A New Numerical Fourier Transform in d-Dimensions

Normand Beaudoin and Steven S. Beauchemin

***Abstract*—The classical method of numerically computing Fourier transforms of digitized functions in one or in d-dimensions is the so-called *discrete Fourier transform* (DFT) efficiently implemented as *fast Fourier transform* (FFT) algorithms. In many cases, the DFT is not an adequate approximation to the continuous Fourier transform, and because the DFT is periodical, spectrum aliasing may occur. The method presented in this contribution provides accurate approximations of the continuous Fourier transform with similar time complexity. The assumption of signal periodicity is no longer posed and allows the computation of numerical Fourier transforms in a broader domain of frequency than the usual half-period of the DFT. In addition, this method yields accurate numerical derivatives of any order and polynomial splines of any odd degree. The numerical error on results is easily estimated. The method is developed in one and in d dimensions, and numerical examples are presented.**

***Index Terms*—Algorithm, aliasing, analysis, analytical, approximation, boundary, convolution, deconvolution, derivation, discrete, discrete Fourier transform (DFT), fast, fast Fourier transform (FFT), Fourier, integration, interpolation, numerical, Nyquist, spectrum, splines.**

I. Introduction

THE UBIQUITOUS Fourier transform and its numerical counterpart—the discrete Fourier transform (DFT)—in one or in more dimensions are used in many fields such as mathematics (linear systems, random process, probability, boundary-value problems), physics (quantum mechanics, optics, acoustics, astronomy), chemistry (spectroscopy, crystallography), and engineering (telecommunications, signal processing, image processing, computer vision, multidimensional, and signal processing) [1]–[5].

"The DFT is of interest primarily because it approximates the continuous Fourier transform." [1]. In this regard, the DFT, which is usually computed via a fast Fourier transform (FFT) algorithm, must be used with caution since it is not a correct approximation in all cases [6]–[9]. First, the DFT is periodical and constitutes a valid approximation of the Fourier transform in only one half of a period. Second, the sampling rate of the function to be submitted to the DFT is a critical issue. Without sampling the time[1] function at a sufficiently high rate, a phenomenon known as aliasing may become intolerable and spoil the accuracy of the DFT as an approximation of the Fourier transform. It could be thought that if the Nyquist criterion is fulfilled, everything should come out fine. The Nyquist criterion states that the sampling rate must be at least twice the highest frequency of the initial function [1], [2], [10]. However, a function may be defined between 0 and T only, and the highest frequency of such a time-limited function is infinite. Consequently, the DFT produces aliasing.[2] One could argue that even though the highest frequency is infinite, it is always possible to sufficiently increase the number of sampled data points such that the error of the DFT becomes as small as one wants. However, the required number of data points could be exceedingly large. As an example, for the function $h(t) = e^{-50t}$, $t \in [0, 1]$, the error on DFT$\{h\}$, around $f = 64$, decreases roughly as $N^{-1/3}$. Hence, one must increase N by a factor of 1000 to decrease the error by a factor of 10.

In some cases where the result of the DFT is used qualitatively, for example, in some Fourier transform infrared spectrometer (FTIR) experiments where the results are plotted and visually examined by an experienced spectroscopist, a high accuracy is not absolutely mandatory, but in some applications, such as in deconvolution, where a division is performed in the frequency domain, a slight error in the denominator function, particularly when it is close to zero, can seriously distort the results [11].

However, one may increase the accuracy of the numerical Fourier transform when the number of sampled data points is limited. This can be implemented through the assumption that the function, from which the sampled data points are extracted, and its derivatives are continuous. The sampling process, which is performed through the Dirac comb [1], in a sense, isolates each data point and renders them independent from each other. The function and its derivatives are no longer continuous. By re-establishing the continuity between the sampled data points, a method that yields an accurate numerical Fourier transform can be devised.

II. Theory in d-Dimension

Let $\vec{t} = (t_1, t_2 \cdots t_d) \in \mathbb{R}^d$ and $\vec{f} = (f_1, f_2 \cdots f_d) \in \mathbb{R}^d$, $d \in \mathbb{N}^*$. $\mathbb{R}$ is the set of real numbers, $\mathbb{N}$ the set of nonnegative integers, and $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$. Let us define Heaviside's function in d-dimensions:

$$\chi : \mathbb{R}^d \to \mathbb{R}, \quad \chi\left(\vec{t}\right) = \prod_{i=1}^{d} \chi\left(t_i\right) \tag{1}$$

Manuscript received August 13, 2001; revised November 19, 2002. The associate editor coordinating the review of this paper and approving it for publication was Dr. Xiang-Gen Xia.

The authors are with the Department of Computer Science, Middlesex College, University of Western Ontario, London, ON Canada N6A 5B7 (e-mail: normand-beaudoin@sympatico.ca; normand@csd.uwo.ca; beau@csd.uwo.ca).

Digital Object Identifier 10.1109/TSP.2003.810285

[1]Without loss of generality, the reciprocal variables *time* (t) and *frequency* (f) are used throughout this paper.

[2]The usual method of avoiding aliasing is to filter out the high-frequency components, thus modifying the original signal.

in which χ and χ are Heaviside's functions in d-dimensions and in one dimension, respectively. Let us define a d-dimensional rectangular functions such as

$$R\left(\vec{t}\right) = \chi\left(\vec{t} - \vec{0}^{-}\right)\chi\left(-\vec{t} + \vec{T}^{+}\right) \tag{2}$$

with $\vec{0}^{-} = (0^{-}, 0^{-}, \ldots 0^{-})$ and $\vec{T}^{+} = \left(T_1^{+}, T_2^{+}, \ldots, T_d^{+}\right)$, $T_\alpha \in \mathbb{R}$, $T_\alpha > 0$, $\forall\, \alpha$, and in which

$$0^{-} = \lim_{\varepsilon \to 0}(0 - \varepsilon), \quad T_\alpha^{+} = \lim_{\varepsilon \to 0}(T_\alpha + \varepsilon), \quad \varepsilon \in \mathbb{R}, \varepsilon > 0. \tag{3}$$

Let g: $\mathbb{R}^d \to (\mathbb{R}$ or $\mathbb{C})$ ($\mathbb{C}$ is the field of complex numbers) be a continuous function that admits directional derivatives of any order in any direction for all $\vec{t}\epsilon A$. A is any convex set such that $\left\{\vec{t} | R\left(\vec{t}\right) \neq 0\right\} \subset A \subseteq \mathbb{R}^d$. We now define the function

$$h\left(\vec{t}\right) = R\left(\vec{t}\right) g\left(\vec{t}\right) \tag{4}$$

and adopt the following definition for the Fourier transform:

$$\mathcal{F}\left\{h\left(\vec{t}\right)\right\} = \int_{\mathbb{R}^d} h\left(\vec{t}\right) e^{-i2\pi \vec{f}\cdot\vec{t}} d\vec{t}. \tag{5}$$

By expanding the inner product and reorganizing the terms, (5) becomes

$$\mathcal{F}\left\{h\left(\vec{t}\right)\right\} = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} h\left(t_1, \ldots, t_d\right) e^{-i2\pi f_1 \cdot t_1} dt_1\right] e^{-i2\pi f_2 \cdot t_2} dt_2 \cdots e^{-i2\pi f_d \cdot t_d} dt_d. \tag{6}$$

It is a well-known fact, which is evident from (6), that one d-dimensional Fourier transform of a function can be performed by d successive one-dimensional (1-D) Fourier transforms. Consequently, in the next section, we develop the theory in one dimension. In that case, generic nonindexed variables as t, f, $T \cdots$ that stand for any indexed variable of a particular dimension of the d-dimensional space are used.

