SIAM J. SCI. COMPUT.
Vol. 32, No. 6, pp. 3228–3250

EFFICIENT SPECTRAL SPARSE GRID METHODS AND APPLICATIONS TO HIGH-DIMENSIONAL ELLIPTIC PROBLEMS*

JIE SHEN† AND HAIJUN YU‡

Abstract. We develop in this paper some efficient algorithms which are essential to implementations of spectral methods on the sparse grid by Smolyak's construction based on a nested quadrature. More precisely, we develop a fast algorithm for the discrete transform between the values at the sparse grid and the coefficients of expansion in a hierarchical basis; and by using the aforementioned fast transform, we construct two very efficient sparse spectral-Galerkin methods for a model elliptic equation. In particular, the Chebyshev–Legendre–Galerkin method leads to a sparse matrix with a much lower number of nonzero elements than that of low-order sparse grid methods based on finite elements or wavelets, and can be efficiently solved by a suitable sparse solver. Ample numerical results are presented to demonstrate the efficiency and accuracy of our algorithms.

Key words. sparse grid, spectral method, high-dimensional problem, Chebyshev–Gauss–Lobatto quadrature

AMS subject classifications. 65N35, 65N22, 65F05, 35J05

DOI. 10.1137/100787842

1. Introduction.

Many scientific and engineering applications require solving high-dimensional partial differential equations (PDEs). Current spectral methods for solving PDEs in multidimensions are usually based on tensor product formulation which quickly becomes nonfeasible for high-dimension ($d > 3$) problems due to the so-called curse of dimensionality as the degree of freedom grows like N^d if N points are used in each direction. Thus, one has to resort to some sort of sparse discretizations. A particularly efficient class consists of the so-called sparse grids based on Smolyak's algorithm (cf. [22]), which is based on a hierarchy of one-dimensional quadrature.

In recent years, sparse grids have become a major approximation tool for high-dimensional problems. It has been successfully used in many scientific and engineering applications (cf., for instance, [6]). Most of the work on sparse grids is based on basis functions in the physical space, e.g., piecewise linear multiscale bases (cf. [3, 5]), hierarchical Lagrangian polynomial bases (cf. [4]), or wavelets (cf. [6, 17]). There have also been attempts to use basis functions from hyperbolic cross in the frequency space (cf. [11, 12, 14]), but these are usually based on Fourier series and are thus restricted to periodic problems. Most recently, Shen and Wang [21] considered hyperbolic cross approximations based on Jacobi polynomials and studied their basic approximation properties as well as their applications to high-dimensional elliptic equations. However, the original hyperbolic cross approximation is not suitable for most practical computations as it lacks a representation in the physical space. We note that the hyperbolic cross approximation in frequency space can be closely related to Smolyak's sparse grid with a nested quadrature through an interpolation operator based on hierarchical basis functions. Therefore, the main purpose of this work is

*Received by the editors March 8, 2010; accepted for publication (in revised form) August 13, 2010; published electronically November 9, 2010. This work was partially supported by AFOSR grant FA9550-08-1-0416 and NFS grant DMS-0915066.

http://www.siam.org/journals/sisc/32-6/78784.html

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (shen7@purdue.edu).

‡Department of Mathematics, Purdue University, West Lafayette, IN 47907. Current address: LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing 100190, China (hyu@lsec.cc.ac.cn).

to develop efficient algorithms for implementing sparse spectral methods based on the hierarchical polynomials for various situations. In particular, we shall consider the sparse grid based on the nested Chebyshev–Gauss–Lobatto quadrature (cf., for instance, [2, 15]), develop a fast transform algorithm between the sparse grid and the coefficients of hierarchical expansion, and use it to construct efficient sparse spectral solvers for high-dimensional elliptic equations.

In order to better understand the crucial issues in solving high-dimensional PDEs, we consider a generic evolution PDE written in the form

$$u_t + Au + N(u) = 0, \quad x \in \Omega \in \mathbb{R}^d, \tag{1.1}$$

where A is a leading linear operator and N is a nonlinear operator. This class of equations includes, as examples, nonlinear Schrödinger equations, Fokker–Planck equations, and some equations in mathematical finance. Assuming that we use a linearly implicit scheme to discretize (1.1) in time, then at each time step, one needs to solve a d-dimensional linear equation

$$\alpha u + Au = f, \quad x \in \Omega, \tag{1.2}$$

with appropriate boundary conditions. Our task is to develop an efficient sparse spectral method for (1.2). Let Y_N be a finite-dimensional polynomial space, and let I_N be an interpolation operator from a sparse grid to Y_N, both to be specified. A weighted spectral-Galerkin method for (1.2) is to find $u_N \in X_N = Y_N \cap X$ such that

$$\alpha(u_N, v_N)_\omega + (Au_N, v_N)_\omega = (I_N f, v_N)_\omega \quad \forall v_N \in X_N, \tag{1.3}$$

where ω is a positive weight function, X is a functional space with the underlying boundary conditions, and $(\cdot,\cdot)_\omega$ represents the weighted L^2 inner product.

Then, two main issues need to be addressed:

(i) The first issue is to construct an efficient algorithm to compute expansion coefficients of $I_N f$ in Y_N from the values of f at the given sparse grid. A particularly interesting sparse grid Σ_N is given by Smolyak's algorithm applied to the *nested* Chebyshev–Gauss–Lobatto quadrature. One can then construct a set of hierarchical basis functions and define an interpolation operator based on the sparse grid onto the space Y_N spanned by these hierarchical basis functions. We note that Y_N is closely related to the hyperbolic cross space. Thanks to the fact that the Chebyshev polynomials (resp., Chebyshev–Gauss–Lobatto points) are the image of trigonometric polynomials (resp., equally spaced points) under the mapping $\theta = \cos^{-1} x$, the transforms between the expansion coefficients in these hierarchical basis functions and the function values at the Chebyshev sparse grid Σ_N can also be performed in a fast and stable manner using the fast fourier transform (FFT).

(ii) The second issue is to develop an efficient algorithm for (1.3). The structure of the matrix associated with (1.3) depends on the basis functions of X_N. However, since X_N is not constructed with a usual tensor-product procedure, one can no longer apply the usual technique (for the full grid) of matrix diagonalization/decomposition (cf. [13, 18]) to efficiently solve these linear systems. We shall propose several different methods for dealing with these linear systems and provide some guidelines on what to use in different situations.

We shall perform numerical experiments for several classes of functions with different smoothness and assess the relative performance of the proposed sparse spectral methods.

The rest of paper is organized as follows. In section 2, we consider the sparse grid by Smolyak's algorithm based on a nested quadrature, construct the corresponding hierarchical bases, and present a detailed fast algorithm for the discrete transform between the values at the sparse grid and the coefficients of the expansion in the hierarchical bases. In section 3, we consider two sparse spectral-Galerkin schemes for model elliptic equations, namely, the Chebyshev–Galerkin method and the Chebyshev–Legendre–Galerkin method; construct efficient algorithms for the matrix-vector product and for forming the system matrix in the latter method; and present several options for solving the resultant linear systems. In section 4, we present some illustrative numerical results which are aimed at assessing the efficiency and accuracy of the sparse spectral methods for different classes of exact solutions. We end with some concluding remarks in the final section.

2. Hierarchical basis, sparse grid, and discrete transform. We consider in this section the sparse grids constructed by Smolyak's algorithm. If one starts with a nested one-dimensional quadrature, then Smolyak's algorithm leads to a sparse grid with many nice properties. In particular, one can build corresponding hierarchical basis functions and construct fast transforms between the expansion in the hierarchical bases and the values at the sparse grid. The commonly used high-order, nested one-dimensional quadratures include Fourier [12], Chebyshev [2], and Gauss–Patterson [10].

For a nonnested one-dimensional quadrature, Smolyak's algorithm leads to a sparse grid with many more points than a sparse grid based on a nested quadrature. Furthermore, it is difficult to construct a standard interpolation scheme because of the mismatch between the number of interpolation points and the dimension of a suitable interpolating space. The commonly used nonnested one-dimensional high-order quadratures include Gauss–Legendre, Gauss–Laguerre, and Gauss–Hermite schemes.

We shall concentrate in this paper on the sparse grids based on nested one-dimensional quadratures.

2.1. One-dimensional hierarchical basis. Let $I = (-1, 1)$. Let $\mathcal{U}^i$ be a scheme ($\{\mathcal{U}^i\}$ is a sequence of functionals for quadrature or a sequence of operators for interpolation) which uses N_i grid points $\mathcal{X}^i$ in I,

$$\mathcal{X}^i = \{x_0^i, x_1^i, \ldots, x_{N_i-1}^i\}, \quad i = 1, 2, \ldots.$$

As a convention, we always set $\mathcal{X}^0 = \emptyset$ and $\mathcal{U}^0$ to be the zero functional/operator.

The grids $\{\mathcal{X}^i\}$ are called *nested grids* if $\mathcal{X}^1 \subset \mathcal{X}^2 \subset \cdots$. For nested grids, we can rearrange the grid points in such a way that

$$\mathcal{X} = \mathcal{X}^1 \cup (\mathcal{X}^2 \backslash \mathcal{X}^1) \cup (\mathcal{X}^3 \backslash \mathcal{X}^2) \cup \cdots = \{x_0, x_1, x_2, \ldots\}$$

with $\{x_j, j \in \mathcal{I}^i\} = \mathcal{X}^i$, where $\mathcal{I}^i = \{0, 1, \ldots, N_i - 1\}$.

Let $\omega(x) > 0$ $(x \in I)$ be a weight function, $V_1 \subset V_2 \subset \cdots \subset V_i \cdots$ be a sequence of finite-dimensional spaces in $L^2_\omega(I)$, and $\{\phi_k(x), k = 0, 1, \ldots\}$ be a set of basis functions of $L^2_\omega(I)$ with

$$V_i = \text{span}\{\phi_k, k \in \mathcal{I}^i\}.$$

Then, for $f \in C(\bar{I})$, we can determine a unique set of coefficients $\{b_k^i\}$ such that

$$f(x_j^i) = \sum_{k \in \mathcal{I}^i} b_k^i \phi_k(x_j^i) \quad \text{for any } j = 0, 1, \ldots, N_i - 1. \tag{2.1}$$

Note that fast transforms between $\{f(x_j^i), x_j^i \in \mathcal{X}^i\}$ and $\{b_k^i, k \in \mathcal{I}^i\}$ are available if the basis functions $\{\phi_k\}$ are Fourier series or Chebyshev polynomials.

For schemes with nested grids, if one can find a set of basis functions $\{\tilde{\phi}_k\}$, such that $V_i = \text{span}\{\tilde{\phi}_k, k \in \mathcal{I}^i\}$ and

$$\tilde{\phi}_k(x_j) = 0 \quad \text{for any } j \in \mathcal{I}^i,\ k \notin \mathcal{I}^i, \tag{2.2}$$

then $\{\tilde{\phi}_k\}$ is called a set of *hierarchical bases*.

