Some Remarks on Quadrature Formulas for Refinable Functions and Wavelets

Arne Barinka*, Titus Barsch, Stephan Dahlke†, Michael Konik

Abstract

This paper is concerned with the efficient computation of integrals of (smooth) functions against refinable functions and wavelets, respectively. We derive quadrature formulas of Gauss–type using these functions as weight functions. The methods are tested for several model problems and possible practical applications are discussed.

Key Words: Gauss quadrature, scaling functions, wavelets.
AMS subject classification: Primary 65D32, secondary 41A30, 42C05, 42C15.

1 Introduction

In recent years, much progress has been made in establishing wavelet analysis as a competitive tool for image analysis/compression and for the numerical treatment of operator equations. Nevertheless, there are still some important problems which have not been solved satisfactorily yet. In this paper, we give a contribution to one of these questions as we shall now explain. Whenever wavelet analysis is used in practice, sooner or later one has to compute inner products with wavelets or with an associated refinable function. We refer to Section 2 for a short overview on wavelet analysis. In some exceptional cases, it is possible to compute these inner products directly by finding primitives, however, usually they are not accessable and therefore one needs suitable quadrature rules. Of course, one could simply apply one of the well–known quadrature formulas which can be found in any book on numerical analysis. However, this usually leads to serious trouble. First of all, neither the refinable function nor the wavelet are necessarily very smooth so that a classical quadrature rule may not perform satisfactorily. Moreover, in many cases, these functions are not known explicitly but only via certain functional equations from which the function values have to be computed or approximated. This is possible in principle, however, these kinds of function evaluations may be expensive and/or inaccurate.

In the last few years, several approaches to this problem have been suggested. Dahmen and Micchelli [DM2] developed a method which is based on the iteration of a specific

*The work of this author has been supported by the Volkswagen Stiftung
†The work of this author has been supported by Deutsche Forschungsgemeinschaft, Grant Da 117/13–1

operator derived from a related subdivision scheme. It works in any dimension and under very low regularity assumptions on the function f, but it might be somewhat expensive. A different approach is presented in [PS1, PS2], where quadrature formulas for refinable functions are considered. Several types of Newton–Cotes quadratures are discussed, which are determined using Tschebyscheff polynomials.

In this paper, we are also concerned with the construction of quadrature formulas. Our approach is mainly based on *Gauss quadrature*, which offers the optimal exactness for a given number of quadrature points. The idea is to use the refinable function and the wavelet, respectively, as the weight function for such a quadrature rule. Note that in this construction the weight is only required to be non–negative. Of course, one has to be able to compute the moments of the weight functions with high accuracy. Fortunately, this is no problem in the wavelet setting. In fact, all moments can be computed recursively and very cheaply, see Section 4 for details. To our knowledge, the first approach in this direction was given by Gautschi et.al. [GGP]. However, their approach only works for refinable functions with some smoothness whereas in our setting we do not need this assumption. Moreover, in [GGP] only refinable functions are considered while in this paper we also study the wavelet case. (Nevertheless, let us mention that simultaneously to our work, generalizations to these cases are in preparation [GS]). Furthermore, we also give an outline how our approach can be generalized to higher dimensions.

This paper is organized as follows. In Section 2, we give a very short summary of the basic facts from wavelet analysis. In Section 3, we briefly recall the setting of Gauss quadrature as far as it is needed for our purposes. In Section 4, we discuss the special features appearing in the application of Gauss quadrature in the wavelet setting. Then, in Section 5, we apply the theory developed so far to the important special case of cardinal B–splines. In Section 6, we present some ways how our approach can be generalized to higher dimensions. This turns out to be quite easy in the tensor product case, however, leaving the Gauss quadrature setting, the general case is also briefly discussed. We have tested our method for some important model problems in one and two dimensions, and the results are presented in Section 7. It turns out that the performance was quite satisfactorily in all cases. Finally, in Section 8, we discuss the application of these quadrature rules to some practical problems. So far, it seems that the approach presented here works very efficiently for the computation of farfield integrals arising in boundary element methods. We therefore discuss one typical example in this regard in detail.

2 The wavelet setting

In this section, we shall briefly recall the basic setting of wavelet analysis as far as it is needed for our purposes. In general, a function ψ is called a *wavelet* if all its scaled, dilated, and integer translated versions

$$\psi_{j,k}(x) := 2^{j/2}\psi(2^j x - k), \quad j,k \in \mathbb{Z}, \tag{1}$$

form a Riesz basis of $L_2(\mathbb{R})$. Usually, they are constructed by means of a *multiresolution analysis* introduced by Mallat [M] which is a nested sequence $\{V_j\}_{j\in\mathbb{Z}}$ of closed subspaces of $L_2(\mathbb{R})$ such that their union is dense while their intersection is zero. Furthermore, one assumes that the spaces $\{V_j\}_{j\in\mathbb{Z}}$ are related by

$$f(\cdot)\in V_j \Longleftrightarrow f(2\cdot)\in V_{j+1}, \tag{2}$$

and that the space V_0 is shift–invariant and spanned by the integer translates of one single function φ called the *generator* of the multiresolution analysis. Assuming that φ has *stable* integer translates, it can be shown that φ is *refinable*, i.e., it satisfies a *two–scale relation*

$$\varphi(x)=\sum_{k\in\mathbb{Z}} a_k\varphi(2x-k) \tag{3}$$

with the *mask* $\mathbf{a}=\{a_k\}_{k\in\mathbb{Z}}\in\ell_2(\mathbb{Z})$. In the sequel, we shall restrict ourselves to refinable functions with compact support and we will always assume that

$$\operatorname{supp}(\mathbf{a}):=\{k\in\mathbb{Z}|a_k\neq 0\}\subseteq[m_1,m_2]. \tag{4}$$

It can be checked that (4) implies $\operatorname{supp}\varphi\subseteq[m_1,m_2]$. Because the union of the spaces $\{V_j\}_{j\in\mathbb{Z}}$ is dense in $L_2(\mathbb{R})$, it is easy to see that the construction of a wavelet basis reduces to finding a function whose translates span a complement space W_0 of V_0 in V_1. Hence (2) implies that the wavelet ψ can be found by means of a functional equation of the form

$$\psi(x)=\sum_{k\in\mathbb{Z}} b_k\varphi(2x-k), \tag{5}$$

where the sequence $\mathbf{b}:=\{b_k\}_{k\in\mathbb{Z}}$ has to be judiciously chosen, see, e.g., [Ch, Dau, Me] for details. We shall henceforth always assume that $\operatorname{supp}(\mathbf{b})\subset[n_1,n_2]$, so that the resulting wavelet is compactly supported.

There are several methods to generalize this concept to higher dimensions. The simplest way is to use *tensor products*. Given a univariate multiresolution analysis with generator φ, it turns out that

$$\phi(x_1,\ldots,x_d):=\varphi(x_1)\ldots\varphi(x_d) \tag{6}$$

generates a multiresolution analysis of $L_2(\mathbb{R}^d)$. Let E denote the vertices of the unit cube in $\mathbb{R}^d$. Defining $\psi^0:=\varphi$ and $\psi^1:=\psi$, it can be shown that the set Ψ of 2^d-1 functions

$$\psi^e(x_1,\ldots,x_d):=\prod_{l=1}^{d}\psi^{e_l}(x_l) \qquad e=(e_1,\ldots,e_d)\in E\backslash\{0\}, \tag{7}$$

generates by shifts and dilates a basis of $L_2(\mathbb{R}^d)$. There also exist multivariate wavelet constructions with respect to *non–separable* refinable functions ϕ satisfying

$$\phi(x)=\sum_{k\in\mathbb{Z}^d} a_k\phi(2x-k), \qquad \{a_k\}_{k\in\mathbb{Z}^d}\in\ell_2(\mathbb{Z}^d), \tag{8}$$

see, e.g., [JM] for details. Analogously to the tensor product case, a family ψ^i, $i=1,\ldots,2^d-1$, of wavelets is needed. Each ψ^i satisfies a functional equation similar to (5),

$$\psi^i(x)=\sum_{k\in\mathbb{Z}^d} b_k^i\phi(2x-k). \tag{9}$$

As in the univariate case, we shall henceforth assume that the scaling functions and wavelets under consideration are all compactly supported.

In practice, it is often very convenient to have access to a suitable *biorthogonal* wavelet basis. For a given (univariate) wavelet basis $\{\psi_{j,k},\ j,k \in \mathbb{Z}\}$, one is interested in finding a second system $\{\tilde{\psi}_{j,k},\ j,k \in \mathbb{Z}\}$ satisfying

$$(\psi_{j,k}(\cdot), \tilde{\psi}_{j',k'}(\cdot)) = \delta_{j,j'}\delta_{k,k'}, \quad j,j',k,k' \in \mathbb{Z}. \tag{10}$$

Here $(\cdot,\cdot)$ clearly denotes the usual L_2–inner product. The construction of such a biorthogonal system essentially relies on a suitable second generator $\tilde{\varphi}$ such that φ and $\tilde{\varphi}$ form a *dual pair*,

$$(\varphi(\cdot), \tilde{\varphi}(\cdot - k)) = \delta_{0,k}. \tag{11}$$

Elegant constructions can be found, e.g., in [CDF]. Generalizations to higher dimensions also exist [CD].

For further information on wavelet analysis, the reader is referred to one of the excellent textbooks on wavelets which have appeared quite recently [Ch, D, Dau, KL, Me, W].

3 Gauss quadrature

A univariate Gauss quadrature rule replaces an integral by a weighted sum of point evaluations of f, i.e.,

$$\int\limits_{[a,b]} f(x)w(x)dx \approx I_w^n(f) := \sum_{i=1}^{n} \lambda_i f(x_i), \qquad [a,b] \subset \mathbb{R}, \quad n \in \mathbb{N}, \tag{12}$$

with *knots* x_i and *weights* λ_i, $i = 1,\ldots n$. Here $w(x)$ is called the *weight function*, which in the most classical case is chosen to be $w(x) \equiv 1$. However, the theory can be developed for a hugh class of functions. In fact, in order to get an appropriate formula, $w(\cdot)$ is in general only required to be non–negative.

