SIAM J. SCI. COMPUT.
Vol. 34, No. 2, pp. A1141–A1164

EFFICIENT SPECTRAL SPARSE GRID METHODS AND APPLICATIONS TO HIGH-DIMENSIONAL ELLIPTIC EQUATIONS II. UNBOUNDED DOMAINS*

JIE SHEN† AND HAIJUN YU‡

Abstract. This is the second part in a series of papers on using spectral sparse grid methods for solving higher-dimensional PDEs. We extend the basic idea in the first part [J. Shen and H. Yu, *SIAM J. Sci. Comp.*, 32 (2010), pp. 3228–3250] for solving PDEs in bounded higher-dimensional domains to unbounded higher-dimensional domains and apply the new method to solve the electronic Schrödinger equation. By using modified mapped Chebyshev functions as basis functions, we construct mapped Chebyshev sparse grid methods which enjoy the following properties: (i) the mapped Chebyshev approach enables us to build sparse grids with Smolyak's algorithms based on nested, spectrally accurate quadratures and allows us to build fast transforms between the values at the sparse grid points and the corresponding expansion coefficients; (ii) the mapped Chebyshev basis functions lead to identity mass matrices and very sparse stiffness matrices for problems with constant coefficients and allow us to construct a matrix-vector product algorithm with quasi-optimal computational cost even for problems with variable coefficients; and (iii) the resultant linear systems for elliptic equations with constant or variable coefficients can be solved efficiently by using a suitable iterative scheme. Ample numerical results are presented to demonstrate the efficiency and accuracy of the proposed algorithms.

Key words. spectral method, sparse grid, higher-dimensional problems, unbounded domains, mapped Chebyshev functions, electronic Schrödinger equation

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

DOI. 10.1137/110834950

1. Introduction.

Many fundamental equations in mathematical physics involve higher-dimensional unbounded domains. These include the electronic Schrödinger equation in quantum mechanics, Boltzmann equations in kinetic theory, and the Black–Scholes equation in mathematical finance. While there is a significant body of work devoted to problems in high-dimensional bounded domains (cf. [4] and the references therein), little attention has been paid to developing dedicated numerical algorithms for high-dimensional unbounded domains. Often, a domain truncation method is used to reduce the underlying problem to a bounded domain (cf. [10]). While this approach can be effective in many situations where the solutions decay rapidly at infinity, it is prone to the so-called finite-size effect (see, for instance, [7]), which can be difficult to quantify. Therefore, it is important to develop dedicated numerical algorithms to treat unbounded domains directly. Much progress, mostly for one-dimensional problems, has been made in this direction by using the spectral methods (cf. [3, 17] and the references therein). Among the many approaches discussed in [3, 17], a particularly effective one is to employ a suitable transform which maps

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 23, 2011; accepted for publication (in revised form) January 3, 2012; published electronically April 19, 2012. This work is partially supported by AFOSR grants FA9550-08-1-0416 and FA9550-11-1-0328 and NFS grant DMS-0915066.

http://www.siam.org/journals/sisc/34-2/83495.html

†School of Mathematical Science, Xiamen University, Xiamen, China, and Department of Mathematics, Purdue University, West Lafayette, IN 47907 (shen7@purdue.edu).

‡Corresponding author. LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing, 100190, China (hyu@lsec.cc.ac.cn).
This author's work was partially supported by the ICMSEC Director Fund.

a bounded domain to an unbounded domain and then use the mapped orthogonal functions (see, for instance, [2, 13]) on the unbounded domain.

In a recent paper [18], we developed an efficient spectral sparse grid method for a class of partial differential equations (PDEs) in higher-dimensional bounded domains. The method is based on (i) a hyperbolic-cross type approximation by orthogonal polynomials and (ii) a sparse grid via Smolyak's construction [19] using the nested Chebyshev–Gauss–Lobatto quadrature, which is the only known nested quadrature with spectral accuracy. The main purposes of this paper are to extend the method developed in [18] for higher-dimensional bounded domains to higher-dimensional unbounded domains and to apply it to solve the electronic Schrödinger equation.

The obvious choices of orthogonal systems in unbounded domains are Hermite and Laguerre functions. However, the corresponding Gaussian-type quadrature points are not nested. Since the number of nodes in the Smolyak's sparse grid based on nonnested quadrature points grows much faster than that based on nested quadrature points, it appears natural to use a mapped Chebyshev approach, as in [2, 13], which will enable us to take advantage of some of the basic ingredients in the spectral sparse grid method developed in [18]. However, there are several significant challenges we have to overcome. How to choose suitable mappings and construct sparse grids such that they are tailored to different decay behaviors at infinity? How to efficiently build the system matrices and solve the discrete linear systems for elliptic equations? How to deal with more general elliptic equations with nonconstant coefficients and the discrete eigenvalue problem for the electronic Schrödinger equations?

We shall construct efficient mapped Chebyshev sparse grid (MCSG) methods for higher-dimensional unbounded domains in this paper and apply them to solve higher-dimensional elliptic equations and the electronic Schrödinger equation. In particular, our methods enjoy the following advantages: (i) using the mapped Chebyshev approach enables us to build sparse grids with Smolyak's algorithms based on nested, spectrally accurate quadratures and allows us to develop fast transforms between the values at the sparse grid and the corresponding expansion coefficients; and (ii) the mapped Chebyshev basis functions lead to, for problems with constant coefficients, identity mass matrices and stiffness matrices which are much sparser, particularly in the high-dimensional case, than the corresponding ones in bounded domains and allow us to construct a matrix-vector product algorithm with quasi-optimal computational cost for problems with variable coefficients.

As a comparison, we also propose a mapped Chebyshev–Hermite sparse grid (MCHSG) method which uses Hermite functions as basis functions but with sparse grids constructed by the mapped Chebyshev approach. Due to the lack of a fast transform between the values at the mapped Chebyshev sparse grid and the Hermite expansion coefficients, this method is more expensive than the MCSG method.

The rest of this paper is organized as follows. In section 2, we describe in detail the main ingredients of the MCSG and MCHSG methods and present ample numerical results and discussions. In section 3, we construct an MCSG method for solving the electronic Schrödinger equation. We conclude with a few remarks in the final section.

2. MCSG and MCHSG methods for elliptic equations in $\mathbb{R}^d$. We construct in this section spectral sparse grid methods for solving the elliptic equation

$$\begin{aligned} &\kappa(\boldsymbol{x})\, u(\boldsymbol{x}) - \nabla\,(\alpha(\boldsymbol{x})\nabla u(\boldsymbol{x})) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d, \\ &\lim_{|x|\to\infty} u(\mathbf{x}) = 0, \end{aligned} \tag{2.1}$$

where $\kappa(\boldsymbol{x})$, $\alpha(\boldsymbol{x}) > 0$. The weak formulation for the above equation is as follows: Find $u \in H^1(\mathbb{R}^d)$ such that

$$(\kappa(\boldsymbol{x})\, u, v) + (\alpha(\boldsymbol{x})\nabla u, \nabla v) = (f, v) \qquad \forall v \in H^1(\mathbb{R}^d), \tag{2.2}$$

where $(u, v) = \int_{\mathbb{R}^d} u\, v d\mathbf{x}$.

Given a sparse spectral approximation space $V_d^q \subset H^1(\mathbb{R}^d)$ (where $q \geq d$ is the level index) and an associated interpolation operator $\mathcal{U}_d^q$, the pseudospectral–Galerkin sparse grid method for (2.2) is as follows: Find $u \in V_d^q$ such that

$$(\mathcal{U}_d^q[\kappa(\boldsymbol{x})\, u], v) + \left(\mathcal{U}_d^{q+1}\left[\alpha(\boldsymbol{x})\nabla u\right], \nabla v\right) = (\mathcal{U}_d^q f, v) \quad \forall v \in V_d^q. \tag{2.3}$$

We will explain in section 2.6 why $\mathcal{U}_d^{q+1}$ is used for the stiff term instead of $\mathcal{U}_d^q$. In the constant coefficient case, i.e., $\kappa(\boldsymbol{x}) \equiv \kappa$ and $\alpha(\boldsymbol{x}) \equiv 1$, the above scheme simplifies to the following: Find $u \in V_d^q$ such that

$$\kappa(u, v) + (\nabla u, \nabla v) = (\mathcal{U}_d^q f, v) \quad \forall v \in V_d^q. \tag{2.4}$$

Adapting the general framework established in [18], an efficient pseudospectral–Galerkin sparse grid method for (2.2) consists of three essential ingredients:

- a sparse grid $\mathcal{X}_d^q$ in $\mathbb{R}^d$ with the index set $\mathcal{I}_d^q$ using Smolyak's algorithm from a one-dimensional (1-D) nested spectrally accurate quadrature, and the associated sparse approximation space V_d^q;
- an interpolating operator $\mathcal{U}_d^q : C(\mathbb{R}^d) \to V_d^q$ based on $\mathcal{X}_d^q$;
- a set of well-designed basis functions for V_d^q so that the corresponding linear system (2.3) can be efficiently solved.

To simplify the presentation, we shall first consider the constant coefficient case (2.4) and then discuss the general case (2.3) in the last subsection.

2.1. MCSGs in $\mathbb{R}^d$ and associated interpolation operators. We recall that Smolyak's algorithm applied to nonnested quadrature points will lead to sparse grids with many more grid points so they are not computationally efficient. Hence, we shall concentrate on sparse grids generated from nested quadrature points. Since the only spectrally accurate nested quadrature is the Chebyshev–Gauss–Lobatto quadrature, we shall restrict our attention to the sparse grid in $\mathbb{R}^d$ constructed from the *mapped* Chebyshev–Gauss–Lobatto quadrature.

It is shown in [18] that for every sparse grid constructed with nested quadrature points, one can construct corresponding hierarchical bases and fast transforms between the values at sparse grid points and expansion coefficients in hierarchical bases.

The Chebyshev–Gauss–Lobatto quadrature points are given by

$$\xi_j^N = \cos\left(\frac{j\pi}{N-1}\right), \quad j = 0, \ldots, N-1.$$

For odd number N, the grid points are nested.

Given a one-to-one map $x = x(\xi) : (-1, 1) \to \mathbb{R}$ and its inverse mapping $\xi = \xi(x) : \mathbb{R} \to (-1, 1)$, we consider the mapped Chebyshev functions

$$\hat{T}_k(x) = \frac{1}{\sqrt{c_k}} T_k(\xi(x))\mu(\xi(x)), \tag{2.5}$$

where $c_0 = \pi$ and $c_k = \pi/2$ for $k \geq 1$, and

$$\mu(\xi) = \sqrt{\omega(\xi)/x'(\xi)} \text{ with } \omega(\xi) = 1/\sqrt{1-\xi^2}. \tag{2.6}$$

By construction, we have

$$(\hat{T}_k, \hat{T}_j) = \int_{\mathbb{R}} \hat{T}_k(x)\,\hat{T}_j(x)dx = \frac{1}{\sqrt{c_k\,c_j}} \int_{-1}^{1} T_k(\xi)T_j(\xi)\omega(\xi)d\xi = \delta_{k,j}. \tag{2.7}$$

Hence, $\{\hat{T}_k\}$ forms an orthonormal system on $L^2(\mathbb{R})$.

Remark 2.1. The definition of $\hat{T}_k$ is different from the usual definition of mapped Chebyshev functions $R_k(x) = T_k(\xi(x))$ (cf. [2, 17]). The main advantage of using $\{\hat{T}_k\}$ is that they are orthonormal, which leads to, in the case of constant coefficients, the identity mass matrix and the sparse symmetric stiffness matrix. As we shall see below, the diagonal mass matrix is a very desirable property in high-dimensional problems. The disadvantage is that $\{\hat{T}_k\}$ are no longer "arbitrarily smooth" in the usual sense due to the factor $\mu(\xi(x))$. Therefore, fast convergences by the expansion in $\{\hat{T}_k\}$ can be obtained only for functions with fast decays at infinities.

