MATHEMATICS OF COMPUTATION
VOLUME 39, NUMBER 160
OCTOBER 1982, PAGES 535-548

Evaluation of Fourier Integrals Using B-Splines

By M. Lax and G. P. Agrawal

Abstract. Finite Fourier integrals of functions possessing jumps in value, in the first or in the second derivative, are shown to be evaluated more efficiently, and more accurately, using a continuous Fourier transform (CFT) method than the discrete transform method used by the fast Fourier transform (FFT) algorithm. A B-spline fit is made to the input function, and the Fourier transform of the set of B-splines is performed analytically for a possibly nonuniform mesh. Several applications of the CFT method are made to compare its performance with the FFT method. The use of a 256-point FFT yields errors of order 10^{-2}, whereas the same information used by the CFT algorithm yields errors of order 10^{-7}—the machine accuracy available in single precision. Comparable accuracy is obtainable from the FFT over the limited original domain if more than 20,000 points are used.

I. Introduction. An accurate fast evaluation of the Fourier integral is of considerable interest because of the application of the Fourier transform techniques to a wide variety of problems. In the physics literature, applications have been made to the solution of nonlinear partial differential equations, and it is customary to evaluate the Fourier integral using some version of the fast Fourier transform (FFT) algorithm [10]–[12], [15], [22], [25]. The FFT procedure is applied with "guard bands" (extra points) even when sharp edges, such as mirror edges, produce discontinuities in the function. The purpose of this paper is to demonstrate that if the function or one of its derivatives is discontinuous, reasonable accuracy can be obtained using the FFT only if an extraordinary number of sample points are used. This point will be demonstrated by considering several examples and reporting (a) exact results, (b) FFT results and (c) CFT results, both as to accuracy and as to CPU time. The CFT (continuous Fourier transform) algorithm is an algorithm we have applied to the calculation of electromagnetic fields in a three-dimensional loaded (nonlinear) laser resonator with sharp mirror edges [14]. This algorithm is based on making a B-spline fit to the original function with the help of PORT, the Bell Laboratories Mathematical Subroutine Library. An analytic integration of the Fourier transform of the B-splines for arbitrary nonuniform spacing permits the construction of an algorithm that takes advantage of uniform interior spacing (permitting the internal use of FFT on the spline coefficients) yet handles end regions that require nonuniformly spaced mesh points correctly.

II. Historical Perspective. Although the primary aim of this paper is to indicate and overcome the limitations of the FFT, we shall comment briefly here on other

---

Received December 10, 1979; revised May 7, 1981 and January 29, 1982.
1980 *Mathematics Subject Classification.* Primary 42A38; Secondary 65D30, 41A15.
Key words and phrases. Fourier integral, B-splines, fast Fourier transform, continuous Fourier transform.

0025-5718/82/0000-0333/$03.25

algorithms that may be used to accomplish a similar purpose. We have not made numerical comparisons because in most cases the algorithms are not specifically suited to the solution of partial differential equations. The latter problem supplies its input in the form of an array of values at regularly spaced points, and values at other points are, in general, unavailable. Moreover, the input is not analytic so that derivatives are not directly available.

Elementary approaches, such as a trapezoidal [1] FFT or an algorithm such as Filon's [9] based on a parabolic fit, suffer from the presence of kinks (jumps in slope) in the fitting function. Such kinks, by Tauberian arguments, generate incorrect high frequency tails in the Fourier transform. A higher order procedure, such as the Chebyshev fitting procedure of Piessens and Branders [18], has the advantage of being adaptive and automatic, although it does not completely avoid the kink problem. The Piessens-Branders automatic Chebyshev algorithm, moreover, is appropriate when the function $f(x)$, whose Fourier integral is desired, is available analytically, or at least numerically at points chosen by the program rather than specified in advance. Similar restrictions also apply to the Piessens-Haegemans Gaussian quadrature procedure [19].

Numerical evaluation of Fourier integrals with the help of cubic splines was proposed by Einarsson [7], [8] and by Silliman [23]. These methods can be combined with Richardson's extrapolation, and the FFT may be used to sum the resulting series. This procedure, in common with ours, avoids kinks. The only disadvantage of the Einarsson-Silliman procedure is that it requires a knowledge of the second derivative at the end points. Although these derivatives can be computed from a table of values $f(x_j)$, we found it more convenient to use the PORT library to make a spline fit to the sample values $f(x_j)$:

1. The PORT spline program uses the first and last *interior* points to provide information equivalent to derivative information at the boundaries.

2. The order k of the spline fit is not restricted to $k = 4$.

3. Discontinuities at the end points are handled easily using multiple mesh points at the ends.

The package of spline programs due to de Boor [6] could equally well have been used since the PORT subroutines are based on de Boor's original algorithms [5].

Most directly related to our work is Marti's algorithm [17] for recursively computing the Fourier coefficients of B-splines with nonequidistant knots. Our paper differs from his in having supplied (1) an analytic formula for the B-spline Fourier coefficients, (2) an algorithm for combining these coefficients to obtain a Fourier integral, (3) a simplification of the Fourier coefficients for the uniformly spaced case that permits internal use of the FFT and (4) explicit determination of end corrections to supplement (3) above since the end mesh points will be multiple points. Marti's procedure does not have end corrections because he does not take advantage of uniformity in the interior, but treats all cases by a general formula. This is logically simpler, but produces a slower code that takes no advantage of the FFT. We have not proposed a specific algorithm for evaluating our analytic expression for the general nonequidistant case, and Marti's algorithm is probably excellent for this case.

