International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:1, 2010

Quadrature formula for sampled functions

Khalid Minaoui, Thierry Chonavel, Benayad Nsiri, Driss Aboutajdine

Abstract—This paper deals with efficient quadrature formulas involving functions that are observed only at fixed sampling points. The approach that we develop is derived from efficient continuous quadrature formulas, such as Gauss-Legendre or Clenshaw-Curtis quadrature. We select nodes at sampling positions that are as close as possible to those of the associated classical quadrature and we update quadrature weights accordingly. We supply the theoretical quadrature error formula for this new approach. We show on examples the potential gain of this approach.

Keywords—Gauss-Legendre, Clenshaw-Curtis, quadrature, Peano kernel, irregular sampling.

I. Introduction

IN general, integration of a known function, say f, is achieved through quadrature formula. We assume that f is continuous on the interval $[-1, 1]$ and we look for its integral on this interval. Adapting results to any finite interval of the form $[a, b]$ would be straightforward. Classical quadrature formulas are calculated from the value of f at n prescribed nodes and quadrature yields the integral of the corresponding Lagrange interpolant [1] of f. As a consequence, such a quadrature formula is exact whence f is a polynomial with degree at most $n - 1$.

Unfortunately, when calculated at regularly spaced points of an interval, Lagrange interpolants are flawed due to large oscillations at the interval ends that increase with the number n of interpolation points, even for flat functions : although the interpolation error of Lagrange interpolants is zero at interpolation points, it increases with n between successive interpolation points. This phenomenon is known as Runge phenomenon [2][3]. In particular, Newton-Cotes quadrature that performs the integral of Lagrange interpolant from regularly sampled values of f becomes numerically unstable for n as small as a few tenth. However, the situation is not desperated since Weierstrass' theorem states that if f is continuous on a closed interval, then there exists a sequence of polynomials with increasing degrees that uniformly converges to it [4].

One way to get a better polynomial interpolant for integrand $f(x)$ consists in building it from irregularly spaced points. More precisely, nodes should be chosen in such a way that their asymptotic density distribution is proportional to $1/\sqrt{1-x^2}$ [2]. In particular, Gauss-Legendre (GL) quadrature [5] and the more recent Clenshaw-Curtis (CC) quadrature introduced in 1960 [6] have quadrature nodes that share this property. Both show very good and quite similar performance for a large variety of integrands [7].

CC quadrature is a bit less precise than GL quadrature; In particular, CC quadrature integrates exactly polynomials with degree up to $n-1$, while GL quadrature is exact up to degree $2n-1$ [8]. GL quadrature nodes and weights can be computed at the expense of $O(n^2)$ operations by solving an eigenvalue problem [9], while for CC quadrature they can be calculated at the expense of $O(n.log_2(n))$ operations only, thanks to the FFT [10].

In signal processing one often has to integrate a function f that is observed only at fixed sampling points $\mathbf{y} = (y_1, \ldots, y_m)^T$, with $-1 \leq y_k < y_{k+1} \leq 1$ for $k = 1, \ldots, m-1$. Ambiguity function calculation [11], numerical solution of first kind integral equations [12] or filtering [13] are classical examples where numerical integration is useful. In this paper, we are looking for the calculation of $\int_{[-1,1]} f(x)dx$ from $f(y_1), \ldots, f(y_m)$. In order to perform this task in an efficient way, we must account for the above discussion about the optimum node location. To this end, we are going to perform approximate GL or CC quadrature by selecting n nodes in $\mathbf{y}$ that are closest to nodes of a GL or CC quadrature with n nodes. As the density of samples increases (m becomes large), these n nodes tend to approach the true GL or CC nodes. In addition, quadrature weights will be chosen accordingly, that is, to ensure exact quadrature for polynomials up to degree $n - 1$.

The rest of the paper is organized as follows. In section 2 we recall basics about GL and CC quadrature and we describe the new, data fitting, quadrature scheme and we derive the expression of its error function. The examples in section 3 show the good behavior of this approach. In particular, we see how it can be used in ambiguity functions calculation.

II. Quadrature formulas

A. Gauss-Legendre and Clenshaw-Curtis quadrature

Letting $f(x)$ and $w(x) > 0$ denote two continuous functions, we consider a quadrature formula of the form

$$I = \int_{-1}^{1} w(x)f(x)dx \approx \sum_{i=1,m} w_i f(x_i). \qquad (1)$$

For the scalar product associated to $w(x)$ and defined by

$$< g_1, g_2 >= \int_{-1}^{1} w(x)g_1(x)g_2(x)dx, \qquad (2)$$

K. Minaoui is with Lab-STICC (CNRS FRE 3167), Institut Telecom, Telecom Bretagne, Technopole Brest Iroise, 29238 Brest, France and with Laboratoire LRIT, unité associé au CNRST, Faculté des Sciences de Rabat, Université Mohammed V Agdal, Morocco. khalid.minaoui@telecom-bretagne.eu.

T. Chonavel is with Lab-STICC (CNRS FRE 3167), Institut Telecom, Telecom Bretagne, Technopole Brest Iroise, 29238 Brest, France. Thierry.chonavel@telecom-bretagne.eu.

