fails due to numerical instability.

## 7. Numerical Computation of the Direct Laplace Transform

The Laplace transform of a function $f(t)$ is defined by (1) on the interval $[0,\infty]$. The problem of an infinite upper limit of integration may be removed by the substitution $t=-\ln(u)$, $\mathrm{d}t=u^{-1}\mathrm{d}u$ which replaces infinite by finite limits.

When $t=0$, $u=1$ and when $t\to\infty$, $u\to 0$. Then

$$\int_0^{\infty}e^{-st}f(t)\mathrm{d}t=\int_0^{\infty}e^{\ln\left(u^s\right)}f\left(-\ln(u)\right)u^{-1}\mathrm{d}u=\int_0^1 u^{s-1}f\left(-\ln(u)\right)\mathrm{d}u \tag{23}$$

The behaviour of the function to be transformed must be considered at the new limits, and the exponential function inside the integral requires special examination in terms of high accuracy.

### Compute the Direct Laplace Transform by Composite Simpson's Rule

For integration over the interval $[a,b]$, an even $n$ is chosen such that the function is adequately smooth over each subinterval $\left[x_j,x_{j+1}\right]$ where $x_j=a+jh$ for all $j\in\{0,1,2,\cdots,n\}$ with $h=\frac{b-a}{n}$. In particular, $x_0=a$ and $x_n=b$. Then, the composite Simpson's Rule is given by [25]:

$$\int_a^b f(x)\mathrm{d}x\approx\frac{h}{3}\left[f(x_0)+2\sum_{j=1}^{n/2-1}f\left(x_{2j}\right)+4\sum_{j=1}^{n/2}f\left(x_{2j-1}\right)+f(x_n)\right] \tag{24}$$

Applying this to the transformed integrand from the Equation (23) we get $u_j=jh$ for all $j\in\{0,1,2,\cdots,n\}$ with $h=\frac{1}{n}$. Therefore,

$$\begin{aligned}C(s)\approx\frac{1}{3n}\Bigg[&0^{s-1}f\left(-\ln(0)\right)+2\sum_{j=1}^{n/2-1}u_{2j}^{s-1}f\left(-\ln\left(u_{2j}\right)\right)\\&+4\sum_{j=1}^{n/2}u_{2j-1}^{s-1}f\left(-\ln\left(u_{2j-1}\right)\right)+1^{s-1}f\left(-\ln(1)\right)\Bigg]\end{aligned} \tag{25}$$

The basic Simpson's rule formula divides the interval $[a,b]$ of integration into two pieces. To apply the composite Simpson's rule, the interval $[a,b]$ must be divided into an even number of subintervals $n=2m$. Then $h=\frac{b-a}{n}=\frac{b-a}{2m}$.

Let $f$ be a function with four continuous derivatives. Then, the composite Simpson's rule converges to the true value of the integral with rapidity $n^{-4}$ at worst. The error committed by the composite Simpson's rule is bounded in absolute value by [25].

$$E_n=-\frac{(b-a)^5}{180n^4}\max\left|f^4(\xi)\right|,\quad a<\xi<b \tag{26}$$

For numerical Laplace transform in the obtaining approximate integral by this rule, some modifications must be made. First, the term $0^{s-1}f\left(-\ln(0)\right)$ in (25) must be addressed. This term addresses the behaviour of the function at infinity. If the Laplace transform exists, the $\lim_{t\to\infty}e^{-st}f(t)\to 0$, meaning that the exponential dampening term outweighs the value of $f(t)$ at infinity. Therefore, $\lim_{t\to\infty}f(t)$ must either exist or oscillate between some finite bounds. As such, it may be concluded that the term $0^{s-1}f\left(-\ln(0)\right)$ vanishes and may be dropped from the formulation.

Next, we need to examine the last term, $1^{s-1}\cdot f\left(-\ln(1)\right)=f(0)$. Evaluating this term should require that the function be defined at $t=0$. This can however prove problematic since many applications of the Laplace transform result in $t$-domain functions that have singularities at $t=0$. As such, we change the domain of integration to be $(0,1-\epsilon)$ as opposed to $(0,1)$, where $\epsilon$ is the machine epsilon depending on the precision level used. For example, in double precision $(N=16)$, $\epsilon\approx 2.22\times 10^{-16}$.

Therefore we have:

$$C(s)\approx\frac{1}{3n}\left[2\sum_{j=1}^{n/2-1}u_{2j}^{s-1}f\left(-\ln\left(u_{2j}\right)\right)+4\sum_{j=1}^{n/2}u_{2j-1}^{s-1}f\left(-\ln\left(u_{2j-1}\right)\right)+u_n^{s-1}f\left(-\ln\left(u_n\right)\right)\right], \tag{27}$$

where $n$ is the number of subintervals, $h=\frac{1-\epsilon}{n}$, and $\epsilon$ is the machine epsilon at the precision level.

Our C++ software implementation of the numerical Laplace transform is based on Equation (27). The following improvements were made to speed up the calculations. Notice that only the powers of $u$ depend on $s$ in the Equation (27). As such, the function evaluations, $f\left(-\ln\left(u_j\right)\right)$, need not be evaluated every time a new $s$ value is calculated. This is especially useful in the double transformation calculations because each evaluation of the function $\hat{f}(t)$ is the numerical inversion of $C(s)$ at some point $t$. Depending on the precision level, this can be an extremely time consuming step. The implementation of the method is shown in Appendix.