Marsden and Taylor [16] derive a quadrature formula for a Fourier integral whose form is similar to the Euler-Maclaurin relation between sums and integrals with end corrections involving derivatives of $f(x)$ at the end points. Splines are involved only indirectly in that the coefficients are chosen to make the error vanish if $f(x)$ is a spline of degree k. In principle, their procedure is equivalent to an integration over the B-spline fit to the integral, and their results should agree with ours. In practice, however, the conditions they impose to determine the integral do not uniquely specify the interpolating spline for $k > 3$. Moreover, their algorithm involves the evaluation of derivatives at the end points. For our purposes, the derivatives must be replaced by appropriate finite difference formulas, and the algorithm for doing so is not specified by them. Thus their algorithm and ours should be exact for polynomials of degree k, but the errors will be different. The Marsden-Taylor work is quite interesting and deserves further analysis. Since we were made aware of this work by one of the referees and an explicit code is not available, we shall not attempt at this time to make a numerical comparison between their algorithm and ours.

III. B-Spline Fit. Consider the Fourier integral over the finite domain $[a, b]$

$$g(\mu) = \int_a^b f(x)e^{i\mu x}\,dx, \tag{1}$$

where the input function $f(x)$ may have a discontinuity in itself or in one of its derivatives at certain points in its domain. Consider a nonuniform mesh $x_j, j = 1$ to N; $a = x_1 \leqslant \cdots \leqslant x_N = b$. Two neighboring mesh points may coincide. We assume that $f(x)$ has an expansion in terms of basis splines (B-splines) [5], [6], [20], [21]

$$f(x) = \sum_{j=1}^{N-k} a_j B_{j,k}(x), \tag{2}$$

where the kth order B-spline $B_{j,k}(x)$ is a polynomial of degree $(k-1)$ in the nonempty interval (x_j, x_{j+k}) and can be obtained from the recurrence relation [5], [6]

$$B_{j,k}(x) = \frac{x - x_j}{x_{j+k-1} - x_j} B_{j,k-1}(x) + \frac{x_{j+k} - x}{x_{j+k} - x_{j+1}} B_{j+1,k-1}(x), \tag{3}$$

where $x_j \leqslant x \leqslant x_{j+k}$ and $B_{j,1}(x) = 1$ if $x_j \leqslant x \leqslant x_{j+1}$ and zero otherwise. It follows from (3) that $0 \leqslant B_{j,k}(x) \leqslant 1$; $B_{j,k}(x)$ is zero outside the interval $[x_j, x_{j+k}]$ and possesses only one maximum inside it.

To represent a function $f(x)$ with a discontinuity in its value (or its sth derivative) at x_i, we must choose the multiplicity m_i of the mesh point x_i to be k (or $k - s$). The expansion coefficients a_j in (2) are obtained by a least-squares fitting procedure.

IV. Fourier Integral. On substituting (2) in (1) we obtain

$$g(\mu) = \sum_{j=1}^{N-k} a_j \int_{x_j}^{x_{j+k}} e^{i\mu x} B_{j,k}(x)\,dx, \tag{4}$$

where we have used the property that $B_{j,k}(x) = 0$ outside the interval $[x_j, x_{j+k}]$. The integral in (4) is evaluated using the identity [21]

$$[x_j x_{j+1} \cdots x_{j+k}] F(x) = \frac{1}{(k-1)!\,(x_{j+k} - x_j)} \int_{x_j}^{x_{j+k}} F^{(k)}(x) B_{j,k}(x)\,dx,$$

which can be obtained using Peano's theorem (for example, see Section 3.7 of [4]). Here $[x_j x_{j+1} \cdots x_{j+k}]F(x)$ and $F^{(k)}(x)$ are, respectively, the kth divided difference and the kth derivative of the function $F(x)$. By choosing $F(x) = \exp(i\mu x)$ and using the result in (4), we obtain

$$g(\mu) = \sum_{j=1}^{N-k} a_j e^{i\mu x_j} \Phi_{j,k}(\mu), \tag{5}$$

where

$$\Phi_{j,k}(\mu) = \frac{(k-1)!\,(x_{j+k} - x_j)}{(i\mu)^k \exp(i\mu x_j)} \left[x_j x_{j+1} \cdots x_{j+k}\right] e^{i\mu x} \tag{6a}$$

$$= \frac{(k-1)!}{(i\mu)^k} h_{j+k} \left[0\, h_{j+1} \cdots h_{j+k}\right], \tag{6b}$$

and $h_{j+i} = (x_{j+1} - x_i)$. In the following and in (6b) it is implicitly assumed that all divided differences refer to the exponential function $e^{i\mu x}$.