Remark 2.2. The convergence behavior by the usual mapped Chebyshev functions, i.e., without the factor $\mu(\xi(x))$ in (2.5), has been studied by many authors; we refer to [3] (resp., [17]) for a qualitative (resp., quantitative) analysis. However, due to the factor $\mu(\xi(x))$ in (2.5), the general framework developed in [17] does not directly apply (see, however, [11, 12] for some related analysis). In this paper, we shall only provide some heuristic arguments along with ample numerical results for the error behaviors by these new mapped Chebyshev functions, and their rigorous numerical analysis in one dimension and in high dimensions will be carried out in [14].

The mapped Chebyshev–Gauss–Lobatto quadrature points $\{x_j^N = x(\xi_j^N),\ j = 0, 1, \ldots, N-1\}$ have a natural hierarchical structure. Denote by $\mathcal{X}^i = \{x_0^i, x_1^i, \ldots, x_{N_i-1}^i\}$ the Chebyshev–Gauss–Lobatto grids with $N_i = 2N_{i-1} - 1$ for $i \geqslant 2$, and $N_0 = 0$, $N_1 = 3$. Denote by $\bar{\mathcal{X}}^i$ the grid increment: $\mathcal{X}^i \backslash \mathcal{X}^{i-1}$. The first level grid $\mathcal{X}^1$ has three points $\{x(-1), x(1), x(0)\}$; the second level grid $\mathcal{X}^2$ is formed by adding two additional points $\{x(-\sqrt{2}/2), x(\sqrt{2}/2)\}$ to the first level grid.

Since the only difference between the usual Chebyshev transform in $\{T_k\}$ and the mapped Chebyshev transform in $\{\hat{T}_k\}$ is a mapping and an additional factor $\mu(x)$, it is clear that the interpolation of a function $f(x)$ on grid $\mathcal{X}^i$

$$\mathcal{U}^i f(x) = \sum_{k=0}^{N_i-1} b_k \hat{T}_k(x), \quad \text{with} \quad \sum_{k=0}^{N_i-1} b_k \hat{T}_k(x_j^i) = f(x_j^i), \quad j \in \mathcal{I}^i = \{0, \ldots, N_i - 1\},$$

can be determined by using the fast Chebyshev transform.

With the above 1-D hierarchical grid $\{\mathcal{X}^i\}$ in $\mathbb{R}$, we can construct the sparse grid in $\mathbb{R}^d$ using Smolyak's algorithm [19]:

$$\mathcal{X}_d^q := \bigcup_{|\boldsymbol{i}|_1 = i_1 + \cdots + i_d \leqslant q} \mathcal{X}^{i_1} \otimes \mathcal{X}^{i_2} \otimes \cdots \otimes \mathcal{X}^{i_d} = \sum_{|\boldsymbol{i}|_1 \leqslant q} \bar{\mathcal{X}}^{i_1} \otimes \bar{\mathcal{X}}^{i_2} \otimes \cdots \otimes \bar{\mathcal{X}}^{i_d}. \tag{2.8}$$

The interpolation operator $\mathcal{U}_d^q$ is defined by

$$\mathcal{U}_d^q f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} \hat{T}_{\boldsymbol{k}}(\boldsymbol{x}) \quad \text{with} \quad \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} \hat{T}_{\boldsymbol{k}}(\boldsymbol{x_j}) = f(\boldsymbol{x_j}) \quad \forall \boldsymbol{x_j} \in \mathcal{X}_d^q, \tag{2.9}$$

where $\boldsymbol{k}$ and $\boldsymbol{j}$ are multi-indices, $\boldsymbol{x_j} = (x_{j_1}, x_{j_2}, \ldots, x_{j_d})$ is the multidimensional coordinates, and $\mathcal{I}_d^q$ is the sparse grid index set given by

$$\mathcal{I}_d^q := \bigcup_{|\boldsymbol{i}| \leqslant q} \left(\mathcal{I}^{i_1} \backslash \mathcal{I}^{i_1 - 1}\right) \otimes \cdots \otimes \left(\mathcal{I}^{i_d} \backslash \mathcal{I}^{i_d - 1}\right). \tag{2.10}$$

Then, our MCSG approximation space is defined as

$$V_d^q := \text{span}\{\phi_{\boldsymbol{k}}(\boldsymbol{x}) : \boldsymbol{k} \in \mathcal{I}_d^q\} \quad \text{with} \quad \phi_{\boldsymbol{k}}(\boldsymbol{x}) = \Pi_{i=1}^d \hat{T}_{k_i}(x_i). \tag{2.11}$$

As in the 1-D case, the fast algorithm developed in [18] for the Chebyshev transform on the sparse grid in $[-1,1]^d$ can be used for the mapped Chebyshev transform on the sparse grid in $\mathbb{R}^d$. In fact, Algorithm 1 in [18] is applicable to calculating a general transform on the sparse grid spectral representation

$$\{q_{\boldsymbol{j}}, \boldsymbol{j} \in \mathcal{I}_d^q\} = \mathcal{T}\left[\{b_{\boldsymbol{k}}, \boldsymbol{k} \in \mathcal{I}_d^q\}\right] = \left\{ \sum_{\boldsymbol{k} \in \mathcal{I}_d^q} b_{\boldsymbol{k}} t_{\boldsymbol{k},\boldsymbol{j}}, \quad \boldsymbol{j} \in \mathcal{I}_d^q \right\}$$

when $(t_{\boldsymbol{k},\boldsymbol{j}})$ is a tensor product type d-dimensional transform matrix

$$t_{\boldsymbol{k},\boldsymbol{j}} = t^1_{k_1,j_1} \cdots t^d_{k_d,j_d}$$

with all the 1-D transform matrices $t^1_{k,j}, \ldots, t^d_{k,j}$ being block triangular. The block sizes are exactly the sizes of grid increments of consecutive levels. See Remark 2.1 and section 3.3 in [18].

When the 1-D transform matrices are not block triangular matrices, such as the mass matrices of the Chebyshev–Galerkin method for bounded domains and the mass matrices of the finite elements methods, the usual $L+U$ splitting method will add a 2^{d-1} factor to the computational cost (see section 3.3 in [18]); when d is large, this is not favorable. In this paper, we propose an LU decomposition to get rid of this factor. We provide the details in the next subsection, where we apply this technique to compute the right-hand side of (2.26).

We note that Algorithm 1 in [18] is quite general. In a practical application, it is not necessary to set $N_1 = 3$. It can be $5, 7$, or any odd number. In (2.8), we do not have to use the grid indices satisfying $|\boldsymbol{i}|_1 \leqslant q$. In fact, the algorithms proposed here and in [18] work for all grids $\mathcal{I}_d$ with a hierarchical structure, i.e., if $\boldsymbol{i} \in \mathcal{I}_d$, then $\boldsymbol{j} \in \mathcal{I}_d$ for all $\boldsymbol{j} \leqslant \boldsymbol{i}$. Here $\boldsymbol{j} \leqslant \boldsymbol{i}$ means $j_l \leqslant i_l$, $l = 1, \ldots, d$.

2.2. MCSG method. We describe now the scheme (2.4) with V_d^q being the MCSG approximation space defined in (2.11).

2.2.1. Choices of mapping and corresponding stiffness matrices. While there is an infinite choice of mappings, in view of the Chebyshev weight involved in $\hat{T}_k$, we shall restrict ourselves to a family of mapping functions $\{x(\xi)\}$ such that

$$x'(\xi) = \frac{L}{(1-\xi^2)^{1+r/2}}, \quad r \geqslant 0, \tag{2.12}$$

where $L > 0$ is a scaling constant and r determines how fast the mapping $x(\xi)$ goes to infinity as ξ goes to ± 1. We refer to [3] for a thorough discussion on the pros and cons of different mappings.

With (2.12), we have

$$\mu(\xi) = \frac{1}{\sqrt{L}} \left(1-\xi^2\right)^{(1+r)/4} \text{ and } \hat{T}_k(x) = \frac{1}{c_k L} T_k(\xi) \left(1-\xi^2\right)^{(1+r)/4}. \tag{2.13}$$

For $r = 0, 1$, it is easy to verify that

$$x(\xi) = \begin{cases} \frac{L}{2} \log \frac{1+\xi}{1-\xi}, & r = 0, \\ \frac{L\xi}{\sqrt{1-\xi^2}}, & r = 1, \end{cases} \qquad \xi(x) = \begin{cases} \tanh(\frac{x}{L}), & r = 0, \\ \frac{x/L}{\sqrt{x^2/L^2+1}}, & r = 1. \end{cases} \tag{2.14}$$

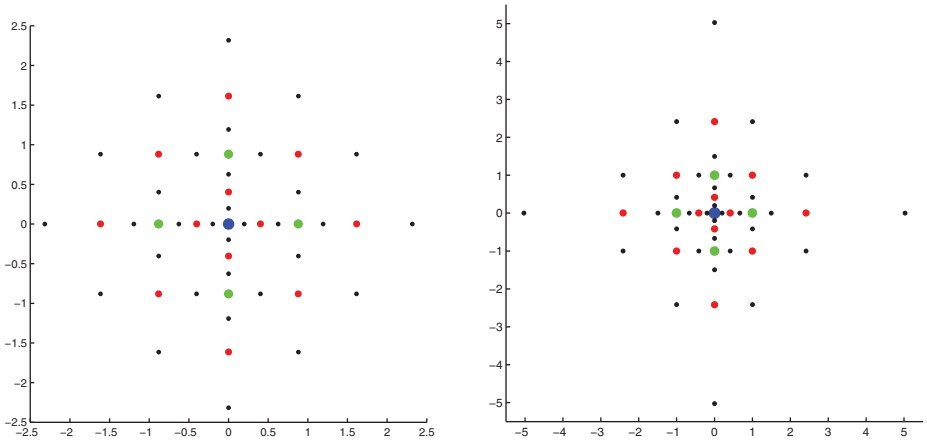

FIG. 2.1. *MCSGs. Left:* $\mathcal{X}_2^5$ *with* $r = 0$*; Right:* $\mathcal{X}_2^5$ *with* $r = 1$*. The points at infinity are not plotted but are used in the transforms.*

For other positive integers r, one can always use an algebraic computing software to derive the explicit expression of the mapping $x(\xi)$. We note that the mapping with $r = 0$ is often referred as the logarithmic mapping, while the mappings with $r > 0$ are referred as algebraic mapping.

We now investigate the behavior of $\hat{T}_k(x)$ as $x \to \infty$ through the relation $\hat{T}_k(x) \propto \mu(\xi) \propto (1-\xi^2)^{(1+r)/4}$.

- $r = 0$: In this case, $\xi = \tanh(x/L)$, we have $\mu(\xi(x)) \propto e^{-|x|/2}$ which decays exponentially fast. Hence, it is good to use the mapping with $r = 0$ if the underlying function decays faster than $e^{-|x|/2}$.
- $r > 0$: In this case, $x(\xi) \propto ((1-\xi^2)^{-r/2})$, so we have $\mu(\xi(x)) \propto |x|^{-(1+r)/2r}$. Therefore, it is preferable to use the mapping with $r > 0$ if the underlying function decays faster than $|x|^{-(1+r)/2r}$.

The actual convergence rates for both $r = 0$ and $r = 1$ will depend on how fast the underlying function decays and also on its smoothness. A rigorous error analysis will be carried out in [14].

We plot in Figure 2.1 the sparse grids $\mathcal{X}_2^5$ generated by the mappings with $r = 0$ and $r = 1$. Since we are not exploring effects of the scaling parameter L, we fix $L = 1$ here and in the rest of the computations.

