# Supplement to
# "Power Algorithms for Inverting Laplace Transforms"

Efstathios Avdis
Department of Industrial Engineering and Operations Research, Columbia University, New York, New York 10027-6699, USA, stathi.avdis@gmail.com

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

This is a supplement to the main paper, presenting additional experimental results not included in the main paper due to lack of space. The overall study investigates ways to create algorithms to numerically invert Laplace transforms within a unified framework proposed by Abate and Whitt (2006). That framework approximates the desired function value by a finite linear combination of transform values, depending on parameters called weights and nodes, which are initially left unspecified. Alternative parameter sets, and thus algorithms, are generated and evaluated here by considering power test functions. The resulting power algorithms are shown to be effective, with the parameter choice being tunable to the transform being inverted. The powers can be advantageously chosen from series expansions of the transform. Real weights for a real-variable power algorithm are found for specified real powers and positive real nodes by solving a system of linear equations involving a generalized Vandermonde matrix, using *Mathematica*. Experiments show that the power algorithms are robust in the nodes; it suffices to use the first $n$ positive integers. The power test functions also provide a useful way to evaluate the performance of other algorithms.

*History:* draft aiming for JoC. Last modified on December 9, 2006.

# Contents

# 1. Introduction

In the main paper we propose a new class of algorithms for numerically inverting Laplace transforms, called *power algorithms*, with parameters that are tunable to the transform being inverted. Our power algorithms are constructed using *power test functions* within a *unified framework* for constructing algorithms to numerically invert Laplace transforms proposed by Abate and Whitt (2006).

The framework approximates the function $f$ by a finite linear combination of transform values; specifically,

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

where $\alpha \equiv (\alpha_1, \ldots, \alpha_n)$ and $\omega \equiv (\omega_1, \ldots, \omega_n)$ are vectors of complex numbers, called *nodes* and *weights*, respectively. The nodes and weights do not necessarily depend on the transform $\hat{f}$ or the function argument $t$, but typically depend upon $n$. This is a framework rather than a single algorithm because the nodes and weights are initially left unspecified. We consider the case in which the function $f$ is real-valued and the weights and nodes are real, so that (1) applies directly in the real domain.

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

The framework (1) is exact, using real weights and nodes, for the $p^{\text{th}}$ power for $p \in \mathcal{P} \equiv \{p \in \mathbb{R} : p > -1\}$ if

$$\frac{1}{t}\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\left(\frac{\alpha_k}{t}\right)^{p+1}}\,\omega_k = t^p, \tag{2}$$

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

$$\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}}\,\omega_k = 1 \ . \tag{3}$$

We consider $n$ given (distinct) positive real nodes and $n$ distinct powers when we look for the $n$ weights. Then the power relations in (3) for the $n$ values of $p$ become a system of $n$ linear equations in $n$ unknowns, where the weights are the unknowns.

We can write the linear system in matrix form as

$$A\omega = b \equiv \left[\frac{1}{\Gamma(p_1+1)}, \ldots, \frac{1}{\Gamma(p_n+1)}\right]^T \ , \tag{4}$$

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

$$A \equiv A_{n,\mathcal{P}} \equiv \begin{bmatrix} \left(\frac{1}{\alpha_1}\right)^{p_1+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_1+1} \\ \left(\frac{1}{\alpha_1}\right)^{p_2+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_2+1} \\ \vdots & & \vdots \\ \left(\frac{1}{\alpha_1}\right)^{p_n+1} & \cdots & \left(\frac{1}{\alpha_n}\right)^{p_n+1} \end{bmatrix} \tag{5}$$

Fortunately, it is not difficult to solve the linear system $A\omega = b$ in (4) and (5) with *Mathematica* because the node matrix $A$ in (5) is a *generalized Vandermonde matrix* with positive real "points" $1/\alpha_k$ and "exponents" $p_k + 1$, $k = 1, \ldots, n$, with $p_k > -1$. Since these generalized Vandermonde matrices are notoriously ill-conditioned, we rely on the high precision of *Mathematica*.

Restricting attention to real-variable algorithms, we investigate possible node sets and possible power sets. We will be giving more details about those investigations here in this supplement. We find that the default node set $\mathcal{N}^* \equiv \{1, 2, \ldots, n\}$ is effective. We find no incentive to use other real node sets. In contrast, efficiency gains can be achieved by carefully considering the power set, as we show in Section 3 of the main paper. But we suggest the following power set as a default power set:

$$\mathcal{P}^* \equiv \mathcal{P}\left(-\frac{5j}{n}; \frac{k}{2}\right) \equiv \left\{-\frac{5j}{n} : 1 \le j \le \frac{n}{5} - 1\right\} \cup \left\{\frac{k}{2} : 0 \le k \le \frac{4n}{5}\right\} \ , \tag{6}$$

as in Step 3 in Figure 1.

**Organization of this Supplement.** We start in Section 2 be summarizing the algorithm to create power algorithms and the default power algorithm; i.e., we repeat Figure 1 from the main paper. Next, in Section 3, we list the Laplace transforms and their known inverses that we consider in this experimental study. They are mostly taken from Table 1 of Valko and Abate (2004).

In Section 4 we present additional material on choosing the powers. In Section 5 we present additonal material on choosing the nodes. In Section 6 we give additional detail about the default power weights. We show that the default power weights are similar to the Gaver-Stehfest weights. In Section 7 we discuss accuracy and precision.

In Section 8 we give additional information about Zakian's algorithm. In Section 9 we show how the power test functions can be used to evaluate existing algorithms $\mathcal{T}$, $\mathcal{E}$, $\mathcal{Z}$ and $\mathcal{G}$. We start by reviewing the structure of the nodes and weights for these other algorithms. The nodes are complex for $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{Z}$. In Section 10 we compare all the main inversion algorithms applied to the test functions in Section 3 over a range of function arguments. In Section 11 we discuss these numerical results. Finally, in Section 12 we discuss difficulties encountered with *Mathematica* in properly treating complex functions involving $\sqrt{s}$.

## 2. The Default Power Algorithm

In this section we repeat Figure 1 of the main paper, which summarizes the algorithm to create power inversion algorithms. It also specifies the default power algorithm, which is based on the default node set $\mathcal{N}^*$ and the default power set $\mathcal{P}^*$. Furthermore, we give the numerical values of the nodes and the weights for the base case $n = 30$ in table 1.

**Constructing A Real-Variable Power Inversion Algorithm**

1. Let the number of terms be $n$; e.g., $n = 30$.

2. Let the computer precision be $1.5n$; e.g., 45 significant digits.

3. Choose $n$ distinct real numbers $p > -1$ to form the power set $\mathcal{P}$; e.g., with an integer $n$ divisible by 5, let
$$\mathcal{P} = \mathcal{P}^* \equiv \mathcal{P}\left(-\frac{5j}{n}; \frac{k}{2}\right) \equiv \left\{-\frac{5j}{n} : 1 \leq j \leq \frac{n}{5} - 1\right\} \cup \left\{\frac{k}{2} : 0 \leq k \leq \frac{4n}{5}\right\} \ .$$

4. Choose $n$ distinct positive real numbers to serve as the node set $\mathcal{N}$; e.g.,
$$\mathcal{N} \equiv \{\alpha_k : 1 \leq k \leq n\} = \mathcal{N}^* \equiv \{1, \ldots, n\} \ .$$

5. Using the specified computer precision, solve the system of $n$ linear equations
$$\sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}} \omega_k = 1, \quad p \in \mathcal{P},$$
to obtain the $n$ real weights $\omega_1, \ldots, \omega_n$.

6. **Apply the created power inversion algorithm:** Using the computer precision, nodes and weights specified above, calculate the required transform values $\hat{f}(\alpha_k/t)$ and the weighted sum to obtain the real-variable numerical inversion
$$f_n(t) \equiv f_{n,\alpha,\omega}(t) = \frac{1}{t} \sum_{k=1}^{n} \omega_k \hat{f}\left(\frac{\alpha_k}{t}\right) \ .$$

Figure 1: Summary of the $(\mathcal{P}^*, \mathcal{N}^*)$ power algorithm.

Table 1: The nodes and weights of the default power algorithm $(\mathcal{P}^*, \mathcal{N}^*)$

| $k$ | $\alpha_k$ | $\omega_k$ |
|---|---|---|
| 1 | 1 | $-2.75248363738488651043256823709168274464977829e-13$ |
| 2 | 2 | $9.20645093009887617275157098068080118815800107e-7$ |
| 3 | 3 | $-1.69586066739768270489313481433687452763380669e-2$ |
| 4 | 4 | $2.96840624780697663383723630134017658774355531e1$ |
| 5 | 5 | $-1.27272469564568042787911496756761076789273671e4$ |
| 6 | 6 | $2.08011611751266511621086464984984205768018921e6$ |
| 7 | 7 | $-1.65675778556606368309332229648401768030717922e8$ |
| 8 | 8 | $7.48521249402337956613097957022842092940812989e9$ |
| 9 | 9 | $-2.12221838175513680715102491731147076398612972e11$ |
| 10 | 10 | $4.05290116724950279365530679075090290442765742e12$ |
| 11 | 11 | $-5.48956264084973653720704057905070525888456048e13$ |
| 12 | 12 | $5.48155095104990024537080179875025820690615774e14$ |
| 13 | 13 | $-4.15633755181301601131302840919914778900500071e15$ |
| 14 | 14 | $2.44852229905872661078229716626179618354688137e16$ |
| 15 | 15 | $-1.14079385506738046123244067984455278150438959e17$ |
| 16 | 16 | $4.26166377320308832022702693917573366748815190e17$ |
| 17 | 17 | $-1.28979231411823273080567708258207421725694197e18$ |
| 18 | 18 | $3.18617733295292344807571229058049006278936549e18$ |
| 19 | 19 | $-6.45553745889958591580246822937070819361079826e18$ |
| 20 | 20 | $1.07528497188725690271049372122289383916645564e19$ |
| 21 | 21 | $-1.47216876376562228294233658264363034472139148e19$ |
| 22 | 22 | $1.65177994616022494204123150444053914868052786e19$ |
| 23 | 23 | $-1.50948891221291777328432190631852972861972162e19$ |
| 24 | 24 | $1.11214425815903251021975827971606112258426328e19$ |
| 25 | 25 | $-6.50337979051705418381739977388051672879744064e18$ |
| 26 | 26 | $2.94750123102732477941355507576088336904017541e18$ |
| 27 | 27 | $-9.97950885353242690031983522450358760907704068e17$ |
| 28 | 28 | $2.37488003652140499897801060815372177354807782e17$ |
| 29 | 29 | $-3.54258304870155391524751812346204148321823780e16$ |
| 30 | 30 | $2.49172474102460812600448155993636194027580666e15$ |

# 3. Test Transforms

Table 2 shows the transforms we used to test our numerical inversion procedures. They all have known inverses. We started with transforms in Table 1 of Valko and Abate (2004). Many appear again here with the same indices. Transforms $\hat{f}_2$ and $\hat{f}_4$ from this list were also used by Abate and Whitt (2006). We then considered additional transforms. The new ones introduced here are $\hat{f}_9$, $\hat{f}_{12}$, $\hat{f}_{13}$ and $\hat{f}_{14}$.

Table 2: The transforms and their inverses

| Name | $\hat{f}(s)$ | $f(t)$ |
|---|---|---|
| $f_1$ | $\frac{1}{s^2+1}$ | $\sin(t)$ |
| $f_2$ | $\frac{1}{\sqrt{s}+\sqrt{s+1}}$ | $\frac{1-e^{-t}}{2\sqrt{\pi t^3}}$ |
| $f_3$ | $\frac{1}{\sqrt{s(s+2)}}$ | $e^{-t}I_0(t)$ |
| $f_4$ | $\frac{1}{s+\sqrt{s}}$ | $e^t\mathrm{erfc}(\sqrt{t})$ |
| $f_5$ | $e^{-2\sqrt{s}}$ | $\frac{e^{-1/t}}{\sqrt{\pi t^3}}$ |
| $f_6$ | $\frac{e^{-\sqrt{s}}\cos(\sqrt{s})}{\sqrt{s}}$ | $\frac{\cos(1/2t)}{\sqrt{\pi t}}$ |
| $f_7$ | $\frac{e^{-1/s}}{\sqrt{s^3}}$ | $\frac{\sin(2\sqrt{t})}{\sqrt{\pi}}$ |
| $f_8$ | $\frac{\log(s)}{s}$ | $\log(t)+\gamma$ |
| $f_9$ | $\frac{1}{s+1}$ | $e^{-t}$ |
| $f_{10}$ | $\frac{1}{2}\log^2(s)$ | $\frac{\log(t)+\gamma}{t}$ |
| $f_{11}$ | $s^3\log(s)$ | $\frac{6}{t^4}$ |
| $f_{12}$ | $\frac{s}{s^2+1}$ | $\cos(t)$ |
| $f_{13}$ | $\hat{f}_4(s)+\frac{\Gamma(1/2)}{\sqrt{s}}$ | $f_4(t)+\frac{1}{\sqrt{t}}$ |
| $f_{14}$ | $\frac{\Gamma(5/4)}{s^{5/4}}+\frac{\Gamma(11/4)}{s^{11/4}}$ | $t^{1/4}+t^{7/4}$ |

# 4. Choosing the Powers

In this section we present additional material related to our investigation into the choice of the power set. As indicated in Section 3 of the main paper, we considered the following 6 alternatives for the power set, each containing $n = 30$ powers:

1. Nonnegative integers: $\mathcal{P}(k) \equiv \{k : 0 \le k \le n-1\}$

2. Nonnegative even integers: $\mathcal{P}(2k) \equiv \{2k : 0 \le k \le n-1\}$

3. Nonnegative odd integers: $\mathcal{P}(2k+1) \equiv \{2k+1 : 0 \le k \le n-1\}$

4. Nonnegative integer multiples of 1/2: $\mathcal{P}(k/2) \equiv \{k/2 : 0 \le k \le n-1\}$

5. All integer multiples of 1/2: $\mathcal{P}((k-1)/2) \equiv \{k/2 : -1 \le k \le n-2\}$

6. The default power set with $n/5$ negative fractional powers and $4n/5$ nonnegative integer multiples of 1/2, i.e., $\mathcal{P}^*$ in (6) .

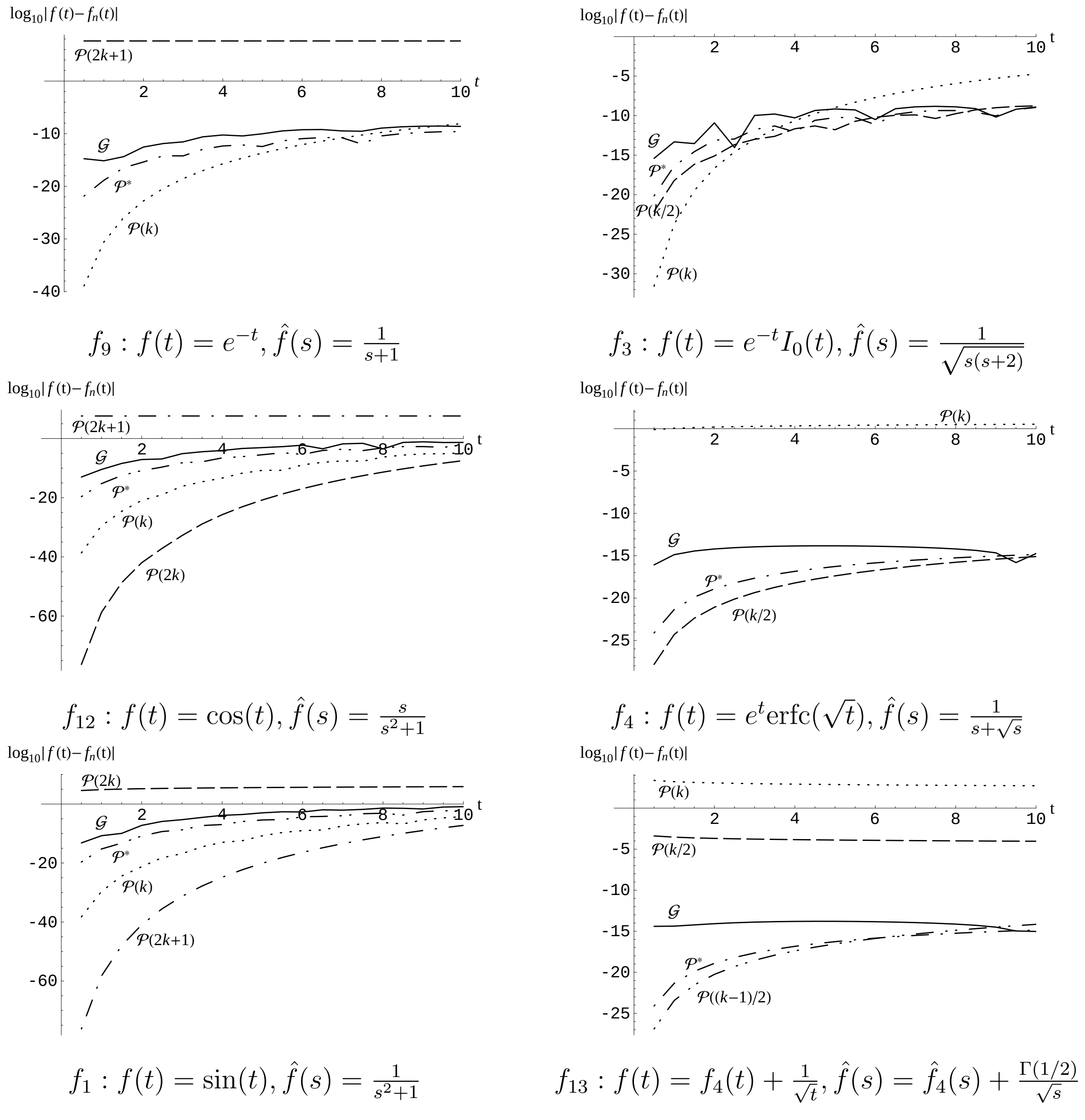
We repeat Figure 2 in the main paper here in Figure 2. It displays the base-10 logarithm of the absolute error $|f(t) - f_n(t)|$ between the function $f$ and the numerical inversion $f_n$ as a function of $t$, $0.5 \le t \le 10.0$, for the six different transforms $\hat{f}$ with known inverses $f$. For clarity, we do not show the performance of all power sets in each plot.

We also conducted several related investigations not shown in the main paper. In each case we use $n = 30$ and the default node set $\mathcal{N}^*$. First, we found that the Gaver-Stehfest algorithm outperformed the six power algorithms above for the transform $\hat{f}_5$, as shown in Figure 4. So we considered alternative power sets. Figure 4 shows the inversion results with new specially constructed power sets. Figure 4 shows that these alternative power algorithms do outperform Gaver-Stehfest.

We found that negative powers greatly help inverting $\hat{f}_5$. The set “special” has two powers in $(-1, 0]$, i.e., it is the set $\{-\frac{9}{10}, -\frac{1}{2}, 0, \frac{1}{2}, 1, \ldots, \frac{27}{2}\}$. The set “special 2” has one third of the powers in $(-1, 0]$, $\{-\frac{9}{10}, -\frac{8}{10}, \ldots, -\frac{1}{10}, 0, \frac{1}{2}, 1, \ldots, 10\}$. The set “special 3” has just as many negative powers as positive, $\{-\frac{14}{15}, \ldots, -\frac{1}{15}, 0, \frac{1}{2}, 1, \ldots, \frac{15}{2}\}$. Finally, the set “special 4”, which gives the best performance, has all 30 powers in the interval $(-1, 0]$, $\{-\frac{29}{30}, \ldots, -\frac{1}{30}, 0\}$.

We can give a partial explanation. The function $f_5(t)$ does not have a series expansion about $t = 0$. Instead, it has an integral representation. Indeed, from the dual pair of functions $f_5$ and $f_7$, we can immediately identify the spectral representation. Since the transform $\hat{f}_7$, as a function of $t$, coincides with the function $f_5(t)$, we know that the function $f_5(t)$ has the integral representation

$$f_5(t) = \int_0^\infty e^{-xt} \frac{\sin(2\sqrt{t})}{\sqrt{\pi}}\, dt \ .$$



$f_9 : f(t) = e^{-t}, \hat{f}(s) = \frac{1}{s+1}$

$f_3 : f(t) = e^{-t} I_0(t), \hat{f}(s) = \frac{1}{\sqrt{s(s+2)}}$

$f_{12} : f(t) = \cos(t), \hat{f}(s) = \frac{s}{s^2+1}$

$f_4 : f(t) = e^t \mathrm{erfc}(\sqrt{t}), \hat{f}(s) = \frac{1}{s+\sqrt{s}}$

$f_1 : f(t) = \sin(t), \hat{f}(s) = \frac{1}{s^2+1}$

$f_{13} : f(t) = f_4(t) + \frac{1}{\sqrt{t}}, \hat{f}(s) = \hat{f}_4(s) + \frac{\Gamma(1/2)}{\sqrt{s}}$

Figure 2: Comparison of performance for different $\mathcal{P}$ with the Gaver-Stehfest algorithm.

To provide additional insight, we plot $f_5$ and $f_7$ in Figures 5 and 6. We plot each in two time scales in order to show the behavior near 0 as well as the values over a longer interval. Note that $f_5(t)$ is essentially 0 in a neighborhood of $t = 0$.

Several of the transforms on our list involve half powers. However, we could require other fractional powers. To illustrate, we considered the function $f_{14}$, which is the sum of two quarter powers. The associated Laplace transform $\hat{f}_{14}(s)$ would be inverted exactly if we used quarter powers. Figure 7 shows how our standard power algorithms perform on this

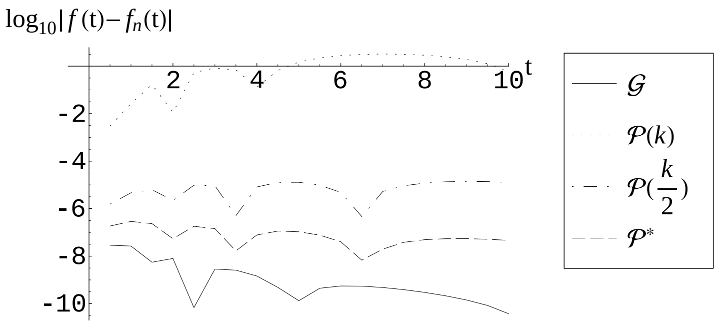


Figure 3: Comparison of some standard $\mathcal{P}$ for $f_5 : f(t) = \frac{e^{-1/t}}{\sqrt{\pi t^3}}, \hat{f}(s) = e^{-2\sqrt{s}}$.

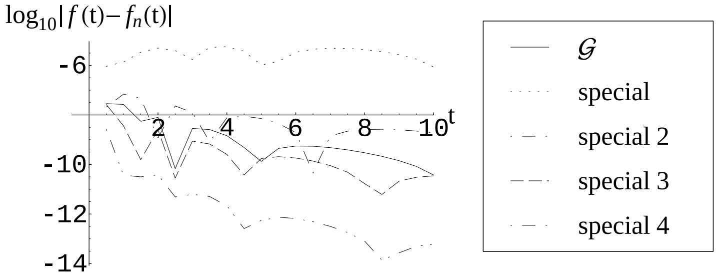


Figure 4: Comparison of some "exotic" $\mathcal{P}$ for $f_5 : f(t) = \frac{e^{-1/t}}{\sqrt{\pi t^3}}, \hat{f}(s) = e^{-2\sqrt{s}}$.

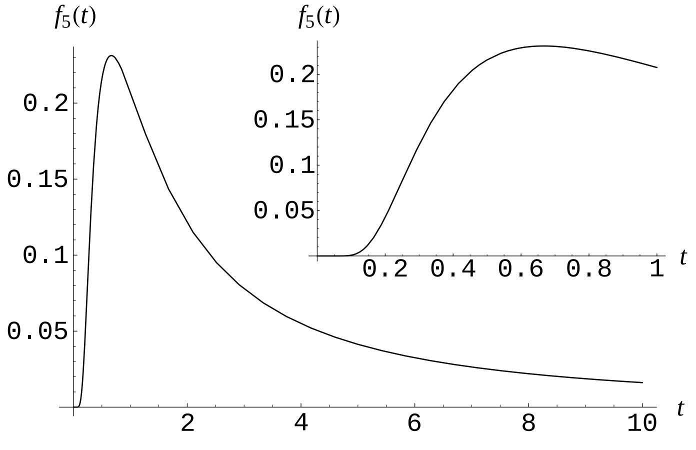


Figure 5: The function $f_5(t) = \frac{e^{-1/t}}{\sqrt{\pi t^3}}$.

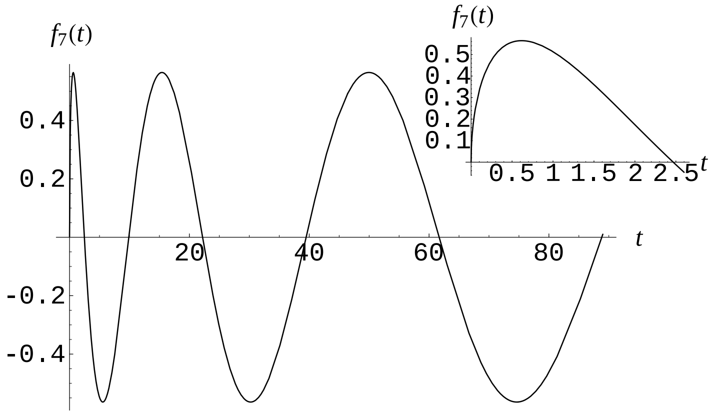


Figure 6: The function $f_7(t) = \frac{\sin(2\sqrt{t})}{\sqrt{\pi}}$.

example. This example further shows the possible improvements in inversion efficiency if we carefully tune the powers.