III. Theory in One Dimension

By virtue of the properties of the differentiation of Heaviside's and Dirac-delta functions (δ) [2], [12], the nth derivative of h with respect to t is

$$h^{(n)}(t) = \chi\left(t - 0^{-}\right)\chi\left(-t + T^{+}\right) g^{(n)}(t) + D_n(t) \tag{7}$$

in which $D_n(t)$ is defined as

$$D_n(t) = \begin{cases} 0, & \text{if } n = 0 \\ \displaystyle\sum_{m=0}^{n-1} \left\{g^{(m)}\left(0^{-}\right)\delta^{(n-m-1)}\left(t - 0^{-}\right) - g^{(m)}\left(T^{+}\right)\delta^{(n-m-1)}\left(t - T^{+}\right)\right\}, & \text{if } n \in \mathbb{N}^{*}. \end{cases} \tag{8}$$

Equations (7) and (8) express the fact that the nth derivative of h with respect to t is the ordinary nth derivative of the function h strictly inside the rectangular box where it is continuous and differentiable, in addition to the nth derivative of h in the regions where it is discontinuous.

According to our definition of the Fourier transform, we have

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \int_{-\infty}^{\infty} h^{(n)}(t)\, e^{-i2\pi f t} dt. \tag{9}$$

We can expand the integral in (9) into parts to form

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \int_{-\infty}^{0} h^{(n)}(t) e^{-i2\pi f t} dt + \int_{0}^{T} h^{(n)}(t) e^{-i2\pi f t} dt + \int_{T}^{\infty} h^{(n)}(t) e^{-i2\pi f t} dt. \tag{10}$$

The sum of the first and last integrals of the right-hand side of (10) clearly is $\mathcal{F}\{D_n(t)\}$. Hence, (10) becomes

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \int_{0}^{T} h^{(n)}(t) e^{-i2\pi f t} dt + \mathcal{F}\{D_n(t)\}. \tag{11}$$

By separating the interval $[0, T]$ into N equal to $\Delta t = T/N$ subintervals, (11) can be rewritten as

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \sum_{j=0}^{N-1} \left\{\int_{j\Delta t}^{(j+1)\Delta t} h^{(n)}(t) e^{-i2\pi f t} dt\right\} + \mathcal{F}\{D_n(t)\}, \quad j \in \mathbb{N}. \tag{12}$$

Since $h^{(n)}$ is continuous and differentiable between and at 0 and T, it can be approximated for $t \in [j\Delta t, (j+1)\Delta t]$ for each $j \in [0, N-1]$, with a Taylor expansion

$$h^{(n)}(t) = \sum_{p=0}^{\infty} \frac{h_j^{(p+n)}(t - j\Delta t)^p}{p!}, \quad p \in \mathbb{N} \tag{13}$$

where $h_j^{(m)}$ is the mth derivative of h at point $t = j\Delta t$. Merging (12) and (13) yields

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \sum_{j=0}^{N-1} \left\{\int_{j\Delta t}^{(j+1)\Delta t} \left(\sum_{p=0}^{\infty} \frac{h_j^{(p+n)}(t - j\Delta t)^p}{p!}\right) e^{-i2\pi f t} dt\right\} + \mathcal{F}\{D_n(t)\}, \quad j \in \mathbb{N}. \tag{14}$$

With the substitution $\tau = t - j\Delta t$ and an adequate permutation of the integral and sums on j and p, (14) becomes

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \sum_{p=0}^{\infty} \left\{\left(\int_{0}^{\Delta t} \frac{\tau^p e^{-i2\pi f \tau}}{p!} d\tau\right)\left(\sum_{j=0}^{N-1} h_j^{(p+n)} e^{-i2\pi f j \Delta t}\right)\right\} + \mathcal{F}\{D_n(t)\}. \tag{15}$$

To numerically compute the Fourier transform of h, we must evaluate it for some discrete values of f. Let $f = k\Delta f = k/T$, $k \in \mathbb{N}$ be these discrete variables. In addition, let us define H_k as the discrete version of $\mathcal{F}\left\{h^{(n)}(t)\right\}$. The integral in (15) depends only on the variable f (or k) and on parameters p and

Δt and can be evaluated analytically, whether f is continuous or discrete, once and for all, for each value of p as

$$I_p = \frac{1}{p!}\int_0^{\Delta t} \tau^p e^{-i2\pi f\tau} d\tau. \tag{16}$$

Since the integral in the definition of I_p is always finite and, in the context of the Gamma function [13], $p! = \pm\infty$ when p is a negative integer, then $I_p = 0$ for $p < 0$. When $p = 0$, by direct integration of (16), we obtain

$$I_0 = \frac{1 - e^{-i2\pi f\Delta t}}{i2\pi f} \tag{17}$$

and by integration by parts, we have a recursive form for I_p for any $p \in \mathbb{N}^*$

$$I_p = \frac{1}{i2\pi f}\left(I_{p-1} - \frac{\Delta t^p}{p!} e^{-i2\pi f\Delta t}\right). \tag{18}$$

The summation on j in (15), when $f = k\Delta f = k/T$, is the discrete Fourier transform of the sequence $h_j^{(p+n)}$, $j \in [0,\, N-1] \subset \mathbb{N}$ [1]. We denote it as $F_{p+n,\,k}$. Since $\Delta t = T/N$ and $f = k/T$, we have

$$F_{p+n,\,k} = \sum_{j=0}^{N-1} h_j^{(p+n)} e^{-i2\pi(kj/N)}. \tag{19}$$

One should note that although we wrote I_p and $F_{p+n,\,k}$ instead of $I_p\,(f \text{ or } k)$ and $F_{p+n,\,k}\,(f \text{ or } k)$, these functions always depend on f or k.

