Laplace Transform Inversion and Viscoelastic Wave Propagation

Richard R. Laverty
Davies Fellow
USMA Department of Mathematical Sciences
West Point, New York

Sponsor: George Gazonas
Research Physicist
USARL Weapons and Materials Research Directorate
Aberdeen Proving Ground, Maryland

Abstract

Inverting Laplace Transforms via the Bromwich Integral has proven to be too difficult, in general, to be a practical method for large classes of problems; therefore, approximation techniques abound. In this paper we will examine four commonly used numerical algorithms for Laplace Transform inversion: The Bellman, Kalaba and Lockett (BKL) method, Jacobi Polynomials, Heaviside Interpolation and the Dubner, Abate and Crump (DAC) algorithm. Each will be evaluated on a broad class of functions to test its robustness, then we will examine how each performs when applied to problems of viscoelastic wave propagation.

Background: The Transform Method

Integral transforms, as a method for solving Initial Value Problems (IVPs) and Boundary Value Problems (BVPs), compete with Separation of Variables and Green's Functions as the most practical method for constructing analytic solutions to large classes of problems. The Laplace Transform has found a tremendous amount of utility in problems that have a time evolution component. It is the goal of this section to introduce what we will call the Laplace Transform method for solving Initial Boundary Value Problems (IBVPs). In subsequent sections we will examine some methods for inverting the Laplace Transform numerically, test each of these methods on some common functions and finally test each method's performance on problems in viscoelastic wave propagation.

The Laplace Transform of a time domain function $f(t)$ is defined by the integral (when it exists):

$$L[f(t)] = \int_0^\infty f(t)e^{-st}\,dt$$

The relevant property of the Laplace Transform that makes it a common tool for solving IVPs can be derived by transforming the time derivatives of $f(t)$ and then performing the necessary number of integrations by parts:

$$\begin{aligned}
L[f'(t)] &= sL[f(t)] - f(0) \\
L[f''(t)] &= s^2 L[f(t)] - sf(0) - f'(0) \\
&\vdots \\
L\left[f^{(n)}(t)\right] &= s^n L[f(t)] - s^{n-1} f(0) - \cdots - sf^{(n-2)}(0) - f^{(n-1)}(0)
\end{aligned}$$

Thus, the process of differentiation in the time domain (t-variable) is reduced to multiplication in the transform domain (s-variable). Therefore, the transform of a differential equation becomes an equation with one less dimension to the domain of its independent variables, i.e. an ODE becomes an algebraic equation, a PDE in two variables becomes and ODE and so on. After being transformed the equation can be solved by the appropriate methods in the reduced domain. Then the transformed solution must be inverted to find the solution to the original problem in the original domain.

As a simple example, consider the IBVP:

$$\begin{aligned}
&\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1 \text{ and } t > 0 \\
&u(x,0) = u_t(x,0) = 0 \quad \text{for } 0 < x < 1 \\
&u(0,t) = 0 \quad \text{for } t > 0 \\
&u_x(1,t) = 1 \quad \text{for } t > 0
\end{aligned}$$

This is the equation for uniaxial displacement, u, of an elastic body initially bounded by the planes $x = 0, 1$, subject to a unit step in stress at its right end and fixed at the left. Although this problem falls within the class that could easily be solved by other methods, we will construct a solution via the Laplace Transform.

First, take the transform of all the quantities in this problem. From now on, we will use a 'hat' to denote transformed quantities and drop any explicit reference to dependence on the transformed variable s; for instance, $\hat{u}(x) = L[u(x,t)]$.

$$\begin{aligned}
&s^2 \hat{u} = \frac{d^2 \hat{u}}{dx^2}, \quad 0 < x < 1 \\
&\hat{u}(0) = 0, \quad \frac{d\hat{u}(1)}{dx} = \frac{1}{s}
\end{aligned}$$

We treat the transform variable s as a parameter and solve the BVP by elementary means.

$$\hat{u}(x) = \frac{sinh(sx)}{s^2\, cosh(s)}$$

By rewriting $\hat{u}(x)$ and using a table of transform pairs we can solve for the displacements:

$$\begin{aligned}
\hat{u}(x) &= \frac{1}{s^2}\frac{e^{sx-s} - e^{-sx-s}}{1+e^{-2s}} \\
&= \sum_{n=0}^{\infty}(-1)^n \frac{e^{-(1+2n-x)s} - e^{-(1+2n+x)s}}{s^2} \\
u(x,t) &= \sum_{n=0}^{\infty}(-1)^n G(1+2n-x, 1+2n-x, t),
\end{aligned}$$

where

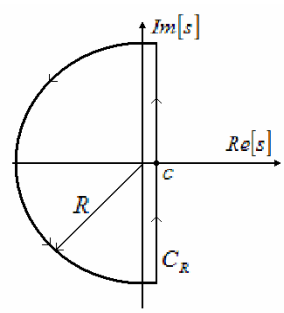
$$G(a,b,t) = \begin{cases} 0, & t < a \\ t-a, & a < t < b \\ b-a, & t > b \end{cases}.$$

Alternatively, we could have inverted $\hat{u}(x)$ by evaluating the Bromwich Integral via residue calculus

$$\begin{aligned}
u(x,t) &= \int_{c-i\infty}^{c+i\infty} e^{st}\hat{u}(x)ds \\
&= \int_{c-i\infty}^{c+i\infty} e^{st}\frac{sinh(sx)}{s^2\, cosh(s)}ds,
\end{aligned}$$

where c is a positive real number chosen so that all the poles of the integrand lie to the left of the line $s = c$.

Evaluation is accomplished by approximating the path of integration from $c - i\infty$ to $c + i\infty$ with a closed contour C_R (called the Bromwich Contour) shown below.



The integral around C_R is found by summing the residues of the integrand that lie within the contour. Then in the limit as $R \to \infty$ the integral around this closed contour will equal the Inverse Laplace Transform provided that the contribution along the half circle of radius R vanishes.

