A Fast Laplace Transform Based on Laguerre Functions*

John Strain†

Courant Institute of Mathematical Sciences

251 Mercer Street

New York, NY 10012

January 13, 2000

Keywords: Laplace Transform, Fast Algorithms, Laguerre Polynomials.

AMS(MOS) Subject Classifications: 44A10, 33A65, 65R10.

Electronic Mail: strain@math.princeton.edu

*Supported by DARPA/AFOSR Contract No. F-49620-87-C-0065 and a NSF Mathematical Sciences Postdoctoral Research Fellowship.

†Current address: Department of Mathematics, Princeton University, Fine Hall, Washington Road, Princeton, NJ 08544.

Abstract

In this paper, we present a fast algorithm which evaluates a discrete Laplace transform with N points at M arbitrarily distributed points in $C(N+M)$ work, where C depends only on the precision required. Our algorithm breaks even with the direct calculation at $N = M = 20$, and achieves a speedup of 1000 with 10000 points. It is based on a geometric divide and conquer strategy, combined with the manipulation of Laguerre expansions, via a dilation formula for Laguerre functions.

1 Introduction

Many situations in applied mathematics require the use of the *Laplace transform*

$$\hat{f}(t) = \int_0^\infty e^{-ts} f(s)\, ds \tag{1}$$

of a function f defined on $R^+ = (0, \infty)$. Practically, one often evaluates $\hat{f}$ by applying a numerical quadrature rule to the integral (1). Thus practical situations usually require evaluation of the discrete Laplace transform

$$\hat{f}_i = \sum_{j=1}^{N} f_j\, e^{-t_i s_j}\ , \tag{2}$$

for $i = 1, \ldots, M$, where the coefficients f_j depend on the values $f(s_j)$ and the weights of the quadrature formula and t_i and s_j are given positive real numbers. A special case is the evaluation of polynomials, or more generally the discrete Mellin transform.

A difficulty, however, is the high cost of evaluating (2) when N or M is large; direct evaluation of (2) costs $O(NM)$ work. In this paper, we present

an algorithm for evaluating the discrete Laplace transform (2), within a user-specified precision ϵ relative to $F = \sum |f_j|$, in $O(M+N)$ arithmetic operations, with a constant in $O(M+N)$ depending only on ϵ and the interval in which all the points t_i and s_j lie. Our algorithm is based on certain properties of Laguerre polynomials, and takes advantage of the multiplicative convolution structure of the transform. It will speed up calculations in which M and N are large and the work is dominated by the arithmetic operations necessary to evaluate the sums (2).

Our algorithm generalizes to evaluate convolutions with a fixed Laguerre function. Since any smooth rapidly decaying function on R^+ has a rapidly converging Laguerre expansion, our algorithm thus generalizes to permit fast multiplicative convolution on R^+, with any such kernel.

A fast Laplace transform has also been developed by Rokhlin [2] using an approximation-theoretic approach quite different from the special-function-theoretic approach presented here. We have not attempted to make a detailed comparison of the two algorithms. They are similar in some respects; both use a geometric subdivision of R^+ but Rokhlin's algorithm is based on Chebyshev expansions, while our algorithm uses special functions especially suited to the Laplace transform.

The structure of the paper is standard. §1 presents the special function definitions and identities we will need, §2 explains the algorithm, and §3 presents the numerical results. §4 discusses convolution with Laguerre functions and more general kernels, and §5 states our conclusions.