An important property of the hierarchical bases is that the expansion coefficients $\{b_k^i\}$ do not depend on the level index i; i.e., we can write

$$f(x_j) = \sum_{k \in \mathcal{I}^i} b_k \tilde{\phi}_k(x_j) \quad \text{for any } j \in \mathcal{I}^i, \quad i = 1, 2, \ldots. \tag{2.3}$$

THEOREM 2.1. *Hierarchical bases always exist for nested schemes. Furthermore, a set of hierarchical bases is given by*

$$\tilde{\phi}_k(x) = \phi_k(x) + \sum_{l \in \mathcal{I}^i} c_{k,l} \phi_l(x) \quad \textit{for } k \in \tilde{\mathcal{I}}^{i+1}, \quad i = 0, 1, 2, \ldots, \tag{2.4}$$

where $\tilde{\mathcal{I}}^{i+1} = \mathcal{I}^{i+1} \backslash \mathcal{I}^i$, $\mathcal{I}^0 = \varnothing$, *and* $c_{k,l} = -\phi_k(x_j) A_{jl}$ *with* $A = (A_{jl})_{l,j \in \mathcal{I}^i}$ *being the inverse matrix of* $B = (\phi_l(x_j))_{l,j \in \mathcal{I}^i}$.

Proof. Evaluating (2.4) at $\mathcal{X}^i$, we derive by using (2.2) that

$$\sum_{l \in \mathcal{I}^i} c_{k,l} \phi_l(x_j) = -\phi_k(x_j) \quad \text{for } j \in \mathcal{I}^i. \tag{2.5}$$

Since $\{\phi_l(x),\ l \in \mathcal{I}^i\}$ are basis functions and $\{x_j,\ j \in \mathcal{I}^i\}$ are quadrature points, $(\phi_l(x_j))_{l,j \in \mathcal{I}^i}$ is a full rank matrix. We can then determine $\{c_{k,l}\}$ from (2.5) and the the hierarchical bases in (2.4). ∎

We now describe in some detail two practical hierarchical bases based on the Chebyshev–Gauss–Lobatto quadrature.

(CH1) Let $\mathcal{X}^i = \left\{x_j^i = \cos(\frac{j\pi}{2^i}),\ j = 0, \ldots, 2^i\right\}$ for $i \geqslant 1$ be the Chebyshev–Gauss–Lobatto grid on the level i with the number of grid points being $N_i = 2^i + 1$ $(i \geq 1)$; for $i = 0$, we set $N_0 = 0$. We rearrange the grid points into a hierarchical order $x_j = x^i_{\alpha^i(j)}$, where $\alpha^i(j)$ is the reorder vector for grid level $i > 0$. For example, one may take

$$\alpha^i(j) = \begin{cases} 0, & j = 0, \\ \frac{2^i}{2^l}\left[1 + 2(2^l - j)\right], & j > 0,\ l = \min\limits_{2^s \geq j} s. \end{cases}$$

The original spectral basis functions are Chebyshev polynomials $T_k(x) := \cos(k \arccos(x))$. The corresponding hierarchical bases are given by

$$\tilde{T}_k = T_k \quad \text{for } k \in \mathcal{I}^1, \qquad \tilde{T}_k = T_k - T_{2^i - k} \quad \text{for } k \in \tilde{\mathcal{I}}^i,\ i > 1. \tag{2.6}$$

(CH2) Here we consider the Chebyshev scheme for functions satisfying homogeneous Dirichlet boundary conditions. We set $\mathcal{X}^i = \left\{x_j^i = \cos(\frac{(j+1)\pi}{2^i}), j = 0, \ldots, 2^i - 2\right\}$ with $N_i = 2^i - 1$ for $i \geqslant 1$. A set of spectral bases with homogeneous Dirichlet boundary condition is given by

$$\hat{T}_{2k} = T_{2k+2} - T_0, \quad \hat{T}_{2k+1} = T_{2k+3} - T_1, \quad k = 0, 1, \ldots.$$

Then, a set of hierarchical bases is given by

(2.7)
$$\bar{T}_k = \hat{T}_k \quad \text{for } k \in \mathcal{I}^1, \quad \bar{T}_k = \hat{T}_k - \hat{T}_{2^i-2-(k+2)} = T_{k+2} - T_{2^i-(k+2)} \quad \text{for } k \in \tilde{\mathcal{I}}^i,\ i > 1.$$

These two schemes will be used in the sparse spectral algorithms for elliptic PDEs presented in the next section. In particular, the first scheme is used when calculating the interpolation $I_N f$ in (1.3); the second scheme is used in transforming the solution to the physical space after solving the resulting linear algebraic system of (1.3).

2.2. Multidimensional hierarchical bases and sparse grid. Given a one-dimensional scheme $\mathcal{U}^i$, the d-dimensional sparse grid by Smolyak's construction is

$$\mathcal{U}_d^q = \sum_{d \leqslant |\boldsymbol{i}|_1 \leqslant q} \Delta^{i_1} \otimes \Delta^{i_2} \otimes \cdots \otimes \Delta^{i_d}, \tag{2.8}$$

where $\boldsymbol{i} = (i_1, i_2, \ldots, i_d)$ are the multiple indices of grid levels with all indices starting from 1, and $|\boldsymbol{i}|_1 = i_1 + i_2 + \cdots + i_d$, $\Delta^i = \mathcal{U}^i - \mathcal{U}^{i-1}$, for $i = 1, 2, \ldots$.

It is clear that all the points in the sparse grid are in the set

$$\mathcal{X}_d^q = \bigcup_{d \leqslant |\boldsymbol{i}|_1 \leqslant q} \mathcal{X}^{i_1} \times \mathcal{X}^{i_2} \times \cdots \times \mathcal{X}^{i_d}.$$

It is shown (cf. [2]) that (2.8) is equivalent to

$$\mathcal{U}_d^q = \sum_{q-d < |\boldsymbol{i}|_1 \leqslant q} (-1)^{q-|\boldsymbol{i}|_1} \binom{d-1}{q-|\mathbf{i}|_1} \mathcal{U}^{i_1} \otimes \mathcal{U}^{i_2} \otimes \cdots \otimes \mathcal{U}^{i_d}, \tag{2.9}$$

where $\binom{n}{k}$ is the binomial coefficient of selecting k elements from an n-element set. In order to use (2.9) as a quadrature rule or interpolation operator, we need to compute $\mathcal{U}^{i_1} \otimes \mathcal{U}^{i_2} \otimes \cdots \otimes \mathcal{U}^{i_d}$ for every $q - d < |\boldsymbol{i}|_1 \leqslant q$. Therefore, the computational complexity of a direct evaluation of (2.9) is about

$$C \sum_{q-d < |\boldsymbol{i}|_1 \leqslant q} N_{i_1} N_{i_2} \cdots N_{i_d},$$

regardless of whether the one-dimensional grids are nested or not. However, for nested grids, we can use the properties of hierarchical bases to design more efficient algorithms as described below.

Given $f \in C(\bar{I}^d)$, for $d = 1$, we can write the interpolation operator $\mathcal{U}^i$ as

$$\mathcal{U}^i(f)(x) = \sum_{k \in \mathcal{I}^i} b_k^i \phi_k(x),$$

where b_k^i is determined by (2.1). However, by using the hierarchical bases, we have

$$\mathcal{U}^i(f)(x) = \sum_{k \in \mathcal{I}^i} b_k \tilde{\phi}_k(x), \quad \Delta^i(f)(x) = \sum_{k \in \tilde{\mathcal{I}}^i} b_k \tilde{\phi}_k(x).$$

Therefore, (2.8) becomes

$$\begin{aligned} \mathcal{U}_d^q(f)(\boldsymbol{x}) &= \sum_{d \leqslant |i|_1 \leqslant q} \sum_{\boldsymbol{k} \in \tilde{\mathcal{I}}^{i_1} \times \cdots \times \tilde{\mathcal{I}}^{i_d}} b_{k_1,k_2,\ldots,k_d} \tilde{\phi}_{k_1}(x_1) \tilde{\phi}_{k_2}(x_2) \cdots \tilde{\phi}_{k_d}(x_d) \\ &= \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x}), \end{aligned}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, $\tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x}) = \tilde{\phi}_{k_1}(x_1)\tilde{\phi}_{k_2}(x_2)\cdots\tilde{\phi}_{k_d}(x_d)$, and

$$\mathcal{I}_d^q = \bigcup_{d \le |\boldsymbol{i}|_1 \le q} \tilde{\mathcal{I}}^{i_1} \times \cdots \times \tilde{\mathcal{I}}^{i_d}. \tag{2.10}$$

The expansion coefficients $\{b_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}_d^q\}$ can be determined by

$$f(\boldsymbol{x}_{\boldsymbol{j}}) = \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{j}}) \quad \forall \boldsymbol{j} \in \mathcal{I}_d^q. \tag{2.11}$$

Hence, $\mathcal{U}_d^q$ defines an interpolation operator which maps the function values on the grid $\mathcal{X}_d^q$ onto the space

$$V_d^q = \operatorname{span}\{\tilde{\phi}_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}_d^q\}. \tag{2.12}$$

Figure 2.1 shows a sparse grid $\mathcal{X}_2^5$ based on the one-dimensional Chebyshev–Gauss–Lobatto quadrature and the corresponding index set $\mathcal{I}_2^5$ in the interpolation space.

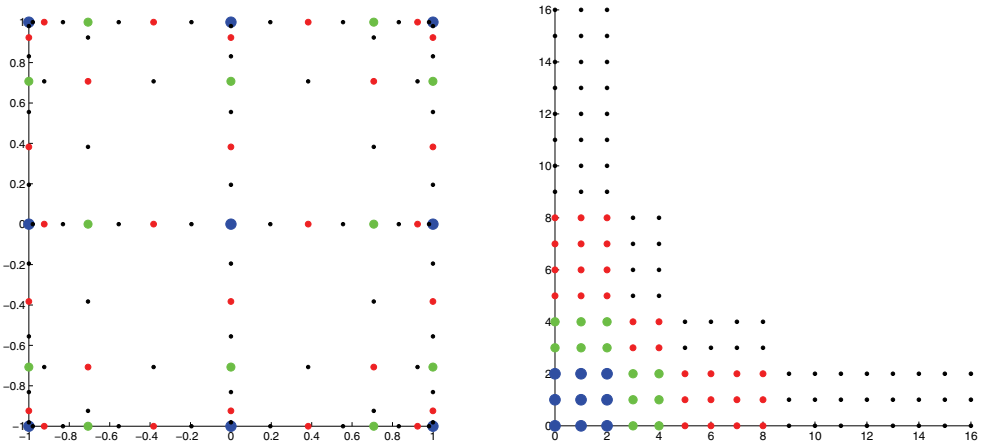

FIG. 2.1. *Left: the sparse grid $\mathcal{X}_2^5$ constructed from the Chebyshev–Gauss–Lobatto quadrature. Right: the corresponding index set $\mathcal{I}_2^5$ in the frequency space. The one-to-one mapping between the points in the left and right figures is given by $\mathcal{X}_2^5 = \{(x_{j_1}, x_{j_2}), (j_1, j_2) \in \mathcal{I}_2^5\}$.*

2.3. Discrete transform. We now describe in detail an efficient algorithm for performing the discrete transform between $\{f(\boldsymbol{x}_{\boldsymbol{j}})\}$ and $\{b_{\boldsymbol{k}}\}$ in (2.11).

