A Modification of Weeks' Method for Numerical Inversion of the Laplace Transform in the Real Case Based on Automatic Differentiation

Salvatore Cuomo[1], Luisa D'Amore[1], Mariarosaria Rizzardi[2], and Almerico Murli[1]

[1] University Federico II, Via Cintia, Naples, Italy, salvatore.cuomo@unina.it, [luisa.damore,almerico.murli]@dma.unina.it
[2] University Parthenope, Naples, Italy, mariarosaria.rizzardi@uniparthenope.it

Summary. Numerical inversion of the Laplace transform on the real axis is an inverse and ill-posed problem. We describe a powerful modification of Weeks' Method, based on automatic differentiation, to be used in the real inversion. We show that the automatic differentiation technique assures accurate and efficient numerical computation of the inverse Laplace function.

Keywords: Automatic differentiation, Laguerre expansion, numerical Laplace inversion

1 Introduction

Automatic differentiation (AD) is having a deep impact in many areas of science and engineering. AD plays an important role in a variety of scientific applications including meteorology, solution of nonlinear systems and inverse problems. Here we are dealing with the Laplace transform inversion (Lti) in the real case. Given a Laplace transform function $F(z)$:

$$F(z) = \int_0^\infty e^{-zt} f(t)dt, \quad z = Re(z) > \sigma_0, \tag{1}$$

where σ_0 is the abscissa of convergence of Laplace transform, we focus on the design of algorithms which obtain $f(t)$, at a given selection of values of t under the hypothesis that $F(z)$ is only computable on the real axis.

We consider Weeks' Method, introduced in [10] and developed as numerical software in [3] for complex inversion, i.e. when F is known on the complex plane. The inverse function $f(t)$ is obtained as a Laguerre expansion:

$$f(t) = e^{\sigma t} \sum_{k=0}^{\infty} c_k e^{-bt} L_k(2bt), \quad c_k = \frac{\Phi^{(k)}(0)}{k!} \tag{2}$$

where $L_k(2bt)$ is the Laguerre polynomial of degree k, $\sigma > \sigma_0$ and b are parameters. The c_k values are McLaurin's coefficients of the function Φ obtained from F. The success or failure of such an algorithm depends on the accuracy of the approximated coefficients c_k (*discretization error*) and on the number of terms in (2) (*truncation error*). In [3, 10] the c_k are computed by considering the Cauchy integral representation of the derivative:

$$c_k = \int_C \frac{1}{z^{k+1}} \Phi(z) dz$$

where C is any contour in the complex plane which includes the origin and does not include any singularities of Φ. For instance, it is a circular contour centered at the radius origin r.

This representation is not feasible in both cases if the Laplace Transform is only known on real axis, and in many applicative domains, such as the Nuclear Magnetic Resonance (NMR), where experimentally preassigned real values are determined. In [5], the authors suggested the computation of c_k using the finite difference schemes for approximating the derivatives. Unfortunately, as the authors state, the instability of high order finite difference schemes puts strong limitations on the maximal attainable accuracy of the computed solution. Moreover, in [2], the authors proposed a collocation method (C-method) for computing the c_k based on the solution of a Vandermonde linear system by using the Bjorck Pereira algorithm.

In all cases, the numerical performance of (2) depends on the choice of suitable values of σ and b. In particular, regarding the parameter σ,

1. if $\sigma - \sigma_0$ is "too small" (i.e. near to zero) a lot of terms of the Laguerre expansion is needed (slow convergence of (2)). In this case, the truncation error predominates.
2. $\sigma - \sigma_0$ is "too large" (i.e. much greater than 1) we can not compute an accurate numerical solution because of the exponential factor $e^{\sigma t}$ in the series expansion (2) that amplifies the errors occurring on the c_k coefficients computation. In this case, the roundoff errors predominate.
3. The choice of σ is also related to the t value. Indeed, because of the exponential growth factor $e^{\sigma t}$ in (2), the accuracy of $f(t)$ degrades as t grows. To address this problem, numerical methods for Lti measure the accuracy in terms of the so-called *pseudoaccuracy*, that provides a uniform accuracy scaled considering $e^{\sigma t}$:
$$\varepsilon_{pseudo}(t) = \frac{|f(t) - \tilde{f}_N(t)|}{e^{\sigma t}}.$$
4. Regarding the parameter b, in [4] the connection between σ and b is investigated and their choices is also discussed. In particular, if z_j is a singularity of $F(z)$ nearest to σ_0 it holds that:
$$\frac{b}{2} \geq \min_{\sigma > \sigma_0} |\sigma - z_j| \tag{3}$$
In [11] two Matlab codes are derived to find σ and b. In [3], a choice of σ and b is given based on experimental considerations.