**Example 8.** The numerical Laplace transform for the Bessel function $J_0(t)$ is plotted in **Figure 11**. We used a composite Simpson's Rule calculation with up to 25,000 subintervals to ensure high accuracy. Comparing Laplace transform

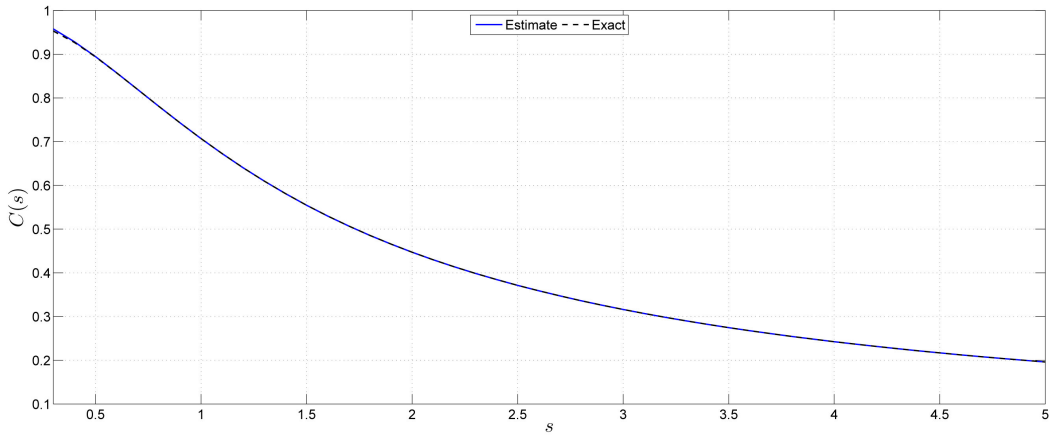



**Figure 11.** Given $f(t)=\sum_{n=0}^{\infty}\frac{(-1)^n t^{2n}}{2^{2n}(n!)^n}$. Compute $\tilde{C}(s)=\mathcal{L}\{f(t)\}$. The determining function is $C(s)=\frac{1}{\sqrt{1+s^2}}$.

$C(s)=\mathcal{L}\{J_0(t)\}$ and the exact solution $C(s)=\frac{1}{\sqrt{1+s^2}}$ for Bessel function $J_0(t)$ is shown in Table 5.

**Example 9.** Consider the following Laplace transform example

$$f(t)=\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}},\quad \text{and}\quad C(s)=\frac{e^{-1/s}}{s^{3/2}} \tag{28}$$

Numerical Laplace transform results are shown in Figure 12 and Table 6.

## 8. Validation of the Numerical Inversion Using Double Transformation Technique

**Example 10.** The first computation presented in this section is the theoretical error for Numerical Laplace transform of the function

$$f(t)=e^{-2t},\quad \text{with}\quad C(s)=\frac{1}{s+2}, \tag{29}$$

where $C(s)$ is the exact Laplace transform solution. Let the number of subintervals $n=2m=256$. From the Equation (26) we have

$$E_n\le\frac{1}{180\times 256^4}\max\left|f^{(4)}(u)\right| \tag{30}$$

The integral $\int_0^1 u^{s-1}f\left(-\ln(u)\right)\mathrm{d}u$ is used, where $u=e^{-t}$. Now $f(u)=u^2$. This yields

$$\int_0^1 u^{s-1}f\left(-\ln(u)\right)\mathrm{d}u=\int_0^1 u^{s-1}u^2\mathrm{d}u=\int_0^1 u^{s+1}\mathrm{d}u \tag{31}$$

The integrand $f(u)=u^{s+1}$ has four continuous derivatives and $f^{(4)}(u)=s(s-1)(s+1)(s-2)u^{s-3}$.

Because $0\le u\le 1$, then for example if $s=0.7$,

$$E_n\le\frac{1}{180\times 256^4}\max\left|f^{(4)}(u)\right|=\frac{1}{180\times 256^4}\times 0.464\approx 6\times 10^{-13}$$

**Table 5.** Comparison of the Laplace transform $\tilde{C}(s)=\mathcal{L}\{J_0(t)\}$ with the exact solution $C(s)=\frac{1}{\sqrt{1+s^2}}$ for Bessel function $J_0(t)$. Values 5.26e−2 $\equiv 5.26\times10^{-2}$.

| $s$ | $\mathcal{L}\{J_0(t)\}$ | $\frac{1}{\sqrt{1+s^2}}$ | Error |
|---|---|---|---|
| 0.1 | 0.942376 | 0.995037 | 5.266103e−2 |
| 0.2 | 0.964656 | 0.980581 | 1.592439e−2 |
| 0.3 | 0.953056 | 0.957826 | 4.770068e−3 |
| 1 | 0.707105 | 0.707107 | 1.438876e−06 |
| 2 | 0.447214 | 0.447214 | 3.906561e−11 |
| 3 | 0.316228 | 0.316228 | 1.001639e−13 |
| 4 | 0.242536 | 0.242536 | 9.999995e−14 |
| 5 | 0.196116 | 0.196116 | 9.999975e−14 |

**Table 6.** Comparison of the Laplace transform $\tilde{C}(s)=\mathcal{L}\left(\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}\right)$ with the exact solution $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. Values 5.62E−6 $\equiv 5.62\times10^{-6}$.

| $s$ | $\mathcal{L}\left(\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}\right)$ | $\frac{e^{-1/s}}{s^{3/2}}$ | Error |
|---|---|---|---|
| 1 | 3.678744e−1 | 3.678794e−1 | 5.625486e−06 |
| 2 | 2.144409e−1 | 2.144410e−1 | 2.318805e−08 |
| 10 | 2.861345e−2 | 2.861347e−2 | 2.317171e−08 |
| 30 | 5.886267e−3 | 5.886290e−3 | 2.317492e−08 |
| 50 | 2.772397e−3 | 2.772421e−3 | 2.317815e−08 |
| 100 | 1.719596e−3 | 1.719619e−3 | 2.318121e−08 |
| 120 | 7.543895e−4 | 7.544127e−4 | 2.318945e−08 |
| 140 | 5.993618e−4 | 5.993850e−4 | 2.319269e−08 |