2 Formulas and special functions

The Laguerre polynomials are discussed in [3]. We shall define them by the Rodrigues formula

$$L_n(t) = \frac{e^t}{n!} D^n \left(t^n e^{-t} \right) \qquad\qquad t \in \mathbf{R}$$

where $D = d/dt$, though they are most conveniently evaluated by the three-term recurrence

$$(n+1)L_{n+1}(t) - (2n+1-t)L_n(t) + nL_{n-1}(t) = 0.$$

We will need only one identity involving Laguerre polynomials. This is the dilation formula which expresses the action of the group of dilations of R^+ on a given Laguerre polynomial. It reads [3]

$$L_n(ts) = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} t^{n-k}(1-t)^k L_{n-k}(s) \tag{3}$$

From this identity, we can derive the main formula used in our fast Laplace transform. We define the Laguerre functions by

$$\mathcal{L}_n(t) = L_n(t)e^{-t}.$$

This is not the standard definition, but the appropriate one for our situation. These Laguerre functions form a biorthonormal set with Laguerre polynomials rather than themselves forming an orthonormal set.

We now require a dilation formula for Laguerre functions. To derive such a formula, fix t and assume a Laguerre expansion of the form

$$\mathcal{L}_n(ts) = \sum_{k=0}^{\infty} g_k^n(t)\mathcal{L}_k(s).$$

Such an expansion exists, since $\mathcal{L}_n(ts)$ is a smooth function on R^+ which decreases rapidly at ∞. The coefficients are found by using the biorthogonality relation [3]

$$\int_0^\infty \mathcal{L}_k(t)L_m(t)dt = \delta_{km}.$$

Indeed, multiply the definition of $g_k^n(t)$ by $L_m(s)$ and integrate term by term to get

$$g_m^n(t) = \int_0^\infty \mathcal{L}_n(ts)L_m(s)ds = \frac{1}{t}\int_0^\infty \mathcal{L}_n(s)L_m(s/t)ds.$$

Use the dilation formula for Laguerre polynomials and biorthogonality to get

$$\begin{aligned} g_m^n(t) &= \frac{1}{t}\int_0^\infty \mathcal{L}_n(s)\sum_{k=0}^{m}\frac{m!}{k!(m-k)!}\left(\frac{1}{t}\right)^{m-k}\left(1-\frac{1}{t}\right)^k L_{m-k}(s)ds \\ &= 0 \quad \text{if} \quad m<n \\ &= \frac{1}{t}\frac{m!}{n!(m-n)!}\left(\frac{1}{t}\right)^n\left(1-\frac{1}{t}\right)^{m-n} \qquad \text{if} \quad m \geq n. \end{aligned}$$

Thus we have a dilation formula for Laguerre functions:

$$\mathcal{L}_n(ts) = \sum_{k=0}^{\infty}\left(\frac{1}{t}\right)^{n+1}\left(1-\frac{1}{t}\right)^k\frac{(n+k)!}{n!k!}\mathcal{L}_{n+k}(s). \tag{4}$$

(This formula can be found in the textbook [1], on page 215. A quicker proof can be constructed by using the generating function of Laguerre polynomials.) An important special case ($n=0$) is

$$e^{-ts} = \frac{1}{t}\sum_{k=0}^{\infty}\left(1-\frac{1}{t}\right)^k\mathcal{L}_k(s). \tag{5}$$

Since $\mathcal{L}_k(s) \leq e^{-s/2}$, the series (4) and (5) converge geometrically fast in the region $|1-1/t| \leq r < 1$.

We are now ready to derive the expansion which forms the basis of our algorithm: Put $t=xy$ and $s=z$ in (5) to get

$$e^{-xyz} = \sum_{k=0}^{\infty}\frac{1}{xy}\left(1-\frac{1}{xy}\right)^k\mathcal{L}_k(z). \tag{6}$$

Apply the binomial theorem in the form

$$\begin{aligned}\left(1-\frac{1}{xy}\right)^k &= 2^{-k}\left[(1+1/x)(1-1/y)+(1+1/y)(1-1/x)\right]^k \\ &= 2^{-k}\sum_{j=0}^{k}\frac{k!}{j!(k-j)!}(1+1/x)^{k-j}(1-1/y)^{k-j}(1+1/y)^j(1-1/x)^j.\end{aligned}$$

Thus (6) becomes, after reversing the order of summation and shifting the index k,

$$e^{-xyz} = \frac{1}{xy}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty}2^{-(j+k)}\frac{(j+k)!}{j!k!}(1+1/x)^k(1-1/y)^k(1+1/y)^j(1-1/x)^j\mathcal{L}_{j+k}(z). \tag{7}$$

This series separates the variables x, y and z in a way convenient for the algorithm.