For the function $e^{st}\hat{u}(x)$, there is a second order pole at $s = 0$ and simple poles at $s = i\frac{(2n-1)}{2}\pi$, $n = 0, \pm 1, \pm 2, \ldots$, therefore,

$$
\begin{aligned}
u(x,t) &= \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} e^{st}\frac{sinh(sx)}{s^2\,cosh(s)}ds \\
&= \lim_{R\to\infty}\frac{1}{2\pi i}\oint_{C_R} e^{st}\frac{sinh(sx)}{s^2\,cosh(s)}ds \\
&= \underset{s=0}{Res}\left[e^{st}\frac{sinh(sx)}{s^2\,cosh(s)}\right] + \sum_{n=-\infty}^{\infty}\underset{s=i\frac{(2n-1)\pi}{2}}{Res}\left[e^{st}\frac{sinh(sx)}{s^2\,cosh(s)}\right] \\
&= x + \frac{8}{\pi^2}\sum_{n=1}^{\infty}\frac{(-1)^n}{(2n-1)^2}sin\left(\frac{2n-1}{2}\pi x\right)cos\left(\frac{2n-1}{2}\pi t\right).
\end{aligned}
$$

By the uniqueness of the solution to this IBVP, we've demonstrated that these two series are equal, but more importantly to our purpose we've demonstrated that inversion via the Bromwich Integral is a nontrivial matter in the simplest of all cases; elastic waves. Since we wish to consider more complicated cases of viscoelastic waves we need to consider other methods for inversion.

Approximation of Transform Inversion

This is the situation in which we find ourselves: We have an IBVP where the Laplace Transform seems to be the natural method with which to construct a solution. However, the solution we have in the transformed space does not lend itself nicely to inversion, i.e., it is not tabulated and the inversion integral is not easily evaluated.

Having invested some time and effort into constructing a solution in the transformed space, we would like to use that solution as the basis for a numerical approximation. Each of the following subsections introduces a method that has proved effective in numerically inverting Laplace Transforms. Then the following section will compare all of the methods (presented here) on a broad range of inversion problems to test the robustness of each method. The last section will then compare the methods within the field of wave propagation in viscoelastic materials.

Bellman, Kalaba and Lockett (BKL)

The BKL method [1] is a specific instance of a more general class of methods known as Nystrom (or quadrature) methods for solving integral equations, [2]. Right

now, we will describe the general Nystrom approach in a very simple situation. Then, after a quick example, we will show how to implement the BKL method.

For Nystrom methods in general, consider the definition of the transform itself:

$$\hat{f}(s) = \int_0^\infty f(t) e^{-st} dt \, .$$

The first step is to perform a change of variables so that the domain of integration is finite, then perform a (any) numerical quadrature routine to approximate the value of the integral on the finite domain.

$$\begin{aligned} \hat{f}(s) &= \int_{\phi(0)}^{\phi(\infty)} f\left(\phi^{-1}(x)\right) e^{-s\phi^{-1}(x)} \frac{d\phi^{-1}}{dx} dx \\ &\approx \sum_{i=1}^{N} w_i h_i f_i \end{aligned}$$

Here, ϕ is smooth, monotonic and maps $[0,\infty)$ to a finite domain. The sum represents our quadrature rule with weights w_i and nodes x_i. We have also employed the shorthand:

$$\begin{aligned} h_i &= e^{-s\phi^{-1}(x_i)} \frac{d\phi^{-1}(x_i)}{dx} \\ f_i &= f\left(\phi^{-1}(x_i)\right). \end{aligned}$$

We split the integrand into two pieces (h_i and f_i) for convenience. In our final expression we will have a linear system of algebraic equations for the values of f evaluated at the nodes. Therefore, the f_i will be the components of the vector of unknowns.

To be more clear, we can explicitly denote those terms that depend on s in the approximation of the transform.

$$\hat{f}(s) \approx \sum_{i=1}^{N} w_i h_i(s) f_i$$

If we were to fix a value for s this expression would still hold. In fact, we could arbitrarily choose N different s values and get N separate equations whose only unknowns are the f_i.

$$\hat{f}(s_1) \approx \sum_{i=1}^{N} w_i h_i(s_1) f_i$$
$$\hat{f}(s_2) \approx \sum_{i=1}^{N} w_i h_i(s_2) f_i$$
$$\vdots$$
$$\hat{f}(s_N) \approx \sum_{i=1}^{N} w_i h_i(s_N) f_i$$

Now we can see that our approximation has led us to a linear algebraic system,

$$\overline{c} = B\overline{y} \,,$$

where $B_{ki} = w_i h_i(s_k)$, $y_i = f_i$ and $c_k = \hat{f}(s_k)$.

Thus, we have reduced the approximate inversion of the Laplace Transform to solving a system of linear algebraic equations. Of course, the accuracy of our results will depend heavily upon our quadrature method. We are also limited to finding values of f at the nodes of the quadrature method, which are themselves not in the time domain, but in the image of ϕ. So, finally, to find f in the time domain we must invert ϕ at the nodes.

One inescapable drawback that we will find with this method, regardless of quadrature method, is the conditioning of the linear system. The matrix B will always have a large condition number. This is more of an artifact of the sensitivity of the process of inverting Laplace Transforms than anything easily remedied within the context of this particular approach.

We can illustrate this point with a simple example (this example does not use the BKL method). Consider the transform pair:

$$f(t) = sin(2\pi t)$$
$$\hat{f}(s) = \frac{2\pi}{s^2 + 4\pi^2}$$

If we pretend that we know the transform, $\hat{f}(s)$, but not the time domain function, then we would be attempting to solve the equation

$$\frac{2\pi}{s^2 + 4\pi^2} = \int_0^\infty f(t) e^{-st} dt$$

for $f(t)$. To employ the Nystrom method, we first map $[0,\infty)$ to a finite domain. If we use $x(t) = e^{-t}$, then the integral equation becomes

$$\frac{2\pi}{s^2+4\pi^2} = \int_0^1 x^{s-1} f\left(ln\left(\frac{1}{x}\right)\right)dx .$$

Using the midpoint rule, we can approximate the integral:

$$\frac{2\pi}{s^2+4\pi^2} \approx \sum_{i=1}^{N} \frac{1}{N}\left(\frac{2i-1}{2N}\right)^{s-1} f\left(ln\left(\frac{2N}{2i-1}\right)\right).$$

The midpoint rule on $(0,1)$ uses the weights $w_i = \frac{1}{N}$ and the nodes $x_i = \frac{2i-1}{2N}$, $i = 1,\ldots,N$.