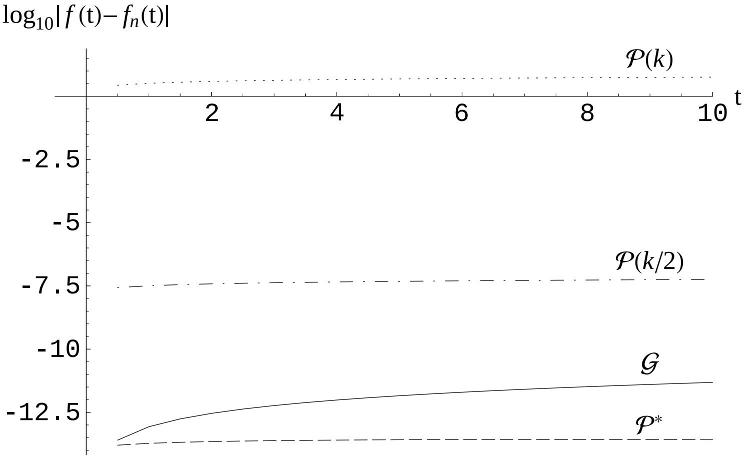


Figure 7: Comparison of different $\mathcal{P}$ for $f_{14} : f(t) = t^{1/4} + t^{7/4}, \hat{f}(s) = \frac{\Gamma(5/4)}{s^{5/4}} + \frac{\Gamma(11/4)}{s^{11/4}}$.

# 5. Choosing the Nodes

As indicated in Section 4 of the main paper, we examined the impact of the node set on the inversion, restricting attention to positive real nodes. We did many more experiments than described in the main paper. We describe some of those here.

We carry out a series of numerical experiments on inversions of three transforms, $f_9$, $f_2$ and $f_4$ from the Table 2. Our reference node set is the default node set $\mathcal{N}^* = \{1, 2, \ldots, n\}$.

We examine the effects of the following modifications: increasing and decreasing the length of the node interval, increasing the value of the largest node, spacing the nodes geometrically instead of linearly, varying the spacing between the nodes and the distance of the nodes from the origin linearly and, lastly, selecting and perturbing the nodes randomly.

## 5.1 Varying the Node Interval

We start our investigation of where to place the nodes by considering $n = 30$ nodes that are spaced equally in an interval of varying size. In our first experiment we hold the leftmost node fixed at $\alpha_1 = 1$ and we place the rightmost node at $\alpha_n = k, k = 2, 3, \ldots, 62$, thus obtaining 61 node intervals of increasing length. Then we solve the power relations to get the corresponding weights for each interval and we carry out the inversions for transforms $f_9$, $f_2$ and $f_4$. We show the absolute errors of every fifth inversion thus obtained (i.e. inversions with $\alpha_n = 2, 7, \ldots, 62$) in figures 8, 9 and 10. We can see that increasing the length of the interval initially helps improve inversion performance, but then the marginal gain decreases. These pictures support the notion that there is great flexibility in the node set provided that it is not put in too small an interval.

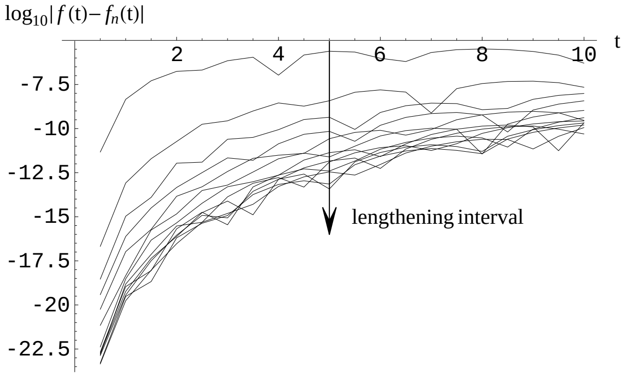


Figure 8: The error plot as the node interval lengthens for $f_9 : f(t) = e^{-t}, \hat{f}(s) = \frac{1}{s+1}$.

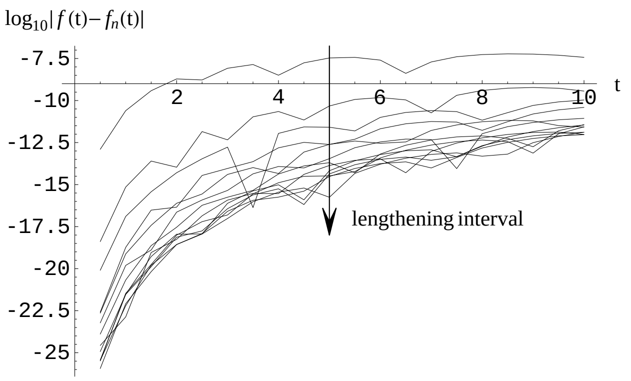


Figure 9: The error plot as the node interval lengthens for $f_2 : f(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}, \hat{f}(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

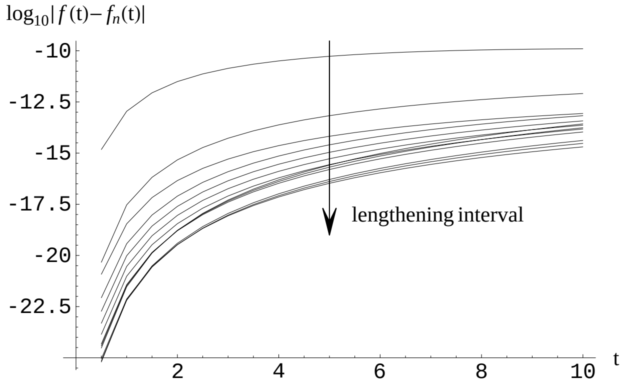


Figure 10: The error plot as the node interval lengthens for $f_4 : f(t) = e^t \text{erfc}(\sqrt{t}), \hat{f}(s) = \frac{1}{s+\sqrt{s}}$.

Next we turn to shortening the interval of the node set. We perform the same experiments as above except that we now hold the rightmost node fixed at $\alpha_1 = 62$ (the largest node value tested above) and we place the leftmost node at $\alpha_n = k, k = 61, 60, \ldots, 1$, thus obtaining 61 node intervals of decreasing length. We solve for the weights for each interval and we carry out the inversions for $f_9$, $f_2$ and $f_4$. We again show the absolute errors of every fifth inversion thus obtained (i.e. inversions with $\alpha_n = 61, 56, \ldots, 1$) in Figures 11, 12 and 13. These results show less variation than in Figures 8, 9 and 10, so that we conclude that having a relatively large node is the key requirement for a good inversion.

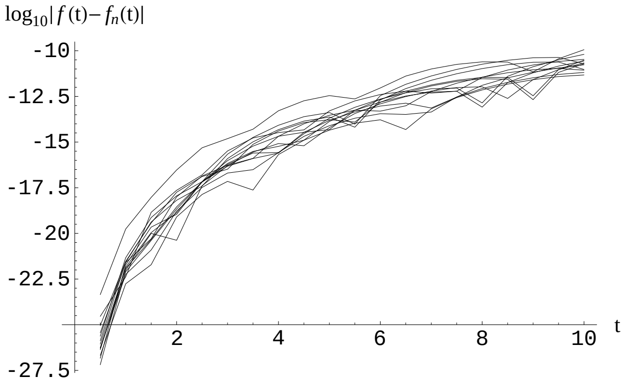


Figure 11: The error plot as the node interval shortens for $f_9 : f(t) = e^{-t}, \hat{f}(s) = \frac{1}{s+1}$.

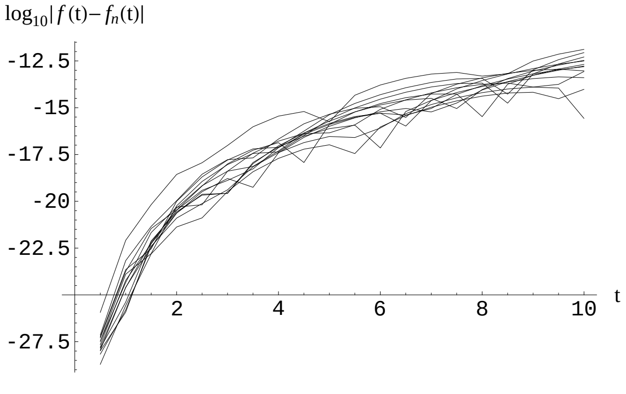


Figure 12: The error plot as the node interval shortens for $f_2 : f(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}, \hat{f}(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

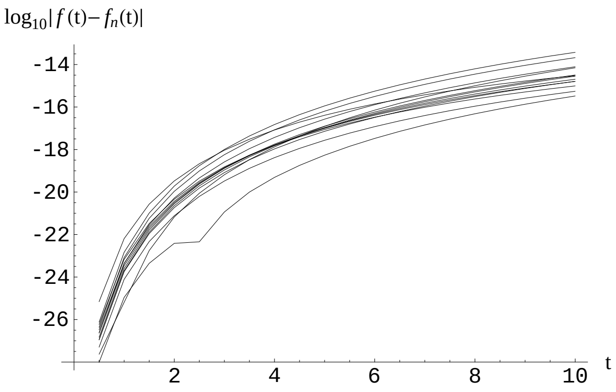


Figure 13: The error plot as the node interval shortens for $f_4 : f(t) = e^t \mathrm{erfc}(\sqrt{t})$, $\hat{f}(s) = \frac{1}{s+\sqrt{s}}$.

## 5.2 Moving Only the Largest Node

Section 5.1 shows that having larger nodes is beneficial, but it seems that if a few large nodes are present in the node set, the effect of adding even larger nodes is not adding much. We investigate this further by moving only the largest node. We move the largest node of the default node set $\mathcal{N}^* = \{1, 2, \ldots, n\}$ while leaving the rest of the nodes in $\mathcal{N}^*$ intact. In particular, we perform inversions after replacing the largest node at $n$ with $n+k, k = 0, \ldots, n$ (and re-solving to get new weights) and examine the effect on the inversion error. We plot the superposition of the resulting error plots in Figures 14, 15 and 16 for three different transforms. We observe that the dispersion in the error plots is not significant, implying that placing the largest node further out than at $n$ does not yield much improvement.

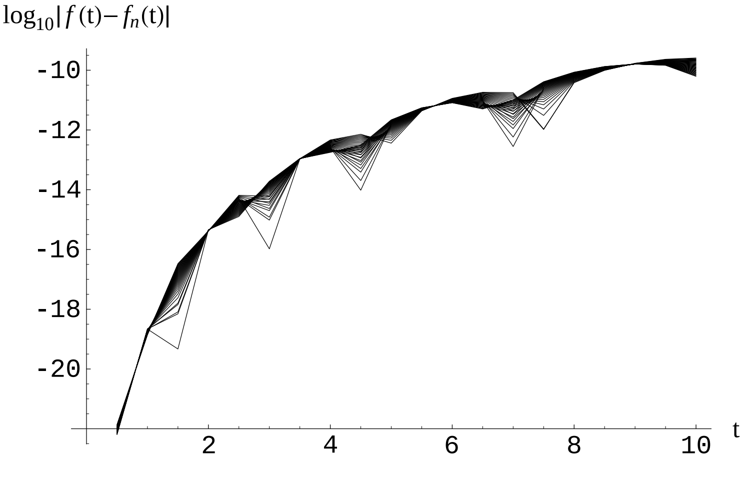


Figure 14: The error plot with increasing only the largest node for $f_9(t) = e^{-t}$, $\hat{f}_9(s) = \frac{1}{s+1}$.

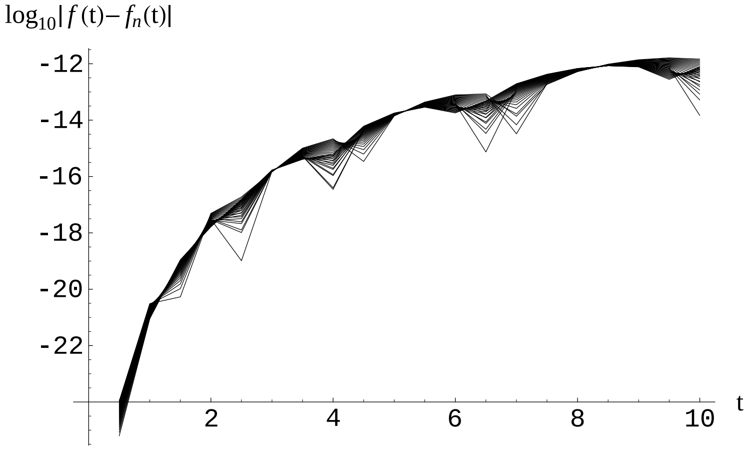


Figure 15: The error plot with increasing only the largest node for $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}$, $\hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

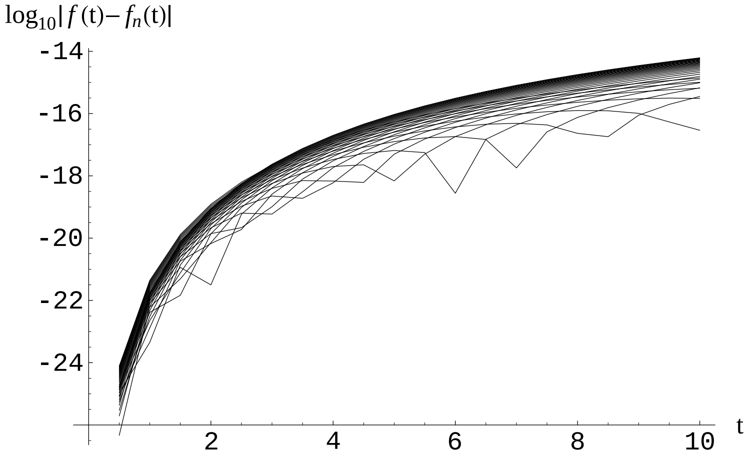


Figure 16: The error plot with increasing only the largest node for $f_4(t) = e^t \text{erfc}(\sqrt{t})$, $\hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$.

## 5.3 Linear versus Geometric Spacing

All the examples above have the nodes spaced linearly. We investigated whether or not it would be better to have the nodes spaced geometrically in the same interval $[1, 30]$. Specifically, we considered letting node $k$ be placed at $r^{k-1}$, $1 \leq k \leq 30$, where $r^{29} = 30$. The results are displayed in Figures 17, 18 and 19. Linear spacing is evidently better.

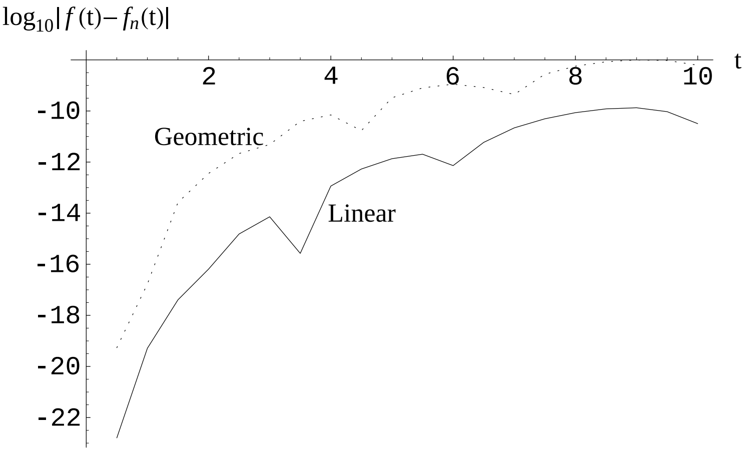


Figure 17: Linear versus geometric spacing for $f_9(t) = e^{-t}$, $\hat{f}_9(s) = \frac{1}{s+1}$.

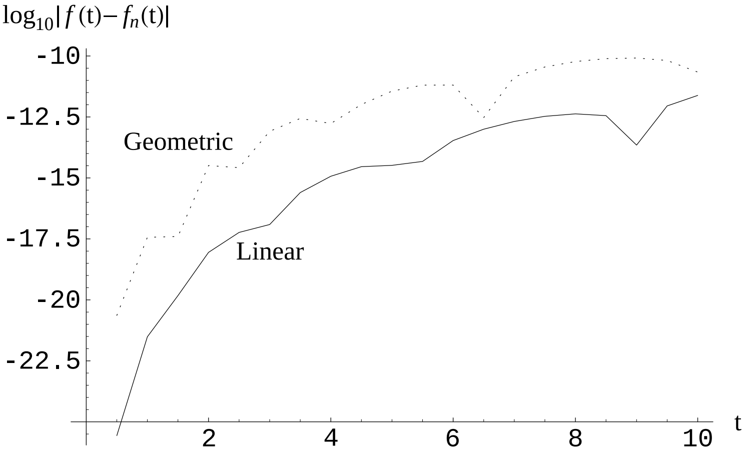


Figure 18: Linear versus geometric spacing for $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}$, $\hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

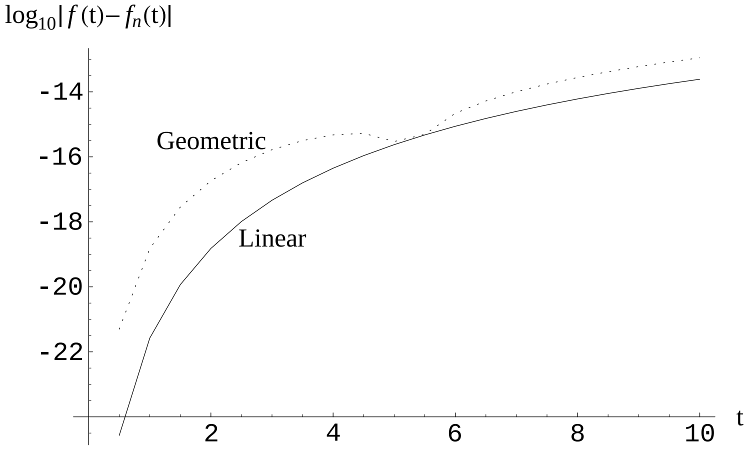


Figure 19: Linear versus geometric spacing for $f_4(t) = e^t \mathrm{erfc}(\sqrt{t})$, $\hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$.

### 5.4 Spacing and Offset

As described in Section 4 of the main paper, we considered node sets of the form

$$\alpha_k = \theta + k\delta, \quad 1 \leq k \leq n = 30 \ , \tag{7}$$

as a function of a positive real *shift* $\theta$ and a positive real *spacing* $\delta$. For each candidate node set over a wide range of parameters $\theta$ and $\delta$, we solve the system of linear equations to obtain the corresponding weights and examine the resulting inversion error. Figure 4 in the main paper shows the surface of the average error (measured in the logarithm to base 10) in the inversion of the standard exponential transform $\hat{f}_9$, for $\theta \in \{0, 0.25, \ldots, 4.75\}$ and $\delta \in \{0.1, 0.25, \ldots, 2.95\}$.

Below we display figures like Figure 4 in the main paper. Here we treat three different transforms and consider three values of $n$: $n = 20$, $n = 30$ and $n = 40$. We also give two different views of each surface. These appear in Figures 20, 21 and 22.

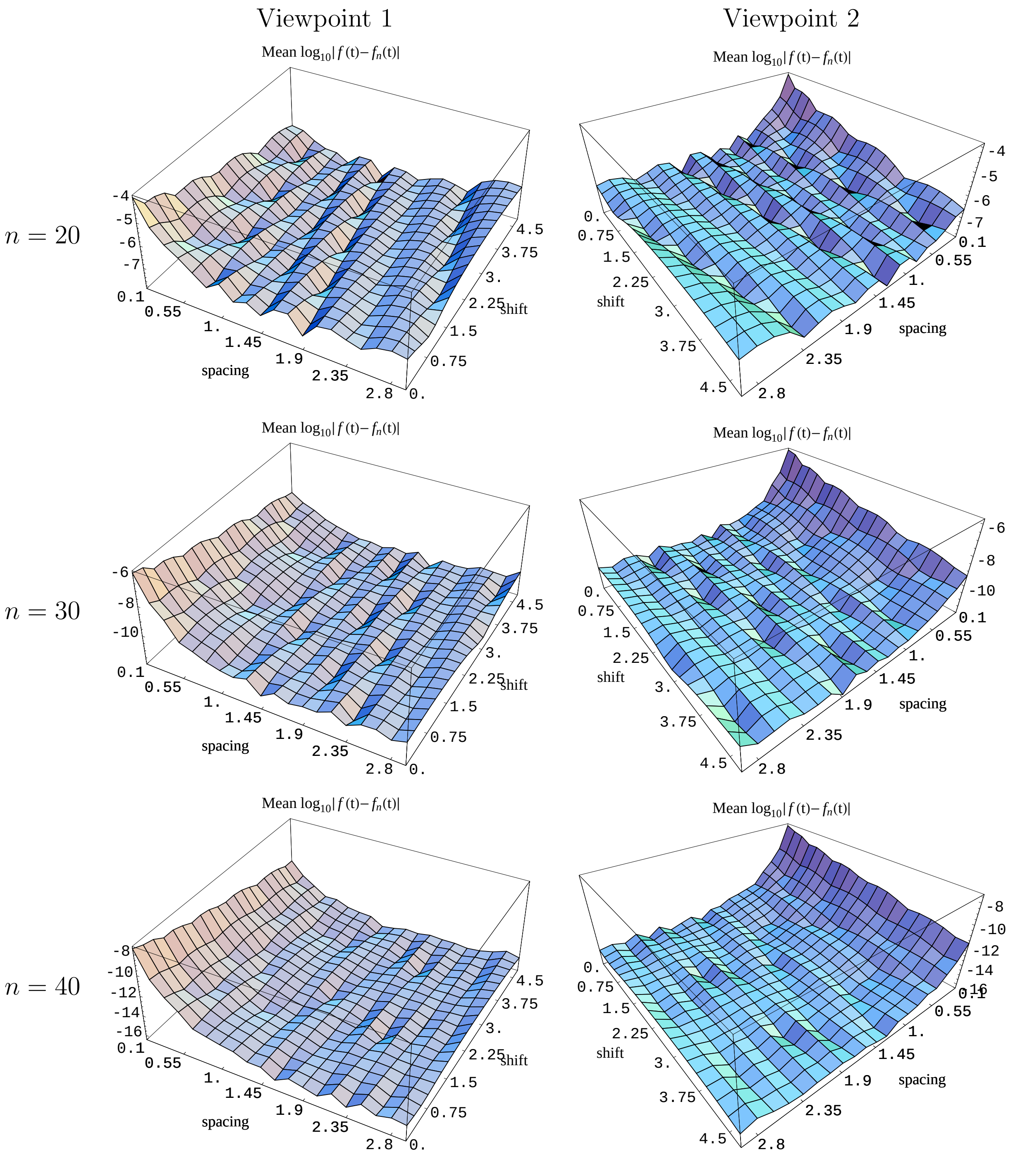


Figure 20: The error surfaces for $f_9(t) = e^{-t}$, $\hat{f}_9(s) = \frac{1}{s+1}$.