2.2.2. Stiffness matrices in one dimension. We now examine the sparsity of the 1-D stiffness matrices with the bases $\{S_k\}$ and $\{R_k\}$. By direct calculation,

$$\begin{aligned} \hat{T}_k'(x) &= \frac{1}{\sqrt{c_k}} \left(T_k'(\xi)\, \mu(\xi) + T_k(\xi) \mu'(\xi) \right) \frac{\mathrm{d}\xi}{\mathrm{d}x} \\ &= \frac{1}{L\sqrt{c_k}} \left((1-\xi^2) T_k'(\xi) \mu(\xi) - \frac{1+r}{2} \xi T_k(\xi) \mu(\xi) \right) (1-\xi^2)^{r/2}. \end{aligned} \tag{2.15}$$

When r is zero or a nonnegative integer, by using the properties of Chebyshev polynomials

$$\begin{aligned} T_0(x) = 1, \quad T_1(x) = x, \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geqslant 1, \end{aligned} \tag{2.16}$$

and

$$(1-x^2)T_n'(x) = \frac{n}{2}\left(T_{n-1}(x) - T_{n+1}(x)\right), \quad n \geqslant 1, \tag{2.17}$$

one can show that the stiffness matrix with components $s_{k,j} = (\hat{T}_k'(x), \hat{T}_j'(x))$ is sparse, and smaller r leads to smaller bandwidth. We provide below the explicit formula of the stiffness matrices for the two cases $r = 0, 1$.

- $r = 0$: We denote

$$S_k(x) := \frac{1}{\sqrt{c_k}} T_k(\tanh(x/L)) \frac{1}{\sqrt{L\cosh(x/L)}}. \tag{2.18}$$

By direct calculation, we find

$$\begin{aligned} s_{k,j} &= \left(S_k', S_j'\right) \\ &= \frac{\pi}{32L^2\sqrt{c_k c_j}} \begin{cases} -(2k-1)(2k-3) - 3\delta_{k,2}, & j = k-2, \\ (2k-1)^2 + (2k+1)^2 + 2\delta_{k,0} + \delta_{k,1}, & j = k, \\ -(2k+1)(2k+3) - 3\delta_{k,0}, & j = k+2, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.19}$$

- $r = 1$: We denote

$$R_k(x) := \frac{1}{\sqrt{c_k}} T_k\left(\frac{x/L}{\sqrt{1+(x/L)^2}}\right) \frac{1}{\sqrt{L}} \frac{1}{\sqrt{1+(x/L)^2}}. \tag{2.20}$$

By direct calculation, we find

$$r_{k,j} = (R_k', R_j') = \frac{\pi}{32L^2\sqrt{c_k c_j}} \begin{cases} k^2 - 4k + 3 + 3\delta_{k,4}, & j = k-4, \\ -4(k^2 - 2k + 1) - 4\delta_{k,2}, & j = k-2, \\ 6k^2 + 2 + 2\delta_{k,0} - \delta_{k,2}, & j = k, \\ -4(k^2 + 2k + 1) - 4\delta_{k,0}, & j = k+2, \\ k^2 + 4k + 3 + 3\delta_{k,0}, & j = k+4, \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

Note that the two sets of basis functions are orthonormal in $L^2(\mathbb{R})$ and thus lead to identity mass matrices.

2.2.3. Stiffness matrices in the *d*-dimensional case. Since the formulation of the stiffness matrices with the two sets of basis functions are exactly the same, we shall give the formulation with a generic set of basis functions $\{\Phi_k\}$.

Letting $u = \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}\Phi_{\boldsymbol{k}}$ in (2.4), and replacing v with $\Phi_{\boldsymbol{k}'}, \boldsymbol{k}' \in \mathcal{I}_d^q$, we obtain

$$\kappa \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}(\Phi_{\boldsymbol{k}}, \Phi_{\boldsymbol{k}'}) + \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}\left(\nabla\Phi_{\boldsymbol{k}}, \nabla\Phi_{\boldsymbol{k}'}\right) = (\mathcal{U}_d^q f, \Phi_{\boldsymbol{k}'}) \quad \forall \boldsymbol{k}' \in \mathcal{I}_d^q. \tag{2.22}$$

Since the mass matrix $(\Phi_{\boldsymbol{k}}, \Phi_{\boldsymbol{k}'})$ for the two bases is the identity matrix, the linear system for (2.4) can be rewritten as

$$\begin{aligned} &\kappa b_{\boldsymbol{k}'} + \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}} r_{k_1,k_1'}\delta_{k_2,k_2'}\cdots\delta_{k_d,k_d'} \\ &\qquad + \cdots + \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}\delta_{k_1,k_1'}\cdots\delta_{k_{d-1},k_{d-1}'} r_{k_d,k_d'} = f_{\boldsymbol{k}'} \quad \forall \boldsymbol{k}' \in \mathcal{I}_d^q, \end{aligned} \tag{2.23}$$

where $f_{\boldsymbol{k}'} = (\mathcal{U}_d^q f, \Phi_{\boldsymbol{k}'})$.

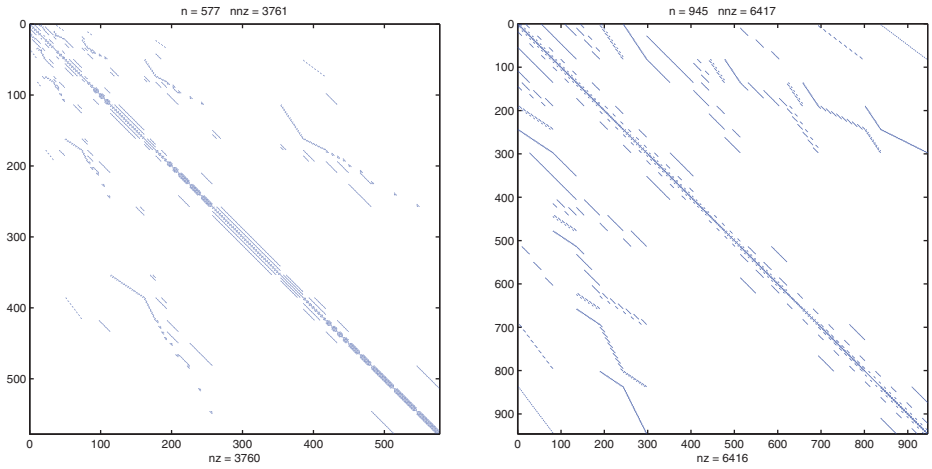

FIG. 2.2. *The sparsity patterns of system matrices of the MCSG method with* $r = 1$. *Left:* $d = 2$, $q = 7$. *Right:* $d = 4, q = 6$.

Thanks to the sparsity of the 1-D stiffness matrix and in particular to the fact that the mass matrix is the identity, the system matrix of the d-dimensional linear problem (2.23),

$$A_{\boldsymbol{k},\boldsymbol{k}'} = \kappa\, \delta_{\boldsymbol{k},\boldsymbol{k}'} + \sum_{i=1}^{d} A^{(i)}_{\boldsymbol{k},\boldsymbol{k}'} \qquad \text{with} \quad A^{(i)}_{\boldsymbol{k},\boldsymbol{k}'} = r_{k_i,k'_i} \prod_{j\neq i} \delta_{k_j,k'_j},$$

is also quite sparse. More precisely, the number of nonzeros per row for the mapping with $r = 1$ (resp., the mapping with $r = 0$ and the Hermite case) is about but less than $1 + 4d$ (resp., $1 + 2d$). The sparsity of this system matrix is essentially the same as that using a finite-difference-based sparse grid (cf. [4]).

Due to the form of the d-dimensional tensor product, these system matrices are much sparser than those for the spectral sparse grid methods in bounded domain presented in [18], where the 1-D stiffness matrix (in the Legendre case) is the identity but the 1-D mass matrix has three nonzero diagonals. Moreover, the system matrix $A_{\boldsymbol{k},\boldsymbol{k}'}$ is very easy to build. The computational cost of building the system matrix is essentially proportional to the number of nonzeros, more precisely, $O(5dn)$ for the $r = 1$ case and $O(3dn)$ for the $r = 0$ case. In Figures 2.2 and 2.3, we plot the sparsity patterns of the two cases, respectively.

2.3. MCHSG method. A natural set of orthonormal basis function in $L^2(\mathbb{R})$ is the normalized Hermite functions defined by

$$\hat{H}_n(x) = \frac{1}{\sqrt{\gamma_n}\sqrt{L}} H_n(x/L), \tag{2.24}$$

where $H_n(x)$ is the Hermite polynomials of degree n and $\gamma_n = n!2^n\sqrt{\pi}$.

Thanks to the properties

$$\hat{H}'_n(x) = \sqrt{\frac{n}{2}}\hat{H}_{n-1}(x) - \sqrt{\frac{n+1}{2}}\hat{H}_{n+1}(x)$$

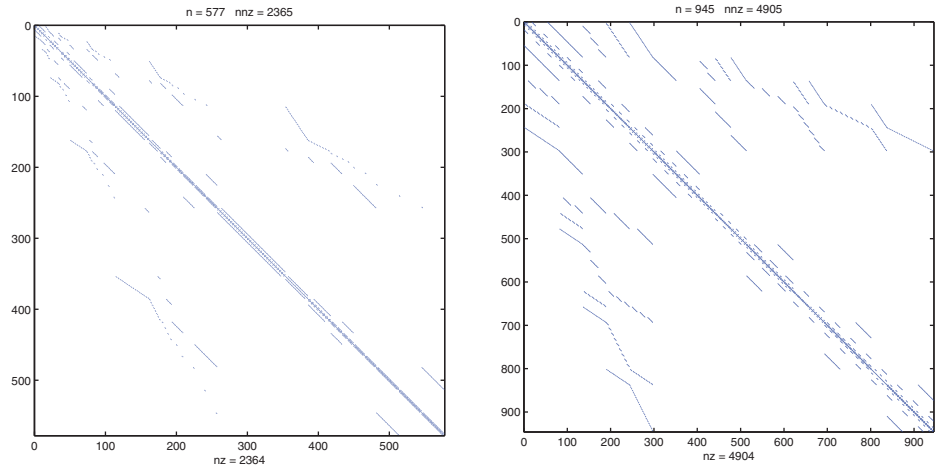

FIG. 2.3. *The sparsity patterns of system matrices of the MCSG method with $r = 0$ and the MCHSG method. Left: $d = 2$, $q = 7$. Right: $d = 4, q = 6$.*

and

$$\int_{\mathbb{R}} \hat{H}_n(x)\hat{H}_m(x)\mathrm{d}x = \delta_{n,m},$$

the stiffness matrix when using bases function $\hat{H}_k(x)$ is also a penta-diagonal banded matrix with nonzeros given by

$$h_{k,j} = (\hat{H}_k', \hat{H}_j') = \frac{1}{2L^2}\begin{cases} \sqrt{k(k-1)}, & j = k-2, \\ 2k+1, & j = k, \\ \sqrt{(k+1)(k+2)}, & j = k+2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.25}$$

However, because Hermite–Gauss points are not nested, the sparse grid generated by Smolyak's algorithm with the Hermite–Gauss quadrature contains many more points than the corresponding mapped Chebyshev sparse grid. In Figure 2.4, we plot the sparse grid $\mathcal{X}_2^5$ generated by Smolyak's algorithm with the Hermite–Gauss quadrature and the corresponding index set $\mathcal{I}_2^5$ in the frequency space.

Instead of using the Hermite sparse grids generated from the nonnested Hermite–Gauss quadrature, we propose the MCHSG method, whose principle is similar to the Chebyshev–Legendre–Galerkin method (cf. [6, 16]). More precisely, given a MCSG $\mathcal{X}_d^q$ and corresponding index set $\mathcal{I}_d^q$, we denote the sparse Hermite approximation space by $\hat{V}_d^q = \mathrm{span}\{\hat{H}_{\boldsymbol{k}} : \boldsymbol{k} \in \mathcal{I}_d^q\}$. Then the MCHSG method is as follows: Find $u \in \hat{V}_d^q$ such that

$$\kappa(u, v) + (\nabla u, \nabla v) = (\mathcal{U}_d^q f, v) \quad \forall v \in \hat{V}_d^q, \tag{2.26}$$

where $\mathcal{U}_d^q$ is the interpolation operator onto the MCSG space V_d^q defined in (2.11).

