FAST ALGORITHMS FOR PERIODIC SPLINE WAVELETS ON SPARSE GRIDS

KAI BITTNER*

Abstract. We consider Boolean sums of univariate interpolation operators which define multivariate j-th order blending interpolation operators on sparse grids. Sample spaces are defined as range of the blending operators. Sample and wavelet spaces have significantly lower dimension and good approximation order for certain function spaces. Fast decomposition and reconstruction algorithms for bivariate spline wavelets, based on algorithms for univariate functions, are described. Operation counts for the algorithms are given and it is shown, that the complexity depends linearly on the dimension of sample spaces.

Key words. wavelets, splines, multivariate periodic interpolation, Boolean sums, sparse grids

AMS subject classifications. 41A15, 41A63, 65D05, 65D07, 65Y20

1. Introduction. Wavelet analysis has evolved in the last few years as a powerful method in numerical analysis and signal processing. An important question is the development of efficient algorithms for multivariate wavelet transforms. The classical approach for the construction of bivariate wavelets from given univariate functions by tensor products leads to a fast increasing dimension of space of order $\mathcal{O}(2^{2j})$. Therefore, one has to take into account a high complexity of the corresponding algorithms.

To avoid this problem the concept of Boolean sums and sparse grids is often used in multivariate approximation and interpolation (see e.g. Delvos and Schempp [5]) as well as in finite element methods (see e.g. Zenger [16]). Applying the Boolean sum method sample spaces of dimension $\mathcal{O}(j2^j)$ are described in [1] and [5]. It is shown there that for functions from bivariate Korobov spaces (see [7]) one obtains only an extra logarithmic factor in the approximation order, while the dimension of the sample spaces is significantly lower. A unified approach to error estimates for Boolean sums and tensor products in Korobov and Sobolev spaces can be found in [10].

The Boolean sum method is applied to wavelet theory by Sprengel in [14], where corresponding sample and wavelet spaces of bivariate periodic functions are considered. A basis for sample spaces is given by fundamental interpolants on sparse grids. It is crucial for this approach, that the underlying univariate wavelets and scaling functions are fundamental interpolants. In [15], a first algorithm for complete decomposition and reconstruction with complexity $\mathcal{O}(j^2 2^j)$ was described by using matrix-vector notations and the fast Fourier transform (FFT).

The aim of this paper is to describe faster algorithms by using periodic spline wavelets with minimal support. Therefore, we establish in §2 an approach, where the underlying univariate wavelet and scaling functions need not be fundamental interpolants. We demand only, that for the sample spaces a certain interpolation problem has a unique solution. Basis functions for the bivariate sample spaces are still the fundamental interpolants from [14], for which we establish the two-scale relation in Theorem 2.1. However, for our applications, we mainly use a generating system which consists of the translates of certain tensor products of univariate scaling functions. This approach leads to algorithms, which are different from the algorithms of Sprengel [15] and have some particular advantages.

*Fachbereich Mathematik, Universität Rostock, Universitätsplatz 1, D-18051 Rostock, Germany, (kai.bittner@mathematik.uni-rostock.de)

For our algorithms, we use periodic spline wavelets with minimal support as described by Plonka and Tasche [8]. In §3, we give a short introduction to these wavelets. In §4, we study fast algorithms for univariate periodic spline wavelets, which are based on a method described in [2]. The idea of recursive filters used there yields complexity $\mathcal{O}(2^j)$ in the univariate setting.

In §5, we present new algorithms for wavelets on sparse grids, which do not require the complete decomposition of a function until the lowest level as in [15] and therefore improve the algorithms presented there. The main idea is to use the representation of functions from the sample spaces by the generating system mentioned above. In this way, we can apply ideas from the tensor product setting, such that our algorithms make use of the algorithms for the univariate case from §4. Beyond the decomposition and reconstruction algorithm, we give also an interpolation algorithm and an algorithm for the computation of function values on the sparse grid. For all algorithms, we count the floating point operations and compare it with the corresponding numbers of the tensor product algorithms.

A numerical example is presented in §6. For this we compare the results of Boolean sum and tensor product algorithms. We describe some special effects of the Boolean sum method and show that for suited functions the Boolean sum method is more efficient than tensor product methods.

2. Sample and Wavelet spaces. Bivariate interpolating scaling functions and wavelets for sparse grids have been investigated for the first time by Sprengel in [14, 15] using Boolean sums of univariate interpolation operators (see [1, 5]). In the following we shortly recall this definitions and establish a more general approach for wavelets on sparse grids.

By $C(\mathbb{T}^n)$ we denote the space of all continuous, n-variate 2π-periodic functions and by $L^2(\mathbb{T}^n)$ the space of square-integrable, n-variate 2π-periodic functions with the inner product $\langle f,g\rangle_{\mathbb{T}^n} := (2\pi)^{-n}\int f(x)\overline{g(x)}\,dx$.

For a fixed $d \in \mathbb{N}$, we put $d_j := d\,2^j$, $j \in \mathbb{N}_0$. For each $j \in \mathbb{N}_0$, we introduce univariate scaling functions $\phi_{j,k}$ as translates of a single function, i.e.,

$$\phi_{j,k} = \phi_j\left(\cdot - \tfrac{2\pi k}{d_j}\right), \quad k = 0,\ldots,d_j-1.$$

In the following, we demand that the scaling functions $\phi_{j,k}$ are linear indepentent. Then we define corresponding univariate sample spaces by

$$V_j := \operatorname{span}\{\phi_{j,k} :\ k = 0,\ldots,d_j-1\}\,. \tag{2.1}$$

Further, we demand the existence of a linear interpolation operator L_j with the precision set

$$\operatorname{prec} L_j = \mathcal{T}_j := \left\{t_{j,k} := \frac{2\pi k}{d_j} : k = 0,\ldots,d_j-1\right\} \subset [0,2\pi)$$

and the image $\operatorname{Im} L_j = V_j$, i.e., for every function $f \in C(\mathbb{T})$, there have to exist coefficients $a_{j,k} \in \mathbb{R}$ such that

$$L_j f(t_{j,l}) = \sum_{k=0}^{d_j-1} a_{j,k}\phi_{j,k}(t_{j,l}) = f(t_{j,l}), \qquad l = 0,\ldots,d_j-1. \tag{2.2}$$

Obviously, the interpolation projectors L_j satisfy the embedding relations $\operatorname{Im} L_j \subset \operatorname{Im} L_{j+1}$ and $\operatorname{prec} L_j \subset \operatorname{prec} L_{j+1}$.

For the bivariate setting, we introduce the projector of *j-th order blending interpolation* defined by

$$B_j := \bigoplus_{r=0}^{j} L_r \otimes L_{j-r} = \sum_{r=0}^{j} L_r \otimes L_{j-r} - \sum_{r=0}^{j-1} L_r \otimes L_{j-r-1} \,,$$

where $A \oplus B := A + B - AB$ denotes the Boolean sum and $A \otimes B$ the tensor product of projectors (cf. [5]). Then, the interpolation knots form a sparse grid (Figure 2.1) given by

$$\mathcal{T}_j^B := \operatorname{prec} B_j = \bigcup_{r=0}^{j} \operatorname{prec} L_r \times \operatorname{prec} L_{j-r} = \bigcup_{r=0}^{j} \mathcal{T}_r \times \mathcal{T}_{j-r} \,.$$

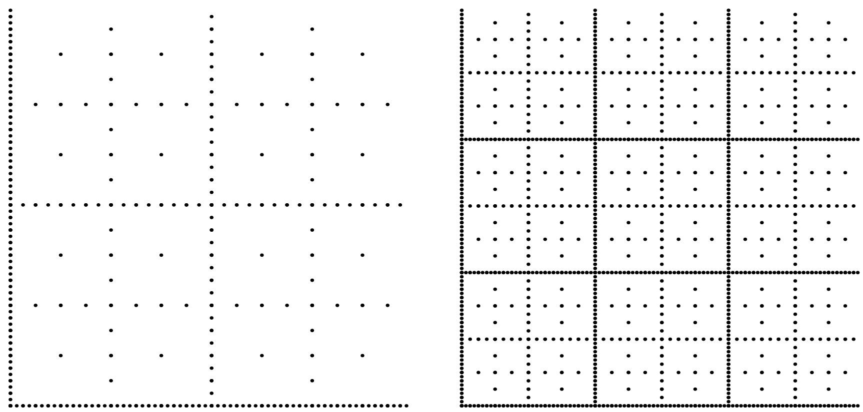

FIG. 2.1. *Sparse grids $\mathcal{T}_6^B$ for $d = 1$ and $\mathcal{T}_5^B$ for $d = 3$.*

Now we define the bivariate sample spaces as the images of B_j, i.e.,

$$V_j^B := \operatorname{Im} B_j = \sum_{r=0}^{j} \operatorname{Im} L_r \otimes \operatorname{Im} L_{j-r} = \sum_{r=0}^{j} V_r \otimes V_{j-r} \,.$$

A generating system of V_j^B is given by

$$\bigcup_{r=0}^{j} \{\phi_{r,k} \otimes \phi_{j-r,l} : \, k = 0, \ldots, d_r - 1; \, l = 0, \ldots, d_{j-r} - 1\} \,. \tag{2.3}$$

This system consists of the translates of $j+1$ different tensor product functions and therefore it is very useful in applications. But for the unique representation of a function $f \in V_j^B$ a basis is necessary. To construct such a basis of V_j^B we employ the interpolation properties of B_j. For this we consider the univariate fundamental interpolants $\phi_{j,k}^\delta \in V_j$, which are uniquely determined (cf. (2.2)) by

$$\phi_{j,k}^\delta(t_{j,l}) = \delta_{k,l}, \quad k, l = 0, \ldots, d_j - 1$$

and which form a basis of V_j. Analogously, a basis of V_j^B is given by the bivariate fundamental interpolants for the sparse grid $\mathcal{T}_j^B$

$$\Phi_{j,(k,l)} = B_j(\phi_{j,k}^\delta \otimes \phi_{j,l}^\delta) = \sum_{r=0}^{j} \phi_{r,2^{r-j}k}^\delta \otimes \phi_{j-r,2^{-r}l}^\delta - \sum_{r=0}^{j-1} \phi_{r,2^{r-j}k}^\delta \otimes \phi_{j-r-1,2^{-r-1}l}^\delta \,,$$

where $(k,l) \in \mathcal{J}_j := \{(k,l) \in \mathbb{Z}^2 :\ (t_{j,k}, t_{j,l}) \in \mathcal{T}_j^B\}$ and $\phi_{j,s} \equiv 0$ for $s \notin \mathbb{Z}$. These functions are not translates of each other and have a more difficult structure. But it turns out, that for our algorithms, we need not use the functions $\phi_{j,k}^\delta$ and $\Phi_{j,(k,l)}$ explicitly. However, it should be useful to investigate the functions $\Phi_{j,(k,l)}$ in more detail. Therefore, we will establish and prove a refinement relation for these functions.

