

Software for an Implementation of Weeks' Method for the Inverse Laplace Transform Problem

B. S. GARBOW, G. GIUNTA and J. N. LYNESS

Argonne National Laboratory

and

A. MURLI

Università di Napoli

A software package based on a modification of the Weeks' method is presented for calculating function values $f(t)$ of the inverse Laplace transform. This method requires transform values $F(z)$ at arbitrary points in the complex plane, and is suitable when $f(t)$ has continuous derivatives of all orders; it is especially attractive when $f(t)$ is required at a number of different abscissas t .

Categories and Subject Descriptors: G.1.2 [Numerical Analysis]: Approximation—*nonlinear approximation*; G.1.4 [Numerical Analysis]: Quadrature and Numerical Differentiation—*finite difference methods*; G.1.9 [Numerical Analysis]: Integral Equations—*Fredholm equations, Laplace transform*

General Terms: Algorithms

Additional Key Words and Phrases: Cauchy integral, inverse Laplace transform, Laguerre polynomials, numerical differentiation, numerical software, series evaluation, Taylor series, WEEKS

1. INTRODUCTION

In this paper we present a software package, called WEEKS, for the numerical inversion of the Laplace transform. The package obtains approximations for $f(t)$ when numerical values of the Laplace transform function

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt, \quad \text{Re } s > \sigma_0, \quad (1.1)$$

are available for all complex s in the form of a user-supplied subprogram. The user also supplies the numerical value or an upper bound for σ_0 , the Laplace transform convergence abscissa, and an accuracy parameter ϵ_{tol} .

This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

Authors' present addresses: B. S. Garbow and J. N. Lyness, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439; and G. Giunta and A. Murlì, Dipartimento di Matematica e Applicazioni, Università di Napoli, Via Mezzocannone 8, 80134 Napoli, Italy.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1988 ACM 0098-3500/88/0600-0163 \$01.50

The method implemented here is based on Weeks' Method [8], as modified by Lyness and Giunta [5]. This method is suitable when $f(t)$ has continuous derivatives of all orders, and is especially attractive when $f(t)$ is to be evaluated for a number of different values of t .

In the following section, we describe the underlying mathematics of the algorithm. The quality of the results returned by our method depends on the choice of certain incidental parameters σ and b . In Section 3 we give some criteria that could be helpful in making this choice; however, a proper choice requires information that is rarely available to the user. In Section 4 we describe some of the diagnostic output from the program designated *modul1*. Our hope is that in a particular problem such output, considered in conjunction with the criteria presented in Section 3, might be helpful in improving the choice of these incidental parameters. In section 5 we present some of the results of numerical comparisons between our software and three other available software items.

2. OUTLINE OF THE ALGORITHM

Here we present a brief outline of the mathematical theory on which the software is based.

Let $\sigma > \sigma_0$ and $b > 0$ be parameters, and let $L_s(x)$ be the Laguerre polynomial of degree s ($e^{-x/2}L_s(x)$ is the corresponding Laguerre function). Let

$$\phi(z) = \frac{b}{1-z} F\left(\frac{b}{1-z} + \sigma - \frac{b}{2}\right), \quad (2.1)$$

where $F(s)$ is the Laplace transform in (1.1). Let a_s be the Taylor coefficients of $\phi(z)$:

$$\phi(z) = \sum_{s=0}^{\infty} a_s z^s, \quad |z| < R_{\text{conv}}. \quad (2.2)$$

Then the following expansion for $f(t)$ exists:

$$f(t) = e^{\sigma t} \sum_{s=0}^{\infty} a_s e^{-bt/2} L_s(bt); \quad \sigma > \sigma_0, \quad b > 0. \quad (2.3)$$