To get N equations for the N unknowns, $f_1, f_2, \ldots, f_N$, we still need to choose N distinct values for s. For convenience we will choose $s_k = k$, $k = 1,\ldots,N$. (We will make the same choice for s in the BKL method.) Therefore, we arrive at our linear system $\bar{c} = B\bar{y}$ with

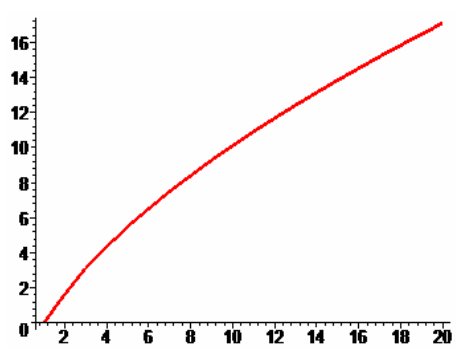
$$c_k = \frac{2\pi}{k^2+4\pi^2},$$

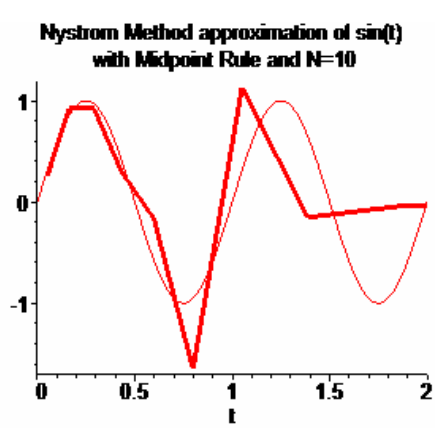
$$B_{ki} = \frac{1}{N}\left(\frac{2i-1}{2N}\right)^{k-1},$$

$$y_i = f\left(ln\left(\frac{2N}{2i-1}\right)\right).$$

It is good to notice that the matrix depends only on our choice of quadrature rule (in this case, the midpoint rule). Only the vector $\bar{c}$ depends on the particular function we wish to invert.

The following figures are of the condition number of the system as a function of N and the final approximation of $f(t)$ using $N = 10$, respectively. Notice that the graph of the condition numbers is nearly linear on the log-plot. Therefore, the condition number grows exponentially with N.





Now, we will examine the BKL method in particular. The BKL method is a Nystrom method, so the general approach is identical to the previous example. In fact, we will perform the same change of variables, $x(t) = e^{-t}$, so that the transform is:

$$\hat{f}(s) = \int_0^1 x^{s-1} f\left(ln\left(\frac{1}{x}\right)\right) dx$$

We could further scale and shift our integration variable x, so that the interval of integration is $(-1,1)$. This is the domain for the Legendre Polynomials which are used to compute the weights and nodes of Gauss-Legendre quadrature; however, we may also use the so-called Shifted Legendre Polynomials [1] and achieve equally accurate results on $(0,1)$ without any further effort.

We will denote the Nth order shifted Legendre polynomial by $P_N(x)$. The nodes of the quadrature method for BKL will be the roots $x_1,\ldots,x_N$ of $P_N(x)$. The weights are calculated from the integral:

$$w_i = \int_0^1 \frac{P_N(x)}{(x - x_i)P_N'(x)} dx .$$

This is enough information to establish the system of linear algebraic equations. However, as we saw with the simple midpoint method, the system will be poorly conditioned. We will side-step some of the consequences of poor conditioning if we can avoid the need to numerically solve the linear system, i.e. we want to find an explicit expression for the components of the vector of unknowns:

$$f_i = \sum_{k=0}^{N-1} a_{ik} \hat{f}(k+1).$$

This can be accomplished, see [1] for details, if we choose $a_{ik} = \frac{q_{ki}}{w_i}$, where the w_i are the weights calculated above and the q_{ki} are the coefficients of a polynomial

$$p_i(x) = q_{0i} + q_{1i}x + q_{2i}x^2 + \cdots + q_{N-1i}x^{N-1}$$

with the property at the nodes

$$p_i(x_j) = \delta_{ij} .$$

So, we have an explicit formula for f at the zeros of the shifted Legendre polynomials. Once we invert the change of variables, so that we can find f as a function of t, then we have completed the BKL method.

Jacobi Polynomials

The Jacobi polynomial method is one of several methods that are referred to as orthogonal polynomial methods, [3]. All orthogonal polynomial methods begin by expressing the time domain function as a series in terms of the relevant set of basis functions.

$$f(t) = \sum_n c_n \varphi_n(t)$$

Transforming the series,

$$\hat{f}(s) = \sum_n c_n \hat{\varphi}_n(s).$$

The orthogonality of the basis functions will provide some way to express the first N coefficients of the transformed series in terms of $\hat{f}$ evaluated at a set of discrete points. These coefficients are then employed to build a truncated expansion for $f(t)$.

We will choose $\varphi_n(t) = P_n^{(0,\beta)}(t)$, the nth order Jacobi polynomial defined by

$$P_n^{(0,\beta)}(x) = \frac{(-1)^n}{2^n n!}(1+x)^{-\beta}\frac{d^n}{dx^n}\left[(1-x)^n(1+x)^{n+\beta}\right].$$

The Jacobi polynomials are orthogonal on the interval $[-1,1]$ with respect to the weight function $(1+x)^\beta$, $\beta > 1$. Therefore, it is necessary to initially change variables so that f is defined on $[-1,1]$:

$$g(x) \equiv f\left(-\frac{1}{\delta}ln\left(\frac{x+1}{2}\right)\right) = \sum_{n=0}^{\infty} c_n P_n^{(0,\beta)}(x)$$

Under this change of variables, $\hat{f}$, the Laplace Transform of f is found by

$$\begin{aligned}\hat{f}(s) &= \frac{1}{2\delta}\int_{-1}^{1}\left(1+\frac{x}{2}\right)^{\frac{s}{\delta}-1} g(x)dx \\ &= \frac{1}{2\delta}\int_{-1}^{1}\left(1+\frac{x}{2}\right)^{\frac{s}{\delta}-1}\left[\sum_{n=0}^{\infty} c_n P_n^{(0,\beta)}(x)\right]dx.\end{aligned}$$

If this expression is evaluated at $s = \delta(\beta + k + 1)$, where k is an arbitrary positive integer, then

$$\hat{f}(\delta(\beta+k+1)) = \frac{2^{\beta+k+1}}{\delta}\int_{-1}^{1}(1+x)^{\beta+k}\left[\sum_{n=0}^{\infty} c_n P_n^{(0,\beta)}(x)\right]dx$$
$$= \frac{2^{\beta+k+1}}{\delta}\int_{-1}^{1}(1+x)^{\beta}\left[\sum_{n=0}^{\infty}(1+x)^k c_n P_n^{(0,\beta)}(x)\right]dx.$$

The term $(1+x)^k$ can be expressed as a finite sum of Jacobi polynomials. Using the orthogonality of the Jacobi polynomials, the integral on the right can be evaluated exactly as a finite sum. Therefore, by choosing N different values of k we can solve for $c_0,\ldots,c_{N-1}$. This will provide us with the means to approximate $g(x)$ with a truncated series of Jacobi polynomials. Then it is just a matter of inverting the change of variables to have an equivalent expression that approximates $f(t)$.