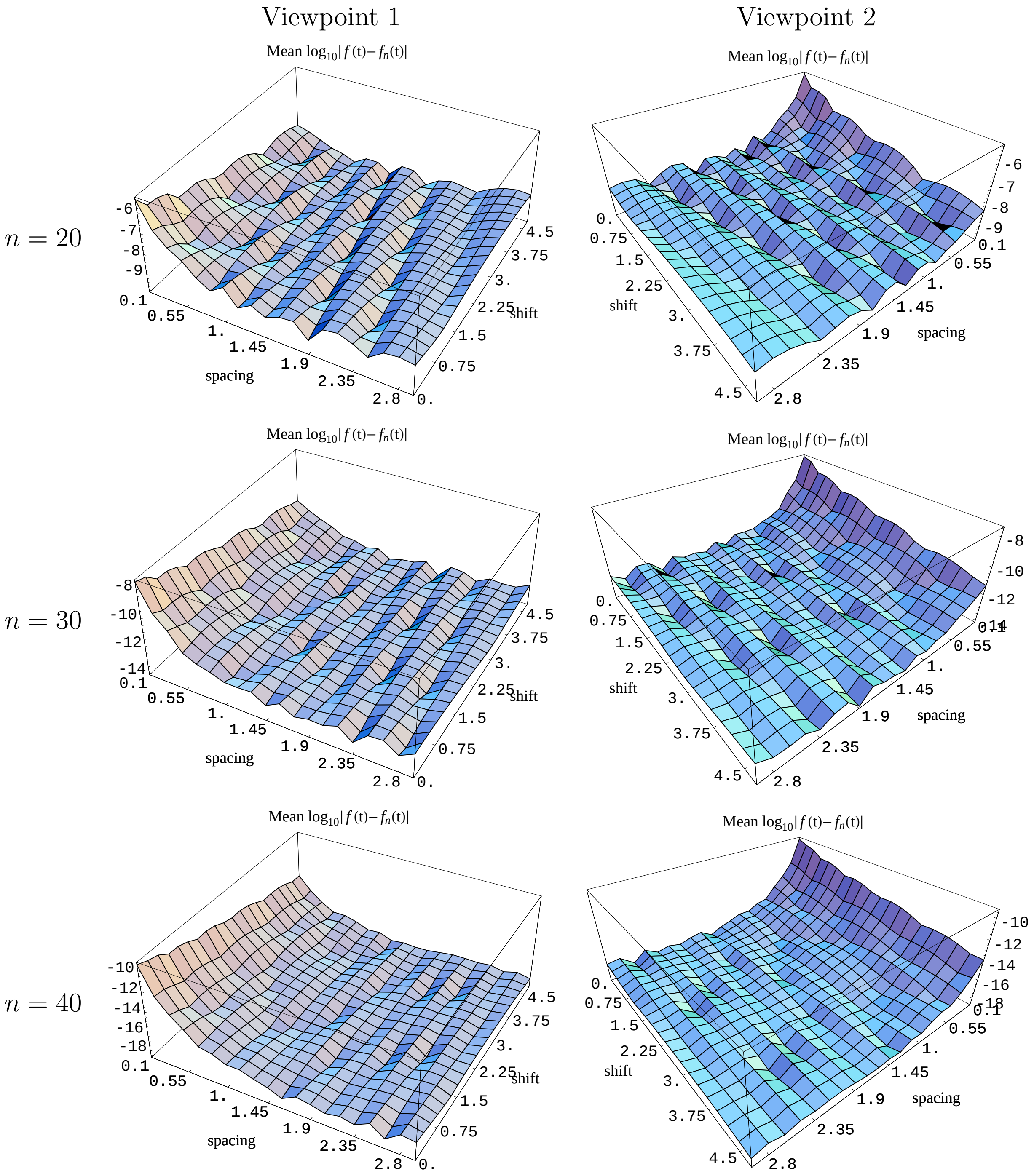


Figure 21: The error surfaces for $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}$, $\hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

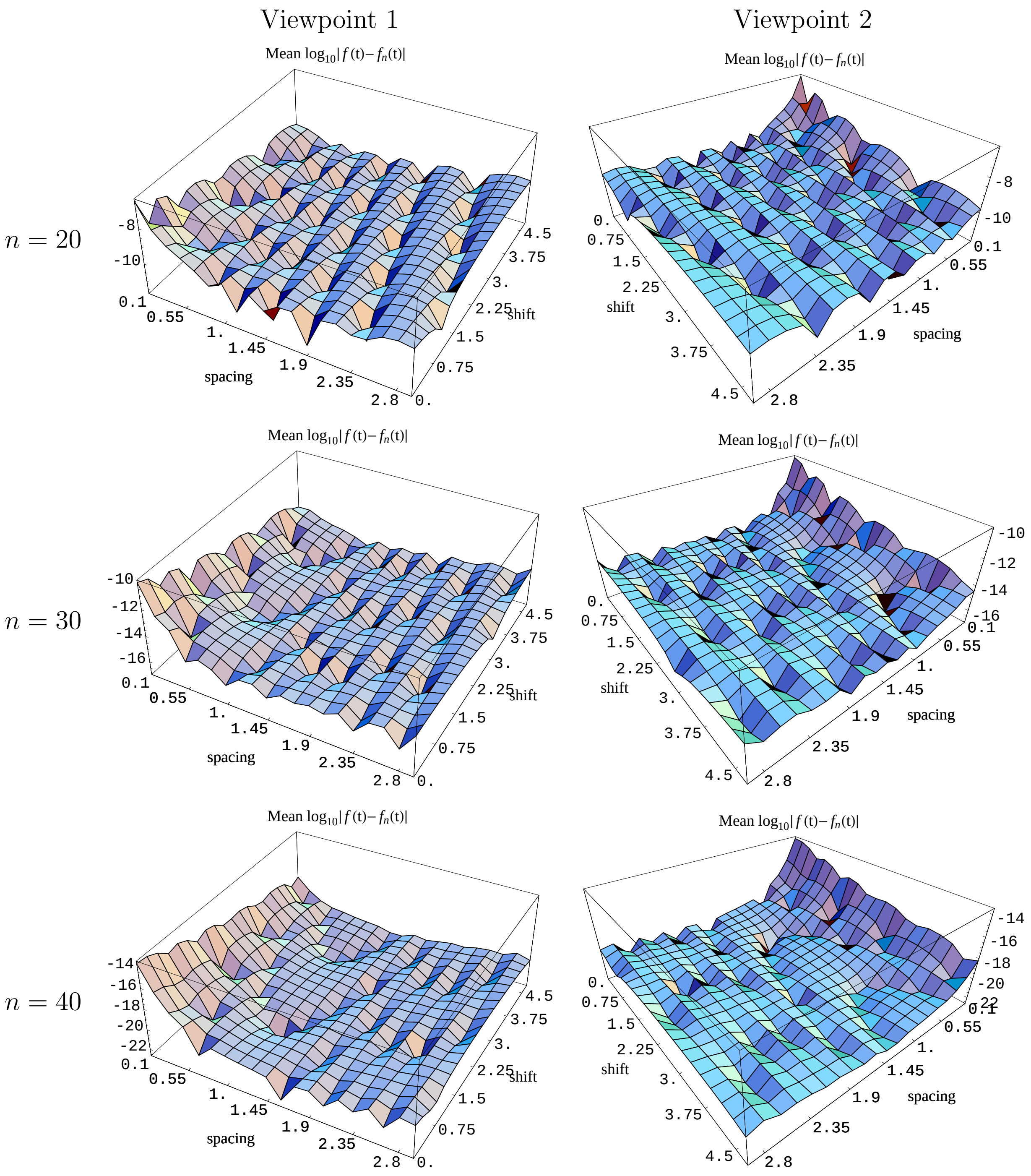


Figure 22: The error surfaces for $f_4(t) = e^t \mathrm{erfc}(\sqrt{t})$, $\hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$.

## 5.5 Sensitivity Analysis

Perturbing all the nodes or all the weights at the same time destroys the inversion, provided that we do not re-solve the linear system. This also extends to perturbing a single node or a single weight.

## 5.6 The Impact of Randomization

We now describe experiments with randomly generated sets of nodes to check the impact of randomization. We generate $n = 30$ i.i.d. nodes, uniformly distributed in the closed interval $[1, 30]$. We then solve the system of linear equations based on these random nodes to obtain the weights and to do the inversion. We carry out $M = 100$ independent replications of this experiment for transforms $f_9$, (Figure 23), $f_2$, (Figure 24) and $f_4$, (Figure 25). In each case, we observe a relatively narrow band of performance results.

Let $f_n^i(t)$ be the inverse obtained by the nodes and weights of the $i^{\text{th}}$ replication. Define

$$\begin{aligned}
\overline{f_n}(t) &= \frac{1}{M}\sum_{i=1}^{M} f_n^i(t) \\
|f - \overline{f_n}|(t) &= |f(t) - \overline{f_n}(t)| \\
\overline{|f - f_n|}(t) &= \frac{1}{M}\sum_{i=1}^{M} |f(t) - f_n^i(t)| \\
|f - f_n|^*(t) &= \max_{1\le i\le M} |f(t) - f_n^i(t)| \\
|f - f_n|_*(t) &= \min_{1\le i\le M} |f(t) - f_n^i(t)|
\end{aligned}$$

and let $|f - f_n|_q(t)$, $0 < q < 1$ denote the $q^{\text{th}}$-quantile of the values $\{|f(t) - f_n^1(t)|, \ldots, |f(t) - f_n^M(t)|\}$. Figures 23, 24 and 25 show these errors together with the absolute error of $(\mathcal{N}^*, \mathcal{P}^*)$ (the unperturbed version of the inversion) over $t$ for functions $f_9$, $f_2$ and $f_4$. We note that, although not shown in the plots, the error of the Gaver-Stehfest algorithm is similar to $|f - f_n|^*$ for functions $f_9$ and $f_2$. On the other hand, for $f_4$, the error of $\mathcal{G}$ is worse than the error of $(\mathcal{N}^*, \mathcal{P}^*)$ for small $t$ and crosses over at around $t = 10$.

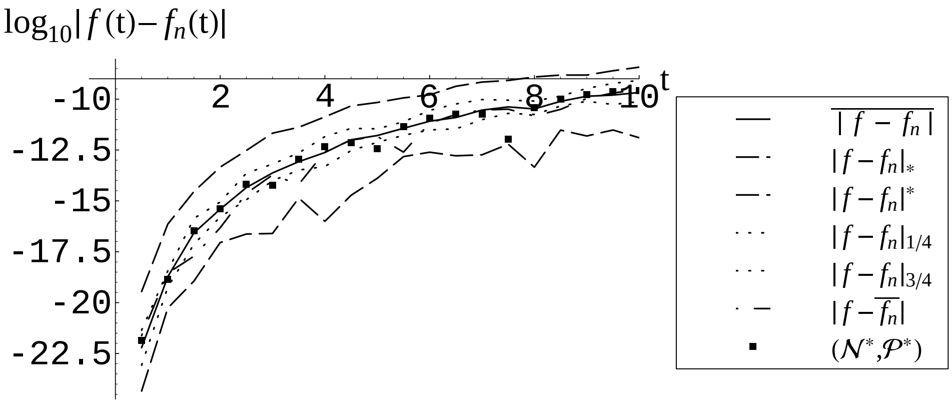


Figure 23: Performance of the randomized inversions and $(\mathcal{N}^*, \mathcal{P}^*)$ for $f_9(t) = e^{-t}, \hat{f}_9(s) = \frac{1}{s+1}$.

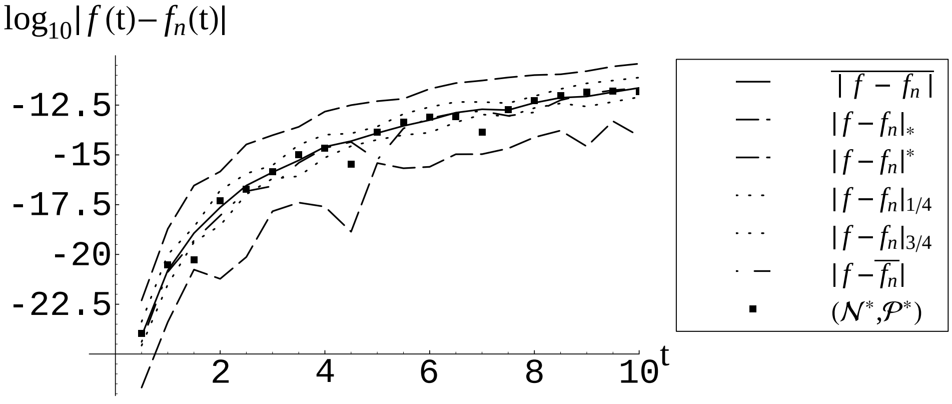


Figure 24: Performance of the randomized inversions and $(\mathcal{N}^*, \mathcal{P}^*)$ for $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}, \hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$.

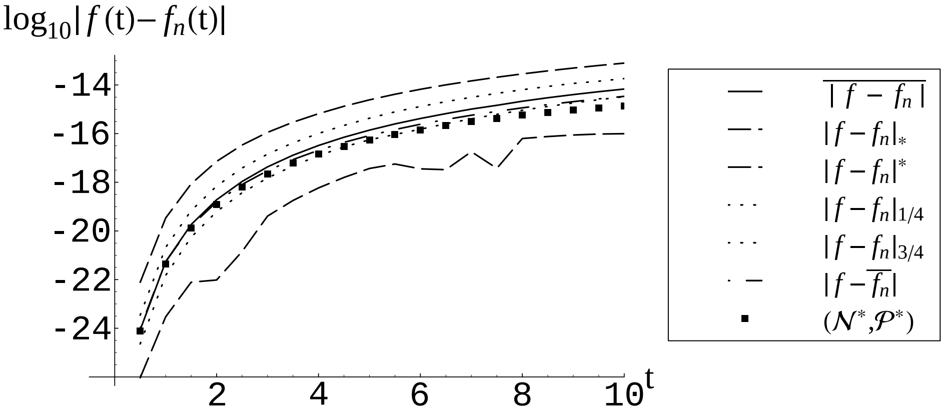


Figure 25: Performance of the randomized inversions and $(\mathcal{N}^*, \mathcal{P}^*)$ for $f_4(t) = e^t\mathrm{erfc}(\sqrt{t})$, $\hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$.

# 6. The Structure of the Weights

Here we compare the weights of the default $(\mathcal{P}^*, \mathcal{N}^*)$ power algorithm to the the weights of $\mathcal{G}$. A fundamental property of Gaver-Stehfest is that the sign of its weights alternates. As we can see in Figures 26–30, the $(\mathcal{P}^*, \mathcal{N}^*)$ power algorithm closely mimics this pattern, but does not follow it exactly. These figures depict the base-10 logarithm of the absolute value of the weights $\omega_1, \ldots, \omega_n$ of $\mathcal{G}$ and $\mathcal{P}^*$, $\mathcal{N}^*$ for a few values of $n$. The $\mathcal{G}$ weights appear as circles, filled in if $\omega_k > 0$ and empty if $\omega_k < 0$ whereas the $(\mathcal{P}^*, \mathcal{N}^*)$ weights appear as squares, filled in if $\omega_k > 0$ and empty if $\omega_k < 0$.

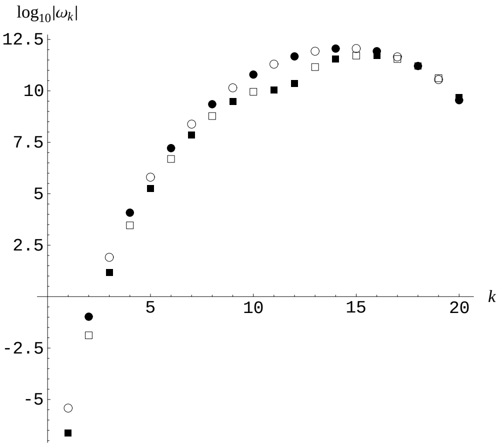


Figure 26: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^*)$ (squares) for $n = 20$.

Let $\mathcal{N}^{\mathcal{G}}$ denote the node set of the Gaver-Stehfest algorithm $\mathcal{G}$, and let $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ denote the algorithm given if we solve the power relations using the node set $\mathcal{N}^{\mathcal{G}}$ with the power set $\mathcal{P}^*$. Then we can also examine how the application of $\mathcal{P}^*$ (via the power relations) on the weights of $\mathcal{G}$ affects the structure of the weights. We show this in Figures 31–35.

We have also compared the weights of $(\mathcal{P}^*, \mathcal{N}^*)$ to the weights of $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ to get a feel of the sensitivity of the weights to changes in the node sets. These two sets of weights have almost the same sign change structure, but the magnitude of the weights are slightly different.

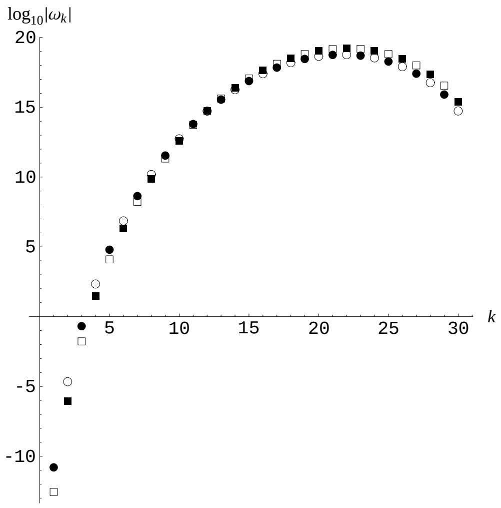


Figure 27: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^*)$ (squares) for $n = 30$.

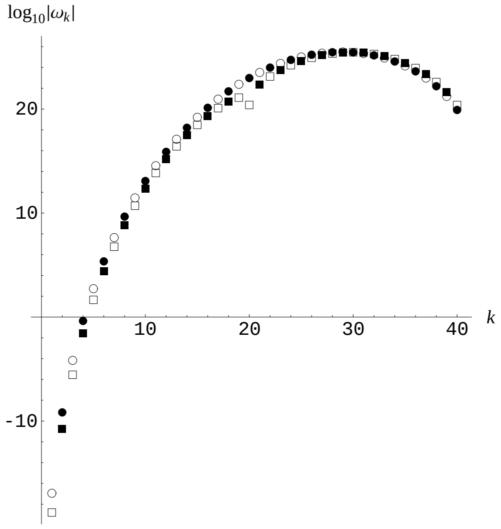


Figure 28: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^*)$ (squares) for $n = 40$.

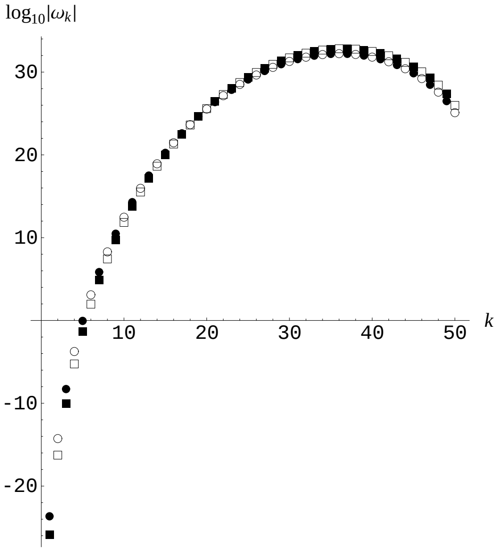


Figure 29: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^*)$ (squares) for $n = 50$.

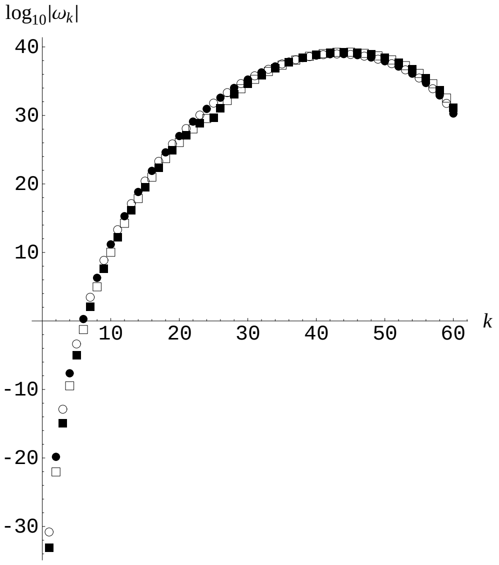


Figure 30: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^*)$ (squares) for $n = 60$.

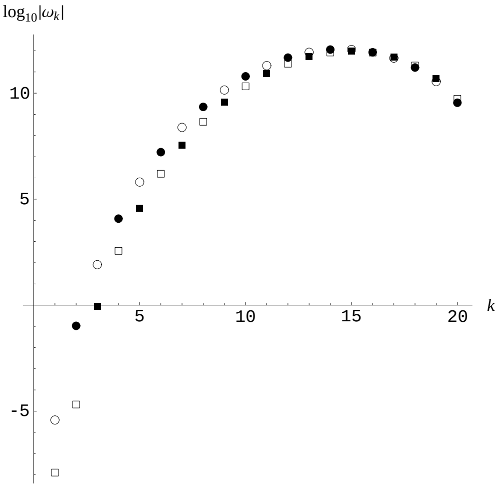


Figure 31: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ (squares) for $n = 20$.

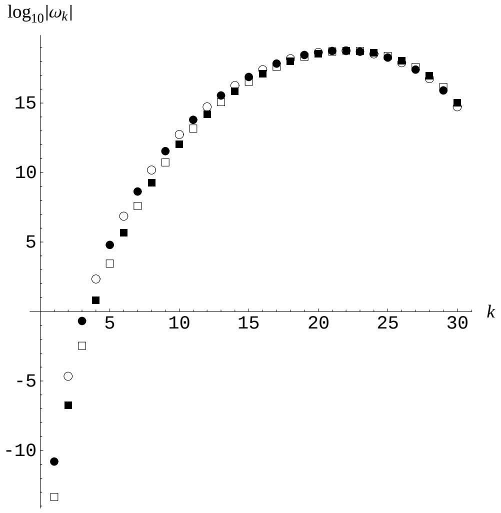


Figure 32: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ (squares) for $n = 30$.

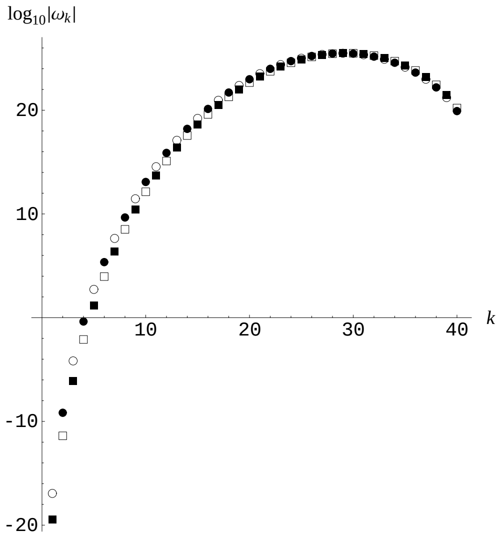


Figure 33: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ (squares) for $n = 40$.

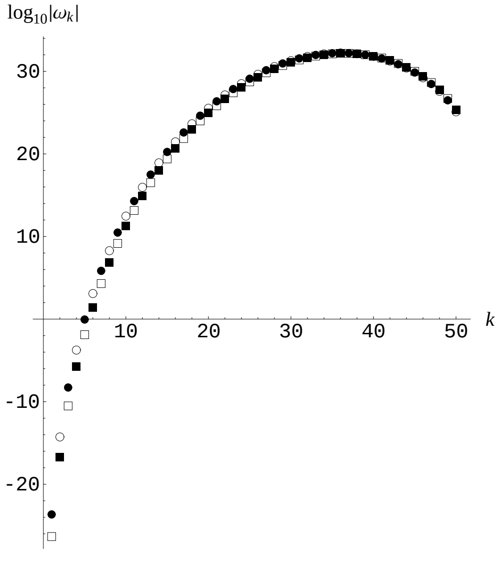


Figure 34: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ (squares) for $n = 50$.

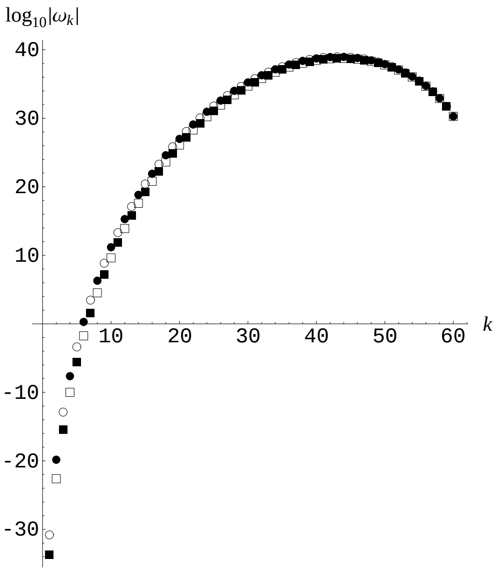


Figure 35: The weights of $\mathcal{G}$ (circles) and $(\mathcal{P}^*, \mathcal{N}^{\mathcal{G}})$ (squares) for $n = 60$.

# 7. Accuracy and Precision Requirements

Here we slightly amplify Section 5 of the main paper. We investigate accuracy and precision, measured in the number of digits. We first investigate how the number $n$ of terms in the sum (1) and the computer precision affect the precision of the weights obtained by solving the linear system (4) using the default node set $\mathcal{N}^*$ and power set $\mathcal{P}^*$. Table 3 shows results as a function of the two variables, with each ranging over several multiples of 10. From Table 3, we see that the required computer precision to solve the linear system as a function of $n$ is approximately $1.2n$. We have used $1.5n$ as the precision requirement in Figure 1 to be safe. For any given $n$, the precision of the weights increases with the computer precision, as shown in Table 3.

We find that the precision of the inversion, as measured by the base-10 logarithm of the absolute error $|f_n(t) - f(t)|$ primarily depends on $n$ provided that the computer precision is above the threshold $1.2n$. The performance of the default power algorithm with node set $\mathcal{N}^*$ and power set $\mathcal{P}^*$ tends to be similar to Gaver-Stehfest, as described in Section 7 of Abate and Whitt (2006), but the power algorithm performs slightly better for $0 < t \leq 10$, significantly so for smaller values of $t$. The ordering is reversed for larger $t$, though; see Section 10. Specific comparisons appear in Figures 36, 37 and 38.

Table 3: $\left|\log_{10}\left|\frac{w_{p+1}-w_p}{w_p}\right|\right|$ as a function of $n$, where $w_p$ stands for the weight computed with precision $p$.