Moreover, the radius of convergence R_{conv} in (2.2) satisfies

$$R_{\text{conv}} \geq 1; \quad \sigma > \sigma_0, \quad b > 0. \quad (2.4)$$

This algorithm is only suitable when $R_{\text{conv}} > 1$, a condition that pertains when $f(t)$ is $C^\infty[0, \infty)$. It provides an approximation $\tilde{f}(t)$ to $f(t)$ by truncating expansion (2.3) and approximating individual coefficients. Thus,

$$\tilde{f}(t) = e^{\sigma t} \sum_{s=0}^{m-1} \tilde{a}_s e^{-bt/2} L_s(bt). \quad (2.5)$$

The approximations \tilde{a}_s to a_s are based on the Cauchy integral representation

$$\begin{aligned} a_s &= \frac{\phi^{(s)}(0)}{s!} = \frac{1}{2\pi i} \int_{|z|=r} \phi(z) z^{-1-s} dz \\ &= \frac{1}{2\pi r^s} \int_0^{2\pi} \phi(re^{i\theta}) e^{-is\theta} d\theta, \quad r < R_{\text{conv}}. \end{aligned} \quad (2.6)$$

Since $\phi(re^{i\theta})$ is periodic in θ , the m -panel trapezoidal rule is suitable. Consequently, we employ the following approximations:

$$\tilde{a}_s = a_s^{[m,1]} = \frac{1}{mr^s} \sum_{j=1}^m \phi(re^{2\pi i j/m}) e^{-2\pi i j s/m}, \quad s = 0, 1, \dots, m-1. \quad (2.7)$$

Since $\phi(\bar{z}) = \overline{\phi(z)}$, only $m/2 + 1$ distinct evaluations of $\phi(z)$ are required to evaluate (2.7).

Note that, in addition to an algorithm to evaluate $\phi(z)$, the evaluation of $\tilde{f}(t)$, defined by (2.5) and (2.7), requires only the values of t and the values of parameters σ , b , m and r .

In our implementation the user does not provide the values of m and r directly, but instead provides m_{top} , an upper limit on possible values of m , and ϵ_{tol} , a required *pseudouniform accuracy*. The software is designed to return a result $\tilde{f}(t)$, which satisfies

$$\left| \frac{\tilde{f}(t) - f(t)}{e^{\sigma t}} \right| < \epsilon_{\text{tol}}. \quad (2.8)$$

The theory, described in detail by Lyness and Giunta [5], shows that this is a natural error criterion for this method.

A convenient existing software item for calculating $a_s^{[m,1]}$ in (2.7) is *entcre* (see [6]); it requires input parameters r and ϵ_{req} and a function subprogram to evaluate $\phi(z)$. *entcre* itself determines a suitable value of m (which is a power of 2) and returns the quantities specified in (2.7). In general, the value of m returned is the smallest power of 2 for which

$$\epsilon_{\text{est}}^{[m,1]} = |a_0^{[m,1]} - a_0| < \epsilon_{\text{req}}. \quad (2.9)$$

entcre is able to calculate $\epsilon_{\text{est}}^{[m,1]}$ since it makes an additional function call with $z = 0$, and $a_0 = \phi(0)$. *entcre* is called with

$$r = \max(e^{-(1/1024)}, e^{-1/m_{\text{top}}}), \quad \epsilon_{\text{req}} = \epsilon_{\text{tol}}/e. \quad (2.10)$$

Thus, generally the specification of m occurs automatically in *entcre*, this being the smallest power of 2 satisfying (2.9). However, the limit $m \leq m_{\text{top}}$ is applied.

The software package comprises two user-visible subprograms, *modul1* and *modul2*. Also required for running and included in the package is a modified copy of *entcre* (with *hfcf*) and function subprogram *phifun* for implementing (2.1). The user must provide a function subprogram for $F(s)$.

In the calling program, *modul1* is called only once. *modul1* calls *entcre*, which invokes *phifun*, which invokes the subprogram for $F(s)$. The user is obliged to provide *modul1* with numerical values for σ_0 and ϵ_{tol} . The user may provide values for σ and b ; otherwise, *modul1* will set default options. This module returns the values of m , $a_s^{[m,1]}$, $s = 0, 1, \dots, m-1$.