B. Nsiri is with Faculté des Sciences Ain Chock, Université Hassan II, Casablanca and with Laboratoire LRIT, unité associé au CNRST, Faculté des Sciences de Rabat, Université Mohammed V Agdal, Morocco. Benayad.nsiri@telecom-bretagne.eu.

D. Aboutajdine is with Laboratoire LRIT, unité associé au CNRST, Faculté des Sciences de Rabat, Université Mohammed V Agdal, Morocco. aboutaj@ieee.org.

we denote by $p_k(x)$ the degree k orthonormal polynomial: $< p_i(x), p_j(x) >= \delta_{i,j}$, where $\delta_{i,j}$ is the Kronecker's delta function. Then, the quadrature formula (1) is exact for all polynomials of degree at most $2n-1$, provided nodes $x_1, \ldots, x_n$ are the zeros of $p_n(x)$ ([5] p. 74).

In particular, GL quadrature follows this principle, where $w(x) = 1$ for all x in $[-1, 1]$. Thus, nodes x_i are defined by the roots of the degree n Legendre polynomial. The weights $\mathbf{w} = (w_1, \ldots, w_n)^T$ are the solutions of equations

$$\int_{-1}^{1} x^k dx = \sum_{i=1,n} w_i x_i^k, \quad k = 0, ..., n-1. \tag{3}$$

Efficient algorithms for computing quadrature rules can be found in the literature [9] (see also [5]) and practical implementations are available for many programming languages (see for instance [5] and [7] for a MATLAB implementation). CC quadrature can be seen as a modified GL quadrature, where nodes are simply given by [6]

$$x_k = \cos \frac{n-k}{n-1}\pi, \qquad k = 1, \ldots, n. \tag{4}$$

CC nodes are in fact the extrema of Chebyshev polynomials in $[-1, 1]$ and they remain quite close to GL nodes that can be written in the form ([5] p.89)

$$x_k = \cos\left(\frac{n-k-1/4}{n+1/2}\pi + o(\frac{1}{n})\right), \qquad k = 1, \ldots, n. \tag{5}$$

B. Quadrature from sampled functions

For a continuous function f sampled at points $\mathbf{y} = (y_1, \ldots, y_m)^T$, there is generally no reason for points $\mathbf{y}$ to contain nodes of any standard quadrature formula, except when considering the particular case of regular sampling together with Newton-Cotes quadrature. But in this case, quadrature performs poorly as discussed earlier.

Thus, with a view to get quadrature nodes that tend to be distributed as GL or CC nodes, we consider a GL or CC quadrature, with n nodes and $n < m$, and we select the n points in $\mathbf{y}$ that are closest to GL or CC nodes. These points in $\mathbf{y}$ define the nodes for the new quadrature. Let us denote these points by $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)^T$. These nodes are associated with weights $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_n)^T$ that ensure exact quadrature for polynomials with degree up to $n-1$. In other words, $\hat{\mathbf{w}}$ is the solution of the following linear set of equations:

$$\int_{-1}^{1} x^k dx = \frac{(1-(-1)^{(k+1)})}{k+1} = \sum_{i=1}^{n} \hat{w}_i \hat{x}_i^k, \quad k = 0, \ldots, n-1. \tag{6}$$

Thus

$$\begin{pmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \vdots \\ \hat{w}_n \end{pmatrix} = \begin{pmatrix} 1 & ... & 1 \\ \hat{x}_1 & ... & \hat{x}_n \\ \vdots & & \vdots \\ \hat{x}_1^{n-1} & ... & \hat{x}_n^{n-1} \end{pmatrix}^{-1} \begin{pmatrix} 2 \\ 0 \\ \vdots \\ \frac{1-(-1)^n}{n} \end{pmatrix}. \tag{7}$$

In a matrix form, we get

$$\hat{\mathbf{w}} = \hat{\mathbf{M}}^{-1}\hat{\mathbf{c}}. \tag{8}$$

Note that the inverse matrix can be calculated in closed form (see Appendix A).

For $f \in C^k[-1, 1]$ and a quadrature defined by nodes $\mathbf{x}$ and weights $\mathbf{w}$ that integrates exactly polynomials up to degree $k-1$ at least, the quadrature error is given by ([5] pp. 218-223)

$$\begin{aligned} E(f) &= \int_{-1}^{1} f(x)dx - \sum_{i=1}^{n} w_i f(x_i) \\ &= \int_{-1}^{1} f^{(k)}(x) N_k(x) dx, \end{aligned} \tag{9}$$

where $N_k(x)$ is Peano's Kernel:

$$N_k(x) = \frac{(1-x)^k}{k!} - \sum_{i=1}^{n} w_i \frac{(x_i - x)_+^{k-1}}{(k-1)!}, \tag{10}$$

and $g_+(x) = \max(0, g(x))$.

In the same way, we have proved that, up to a second order term, the theoretical error for the approximate GL or CC quadrature rule that we have defined here above is equal to the GL or CC quadrature error with an additional term that linearly depends on nodes perturbations. We shall denote by $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{w} = (w_1, \ldots, w_n)^T$ the GL or CC quadrature nodes and weights repectively, and we let $\hat{\mathbf{x}} = \mathbf{x} + \delta_{\mathbf{x}}$ and $\hat{\mathbf{w}} = \mathbf{w} + \delta_{\mathbf{w}}$ the perturbated nodes and weights. The following theorem supplies the expression of the error for quadrature parameters $(\hat{\mathbf{x}}, \hat{\mathbf{w}})$ and its proof is supplied in appendix A.

