A Generalized Poisson Summation Formula and its Application to Fast Linear Convolution

Jorge Martinez, Richard Heusdens, and Richard C. Hendriks

***Abstract*—In this letter, a generalized Fourier transform is introduced and its corresponding generalized Poisson summation formula is derived. For discrete, Fourier based, signal processing, this formula shows that a special form of control on the periodic repetitions that occur due to sampling in the reciprocal domain is possible. The present paper is focused on the derivation and analysis of a *weighted* circular convolution theorem. We use this specific result to compute linear convolutions in the generalized Fourier domain, without the need of zero-padding. This results in faster, more resource-efficient computations. Other techniques that achieve this have been introduced in the past using different approaches. The newly proposed theory however, constitutes a unifying framework to the methods previously published.**

***Index Terms*—Generalized Poisson summation formula, linear filtering, weighted circular convolution.**

I. Introduction

THE classical Poisson summation formula, expresses the fact that discretization in one domain implies periodicity in the reciprocal domain [1]. This periodicity comes in the form of a periodic summation of the signal values. In Fourier based digital signal processing (DSP), we have to deal with the periodic repetitions that appear due to the discrete nature of the domains where our signals are defined. In virtually all applications, the effect of overlapped repetitions is an issue that must be avoided, or at least controlled. In many cases the only way to achieve this is by increasing the sampling rate in the reciprocal domain [1], a costly operation in terms of memory and computational resources. In this letter we introduce a generalized Fourier domain (GFD) and derive its generalized Poisson summation formula. The newly proposed equation, relates the samples of the continuous generalized spectrum of a signal, with a geometrically *weighted* periodic extension of the signal. As it will be explained, this formula shows that a parametric form of control on the periodic repetitions that occur due to sampling in the GFD is possible, without the need to increase the sampling rate. This result has in principle many potential applications, the work presented here however, will be focused on the derivation and analysis of the *weighted* circular convolution theorem for the generalized discrete Fourier transform (GDFT).

Manuscript received May 25, 2011; accepted June 18, 2011. Date of publication June 30, 2011; date of current version July 11, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hsiao-Chun Wu.

The authors are with the Signal and Information Processing Lab (SIPL), Department of Mediamatics, Faculty of Electrical Engineering, Mathematics and Informatics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: J.A.MartinezCastaneda@TuDelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2011.2161078

For finite discrete-time signals of length $N \in \mathbb{Z}$, point-wise multiplication of the discrete spectra of the signals corresponds to circular convolution [1]. In order to perform a linear convolution (i.e., LTI filtering), the input signals are first zero-padded to at least length $2N-1$ and then transformed, increasing the number of computations needed [1]. In this letter we show how the newly proposed theorem can be used to perform linear convolutions in the GFD without the need of zero-padding, implementing the GDFT by means of the FFT. As we show, this results in a more efficient computation in terms of memory locations and operations needed. Although techniques to obtain a linear convolution without zero-padding have been developed in different contexts, e.g., [2] and [3], we show that the proposed theory constitutes a unifying framework to these approaches.

II. Poisson Summation Formula and Circular Convolution

Consider the Fourier transform for a signal $f(t) \in L^2(\mathbb{R})$ given by

$$F(\Omega) = \int_{-\infty}^{\infty} f(t) e^{-j\Omega t}\, dt \tag{1}$$

where $\Omega \in \mathbb{R}$ is the angular frequency variable, and $t \in \mathbb{R}$ represents time. If (1) exists, then we call F the spectrum of the signal f. The inverse transformation is given by [1]

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\Omega) e^{j\Omega t}\, d\Omega. \tag{2}$$

The Poisson summation formula then links the signal f to the samples of its spectrum F [1], i.e.,

$$\sum_{p\in\mathbb{Z}} f(t+pT_p) = \frac{1}{T_p} \sum_{k\in\mathbb{Z}} F\left(\frac{2\pi k}{T_p}\right) e^{j(2\pi k t/T_p)}. \tag{3}$$