The apparent similarity of (5) with the discrete Fourier transform is striking. The expansion coefficient a_j, obtained through a B-spline fit, is multiplied by a μ- and k-dependent correction factor $\Phi_{j,k}(\mu)$, and the product is to be used in place of the function value $f(x_j)$. In effect, the FFT sum acquires a separate "window factor" at each mesh point.

Equation (5) is derived for a mesh of arbitrary spacing. However, the case of a uniform mesh is of practical importance: The window factors except for end corrections become uniform; the sum, in (5) acquires the form of a discrete Fourier transform, and FFT algorithms can be used to perform this sum efficiently. To take advantage of the simplifications of a uniform mesh, it is convenient to break up the Fourier integral, (1), into regions connecting points of discontinuity. In this way, code corrections associated with nonuniform spacing of knots need only be applied as end corrections. The end points are chosen to have multiplicity k. A uniform mesh of N points in the interval $[a, b]$ with end-point multiplicity k is given by

$$\begin{gathered} x_1 = x_2 = \cdots = x_k = a, \quad x_{N-k+1} = \cdots = x_{N-1} = x_N = b, \\ (x_{j+1} - x_j) = h, \qquad j = k, k+1, \ldots, N-k. \end{gathered} \tag{7}$$

The evaluation of the correction factor $\Phi_{j,k}(\mu)$ is readily carried out using (6) and we find

$$\Phi_{j,k}(\mu) \equiv \Phi(\mu) = h\left(\frac{e^{i\mu h} - 1}{i\mu h}\right)^k, \qquad j = k \text{ to } (N + 1 - 2k). \tag{8}$$

At the left-hand end point $\Phi_{j,k}$ for $j = 1$ to $(k-1)$ and at the right-hand end point $\Phi_{j,k}$ for $j = N - 2(k-1)$ to $(N-k)$ are to be evaluated separately because in evaluating the divided difference in (6) one or more points coincide. The procedure is however straightforward, and we give the details in the Appendix. It should be remarked that evaluation of these end corrections requires a number of steps proportional to the spline order k independent of the total number of mesh points N (for the case of a uniform mesh) for each μ.

We have developed a continuous Fourier transform (CFT) algorithm based on (5) with proper end corrections. The FFT is used to sum the series in (5). For each μ the

correction factor $\Phi_{j,k}(\mu)$ is applied to obtain the Fourier integral $g(\mu)$. In the next section we compare the performance of the CFT and the FFT for three test problems.

V. Applications. As we have mentioned in the introduction, the FFT does not yield accurate results, unless extraordinarily large values of N are used, for an input function with discontinuous behavior or with rapid oscillations. In this section we illustrate the performance of the CFT and compare it with the FFT for three test functions. All calculations are done on a DEC-10 machine in single precision.

Case 1—*Square Pulse*. The input function $f(x)$ is assumed to be

$$f(x) = \begin{cases} 1 & \text{if } |x| \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

From (1) the Fourier transform of $f(x)$ is readily obtained,

$$g(\mu) = (2 \sin \mu)/\mu. \tag{10}$$

In using the FFT, $f(x)$ is supplied on the interval $[-8, 8]$ to provide the necessary "guard band", and the resulting $g(\mu)$ is compared with exact values given by (10). The calculations are done with $N_F = 64$, 256, and 1024 FFT points. In general the

TABLE 1
Comparison of CFT and FFT for a square pulse

μ	FFT		CFT	EXACT
	N_F	$g(\mu)$	N=10	
$\pi/8$	64	1.9474251		
	256	1.9488929	1.9489906	1.9489907
	1024	1.9489846		
$7\pi/8$	64	0.2673821		
	256	0.2777420	0.2784273	0.2784273
	1024	0.2783844		
$17\pi/8$	64	0.0086711		
	256	0.1129784	0.1146465	0.1146465
	1024	0.1145425		
$15\pi/4$	64	-0.0174109		
	256	-0.1145691	-0.1200422	-0.1200422
	1024	-0.1197030		

accuracy is poor at high frequencies. While using CFT, a uniform mesh with multiplicity $k = 3$ (spline-order) at the end points (-1 and 1) (which are also the points of discontinuity) is chosen. Since $f(x)$ is constant inside $[-1, 1]$, only 10 mesh points (including multiplicities) are chosen. In Table 1 we have compared the CFT and FFT results at some frequencies. It is evident from Table 1 that the CFT with 10 points gives considerably more accurate results than the FFT method with 1024 points.

Case 2—Exponential Function $e^{-p|x|}$. This case is interesting because, unlike Case 1, the input function $f(x)$ is continuous at all points with a kink at the origin, i.e., its first derivative has a jump at $x = 0$. Substituting $f(x) = \exp[-p\,|\,x\,|]$ in (1) and taking $a = -b$, we obtain

$$g(\mu) = \frac{2p}{p^2 + \mu^2} + \frac{2e^{-pb}}{p^2 + \mu^2}(\mu \sin \mu b - p \cos \mu b). \tag{11}$$