For this, we consider the following finer splitting of $\mathcal{J}_j$

$$\mathcal{J}_j = \bigcup_{\kappa=0}^{j} 2^{j-\kappa}(\mathcal{J}_\kappa \setminus 2\mathcal{J}_{\kappa-1}) = \bigcup_{\kappa=0}^{j} 2^{j-\kappa} \bigcup_{r=0}^{\kappa} \mathcal{K}_{\kappa,r}$$

into the disjoint sets $\mathcal{K}_{\kappa,r}$ with $\mathcal{K}_{0,0} = \mathcal{J}_0$ and for $\kappa \geq 1$

$$\begin{aligned}
\mathcal{K}_{\kappa,r} &= \{(2^{\kappa-r}(2\nu+1), 2^r(2\mu+1)) :\ \nu = 0,\ldots,d_{r-1}-1;\ \mu = 0,\ldots,d_{\kappa-r-1}-1\}, \\
&\quad r = 1,\ldots,\kappa-1\,, \\
\mathcal{K}_{\kappa,0} &= \{(2^\kappa \nu, 2\mu+1) :\ \nu = 0,\ldots,d_0-1;\ \mu = 0,\ldots,d_{\kappa-1}-1\}\,, \\
\mathcal{K}_{\kappa,\kappa} &= \{(2\nu+1, 2^\kappa \mu) :\ \nu = 0,\ldots,d_{\kappa-1}-1;\ \mu = 0,\ldots,d_0-1\}\,.
\end{aligned}$$

For $(k,l) \in 2^{j-\kappa}\mathcal{K}_{\kappa,r}$, we obtain

$$\Phi_{j,(k,l)} = \sum_{s=r}^{j+r-\kappa} \phi^\delta_{s,2^{s-j}k} \otimes \phi^\delta_{j-s,2^{-s}l} - \sum_{s=r}^{j+r-\kappa-1} \phi^\delta_{s,2^{s-j}k} \otimes \phi^\delta_{j-s-1,2^{-s-1}l} \tag{2.4}$$

and in particular for $(k,l) \in \mathcal{J}_j \backslash 2\mathcal{J}_{j-1}$, this simplifies to

$$\Phi_{j,(k,l)} = \phi^\delta_{r,2^{r-j}k} \otimes \phi^\delta_{j-r,2^{-r}l}\,. \tag{2.5}$$

Now we establish the refinement equation of the bivariate fundamental interpolants.

THEOREM 2.1. *Let* $(k,l) \in 2^{j-\kappa}\mathcal{K}_{\kappa,r}$, $0 \leq r \leq \kappa \leq j$. *With* $\alpha_{j,k} := \phi^\delta_{j,0}(t_{j+1,2k+1})$ *we have*

$$\begin{aligned}
\Phi_{j,(k,l)} = \Phi_{j+1,(2k,2l)} &+ \sum_{m=0}^{d_{j-r}-1} \alpha_{j-r,m-2^{-r}l} \Phi_{j+1,(2k,2^r(2m+1))} \\
&+ \sum_{m=0}^{d_{j+r-\kappa}-1} \alpha_{j+r-\kappa,m-2^{r-\kappa}k} \Phi_{j+1,(2^{\kappa-r}(2m+1),2l)} \\
&- \sum_{s=r+1}^{j+r-\kappa} \sum_{m=0}^{d_{s-1}-1} \sum_{n=0}^{d_{j-s}-1} \alpha_{s-1,m-2^{s-j-1}k} \alpha_{j-s,n-2^{-s}l} \Phi_{j+1,(2^{j-s+1}(2m+1),2^s(2n+1))}\,.
\end{aligned} \tag{2.6}$$

Proof. From the interpolation properties of $\phi^\delta_{j,k}$, we conclude

$$\phi^\delta_{j,k} = \phi^\delta_{j+1,2k} + \sum_{l=0}^{d_j-1} \alpha_{j,l-k} \phi^\delta_{j+1,2l+1}, \quad k = 0,\ldots,d_j-1. \tag{2.7}$$

Let now j, k, l be fixed. We introduce the notations

$$\sigma_s^1 := \phi_{s,2^{s-j}k} \otimes \phi_{j-s+1,2^{-s+1}l}\ , \quad \sigma_s^2 := \phi_{s,2^{s-j}k} \otimes \phi_{j-s,2^{-s}l}\ ,$$
$$\sigma_s^3 := \sum_{m=0}^{d_{j-s}-1} \alpha_{j-s,m-2^{-s}l}\ \phi_{s,2^{s-j}k} \otimes \phi_{j-s+1,2m+1}\ ,$$
$$\sigma_s^4 := \sum_{m=0}^{d_{s-1}-1} \alpha_{s-1,m-2^{s-j-1}k}\ \phi_{s,2m+1} \otimes \phi_{j-s+1,2^{-s+1}l}\ ,$$
$$\sigma_s^5 := \sum_{m=0}^{d_{s-1}-1} \sum_{n=0}^{d_{j-s}-1} \alpha_{s-1,m-2^{s-j-1}k}\, \alpha_{j-s,n-2^{-s}l}\ \phi_{s,2m+1} \otimes \phi_{j-s+1,2n+1}\,.$$

For $(k,l) \in 2^{j-\kappa}\mathcal{K}_{\kappa,r}$, we have with (2.4)

$$\Phi_{j+1,(2k,2l)} = \sum_{s=r}^{j+r-\kappa+1} \sigma_s^1 - \sum_{s=r}^{j+r-\kappa} \sigma_s^2\,. \tag{2.8}$$

From (2.7), it follows that

$$\begin{aligned}
&\phi_{s,2^{s-j}k} \otimes \phi_{j-s,2^{-s}l} \\
&= \ \phi_{s,2^{s-j}k} \otimes \left(\phi_{j-s+1,2^{-s+1}l} + \sum_{m=0}^{d_{j-s}-1} \alpha_{j-s,m-2^{-s}l}\phi_{j-s+1,2m+1}\right) \\
&\quad + \left(\phi_{s+1,2^{s-j+1}k} + \sum_{m=0}^{d_s-1} \alpha_{s,m-2^{s-j}k}\phi_{s+1,2m+1}\right) \otimes \phi_{j-s,2^{-s}l} \\
&\quad - \phi_{s,2^{s-j}k} \otimes \phi_{j-s,2^{-s}l} \\
&= \sigma_s^1 + \sigma_s^3 + \sigma_{s+1}^1 + \sigma_{s+1}^4 - \sigma_s^2
\end{aligned}$$

and

$$\begin{aligned}
\phi_{s,2^{s-j}k} \otimes \phi_{j-s-1,2^{-s-1}l} = &\left(\phi_{s+1,2^{s-j+1}k} + \sum_{m=0}^{d_s-1} \alpha_{s,m-2^{s-j}k}\phi_{s+1,2m+1}\right) \\
&\otimes \left(\phi_{j-s,2^{-s}l} + \sum_{m=0}^{d_{j-s-1}-1} \alpha_{j-s-1,m-2^{-s-1}l}\phi_{j-s,2m+1}\right) \\
&= \sigma_{s+1}^1 + \sigma_{s+1}^3 + \sigma_{s+1}^4 + \sigma_{s+1}^5\,.
\end{aligned}$$

With (2.4), we obtain

$$\Phi_{j,(k,l)} = \sum_{s=r}^{j+r-\kappa} (\sigma_s^1 - \sigma_s^2) + \sigma_{j+r-\kappa+1}^1 \ + \ \sigma_r^3 \ + \ \sigma_{j+r-\kappa+1}^4 \ - \sum_{s=r+1}^{j+r-\kappa} \sigma_s^5.$$

Taking into account (2.8) and (2.5), the assertion (2.6) follows. $\square$

Now we introduce the univariate wavelet space W_j as the orthogonal complement of V_j in V_{j+1}, i.e. $V_{j+1} = V_j \oplus^\perp W_j$, where $\oplus^\perp$ denotes the orthogonal sum in $L^2(\mathbb{T}^n)$. We choose a wavelet function $\psi_j \in W_j$, such that

$$\{\psi_{j,k} = \psi_j(\cdot - t_{j,k})\ :\ k = 0, \ldots, d_j - 1\}$$

is a basis of W_j.

The bivariate wavelet spaces are given by

$$W_j^B = \bigoplus_{r=0}^{j+1}{}^{\perp}\, W_{r-1} \otimes W_{j-r}$$

with $W_{-1} := V_0$. In [14], it has been proved that $V_{j+1}^B = V_j^B \oplus^{\perp} W_j^B$. A basis of W_j^B is given in a natural way by

$$\bigcup_{r=0}^{j+1} \{\psi_{r-1,k} \otimes \psi_{j-r,l} :\ k = 0,\ldots,d_{r-1}-1;\ l = 0,\ldots,d_{j-r}-1\}, \tag{2.9}$$

where $d_{-1} := d_0$ and $\psi_{-1,k} := \phi_{0,k}$. By counting the basis functions we obtain

$$\dim W_j^B = d^2(j2^{j-1} + 2^{j+1}) \quad \text{and} \quad \dim V_j^B = d^2(j2^{j-1} + 2^j)\,. \tag{2.10}$$

In [14], Theorem 2, it is shown that the bivariate wavelet basis (2.9) is stable, if the univariate scaling functions and wavelets form stable bases.