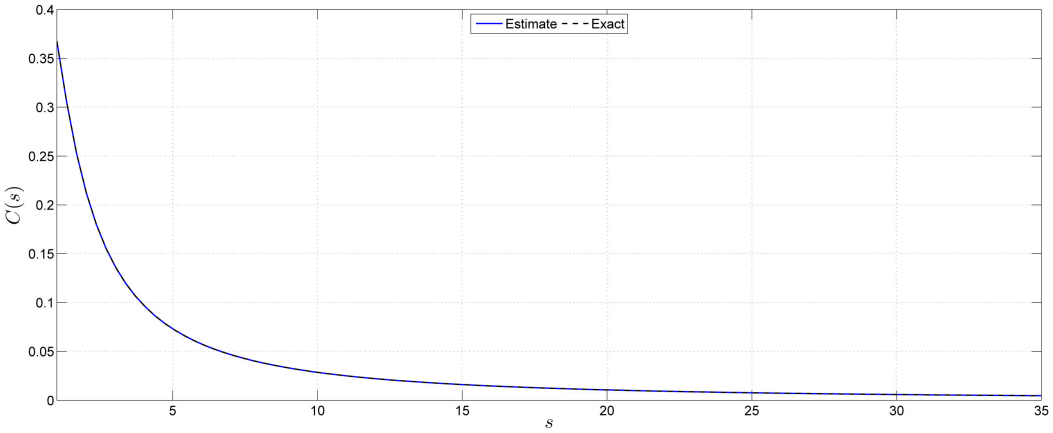



**Figure 12.** Given $f(t)=\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}$. Compute $\tilde{C}(s)=\mathcal{L}\left(\frac{\sin\left(2\sqrt{t}\right)}{\sqrt{\pi}}\right)$. The determining function is $C(s)=\frac{e^{-1/s}}{s^{3/2}}$.

Next we compare this theoretical error bound result with the numerical error obtaining by two step double transformation technique, on

$$\tilde{\tilde{C}}(s)=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{1}{s+2}\right\}\right\} \tag{32}$$

Figure 13 displays the numerical error $\varepsilon_N(s)$ in varying precision levels for some range of $s$ beginning at very low values. From Figure 13 it is clear that increasing the decimal precision greatly increases the accuracy of the estimates for low values of $t$. Each time $N$ is doubled, $\varepsilon_N(1)$ and $\varepsilon_N(2)$ are nearly halved. Examining lower values of $s$ reveals that the difference between the various $\varepsilon_N(s)$ is not as pronounced as for $s=1$ or $s=2$. This suggests that increasing $N$ does not have a very significant impact on the double transformation accuracy for higher values of $t$. As such, depending on the needs of the application, one might stop increasing precision past $N=64$ since the double transformation errors are not showing a worthwhile improvement for the large increase in computational cost.

The error at $s\approx 0.7\left(\log_2 s=-0.5\right)$ for $N=250$ subintervals. For the precision level 64 the error is at the order of $10^{-17}$, and for the precision level 128 the accuracy is much better, at the order $10^{-30}$.

For the precision levels 16 and 32 the accuracy are at the order $10^{-7}$ and $10^{-12}$ respectively. For the precision levels 16 and 32, the accuracy of the answer has deteriorated due to roundoff error. Note that this error of the inverse Laplace transform is ignored in the first step of the calculation as it is much smaller than suggested by the composite Simpson's rule in the second step of the double transformation algorithm.

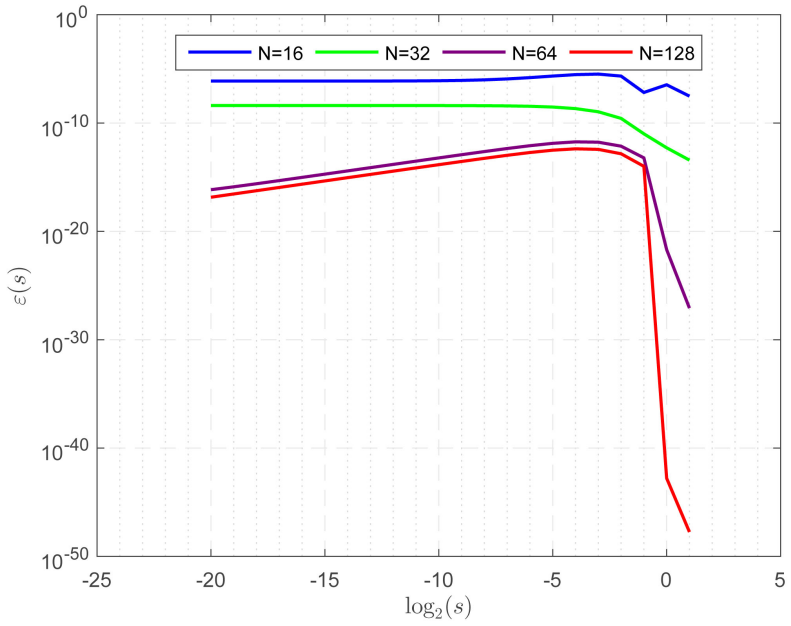



**Figure 13.** Numerical error $\varepsilon_N(s)=\left|C(s)-\tilde{\tilde{C}}_N(s)\right|$ for double transformation $\tilde{\tilde{C}}_N(s)=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{1}{s+2}\right\}\right\}$ in varying precision levels.

**Example 11.** This example illustrates two steps of the numerical double transformation calculation

$$\tilde{\tilde{C}}(s)=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{e^{-1/s}}{s^{3/2}}\right\}\right\} \tag{33}$$

Figure 14 depicts $\varepsilon_N(s)$ in varying precision levels for some range of $s$ from the very low values.

**Example 12.** Numerical double transformation for Bessel function of order 0 is given in Figure 15 and Table 7. The plots in Figure 11 and Figure 15 are indistinguishable. High accuracy is borne out by comparison of the approximation with the exact solution.

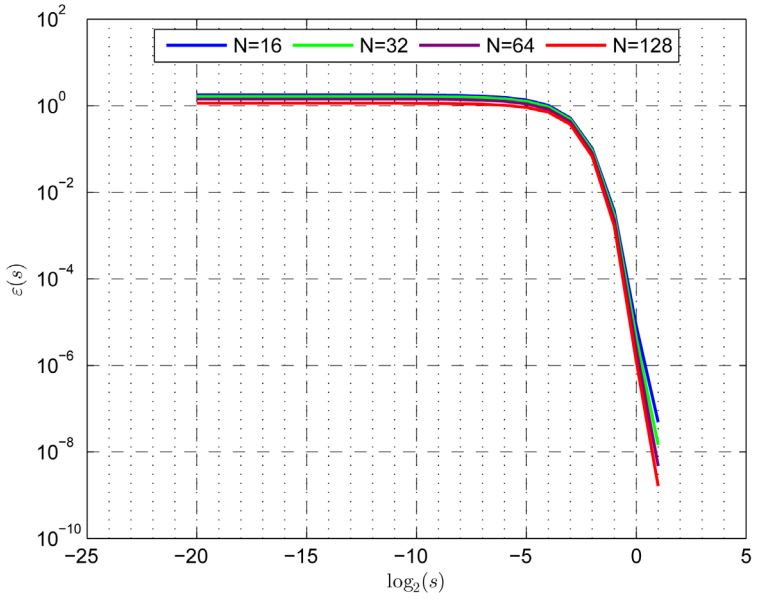



Figure 14. Numerical error $\varepsilon_N(s)=\left|C(s)-\tilde{\tilde{C}}_N(s)\right|$ for double transformation $\tilde{\tilde{C}}_N(s)=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{e^{-1/s}}{s^{3/2}}\right\}\right\}$ in varying precision levels.