From this equation we see that the repetitions of the periodic summation at the right-hand side will overlap if the length of the support of signal f is larger than T_p. Thus, given a fixed signal with finite-length (compact) support, the only way to avoid overlapping using (3) is to increase the value of T_p. This means a smaller spectral sampling interval $2\pi k/T_p$, which implies increased spectral sampling rate, and consequently, more memory and computational resources. This formula also allows us to easily understand the effect that frequency discretization has on the convolution product of two signals, say $f(t)$ and $h(t)$, when performed in the frequency-domain. That is

$$\sum_{p\in\mathbb{Z}} (f * h)(t+pT_p) = \frac{1}{T_p} \sum_{k\in\mathbb{Z}} (FH)\left(\frac{2\pi k}{T_p}\right) e^{j(2\pi k t/T_p)} \tag{4}$$

where $*$ denotes the linear convolution operator, and where we have made use of the convolution theorem [1]. The periodic repetitions that appear in the time domain will overlap when the

support of the convolution between f and h is larger than T_p, and consequently, it will be impossible to obtain the linear convolution result using (4). This is exactly how circular convolution is defined, as a periodic or cyclic version of linear convolution [1]. We will show that, by introducing a generalized Poisson summation formula, a special form of control on these repetitions can be obtained.

III. A Generalized Poisson Summation Formula

We now define a generalized Fourier transform for $f \in L^2(\mathbb{R})$, and $\alpha \in \mathbb{C}\backslash\{0\}$ as follows:

$$F_\alpha(\Omega) = \int_{-\infty}^{\infty} f(t) e^{\beta t} e^{-j\Omega t}\, dt \tag{5}$$

where $\beta = \log(\alpha)/T_p$. This is equivalent to the ordinary Fourier transform (if it can be defined) of the modulated signal $f(t)e^{\beta t}$. Note that (5) can be seen as a particular case of the Laplace transform, therefore for $|\Re\{\beta\}| > 0$, with $\Re\{\beta\}$ the real part of β, we have that (5) would not be defined on all $L^2(\mathbb{R})$, but only on the dense subset of all causal and anticausal (single-sided) functions [4]. Hence, for finite-length signals, operation (5) can always be defined, since these signals are special cases of single-sided functions. The inverse transformation follows as

$$f(t) = \frac{e^{-\beta t}}{2\pi} \int_{-\infty}^{\infty} F_\alpha(\Omega) e^{j\Omega t}\, d\Omega. \tag{6}$$

Evaluating F_α in (3), we obtain a generalization of the Poisson summation formula:

$$\sum_{p\in\mathbb{Z}} e^{\beta(t+pT_p)} f(t+pT_p) = \frac{1}{T_p} \sum_{k\in\mathbb{Z}} F_\alpha\left(\frac{2\pi k}{T_p}\right) e^{j(2\pi k t/T_p)}$$

so that

$$\sum_{p\in\mathbb{Z}} e^{\beta p T_p} f(t+pT_p) = \frac{e^{-\beta t}}{T_p} \sum_{k\in\mathbb{Z}} F_\alpha\left(\frac{2\pi k}{T_p}\right) e^{j(2\pi k t/T_p)}$$

or

$$\sum_{p\in\mathbb{Z}} \alpha^p f(t+pT_p) = \frac{e^{-\beta t}}{T_p} \sum_{k\in\mathbb{Z}} F_\alpha\left(\frac{2\pi k}{T_p}\right) e^{j(2\pi k t/T_p)} \tag{7}$$

since $\alpha = e^{\beta T_p}$. This equation, relates the samples of the continuous generalized spectrum of a signal, with a geometrically *weighted* periodic extension of the signal. Therefore, an extra form of control can be obtained over the repetitions via the parameter α. In analogy, let us now define the generalized discrete Fourier transform (GDFT) for finite length signals $x(n)$, $n = \{0, \ldots, N-1\}$, $k = \{0, \ldots, N-1\}$, as follows:

$$X_\alpha(k) = \sum_{n=0}^{N-1} x(n) e^{\beta n} e^{-j(2\pi/N)kn} \tag{8}$$

where $\beta = \log(\alpha)/N$. The inverse GDFT is given by

$$x(n) = \frac{e^{-\beta n}}{N} \sum_{k=0}^{N-1} X_\alpha(k) e^{j(2\pi/N)kn}. \tag{9}$$