3. Periodic spline wavelets. Now we will consider an example for suitable univariate sample and wavelet spaces. By Sprengel [14, 15], interpolating trigonometric polynomials as described in [11, 12] are used as univariate wavelet and scaling functions. In this case one has $\phi_{j,k} = \phi_{j,k}^{\delta}$ and the wavelets satisfy an interpolation property, too. This has some advantages in the notation. On the other hand, this approach leads to algorithms with complexity $\mathcal{O}(j2^j)$ such that the algorithms for the bivariate case have complexity $\mathcal{O}(j^2 2^j)$. To improve the algorithms we propose another approach. Because of the wide applicability and the simple structure of spline interpolation, we will restrict ourselves to spline wavelets in the sequel.

Here, we consider periodic spline wavelets with minimal support as described in [8, 9]. We briefly recall some definitions. The k–th Fourier coefficient of $f \in L^2(\mathbb{T})$ is given by $c_k(f) := \langle f, e^{ik^T\cdot}\rangle_{\mathbb{T}}$, $k \in \mathbb{Z}$. Further, we define the discrete Fourier transform of length N by $\hat{x}_n := \sum_{k=0}^{N-1} x_k w_N^{nk}$, $n \in \mathbb{Z}$, where $w_N := e^{-2\pi i/N}$ is a primitive N–th root of unity.

The cardinal B-spline of order $m \geq 1$ is defined by

$$N_m := N_{m-1} * N_1 = \int_0^1 N_{m-1}(\cdot - t)\, dt\,, \quad m \geq 2$$

with $N_1 := \frac{1}{2}(\chi_{[0,1)} + \chi_{(0,1]})$. By the convolution property, the Fourier transform of N_m reads as follows

$$\widehat{N}_m(u) := \int_{-\infty}^{\infty} N_m(t)\, e^{-iut}\, dt = \begin{cases} 1 & \text{for} \quad u = 0\,, \\ \left(\frac{1-e^{-iu}}{iu}\right)^m & \text{otherwise}\,. \end{cases} \tag{3.1}$$

The scaling function ϕ_j $(j \in \mathbb{N}_0)$ is defined as 2π-periodization of $N_m(2^j\cdot)$, i.e.

$$\phi_j := \sum_{k=-\infty}^{\infty} N_m\left(d_j\left(\frac{\cdot}{2\pi} + k\right)\right)\,. \tag{3.2}$$

By the Poisson summation formula, one obtains the Fourier coefficients

$$c_n(\phi_j) = \frac{1}{d_j}\widehat{N}_m\left(\frac{2\pi n}{d_j}\right) \quad \text{and} \quad c_n(\phi_{j,l}) = w_{d_j}^{nl}\, c_n(\phi_j). \tag{3.3}$$

The *two–scale relation* is given by $\phi_j = \sum_{k=0}^{m} p_k \phi_{j+1,k}$ or in the Fourier transformed form $c_n(\phi_j) = 2P(w_{d_{j+1}}^n)c_n(\phi_{j+1})$ with the *two–scale symbol*

$$P(z) := \tfrac{1}{2}\sum_{k=0}^{m} p_k z^k = \left(\frac{1+z}{2}\right)^m .$$

Using the results of Locher [6] for construction of periodic fundamental interpolants in shift invariant spaces, we obtain the Fourier coefficients of the fundamental interpolant $\phi_{j,0}^{\delta}$ of even spline order $m = 2r$ by

$$c_n(\phi_j^{\delta}) = \frac{c_n(\varphi_j)}{\sum_{k=0}^{d_j-1}\phi_j(t_{j,k})w_{d_j}^{kn}} = \frac{w_{d_j}^{-nr}}{\Phi_{2r}(w_{d_j}^n)}c_n(\varphi_j) = \frac{1}{d_j\Phi_{2r}(w_{d_j}^n)}\mathrm{sinc}^{2r}\left(\frac{\pi n}{d_j}\right) \tag{3.4}$$

with the m–th *Euler–Frobenius function* $\Phi_m(z) := \sum_{k=-\infty}^{\infty} N_m\left(\frac{m}{2}+k\right)z^k$. In Figure 3.1, we show bivariate basis functions constructed from these splines. For odd spline order, there does not exist a unique spline interpolant on the grid $\mathcal{T}_j$. Consequently, we will deal in the following only with splines of even order $m = 2r$.

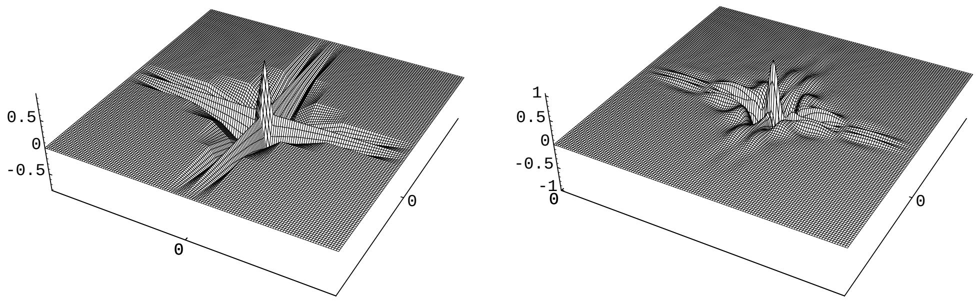

FIG. 3.1. *Bilinear* ($m = 2, d = 2$) *spline* $\Phi_{4,(0,0)}$ *and bicubic* ($m = 4, d = 2$) *spline* $\Phi_{4,(8,0)}$.

The wavelet ψ_j is given as the 2π-periodization of the the cardinal spline wavelet introduced by C. K. Chui and J. Z. Wang [4, 3]. Therefore, we define $\psi_j \in W_j$ by its *two–scale relation* $\psi_j = \sum_{k=0}^{3m-2} q_k \phi_{j+1,k}$ or in the Fourier transformed form $c_n(\psi_j) = 2Q(w_{d_{j+1}}^n)c_n(\phi_{j+1})$ with the *two–scale symbol*

$$Q(z) := \tfrac{1}{2}\sum_{k=0}^{3m-2} q_k z^k = (-z)^{m-1}\left(\frac{1-z}{2}\right)^m \Phi_{2m}(-z).$$

Figure 3.2 shows bivariate wavelet functions constructed from the periodized Chui-Wang wavelets.

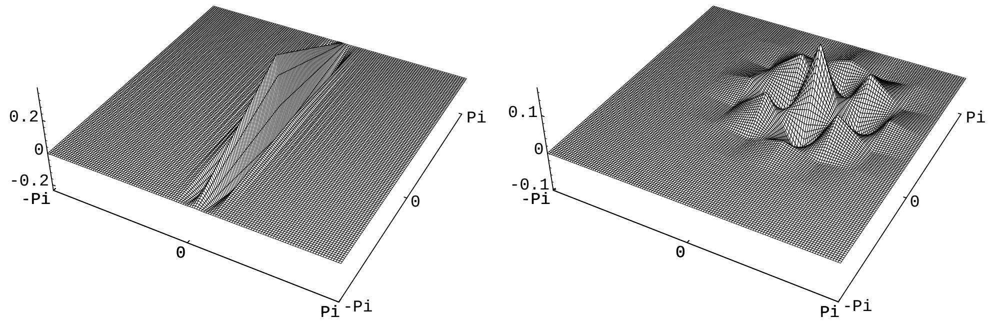

FIG. 3.2. *Bilinear* $(m = 2, d = 2)$ *spline wavelets* $\phi_0 \otimes \psi_3$ *and* $\psi_1 \otimes \psi_1$.

4. Fast univariate spline algorithms. Fast algorithms for spline interpolation, decomposition, reconstruction and computation of function values based on the FFT are described in [8]. Unfortunately this algorithms have complexity $\mathcal{O}(j2^j)$. Our goal is to apply algorithms of complexity $\mathcal{O}(2^j)$ to achieve the complexity $\mathcal{O}(j2^j)$ in the bivariate case.

Because we consider periodic functions we can assume that the coefficients are extended periodically, e.g. $a_{j,k} = a_{j,d_j+k}$. For the reconstruction it is well known, that in

$$\sum_{k=0}^{d_{j+1}-1} a_{j+1,k}\,\phi_{j+1,k} = \sum_{k=0}^{d_j-1} a_{j,k}\,\phi_{j,k} + \sum_{k=0}^{d_j-1} b_{j,k}\,\psi_{j,k} \tag{4.1}$$

the coefficients $a_{j+1,k}$ can be computed by

$$\begin{aligned} a_{j+1,2k} &= \sum_{l=0}^{m/2} p_{2l}\,a_{j,k-l} + \sum_{l=0}^{3m/2-1} q_{2l}\,b_{j,k-l}, \\ a_{j+1,2k+1} &= \sum_{l=0}^{m/2-1} p_{2l+1}\,a_{j,k-l} + \sum_{l=0}^{3m/2-2} q_{2l+1}\,b_{j,k-l}, \qquad k = 0,\dots,d_j-1 \end{aligned} \tag{4.2}$$

with $4m\,d_j$ real additions and multiplications. In the following we will count one real addition and one real multiplication together as one real operation.

