JOURNAL OF COMPUTATIONAL PHYSICS 110, 164–170 (1994)

Methods for Fast Computation of Integral Transforms

SHAY GUERON

Center For Applied Mathematics, 504 ETC Building, Cornell University, Ithaca, New York 14853

Received August 1, 1990; revised November 19, 1992

This paper is concerned with two aspects of the numerical calculation of integral transforms. The first is finding a necessary and sufficient condition that enables converting an integral transform into a correlation (convolution) form. The condition and the transformation that implements it are generalizations of the Gardner transformation and derived in the paper. This technique can be applied to a wide class of integral transforms and is shown to reduce the computational complexity and storage requirements of the resulting algorithm. The second issue addressed in the paper is the accuracy of the calculation of the correlation integral, obtained by the above transformation, for a given number of samples. It is shown how the standard FFT method can be applied in combination with various numerical integration rules. This proves to be an important factor in expediting the computations, reducing the storage requirements, and improving the accuracy.

1. INTRODUCTION

The numerical approximation of integral transforms is important in many scientific and engineering problems and can present a formidable task in terms of computer time and storage requirements. The need for fast algorithms becomes especially crucial when the kernels of the transforms (or the transformed functions) are oscillatory.

In this paper we treat two problems related to the numerical calculation of integral transforms. The first is finding a necessary and sufficient condition that enables the conversion of a given integral transform into a correlation (convolution) integral. The condition and the corresponding change of variables are derived in Section 2 and generalize the well-known Gardner transformation (Gardner *et al.* [8]). We show that many of the commonly used integral transforms can be converted into a correlation form by means of a change of variables.

Correlation integrals can be calculated with significantly reduced computational effort and memory requirements. This reduction of effort can be achieved by applying the FFT algorithm to the discretized problem. Traditionally, the discrete evaluation of the transform is performed by means of the rectangular quadrature rule. In Section 2 we introduce a modification of this standard approach. At the cost of some extra multiplications, finer quadrature rules can be applied, thereby reducing the number of discretization points needed to obtain a desired accuracy. This modification is fast, easy to implement, reduces storage requirements and improves the accuracy.

In Sections 3 and 4 we give two examples that present typical difficulties involved with the change of variables and demonstrate ways to overcome them. With our approach, one can apply the same integration rules that might have been used for the original problem. Consequently, the advantage of the transformation depends only on the inherent properties of the problem (the integrands). The first example shows how our transformation can change the properties of the integrand and make it singular and that the flexibility to use different integration rules is crucial. In the second example we use the new technique to improve the fast Hankel transform (Siegman [17]).

2. CHANGING AN INTEGRAL TRANSFORM INTO A CORRELATION FORM

Consider the integral transform

$$F(x) = \int_a^b K(x, y)\, g(y)\, dy, \qquad x \in [c, d], \tag{2.1}$$

where the kernel, $K(x, y)$, and the transformed function, $g(y)$, may be real or complex. Computing $F(x)$ can be a time-consuming task requiring a large storage space. In this section we investigate the cases where Eq. (2.1) can be converted into the correlation integral

$$F_1(u) = \int_{a_1}^{b_1} H(u + v)\, \tilde{g}(v)\, dv, \qquad u \in [c_1, d_1]. \tag{2.2}$$

(Note that replacing v with $-v$ in Eq. (2.2) results a convolution form. Thus, we do not distinguish between correlations and convolutions through our discussion.) We shall show that an integral transform of the form shown above has properties which may be exploited to reduce the

0021-9991/94 $6.00

computational effort required to evaluate F_1. Consequently, finding the cases where there exists a one-to-one (differentiable) substitution $x=\phi(u)$, $y=\psi(v)$, that converts (2.1) to (2.2), is of interest.