In this case, the generalized Poisson summation formula takes the following form:

$$\sum_{p\in\mathbb{Z}} \alpha^p x(n+pN) = \frac{e^{-\beta n}}{N} \sum_{k=0}^{N-1} X_\alpha(k) e^{j(2\pi k n/N)} \tag{10}$$

since $\alpha = e^{\beta N}$. Taking $Z_\alpha(k) = (X_\alpha Y_\alpha)(k)$ in (10), and applying the convolution theorem we obtain

$$z(n) = \sum_{p\in Z} \alpha^p (x * y)(n+pN). \tag{11}$$

For $\alpha = 1$, this corresponds to the standard Fourier case as already shown for the continuous time case in (4). From (11) we see how circular convolution is related to linear convolution. Although the original sequences x and y are of length N, its convolution product is however of length $2N-1$. Then it follows that $z(n)$ for $n = \{0, \ldots, N-1\}$, represents the linear convolution of x and y plus the last $N-1$ terms of one overlapped repetition. The classical approach of zero-padding avoids this situation by making N large enough, so that the periodic repetitions in (11) are sufficiently appart to avoid overlapping. In Section IV we will derive the *weighted* circular convolution theorem for the GDFT pair (8) and (9). This theorem will be used to perform linear convolutions in the GFD without the need of zero padding, taking advantage of the weighting effect that factor α^p has on the repetitions of the signal as expressed by the generalized Poisson summation formula (10).

IV. Weighted Circular Convolution Theorem

We have the following result.

Proposition 1: Let $x(m)$, $m \in \{0, \ldots, N-1\}$ and $y(l)$, $l \in \{0, \ldots, N-1\}$ be the signals to be convolved. For $n \in \{0, \ldots, N-1\}$, we have

$$\frac{e^{-\beta n}}{N} \sum_{k=0}^{N-1} X_\alpha(k) Y_\alpha(k) e^{j(2\pi/N)nk} = \tag{12}$$

$$\sum_{m=0}^{n} x(m) y(n-m) + \alpha \sum_{m=n+1}^{N-1} x(m) y(N+n-m) \tag{13}$$

where the left hand summation in (13) represents the contribution of N linear convolution terms, and the right hand summation the contribution of N circular convolution terms (which are in fact the last terms of the linear convolution). The factor α effectively weights the amount of circular convolution that is obtained. Thus, given two discrete-time, finite length signals, $x(n)$ and $y(n)$, point-wise multiplication of their generalized discrete spectra, $X_\alpha(k)$ and $Y_\alpha(k)$, corresponds to a *weighted* circular convolution in the time-domain. The proof is given in the Appendix.

V. Linear Convolution Using the GDFT

To compute the weighted circular convolution operation given by (12), the generalized spectrum of the signals to be convolved is obtained using the GDFT (8). Note that this operation can be implemented, by taking the FFT of the modulated signal, $x(n)e^{\beta n}$ with $\beta = \log(\alpha)/N$. The inverse transform (9) can be obtained by multiplying the IFFT of the generalized spectrum with the inverse function $e^{-\beta n}$. For finite energy signals, the theory of Laplace integrals [4], ensures

the existence of the GDFT for any value of $\alpha \in \mathbb{C}\backslash\{0\}$, as previously mentioned in Section III. Let us now analyze a special case of the weighted convolution theorem, setting $\alpha = j$, with j the imaginary unit. For real signals x and y, the resulting signal obtained after applying (12) becomes complex (its spectrum not being Hermitian symmetric). By a careful inspection of (13), it is noted that as a result, the first N values of the linear convolution are stored in the real part of the signal, and the remaining $N-1$ terms are perfectly preserved in its imaginary part. Hence, concatenating the real and imaginary parts, a $2N-1$ point, error-free convolution can be obtained in the GFD without the need of zero-padding. If x and y are complex, then two transforms are necessary to obtain a linear convolution, i.e., for $\alpha = \pm j$. Two show this, let us denote by z_j the result of (12) setting $\alpha = j$, and by z_{-j} the result setting $\alpha = -j$. Further denoting by $z_{\Re}$ and $z_{\Im}$ the real and imaginary parts respectively, of the linear convolution between x and y, then by (13) we obtain