Table 7. Comparison of $\tilde{\tilde{C}}(s)=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{C(s)\right\}\right\}$ for Bessel function in Table 1 with the exact solution $C(s)=\frac{1}{\sqrt{1+s^2}}$. Values 7.3e−3 $\equiv 7.3\times10^{-3}$.

| $s$ | $\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{1}{\sqrt{1+s^2}}\right\}\right\}$ | $\frac{1}{\sqrt{1+s^2}}$ | Error |
|---|---|---|---|
| 0.1 | 0.987647 | 0.995037 | 7.390034e−3 |
| 0.2 | 0.979209 | 0.980581 | 1.371815e−3 |
| 0.3 | 0.957524 | 0.957826 | 3.018401e−4 |
| 1 | 0.707107 | 0.707107 | 3.477729e−07 |
| 2 | 0.447214 | 0.447214 | 5.855865e−07 |
| 3 | 0.316228 | 0.316228 | 1.241552e−07 |
| 4 | 0.242536 | 0.242536 | 4.256322e−08 |
| 5 | 0.196116 | 0.196116 | 1.135660e−08 |

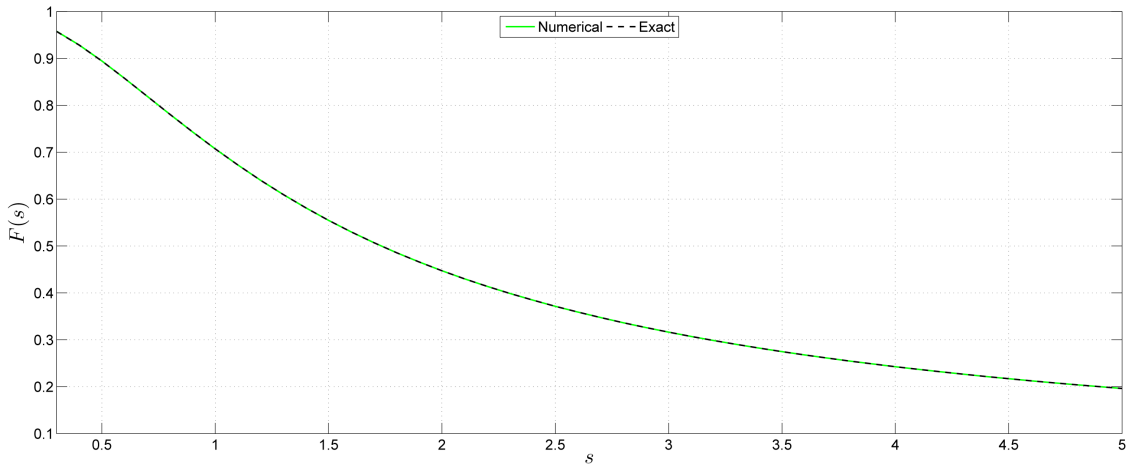



**Figure 15.** Given $C(s)=\frac{1}{\sqrt{1+s^2}}$. Compute $\tilde{\tilde{C}}=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{C(s)\right\}\right\}$.

**Example 13.** Numerical double transformation results for the function $C(s)=\frac{e^{-1/s}}{s^{3/2}}$ are given in Figure 16 and in Table 8. The plots in Figure 12 and Figure 16 are nearly identical. As can be seen from Table 6 and Table 8, the accuracy is very high.

Consider again double transformation results for the function $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. Let $s=2\left(\log_2 s=1\right)$ and the precision level $N=32$. The results illustrated in the four plots (Figure 17) correspond to different number of subintervals $n$ in composite Simpson's rule to approximate the Laplace transform. The error estimations are at the order $10^{-5},10^{-6},10^{-7}$ and $10^{-8}$ respectively with $n$ = 32, 64, 128 and 512 subintervals.

## 9. Comparison of Running Time in Arbitrary Precision Calculations

The execution time for the test functions is shown in Table 1. The direct Laplace transform and inverse Laplace transform are computed using different precision levels $N$. The number of subintervals used in the direct Laplace transform calculations is $L=512$. Table 9 gives the relative CPU time of the numerical solutions. All times are relative runs performing in double precision (16 digits). Computer configuration used: AMD 8350 8-Core processor 4 GHz, 8 GB RAM, 240 GB SDD, Windows 10 Pro.

Let $N$ be the precision level. We approximate the relative CPU time $t(N)$ for inverse and direct Laplace transform algorithms by the polynomial $p(N)$ of degree 6. There are seven coefficients and the polynomial is:

$$p(N)=p_1*x^6+p_2)*x^5+p_3*x^4+p_4*x^3+p_5*x^2+p_6*x+p_7 \qquad (34)$$

For inverse Laplace transform algorithm, the coefficients in the polynomial $p(N)$ of degree 6 are:

