16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

CALCULATION OF AMBIGUITY FUNCTIONS WITH FERMAT NUMBER TRANSFORM

K. Minaoui[1,2], *T. Chonavel*[1], *B. Nsiri*[2] *and D. Aboutajdine*[2]

[1]SC Department, Telecom Bretagne, CNRS (UMR 3192)
Technopôle Brest Iroise CS 83818, 29238, Brest Cedex 3, France
[2]GSCM_LRIT, Faculty of Sciences Rabat, Mohammed V-Agdal University
B.P. 1014, Rabat, Morocco
email:{khalid.minaoui, thierry.chonavel, benayad.nsiri}@telecom-bretagne.eu, aboutaj@ieee.org

ABSTRACT

This paper deals with fast calculation of cross-ambiguity functions. The approach that we develop is based on Gauss-Legendre quadrature associated with Fermat Number Transform. For fixed number of quadrature nodes, these nodes are approximated by their closest neighbors on a regular sampling grid. This enables Gauss quadrature good approximation while preserving the convolution structure of the grid quantized quadrature. The interest of preserving the convolution structure of the cross ambiguity terms in the corresponding discretized problem lies in the possibility of using fast transform Fourier-like algorithms. In a digital processing context, Number Theoretic Transforms (NTT) in finite fields of order a Fermat number are known to be particularly well suited to achieve convolution at very low computational cost. The contribution of this paper lies in the association of both powerful concepts of Gauss quadrature and NTT to realize fast convolution, and in particular fast cross-ambiguity calculation. Simulations are carried out to illustrate calculation of a few standard radar waveforms ambiguity functions.

1. INTRODUCTION

Convolution is probably the most standard operator in signal processing and Fast Fourier Transform (FFT) probably the most famous digital signal processing algorithm, partly because it enables fast convolution computations. On the other hand, cross-ambiguity calculations is a time frequency representation of much importance in applications such as radar [1], the expression of which can be seen as a set of convolutions; Letting $r(t)$ denote an observed signal and $u(t)$ a reference waveform, the cross ambiguity function, as defined in [1, 2, 3, 4], is given by

$$\chi(\tau,f) = \int_{-\infty}^{\infty} r(t)u^*(t-\tau)\exp(j2\pi ft)dt, \qquad (1)$$

where τ and f are the time delay and frequency parameters respectively. Clearly, for fixed f, $\tau \rightarrow X(\tau,f)$ is a convolution operation. More precisely, $\chi(\tau,f)$ can be seen as the output at time τ of $r(t)$ demodulated at frequency f and passed into the match filter associated with waveform $u(t)$. In particular, when $r(t)=u(t)$, Eq. (1) is called the ambiguity function of the waveform $u(t)$.

The ambiguity function plays an important role in radar detection by supplying information about the radar waveform time and frequency resolution capability and in studying performance of targets distance and speed estimators, since the Cramer-Rao bound for these parameters can be derived from the ambiguity function [1].

The cross-ambiguity between an echoed radar signal and the emitted waveform $u(t)$ shows attenuated and Doppler-and-delay shifted versions of the ambiguity function of $u(t)$ that represent illuminated targets contributions. Of course, for a given radar waveform one may derive specific fast algorithms for targets detection and parameters estimation (see for instance [5] for the case of linear chirp waveforms). But, working with $\chi(\tau,f)$ can be seen as a general way to perform this task for any kind of waveform. In addition, working with function $\chi(\tau,f)$ enables recovery of possibly very closely located sources in the time-frequency domain [6].

The main drawback in using $\chi(\tau,f)$ in a real time processing context is that its calculation requires high computational effort. Then FFT based algorithms have been proposed to calculate precise or decimated versions of the ambiguity function on a regular sampling grid (see [7] and reference therein). All these approaches for calculating cross-ambiguity functions start from a discretized version of the convolution equation. In other words, they consider a Riemann sum approximation of the convolution equation.