For the calculations we chose $p = 1$, $b = 16$. In using the FFT, $f(x)$ is supplied inside the interval $[-16, 16]$ with $N_F = 256$. For the CFT the integral was performed only over the range [0, 16]. A mesh point with multiplicity k must be assigned at the point of discontinuity $x = 0$. Since $f(x) = \exp[-p\,|\,x\,|]$ is an even function of x, the value of the Fourier integral $g(\mu)$ is obtained by doubling the numerical value. By choosing the number of sample points $N_s = 128$, $f(x)$ in the CFT is supplied at the same points as in the FFT. This ensures that the information about $f(x)$ is given to the same extent to both programs. The CFT calculations were done with $N = 107$ mesh points and $k = 4$ ("cubic splines"). In Table 2 we compare the CFT and FFT outputs with the exact values of $g(\mu)$ obtained from (11). It is evident that the CFT is superior to the FFT. At $\mu = 2\pi$ the FFT yields only one significant digit, while the CFT is accurate up to six significant digits. It is to be noted that at $\mu = 4\pi$ the relative error in the FFT value is 24%, and the situation gets worse at higher values of μ.

TABLE 2

Comparison of CFT and FFT for an exponential function $e^{-|x|}$

μ	FFT		CFT	EXACT
	N_F=256	N_F=1024	N=107	
0	2.0026033	2.0001625	1.9999999	1.9999998
$\pi/2$	0.5794093	0.5769636	0.5768016	0.5768006
π	0.1866230	0.1841621	0.1839995	0.1839993
2π	0.0520947	0.0495721	0.0494090	0.0494090
4π	0.0155441	0.0127495	0.0125852	0.0125854
7π	0.0081098	0.0042937	0.0041270	0.0041270

In order to see how the FFT performance improves as N_F increases, we made a 4-fold increase in the number of points, $N_F = 1024$, and the corresponding values of $g(\mu)$ are also shown in Table 2. We find that only one significant digit is gained by increasing N_F from 256 to 1024. The convergence of the FFT values to the exact values appears to be slow.

Case 3—*Truncated Cornu Spiral.* This example is chosen to illustrate the usefulness of our CFT algorithm in unstable resonator problems of optics [14]. The finite size of the output mirror produces sharp discontinuities in the field distribution at the position of the mirror edges whenever the optical field is reflected at the output mirror. If we assume that initially the field distribution is uniform, on first reflection the field distribution $f(x)$ is zero outside the mirror dimensions. If we consider only one transverse dimension, $f(x)$ is given by (9), and we have compared the performance of the CFT and the FFT in Case 1. On second reflection at the output mirror the optical field $f(x)$ is a truncated Cornu spiral. The following analytic expressions for the Fourier transform pair $f(x)$ and $g(\mu)$ are obtained [3] after solving the paraxial wave equation:

$$f(x) = \frac{1}{\sqrt{i}}\left[F\left(\frac{1+x}{2p}\right) + F\left(\frac{1-x}{2p}\right)\right], \tag{12}$$

for $|x| < 1$ and zero otherwise, and

$$g(\mu) = \frac{1}{\sqrt{2i}}\left\{\frac{2\sin\mu}{\mu}F\left(\frac{1}{p}\right) + \frac{e^{-ip^2\mu^2}}{2i\mu}\left[2F(p\mu)\cos\mu + e^{i\mu}F\left(\frac{1-\mu p^2}{p}\right) - e^{-i\mu}F\left(\frac{1+\mu p^2}{p}\right)\right]\right\}, \tag{13}$$

where $F(t) = C_1(t) + iS_1(t)$ is the complex Fresnel integral [2]

$$F(t) = \sqrt{\frac{2}{\pi}}\int_0^t e^{iu^2}\,du. \tag{14}$$

In the numerical results reported below the parameter p was chosen to be 0.158114.

TABLE 3
Comparison of CFT and FFT for a truncated Cornu spiral

μ	FFT		CFT		EXACT	
	Re $g(\mu)$	Im $g(\mu)$	Re $g(\mu)$	Im $g(\mu)$	Re $g(\mu)$	Im $g(\mu)$
0	1.8735905	-0.1282492	1.8736377	-0.1283733	1.8736373	-0.1283736
$\pi/2$	1.2719076	-0.0392334	1.2719424	-0.0392336	1.2719421	-0.0392337
π	0.1449800	0.1056147	0.1449329	0.1057388	0.1449329	0.1057390
2π	-0.1697461	-0.0233831	-0.1696989	-0.0235073	-0.1696986	-0.0235074
6π	-0.0344188	-0.0501655	-0.0343715	0.0500408	-0.0343712	0.0500407

In Table 3 we compare $g(\mu)$ obtained by the CFT and FFT methods with the exact value from (13). To use the FFT the input function $f(x)$ was provided over an interval $[-8, 8]$ with $N_F = 1024$. For the CFT algorithm, $f(x)$ is supplied within the interval $[-1, 1]$, and $N_s = N_F/8 = 128$ sample points describe the input function $f(x)$ with the same accuracy. Calculations were done using $k = 4$ and $N = 107$. It is evident from Table 3 that, while the CFT method yields six significant digits, the FFT is accurate only up to four significant digits near $\mu = 0$ and the accuracy gets worse at high μ; for example, at $\mu = 6\pi$ only two significant digits are obtained.