After *modul1* has returned, *modul2* is called once for each value of t for which $\tilde{f}(t)$ is required. This module simply evaluates expression (2.5) for $\tilde{f}(t)$; the user provides the value of t . The values of σ , b , m , and $\tilde{a}_s = a_s^{[m,1]}$ are those output from *modul1*. The complete calling sequences of both modules are given on pages 173–175 of this issue [2].

3. INCIDENTAL PARAMETERS AND SCALING

3.1 The Role of σ_0 : The Laplace Convergence Abscissa

Nearly all the techniques known to the authors for the inverse Laplace transform problem require that the user give the numerical value of σ_0 , the Laplace transform convergence abscissa, or an upper bound $\tilde{\sigma}_0$ on σ_0 . σ_0 is defined in (1.1) in terms of the existence of an integral. It may also be defined as the real part of the rightmost singularity of $F(s)$.

One of the reasons for having to know the value of σ_0 is that the parameter σ must exceed σ_0 in value. If it does not, the series for $\tilde{f}(t)$ will not converge. If the value of σ_0 is not known, the user must be prepared for significant preliminary effort, either in using the method and obtaining chaotic results or in attempting to locate the rightmost singularities of $F(s)$. Incidentally, note that other methods with which the authors are familiar also require the user to set a parameter corresponding to σ greater than σ_0 , with the risk of convergence to an incorrect result if this is not done.

The numerical value of σ_0 is also relevant in providing a somewhat crude measure of the magnitude of $f(t)$ for large t . It is readily proved that

$$\begin{aligned} \limsup_{t \rightarrow \infty} |f(t)e^{-\sigma_0 t}| e^{\epsilon t} &= 0, & \epsilon < 0 \\ &= \infty, & \epsilon > 0. \end{aligned} \quad (3.1)$$

Different users naturally have different accuracy requirements. When one seeks values of $f(t)$ over a range of values of t , to ask for uniform relative accuracy may be unrealistic, particularly when $f(t)$ is oscillatory. To request uniform accuracy with respect to the function $e^{\sigma_0 t}$, however, is reasonable, since, for large t , $f(t)$ is of uniform numerical order $ke^{\sigma_0 t}$. This leads us to define $\epsilon_{\text{nat}}(t)$, the *natural relative accuracy* of approximation $\tilde{f}(t)$ to $f(t)$, as

$$\epsilon_{\text{nat}}(t) = (\tilde{f}(t) - f(t))/e^{\sigma_0 t}. \quad (3.2)$$

3.2 Choice of Parameter σ

Unfortunately, even when σ_0 is known, the choice of the parameter σ is not easy. Briefly, the situation is that series (2.3) converges slowly when $\sigma - \sigma_0$ is small, and faster when $\sigma - \sigma_0$ is larger. However, the method produces a result in which the accuracy with respect to $e^{\sigma t}$, but not to $e^{\sigma_0 t}$, is uniform. Specifically, we assign a value ϵ_{tol} , and the routine obtains results $\tilde{f}(t)$ satisfying

$$|\tilde{f}(t) - f(t)| e^{\sigma t} \leq \epsilon_{\text{tol}}, \quad \text{for all } t \geq 0. \quad (3.3)$$

The natural accuracy measure $\epsilon_{\text{nat}}(t)$ then satisfies

$$|\epsilon_{\text{nat}}(t)| < \epsilon_{\text{tol}} e^{(\sigma - \sigma_0)t} \quad (3.4)$$

and degrades exponentially with t , the exponential constant being $\sigma - \sigma_0$. Thus, if one requires meaningful results for $0 \leq t \leq T$ with a large value of T , one needs to take $\sigma - \sigma_0$ small, in which case the series (2.3) converges slowly and requires many terms to converge. On the other hand, for a smaller value of T one can allow $\sigma - \sigma_0$ to be larger, resulting in faster convergence and a less expensive result.