Assuming that such a substitution exists we can write

$$K(x, y)=H(u+v)=H(\phi^{-1}(x)+\psi^{-1}(y)). \quad (2.3)$$

Therefore, it is clear that such a substitution exists only if the original kernel is itself a function of *one* variable, which implies that

$$K(x, y)=K_1(a(x)+b(y)) \quad (2.4)$$

for some one-to-one differentiable functions $a(x)$ and $b(y)$. Moreover, a kernel, given as in (2.4), can be written as a sum kernel in more than one way. In fact, one may choose any constants α, β_1, β_2 and substitute

$$a(x)=\alpha u+\beta_1, \qquad b(y)=\alpha v+\beta_2 \quad (2.5)$$

to obtain

$$K(x, y)=K_1(\alpha(u+v)+\beta_1+\beta_2)=H(u+v). \quad (2.6)$$

Making appropriate choices for the constants α, β_1, β_2 plays an important role in the process and it is discussed in the following sections. Applying the substitution (2.5) to the transform (2.1) one obtains equation (2.2) with

$$F_1(u)=F(a^{-1}(\alpha u+\beta_1)) \quad (2.7)$$

and

$$\tilde{g}(v)=g(b^{-1}(\alpha v+\beta_2))\cdot\frac{d}{dv}[b^{-1}(\alpha v+\beta_2)] \quad (2.8)$$

and with new integration limits a_1, b_1, c_1, d_1.

The kernel $K(x, y)$ can appear in different forms while still being essentially a sum kernel, as illustrated by the following examples:

• *A product kernel.* Suppose that $K(x, y)=K(a(x)\cdot b(y))$ for some positive one-to-one differentiable functions $a(x)$ and $b(y)$. The kernel, K, can then be written as

$$K(x, y)=K(a(x)\cdot b(y))=K(e^{\ln(a(x))+\ln(b(y))}) \quad (2.9)$$

which is a sum kernel to which our change of variables may be applied. The Hankel transform ($K=yJ_0(2\pi xy)$, $a=0$, $b=\infty$) and the Laplace transform ($K=e^{-xy}$, $a=0$, $b=\infty$) are examples of such product kernels. The change of variables that converts a product kernel into a sum kernel has been applied to these cases before. Gardner *et al.* [8] initiated the use of such transformations to speed up the calculation of the Laplace transform, and Siegman [17] used this idea for the Hankel transform. From Eq. (2.9) it is clear why we require that $a(x)$ and $b(y)$ are to be positive. In cases where $a(x)$ (or $b(y)$) is negative, one may use its absolute value in Eq. (2.9) (regarding the negative sign as a coefficient). In the more general case where $a(x)$ changes sign, the transform can be represented as a sum over intervals where $a(x)$ has the same sign.

• *Exponential kernel.* Here K takes the form of $K(x, y)=K(a(x)^{b(y)})$ for some positive differentiable functions $a(x)$ and $b(y)$. Writing K as $K(x, y)=K(e^{b(y)\cdot\ln(a(x))})$ reduces it to the form of a product kernel and the results above may be applied. An example is the Mellin transform for which $K(x, y)=K(x^y)$, $a=0$, and $b=1$.

• *Degenerate kernel.* An integral transform with a degenerate kernel is the case where K is a finite sum of terms of the form $a(x)\cdot b(y)$ (e.g., powers of x and y). It can be written as a sum of product-kernel transforms, each of which can be treated separately as explained above.

3. IMPROVED PROCEDURES FOR CALCULATING CORRELATION INTEGRALS

Numerical computation of an integral transform starts with the discretization of a problem which, for a sum-kernel transform (2.2), yields a discrete correlation. Discrete correlations can be efficiently calculated (i.e., with reduced computational complexity) by means of the fast Fourier transform (FFT) (Davis and Rabinowitz [7], Brigham [3]). The idea of converting an integral transform into a sum-kernel form to use the FFT has been applied before, although only for special cases of product-kernel transforms. It was pioneered by Gardner [8] and has been applied since in several cases (e.g., Cohn-Sfetcu *et al.* [6], Siegman [17], Murphy and Bernabe [11], Cheng *et al.* [21]). The FFT is traditionally associated with using the rectangular quadrature rule, the exclusive technique applied to the resulting sum-kernel transforms in the above references.

When estimating the efficiency of the transformed approach, one should remember that any integration rule may be applied to the original problem (Eq. (2.1)). Consequently, transforming (2.1) to Eq. (2.2) and then applying *only* an inferior integration rule weakens the whole procedure. In extreme cases, it might make the transformation worthless *despite* the efficiency in computing (2.2). For this reason we suggest a variation on the traditional algorithm. Our modification enables the application of the same fast algorithms while retaining the flexibility to choose appropriate quadrature rules for approximating (2.2). This allows a comparison between the direct and the transformed approaches that depends only on the inherent properties of the integrands.