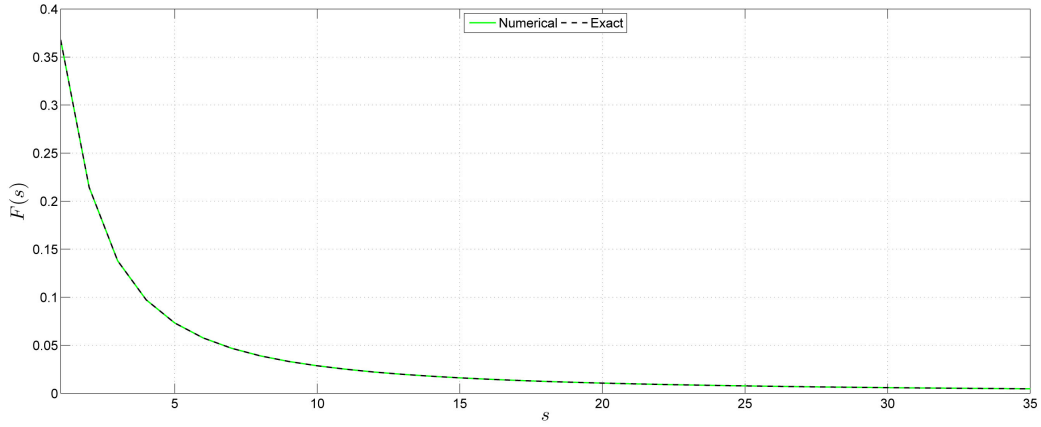



**Figure 16.** Given $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. Compute $\tilde{\tilde{C}}=\mathcal{L}\left\{\mathcal{L}^{-1}\left\{C(s)\right\}\right\}$.

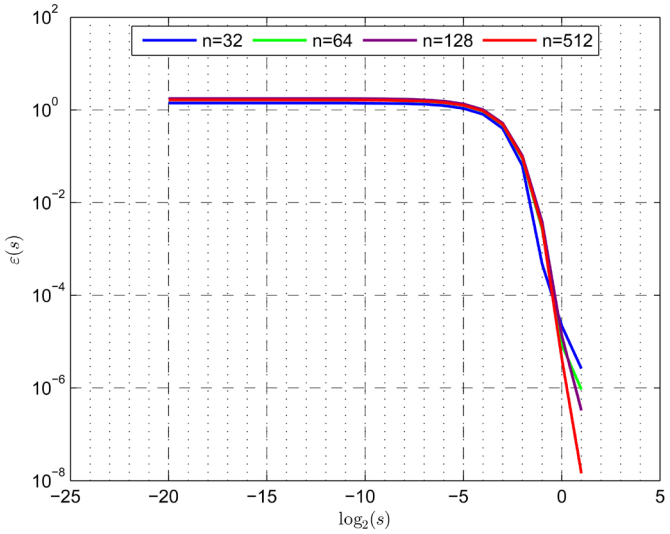



**Figure 17.** Given $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. The precision level $N=32$. Compute the error estimations for different number of subintervals with $n$ = 32, 64, 128 and 512 in composite Simpson's rule to approximate the Laplace transform.

**Table 8.** Comparison of $\tilde{C}(s)=$ with original Laplace transform $C(s)=\frac{e^{-1/s}}{s^{3/2}}$. Values 4.3e−07 $\equiv 4.3\times10^{-7}$.

| $s$ | $\mathcal{L}\left\{\mathcal{L}^{-1}\left\{\frac{e^{-1/s}}{s^{3/2}}\right\}\right\}$ | $\frac{e^{-1/s}}{s^{3/2}}$ | Error |
|---|---|---|---|
| 1 | 3.678790e−1 | 3.678794e−1 | 4.315168e−07 |
| 2 | 2.144410e−1 | 2.144410e−1 | 8.855806e−11 |
| 10 | 2.861347e−2 | 2.861347e−2 | 9.927583e−11 |
| 30 | 5.886290e−3 | 5.886290e−3 | 1.452239e−10 |
| 50 | 2.772420e−3 | 2.772420e−3 | 1.461473e−10 |
| 100 | 9.900497e−4 | 9.900498e−4 | 1.436632e−10 |
| 120 | 7.544126e−4 | 7.544127e−4 | 1.431769e−10 |
| 140 | 5.993848e−4 | 5.993850e−4 | 1.4283671e−10 |

**Table 9.** Relative CPU time for inverse and direct Laplace transform algorithms as a function of the precision level. All times are relative to run in double precision (16 digits).

| Precision Level ($N$) | 16 | 32 | 64 | 128 | 256 | 512 | 750 | 1000 |
|---|---|---|---|---|---|---|---|---|
| Inverse Laplace Transform | 1 | 3 | 25 | 200 | 1.80e3 | 1.91e4 | 7.95e4 | 2.17e5 |
| Direct Laplace Transform | 1 | 1.65 | 2.04 | 4.52 | 10.25 | 22.07 | 37.33 | 70.51 |

$$\begin{aligned} p(6) &= (p_1, p_2, p_3, p_4, p_5, p_6, p_7) \\ &= (-4.673630e-13, 9.235484e-10, -4.371448e-07, \\ &\quad 2.108954e-04, -1.350226e-02, 5.725313e-01, -6.348896e+00). \end{aligned}$$

For the direct Laplace transform, the coefficients in the approximation polynomial $p(N)$ of degree 6 are:

$$\begin{aligned} p(6) &= (p_1, p_2, p_3, p_4, p_5, p_6, p_7) \\ &= (-6.42651e-17, 2.43029e-14, 3.199151e-10, \\ &\quad -4.079160e-07, 1.878244e-04, 9.7309e-03, 9.178160e-01). \end{aligned}$$

In Figure 18 we plotted relative CPU time for inverse Laplace transform and for direct Laplace transform as a function of the precision level $N$.

## 10. Conclusion

Laplace Transform applications often require high accuracy beyond IEEE double precision. Common situations involve calculations that are numerically unstable, and even double-double precision is not sufficient to reach the necessary accuracy. Roundoff and underflow/overflow errors that occur during the computations can cause severe stability problems. There are several numerical inverse Laplace transform methods, each successful in some fields. The problem is to find methods successful in stability, accuracy and computational efficiency. Overall, the presented double transformation approach provides an effective way to compare the effectiveness of numerical inversion methods. In order to validate and improve the inversion solution, the numerical direct Laplace transform are used for this inversion, and compared it with the original Laplace transform. We implemented the composite Simpson's Rule for direct Laplace transform, and investigate the role that extended precision can play obtaining accurate transform inversions. The high precision approach seems to be an effective way to handle the challenging problems dealing with periodic functions. We demonstrated that the level of precision chosen must match algorithms properly. In the Gaver-Stehfest algorithm the balance is between the truncation error and roundoff error. High precision in an inaccurate algorithm yield little benefit, while a potentially highly accurate algorithm may be defeated by roundoff error if inadequate accuracy is used.