Theorem 1: For a k differentiable function $f : [-1, 1] \to \mathbb{R}$, denoting by $N_k(\tau)$ its Peano Kernel for the quadrature parameters $(\mathbf{x}, \mathbf{w})$, then for parameters $(\hat{\mathbf{x}}, \hat{\mathbf{w}}) = (\mathbf{x} + \delta_{\mathbf{x}}, \mathbf{w} + \delta_{\mathbf{w}})$ the quadrature error is given by the following expressions, up to a second order term in $\delta_{\mathbf{x}}$:

$$E(f) = \int_{-1}^{1} f^{(k)}(x) \hat{N}_k(x) dx, \tag{11}$$

where

$$\begin{aligned} \hat{N}_k(x) \approx\ & N_k(x) - \sum_{i=1}^{n} \delta_{x_i} w_i \left(\frac{(x_i - x)_+^{k-2}}{(k-2)!} \right. \\ & \left. - \sum_{j=1}^{n} \frac{\sum_{u \neq j}^{n} \left(\prod_{r \neq j,u}^{n} (x_i - x_r) \right)}{\prod_{t \neq j}^{n} (x_j - x_t)} \frac{(x_j - x)_+^{k-1}}{(k-1)!} \right). \end{aligned} \tag{12}$$

Theorem 1 shows in particular that when the number of sampling points increases with maximum distance between contiguous points decreasing to 0 then $\max_{k=1,m} \delta_{x_k} \to 0$ and $\hat{N}_k(x) \to N_k(x)$ as one may expect. Furthermore, although minimization of the amplitude of $\hat{N}_k$ over possible nodes would be a challenging combinatorial problem, Eq. (12) suggests the simpler approach that consists in minimizing errors $|\delta_{x_k}|$ by selecting $\hat{\mathbf{x}}$ as close as possible to exact GL or CC nodes.

We shall call quadrature with parameters $(\hat{\mathbf{x}}, \hat{\mathbf{w}})$ the quantized GL (QGL) or quantized CC (QCC) quadrature respectively, depending whether initial parameters $(\mathbf{x}, \mathbf{w})$ are GL or CC parameters.

III. Examples

A. Basic examples

The higher the number of nodes, the more accurate the quadrature with parameters $(\hat{\mathbf{x}}, \hat{\mathbf{w}})$ should be. This is illustrated in Figure 1 for $n = 10$ nodes where the quadrature errors for $f(x) = x^k$ are plotted for increasing values of k. For CC and GL quadrature, we check that error can be canceled (up to machine precision) for $k \leq 9$ with CC and $k \leq 19$ with GL. QGL and QCC performance are plotted for $m = 50, 150$ and 300 regularly sampled data. Both QGL and QCC have performance that are little degradated compared to what would supply CC if samples at exact CC nodes (Eq. (4)) were available.

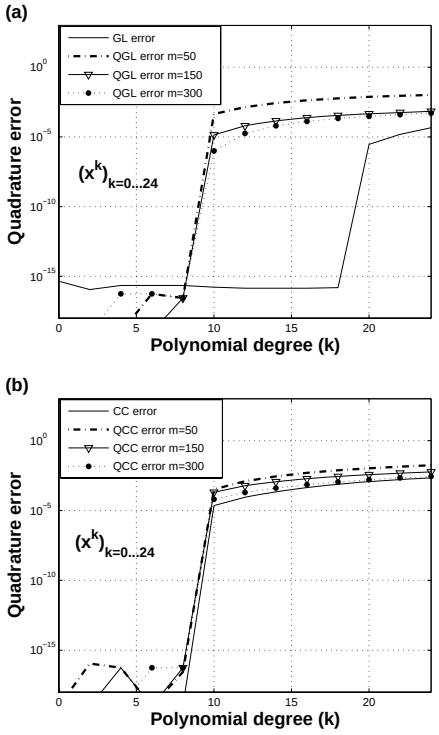

Fig. 1. Integration error for $f(x) = x^k$ as a function of k for several data sizes (50,150,300) (a) GL and QGL (b) GC and QCC.

On another hand, we have checked that using QGL or QCC with nodes $\hat{\mathbf{x}}$ but with GL or CC weights $\mathbf{w}$ instead of $\hat{\mathbf{w}}$ results in serious performance loss, especially for low degree polynomials quadrature.

Figure 2 shows quadrature error for polynomial x^{20} as a function of the number of quadrature nodes and for 100 regularly spaced samples. We check that the theoretical error supplied by theorem 1 is very close to simulation results.

Now, we consider $\int_{-1}^{1}(1+x^2)^{-1}dx$ calculated from 50 samples obtained at random points. Mean QGL and QCC error with dispersion bars for 10 experiments are plotted in Figure 3. QGL and QCC achieve quite similar performance. In addition, we have checked that they are very good compared to Riemann or Simpson quadrature that yield error no better than $3\ 10^{-2}$ when using all data.