For reference, we consider a common computational procedure for the original problem. The intervals $[a, b]$ and $[c, d]$ are divided into subintervals $c = x_1 < x_2 < \cdots < x_{m-1} < x_m = d$ and $a = y_1 < y_2 < \cdots < y_{n-1} < y_n = b$. We define $F_i = F(x_i)$, $g_j = g(y_j)$, $K_{ij} = K(x_i, y_j)$. An integration rule applied to (2.1) yields

$$F_i = \sum_{j=1}^{n} K_{ij} g_j w_j, \qquad i = 1, ..., m. \tag{3.1}$$

Using this notation, $F(x)$ is evaluated on m points (x_i), where the weights and the abscissa coordinates used in the integration rule are w_j and y_j, respectively. In practical applications equally spaced (ES henceforth) grids on $[a, b]$ and $[c, d]$ are used. Moreover, for arbitrary grids, the choice of different integration rules affects both the weights w_j and the abscissas y_j used, while for integration rules applied to ES grids only the w_j are affected. Therefore, for simplicity, we restrict attention in this paper to ES meshes for the intervals $[a, b]$ and $[c, d]$. For example, letting h denote a fixed stepsize, the choice $w_j = h$ for all j yields the rectangular rule while the choice $w_1 = w_n = h/2$ with $w_j = h$ otherwise, yields the (generally more accurate) trapezoidal rule. We also mention that the midpoint rule ($w_j = h$ and $y_j = (((2j+1)/2)\, h)$ has accuracy equivalent to that of the trapezoidal rule. We point out that by requiring an ES grid we eliminate the option of using other rules like Gaussian quadrature rules. Indeed, these methods cannot be implemented straightforwardly with our approach, but require a further substitution and some interpolation.

The summation in (3.1) yields discretization errors that depend on the applied integration rule and on the number of discretization points (n). To achieve a desired accuracy with a certain rule n has to be chosen appropriately. Generally, higher order rules increase the accuracy of the computations, and one may apply them in order to satisfy the precision requirements with the smallest possible value of n. The number of points (m) at which $F(x)$ is approximated depends on the desired sampling resolution on $[c, d]$ (denoted by δ_1). For example, the grid resolution might be determined by the need to interpolate F with a certain accuracy. For many problems it is convenient to have identical grids on the x and the y domains (or at least $n = m$), especially if the transform is computed repeatedly. In these cases the number of sample points should be chosen to satisfy simultaneously the accuracy *and* the resolution requirements.

In this paper we measure the "cost" of the calculation in terms of required memory and the number of arithmetical operations (which is translated to the computation time). In order to compute Eq. (3.1), one should generate and store a look-up table K_{ij} consisting of nm elements ($\frac{1}{2}nm$ if the kernel is symmetric in its arguments (i.e., $k(x, y) = k(y, x)$) and the discretization is identical for x and y). The size of this table determines the storage requirements (and indirectly affects the calculation speed) and it is important to minimize its dimensions. The arithmetical operations associated with the direct approach are $(m-1)$ additions for each of the n values of F_i (ignoring the n multiplications, by the real coefficient w_j, needed to compute $w_j g_j$). As an overall estimate, both memory requirements and computational time for the direct method are proportional to nm.

To describe our more efficient algorithm we use notations analogous that used above, for the sample points on $[a_1, b_1]$ and $[c_1, d_1]$. With an equally spaced discretization and the same spacing for both the u and the v domains, $u_i + v_j$ is constant for choices of i and j such that $i + j$ is a fixed value. Consequently, $H_{ij} = H(u_i + v_j)$ can be denoted by a single subscript, namely H_{i+j}.

An integration rule applied to the transformed method (2.2) can be written as

$$F_i = \sum_{j=1}^{N} H_{i+j} \tilde{g}_j w_j, \qquad i = 1, ..., M, \tag{3.2}$$

where N is the number of discretization points (on $[a_1, b_1]$) needed to achieve the desired accuracy. Clearly, N depends on the intrinsic properties of the transformed integrand and on the applied integration rule. One should note at this point that the numbers n and N differ even when the same rule is used in Eqs. (3.1) and (3.2), since the integrand and the integration limits are changed. The number of points at which F_i is evaluated will be denoted by M, which may be different (generally greater) from m. To maintain the desired grid resolution (δ_1) after applying the nonlinear transformation (given by 2.5) the *maximal* spacing on $[c_1, d_1]$ should satisfy $\Delta u \leqslant (\delta_1/\alpha) \max(a')$. This consideration determines the parameters α, β_1, β_2, N, M as demonstrated in the examples. As before, in order to have the same grid for u and v (or $N = M$) the number of sample points should be chosen large enough to satisfy both the accuracy and the resolution requirements.