Defining the inner product corresponding to $w(\cdot) \geq 0$ as

$$(f,g)_w := \int\limits_{[a,b]} f(x)g(x)w(x)dx,$$

the knots x_i of the Gauss rule are the zeros of n–th orthogonal polynomial π_n with respect to this scalar product. The orthogonal polynomials satisfy the following three–term recurrence relation

$$\pi_{k+1}(x) = (x - \alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x),\ k = 0,\ldots,n-1, \tag{13}$$

$$\pi_{-1}(x) = 0,\ \pi_0(x) = 1.$$

The coefficients α_k, β_k can be computed as

$$\alpha_k = \frac{(x\,\pi_k, \pi_k)_w}{(\pi_k,\pi_k)_w}, \quad \beta_k = \frac{(\pi_k,\pi_k)_w}{(\pi_{k-1},\pi_{k-1})_w}. \tag{14}$$

Note that all α_k vanish if the weight function is symmetric to the origin.

The computation of the Gauss quadrature rule according to a specific weight function w can be done in several ways, see, e.g., [DR], Chapter 2.7. One classical method is the following. The first step is to compute the recursion coefficients α_k, β_k according to (14) and (13). In a second step, solving an eigenvector/eigenvalue problem with a tridiagonal matrix consisting of the coefficients α_k, β_k, leads to the desired values for x_i and λ_i, see, e.g., [DR, G].

Let us define the Gauss quadrature error on the interval $[a,b]$ by

$$E_w^n(f) := \Big| \int\limits_{[a,b]} f(x)w(x)dx - I_w^n(f)\Big|. \tag{15}$$

A classical result states that if $f \in C^{2n}(I\!R)$ and $w(\cdot) \geq 0$, then

$$E_w^n(f) = \frac{f^{(2n)}(\xi)}{(2n)!k_n^2}, \qquad \text{for some } a < \xi < b, \tag{16}$$

see again [DR] for details. Here k_n denotes the leading coefficient of the n-th orthonormal polynomial with respect to the scalar product $(\cdot,\cdot)_w$. In the case that w is symmetric to the origin, we have

$$k_n = \left(\prod_{i=0}^{n} \left(\pi_i(\cdot), \pi_i(\cdot)\right)_w \right)^{-1}.$$

A quadrature rule is said to be of *degree* m if it is exact for all polynomials with order up to m. Gauss quadrature rules with n points are of degree $m = 2n$, cf. (16).

Generalizations to higher dimensions also exist. For certain integration domains such as rectangles, Gauss quadrature rules can be derived by a product formula made up of univariate Gauss formulas. We will come back to this issue in Section 6.1. In certain cases, it is also possible to construct non–product formulas, which will be discussed in Section 6.2.

For further information on quadrature, the reader is referred, e.g., to [DR, K, St] and the references therein.

4 Gauss quadrature in the wavelet context

Whenever wavelet analysis is used in practice, sooner or later one has to compute inner products of the form

$$(f, \theta_{j,k}) = \int\limits_{\text{supp}(\theta_{j,k})} f(x)\theta_{j,k}(x)\, dx, \tag{17}$$

where f is some given function and $\theta_{j,k}$ denotes a wavelet $\psi_{j,k}(x)$ according to (1) or an associated dilated and translated version of a refinable function,

$$\varphi_{j,k}(x) := 2^{j/2}\varphi(2^j x - k), \quad j,k \in Z\!\!\!Z, \tag{18}$$

cf. Section 2. Integrals of the form (17) occur quite naturally in image/signal processing when projections of the signal f onto the spaces V_j and W_j have to be computed. They also have to be dealt with when wavelets are used as basis functions in a Galerkin approach to solve, e.g., an elliptic operator equation. In fact, the entries in the right–hand side of the resulting linear system are of the form (17), compare with Section 8.

By means of a substitution, integrals of the form (17) can be transfered to integrals involving only θ,

$$\int\limits_{\mathrm{supp}(\theta_{j,k})} f(x)\theta_{j,k}(x)\,dx \;=\; 2^{-j/2}\int\limits_{\Omega} f(2^{-j}(u+k))\,\theta(u)\,du, \tag{19}$$

where $\Omega := \mathrm{supp}(\theta)$. Recall that in either case Ω is assumed to be compact, cf. Section 2. Note that if we have determined quadrature rules on Ω for

$$\int\limits_{\Omega} f(x)\theta(x)\,dx, \quad \theta\in\{\varphi,\psi\}, \tag{20}$$

all integrals of the form (17) can be computed by the substitution (19). Hence, knots and weights have to be calculated only once and for all on Ω. Therefore, the remaining question is how to find appropriate rules for the integrals in (20).

4.1 The general idea

A classical way to deal with integrals of the form (20) is to use a usual Gauss quadrature, namely to choose $f(\cdot)\theta(\cdot)$ as the integrand and $w(x)\equiv 1$ as the weight function. But even if $f\in C^\infty$, this method might run into trouble because in general θ needs not to be very smooth. In special cases, such as the cardinal B–splines, see Section 5, it is possible to get around this problem by dividing the integration domain D into subdomains $D_l, l=1,\dots,k$, on which the integrand is sufficiently smooth and to apply a Gauss rule on each of these. Such a scheme is referred to as a *composite quadrature rule.* They are in general quite expensive, because in order to achieve a formula of degree $2m,\ m\in\mathbb{N}$, on D, one has to perform km point evaluations. Therefore we will present an alternative approach to circumvent the smoothness problem.

Instead of using the classical Gaussian rule, a modified formula according to $w(x)=\theta(x),\ \theta\in\{\varphi,\psi\}$ is used. As the weight function is only required to be non–negative, the lacking smoothness of θ is no longer a problem in the sense that the performance of this Gauss quadrature only depends on the number of knots and the regularity of f, but not on the smoothness of w as one can see from (16). Hence, if $f\in C^{2n},\ n\in\mathbb{N}$, Gauss quadrature rules of degree $2m, m\le n$ can be achieved with m knots.

All in all the approximation of integrals of the form (20) boils down to the construction of a Gauss rule for the weight function $w(x)=\theta(x)$, e.g, according to the method described in Section 3. Hence the remaining key ingredient is the efficient computation of integrals of polynomials against refinable functions and wavelets, respectively.

Before we turn to this issue in the next subsection, let us briefly deal with the following. Viewing θ as a weight function in general requires θ to be non–negative. This covers important scaling functions such as the cardinal B–splines or the box splines, see Sections 5 and 7.2. However, there are prominent cases of scaling functions and wavelets not being non–negative, such as the dual scaling functions and the wavelets according to the above cases. This problem can be fixed with the following *lifting trick*. If $\theta \not\geq 0$ one can find an appropriate constant $c > 0$ such that on $\operatorname{supp}(\theta)$

$$\theta^c(x) := \theta(x) + c\chi_{[l_1,l_2]}(x) \geq 0, \tag{21}$$

where $\chi_{[l_1,l_2]}$ denotes the characteristic function of $[l_1, l_2] \supseteq \operatorname{supp}(\theta)$, cf. (4) and (5). Obviously, the moments of $\chi_{[l_1,l_2]}$ can be computed as

$$\gamma_\beta(\chi_{[l_1,l_2]}) := \int\limits_{l_1}^{l_2} x^\beta \, dx = \frac{l_2^{\beta+1} - l_1^{\beta+1}}{\beta+1}, \quad \beta \in I\!N \cup \{0\}.$$

Using the methods described in Section 4.2, the moments of θ are also accessable. From this the moments of θ^c are easily computable and the corresponding quadrature rule with weights λ_i^c and knots x_i^c can be determined. Then a second Gauss rule with weights λ_i^χ and knots x_i^χ for the characteristic function $\chi_{[l_1,l_2]}$ has to be set up using its already computed moments. The sum (12) is then in fact replaced by the difference of two sums

$$\int\limits_\Omega f(x)\theta(x)dx \approx I_\theta^n(f) = \sum_{i=1}^n \lambda_i^c f(x_i^c) - c\sum_{i=1}^n \lambda_i^\chi f(x_i^\chi). \tag{22}$$

To reduce effects of cancellation, c should be chosen as small as possible. In the case that θ is explicitly known, this is no problem. But even if it is given only implicitly by means of the mask, by employing the algorithm in [DM2], point values are still accessable, at least on a dyadic grid. Therefore c can be determined by a simple plot. This might be expensive when it comes to wavelets with large supports or to higher dimensions, but recall that this has to be done only once and for all.

Remark 4.1 *Let us briefly discuss a slightly different way to deal with integrals involving wavelets. One could apply the above quadrature method only to the case $\theta = \varphi$ and treat the wavelet case by using the scaling function representation* (5) *of ψ. This results in a quadrature formula made up of $k = n_2 - n_1 + 1$ quadrature rules for φ, and hence invoking kn point evaluations of f in a formula of degree $2n$. This is in general more expensive then the ansatz proposed here, where the number of point evaluations for a wavelet is at most $2n$, independent of the length of* $\mathbf{b}$ *in* (5)*.*

4.2 Computing the moments

We are left with the determination of Gauss quadrature rules for a non–negative weight function $\theta^c \in \{\varphi^c, \psi^c\}$, (21). This can be done e.g. according to Section 3. Taking the above lifting trick in account, the key step of this construction is the computation of the moments of θ. In some special cases this can be done directly, e.g. in the case of cardinal B–spline wavelets, see Section 5. However, in general, this is not the case. As one possible solution, in [GGP] Simpson's quadrature rules are applied. By consequence, this approach is limited to 'smooth' refinable functions. But many scaling functions do not allow quadrature rules to compute their moments because their regularity is too low. We therefore suggest in the following a different method avoiding quadrature.