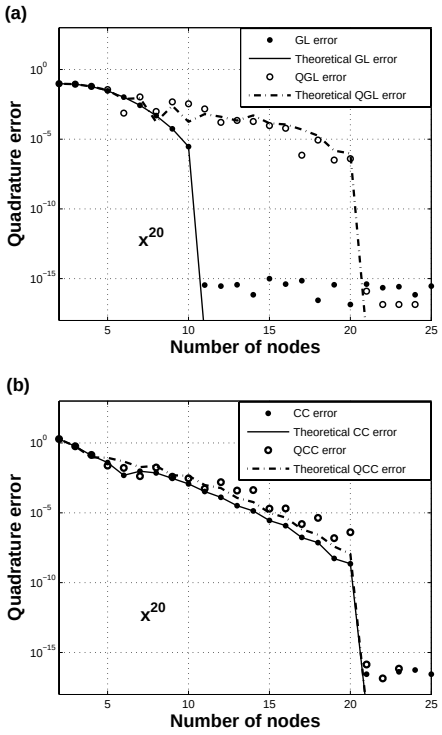

Fig. 2. Integration error for $f(x) = x^{20}$ as a function of number of nodes for data size = 100 (a) GL and QGL (b) CC and QCC.

B. Quadrature using all samples

We have seen that good results are obtained when using quadrature parameters $(\hat{\mathbf{x}}, \hat{\mathbf{w}})$ from sampled data. Let us consider the issue of further quadrature error reduction when using all m available data points. Whence the n QGL or QCC nodes and weight are fixed and denoted by $(\hat{x}_i^k, \hat{w}_i)_{i=1,\ldots,n}$. The quadrature in this case will be written as,

$$I = \int_{-1}^{1} w(x)f(x)dx \approx \sum_{i=1,n} \hat{w}_i f(\hat{x}_i) + \sum_{j=1,s} \bar{w}_j f(\bar{x}_j), \quad (13)$$

where $s = m - n$. The $\bar{x}_j$'s are samples that are not used in QGL or QCC quadrature. Then, to calculate the weights $\bar{w}_j$ we minimize the quantity

$$\sum_{k=0}^{L-1} \Big\| [\int_{-1}^{1} x^k dx - \sum_{i=1,n} \hat{w}_i \hat{x}_i^k] - \sum_{j=1,s} \bar{w}_j \bar{x}_j^k \Big\|^2, \quad (14)$$

for some order L. We use the following notations : $\hat{\mathbf{w}} = (\hat{w}_1, \cdots, \hat{w}_n)^T$, $\bar{\mathbf{w}} = (\bar{w}_1, \cdots, \bar{w}_s)^T$,

$$\mathbf{a} = \begin{pmatrix} 2 \\ 0 \\ \vdots \\ \frac{(1-(-1)^{(L)})}{L} \end{pmatrix} - \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \hat{x}_1 & \hat{x}_2 & \ldots & \hat{x}_n \\ \vdots & \vdots & & \vdots \\ \hat{x}_1^{L-1} & \hat{x}_2^{L-1} & \ldots & \hat{x}_n^{L-1} \end{pmatrix} \hat{\mathbf{w}},$$

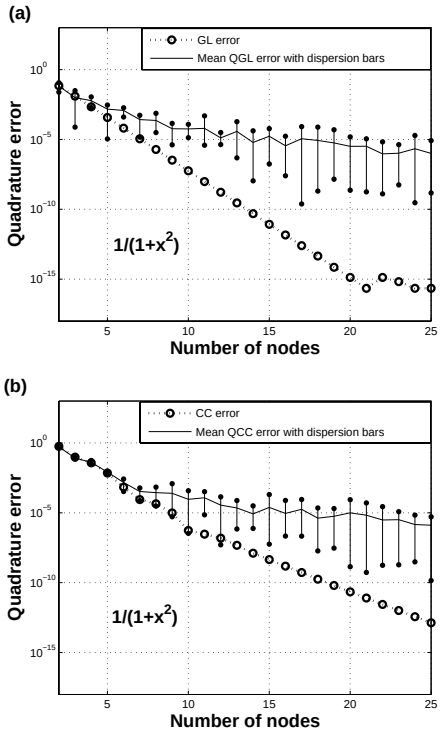

Fig. 3. (a) QGL and (b) QCC quadrature error as a function of the number of nodes for $f(x) = (1+x^2)^{-1}$ and 50 randomly sampled data.

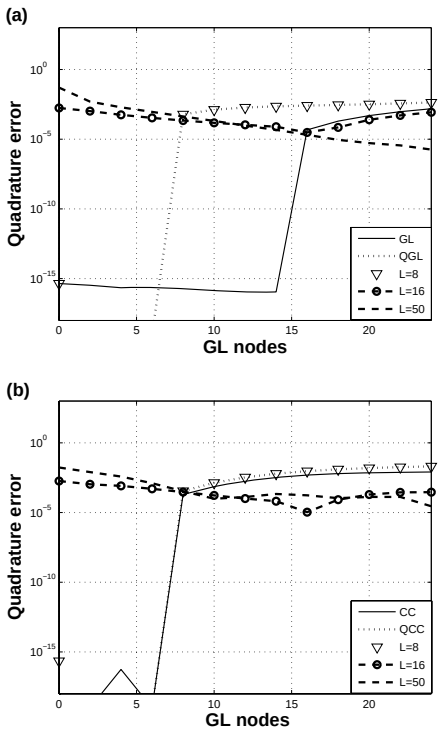

Fig. 4. Integration error for $f(x) = x^k$ as a function of k when using all the points of the grid (a) GL, QGL and grid error (b) GC, QCC and grid error.