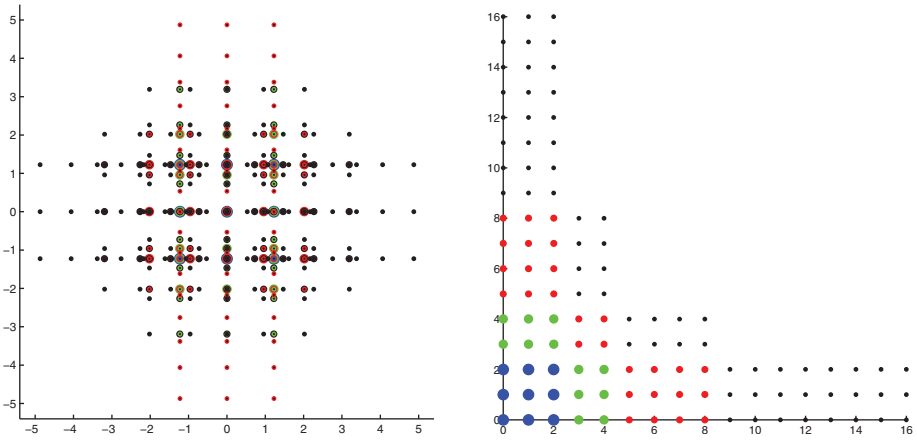

FIG. 2.4. *Left: Hermite sparse grid $\mathcal{X}_2^5$. Right: the corresponding index set $\mathcal{I}_2^5$ in the frequency space. It is obvious that the number of points in physical space (left) is larger than the number of spectral coefficients (right).*

Let $u = \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}\hat{H}_{\boldsymbol{k}}$ and replace v by $\hat{H}_{\boldsymbol{k}'}, \boldsymbol{k}' \in \mathcal{I}_d^q$ in the above equation; we get

$$\kappa \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}}(\hat{H}_{\boldsymbol{k}}, \hat{H}_{\boldsymbol{k}'}) + \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}} \left(\nabla\hat{H}_{\boldsymbol{k}}, \nabla\hat{H}_{\boldsymbol{k}'}\right) = (\mathcal{U}_d^q f, \hat{H}_{\boldsymbol{k}'}) \quad \forall \boldsymbol{k}' \in \mathcal{I}_d^q \tag{2.27}$$

or

$$\kappa b_{\boldsymbol{k}'} + \sum_{i=1}^{d} \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} b_{\boldsymbol{k}} \delta_{k_1,k_1'} \cdots \delta_{k_{i-1},k_{i-1}'} h_{k_i,k_i'} \delta_{k_{i+1},k_{i+1}'} \cdots \delta_{k_d,k_d'} = f_{\boldsymbol{k}} t_{k_1 k_1'} \cdots t_{k_d k_d'} \quad \forall \boldsymbol{k}' \in \mathcal{I}_d^q, \tag{2.28}$$

where

$$t_{k,j} = \left(R_k(x), \hat{H}_j(x)\right)$$

is the transform matrix from mapped Chebyshev representation to Hermite representation.

We note that the 1-D stiffness matrices $\{s_{k,j}\}$ and $\{h_{k,j}\}$ have exactly the same sparse pattern. Therefore, the only difference in implementation between the MCSG method and the MCHSG method is the evaluation of the right-hand side. The latter case is much more complicated, but it can still be evaluated by using the fast sparse grid transform with LU decomposition given by Algorithm 1.

2.4. Efficient implementations.

2.4.1. A general sparse grid transform with an LU decomposition. We need an LU decomposition because, the tensor-product subgrids in a sparse grid have different sizes. Consider, for example, the transform on the two-dimensional sparse grid $\mathcal{X}_2^3$:

$$\hat{f}_{k_1',k_2'} = \sum_{(k_1,k_2)\in\mathcal{I}_2^3} f_{k_1,k_2} t_{k_1,k_1'} t_{k_2,k_2'} \quad \forall (k_1', k_2') \in \mathcal{I}_2^3. \tag{2.29}$$

For the indices $k_1 \in \mathcal{I}^1, k_2 \in \mathcal{I}^2, k_1' \in \mathcal{I}^2 \backslash \mathcal{I}^1$, we know $(k_1, k_2) \in \mathcal{I}_2^3$, but $(k_1', k_2) \notin \mathcal{I}_2^3$ for some k_1', k_2. However, $(k_1', k_2') \in \mathcal{I}_2^3$ for all the $k_2' \in \mathcal{I}^1$. So if we first calculate $f^1_{k_1',k_2} = \sum_{(k_1,k_2)\in\mathcal{I}_2^3} f_{k_1,k_2} t_{k_1,k_1'}$, then the index set of the nonzeros of $f^1_{k_1',k_2}$ will be larger than $\mathcal{I}_2^3$. But if the transform matrix $t_{k,j}$ is a block lower triangular matrix, then the index set of the nonzeros of $f^1_{k_1',k_2}$ will stay in $\mathcal{I}_2^3$. When the transform matrix is an upper block triangular matrix, to calculate $f^2_{k_1',k_2'} = \sum_{(k_1',k_2)\in\mathcal{I}_2^3} f^1_{k_1',k_2} t_{k_2,k_2'}$ with $(k_1', k_2') \in \mathcal{I}_2^3$, we only need the values of $f^1_{k_1',k_2}$ with $(k_1', k_2) \in \mathcal{I}_2^3$. This argument shows that we should first calculate the 1-D transforms along the directions associated with a block lower triangular matrix, followed by at most one 1-D transform with a general matrix, and then calculate all the 1-D transforms which have a block upper triangular structure.

It is easy to see that the LU decomposition $t_{k,j} = \sum_{m\leqslant \min\{k,j\}} l_{k,m} u_{m,j}$ for different grid sizes will have a hierarchical structure. So if we use the same 1-D basis functions in all d-dimensions, then only one LU decomposition for a single transform matrix is needed. After the decomposition, the transform (2.29) is equivalent to

$$\begin{aligned}\hat{f}_{k_1',k_2'} &= \sum_{(k_1,k_2)\in\mathcal{I}_2^3} \sum_{m_i<\min\{k_i,k_i'\}} f_{k_1,k_2} \left(l_{k_1,m_1} u_{m_1,k_1'}\right)\left(l_{k_2,m_2} u_{m_2,k_2'}\right) \\ &= \sum_{(m_1,m_2)\in\mathcal{I}_2^3} \left(\sum_{(k_1,k_2)\in\mathcal{I}_2^3} f_{k_1,k_2} l_{k_1,m_1} l_{k_2,m_2} \right) u_{m_1,k_2'} u_{m_2,k_2'} \quad \forall (k_1', k_2') \in \mathcal{I}_2^3,\end{aligned}$$

which involves two simple transforms:

$$f^l_{m_1,m_2} = \sum_{(k_1,k_2)\in\mathcal{I}_2^3} f_{k_1,k_2} l_{k_1,m_1} l_{k_2,m_2}, \quad \hat{f}_{k_1',k_2'} = \sum_{(m_1,m_2)\in\mathcal{I}_2^3} f^l_{m_1,m_2} u_{m_1,k_2'} u_{m_2,k_2'}.$$

The details are given as Algorithm 1.

ALGORITHM 1. Fast transform on sparse grid with LU decomposition.

Input: q, d, $\mathcal{I}_d^q$, $\left\{ t^i_{k,j},\ k,j = 1,\ldots,N_{q-d+1},\ i = 1,\ldots,d \right\}$, and $\left\{ f_{\boldsymbol{k}},\ \boldsymbol{k} \in \mathcal{I}_d^q \right\}$.
Output: $\left\{ b_{\boldsymbol{j}}, = \sum_{\boldsymbol{k}\in\mathcal{I}_d^q} f_{\boldsymbol{k}} t^1_{k_1,j_1} \cdots t^d_{k_d,j_d},\ \boldsymbol{k} \in \mathcal{I}_d^q \right\}$.

```
Let n = N_{q-d+1};
for i = 1 to d do
   Calculate (block) LU decomposition:
   t^i_{k,j} = Σ_{m=1}^{n} l^i_{k,m} u^i_{m,j},  k,j = 1,...,n.
end for
b_k ← f_k , for all k ∈ I^q_d.
for i = 1 to d do
   b_{k_1,...,k_{i-1},k_i',k_{i+1},...,k_d} ← Σ_{k_i'≤k_i} b_k l^i_{k_i,k_i'}, for all k ∈ I^q_d.
end for
for i = 1 to d do
   b_k ← Σ_{k_i'≤k_i} b_{k_1,...,k_{i-1},k_i',k_{i+1},...,k_d} u^i_{k_i',k_i}, for all k ∈ I^q_d.
end for
```

Note that the LU decomposition (or Cholesky decomposition for the symmetric case) preserves the sparse property for banded transform matrices. The technique we introduced here can be used to decompose the mass matrices in sparse solvers for

bounded domains [18] and speed up the matrix-vector product algorithm with large dimensions.

Note that to maintain the hierarchical structure, the LU decomposition cannot pivot from all the rows and columns, only from the same grid increment.

Algorithm 1 is used to calculate the right-hand side of the discrete linear system of the MCHSG method. First, $f_{\boldsymbol{k}}$ in (2.28) is evaluated by applying the forward sparse grid fast Chebyshev transform on

$$\left\{ f(\boldsymbol{x}(\boldsymbol{\xi_j}))/\mu(\boldsymbol{\xi_j}), \boldsymbol{\xi_j} \in \mathcal{X}_d^q \right\};$$

then Algorithm 1 is used to apply the transforms $t_{k_i,k_i'}, i = 1, \ldots, d$, to $f_{\boldsymbol{k}}$.

2.4.2. Iterative and direct solvers. We consider now three options for solving the linear algebraic system for the MCSG and MCHSG methods.

- The first option is to use an iterative method, e.g., the preconditioned conjugate gradient (PCG) method, which only needs the matrix-vector product without building the system matrix explicitly. This can be done with linear computational cost by using Algorithm 1 proposed in [18]. However, the simple diagonal preconditioner does not perform well, and it is not clear how to construct better preconditioners.
- The second option is to use black-box iterative solvers such as the algebraic multigrid method (AMG) [15]. AMG works very well for the MCHSG method and the MCSG method with $r = 0$. The iteration numbers are bounded, which indicates that the computational costs are quasi-optimal.
- The third option is to use direct sparse solvers such as CHOLMOD [5] and SuperMF [21]. Theoretically, CHOLMOD has a decomposition complexity of order $O(N^{3/2})$, with memory use of order $O(N \log(N))$, where N is the number of grid points. SuperMF has a theoretical decomposition complexity of order $O(r^2 N)$, with memory use of order $O(rN)$, where r is a problem dependent constant.

In practice, SuperMF is faster than CHOLMOD, but CHOLMOD is more robust. In Table 1, we provide a performance comparison of PCG, AMG, and CHOLMOD for the MCSG methods with $r = 0$ and $r = 1$. The MCHSG method is similar to the MCSG method with mapping parameter $r = 0$. It is clear that for small to medium-size problems, the CHOLMOD is the best. It is also suitable to problems with multiple right-hand sides. However, when the system size becomes large, the AMG method will eventually outperform the direct solvers.