To obtain a decomposition algorithm of complexity $\mathcal{O}(2^j)$ we refer to Berger and Strömberg [2], where an algorithm for fast decomposition based on recursive filters is described. Let $\lambda_{m,\nu} < 0$, $\nu = 1,\dots,r-1$ denote the zeros of the Euler-Frobenius polynomial inside the unit circle, which are all real and negative (see [13]). Further, we consider the coefficients r_k and s_k of the numerators of the decomposition symbols $H(z) = R(z)/T(z)$ and $G(z) = S(z)/T(z)$ (see [2, 8]) with

$$\begin{aligned} R(z) &:= \sum_{k=1-2m}^{m-1} r_k z^k = \frac{\Phi_{2m}(z)}{N_{2m}(1)} \left(\frac{1+z^{-1}}{2}\right)^m, \\ S(z) &:= \sum_{k=1-2m}^{1-m} s_k z^k = \frac{(-z)^{1-m}}{N_{2m}(1)} \left(\frac{1-z^{-1}}{2}\right)^m, \\ T(z) &:= \frac{\Phi_{2m}(z)}{N_{2m}(1)} = \prod_{\nu=1}^{m-1} (z-\lambda_{2m,\nu})(1-z^{-1}\lambda_{2m,\nu}^{-1}). \end{aligned}$$

With these notations the decomposition algorithm of Berger and Strömberg, which determines the coefficients $a_{j,k}$ and $b_{j,k}$ in (4.1) from given $a_{j+1,k}$ reads as follows.

ALGORITHM 4.1.
spline_dec $\Big((a_{j+1,k})_{k=0}^{d_{j+1}-1}\Big)$

1. For $k = 0, \ldots, d_j - 1$

$$u_k^{(1)} = \sum_{l=1-2m}^{m-1} r_l \, a_{j,2k-l}, \quad v_k^{(1)} = \sum_{l=1-2m}^{1-m} s_l \, a_{j,2k-l}$$

2. For $\nu = 1, \ldots, m-1$

(a) $u_0^{(2\nu)} = \dfrac{1}{1-\lambda_{2m,\nu}^{d_j}} \displaystyle\sum_{s=0}^{d_j-1} \lambda_{2m,\nu}^s u_{s+1}^{(2\nu-1)}, \quad v_0^{(2\nu)} = \dfrac{1}{1-\lambda_{2m,\nu}^{d_j}} \displaystyle\sum_{s=0}^{d_j-1} \lambda_{2m,\nu}^s v_{s+1}^{(2\nu-1)}$

(b) For $k = d_j$ *downto* 2

$$u_{k-1}^{(2\nu)} = u_k^{(2\nu-1)} + \lambda_{2m,\nu} u_k^{(2\nu)}, \quad v_{k-1}^{(2\nu)} = v_k^{(2\nu-1)} + \lambda_{2m,\nu} v_k^{(2\nu)}$$

(c) $u_0^{(2\nu+1)} = \dfrac{-1}{1-\lambda_{2m,\nu}^{d_j}} \displaystyle\sum_{s=1}^{d_j} \lambda_{2m,\nu}^s u_{-s}^{(2\nu)}, \quad v_0^{(2\nu+1)} = \dfrac{-1}{1-\lambda_{2m,\nu}^{d_j}} \displaystyle\sum_{s=1}^{d_j} \lambda_{2m,\nu}^s v_{-s}^{(2\nu)}$

(d) For $k = 0, \ldots, d_j - 2$

$$u_{k+1}^{(2\nu+1)} = \lambda_{2m,\nu}(u_k^{(2\nu+1)} - u_k^{(2\nu)}), \quad v_{k+1}^{(2\nu+1)} = \lambda_{2m,\nu}(v_k^{(2\nu+1)} - v_k^{(2\nu)})$$

3. For $k = 0, \ldots, d_j - 1 \qquad a_{j,k} = u_k^{(2m-1)}, \quad b_{j,k} = v_k^{(2m-1)}$

Return $\Big((a_{j,k})_{k=0}^{d_j-1}, \ (b_{j,k})_{k=0}^{d_j-1}\Big)$.

The number of real operations yields as follows. Let us assume that the values of $\lambda_{2m,\nu}^s$ are precomputed. For the first step, we need $4m\,d_j$ operations. In the second step, for each ν we need $2(d_j+1)$ operations for *(a)* as well as for *(c)* and $2(d_j-1)$ operations for *(b)* and *(d)* each. Therefore, the second step needs $(8m-8)d_j$ operations at all. Together these are $(12m-8)d_j$ operations for the decomposition algorithm.

Furthermore, we have to consider an interpolation algorithm. For given function values $y_{j,k} = f(t_{j,k})$, we want to determine the coefficients $a_{j,k}$ such that

$$\sum_{k=0}^{d_j-1} a_{j,k} \phi_{j,k}(t_{j,l}) = y_{j,l}.$$

In [8], the Fourier transformed solution for even spline order $m = 2r$ is given by

$$\hat{a}_{j,n} = \frac{w_{d_j}^{-nr}}{\Phi_{2r}(w_{d_j}^n)} \, \hat{y}_{j,n}, \quad n \in \mathbb{Z}.$$

Because the denominator of the symbol $w_{d_j}^{-nr}/\Phi_{2r}(w_{d_j}^n)$ is similar to the denominator of the decomposition symbols, we can apply again the idea of Berger and Strömberg, which leads to the following interpolation algorithm.

ALGORITHM 4.2.
spline_int $\left((y_{j,k})_{k=0}^{d_j-1}\right)$

1. *For* $k = 0, \dots, d_j - 1$ $\qquad u_k^{(1)} = \dfrac{y_{j,k+r}}{N_{2r}(1)}$
2. *For* $\nu = 1, \dots, r-1$
   (a) $\quad u_0^{(2\nu)} = \dfrac{1}{1-\lambda_{2r,\nu}^{d_j}} \sum_{s=0}^{d_j-1} \lambda_{2r,\nu}^s u_{s+1}^{(2\nu-1)}$
   (b) *For* $k = d_j$ *downto* 2 $\qquad u_{k-1}^{(2\nu)} = u_k^{(2\nu-1)} + \lambda_{2r,\nu} u_k^{(2\nu)}$
   (c) $\quad u_0^{(2\nu+1)} = \dfrac{-1}{1-\lambda_{2r,\nu}^{d_j}} \sum_{s=1}^{d_j} \lambda_{2r,\nu}^s u_{-s}^{(2\nu)}$
   (d) *For* $k = 0, \dots, d_j - 2$ $\qquad u_{k+1}^{(2\nu+1)} = \lambda_{2r,\nu}(u_k^{(2\nu+1)} - u_k^{(2\nu)})$
3. *For* $k = 0, \dots, d_j - 1$ $\qquad a_{j,k} = u_k^{(2m-1)}$

Return $(a_{j,k})_{k=0}^{d_j-1}$.

For this algorithm, we count d_j operations in the first step and $(4r-4)d_j$ operations in the second step. Hence, the absolute number of real operations for this algorithm is $(2m-3)d_j$.

5. Algorithms for bivariate spline wavelets on sparse grids. In this section we will describe algorithms for the decomposition of a bivariate function $f_{j+1} \in V_{j+1}^B$ into a scaling part $f_j \in V_j^B$ and a wavelet part $g_j \in W_j^B$ as well as for the reconstruction of f_{j+1} from f_j and g_j. The main difficulty of the Boolean sum approach is the structure of the basis functions $\Phi_{j,(k,l)}$. That is why we propose the generating system (2.3) for our algorithms. In this way, we can use the ideas from the tensor product case. Thus, we can establish fast algorithms, which have a simple structure and are based on the univariate algorithms. The algorithms are independent from the underlying univariate wavelets. For simplicity and because we want to give exact operation counts we will restrict ourselves to spline wavelets.

5.1. Interpolation algorithm for sparse grids. To obtain a representation of a function $f \in V_j^B$ as a linear combination of elements from the generating system (2.3) we have to compute coefficients $a_{r,(k,l)}$ with $f = \sum_{r,k,l} a_{r,(k,l)} \phi_{r,k} \otimes \phi_{j-r,l}$. From $f = B_j f$, it follows that the coefficients $a_{r,(k,l)}$ are a solution of the following interpolation problem. For all $(\mu,\nu) \in \mathcal{J}_j$

$$f(t_{j,\mu}, t_{j,\nu}) = B_j f(t_{j,\mu}, t_{j,\nu}) = \sum_{r=0}^{j} \sum_{k=0}^{d_r-1} \sum_{l=0}^{d_{j-r}-1} a_{r,(k,l)} \phi_{r,k}(t_{j,\mu}) \phi_{j-r,l}(t_{j,\nu}).$$

Because we have not a basis the interpolation problem has several solutions $a_{r,(k,l)}$, which represent all the same function. To obtain one of these solutions we will first describe an interpolation algorithm for sparse grids. With

$$f_{j,r} := L_r \otimes (L_{j-r} - L_{j-r-1}) f = \sum_{k=0}^{d_r-1} \sum_{l=0}^{d_{j-r}-1} a_{r,(j,k)} \phi_{r,k} \otimes \phi_{j-r,l} \in V_r \otimes V_{j-r},$$

where $L_{-1} := 0$, it holds that $B_j f = \sum_{r=0}^{j} f_{j,r}$. Now we write

$$f_{j,r}(x,y) = L_r\Big(\underbrace{(L_{j-r} - L_{j-r-1}) f(\cdot, y)}_{g_r(\cdot,y)}\Big)(x).$$

This means for each $l = 0, \ldots, d_{j-r} - 1$ we have to interpolate the values

$$g_r(t_{r,k}, t_{j-r,l}) = \big((L_{j-r} - L_{j-r-1}) f(\cdot, t_{j-r,l})\big)(t_{r,k}), \quad k = 0, \ldots, d_r - 1.$$

Applying the univariate interpolation algorithm for y, we obtain for $r < j$

$$g_r(t_{r,k}, y) = \sum_{l=0}^{d_{j-r}-1} u_{r,(k,l)} \phi_{j-r,l}(y) - \sum_{l=0}^{d_{j-r-1}-1} u_{r+1,(2k,l)} \phi_{j-r-1,l}(y)$$

and $g_j(t_{j,k}, y) = \sum_{l=0}^{d_j-1} u_{j,(k,l)} \phi_{j,l}(y)$ with

$$(u_{r,(k,l)})_{l=0}^{d_{j-r}-1} = \textbf{spline\_int}\left(\big(f(t_{r,k}, t_{j-r,\nu})\big)_{\nu=0}^{d_{j-r}-1}\right), \quad k = 0, \ldots, d_r - 1.$$