and

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & ... & 1 \\ \bar{x}_1 & \bar{x}_2 & ... & \bar{x}_s \\ \vdots & \vdots & & \vdots \\ \bar{x}_1^{L-1} & \bar{x}_2^{L-1} & ... & \bar{x}_s^{L-1} \end{pmatrix},$$

where $.^T$ is the transpose operator. Then, we rewrite criterion (14) in a matrix form as follows :

$$\left\| \mathbf{a} - \mathbf{X}\bar{\mathbf{w}} \right\|^2, \tag{15}$$

and minimization of criterion (15) yields

$$\widehat{\bar{\mathbf{w}}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{a}. \tag{16}$$

As an example we choose $n = 8$ nodes from 50 regularly sampled data. Figure 4 shows the quadrature errors for $f(x) = x^k$ for increasing values of k. Performance are plotted for exact and quantized quadrature as well as for $L = 8, 16, 50$ in criterion (15). We check that using all samples with $L \leq 8$, performance are similar to the QGL/QCC. Indeed, in this case, we get $\bar{w} = 0$. For $L > 8$ performance are seriously deteriorated for low degree polynomials. In addition we check that matrix $\mathbf{X}^T\mathbf{X}$ is rather ill-conditioned. This shows that there is no gain in trying to use sampled points that are not in $\hat{\mathbf{x}}$ by minimizing criterion (14) over $\bar{\mathbf{w}}$.

C. Computation of ambiguity functions

In radar processing, an observed signal $r(x)$ is issued from reflections of an emitted waveform, say $u(x)$. The cross ambiguity function between r and u is a useful tool for targets detection and parameters estimation [11], [14], [15], [16]. It is defined by

$$\chi_{ru}(\tau, f) = \int_{-\infty}^{\infty} r(x)u^*(x-\tau)\exp(j2\pi f x)dx, \tag{17}$$

where τ and f are the time delay and doppler frequency parameters respectively. Clearly, for fixed f, $\tau \rightarrow \chi_{ru}(\tau, f)$ is a convolution operation. More precisely, $\chi_{ru}(\tau, f)$ can be seen as the output, at time τ, of the matched filter associated with the waveform $u(x)$ when $r(x)$ demodulated at frequency f is present at the input. When $r(x) = u(x)$, χ_{uu} is called the ambiguity function of the waveform $u(x)$.

The cross-ambiguity between an echoed radar signal and the emitted waveform $u(x)$ shows attenuated and Doppler-and-delay shifted versions of the ambiguity function of $u(x)$ 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 [17] for the case of linear chirp waveforms). But, working with $\chi_{ru}(\tau, f)$ can be seen as a general way to perform this task for any kind of waveform. In addition, working with function $\chi_{ru}(\tau, f)$ enables recovery of possibly very closely located sources in the time-frequency domain [18].

Restricting here our interest to the ambiguity function

calculation, we consider a pulse train waveform defined by

$$u(t) = \frac{1}{\sqrt{N_p}} \sum_{k=0}^{N_p-1} \frac{1}{\sqrt{d}} \text{ rect}\left(\frac{t-kT}{d}\right), \qquad (18)$$

where d is the pulse duration, N_p the number of the pulses in the train, T the pulse repetition interval and

$$\text{rect}(t) = \begin{cases} 1 & |t| \leq 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding ambiguity function is expressed as follow [11], [14]:

$$\begin{aligned} \chi_{uu}(\tau, f) &= \big(1 - \frac{|\tau|}{d}\big) \frac{\sin\left(\pi f(d-|\tau|)\right)}{\pi f(d-|\tau|)} \text{ rect}(\frac{\tau}{2d}) \\ &\times \sum_{q=-N_p+1}^{N_p-1} \left| \frac{\sin\left(\pi f(N_p-|q|)T\right)}{N_p \sin\left(\pi f T\right)} \right|. \end{aligned} \qquad (19)$$

This ambiguity function is plotted in Figure 5 with parameters $d = 0.25s$, $N_p = 4$ and $T = 1s$.

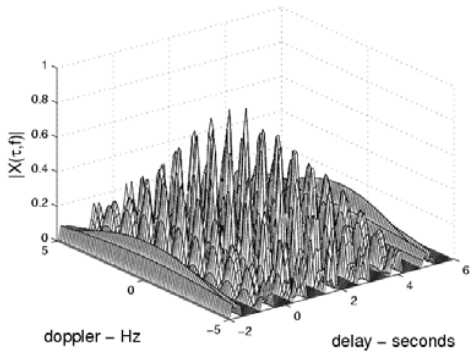

Fig. 5. Ambiguity functions for a pulse train.