In this paper we propose a direct computation of c_k by using forward AD and we show that AD assures significant advantages in terms of accuracy and efficiency. Computation of each value c_N is accurate up to the maximal relative accuracy and its precision scales as N^2. As a consequence, the discretization error becomes negligible and we can relate the choice of the parameter σ (therefore of b) essentially to the t value without degrading the final accuracy. We feel that this approach seems to be a good candidate to lead to an effective numerical software for Laplace inversion in the real case.

The paper is organized as follows: in Sect. 2 we give some preliminaries; in Sect. 3 we discuss remarks on AD and finally numerical experiments are provided in Sect. 4, where comparisons are reported.

2 Preliminaries

Here we give same basic definitions.

Definition 1. *Let $\gamma > 0$ be an integer number. The space S_γ is the set of all functions whose analytical continuation, $H(z)$, can be assumed in the form:*

$$H(z) = z^{-\gamma} G(z), \tag{4}$$

where G is analytic at infinity.

In the following, we assume $F(z) \in S_1$. Let $\sigma > \sigma_0$ and $b > 0$ be fixed. We define the operator Ψ onto S_1 such as:

$$\Psi : h \in S_1 \to \Psi[h(z)] = \frac{2 \cdot b}{1-z} h\left(\frac{2 \cdot b}{1-z} + \sigma - b\right) \in S_1,$$

and, by applying Ψ to F it follows:

$$\Psi[F(z)] = \frac{2 \cdot b}{1-z} F\left(\frac{2 \cdot b}{1-z} + \sigma - b\right) = \Phi(z). \tag{5}$$

To characterize the errors introduced by the computational approach we recall some basics definitions.

Definition 2. *The truncation error is defined as follows:*

$$\varepsilon_{trunc}(t, \sigma, b, N) = e^{\sigma t} \sum_{k=N}^{\infty} e^{-bt} c_k L_k(2bt)$$

The truncation error occurs substituting the infinite series in (2) by the finite sum consisting of the first N terms.

Definition 3. *Let $\Im$ be a finite arithmetic system with u as maximum relative accuracy. The roundoff error on $f_N(t)$ is defined as:*

$$\varepsilon_{cond}(t, \sigma, b, N) = f_N(t) - \tilde{f}_N(t)$$

where $\tilde{f}_N(t) = e^{\sigma t} \sum_{k=0}^{N} e^{-bt} \tilde{c}_k L_k(2bt)$ where $\tilde{c}_k$ are the approximated coefficients.

Table 1. AD-based scheme for Lti.

:: step 1: computation of the coefficients c_k by using AD
:: step 2: evaluation of the function $f_N(t) = e^{\sigma t} \sum_{k=0}^{N} c_k e^{-bt} L_k(2bt)$.

Definition 4. *Let $\Im$ be a finite arithmetic system with u as maximum relative accuracy. The discretization error on $f_N(t)$ is defined as:*

$$\varepsilon_{disc}(\sigma, b, N) = |c_N - \tilde{c}_N|$$

where $\tilde{c}_k$ are the coefficients obtained in $\Im$.