The look-up table for Eq. (3.2) consists of $N + M$ elements. Generating $\tilde{g}_j w_j$ requires N additional multiplications. The last N multiplications are "saved" if we restrict ourselves only to the rectangular quadrature rule, as has been done in previous work (e.g., Brigham [3], Siegman [3], Murphy and Bernabe [11], Agrawal and Lax [1], Cheng *et al.* [21]). The option to apply other integration rules was not considered and this is the point where our approach differs from the standard algorithm. One should note that application of the standard method has the implicit effect of increasing the value of N, due to the errors involved with the rectangular rule. The latter converges to the exact value with a rate proportional to $1/N$ while more accurate rules have faster convergence rates (for example, $1/N^2$ for the trapezoidal rule and $1/N^4$ for Simpson's rule. See Davis and Rabinowitz [7] for details). We assert that

the extra N multiplications result in an improvement that justifies the marginal additional complexity. Furthermore, in some integration rules only a fraction of these additional multiplications needs to be performed since most of the weights are unity. The trapezoidal rule, for example, requires only two multiplications. In Simpson's rule, the weights alternate: $w_{2j}=\frac{4}{3}$ and $w_{2j+1}=\frac{2}{3}$ for $j=1\cdots(N-1)/2$ and $w_1=w_N=\frac{1}{3}$, so only $N/2$ multiplications must be executed. Since Simpson's rule restricts N to be odd we prefer in our numerical experiments to use a modified version that works for any value of N. This is achieved with the weights: $w_1=w_N=\frac{17}{48}$, $w_2=w_{N-1}=\frac{59}{48}$, $w_3=w_{N-2}=\frac{43}{48}$, $w_4=w_{N-3}=\frac{49}{48}$ and otherwise $w_j=1$ (e.g., Press *et al.* [13]). Only eight multiplications are needed to generate $\tilde{g}_j w_j$ with these weights. It follows that the additional computational cost of N multiplications should be regarded only as an upper bound which is not attained in many practical cases. Also, flexibility in choosing the quadrature rules, enables one to use "open" and "semi-open" rules to skip over integrable discontinuities (Davis and Rabinowitz [7], Burden and Fairs [4]).

We shall now briefly describe an efficient method for calculating Eq. (3.2). First, by appending M zeros to the vector $\tilde{g}_j w_j$ and extending the summation to $N+M$, Eq. (3.2) can be regarded as a circular correlation and the FFT can be applied as follows: transform both of the ($N+M$ components) vectors, H_i and $\tilde{g}_j w_j$. Then multiply, term by term, $FFT(H_i)$ by the complex conjugate of $FFT(\tilde{g}_j w_j)$ and compute the inverse FFT of the result. The last M components of the resulting vector yield the desired result. It follows that Eq. 3.2 is obtained at the cost of three FFT of vectors with $N+M$ components, $N+M$ (complex) multiplications, and (at most) N additional multiplications.

The actual number of arithmetical operations depends on the FFT algorithm that is used. The computational complexity of an FFT algorithm applied to a p-dimensional vector depends on the factorization of p and is, roughly, proportional to $p\log_2(p)$. The FFT is most efficient when p is a power of two [Brigham [3]]. However, efficient algorithms exist for other values of p (see Brigham [3], Oppenheim and Schafer [14], Temperton [19, 20]).

Our method requires storage for two complex vectors of $N+M$ elements (i.e., $4(N+M)$ real numbers). Therefore, assuming the look-up table K_{ij} consists of real numbers the ratio $4(N+M)/nm$ ($8N/n^2$ for the case of identical grids) relates the storage requirements of the direct to that for the transformed approach. Consequently, reduction in storage requirements can be expected even when $N>n$ and $M>m$.

The overall number of operations for the transformed method is proportional to $(N+M)\log_2(N+M)$. Recall that the number of operations required for the direct approach (3.1) is proportional to nm. Thus, one can expect time reduction even if $N>n$ and $M>m$.