This can be put on a sound footing. The natural accuracy $\varepsilon_{\text{nat}}(t)$ will degrade over the range $0 \leq t \leq T$. If a natural accuracy degrading from ε_0 to ε_T is required, the user should set

$$\begin{aligned}\sigma - \sigma_0 &= \frac{1}{T} \ln(\varepsilon_T/\varepsilon_0); \\ \varepsilon_{\text{tol}} &= \varepsilon_0.\end{aligned}\tag{3.5}$$

For example, if $\varepsilon_T/\varepsilon_0 = 20$, $\sigma - \sigma_0 \simeq 3/T$.

We refer generically to accuracy criteria of form (3.2) or (3.3) as *pseudorelative accuracies* with respect to the corresponding exponential function. The pseudo-relative accuracy with respect to $e^{\sigma_0 t}$ is what we call *natural relative accuracy*.

3.3 A Remark about Scaling

The type of tolerance required makes prior scaling of the input Laplace transform unnecessary. Mathematically, the following problems are equivalent:

- (a) given $F(s)$, find $f(t)$;
- (b) given $G(s) = F(s + k)$, find $g(t)$;
- (c) given $H(s) = F(\alpha s)$, find $h(t)$;

where $f(t)$, $g(t)$, and $h(t)$ are the inverse Laplace transforms of $F(s)$, $G(s)$, and $H(s)$, respectively. The answers are related by

$$g(t) = f(t)e^{-kt}\tag{3.6}$$

and

$$h(t) = \alpha^{-1}f(\alpha^{-1}t),\tag{3.7}$$

and the respective Laplace convergence abscissas by

$$\sigma_{0F} = \sigma_{0G} + k = \sigma_{0H}\alpha.$$

At first glance, it looks as if one could obtain better accuracy in $f(t)$ by making k large and solving problem (b) instead of (a), and then using (3.6) to obtain the result of problem (a); however, this is not true. A moment's reflection reveals that, if the same values of b and $\sigma - \sigma_0$ are used, the calculations are completely equivalent and the results satisfy

$$\frac{\tilde{g}(t) - g(t)}{e^{\sigma_G t}} = \frac{\tilde{f}(t) - f(t)}{e^{\sigma_F t}}.$$

Moreover, if $b_H = b\alpha^{-1}$ and $\sigma_H = \sigma\alpha^{-1}$ are used for the calculation of $\tilde{h}(t)$ from $H(s)$, this is also completely equivalent. Since the user's problem could have been equally well formulated in terms of $F(s)$, $G(s)$, or $H(s)$, it is dangerous to unthinkingly assume that the user requires uniform accuracy in $f(t)$ since this does not translate into uniform accuracy in $g(t)$ or $h(t)$. For a meaningful calculation, it is imperative that the user look a stage ahead to see what use is to be made of the results and to assess the accuracy needed for that purpose.

3.4 Choice of b

Assume σ has been chosen. We now give some suggestions for the subsequent choice for b , but only rarely will the user have enough information available to use other than (a) below:

- (a) Set $b/2 \geq \sigma - \sigma_0$.
- (b) In terms of the locations of the singularities s_j of $F(s)$ nearest to σ_0 , set

$$\frac{b}{2} \geq \min_j (|\sigma - s_j|).$$

- (c) Let C be the set of circles each of which includes all the singularities of $F(s)$. Let C_1 be that circle of the set C that subtends the smallest angle at σ . Then choose $b/2$ to be the length of the tangent from σ to C_1 .

It is relatively straightforward to justify (a) and (b) mathematically on the basis of the ultimate convergence rate of series (2.3). Suggestion (c) may be justified on the same basis for cases in which $F(s)$ is a rational function, or has no singularities other than poles and no pole at infinity.