## Acknowledgements

Matt Davison thanks to Natural Sciences and Engineering Research Council of

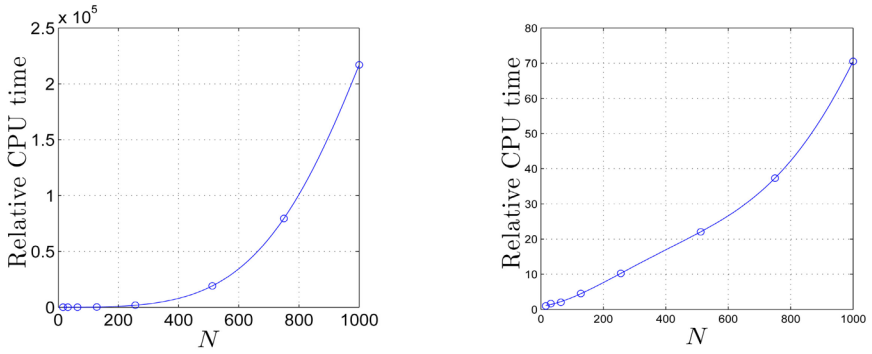



**Figure 18.** Relative CPU time for inverse Laplace transform (left) and for direct Laplace transform (right) as a function of the precision level $N$. All times are relative to run in double precision (16 digits).

Canada for research funding. We are also grateful to the anonymous referees for useful suggestions which improved the present work.

## References

[1] Cengel, Y. and Palm, W. (2013) Differential Equations for Engineers and Scientists. McGraw-Hill, New York.

[2] Edwards, C. and Penney, C. (2015) Differential Equations and Boundary Value Problems. 5th Edition, Pearson, London.

[3] Polyanin, A.D. (2016) Handbook of Linear Partial Differential Equations for Engineers and Scientists. 2nd Edition, Chapman and Hall/CRC, UK. https://doi.org/10.1201/b19056

[4] Cohen, A.M. (2007) Numerical Methods for Laplace Transform Inversion. Springer, Berlin.

[5] Kreyszig, E. (2011) Advanced Engineering Mathematics. 10th Edition, Wiley, Hoboken.

[6] Abate, J., Chudhuru, G. and Whitt, W. (2000) An Introduction to Numerical Transform Inversion and Its Application to Probability Models. In: Grassmann, W., Ed., *Computational Probability*, Springer, Berlin, 257-323.

[7] Badescu, A., Breuer, L., Da Silva Soares, A., Latouche, G., Remiche, M.A. and Stanford, D.A. (2005) Risk Processes Analyzed as Fluid Queues. *Scandinavian Actuarial Journal*, **2005**, 127-141. https://doi.org/10.1080/03461230410000565

[8] Kao, E. (1997) An Introduction to Stochastic Processes. Duxbury Press, Pacific Grove, CA.

[9] Nadarajah, S. and Kotz, S. (2006) On the Laplace transform of the Pareto Distribution. *Queueing Systems*, **54**, 243-244. https://doi.org/10.1007/s11134-006-0299-1

[10] Davison, M. and Doeschl, A. (2004) A Hyperbolic PDE with Parabolic Behavior. *SIAM Reviev*, **46**, 115-127. https://doi.org/10.1137/S0036144502409007

[11] Cruz-Baez, D.I. and Gonazalex-Rodriguez, J.M. (2008) A Different Approach for Pricing Asian Options. *Applied Mathematics Letters*, **21**, 303-306.

[12] Deakin, A. and Davison, M. (2010) An Analytic Solution for a Vasicek Interest Rate Convertible Bond Model. *Journal of Applied Mathematics*, **2010**, Article ID: 263451. https://doi.org/10.1155/2010/263451

[13] Fusai, G. and Roncoroni, A. (2008) Implementing Models in Quantitative Finance: Methods and Cases. In: Avellaneda, M., Barone-Adesi, G., Broadie, M., Davis, M.,

Derman, E., Klüppelberg, C. and Schachermayer, W., Eds., *Springer Finance*, Springer-Verlag, Berlin, Heidelberg.

[14] Kuhlman, K.L. (2013) Review of Inverse Laplace Transform Algorithms for Laplace-Space Numerical Approaches. *Numerical Algorithms*, **63**, 339-355. https://doi.org/10.1007/s11075-012-9625-3

[15] Stehfest, H. (1970) Algorithm 368: Numerical Inversion of Laplace Transform. *Communications of the ACM*, **13**, 47-49. https://doi.org/10.1145/361953.361969

[16] Kuznetsov, A. (2013) On the Convergence of the Gaver—Stehfest Algorithm. *SIAM Journal of Numerical Analysis*, **51**, 2984-2998. https://doi.org/10.1137/13091974X

[17] Abate, A. and Valko, P. (2004) Multi-Precision Laplace Transform Inversion. *International Journal for Numerical Methods in Engineering*, **60**, 979-993. https://doi.org/10.1002/nme.995

[18] Duffy, D.G. (2015) Green's Functions with Applications. 2nd Edition, CRC Press, Boca Raton, FL. https://doi.org/10.1201/b18159

[19] Davies, B. and Martin, B. (1979) Numerical Inversion of the Laplace Transform, a Survey and Comparison of Methods. *Journal of Computational Physics*, **33**, 1-32.

[20] Murli, A. and Rizzardi, M. (1990) Algorithm 682: Talbot's Method for the Laplace Inversion Problem. *ACM Transactions on Mathematical Software*, **16**, 158-168. https://doi.org/10.1145/78928.78932

[21] Bailey, D.H., Hida, Y., Li, X.S and Thompson, B. (2002) ARPREC: An Arbitrary Precision Computation Package. Tech. Rep. LBNL-53651, Lawrence Berkley National Lab.

[22] Krougly, Z.L. and Jeffrey, D.J. (2009) Implementation and Application of Extended Precision in Matlab. *Proceedings of the* 11*th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering*, Athens, 28-30 September 2009, 103-108.

[23] Krougly, Z.L., Jeffrey, D.J. and Tsarapkina, D. (2013) Software Implementation of Numerical Algorithms in Arbitrary Precision. 15*th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (*SYNASC*), Timisoara, 23-26 September 2013, 131-137.