$$z_j(n) = z_{\Re}(n) + jz_{\Im}(n) + j(z_{\Re}(N+n) + jz_{\Im}(N+n))$$
$$z_{-j}(n) = z_{\Re}(n) + jz_{\Im}(n) - j(z_{\Re}(N+n) + jz_{\Im}(N+n)).$$

Clearly, the first N samples of the linear convolution are obtained by $(z_j(n) + z_{-j}(n))/2$ and the remaining samples by $(z_j(n) - z_{-j}(n))/(2j)$.

If one takes $\alpha \in \mathbb{R}$, with $0 < \alpha \ll 1$ in (12), then not an exact, but an approximation of linear convolution is obtained. In fact, only the first N values of the linear convolution are approximated, since the circular convolution terms in (13) are attenuated by a very small factor. The approximation can be made as accurate as possible (making α as small as possible), up to the limits imposed by finite word-length arithmetic. This holds for both complex or real inputs. In this case for real time-domain signals, Hermitian symmetry still holds. Therefore, the point-wise multiplication of the spectra can be performed over the first $N/2+1$ DFT coefficients if N is even, and $(N-1)/2+1$ coefficients if N is odd.

Two techniques to obtain a linear convolution without the need of zero-padding have been previously introduced in [2] and [3], following different approaches. By setting $\alpha = j$ we directly obtain what the authors in [2] call the right-angle circular convolution (RCC), and for $\alpha = -j$ the left-angle circular convolution (LCC). Further in [3], the authors propose to multiply the input sequences by a scaling factor $s^n \in \mathbb{R}$. This is equivalent to use GDFTs with parameter $\alpha = s^N$ in our proposed derivation. Hence, these techniques can be seen as particular cases of the weighted circular convolution theorem. The newly proposed theory constitutes therefore a generalization that not only provides a unifying framework for previous methods, but also allows for a deeper insight into the problem. This leads to the formulation of new applications, such as the approximation here proposed setting $0 < \alpha \ll 1$.

VI. Computational Complexity Analysis and Experiments

In real applications, one would like to use the convolution theorem together with the FFT in order to perform linear convolutions in the frequency-domain with reduced complexity [1]. In some of these applications, the goal is to convolve two finite length signals without worrying about causality or delay (like in the multiplication of two long polynomials). In many real-time DSP problems however, the goal is to filter a long signal, using a finite length (in general much shorter) LTI filter. This is achieved in a block-by-block basis, performing shorter convolutions at each step using well known approaches [1]. In all cases, the operation at hand can be seen as the linear convolution of a length-N signal, say $x(n)$ with a length-M filter, say $h(n)$, where without loss of generality the assumption $M \leq N$ is made. The result of the convolution is therefore of size $N+M-1$.

An exact complexity analysis is a lengthy and complex task. State-of-the-art FFT algorithms deliver quite inhomogeneous (although asymptotically "equivalent") performance [5]. In each specific case, advantageous conditions can be exploited by the algorithms. The complexity then, becomes a function of the FFT length, the signals class, symmetries present in the input and output signals, hardware architecture, etc [5].

Let us now analyze the total number of multiplications needed as a reasonable basis for comparing the computational complexity. For simplicity, real-valued signals are considered. Thus, given $x(n)$ and $h(n)$, the classical frequency-domain approach requires to pad both signals to at least a size of $N+M-1$. Moreover, we have that $1 < M \leq N$, hence the complexity can be expressed as a function of N and the ratio, $\lambda = (M-1)/N$, where $1/N \leq \lambda < 1$. Then, we have that three (I)FFTs are needed to transform the signals to the frequency-domain and back to the time-domain. For each of these, the FFT (or IFFT) requires approximately $N(\lambda+1)\log_2(N(\lambda+1))$ real multiplications [5]. The point-wise multiplication of the spectra requires $2N(\lambda+1)+1$ multiplications, since Hermitian symmetry can be exploited [1]. Hence, the total number of multiplications performed is $3N(\lambda+1)\log_2(N(\lambda+1)) + 2N(\lambda+1) + 1$.