Substituting (16) and (19) in (15), we obtain the following result:

$$\mathcal{F}\left\{h^{(n)}(t)\right\} = \sum_{p=0}^{\infty} I_p F_{p+n,\,k} + \mathcal{F}\left\{D_n(t)\right\}. \tag{20}$$

When $n = 0$, (20) becomes

$$H_k = \sum_{p=0}^{\infty} I_p F_{p,\,k}. \tag{21}$$

Now, integrating by parts, the right-hand side of (9) yields

$$\mathcal{F}\left\{h^{(n+1)}\right\} = i2\pi f\mathcal{F}\left\{h^{(n)}\right\}. \tag{22}$$

Defining $b_n = i2\pi f\mathcal{F}\{D_n\} - \mathcal{F}\{D_{n+1}\}$, combining (20) and (22), and reorganizing the terms yields

$$-i2\pi f I_0 F_{n,\,k} + \sum_{p=1}^{\infty}\left(I_{(p-1)} - i2\pi f I_p\right) F_{p+n,\,k} = b_n. \tag{23}$$

With the definition

$$J_\alpha = I_{\alpha-1} - i2\pi f I_\alpha \tag{24}$$

(23) becomes

$$J_0 F_{n,\,k} + \sum_{p=1}^{\infty} J_p F_{p+n,\,k} = b_n. \tag{25}$$

Given the definition of g and h, we have $g^{(n)}(0^-) = g^{(n)}(0) = h^{(n)}(0)$ and $g^{(n)}(T^+) = g^{(n)}(T) = h^{(n)}(T)$. Using these facts in addition to the properties of Fourier transform and those of Dirac delta functions [12], one easily observes that expanding b_n results in

$$b_n = h^{(n)}(T)\, e^{-i2\pi fT} - h^{(n)}(0). \tag{26}$$

In the discrete context, where $f = k/T$, (26) takes the following simple and significant form:

$$b_n = h^{(n)}(T) - h^{(n)}(0) = h_N^{(n)} - h_0^{(n)}. \tag{27}$$

We refer to b_n, $n \in \mathbb{N}$ as the boundary conditions of the system.

Up to this point, all equations are rigorously exact since p tends toward infinity. However, in practical situations, we introduce approximations by limiting the range on p. Let us define $\theta \in \mathbb{N}$ as the truncating parameter, which, for reasons discussed later, is always chosen as an odd integer. We refer to it as the order of the system.

Let us expand (25) for each value of $n \in [0,\, \theta - 1] \subset \mathbb{N}$. This generates a system of θ different equations, and for each of these, we let p range from 1 to $\theta - n$. This gives the following system written in a matrix form:

$$\begin{bmatrix} J_1 & J_2 & \cdots & J_\theta \\ J_0 & J_1 & \cdots & J_{\theta-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_1 \end{bmatrix} \begin{bmatrix} F_{1,\,k} \\ F_{2,\,k} \\ \vdots \\ F_{\theta,\,k} \end{bmatrix} \simeq \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{\theta-1} \end{bmatrix} + \begin{bmatrix} -J_0 F_{0,\,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{28}$$

or, more compactly, as

$$\mathtt{MF} \simeq \mathtt{B} + \mathtt{C}. \tag{29}$$

The general expression for elements of $\mathtt{M}$ is

$$(\mathtt{M})_{\mu\nu} = J_{\nu-\mu+1}. \tag{30}$$

Let us now write (29) as

$$\mathtt{F} \simeq \mathtt{M}^{-1}(\mathtt{B} + \mathtt{C}). \tag{31}$$

On the right side of (31), every term, except $\mathtt{B}$, is known. $\mathtt{C}$ contains the term $F_{0,\,k}$, which is the DFT of the initial function. All other terms in $\mathtt{C}$ and $\mathtt{M}^{-1}$ depend on parameters (N and T) of the function to be transformed and not on its data. (The evaluation of $\mathtt{B}$, which are the boundary conditions, is the subject of the next section.) With the knowledge of $\mathtt{F}$ from (31), the terms of (21), for $p \in [0,\, \theta]$, are completely determined. Thus, the truncated version of (21) can be written as

$$H_k \simeq \sum_{p=0}^{\theta} I_p F_{p,\,k}. \tag{32}$$

Let us define a one row matrix as $\mathtt{I}_\theta = [I_1\ I_2\ \cdots\ I_\theta]$ and write (32) as follows:

$$H_k \simeq I_0 F_{0,\,k} + \mathtt{I}_\theta \mathtt{F}. \tag{33}$$

With (33), we approximate the Fourier transform (or its inverse) of a digitized function in one dimension. The digitized Fourier transform obtained with (33) is not bandlimited (as with the DFT, which is periodical). Equation (33) is valid as an accurate approximation of the analytical Fourier transform for all values of $k \in \mathbb{Z}$ [14].

One should note that in (32), I_p is undetermined when $f = 0$. This can always be solved by Hospital's rule or by direct integration of (16) with $f = 0$. Once these undeterminations due to $I_p\,(f = 0)$ are solved, every term of (32) or (33) behaves well as long as θ is an odd integer. For even values of θ, there are singularities at $f = (\kappa + 1/2)\, N\Delta f$, $\kappa \in \mathbb{N}$ that cannot be removed. These are due to the fact that when θ is even, some terms in $\mathtt{M}^{-1}$ (involved in the computation of $F_{p,\,k}$) have denominators that go to 0, whereas the numerators remain nonzero. This is an imperative reason to choose odd values for θ.

It is important to observe that although (31) involves the inversion of $\mathtt{M}$, it does not mean that one has to invert the matrix each time a numerical Fourier transform is computed. Equation (31) is a part of the mathematical development of the formula used to compute the numerical Fourier transform. Each of its terms can be handled symbolically. For a specific order θ, the matrix $\mathtt{M}$ is inverted only once to establish the formula for the numerical Fourier transform.

Equation (33) can be thought of as the Fourier domain counterpart for a complete function of the Taylor series in the time domain for a single point. In the same way, a Taylor expansion is more complex than its first term, and (33) is more complex than the standard FFT. If, in a specific application, one does not require to get rid of the periodicity of the DFT or is not bothered by the aliasing, if one does not need more accuracy or does not have to compute the Fourier transform for values of f beyond those inside the usual frequency band of the DFT, then there is no point in using anything else than the usual DFT (FFT), which remains a simple and valid tool in these circumstances. However, if one or more of the aforementioned virtues of (33) are desirable, the benefits of these few more terms in the series are welcomed, and the burden of the added mathematical complexity is compensated, as we will see, by the fact that the time complexity of (33) remains that of the FFT.

Furthermore, although (33) contains the symbolic form of $\mathtt{F}$, which can be used as is to form a single formula without numerically evaluating each term of $\mathtt{F}$ separately, (31) can be used to compute each term of $\mathtt{F}$ for values of k from 0 to $N-1$ to produce θ accurate approximations of the DFT of the derivatives $h_j^{(p)}$ for values of $p \in [1,\, \theta]$. Thus, applying the inverse DFT operation to each of these approximations generates the corresponding sequences $h_j^{(p)}$ that are accurate numerical derivatives of all orders from 1 to θ of the initial function $h_j^{(0)}$. This implies that one can accurately compute the derivatives of any order of a digitized function or signal [14].

Derivatives calculated in this manner are continuous in between and at each data point. Thus, with the $DFT\,(FFT)$, we obtain spline polynomials of any odd degree, with their corresponding properties [14]. Hence, such high-order spline interpolation polynomials allow integrals between any limit to be accurately computed [14].

Let R_θ be any result (Fourier transform, derivative, or integral) obtained with an arbitrary order θ. Once the boundary conditions are established, the error on R_θ (noted E_θ) can be fairly estimated since $R_{\theta+2}$, relative to R_θ, can be considered almost as the exact result. To perform error estimation, one can use the following relation: $E_\theta = \mathcal{O}\,(R_\theta,\, R_{\theta+2})$, where $\mathcal{O}$ is any operator one can define to meet specific needs [14].