By the univariate reconstruction formula (4.2), we compute coefficients $v_{r,(k,l)}$, such that

$$g_r(t_{r,k}, y) = \sum_{l=0}^{d_{j-r}-1} v_{r,(k,l)} \phi_{j-r,l}(y).$$

Finally, we have

$$\begin{aligned} f_{j,r}(t_{r,\mu}, t_{j-r,\nu}) = g_r(t_{r,\mu}, t_{j-r,\nu}) &= \sum_{l=0}^{d_{j-r}-1} v_{r,(\mu,l)} \phi_{j-r,l}(t_{j-r,\nu}) \\ &= \sum_{k=0}^{d_{j-r}-1} \left(\sum_{k=0}^{d_r-1} a_{r,(j,k)} \phi_{r,k}(t_{r,\mu}) \right) \phi_{j-r,l}(t_{j-r,\nu}) \end{aligned}$$

and therefore

$$(a_{r,(k,l)})_{k=0}^{d_r-1} = \textbf{spline\_int}\left((v_{r,(\mu,l)})_{\mu=0}^{d_r-1}\right), \quad l = 0, \ldots, d_{j-r} - 1.$$

Hence, the spline interpolation algorithm for sparse grids reads as follows.

ALGORITHM 5.1.

bool_int $\big((f(t_{j,\mu}, t_{j,\nu}))_{(\mu,\nu)\in\mathcal{J}_j}\big)$

1. For $r = 0, \ldots, j$

For $k = 0, \ldots, d_r - 1$

$$(u_{r,(k,l)})_{l=0}^{d_{j-r}-1} = \textbf{spline\_int}\left(\big(f(t_{r,k}, t_{j-r,\nu})\big)_{\nu=0}^{d_{j-r}-1}\right)$$

2. For $r = 0, \ldots, j-1$

For $k = 0, \ldots, d_r - 1;\quad l = 0, \ldots, d_{j-r-1} - 1$

$$\begin{aligned} v_{r,(k,2l)} &= u_{r,(k,2l)} - \textstyle\sum_{s=0}^{m/2} p_{2s}\, u_{r+1,(2k,l-s)} \\ v_{r,(k,2l+1)} &= u_{r,(k,2l+1)} - \textstyle\sum_{s=0}^{m/2-1} p_{2s+1}\, u_{r+1,(2k,l-s)} \end{aligned}$$

For $k = 0, \ldots, d_j - 1;\quad l = 0, \ldots, d_0 - 1 \qquad v_{j,(k,l)} = u_{j,(k,l)}$

3. For $r = 0, \ldots, j$

For $l = 0, \ldots, d_{j-r} - 1$

$$(a_{r,(k,l)})_{l=0}^{d_r-1} = \textbf{spline\_int}\left((v_{r,(\mu,l)})_{\mu=0}^{d_r-1}\right)$$

Return $(a_{r,(k,l)})_{r,k,l}$.

In this algorithm, we need $\sum_{r=0}^{j}(2m-3)d_{j-r}d_r$ real operations in the first as well as in the third step and $\sum_{r=0}^{j-1}(m+1)d_{j-r-1}d_r$ operations in the second step. This are altogether

$$d^2(j+1)(4m-6)2^j + d^2 j(m+1)2^{j-1} = (9mj - 11j + 8m - 12)d^2 2^{j-1}$$

operations.

5.2. Decomposition. Now we want to decompose a given function $f_{j+1} \in V_{j+1}^B$ into functions $f_j \in V_j^B$ and $g_j \in W_j^B$ with $f_j + g_j = f_{j+1}$, i.e., we have to determine coefficients $a_{r,(k,l)}$ and $b_{r,(k,l)}$ such that

$$\begin{aligned}
&\sum_{r=0}^{j+1}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j+1-r}-1} c_{r,(k,l)}\phi_{r,k}\otimes\phi_{j+1-r,l} \\
&= \sum_{r=0}^{j}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1} a_{r,(k,l)}\,\phi_{r,k}\otimes\phi_{j-r,l} + \sum_{r=0}^{j+1}\sum_{k=0}^{d_{r-1}-1}\sum_{l=0}^{d_{j-r}-1} b_{r,(k,l)}\,\psi_{r-1,k}\otimes\psi_{j-r,l}
\end{aligned}\tag{5.1}$$

for given $c_{r,(k,l)}$. Applying the univariate decomposition algorithm first with respect to y and then with respect to x we obtain

$$\begin{aligned}
f_{j+1} &= \sum_{r=0}^{j}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1}\left(\tilde{u}_{r,(k,l)}\,\phi_{r,k}\otimes\phi_{j-r,l} + \tilde{v}_{r,(k,l)}\,\phi_{r,k}\otimes\psi_{j-r,l}\right) \\
&\quad + \sum_{k=0}^{d_{j+1}-1}\sum_{l=0}^{d_0-1}\tilde{v}_{j+1,(k,l)}\,\phi_{j+1,k}\otimes\phi_{0,l} \\
&= \sum_{r=0}^{j}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1}\tilde{u}_{r,(k,l)}\,\phi_{r,k}\otimes\phi_{j-r,l} + \sum_{l=0}^{d_j-1}\sum_{k=0}^{d_{-1}-1} b_{0,(k,l)}\,\psi_{-1,k}\otimes\psi_{j,l} \\
&\quad + \sum_{r=1}^{j}\sum_{l=0}^{d_{j-r}-1}\sum_{k=0}^{d_{r-1}-1}\left(\tilde{w}_{r,(k,l)}\,\phi_{r-1,k}\otimes\psi_{j-r,l} + b_{r,(k,l)}\,\psi_{r-1,k}\otimes\psi_{j-r,l}\right) \\
&\quad + \sum_{k=0}^{d_j-1}\sum_{l=0}^{d_0-1}\left(\tilde{w}_{j+1,(k,l)}\,\phi_{j,k}\otimes\phi_{0,l} + b_{j+1,(k,l)}\,\psi_{j,k}\otimes\psi_{-1,l}\right).
\end{aligned}$$

By the univariate reconstruction formula (4.2), we can then compute the values $a_{r,(k,l)}$ such that

$$f_{j+1} = \underbrace{\sum_{r=0}^{j}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1} a_{r,(k,l)}\,\phi_{r,k}\otimes\phi_{j-r,l}}_{f_j\in V_j^B} + \underbrace{\sum_{r=0}^{j+1}\sum_{k=0}^{d_{r-1}-1}\sum_{l=0}^{d_{j-r}-1} b_{r,(k,l)}\,\psi_{r-1,k}\otimes\psi_{j-r,l}}_{g_j\in W_j^B}.$$

From this equalities, we derive the bivariate decomposition algorithm, which determines the coefficients $a_{r,(k,l)}$ and $b_{r,(k,l)}$.

ALGORITHM 5.2.
bool_dec$\left((c_{r,(k,l)})_{r,k,l}\right)$

1. *For* $r = 0, \dots, j$
   For $k = 0, \dots, d_r - 1$
   $$\left(\left(\tilde{u}_{r,(k,l)}\right)_{l=0}^{d_{j-r}-1}, \tilde{v}_{r,(k,l)}\right)_{l=0}^{d_{j-r}-1}\right) = \textbf{spline\_dec}\left(\left(c_{r,(k,l)}\right)_{l=0}^{d_{j+1-r}-1}\right)$$
   For $k = 0, \dots, d_{j+1} - 1;\quad l = 0, \dots, d_0 - 1 \qquad \tilde{v}_{j+1,(k,l)} = c_{j+1,(k,l)}$
2. *For* $k = 0, \dots, d_0 - 1;\quad l = 0, \dots, d_j - 1 \qquad b_{0,(k,l)} = \tilde{v}_{0,(k,l)}$
   For $r = 1, \dots, j + 1$
   For $l = 0, \dots, d_{j-r} - 1$
   $$\left(\left(\tilde{w}_{r,(k,l)}\right)_{k=0}^{d_{r-1}-1}, b_{r,(k,l)}\right)_{k=0}^{d_{r-1}-1}\right) = \textbf{spline\_dec}\left(\left(\tilde{v}_{r,(k,l)}\right)_{k=0}^{d_r-1}\right)$$
3. *For* $r = 1, \dots, j - 1$
   For $k = 0, \dots, d_r - 1;\quad l = 0, \dots, d_{j-r-1} - 1$
   $$\begin{aligned} a_{r,(k,2l)} &= \tilde{u}_{r,(k,2l)} + \sum_{s=0}^{3m/2-1} q_{2s}\, \tilde{w}_{r+1,(k,l-s)} \\ a_{r,(k,2l+1)} &= \tilde{u}_{r,(k,2l+1)} + \sum_{s=0}^{3m/2-2} q_{2s+1}\, \tilde{w}_{r+1,(k,l-s)} \end{aligned}$$
   For $k = 0, \dots, d_j - 1;\quad l = 0, \dots, d_0 - 1 \qquad a_{j,(k,l)} = \tilde{u}_{j,(k,l)} + \tilde{w}_{j+1,(k,l)}$

Return $\left((a_{r,(k,l)})_{r,k,l},\ (b_{r,(k,l)})_{r,k,l}\right)$.

This algorithm needs $\sum_{r=0}^{j} d_r(12m-8)d_{j-r}$ real operations for the first, $d_0(12m-8)d_j + \sum_{r=1}^{j} d_{j-r}(12m-8)d_{r-1}$ operations for the second step and finally $\sum_{r=1}^{j-1} d_r(3m-1)d_{j-r-1}$ operations for the third step. Therefore, the decomposition algorithm needs

$$d^2(3j+4)(6m-4)2^j + d^2(j-1)(3m-1)2^{j-1} = (39mj + 45m - 25j - 31)d^2 2^{j-1}$$

operations.