In this paper, we do not address convolutions calculation directly through regular sampling. Indeed, standard Riemann integral approximations or more general Newton-Cotes quadrature formulas calculate integrals through regular sampling, while more efficient Gauss techniques lead to quadrature formulas that are calculated from irregularly spaced nodes [8, 9]; For a fixed number of nodes, Gauss formula yield much lower integral approximation error than their Newton counterparts.

However, irregular sampling does not preserve convolutional structure of the initial integral expression. To cope with this drawback we consider approximate Gauss nodes, chosen as closest neighbours of the Gauss nodes number on a regular sampling grid. In order to ensure little error resulting from this quantization, the grid has more points than the number of nodes. Quadrature weights are updated accordingly. In this new representation, $\chi(\tau,f)$ is approximated as a weighted, regularly sampled quadrature expression with many zero terms. This enables using FFT-like convolution algorithms if working from direct discretization of convolution integral.

With a view to digital processors implementation, we add a further step for complexity reduction of the algorithm. This is achieved by replacing standard FFT by Number Theoretic Transforms (NTT) that are discrete Fourier like operations that transform cyclic convolution operations on finite rings of integers into multiplications. In particular, it has been shown in [10] that the case of rings of integers modulo a Fermat number, that is, a number of the form $2^{2^t}+1$ ($t \in \mathbb{N}$), is of particular interest. Then, this transform is called a Fermat Number Transform (FNT). Using FNT for calculating convolutions leads to operations on a finite ring of integers, avoiding thus floating point calculations. In addition, the number of operations is small and multiplications simply amount to registers shifts.

The rest of this paper is organized as follows: in section 2 we recall basics about Gauss quadrature and Legendre polynomials. In section 3, we consider its approximation on a grid. NTT and FNT are recalled in section 4. Application of the Gauss-Legendre quadrature and FNT for ambiguity functions calculation is presented in section 5 where examples of standard radar waveforms ambiguity functions calculation are presented.

2. GAUSS-LEGENDRE QUADRATURE

The Gauss-Legendre quadrature method is a member of the Gaussian quadrature family [8]. Let us consider an integral expression over interval $[0,1]$ (generalization to more general finite intervals is straightforward), of the form

$$I = \int_0^1 g(x)dx. \tag{2}$$

For a given weight function $w(x)$ on $[0,1]$, we define the inner product of g and h as

$$< g,h >= \int_0^1 w(x)g(x)h(x)dx. \tag{3}$$

For this inner product, orthogonal polynomials sequences $(p_i(x))_{i\in\mathbb{N}}$ are sequences of polynomials with degree of p_i equal to i that satisfy $< p_i, p_j >= \delta_{i,j}$, where $\delta_{i,j}$ is Kronecker's delta function.

Legendre polynomials form such an orthogonal family for weight function $w(x) = 1$, for all x. They can be defined in several ways. In particular. In particular, they satisfy the following recurrence relationships:

$$p_0(x) \quad = \quad 1 \qquad p_1(x) = x \tag{4}$$

$$(n+1)P_{n+1}(x) \quad = \quad (2n+1)xp_n(x) - np_{n-1}(x), \quad n \geq 1.$$

Alternatively, they can be described by Rodrigues' representation:

$$p_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}((x^2-1)^n). \tag{5}$$

For a fixed positive integer m, it is possible to compute a quadrature formula for I (see Eq. (2)), that is, a weighted discrete sum approximation of the integral, of the form

$$\sum_{i=1,m} w_i g(x_i) \tag{6}$$

such that this approximation matches exactly I whenever g is a polynomial of degree at most $2m-1$. Then, quadrature is said to be of order $2m$. In addition, if g is a more general function, the quadrature approximation error is of the form $g^{(2m)}(\xi)/((2m)!p_{m,m})$ for some $\xi \in]0,1[$, provided $g^{(2m)}$ exists, and with $p_{m,m}$ the leading coefficient of $p_m(x)$.