VI. Discussion of Accuracy. It is clear from Tables 1–3 that the accuracy of the CFT program is uniformly good for all values of μ. However, the errors associated with the FFT increase with μ. This conclusion is displayed in Figure 1, for (1) the square pulse, (2) the exponential function, and (3) the truncated Cornu spiral. Figure 1 plots the absolute error in the FFT value of $g(\mu)$ versus $\mu = 2\pi n/(\Delta x)$, with n displayed as the abscissa. Here Δx is the uniform interval between sample positions x_j where $j = 1, 2, \ldots, N_F$ and $N_F = 256$. In cases (1) and (3), the error oscillates periodically. We have therefore plotted the envelope of successive maximum errors as the conservative measure of validity of the FFT. In both of these cases, which are characterized by a jump in $f(x)$, the error is a strong superlinear function of n. For the square pulse, the error at the first maximum is 1.18×10^{-3}, whereas that at the last maximum is 3.80×10^{-2}, an increase of a factor of 32. For the truncated Cornu spiral, the corresponding numbers are 1.75×10^{-3} and 2.31×10^{-2} for an increase of a factor of 13. For the exponential function the increase is much less: from 2.60×10^{-3} to 4.60×10^{-3} with a ratio of 1.76. The errors in the imaginary part of $g(\mu)$ for the Cornu spiral case are similar to those for the exponential function case and are therefore not plotted.

The improved performance of the exponential function is caused by the fact that the discontinuity is in the first derivative rather than the function. The improved performance for the imaginary part of $g(\mu)$ for the Cornu spiral case is caused by the smaller jump in the imaginary part of $f(x)$.

In all cases, however, the errors associated with the FFT are of the order of 10^{-2} for $N_F = 256$, whereas the corresponding errors in the CFT program are not plotted because they are of the order of 10^{-7} (which is limited by round-off error in these single precision calculations). A comparable accuracy can be achieved over the limited region of μ plotted in Figure 1 using the FFT, but only if more than 20,000 points are used. Of course, $g(\mu)$ is then evaluated over a large range, and its values over that larger range are less reliable.

VII. Discussion of CPU Time. As mentioned previously, it is difficult to make a simple comparison between CPU times for the FFT and the CFT programs because they serve different purposes. To evaluate a single Fourier integral, the CFT will win "hands down". To compute precisely the N Fourier transforms yielded by the FFT when there are N input points, the FFT will win "hands down" if the input function is smooth enough. The question that we can resolve, here, is which algorithm will take less time if a given accuracy is desired, and a given fixed number of output points is required when discontinuities or kinks are present. In general, one only needs enough points in the transform space to adequately characterize the output function.

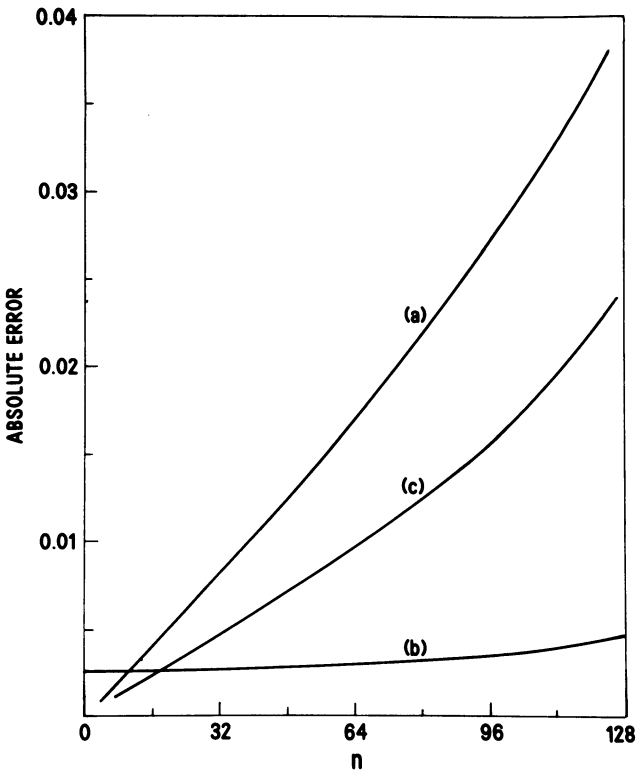

FIGURE 1

Variation of the absolute error in the FFT values of $g(\mu)$, $\mu = 2\pi n/(\Delta x)$ at various points in its domain for three cases considered in Section 4. A 256 point FFT is used to obtain the Fourier transform. As $g(\mu)$ is an even function in all three cases, only the region $\mu \geq 0$ is shown. The curves are plotted for (a) the square pulse, (b) the exponential function, and (c) the truncated Cornu spiral. In the last case the absolute error occuring in the real part of $g(\mu)$ are similar to those for case (b) and are therefore not plotted. In all three cases the absolute errors associated with the CFT are of the order of 10^{-7}, and on this scale the plot is indistinguishable from the abscissa-axis.

To make the required comparison we shall use the truncated Cornu spiral, Case 3 of Section 4. Calculations were done with the minimum number of points required to achieve an accuracy of 10^{-7} for each program, and an accuracy of 5×10^{-3} for each program. The results are summarized in Table 4. This table demonstrates that the time taken for the FFT sum scales with the number of sample points N_s as $N_s \log N_s$ for both the FFT method and the FFT portion of the CFT method. Three CFT runs are included in this table to demonstrate this point from a large number of auxiliary runs.