5.3. Reconstruction. Analogously, we can derive a reconstruction algorithm, which determines the coefficients $c_{r,(k,l)}$ from given coefficients $a_{r,(k,l)}$ and $b_{r,(k,l)}$ in (5.1).

ALGORITHM 5.3.
bool_rec$\left((a_{r,(k,l)})_{r,k,l},\ (b_{r,(k,l)})_{r,k,l}\right)$

1. *For* $l = 0, \dots, d_j - 1;\quad k = 0, \dots, d_0 - 1 \qquad \breve{u}_{0,(k,l)} = b_{0,(k,l)}$
   For $r = 1, \dots, j + 1$
   For $l = 0, \dots, d_{j-r} - 1;\quad k = 0, \dots, d_{r-1} - 1$
   $$\begin{aligned} \breve{u}_{r,(2k,l)} &= \sum_{s=0}^{3m/2-1} q_{2s}\, b_{r,(k-s,l)} \\ \breve{u}_{r,(2k+1,l)} &= \sum_{s=0}^{3m/2-2} q_{2s+1}\, b_{r,(k-s,l)} \end{aligned}$$
2. *For* $r = 0, \dots, j$
   For $k = 0, \dots, d_r - 1;\quad l = 0, \dots, d_{j-r} - 1$
   $$\begin{aligned} c_{r,(k,2l)} &= \sum_{s=0}^{m/2} p_{2s}\, a_{r,(k,l-s)} + \sum_{s=0}^{3m/2-1} q_{2s}\, \breve{u}_{r,(k,l-s)} \\ c_{r,(k,2l+1)} &= \sum_{s=0}^{m/2-1} p_{2s+1}\, a_{r,(k,l-s)} + \sum_{s=0}^{3m/2-2} q_{2s+1}\, \breve{u}_{r,(k,l-s)} \end{aligned}$$
   For $k = 0, \dots, d_{j+1} - 1;\quad l = 0, \dots, d_0 - 1 \qquad c_{j+1,(k,l)} = \breve{u}_{j+1,(k,l)}$

Return $(c_{r,(k,l)})_{r,k,l}$.

This reconstruction algorithm needs $\sum_{r=1}^{j} d_{j-r}(3m-1)d_{r-1} + d_0(3m-1)d_j$ operations in the first step and $\sum_{r=0}^{j} d_r\, 4m\, d_{j-r}$ operations in the second step. Summing up, these are

$$d^2(j+1)4m\,2^j + d^2(j+2)(3m-1)2^{j-1} = (11mj + 14m - j - 2)d^2 2^{j-1}$$

real operations.

5.4. Computation of function values on the sparse grid. Finally, we consider the question how to compute function values of $f \in V_j^B$. In particular, we have to determine the values for all points of the sparse grid $\mathcal{T}_j^B$ to obtain the coefficients $\gamma_{k,l} = f(t_{j,k}, t_{j,l})$ in

$$f = \sum_{(k,l)\in\mathcal{J}_j} \gamma_{k,l}\Phi_{j,(k,l)}.$$

Because of supp $\phi_{j,k} = [t_{j,k}, t_{j,k+m}]$, function values of

$$f_{j,r} = \sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1} a_{r,(k,l)}\phi_{r,k}\otimes\phi_{j-r,l}$$

can be computed for $(x,y) \in [t_{r,\mu}, t_{r,\mu+1}) \times [t_{r,\nu}, t_{r,\nu+1})$ with $c\,m^2$ operations by

$$f_{j,r}(x,y) = \sum_{k=\mu}^{\mu+m-1}\sum_{l=\nu}^{\nu+m-1} a_{r,(k,l)}\phi_{r,k}(x)\phi_{j-r,l}(y).$$

To determine the sum $f(x,y) = \sum_{r=0}^{j} f_{j,r}(x,y)$ we need then $cj\,m^2$ operations. In this way, the computation of all coefficients $\gamma_{k,l}$, $(k,l)\in\mathcal{J}_j$ would require $\mathcal{O}(j^2 2^j)$ operations. To reduce the complexity further to $\mathcal{O}(j2^j)$ we will use the refinability of ϕ_j.

First we introduce index sets $\mathcal{I}_{j,r}$ with $\mathcal{I}_{j,0} := \{2^j k : k = 0,\ldots,d_0-1\}$ and $\mathcal{I}_{j,r} := \{2^{j-r}(2k+1) : k = 0,\ldots,d_{r-1}-1\}$ for $r = 1,\ldots,j$. By this definition we have $\{k :\ k = 0,\ldots,d_j-1\} = \bigcup_{r=0}^{j}\mathcal{I}_{j,r}$ and

$$\mathcal{T}_j = \bigcup_{r=0}^{j}\{(t_{r,\mu}, t_{j,\nu}) :\ \nu\in\mathcal{I}_{j,r},\ \mu = 0,\ldots,d_{j-r}-1\}.$$

Now we consider for $\nu\in\mathcal{I}_{j,r}$ the values

$$b_{\mu,\nu} := \sum_{s=r}^{j} f_{j,s}(t_{r,\mu}, t_{j,\nu}) = \sum_{s=r}^{j}\sum_{l=0}^{d_{j-s}-1} u_{s,(2^{s-r}\mu,l)}\phi_{j-s,l}(t_{j,\nu}),$$

where $u_{s,(\mu,l)} = \sum_{k=0}^{d_s-1} a_{s,(k,l)}\phi_{s,k}(t_{s,\mu})$. By the reconstruction formula (4.2) we compute coefficients $v_{\mu,l}$, such that $b_{\mu,\nu} = \sum_{l=0}^{d_{j-r}-1} v_{\mu,l}\phi_{j-r,l}(t_{j,\nu})$. Analogously, we determine the values $c_{\mu,\nu} := \sum_{s=0}^{j-r} f_{j,s}(t_{j,\mu}, t_{r,\nu})$, if $\mu\in\mathcal{I}_{j,r}$.

Let $(\mu,\nu) \in 2^{j-\kappa}\mathcal{K}_{\kappa,r}$. Then it follows that $\mu \in \mathcal{I}_{j,r}$ and $\nu \in \mathcal{I}_{j,\kappa-r}$ and we can determine the values of f by

$$\begin{aligned} f(t_{j,\mu},t_{j,\nu}) &= \sum_{s=0}^{j-r} f_{j,s}(t_{j,\mu},t_{r,2^{r-j}\nu}) + \sum_{s=\kappa-r}^{j} f_{j,s}(t_{\kappa-r,2^{\kappa-r-j}\mu},t_{j,\nu}) \\ &\qquad - \sum_{s=\kappa-r}^{j-r} f_{j,s}(t_{s,2^{s-j}\mu},t_{j-s,2^{-s}\nu}) \\ &= c_{\mu,2^{r-j}\nu} + b_{2^{\kappa-r-j}\mu,\nu} - \sum_{s=\kappa-r}^{j-r} f_{j,s}(t_{s,2^{s-j}\mu},t_{j-s,2^{-s}\nu}). \end{aligned}$$

To obtain the function values $\phi_j(t_{j,k})$ we use the definition (3.2), such that we only need the values $N_m(k)$ of the cardinal B-spline for $k = 1,\dots,m-1$. Hence, the function values $y_{\mu,\nu} = f(t_{j,\mu},t_{j,\nu})$ on the sparse grid can be computed with the following algorithm.

ALGORITHM 5.4.

sparse_val$\big((a_{r,(k,l)})_{r,k,l}\big)$

1. *For* $r = 0,\dots,j$
   For $\nu = 0,\dots,d_{j-r}-1;\quad \mu = 0,\dots,d_r-1$
$$\begin{aligned} u^{(0)}_{r,(\mu,\nu)} &= \textstyle\sum_{k=1}^{m-1} a_{r,(\mu-k,\nu)}\, N_m(k) \\ u^{(1)}_{r,(\mu,\nu)} &= \textstyle\sum_{l=1}^{m-1} a_{r,(\mu,\nu-l)}\, N_m(l) \\ w_{r,(\mu,\nu)} &= \textstyle\sum_{l=1}^{m-1} u^{(0)}_{r,(\mu,\nu-l)}\, N_m(l) \end{aligned}$$
   (* $w_{r,(\mu,\nu)} = f_{j,r}(t_{r,2^{r-j}\mu},t_{j-r,2^{-r}\nu})$ *)
2. *For* $\kappa = 0,\dots,j;\quad k \in \mathcal{I}_{j,\kappa}$
   For $r = 1,\dots,j-\kappa$
   For $l = 0,\dots,d_{r-1}-1$
$$\begin{aligned} u^{(0)}_{j-r,(2^{j-r-\kappa}k,2l)} &= u^{(0)}_{j-r,(2^{j-r-\kappa}k,2l)} + \textstyle\sum_{s=0}^{m/2} p_{2s}\, u^{(0)}_{j-r+1,(2^{j-r+1-\kappa}k,l-s)} \\ u^{(0)}_{j-r,(2^{j-r-\kappa}k,2l+1)} &= u^{(0)}_{j-r,(2^{j-r-\kappa}k,2l+1)} + \textstyle\sum_{s=0}^{m/2-1} p_{2s+1}\, u^{(0)}_{j-r+1,(2^{j-r+1-\kappa}k,l-s)} \end{aligned}$$
$$\begin{aligned} u^{(1)}_{r,(2l,2^{j-r-\kappa}k)} &= u^{(1)}_{r,(2l,2^{j-r-\kappa}k)} + \textstyle\sum_{s=0}^{m/2} p_{2s}\, u^{(1)}_{r-1,(l-s,2^{j-r+1-\kappa}k)} \\ u^{(1)}_{r,(2l+1,2^{j-r-\kappa}k)} &= u^{(1)}_{r,(2l+1,2^{j-r-\kappa}k)} + \textstyle\sum_{s=0}^{m/2-1} p_{2s+1}\, u^{(1)}_{r-1,(l-s,2^{j-r+1-\kappa}k)} \end{aligned}$$
3. *For* $\kappa = 0,\dots,j;\quad k \in \mathcal{I}_{j,\kappa}$
   $l = 0,\dots,d_{j-\kappa}-1$
$$\begin{aligned} b_{l,k} &= \textstyle\sum_{s=1}^{m-1} u^{(0)}_{\kappa,(l,2^{\kappa-j}k-s)}\, N_m(s) \\ c_{k,l} &= \textstyle\sum_{s=1}^{m-1} u^{(1)}_{\kappa,(2^{\kappa-j}k-s,l)}\, N_m(s) \end{aligned}$$
4. *For* $\kappa = 0,\dots,j;\quad s = 0,\dots,\kappa;\qquad (\mu,\nu) \in \mathcal{K}_{\kappa,r}$
$$y_{\mu,\nu} = c_{\mu,2^{r-j}\nu} + b_{2^{\kappa-r-j}\mu,\nu} - \sum_{s=\kappa-r}^{j-r} w_{s,(2^{s-j}\mu,2^{-s}\nu)}$$

Return $(y_{\mu,\nu})_{(\mu,\nu)\in\mathcal{J}_j}$.