Nodes x_i must be chosen as the roots of $p_m(x)$, that can be proved to be distinct, and coefficients w_i are solutions of equations

$$\int_0^1 x^k dx = \sum_{i=1,m} w_i x_i^k, \quad k = 0,...,m-1. \tag{7}$$

Gauss quadrature has become very popular with development of computers. Efficient algorithms for computing quadrature rules can be found in the literature [11]. Practical implementations are also available for many programming languages (see for instance [12], function gauss.m, for a MATLAB implementation).

3. APPROXIMATE GAUSS QUADRATURE

As discussed in the introduction, irregular sampling node spacing of Gauss quadrature does not preserves convolution structure in quadrature formula of a convolution integral. This drawback can be overcome by approximating nodes of the quadrature formula (that is zeroes of Legendre polynomial) by their closest neighbors on a regular grid. Thus, nodes $(x_i)_{i=1,m}$ are replaced by new ones, located on the grid at locations $(\hat{x}_i)_{i=1,m}$. Of course, quadrature order is no longer $2m$, but, at least, it can be made equal to m by slightly moving coefficients w_i. Indeed, once the $(\hat{x}_i)_{i=1,m}$ are fixed, it suffices to replace the $(w_i)_{i=m}$ by coefficients $(\hat{w}_i)_{i=m}$ such that

$$\int_0^1 x^k dx = \sum_{i=1,m} \hat{w}_i \hat{x}_i^k, \quad k = 0,\dots,m-1. \tag{8}$$

The smaller the grid step, the more accurate the Gauss quadrature approximation supplied by node-weight pairs $(\hat{x}_i, \hat{w}_i)_{i=1,m}$ will be.

On another hand we have checked that using quantized Gauss-Legendre nodes without updating corresponding weights leads to poor error performance, especially for low degree polynomials integration.

Letting $\hat{x}_i = k_i\Delta$, where Δ is the grid stepsize, the quadrature formula for convolution integral $\int_0^1 g(x-u)h(u)du$ with $x = t_0\Delta$ leads to a discrete convolution equation of the form

$$\sum_{t=0}^{1/\Delta} \hat{g}(t_0 - t)\hat{h}(t), \tag{9}$$

where $\hat{g}(t_0-t) = g((t_0-t)\Delta)$ and either $\hat{h}(t) = \hat{w}_i h(k_i\Delta)$ if $t = k_i$ for some $i \in \{1,\dots,m\}$ or $\hat{h}(t) = 0$ otherwise. Thus, the FFT of $\hat{h}$ can be achieved with reduced computional effort since it has many zero entries.

When considering function $\chi(\tau, f)$, since $r(t)$ and $u(t)$ can have complex values, the convolution can be splitted into a sum of four integrals each representing a convolution where $\hat{g}(t)$ is the real or imaginary part of the sampled $u^*(x)$, while $\hat{h}(t)$ corresponds to the weighted real or imaginary part of $r(x)\exp(2i\pi fx)$. Note that if targets contributions are such that large delays should be considered, then $\hat{h}(t)$ should be much longer than $\hat{g}(t)$ which makes even more interesting the presence of many zeros in sequence $\hat{h}$.

Fig.1. shows the quadrature error for polynomials $(x^k)_{k=0,20}$ integrated on $[0,1]$ as a function of polynomial degrees (Fig.1-a) and of the number of quadrature nodes (Fig.1-b) respectively.

Here, the exact Gauss-Legendre (GL) quadrature is achieved with $m = 8$ nodes, which yields exact quadrature formulas for polynomial degrees up to 15. The $m = 8$ nodes Newton-Cotes (NC) quadrature is exact only for polynomial degrees up to 7. The grid approximated nodes of the quantized GL that we introduced here above also supplies exact quadrature for polynomial degrees up to 7, thanks to weights updating (weights $(\hat{w}_i)_{i=1,m}$). Only true GL quadrature achieves negligible error as degree grows form 8 to 20 (Fig 1-a). However, the error of our quantized GL quadrature remains much lower than NC quadrature (about for times smaller for $\int_0^1 x^{20}dx$), while the sampling grid size is 32 points.