In our problem, the minimal value of $N+M$ is a function of the new integrand, the desired grid-density on $[c_1, d_1]$, and the integration rules. Sometimes it is even worthwhile to deliberately increase the number of sample points (compared with the minimal required) in order to work with vectors whose length is highly composite and use an efficient FFT. We also remark that the circular correlation can be computed by efficient methods without using an FFT (e.g., by means of the Cook–Toom algorithm, or the Chinese remainder theorem (Rao [15])).

Finally, routines that implement fast algorithms to compute the correlation (or convolution) of two input series are standard features in numerical packages (e.g., *Numerical Recipes* [13]), libraries (e.g., the IMSL Library), and computer-algebra packages (e.g., Matlab, Maple, Mathematica). The modification we suggest here differs from the standard approach only by the introduction of the extra multiplications by the weights w_j. Moreover, incorporating the variant described above into existing routines is simple and does not require changes in the library codes since it can be implemented externally at the stage where the input vector $\tilde{g}_j w_j$ is generated.

4. EXAMPLE 1

We use the integral transform

$$F(x)=\int_0^1 \sin(\lambda(x^2+y^2))\, g(y)\, dy, \qquad x\in[0,1], \tag{4.1}$$

to demonstrate how converting the kernel to a correlation form may introduce singularities with the result that N becomes larger than n. Application of accurate integration rules to the transformed problem becomes essential if it is to remain more efficient than the direct approach. We want F to be computed on a grid with resolution $\delta_1=0.01$, with accuracy of $\varepsilon=0.01$, and choose $g(y)\equiv 1$ and $\lambda=100$.

The number of grid points (n) needed to achieve the desired accuracy depends on the applied integration rule. Here we compare the rectangular, the trapezoidal, and Simpson's rules. The errors associated with these rules are E_R, E_T, E_S, respectively,

$$E_R=\frac{h}{2}f'(\theta_1) \tag{4.2}$$

$$E_T=\frac{h^2}{12}f''(\theta_2) \tag{4.3}$$

$$E_S=\frac{h^4}{180}f^{IV}(\theta_3), \tag{4.4}$$

where h denotes the step size, f denotes the integrand, and $\theta_1, \theta_2, \theta_3\in[0,1]$ are points on the integration interval.

We evaluate the maximal values of the first, the second, and the fourth derivatives (with respect to y) of the kernel in Eq. (4.1) (over $x \in [0, 1]$). The minimal number of sample points that ensures accuracy of ε can then be calculated. For the rectangular, trapezoidal, and Simpson's rules we obtain

$$n_R = 10000, \qquad n_T = 578, \qquad n_S = 174, \tag{4.5}$$

respectively. Clearly, among these three, Simpson's rule is the obvious choice.

Applying the substitution

$$x^2 = \alpha u + \beta_1, \qquad y^2 = \alpha v + \beta_2 \tag{4.6}$$

to Eq. (4.1), yields the correlation integral

$$F(\sqrt{\alpha u + \beta_1}) = \int_{a_1}^{b_1} \sin(\lambda(\alpha(u+v) + \beta_1 + \beta_2)) \times \frac{\alpha}{2\sqrt{\alpha v + \beta_2}}\, dv, \tag{4.7}$$

where

$$a_1 = \frac{-\beta_2}{\alpha}, \qquad b_1 = \frac{1-\beta_2}{\alpha}. \tag{4.8}$$

Note that the new kernel is singular at $v = -\beta_2/\alpha$. The singularity is skipped by integrating Eq. (4.7) over the interval $[0, b_1]$. The contribution over $[-\beta_2/\alpha, 0]$ (which is equivalent to integrating the original kernel over $[0, \sqrt{\beta_2}]$) is replaced with an approximation given by the first two terms of the Taylor series:

$$\int_0^{\sqrt{\beta_2}} \sin(\lambda(x^2 + y^2))\, dy \sim \sqrt{\beta_2}\sin(\lambda x^2) + \frac{\lambda \beta_2^{1.5}}{3}\cos(\lambda x^2). \tag{4.9}$$

A simple calculation shows that this approximation introduces the error (E_1) that satisfies

$$E_1 \leqslant 0.1\lambda^2 \beta_2^{2.5}. \tag{4.10}$$

Thus, the overall error is the sum of E_1 and the error from the integration rule. We choose to satisfy $E_1 \leqslant 0.005$ (allowing an error of 0.005 in the integration). Rounding $\beta_2 = 0.007$ and setting $b_1 = 1$ yields $\alpha = 0.993$. We choose $\beta_1 = \beta_2$ to have the same range over both the u and the v domains.