2.5. Numerical results and discussion. We now examine the accuracy and efficiency of the proposed algorithms by solving (2.4) with exact solutions:

$$u_1(\boldsymbol{x}) = \frac{1}{\sqrt{c_1}^d} e^{-\frac{\sum_i |x_i|^k}{2}}, \quad c_1 = \frac{2\Gamma(1/k)}{k}; \tag{2.30}$$

$$u_2(\boldsymbol{x}) = \frac{1}{\sqrt{c_2}^d} \frac{1}{\prod_i (1 + |x_i|^k)}, \quad c_2 = \frac{2\,\mathrm{Beta}(1/k, 2 - 1/k)}{k}; \tag{2.31}$$

$$u_3(\boldsymbol{x}) = \frac{1}{\sqrt{c_3}^d} \frac{2^d}{\prod_i (e^{x_i} + e^{-x_i})^k}, \quad c_3 = \frac{2^{2k}\,\mathrm{Beta}(k, k)}{2};$$

$$u_4(\boldsymbol{x}) = \frac{1}{(1 + \sum_i |x_i|^3)^{d/3+k}};$$

TABLE 1

The performance comparison among PCG, AMG and CHOLMOD. n_{pcg} and n_{amg} stand for the iteration number of PCG method and AMG method, respectively; t_{pcg}, t_{amg}, t_{chf}, and t_{chs} stand for the CPU time of PCG solver, AMG solver, the factorization step of CHOLMOD, and the solution step of CHOLMOD, respectively. The maximum 1-D scheme size in the sparse grid is set to 1025.

$r=0$									
d	q	DoF	nnz	n_{pcg}	n_{amg}	t_{pcg}	t_{amg}	t_{chf}	t_{chs}
2	13	38913	186365	1826	32	5.44(0)	2.54(−1)	5.00(−2)	0
	14	67585	329725	1915	32	1.05(1)	5.12(−1)	1.20(−1)	1.00(−2)
	15	116737	575485	2047	32	2.02(1)	9.06(−1)	2.80(−1)	1.00(−2)
	16	198657	985085	2149	33	3.86(1)	1.68(0)	1.14(0)	1.10(−1)
4	10	52993	333561	272	26	1.62(0)	3.53(−1)	4.40(−1)	3.00(−2)
	11	133889	865017	523	28	8.70(0)	1.12(0)	2.56(0)	1.10(−1)
	12	331777	2191353	997	30	4.31(1)	3.25(0)	1.88(1)	3.80(−1)
	13	808961	5445625	1901	31	2.02(2)	9.08(0)	9.44(1)	1.24(0)
$r=1$									
d	q	DoF	nnz	n_{pcg}	n_{amg}	t_{pcg}	t_{amg}	t_{chf}	t_{chs}
2	13	28913	325617	1424	417	5.31(0)	3.94(0)	2.30(−1)	2.00(−2)
	14	67585	583665	1416	413	9.25(0)	7.68(0)	6.80(−1)	4.00(−2)
	15	116737	1026033	1423	407	1.70(1)	1.36(1)	2.01(0)	1.10(−1)
	16	198657	1763313	1446	546	3.01(1)	3.17(1)	6.26(0)	2.50(−1)
4	10	52993	512737	680	106	4.61(0)	1.52(0)	2.71(0)	7.00(−2)
	11	133889	1354465	1389	163	2.64(1)	6.59(0)	1.79(1)	2.70(−1)
	12	331777	3483617	2491	268	1.26(2)	3.27(1)	1.04(2)	8.90(−1)
	13	808961	8767457	2679	390	3.29(2)	1.26(2)	6.63(2)	3.11(0)

where Γ is the Gamma function, and Beta is the Beta function. The corresponding right-hand-side functions $f = \kappa\, u - \Delta u$ are given by

$$f_1(\boldsymbol{x}) = u_1(\boldsymbol{x}) \left\{ \kappa - \sum_i \left(\frac{k^2}{4} |x_i|^{2k-2} - \frac{k(k-1)}{2} |x_i|^{k-2} \right) \right\}, \quad k \geqslant 2,$$

$$f_2(\boldsymbol{x}) = u_2(\boldsymbol{x}) \left\{ \kappa - \sum_i \left(\frac{2k^2 |x_i|^{2k-2}}{(1+|x_i|^k)^2} - \frac{k(k-1)|x_i|^{k-2}}{1+|x_i|^k} \right) \right\}, \quad k \geqslant 2,$$

$$f_3(\boldsymbol{x}) = u_3(\boldsymbol{x}) \left\{ \kappa - \sum_i \left(k(k+1) \left(\frac{e^{2x_i}-1}{e^{2x_i}+1} \right)^2 - k \right) \right\}, \quad k > 0,$$

$$f_4(\boldsymbol{x}) = u_4(\boldsymbol{x}) \left\{ \kappa - \sum_i (d+3k) \left(\frac{(d+3k+3)|x_i|^4}{(1+\sum_i |x_i|^3)^2} - \frac{2|x_i|^1}{1+\sum_i |x|^3} \right) \right\}, \quad k \geqslant 2.$$

We fix $L = 1$ in all the numerical tests. We note that the tuning of the parameter L may result in significant convergence improvement (cf. [20, 1]) but we shall not address this issue here.

2.5.1. Tensor product of 1-D functions with algebraic convergence. Consider first functions $F(\boldsymbol{x})$ which are constructed as a tensor product of 1-D function $f(x)$ with algebraic convergence for a given set of bases functions $\{\Phi_k\}$. Namely,

$$f(x) = \sum_k \tilde{f}_k \Phi_k(x) \quad \text{with} \quad |\tilde{f}_k| \sim k^{-\alpha}$$

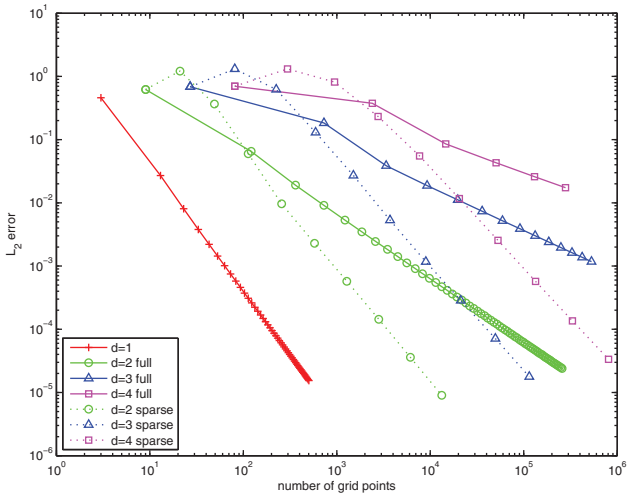

FIG. 2.5. *Comparison of MCFG and MCSG methods (with $r = 1$) for solving* (2.1) *with $f = f_2$, $k = 3$.*

and

$$F(\boldsymbol{x}) = \Pi_{i=1}^{d} f(x_i) = \sum_{i=1}^{d} \sum_{k_i} \tilde{f}_{k_1,k_2,\ldots,k_d} \Pi_{i=1}^{d} \Phi_{k_i}(x_i),$$

for example, u_1 with odd number k for the MCSG methods with $r = 0, 1$; u_2 with any $k \geqslant 2$ for the MCSG method with $r = 1$; u_3 with $k \geqslant 1$ for the MCSG method with $r = 0$.

It is clear that $|\tilde{f}_{k_1,k_2,\ldots,k_d}| \sim (\prod_{i=1}^{d} k_i)^{-\alpha}$. Therefore, for a fixed accuracy expressed as $K^{-\alpha}$, it is clear that all the terms with index $(k_1, k_2, \ldots, k_d)$ such that $\prod_{i=1}^{d} k_i \leq K$ should be kept. This is the ideal case for the usual sparse grid/hyperbolic cross approximation. So we should fix N_1 (the number of quadrature points in the coarsest grid) and increase q until the required accuracy is achieved.

A comparison of the solution error versus the number of grid points between the mapped Chebyshev full grid (MCFG) method and the MCSG method with $r = 1$ for the test function u_2 with $k = 3$ is given in Figure 2.5. Clearly, the MCSG method converges at a rate that is almost independent of d.

2.5.2. Tensor product of 1-D functions with geometric/subgeometric convergence. Consider now functions $F(\boldsymbol{x})$ which are constructed as a tensor product of 1-D function $f(x)$ with geometric/subgeometric convergence for a given set of bases functions $\{\Phi_k\}$. Namely,

$$f(x) = \sum_{k} \tilde{f}_k \Phi_k(x) \quad \text{with} \quad |\tilde{f}_k| \sim \exp(-\alpha k^{\beta}),$$

where $\beta = 1$ (resp., $0 < \beta < 1$) is referred to as geometric (resp., subgeometric) convergence, and

$$F(\boldsymbol{x}) = \Pi_{i=1}^{d} f(x_i) = \sum_{i=1}^{d} \sum_{k_i} \tilde{f}_{k_1,k_2,\ldots,k_d} \Pi_{i=1}^{d} \Phi_{k_i}(x_i),$$

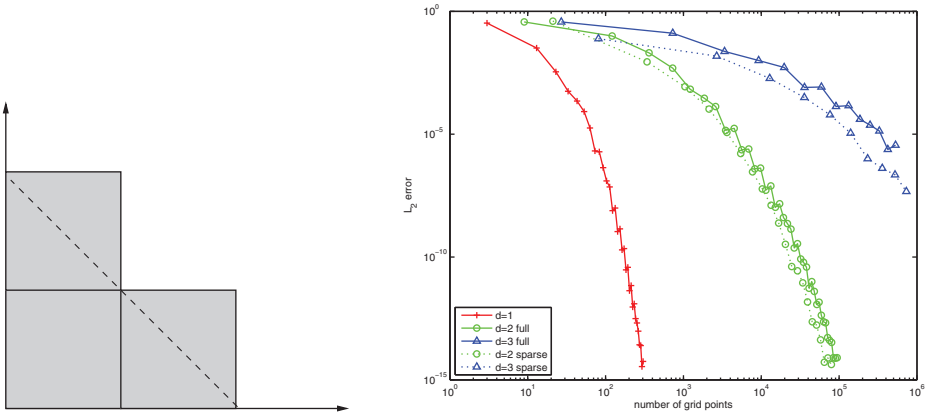

FIG. 2.6. *Left: the optimal frequency index set for tensor product of* 1*-D function with geometric convergence. Right: Comparison of MCFG and MCSG methods (with* $r = 1$*)* $\mathcal{X}_d^{d+1}$ *for solving* (2.1) *with* $f = f_1$*,* $k = 2$*.*

for example, smooth functions with sufficiently fast exponential decays, such as $u_1(\boldsymbol{x})$ with even k and $u_3(\boldsymbol{x})$, for both the MCHSG method and MCSG methods with $r = 0, 1$.

It is clear that $|\tilde{f}_{k_1,k_2,\ldots,k_d}| \sim \exp(-\alpha \sum_{i=1}^d k_i^\beta)$. Therefore, for a fixed accuracy expressed as $\exp(-\alpha K)$, it is clear that all the terms with index $(k_1, k_2, \ldots, k_d)$ such that $\sum_{i=1}^d k_i^\beta \le K$ should be kept.

Consider first the geometric convergence case, i.e., $\beta = 1$; the marginal curve of the required index set is a hyperplane $\sum_{i=1}^d k_i = K$ (cf. the left side of Figure 2.6). In this case, the usual sparse grid with $N_1 = 3$ (N_1 is the number of grid points at the first level) is not suitable. The best hierarchical grid that covers this curve is a two-level sparse grid $\mathcal{X}_d^{d+1}$ (cf. the left side of Figure 2.6). Therefore, to obtain a better convergence rate, we should fix $q = d + 1$ and increase N_1. The ratio of the number of grid points in a two-level sparse grid to that in a full grid is $\frac{1+d}{2^d}$. We plot on the right side of Figure 2.6 the error versus number of grid points when using full grid and two-level sparse grids $\mathcal{X}_d^{d+1}$ with different N_1. We observe that even in the geometric convergence case which is usually unfavorable to sparse grid approximations, the two-level MCSG method still outperforms the MCFG method.