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

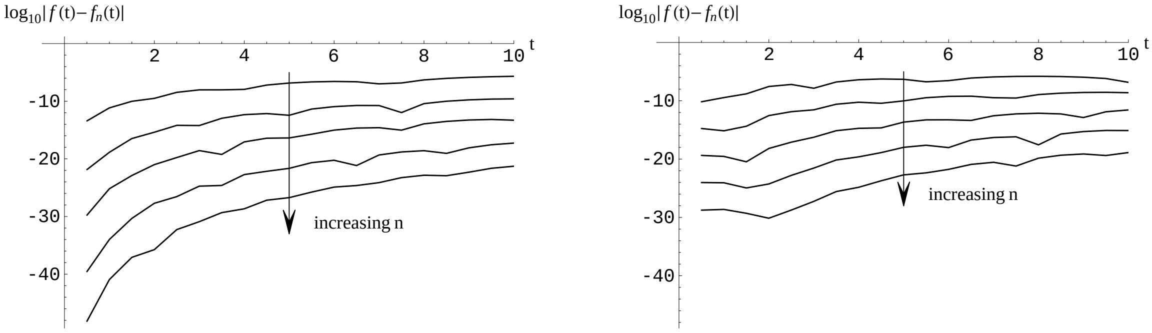


Figure 36: Inversion of $f_9(t) = e^{-t}, \hat{f}_9(s) = \frac{1}{s+1}$ with $\mathcal{P}^*$, $\mathcal{N}^*$ (left) versus $\mathcal{G}$ (right) for $n = 20, 30, 40, 50, 60$

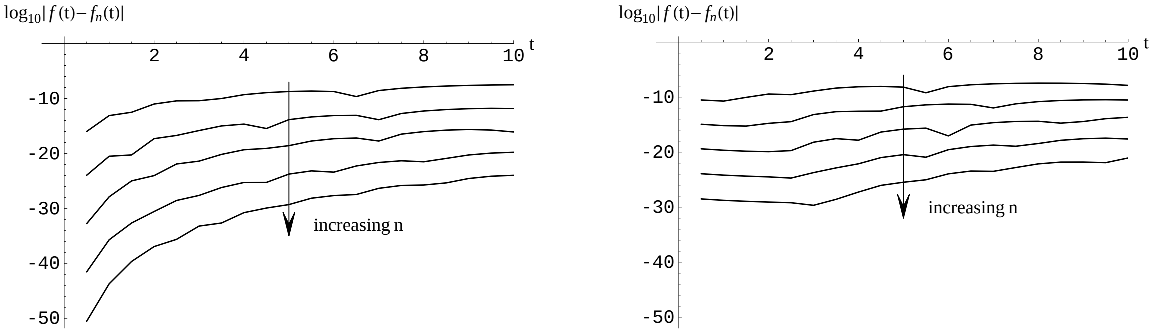


Figure 37: Inversion of $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}, \hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$ with $\mathcal{P}^*$, $\mathcal{N}^*$ (left) versus $\mathcal{G}$ (right) for $n = 20, 30, 40, 50, 60$.

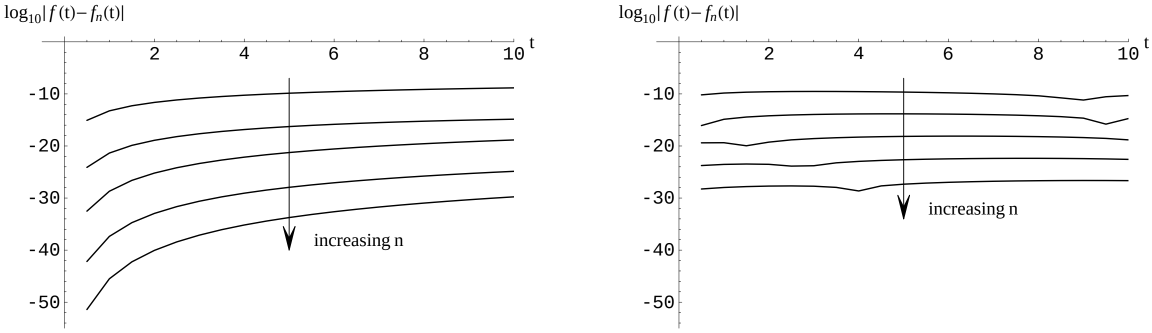


Figure 38: Inversion of $f_4(t) = e^t \mathrm{erfc}(\sqrt{t}), \hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$ with $\mathcal{P}^*$, $\mathcal{N}^*$ (left) versus $\mathcal{G}$ (right) for $n = 20, 30, 40, 50, 60$.

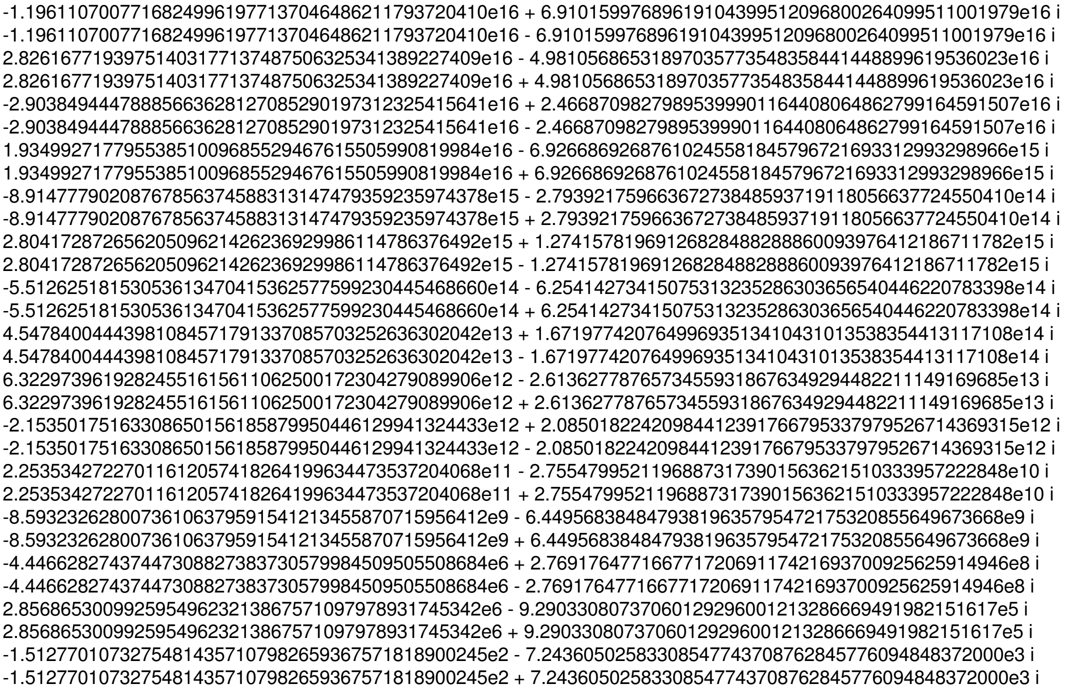
# 8. Zakian's Algorithm

Zakian's algorithm obtains nodes and weights through the Padé approximant of the complex exponential function $e^{-z}$; see Baker and Graves-Morris (1996). We have implemented Zakian's algorithm based on the $((n-1)/n)$ Padé approximant of $e^{-z}$. The details of how to compute the nodes and weights of the algorithm are given on pages 522 and 523 of Zakian and Edwards (1978). The nodes and weights are displayed below in Figures 39 and 40 below for the case $n = 30$. Note that both the nodes and weights appear in complex conjugates

```
3.93956425452757459825745847076077335755988546e1 + 1.71198870255437172929360044776474931594386669e0 i
```

Figure 39: The Zakian complex nodes for $n = 30$.

We display the Zakian nodes in the complex plane for five values of $n$, ranging from $n = 20$ to $n = 60$ in Figure 41.



```
-1.1961107007716824996197713704648621179372041 0e16 + 6.9101599768961910439951209680026409951100197 9e16 i
```

Figure 40: The Zakian complex weights for $n = 30$.

In the main paper we observe that the Zakian algorithm ($\mathcal{Z}$) is especially effective for smooth functions. Unfortunately, that special focus comes at a penalty, because the Zakian algorithm tends to perform poorly for non-smooth functions. We illustrate this sharply diverging performance in Figure 42. For comparison, we show results for the algorithms $\mathcal{G}$, $\mathcal{E}$ and $\mathcal{T}$ together with $\mathcal{Z}$ in Figure 42. For the smooth function $f_1 \equiv \sin(t)$, Zakian produces far better accuracy than any of the other algorithms, but for the non-smooth function $f_2$, Zakian performs worse than the other algorithms.

We also exhibit this diverging performance by considering inversions of different functions only for $\mathcal{Z}$. In Figure 43 we show the error performance of $\mathcal{Z}$ for the two smooth functions $f_9$ and $f_3$, which correspond to the transforms that $\mathcal{Z}$ inverts the best, versus the two non-

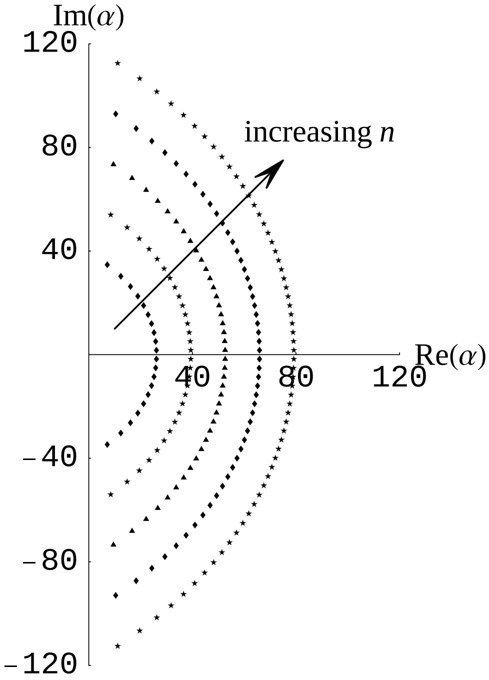
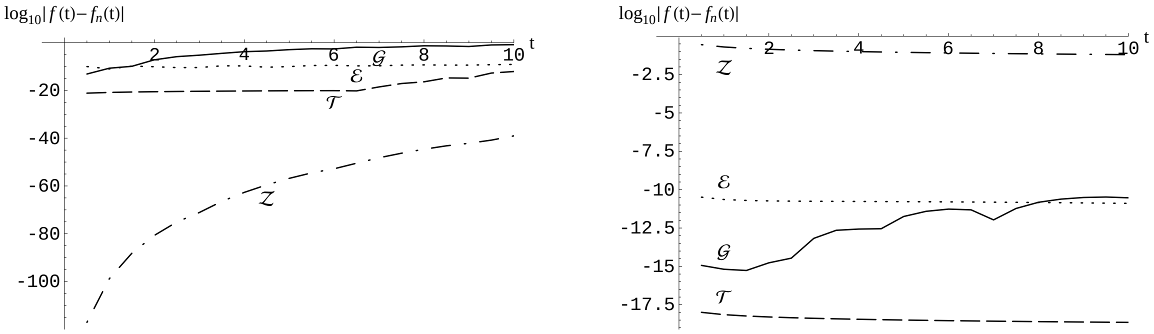


Figure 41: The nodes of Zakian ($\mathcal{Z}$) for $n = 20, 30, 40, 50, 60$.

smooth functions $f_2$ and $f_4$, which correspond to transforms that $\mathcal{Z}$ inverts the worst.



$f_1 : f(t) = \sin(t), \hat{f}(s) = \frac{1}{s^2+1}$

$f_2 : f(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}, \hat{f}(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$

Figure 42: The two extremes of Zakian's algorithm.

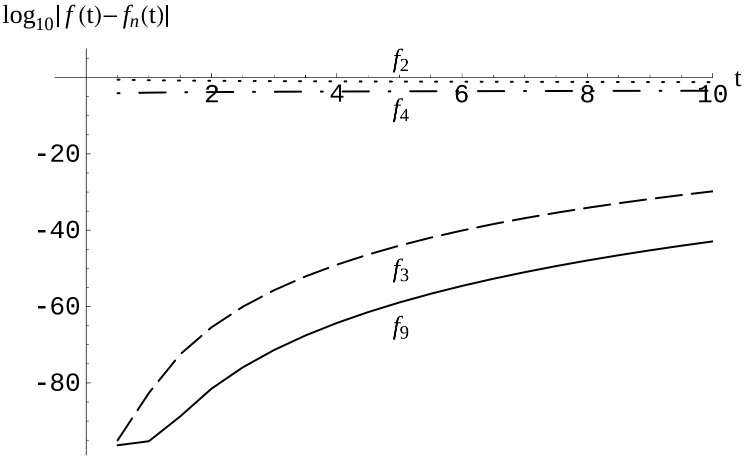


Figure 43: The extremes of Zakian's algorithm.

# 9. Evaluating Established Algorithms

In this section we supplement Section 7 of the main paper, where we showed that the power test functions can be used to evaluate other algorithms. In particular, we used the power functions to evaluate Talbot's algorithm ($\mathcal{T}$), the Euler algorithm ($\mathcal{E}$), the Gaver-Stehfest algorithm ($\mathcal{G}$) and Zakian's algorithm ($\mathcal{Z}$). As shown by Abate and Whitt (2006), these other algorithms all can be expressed in the unified framework (1) by appropriate node and weight vectors $\alpha$ and $\omega$. Thus the algorithms are characterized by these vectors $\alpha$ and $\omega$, which are specified in Abate and Whitt (2006). Here we present some additional detail.

## 9.1 Nodes and Weights

We start by giving more information about these other algorithms. We described Zakian's algorithm in the previous section. For example, Figure 41 shows the Zakian nodes for various $n$. The algorithms $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{G}$ are described in Abate and Whitt (2006).

Figures 44 and 45 show the Talbot nodes for various $n$. Figures 46 and 47 show the nodes of the Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$) and Zakian ($\mathcal{Z}$) algorithms for $n = 30$ in the same plot. Figures 48 and 49 show the Talbot and Euler weights for $n = 30$.

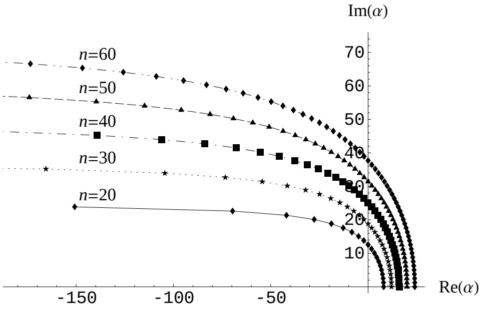


Figure 44: The nodes of Talbot ($\mathcal{T}$) for various $n$. The last few nodes for high $n$ are not shown.

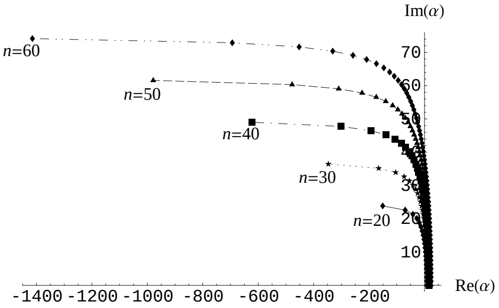


Figure 45: The entire range of the Talbot nodes for various $n$.

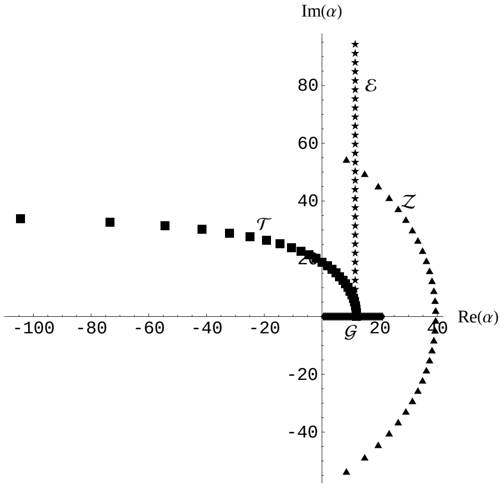


Figure 46: The nodes of Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$) and Zakian ($\mathcal{Z}$) for $n = 30$. The last two nodes of $\mathcal{T}$, at $(56\pi/5)(i + \cot(14\pi/15))$ and $(58\pi/5)(i + \cot(29\pi/30))$ are not shown.

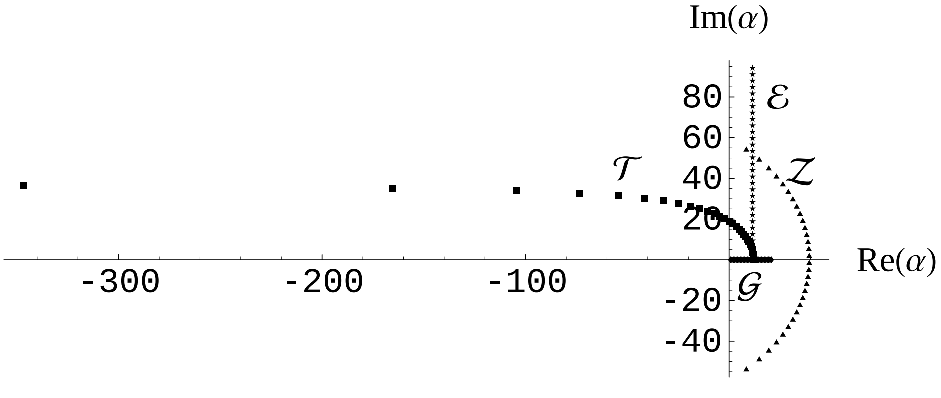


Figure 47: All the nodes of the Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$) and Zakian ($\mathcal{Z}$) for $n = 30$.

```
3.25509582838007841616010409796973566340418569e4
1.51106571900048712806711917225254391493891027e4 + 6.06026059986219652370507043327928844958043595e4 i
-4.86770730215046319485101992546477763202185679e4 + 2.58877364259163819523143321915344193863414624e4 i
-2.99384175448156386944639984917936364411644951e4 - 3.32294147873374662660282920724409086738831200e4 i
1.85917568584221435003980435576517466756450831e4 - 2.76069824317462854316469562272054536293827672e4 i
2.13009180888419515085285936831089514039833311e4 + 7.71826614401200062831547561111958546436073560e3 i
-1.43419349690908332394898964098887659546393375e3 + 1.39811435573641275471171534375831732238511691e4 i
-7.82590996007218345235605437314433075552332778e3 + 1.13173832954897563935296789393198720556146274e3 i
-1.52463412969585716743694237772996376247892229e3 - 3.70928921218008889694699423853608123928479081e3 i
1.46288781570624110683156531346513437045140800e3 - 1.08226649105411038451345000352403423053202639e3 i
5.66176985788079650645966104556737723976757649e2 + 4.63649768927811872234861303997686475472251975e2 i
-1.09269144015688774359977330038008218311538053e2 + 2.33508362383850773827330791519948371622336162e2 i
-7.70381813691551500445033973311886328173638736e1 - 1.46768173055997074910529973998355235278214536e1 i
-1.25782591231270483200063578812453153207050920e0 - 2.01906650876328610337480682555318945648347952e1 i
4.11431879677159765335344870829689000777801020e0 - 1.33558853387101327990102690072754558294893351e0 i
4.00000000000000000000000000000000000000000000e-1 + 6.28318530717958647692528676655900576839433880e-1 i
-6.77726615261110788331491016691701969237929446e-2 + 7.28462794468055965707978472174725497849057342e-2 i
-8.64967044374482473294324192612359186525155580e-3 - 4.64765535377239337651689943954476160126659027e-3 i
1.56068108910999024124543187440389186741123139e-4 - 6.52320853621824356842016512399638197369351131e-4 i
2.90046837237078481112254920465093802128884343e-5 - 7.42074632164382304452874611977165812800536525e-7 i
1.99652866533243689754398467406208838680919082e-7 + 6.72805617365432550696860596646483082131007119e-7 i
-6.74380619552938816602872234038384012118414977e-9 + 4.17328434264709317965458255439941142958172398e-9 i
-2.34433100130619613941414707744196580894541977e-11 - 2.17441939777738696274602983820253351019522884e-11 i
1.37819009340326527821358683247252767449107578e-14 - 2.67700022095979265499900953060898158311275642e-14 i
3.20236436867707786781551184817779550489943042e-18 + 6.46067232004637689533943990478344375835260475e-19 i
9.34046153545159597001968615361755333169010662e-25 + 1.13991238415254219035455483558230729129472247e-23 i
-9.42798794794905794254603656127467129359498022e-32 + 3.63046978611772777610535491623211952842728701e-32 i
-3.57342331313026855182704805169930016549475071e-45 - 4.61475455595475465428611931487755359119222216e-45 i
2.14776405051308980691897977765435390915282351e-71 - 3.04350454218261916322519072908430460667195271e-71 i
2.86889736450694453760188977841240094622214019e-149 + 9.21139863842759136712884894166605210811283056e-150 i
```

Figure 48: The Talbot complex weights for $n = 30$.

```
5.0000000000000000000000000000000000000000e4
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
1.0000000000000000000000000000000000000000e5
-1.0000000000000000000000000000000000000000e5
9.9969482421875000000000000000000000000000e4
-9.9951171875000000000000000000000000000000e4
9.9630737304687500000000000000000000000000e4
-9.8242187500000000000000000000000000000000e4
9.4076538085937500000000000000000000000000e4
-8.4912109375000000000000000000000000000000e4
6.9638061523437500000000000000000000000000e4
-5.0000000000000000000000000000000000000000e4
3.0361938476562500000000000000000000000000e4
-1.5087890625000000000000000000000000000000e4
5.9234619140625000000000000000000000000000e3
-1.7578125000000000000000000000000000000000e3
3.6926269531250000000000000000000000000000e2
-4.8828125000000000000000000000000000000000e1
3.0517578125000000000000000000000000000000e0
```

Figure 49: The Euler real weights for $n = 30$.

## 9.2 Scoring the Algorithms with Power Test Functions

We now turn to evaluating the algorithms using the power test functions. In order to see how well these inversion algorithms perform in the inversion of power functions, we numerically identify the set of powers $p$ for which the power relation (3) holds to within a small specified error $\varepsilon$, and the complementary set where it is violated. For that purpose, let $p_\varepsilon^+(n)$ to be the smallest nonnegative number $p$ for which the $p^{\text{th}}$ power condition is violated by at least a small positive number $\varepsilon$,

$$p_\varepsilon^+(n) \equiv \min\{p \geq 0 : \left| \Re \left\{ \sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}} \omega_k \right\} - 1 \right| > \varepsilon\} \ . \tag{8}$$

Similarly, let

$$p_\varepsilon^-(n) \equiv \max\{p \leq 0 : \left| \Re \left\{ \sum_{k=1}^{n} \frac{\Gamma(p+1)}{\alpha_k^{p+1}} \omega_k \right\} - 1 \right| > \varepsilon\} \ . \tag{9}$$

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

Thus the range $(p_\varepsilon^-(n), p_\varepsilon^+(n))$ is the smallest contiguous interval of $p$ among half powers for which the $p^{\text{th}}$ power relation is satisfied within $\varepsilon$. In Figure 50 we show $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for the four algorithms $\mathcal{G}$, $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{Z}$ for $2 \leq n \leq 40$ and $\varepsilon = 0.05$. We obtain the points shown by fixing $n$ and evaluating the power conditions numerically. For $p \geq 0$, the points to the right and under each curve meet the power conditions within $\varepsilon$, while the points to the left and over do not. Similarly, for $p \leq 0$, the points to the right and over each curve meet the power conditions within $\varepsilon$, while the points to the left and over do not.

The single most striking observation from Figure 50 is that the positive and negative power thresholds $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ are *linear* in $n$, demonstrating consistent regular behavior as $n$ changes. Moreover, for $p \geq 0$, all four existing algorithms meet the first few power conditions within $\varepsilon$, although they vary widely in how many they meet. Figure 50 shows that Euler performs slightly better than Gaver-Stehfest and that Talbot performs much better than either of them, which is consistent with extensive experience, including numerical inversion examples here in Section 10.

Table 4: The positive power threshold $p_\varepsilon^+(n) = c_n \cdot n + c_\varepsilon \cdot \log_{10}(\varepsilon) + c$ for different algorithms

| Algorithm | $c_n$ | $c_\varepsilon$ | $c$ |
|---|---|---|---|
| Gaver-Stehfest | 0.475 | 0.794 | 0.239 |
| Euler | 0.701 | 2.08 | $-0.523$ |
| Talbot | 1.39 | 0.474 | $-0.174$ |

Zakian's algorithm is the best for $p \geq 0$, which translates into the spectacular inversion for $f_1$ shown in Sections 8 and 10. On the other hand, for $p < 0$, Zakian's algorithm does not satisfy any power conditions; $p_\varepsilon^-(n) = 0$ for all $n$. In contrast, the Gaver-Stehfest, Euler and Talbot algorithms do meet some power conditions within $\varepsilon$ for $p < 0$ with the same ranking as for $p \geq 0$. Figure 50 helps explain why the Zakian algorithm is fundamentally different from the other algorithms. The Zakian algorithm evidently is highly tuned for smooth functions (i.e., analytic functions, with derivatives of all orders) at the expense of non-smooth functions.

We also observed that when $p \notin (p_\varepsilon^-(n), p_\varepsilon^+(n))$, the $p^{\text{th}}$ power conditions rapidly diverge from 1. Further analysis indicates that the lines in Figure 50 have additional regularity as we change the error tolerance $\epsilon$. That is illustrated by Figures 51, 52 and 53, which plots $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ as functions of $\varepsilon$ for each algorithm. Our experiments lead us to conclude that the power thresholds are approximately linear in the two variables $n$ and $\log_{10}(\varepsilon)$. The positive power threshold $p_\varepsilon^+(n)$ has approximately the linear formula

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

where the three parameters $c_n$, $c_\varepsilon$ and $c$ depend on the algorithm, as indicated in Table 4.

A similar, but less precise, linear relation holds for the negative power threshold, as shown in Table 5. We note that for Euler and Talbot, the slope $c_\varepsilon$ of the negative power threshold $p_\varepsilon^-(n)$ does vary a little. In this case, the value quoted in table is the average of the 5 slightly different slopes.