$$c_0 = (\beta+1)\delta\hat{f}(\delta(\beta+1))$$
$$c_1 = (\beta+2)(\beta+3)\left[\delta\hat{f}(\delta(\beta+2)) - \frac{c_0}{\beta+2}\right]$$
$$c_n = \frac{\prod_{j=1}^{n+1}(n+\beta+j)}{\prod_{j=0}^{n-1}(n-j)}\left[\delta\hat{f}(\delta(\beta+n)) - \sum_{m=0}^{n-1} c_m \frac{\prod_{j=0}^{m-1}(n-j)}{\prod_{j=1}^{m+1}(n+\beta+j)}\right], \quad n \geq 2$$
$$f(t) \approx \sum_{n=0}^{N-1} c_n P_n^{(0,\beta)}\left(2e^{-\delta t} - 1\right)$$

The other orthogonal polynomial methods follow similar steps and yield the same type of result; a truncated expansion of $f(t)$ in terms of the appropriate basis functions (after a change of variables). One advantage of these methods, in general, is that the change of variables to the proper domain for the basis functions is never unique. This allows some flexibility by providing a 'free' parameter. In the case of Jacobi polynomials, δ is an adjustable parameter whose only restriction is that it must be positive. We will see that the choice of δ does have a measurable effect on the performance of this method.

Heaviside Interpolation

The Heaviside Interpolation method is carried out in two steps. First, the transformed function, $\hat{f}(s)$, is approximated by a rational function $\hat{F}(s)$. Second, the approximating function is inverted using the Heaviside Expansion Theorem, [4]. We presume that $f(t) \approx F(t)$, provided that $\hat{F}$ is an accurate approximation of $\hat{f}$.

We construct the approximating function, $\hat{F}$, by interpolating $\hat{f}$. We will see that the accuracy of this approximation depends heavily on our choice of nodes for interpolation, $s_1, s_2, \ldots, s_N$. Therefore, our end result (in the time domain) is also sensitive to this choice. At present, we do not have an effective, algorithmic method for choosing the nodes that will yield the best approximation to $\hat{f}$, and by association the best approximation to f. It does seem reasonable however, that search methods would be an effective approach to discovering good, if not optimal, nodes. (Our suspicion is that direct optimization will prove near impossible due to the complicated nature of any error valued functional.)

We will demonstrate this method on a simple example, to illustrate the process as well as indicate the sensitivity of the final result to our choice of nodes. We begin with a statement of the Heaviside Expansion Theorem, [4].

<u>Theorem</u>: Let $\hat{F}(s)$ be a rational function with the degree of the denominator at least one greater than the numerator.

$$\hat{F}(s) = \frac{P(s)}{Q(s)}$$

Then we may find a partial fraction decomposition of $\hat{F}(s)$

$$\hat{F}(s) = \sum_{k=1}^{M} \sum_{n=1}^{N_n} \frac{a_{n,k}}{(s - s_k)^n},$$

where $s_1, s_2, \ldots, s_M$ are the zeros of $Q(s)$, (not to be confused with the interpolation nodes mentioned above), and $N_1, N_2, \ldots, N_M$ are the orders of the associated zeros. Furthermore, the Inverse Laplace Transform of $\hat{F}(s)$ is

$$F(t) = \sum_{k=1}^{M} \sum_{n=1}^{N_k} \frac{a_{n,k}}{(n-1)!} t^{n-1} e^{s_k t}.$$

<u>Proof</u>: Since $Q(s)$ is a polynomial with real coefficients, we will always be able to construct a factored form for $Q(s)$:

$$Q(s) = (s - s_1)^{N_1} (s - s_2)^{N_2} \cdots (s - s_M)^{N_M},$$

where the s_k are the (possibly complex) roots of $Q(s)$ and the N_k are the corresponding orders, $k = 1, \ldots, M$. From this point, construction of the partial fraction decomposition is an arithmetic exercise.

Inversion of the decomposed $Q(s)$ follows from an evaluation of the inversion integral. Since all sums are finite, the linearity of the transform and its inverse allow

us to examine the inversion of a single representative term. Therefore, consider the following integral

$$\begin{aligned}
\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} e^{st}\frac{a_{n,k}}{(s-s_k)^n}ds &= \lim_{R\to\infty}\frac{1}{2\pi i}\oint_{C_R} e^{st}\frac{a_{n,k}}{(s-s_k)^n}ds \\
&= \operatorname*{Res}_{s=s_k}\left[e^{st}\frac{a_{n,k}}{(s-s_k)^n}\right] \\
&= \frac{a_{n,k}}{(n-1)!}\frac{d^{n-1}}{ds^{n-1}}\left(e^{st}\right)\Big|_{s=s_k} \\
&= \frac{a_{n,k}}{(n-1)!}t^{n-1}e^{s_k t}
\end{aligned}$$

The contour C_R is the Bromwich Contour, and the result follows.

Now, to demonstrate the sensitivity of the Heaviside Interpolation method to the choice of nodes consider the transform pair

$$\begin{aligned}
f(t) &= J_0(t) \\
\hat{f}(s) &= \frac{1}{\sqrt{s^2+1}}.
\end{aligned}$$

If we choose the interpolation nodes $s_1 = 1,\ s_2 = 2, \ldots, s_5 = 5$ then the rational function that interpolates $\hat{f}$ is

$$\hat{F}(s) = \frac{0.025s^2 + 31.68s + 17.62}{32.06s^2 + 15.35s + 22.34},$$

(coefficients have been rounded). We can invert this $\hat{F}$ exactly using the Heaviside Expansion Theorem

$$F(t) = 0.99e^{-0.24t}\,cos(0.80t) + 0.39e^{-0.24t}\,sin(0.80t) + 7.69\times10^{-4}\,\delta(t).$$

Again, coefficients have been rounded. One quantitative measure of the accuracy of this approximation is the Mean Squared Error (MSE). For this problem, we will use the interval $[0,2\pi]$.

$$\begin{aligned}
MSE &= \frac{1}{2\pi}\int_0^{2\pi}\left[J_0(t) - F(t)\right]^2 dt \\
&= 7.970\times10^{-3}
\end{aligned}$$

This is not such a bad approximation, especially when we consider that we only used values of the transform at 5 locations to perform the inversion. Now, suppose

we choose 5 different locations to interpolate $\hat{f}$: $s_1 = \frac{1}{2}$, $s_2 = 1, \ldots, s_5 = \frac{5}{2}$. Then the rational approximation to $\hat{f}$ will be

$$\hat{F}(s) = \frac{s^2 + 2s}{s^2 - 10s + 8},$$

and this can be inverted:

$$F(t) = \delta(t) - 0.31e^{0.88t} + 12.31e^{9.12t}$$
$$MSE = 8.133 \times 10^{49}.$$

This example was chosen because of its sensitivity to the choice of interpolation nodes, but it is not a rare example. Almost any function, in the transform domain, that can't be expressed exactly as a ratio of polynomials will experience a similar dependence.