We illustrate the procedure by using the two-dimensional sparse grid in Figure 2.1. Let us first consider the transform on the regular tensor product grid $\mathcal{X}^1 \times \mathcal{X}^1 = \{(x_{j_1}, x_{j_2}),\ j_1 \in \mathcal{I}^1,\ j_2 \in \mathcal{I}^1\}$ (which contains the nine biggest dots in Figure 2.1). We get

$$f(\boldsymbol{x}_{\boldsymbol{j}}) = \sum_{\boldsymbol{k} \in \mathcal{I}^1 \times \mathcal{I}^1} b_{\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{j}}) \quad \forall \boldsymbol{j} \in \mathcal{I}^1 \times \mathcal{I}^1.$$

The coefficients $\{b_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}^1 \times \mathcal{I}^1\}$ can be computed by two one-dimensional transforms on $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}^1 \times \mathcal{I}^1\}$, one along the first dimension, and another along the second dimension, i.e.,

$$b'_{k_1,j_2} = \sum_{j_1 \in \mathcal{I}^1} f(x_{j_1}, x_{j_2}) T_{j_1,k_1}, \quad b_{k_1,k_2} = \sum_{j_2 \in \mathcal{I}^1} b'_{k_1,j_2} T_{j_2,k_2}$$

or

$$b''_{j_1,k_2} = \sum_{j_2 \in \mathcal{I}^1} f(x_{j_1}, x_{j_2}) T_{j_2,k_2}, \quad b_{k_1,k_2} = \sum_{j_1 \in \mathcal{I}^1} b''_{j_1,k_2} T_{j_1,k_1},$$

where T is the inverse matrix of $(\tilde{\phi}_k(x_j))_{k,j}$.

Thanks to property (2.2), values of b'_{j_1,k_2} and b''_{k_1,j_2} do not depend on grid level. Therefore, we can compute all of the coefficients $\{b_{\boldsymbol{k}},\ \boldsymbol{k} \in \mathcal{I}_2^5\}$ (cf. Figure 2.1) as follows.

First we perform a one-dimensional transform on $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}^4 \times \mathcal{I}^1\}$ along the first dimension. We get $\{b'_{k_1,j_2}, (k_1, j_2) \in \mathcal{I}^4 \times \mathcal{I}^1\}$, which includes $\{b'_{k_1,j_2}, (k_1, j_2) \in \mathcal{I}^1 \times \mathcal{I}^1\}$. Then by performing similar transforms on $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}^3 \times \tilde{\mathcal{I}}^2\}$, $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}^2 \times \tilde{\mathcal{I}}^3\}$, and $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}^1 \times \tilde{\mathcal{I}}^4\}$, we get all of the values of b'_{k_1,j_2} with $(k_1, k_2) \in \mathcal{I}_2^5$. Next, applying one-dimensional transforms along the second dimension on $\{b'_{k_1,j_2}$, $(k_1, j_2) \in \mathcal{I}^1 \times \mathcal{I}^4\}$, $\{b'_{k_1,j_2}, (k_1, j_2) \in \tilde{\mathcal{I}}^2 \times \mathcal{I}^3\}$, $\{b'_{k_1,j_2}, (k_1, j_2) \in \tilde{\mathcal{I}}^3 \times \mathcal{I}^2\}$, and $\{b'_{k_1,j_2}, (k_1, j_2) \in \tilde{\mathcal{I}}^4 \times \mathcal{I}^1\}$, we obtain all of the values of $\{b_{k_1,k_2}, (k_1, k_2) \in \mathcal{I}_2^5\}$.

This procedure can be easily extended to d-dimensional cases. The exact procedures are described in *Algorithms* 1 and 2 below.

Algorithm 1. Fast forward transform on sparse grid.

Input: q, d, $\mathcal{X}_d^q$, and $\{\, f(\boldsymbol{x_j}),\, \boldsymbol{j} \in \mathcal{I}_d^q \,\}$.
Output: $\{\, b_{\boldsymbol{k}},\, \boldsymbol{k} \in \mathcal{I}_d^q \mid f(\boldsymbol{x_j}) = \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x_j}) \,\}$

```
b_j ← f(x_j) for all j ∈ I_d^q
for d' = 1 to d do
  for all i' ∈ {i' = (i_1, ..., i_{d'-1}, i_{d'+1}, ..., i_d) | |i'|_1 ⩽ q − 1} do
    one-dimensional transforms on {b_j, j ∈ Ĩ^{i_1} × ··· × Ĩ^{i_{d'-1}} × I^{q−|i'|_1} × Ĩ^{i_{d'+1}} ×
    ··· × Ĩ^{i_d}} along the d'th dimension.
  end for
end for
```

The total computational complexity of the fast forward and inverse transforms depends on the complexity of the one-dimensional transforms. We have the following results.

THEOREM 2.2. *Let n be the number of grid points in $\mathcal{X}_d^q$, and let C_d^q be the computational complexity of the fast transforms described in Algorithms* 1 *and* 2. *Suppose $N_i \sim 2^i$; then we have the following:*

(i) *If the cost of one-dimensional transforms is linear, i.e., $O(N)$, then $C_d^q = \mathcal{O}(dn)$.*

(ii) *If the cost of one-dimensional transforms is log-linear, i.e., $\mathcal{O}(N \log N)$, then $C_d^q = \mathcal{O}(nq)$.*

(iii) *If the cost of one-dimensional transforms is quadratic, i.e., $\mathcal{O}(N^2)$, then*

$$2^{2(q-d+1)} \lesssim C_d^q \lesssim 2^{2(q-d+1)} \binom{q-1}{d-1}.$$

Algorithm 2. Fast inverse transform on sparse grid.

Input: q, d, $\mathcal{X}_d^q$, and $\{b_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}_d^q\}$.

Output: $\{f(\boldsymbol{x_j}), \boldsymbol{j} \in \mathcal{I}_d^q \mid f(\boldsymbol{x_j}) = \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}\tilde{\phi}_{\boldsymbol{k}}(\boldsymbol{x_j})\}$

for $d' = 1$ to d **do**

for all $\boldsymbol{i}' \in \{\boldsymbol{i}' = (i_1, \ldots, i_{d'-1}, i_{d'+1}, \ldots, i_d) \mid |\boldsymbol{i}'|_1 \leqslant q-1\}$ **do**

one-dimensional inverse transforms on $\{b_{\boldsymbol{j}}, \boldsymbol{j} \in \tilde{\mathcal{I}}^{i_1} \times \cdots \times \tilde{\mathcal{I}}^{i_{d'-1}} \times \mathcal{I}^{q-|\boldsymbol{i}'|_1} \times \tilde{\mathcal{I}}^{i_{d'+1}} \times \cdots \times \tilde{\mathcal{I}}^{i_d}\}$ along the d'th dimension.

end for

end for

$f(\boldsymbol{x_j}) \leftarrow b_{\boldsymbol{j}}$ for all $\boldsymbol{j} \in \mathcal{I}_d^q$

Proof. It is easy to see that $C_d^q = \mathcal{O}(dn)$ if the cost of one-dimensional transforms is linear.

Next we consider the log-linear case. To fix the idea, we consider the case $N_i = 2^i - 1$ for $i \geq 1$. The case with $N_i = 2^i$ or $2^i + 1$ can be treated similarly. We first estimate the number of grid points n in $\mathcal{X}_d^q$:

$$
\begin{aligned}
n &= \mathrm{Card}\Big\{ \bigcup_{d\leqslant|\boldsymbol{i}|\leqslant q} (\mathcal{X}^{i_1}\backslash\mathcal{X}^{i_1-1}) \times \cdots \times (\mathcal{X}^{i_d}\backslash\mathcal{X}^{i_d-1}) \Big\} \\
&= \mathrm{Card}\Big\{ \bigcup_{d-1\leqslant i_1+\cdots+i_{d-1}\leqslant q-1} (\mathcal{X}^{i_1}\backslash\mathcal{X}^{i_1-1}) \times \cdots \times (\mathcal{X}^{i_{d-1}}\backslash\mathcal{X}^{i_{d-1}-1}) \times \mathcal{X}^{q-i_1-\cdots-i_{d-1}} \Big\} \\
&\sim 2^{q-d+1}\mathrm{Card}\{d-1 \leqslant i_1 + \cdots + i_{d-1} \leqslant q-1\} \\
&= 2^{q-d+1}\binom{q-1}{d-1}.
\end{aligned}
$$

The complexity of the fast transforms on the sparse grid $\mathcal{X}_d^q$ can be estimated as follows:

$$
\begin{aligned}
C_d^q &\lesssim d\sum_{p=1}^{q-d+1} p 2^{q-d+1}\mathrm{Card}\,\{d-1 \leqslant i_1 + \cdots + i_{d-1} = q-p\} \\
&= d2^{q-d+1}\sum_{p=1}^{q-d+1} p\binom{q-p-1}{d-2} \\
&= d2^{q-d+1}\left\{ q\sum_{p=1}^{q-d+1}\binom{q-p-1}{d-2} - \sum_{p=1}^{q-d+1}(q-p)\binom{q-p-1}{d-2} \right\} \\
&= d2^{q-d+1}\left\{ q\binom{q-1}{d-1} - \sum_{p=1}^{q-d+1}\frac{(q-p)(q-p-1)\cdots(q-p-d+2)}{(d-2)!} \right\} \\
&= d2^{q-d+1}\left\{ q\binom{q-1}{d-1} - \frac{1}{d}\frac{q(q-1)\cdots(q-d+1)}{(d-2)!} \right\} \\
&= 2^{q-d+1}q\binom{q-1}{d-1}.
\end{aligned}
$$

Hence, $C_d^q \sim \mathcal{O}(nq)$.

Similarly, we can show that if the cost of one-dimensional transforms is $\mathcal{O}(N^2)$, then we have

$$2^{2(q-d+1)} \lesssim C_d^q \lesssim 2^{2(q-d+1)} \binom{q-1}{d-1}. \qquad \square$$

Remark 2.1. Algorithms 1 and 2 can also be applied to compute

(2.13)
$$v_{k_1,k_2,\ldots,k_d} = \sum_{(k_1',k_2',\ldots,k_d')\in\mathcal{I}_d^q} u_{k_1',k_2',\ldots,k_d'} t^{(1)}_{k_1',k_1} t^{(2)}_{k_2',k_2} \cdots t^{(d)}_{k_d',k_d} \quad \forall\ (k_1,k_2,\ldots,k_d)\in\mathcal{I}_d^q,$$

where each $T^{(i)} = (t^{(i)}_{k_i',k_i})$ is either a block diagonal matrix, or (block) upper triangular matrix, or (block) lower triangular matrix. This nice property allows us to build a fast algorithm for the matrix-vector multiplication in the next section. More details are given in subsection 3.3.

3. Sparse spectral-Galerkin methods for elliptic equations. In this section, we construct efficient sparse spectral algorithms for the model problem

(3.1)
$$\begin{cases} \alpha u - \Delta u = f, & \boldsymbol{x}\in\Omega = (-1,1)^d, \\ u|_{\partial\Omega} = 0. \end{cases}$$

Given an integer $q \geq d$, let us denote by $\mathcal{I}_d^q$ and $\mathcal{J}_d^q$ the index sets associated with the schemes (CH2) and (CH1), respectively. More precisely,

$$\mathcal{I}_d^q = \left\{ (k_1,k_2,\ldots,k_d) : 0 \leqslant k_s < 2^{i_s} - 1,\ i_s \geqslant 0,\ \sum_{s=1}^d i_s = q \right\},$$

$$\mathcal{J}_d^q = \left\{ (k_1,k_2,\ldots,k_d) : 0 \leqslant k_s < 2^{i_s} + 1,\ i_s \geqslant 0,\ \sum_{s=1}^d i_s = q \right\}.$$

We set the sparse approximation space

(3.2)
$$V_d^q = \operatorname{span}\{\tilde{\phi}_{\boldsymbol{k}} : \boldsymbol{k} = (k_1,\ldots,k_d)\in\mathcal{I}_d^q\},$$

where $\{\tilde{\phi}_{\boldsymbol{k}}\}$ are the d-dimensional hierarchical basis functions based on (2.7). We also denote by $\mathcal{U}_d^q$ the interpolation operator (cf. (2.9)) associated with the scheme (CH1). Then, the sparse spectral-Galerkin method for (3.1) is as follows: find $u_d^q \in V_d^q$ such that

(3.3)
$$\alpha(u_d^q, v)_\omega - (\Delta u_d^q, v)_\omega = (\mathcal{U}_d^q f, v)_\omega \quad \forall v \in V_d^q,$$

where $(u,v)_\omega = \int_\Omega uv\omega\, d\boldsymbol{x}$ and $\omega(\boldsymbol{x}) = \Pi_{i=1}^d (1-x_i^2)^{-1/2}$ in the Chebyshev case and $\omega(\boldsymbol{x}) = 1$ in the Legendre case.