[24] Abramowitz, M. and Stegun, I.A. (1964) Handbook of Mathematical Functions. National Bureau of Standards.

[25] Atkinson, K.E. (1989) An Introduction to Numerical Analysis. 2nd Edition, John Wiley & Sons, Hoboken.

## Appendix: C++ Software Implementation of Numerical Laplace and Inverse Laplace Transform in Arbitrary Precision

The Gaver-Stehfest algorithm of inverse Laplace transform was implemented in multiple precision as described in Section 4. Numerical direct Laplace transform was implemented in multiple precision by the composite Simpson's Rule, as covered in details in Section 7. The calculations can be performed in C++ up to 1000 places of decimals/1000 significant digits.

The C++ code of the inverse Laplace transform and direct Laplace transform are given below.

**1. C++ implementation of the Gaver-Stehfest algorithm for inverse Laplace transform**

```
// CLASS LaplaceInverter ------------------------------------------------------
class LaplaceInverter {
private:
   mp_real *V;
   double L;
public:
   LaplaceInverter(int L_in); // Constructor
   mp_real evaluate(mp_real(*F)(mp_real), mp_real t); // Function to evaluate inverse Laplace transform at t
};

// CONSTRUCTOR ----------------------------------------------------------------
LaplaceInverter::LaplaceInverter(int L_in) { // Number of terms in Gaver-Stehfest summation
   // Define properties
   L = L_in;
   V = new mp_real[L_in];

   int nn2 = L / 2;
   // Populate coefficient array for arbitrary precision
   for (int n = 1; n <= L; n++){
      mp_real z = mp_real(0.0);
      for (int k = (int)((n + 1) / 2); k <= min(n, nn2); k++){
         z += ((pow((mp_real)k, nn2))*arfact(2 * k))/  (arfact(nn2 - k)*arfact(k)*arfact(k - 1)*arfact(n - k)*
                  arfact(2 * k - n));
      }
      V[n - 1] = pow(-1.0, (n + nn2)) * z;
   }
}

// MEMBER FUNCTION ------------------------------------------------------
mp_real LaplaceInverter::evaluate(mp_real(*F)(mp_real), mp_real t) {
   mp_real sum = mp_real(0.0);
   mp_real _log2_t = _log2 / t;
   for (int n = 1; n <= L; n++)
   {
      sum += V[n - 1] * F((mp_real)n * _log2_t);
   }
   return sum * _log2_t;
}
```

**2. C++ implementation of the composite Simpson's Rule for direct Laplace transform**

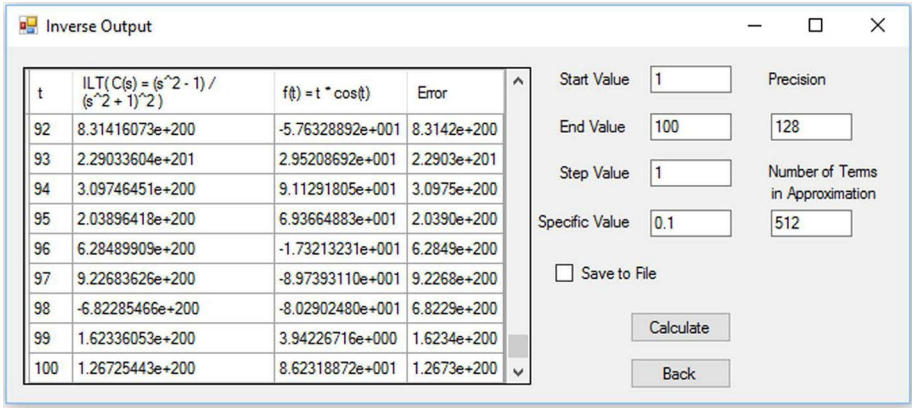
```
// DECLARATION ----------------------------------------------------------------
int num_nodes = 25000;
mp_real *U_t = new mp_real[num_nodes + 1]; // Vector of u values
mp_real *F_u = new mp_real[num_nodes + 1]; // Vector of f(u) values
mp_real h = (1 - _eps) / num_nodes; // _eps has been pre-defined on a global scope
// Pre-calculate the function evaluation to speed up the calculations
for (int j = 1; j <= num_nodes; j++) {
   U_t[j] = 0 + j*h;
   F_u[j] = f(-log(U_t[j]));
}

// IMPLEMENTATION -------------------------------------------------------------
```

```
mp_real LaplaceTransform(mp_real *U_t, // n+1 length vector of u values. First element is 0.
             mp_real *F_u, // n+1 length vector of f(u) values. E.g. F_u[1] = F(U_t[1])
             mp_real s, // Value of s to evaluate numerical Laplace transform for
             int n){ // Must be one less than length of U_t & F_u.
   mp_real sum = mp_real(0.0); // Initialization
   mp_real h = (1 - _eps) / n; // Define subinterval length
   for (int j = 2; j < n; j += 2) {
      sum += 2 * pow(U_t[j], s - 1)*F_u[j]; // Add even terms
   }
   for (int j = 1; j < n; j += 2) {
      sum += 4 * pow(U_t[j], s - 1)*F_u[j]; // Add odd terms
   }
   sum += pow(U_t[n], s - 1)*F_u[n]; // Add last term
   return h / 3 * sum; // Return numerical Laplace transform estimate
}
// EXAMPLE ---------------------------------------------------------------
mp_real s = mp_real(0.5);
mp_real result = LaplaceTransform(U_t, F_u, s, num_nodes);
```

### 3. Demonstration the accuracy of Laplace transform inversions

This demonstration shows the role that extended precision can play in accuracy of Laplace transform inversions by six screenshots (Figures 19-21) corresponding to the following periodic function. Given $C(s)=\dfrac{s^2-1}{\left(s^2+1\right)^2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=t\cos(t)$.