Finally, we will need to estimate the truncation error incurred by truncating (7) after say p^2 terms. The error is $E_p - F_p$ where

$$E_p = \frac{1}{xy}\left[\sum_{j=p+1}^{\infty}\sum_{k=0}^{\infty}+\sum_{k=p+1}^{\infty}\sum_{j=0}^{\infty}\right]z_1^k z_2^j\frac{(j+k)!}{j!k!}\mathcal{L}_{j+k}(z),$$

$z_1 = (1+1/x)(1-1/y)/2$, $z_2 = (1+1/y)(1-1/x)/2$ and

$$F_p = \frac{1}{xy}\sum_{j=p+1}^{\infty}\sum_{k=p+1}^{\infty}z_1^k z_2^j\frac{(j+k)!}{j!k!}\mathcal{L}_{j+k}(z).$$

We have split the error up like this so that the largest error term E_p can be bounded by the dilation formula (4) for Laguerre functions: we have

$$E_p = \frac{1}{xy}\left[\sum_{j=p+1}^{\infty}z_2^j\xi_1^{j+1}\mathcal{L}_j(\xi_1 z)+\sum_{k=p+1}^{\infty}z_1^k\xi_2^{k+1}\mathcal{L}_k(\xi_2 z)\right] \tag{8}$$

where $z_1 = 1-1/\xi_1$ and $z_2 = 1-1/\xi_2$. The uniform bound $|\mathcal{L}_k(z)| \le e^{-z/2}$ and the formula for the tail of a geometric series give the error bound

$$|E_p| \le \frac{\xi_1}{xy}e^{-\xi_1 z/2}\frac{(\frac{z_2}{1-z_1})^{p+1}}{1-\frac{z_2}{1-z_1}}+\frac{\xi_2}{xy}e^{-\xi_2 z/2}\frac{(\frac{z_1}{1-z_2})^{p+1}}{1-\frac{z_1}{1-z_2}}.$$

Some algebra reduces this expression to

$$|E_p| \le e^{-xyz/(1+x-y+xy)} \left(1 - \frac{2}{1+x-y+xy}\right)^{p+1} + e^{-xyz/(1-x+y+xy)} \left(1 - \frac{2}{1-x+y+xy}\right)^{p+1}$$

This bound is useful when z lies anywhere in $(0,\infty)$ and x and y are restricted to lie near 1: If we take x and y in a geometric interval $(1/q, q)$ with $1 \le q < \frac{1+\sqrt{5}}{2}$, an easy calculation shows that

$$|E_p| \le 2e^{-z/q^2(1+q^2)} \left[\frac{q - 1/q}{2 - (q - 1/q)}\right]^{p+1}. \tag{9}$$

Note that $q - 1/q$ is just the length of the interval containing x and y.

The other error term F_p cannot be bounded by the dilation formula, because the sums both run from $p+1$ to ∞. We bound it crudely, assuming that q is small, by using that $(j+k)!/j!k! \le 2^{j+k}$ and that $\mathcal{L}_{j+k}(z) \le e^{-z/2} \le 1$. The resulting bound tremendously overestimates F_p, but it is sharp enough for our purposes. The formula for the partial sum of a geometric series implies that

$$|F_p| \le \frac{1}{xy} \frac{(2z_1)^{p+1}}{1-2z_1} \frac{(2z_2)^{p+1}}{1-2z_2}.$$

If x and y lie in the interval $1/q, q$ with $1 \le q < \sqrt{2}$, then F_p is bounded by

$$|F_p| \le 2q^2 \frac{(q^2-1)^{2p+2}}{(2-q^2)^2}.$$

Note that the geometric convergence factor $q^2 - 1 = q(q - 1/q)$ is larger than the convergence factor $q - 1/q$ in the error bound for E_p, but that it is raised to a higher power $2p+2$ rather than $p+1$. It turns out that our bound for F_p is comparable to our bound for E_p roughly when $q = 1.23$ and negligible roughly when $q = 1.13$. Our numerical results use the value $q = 1.125$, and F_p is indeed negligible.