Using Eqs. (4.2), (4.3), (4.4), it follows that the minimal value of N which ensures accuracy of 0.005 is

$$N_R = 66000, \qquad N_T = 1155, \qquad N_S = 359. \tag{4.11}$$

for the rectangular, trapezoidal, and Simpson's rules, respectively.

Obviously, the rectangular rule is impractical here. Since it requires so many sample points the whole transformation is worthless, compared with the effort involved with Simpson's rule applied directly. However, with our approach, we are free to apply the same higher order rules that may be applied to the original problem, to the transformed problem. Consequently, we may assume $N = N_S$.

From Eq. (4.6) we have $2x\, dx = \alpha\, du$; thus, the sampling is "diluted" as x approaches its lower limit ($\sqrt{\beta_2}$). In order to maintain the same resolution obtained with m equally spaced grid points on $[a, b]$, compared with the direct method, one has to satisfy $M > \alpha m/2\sqrt{\beta_2}$. With the values chosen for α and β_1 this implies $M \sim 6m$. This oversampling is an inevitable outcome of the nonlinear transformation (4.6) and the singularity associated with its inverse.

However, despite the increase in both n and m, the transformed method is still efficient. The direct method, with $n = 170$ and $m = 100$, requires $\sim$34 Kb storage (assuming four bytes per floating point number) while the transformed method, with $N + M = 1024$ (we deliberately "round up" $N + M$ in order to apply a power of two FFT algorithm) requires only about 16.4 Kb storage, which is a reduction by a factor of about two. The measured computation time with the transformed method was around 20% less than with the direct technique, both computed on a 386 personal computer.

We finally remark that the advantage of the transformed approach becomes more noticeable as λ increases. The direct method becomes prohibitively expensive, whereas the transformed method (with storage requirements and computational time growing at a much slower rate) remains feasible.

5. EXAMPLE 2

In this example we apply our method to the Hankel transform (HT), frequently used in many applications such as optics (elements with radial symmetry), two-dimensional image processing (Goodman [9]), and numerical solution of partial differential equations (Le Bail [2]). Since the straightforward calculation of the HT involves a heavy computational burden and high storage requirements, considerable effort has been directed towards developing fast and efficient HT algorithms. Siegman [17] introduced an algorithm for a fast Hankel transform (FHT), based on Gardner's transformation. A subsequent paper by Agrawal and Lax [1] suggested further improvements on Siegman's

FHT algorithm. Other fast HT algorithms were introduced by Talman [18], Candel [5] and by Murphy and Gallagher [12]. We will show here that the exponential transformation required for the FHT yields an inevitable increase in N and M, compared with n and m. The flexibility to apply any integration rule to the transformed problem, as it is with our approach, yields a faster and more accurate HT.

The HT of order l is defined by

$$F(x) = 2\pi \int_0^\infty J_l(2\pi xy)\, yg(y)\, dy, \qquad x \in [0, \infty). \tag{5.1}$$

Following Eq. (2.9), we use the transformation

$$x = e^{\alpha u + \beta_1}, \qquad y = e^{\alpha v + \beta_2} \tag{5.2}$$

to convert Eq. (5.1) into the form

$$F(e^{\alpha u + \beta_1}) = 2\pi \int_{-\infty}^{\infty} J_l(2\pi e^{\alpha(u+v)+\beta_1+\beta_2})\, \alpha e^{2\alpha v + 2\beta_2} \times g(e^{\alpha v + \beta_2})\, dv, \qquad u \in (-\infty, \infty). \tag{5.3}$$

We may now use Eq. (3.2) as a discrete approximation of Eq. (5.3), with $u_i = i - 1$ and $v_j = j - 1$ for $i = 1, ..., M$, $j = 1, ..., N$, and the appropriate expressions for H and $\tilde{g}$.

The choice of $u \in [0, M-1]$ and $v \in [0, N-1]$ in Eq. (5.3) is equivalent to $x \in [e^{\beta_1}, e^{\alpha(M-1)+\beta_1}]$ and $y \in [e^{\beta_2}, e^{\alpha(N-1)+\beta_2}]$ in Eq. (5.1). The values $x = 0$ and $y = 0$ are not attained with the exponential transformation (5.2), to which we refer as an unavoidable "unsampled interval." However, one can make this unsampled interval arbitrarily small by choosing β_1 and β_2 appropriately.