Given a set of basis functions $\{\phi_{\boldsymbol{k}}\}$ (not necessarily the hierarchical bases) for V_d^q, the sparse spectral-Galerkin method for (3.3) can be written as a linear system

(3.4)
$$(\alpha M + S)\boldsymbol{u} = \boldsymbol{f},$$

where M and S are, respectively, the mass and the stiffness matrix associated with $\{\phi_{\boldsymbol{k}}\}$, and $\boldsymbol{f}$ is a vector with component $\boldsymbol{f}_{\boldsymbol{k}} = (\mathcal{U}_d^q f, \phi_{\boldsymbol{k}})_\omega$.

While it appears to be natural to use the hierarchical bases $\{\tilde{\phi}_{\boldsymbol{k}}\}$, it is, however, not the most efficient choice as the number of nonzero elements in the mass and

stiffness matrices increase rapidly as d increases. On the other hand, the spectral-Galerkin basis functions $\psi_k(x) := T_k(x) - T_{k+2}(x)$ in the Chebyshev case and $\varphi_k(x) := L_k(x) - L_{k+2}(x)$ in the Legendre case lead to very simple mass and stiffness matrices and enjoy many other nice properties (cf. [18, 19]). Therefore, we shall use these as basis functions for V_d^q.

Notice that we still use the hierarchical bases (2.6) to obtain $\mathcal{U}_d^q f$, the expansion in the hierarchical bases, from the values of f at the sparse grid. Therefore, in order to compute the right-hand side vector $\boldsymbol{f}$, we first transform $\mathcal{U}_d^q f$ into an expansion based on standard Chebyshev or Legendre polynomials; then we use the orthogonality of Chebyshev or Legendre polynomials to compute $\boldsymbol{f}_{\boldsymbol{k}} = (\mathcal{U}_d^q f, \phi_{\boldsymbol{k}})_\omega$.

We shall show below, for both the Chebyshev and Legendre weight functions $\boldsymbol{\omega}$, that the approximate solution u_d^q for (3.3) can be efficiently obtained.

3.1. Sparse Chebyshev–Galerkin method. We consider now the scheme (3.3) with the Chebyshev weight.

We start with the one-dimensional case. Setting $\psi_k(x) = T_k(x) - T_{k+2}(x)$ as mentioned above, we have $V_1^q = \operatorname{span}\{\psi_k : k = 0, 1, \ldots, 2^q - 2\}$. Denote $a_{kj} = -(\psi_j''(x), \psi_k(x))_\omega$, $b_{kj} = (\psi_j(x), \psi_k(x))_\omega$; then the one-dimensional mass matrix and the stiff matrix are given by $B = (b_{kj})_{0 \leqslant k,j \leqslant 2^q-2}$, $A = (a_{kj})_{0 \leqslant k,j \leqslant 2^q-2}$. Writing $u_1^q = \Sigma_{n=0}^{2^q-2} \hat{u}_n \psi_n(x)$ and taking $v = \psi_k$ $(k = 0, 1, \ldots, 2^q - 2)$ in (3.3), we find

$$(\alpha B + A)\boldsymbol{u} = \boldsymbol{f}, \tag{3.5}$$

where $\boldsymbol{u} = (\hat{u}_0, \ldots, \hat{u}_{2^q-2})^T$, $\boldsymbol{f} = (f_0, \ldots, f_{2^q-2})^T$ with $f_k = (\mathcal{U}_1^q f, \psi_k(x))_\omega$. It can be easily shown that $b_{kj} = 0$ for $|k - j| > 2$ and $a_{kj} = 0$ for $j > k$. We note in particular that, even though the matrix A is not sparse, due to its special structure (cf. [19]), the matrix-vector product $(\alpha B + A)\boldsymbol{u}$ can be computed in linear complexity, i.e., in $\mathcal{O}(2^q)$ operations. This property plays a crucial role below.

Now we consider the multidimensional cases. Let us denote

$$\psi_{\boldsymbol{k}}(\boldsymbol{x}) = \psi_{k_1}(x_1)\psi_{k_2}(x_2)\cdots\psi_{k_d}(x_d), \quad T_{\boldsymbol{k}}(\boldsymbol{x}) = T_{k_1}(x_1)T_{k_2}(x_2)\cdots T_{k_d}(x_d).$$

Then, we have $V_d^q = \operatorname{span}\{\psi_{\boldsymbol{k}} : \boldsymbol{k} = (k_1, \ldots, k_d) \in \mathcal{I}_d^q\}$.

We set

$$\begin{aligned} &u_d^q = \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} \hat{u}_{\boldsymbol{k}} \psi_{\boldsymbol{k}}(\boldsymbol{x}), \quad \boldsymbol{u} = (\hat{u}_{\boldsymbol{k}})_{\boldsymbol{k} \in \mathcal{I}_d^q}; \\ &f_{\boldsymbol{k}} = (\mathcal{U}_q^d f, \psi_{\boldsymbol{k}})_\omega, \quad \boldsymbol{f} = (f_{\boldsymbol{k}})_{\boldsymbol{k} \in \mathcal{I}_d^q}^T; \\ &M = ((\psi_{\boldsymbol{j}}, \psi_{\boldsymbol{k}})_\omega)_{\boldsymbol{k},\boldsymbol{j} \in \mathcal{I}_d^q}, \quad S = (-(\Delta\psi_{\boldsymbol{j}}, \psi_{\boldsymbol{k}})_\omega)_{\boldsymbol{k},\boldsymbol{j} \in \mathcal{I}_d^q}. \end{aligned} \tag{3.6}$$

Note that the exact forms of $M, S, \boldsymbol{u}, \boldsymbol{f}$ depend on how the index set $\mathcal{I}_d^q$ is arranged into a row vector.

With the above notation, the sparse Chebyshev–Galerkin method (3.3) also reduces to the linear system (3.4), where the vector $\boldsymbol{f}$ on the right-hand side can be evaluated with the fast one-dimensional Chebyshev transform as follows:

1. Use Algorithm 1 to transform the function values of $f(\boldsymbol{x})$ at Chebyshev–Lobatto sparse grid points to the expansion coefficients in hierarchical bases (2.6), namely,

$$\mathcal{U}_q^d f = \sum_{\boldsymbol{k} \in \mathcal{J}_d^q} \tilde{f}_{\boldsymbol{k}} \tilde{T}_{\boldsymbol{k}}(\boldsymbol{x}),$$

where $\tilde{T}_{\boldsymbol{k}}(\boldsymbol{x})$ is the multidimensional hierarchical basis function based on (2.6).

2. Rewrite the expansion in standard Chebyshev polynomials, namely,

$$\mathcal{U}_q^d f = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \tilde{f}_{\boldsymbol{k}} \tilde{T}_{\boldsymbol{k}}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \hat{f}_{\boldsymbol{k}} T_{\boldsymbol{k}}(\boldsymbol{x}).$$

3. Compute $\boldsymbol{f}_{\boldsymbol{k}} = (\mathcal{U}_q^d f, \psi_{\boldsymbol{k}})_\omega = \sum_{\boldsymbol{j}\in\mathcal{J}_d^q} \hat{f}_{\boldsymbol{j}} (T_{\boldsymbol{j}}, \psi_{\boldsymbol{k}})_\omega$.

Note that each of the above three steps is essentially equivalent to a transform with a triangular transform matrix. According to Remark 2.1, the cost of the last two steps is proportional to the number of grid points, so the cost of evaluating $\boldsymbol{f}$ is essentially the cost of one fast Chebyshev transform using Algorithm 1.

Using the notation in one dimension, the entries of the mass and stiffness matrices can be computed as follows:

$$(\psi_{\boldsymbol{j}}, \psi_{\boldsymbol{k}})_\omega = (\psi_{j_1}, \psi_{k_1})_\omega (\psi_{j_2}, \psi_{k_2})_\omega \cdots (\psi_{j_d}, \psi_{k_d})_\omega = b_{k_1 j_1} b_{k_2 j_2} \cdots b_{k_d j_d}$$

and

$$-(\Delta\psi_{\boldsymbol{j}}, \psi_{\boldsymbol{k}})_\omega = \sum_{s=1}^{d} b_{k_1 j_1} \cdots b_{k_{s-1} j_{s-1}} a_{k_s j_s} b_{k_{s+1} j_{s+1}} \cdots b_{k_d j_d}.$$

The main advantage of the Chebyshev method is that the FFT can be used. However, this leads to a nonsymmetric, nonsparse stiffness matrix which is more difficult to deal with by an iterative method.

Unlike in the regular spectral method where the linear system can be efficiently solved by the matrix decomposition method (cf., for instance, [13, 19]), the matrix decomposition method cannot be applied to the system (3.4) since the space V_d^q is not separable. On the other hand, since the one-dimensional stiffness matrix A is not sparse, it is prohibitively expensive, in terms of both CPU time and storage, to build and store the stiffness and mass matrices in the high-dimension case. Therefore, we shall consider an iterative approach for which we need only provide an efficient matrix-vector multiplication, i.e., given $\boldsymbol{u}$, compute $\boldsymbol{r} = (\alpha M + S)\boldsymbol{u}$. A fast algorithm for doing this matrix-vector multiplication is described in subsection 3.3.

3.2. The sparse Chebyshev–Legendre–Galerkin method. The Chebyshev–Galerkin method above leads to a nonsymmetric, nonsparse system which can be solved only by an iterative method. However, due to the lack of optimal preconditioners, this procedure may become very slow as the problem size increases. We shall now describe a Chebyshev–Legendre method (cf. [9, 20]) which combines the advantages of Chebyshev and Legendre methods. More precisely, we consider the scheme (3.3) with $\omega = 1$.

Let $\mathcal{U}_d^q$ be the sparse interpolation operator as in the above, and let

$$\varphi_k(x) = L_k(x) - L_{k+2}(x), \quad \varphi_{\boldsymbol{k}}(\boldsymbol{x}) = \varphi_{k_1}(x_1)\varphi_{k_2}(x_2)\cdots\varphi_{k_d}(x_d).$$

Then we have $V_d^q = \mathrm{span}\{\varphi_{\boldsymbol{k}} : \boldsymbol{k} \in \mathcal{I}_d^q\}$.