Consider now a subgeometric convergence case with $\beta = 1/2$; the marginal curve $\sum_{i=1}^d k_i^{1/2} = K$ of the required index set is shown on the left side of Figure 2.7. In this case, the best hierarchical grid that covers this curve is a four-level grid $\mathcal{X}_d^{d+3}$. The ratio of the number of grid points in such a sparse grid to that in a full grid is $((d + 1)2^d + 4d)/8^d$. The right side of Figure 2.7 shows the error versus number of grid points when using the MCFG method and the four-level MCSG method with different sizes of first-level grid. We observe that the four-level MCSG method also outperforms the MCFG method, and the relative efficiency is better than in the geometric convergence case.

2.5.3. An example of a nontensor product function. Most functions in real applications are not of tensor product type. As long as they are not isotropically smooth, a spectral sparse grid method will usually outperform a spectral full grid method. More importantly, only sparse grid methods are feasible for higher-dimensional problems. In Figure 2.8, we compare the MCSG method with the MCFG

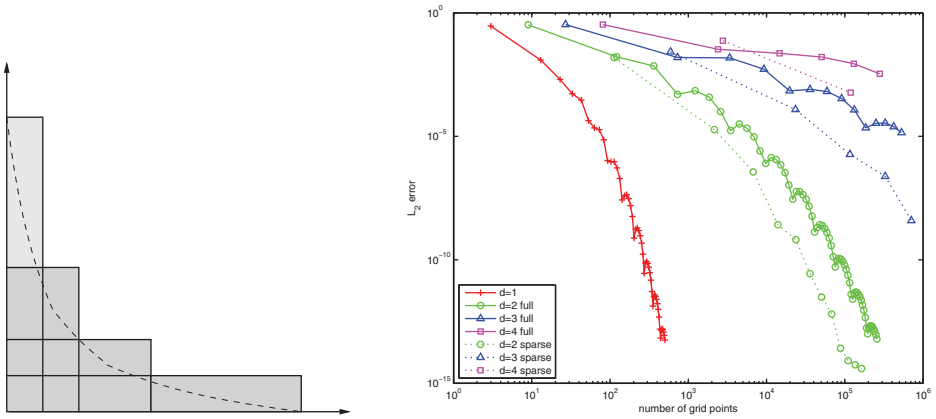

FIG. 2.7. *Left: the optimal frequency index set for tensor product of* 1*-D function with sub-geometric convergence. Right: Comparison of MCFG and MCSG methods (with* $r = 1$*)* $\mathcal{X}_d^{d+3}$ *for solving* (2.1) *with* $f = f_3$*,* $k = 1$*.*

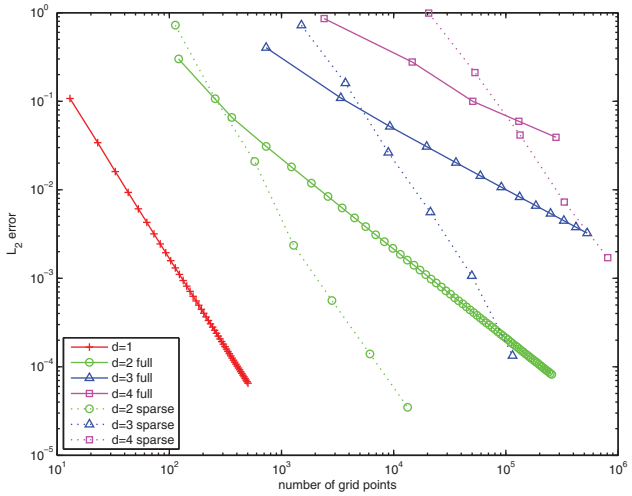

FIG. 2.8. *Comparison of MCSG and MCFG methods (with* $r = 1$*) for solving* (2.1) *with* $f = f_4$*,* $k = 3$*.*

methods (with $r = 1$) on the test function u_4 with $k = 3$. It is observed that the MCSG method converges much faster than the MCFG method for $d > 1$.

2.5.4. Comparison between MCSG methods with $r = 0, 1$ and the MCHSG method. In general, the MCSG method with $r = 1$ and the MCHSG method can be applied to problems with both exponentially and algebraically decaying functions, while the MCSG method with $r = 0$ is recommended only for exponentially decaying functions.

In terms of computational cost per node, the MCSG method with $r = 0$ is the most efficient as its system matrix is more compact than the MCSG method with $r = 1$, and the Hermite method is not as efficient as the MCSG methods since it lacks a fast transform between physical values and spectral coefficients. In Figure 2.9, we show the numerical results of solving (2.4) with $f = f_1$, $k = 3$ by using the MCSG

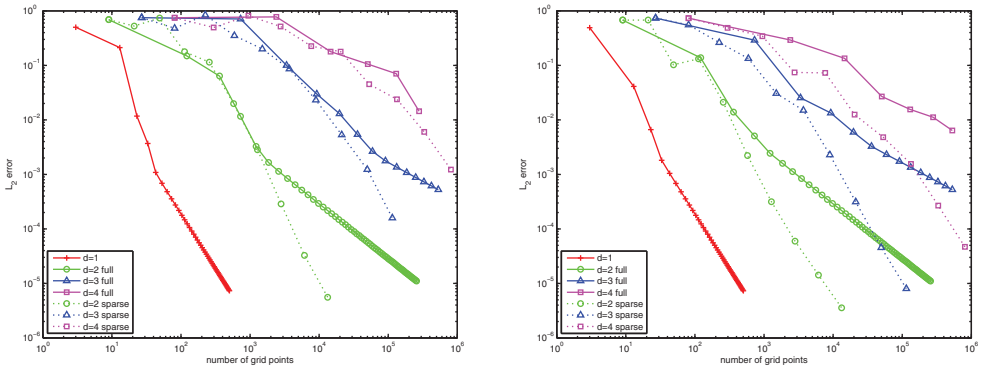

FIG. 2.9. *Comparison of two MCSG methods when solving* (2.1) *with* $f = f_1$, $k = 3$. *Left: MCSG method with* $r = 1$. *Right: MCSG method with* $r = 0$.

methods with $r = 0, 1$. For this particular example, the two methods have a similar asymptotic convergent rate, but the MCSG method with $r = 0$ performs better when only a few grid points are used in the 1-D case, leading to better approximation results in the preasymptotic range in the higher-dimensional case.

2.6. Elliptic equations with variable coefficients. For problems with variable coefficients, the mass and stiffness matrices in the pseudospectral–Galerkin sparse grid method (2.3) are no longer sparse, so the explicit evaluation of these matrices is prohibitively expensive. Therefore, a suitable preconditioned Krylov-type method requiring only matrix-vector products should be employed. An effective preconditioner is the MCSG method for the constant coefficient problem (2.4). In fact, this preconditioner is optimal if the variable coefficients κ and α are bounded above and below by positive constants. Due to the presence of interpolation operators, the system (2.3) is no longer symmetric but it is still positive definite, so we shall employ the BICGSTAB iteration, which only requires matrix-vector products.

Hence, we only need to provide algorithms to calculate

$$\{u_{\boldsymbol{k}}\} \to \mathcal{M}(u)_{\boldsymbol{j}} = \left(\mathcal{U}_d^q\left[\kappa(\boldsymbol{x})\sum_{\boldsymbol{k}} u_{\boldsymbol{k}}\hat{T}_{\boldsymbol{k}}(\boldsymbol{x})\right], \hat{T}_{\boldsymbol{j}}(\boldsymbol{x})\right), \quad \boldsymbol{j} \in \mathcal{I}_d^q, \tag{2.32}$$

and

$$\begin{aligned}\{u_{\boldsymbol{k}}\} \to \mathcal{S}(u)_j &= \left(\mathcal{U}_d^{q+1}\left[\alpha(\boldsymbol{x})\nabla\sum_{\boldsymbol{k}} u_{\boldsymbol{k}}\hat{T}_{\boldsymbol{k}}(\boldsymbol{x})\right], \nabla\hat{T}_{\boldsymbol{j}}(\boldsymbol{x})\right)\\ &= -\left(\nabla\mathcal{U}_d^{q+1}\left[\alpha(\boldsymbol{x})\nabla\sum_{\boldsymbol{k}} u_{\boldsymbol{k}}\hat{T}_{\boldsymbol{k}}(\boldsymbol{x})\right], \hat{T}_{\boldsymbol{j}}(\boldsymbol{x})\right), \quad \boldsymbol{j} \in \mathcal{I}_d^q.\end{aligned} \tag{2.33}$$

The first term (2.32) can be computed by using the usual procedure: (i) transform u from spectral representation $u_{\boldsymbol{k}}$ to physical representation $u(\boldsymbol{x})$, (ii) multiply it by $\kappa(\boldsymbol{x})$, and (iii) transform the result from physical values to spectral representation; the resulting spectral coefficients are the components $\mathcal{M}(u)_{\boldsymbol{j}}, \boldsymbol{j} \in \mathcal{I}_d^q$.

The second term is much more complicated. The main difficulty is that the MCSG space V_d^q is not closed under differentiation, i.e., $u \in V_d^q$ does not imply $\nabla u \in V_d^q$. This is why we need to perform the multiplication on a finer sparse grid $\mathcal{X}_d^{q+1}$.

First, we derive from (2.15) and the basic properties of Chebyshev polynomials (2.16) and (2.17) that

$$\sqrt{c_k}\hat{T}_k'(x) = \frac{(1-\xi^2)^{r/2}}{L}\sum_{j=k-1}^{k+1}\beta_j\left(\sqrt{c_j}\hat{T}_j(x)\right), \tag{2.34}$$

where

$$\beta_{k-1} = \frac{k}{2} - \frac{1+r}{4}, \quad \beta_k = 0, \quad \beta_{k+1} = -\frac{k}{2} - \frac{1+r}{4}, \quad \text{for } k \geqslant 1;$$
$$\beta_{k-1} = 0, \quad \beta_k = 0, \quad \beta_{k+1} = -\frac{1+r}{2}, \quad \text{for } k = 0.$$

We now proceed separately for $r = 0$ and $r \neq 0$.

- Case $r = 0$: In this case, the formula (2.34) simplifies to a summation of mapped Chebyshev functions. We first extend u to the finer sparse grid space V_d^{q+1}, take the derivatives using (2.34), then transform to physical space and multiply the result by $\alpha(\boldsymbol{x})$, and then transform the result back to spectral representation, then take the outer differentiation. In the end, project the resulting function from V_d^{q+1} to V_d^q.
- Case $r \neq 0$: We still use the formula (2.34); after taking the inner differentiation, we transform $\frac{L}{(1-\xi_i^2)^{r/2}}\partial_{x_i}u(\boldsymbol{x}) = \sum_{\boldsymbol{k}} u_{\boldsymbol{k}}\sum_{j=k-1}^{k+1}\beta_j(\sqrt{c_j}\hat{T}_j(x))$ to physical space. Then we multiply the result by $(1-\xi_i^2)^r\alpha(\boldsymbol{x})$ and transform the new result to spectral representation $\sum_{\boldsymbol{k}} h_{\boldsymbol{k}}\hat{T}_{\boldsymbol{k}}(\boldsymbol{x})$. In other words, we have

$$\partial_{x_i}\left(\alpha(\boldsymbol{x})\partial_{x_i}u(\boldsymbol{x})\right) = \frac{1}{L}\partial_{x_i}\left((1-\xi_i^2)^{-r/2}\sum_{\boldsymbol{k}} h_{\boldsymbol{k}}\hat{T}_{\boldsymbol{k}}(\boldsymbol{x})\right).$$

  Finally, the outer differentiation in the above can be calculated by using the following 1-D formula:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}x}\left(\left(1-\xi^2\right)^{-\frac{r}{2}}\sqrt{c_k}\hat{T}_k(x)\right)\\
&\quad = \left(1-\xi^2\right)^{-\frac{r}{2}}\frac{\mathrm{d}}{\mathrm{d}x}\left(\sqrt{c_k}\hat{T}_k(x)\right) + \sqrt{c_k}\hat{T}_k(x)\cdot r\xi\left(1-\xi^2\right)^{-1-\frac{r}{2}}\frac{\mathrm{d}\xi}{\mathrm{d}x}\\
&\quad = \sum_{j=k-1}^{k+1}\tilde{\beta}_j\tilde{\phi}_j(x),
\end{aligned}$$

  where

$$\tilde{\beta}_{k-1} = \frac{k}{2} + \frac{r-1}{4}, \quad \tilde{\beta}_k = 0, \quad \tilde{\beta}_{k+1} = -\frac{k}{2} + \frac{r-1}{4}, \quad k \geqslant 1;$$
$$\tilde{\beta}_{k-1} = 0, \quad \tilde{\beta}_k = 0, \quad \tilde{\beta}_{k+1} = \frac{r-1}{2}, \quad k = 0.$$

  In the end, we project the resulting function from V_d^{q+1} to V_d^q.