The computation of integrals with monomials can be done efficiently and (up to round off) exactly by using recursion relations, see [DM2]. For reader's convenience, we briefly recall the basic ideas. As the method can be easily described for the multivariate case and as we will need it later in Section 6 in this shape, let us present the scheme in this general form. We start with the case of an arbitrary refinable function ϕ in d dimensions. Recall, that we always assume $\operatorname{supp}\phi$ and $\operatorname{supp}\psi^i, i \in I$, to be compact in $\mathbb{R}^d$. Let us define

$$\gamma_\beta := \int\limits_{\mathbb{R}^d} x^\beta \phi(x)dx, \quad \beta = (\beta_1, \ldots, \beta_d), \quad x^\beta = x_1^{\beta_1} \ldots x_d^{\beta_d}. \tag{23}$$

The normalization

$$\int\limits_{\operatorname{supp}(\phi)} \phi(x)dx = 1$$

leads to $\gamma_0 = 1$. Defining

$$c_\alpha^\beta := 2^{-|\beta|-d} \binom{\beta}{\alpha} \sum_{k \in \mathbb{Z}^d} a_k k^\alpha, \quad |\beta| := \sum_{l=1}^d \beta_l,$$

we obtain

$$\gamma_\beta = \sum_{0 \le \alpha \le \beta} c_{\beta-\alpha}^\beta \gamma_\alpha = c_0^\beta \gamma_\beta + \sum_{0 \le \alpha < \beta} c_{\beta-\alpha}^\beta \gamma_\alpha. \tag{24}$$

Here $\alpha < \beta$ means $\alpha_j \le \beta_j,\ j = 1, \ldots, d$, and $\alpha_i < \beta_i$ for at least one $i \in \{1, \ldots, d\}$. Hence we end up with the recursion

$$\gamma_\beta = (1 - 2^{-|\beta|}) \sum_{0 \le \alpha < \beta} c_{\beta-\alpha}^\beta \gamma_\alpha. \tag{25}$$

Now we want to use the method presented above to compute integrals involving wavelets instead of scaling functions. Since wavelets are linear combinations of translated versions of refinable functions on the next higher level, compare with (9), this task is not too difficult. The integrals of wavelets with monomials can be computed as follows

$$\begin{aligned}
\int\limits_{\operatorname{supp}(\psi^i)} x^\beta \psi^i(x)dx &= \int\limits_{\mathbb{R}^d} x^\beta \sum_{k \in \mathbb{Z}^d} b_k^i\, \phi(2x-k)dx = \frac{1}{2^{|\beta|+d}} \int\limits_{\mathbb{R}^d} x^\beta \sum_{k \in \mathbb{Z}^d} b_k^i\, \phi(x-k)dx \\
&= \frac{1}{2^{|\beta|+d}} \sum_{k \in \mathbb{Z}^d} b_k^i \int\limits_{\mathbb{R}^d} \left(\sum_{0 \le \alpha \le \beta} \binom{\beta}{\alpha} x^\alpha k^{\beta-\alpha} \right) \phi(x)dx \\
&= \frac{1}{2^{|\beta|+d}} \sum_{k \in \mathbb{Z}^d} \sum_{0 \le \alpha \le \beta} b_k^i \binom{\beta}{\alpha} k^{\beta-\alpha} \gamma_\alpha
\end{aligned} \tag{26}$$

where again $\beta = (\beta_1, \ldots, \beta_d)$.

5 Cardinal B–splines and their duals

In this section, we shall apply the theory investigated above to the case that the underlying refinable function is a cardinal B–spline or one of the corresponding dual functions, compare with Section 2. These special functions are frequently used in numerical analysis. In general, the *cardinal B–spline* $\mathcal{N}_\rho$ of order $\rho \in I\!N$ is a piecewise polynomial function which is defined recursively as a convolution product, i.e.,

$$\begin{aligned} \mathcal{N}_1(x) &:= \chi_{[0,1)}(x), \\ \mathcal{N}_\rho(x) &:= (\mathcal{N}_{\rho-1} * \mathcal{N}_1)(x) = \int_{I\!R} \mathcal{N}_{\rho-1}(x-y)\mathcal{N}_1(y)dy. \end{aligned} \tag{27}$$

Obviously, all these functions are positive so that we may use the algorithm outlined in Section 4 without using the lifting trick (22). We shall mainly be concerned with the B–spline $\mathcal{N}_2$. By an integer shift, we obtain the classical symmetric hat function

$$N_2(x) := \mathcal{N}_2(x+1) = \begin{cases} 1+x, & -1 \le x < 0, \\ 1-x, & 0 \le x \le 1, \\ 0, & \text{otherwise}. \end{cases} \tag{28}$$

We list the knots and weights for this case in Table A in the appendix. Let us remark that for the spline case these knots and weights can be determined without the recursion formula (24) because the moments γ_β can be computed directly. For instance, for the spline N_2, we obtain

$$\gamma_\beta = \begin{cases} 0, & \beta \text{ odd}, \\ \frac{2}{(\beta+1)(\beta+2)}, & \beta \text{ even}. \end{cases}$$

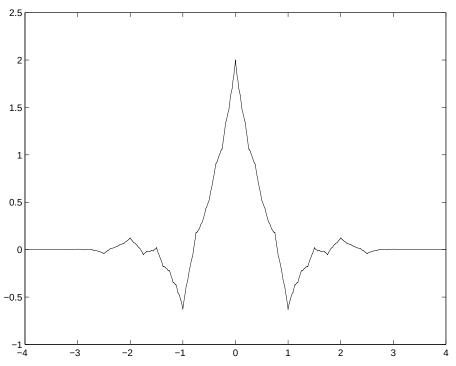
In the following table, we have listed the leading coefficients k_l, $l = 1, \ldots, 10$, of the orthonormal polynomials corresponding to N_2. We see that the sequence k_l, $l = 1, \ldots, 10$, approximately behaves like $2^{(l-1)}$. Because k_l^{-2} enters in the error estimate (16), we expect that the resulting quadrature rules perform very accurately as l increases. This conjecture is confirmed by our numerical experiments, see Section 7.

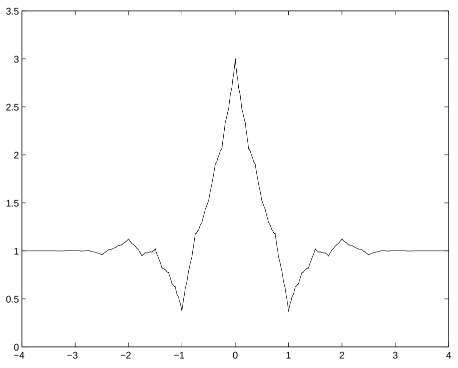
k_1	1.0000000000000000000000000000
k_2	2.449489742783178098197284074700
k_3	5.070925528371099465057709964190
k_4	10.513149660756936271463359120000
k_5	21.259740683707940951376792519400
k_6	43.169070646570870905645790976900
k_7	86.801073005286568361974542381700
k_8	175.063172739382196824342750901000
k_9	351.225962331516688455428359048000
k_{10}	706.201232618915412159855112503000

For cardinal B–splines, suitable dual generators are known. In fact, in [CDF], families of refinable functions $\tilde{\mathcal{N}}_{\rho,\tilde{\rho}}$, $\rho+\tilde{\rho}$ even, were constructed such that a biorthogonality relation of the form (11) is satisfied, i.e.,

$$\Big(\mathcal{N}_\rho(\cdot), \tilde{\mathcal{N}}_{\rho,\tilde{\rho}}(\cdot - k)\Big) = \delta_{0,k}. \tag{29}$$

We also want to derive quadrature rules using these dual functions as weight functions. In general, they are no longer positive so that we have to work with the lifting trick (22) in this case. Once again, we shall focus on the case $\rho = 2$. The next figure shows both, the (centralized) dual function $\tilde{N}_{2,4}$ which is biorthogonal to the usual hat function (28), and its lifted version $\tilde{N}^1_{2,4}$.

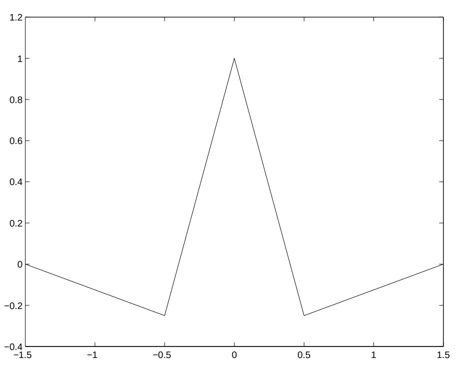

$\tilde{N}_{2,4}$


$\tilde{N}^1_{2,4} := \tilde{N}_{2,4} + \chi_{[-4,4]}$

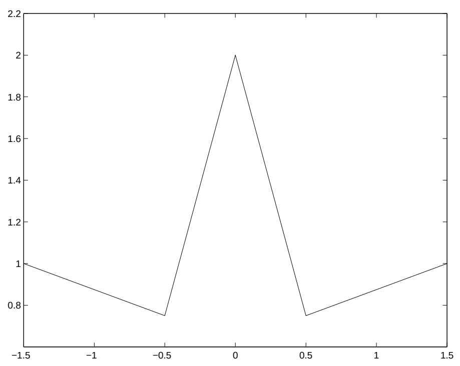
In the next table we present the recursion coefficients β_k entering the construction of the two rules needed in the lifting trick, namely the one for $\tilde{N}^1_{2,4}$ and the one for the characteristic function $\chi_{[-4,4]}$.

k	$\beta_k(\tilde{N}^1_{2,4})$	$\beta_k\left(\chi_{[-4,4]}\right)$	k	$\beta_k(\tilde{N}^1_{2,4})$	$\beta_k\left(\chi_{[-4,4]}\right)$
0	9.00000000000000	8.00000000000000	5	3.43508129707940	4.04040404040410
1	4.72222222222222	5.33333333333333	6	4.56778842439712	4.02797202797139
2	4.91777777777778	4.26666666666666	7	3.70208113647468	4.02051282051267
3	3.50852646887946	4.11428571428572	8	4.10494370822309	4.01568627450075
4	4.74217628189890	4.06349206349206	9	4.10652830708695	4.01238390088814

Once a dual generator is found, a biorthogonal wavelet basis satisfying (10) can be easily constructed, see again [CDF] for details. By using the lifting trick one more time, we may determine a quadrature formula according to Section 4 for the weight function ψ. In the next figure, we have depicted the primal wavelet $\psi_{2,2}$ corresponding to N_2 and $\tilde{N}_{2,2}$ in the original and in the lifted form.