In Fig.1-b, we compare quadrature formulas for $\int_0^1 x^{15}dx$. The error is drawn for the three methods as a function of the number of quadrature nodes, ranging from 1 to 8. Here again the quantization grid has 32 regularly spaced points.

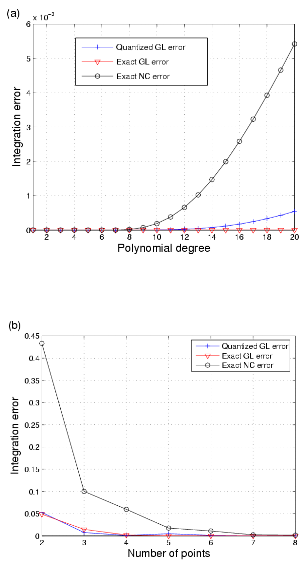

Figure 1: Integration error as a function of a) polynomial degree b) number of points, for exact and quantized Gauss-Legendre (GL) and Newton-Cotes (NC) quadrature. Quantization grid has 16 points. In b), integrated polynomial order is 15.

4. NUMBER THEORETICAL TRANSFORMS

4.1 Modular arithmetic reminders

Let us begin with some reminders of modular arithmetic. Two integers a and b are said to be congruent modulo M if

$$a = b + kM, \tag{10}$$

where k is some integer. This is written as $a = b\ (mod\ M)$. All integers are congruent modulo M to some integer in the finite set $(0,1,2,...,M-1)$ which is called the set of integers modulo M and denoted by $\mathbb{Z}_M$ (or $\mathbb{Z}/M\mathbb{Z}$). $\mathbb{Z}_M$ is a ring. All nonzero integers of $\mathbb{Z}_M$ have an inverse for multiplication if and only if M is prime, in which case $\mathbb{Z}_M$ is a finite field.

Let α denote an integer relatively prime to M. If N is the smallest positive integer such that

$$\alpha^N = 1 \quad (mod\ M), \tag{11}$$

then α is said to be a root of unity of order N. This concept is of particular interest when α is a generator of the field, that is, when

$$\mathbb{Z}_M = \{\alpha^i; i = 0,1,\ldots,M-2\} \cup \{0\}. \tag{12}$$

In such a case, α is called a primitive root. Clearly, the interest of this property lies in simplified products calculation: if $b = \alpha^i$ and $c = \alpha^j$, then $bc = \alpha^{i+j}\ (mod\ M)$. In particular, if $\alpha = 2$ is a generator, multiplications in $\mathbb{Z}_M$ simply amount to register shifts. Further elements of number theory basics can be found in books [13, 14] for instance.

4.2 Number Theoretic Transforms

Let α be a root of unity of order N in $\mathbb{Z}_M$ and assume that N has an inverse in $\mathbb{Z}_M$. Then, the NTT and its inverse (INTT) for a sequence $(x_k)_{k=0,N-1}$ of $\mathbb{Z}_M$ are defined by

$$\begin{aligned} X_k &= \textstyle\sum_{n=0}^{N-1} x_n \alpha^{nk} \quad (mod\ M), & k = 0,1,...,N-1; \\ x_n &= N^{-1} \textstyle\sum_{k=0}^{N-1} X_k \alpha^{-nk}\ (mod\ M), & n = 0,1,...,N-1. \end{aligned} \tag{13}$$

These transforms implement cyclic convolution. Letting $(x_n)_{n=0,N-1}$ and $(h_n)_{n=0,N-1}$ denote sequences of $\mathbb{Z}_M$, and $(y_n)_{n=0,N-1}$ their cross-correlation defined by [15]

$$y_n = \sum_{m=0}^{n-1} x_m h_{n+m} \quad (mod\ M), \qquad n = 0,1,...,N-1, \tag{14}$$

y_n can also be calculated as

$$y_n = \text{INTT}[X_{N-k} \times H_k \quad (modM)]. \tag{15}$$

In order to avoid congruence ambiguity in practical calculations, one should ensure that for any n, in Eq. (14) we have $|\sum_{m=0}^{n-1} x_m h_{n+m}| < M/2$. This is true whence the following condition is satisfied [10, 15]

$$|y_n| \leq \max_k |x_k| \sum_{n=0}^{N-1} |h_n| \leq \frac{M}{2}. \tag{16}$$

It seems that using NTT for convolution has been introduced by Rader in [16, 17], where Mersenne numbers and Fermat numbers respectively are considered for M. Mersenne numbers are of the form $M = 2^p - 1$, where p is prime, while Fermat numbers are of the form $M = 2^{2^t} + 1$.