We now present some numerical results. We take two sets of nonconstant coefficients as test examples:

$$\kappa(\boldsymbol{x}) = \kappa\left(1 + 0.5\cos(2\pi x_1)\right), \quad \alpha(\boldsymbol{x}) = 1 + 0.5\cos(2\pi x_1), \tag{2.35}$$

TABLE 2
Numbers of iterations and CPU time using BICGSTAB to solve equations with nonconstant coefficients. $\kappa = 1, \alpha = 1$. BICGSTAB tolerance 1×10^{-12}. $k = 3$ in u_1 and u_2.

		(2.35), u_1, $r = 0$		(2.35), u_2, $r = 1$		(2.36), u_1, $r = 0$		(2.36), u_2, $r = 1$	
d	q	iter#	CPU	iter#	CPU	iter#	CPU	iter#	CPU
1	7	13	3.000(−3)	13	4.000(−3)	6	1.000(−3)	6	1.000(−3)
	8	13	7.000(−3)	13	6.000(−3)	5	2.000(−3)	6	3.000(−3)
	9	15	1.000(−2)	13	1.000(−2)	5	4.000(−3)	6	4.000(−3)
2	8	12	4.400(−2)	13	5.200(−2)	6	2.600(−2)	6	2.000(−2)
	9	13	1.040(−1)	14	1.150(−1)	5	4.200(−2)	6	4.200(−2)
	10	13	2.270(−1)	14	2.410(−1)	5	8.700(−2)	6	9.200(−2)
3	9	14	6.560(−1)	13	7.780(−1)	6	2.920(−1)	6	3.030(−1)
	10	14	1.510(−0)	13	1.911(−0)	6	6.830(−1)	6	7.500(−1)
	11	14	4.071(−0)	13	4.749(−0)	6	1.857(−0)	6	1.870(−0)

and

$$\kappa(\boldsymbol{x}) = \kappa\left(1 + 0.5e^{-\frac{1}{1+|\boldsymbol{x}|^2}}\right), \quad \alpha(\boldsymbol{x}) = 1 + 0.5e^{-\frac{1}{1+|\boldsymbol{x}|^2}}, \tag{2.36}$$

and let the exact solution be u_1 and u_2 given in (2.30) and (2.31). We use the MCSG method with $\kappa = 1, \alpha = 1$ as the preconditioner. For a tolerance of 1×10^{-12}, BICGSTAB converges in about 12 to 14 iterations for the first case (2.35) and in about 6 iterations for the second case (2.36). No obvious dependence on r, d, q is observed. Details are given in Table 2.

3. Application to the electronic Schrödinger equation. The electronic Schrödinger equation plays a fundamental role in quantum chemistry. Its accurate numerical solution is a great challenge mainly due to its high dimensionality. It is rigorously shown in [22] and numerically verified in [23, 10, 9] that the sparse grid approach is suitable for numerically solving the electronic Schrödinger equation. However, previous works all employ the Fourier sparse grid approach, which replaces the problem in the unbounded domain $\mathbb{R}^d$ by a finite domain $(0, L)^d$ with periodic boundary conditions. While this approach greatly simplifies the implementation, it introduces the finite-size effect (see, for instance, [7]), which is difficult to quantify.

In this section, we present a pseudospectral sparse grid approach to solve the electronic Schrödinger equation in $\mathbb{R}^d$ directly.

3.1. Description of the problem and numerical methods. As a starting point, we shall restrict our attention in this paper to the case of one spatial dimension with N electrons. Therefore, we are dealing with an N-dimensional problem. In this case, the problem (cf. [22]) is to look for eigenvalues and eigenfunctions (E, ψ) of the electronic Schrödinger equation

$$H\psi(x_1, \ldots, x_N) = E\psi(x_1, \ldots, x_N), \tag{3.1}$$

where H is the Hamilton operator

$$H = T + U + V = -\frac{1}{2}\sum_{i=1}^{N} \partial_{x_i}^2 + \sum_{i=1}^{N} Z|x_i| - \sum_{i=1}^{N}\sum_{j>i}^{N} |x_i - x_j|. \tag{3.2}$$

In the above, $x_i \in \mathbb{R}$ denotes the position of the ith electron ($i = 1, 2, \ldots, N$), and Z is the charge of the fixed nucleus at the origin.

Since it is known that the eigenfunctions of (3.1) decay exponentially as $x_i \to \infty$, we shall use the mapped Chebyshev (with $r = 0$) sparse grid method. Let us denote $(x_1, \ldots, x_N) = \boldsymbol{x}$. As usual, $V_N^q = \{\phi_{\boldsymbol{k}}(\boldsymbol{x}), \boldsymbol{k} \in \mathcal{I}_N^q\}$ denotes the sparse approximation space with $\phi_{\boldsymbol{k}}(\boldsymbol{x}) = S_{\boldsymbol{k}}(\boldsymbol{x})$ being the basis function defined in (2.18). Then, the pure Galerkin version of the MCSG method for (3.1) is as follows: Find $\psi_N^q = \sum_{\boldsymbol{k}\in\mathcal{I}_N^q} b_{\boldsymbol{k}}\phi_{\boldsymbol{k}}(\boldsymbol{x})$ such that

$$\sum_{\boldsymbol{k}\in\mathcal{I}_N^q} b_{\boldsymbol{k}}(H\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = \sum_{\boldsymbol{k}\in\mathcal{I}_N^q} E(\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) \quad \forall \quad \boldsymbol{l} \in \mathcal{I}_N^q. \tag{3.3}$$

Since $\{S_k(x)\}$ are orthonormal in $L^2(\mathbb{R}^d)$, the right-hand side is an identity matrix. The left-hand side can be split into three terms:

$$(T\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}\left(\partial_{x_i}\phi_{\boldsymbol{k}}(\boldsymbol{x}), \partial_{x_i}\phi_{\boldsymbol{l}}(\boldsymbol{x})\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}\frac{\delta_{\boldsymbol{k},\boldsymbol{l}}}{\delta_{k_i,l_i}}s_{k_i,l_i},$$

$$(U\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = Z\sum_{i=1}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}(|x_i|\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = Z\sum_{i=1}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}\frac{\delta_{\boldsymbol{k},\boldsymbol{j}}}{\delta_{k_i,l_i}}u_{k_i,l_i},$$

$$(V\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = -\sum_{i=1}^{N}\sum_{j>i}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}(|x_i - x_j|\phi_{\boldsymbol{k}}, \phi_{\boldsymbol{l}}) = -\sum_{i=1}^{N}\sum_{j>i}^{N}\sum_{\boldsymbol{k}\in\mathcal{I}_N^q}\frac{\delta_{\boldsymbol{k},\boldsymbol{j}}}{\delta_{k_i,l_i}\delta_{k_j,l_j}}v_{k_i,l_i;k_j,l_j},$$

where $s_{k,j}$ is given by (2.19),

$$u_{k,l} = \int_{\mathbb{R}} |x|\phi_k(x)\phi_l(x)\mathrm{d}x, \tag{3.4}$$

and

$$v_{k_1,l_1;k_2,l_2} = \int\int |x - y|\phi_{k_1}(x)\phi_{k_2}(y)\phi_{l_1}(x)\phi_{l_2}(y)\mathrm{d}x\mathrm{d}y. \tag{3.5}$$

The main difficulty in solving (3.3) is that both $(u_{k,l})$ and $(v_{k_1,l_1;k_2,l_2})$ are dense matrices. While they can be accurately precomputed by using some special tricks and numerical quadratures, the cost of applying these dense transform matrices makes it prohibitively expensive. Hence, instead of (3.3), we shall consider its pseudospectral Galerkin version: Find $\psi_N^q = \sum_{\boldsymbol{k}\in\mathcal{I}_N^q} b_{\boldsymbol{k}}\phi_{\boldsymbol{k}}(\boldsymbol{x})$ such that

$$(T\psi_N^q + \mathcal{U}_N^q\left[(U + V)\psi_N^q\right], \phi_{\boldsymbol{l}}) = E(\psi_N^q, \phi_{\boldsymbol{l}}) \quad \forall \quad \boldsymbol{l} \in \mathcal{I}_N^q, \tag{3.6}$$

where $\mathcal{U}_N^q$ is the interpolation operator based on the sparse grid $\mathcal{X}_N^q$ defined in (2.9). Namely, the terms involving $\{u_{k,l},\ v_{k_1,l_1;k_2,l_2}\}$ are replaced by

$$\tilde{b}_{\boldsymbol{l}} = \left(\mathcal{U}_N^q\left[(U + V)\,\psi_N^q\right], \phi_{\boldsymbol{l}}\right).$$

The above can be efficiently computed by using the approach presented in the last section for problems with variable coefficients, similar to the computation for (2.32).

The above pseudospectral approach leads to a nonsymmetric eigenvalue problem, so we shall employ the Arnoldi method, which only requires us to compute the matrix-vector products. However, direct application of the Arnoldi method to the linear eigenvalue problem (3.6) converges very slowly. To speed up the convergence, we

adopt a shift-invert procedure [8] to compute the smallest eigenvalues. In the shift-invert procedure, we need to solve (invert) the shifted problem

$$(H - \delta I)\psi(x_1, x_2, \ldots, x_N) = f(x_1, x_2, \ldots, x_N) \tag{3.7}$$

with the shifting constant δ being as close to the desired eigenvalue as possible. In practice, we adaptively choose δ starting with a coarse-grid (with small q) approximation of the eigenvalue.

The above equation is solved by using a BICGSTAB method with the following preconditioner:

$$\left\{ -\frac{1}{2}\sum_{i=1}^{N} \partial_{x_i}^2 + \beta Z \sum_{i=1}^{N} \xi^2(x_i) \right\}. \tag{3.8}$$

In the above, the scalar number β is a stretching parameter that depends on the dimension of the problem and the discrete grid size. $\xi(x) = \tanh(x)$ is the invert mapping function that maps $x \in \mathbb{R}$ to $\xi \in [-1, 1]$. When discretized using the Galerkin method with mapped Chebyshev bases, this preconditioning operator leads to a very sparse algebraic system, which can be solved efficiently by using a direct solver.

3.2. Numerical results. In Tables 3–6, we present numerical results for the electronic Schrödinger equation with one, two, six, and eight electrons in one spatial dimension by using the mapping (2.14) with $r = 0$ and a scaling parameter L. All results, except the last case with six and eight electrons, are carried out on a Thinkpad T410s with 4 GB RAM and one Intel i5 Core at 2.4 GHz. For example, the case $N = 6, q = 12$ with 1.4 million degree of freedoms took less than 3 hours.