$\psi_{2,2}$



$\psi^1_{2,2} := \psi_{2,2} + \chi_{[-1.5,1.5]}$

The next tabular lists the coefficients β_k for the lifted wavelet and for the characteristic function $\chi_{[-1.5,1.5]}$.

k	$\beta_k(\psi^1_{2,2})$	$\beta_k(\chi_{[-1.5,1.5]})$	k	$\beta_k(\psi^1_{2,2})$	$\beta_k(\chi_{[-1.5,1.5]})$
0	3.0000000000000	3.0000000000000	5	0.5371914738074	0.5681818181818
1	0.6616116523517	0.7500000000000	6	0.5688216835388	0.5664335664335
2	0.7351467769579	0.6000000000000	7	0.5777163727584	0.5653846153843
3	0.4703057627907	0.5785714285714	8	0.5494010683153	0.5647058823531
4	0.6434383764919	0.5714285714286	9	0.5746251705822	0.5642414860593

We want to finish this section with another remark. As already stated above, the cardinal B–spline $\mathcal{N}_\rho$ is a piecewise polynomial function and its singular support can be determined exactly. Therefore another method frequently used in practice is to subdivide the support of $\mathcal{N}_\rho$ and of the corresponding primal wavelet according to their singular supports and to employ a usual Gauss quadrature (with $w \equiv 1$) on each of the resulting subintervals. However, this approach is only applicable to the primal functions and has the serious disadvantage that the number of subdivisions, which is proportional to the numerical cost, grows like ρ for the generator and like $\rho + \tilde{\rho}$ for the corresponding primal wavelet. Moreover, in higher dimensions the number of subdivisions grows exponentially with the spatial dimension d. In contrary to this, the method presented here requires only the evaluation of one integral independent of the order of the wavelet and of the scaling function.

6 The multivariate case

In this section, we investigate how the approach outlined above can be generalized to *multivariate* refinable functions and wavelets. We first discuss the important special case of scaling functions and wavelets of tensor product type, i.e.,

$$\Theta(x_1, \ldots, x_d) = \theta_1(x_1)\theta_2(x_2)\cdots\theta_d(x_d), \tag{30}$$

where $\theta_i,\ i=1,\ldots,d$, denote univariate refinable functions or wavelets, compare with Section 2. It turns out that this case can in fact be reduced to the univariate setting presented above. Moreover, we also briefly explain how general scaling functions and wavelets can be handled.

6.1 The tensor product case

Let us first assume that all the $\theta_i,\ i=1,\ldots,d$, on the right–hand side of (30) are non–negative. We want to find a quadrature formula of degree $N, N \in \mathbb{N}$, of the form

$$\int_{\Omega_1}\cdots\int_{\Omega_d} f(x_1,\ldots,x_d)\theta_1(x_1)\cdots\theta_d(x_d)\,dx_1\cdots dx_d \sim \sum_{i=1}^{n(N)} \lambda_i f(x_{i,1},\ldots,x_{i,d}). \tag{31}$$

Here $\Omega_i=\operatorname{supp}(\theta_i),\ i=1,\ldots,d$. It is easy to derive such a formula from the univariate rule constructed in Sections 3 and 4 by using the following general result shown in [St]. Let $\mathbb{R}^d \supset I = I_1 \times I_2,\ I_1 \subset \mathbb{R}^{d_1},\ I_2 \subset \mathbb{R}^{d_2}$ be a cube. Suppose that we are given quadrature rules on I_1 and I_2 with respect to weight functions g_1 and g_2, respectively, and with points and weights

$$(x_{i,1},\ldots,x_{i,d_1}),\ \lambda_i,\ i=1,\ldots,n_1, \quad \text{and} \quad (x_{j,d_1+1},\ldots,x_{j,d}),\ \mu_j,\ j=1,\ldots,n_2,$$

which are of degree N_1 and N_2, respectively. Let g be defined by

$$g(x_1,\ldots,x_d)=g_1(x_1,\ldots,x_{d_1})g_2(x_{d_1+1},\ldots,x_d).$$

Then the following theorem holds, see [St], Chapter 2, Theorem 2.3–1.

Theorem 6.1 *The $n=n_1n_2$ points and weights*

$$(x_{i,1},\ldots,x_{i,d_1},x_{j,d_1+1},\ldots,x_{j,d}),\ \lambda_i\mu_j,\ i=1,\ldots,n_1,\ j=1,\ldots,n_2,$$

are an integration formula for the weight function g on I which is of degree $N=\min(N_1,N_2)$.

Applying Theorem 6.1 $(d-1)$ times ends up in a product quadrature formula for a function Θ of the form (30).

In case that the functions $\theta_i,\ i=1,\ldots,d$, are not non–negative, things are slightly more complicated. Nevertheless, it turns out that one can again use the lifting trick (22). To avoid unnecessary technical difficulties, we shall only write down the details for the case $d=2$. The higher dimensional cases can be treated analogously. Let us consider the following decomposition

$$\begin{aligned}\int_{\Omega_1}\int_{\Omega_2} f(x_1,x_2)\theta_1(x_1)\theta_2(x_2)\,dx_1dx_2 &= \int_{\Omega_1}\int_{\Omega_2} f(x_1,x_2)(\theta_1(x_1)+c_1)(\theta_2(x_2)+c_2)\,dx_1dx_2\\ &\quad -\int_{\Omega_1}\int_{\Omega_2} f(x_1,x_2)(\theta_1(x_1)+c_1)c_2\,dx_1dx_2\end{aligned}$$

$$- \int_{\Omega_1} \int_{\Omega_2} f(x_1, x_2)(\theta_2(x_2) + c_2)c_1 \; dx_1 dx_2$$

$$+ \int_{\Omega_1} \int_{\Omega_2} f(x_1, x_2) c_1 c_2 \; dx_1 dx_2. \tag{32}$$

By using Theorem 6.1 for each term on the right–hand side one can construct a quadrature formula made up of univariate rules. Note that if c_1 and c_2 are appropriately chosen, all weight functions for these cases are positive.

6.2 The general case

Let us now consider the case that Θ is a multivariate scaling function or a multivariate wavelet which is not necessarily of the form (30). Then there is no way to use a product formula according to Theorem 6.1 and therefore we have to derive directly a (non–product) quadrature rule. Leaving the usual Gauss quadrature setting, this is possible, e.g., by employing the results from [St]. Let us briefly recall the ideas. We shall focus on the 2D–case, for the general case, the reader is again referred to [St]. Let us first assume that Θ is non–negative. We again want to find a quadrature rule of degree N having the form

$$\int_{\mathrm{supp}\Theta} f(x,y)\Theta(x,y) \; dxdy \sim \sum_{i=1}^{n(N)} \lambda_i f(x_i, y_i). \tag{33}$$

To this end, one has to choose the points

$$(x_i, y_i), \quad i = 1, \dots, n, \; n = n(N) = \frac{(2+N)(1+N)}{2} \tag{34}$$

in such a way that they do not all lie on a curve $Q_N(x,y) = 0$, where Q_N is a polynomial of degree N. Such a set of points can always be found, see, e.g., [St], Ch. 1, Theorem 1.8–3 for details. Under this condition, the $(n \times n)$–matrix with rows

$$(x_1^{\beta_1} y_1^{\beta_2}, \dots, x_n^{\beta_1} y_n^{\beta 2}), \quad 0 \le \beta_1 + \beta_2 \le N$$

is non–singular and the coefficients λ_i can be found as solutions of the linear system

$$\lambda_1 x_1^{\beta_1} y_1^{\beta_2} + \dots + \lambda_n x_n^{\beta_1} y_n^{\beta_2} = \int_{\mathrm{supp}\Theta} x^{\beta_1} y^{\beta_2} \Theta(x,y) \; dxdy, \qquad 0 \le |\beta| \le N. \tag{35}$$

Let us stress that again the right–hand side in (35) can be computed exactly without any regularity assumption on Θ, compare with Section 4.2. Scaling functions and wavelets taking negative values can be handled by using the lifting trick.

Remark 6.2 *It might be somewhat expensive to construct non–product quadrature rules of a very high degree, for then a huge linear system has to be assembled and to be solved. However, the reader should again observe that these expensive computations have only to be done once and for all.*

7 Numerical examples

Once the quadrature rules for refinable functions and wavelets are established, one clearly wishes to see how they behave in practice. Therefore we have tested our formulas for several model problems, and the results are presented in this section. In Subsection 7.1, the univariate formulas are discussed. We use the B–splines, their duals and the corresponding wavelets as introduced in Section 5 as weight functions. Furthermore, in Subsection 7.2, we test the multivariate rules. We focus on the non–product formulas constructed in Section 6.2 and we use box splines as weight functions. It turned out that the performance was quite satisfactorily in all cases.