In practice, we approximate Eq. (5.1) to a finite domain $x \in [0, L_1]$ and truncate the infinite integration limits to $y \in [0, L_2]$. L_1 should be chosen large enough to assure that the truncated integral represents a good approximation of (5.1) (for example, one criterion can be enabling an accurate inversion of the HT). In many cases we have $g(y) = 0$ for $y \geqslant L_2$ thus the truncation does not introduce any error at all (e.g., when L_2 is the maximal aperture involved with an optical problem). L_2 represents the largest value of x for which $F(x)$ is required. We also choose values for the sampling resolution (δ_1), the maximal length of the unsampled interval (γ_1) on the x domain, and the desired accuracy in the computations (ε). This determines the values of α, β_1, β_2, N, M used in the algorithm.

The choice for L_1 and γ_1 gives

$$e^{\beta_1} = \gamma_1, \qquad e^{\alpha(M-1)+\beta_1} = L_1. \tag{5.4}$$

The nonlinear transformation given in Eq. (5.2) yields unequally spaced samples of x (and y). Thus, in order to maintain the required sampling resolution, we must ensure that the maximal distance between two adjacent sample points does not exceed δ_1. The spacing is maximal for $i = M$ and we can approximate $\Delta x_{\max} \sim \alpha L_1$ (using Eq. (5.4)). Therefore, it is enough to satisfy

$$\alpha L_1 = \delta_1. \tag{5.5}$$

Equations (5.4) and (5.5) determine α, β_1, and M by means of some simple calculations. Having fixed the value of α, we can determine β_2 and N by an expression analogous to Eqs. (5.4), namely,

$$e^{\beta_2} = \gamma_2, \qquad e^{\alpha(N-1)+\beta_2} = L_2. \tag{5.6}$$

Here γ_2 is the unsampled interval in the y domain and should be set as part of the accuracy requirements, as we shall describe later. Obviously, the order of these calculations may be reversed to determine α, β_2, and N first, followed by β_1 and M. However, after setting γ_1, γ_2, L_1, L_2 we can determine the resolution on *only* one of the domains (i.e., δ_1 *or* δ_2), which then implies the values of *both* N and M.

The parameters γ_2 and N are directly related to the accuracy of the calculations. Since the interval $[0, \gamma_2]$ is not sampled, an error (which depends on γ_2) is introduced. The integral over $[0, \gamma_2]$ (denoted $I(x)$) can be approximated by means of its Taylor expansion (see Agrawal and Lax [1]). Assuming that g varies slowly near 0 (and for $l = 0$) one obtains

$$I(x) \sim \pi g(\gamma_2)(\gamma_2^2 - \tfrac{1}{2}\pi\gamma_2^4 x^2) \tag{5.7}$$

with an error of the order of $O(\gamma_2^6)$.

The minimal required number of sample points, N, can be estimated by computing upper bounds for the integrand's derivatives and using them in the expressions for the error corresponding to the applied integration rule.

Often, in practice, the Hankel transform is computed repeatedly and it is convenient to have $N = M$. Consequently, N should be chosen large enough to satisfy both sampling resolution and accuracy requirements. Using the rectangular (weak) integration rule causes an increase in N, compared with better rules, and makes the procedure less effective. To illustrate, we compare computations with our approach with the example presented by Siegman [17]. There, the resolution was set so that the most oscillatory component of the kernel ($J_0(L_1 L_2)$) is sampled k_1 times and the length of the unsampled interval does not exceed $1/k_2$ of the fastest oscillation. The grids on the x and the y domains were identical. It was assumed implicitly that a fairly fine mesh on x ensures "good" accuracy in the calculations, although the actual required accuracy was not mentioned.

TABLE I

Accuracy Achieved with the Direct Method and the Transformed Method with Different Integration Rules

Method	Rectangular rule	Trapezoidal rule	Simpson's rule
Direct ($n=60$)	10^{-1}	10^{-3}	10^{-5}
Transformed ($N=256$)	10^{-1}	10^{-3}	10^{-6}

This is not necessarily true, especially since the rectangular integration rule was used.