It is known that the eigenfunctions of the electronic Schrödinger equation decay exponentially as $|x|$ goes to infinity. Hence, we choose to use the mapping (2.14) with $r = 0$, which is more suitable for such solutions. It is also well known that a properly chosen scaling parameter L can significantly improve the accuracy with a given number of unknowns. However, there is no obvious way to find the optimal scaling parameter without prior knowledge on the decay rate of the solution. We list in Tables 3–6 numerical results with four different scaling parameters.

We observe from Tables 3–6 that as q increases, the first eigenvalue always increases with $L = 1$ and decreases with $L = 0.5$. So the value with $L = 1$ (resp., $L = 0.5$) provides a lower (resp., upper) bound for the first eigenvalue. For $L = 0.6$

TABLE 3
Smallest eigenvalue of the electronic Schrödinger equation with one electron in one spatial dimension.

q	DoF	$L = 0.5$	$L = 0.6$	$L = 0.75$	$L = 1$
1	3	2.338246	1.766016	1.190107	0.68917
2	5	1.404767	1.154315	0.932692	0.77458
3	9	1.060249	0.934986	0.842216	0.79641
4	17	0.908796	0.848740	0.814735	0.80484
5	33	0.844709	0.819171	0.809213	0.807628
6	65	0.820123	0.810868	0.8085893	0.808368
7	129	0.811832	0.809005	0.8085895	0.808554
8	257	0.809405	0.8086709	0.8086082	0.808601
9	513	0.808787	0.8086225	0.8086144	0.8086126
10	1025	0.808649	0.8086169	0.8086160	0.8086155

TABLE 4

Smallest eigenvalue of the electronic Schrödinger equation with two electrons in one spatial dimension.

q	DoF	$L = 0.5$	$L = 0.6$	$L = 0.75$	$L = 1$
2	9	4.6887345	3.5337141	2.3803513	1.3783412
3	21	3.0801090	2.5024530	1.9562296	1.5447713
4	49	2.4610299	2.1407780	1.8823848	1.7267759
5	113	2.1449360	1.9621990	1.8462236	1.7944285
6	257	2.0139266	1.9089328	1.8596860	1.8463230
7	577	1.9411353	1.8801002	1.8579695	1.8511776
8	1281	1.9053250	1.8713405	1.8624749	1.8607346
9	2817	1.8838480	1.8657901	1.8622006	1.8610317
10	6145	1.8734093	1.8646394	1.8634796	1.8632008
11	13313	1.8678381	1.8638844	1.8634728	1.8632195
12	28673	1.8655343	1.8639148	1.8638087	1.8637460
13	61441	1.8644735	1.8638533	1.8638094	1.8637472
14	106497	1.8640681	1.8639032	1.8638934	1.8638778
15	188417	1.8639412	1.8639003	1.8638934	1.8638778
16	335873	1.8639259	1.8639160	1.8639144	1.8639104

TABLE 5

Smallest eigenvalue of the electronic Schrödinger equation with six electrons in one spatial dimension.

q	DoF	Memory	$L = 0.5$	$L = 0.6$	$L = 0.75$	$L = 1$
6	729		14.19322	10.62023	7.14268	4.13509
7	3645	1M	11.53182	8.98147	6.56414	4.57501
8	14337	7M	9.84225	8.20117	6.84163	6.28908
9	49761	30M	9.13927	8.08522	7.40965	7.55470
10	159489	158M	8.81094	8.15501	7.88803	8.19936
11	483201	879M	8.65394	8.26734	8.21052	8.36132
12	1403137	5163M	8.57456	8.35870	8.37154	8.44094
13	3940609	16689M	8.52448	8.42065	8.44336	8.46657

TABLE 6

Smallest eigenvalue of the electronic Schrödinger equation with eight electrons in one spatial dimension.

q	DoF	Memory	$L = 0.5$	$L = 0.6$	$L = 0.75$	$L = 1$
8	6561	2M	19.011	14.176	9.525	5.514
9	41553	30M	16.487	12.652	9.076	6.059
10	193185	312M	14.387	11.764	9.591	8.622
11	768609	1621M	13.428	11.723	10.682	10.210
12	2772225	16587M	13.054	11.995	11.601	11.339

and 0.75, we observe that in most cases it also changes monotonically when q is large enough such that the solution enters the "asymptotic range." Therefore, by carefully examining these tables, we can deduce that the exact value of the first eigenvalue for the one-electron, two-electron, six-electron, and eight-electron system should lie between [0.8080160, 0.8080169], [1.8639144, 1.8639259], [8.46657, 8.52448], and [11.339, 13.054], respectively. Hence, by taking the midpoints of these intervals as the best approximations, we find that their relative errors are 5.6×10^{-7}, 3.1×10^{-6}, 3.4×10^{-3}, and 7.0×10^{-2}, respectively, for the one, two, six, and eight electrons.

In Table 7, we list the numbers of outer (ARPack) and inner (BICGSTAB) iterations used for computing the first eigenvalue with one, two, four, and six electrons. Observe that the shift-invert procedure is very effective as the numbers of outer (ARPack) iteration remain low for all cases, particularly for larger q. On the

TABLE 7

Iteration numbers of the MCSG method with $r = 0, L = 1$ for the electronic Schrödinger equation with one, two, four, and six electrons in one spatial dimension: n_1 and n_2 are the numbers of iterations used in ARPack and BICGSTAB, respectively. The tolerance is set to 10^{-12} for $N = 1, 2$, while it is set to 10^{-7} for $N = 4, 6$. The stretching parameter β in the preconditioner is set to $(q - N + 1)\log 2 - \log(\pi/2)$.

$N=1$			$N=2$			$N=4$			$N=6$		
q	n_1	n_2	q	n_1	n_2	q	n_1	n_2	q	n_1	n_2
2	9	6	3	11	22	5	7	27	7	9	37
3	7	10	4	13	28	6	11	34	8	21	49
4	7	10	5	9	31	7	9	41	9	23	80
5	7	12	6	9	31	8	9	44	10	13	110
6	7	13	7	7	33	9	7	51	11	9	128
7	7	14	8	7	36	10	7	56	12	7	141
8	7	15	9	7	36	11	5	69	13	7	148
9	7	16	10	9	37	12	5	74			
10	7	17	11	9	37	13	5	87			
11	7	18	12	9	38	14	7	112			

other hand, the number of inner (BICGSTAB) iterations increases as q increases but remains reasonable for all reported cases.

We now make some comments on our results in relation to that of Griebel and Hamaekers [10]. First, we did not take into account the Pauli exclusion principle, while they did. Hence, the energy for the ground state we obtained is smaller than that reported in [10]. Second, their method is based on a pure Galerkin approach (similar to (3.3)) which leads to a much large number of nonzeroes in the system matrix, while ours is based on a pseudospectral Galerkin approach which is much faster, in terms of number of operations per unknown. Third, for the cases of six and eight electrons, it appears that our method leads to relatively better convergence results. Note that the computational cost of our method can be further reduced if we take into account the Pauli exclusion principle. Thus, our method has potential to be an efficient and feasible approach for solving the electronic Schrödinger equation.

4. Concluding remarks. We developed in this paper the MCSG and MCHSG methods for elliptic equations in $\mathbb{R}^d$. The MCSG method is based on nested, spectrally accurate quadratures and enjoys fast transforms between the values at the sparse grid and the corresponding expansion coefficients. Moreover, the MCSG and MCHSG methods lead to the identity mass matrix and very sparse stiffness matrix for problems with constant coefficients and allow us to construct a matrix-vector product algorithm with quasi-optimal computational cost even for problems with variable coefficients. Furthermore, we proposed several efficient alternatives to solve the resultant linear systems for elliptic equations with constant or variable coefficients, and we presented ample numerical results to demonstrate the efficiency and accuracy of the proposed algorithms.

As an example of applications, we considered the electronic Schrödinger equation which plays a fundamental role in quantum chemistry. The preliminary numerical results indicate that our MCSG method with $r = 0$ is very competitive in solving the electronic Schrödinger equation with a moderate number of electrons in one spatial dimension and has great potential to be a viable approach for solving the electronic Schrödinger equation with a moderate number of electrons in three spatial dimensions.

The algorithms developed in this paper set a solid foundation for solving higher-dimensional problems in unbounded domains. In forthcoming papers, we shall apply

the basic algorithms developed in this paper to more complicated and challenging problems such as the electronic Schrödinger equations with a few electrons in three spatial dimensions and the Boltzmann equation.

REFERENCES

[1] J. P. Boyd, *The optimization of convergence for Chebyshev polynomial methods in an unbounded domain*, J. Comput. Phys., 45 (1982), pp. 43–79.

[2] J. P. Boyd, *Spectral methods using rational basis functions on an infinite interval*, J. Comput. Phys., 69 (1987), pp. 112–142.

[3] J. P. Boyd, *Chebyshev and Fourier spectral methods*, 2nd ed., General Publishing Company, North York, ON, Canada, 2001.

[4] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 1–123.

[5] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, *Algorithm* 887: *CHOLMOD, Supernodal sparse cholesky factorization and Update/Downdate*, ACM Trans. Math. Softw., 35 (2008), pp. 1–14.

[6] 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.

[7] M. E. Fisher and M. N. Barber, *Scaling theory for finite-size effects in the critical region*, Phys. Rev. Lett., 28 (1972), pp. 1516–1519.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, MD, 1989.

[9] V. Gradinaru, *Strang splitting for the time-dependent Schrödinger equation on sparse grids*, SIAM J. Numer. Anal., 46 (2008), pp. 103–123.

[10] M. Griebel and J. Hamaekers, *Sparse grids for the Schrödinger equation*, Math. Model. Numer. Anal., 41 (2007), pp. 215–247.

[11] B.-Y. Guo and J. Shen, *On spectral approximations using modified Legendre rational functions: Application to the Korteweg-de Vries equation on the half line*, Indiana Univ. Math. J., 50 (2001), pp. 181–204.

[12] B.-Y. Guo and Z.-Q. Wang, *Modified Chebyshev rational spectral method for the whole line*, Discrete Contin. Dyn. Syst., (suppl.) (2003), pp. 365–374.

[13] B.-Y. Guo, J. Shen, and Z. Wang, *A rational approximation and its applications to differential equations on the half line*, J. Sci. Comput., 15 (2000), pp. 117–147.

[14] L.-L. Wang, J. Shen, and H. Yu, *Error analysis of orthonormal mapped Chebyshev rational functions in one-dimension and high-dimensions*, in preparation.

[15] Y. Notay, *An Aggregation-Based Algebraic Multigrid Method*, Technical report GANMN 08-02, Universite Libre de Bruxelles, 2008.

[16] J. Shen, *Efficient Chebyshev-Legendre Galerkin methods for elliptic problems*, in Proceedings of ICOSAHOM'95, A. V. Ilin and R. Scott, eds., 1996, pp. 233–240.

[17] J. Shen and L.-L. Wang, *Some recent advances on spectral methods for unbounded domains*, Commun. Comput. Phys., 5 (2009), pp. 195–241.

[18] J. Shen and H. Yu, *Efficient spectral sparse grid methods and applications to high dimensional elliptic problems*, SIAM J. Sci. Comput., 32 (2010), pp. 3228–3250.

[19] S. A Smolyak, *Interpolation and quadrature formulas for the classes w_s^a and e_s^a*, Soviet Math., 1 (1963), pp. 384–387.

[20] T. Tang, *The Hermite spectral method for Gaussian-type functions*, SIAM J. Sci. Comput., 14 (1993), p. 594.

[21] 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.

[22] H. Yserentant, *On the regularity of the electronic Schrödinger equation in Hilbert spaces of mixed derivatives*, Numer. Math., 98 (2004), pp. 731–759.

[23] H. Yserentant, *Sparse grid spaces for the numerical solution of the electronic Schrödinger equation*, Numer. Math., 101 (2005), pp. 381–389.