Dubner, Abate and Crump (DAC)

The algorithm attributed to Dubner and Abate was originally derived by applying a Finite Fourier Cosine series to approximate the Laplace Inversion integral, [5]. It was observed at the time that an alternative approach that would lead to the same algorithm was to apply a trapezoid rule approximation to the inversion integral, which can be expressed as

$$f(t) = \frac{2e^{ct}}{\pi} \int_0^\infty Re\left[\hat{f}(s)\right] cos(\omega t) d\omega,$$

when one takes into account that the time domain function $f(t)$ is real valued and we limit our attention to the even, periodic extension of f on some suitably chosen interval $[0, T]$. The resulting algorithm produced excellent results on $\left[0, \frac{T}{2}\right]$.

Later, Crump re-evaluated the approach of Dubner and Abate by loosening the restriction on the symmetry of f and relying merely on the real-valued nature of the time domain function, [6]. Therefore, Crump was approximating the integral

$$f(t) = \frac{e^{ct}}{\pi} \int_0^\infty \left[Re\left\{\hat{f}(c + i\omega)\right\} cos(\omega t) - Im\left\{\hat{f}(c + i\omega)\right\} sin(\omega t)\right] d\omega$$

with a trapezoidal rule. By choosing the step size

$$\Delta\omega = \frac{\pi}{T}$$

he arrived at the approximation

$$f(t) \approx \frac{e^{ct}}{T}\left[\frac{\hat{f}(c)}{2} + \sum_{n=1}^{\infty}\left(Re\left[\hat{f}\left(c + i\frac{n\pi}{T}\right)\right]cos\left(\frac{n\pi}{T}t\right) - Im\left[\hat{f}\left(c + i\frac{n\pi}{T}\right)\right]sin\left(\frac{n\pi}{T}t\right)\right)\right].$$

This improves the approximation by using the information about the imaginary part of $\hat{f}$, information that is discarded by Dubner and Abate. As a result, the interval upon which error estimates remain reliable is extended to the interval $[0,2T]$. This improved algorithm has now come to be known as the Dubner, Abate and Crump algorithm, or just DAC.

Performance of Approximation Methods in General

Now, we will examine how each method performs, in general. Each will be tested on a selection of typical functions and the results will be tabulated. For each method, instances where performance was exceptionally good or bad will be further illustrated with plots of the exact inverse and the outcome of the particular approximation procedure.

As a general measure of performance, we will use the Mean Squared Error (MSE) defined as:

$$MSE = \frac{1}{T}\int_0^T \left[f(t) - f_{approx}(t)\right]^2 dt$$

where $f_{approx}(t)$ is the approximation to the exact solution.

BKL Method

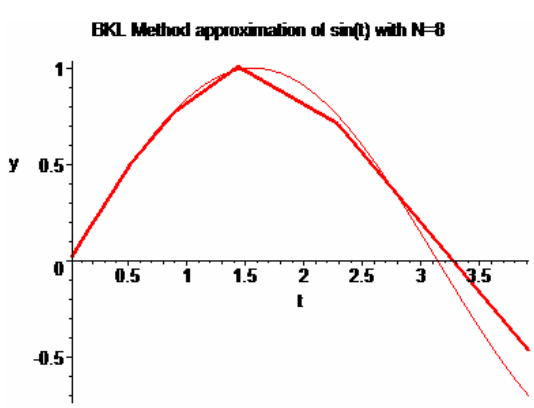
The following table summarized the results of the BKL method. There are two deviations in this table from the data in the tables that follow. First, the intervals upon which the approximation was measured. Using the BKL method we are locked-in to an interval, determined by the zeros of the Shifted Legendre Polynomials. We could alter the location of these points in the time domain by means of a different change of variables (we used $x = e^{-t}$), but the effort is not justified for a this type of quick examination. Also, in [1], methods for extending the domain of the approximation are discussed, but the accuracy will be the same. The second deviation is the calculation of the MSE. All of the methods that follow produce continuous approximations to $f(t)$; the BKL method generates a discrete approximation, therefore it seemed more appropriate to use the following measure of error:

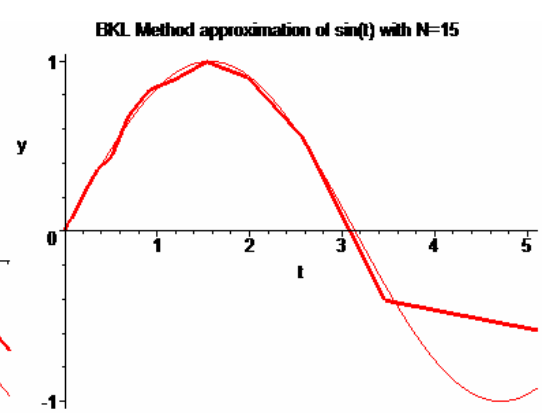
$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left[f(t_i) - f_i\right]^2$$

where the t_i correspond to the zeros of the Shifted Legendre polynomials and the f_i are the method's approximations to $f(t_i)$.

$f(t)$	$\hat{f}(s)$	**Interval**	**MSE**	**Comments**
t	$\frac{1}{s^2}$	$(0,4.339)$	0.0117	$N=10$
$\frac{t^2}{2}$	$\frac{1}{s^3}$	$(0,4.339)$	0.4662	$N=10$
$\frac{t^3}{6}$	$\frac{1}{s^4}$	$(0,4.339)$	4.617	$N=10$
e^{-t}	$\frac{1}{s+1}$	$(0,4.339)$	9.462×10^{-9}	$N=10$
$sin(t)$	$\frac{1}{s^2+1}$	$(0,4.339)$	0.0111	$N=10$
$cos(t)$	$\frac{s}{s^2+1}$	$(0,4.339)$	0.0029	$N=10$
$ln(t)$	$\frac{-(\gamma+ln(s))}{s}$, γ Euler's Constant	$(0,4.339)$	0.0122	$N=10$
$J_0(t)$	$\frac{1}{\sqrt{s^2+1}}$	$(0,4.339)$	0.0014	$N=10$
$H(t-1)=\begin{cases}0, & t<1\\ 1, & t>1\end{cases}$	$\frac{e^{-s}}{s}$	$(0,4.339)$	0.0021	$N=10$

Clearly, performance was best on the exponentially damped function and worst on the high order polynomials. One thing about this method that is not clear from this table is the relative improvement as N is increased. Unlike the methods that follow, there is little gained by increasing N. As an example, consider $f(t)=sin(t)$:





On the left is the $N = 8$ approximation and on the right $N = 15$. The higher order approximation is only slightly better, with an MSE of 0.00602 as opposed to 0.0072. (Note also that the domains are slightly different: $N = 8$ is on (0,3.919) and $N = 15$ is on (0,5.115).)