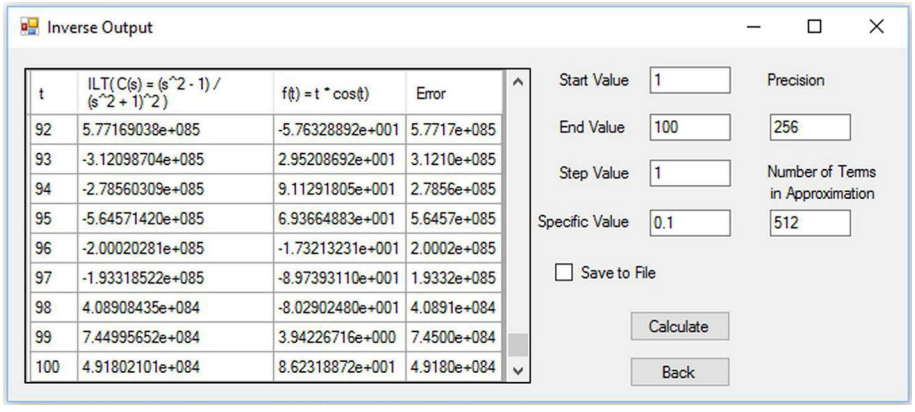




Figure 19. Given $C(s)=\dfrac{s^2-1}{\left(s^2+1\right)^2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=t\cos(t)$. Two screenshots are given for 512 number of terms and the precision level 128 and 256. We failed to get the solution.

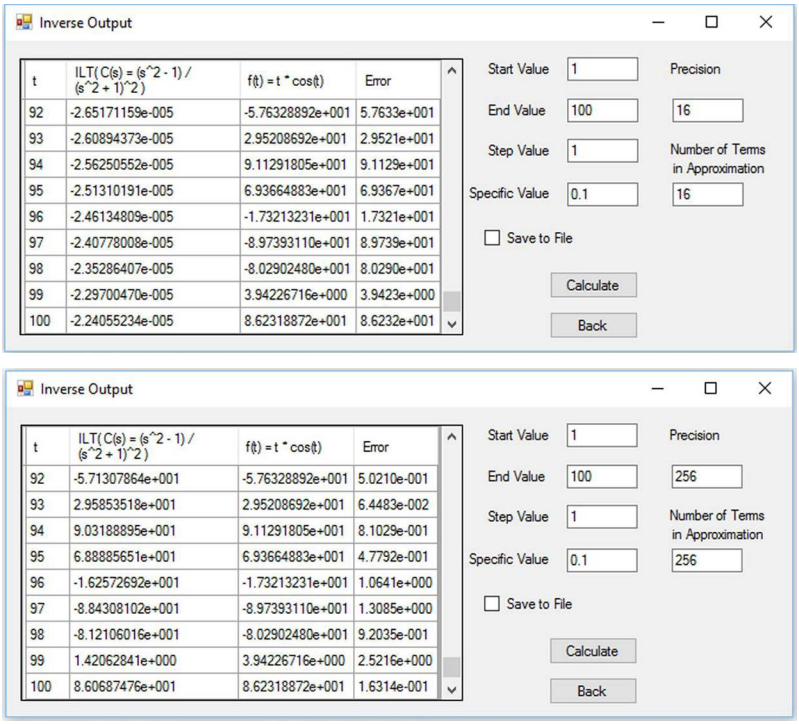



**Figure 20.** Given $C(s)=\dfrac{s^2-1}{\left(s^2+1\right)^2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=t\cos(t)$. Two screenshots are given for 16 and 256 precision levels.

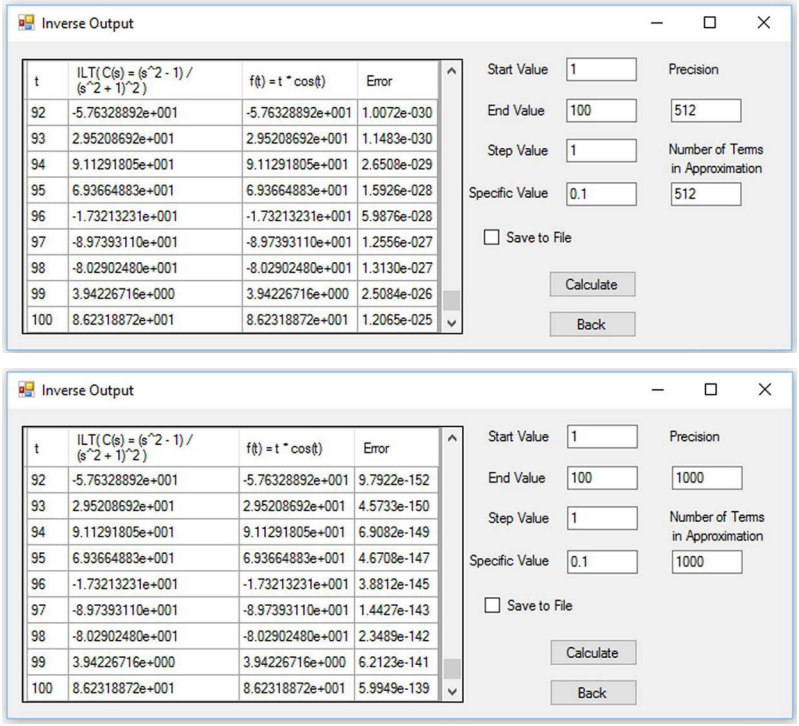



**Figure 21.** Given $C(s)=\dfrac{s^2-1}{\left(s^2+1\right)^2}$. Compute $\tilde{f}(t)=\mathcal{L}^{-1}\{C(s)\}$. The determining function is $f(t)=t\cos(t)$. Two screenshots are given for 512 and 1000 precision levels.

Previously Figure 4 illustrated a good approximation for $t$ values, up to 35, if the precision level $N = 128$.

Let increase $t$ up to 100. The results in Figure 19 leads to unstable solutions, for the number of expansion terms $L = 512$, if the precision level much smaller than $L$, as illustrated for $N = 128$ and 256. In this case, using too many terms causes rounding error to overtake the numerical solution, and we essentially obtain noise.

Next (Figure 20 and Figure 21) illustrate the solutions correspond to the precision level $N$ equals 16, 256, 512 and 1000. The same number of expansion terms $L$ used as the precision level $N$. The accuracy is very poor for $N$ is 16 and 256, at order at most roughly $10^{-2}$. We see significant improvements in accuracy as precision level increased to 512 and 1000. They are at order at least roughly $10^{-25}$ and $10^{-139}$ accordingly.