Using the same notions as in the Chebyshev–Galerkin case, we have (cf. [18]) $a_{kj} = -(\varphi_j''(x), \varphi_k(x)) = (4k+6)\delta_{kj}$ and $b_{kj} = (\varphi_j(x), \varphi_k(x)) = 0$ for $k \neq j, j \pm 2$. Let us denote $M = ((\varphi_{\boldsymbol{j}}, \varphi_{\boldsymbol{k}}))_{\boldsymbol{k},\boldsymbol{j}\in\mathcal{I}_d^q}$, $S = (-(\Delta\varphi_{\boldsymbol{j}}, \varphi_{\boldsymbol{k}}))_{\boldsymbol{k},\boldsymbol{j}\in\mathcal{I}_d^q}$, $\boldsymbol{u} = (\hat{u}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_d^q}^T$, $\boldsymbol{f} = (f_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_d^q}^T$ with $f_{\boldsymbol{k}} = (\mathcal{U}_d^q f, \varphi_{\boldsymbol{k}})$. Equation (3.3) with $\omega = 1$ reduces again to the linear system (3.4). However, thanks to the nice structures of the one-dimensional stiffness and mass matrices, the number of nonzeros of the system matrix in (3.4) in this case

is proportional to number of unknowns for fixed d, making it feasible to use a suitable sparse solver. The price we pay for the sparsity of the stiffness matrix is an extra step in evaluating $\boldsymbol{f}$. More precisely, we now compute $\boldsymbol{f}$ as follows:

1. Use Algorithm 1 to compute

$$\mathcal{U}_q^d f = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \tilde{f}_{\boldsymbol{k}} \tilde{T}_{\boldsymbol{k}}(\boldsymbol{x}),$$

where $\tilde{T}_{\boldsymbol{k}}(\boldsymbol{x})$ is the multidimensional hierarchical basis function based on (2.6).

2. Rewrite the expansion in standard Chebyshev polynomials, namely,

$$\mathcal{U}_q^d f = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \tilde{f}_{\boldsymbol{k}} \tilde{T}_{\boldsymbol{k}}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \hat{f}_{\boldsymbol{k}} T_{\boldsymbol{k}}(\boldsymbol{x}).$$

3. Perform a Chebyshev to Legendre transform to obtain $\mathcal{U}_d^q f = \sum_{\boldsymbol{k}\in\mathcal{J}_d^q} \check{f}_{\boldsymbol{k}} L_{\boldsymbol{k}}(\boldsymbol{x})$; since the transform matrices between Chebyshev coefficients and Legendre coefficients are triangular matrices [20], we can apply Algorithm 1 directly.

4. Compute $\boldsymbol{f}_{\boldsymbol{k}} = (\mathcal{U}_q^d f, \varphi_{\boldsymbol{k}}) = \sum_{\boldsymbol{j}\in\mathcal{J}_d^q} \check{f}_{\boldsymbol{j}} (L_{\boldsymbol{j}}, \varphi_{\boldsymbol{k}})$.

Since the system (3.4) in this case is symmetric and positive definite, it can be solved by using the preconditioned conjugate gradient (PCG) iterative method which requires efficient system matrix-vector multiplication. On the other hand, since the system matrix is sparse, it can also be solved by using a sparse solver which requires explicitly building and storing the stiffness and mass matrices S and M. These issues are discussed in the next subsection.

3.3. Fast matrix-vector multiplication. In order to solve the linear systems (3.4) efficiently by using an iterative method, we need to construct an efficient algorithm to perform the matrix-vector product. We describe below a fast algorithm for the matrix-vector multiplication using the hierarchical structure in $\mathcal{I}_d^q$.

The algorithm is similar to that in [1] for a linear finite element sparse grid solver. More precisely, we write

$$\boldsymbol{r} = (\alpha M + S)\boldsymbol{u} = \alpha M\boldsymbol{u} + S_1\boldsymbol{u} + S_2\boldsymbol{u} + \cdots + S_d\boldsymbol{u}. \tag{3.7}$$

Each term of the right-hand side is a special case of the general form

$$v_{k_1,k_2,\ldots,k_d} = \sum_{(k_1',k_2',\ldots,k_d')\in\mathcal{I}_d^q} u_{k_1',k_2',\ldots,k_d'} t^{(1)}_{k_1',k_1} t^{(2)}_{k_2',k_2} \cdots t^{(d)}_{k_d',k_d} \quad \forall\ (k_1,k_2,\ldots,k_d)\in\mathcal{I}_d^q. \tag{3.8}$$

For the term $M\boldsymbol{u}$, we have $t^{(s)}_{k',k} = b_{k,k'}$ for all s; for the terms $S_\lambda\boldsymbol{u}$, we have $t^{(\lambda)}_{k',k} = a_{k,k'}$ and $t^{(s)}_{k',k} = b_{k,k'}$ for all $s \neq \lambda$.

If all of the one-dimensional matrices $T^{(s)} = (t^{(s)}_{k_s',k_s})$ are block diagonal matrices, we can use Algorithm 1 to compute the matrix-vector multiplications. However, when $T^{(s)}$ is not a block diagonal matrix,

$$v_{k_1,\ldots,k_s,k'_{s+1},\ldots,k'_d} = \sum_{\{k_s'|(k_1,\ldots,k_{s-1},k_s',\ldots,k_d')\in\mathcal{I}_d^q\}} u_{k_1,\ldots,k_{s-1},k_s',\ldots,k_d'} t^{(s)}_{k_s',k_s} \tag{3.9}$$

may have nonzero values with indices $(k_1,\ldots,k_s,k'_{s+1},\ldots,k'_d) \notin \mathcal{I}_d^q$. But if $T^{(s)}$ is a block lower triangular matrix, then all indices of the nonzero values of v are

in the index set of the sparse grid. On the other hand, if $T^{(s)}$ is a block upper triangular matrix, then no values of u with index outside of $\mathcal{I}_d^q$ will be used to calculate $\{v_{k_1,\ldots,k_s,k'_{s+1},\ldots,k'_d}, (k_1,\ldots,k_s,k'_{s+1},\ldots,k'_d) \in \mathcal{I}_d^q\}$. Based on these observations, one can use an upper-lower triangular splitting to calculate (3.8). Namely,

$$v_{k_1,\ldots,k_d} = \sum_{(k'_1,\ldots,k'_d)\in\mathcal{I}_d^q} u_{k'_1,\ldots,k'_d} \left(t^{(1L)}_{k'_1,k_1} + t^{(1U)}_{k'_1,k_1}\right)\cdots\left(t^{(dL)}_{k'_d,k_d} + t^{(dU)}_{k'_d,k_d}\right) \quad \forall\ (k_1,k_2,\ldots,k_d) \in \mathcal{I}_d^q. \tag{3.10}$$

The above summation involves 2^d terms. Each term can be calculated by performing a sequence of unidirectional downward/upward transforms on u along each dimension. By unidirectional downward (resp., upward) transform, we mean the transform (3.9) where the one-dimensional transform matrix $T^{(s)}$ is a lower (resp., upper) block triangular matrix. To keep the intermediate values inside $\mathcal{I}_d^q$, all of the downward transforms should be performed before all of the upward transforms. There are in total $2^d d$ unidirectional transforms; each can be computed in $\mathcal{O}(n)$ operations. Since a lower triangular transform and an upper triangular transform together can be computed in $\mathcal{O}(n)$ operations, the total number of operations to calculate (3.8) is $\mathcal{O}(2^{(d-1)}dn)$. The total memory usage is $\mathcal{O}(3n)$. Note that if we use a recursive implementation of the splitting technique, (3.8) can be calculated in $\mathcal{O}((2^d-1)n)$ operations, using $\mathcal{O}(dn)$ memory.

By using the same idea as for the matrix-vector multiplication, we can efficiently build the stiffness and mass matrices in (3.4) in the Chebyshev–Legendre case. First, we perform an upper-lower block triangular splitting on every one-dimensional transform matrix; a general form of system matrix will become a sum of 2^d sub-matrices built from d one-dimensional upper block or lower block matrices. Each submatrix can be calculated by successively applying all the unidirectional downward transforms to an $n \times n$ identity matrix, followed by successively applying all the unidirectional upward transforms. The total number of nonzeros in the system matrix $\alpha M + S$ is about $\mathcal{O}(3^d n)$; this is much smaller than the sparse grid methods based on hierarchical finite elements or wavelets, in which the number of nonzeros is not proportional to n for fixed d. In Figure 3.1, we plot the nonzero patterns of several system matrices with mass matrix ($\alpha \neq 0$) and without mass matrix ($\alpha = 0$) in the Chebyshev–Legendre case.

3.4. Iterative and direct solvers. In this subsection we present several options for solving the linear systems from the Chebyshev–Galerkin and Chebyshev–Legendre–Galerkin methods.

The first option is to use an iterative method which does not need explicit formation of the system matrix. This is also the only feasible option for the Chebyshev–Galerkin method. Since the system matrix comes from the discretization of an elliptic equation, we can use a preconditioned BICGSTAB for (3.4) in the Chebyshev case and PCG for (3.4) in the Chebyshev–Legendre case. Due to the structure of the Chebyshev sparse grid, it is not an easy matter to construct optimal preconditioners. So we will consider here only the simple diagonal preconditioner. Thanks to the special basis functions, which are H^1-orthogonal in one dimension, that we use in the Chebyshev–Legendre–Galerkin method, the diagonal preconditioner, although not optimal, does a remarkable job of reducing condition numbers of the system matrix. In Table 3.1, we show the condition numbers of the system matrix of the Chebyshev–Legendre–Galerkin method with and without the diagonal preconditioner. It also

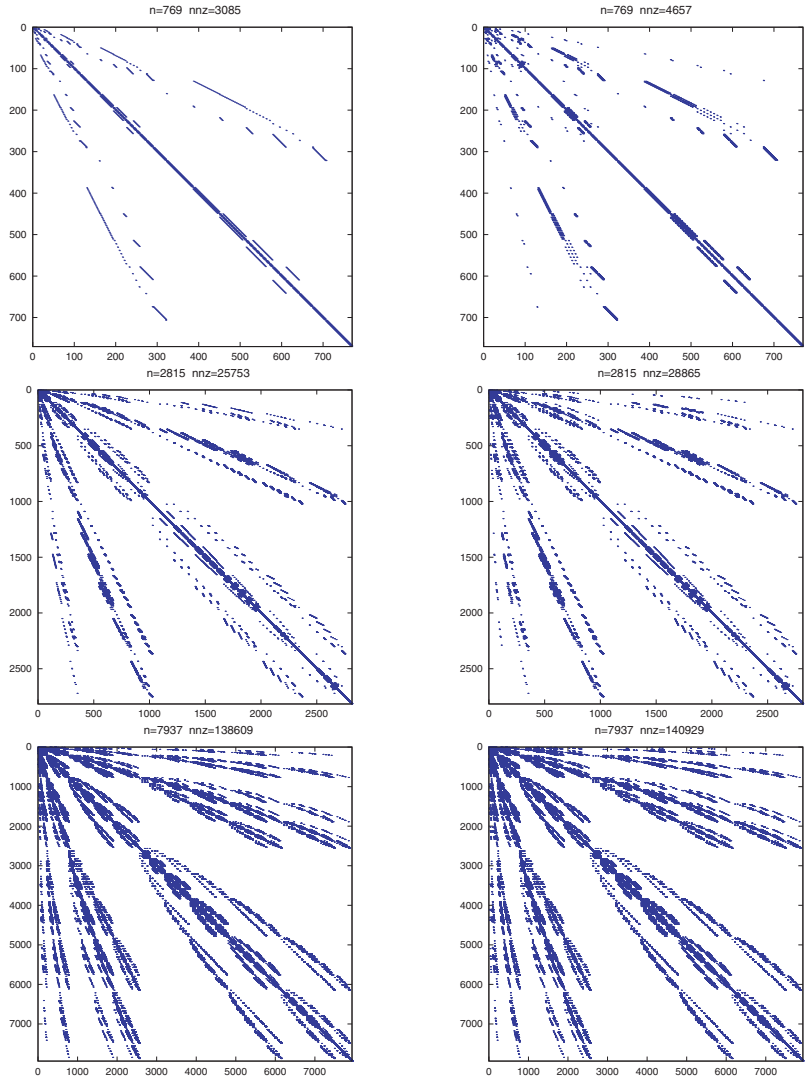

FIG. 3.1. *The structure of system matrices of the Chebyshev–Legendre–Galerkin sparse grid method. Left column: stiffness matrix. Right column: the sum of the stiffness matrix and the mass matrix. First row:* $d = 2$, $q = 8$*; second row:* $d = 3$, $q = 9$*; third row:* $d = 4$, $q = 10$.

shows that in higher-dimensional cases, the condition number is smaller than that in the lower-dimensional cases with the same number of unknowns.