3.5 Choice of m_{top}

The user-provided parameter m_{top} constitutes a physical limit on m and hence on the size of the calculation. Several work arrays are of dimension m_{top} (see [2]). The authors feel that a value of m_{top} such as 64 or 128 should be adequate for any problem for which this is a suitable method, as long as the parameters σ and b have been carefully chosen. Only in exceptional situations should a value of m_{top} exceeding 1024 be required.

3.6 Default Options

The user has to provide numerical values for σ_0 , σ , and b . Naturally, the software has no control whatsoever over the values provided by the user. Nevertheless, it provides default options that override σ and b as follows:

- (i) If $\sigma \leq \sigma_0$, σ is replaced by $\sigma_0 + 0.7$.
- (ii) If $b < 2(\sigma - \sigma_0)$, b is replaced by $2.5(\sigma - \sigma_0)$.

Then, the routine disregards σ_0 completely and bases its calculation on σ and b . Section 3.4 indicates that when σ_0 is correct, default option (ii) is quite reasonable, and our numerical experiments have confirmed this. However, there is no proper theoretical justification for (i). The number 0.7 was chosen because it is near to an optimum overall choice for the set of 16 functions $F_j(s)$ used by Davies and Martin [1] in their numerical experiments, in which $f_j(t)$ is evaluated for $0.5 \leq t \leq 15 = T$. However, reference to Sections 3.2 and 3.3 indicates that, if Davies and Martin had happened to choose $T = 1.5$ or $T = 150$, or used $\tilde{F}_j(s) = F_j(s/10)$ or $\tilde{F}_j(s) = F_j(10s)$ instead, we should have replaced 0.7 by 7 or by 0.07.

4. DIAGNOSTIC INFORMATION

The heart of our method is the attempt to evaluate series (2.3). If this series converges quickly, the approximation (2.5) may be good and inexpensive. If it converges too slowly, the method should be abandoned. The rate of convergence

depends principally on the values of a_s . Information about these values is available from *modul1* and processed into information about the expected accuracy of $\tilde{f}(t)$, before *modul2*, which calculates $\tilde{f}(t)$, is called.

As soon as the values of \tilde{a}_s are available, *modul1* determines values of K and $R > 1$ having the property

$$|\tilde{a}_s| < KR^{-s}, \quad s = m/2, m/2 + 1, \dots, m - 1. \quad (4.1)$$

An algorithm for this is given in [3], pp. 17–18. A set of approximate pseudoerror bounds is then constructed based on the assumption that $|a_s| < KR^{-s}$ for all s . Details are given in [5]. These bounds are ϵ_{disc} , the error resulting from using \tilde{a}_s in place of a_s in (2.3); ϵ_{trunc} , the error resulting from truncating (2.3); and ϵ_{cond} , an estimate of the condition error based on the optimistic assumption that the noise level of function values is near the machine accuracy parameter. We have found

$$\epsilon_{\text{index}} = \epsilon_{\text{disc}} + \epsilon_{\text{trunc}} + \epsilon_{\text{cond}} \quad (4.2)$$

to be a highly reliable bound on the pseudoerror with respect to $e^{\sigma t}$ in the results subsequently returned by *modul2*.

Numerical values of these approximate error bounds, and of K and R , are returned by *modul1* in a diagnostic vector.

5. RESULTS OF NUMERICAL COMPARISONS

We have carried out a brief set of numerical experiments in which the performance of our software WEEKS was compared with the following three pieces of software:

- (1) FLINV: from the IMSL Library, based on Crump's method;
- (2) LAPIN: Algorithm 27 from *Journal of Computational and Applied Mathematics* [4]; and
- (3) DLAINV: Algorithm 619 from *ACM Transactions on Mathematical Software* [7].

Note that the Mark X (1985) version of the NAG Library contains no software for this problem.

Readers interested in the results of a major testing effort (15 methods, 16 functions, and 30 values of t for various choices of incidental parameters) should examine the article by Davies and Martin [1]. Our results, outlined below, are in keeping with those published in that article; our experiments confirm their conclusions in general, and extend some in particular.

