Advances in Computational Mathematics (2022) 48:71
https://doi.org/10.1007/s10444-022-09988-6

An $O(N)$ algorithm for computing expectation of N-dimensional truncated multi-variate normal distribution II: computing moments and sparse grid acceleration

Chaowen Zheng[1] · Zhuochao Tang[2] · Jingfang Huang[2] · Yichao Wu[3]

Received: 6 March 2022 / Accepted: 29 September 2022
© Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
In a previous paper (Huang et al., *Advances in Computational Mathematics* 47(5):1–34, 2021), we presented the fundamentals of a new hierarchical algorithm for computing the expectation of a N-dimensional function $H(\mathbf{X})$ where $\mathbf{X}$ satisfies the truncated multi-variate normal (TMVN) distribution. The algorithm assumes that $H(\mathbf{X})$ is low-rank and the covariance matrix Σ and precision matrix $A = \Sigma^{-1}$ have low-rank blocks with low-dimensional features. Analysis and numerical results were presented when A is tridiagonal or given by the exponential model. In this paper, we first demonstrate how the hierarchical algorithm structure allows the simultaneous calculations of all the order M and less moments $E(H(\mathbf{X}) = X_1^{m_1} \cdots X_N^{m_N} | a_i < X_i < b_i,\ i = 1, \ldots, N),\ \sum_i m_i \le M$ using asymptotically optimal $O(N^M)$ operations when $M \ge 2$ and $O(N \log(N))$ operations when $M = 1$. These $O(N^M)$ moments are often required in the Expectation Maximization (EM) algorithms. We illustrate the algorithm ideas using the case when A is tridiagonal or the exponential model where the off-diagonal matrix block has rank $K = 1$ and number of effective variables $P \le 2$ for each function associated with a hierarchical tree node. The smaller K and P values allow the use of existing FFT and Non-uniform FFT (NuFFT) solvers to accelerate the computation of the compressed features in the system. To handle cases with higher K and P values, we introduce the sparse grid technique aimed at problems with $K + P \approx 5 \sim 20$. We present numerical results for computing both the moments and higher K and P values to demonstrate the accuracy and efficiency of the algorithms. Finally, we summarize our results and discuss the limitations and generalizations, in particular, our algorithm capability is limited by the availability of mathematical tools in higher dimensions. When $K + P$ is greater than 20, as far as we know, there are no practical tools available for problems with 20 truly independent variables.

Communicated by Zydrunas Gimbutas.

Extended author information available on the last page of the article

Published online: 02 November 2022

Keywords Exponential covariance model · Sparse grids · Hierarchical algorithm · Low-dimensional structure · Low-rank structure · Truncated multi-variate normal distribution

Mathematics Subject Classification (2010) 03D20 · 41A30 · 62H10 · 65C60 · 65D30 · 65T40

1 Introduction

Suppose $\mathbf{X} = (X_1, \ldots, X_N)^T$ follows a multi-variate normal distribution with mean $\mathbf{0}$ and positive definite covariance matrix Σ. We are interested in computing $E(H(\mathbf{X})|a_i < X_i < b_i, i = 1, \ldots, N) = \int_{\mathbf{a}}^{\mathbf{b}} H(\mathbf{x}) \frac{p(\mathbf{x};\Sigma)}{\int_{\mathbf{a}}^{\mathbf{b}} p(\mathbf{x};\Sigma)} d\mathbf{x}$ for a N-dimensional function $H(\cdot)$, where $p(\mathbf{x}, \Sigma)$ denotes the probability density function (pdf) of $\mathbf{X}$, and $\frac{p(\mathbf{x};\Sigma)}{\int_{\mathbf{a}}^{\mathbf{b}} p(\mathbf{x};\Sigma)}$ denotes the pdf of a truncated multi-variate normal (TVMN) distribution, with X_i truncated at the lower limit a_i and upper limit b_i. In this expectation calculation, the essential part is to compute the integral $\phi(\mathbf{a}, \mathbf{b};A) = \int_{\mathbf{a}}^{\mathbf{b}} H(\mathbf{x}) f(\mathbf{x}|A) d\mathbf{x}$ where the matrix A denotes the precision matrix $A = \Sigma^{-1}$ and $f(\mathbf{x}|A) = e^{-\frac{1}{2}\mathbf{x}^T A \mathbf{x}}$. Without loss of generality, we focus on evaluating this integral instead of the exact expectation for the rest of the paper.

In a previous paper [9], we presented the fundamentals of a hierarchical algorithm for computing the integral $\phi(\mathbf{a}, \mathbf{b};A)$ under the assumption that $H(\mathbf{X})$ is low-rank and A has low-rank blocks with low-dimensional features. Numerical results were presented for cases when A is tridiagonal or when it is given by the exponential model. The ranks of the off-diagonal blocks for both cases are $K = 1$ and the number of effective variables for the function associated with each hierarchical tree node is bounded by $P = 2$. The smaller K and P values in [9] simplify the discussions and allow the use of existing FFT and Non-uniform FFT (NuFFT) solvers. Extremely accurate results (allowed by the problem condition number) were derived for dimension N as large as 2048 on a personal desktop computer. These highly accurate results also provide valuable data for validating other high dimensional data analysis tools.