This completes the formulas and estimates we will need to derive the fast Laplace transform.

3 The fast Laplace transform

We now explain our algorithm for evaluating the discrete Laplace transform

$$\hat{f}_i = \sum_{j=1}^{N} f_j e^{-t_i s_j}, \tag{10}$$

within an error bounded by $\epsilon F = \epsilon \sum |f_j|$. We use the series (7) in a natural way. Since (7) converges rapidly for x and y near 1, we scale t_i and s_j into groups lying near 1. The scaling factors then comprise z which is allowed to lie anywhere. We begin by dividing the interval $(0, \infty)$ in which the "sources" s lie into geometric intervals of ratio $q > 1$. Such an interval B_n is given by $B_n = (q^{2n-1}, q^{2n+1}]$, for any integer n, and the geometric center of B_n is $s_B = q^{2n}$. Thus s lies in B_n whenever $1/q \le s/s_B \le q$. The ratio q will be chosen later, to balance work and error. Similarly, we cut the interval $(0, \infty)$ in which the "targets" t_i lie into geometric intervals C_m with geometric centers t_C and ratio q. (We take the same ratio for simplicity of exposition; clearly one can take different ratios for targets and sources.) Now each source s_j and target t_i lies in a geometric interval, say B or C respectively. Consider targets t_i lying in C. We want to evaluate

$$\begin{aligned} \hat{f}_i &= \sum_{j=1}^{N} f_j e^{-t_i s_j} \\ &= \sum_{B} \sum_{s_j \in B} f_j e^{-t_i s_j} \\ &= \sum_{B} \sum_{s_j \in B} f_j e^{-(t_i/t_C)(s_j/s_B) t_C s_B}. \end{aligned}$$

Here we have broken up the sum over j by summing over sources s_j in each box B separately, and scaled t_i and s_j to lie near 1.

Consider all sources and targets lying in a fixed pair of boxes B and C,

and apply (7):

$$\begin{aligned}\hat{f}_i(B) &= \sum_{s_j \in B} f_j e^{-(t_i/t_C)(s_j/s_B)t_C s_B} \\ &= \sum_{j=0}^{p}\sum_{k=0}^{p}\sum_{s_j\in B} f_j \frac{s_B}{s_j}\left(\frac{1+s_B/s_j}{2}\right)^j \left(1-\frac{s_B}{s_j}\right)^k \frac{t_C}{t_i} \\ &\cdot \left(\frac{1+t_C/t_i}{2}\right)^k \left(1-\frac{t_C}{t_i}\right)^j \frac{(j+k)!}{j!k!}\mathcal{L}_{j+k}(t_C s_B) + \mathcal{E}_p.\end{aligned}$$

Here we choose p and q, depending only on ϵ, so that $\mathcal{E}_p \le \epsilon F$ where $F = \sum |f_j|$. We have now separated the variables t_i and s_j in such a way that a fast algorithm is possible. It proceeds as follows.