Observe that, for the choices (1)–(4) in Sect. 1, the parameter σ influences both ε_{trunc} and ε_{cond}. The best suitable value of σ should balance these two quantities. We propose the computation of the McLaurin coefficients using AD. Our approach is sketched in Table 1. By following [2, 10], we refer to the *pseudoaccuracy* defined as follows:

$$\varepsilon_{pseudo}(t) = \frac{|f(t) - \tilde{f}_N(t)|}{e^{\sigma t}}$$

and, by using the same arguments as in [10], it is

$$\varepsilon_{pseudo}(t) = \frac{|f(t) - \tilde{f}_N(t)|}{e^{\sigma t}} \leq ||T|| + ||R||$$

where $||T|| = \sqrt{\sum_{k=N}^{\infty} |c_k|^2}$, $||R|| = \eta(u)\sqrt{\sum_{k=0}^{N} |c_k|^2}$, and $\eta(u)$ depends on the maximum relative accuracy u. Therefore, it follows that:

$$AbsErr(t) = |f(t) - \widetilde{f}_N(t)| = e^{\sigma t}\varepsilon_{pseudo}(t) \leq e^{\sigma t}(||T|| + ||R||) = AbsErrEst(t).$$

In [2] the upper bound of $||T|| \leq \frac{K(r)}{r^N(r-1)}$ was given, $K(r)$ depending on Φ and on the radius of convergence of the MacLaurin series expansion. In Sect. 4 we provide a computable estimate of $||T||$.

3 Remarks on Automatic Differentiation

The computation of Taylor series using AD is a well known technique and many different software tools are available [6]. In Sect. 4 we use the software `TADIFF` [1].

To state both, the reliability and the efficiency of AD we analyze the performance of the approach in terms of the discretization error ε_{disc} and its computational cost. To achieve this aim, for computing the c_k using the derivatives of Φ, we first follow a straightforward approach based on a variant of Horner's method. The error analysis shows that this approach returns satisfying accuracy on the computed coefficients. Then, we experimentally verify that the error estimate still holds using `TADIFF`.

Table 2. Horner's method for evaluating the polynomial $T(z)$ of degree n at $z = z_0$. The a_i $i = 0,...n$ are the coefficients of T. The derivate S is computed simultaneously.

1.	$T = a_n$	$S = 0$
2.	$S = T + z_0 * S$	$n-1 \geq i \geq 0$
3.	$T = a_i + z_0 * T$	$n-1 \geq i \geq 0$
4.	The value of $T(z_0)$ is T	
5.	The value of $T'(z_0)$ is S	

We assume that $\Phi(z)$ is a rational polynomial of the following type[3]:

$$\Phi(z) = \frac{P(z)}{Q(z)} \tag{6}$$

where the numerator is a p-degree and the denominator is a q-degree polynomial and assume $p < q$. In Table 2 we show a variant of Horner's method used in the *evaluation trace*, which is the hand-coded implementation developed to evaluate Φ and its derivatives.

Theorem 1. *For each function $F(z) \in S_1$ we can determine the algorithm for the evaluation trace of $\Phi(0)$ based on the algorithms as described in Table 2, see [6].*

Remark 1. $c_0 = \Phi(0)$. If c_0 is computed in a finite arithmetic system $\Im$ where u is the relative maximum accuracy then, by applying the Forward Error Analysis (FEA) to the algorithm described in Table 2, we have that:

$$\widetilde{c_0} = c_0 + \mu \tag{7}$$

where $\mu = (2q+1+p)^2\delta_0$ and $\delta_0| \leq u$. From (7), it follows that c_0 can be computed within the relative maximum accuracy.

Remark 2. The c_N are the MacLaurin coefficients of $\Phi(z)$. In our case the $N-th$ derivative of $\Phi(z)$ is a rational polynomial too. In order to derive the error estimate on c_N we use the FEA too, and we have:

$$\widetilde{c_1} = c_1 + 2\mu$$

$$\widetilde{c_2} = c_2 + 3\mu$$

$$\widetilde{c_3} = c_3 + 6\mu$$

$$\widetilde{c_N} = c_N + 1\mu + 2\mu + ... + N\mu = O(N^2)\mu \tag{8}$$

From (8) it follows:

$$|\widetilde{c_N} - c_N| \leq N^2(2q+1+p)^2u \tag{9}$$

The round-off error on c_N is bounded below by a quantity which is proportional to the maximum relative accuracy, and it scales as N^2. This result is quite useful both in terms of discretization error estimate and of its computational cost.

[3] This assumption is not restrictive because any function $F(z) \in S_1$ behaves like a rational polynomial as $|z| \to \infty$.