The table indicates that, to achieve machine accuracy (10^{-7} in single precision), the FFT program required 20,000 points and took longer than the CFT program to execute. On the other hand, for low accuracy, 5×10^{-3}, the FFT program was 7 times faster than the CFT program.

We should mention that the FFT program used was the Singleton [24] algorithm as incorporated in the PORT library. This is the fastest of the programs among those

available to us. The spline fit also came from the PORT library. No attempt has been made to optimize our end corrections or the spline fitting program.

TABLE 4
Comparison of CPU time for CFT and FFT algorithms

Program	Absolute accuracy	Sample points N_s	Core Kwords	CPU time in seconds for different parts of the program			
				FFT sum	Spline fit	End corrections	Total
FFT	1×10^{-7}	20,000	46	37.918	-	-	37.918
FFT	5×10^{-3}	256	7	0.250	-	-	0.250
CFT	1×10^{-7}	510	19	0.667	18.990	11.539	31.196
CFT	1×10^{-7}	240	17	0.233	8.840	4.671	13.744
CFT	5×10^{-3}	32	16	0.033	1.251	0.578	1.862

VIII. Conclusions. We have presented an algorithm to calculate the finite Fourier integral $g(\mu)$ of an input function $f(x)$ based on a spline fit to $f(x)$ and an analytic integration of the spline functions. This program is particularly useful as a Fourier transform when $f(x)$ has jumps in value, kinks (jumps in slope), or jumps in higher derivatives. In all these cases, $f(x)$ is not band limited, and the performance of the FFT is poor. For low desired accuracy, the FFT program with enough points to achieve this accuracy will run faster than the CFT program. For high accuracy, especially close to machine accuracy (10^{-7}), the CFT program will provide the required accuracy with shorter CPU time and a significantly lower core requirement. See column 4 of Table 4.

Because of the large space taken up by the compiled portion of the PORT library, the FFT program would appear to use less core when 5000 or fewer points are needed. However, in solving wave-propagation equations in three dimensions [14], two-dimensional Fourier transforms are needed, and the core requirements are proportional to N^2. Thus, even when the desired accuracy can be achieved using the FFT algorithm, this may not be feasible for several reasons: (i) the amount of core needed may not be available, (ii) $f(x)$ is measured only at certain values of x in a given experiment, (iii) $f(x)$ is calculated at discrete values of x numerically, and it may be expensive to compute it at a larger number of points.

It should be remembered that the CFT and the FFT do not in fact perform the same function. The FFT provides a discrete Fourier series representation of a function $f(x)$ based on requiring $f(x)$ to be periodic over an extended domain. The CFT, on the other hand, actually computes the Fourier integral of $f(x)$ over a finite

domain. The latter is often what the user wants. Our program satisfies this need, and for poorly band limited functions it certainly saves space and sometimes saves CPU time.

Acknowledgements. The work at CCNY was supported in part by the Army Research Office, by the Department of Energy and a grant from the City University of New York PSC-CUNY Research Award Program. We acknowledge the use of the Bell Laboratory Mathematical Subroutine Library (PORT).

Appendix—End Corrections for Uniform Mesh. In this appendix we evaluate

$$\Phi_{j,k}(\mu) = h_{j+k}\frac{(k-1)!}{(i\mu)^k}[0\, h_{j+1} \cdots h_{j+k}], \tag{A1}$$

for $j = 1, k-1$ and $j = N - 2(k-1), (N-k)$. At these values of j two or more mesh points coincide and care should be exercised in evaluating the divided difference in (A1). For the case of $\Phi_{1,k}$, there are k points which coincide, and we have

$$\Phi_{1,k}(\mu) = h\frac{(k-1)!}{(i\mu)^k}[0\,0 \cdots (k - \text{times})\, h]. \tag{A2}$$

Using the definition of divided differences [13], $[0\, h] = (e^{i\mu h} - 1)/h$. The second divided difference is given by

$$\begin{aligned}[0\,0\,h] &= \frac{[0\,h] - [0\,0]}{h} = \frac{1}{h}\left[\frac{(e^{i\mu h}-1)}{h} - \lim_{\varepsilon\to 0}\frac{(e^{i\mu\varepsilon}-1)}{\varepsilon}\right] \\ &= \frac{1}{h^2}\left[e^{i\mu h} - (1 + i\mu h)\right].\end{aligned} \tag{A3}$$

One can continue similarly using the fact that for a general function $f(x)$ the ith divided difference when all points coincide is given by $f^{(i-1)}(x)/(i-1)!$. We then obtain

$$\Phi_{1,k}(\mu) = h\frac{(k-1)!}{(i\mu h)^k}\left(e^{i\mu h} - \left(1 + i\mu h + \cdots + \frac{(i\mu h)^{k-1}}{(k-1)!}\right)\right) \tag{A4}$$

$$= h(k-1)!\sum_{p=k}^{\infty}\frac{(i\mu h)^{p-k}}{p!}, \tag{A5}$$

where the latter form is to be used for small values of μ.