The problem of a good choice for M is addressed in [10]. The authors show that such a good choice should respect the following conditions. First, there must exist a root of unity of order N such that the prime decomposition of N has many factors (and preferably is a power of 2) for a fast FFT-like algorithm to exist. In addition, N should be large enough to enable calculation of large sequences convolution. Furthermore, the binary representation of α should be simple enough to enable fast calculation of powers of α. Idealy, α should be a power of two. And the binary representation of M should be simple too, to ensure easy modulo M operations.

In [10], the authors have shown that this leads to select M as a Fermat number. Then, the particular choice $\alpha = 2$ is of order $N = 2^{t+1}$ while $\alpha = 2^{2^t-2}(2^{2^{t-1}} - 1)$ $(\alpha^2 = 2)$ has maximum possible order $N = 2^{t+2}$.

Note that applying the Fast Fermat Number Transform (FFNT) requires about $N \log_2 N$ simple operations that are bits shifts and additions but not multiplication, while FFT requires about $N \log_2 N$ multiplications. About 30% of computational effort can be spared for convolving two sequences if FFNT implementation instead of FFT is considered [10].

5. APPLICATION TO AMBIGUITY FUNCTION COMPUTATION

As discussed in the introduction, the radar ambiguity function is a tool for radar designers and for radar detection and estimation. For targets detection, one looks for time and delay shifted versions of the ambiguity function in the cross-ambiguity representation calculated between a reflected radar signal and the emitted waveform.

Restricting here our interest to the ambiguity function calculation, we consider three particular standard waveforms, namely a single pulse, a Linear Frequency Modulation Waveform (LFM) and a pulse train. Ambiguity functions of these waveforms can be calculated in closed form; They are given respectively by

$$\begin{aligned} u_1(t) &= \tfrac{1}{\sqrt{d}}\mathrm{rect}\big(\tfrac{t}{d}\big) \\ u_2(t) &= \tfrac{1}{\sqrt{d}}\mathrm{rect}\big(\tfrac{t}{d}\exp(2i\pi\beta t^2)\big) \\ u_3(t) &= \tfrac{1}{\sqrt{N}}\textstyle\sum_{k=0}^{N-1} u_1(t-kT) \end{aligned} \qquad (17)$$

where

$$\mathrm{rect}\big(\frac{t}{d}\big) = \begin{cases} 1 & -\frac{d}{2} \le t \le \frac{d}{2} \\ 0 & \text{otherwise.} \end{cases}$$

and d is the pulse duration. The corresponding ambiguity functions are expressed as follows:

$$\begin{aligned} \chi_1(\tau,f) &= \left|\big(1-\tfrac{|\tau|}{d}\big)\tfrac{\sin\pi f(d-|\tau|)}{\pi f(d-|\tau|)}\right|, \qquad |\tau| \le d \\ \chi_2(\tau,f) &= \left|\big(1-\tfrac{|\tau|}{d}\big)\tfrac{\sin\pi d(2\beta\tau+f)\left(1-\frac{|\tau|}{d}\right)}{\pi d(2\beta\tau+f)\left(1-\frac{|\tau|}{d}\right)}\right|, \quad |\tau| \le d \\ \chi_3(\tau,f) &= \textstyle\sum_{k=-N+1)}^{N-1} |\chi_1(\tau-kT,f)| \times \left|\tfrac{\sin\pi f(N-|k|T)}{N\pi f T}\right| \end{aligned} \qquad (18)$$