7.1 Univariate examples

To test the whole procedure, we have applied it to the computation of the integrals

$$\int_{-4}^{4} e^x \tilde{N}_{2,4}(x)dx \qquad \text{and} \qquad \int_{-1}^{1} e^x N_2(x)dx.$$

The results are displayed in the following table.

n	$I^n_{\tilde{N}_{2,4}}$	$I^n_{N_2}$	n	$I^n_{\tilde{N}_{2,4}}$	$I^n_{N_2}$
1	1.0000000000000	1.00000000000000	6	0.9233247035827	1.08616126977393
2	-0.6252681537938	1.08449718995440	7	0.9233379806840	1.08616126974154
3	0.9543890162969	1.08614841390864	8	0.9233380192360	1.08616126953389
4	0.9088372144081	1.08616121370867	9	0.9233380209880	1.08616126997847
5	0.9234686807508	1.08616126928519	10	0.9233380212325	1.08616126988915

For the second integral, we can investigate the quality of the process somewhat more precisely because in this case the exact solution $e + e^{-1} - 2 \approx 1.086161269630487$ can be easily computed. It turns out that already for $n = 5$ the error is about 10^{-10}. Moreover, this example enables us to test the error estimate (16). Indeed, since the exponential function is strictly increasing on $[-1, 1]$, we obtain the rough estimate

$$E_n^{N_2}(e^x) \leq \frac{e}{(2n)!k_n^2}. \tag{36}$$

In the following table, we have compared this estimate with the true error. We see that (36) is in general too pessimistic by one power of ten.

n	$\lvert \int_{-1}^{1} e^x N_2(x)dx - I_{N_2}^n \rvert$	$((2n)!k_n^2)^{-1}e$
1	0.086161269630487	1.3591409
2	0.00166408	0.037753914
3	0.000012856	0.0001468207
4	0.000000056	0.0000006099697
5	$\sim 10^{-10}$	0.000000001657354

We have also tested the quadrature formula obtained for the B–spline wavelet $\psi_{2,2}$ as the weight function. We used it to approximate the integral

$$\int_{\mathbb{R}} e^x \ \psi_{2,2}(x)dx.$$

Since an explicit representation of the wavelet is known, it is again easy to compute the exact value of the integral for comparison. The results, showing a rapid decrease of the error, are displayed in the next table.

n	$I_{\psi_{2,2}}^n$	$\lvert \int_{\mathbb{R}} e^x\psi_{2,2}dx - I_{\psi_{2,2}}^n \rvert$	n	$I_{\psi_{2,2}}^n$	$\lvert \int_{\mathbb{R}} e^x\psi_{2,2}dx - I_{\psi_{2,2}}^n \rvert$
1	0.0000000000000	1.440913408e-01	6	-0.1440913412020	3.978017359e-10
2	-0.1487392673555	4.647926551e-03	7	-0.1440913405939	2.102940599e-10
3	-0.1439509995222	1.403412819e-04	8	-0.1440913412644	4.602584424e-10
4	-0.1440912053939	1.354102581e-07	9	-0.1440913407264	7.778050425e-11
5	-0.1440913387957	2.008493871e-09	10	-0.1440913404450	3.591145714e-10

7.2 Multivariate examples

In this subsection we present tests concerning the general multivariate approache described in Section 6.2. The discussion of the tensor product case as explained in Section 6.1 is postponed to Section 8, where we study a practical application in accordance to this in detail.

The method for non tensor product weight functions has been applied to the maybe most prominent family of non–separable multivariate scaling functions, i.e., to box splines. Let us briefly recall the definition and the basic facts. Let
$X = \{x^1, \ldots, x^\mu\} \subset \mathbb{Z}^d \backslash \{0\}$, $x^l = (x_1^l, \ldots, x_d^l)^T$ denote a set of not necessarily distinct vectors satisfying $\mu \geq d$ and

$$< X >= \text{span } X = \mathbb{R}^d. \tag{37}$$

Then the *box spline* $B(\cdot|X)$ is defined by requiring that the equation

$$\int_{\mathbb{R}^d} f(x)B(x|X)dx = \int_{[0,1]^\mu} f(Xu)du \tag{38}$$

holds for any continuous function f on $\mathbb{R}^d$. The vectors $x^1, \ldots, x^\mu$ are called the *direction vectors* of $B(\cdot|X)$. Every box spline is a refinable function,

$$B(\cdot|X) = \sum_{k\in\mathbb{Z}^d} a_k B(2\cdot -k|X), \tag{39}$$

where the mask $\mathbf{a} = \{a_k\}_{k\in\mathbb{Z}^d}$ is given by

$$\sum_{k\in\mathbb{Z}^d} a_k z^k = 2^{d-\mu}\prod_{l=1}^{\mu}(1+z^{x^l}). \tag{40}$$

It can be shown that a box spline is a piecewise polynomial. Moreover, it is positive and satisfies

$$B(u|X) = 0, \quad u \notin [X[, \ [X[:= \{t_1x^1 + \cdots + t_\mu x^\mu : \ 0 \le t_l < 1, \ 1 \le l \le \mu\}. \tag{41}$$

For further information on box splines, the reader is referred to [DM].

We set up a simple algorithm to generate a specific set of $n(N)$ points in the domain of integration according to (34) and to construct the corresponding non–product quadrature rule for a bivariate weight, cf. Section 6.2. In our test examples, the resulting quadrature rules were surprisingly weakly dependent on the location of these points, however, the algorithm used in the following examples tends to equidistribute them.

First of all, we applied our algorithm to the famous *Courant finite element* $B(\cdot\,|\,{}^{1\,0\,1}_{0\,1\,1})$ i.e., to the bivariate box spline with direction vector $\binom{1}{0}, \binom{0}{1}, \binom{1}{1}$. The graph of this function looks like a hexagonal pyramid. According to (41), its support is contained in the cube $[0,2]\times[0,2]$. In Table B in the appendix, we have listed the resulting Gauss points and the corresponding weights for this cube as N and therefore $n(N)$ increases, cf. (34). With these points and weights, we have tested the method described in Section 6.2 for the function

$$f(x,y) = \sin(x+y).$$

The results are depicted in the following table.

N	n	$I^n_{B(\cdot\,|\,{}^{1\,0\,1}_{0\,1\,1})}$	N	n	$I^n_{B(\cdot\,|\,{}^{1\,0\,1}_{0\,1\,1})}$
1	3	0.29788436073635405	7	36	0.70347319558925836
2	6	0.70724512563270003	8	45	0.70347294122562543
3	10	0.70534367984966273	9	55	0.70347299009047892
4	15	0.70336387961090840	10	66	0.70347299263743424
5	21	0.70345600023785304	11	78	0.70347299204674396
6	28	0.70347597675687457	12	91	0.70347299203136016

We have also tested our method for the tensor product cardinal B–spline $\mathcal{N}_2 \otimes \mathcal{N}_2(x,y)$, i.e., for the bivariate box spline with direction vectors $\binom{1}{0}, \binom{1}{0}, \binom{0}{1}, \binom{0}{1}$. We applied the resulting quadrature rule to the functions

$$f(x,y) = \sin(x+y) \qquad \text{and} \qquad g(x,y) = \exp(x+2y).$$

In both cases, the exact solutions can be easily computed. Consequently, we are able to calculate the errors of the quadrature rules. It seems to be most meaningful to consider the relative errors

$$\mathcal{E}(n, f, \mathcal{N}_2 \otimes \mathcal{N}_2) := \frac{|\int (f(x,y)\mathcal{N}_2 \otimes \mathcal{N}_2(x,y)\,dxdy - I^n_{\mathcal{N}_2 \otimes \mathcal{N}_2}|}{|\int f(x,y)\mathcal{N}_2 \otimes \mathcal{N}_2(x,y)\,dxdy|}. \tag{42}$$

The results are listed below.

N	n	$I^n_{\mathcal{N}_2 \otimes \mathcal{N}_2}(f)$	$\mathcal{E}(n, f, \mathcal{N}_2 \otimes \mathcal{N}_2)$	$I^n_{\mathcal{N}_2 \otimes \mathcal{N}_2}(g)$	$\mathcal{E}(n, g, \mathcal{N}_2 \otimes \mathcal{N}_2)$
2	6	0.764912463322939	$0.4821e-02$	29.787140498052480	$0.1139e-1$
3	10	0.768230405045909	$0.5043e-03$	30.254584373509807	$0.4128e-2$
4	15	0.768619073231594	$0.1273e-05$	30.131265424222669	$0.3492e-4$
5	21	0.768611816978489	$0.8166e-05$	30.133087211221344	$0.9538e-4$
6	28	0.768618283581450	$0.2464e-06$	30.130175412991949	$0.1257e-5$
7	36	0.768618204690890	$0.1437e-06$	30.130286953988257	$0.2445e-5$
8	45	0.768618091001388	$0.4129e-08$	30.130214729062992	$0.4750e-7$
9	55	0.768618093017487	$0.1506e-08$	30.130214524534466	$0.4071e-7$
10	66	0.768618094207751	$0.4247e-10$	30.130213355947507	$0.1929e-8$
11	78	0.768618094183085	$0.1037e-10$	30.130213314885822	$0.5659e-9$
12	91	0.768618094174823	$0.3695e-12$	30.130213298960925	$0.3737e-10$

As expected, in both cases the error decreases very rapidly as N grows.

Finally, we have studied our algorithm for the box spline $B(\,\cdot\,|\begin{smallmatrix}1&1&0&0&1\\0&0&1&1&1\end{smallmatrix})$. Once again, we used $f(x,y) = \sin(x+y)$ as test function. The results are as follows.

N	n	$I^n_{B(\cdot\,|\begin{smallmatrix}1&1&0&0&1\\0&0&1&1&1\end{smallmatrix})}$	N	n	$I^n_{B(\cdot\,|\begin{smallmatrix}1&1&0&0&1\\0&0&1&1&1\end{smallmatrix})}$
1	3	−0.45348505681758178	7	36	0.10037633276022173
2	6	0.04964854550337172	8	45	0.10037668586593097
3	10	0.09289718468658661	9	55	0.10037657872727099
4	15	0.10098966569179155	10	66	0.10037657503752581
5	21	0.10042928697158683	11	78	0.10037657664472151
6	28	0.10037031312195928	12	91	0.10037657667564549

8 Practical applications

One possible application of the described method is the computation of right–hand sides in (adaptive) Galerkin schemes for elliptic operator equations. There, vectors of the form

$$\mathbf{F}_\Lambda := (f, \psi_{j,k})_{(j,k)\in\Lambda}$$

occur. Here Λ is a possibly lacunary set of wavelet indices. The application of the above method is straightforward. One reduces all integrals to the standard situation by means of the substitution (19) and uses the lifting trick (22) if necessary. However, the performance of the resulting quadrature formulas depends on the smoothness of the right–hand side f, see (16), which might not always be adequate.