The procedure outlined above can be carried out to obtain $\Phi_{2,k}$, $\Phi_{3,k}, \ldots$, etc., in a similar way. The algebra is lengthy but straightforward. We obtain

$$\Phi_{2,k}(\mu) = 2h(k-1)!\left[\sum_{p=k}^{\infty}(2^{p+1-k} - 1)\frac{(i\mu h)^{p-k}}{p!}\right], \tag{A6}$$

$$\Phi_{3,k}(\mu) = 3h(k-1)!\left[\sum_{p=k}^{\infty}\left(\frac{3^{p+3-k}}{3!} - \frac{2^{p+3-k}}{2!} + \frac{1}{2}\right)\frac{(i\mu h)^{p-k}}{p!}\right]. \tag{A7}$$

Further calculations are needed to obtain the end corrections at the other end point. For instance

$$\Phi_{N-k,k}(\mu) = \frac{(k-1)!}{(i\mu)^k}[h\,h\cdots h\,0]$$

$$= \frac{(-1)^k h(k-1)!}{(i\mu h)^k}\left[1 - e^{i\mu h}\left\{1 - i\mu h + \cdots + \frac{(-i\mu h)^{k-1}}{(k-1)!}\right\}\right] \tag{A8}$$

$$= h(k-1)!e^{i\mu h}\sum_{p=k}^{\infty}\frac{(-i\mu h)^{p-k}}{p!} = e^{i\mu h}\Phi_{1,k}(-\mu).$$

Similar algebra shows that the right-hand end corrections are related to the left-hand end corrections by the formula

$$\Phi_{N-k-(j-1),k}(\mu) = e^{i\mu jh}\Phi_{j,k}(-\mu), \qquad j = 1 \text{ to } (k-1). \tag{A9}$$

We now present an explicit form of the left-hand end corrections $\Phi_{j,k}(\mu), j = 1$ to $k-1$ for the spline orders $k = 2$, 3 and 4. The end corrections at the right-hand side are readily obtained from the following expressions by using the relation given by (A9). The series (A5)–(A7) can be summed to obtain an analytic form for $\Phi_{j,k}(\mu)$. For $k = 2$,

$$\Phi_{1,2} = \frac{h}{\lambda^2}[e^{\lambda} - (1+\lambda)]. \tag{A10}$$

For $k = 3$,

$$\Phi_{1,3} = \frac{2h}{\lambda^3}\left[e^{\lambda} - \left(1 + \lambda + \frac{1}{2}\lambda^2\right)\right], \tag{A11}$$

$$\Phi_{2,3} = \frac{h}{\lambda^3}[e^{2\lambda} - 4e^{\lambda} + 3 + 2\lambda]. \tag{A12}$$

For $k = 4$,

$$\Phi_{1,4} = \frac{6h}{\lambda^4}\left[e^{\lambda} - \left(1 + \lambda + \frac{\lambda^2}{2} + \frac{\lambda^3}{6}\right)\right], \tag{A13}$$

$$\Phi_{2,4} = \frac{3h}{2\lambda^4}[e^{2\lambda} - 8e^{\lambda} + 7 + 6\lambda + 2\lambda^2], \tag{A14}$$

$$\Phi_{3,4} = \frac{h}{\lambda^4}\left[e^{3\lambda} - \frac{9}{2}e^{2\lambda} + 9e^{\lambda} - \frac{11}{2} - 3\lambda\right], \tag{A15}$$

where $\lambda = i\mu h$. Care must be exercised to avoid computing errors when implementing the end corrections, (A10)–(A15), for small values of $|\mu h|$. In this case the exponentials in (A10)–(A15) are expanded in a power series which is terminated to achieve required accuracy. For the sake of completeness we give the expressions for $\Phi_{j,k}$ to be used in the CFT code for small $|\mu h|$. For $k = 2$,

$$\Phi_{1,2} = h\left(\frac{1}{2} + \frac{\lambda}{6} + \frac{\lambda^2}{24} + \frac{\lambda^3}{120} + \frac{\lambda^4}{720} + \cdots\right). \tag{A16}$$

For $k = 3$,

$$\Phi_{1,3} = h\left(\frac{1}{3} + \frac{\lambda}{12} + \frac{\lambda^2}{60} + \frac{\lambda^3}{360} + \frac{\lambda^4}{2520} + \cdots\right), \tag{A17}$$

$$\Phi_{2,3} = h\left(\frac{2}{3} + \frac{\lambda}{2} + \frac{\lambda^2}{30} + \frac{\lambda^3}{12} + \frac{31\lambda^4}{1260} + \cdots\right). \tag{A18}$$

For $k = 4$,

$$\Phi_{1,4} = h\left(\frac{1}{4} + \frac{\lambda}{20} + \frac{\lambda^2}{120} + \frac{\lambda^3}{840} + \frac{\lambda^4}{6720} + \cdots\right), \tag{A19}$$

$$\Phi_{2,4} = h\left(\frac{1}{2} + \frac{3\lambda}{10} + \frac{7\lambda^2}{60} + \frac{\lambda^3}{28} + \frac{31\lambda^4}{3360} + \cdots\right), \tag{A20}$$

$$\Phi_{3,4} = h\left(\frac{3}{4} + \frac{9\lambda}{10} + \frac{5\lambda^2}{8} + \frac{9\lambda^3}{28} + \frac{301\lambda^4}{2240} + \cdots\right), \tag{A21}$$

where $\lambda = i\mu h$.