For the GFD method setting $\alpha = j$, only the filter, $h(n)$ must be zero-padded to a size N. To implement a GFFT or IGFFT as proposed in Section V, the (de)modulation of the signals require $2N$ multiplications each. The FFTs require approximately $N\log_2(N)$ multiplications each. The point-wise multiplication of the generalized spectra requires $8N$ multiplications. The total number of multiplications needed is thus $3N\log_2(N) + 14N$. From here, it is clear that the complexity ratio for the GDFT based convolution to the frequency-domain based convolution is approximately, $(1/(1+\lambda))(\log_2(N) + 14/3)/(\log_2(N(1+\lambda)) + 2/3)$. For example setting $N = 256$, in the limiting case $\lambda = 1/N$, which implies $M = 2$, the frequency-domain implementation shows to be about 1.4 times faster than the GDFT approach. This is caused by the very short length of the filter. The overhead produced by the extra multiplications needed in the GDFT method is not compensated. On the other hand, for $M = N$, we have that the complexity ratio can be expressed as $(1/2)(\log_2(N) + 14/3)/(\log_2(N) + 5/3)$. In this case the GDFT based convolution is about 1.5 times faster than the classical frequency-domain approach.

For the GDFT method setting $\alpha \ll 1$, the complexity ratio is approximately $(1/(1+\lambda))(\log_2(N) + 2)/(\log_2(N(1+\lambda)) + 2/3)$, since Hermitian symmetry can be exploited and the

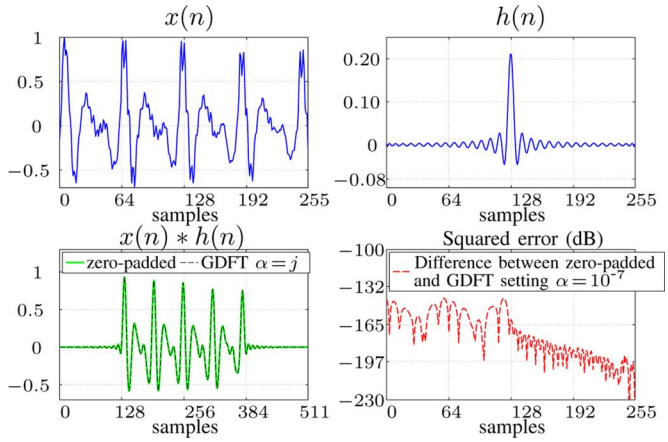

Fig. 1. LTI filtering of a speech signal frame $x(n)$ and a low-pass filter $h(n)$, performed in the spectral domain for $N = 256$. The standard frequency-domain approach using zero-padding to $2N$ samples and the novel GDFT method are compared.

(de)modulation of the signals require a total of $3N$ multiplications. In this case, for $N = 256$ the limiting case $\lambda = 1/N$ shows that the GDFT method is still slightly more complex than the frequency-domain algorithm. For $M = N$ on the other hand, the GDFT method is about two times faster.

From these results we see that for the GDFT approach, the most efficient choice for the filter size is $M = N$. This avoids trivial operations on the zeros added to the filter to be computed, giving the maximum performance increase with respect to the standard frequency-domain method, since the signal and the filter do not need to be zero-padded. In this case, it is also easy to see that the new approach can be implemented using roughly half the amount of memory that it is needed for the standard frequency-domain based convolution.