We compute $\chi_{uu}(\tau, f)$ by dividing the integration interval into N sub-intervals of the form $[t_k, t_{k+1}]$ $(k = 0, \ldots, N-1)$. Thus, letting $h_k = t_{k+1} - t_k$,

$$\begin{aligned} \int u(x)u(x-\tau)e^{j2\pi fx}dx &\approx \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} u(x)u(x-\tau) \\ &\quad \times e^{j2\pi fx}dx \\ &= \sum_{k=0}^{N-1} \frac{h_k}{2} \int_{-1}^{1} u\big(t_k + \frac{(x+1)}{2}h_k\big) \\ &\quad \times u\big(t_k + \frac{(x+1)}{2}h_k - \tau\big) \\ &\quad \times e^{j2\pi f(t_k + \frac{(x+1)}{2}h_k)}dx. \end{aligned} \qquad (20)$$

We calculate integrals at regularly sampled points (τ, f) from regularly sampled u :

$$\begin{aligned} \int u(x)u(x-\tau)e^{j2\pi fx}dx &\approx \sum_{k=0}^{N-1} \sum_{i=1}^{n} \frac{h}{2} \hat{w}_i u(\hat{z}_{ik}) \\ &\quad \times e^{j2\pi f\hat{z}_{ik}} u(\hat{z}_{ik} - \tau), \\ &= \sum_{i,k} v(\hat{z}_{ik}) u(\hat{z}_{ik} - \tau), \end{aligned} \qquad (21)$$

where $\hat{z}_{ik} = t_k + \frac{(\hat{x}_i+1)}{2}h$ and $v(\hat{z}_{ik}) = \frac{h}{2}\hat{w}_i u(\hat{z}_{ik})e^{j2\pi f\hat{z}_{ik}}$.

For QGL or QCC quadratures $\hat{w}_i$ are either zero or equal to quadrature weights, depending wether samples $\hat{z}_{ik}$ are QGL or QCC nodes respectively. For Riemann quadrature, all weights are equal. Then, in Eq. (21), convolution between u and v can be done in the Fourier domain by using FFT and inverse FFT algorithms.

Letting $\hat{\chi}_{uu}(\tau, f)$ denote the calculated approximation of the ambiguity function, we consider the performance indices given by the mean square error, that is defined by

$$MSE = \text{mean}_{(\tau,f)}\left(|\hat{\chi}(\tau, f) - \chi(\tau, f)|^2\right). \qquad (22)$$

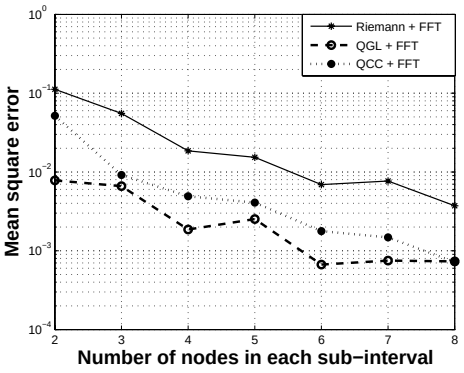

Fig. 6. Mean square error as a function of the number of nodes.

Figure 6 shows mean square error MSE as a function of the number of quadrature nodes. Quadrature is achieved by slicing the convolution interval into 8 sub-intervals. 16 samples are available in each sub-interval from which QGL or QCC nodes are selected.

As far as complexity is considered, for Riemann quadrature, the FFTs for calculating the convolution in Eq. (21) are classically calculated at the expense of $O(Nn \log Nn)$. To get the corresponding complexity for QGL or GCC we account for results supplied in [19] that present a complexity analysis of the FFT of sparse vectors, that is, vectors that contain many zero entries. Then, when QGL or QCC is performed with 8 nodes the FFT requires about 850 multiplications, which is the same order of complexity as for Riemann quadrature with 15 nodes. However, with 8 nodes, $MSE = 7\ 10^{-4}$ for QGL or QCC, while $MSE = 1.6\ 10^{-3}$ for Riemann with 15 nodes. Thus, for similar complexity significantly better accuracy is achieved by QGL and QCC.

IV. Conclusion

In this paper we have introduced a new approach for quadrature of functions observed only at (possibly irregular) sampling positions. We have checked very good theoretical and practical quadrature error performance for this approach, and we have shown that this technique can be useful in particular for calculating ambiguity functions.

APPENDIX A
ERROR FORMULA FOR APPROXIMATE QUADRATUR

Let us consider a quadrature formula with n nodes and parameters $(\mathbf{x}, \mathbf{w})$ and f in $C^k[-1,1]$, with $k \leq n$. Then error formula (11) is satisfied. Now, we consider the perturbated quadrature $(\hat{\mathbf{x}}, \hat{\mathbf{w}}) = (\mathbf{x} + \delta_{\mathbf{x}}, \mathbf{w} + \delta_{\mathbf{w}})$. The Peano kernel of this new quadrature is

$$\hat{N}_k(x) = \frac{(1-x)^k}{k!} - \sum_{i=1}^{n}(w_i + \delta_{w_i})\frac{(x_i + \delta_{x_i} - x)_+^{k-1}}{(k-1)!},$$