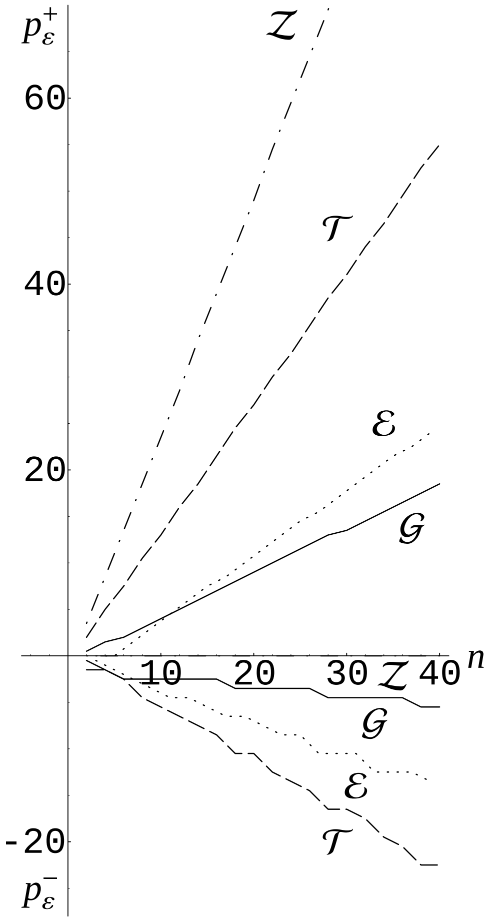


Figure 50: Scoring the Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$) and Zakian ($\mathcal{Z}$) algorithms using power test functions in half powers. The error tolerance is $\varepsilon = 0.05$.

Table 5: The negative power threshold $p_\varepsilon^-(n) = c_n \cdot n + c_\varepsilon \cdot \log_{10}(\varepsilon) + c$ for different algorithms

| Algorithm | $c_n$ | $c_\varepsilon$ | $c$ |
|---|---|---|---|
| Gaver-Stehfest | $-0.095$ | $-0.266$ | $-1.81$ |
| Euler | $-0.341$ | $-0.793$ | $-1.16$ |
| Talbot | $-0.565$ | $-1.06$ | $-1.04$ |

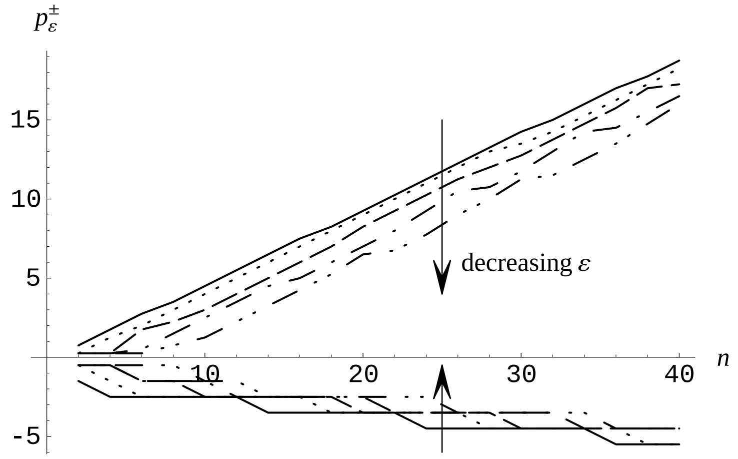


Figure 51: $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for $\varepsilon = 5 \times 10^{-j}, j = 1, 2, 3, 4, 5$ for the Gaver-Stehfest algorithm.

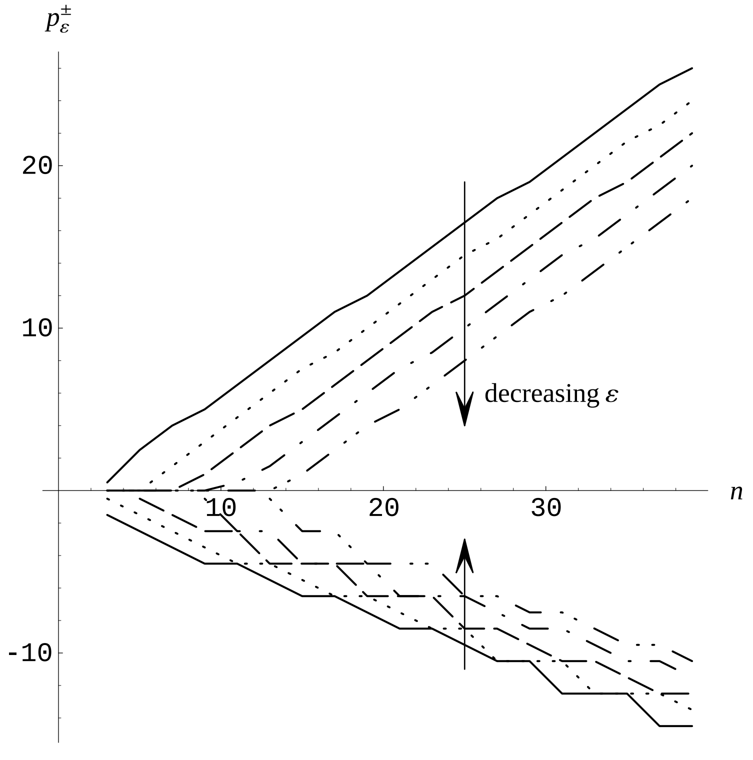


Figure 52: $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for $\varepsilon = 5 \times 10^{-j}, j = 1, 2, 3, 4, 5$ for the Euler algorithm.

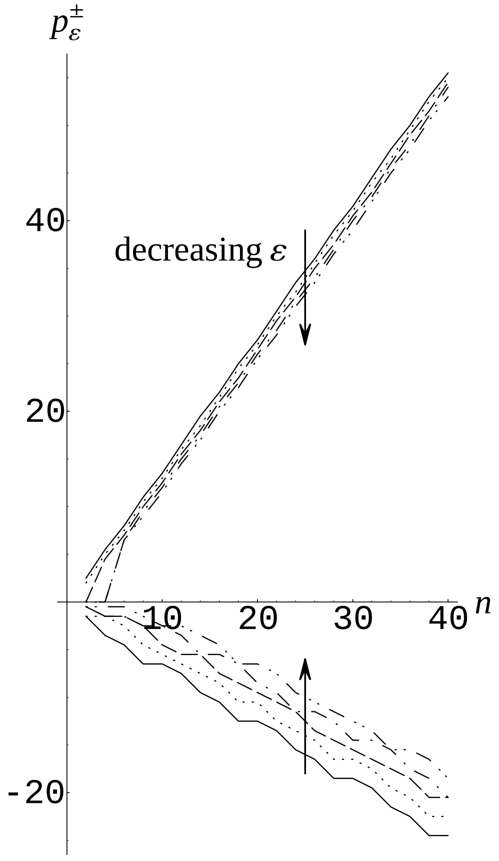


Figure 53: $p_\varepsilon^+(n)$ and $p_\varepsilon^-(n)$ for $\varepsilon = 5 \times 10^{-j}, j = 1, 2, 3, 4, 5$ for the Talbot algorithm.

# 10. Extensive Comparison of Performance

This section compares the performance of the established algorithms Gaver-Stehfest ($\mathcal{G}$), Euler ($\mathcal{E}$), Talbot ($\mathcal{T}$), Zakian ($\mathcal{Z}$) and our default power algorithm ($(\mathcal{N}^*, \mathcal{P}^*)$)). We show inversion errors for the transforms of Table 2 in Figures 54–73. We show both the absolute error $|f_n(t) - f(t)|$ and the relative error $|f_n(t) - f(t)|/|f(t)|$, both measured in log scale, so that we are showing number of digits accuracy. In particular, we display the base-10 logarithms: $\log_{10}(|f_n(t) - f(t)|)$ (absolute error) and $\log_{10}(|f_n(t) - f(t)|/|f(t)|)$ (relative error). Results for one function appear on each page. The absolute errors appear in the left column, while the relative errors appear in the right column. We calculate the values $f(t)$ for a wide range of times (function arguments) $t$: small values ($0.5 \leq t \leq 10$), medium values ($10 \leq t \leq 100$) and large values ($100 \leq t \leq 1000$). The time intervals increase moving down on each page, so that there are 6 plots on each page for each function. We discuss these numerical results in the next section.

For some transforms – $\hat{f}_1$, $\hat{f}_3$, $\hat{f}_9$, and $\hat{f}_{12}$ – $\mathcal{Z}$ performs much better than $\mathcal{G}$, $\mathcal{E}$,$\mathcal{T}$ and $\mathcal{P}^*$, at least for small values of $t$. As a consequence, it is difficult to distinguish visually how the other algorithms perform. For these cases (for small $t$), we show extra plots with the inversions of $\mathcal{G}$, $\mathcal{E}$,$\mathcal{T}$ and $\mathcal{P}^*$ only (leaving out $\mathcal{Z}$).

For the two transforms $\hat{f}_3$ and $\hat{f}_7$, we encounter difficulties because of the way *Mathematica* handles complex functions involving powers of $\sqrt{s}$. The direct versions

$$\hat{f}_3(s) = \frac{1}{\sqrt{s(s+2)}} \quad \text{and} \quad \hat{f}_7(s) = \frac{e^{-1/s}}{\sqrt{s^3}}$$

cause problems, so we label those $\hat{f}_{3w}$ and $\hat{f}_{7w}$ ($w$ for "wrong"). We avoid those difficulties by writing these transforms as

$$\hat{f}_3(s) = \frac{1}{\sqrt{s}\sqrt{s+2}} \quad \text{and} \quad \hat{f}_7(s) = \frac{e^{-1/s}}{s\sqrt{s}}$$

We display inversion results for both cases. We discuss this issue in Section 12.

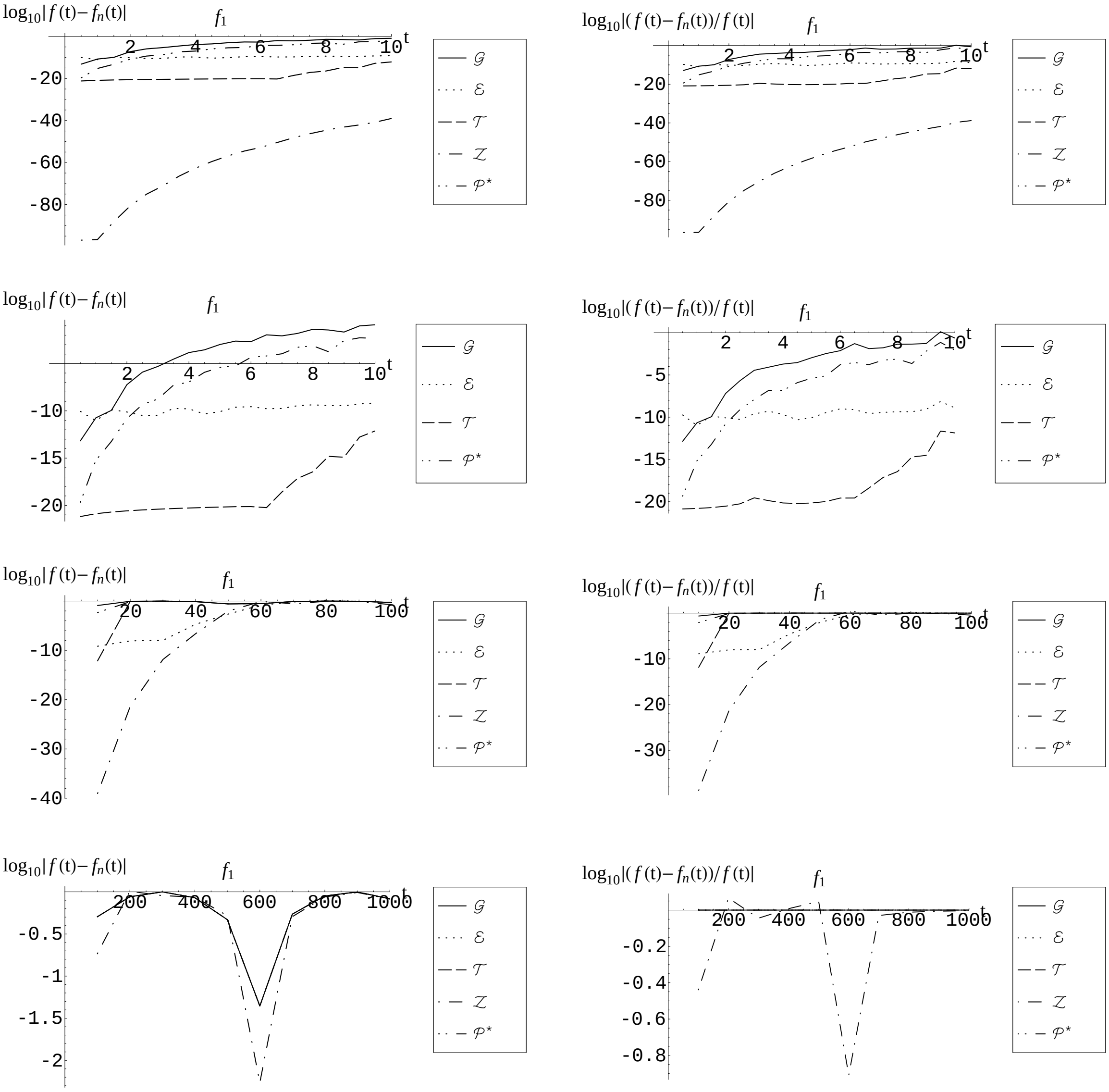


Figure 54: Comparison of inversion for $f_1(t) = \sin(t), \hat{f}_1(s) = \frac{1}{s^2+1}$ with $n = 30$.

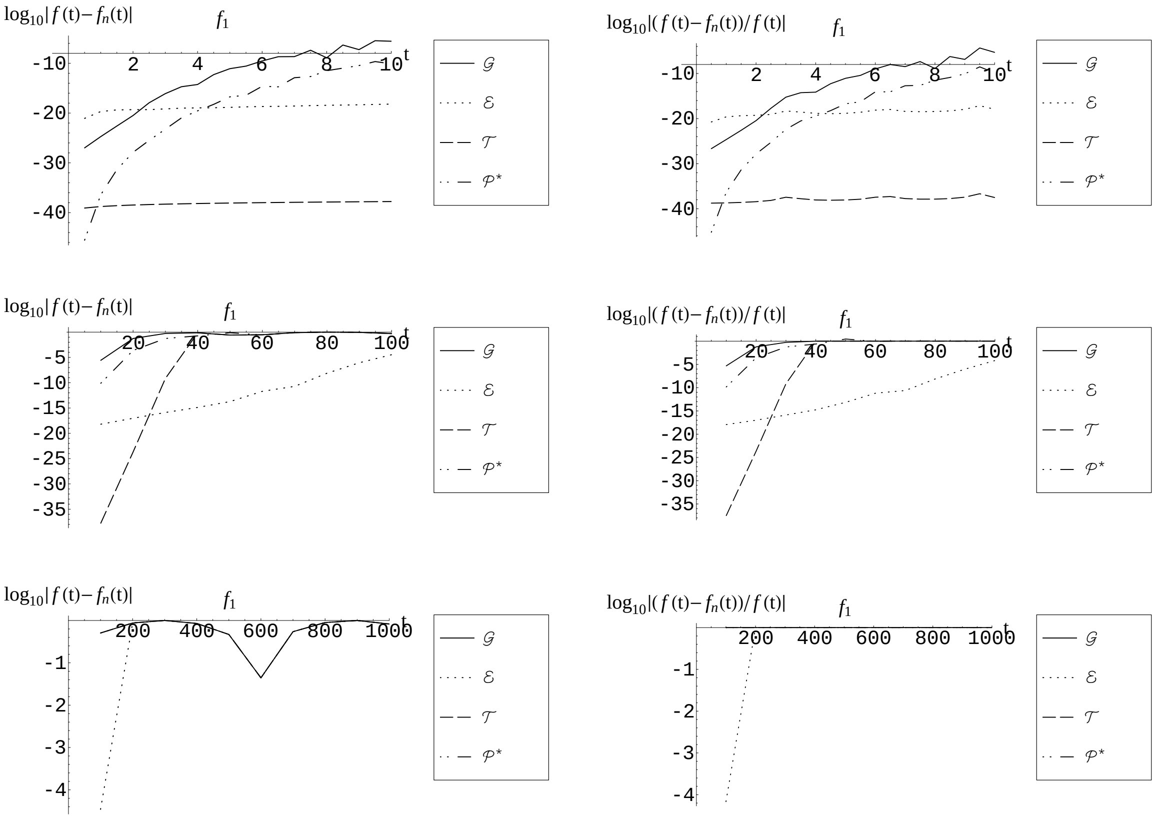


Figure 55: Comparison of inversion for $f_1(t) = \sin(t)$, $\hat{f}_1(s) = \frac{1}{s^2+1}$ with $n = 60$.

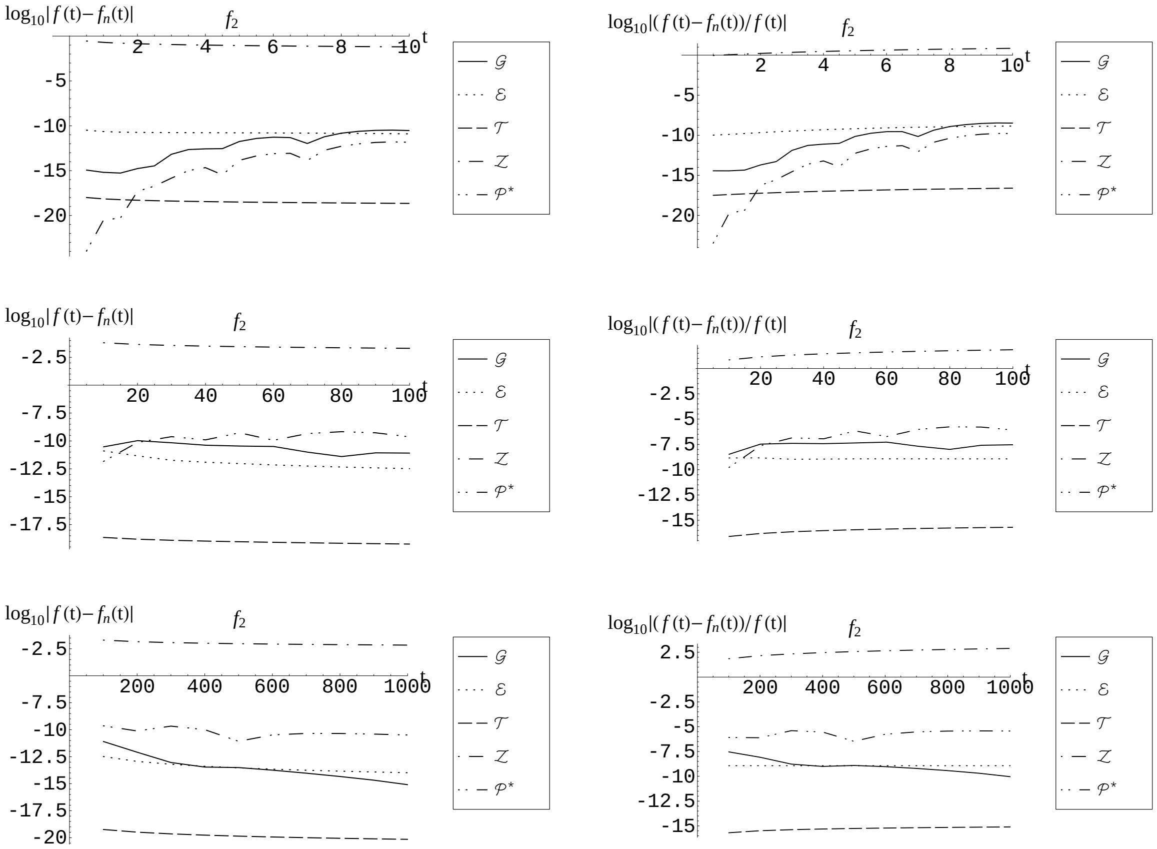


Figure 56: Comparison of inversion for $f_2(t) = \frac{1-e^{-t}}{2\sqrt{\pi t^3}}$, $\hat{f}_2(s) = \frac{1}{\sqrt{s}+\sqrt{s+1}}$ with $n = 30$.

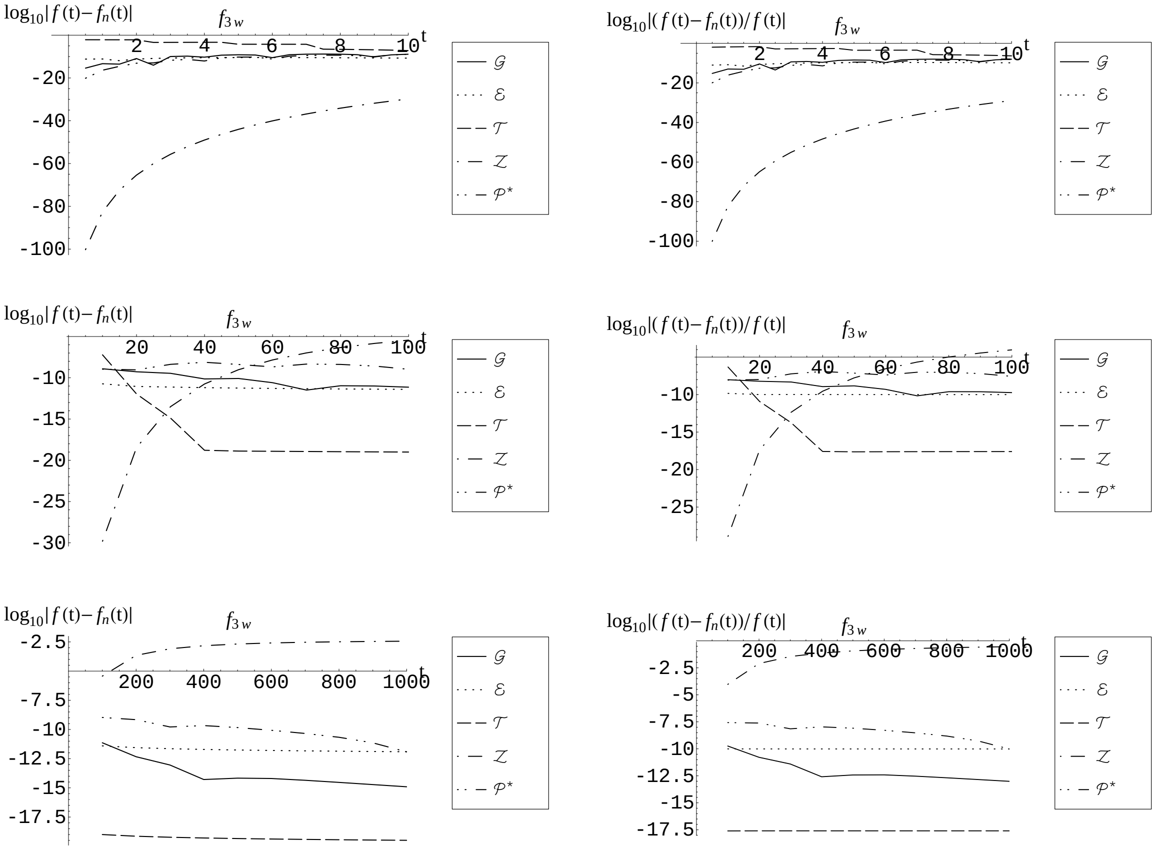


Figure 57: Comparison of inversion for $f_3(t) = e^{-t}I_0(t)$, when the transform is implemented as $\hat{f}_3(s) = \frac{1}{\sqrt{s(s+2)}}$ with $n = 30$.

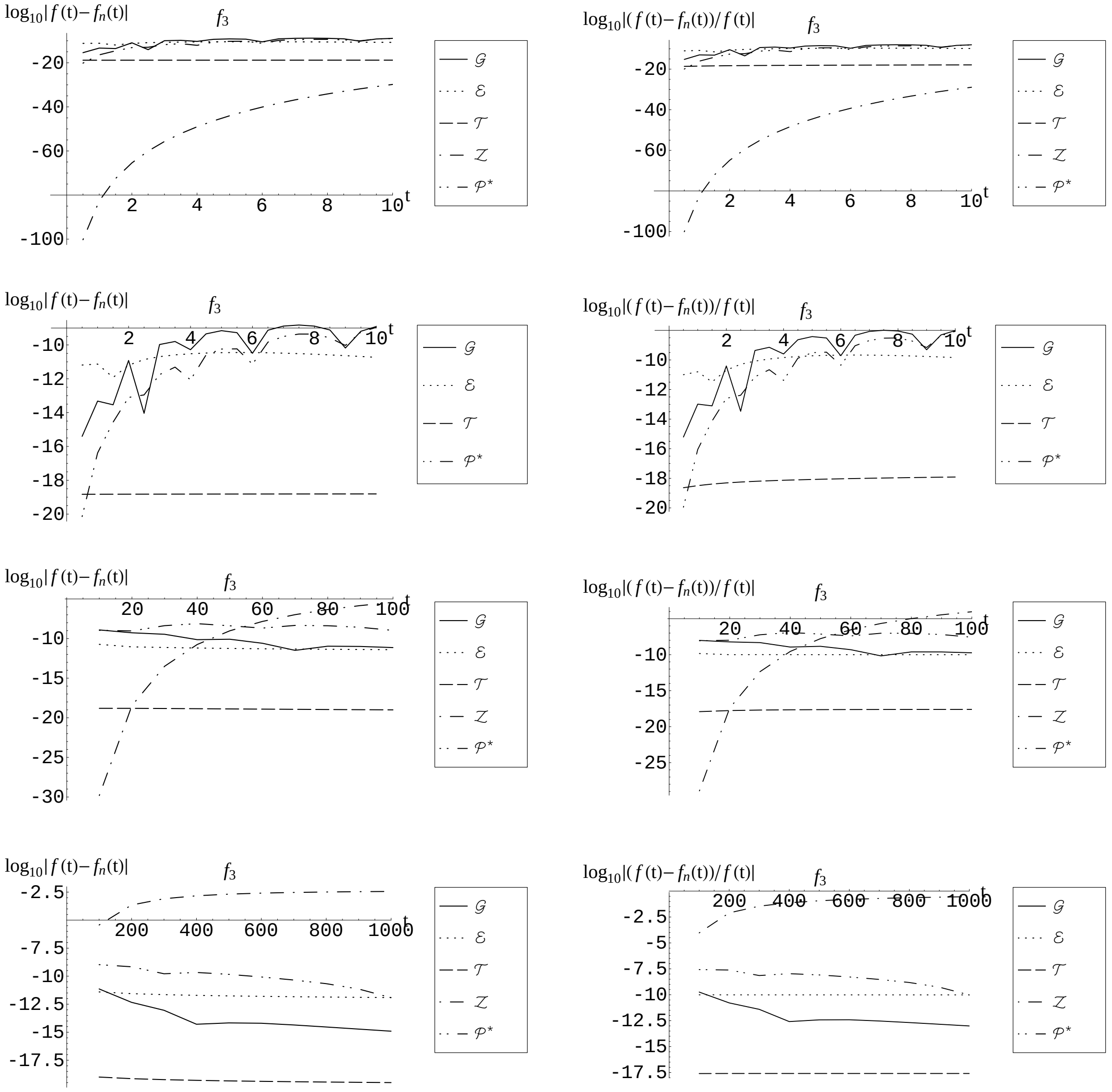


Figure 58: Comparison of inversion for $f_3(t) = e^{-t}I_0(t), \hat{f}_3(s) = \frac{1}{\sqrt{s}\sqrt{s+2}}$ with $n = 30$.