Another application arises in the context of the boundary element method. In this case, one has to solve an operator equation

$$Au = f, \tag{43}$$

where A has a global Schwartz kernel K,

$$(Av)(x) = \int K(x,y)v(y)\, dy, \tag{44}$$

which satisfies the condition

$$|\partial_x^\alpha \partial_y^\beta K(x,y)| \leq c_{\alpha,\beta}\mathrm{dist}(x,y)^{-(d+2t+|\alpha|+|\beta|)}. \tag{45}$$

Here $2t$ is the order of the operator A . The function K is in general a smooth and non–local, but it may have a singularity on the diagonal. For further details, the reader is referred, e.g., to [DPS, Sch] and the references therein.

Let us first consider the case $d = 1$. If one wants to solve (43) by means of a wavelet Galerkin approach, one has to evaluate scalar products of the form

$$(K\varphi_{j,k}, \varphi_{j',k'}) := \int\limits_{\mathrm{supp}(\varphi_{j,k})} \int\limits_{\mathrm{supp}(\varphi_{j',k'})} K(x,y)\varphi_{j,k}(x)\varphi_{j',k'}(y)\, dxdy \tag{46}$$

and

$$(K\psi_{j,k}, \psi_{j',k'}) := \int\limits_{\mathrm{supp}(\psi_{j,k})} \int\limits_{\mathrm{supp}(\psi_{j',k'})} K(x,y)\psi_{j,k}(x)\psi_{j',k'}(y)\, dxdy. \tag{47}$$

By the substitution (19), we get from (46)

$$(K\varphi_{j,k}, \varphi_{j',k'}) = 2^{-(j+j')/2} \int\limits_{\mathrm{supp}\varphi} \int\limits_{\mathrm{supp}\varphi} K\Big(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k')\Big)\varphi(\bar{x})\varphi(\bar{y})\, d\bar{x}d\bar{y}.$$

These integrals can be evaluated by using the ideas described in Section 6.1. Indeed, we may apply Theorem 6.1 to the case $I_1 = I_2 = \mathrm{supp}\varphi$, $g_1 = g_2 = \varphi$, $f(x,y) = K\Big(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k')\Big)$. The reader should observe that the Schwartz kernel K is usually very smooth off the diagonal so that the Gauss quadrature should perform very well for the computation of integrals involving basis functions whose supports are well seperated. These integrals are sometimes called *farfield integrals.*

As an example, we have computed the scalar products (46) for an important special case, i.e., for the *Hilbert transform.* In this case, the kernel is of the form

$$K(x,y) = \frac{1}{x-y}. \tag{48}$$

The scaling function φ was chosen to be the hat function defined in (28). The resulting stiffness matrix for $j = j' = 3$ is illustrated in Figure 1. As discussed above, the method proposed here only performs satisfactorily in regions where $\mathrm{supp}(\varphi_{j,k}) \cap \mathrm{supp}(\varphi_{j,k'}) = \emptyset$. Therefore the entries in the vicinity of the diagonal are not depicted in the following figures. If the corresponding domains of integration are overlapping one gets singular integrals, which require particular attention. Weakly singular kernels can be transformed to analytic ones by nonlinear changes of variables. Higher singularities arising from other operators must be transformed to those of weakly singular type by some regularization. Details of such transformations have been carried out in [S, SS].

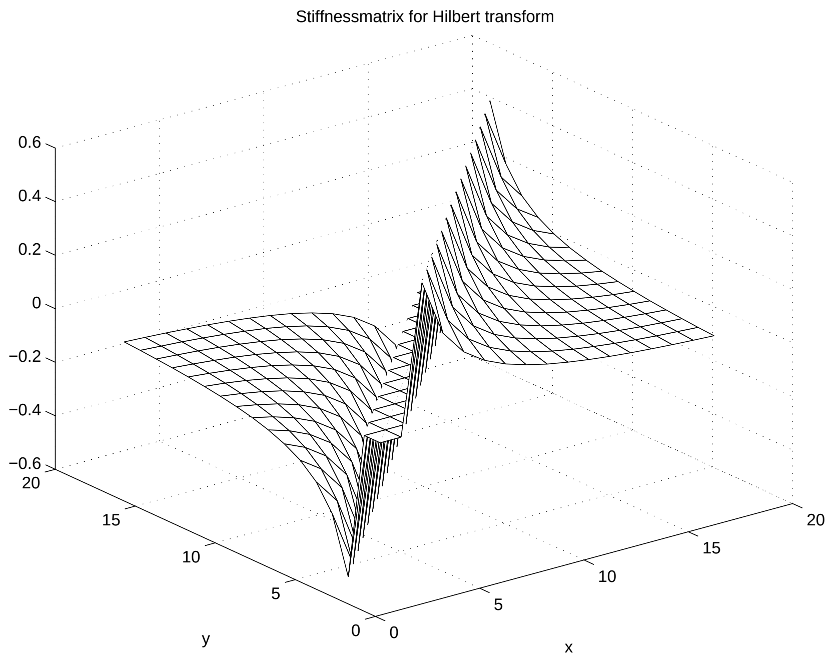


Figure 1: Stiffness matrix corresponding to the Hilbert transform

For the simple case of the hat function, the entries in the stiffness matrix can also be computed directly. This fact can clearly be used to test the performance of the algorithm. In Figure 2, we have depicted the resulting relative error. It turns out that the error is in fact very small, especially in the farfield. Note that for the above reasons, the entries in the vicinity of the diagonal are not displayed.

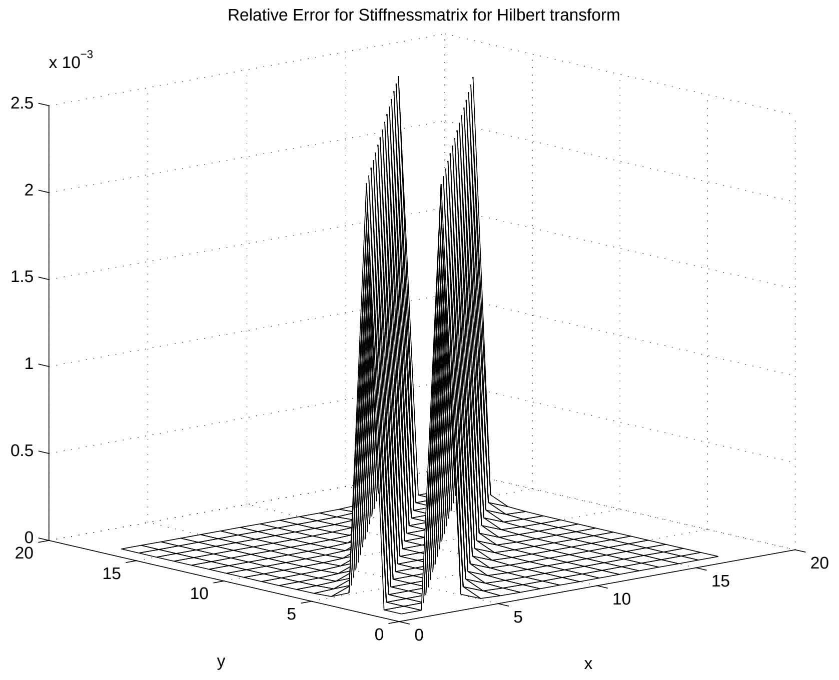


Figure 2: Relative error corresponding to the Hilbert transform

In case that φ is not strictly positive, we may apply the lifting trick (22) using $\varphi_1 = \varphi_2 = \varphi$. We obtain the decomposition

$$\begin{aligned}
&\int\limits_{\mathrm{supp}\varphi}\int\limits_{\mathrm{supp}\varphi} K(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k'))\varphi(\bar{x})\varphi(\bar{y})\, d\bar{x}d\bar{y} = \qquad (49)\\
&\quad \int\limits_{\mathrm{supp}\varphi}\int\limits_{\mathrm{supp}\varphi} K(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k'))(\varphi(\bar{x})+c)(\varphi(\bar{y})+c)\, d\bar{x}d\bar{y}\\
&\quad - \int\limits_{\mathrm{supp}\varphi}\int\limits_{\mathrm{supp}\varphi} K(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k'))(\varphi(\bar{x})+c)c\, d\bar{x}d\bar{y}\\
&\quad - \int\limits_{\mathrm{supp}\varphi}\int\limits_{\mathrm{supp}\varphi} K(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k'))(\varphi(\bar{y})+c)c\, d\bar{x}d\bar{y}\\
&\quad + \int\limits_{\mathrm{supp}\varphi}\int\limits_{\mathrm{supp}\varphi} K(2^{-j}(\bar{x}-k), 2^{-j'}(\bar{y}-k'))c^2\, d\bar{x}d\bar{y}.
\end{aligned}$$

We have tested this idea for the case $\varphi = \tilde{N}_{2,4}$ and the kernel defined in (48). The results are displayed in the following Figure 3, again omitting the entries close to the diagonal.