IV. Boundary Conditions

The use of (33) implies the evaluation of matrix $\mathtt{B}$, which are the boundary conditions of the function, as defined in (27). One can use any method one wants to accomplish that task. However, a method is proposed here that proves to give very accurate boundary conditions, as long as the function to which it is applied is smooth enough. Fortunately, what is meant by *smooth enough* can be determined by the method itself. This is fully discussed toward the end of this section.

Let us expand (25) for each value of $n \in [0,\, \theta - 1] \subset \mathbb{N}$. This generates a system of θ different equations, and for each of these, we let p range from 1 to $\theta - n - 1$. This gives the following system, which is written in matrix form:

$$\begin{bmatrix} J_0 & J_1 & \cdots & J_{\theta-1} \\ 0 & J_0 & \cdots & J_{\theta-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_0 \end{bmatrix} \begin{bmatrix} F_{0,\,k} \\ F_{1,\,k} \\ \vdots \\ F_{\theta-1,\,k} \end{bmatrix} \simeq \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{\theta-1} \end{bmatrix} \tag{34}$$

or, more compactly, as

$$\mathtt{M}_a \mathtt{F}_a \simeq \mathtt{B}. \tag{35}$$

Note that the matrix $\mathtt{M}_a$ is completely known since each and every one of its terms depends only on f. The general expression for the elements of $\mathtt{M}_a$ is

$$(\mathtt{M}_a)_{\mu\nu} = J_{\nu-\mu}. \tag{36}$$

Matrix $\mathtt{B}$ is unknown. If it were, we could evaluate $\mathtt{F}_a$ from (35) as

$$\mathtt{F}_a \simeq \mathtt{M}_a^{-1} \mathtt{B}. \tag{37}$$

Unfortunately, for $f = \kappa N \Delta f$, $\kappa \in \mathbb{N}$, it happens that $\det(\mathtt{M}_a) = 0$ and $\mathtt{M}_a^{-1}$ cannot be computed. However, for values of f around $N\Delta f/2$, the approximation (37) is quite accurate when the function is smooth enough. We take advantage of this fact to compute $\mathtt{B}$.

The first element of $\mathtt{F}_a$, which is $(\mathtt{F}_a)_1 = F_{0,\,k}$, is the DFT of the sequence h_j. It is completely determined for each value of k. It is not the same situation for the other elements of $\mathtt{F}_a$, which are still unknown. In addition, the elements of matrix $\mathtt{M}_a^{-1}$ are given for each value of $f = k\Delta f$. We can then extract the following from (37):

$$F_{0,\,k} \simeq \left(row_1 \mathtt{M}_a^{-1}\right) \mathtt{B} = \sum_{\nu=1}^{\theta} L_\nu\, (\mathtt{B})_\nu \tag{38}$$

in which, for brevity, we set $L_\nu = \left(\mathtt{M}_a^{-1}\right)_{1,\,\nu}$. Let us now define Ω as an interval of θ values of k centered on $N/2$, as follows:

$$\begin{aligned} \Omega &= \left[\frac{N}{2} - \left(\frac{\theta-1}{2}\right),\, \frac{N}{2} + \left(\frac{\theta-1}{2}\right)\right] \\ &= [k_1,\, k_2,\, \ldots,\, k_\theta] \subset \mathbb{N}. \end{aligned} \tag{39}$$

Let us expand (38) for each value of $k \in \Omega$. (It is understood that in practical cases, for each instance of f in each term, one has to replace it by $k\Delta f$.) Doing so yields the following system:

$$\begin{bmatrix} F_{0,\,k_1} \\ F_{0,\,k_2} \\ \vdots \\ F_{0,\,k_\theta} \end{bmatrix} \simeq \begin{bmatrix} L_1(k_1) & L_2(k_1) & \cdots & L_\theta(k_1) \\ L_1(k_2) & L_2(k_2) & \cdots & L_\theta(k_2) \\ \vdots & \vdots & \ddots & \vdots \\ L_1(k_\theta) & L_2(k_\theta) & \cdots & L_\theta(k_\theta) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{\theta-1} \end{bmatrix}. \tag{40}$$

Note that $L_\nu(k_\beta)$ is compact notation for $\left(\mathtt{M}_a^{-1}\right)_{1,\,\nu}$ evaluated at $k = k_\beta$ and $f = k_\beta \Delta f$. Let us express (40) more compactly as $\mathtt{F}_c \simeq \mathtt{WB}$, which yields

$$\mathtt{B} \simeq \mathtt{W}^{-1}\mathtt{F}_c. \tag{41}$$

Equation (41) completely determines B from $F_{0,\,k}$ (the discrete Fourier transform of the digitized initial function h). According to (27), knowing B numerically specifies b_n ($n \in [0,\, \theta - 1]$). They are the boundary conditions of the system.

Equation (24) is the generic expression of terms in $\mathtt{M}_a$ and M. Defining

$$x = e^{-i2\pi f \Delta t} = e^{-i2\pi(k/N)} \tag{42}$$

and successively substituting (17), (18), and (42) in (24) yields

$$J_0 = x - 1 \quad \text{and} \quad J_\alpha = \frac{\Delta t^\alpha}{\alpha!} x. \tag{43}$$

These substitutions lead to dramatic simplifications in the expression of $\mathtt{M}_a$. Nevertheless, terms in matrix W in (40), once expanded, still appear rather cumbersome. Further simplification is desirable. A detailed examination of L_ν reveals that

$$L_\nu = \left(\frac{-\Delta t^{\nu-1}}{(\nu-1)!}\right)\left(\frac{E_{\nu-1}}{(x-1)^\nu}\right) \tag{44}$$

in which E_α is Euler's polynomial [15] of degree α, which is defined as

$$E_\alpha = \sum_{m=0}^{\alpha}\left(\sum_{j=0}^{m}(-1)^j (m-j)^\alpha \frac{(\alpha+1)!}{j!\,(\alpha-j+1)!}\right) x^m. \tag{45}$$

Note that Euler's polynomials can be built with Euler's triangle. Equations (44) and (45) are used to fill matrix W as specified by (40).

To compute B through (41), the matrix W has to be inverted. It should be noted that W does not depend on the values of the function but only on its parameters (N and T). The size of W is $\theta \times \theta$. θ is the order of the system and the degree of each interpolating Taylor polynomials between each data point. Although it is possible and not difficult to do, it is very unlikely that one uses very large values for θ. For example, in the context of this method, $\theta = 21$ is a very high value, but W is still a relatively small numerical matrix that can be inverted efficiently. Moreover, for a class of functions that share the same parameters (as is the case in many dimensions), matrix W is inverted only once. The computation of boundary conditions is then a rapid operation.

In the context of the Nyquist sampling theorem, the function to be sampled is bounded in the frequency domain, and the sampling rate must be, at least, twice the highest frequency of the function [1], [2], [10]. Hence, such a function is always smooth enough. However, in the context of this paper, functions are bounded in the time domain and unbounded in the frequency domain. This is why the boundary conditions become an issue. The method presented in this section to compute boundary conditions is quite efficient as long as the function is smooth enough between the boundaries. Since the Nyquist criterion can no longer be used, one must have another criterion of smoothness. Fortunately, an operational criterion can be derived from the method itself, and it appears that a function does not have to be so smooth to be considered smooth enough by the method.