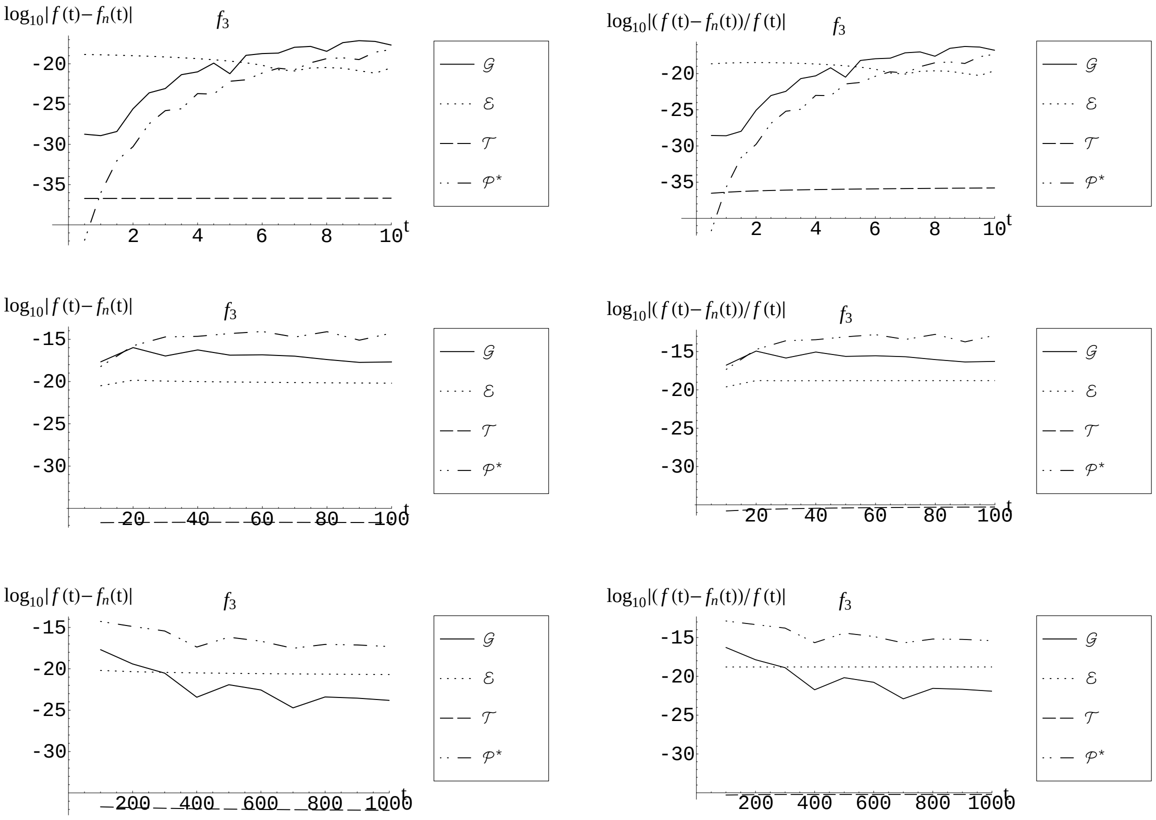


Figure 59: Comparison of inversion for $f_3(t) = e^{-t}I_0(t)$, $\hat{f}_3(s) = \frac{1}{\sqrt{s}\sqrt{s+2}}$ with $n = 60$.

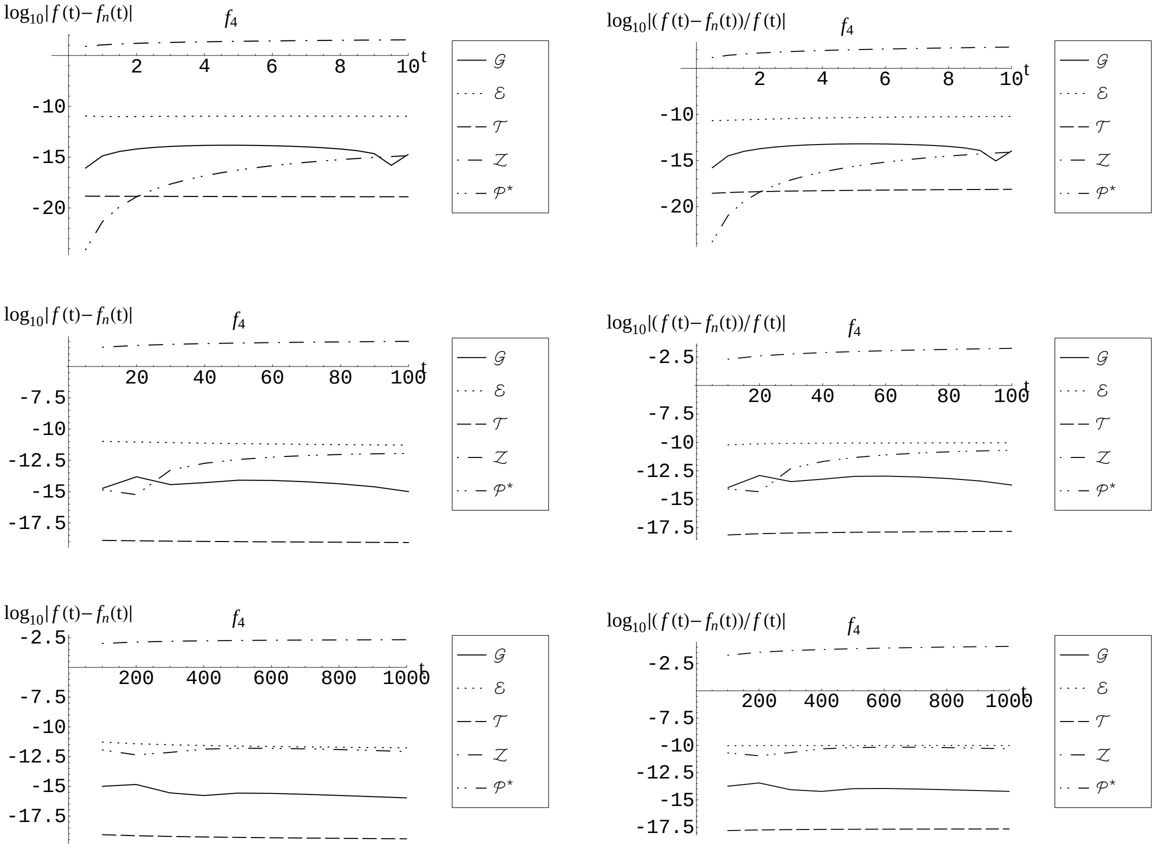


Figure 60: Comparison of inversion for $f_4(t) = e^t \mathrm{erfc}(\sqrt{t})$, $\hat{f}_4(s) = \frac{1}{s+\sqrt{s}}$ with $n = 30$.

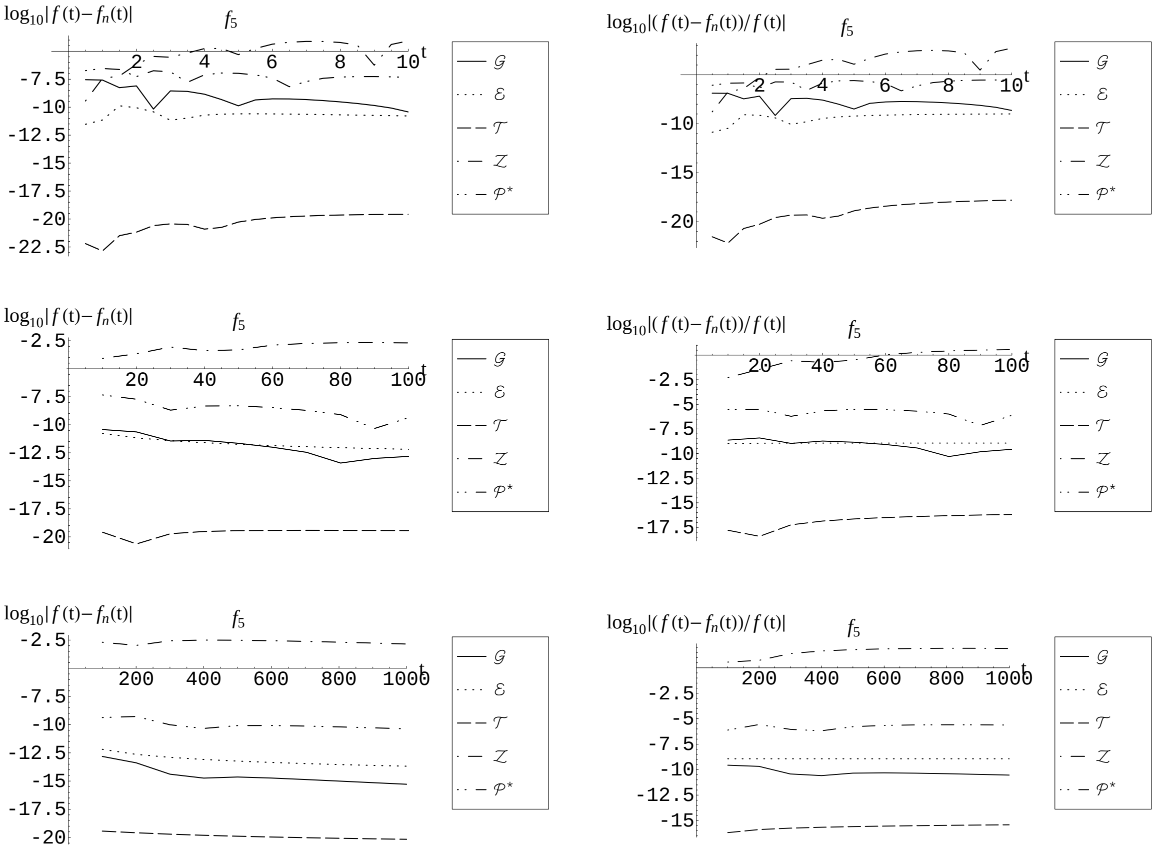


Figure 61: Comparison of inversion for $f_5(t) = \frac{e^{-1/t}}{\sqrt{\pi t^3}}$, $\hat{f}_5(s) = e^{-2\sqrt{s}}$ with $n = 30$.

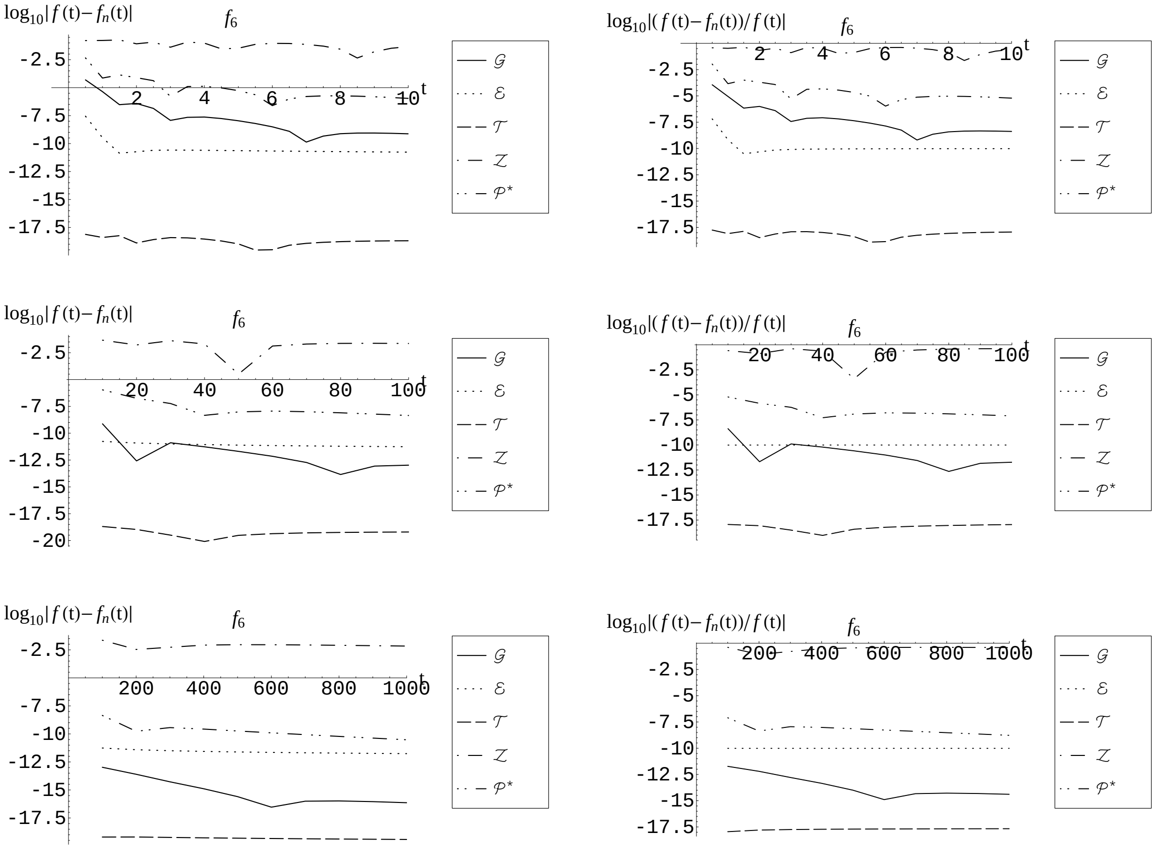


Figure 62: Comparison of inversion for $f_6(t) = \frac{\cos(1/2t)}{\sqrt{\pi t}}$, $\hat{f}_6(s) = \frac{e^{-\sqrt{s}}\cos(\sqrt{s})}{\sqrt{s}}$ with $n = 30$.

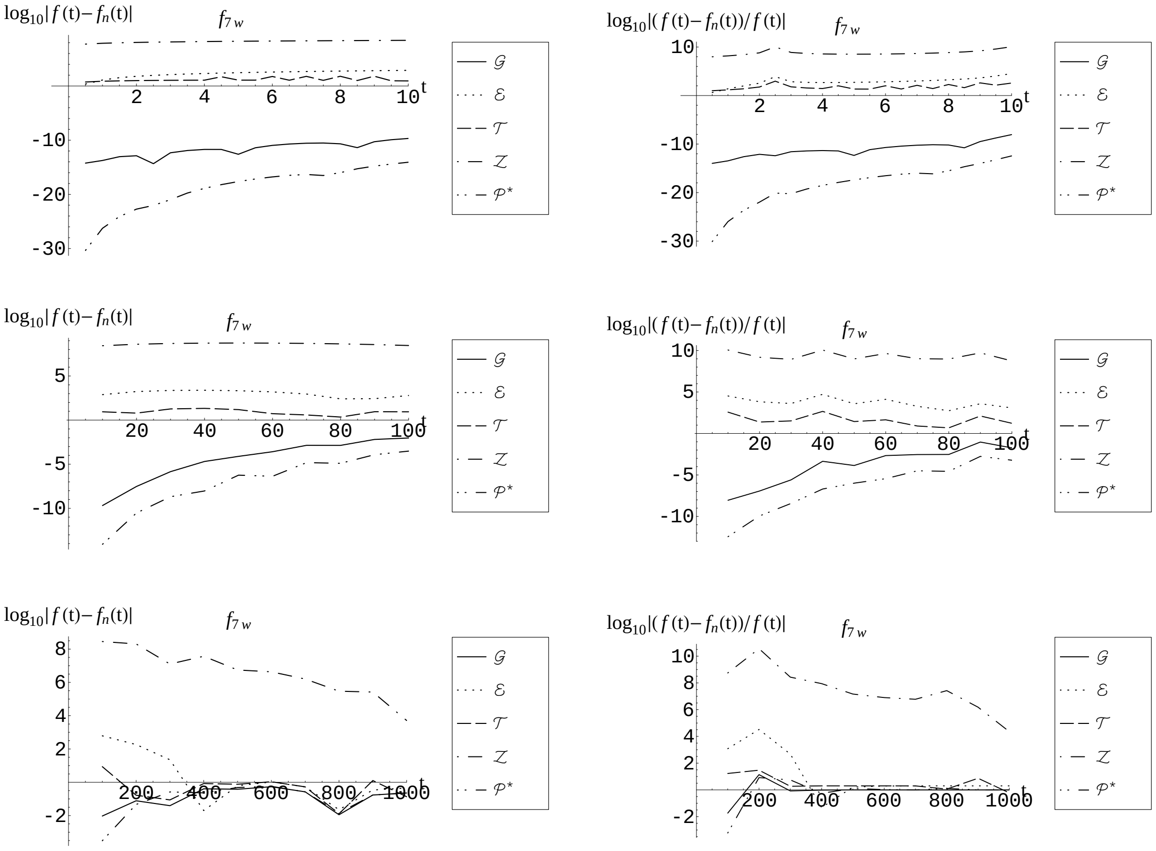


Figure 63: Comparison of inversion for $f_7(t) = \frac{\sin(2\sqrt{t})}{\sqrt{\pi}}$ when the transform is implemented as $\hat{f}_7(s) = \frac{e^{-1/s}}{\sqrt{s^3}}$ with $n = 30$.

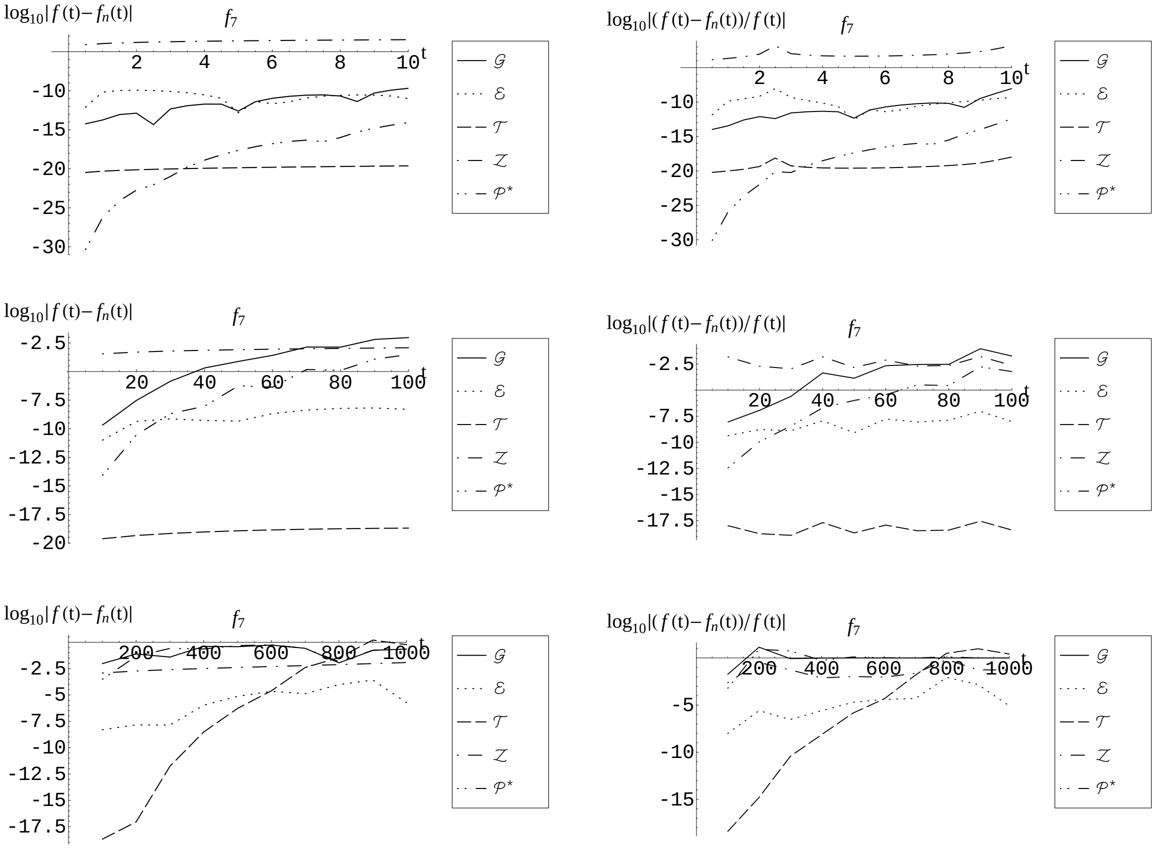


Figure 64: Comparison of inversion for $f_7(t) = \frac{\sin(2\sqrt{t})}{\sqrt{\pi}}$, $\hat{f}_7(s) = \frac{e^{-1/s}}{s\sqrt{s}}$ with $n = 30$.

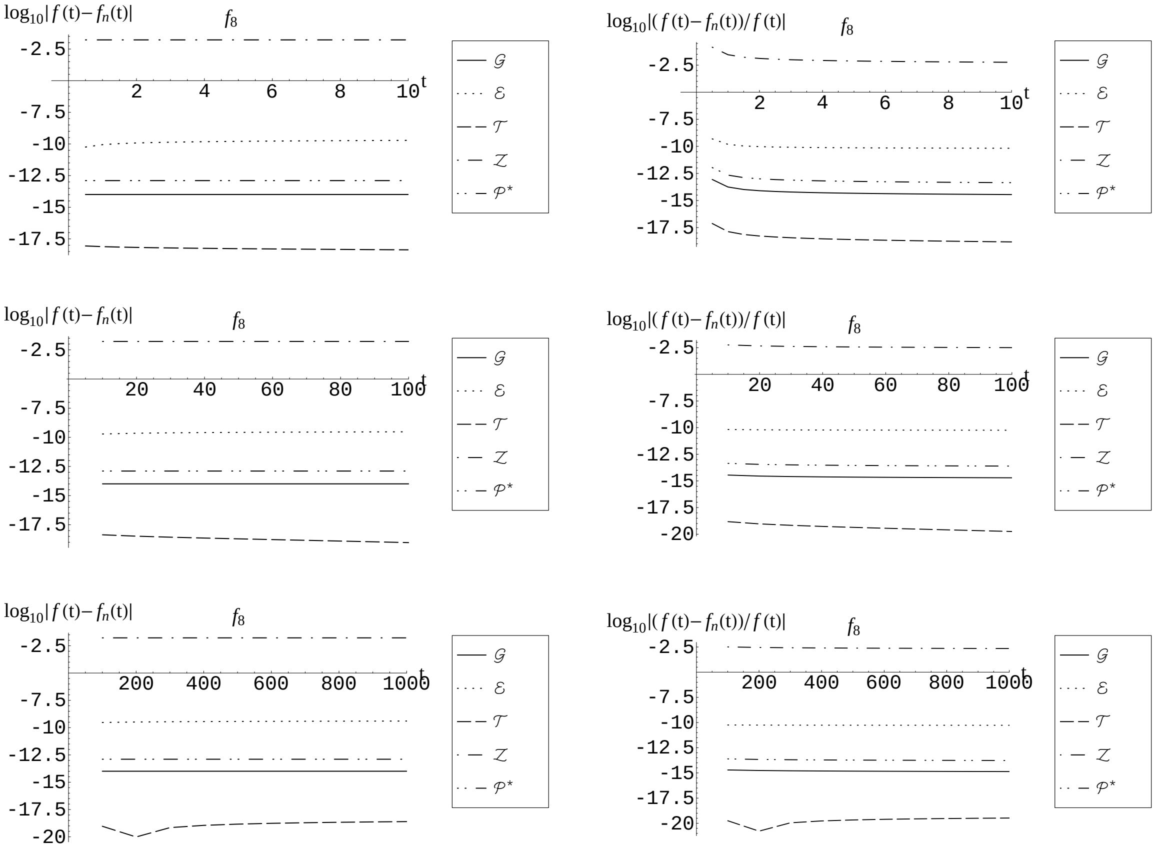


Figure 65: Comparison of inversion for $f_8(t) = \log(t) + \gamma$, $\hat{f}_8(s) = \frac{\log(s)}{s}$ with $n = 30$.