In general, the BKL method performs well when compared to the others but suffers from some serious drawbacks. Chief among its detractors is that to change the time domain is a non-trivial exercise – so we are restricted to an approximation on one particular interval. Another, less severe, disadvantage is that all other methods generate a continuous approximation; BKL generates a set of points. In the physical application of interest (viscoelastic wave propagation), the stresses are the derivatives of the displacement and are often of a greater physical importance. By using a discrete approximation to the displacements we must introduce another level of approximation to find the stresses.

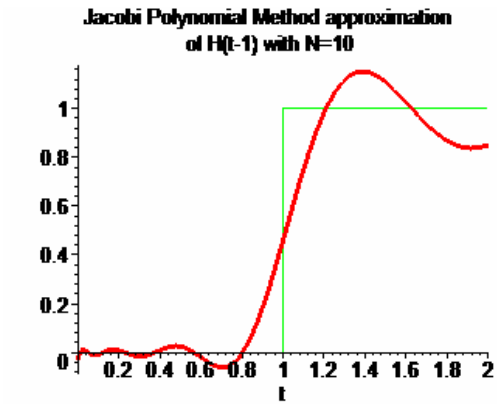
Jacobi Polynomials

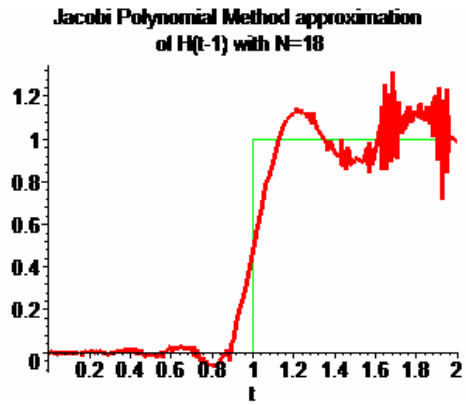
The following table summarized the results of the Jacobi polynomial method. It's not surprising that the polynomial method performed very well on inverting polynomials. The exponentially damped function was a surprise.

$f(t)$	$\hat{f}(s)$	**Interval**	**MSE**	**Comments**
t	$\frac{1}{s^2}$	$(0,1)$	2.154×10^{-7}	$N=5$ $\beta=5,\ \delta=1$
$\frac{t^2}{2}$	$\frac{1}{s^3}$	$(0,1)$	2.091×10^{-6}	$N=5$ $\beta=5,\ \delta=1$
$\frac{t^3}{6}$	$\frac{1}{s^4}$	$(0,1)$	3.76×10^{-6}	$N=5$ $\beta=5,\ \delta=1$
e^{-t}	$\frac{1}{s+1}$	$(0,1)$	1.518×10^{-26}	$N=5$ $\beta=5,\ \delta=1$
$sin(t)$	$\frac{1}{s^2+1}$	$(0,2\pi)$	0.0617	$N=5$ $\beta=0.1,\ \delta=0.9$
$cos(t)$	$\frac{s}{s^2+1}$	$(0,2\pi)$	0.0726	$N=5$ $\beta=3,\ \delta=0.5$
$ln(t)$	$\frac{-(\gamma+ln(s))}{s}$, γ Euler's Constant	$(0,2e)$	0.1505	$N=5$ $\beta=1,\ \delta=1$
$J_0(t)$	$\frac{1}{\sqrt{s^2+1}}$	$(0,4\pi)$	0.0281	$N=5$ $\beta=4,\ \delta=0.1$
$H(t-1)=\begin{cases}0, & t<1\\ 1, & t>1\end{cases}$	$\frac{e^{-s}}{s}$	$(0,2)$	0.0224	$N=5$ $\beta=3,\ \delta=0.5$

The poor performance for the logarithm can be attributed to a general difficulty for polynomials to approximate vertical asymptotes. This causes us no great concern since this is outside of the expected behavior of wave propagation.

Before we leave the Jacobi polynomials it would be nice to take a closer look at how well it works with discontinuities. Of course, with only 5 terms it is difficult to approximate a jump discontinuity; the following plots demonstrate how the method can perform, given a few more terms:





We can see that increasing the order of the approximating polynomial can result in unstable results for fairly low values of N.

Heaviside Interpolation

At a glance, we can see that the Heaviside Interpolation method is very robust for smooth functions. For several of the chosen transforms the method finds the exact inverse.

$f(t)$	$\hat{f}(s)$	**Interval**	**MSE**	**Comments**
t	$\frac{1}{s^2}$	$(0,1)$	0	$N=5, s_j = j$
$\frac{t^2}{2}$	$\frac{1}{s^3}$	$(0,1)$	0.0001	$N=5, s_j = j+1$
$\frac{t^3}{6}$	$\frac{1}{s^4}$	$(0,1)$	0.0004	$N=5, s_j = j$
e^{-t}	$\frac{1}{s+1}$	$(0,1)$	0	$N=5, s_j = j$
$sin(t)$	$\frac{1}{s^2+1}$	$(0,2\pi)$	0	$N=5, s_j = j$
$cos(t)$	$\frac{s}{s^2+1}$	$(0,2\pi)$	0	$N=5, s_j = j$
$ln(t)$	$\frac{-(\gamma+ln(s))}{s}$, γ Euler's Constant	$(0,2e)$	0.0528	$N=5, s_j = j-\frac{1}{2}$

$J_0(t)$	$\frac{1}{\sqrt{s^2+1}}$	$(0,4\pi)$	0.0198	$N=5, s_j=j$
$H(t-1)=\begin{cases}0, & t<1\\ 1, & t>1\end{cases}$	$\frac{e^{-s}}{s}$	$(0,2)$	0.2645	$N=5, s_j=\frac{j}{5}$