Let us assume that $k \geq 2$. The first order Taylor expansion of the term $(x_i + \delta_{x_i} - x)_+^{k-1}$ yields

$$\begin{aligned}(x_i + \delta_{x_i} - x)_+^{k-1} &\approx [(x_i - x)_+^{k-1} + \\ &\quad (k-1)\delta_{x_i}(x_i - x)_+^{k-2}]\mathbb{1}_{x \leq x_i + \delta_{x_i}},\end{aligned}$$

with $\mathbb{1}_A(x) = 1$ if $x \in A$, and $\mathbb{1}_A(x) = 0$ otherwise. Then, up to second order terms in $\delta_{\mathbf{x}}$, we get

$$\hat{N}_k(x) \approx \frac{(1-x)^k}{k!} - \sum_{i=1}^{n}\left(\delta_{w_i}\frac{(x_i - x)_+^{k-1}}{(k-1)!}\right. \quad (23)$$
$$\left. + w_i\Big(\frac{(x_i - x)_+^{k-1}}{(k-1)!} + \delta_{x_i}\frac{(x_i - x)_+^{k-2}}{(k-2)!}\Big)\right)\mathbb{1}_{x \leq x_i + \delta_{x_i}}.$$

Now, if we assume that quadrature is exact for polynomials with degree up to $n-1$ at least, it is clear that $\mathbf{w}$ is completely specified from $\mathbf{x}$. Indeed, since

$$\int_{-1}^{1} x^k dx = \sum_{i=1,n} w_i x_i^k = \frac{(1-(-1)^{(k+1)})}{k+1}, \quad k = 0, ..., m-1, \quad (24)$$

and $\mathbf{w}$ is the solution of the following matrix equation:

$$\begin{pmatrix} 1 & ... & 1 \\ x_1 & ... & x_n \\ \vdots & & \vdots \\ x_1^{n-1} & ... & x_n^{n-1} \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ \vdots \\ \frac{1-(-1)^n}{n} \end{pmatrix}. \quad (25)$$

In a matrix form, we rewrite Eq. (25) as

$$\mathbf{M}\mathbf{w} = \mathbf{c}. \quad (26)$$

In order to supply more precise expression of $\hat{N}_k(x)$ when both $(\mathbf{x}, \mathbf{w})$ and $(\hat{\mathbf{x}}, \hat{\mathbf{w}})$ satisfy Eq. (25), let us express $\hat{N}_k(x)$ only in terms of $(\mathbf{x}, \delta_{\mathbf{x}}, \mathbf{w})$ by relating $\delta_{\mathbf{w}}$ to other parameters. This is achieved as follows: letting

$$\delta_{w_i} = \sum_{j=1}^{n}\frac{\partial w_i}{\partial x_j}\delta_{x_j}, \quad (27)$$

the term

$$T = \sum_{i=1}^{n}\delta_{w_i}(x_i - x)_+^{k-1} \quad (28)$$

in Eq. (23) can be rewritten as

$$\begin{aligned} T &= \begin{pmatrix} \delta_{x_1} & \delta_{x_2} & ... & \delta_{x_n} \end{pmatrix} \\ &\times \begin{pmatrix} \frac{\partial w_1}{\partial x_1} & ... & \frac{\partial w_n}{\partial x_1} \\ \frac{\partial w_1}{\partial x_2} & ... & \frac{\partial w_n}{\partial x_2} \\ \vdots & & \vdots \\ \frac{\partial w_1}{\partial x_n} & ... & \frac{\partial w_n}{\partial x_n} \end{pmatrix}\begin{pmatrix} (x_1 - x)_+^{k-1} \\ (x_2 - x)_+^{k-1} \\ \vdots \\ (x_n - x)_+^{k-1} \end{pmatrix}. \end{aligned}$$

On another hand, derivation of (26) with respect to x_i yields

$$w_i\begin{pmatrix} 0 \\ 1 \\ 2x_i \\ \vdots \\ (n-1)x_i^{n-2} \end{pmatrix} + \mathbf{M}\begin{pmatrix} \frac{\partial w_1}{\partial x_i} \\ \frac{\partial w_2}{\partial x_i} \\ \vdots \\ \frac{\partial w_n}{\partial x_i} \end{pmatrix} = 0.$$

But, $\mathbf{M}$ is a VanderMonde matrix and thus its inverse can be expressed in closed form [20], [21]. In fact, we have

$$\begin{aligned} T &= -\begin{pmatrix} \delta x_1 & ... & \delta x_n \end{pmatrix} \; diag(w_1, ..., w_n) \\ &\times \begin{pmatrix} 0 & 1 & ... & (n-1)x_1^{(n-2)} \\ \vdots & \vdots & & \\ 0 & 1 & ... & (n-1)x_n^{(n-2)} \end{pmatrix}\mathbf{S} \\ &\times diag\left(\frac{1}{\prod_{k\neq 1}^{n}(x_1 - x_k)}, \quad \cdots \quad , \frac{1}{\prod_{k\neq n}^{n}(x_n - x_k)}\right) \\ &\times \big((x_1 - x)_+^{k-1}, \ldots, (x_n - x)_+^{k-1}\big)^T, \end{aligned} \quad (29)$$

where

$$\mathbf{S}_{ij} = (-1)^{n-i}\sum_{\substack{i_1+i_2+...+i_n=m \\ i_s \in \{0,1\}}}^{n}\left(\prod_{s\neq j}^{n} x_s^{i_s}\right).$$