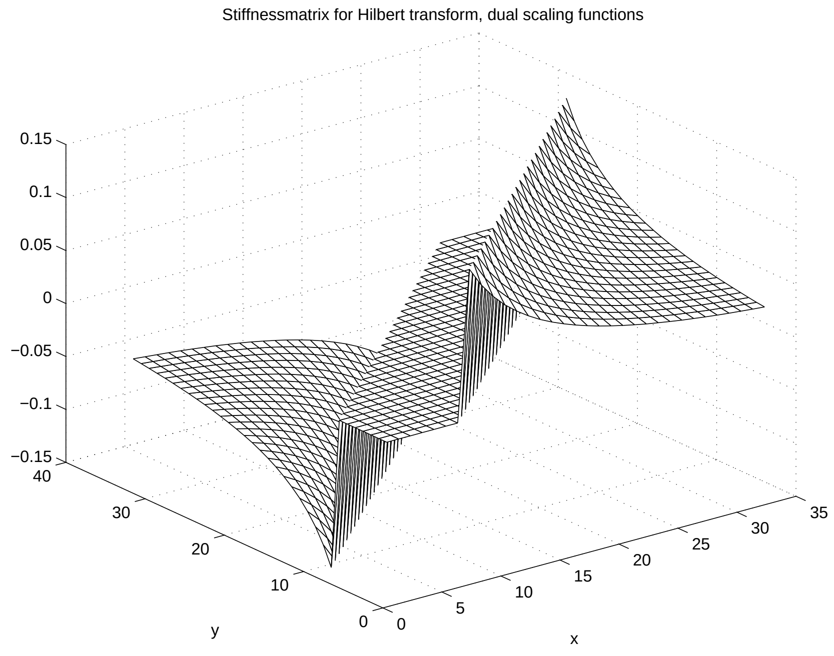


Figure 3: Stiffness matrix with respect to $\tilde{N}_{2,4}$

For the integrals according to (47), we can again use the lifting trick for the wavelet ψ, which is totally analogous to (49). The resulting stiffness matrix is presented in Figure 8 in the appendix. It shows the typical finger structure, that has been investigated, e.g., in [DPS].

Higher dimensional problems can be treated analogously by using suitable tensor products of wavelets and scaling functions or by setting up an appropriate quadrature formula by means of the method described in Section 6.2.

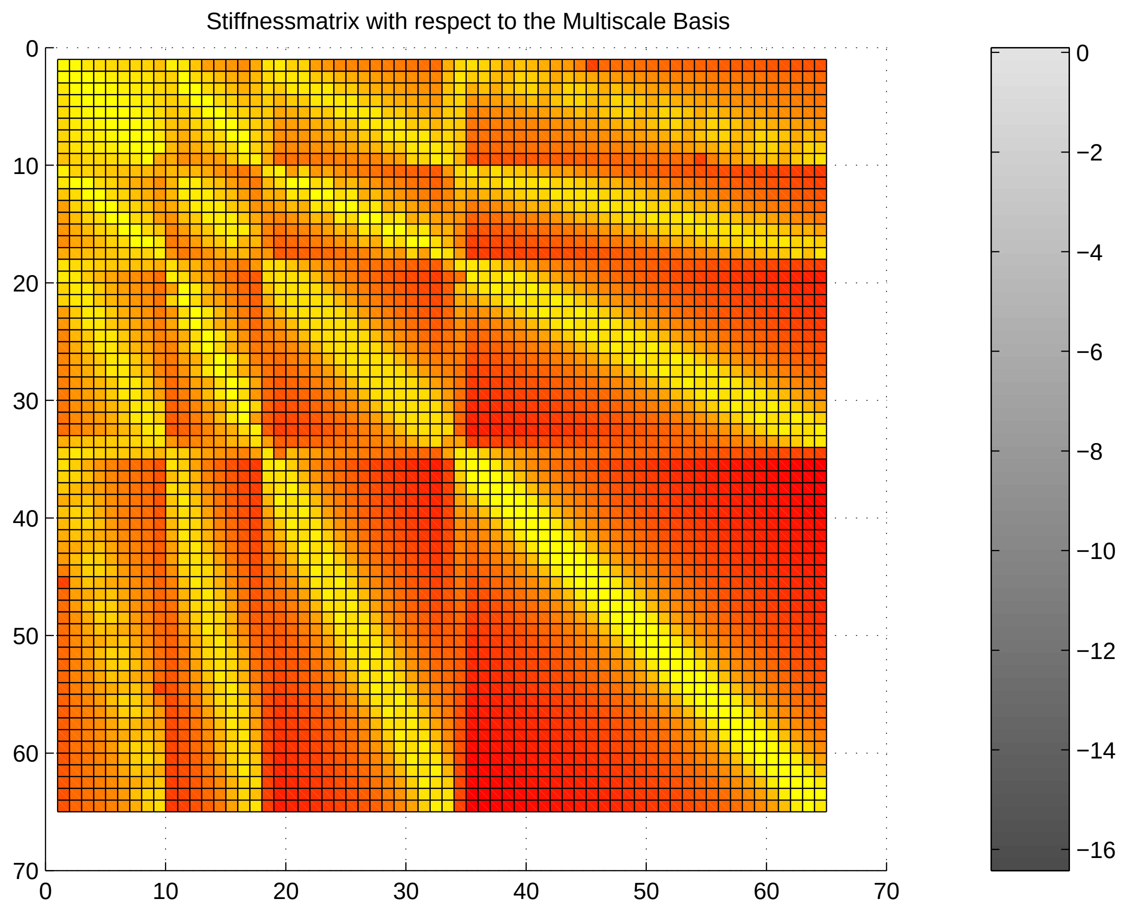


Figure 4: Stiffness matrix associated with the wavelet $\psi_{2,2}$

Acknowledgement: The authors would like to thank Prof. W. Gautschi for an inspiring discussion at the *2. Workshop on Orthogonal Polynomials* at Ballenstedt, Germany, and for putting unpublished material at their disposal. They also feel grateful to Mario Mommer who did a great job working on some of the numerical experiments.

A Appendix

In the following tables we list knots and weights for Gauss quadrature rules with $n = 1, \ldots 10$ knots for the hat function, cf. (28). Note that $w_i^{N_2} = w_{n-i+1}^{N_2}$ and $x_i^{N_2} = -x_{n-i+1}^{N_2}, \quad i = 1 \ldots \lceil n/2 \rceil$.

Table A, Weights and knots for the hat function, see (28)

n	$w_i^{N_2}$	$x_i^{N_2}$
1	1.000000000000000000000000000000	0.000000000000000000000000000000
2	0.500000000000000000000000000000	0.408248290463863016366214012451
3	0.208333333333333333333333333334	0.632455532033675866399778708887
	0.583333333333333333333333333332	0.000000000000000000000000000000
4	0.098866304579875626785852745947	0.750925143304447384549815336033
	0.401133695420124373214147254053	0.262229842652747963829202721001
5	0.051658257765490621790913992431	0.821440599738381527872000286332
	0.239473240705457390441501909707	0.449920352459841963349447292207
	0.417737003058103975535168195725	0.000000000000000000000000000000
6	0.029496561655309346531441703474	0.865738294978819096173689107167
	0.147481233705272848259384512169	0.576614808248487335302143006343
	0.323022204639417805209173784355	0.194263676775038891612011154922
7	0.017940745025631765708397345815	0.895615660710817516798208431832
	0.093925039319343914901843901982	0.665993283705955567193128023493
	0.224746555018036629912163487255	0.348413302656911965593248650426
	0.326775321273975378955190529895	0.000000000000000000000000000000
8	0.011529448606035851690614526463	0.916498601781823410628599506008
	0.062272586069543357120647564671	0.730126110448886200699403394721
	0.157584233713380391549094329339	0.464700844889638967828890258309
	0.268613731611040399639643579526	0.154629080840261133515525835015
9	0.007724672257048883537841173746	0.931753814324370259839777441275
	0.042637377229386622561030714360	0.777874522766333218736187749949
	0.111965186394200942366897985607	0.554352720106100763666240572469
	0.203250033165550156295536408147	0.284230579976574031942210440432
	0.268845461907626790477387436280	0.000000000000000000000000000000

(continued on next page)

Weights and knots for the hat function, see (28) (continued)		
10	0.005369059724471598670424583844	0.943166381725667699755384276105
	0.030113256293466570387688594983	0.814053288655822717949783211207
	0.081230624454969189238806002360	0.623784313633467079000174826997
	0.153837064198188702301836370938	0.388351321334551331283338706126
	0.229449995328903939401244447876	0.128573681331639522292510283016

In the next table, we list the knots and weight corresponding to the Courant finite element $B(\,\cdot\,|\,\begin{smallmatrix}1&0&1\\0&1&1\end{smallmatrix})$, cf. Section 7.2.

Table B: Weights and knots for the Courant FE, see Section (7.2)		
	$w_i^{B(\cdot\,|\,\begin{smallmatrix}1&0&1\\0&1&1\end{smallmatrix})}$	$x_i^{B(\cdot\,|\,\begin{smallmatrix}1&0&1\\0&1&1\end{smallmatrix})}$
N=1 (n=3)	0.49999999999999989	(1.0000000000000000, 2.0000000000000000)
	0.25000000000000000	(0.0000000000000000, 0.0000000000000000)
	0.25000000000000000	(2.0000000000000000, 0.0000000000000000)
N=2 (n=6)	0.21354166666666685	(0.7500000000000000, 0.3333333333333333)
	-0.02604166666666664	(1.7500000000000000, 0.3333333333333333)
	0.13541666666666665555	(0.2500000000000000, 1.0000000000000000)
	0.48958333333333333591	(1.2500000000000000, 1.0000000000000000)
	0.08854166666666667185	(0.7500000000000000, 1.6666666666666667)
	0.09895833333333333148	(1.7500000000000000, 1.6666666666666667)
N=3 (n=10)	0.02343749999999971	(0.0000000000000000, 0.2500000000000000)
	0.09895833333333410	(1.0000000000000000, 0.2500000000000000)
	-0.01822916666666661	(2.0000000000000000, 0.2500000000000000)
	0.23958333333333245	(0.5.00000000000000, 0.7500000000000000)
	0.15624999999999892	(1.5000000000000000, 0.7500000000000000)
	-0.00260416666666644	(0.0000000000000000, 1.2500000000000000)
	0.35937500000000105	(1.0000000000000000, 1.2500000000000000)
	0.03906250000000029	(2.0000000000000000, 1.2500000000000000)
	0.01041666666666670	(0.5.00000000000000, 1.7500000000000000)
	0.09374999999999965	(1.5000000000000000, 1.7500000000000000)
N=4 (n=15)	0.05465494791666419	(0.5000000000000000, 0.2000000000000000)
	0.01152343749999857	(1.1666666666666667, 0.2000000000000000)
	-0.00107421875000024	(1.8333333333333333, 0.2000000000000000)