Since the system matrix in (3.4) in the Chebyshev case is not symmetric, we shall not list the condition numbers. Instead, we list in Table 3.2 the iteration numbers and CPU times of the preconditioned BICGSTAB method for the Chebyshev–Galerkin system, and of the PCG method for the Chebyshev–Legendre–Galerkin system. The diagonal preconditioner is used in both cases. It can be seen that the convergence rates in both cases deteriorate only slightly as the number of unknowns increases, indicating that both schemes, while not with optimal computational complexity, are very efficient in solving the model elliptic equation in high dimensions. We also observe that the Chebyshev–Legendre–Galerkin method with PCG is in general more efficient than

TABLE 3.1

The condition number of diagonal-preconditioned and unpreconditioned system matrix. nnz_0: number of nonzeros of stiffness matrix; $cond_0$: the condition number of stiffness matrix; $pcond_0$: the condition number of stiffness matrix with diagonal preconditioner; nnz_1: number of nonzeros of system matrix ($\alpha = 1$); $cond_1$: the condition number of system matrix ($\alpha = 1$); $pcond_1$: the condition number of system matrix ($\alpha = 1$) with diagonal preconditioner.

d	q	n	nnz_0	$cond_0$	$pcond_0$	nnz_1	$cond_1$	$pcond_1$
2	3	5	9	1.394	1.230	9	1.330	1.298
	4	17	45	5.059	1.979	49	5.008	2.035
	5	49	157	13.312	2.742	193	13.248	2.819
	6	129	461	56.096	5.863	625	55.992	5.941
	7	321	1229	156.359	8.582	1777	156.295	8.670
	8	769	3085	746.708	20.594	4657	746.584	20.689
	9	1793	7437	2125.222	30.359	11569	2125.186	30.454
	10	4097	17421	10883.493	76.538	27697	10883.369	76.639
3	4	7	19	1.487	1.486	19	1.578	1.553
	5	31	133	5.285	2.774	133	5.256	2.855
	6	111	633	30.951	4.679	641	30.652	4.794
	7	351	2457	112.567	9.071	2561	112.335	9.226
	8	1023	8337	573.052	17.965	9017	572.201	18.192
	9	2815	25753	3266.231	39.210	28865	3262.578	39.552
4	5	9	29	1.825	1.767	29	1.938	1.836
	6	49	281	5.506	3.694	281	5.510	3.795
	7	209	1809	33.365	7.141	1809	33.153	7.297
	8	769	9009	214.031	14.285	9025	212.514	14.519
	9	2561	37601	997.260	30.342	37873	995.447	30.715

TABLE 3.2

Iteration numbers and CPU times of PCG for the Chebyshev–Legendre–Galerkin method (ChLeG) and preconditioned BICGSTAB for the Chebyshev–Galerkin method (ChG). The test solution is u_1 with $k = 2$, which is defined in section 4. The iteration tolerance is 10^{-9}.

d	q	n	itr#(ChG)	CPU (ChG)	itr#(ChLeG)	CPU (ChLeG)
2	9	1793	28	0.012	35	0.008
	10	4097	35	0.042	43	0.042
	11	9217	56	0.106	49	0.083
	12	20481	62	0.258	46	0.131
	13	45057	79	0.729	46	0.220
3	10	7423	33	0.136	40	0.085
	11	18943	52	0.453	59	0.290
	12	47103	76	1.694	87	1.090
	13	114687	122	7.651	111	3.830
	14	274431	171	29.533	122	12.070
4	11	23297	26	0.760	27	0.468
	12	65537	56	4.402	58	2.380
	13	178177	86	22.744	93	13.200
	14	471041	173	147.500	138	60.800
5	11	18943	3	0.163	3	0.125
	12	61183	10	1.634	10	0.927
	13	187903	31	21.431	31	11.100
	14	553983	78	207.430	74	98.500

the Chebyshev–Galerkin method with preconditioned BICGSTAB. However, an extra Chebyshev-to-Legendre transform is needed for the Chebyshev–Legendre–Galerkin method.

The second option is to use a robust sparse solver which requires the explicit formation of the system matrix. We shall consider the direct sparse solver UMFPack [7, 8] and an aggregation-based implementation of the AMG method [16].

TABLE 3.3

CPU time comparison of several linear sparse system solvers for (3.4) *in the Chebyshev–Legendre case: PCG, AMG, UMFPack. The iteration tolerance is* 10^{-14}. *(In this table, the biggest level number of one-dimensional grids is bounded by* 10*; i.e.,* $\mathcal{X}_d^q = \bigcup_{d \leqslant |\boldsymbol{i}|_1 \leqslant q,\ |i_s| \leq 10, s=1,\ldots,d.} \mathcal{X}^{i_1} \times \mathcal{X}^{i_2} \times \cdots \times \mathcal{X}^{i_d}$*).*

d	q	n	nnz	PCG	Matrix forming	AMG	UMF fact	UMF solv
2	12	18433	83981	0.403	0.784	0.069	0.081	0.007
	13	34817	165901	0.993	1.500	0.153	0.198	0.017
	14	63489	309261	2.300	2.740	0.341	0.487	0.035
	15	112641	555021	4.720	4.930	0.693	1.210	0.070
	16	194561	964621	13.700	8.740	1.300	3.270	0.133
3	11	18943	203689	0.699	1.800	0.155	0.350	0.025
	12	47103	536721	2.650	4.760	0.656	1.990	0.051
	13	111615	1361505	9.060	12.000	2.300	12.500	0.148
	14	252927	3305961	29.500	29.500	7.680	77.700	0.421
	15	551935	7659745	107.000	68.000	24.200	474.000	1.220
5	9	1471	25941	0.053	0.241	0.009	0.011	0.002
	10	5503	129193	0.242	1.220	0.045	0.107	0.009
	11	18943	559793	1.170	5.200	0.342	1.02	0.047
	12	61183	2182593	6.080	20.200	1.830	11.700	0.208
	13	187903	7835233	31.000	74.200	8.380	154.000	0.949

TABLE 3.4

The memory usage of AMG and UMFPack in MByte.

d	q	n	nnz	AMG memory	UMF memory
2	12	18433	83981	1.80	5.7
	13	34817	165901	3.49	10.7
	14	63489	309261	6.45	19.9
	15	112641	555021	11.51	41.1
	16	194561	964621	19.95	84.3
3	11	18943	203689	3.20	9.8
	12	47103	536721	8.30	34.3
	13	111615	1361505	20.69	120.0
	14	252927	3305961	49.82	401.0
	15	551935	7659745	115.82	1381.8
5	9	1471	25941	0.38	0.9
	10	5503	129193	1.84	3.8
	11	18943	559793	7.88	18.0
	12	61183	2182593	30.56	88.6
	13	187903	7835233	108.64	528.6

In Table 3.3, we present the numerical performance of several linear system solvers on the Chebyshev–Legendre–Galerkin method.

We note that the direct sparse solver UMFPack needs significant CPU time and memory for the initial factorization, so it is well suited only if one needs to solve the equation repeatedly, as in a time-dependent problem. For high-dimensional problems, the PCG method is a better choice as the factorization of the system matrix in UMFPack becomes too expensive in terms of both memory (see Table 3.4) and CPU time. The AMG method is a better choice only for low-dimensional problems ($d = 2, 3$) with a single right-hand side.

In fact, the CPU time and memory cost of UMFPack factorization are of order $\mathcal{O}(n^{1.5})$ and $\mathcal{O}(n \log n)$, respectively, which is not optimal in terms of the number of unknowns. Recently, Xia and Gu [24] proposed a structured multifrontal method for some discretized problems. Under certain conditions, the CPU time and memory

costs for factorization are of order $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$, respectively. Note that, after the factorization, the CPU time spent in solving the linear system is proportional to the memory usage. (See also [23] for an $\mathcal{O}(n)$ cost superfast multifrontal method for a special case.)

Dr. Xia has graciously allowed us to make some initial tests using the prerelease version of his SuperMF software. In Figure 3.2, we present a comparison of the CPU time and memory costs with the UMFPack and SuperMF. It is clear that the SuperMF outperforms the UMFPack significantly and that the CPU time and memory costs of SuperMF do appear to be of order $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$, making the overall Chebyshev–Legendre–Galerkin algorithm quasi-optimal.

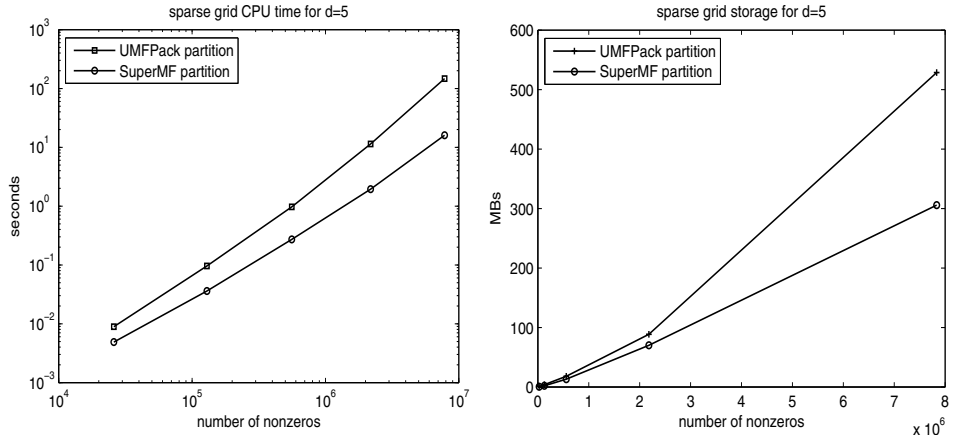

FIG. 3.2. *The comparison of CPU times and memory usage between the UMFPack and SuperMF methods at $d = 5$. The left figure plots the CPU factorization time versus the number of nonzeros. The right figure plots the the memory usage in the factorization process versus the number of nonzeros.*

4. Numerical results and discussion. In this section, we present some numerical results for solving (3.1). The goal is to examine the convergence behavior of the sparse spectral methods for different classes of functions. We consider the following four exact solutions of (3.1):

$$u_1(\boldsymbol{x}) = \prod_{i=1}^{d} \sin\left(k\pi \frac{x_i + 1}{2}\right), \qquad u_2(\boldsymbol{x}) = \sum_{i=1}^{d} \left[\phi_k(x_i) \prod_{j \neq i} \sin\left(\pi \frac{x_j + 1}{2}\right)\right],$$

$$u_3(\boldsymbol{x}) = \prod_{i=1}^{d} g_k(x_i), \qquad u_4(\boldsymbol{x}) = \prod_{i=1}^{d} \left(h_k(x_i) - \frac{1 + x_i}{2}\right),$$

where

$$\phi_k(x) = e^{\sin(k\pi \frac{x+1}{2})} - 1, \quad k = 1, 2, \ldots,$$
$$g_k(x) = (1 - x^2)(1 + x)\log(1 + x + 10^{-k}), \quad k = 1, 2, \ldots,$$
$$h_k(x) = \begin{cases} 0, & x \leqslant 0, \\ x^k, & x > 0, \end{cases} \quad k = 2, 3, \ldots.$$