Now we count the number of real operations for this algorithm. Obviously, the first step needs $d^2(j+1)3(m-1)2^j$ operations. The number of operations for the second step is given by

$$2\left(d_0(m+1)\sum_{r=1}^{j} d_{r-1} + \sum_{\kappa=1}^{j} d_{\kappa-1}(m+1)\sum_{r=1}^{j-\kappa} d_{r-1}\right)$$
$$= d^2 2(m+1)\left(\sum_{r=1}^{j} 2^{r-1} + \sum_{\kappa=1}^{j} 2^{\kappa-1}\sum_{r=1}^{j-\kappa} 2^{r-1}\right) \leq d^2(j+2)(m+1)2^j.$$

Analogously, we obtain $d^2(j+2)(m-1)2^j$ operations in the third step. In the fourth step we have only real additions. Because each value $w_{r,(\mu,\nu)}$ is used once we obtain the number of this additions by

$$\#(\mathcal{J}_j) + \sum_{r=0}^{j}\sum_{k=0}^{d_r-1}\sum_{l=0}^{d_{j-r}-1} 1 = d^2(3j+4)2^{j-1}.$$

Therefore, we can estimate the number of real operations for this algorithm by

$$d^2(5m-3)(j+2)2^j.$$

In the following table we sum up our operation counts for the Boolean sum algorithms and compare it with the operation numbers for the corresponding tensor product methods.

Algorithm	Boolean sum	Tensor product
Interpolation	$(9mj+8m-11j-12)\,d^2\,2^{j-1}$	$(4m-6)\,d^2\,2^{2j}$
Decomposition	$(39mj+45m-25j-31)\,d^2\,2^{j-1}$	$(24m-16)\,d^2\,2^{2j}$
Reconstruction	$(11mj+14m-j-2)\,d^2\,2^{j-1}$	$8m\,d^2\,2^{2j}$
Function values	$(5m-3)(j+2)\,d^2\,2^j$	$(2m-2)\,d^2\,2^{2j}$

Hence, the reconstruction and decomposition algorithms for spline wavelets on sparse grids have the arithmetical complexity $\mathcal{O}(j2^j)$, while the corresponding algorithms for tensor products have complexity $\mathcal{O}(2^{2j})$.

REMARK 5.5. *From the structure of the sparse grids one would expect some problems in the implementation of the algorithm. However, because of the simple structure of the generating system* (2.3) *this problem does not occur in the Algorithms* 5.2 *and* 5.3 *for decomposition and reconstruction. In Algorithm* 5.4, *the second, third and fourth step show how one can deal with the sparse grids using the index sets* $\mathcal{I}_{j,r}$ *and* $\mathcal{K}_{\kappa,r}$ *without a considerable loss of performance. Analogously, we can handle this problem in step 1 of Algorithm* 5.1. *Therefore, the number of floating point operations is suited for a comparison of tensor product methods and our Boolean sum algorithms.*

6. Numerical examples. Finally, we present the results of numerical tests. The detection of higher order discontinuities is an important property of wavelets. The reason for this is, that the wavelet part of a function is the error of the orthogonal projection, which depends on the smoothness of the function such that the large wavelet coefficients are localized near the discontinuities.

Therefore, we consider first the approximation properties of the interpolation projectors. Let $f \in C^{r,r}(\mathbb{T}^2)$ with

$$C^{r,r}(\mathbb{T}^2) = \left\{ f \in C(\mathbb{T}^2) : D^{k,l} f \in C(\mathbb{T}^2),\ k,l \leq r \right\}.$$

In [5], it is shown that for $r \leq m$ the error of spline blending interpolation on sparse grids is

$$\|f - B_j f\|_\infty = \mathcal{O}\left(j 2^{-r\,j}\right), \qquad j \to \infty,$$

while the error of tensor product interpolation on a regular grid is given by

$$\|f - (L_j \otimes L_j) f\|_\infty = \mathcal{O}\left(2^{-r\,j}\right), \qquad j \to \infty.$$

That means, we obtain for Boolean sums only an extra logarithmic factor in approximation error, while the cost for the algorithms is remarkably lower. On the other hand, if

$$f \in C^r(\mathbb{T}^2) = \left\{ f \in C(\mathbb{T}^2) : D^{k,l} f \in C(\mathbb{T}^2),\ k + l \leq r \right\},$$

it can be shown, that for B_j it holds that

$$\|f - B_j f\|_\infty = \mathcal{O}\left(2^{-j\,r/2}\right), \qquad j \to \infty.$$

In this case, the approximation order of the tensor product interpolation operator is

$$\|f - (L_j \otimes L_j) f\|_\infty = \mathcal{O}\left(2^{-j\,r}\right), \qquad j \to \infty.$$

For our tests, we choose bivariate spline functions, because this functions have discontinuities in its derivatives at well known lines. Therefore, we interpolate and decompose the tensor product spline $N_4 \otimes N_4$ as well as the box spline $B_{(222)}$ both centered in the point (π, π) and periodically extended. Both functions are contained in $C^2(\mathbb{T}^2)$. On the other hand, $N_4 \otimes N_4$ is contained in $C^{2,2}(\mathbb{T}^2)$, too, whereas $B_{(222)}$ is only contained in $C^{1,1}(\mathbb{T}^2)$ but not in $C^{2,2}(\mathbb{T}^2)$. With these examples, we demonstrate for which kind of functions the Boolean sum method works well. The error estimates given above let expect good results of the Boolean sum algorithm only for the tensor product spline. We choose the underlying spline wavelets of order $m = 6$ to rule out effects from the spline structure of our test functions. Furthermore, we set $d = 1$, because with this setting the special properties of the Boolean sum approach should appear most visible.

We compare the results of our algorithms with the tensor product methods. For tensor products, we decompose the functions from level 9 to level 8. In the Boolean sum case we choose once the decomposition from level 15 to 14. Because of $\dim(V_9 \otimes V_9) \approx \dim V_{15}^B$ we have around the same amount of memory in this way. On the other hand, we perform the decomposition from level 11 to 10. The results show that for this level the approximation error for $N_4 \otimes N_4$ is roughly the same as in the tensor product approximation to level 8.

Figure 6.1 shows density plots of the absolute values of the wavelet part of the functions. For higher values the shade is darker. As expected the result for the tensor product spline looks better than for the box spline. The pictures for the box spline show, that only discontinuities on lines which are diagonal to the axes are detected. The reason for this is that the function is not smooth here in the sense of $C^{2,2}(\mathbb{T}^2)$. Furthermore, for the Boolean sum approach we have other localization properties, namely for the detection of singularities of derivatives in one direction we have good localization in this variable and bad localization in the other variable. On the other hand, for the detection of singularities in mixed derivatives the localization is equal in both directions.