For each nonempty source interval B, we evaluate $(p+1)^2$ coefficients

$$A_{jk}(B) = \sum_{s_j\in B} f_j \frac{s_B}{s_j}\left(\frac{1+s_B/s_j}{2}\right)^j \left(1-\frac{s_B}{s_j}\right)^k$$

for $0 \le j,k \le p$. This costs $O(p^2)$ work for each nonempty B, which cannot add up to more than $O(p^2 N)$. Note that p depends only on the user-specified tolerance ϵ. Now the algorithm proceeds by running over nonempty target intervals C. For each C, we accumulate a series of the form

$$\sum_{j=0}^{p}\sum_{k=0}^{p} B_{jk}(C)\frac{t_C}{t}\left(\frac{1+t_C/t}{2}\right)^k \left(1-\frac{t_C}{t}\right)^j \tag{11}$$

to be evaluated at all targets $t = t_i$ lying in C. Here the coefficients B_{jk} are given by

$$B_{jk}(C) = \sum_B A_{jk}(B)\frac{(j+k)!}{j!k!}\mathcal{L}_{j+k}(t_C s_B).$$

Hence it costs at most $O(p^2|B||C|)$ to form all coefficients B_{jk} for all target boxes C, where $|B|$ and $|C|$ are the total numbers of nonempty source and target boxes respectively.

The final step in the algorithm is to evaluate the appropriate series (11) at each t_i: this costs $O(p^2 M)$ work. The complete algorithm thus costs $O(N +$

M) work to evaluate the discrete Laplace transform with an error less than ϵF; the constant in $O(N+M)$ depends only on ϵ. The overhead associated with forming the coefficients is bounded by the number of source box-target box interactions, which depends only on the maximum and minimum source and target locations.

An even further cost reduction is effected by cutting off the interaction: if $t \in C$ and $s \in B$, then

$$e^{-ts} \leq e^{-t_C s_B/q^2}.$$

If $t_C = q^{2n}$ and $s_B = q^{2m}$, then

$$e^{-ts} \leq e^{-q^{2(n+m-1)}}$$

which decays rapidly as $n+m$ increases. Thus target box C_n need only interact with source boxes B_m for which $n+m \leq 1 + (\log\log 1/\epsilon)/2\log q$.

4 Numerical results

First, we tested the algorithm on randomly generated points uniformly distributed on the interval $[0,5]$, with weights f_i random and uniformly distributed on $[0,1]$. We took $\epsilon = 10^{-6}$, $q = 1.125$ and $p^2 = 6^2$ terms in the Laguerre series. The results are reported in Table 1; T_f is the time required for the fast evaluation scheme, while T_d is the direct evaluation time. The fast algorithm beats direct evaluation consistently for $N = M \geq 20$. When $N = M = 10240$, the fast algorithm is about a thousand times faster than direct evaluation; thus the projected break-even point is at $N = M = 10$. The column headed E_f reports the errors produced by the fast algorithm, as measured against the direct calculation.

$N = M$	T_f	T_d	E_f
20	.01	.01	4.1-9
40	.01	.02	4.4-8
80	.03	.06	8.5-8
160	.04	.24	1.1-7
320	.06	.96	6.7-8
640	.10	3.78	6.5-8
1280	.16	15.36	6.6-8
2560	.29	61.70	7.7-8
5120	.52	248.32	7.4-8
10240	.99	1022.98	7.0-8

Table 1: Evaluation of a discrete Laplace transform to accuracy 10^{-6}, with points randomly generated on $[0, 5]$.

We also tested the algorithm on equispaced points on the interval $[0, 10]$, to simulate the application of the algorithm to numerical integration with the trapezoidal rule. The weights were randomly distributed on $[0, 1]$, as before. The results are shown in Table 2. Again, the fast algorithm breaks even at $N = M = 20$ and achieves a speedup of 1000 at $N = M = 10240$. The errors are also considerably smaller than the error bound.

$N = M$	T_f	T_d	E_f
20	.01	.01	7.8-8
40	.02	.02	4.9-8
80	.03	.06	5.3-8
160	.04	.26	6.9-8
320	.07	.99	8.2-8
640	.11	3.84	6.4-8
1280	.18	15.49	2.1-8
2560	.30	61.95	8.5-8
5120	.55	247.30	3.6-7
10240	1.04	993.28	3.6-7

Table 2: Evaluation of a discrete Laplace transform to accuracy 10^{-6}, with points equispaced on $[0, 10]$.