In Fig. 1 an example of LTI filtering of a speech signal frame $x(n)$, using a low-pass filter $h(n)$ for $N = 256$ is given. Both the classical frequency-domain approach using zero-padding to $2N$ samples and the novel GDFT approach (12) for $\alpha = j$ and $\alpha = 10^{-7}$ are compared. After repeating the experiment 10^6 times, the classical approach took 62.7 s to complete. The GDFT approach took 45.4 s to complete for the case $\alpha = j$, and 32.2 s for the case $\alpha = 10^{-7}$, showing the advantage of the GDFT method in agreement with the analysis conducted above. For the case $\alpha = 10^{-7}$ the result is not exact, and only the first N values of the operation can be compared. A plot in dB of the square error between both approaches shows that a very accurate approximation of LTI filtering is obtained. All tests were performed using double floating-point precision in MATLAB® on a standard desktop PC.

VII. Conclusions

In this work, a generalized Poisson summation formula has been proposed. It conceptually allows us to obtain a special form of control on the periodic repetitions that occur due to sampling in the reciprocal domain. Using this result, a *weighted* circular convolution theorem for the GDFT is derived, which is used to perform efficient, non zero-padded linear convolutions. Altogether, these results have applications which range from simple multiplication of long polynomials, to Wiener filtering, adaptive filtering, near-field beamforming, and many more.

Appendix
Proof of Proposition 1

Proof: Let

$$\begin{aligned} z(n) &= \frac{e^{-\beta n}}{N} \sum_{k=0}^{N-1} X_\alpha(k) Y_\alpha(k) e^{j(2\pi/N)nk} \\ &= \frac{e^{-\beta n}}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x(m) e^{\beta m} e^{-j(2\pi/N)mk} \right) \\ &\quad \times \left(\sum_{l=0}^{N-1} y(l) e^{\beta l} e^{-j(2\pi/N)lk} \right) e^{j(2\pi/N)nk} \\ &= \frac{e^{-\beta n}}{N} \sum_{m=0}^{N-1} x(m) e^{\beta m} \sum_{l=0}^{N-1} y(l) e^{\beta l} \\ &\quad \times \left(\sum_{k=0}^{N-1} e^{j(2\pi/N)k(n-m-l)} \right). \end{aligned}$$

For $p \in \mathbb{Z}$ we have [1]

$$\sum_{k=0}^{N-1} e^{j(2\pi/N)k(n-m-l)} = \begin{cases} N, & l = n - m + pN \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Therefore, we have that $l = (n-m+pN) \in \{0, \ldots, N-1\}$, and $p = -\lfloor (n-m)/N \rfloor$, where $\lfloor x \rfloor$ is the nearest integer $\leq x$. Further using (14) we obtain

$$\begin{aligned} z(n) &= \sum_{m=0}^{N-1} x(m) y(n-m+pN) e^{\beta m} e^{\beta(n-m+pN)} e^{-\beta n} \\ &= \sum_{m=0}^{N-1} x(m) y(n-m+pN) e^{\beta pN} \\ &= \sum_{m=0}^{N-1} x(m) \alpha^p y(n-m+pN). \end{aligned} \tag{15}$$

Since the output signal is of length N, we have that $(n-m) \in \{-N+1, \ldots, N-1\}$, and thus $p \in \{0, 1\}$, so that (15) can be rewritten as

$$z(n) = \sum_{m=0}^{n} x(m) y(n-m) + \alpha \sum_{m=n+1}^{N-1} x(m) y(N+n-m)$$

■

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing, Principles, Algorithms and Applications*, 4th ed. Upper Saddle River, NJ: Prentice-Hall, 2007.

[2] C. Radhakrishnan and W. Jenkins, "Modified discrete fourier transforms for fast convolution and adaptive filtering," in *Proc. ISCAS*, May-Jun. 2010, pp. 1611–1614.

[3] Z. Babic and D. Mandic, "A fast algorithm for linear convolution of discrete time signals," in *5th Int. Conf. Telecommunications in Modern Satellite, Cable and Broadcasting Service*, Sep. 2001, vol. 2, pp. 595–598.

[4] K. B. Wolf, *Integral Transforms in Science and Engineering*. New York: Plenum, 1979.

[5] M. Frigo and S. Johnson, "The design and implementation of fftw3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.