Example 1. Let $F(z) = \frac{z-1}{(z^2+1)^2}$, $\sigma = 0.7$, $b = 1.7$, $N = 40$, $u = 2.22 \times 10^{-16}$. By using `TADIFF`, we get:

$$\widetilde{c_N} = -2.11246 \times 10^{-9}$$

Let

$$c_N = -2.00604215234895 \times 10^{9}$$

be the value obtained using symbolic derivation of Φ. Then:

$$|\widetilde{c_N} - c_N| = 5.65 \times 10^{-11}$$

and

$$N^2(2q+1+p)^2 u = 40^2(2+1+4)^2 \cdot u = 1.74 \times 10^{-11}$$

This means that the upper bound given in (9) also provides a reliable estimate of the error introduced on c_N obtained using `TADIFF`.

Remark 3. Observe that the number of terms used in the series expansion (2) is small (say $N \leq 70/75$). Indeed, as $N \to \infty$, $c_N \to 0$, and as soon as the computed value of c_{N+1} becomes numerically zero (i.e. less than the maximum relative accuracy times c_N), the Laguerre series expansion should be truncated at that value of N. For instance, regarding the function F introduced in example 1, we get:

$$c_{70} = 4.48 \times 10^{-17}$$

and $N = 70$ should be considered a reliable value of N. This result implies that the factor N^2 in (9) scales the maximum relative accuracy of two orders of magnitude at most and that the computation of the coefficients c_N is not time consuming.

We conclude that, by using `TADIFF` the discretization error becomes negligible, therefore, we mainly refer to the truncation error and to the condition error.

4 Numerical Experiments

In this section we describe numerical simulations carried out using the software TADIFF for computing the coefficients c_k. Moreover, we use the following upper bound of the absolute error $AbsErr(t)$:

$$AbsErrEst(t) \leq e^{\sigma t} \left\{ \sqrt{\sum_{k=N}^{M} |c_k|^2} + u \sqrt{\sum_{k=0}^{N} |c_k|^2} \right\} = CompAbsErr(t)$$

where $M = 2N$ and $u = 2.22 \times 10^{-16}$. Experiments are carried out using the double precision on a Pentium IV 2.8 GHz, with Linux Kernel 2.622-14-386 and gcc 4.1.3 compiler.

4.1 Simulation 1

We consider the following functions:

$$F_1(z) = \frac{z}{(z^2+1)^2} \quad f_1(t) = cos(2t), \quad \sigma_0 = 0.$$

In Table 3 we compare *AbsErr* and *CompAbsErr* at different values of t and using $\sigma = 0.7$ (then $b = 1.7$), on the left, and $\sigma = 2.5$ (then $b = 5$) on the right. The value of N has been fixed at 20. Note that for small σ the accuracy on the computed $f_N(t)$ ranges between four correct digits, at $t = 1$, and two correct digits, at $t = 9$. Because σ is relatively small, the low accuracy is essentially due to the slow convergence of the series expansion. In other words, the truncation error predominates. Conversely, when choosing $\sigma = 2.5$, the accuracy is higher at $t = 1, 1.5, 2...$ than before, while it strongly degrades at $t = 7, 9$. Although the series expansion converges more rapidly than at $\sigma = 0.7$, in this case, due to the higher value of σ, the exponential factor strongly degrades the final accuracy and the condition error predominates, especially as t grows. This experiment suggests that, once N is given (therefore, the truncation error is fixed), the value of σ should change depending on t: it should be large for small t and small for large t, in order to control the error amplification by keeping the exponential factor $e^{\sigma t}$ constant. We are working on the dynamic selection of σ at run time. In Figs. 1, 2 and in Table 4 we compare the AD-based computation with the C-method described in [2], in terms of the maximum absolute error on $[0, 7]$. As before, we consider $\sigma = 3$ and $\sigma = 0.7$, while $N = 28$ and $N = 50$. The numerical results confirm the better accuracy obtained using AD than using the C-method, where the coefficients are obtained by solving a Vandermonde linear system. The different accuracy is mainly due to the amplification of the discretization error introduced by these two methods.

4.2 Simulation 2

We compare the proposed approach with the following numerical codes:

- `InvertLT.m`: implementation (developed in C++ and `Matlab`) of the method proposed in [7], based on the quadrature of the Mellin transform operator. `InvertLT.m` is a DLL (Windows operating system only) that can be used within `Matlab` package.
- `gavsteh.m`[8]: implementation of the Gaver-Stehfest algorithm proposed in [9].