Let us define the following measure of error on boundary conditions for a specific order θ as follows:

$$E_\theta = \mathcal{O}(\mathtt{B}_\theta,\, \mathtt{B}_{\theta+2}). \tag{46}$$

The operator $\mathcal{O}$ can be defined as one wants to fulfill its purpose. For example, one could use, as we do in the examples below, $E_\theta(i) = \left|(\mathtt{B}_\theta)_i - (\mathtt{B}_{\theta+2})_i\right|$, $i = 1, \ldots, \theta$.

For a specific sampled function, one may increase θ as long as E_θ decreases. The value of θ for which the error E_θ is minimum is the optimum value of θ and is denoted θ_{opt}. The smoother the function is between the boundaries, the higher θ_{opt} becomes. If the sampling rate is reduced or if the frequency component of the function is increased, θ_{opt} decreases. If it happens that $\theta_{opt} < 1$, then the method presented in this section to obtain boundary conditions is no longer adequate. It is the case, for example, for a discrete random function or for a high-frequency function that is so sparsely sampled that it appears almost to be random. However, in such situations, no method can claim to be very accurate, and one has to resort to information about the function that is not obtained from mere sampled data points. If such external information is unavailable, then one could use a method as simple as possible to compute the boundary conditions, such as linear interpolation between points.

It should be noted that even though θ_{opt} for boundary conditions may be low, once these boundary conditions are established, a value of θ as high as one wants can be used with (31) or (33).

V. Time Complexity

In one dimension, a close examination of (28) and (33) reveals that the computation of only one FFT is required. The other terms form a correcting operation to be applied only once on each of the N values of the FFT. The time complexity of the entire correcting operation is thus $O(N)$, and the time complexity of the FFT is $O(N \log N)$. Hence, the time complexity of the entire algorithm is $O(N \log N)$ when θ is kept constant.

The time complexity of the correcting operation, relative to θ, is $O\left(\theta^2\right)$, but the error on computed results decreases exponentially with the increase of θ. Hence, as long as one can afford to increase θ, the tradeoff is strongly beneficial [14].

According to (6), this method can be applied sequentially to compute an accurate d-dimensional Fourier transform. In this multidimensional case, for each $\alpha \in \{1, 2, \ldots, d\}$, we have $t_\alpha \in [0,\, T_\alpha]$. This interval is separated into N_α equal $\Delta t_\alpha = T_\alpha / N_\alpha$ subintervals, and $f_\alpha = k_\alpha \Delta f_\alpha = k_\alpha / T_\alpha$. As with the ordinary DFT(FFT), the order in which the dimensions are

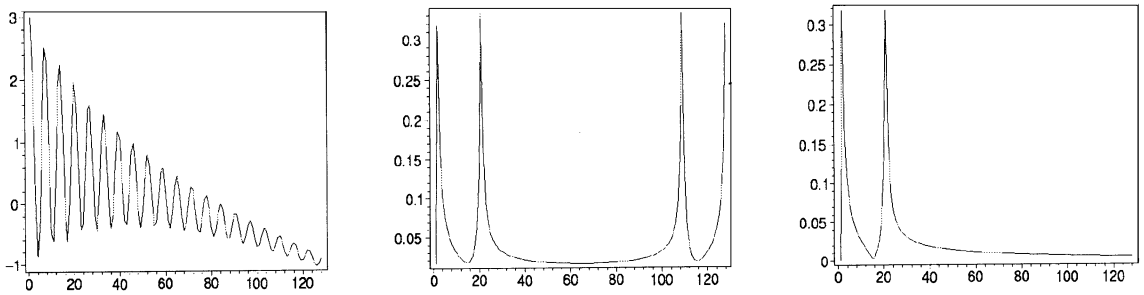


Fig. 1. (a) Function (48) with $fc = 20$. (b) DFT of the function shown in (a). (c) Numerical Fourier transform computed with PM and with $\theta = 9$. $\overline{E}$ is the average of the absolute value of the error. For the DFT, the error is computed for points from 0 to $N/2$. For PM, the error is computed on the full range of frequency.

treated is irrelevant. The number of times (33) has to be applied to completely compute the d-dimensional Fourier transform is

$$PQ \quad \text{where} \quad P = \prod_{\alpha=1}^{d} N_\alpha \quad \text{and} \quad Q = \sum_{\beta=1}^{d} \frac{1}{N_\beta}. \qquad (47)$$

The time complexity is then $O(P \log P)$. Let us set $N_\alpha = a_\alpha N, \ \forall\, \alpha, a_\alpha$ being constants. It is easy to show that the time complexity is $O\left(N^d \log N\right)$, which is the same as that of the DFT in the d-dimension.

VI. Examples in One Dimension

In this section, several examples are presented to illustrate the different aspects of the proposed method and to demonstrate the concepts of the previous sections. (For brevity, in the text and in the figures, PM stands for the proposed methods of this paper. The context makes it clear if it is PM for the boundary conditions or for Fourier transform.) The first set of examples are based on the following function:

$$h(t) = 2e^{-3t}\cos(2\pi f_c t) - 2t + 1, T = 1, N = 128. \quad (48)$$

(This function mimics typical measurements of thermodynamics experiments conducted to determine the ratio C_p/C_v, these parameters being the specific heat of a gas.)

When $f_c = 10$, there are 12.8 data points for each cycle of the cosine term of the function. It is not a very high sampling rate, but it is sufficient to make the function appear relatively smooth between the boundaries. Then, we obtain $\theta_{opt} = 39$, which means that a high accuracy can be reached with the method for boundary conditions described previously. When $f_c = 20$, we have 6.4 data points per cycle, and $\theta_{opt} = 15$. The sampled function begins to appear coarse (see Fig. 1) but is still manageable and the method for boundary conditions performs very well. When $f_c = 25, \theta_{opt} = 1$, there are 5.1 data points for each cycle of the cosine, and the method can still be used but it is at the limit. When the sampling rate relatively to the frequency of the function is reduced, θ_{opt} drops rapidly, but conversely, this means that the sampling rate does not have to be greatly increased to obtain high values for θ_{opt}. In the examples below, when $\theta_{opt} < 1$, the following simple formulas are used for the boundary conditions: $b_0 = h_{N-1} - h_0, b_1 = -(h_1 - h_0)/\Delta t$ and $b_i = 0$ for $i > 1$.

Fig. 1 shows function (48) when $f_c = 20$ and its transforms by the DFT algorithm and by the PM algorithm. The parameters and error measures are indicated in the Fig. 1. The curve given by PM is graphically indistinguishable from the analytical transform.