First, we have limited ourselves to those methods that require and use $F(s)$ with s complex, and that are not furnished with the numerical value of the exponent of the asymptotic decay rate as s becomes infinite. Davies and Martin include five methods (#8, #9, #12, #13, #14) of this type, and examination of their results indicates the strongest performers are #9 and #14, upon which WEEKS and FLINV, respectively, are based. (LAPIN and DLAINV have appeared only recently, and so their algorithms were not included among the Davies–Martin methods.)

We found that WEEKS produced comparable results to those reported by Davies and Martin (for method #9) at about half the cost, exactly as one might

expect. FLINV produced results closely in line with those (for method #14) of Davies and Martin.

A reasonable expectation is that the user will require the values of $f(t)$ for a range of values of t . Indeed, the calling sequences of both FLINV and LAPIN (but not DLAINV) allow for a vector of t -values to be entered. However, all these routines then proceed to perform sequences of independent calculations, based for each value of t on a new set $F(s)$ of transform function values. In contrast, WEEKS, after calculating a single set of transform function values, can evaluate $f(t)$ for as many values of t as one likes. When many values are required, this is a highly significant economic advantage; in the Davies–Martin tests, where the function is evaluated at 30 different values of t , this can give WEEKS an advantage by a factor of 30 when cost is measured by the number of transform values.

The Davies–Martin function list can be divided into two classes:

- (a) $f(t)$ is an entire function (#1, #3–#8, #13, #16); and
- (b) $f(t)$ has a discontinuity (#10, #12), $t^{1/2}$ singularity at the origin (#2, #9, #14), $\log t$ singularity at the origin (#11), or an isolated essential singularity at the origin (#15).

For functions of class (a), the result of our testing is so conclusive that the factor 30 does not affect the conclusions: WEEKS is faster than FLINV, LAPIN, or DLAINV by factors of about 5 for a single function value, or by factors of about 150 in a direct emulation of the Davies–Martin tests. For function #15, performances of the several routines are closely comparable; for the other functions of class (b), WEEKS is not suitable, exhausting its limit of function evaluations and returning results markedly poorer than those from any of the other three routines.

REFERENCES

1. DAVIES, B., AND MARTIN, B. Numerical inversion of the Laplace transform: A survey and comparison of methods. *J. Comput. Phys.* 33, 1 (Oct. 1979), 1–32.
2. GARBOW, B. S., GIUNTA, G., LYNESS, J. N., AND MURLI, A. Algorithm 662: A FORTRAN software package for the numerical inversion of the Laplace transform based on Weeks' method. *ACM Trans. Math. Softw.* 14, 2 (June 1988), 171–176.
3. GIUNTA, G., LYNESS, J. N., AND MURLI, A. An implementation of Weeks' method for the inverse Laplace transform problem. ANL/MCS-TM-39, Argonne National Laboratory, Argonne, Ill., Oct. 1984.
4. HONIG, G., AND HIRDES, U. Algorithm 27: A method for the numerical inversion of Laplace transforms. *J. Comput. Appl. Math.* 10, 1 (Jan. 1984), 113–132.
5. LYNESS, J. N., AND GIUNTA, G. A modification of the Weeks method for numerical inversion of the Laplace transform. *Math. Comput.* 47, 175 (July 1986), 313–322.
6. LYNESS, J. N., AND SANDE, G. Algorithm 413: ENTCAF and ENTCRE. *Commun. ACM* 14, 10 (Oct. 1971), 669–675.
7. PIESSENS, R., AND HUYSMANS, R. Algorithm 619: Automatic numerical inversion of the Laplace transform. *ACM Trans. Math. Softw.* 10, 3 (Sept. 1984), 348–353.
8. WEEKS, W. T. Numerical inversion of Laplace transforms using Laguerre functions. *J. ACM* 13, 3 (July 1966), 419–429.

Received August 1986; revised December 1987; accepted February 1988

ACM Transactions on Mathematical Software, Vol. 14, No. 2, June 1988.