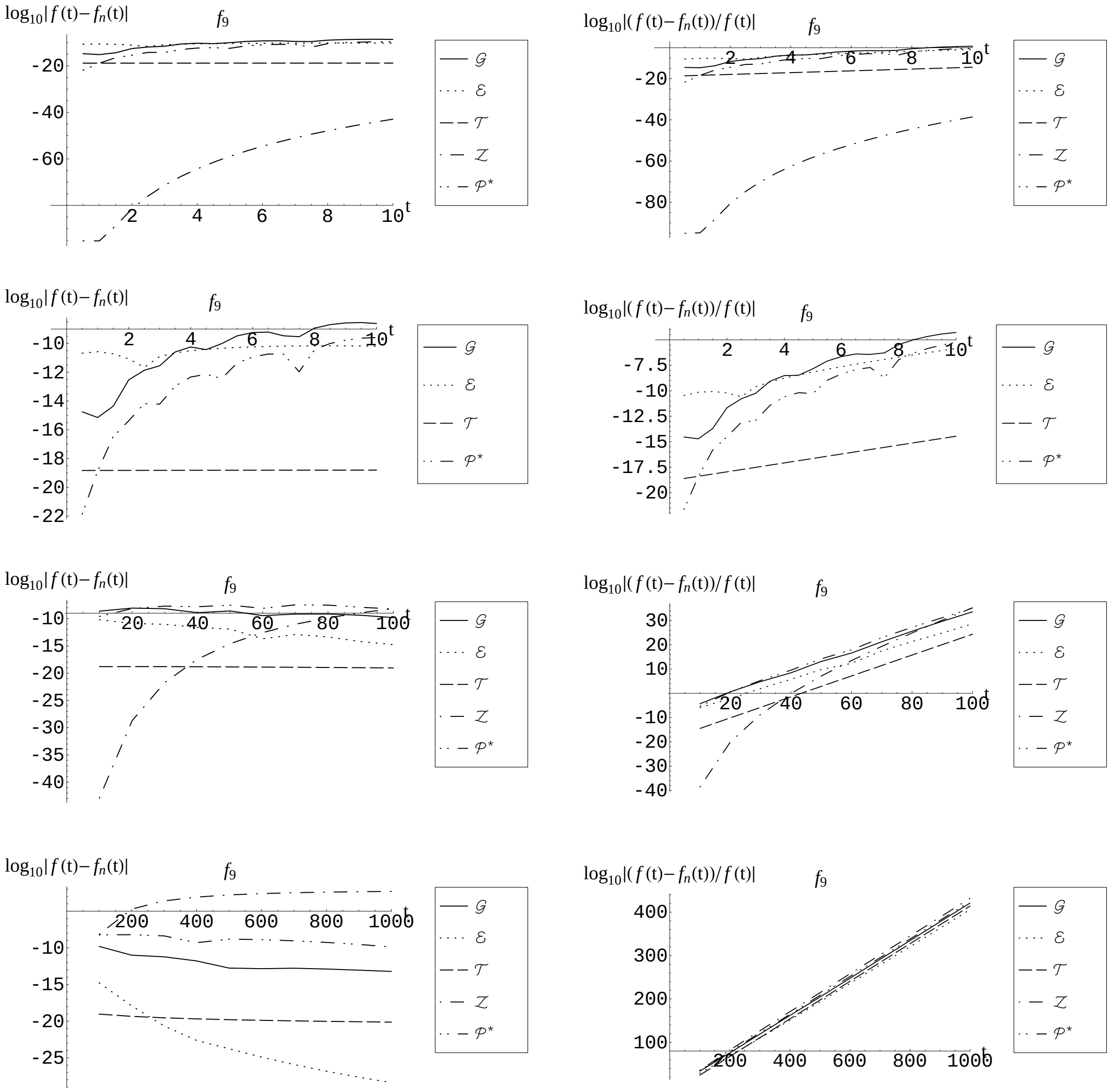


Figure 66: Comparison of inversion for $f_9(t) = e^{-t}$, $\hat{f}_9(s) = \frac{1}{s+1}$ with $n = 30$.

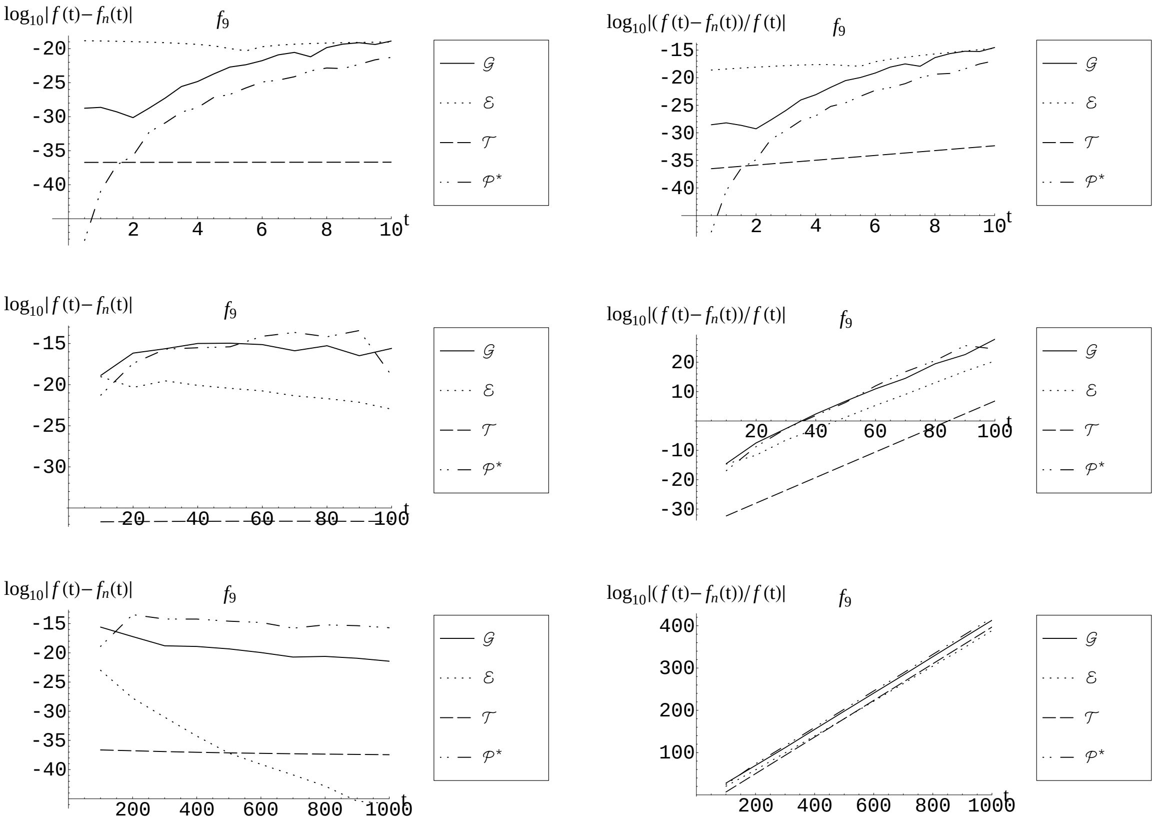


Figure 67: Comparison of inversion for $f_9(t) = e^{-t}$, $\hat{f}_9(s) = \frac{1}{s+1}$ with $n = 60$.

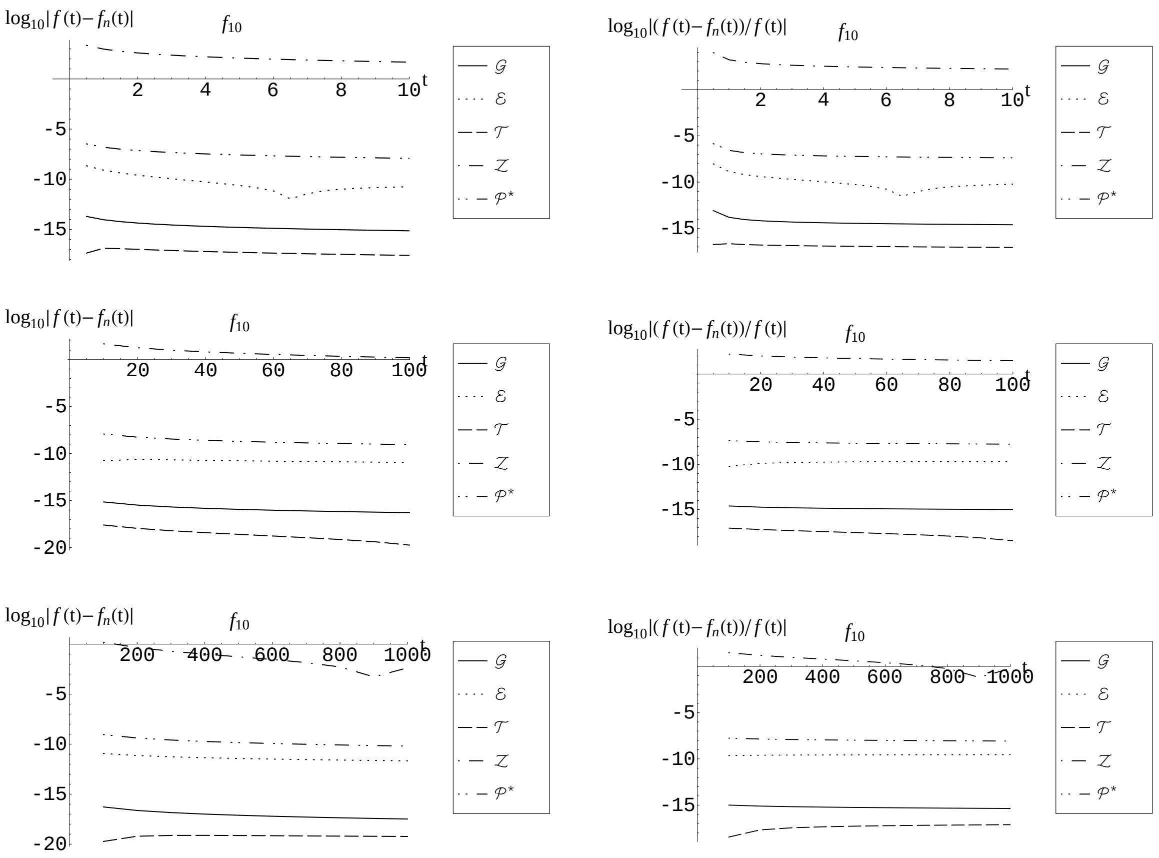


Figure 68: Comparison of inversion for $f_{10}(t) = \frac{\log(t)+\gamma}{t}$, $\hat{f}_{10}(s) = \frac{1}{2}\log^2(s)$ with $n = 30$.

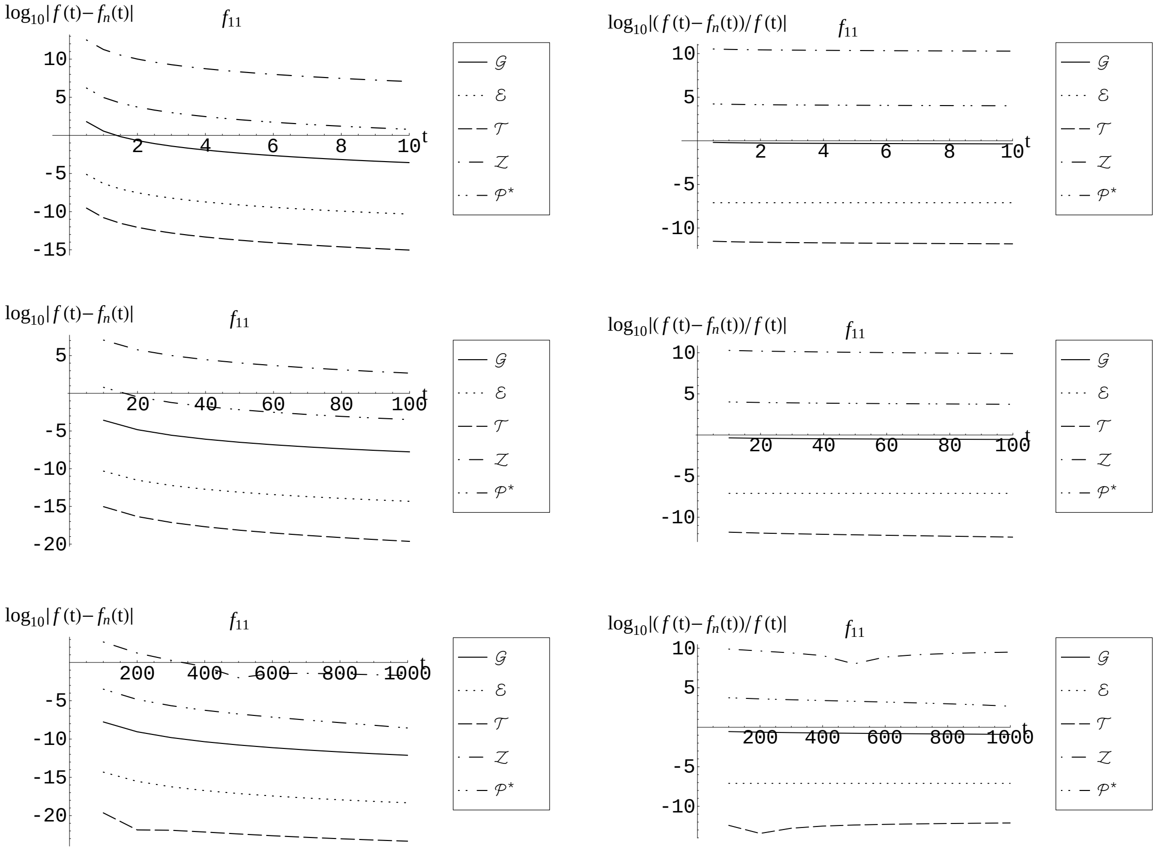


Figure 69: Comparison of inversion for $f_{11}(t) = \frac{6}{t^4}, \hat{f}_{11}(s) = s^3 \log(s)$ with $n = 30$.

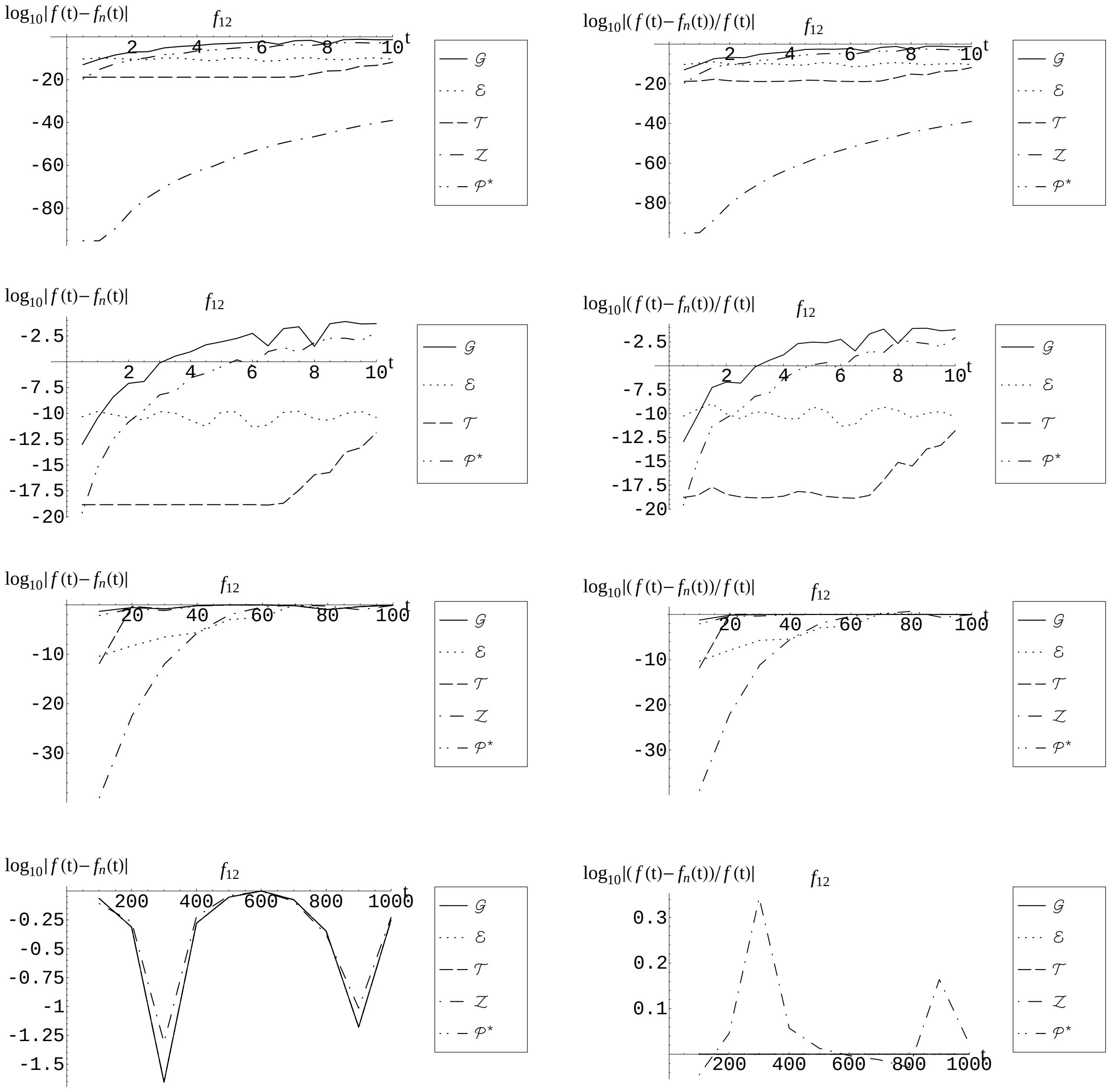


Figure 70: Comparison of inversion for $f_{12}(t) = \cos(t), \hat{f}_{12}(s) = \frac{s}{s^2+1}$ with $n = 30$.

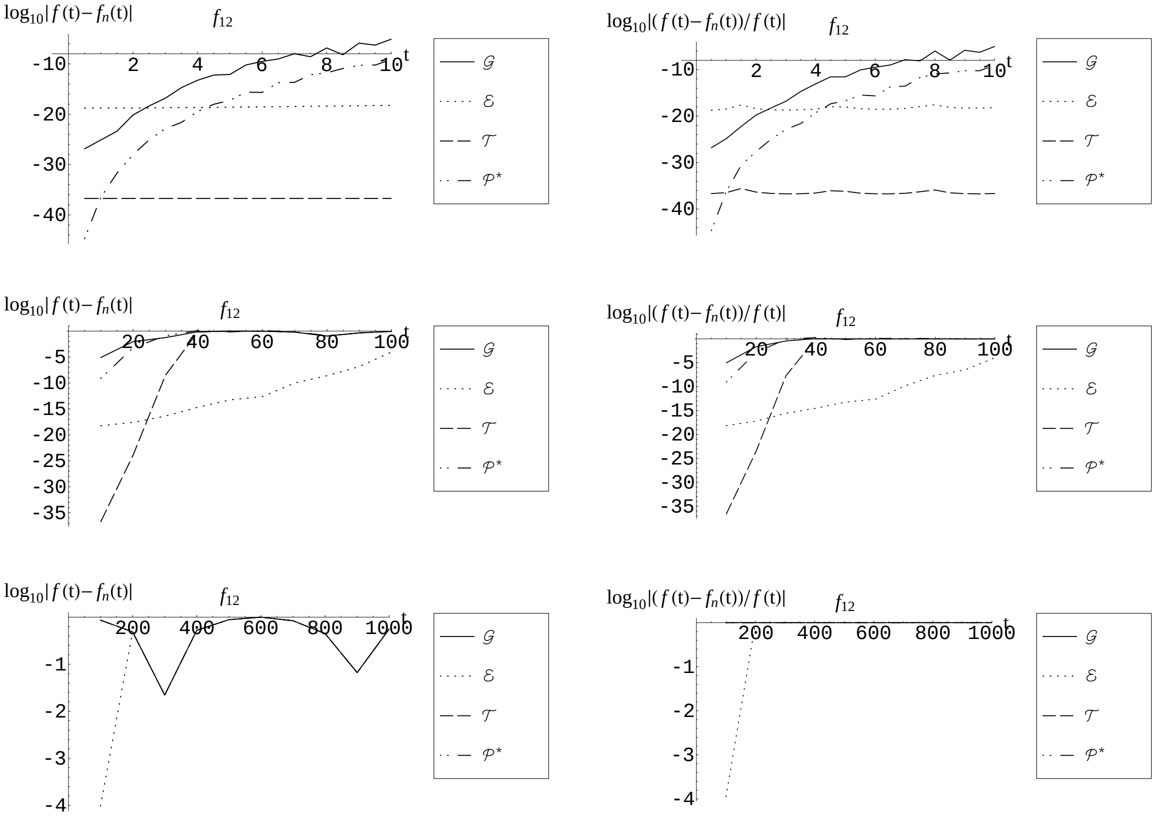


Figure 71: Comparison of inversion for $f_{12}(t) = \cos(t)$, $\hat{f}_{12}(s) = \frac{s}{s^2+1}$ with $n = 60$.

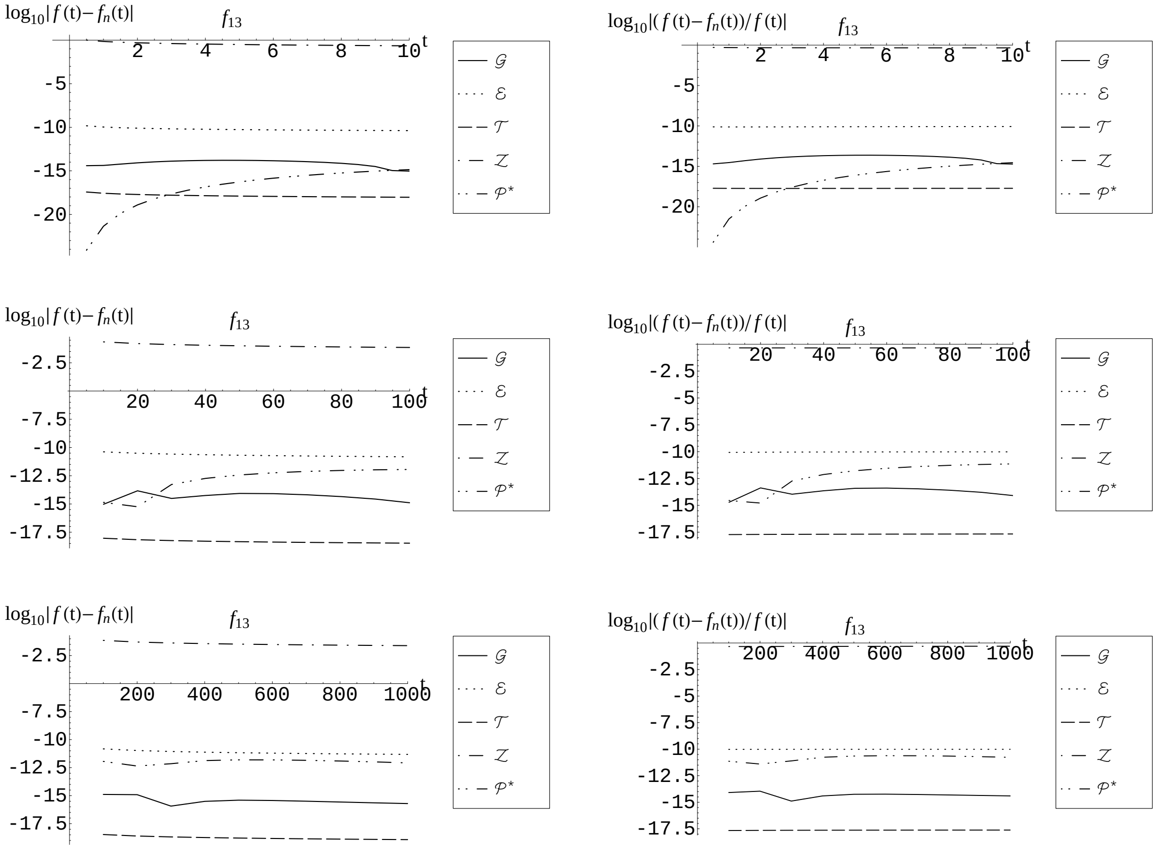


Figure 72: Comparison of inversion for $f_{13}(t) = f_4(t) + \frac{1}{\sqrt{t}}$, $\hat{f}_{13}(s) = \hat{f}_4(s) + \frac{\Gamma(1/2)}{\sqrt{s}}$ with $n = 30$.