Since

$$\sum_{r=1}^{n} x_i^{r-1}\mathbf{S}_{rj} = \prod_{r\neq j}^{n}(x_i - x_r), \quad (30)$$

deriving both sides of Eq. (30) w.r.t x_i yields

$$\sum_{r=1}^{n}(r-1)x_i^{r-2}\mathbf{S}_{rj} = \sum_{u\neq j}^{n}\Big(\prod_{r\neq j,u}(x_i - x_r)\Big). \quad (31)$$

Clearly, Eq. (31) supplies entry (i,j) for the product of the two non-diagonal matrices in the expression of T in Eq. (29). Finally,

$$T = -\sum_{ij}^{n}\delta_{x_i}w_i\frac{\sum_{u\neq j}^{n}\Big(\prod_{r\neq j,u}^{n}(x_i - x_r)\Big)}{\prod_{t\neq j}^{n}(x_j - x_t)}(x_j - x)_+^{k-1}, \quad (32)$$

and

$$\begin{aligned}\hat{N}_k(x) \approx \tfrac{(1-x)^k}{k!} - \textstyle\sum_{i=1}^{n} w_i\mathbb{1}_{x\leq x_i+\delta_{x_i}}\Bigg(\tfrac{(x_i-x)_+^{k-1}}{(k-1)!} + \\ \delta_{x_i}\Big(\tfrac{(x_i-x)_+^{k-2}}{(k-2)!} - \textstyle\sum_{j=1}^{n}\frac{\sum_{u\neq j}^{n}\left(\prod_{r\neq j,u}^{n}(x_i-x_r)\right)}{\prod_{t\neq j}^{n}(x_j-x_t)}\tfrac{(x_j-x)_+^{k-1}}{(k-1)!}\Big)\Bigg).\end{aligned} \quad (33)$$

In addition, noting that for $n \geq 2$,

$$\delta_{x_i} \times (x_i - x)_+^n\mathbb{1}_{x\leq x_i+\delta_{x_i}} = \delta_{x_i} \times (x_i - x)_+^n + o(\delta_{x_i}), \quad (34)$$

we get Eq. (12) for $\hat{N}_k(x)$.

REFERENCES

[1] M. B. Allen and E. L. Isaacson, *Numerical Analysis for Applied Science*, Wiley-Interscience, 1998.
[2] B. Fornberg and P. G. Ciarlet and A. Iserles, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998.
[3] J. P. Berrut and L. N. Trefethen, "Barycentric Lagrange interpolation, SIAM rev.", pp. 501-517, 2004.
[4] H. Jeffreys and B. S. Jeffreys, *Weierstrass's Theorem on Approximation by Polynomials, P 14.08 in Methods of Mathematical Physics*, 3rd ed. England: Cambridge University Press, 1988.
[5] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, N.Y.: Academic Press, 1975.
[6] C. W. Clenshaw A. R. Curtis, "A method for numerical integration on an automatic computer", Numer. Math., vol. 2, pp. 197-205, 1960.
[7] L. N. Trefethen, "Is Gauss Quadrature Better Than Clenshaw-Curtis ?", SIAM Rev., Society for Industrial and Applied Mathematics, vol. 50(1), pp. 67-87, 2008.
[8] P. K. Kythe and M. R. Schäferkotter, *Handbook of Computational Methods for Integration*, CRC Press, 2005.
[9] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules", Math. Comp., vol. 23, pp. 221-230, 1969.
[10] W. M. Gentleman, "Implementing clenshaw-curtis quadrature, i- computing the cosine transformation", Communications of the ACM, vol. 15(5), pp. 337-342, 1972.
[11] V. Levanon, *Radar Principles*, New York: John Wiley & Sons, 1988.
[12] J. Graves and P. M. Prenter, "Numerical iterative filters applied to first kind Fredholm integral equation", Numer. Math., vol. 30, pp. 281-299, 1978.
[13] K. C. Aas and K. A. Duell and C. T. Mullis, "Synthesis of extremal wavelet-generating filters using Gaussian quadrature", IEEE Transaction on signal processing, vol. 43(5), pp. 1045-1057, 1995.
[14] B. R. Mahafza, *Radar Systems Analysis ans Design Using Matlab*, New York: Chapman & Hall/CRC, 2000.
[15] P. M. Woodward, *Probability and information Teory, with Application to Radar*, 2nd ed. Oxford: Pergamon Press, 1964.
[16] A. Papoulis, *Signal Analysis*, New York: McGraw-Hill, 1977.
[17] T. Shan and R. Tao, and R. S. Rong, "A Fast Method for Time Delay, Doppler Shift and Doppler Rate Estimation", International Conference on Radar (CIE'06), Shanghai, China, October 2006, pp. 1-4.
[18] 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, 2007.
[19] R. J. Baxley and G. T. Zhou, "Computational Complexity Analysis of FFT Pruning - A Markov Modeling Approch", Digital Signal processing Workshop, Proc. IEEE 12th Digital Signal Processing Workshop, 2006, pp. 535-539.
[20] N. Macon and A. Spitzbart, "Inverses of vandermonde matrices", The American Mathematical Monthly, vol. 65, pp. 95-100, 1958.
[21] A. Klinger, "The vandermonde matrix", The American Mathematical Monthly, vol. 74(5), pp. 571-574, 1967.