Finally, we tested the logarithmic dependence of the work estimate on the user-specified precision ϵ. According to our estimates, the work required should grow only logarithmically with ϵ. Thus we doubled the number of significant digits required, set $\epsilon = 10^{-12}$, and used $p^2 = 13^2$ terms in each Laguerre series. Here the points were again equispaced on $[0, 10]$. The results are shown in Table 3. Clearly the time has very precisely doubled in

comparison with Table 2.

$N = M$	T_f	T_d	E_f
20	.01	.01	4.2-14
40	.04	.02	3.2-14
80	.07	.07	5.9-14
160	.09	.27	3.6-14
320	.14	1.06	4.3-14
640	.23	3.97	2.9-15
1280	.35	16.00	1.1-14
2560	.60	64.00	3.0-14
5120	1.07	247.81	1.4-13
10240	2.00	1037.31	1.3-13

Table 3: Evaluation of a discrete Laplace transform to accuracy 10^{-12}, with points equispaced on $[0, 10]$.

5 Generalizations

Our algorithm generalizes immediately to evaluate the convolution with a fixed Laguerre function,

$$\hat{f}_i = \sum_{j=1}^{N} f_j \mathcal{L}_n(t_i s_j).$$

(The Laplace transform is the case $n = 0$.) One simply uses the expansion derived in §2,

$$\mathcal{L}_n(ts) = \sum_{k=0}^{\infty} \left(\frac{1}{t}\right)^{n+1} \left(1 - \frac{1}{t}\right)^k \frac{(n+k)!}{n!k!} \mathcal{L}_{n+k}(s),$$

in place of (5) in the calculations following (5). The resulting formula,

$$\begin{aligned}\mathcal{L}_n(xyz) &= \left(\frac{1}{xy}\right)^{n+1} \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} 2^{-(j+k)}(1+1/x)^k(1-1/y)^k \cdot \\ &\cdot \quad (1+1/y)^j(1-1/x)^j \frac{(j+k+n)!}{j!k!n!}\mathcal{L}_{j+k+n}(z),\end{aligned}$$

implies a fast algorithm exactly similar to the fast Laplace transform.

This observation makes an even further generalization possible. One often wants to evaluate multiplicative convolution sums of the form

$$\hat{f}_i = \sum_{j=1}^{N} f_j K(t_i s_j), \tag{12}$$

where K is a smooth function on R^+ which decays rapidly enough at ∞. Such a kernel has a rapidly converging Laguerre expansion

$$K(z) = \sum_{n=0}^{\infty} K_n \mathcal{L}_n(z)$$

which approximates K arbitrarily well if enough terms are taken. Thus one can truncate the series after say P terms, and apply a fast Laguerre transform to each term to get an algorithm which evaluates (12) at M points in $O(N+M)$ work, with a constant depending only on the precision desired.

6 Conclusions

We have presented a fast algorithm which achieves a thousandfold speedup over the direct calculation of the discrete Laplace transform when ten thousand points are used, but requires so little overhead that it is faster even when 20 points are used. The constant in the work estimate depends only on the precision desired, and only logarithmically at that. Thus asking for twice

as many correct digits only costs twice as much. Such an algorithm makes it possible to evaluate Laplace transforms numerically to far higher accuracy in far less time.

A generalization of the algorithm allows the application of any multiplicative convolution operator on R^+ with a smooth kernel, in optimal time.

References

[1] E. D. Rainville, *Special Functions*, New York: Chelsea, 1960.

[2] V. Rokhlin, *A Fast Algorithm for the Discrete Laplace Transformation*, Journal of Complexity 4(1988), 12-32.

[3] G. Szego, *Orthogonal Polynomials*, New York: American Mathematical Society Colloquium Publications, vol. 23, 1939.