Physics Department
City College of the City University of New York
New York, New York 10031

Bell Laboratories
Murray Hill, New Jersey 07974

1. F. ABRAMOVICI, "The accurate calculation of Fourier integrals by the fast Fourier transform technique," *J. Comput. Phys.*, v. 11, 1973, pp. 28–37.

2. M. ABRAMOWITZ & I. A. STEGUN (Editors), *Handbook of Mathematical Functions*, Dover, New York, 1965, p. 300.

3. G. P. AGRAWAL & M. LAX, "Fraunhofer diffraction in the beam approximation from two longitudinally separated slits," *J. Opt. Soc. Amer.*, v. 72, 1982, pp. 164–166.

4. P. J. DAVIS, *Interpolation and Approximation*, Blaisdell, New York, 1963, Section 3.7.

5. C. DE BOOR, "On calculating with B-splines," *J. Approx. Theory*, v. 6, 1972, pp. 50–62.

6. C. DE BOOR, "Package for calculating with B-splines," *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 441–472; C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

7. B. EINARSSON, "Numerical calculation of Fourier integrals with cubic splines," *BIT*, v. 8, 1968, pp. 279–286; *BIT*, v. 9, 1969, pp. 183–184 (errata).

8. B. EINARSSON, "Use of Richardson extrapolation for the numerical calculation of Fourier transforms," *J. Comput. Phys.*, v. 21, 1976, pp. 365–370.

9. L. N. G. FILON, "On a quadrature formula for trigonometric integrals," *Proc. Roy. Soc. Edinburgh*, v. 49, 1928, pp. 38–47.

10. S. M. FLATTE & F. D. TAPPERT, "Calculation of the effect of internal waves on oceanic sound transmission," *J. Acoust. Soc. Amer.*, v. 58, 1975, pp. 1151–1159.

11. J. GAZDAG, "Numerical convective schemes based on accurate computation of space derivatives," *J. Comput. Phys.*, v. 13, 1973, pp. 100–113.

12. R. W. HOCKNEY, "A fast direct solution of Poisson's equation using Fourier analysis," *J. Assoc. Comput. Mach.*, v. 17, 1965, pp. 95–113.

13. H. JEFFREYS & B. S. JEFFREYS, *Methods of Mathematical Physics*, Cambridge Univ. Press, Oxford, 1978, p. 262.

14. M. LAX, G. P. AGRAWAL & W. H. LOUISELL, "Continuous Fourier transform spline solution of unstable resonator field distribution," *Opt. Lett.*, v. 9, 1979, pp. 303–305.

15. R. C. LE BAIL, "Use of Fast Fourier Transforms for solving partial differential equations in physics," *J. Comput. Phys.*, v. 9, 1972, pp. 440–465.

16. M. J. MARSDEN & G. D. TAYLOR, "Numerical evaluation of Fourier integrals," in *Numerische Methoden der Approximations Theorie*, Band I (Vortragsauszuge einer Tagung, Oberwolfach, 1971), pp. 61–76; Internat. Schriftenreibe zur Numer. Math. Band 16, Birkhauser, Basel, 1972.

17. J. MARTI, "An algorithm recursively computing the exact Fourier coefficients of B-splines with non-equidistant knots," *J. Appl. Math. and Phys.*, v. 29, 1978, pp. 301–305.

18. R. PIESSENS & M. BRANDERS, "Computation of oscillating integrals," *J. Comput. Appl. Math.*, v. 1, 1975, pp. 153–164.

19. R. PIESSENS & A. HAEGEMANS, "Algorithm for the automatic integration of highly oscillatory functions," *Computing*, v. 13, 1974, pp. 183–193.

20. I. J. SCHOENBERG, "Contributions to the problem of approximation of equidistant data by analytic functions," *Quart Appl. Math.*, v. 4, 1946, pp. 45–49, 121–141.

21. I. J. SCHOENBERG, *Cardinal Spline Interpolation*, SIAM, Philadelphia, Pa., 1973, p. 2.

22. A. E. SIEGMAN & E. A. SZIKLAS, "Mode calculations in unstable resonators with flowing saturable gain, 2: Fast Fourier transform method," *Appl. Optics*, v. 14, 1975, pp. 1874–1889.

23. S. D. SILLIMAN, "The numerical evaluation by splines of Fourier transforms," *J. Approx. Theory*, v. 12, 1974, pp. 32–51.

24. R. C. SINGLETON, "An algorithm for computing the mixed radix fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, v. AU-17, 1969, pp. 93–103.

25. F. D. TAPPERT, *Numerical Solutions of the Korteweg-deVries Equation and its Generalizations by the Split-Step Fourier Method*, Lectures in Appl. Math., vol. 15, Amer. Math. Soc., Providence, R. I., 1974, pp. 215–216.