Fig. 2 shows function (48) when $f_c = 50$ and its transform by the DFT algorithm and by the PM algorithm for different values of θ. The initial function is quite rough. With $f_c = 50$, there are only 2.56 data points per cycle of the cosine term. The initial function $h(t)$ is unbounded in frequency, and hence, it is not possible to discuss the sampling rate in terms of the Nyquist criterion since the highest frequency is infinite. If the cosine term in the function was alone and unbounded in the time domain, the minimum number of samples per cycle needed to satisfy the Nyquist criterion would be 2. With 2.56 samples per cycle, we are almost at the limit. This function is demanding for both methods, DFT and PM. The DFT is indeed periodical and the two central peaks are very close, which worsen the already present aliasing. Our PM attempts to remove the periodicity and the aliasing to make the computed result as close as possible to the exact result. To achieve this, the order θ has to be increased. With $\theta = 3$, Fig. 2 clearly shows a residual peak around $f = 80\Delta f$. With $\theta = 13$, the result is almost exact. The residual peak disappears and the average error reduces to 4.9×10^{-5}, which is about 408 times more accurate than the DFT. Furthermore, a proper removal of periodicity gives us access to the frequency content of the function, not only between 0 and $(N/2)\,\Delta f$ but also between $(N/2)\,\Delta f$ and $N\Delta f$, and even beyond, as we will see in the next example. In the present case, under integration, the frequency content between $(N/2)\,\Delta f$ and $N\Delta f$ is approximately 16% of the transform between 0 and $N\Delta f$. It may not be negligible. With usual DFT, this information is unavailable.

The next example uses another usual and frequent function: a fast exponential decay

$$h(t) = e^{-100t}. \qquad (49)$$

Fig. 3 shows this function and its transforms. The number of sampled points between 10 and 100% of the amplitude is only four. The frequency spectrum of such a function is particularly wide. It is evident that the aliasing prevents the DFT from revealing the frequency content of $h(t)$. To obtain it accurately, this time, PM has been used to compute the Fourier transform over three periods of the usual DFT. Fortunately, PM does not have to be modified for such a task. The lower curve of Fig. 3(b)

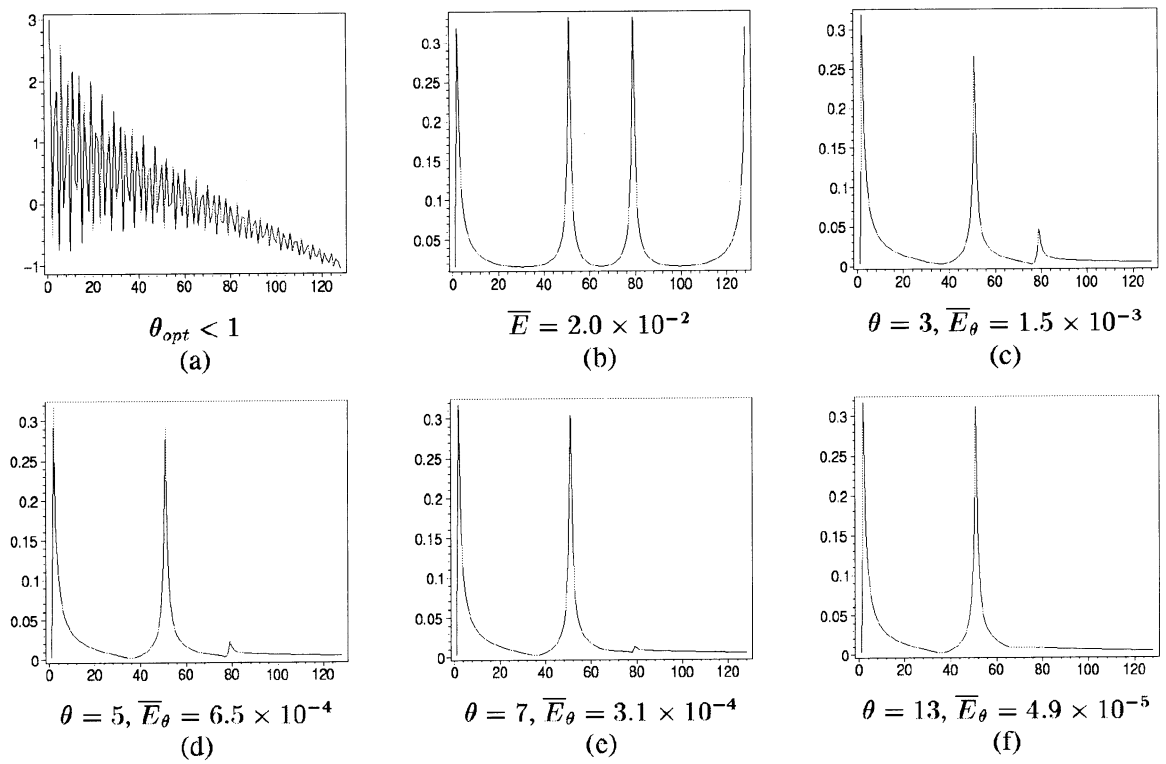


Fig. 2. (a) Function (48) with $fc = 50$. (b) DFT of the function shown in (a). (c) Numerical Fourier transform computed with PM and with $\theta = 3$. (d) $\theta = 5$. (e) $\theta = 7$. (f) $\theta = 13$. $\overline{E}$ is the average of the absolute value of the error. For the DFT, the error is computed for points from 0 to $N/2$. For PM, the error is computed on the full range of frequency.

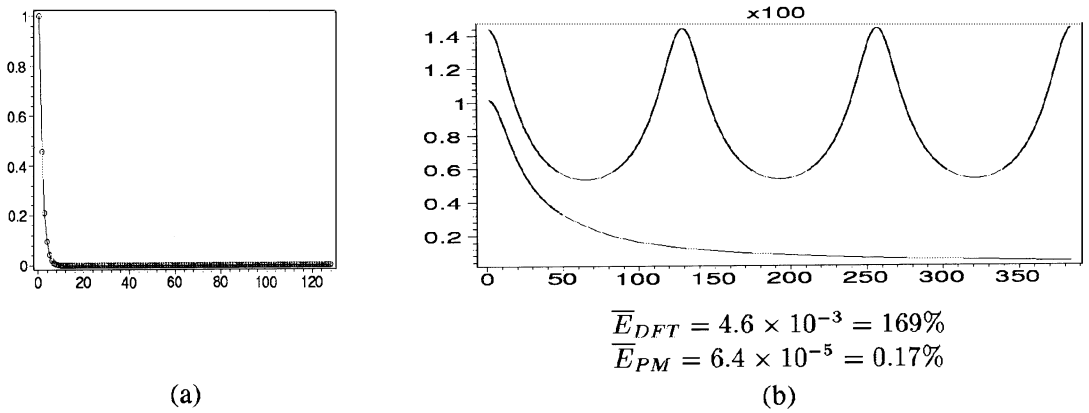


Fig. 3. (a) Function (49). (b) Upper curve is the DFT of the function shown in (a). The lower curve is the numerical Fourier transform computed with PM and with $\theta = 5$. As usual, $\overline{E}$ indicates the average of the absolute value of the error.