Parameter values are $d = 1s$ for the single pulse and LFM waveforms and $d = 0.2s$ for the pulse train, $\beta = 2.5Hz\,s^{-1}$, $N = 4$ and $T = 1s$. These ambiguity functions are plotted in Fig.2. We computed $\chi_k(\tau,f)$ $(k = 1,2,3)$ by using node quantized Gauss quadrature and FNT as described in previous sections. Letting $\hat{\chi}_k(\tau,f)$ $(k = 1,2,3)$ denote the calculated approximation, we shall consider the performance indices given by the mean square error, that is defined by

$$MSE_k \quad = \mathrm{mean}_{(\tau,f)}\left(|\hat{\chi}_k(\tau,f) - \chi_k(\tau,f)|^2\right). \qquad (19)$$

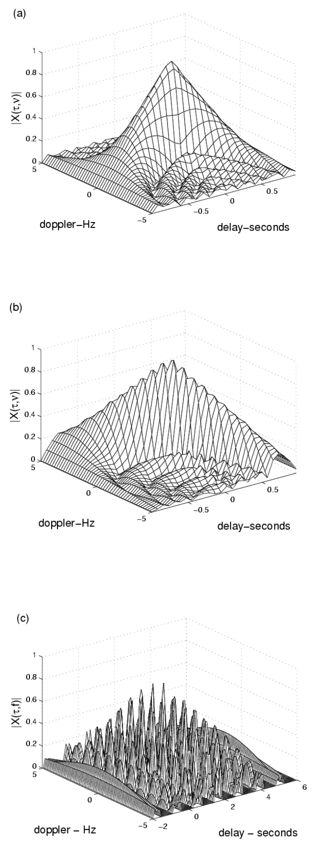

Figure 2: Ambiguity functions for a) a single pulse, b) a Linear Frequency Modulation Waveforms (LFM) c) a pulse train.

Figure 3 shows mean square error MSE_k $(k = 1,2,3)$ as a function of the number of quadrature points. Here, $M = 2^{2^4} + 1 = 65537$. Quadrature is achieved by slicing the convolution interval into 8 sub-intervals for single pulse and LFM waveforms and 64 for the train pulse. Inside each sub-interval, quantized GL quadrature is performed using a 16 points regular grid for single pulse and LFM waveforms and using only 4 nodes for the pulse train, thus leading to the same total number of nodes for each waveform processing. We check that quantized GL quadrature error is lower than NC quadrature while finite field quantization of amplitude, which is required for NTT implementation, has no noticeable effect (curves $\circ$ and $\triangledown$). Here, the FNT is performed on stacked node values grids of all quadrature sub-intervals.

Let us remark that lower performance is achieved by pulse train than by other waveforms, due to more irregular shape of integrated function.

Let n denote the number of equally spaced points on the grid and m the number of nodes of Gauss-Legendre quadrature. As noted in section 4, computing the FNT is a very simple operation on a binary machine. To compute a length nm fast FNT, $nm \log_2 nm$ additions/subtractions, and $(nm/2)\log_2 nm/2$ multiplications by some powers of 2 are required which are implemented as bit shifts and subtractions [10]. If two sequences $(x_i)_{i=1,nm}$ and $(h_i)_{i=1,nm}$ are to

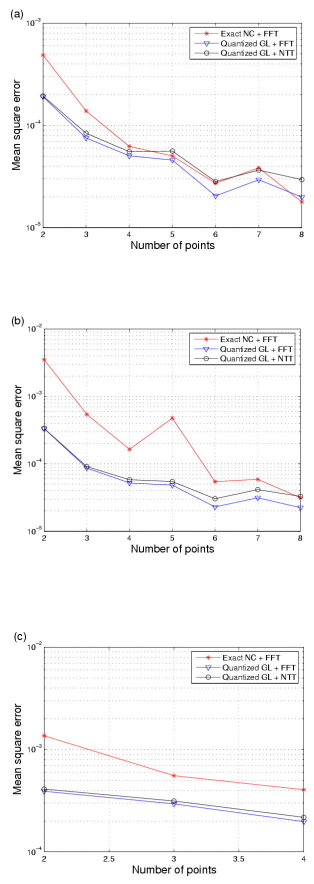

Figure 3: Mean square error as a function of the number of points, for a) single pulse, b) Linear Frequency Modulation (LFM) c) pulse train.

be convolved, the convolution product complexity is of order nm. Finally, if N_f values of the frequency are considered for calculating the ambiguity function, the whole ambiguity function can be calculated at the expense of order $N_f nm$ operations.