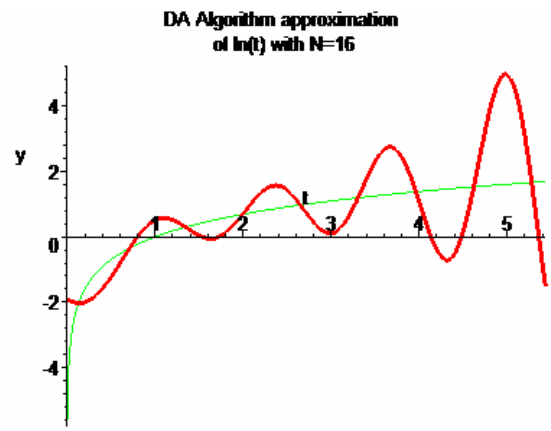
DAC Algorithm

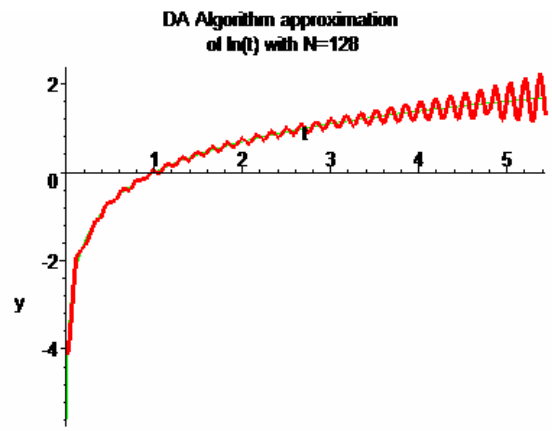
When executing the DAC algorithm we found that the advantages of the Dubner and Abate method [5] performed equal with the full DAC algorithm. Therefore, the "DA" algorithm was used to generate the inverses in this table.

One surprise in this table is the mediocre performance on sinusoids. It seems reasonable, given the Jacobi polynomial method's excellence with polynomials, that a method that finds a truncated Fourier Series would perform well with sines and cosines.

$f(t)$	$\hat{f}(s)$	**Interval**	**MSE**	**Comments**
t	$\frac{1}{s^2}$	$(0,1)$	2.33×10^{-4}	$c=4.605$
$\frac{t^2}{2}$	$\frac{1}{s^3}$	$(0,1)$	2.43×10^{-8}	$c=4.605$
$\frac{t^3}{6}$	$\frac{1}{s^4}$	$(0,1)$	1.51×10^{-9}	$c=4.605$
e^{-t}	$\frac{1}{s+1}$	$(0,1)$	0.0194	$c=4.605$
$sin(t)$	$\frac{1}{s^2+1}$	$(0,2\pi)$	0.0243	$c=0.733$
$cos(t)$	$\frac{s}{s^2+1}$	$(0,2\pi)$	0.0184	$c=0.733$
$ln(t)$	$\frac{-(\gamma+ln(s))}{s}$, γ Euler's Constant	$(0,2e)$	1.932	$c=0.8471$
$J_0(t)$	$\frac{1}{\sqrt{s^2+1}}$	$(0,4\pi)$	0.0376	$c=0.3665$
$H(t-1)=\begin{cases}0, & t<1\\ 1, & t>1\end{cases}$	$\frac{e^{-s}}{s}$	$(0,2)$	0.0226	$c=2.3026$

One positive note for the "DA" algorithm is the ease with which higher order approximations can be computed. By increasing N to 128 we can improve the DA performance on it's otherwise worst transform by two orders of magnitude to an MSE of 0.02686 (from 1.932 with $N=16$).





Performance of Approximation Methods on Viscoelastic IBVPs

We will now consider the performance of each of these methods when inverting Laplace Transforms arising out of viscoelastic IBVPs. The physical problem we will focus on will be the propagation of a wave through a semi-infinite, homogeneous viscoelastic material. The uniaxial equation of motion is

$$\frac{\partial \sigma}{\partial x} = \rho \frac{\partial^2 u}{\partial t^2}, \quad x > 0, \quad t > 0,$$

where σ is the stress, ρ the density and u the displacement. The body is initially at rest and a stress is applied at the free end, $x = 0$, at $t = 0$:

$$\begin{aligned} u(x,0) &= 0 \\ \frac{\partial u(x,0)}{\partial t} &= 0 \\ \sigma(0,t) &= \sigma_0 H(t) \\ \lim_{x\to\infty} \sigma(x,t) &= 0 \end{aligned}$$

We will use an hereditary integral formulation for the constitutive law, relating the stress and strain:

$$\sigma(x,t) = \int_0^t Y(t-T) \frac{\partial \varepsilon(x,T)}{\partial \tau} dT,$$

where $\varepsilon = \dfrac{\partial u}{\partial x}$ is the uniaxial strain, $Y(t)$ is the relaxation modulus and we have assumed that the material is initially unstrained. The relaxation modulus for several materials is summarized in the following table.

Type	Relaxation Modulus
Pure Elastic Material (solid)	$Y(t) = E$ (constant)
Pure Viscous Material (fluid)	$Y(t) = \eta\delta(t)$
Maxwell Material (fluid)	$Y(t) = Ee^{-\frac{Et}{\eta}}$
Voight Material (solid)	$Y(t) = E + \eta\delta(t)$

Regardless of the particular constitutive law, the Laplace Transform (applied to the t-variable) of the equation of motion gives us the ODE

$$s\hat{Y}\frac{d\hat{\varepsilon}}{dx} = \rho s^2\hat{u}, \quad \text{or}$$
$$s\hat{Y}\frac{d^2\hat{u}}{dx^2} = \rho s^2\hat{u},$$

subject to the boundary conditions:

$$\hat{\sigma}(0) = \hat{Y}\frac{d\hat{u}(0)}{dx} = \frac{\sigma_0}{s}, \quad \text{and}$$
$$\lim_{x\to\infty}\hat{u}(x) = 0.$$

The solution to this BVP is

$$\hat{u}(x) = -\frac{\sigma_0}{s^2\hat{Y}\hat{\Omega}}e^{-\hat{\Omega}x},$$

or if we are interested in the stress

$$\hat{\sigma}(x) = -\frac{\sigma_0}{s}e^{-\hat{\Omega}x},$$

where $\hat{\Omega}(s) = \sqrt{\frac{\rho s}{\hat{Y}(s)}}$.

After introducing the non-dimensional variables

$$\xi = \frac{\sqrt{\rho E}}{\eta}x, \quad \tau = \frac{E}{\eta}t$$

and the non-dimensional stress

$$\Sigma(\xi,\tau)=\frac{\sigma(\xi,\tau)}{\sigma_0}$$

we can find the following time domain solutions after analytical inversion of the Laplace Transform, [7,8,9].