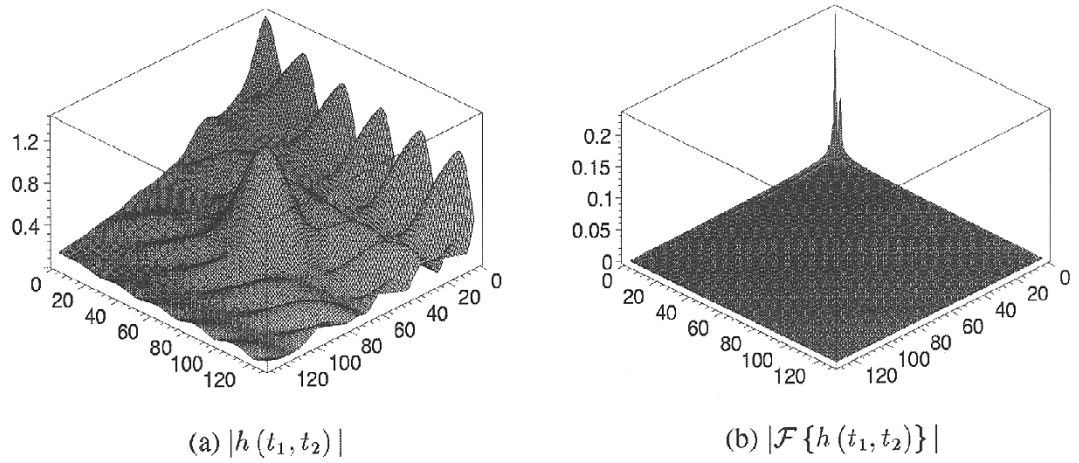
shows this result, which is graphically indistinguishable from the exact transform. This time, under integration, the frequency content between $(N/2)\Delta f$ and $3N\Delta f$ is about 50% of the total. It cannot be neglected. This example clearly shows that PM performs accurately for a function that is far from being smooth, whereas the classical DFT fails completely.

VII. Examples in Two Dimensions

In this section, an example in two dimensions is used to illustrate the algorithm. The choice of such an example is not obvious. That is to say, the function must not be trivial; it must be difficult enough for the computation of the Fourier transform to be numerically demanding. On the other hand, for purposes of comparison, the Fourier transform of the function must be analytically known. The chosen initial function for our example is then the following complex function:

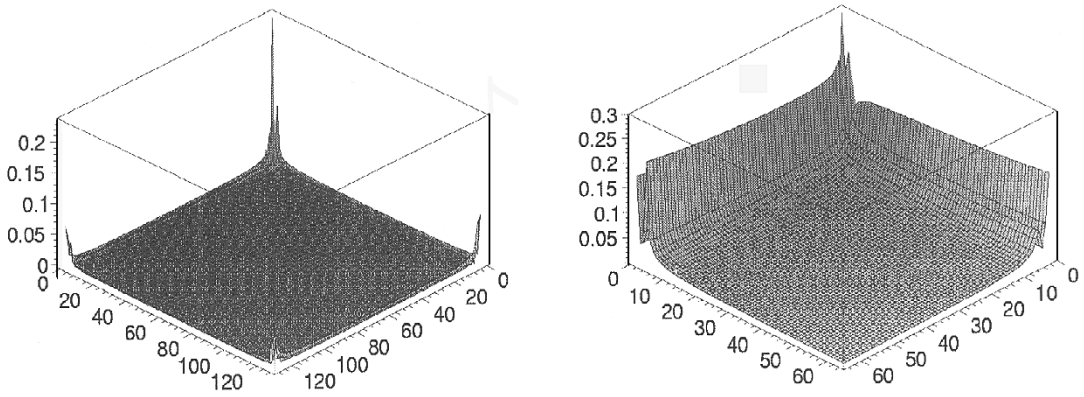
$$h(t_1, t_2) = \Big(\cos(9t_1)\cos(11t_1 + 17t_2)\, e^{-2.5t_1},\ e^{-2(t_1+t_2)} + e^{\left[-100(t_1-0.5)^2-50(t_2-0.5)^2\right]}\Big)$$
$$T_1 = T_2 = 1. \tag{50}$$

The real part is a combination of damped, slanted oscillations. The imaginary part is a nonsymmetrical Gaussian peak purposefully slanted by an exponential to avoid error cancellation by symmetry. For both variables, the function is discontinuous at 0 and at T_α, $\forall\, \alpha$. Fig. 4(a) and (b) shows the modulus of (50) and of its analytical Fourier transform for $N_1 = N_2 = 128$, respec-



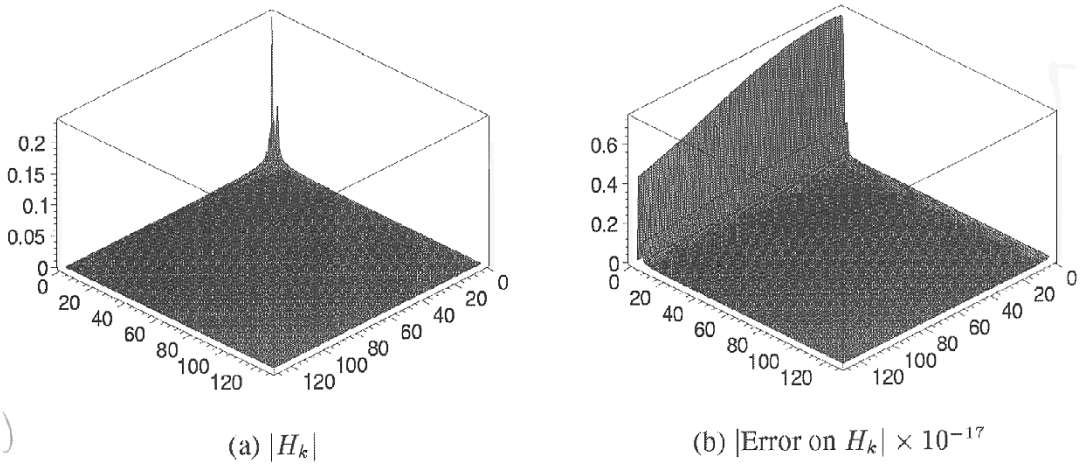
(a) $|h(t_1, t_2)|$ (b) $|\mathcal{F}\{h(t_1, t_2)\}|$

Fig. 4. Modulus of h and of $\mathcal{F}\{h\}$.



(a) $|DFT\{h\}|$ (b) $|\text{Error on } DFT\{h\}| \times 10^{-2}$

Fig. 5. Modulus of DFT$\{h\}$ and of error on DFT$\{h\}$ ($\times 10^{-2}$). The error is computed on the first 64 $\times$ 64 data points only. Maximum error $= 0.3 \times 10^{-2}$.



(a) $|H_k|$ (b) $|\text{Error on } H_k| \times 10^{-17}$

Fig. 6. Modulus of the numerical Fourier transform given by (33), with $\theta = 13$ and modulus of error on H_k ($\times 10^{-17}$). The error is computed on the full range of the 128 $\times$ 128 data points. Maximum error $= 0.7 \times 10^{-17}$.

tively. The formula of the analytical Fourier transform of this 2–D function is not shown here since it is several pages long.

Fig. 5(a) shows the DFT of h. The expected periodical behavior is evident. Fig. 5(b) shows the modulus of the error for the DFT relatively to the analytical Fourier transform. To be fair, this error must indeed be computed on the first $(N_1/2)\times(N_2/2)$ data points only since the DFT is periodical.