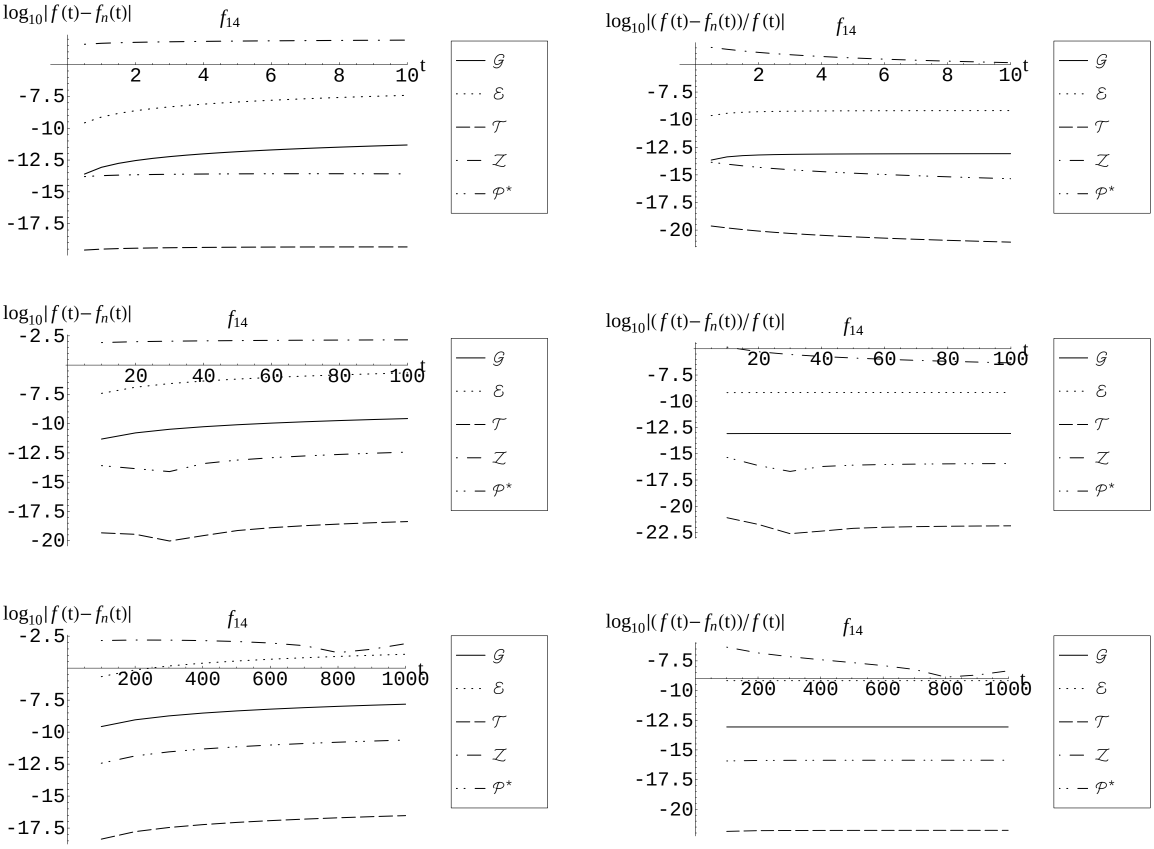


Figure 73: Comparison of inversion for $f_{14}(t) = t^{1/4} + t^{7/4}, \hat{f}_{14}(s) = \frac{\Gamma(5/4)}{s^{5/4}} + \frac{\Gamma(11/4)}{s^{11/4}}$ with $n = 30$.

# 11. Commentary

In this section we discuss the numerical results in Section 10. We applied 5 different numerical inversion algorithms – $\mathcal{T}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{Z}$, and $\mathcal{P}^*$ – to the 14 different Laplace transforms with known inverses displayed in Table 2.

We calculated the values $f(t)$ for a wide range of times (function arguments) $t$: small values ($0.5 \leq t \leq 10$), medium values ($10 \leq t \leq 100$) and large values ($100 \leq t \leq 1000$). The time intervals increase moving down on each page, so that there are 6 plots on each page for each function. We should point out that the function argument has no meaning independent of the function itself. That is easy to see by considering the scaled function

$$f_c(t) \equiv f(ct), \quad t \geq 0 \ , \tag{11}$$

where $c$ is a positive real number. Clearly the value of the function $f_c$ at $t$ is the same as the value of the function $f$ at $ct$.

Perhaps the most important observation from the 14 tables is that the results are not uniformly good. Indeed some of the results are horrendous. Moreover, in many cases, the results strongly depend on the function argument $t$. That is immediately apparent from the first function $f_1(t) \equiv \sin(t)$. For $f_1$, the inversion results – as measured by absolute error or relative error – are useless for all algorithms for all sufficiently large $t$, in particular, for all $t \geq 60$. Essentially the same results are seen for the closely related function $f_{12}(t) \equiv \cos(t)$. On the positive side, for both of these functions the results are good for all suitably small $t$, e.g., for $t \leq 4$.

A similar, and more easily understood, story holds for the simple function $f_9(t) \equiv e^{-t}$. For $t \geq 40$, no algorithm produces a single significant digit: The relative errors satisfy $\log_{10}\{|f_n(t) - f(t)|/|f(t)|\} \geq 0$ for all 5 algorithms. This example is easy to understand, because $f_9(t)$ rapidly approaches 0 as $t$ increases; e.g., $f_9(40) \approx 2.35 \times 10^{-18}$. The first conclusion to draw from these results is that numerical inversion should not be viewed as a method that will be effective for all functions when applied blindly. Some care may be needed.

On the positive side, note that all inversion algorithms with the exception of Zakian ($\mathcal{Z}$) are consistently effective, yielding at least a few significant digits for all times considered, for 9 of the 14 transforms: $f_2$, $f_3$, $f_4$, $f_5$, $f_6$, $f_8$, $f_{10}$, $f_{13}$, and $f_{14}$. The serious difficulties are confined to $f_1$, $f_7$, $f_9$, $f_{11}$, and $f_{12}$.

First, we point out that $\hat{f}_{10}$ and $\hat{f}_{11}$ are *not* bonafide Laplace transforms, because the transform integrals are not properly defined. (The function $1/t^p$ is not integrable over the interval $(0,1)$ for $p \geq 1$.) Instead, $\hat{f}_{10}$ and $\hat{f}_{11}$ are pseudotransforms, as discussed in Sections 12-14 and the Appendix of Deutsch (1974). Thus we should not accept responsibility for numerically inverting them correctly. We should take the good performance for $\hat{f}_{10}$ as an added bonus. Moreover, the difficult function $f_{11}(t) \equiv 6t^{-4}$ is itself a power, but a negative power, so it can be addressed by power algorithms. However, we report results only for the default power set $\mathcal{P}^*$, which contains no powers less than $-1$.

But there are still remaining issues with some of the bonafide transforms. At this point, it is worth repeating a point made in Abate and Whitt (1992, 1995). In many applications, functions of interest are known to have special structure, and that special structure can often be used to advantage to make numerical inversion easier. In particular, we have emphasized probability applications, where it is often of interest to compute probability *cumulative distribution functions* (cdf's) $F(t)$ on the positive halfline, which are nondecreasing functions with $F(0) = 0$ and $\lim_{t\to\infty} F(t) = 1$; i.e., cdf's are monotone functions with $0 \leq F(t) \leq 1$ for all $t$. Equivalently, we can calculate complementary cdf's (ccdf's) $F^c(t) \equiv 1 - F(t)$, which converge to 0 as $t \to \infty$.

In contrast, numerical inversion can be difficult with rapidly oscillating functions such as

$$h_c(t) \equiv \sin(ct), \quad t \geq 0 \ , \tag{12}$$

which has associated Laplace transform

$$\hat{h}_c(s) = \frac{c}{s^2 + c^2} \ . \tag{13}$$

For real $s$, we have

$$|\hat{h}_c(s)| \leq \frac{1}{c} \ , \tag{14}$$

so that $|\hat{h}_c(s)|$ can be made arbitrarily small by making $c$ suitably large (and making $h_c$ rapidly oscillating).

These functions show the fundamental instability of the inverse operation: a small change in any Laplace transform can produce a big change in its inverse function. To see that, let $f$ be a real-valued function with Laplace transform $\hat{f}$. Then let

$$f_c(t) \equiv f(t) + h_c(t), \quad t \geq 0 \ , \tag{15}$$

where $h_c$ is defined in (12). Correspondingly, let

$$\hat{f}_c(s) \equiv \hat{f}(s) + \hat{h}_c(s) \ . \tag{16}$$

As $c$ increases $\hat{f}_c$ approaches $\hat{f}$, but $f_c$ remains distant from $f$. Such difficulties with rapidly oscillating functions are avoided if we restrict attention to monotone functions $f$.

Nevertheless, one of the functions causing difficulties here is the monotone function $f_9(t) \equiv e^{-t}$. Indeed, it is a relatively simple ccdf. We have no difficulties for smaller $t$, such as $t = 1$, but we do have difficulties for large $t$, when $e^{-t}$ itself can be extremely small. Fortunately, that problem for large $t$ for the exponential function is well understood. It is easily addressed by scaling, as shown by Choudhury and Whitt (1997). Indeed, the exponential function is treated in Example 4.1 of Choudhury and Whitt (1977). For that reason, we evaluate the performance of the inversion algorithms for inverting $\hat{f}_9$ by focusing on the single time $t = 1$. We conclude that the bad performance observed for $t \geq 60$ in Figure 66 can be disregarded.

That leaves only the trigonometric functions. From the discussion above, we are not surprised by the inversion problems encountered for the oscillating functions $f_1(t) \equiv \sin(t)$, $f_7(t) \equiv \sin(2\sqrt{t})/\sqrt{\pi}$, $f_{12}(t) \equiv \cos(t)$ for large $t$. They all involve sines and cosines. That corresponds to considering the function $h_c$ for $c = t$ and $t = 1$. Difficulties with the transforms $\hat{f}_1(s) \equiv 1/(1+s^2)$ and $\hat{f}_{12}(s) \equiv s/(1+s^2)$ were noted by Abate and Valko (2004), because they both have singularities off the real axis, in particular at $+i$ and $-i$. Thus these transforms fail to be in their class of "good" transforms $\mathbf{F}$.

# 12. Branch Cuts in Mathematica

There is a difficulty with computing half powers of complex $s$ in *Mathematica* that can cause difficulties in the numerical inversions of $\hat{f}_3$ and $\hat{f}_7$. Direct applications of *Mathematica* put the branch cuts in the positive half plane. That is avoided by expressing

$$\hat{f}_3(s) \equiv \frac{1}{\sqrt{s}\sqrt{s+2}} \quad \text{instead of} \quad \hat{f}_{3w}(s) \equiv \frac{1}{\sqrt{s(s+2)}} \tag{17}$$

and

$$\hat{f}_7(s) \equiv \frac{e^{-1/s}}{s\sqrt{s}} \quad \text{instead of} \quad \hat{f}_{7w}(s) \equiv \frac{e^{-1/s}}{\sqrt{s^3}}\ . \tag{18}$$

We add the subscript $w$ to indicate the natural but *wrong* form to use. We displayed results for both the good and bad forms.

Figure 74 shows the real (left) and imaginary (right) parts of the transform $\hat{f}_3(s) = \frac{1}{\sqrt{s}\sqrt{s+2}}$. We can see that this transform has a branch cut on the left half of the real axis from $s = 0$ to $s = -2$. In Figure 75 we see the transform of the same inverse when it is implemented as $\hat{f}_{3w}(s) = \frac{1}{\sqrt{s(s+2)}}$. The transform $\hat{f}_{3w}(s)$ has, in addition to the branch cut from $s = 0$ to $s = -2$, another branch cut parallel to the imaginary axis along the line $-1 + ic, c \in (-\infty, \infty)$. As we can see in Figure 76, the additional branch cut of this implementation of the transform of $f_3$ intersects with the path of nodes prescribed by the Talbot nodes, which explains why, as shown in Figure 57, Talbot's algorithm does not work well on $\hat{f}_{3w}(s)$.

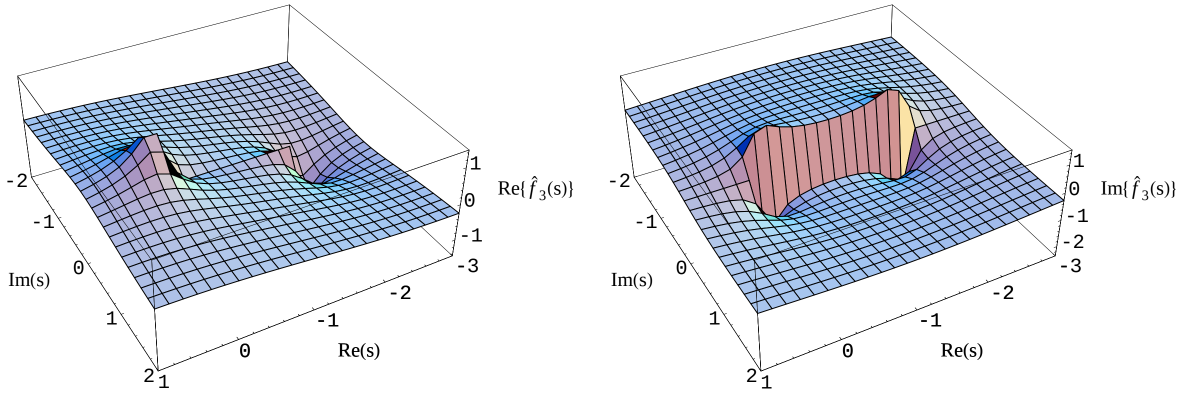


Figure 74: The branch cut of the transform $\hat{f}_3(s) = \frac{1}{\sqrt{s}\sqrt{s+2}}$.

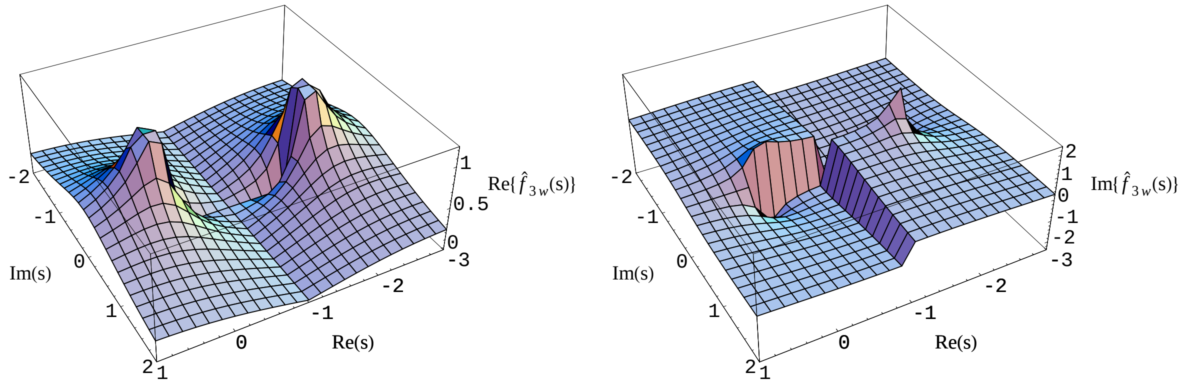


Figure 75: The branch cuts of the transform $\hat{f}_{3w}(s) = \frac{1}{\sqrt{s(s+2)}}$.

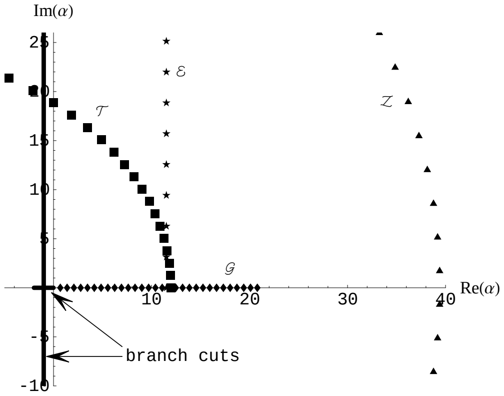


Figure 76: The branch cuts of the transform $\hat{f}_{3w}(s) = \frac{1}{\sqrt{s(s+2)}}$ and the nodes of $\mathcal{G}$, $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{Z}$.

In a similar vain, Figure 77 shows the real (left column) and imaginary (right column) parts of the transform $\hat{f}_7(s) = \frac{e^{-1/s}}{s\sqrt{s}}$ from two different viewpoints. This transform has a branch cut that coincides with the left half of the real axis. In Figure 78 we see the transform of the same inverse when it is implemented as $\hat{f}_{7w}(s) = \frac{e^{-1/s}}{\sqrt{s^3}}$. The transform $\hat{f}_{7w}(s)$ has 3 branch cuts at angles $2\pi/3$ radians with each other, where one of them is the left half of the real axis. As we can see in Figure 79, the 2 additional branch cuts of this implementation of the transform of $f_7$ intersect with the path of nodes prescribed by the nodes of Talbot, Euler and Zakian, which explains why, as shown in Figure 63, the Talbot, Euler and Zakian algorithms do not work well on $\hat{f}_{7w}(s)$. We note that Zakian's algorithm does not work even on $\hat{f}_7(s)$, because the function $f_7$ is not analytic around the origin.

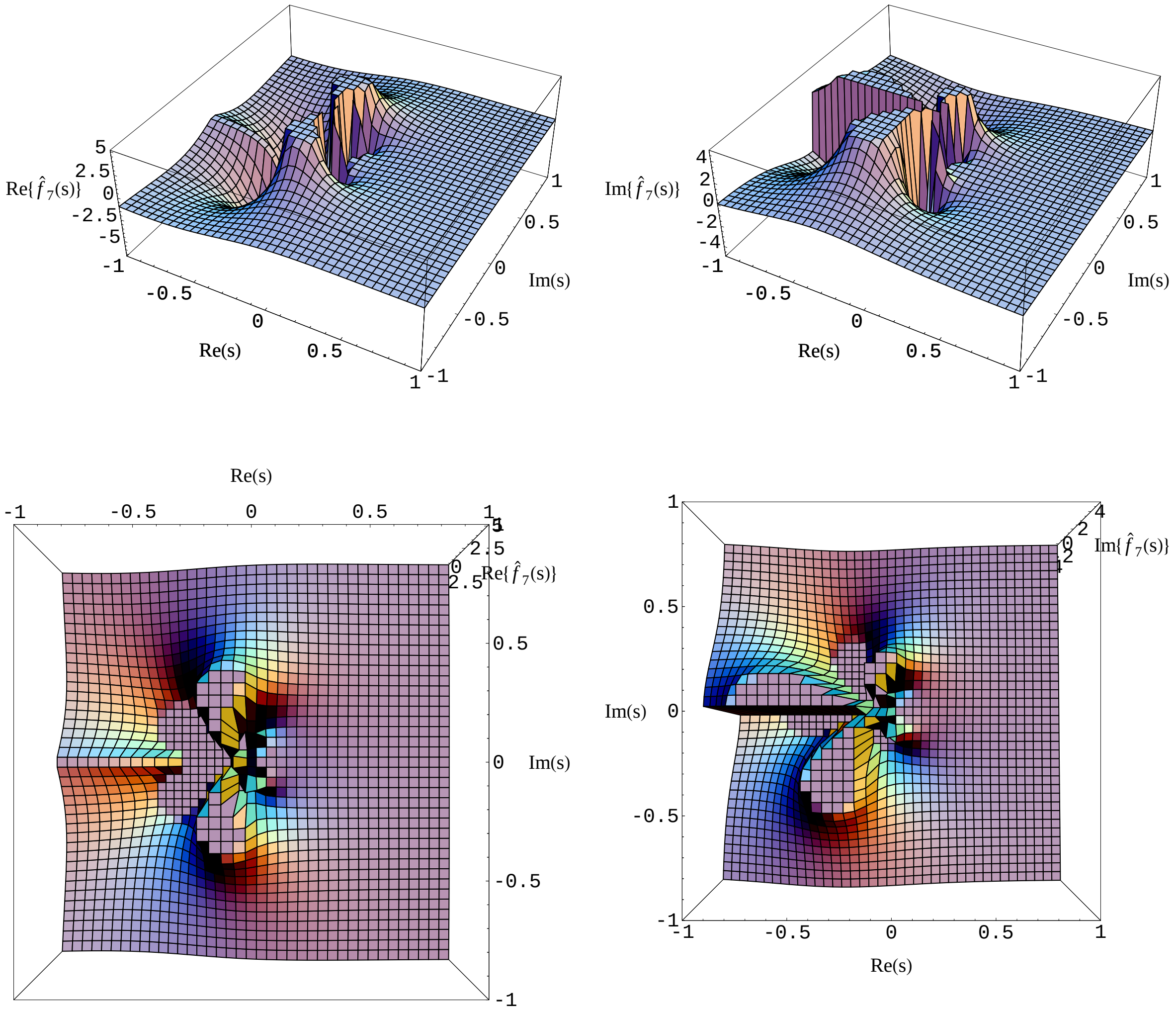


Figure 77: The branch cut of the transform $\hat{f}_7(s) = \frac{e^{-1/s}}{s\sqrt{s}}$ from two different viewpoints.

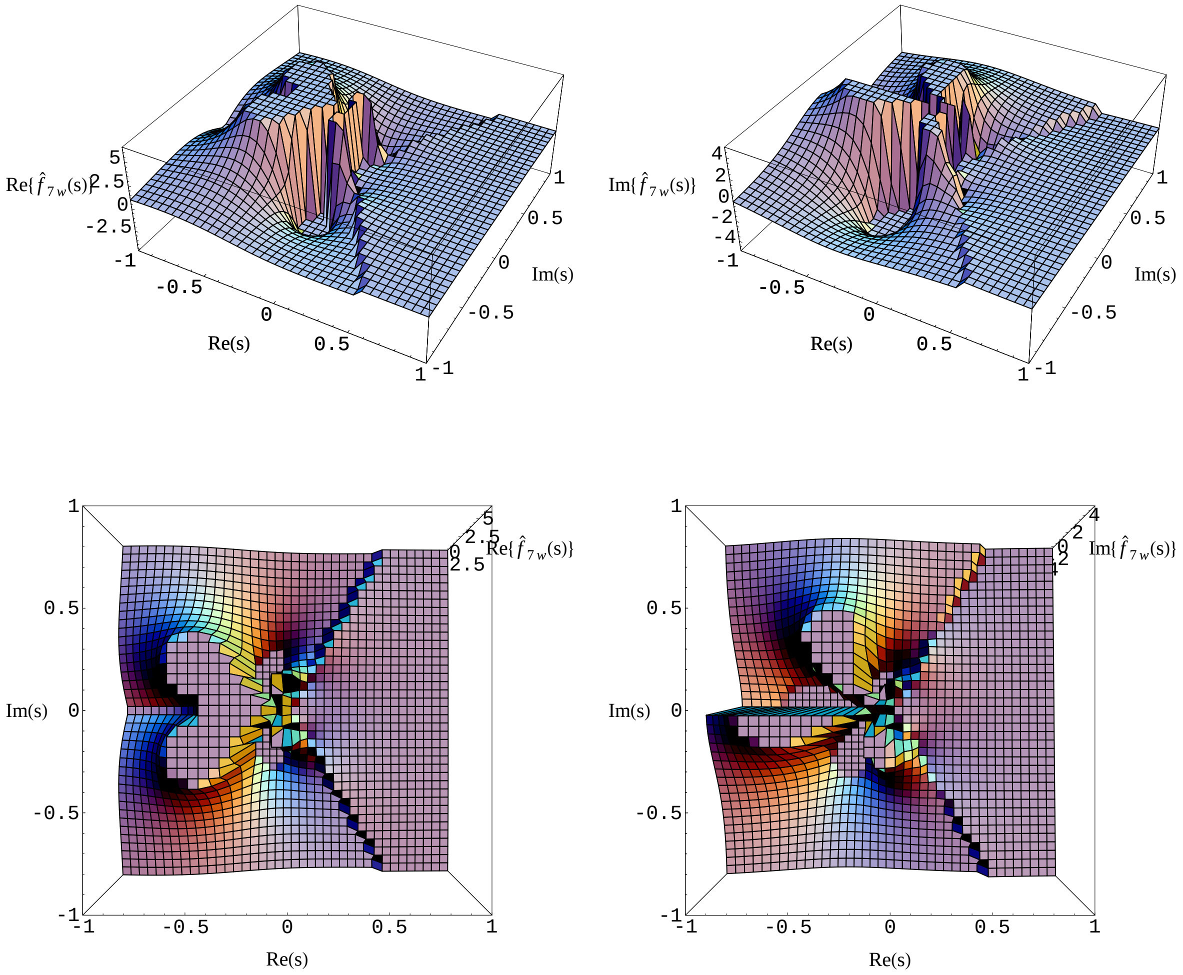


Figure 78: The branch cuts of the transform $\hat{f}_{7w}(s) = \frac{e^{-1/s}}{\sqrt{s^3}}$ from two different viewpoints.

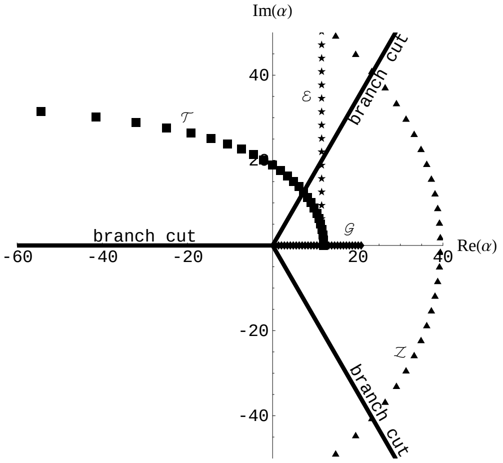


Figure 79: The branch cuts of the transform $\hat{f}_{7w}(s) = \frac{e^{-1/s}}{\sqrt{s^3}}$ and the nodes of $\mathcal{G}$, $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{Z}$.

# References

Abate, J. and P. P. Valko. 2004. Multi-precision Laplace inversion. *Int. J. Numer. Meth. Engng.* 60, 979-993.

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

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

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

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

Avdis, E. and W. Whitt. 2006. Power algorithms for inverting Laplace transforms. Available at http://columbia.edu/∼ww2040.

Baker, G. A. and P. Graves-Morris. 1996. *Padé Approximants*, second edition, Encyclopedia of Mathematics and its Applications 59, Cambridge University Press.

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

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

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

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

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

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

Wellekens, C. J., 1970. Generalization of Vlach's method for the numerical inversion of the

Laplace transform. *Electronic Letters* 6, 742–744.

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

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

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

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