The corresponding forcing functions f are

$$f_1(\boldsymbol{x}) = \alpha u_1(\boldsymbol{x}) + d\frac{k^2\pi^2}{4}u_1(\boldsymbol{x}),$$

$$f_2(\boldsymbol{x}) = \alpha u_2(\boldsymbol{x}) + \sum_{i=1}^{d}\left[\left(\frac{(d-1)\pi^2}{4}\phi_k(x_i) - \phi_k''(x_i)\right)\prod_{j\neq i}\sin\left(\pi\frac{x_j+1}{2}\right)\right],$$

$$f_3(\boldsymbol{x}) = \alpha u_3(\boldsymbol{x}) + \sum_{i=1}^{d}\left[-g_k''(x_i)\prod_{j\neq i}g_k(x_j)\right],$$

$$f_4(\boldsymbol{x}) = \alpha u_4(\boldsymbol{x}) + u_4(\boldsymbol{x})\sum_{i=1}^{d}\frac{k(k-1)h_{k-2}}{h_k(x_i) - (1+x_i)/2}.$$

We recall that the convergence rate of the sparse spectral methods depends on the regularity of solutions in the weighted Korobov spaces [21]. More precisely, let us define the hyperbolic cross space

$$H_N = \text{span}\{\Pi_{i=1}^{d}L_{k_i}(x_i) : \Pi_{i=1}^{d}\max(1,k_i) \le N\} \tag{4.1}$$

and the weighted Korobov space

$$K^m(I^d) = \{v : \partial_{\boldsymbol{x}}^{\boldsymbol{k}}u \in L^2_{\omega^{\boldsymbol{k},\boldsymbol{k}}}(I^d),\ 0 \le k_i \le m,\ i = 1,2,\ldots,d\}, \tag{4.2}$$

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_d)$ and $\omega^{\boldsymbol{k},\boldsymbol{k}} = \Pi_{i=1}^{d}(1-x_i^2)^{k_i}$. It is shown in [21] that

$$\inf_{v_N\in H_N}\|v - v_N\|_{L^2} \le N^{-m}|v|_{K^m}, \tag{4.3}$$

where $|v|_{K^m}^2 = \sum_{|\boldsymbol{k}|_l\infty=m}\|\partial_{\boldsymbol{x}}^{\boldsymbol{k}}u\|^2_{L^2_{\omega^{\boldsymbol{m}+\boldsymbol{k},\boldsymbol{m}+\boldsymbol{k}}}}$.

We note that, for fixed d and $N = N_q$, as $q \to \infty$, the hyperbolic cross space H_N is essentially equivalent to the space V_d^q in (2.12). So the error estimate (4.3) holds with H_N replaced by V_d^q.

Therefore, it is clear that for the sparse spectral method to perform well, the solution should have higher regularity in the Korobov space or, in other words, be well represented by the expansion coefficients in the space V_d^q or the hyperbolic cross space H_N. In Figure 4.1, we plot the amplitude of Chebyshev expansion coefficients for the four exact solutions in the two-dimensional case.

For the sake of comparison, we also recall the error estimates for the usual spectral methods (cf., for instance, [21]):

$$\inf_{v_N\in X_N}\|v - v_N\|_{L^2} \le N^{-m}|v|_{\tilde{H}^m_\omega}, \tag{4.4}$$

where

$$X_N = \text{span}\{\Pi_{i=1}^{d}L_{k_i}(x_i) : k_i \le N,\ i = 1,2,\ldots,d\}, \tag{4.5}$$

and the weighted Sobolev space

$$\tilde{H}^m_\omega(I^d) = \left\{v : \partial_{\boldsymbol{x}}^{\boldsymbol{k}}u \in L^2_{\omega^{\boldsymbol{k},\boldsymbol{k}}}(I^d),\ \sum_{i=1}^{d}k_i \le m\right\}, \tag{4.6}$$

with $|v|^2_{\tilde{H}^m_\omega} = \sum_{\sum_{i=1}^d k_i = m} \|\partial^{\boldsymbol{k}}_{\boldsymbol{x}} u\|^2_{L^2_{\omega^{\boldsymbol{m+k},\boldsymbol{m+k}}}}$. We note that $\mathrm{Card}(H_N) = O(N \log^{d-1} N)$, while $\mathrm{Card}(X_N) = O(N^d)$. Therefore, the sparse spectral methods will be much better than the full grid spectral method for functions with similar regularity index m in both weighted Sobolev and weighted Korobov spaces. Note that we always have $K^m(I^d) \subset \tilde{H}^m_\omega(I^d)$. However, for very smooth functions such as isotropic analytical functions, both the sparse spectral method and the full grid spectral method will converge exponentially fast (with respect to the number of unknowns), so there is not much advantage in using a sparse spectral method.

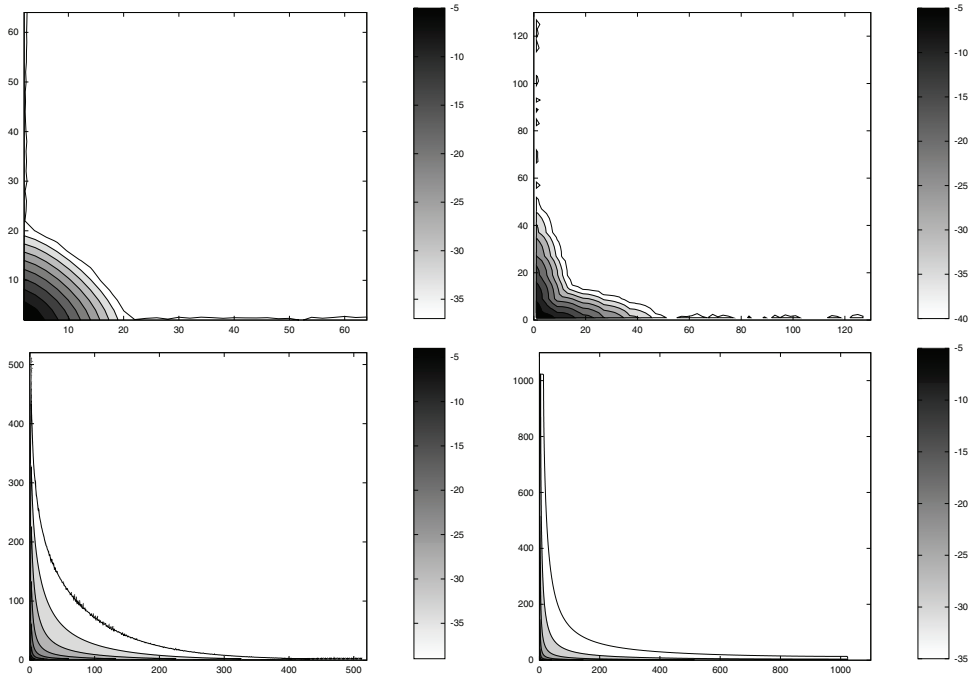

FIG. 4.1. *The two-dimensional Chebyshev spectral coefficients of the exact solutions* $u_1(\boldsymbol{x})$, $u_2(\boldsymbol{x}), u_3(\boldsymbol{x}), u_4(\boldsymbol{x})$ *with* $k = 2, 2, 3, 3$*, respectively. The logarithms of the absolute values of the coefficients are plotted.*

Case 1. With $f(\boldsymbol{x}) = f_1(\boldsymbol{x})$, the exact solution is a tensor product of one-dimensional analytic functions. In this case, both estimates (4.3) and (4.4) are valid for any $m > 0$. So the sparse spectral method does not hold much advantage over the usual full grid spectral method. This observation is also consistent with the spectrum shown in Figure 4.1.

In Figure 4.2, we present the L^2_ω error of the solutions obtained by the Chebyshev–Legendre–Galerkin method on full grid and sparse grid. The results indicate that the sparse grid method has similar convergence rate as the full grid method in terms of the number of unknowns, while the results with the sparse grid method are sightly better in higher dimensions.

Case 2. With $f(\boldsymbol{x}) = f_2(\boldsymbol{x})$. The exact solution is analytic but anisotropic. Figure 4.3 shows that the sparse grid method is much better than the full grid method.

Case 3. $f(\boldsymbol{x}) = f_3(\boldsymbol{x})$ with $k = 3$. The exact solution is a product of several one-dimensional functions with singularity near one boundary. Figure 4.4 shows the results of Chebyshev–Legendre–Galerkin method on full grids and sparse grids. We observe

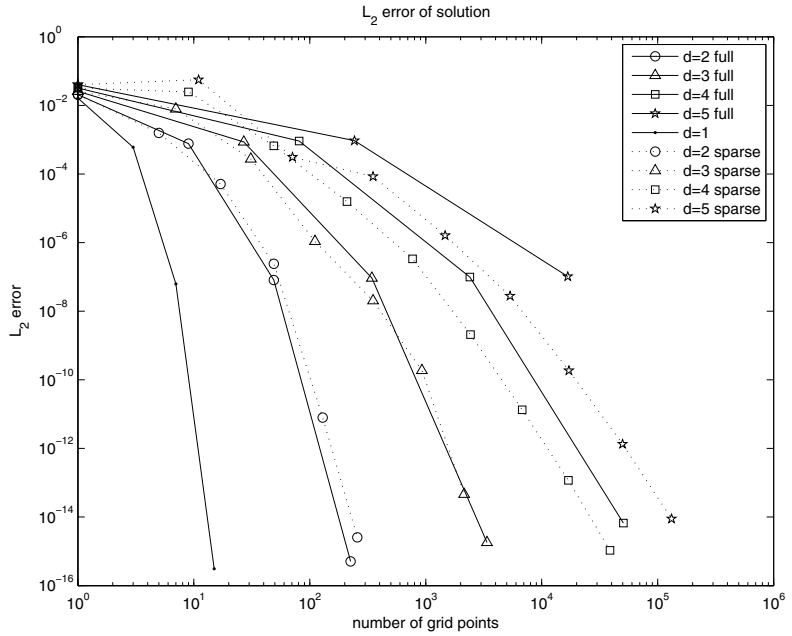

FIG. 4.2. *Convergence history of the Chebyshev–Legendre–Galerkin method on full grid (solid lines) and sparse grid (dotted lines) for Poisson's equation with right-hand side function* $f_1(\boldsymbol{x})$, $k = 1$.

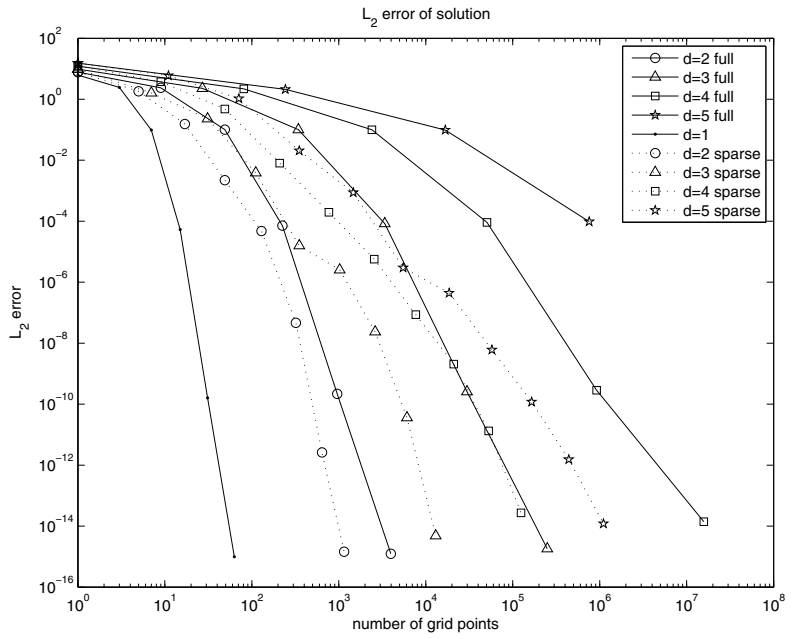

FIG. 4.3. *The numerical results of the Chebyshev–Legendre–Galerkin method on full grid and sparse grid for Poisson's equation with right-hand side function* $f_2(\boldsymbol{x})$, $k = 2$.