The transformed function in Siegman's example was $g(y)=1$ for $0 \leqslant y \leqslant L_2$ and zero otherwise. The choice $L_1=L_2=3.938$, $k_1=k_2=4$, and $\delta_1=\delta_2=0.06349$ yields $\alpha=0.01612$ and $N=256$. Table I presents the typical order of magnitude of the errors involved with the rectangular, the trapezoidal, and Simpson's rules when applied to the problem. In our calculations we included the approximation 5.7 for $I(x)$ (see Agrawal and Lax [1]), which is more accurate than simply neglecting this term (as in Siegman's original paper). The results were compared with the exact values obtained analytically.

One should note that the direct method with $n=60$ requires roughly the same storage ($\sim$7.2 Kb) as the transformed method with $N=256$ ($\sim$8.2 Kb). Simpson's rule, applied to both methods, results in a precision of 10^{-6}. We found that to obtain this precision with the rectangular rule, N should be increased beyond the limit where it still pays to use the transformed method. Moreover, numerical experiments show that high accuracy cannot be achieved with the rectangular rule (probably due to roundoff errors). Nevertheless, higher order integration rules applied to both methods result in errors of comparable order. This proved to be true for other cases that were investigated (e.g., $g(x)=e^{-\pi x^2}$). It is clear that the flexibility to apply accurate integration rules to the transformed method is essential.

We conclude with estimating the dependence of N on n for the case were $L_1=L_2$ and $\delta_1=\delta_2$. Suppose that the target accuracy can be achieved with n equally spaced samples when using the direct method and spacing δ_1. We also add the requirement $\gamma_1=\gamma_2=\frac{1}{4}\delta_1$. It can be shown that ensuring the same accuracy with the transformed method requires $N=n\ln(4n)$ samples. The ratio comparing the storage requirements of the two methods is therefore $16\ln(4n)/n$. The computation time is machine dependent but the improvement is clear (verified numerically on a 386 personal computer). We can easily conclude that the transformed method becomes more advantageous when the problem is more difficult.

ACKNOWLEDGMENTS

The research was supported by the Center for Applied Mathematics and the Theory Center at Cornell University, and by a Rothschild Fellowship. The careful reading and the suggestions of an anonymous referee were very valuable. I also thank John Guckenheimer and Mark Myers for their helpful comments and assistance in editing the manuscript.

REFERENCES

1. G. P. Agrawal and M. Lax, *Opt. Lett.* **6**, 171 (1981).
2. R. C. Le Bail, *J. Comput. Phys.* **9**, 440 (1972).
3. O. E. Brigham, *The Fast Fourier Transform* (Prentice-Hall, Englewood Cliffs, NJ, 1974).
4. R. L. Burden and J. D. Faires, *Numerical Analysis* (PWS, Boston, 1985).
5. S. M. Candel, *Comput. Phys. Commun.* **23**, 343 (1981).
6. S. Cohn-Sfetcu, M. R. Smith, and S. T. Nichols, *Proc. IEEE* **63**, 326 (1975).
7. P. Davis and D. Rabinowitz, *Numerical Integration* (Blaisdell, Boston, 1967).
8. D. G. Gardner, J. C. Gardner, and G. Laush, *J. Chem. Phys.* **31**, 978 (1959).
9. J. W. Goodman, *Introduction to Fourier Optics* (McGraw-Hill, New York, 1968).
10. W. P. Lathman and T. C. Salvi, *Opt. Lett.* **5**, 219 (1980).
11. W. D. Murphy and M. L. Bernabe, *Appl. Opt.* **17**, 2358 (1978).
12. P. K. Murphy and N. C. Callagher, *J. Opt. Soc. Am.* **73**, 1130 (1983).
13. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge Univ. Press, Cambridge, 1986).
14. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing* (Prentice Hall, Englewood Cliffs, NJ, 1975).
15. E. Rao (1982), *Fast Transforms* (Academic Press, New York, 1982).
16. S.-C. Sheng and A. E. Siegman, *Phys. Ref.* **21**, 599 (1980).
17. A. E. Siegman, *Opt. Lett.* **1**, 13 (1977).
18. J. D. Talman, *J. Comput. Phys.* **29**, 35 (1978).
19. C. Temperton, *J. Comput. Phys.* **52**, 1 (1983).
20. C. Temperton, *J. Comput. Phys.* **58**, 283 (1985).
21. Z. Cheng, H. J. J. Seguin, S. K. Nikumb, A. V. Seguin, and H. Reshef, *Appl. Opt.* **27**, 836 (1988).