Material	**Stress Transform**	**Nondimensional Stress**
Elastic	$\hat{\Sigma}(\xi,s)=\frac{e^{-\xi s}}{s}$	$\Sigma(\xi,\tau)=H(\tau-\xi)$
Viscous	$\hat{\Sigma}(\xi,s)=\frac{e^{-\xi\sqrt{s}}}{\sqrt{s}}$	$\Sigma(\xi,\tau)=\frac{1}{\sqrt{\pi\tau}}e^{-\frac{\xi^2}{4\tau}}$
Maxwell	$\hat{\Sigma}(\xi,s)=\frac{e^{-\xi\sqrt{s^2+s}}}{\sqrt{s^2+s}}$	$\Sigma(\xi,\tau)=e^{-\frac{\tau}{2}}I_0\left(\frac{1}{2}\sqrt{\tau^2-\xi^2}\right)H(\tau-\xi)$
Voight	$\hat{\Sigma}(\xi,s)=\sqrt{\frac{1+s}{s^2}}e^{-\xi\sqrt{\frac{s^2}{1+s}}}$	$\Sigma(\xi,\tau)=\frac{e^{\tau}}{\sqrt{\pi}}\left[\frac{e^{-\frac{\xi^2}{4\tau}}}{\sqrt{\tau}}-\int_0^{\tau}\frac{J_1\left(2\sqrt{\lambda\tau-\lambda^2}\right)}{\sqrt{\tau-\lambda}}e^{-2(\tau-\lambda)-\frac{\xi^2}{4\lambda}}d\lambda\right]$

We've already seen how each of these methods performs on the purely elastic material – BKL outperformed all the others by at least one order of magnitude, (although the measurement was made differently). The following tables summarize the performance on the viscous materials, then Maxwell materials.

Performance on Viscous materials

Method	**MSE**	**Interval**	**Notes**
BKL	1.025×10^{-5}	$(0,4.339)$	$N=10$
Jacobi polynomials	0.0233	$(0,4)$	$N=10,\ \beta=3,\ \delta=1$
Heaviside Interpolation	0.003623	$(0,10)$	$N=5,\ s_j=j$
DA algorithm	9.0×10^{-4}	$(0,4e)$	$N=64,\ c=0.8471$

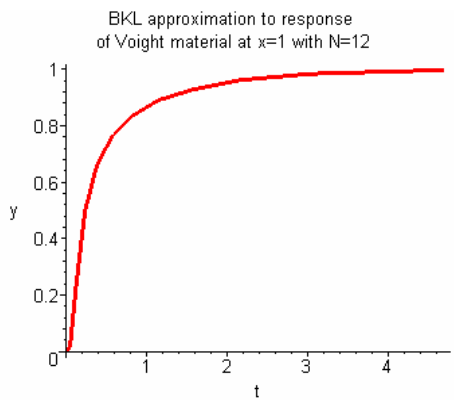
Performance on Maxwell materials

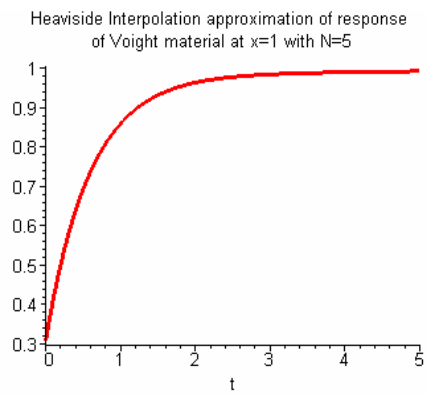
Method	**MSE**	**Interval**	**Notes**
BKL	4.0125×10^{-4}	$(0,4.686)$	$N=12$
Jacobi polynomials	Unreliable	$(0,4)$	$N=12,\ \beta=3,\ \delta=1$
Heaviside Interpolation	0.06158	$(0,5)$	$N=5,\ s_j=\frac{j}{3}$
DA algorithm	0.00142	$(0,4e)$	$N=64,\ c=0.8471$

We can see that there are little surprises, based on experience from the general performance of all of these methods. The Jacobi polynomial method would probably be considered least appropriate due to the general difficulty of mimicking non-polynomial behavior, i.e. approaching a finite limit as $t \to \infty$.

One thing that a quick glance at the MSE of each method does not reveal is the order of the approximation. To achieve similar results, the DA algorithm had to use 64 terms, while the others were between 5 and 12.

We did not include performance of the methods for the Voight material. This is primarily due to the difficulty in accurately computing the MSE (the difficulty lies in evaluating the exact solution, not the approximations). The following two graphs show the BKL and the Heaviside Interpolation approximations to this problem.





Both of these exhibit qualitative behavior consistent with the exact solution, [8]. The other methods could not achieve this. The Jacobi polynomials grow too quickly and the DA algorithm requires too many terms to be competitive with these methods.

Conclusion

We have examined four popular methods for inverting Laplace transforms. First on some simple transforms, to get an idea of where each performs well, then applied to some wave problems. We found that the Jacobi polynomial method does well with polynomials, which is to be expected, but it's ultimate behavior for large t (polynomial behavior) make it a poor method for bounded functions. The Heaviside Interpolation method works well with functions that decay (such as exponentials), but does poorly with discontinuities. The BKL method appears to be very robust, but additional work is needed to extend the time domain of the approximation. And last, the DA (or DAC) algorithm. It also appears very robust, and even performs relatively well on discontinuous functions – a very desirable property when working with wave propagation problems.

When viscoelastic wave problems are considered, it appears that the BKL and the Heaviside Interpolation methods performed best. Especially when considering the more complicated behavior of Voight material. These were the only methods that reproduced the expected behavior with extremely low order approximations. This dependability makes it more than justified to continue to examine ways to adapt each

of these methods to our purposes. The BKL method needs to be able to make approximations on larger time domains and we need to develop an automated method for choosing the nodes of interpolation in the Heaviside method.

References

[1] Bellman, R., Kalaba, R., Lockett, J., *Numerical Inversion of the Laplace Transform: Applications to Biology, Economics, Engineering, and Physics*, Elsevier, NY, 1966.

[2] Delves, L. and Mohamed, J., *Computational methods for integral equations*, Cambridge University Press, Cambridge, 1985.

[3] Miller, M. and Guy, W., Numerical Inversion of the Laplace Transform by use of Jacobi Polynomials, SIAM J. Numer. Anal., vol. 3(4), 1966, 624-635.

[4] LePage, W., *Complex Variables and the Laplace Transform for Engineers*, Dover, NY, 1961.

[5] Dubner, H. and Abate, J., Numerical Inversion of Laplace Transforms by Relating Them to the Finite Fourier Cosine Transform, J. ACM, vol. 15(1), 1968, 115-123.

[6] Crump, K., Numerical Inversion of Laplace Transforms using a Fourier Series Approximation, J. ACM, vol. 23(1), 1976, 89-96.

[7] Lee, H. and Kanter, I., Wave Propagation in Finite rods of Viscoelastic Material, J. Appl. Phys., vol. 24(9), 1953, 1115-1122.

[8] Lee, H. and Morrison, J. A Comparison of the Propagation of Longitudinal Wave in Rods of Viscoelastic Materials, J. Polymer Sci., vol. 19, 1956, 93-110.

[9] Christensen, R., *Theory of Viscoelasticity, 2nd Edition*, Dover, Mineola, NY, 1982.