Table 3. Simulation 1: error estimates at $N = 20$.

$\sigma = 0.7, b = 1.7$	AbsErr	CompAbsErr
$t = 1$	3.5×10^{-5}	1.7×10^{-4}
$t = 1.5$	3.2×10^{-5}	2.5×10^{-4}
$t = 2$	4.5×10^{-5}	3.6×10^{-4}
$t = 7$	1.2×10^{-3}	1.9×10^{-2}
$t = 9$	7.2×10^{-3}	4.8×10^{-2}

$\sigma = 2.5, b = 5$	AbsErr	CompAbsErr
$t = 1$	3.5×10^{-8}	1.7×10^{-7}
$t = 1.5$	3.3×10^{-7}	2.3×10^{-7}
$t = 2$	4.5×10^{-6}	3.6×10^{-5}
$t = 7$	5.2×10^{0}	7.9×10^{0}
$t = 9$	2.2×10^{0}	1.8×10^{1}

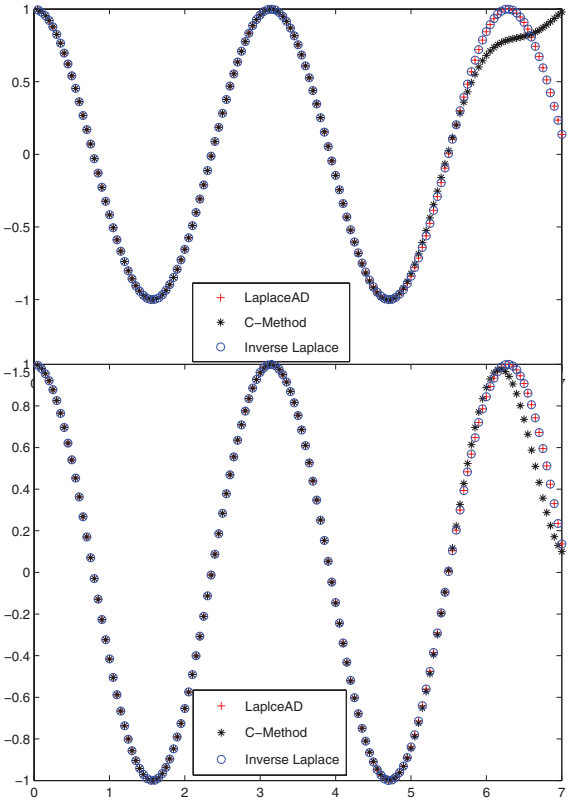

Fig. 1. top: $N = 28, \sigma = 3, b = 6$; bottom: $N = 50, \sigma = 0.7, b = 1.4$.

We choose these two software because, to our knowledge, they are the only ready-to-use software available. As a first test we consider the following functions:

$$F_2(z) = \frac{1}{(z+1)^2}, \quad f_2(t) = texp(-t), \quad \sigma_0 = 0,$$

and we compare the AD-Method ($\sigma = 2$, $b = 4$) with the other two in terms of the absolute error and of the execution time. Results are shown in Fig. 2 and Table 5.

As a second test, we consider the following functions:

$$F_3(z) = \frac{z}{(z^2+1)^2}, \quad f_3(t) = \frac{tsin(t)}{2}, \quad \sigma_0 = 0,$$

and we compare the AD-method ($\sigma = 3$, $b = 6$) with the other two in terms of the absolute error and the execution time. Results are shown in Fig. 2 and Table 5. We note the ability of the proposed scheme to obtain accurate and efficient solutions. A different choice of parameters could potentially achieve better results in [7] and [9].