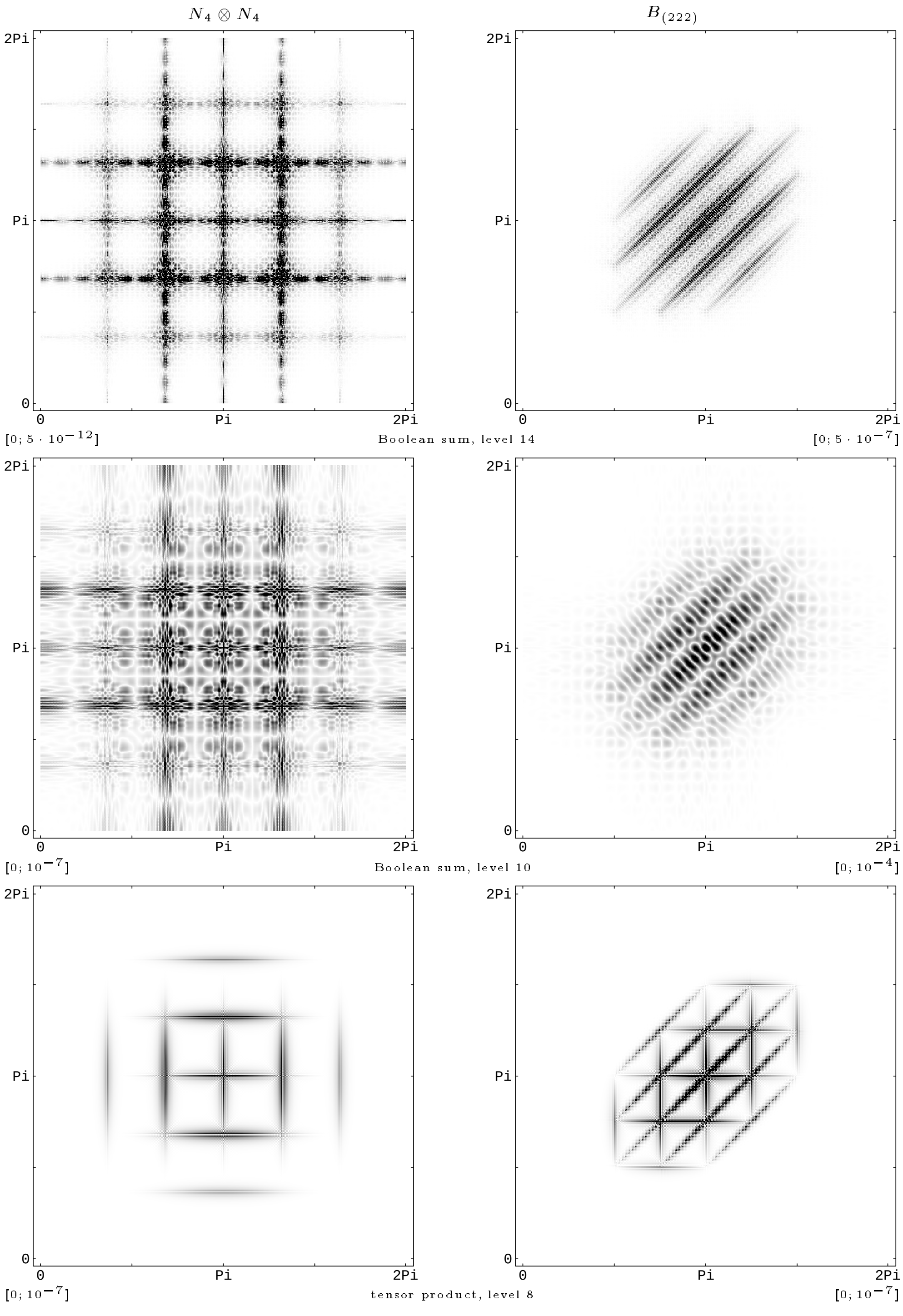

FIG. 6.1. *Density plot of absolute values of the wavelet part for different decompositions. The numbers in the brackets show the plot range (minimum=white; maximum=black).*

TABLE 6.1
Number of wavelet coefficients which are greater than ε for different decompositions.

	$N_4 \otimes N_4$						$B_{(222)}$					
	tens. prod. level 8		Bool. sum level 14		Bool. sum level 10		tens. prod. level 8		Bool. sum level 14		Bool. sum level 10	
ε	total	%	total	%	total	%	total	%	total	%	total	%
10^{-4}	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
10^{-5}	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	68	0.9
10^{-6}	0	0.0	0	0.0	0	0.0	0	0.0	4	0.0	248	3.5
10^{-7}	140	0.1	0	0.0	8	0.1	470	0.2	544	0.4	632	8.8
10^{-8}	2827	1.5	0	0.0	296	4.1	7268	3.7	1944	1.3	1012	14.1
10^{-9}	9036	4.6	0	0.0	1164	16.2	17644	9.0	4292	2.9	1376	19.2
10^{-10}	16688	8.5	0	0.0	1664	23.2	23510	12.0	7408	5.0	1652	23.0
10^{-11}	25112	12.8	136	0.1	1960	27.3	27526	14.0	11836	8.0	1832	25.6
10^{-12}	30068	15.3	1900	1.3	2132	29.7	31406	16.0	15836	10.7	1968	27.5
10^{-13}	33328	17.0	6880	4.7	2220	31.0	35540	18.1	18688	12.7	2100	29.3
10^{-14}	35884	18.3	12362	8.4	2300	32.1	39878	20.3	21056	14.3	2168	30.2
10^{-15}	38847	19.8	17332	11.8	2364	33.0	44264	22.5	23510	15.9	2220	31.0
10^{-16}	43960	22.4	21662	14.7	2436	34.0	48920	24.9	26444	17.9	2256	31.5
10^{-17}	51772	26.3	21662	20.5	2458	34.3	53770	27.3	31697	21.5	2292	32.0
10^{-18}	59846	30.4	30165	25.6	2470	34.5	58804	29.9	35592	24.1	2328	32.5
10^{-19}	66882	34.0	37768	28.8	2478	34.6	64102	32.6	37650	25.5	2352	32.8
10^{-20}	72058	36.7	42420	30.3	2490	34.7	69388	35.3	38958	26.4	2388	33.3
0	196608	100.0	147456	100.0	7168	100.0	196608	100.0	147456	100.0	7168	100.0

In Table 6.1, we see the number of wavelet coefficients with an absolute value greater than some threshold ε. Obviously, for the tensor product spline $N_4 \otimes N_4$ the Boolean sum method is more efficient than the tensor product algorithms. For the box spline, the results of the Boolean sum method are similar as in the tensor product case.

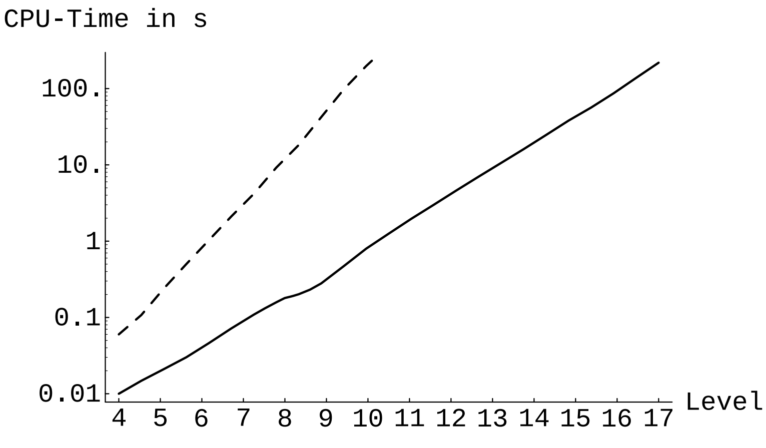

FIG. 6.2. *CPU-time (on a* Sun SPARCstation 20*) of tensor product algorithm (dashed line) and Boolean sum algorithm (continuous line) for $m = 6$ and $d = 1$.*

These examples show that the concept of Boolean sum seems to be more suitable for function spaces of the type $C^{r,r}(\mathbb{T}^2)$ whereas the tensor product methods appear to be better for the spaces of the type $C^r(\mathbb{T}^2)$. Therefore, a possible application for wavelets on sparse grids could be seen in the numerical treatment of PDE's, where the solution is contained in $C^{r,r}(\mathbb{T}^2)$ but not in $C^{r+1}(\mathbb{T}^2)$. For such applications, it should also be useful to investigate a Boolean sum approach for non-periodic wavelets on the square.

Finally, we draw in Figure 6.2 a comparison of the time needed for tensor product and Boolean sum algorithms. It shows that the Boolean sum method is remarkably faster than tensor product algorithms with the same quality of results for suited functions.

Acknowledgments. The author thanks the referees and the editor for their helpful comments and valuable suggestions to improve the manuscript. The research of K. Bittner was supported by the Deutsche Forschungsgemeinschaft. The author thanks Jürgen Prestin for suggesting the problem and the valuable comments.

REFERENCES

[1] G. Baszenski and F. J. Delvos, *A discrete Fourier transform scheme for Boolean sums of trigonometric operators.* in Multivariate Approximation Theory IV, C. K. Chui, W. Schempp and K. Zeller, eds., Birkhäuser, Basel, 1989, pp. 15–24.

[2] T. Berger and O. Strömberg, *Exact reconstruction algorithms for the discrete wavelet transform using spline wavelets*, Appl. Comp. Harmonic Anal., 2 (1995), pp. 392–397

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

[4] C. K. Chui and J. Z. Wang, *On compactly supported spline wavelets and a duality principle*, Trans. Amer. Math. Soc., 330 (1992), pp. 903–915.

[5] F.-J. Delvos and W. Schempp, *Boolean methods in interpolation and approximation*, Longman Scientific & Technical, Harlow 1989.

[6] F. Locher, *Interpolation on uniform meshes by translates of one function and related attenuation factors*, Math. Comp., 37 (1981), pp. 403–416.

[7] N. M. Korobov, *The approximate computation of multiple integrals*, Dokl. Akad. Nauk. SSSR, 124, (1959), pp. 1207–1210.

[8] G. Plonka and M. Tasche, *On the computation of periodic spline wavelets*, Appl. and Comp. Harmonic Anal., 2, (1995), pp. 1–14.

[9] G. Plonka and M. Tasche, *A unified approach to periodic wavelets*, in Wavelets: Theory, Algorithms and Applications, C. K. Chui, L Montefusco, and L. Puccio, eds., Academic Press, San Diego, 1994, pp. 137–151.

[10] G. Pöplau and F. Sprengel, *Some Error Estimates for Periodic Interpolation on Full and Sparse Grids*, in Curves and Surfaces with Applications in CAGD, A. Le Méhauté, C. Rabut, and L. L. Schumaker, eds., Vanderbilt University Press, Nashville, 1997, to appear.

[11] J. Prestin and E. Quak, *Trigonometric interpolation and wavelet decomposition*, Numer. Algorithms, 9 (1995), pp. 293–318.

[12] J. Prestin and K. Selig, *Interpolatory and orthonormal trigonometric wavelets*, Signal and Image Representation in Combined Spaces, in J. Zeevi and R. Coifman, eds., Academic Press, to appear.

[13] I. J. Schoenberg, *Cardinal interpolation and spline functions*, J. Approx. Theory, 2 (1969), pp. 167–206.

[14] F. Sprengel, *Multivariate periodic interpolating wavelets*, in Approximation Theory, Wavelets, and Applications, S. P. Singh, ed., Kluwer Academic Publ., Dordrecht, 1995, pp. 473–483.

[15] F. Sprengel, *Periodic interpolation and wavelets on sparse grids*, Preprint 96/8, Fachbereich Mathematik, Univ. Rostock.

[16] C. Zenger, *Sparse grids*, in Notes on Numerical Fluid Mechanics, volume 31, W. Hackbusch, ed., Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar, Vieweg, Braunschweig, 1991, pp. 297–301.