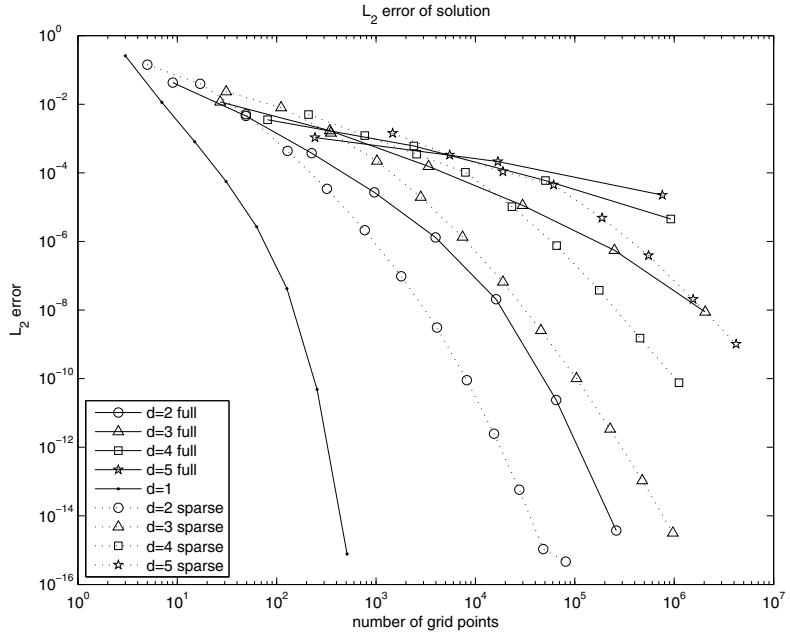

FIG. 4.4. *The Chebyshev–Legendre–Galerkin method on full grid and sparse grid for Poisson's equation with right-hand side function* $f_3(\boldsymbol{x})$, $k = 3$.

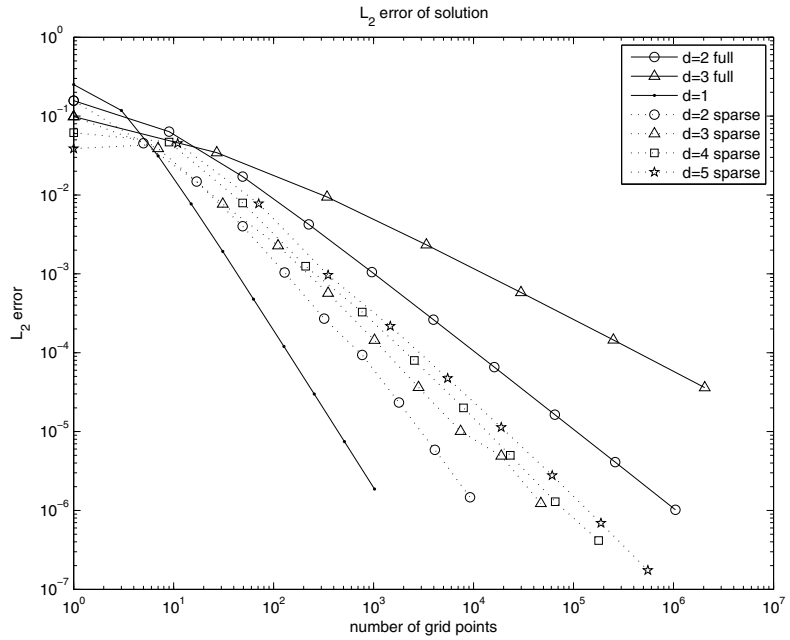

FIG. 4.5. *The Chebyshev–Legendre–Galerkin method on full grid (solid lines) and sparse grid (dotted lines) for Poisson's equation with right-hand side function* $f_4(\boldsymbol{x})$, $k = 3$.

that the sparse grid method is also better than the full grid method for this problem.

Case 4. $f(\boldsymbol{x}) = f_4(\boldsymbol{x})$ with $k = 3$. The exact solution is a product of one-dimensional functions in $H^{k-\frac{1}{2}+\epsilon}(I)$ (for any $\epsilon > 0$). It is easy to see that the solution belongs to $K^m(I^d)$ and $\tilde{H}^m_\omega(I^d)$ with the same index $m = k - \frac{1}{2} + \epsilon$. Therefore, this is an ideal case for the sparse spectral method. Figure 4.5 shows that the sparse grid method is much more efficient than full grid method for this problem.

5. Concluding remarks. We presented some efficient algorithms for the implementation of sparse spectral methods on the sparse grid by Smolyak's construction based on a nested quadrature, e.g., the Chebyshev–Gauss–Lobatto quadrature.

By using the properties of hierarchical bases, we developed a fast algorithm for the discrete transforms between the values at the sparse grid and the coefficients of expansion in the hierarchical bases.

We also proposed two efficient sparse spectral-Galerkin methods for a model elliptic equation in a high-dimensional cube: (i) The Chebyshev–Galerkin approach leads to a structured but nonsparse stiffness matrix. While it is too expensive to generate this matrix explicitly, the product of the system (stiffness + mass) matrix with a vector can be efficiently performed. Therefore, combined with a preconditioned BICGSTAB iterative procedure with a diagonal preconditioner, the sparse Chebyshev–Galerkin method, while not of optimal computational complexity, is easy to implement and very efficient, particularly in the high-dimensional case. (ii) The Chebyshev–Legendre–method leads to a sparse, symmetric and positive definite system that can be solved by using either a PCG iterative method or a sparse solver. It is remarkable that in this case the system matrix has a much lower number of nonzero elements than low-order sparse grid methods based on finite elements or wavelets. We note that in most cases, the Chebyshev–Legendre–Galerkin method is more efficient than the Chebyshev–Galerkin method, even though the former requires an extra Legendre-to-Chebyshev transform for each right-hand side function.

We investigated the relative efficiency of PCG iterative method and several sparse solvers for the Chebyshev–Legendre–Galerkin method. It was found that the PCG method with the diagonal preconditioner is the easiest to implement, uses the least memory, and is very efficient, even though it is not of optimal computational complexity. On the other hand, the prerelease version of SuperMF has a theoretical quasi-optimal memory and CPU complexity, which is confirmed by our numerical results, significantly outperforms both UMFPack and AMG, and outperforms PCG in most cases if the initial factorization cost is not taken into account. However, the factorization process in the current version of SuperMF is still not robust and can be quite expensive.

We also investigated the accuracy and efficiency of the sparse spectral methods for several typical classes of functions and showed that, except for a solution which is an isotropic analytic function, the sparse spectral methods are more accurate and efficient in terms of number of unknowns, than the full grid spectral method.

This paper is a first step toward developing efficient sparse spectral methods for high-dimensional PDEs. The algorithms developed in this paper are essential for any sparse spectral method in high dimension and enable us to solve some high-dimensional equations at a reasonable cost. While we considered a model elliptic equation with Dirichlet boundary conditions only as an example, the methods presented in this paper can be easily extended to the same equation with different boundary conditions and to more general equations with the Laplacian as the principal differential operator.

Acknowledgment. The authors would like to thank Dr. Jianlin Xia for his permission and assistance in using his prerelease version of the SuperMF package.

REFERENCES

[1] R. Balder and C. Zenger, *The solution of multidimensional real Helmholtz equations on sparse grids*, SIAM J. Sci. Comput., 17 (1996), pp. 631–646.
[2] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grids*, Adv. Comput. Math., 12 (2000), pp. 273–288.
[3] H.-J. Bungartz, *An adaptive Poisson solver using hierarchical bases and sparse grids*, in Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen, eds., North–Holland, Amsterdam, 1992, pp. 293–310.
[4] H.-J. Bungartz, *A multigrid algorithm for higher order finite elements on sparse grids*, Electron. Trans. Numer. Anal., 6 (1997), pp. 63–77.
[5] H.-J. Bungartz and M. Griebel, *A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives*, J. Complexity, 15 (1999), pp. 167–199.
[6] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.
[7] T. A. Davis, *Algorithm* 832*: UMFPACK, an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 196–199.
[8] T. A. Davis, *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 165–195.
[9] W. S. Don and D. Gottlieb, *The Chebyshev–Legendre method: Implementing Legendre methods on Chebyshev points*, SIAM J. Numer. Anal., 31 (1994), pp. 1519–1534.
[10] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), pp. 209–232.
[11] V. Gradinaru, *Fourier transform on sparse grids: Code design and the time dependent Schrödinger equation*, Computing, 80 (2007), pp. 1–22.
[12] M. Griebel and J. Hamaekers, *Sparse grids for the Schrödinger equation*, M2AN Math. Model. Numer. Anal., 41 (2007), pp. 215–247.
[13] D. B. Haidvogel and T. A. Zang, *The accurate solution of Poisson's equation by expansion in Chebyshev polynomials*, J. Comput. Phys., 30 (1979), pp. 167–180.
[14] K. Hallatschek, *Fouriertransformation auf dünnen Gittern mit hierarchischen Basen*, Numer. Math., 63 (1992), pp. 83–97.
[15] A. Klimke, *Efficient Construction of Hierarchical Polynomial Sparse Grid Interpolants Using the Fast Discrete Cosine Transform*, IANS preprint 2006/007, Universität Stuttgart, Stuttgart, Germany, 2006.
[16] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.
[17] C. Schwab and R. Stevenson, *Adaptive wavelet algorithms for elliptic PDE's on product domains*, Math. Comp., 77 (2008), pp. 71–92.
[18] J. Shen, *Efficient spectral-Galerkin method* I. *Direct solvers for second- and fourth-order equations using Legendre polynomials*, SIAM J. Sci. Comput., 15 (1994), pp. 1489–1505.
[19] J. Shen, *Efficient spectral-Galerkin method* II. *Direct solvers for second- and fourth-order equations using Chebyshev polynomials*, SIAM J. Sci. Comput., 16 (1995), pp. 74–87.
[20] J. Shen, *Efficient Chebyshev-Legendre Galerkin methods for elliptic problems*, in Proceedings of ICOSAHOM'95, A. V. Ilin and R. Scott, eds., Houston J. Math., 1996, pp. 233–240.
[21] J. Shen and L.-L. Wang, *Sparse spectral approximations of high-dimensional problems based on hyperbolic cross*, SIAM J. Numer. Anal., 48 (2010), pp. 1087–1109.
[22] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Soviet Math. Dokl., 4 (1963), pp. 240–243 (in English); Dokl. Akad. Nauk SSSR, 148 (1963), pp. 1042–1045 (in Russian).
[23] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1382–1411.
[24] J. Xia and M. Gu, *Robust Structured Multifrontal Factorization and Preconditioning for Discretized PDEs*, preprint, 2009; available online at http://www.math.purdue.edu/~xiaj/work/mfhssr.pdf.