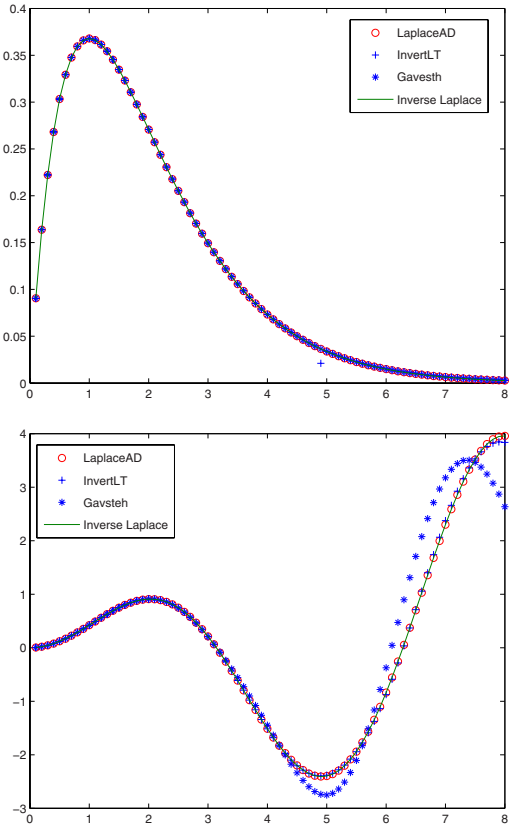

Fig. 2. top: Comparative results on f_2; bottom: Comparative results on f_3.

Table 4. Simulation 1: maximum absolute error in $[0,7]$.

	Max Abserr C-Method	Max Abserr AD-Method
$N=28, \sigma=3, b=6$	0.84	0.9×10^{-4}
$N=50, \sigma=0.7, b=1.4$	0.9×10^{-2}	2.1×10^{-6}

Table 5. left: Comparative results on f_2; right: Comparative results on f_3.

	Abserr	Execution time
InvertLT	0.1×10^{-1}	4.98 sec
Gavsteh	0.3×10^{0}	2.3 sec
AD-Method	0.2×10^{-5}	0.7 sec

	Abserr	Execution time
InvertLT	2.6×10^{-2}	6.7 sec
Gavsteh	0.1×10^{-5}	1.8 sec
AD-Method	0.2×10^{-6}	0.8 sec

5 Conclusions

We use AD for computing the McLaurin coefficients in a numerical algorithm for Laplace transform real inversion. Results confirm the advantages in terms of accuracy and efficiency provided by AD, encouraging the authors to investigate toward the development of a numerical software to be used in applications.

References

1. Bendtsen, C., Stauning, O.: Tadiff, a flexible C++ package for automatic differentiation using Taylor series expansion. http://citeseer.ist.psu.edu/bendtsen97tadiff.html
2. Cuomo, S., D'Amore, L., Murli, A., Rizzardi, M.: Computation of the inverse Laplace transform based on a collocation method which uses only real values. Journal of Computational and Applied Mathematics **1**(198), 98–115 (2007)
3. Garbow, B., Giunta, G., Lyness, N., Murli, A.: Algorithm 662: A Fortran software package for the numerical inversion of a Laplace transform based on Week's method. ACM, Transaction on Mathematical Software **2**(14), 163–170 (1988)
4. Giunta, G., Laccetti, G., Rizzardi, M.: More on Weeks' method for the numerical inversion of the Laplace transform. Numerische Mathematik **2**(54), 193–200 (1988)
5. Giunta, G., Murli, A.: An algorithm for inverting the Laplace transform using real and real sampled function values. In: R. Vichnevetsky, P. Borne, J. Vignes (eds.) Proceedings of IMACS 88: 12th World Congress on Scientific Computation. Paris (1988). Vol. III
6. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. No. 19 in Frontiers in Appl. Math. SIAM, Philadelphia (2000)
7. Kryzhniy, V.: On regularization method for numerical inversion of Laplace transforms. Journal of Inverse and Ill-Posed Problems **12**(3), 279–296 (2004)
8. Srigutomo, W.: Gaver–Stehfest algorithm for inverse Laplace transform, Matlab package, The MathWorks Inc. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9987
9. Stehfest, H.: Algorithm 368: Numerical inversion of Laplace transform. Communication of the ACM **13**, 47–49 (1970)
10. Weeks, W.: Numerical inversion of the Laplace transform using Laguerre functions. Journal of ACM **13**, 419 –429 (1966)
11. Weideman, J.: Algorithms for parameter selection in the weeks method for inverting the Laplace transform. SIAM Journal on Scientific Computing **1**(21), 111–128 (1999)