(continued on next page)

Weights and knots for the Courant FE, see Section (7.2)		
	0.04778645833334450	(0.1666666666666666, 0.6000000000000000)
	0.17630208333334255	(0.8333333333333333, 0.6000000000000000)
	0.03.63281250000011	(1.5000000000000000, 0.6000000000000000)
	0.11243489583330270	(0.5000000000000000, 1.0000000000000000)
	0.21367187499999982	(1.1666666666666667, 1.0000000000000000)
	0.02285156250000006	(1.8333333333333333, 1.0000000000000000)
	-0.00039062499999955	(0.1666666666666666, 1.4000000000000000)
	0.11640625000002323	(0.8333333333333333, 1.4000000000000000)
	0.14440104166666040	(1.5000000000000000, 1.4000000000000000)
	0.00061848958333334	(0.5000000000000000, 1.8000000000000000)
	0.04147135416666046	(1.1666666666666667, 1.8000000000000000)
	0.02301432291666924	(1.8333333333333333, 1.8000000000000000)
N=5	0.00563964843751220	(0.0000000000000000, 0.1666666666666666)
(n=25)	0.04064941406245611	(0.6666666666666666, 0.1666666666666666)
	-0.00017089843749108	(1.3333333333333333, 0.1666666666666666)
	-0.00002441406249938	(2.0000000000000000, 0.1666666666666666)
	0.07250976562499598	(0.3333333333333333, 0.5000000000000000)
	0.09404296875009843	(1.0000000000000000, 0.5000000000000000)
	0.00766601562498629	(1.6666666666666667, 0.5000000000000000)
	0.00424804687499713	(0.0000000000000000, 0.8333333333333333)
	0.16567382812499065	(0.6666666666666666, 0.8333333333333333)
	0.11137695312488399	(1.3333333333333333, 0.8333333333333333)
	-0.00161132812499675	(2.0000000000000000, 0.8333333333333333)
	0.03525390624998361	(0.3333333333333333, 1.1666666666666667)
	0.17324218750004255	(1.0000000000000000, 1.1666666666666667)
	0.07119140625007708	(1.6666666666666667, 1.1666666666666667)
	-0.00051269531249515	(0.000000000000000, 1.5000000000000000)
	0.03430175781251509	(0.666666666666666, 1.5000000000000000)
	0.12941894531245335	(1.3333333333333333, 1.5000000000000000)
	0.01101074218748175	(2.0000000000000000, 1.5000000000000000)
	-0.00151367187500374	(0.3333333333333333, 1.8333333333333333)
	0.02021484374999951	(1.0000000000000000, 1.8333333333333333)
	0.02739257812501232	(1.6666666666666667, 1.8333333333333333)

(continued on next page)

Weights and knots for the Courant FE, see Section (7.2)		
N=6	0.02165244369923851	(0.3750000000000000, 0.1428571428571428)
	0.01092212414607925	(0.8750000000000000, 0.1428571428571428)
	0.00023707233405760	(1.3750000000000000, 0.1428571428571428)
	0.00024138354750510	(1.8750000000000000, 0.1428571428571428)
	0.01956523108091459	(0.1.25000000000000, 0.4285714285714285)
	0.06333050601655035	(0.6.25000000000000, 0.4285714285714285)
	0.05252965695052495	(1.1250000000000000, 0.4285714285714285)
	-0.00197270307585234	(1.6250000000000000, 0.4285714285714285)
	0.05579025430068487	(0.3.75000000000000, 0.7142857142857143)
	0.09904191355256522	(0.8.75000000000000, 0.7142857142857143)
	0.03337605390533229	(1.3750000000000000, 0.7142857142857143)
	0.00133713414426512	(1.8750000000000000, 0.7142857142857143)
	0.00721324133490140	(0.1.25000000000000, 1.0000000000000000)
	0.09532598263317101	(0.6.25000000000000, 1.0000000000000000)
	0.13028669407781066	(1.1250000000000000, 1.0000000000000000)
	0.05507194075043687	(1.625000000000000, 1.0000000000000000)
	0.01838655421356106	(0.3.75000000000000, 1.2857142857142858)
	0.07841338707667969	(0.8.75000000000000, 1.2857142857142858)
	0.07853257517310503	(1.3750000000000000, 1.2857142857142858)
	0.01421283943936253	(1.8750000000000000, 1.2857142857142858)
	-0.00050312829397308	(0.1.25000000000000, 1.5714285714285714)
	0.00710763275300024	(0.6.25000000000000, 1.5714285714285714)
	0.07266312049191456	(1.1250000000000000, 1.5714285714285714)
	0.05418506602132048	(1.6250000000000000, 1.5714285714285714)
	-0.00053315742935788	(0.3.75000000000000, 1.8571428571428572)
	0.00716100218435023	(0.8.75000000000000, 1.8571428571428572)
	0.01765379325419301	(1.3750000000000000, 1.8571428571428572)
	0.00877138571765864	(1.8750000000000000, 1.8571428571428572)

References

[Ch] C.K. Chui, *An Introduction to Wavelets*, Academic Press, Boston, 1992.

[CDF] A. Cohen, I. Daubechies, and J. Feauveau, Bi–orthogonal bases of compactly supported wavelets, *Comm. Pure Appl. Math.* **45** (1992), 485–560.

[CD] A. Cohen and I. Daubechies, Non–separable bidimensional wavelet bases, *Rev. Mat. Iberoamericana* **9** (1993), 51–137.

[D] W. Dahmen, Wavelet and multiscale methods for operator equations, *Acta Numerica* **6** (1997), Cambridge University Press, Cambridge, 55–228.

[DM] W. Dahmen and C.A. Micchelli, Recent progresses in multivariate splines, in: "Approximation Theory IV", (C.K. Chui, L.L. Schumaker, and J.D. Ward, Eds.), Academic Press, New York, 1983, 27–121.

[DM2] W. Dahmen and C.A. Micchelli, Using the refinement equation for evaluating integrals of wavelets, *SIAM J. Numer. Anal.* **30(2)** (1993), 507–537.

[DPS] W. Dahmen, S. Prössdorf, and R. Schneider, Multiscale methods for pseudo-differential equations on manifolds, in: "Wavelets: Theory, Algorithms, and Applications", (C.K. Chui, L. Montefusco, and L. Puccio, Eds.), Academic Press, 1994, 385–424.

[Dau] I. Daubechies, *Ten Lectures on Wavelets*, CBMS–NSF Regional Conference Series in Applied Math. **61**, SIAM, Philadelphia, 1992.

[DR] P. Davis and P. Rabinowitz, *Methods of Numerical Integration*, (W. Rheinboldt, Ed.), Academic Press, New York, 1975.

[G] W. Gautschi, Orthogonal polynomials: applications and computation, *Acta Numerica* **5** (1996), Cambridge University Press, Cambridge, 45–119.

[GGP] W. Gautschi, L. Gori, and F. Pitolli, *Gauss quadrature for refinable weight functions*, preprint, 1999.

[GS] W. Gautschi and T. Sauer, *Moments of refinable functions and Gauss quadrature*, manuscript, 1999.

[JM] R.Q. Jia and C.A. Micchelli, Using the refinement equations for the construction of pre–wavelets II: Powers of two, in: "Curves and Surfaces", (P.J. Laurent, A. LeMéhauté, and L.L. Schumaker, Eds.), Academic Press, New York, 1991, 209–246.

[KL] J.–P. Kahane and P.–G. Lemarié–Rieusset, *Fourier Series and Wavelets*, Gordon and Breach Science Publishers, Luxembourg, 1995.

[K] V.I. Krylov, *Approximate Calculation of Integrals*, translated by A.H. Stroud, Macmillan, New York, 1962.

[M] S. Mallat, Multiresolution approximation and wavelet orthonormal bases of $L^2(I\!R)$, *Trans. Amer. Math. Soc.* **315** (1989), 69–88.

[Me] Y. Meyer, *Wavelets and Operators*, Cambridge Studies in Advanced Mathematics **37**, Cambridge, 1992.

[PS1] R. Piessens and W. Sweldens, Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions, *SIAM J. Numer. Anal.* **31(4)** (1994), 1240–1264.

[PS2] R. Piessens and W. Sweldens, Asymptotic error expansions of wavelet approximations of smooth functions II, *Numer. Math.* **68** (1994), 377–401.

[S] S. Sauter, *Über die effiziente Verwendung des Galerkin–Verfahrens zur Lösung Fredholmscher Integralgleichungen*, PhD. Thesis, Christian-Albrecht-Universität, Kiel, 1992.

[SS] S. Sauter and C. Schwab, Quadrature for hp Galerkin BEM in $I\!R^3$, *Num. Math.* **78** (1997), 211–258.

[Sch] R. Schneider, *Multiskalen– und Wavelet–Matrixkompression: Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme*, Advances in Numerical Mathematics, Teubner, Stuttgart, 1998.

[St] A.H. Stroud, *Approximate Calculation of Multiple Integrals,* Prentice–Hall, New Jersey, 1971.

[StS] A.H. Stroud and D. Secrest, *Gaussian Quadrature Formulas,* Prentice–Hall, New Jersey, 1966.

[W] P. Wojtaszczyk, *A Mathematical Introduction to Wavelets*, Cambridge University Press, Cambridge, 1997.

Arne Barinka, Titus Barsch, Stephan Dahlke
RWTH Aachen
Institut für Geometrie und Praktische Mathematik
Templergraben 55
52056 Aachen
Germany
{barinka,barsch,dahlke}@igpm.rwth-aachen.de

Michael Konik
Fakultät für Mathematik
Technische Universität Chemnitz
09107 Chemnitz
Germany
konik@mathematik.tu-chemnitz.de