6. CONCLUSION

In this paper we have introduced a new approach for fast convolution calculation based on approximate Gauss-Legendre quadrature on a regular grid, associated with FNT algorithm. We have checked good behavior of this procedure when applied to standard radar waveforms ambiguity functions calculation at the expense of low computational complexity. Clearly, for convolutions involving integration on a semi-infinite interval, or on $\mathbb{R}$, this method can be adapted by using Gauss-Laguerre and Gauss-Hermite quadrature respectively. On another hand, one can also benefit from the fact that in many situations phase variation induced by frequency offset little varies inside quadrature intervals leading to further complexity reduction.

7. ACKNOWLEDGMENT

Part of this work has been sponsored by ONERA, the French Aerospace Lab (PEA NFO project) and Maroco Pole STIC (sciences and technologies for information and communications).

REFERENCES

[1] V. Levanon, *Radar Principles*. New York: John Wiley & Sons, 1988.

[2] B. R. Mahafza, *Radar Systems Analysis ans Design Using Matlab*. New York: Chapman & Hall/CRC, 2000.

[3] P. M. Woodward, *Probability and information Teory, with Application to Rdar*. 2nd ed. Oxford: Pergamon Press, 1964.

[4] A. Papoulis, *Signal Analysis*. New York: McGraw-Hill, 1977.

[5] T. Shan, R. Tao, and R. S. Rong, “A Fast Method for Time Delay, Doppler Shift and Doppler Rate Estimation,” in *International Conference on Radar (CIE’06)*, Shanghai, China, October 16-19. 2006, pp. 1–4.

[6] O. Rabaste and T. Chonavel, "Estimation of Multipath Channels With Long Impulse Response at Low SNR via an MCMC Method", *IEEE Trans. Sig. Proc.*, vol. 55, pp. 1312–1325, April 2007.

[7] L. Auslander and R. Tolimieri, "Computing Decimated Finite Cross-Ambiguity Functions", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 36, N.3, pp. 359–364, Mar. 1974.

[8] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*. New York: Academic Press, 1975.

[9] C. Guilpin, *Manuel de Calcul Numérique Appliqué*. Les Ulis: EDP Sciences, 1999.

[10] R. C. Agarwal and C.S. Burrus, "Fast convolution using Fermat number transform with application to digital filtering", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-22, N.2, pp. 87–97, 1974.

[11] G. H. Golub and J. H. Welsch, "Calculation of Gauss quadrature rules", *Math. Comp.*, Vol 23, pp. 221–230, 1969.

[12] L. N. Trefethen, *Spectral Methods in MATLAB*. Philadelphia: SIAM, 2000.

[13] I. M. Niven and H. S. Zuckerman, *An Introduction to the theory of numbers*. New York: John Wiley & Sons, 1980.

[14] S. W. Golomb and G. Gong, *Signal Design for Good Correlation for Wirelless Commununication, Cryptographie, and Radar*. Cambridge: University Press, 2005.

[15] S. Xu, L. Dai, S.C. Lee, "Autocorrelation Analysis of Speech Signals Using Fermat Number Transform", *IEEE Transactions on Signal Processing*, vol. 40, N.8, pp. 1910–1914, August. 1992.

[16] C. M. Rader, “The number theoretic DFT and exact discrete convolution,” in *IEEE Arden House Workshop on Digital Signal Processing*. Harriman, N. Y., Jan. .11, 1972

[17] C. M. Rader, "Discrete convolution via Mersenne transforms", *IEEE Trans. Comput.*, vol. C-21, pp. 1269–1273, December. 1972.