Fig. 6(a) shows the numerical Fourier transform of h computed with (33) for $\theta = 13$. It is clearly seen that this approximation behaves as the analytical Fourier transform and is not periodical. Fig. 6(b) shows the modulus of the error for this approximation relative to the analytical Fourier transform, computed this time on the full range of the $N_1 \times N_2$ data points. For comparison, one should note the vertical scale factors in

TABLE I
AVERAGE OF THE MODULI OF THE ERROR ON THE NUMERICAL FOURIER TRANSFORM OF FUNCTION (50) AND COMPUTED WITH (33) FOR DIFFERENT VALUES OF N AND θ. THE LAST LINE IS THE AVERAGE OF THE MODULUS OF THE ERROR FOR THE DFT OF THE SAME FUNCTION

	N				
θ	8	16	32	64	128
1	1×10^{-2}	1×10^{-3}	2×10^{-4}	2×10^{-5}	3×10^{-6}
3	3×10^{-1}	1×10^{-3}	9×10^{-6}	3×10^{-7}	1×10^{-8}
5		1×10^{-2}	8×10^{-7}	6×10^{-9}	5×10^{-11}
7			4×10^{-6}	1×10^{-10}	3×10^{-13}
9				3×10^{-12}	2×10^{-15}
11				8×10^{-14}	9×10^{-18}
13				2×10^{-15}	8×10^{-20}
DFT	3×10^{-2}	9×10^{-3}	3×10^{-3}	8×10^{-4}	2×10^{-4}

Figs. 5(b) and 6(b), which show that the error is reduced by at least 14 orders of magnitude.

Equation (33), which is used for Fig. 6, is used again, on the same function [see (50)] with different values of $N = N_1 = N_2$ and θ. The average of the moduli of the error on the numerical Fourier transform given by (33) relative to the exact analytical Fourier transform are shown in Table I in addition to the results obtained with the DFT.

We see that for small values of N (actually for $N = 8$), increasing θ does not improve accuracy because the optimum value for θ (θ_{opt}) is already reached. If N is slightly increased, θ_{opt} grows rapidly, and the error decreases dramatically. Note that for $N = 8$, $\theta_{opt} < 3$; for $N = 16$, $\theta_{opt} < 5$; for $N = 32$, $\theta_{opt} = 5$; and for $N \geq 64$, $\theta_{opt} \geq 13$. We also note that, to the exception of small values of N, the error from (33) is always much smaller than the error from the DFT, and for any θ, it decreases more rapidly with the increase of N than does the error from the DFT with the same increase in N.

VIII. CONCLUSION

In principle, when the Fourier integral of a function is analytically obtainable, it is possible to compute its Fourier transform exactly. In all other cases, one must resort to purely numerical techniques. In addition, the sampling process involves the Dirac comb, which destroys the continuity of a sampled analytical function. To make it manageable on a finite computer, the digitized function is defined as a finite number of points only. The usual tools to numerically compute the Fourier transform of a digitized function in one or more dimensions is the DFT, which is efficiently implemented as FFT algorithms. However, in many cases, the DFT is not an adequate numerical approximation of the Fourier transform. On the one hand, there is the analytical Fourier transform that is, most of the time, not known, and on the other hand, one has the efficient DFT, which computes an approximation of the Fourier transform without attempting to reduce the unavoidable ravages of the Dirac comb. The method presented in this contribution is between these two extremes; its *position* being determined by the value of θ, which is the order of the system. The method provides accurate approximations of the continuous Fourier transform, is no longer periodical, and computes the numerical Fourier transform in a broader frequency domain than the usual half-period of the DFT. The method gives accurate numerical partial derivatives of any order and the polynonial splines of any odd degree. The error can be easily computed by comparing the results of two successive odd orders. The time complexity is $N \log N$. The time complexity, relative to θ (independent of the time complexity related to N) is $O\left(\theta^2\right)$, whereas the accuracy increases exponentially with θ. Hence, the numerical accuracy increases much more rapidly than the computational cost of the proposed method. For smooth or rough functions, the proposed method performs better than the classical DFT and yields information that is not obtainable in principle with the DFT. The relative complexity of the proposed method is justified by the nature and the quality of the results it provides.

REFERENCES

[1] E. O. Brigham, *The Fast Fourier Transform*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[2] N. Morrison, *Introduction to Fourier Analysis*. New York: Wiley-Interscience, 1994.

[3] L. Marchildon, *Mécanique Quantique*. Bruxells, Belgium: De Boeck Université, 2000.

[4] C. Kittel, *Physique de l'état Solide*. Paris, France: Dunod Université, 1983.

[5] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 2001.

[6] J. Schutte, "New fast fourier transform algorithm for linear system analysis applied in molecular beam relaxation spectroscopy," *Rev. Sci. Instrum.*, vol. 52, no. 3, p. 400, 1981.

[7] S. Makinen, "New algorithm for the calculation of the fourier transform of discrete signals," *Rev. Sci. Instrum.*, vol. 53, no. 5, p. 627, 1982.

[8] S. Sorella and S. K. Ghosh, "Improved method for the discrete fast fourier transform," *Rev. Sci. Instrum.*, vol. 55, no. 8, p. 1348, 1984.

[9] M. Froeyen and L. Hellemans, "Improved algorithm for the discrete fourier transform," *Rev. Sci. Instrum.*, vol. 56, no. 12, p. 2325, 1985.

[10] J. D. Gaskill, *Linear Systems, Fourier Transform, and Optics*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[11] N. Beaudoin, Ph.D. dissertation, Université du Québec à Trois-Rivières, QC, Canada, 1999.

[12] E. Butkov, *Mathematical Physics*. Reading, MA: Addison-Wesley, 1968.

[13] E. Kreyszig, *Advanced Engineering Mathematics*. New York: Wiley, 1983.

[14] N. Beaudoin, "A high-accuracy mathematical and numerical method for fourier transform, integral, derivative, and polynomial splines of any order," *Can. J. Phys.*, vol. 76, no. 9, pp. 659–677, 1998.

[15] Plouffe's inverter. [Online]. Available: www.lacim.uqam.ca/pi/indexf.html, code A008 292

Normand Beaudoin received the B.S. degree in physics and the Ph.D. degree *cum laude* in biophysics in 1999 from the University of Quebec, Trois-Rivières, QC, Canada.

He is currently a post-doctoral fellow with the Department of Computer Science, The University of Western Ontario, London, ON, Canada. He is an electronics specialist in laboratory research. He designed, built, and operated complete systems of flash photolysis and time-resolved photoacoustic spectroscopy. He taught mathematics, physics, and electronics at different levels including university. He also acted as scientific director in a startup company for research equipment. His current research interests include analytical and discrete Fourier transforms, numerical analysis, spline polynomials, and numerical approximation. His doctoral research was supported by a grant from FCAR-Quebec. His post-doctoral research was funded through the *Natural Sciences and Engineering Research Council of Canada* (NSERC).

Dr. Beaudoin's Ph.D. thesis received the *Best Thesis of the Year* Award from the University of Quebec.

Steven S. Beauchemin received the M.Sc. and Ph.D. in computer science in 1991 and 1997, respectively. from the University of Western Ontario, London, ON, Canada.

He was an NSERC post-doctoral fellow at the University of Pennsylvania, Philadelphia, from 1997 to 1999. His research interests include optical flow, autonomous navigation, and robotics. He is currently and is now an Assistant Professor with the Department of Computer Science, The University of Western Ontario.

Dr. Beauchemin received the Governor General Academic Gold Medal of Canada in 1998.