In this paper, we consider the generalization of the algorithm in [9]. Note that the efficient computation of different order moments of TMVN is important in many applications. One example is the Expectation Maximization (EM) algorithms that require the calculation of all the 0^{th}, 1^{st}, and 2^{nd} order moments, i.e., $E(H(\mathbf{X})|a_i < X_i < b_i, i = 1, \ldots, N)$ where $H(\mathbf{X}) \in \mathcal{H} = \{X_1^{k_1} \cdots X_N^{k_N}, \sum_{i=1}^N k_i \le 2,\ 0 \le k_i \le 2\}$. Most existing algorithms become computationally impossible when the dimension N gets larger or when higher accuracy is required (due to the slow convergence of the Monte Carlo methods). Our hierarchical algorithm in [9] has successfully reduced the computational complexity for each individual moment calculation to asymptotically optimal $O(N)$. However as there are a total number of $O(N^2)$ order≤ 2 moments, a total amount of $O(N^3)$ operations are therefore required if each moment is computed individually. One main contribution of this paper is a novel numerical algorithm utilizing the compressible features in the hierarchical tree structure [9] to avoid any redundant computations and reduce the computational

complexity of evaluating all the 0^{th}, 1^{st}, and 2^{nd} order moments from $O(N^3)$ to asymptotically optimal $O(N^2)$. More precisely, for the N-dimensional truncated normal distribution, the resulting algorithm can calculate the mean $E(\mathbf{X}|a_i < X_i < b_i, i = 1, \ldots, N)$ (1^{st} order moments) using $O(N \log(N))$ operations, and can evaluate the covariance matrix $Cov(\mathbf{X}|a_i < X_i < b_i, i = 1, \ldots, N)$ (2^{nd} order moments) using $O(N^2)$ operations. We demonstrate the new algorithm ideas for the cases when A is the tridiagonal or exponential model with rank $K = 1$ for the off-diagonal blocks and number of effective variables $P \leq 2$ for each function associated with a hierarchical tree node. The smaller K and P values allow the use of existing FFT and Non-uniform FFT (NuFFT) solvers. For simplicity, we focus on the zero mean case in this paper but the algorithm can be easily generalized to non-zero mean cases and we skip the details.

For fixed P and K values, our algorithms are asymptotically optimal $O(N)$ to compute the expectation of one low rank function, $O(N \log N)$ to compute all the 1^{st} order moments, and $O(N^M)$ to compute all the order$\leq M$ ($M > 1$) moments. However, when P and K increase, the prefactor of these asymptotically optimal algorithms, when based on the classical FFT techniques, grows exponentially and one quickly encounters the "curse of dimensionality." This is because the FFT requires uniform discretization in each dimension and the number of total sample points in this "full-grid" discretization grows exponentially when the dimension increases. In [9], we studied how to compress and represent the N-dimensional integrand function in the expectation integral as a hierarchy of functions $h_{i,j}(w)$, where i is the node level in the hierarchical tree structure, j is the index of the associated node in level i, and w is the effective variable vector of size no more than P. The second contribution of this paper is the introduction of the sparse grid technique to further take advantage of any compressible features in each function $h_{i,j}(w)$ on the (compressed) hierarchical tree structure. The sparse grid technique uses another hierarchical basis and sparse tensor ideas for each function $h_{i,j}(w)$ with P effective variables [15]. For a function $h(w)$ with reasonable smoothness, when n discretization levels are used in each dimension and define $h = 2^{-n}$, compared with the conventional full-grid methods (i.e., the uniform grids in the FFT discretization) which require $O(\frac{1}{h^P})$ grid points for an accuracy of $O(h^2)$, the sparse grid method only requires $O(h^{-1} \cdot \log(h^{-1})^{P-1})$ grid points for an approximation error of $O(h^2 \log(h^{-1})^{P-1})$. Our numerical experiments show that when properly implemented, the sparse grid based methods have the potential to handle problems with $K + P \approx 5 \sim 20$. However, when $K + P$ is greater than 20, our numerical experiments also reveal that no current tools can analytically handle problems with 20+ truly independent variables, therefore new strategies become necessary to utilize the compressible features and hierarchical structures, e.g., using the Monte Carlo simulations to fight the curse of dimensionality.

We organize this paper as follows. In Section 2, we provide a brief review of the low-rank properties and key ideas in the original hierarchical algorithm in [9]. In Section 3, we describe the generalized algorithm for simultaneously calculating all the 0^{th}, 1^{st}, and 2^{nd} order moments (a total of $O(N^2)$ moments) for the TMVN using asymptotically optimal $O(N^2)$ operations. The scheme can be further generalized to compute all the M^{th} order ($M > 1$) or less moments using asymptotically optimal $O(N^M)$ operations. In Section 4, we discuss the sparse grid technique and how it can be applied to

handle problems with larger $K + P$ values. In Section 5, we present preliminary numerical results to demonstrate the accuracy, stability and computational efficiency of the presented algorithms for computing all the moments and for larger $K + P$ values using the sparse grid technique. Finally in Section 6, we summarize our results and discuss the limitation of our algorithms and possible future research directions.

2 Review of low-rank structures in the hierarchical algorithm

In [9], by observing the low-rank properties of the off-diagonal blocks in the precision matrix A and assuming $H(\mathbf{x})$ is also a low-rank function, a fast hierarchical algorithm is presented to evaluate the N-dimensional integral

$$\phi(\mathbf{a}, \mathbf{b}; A) = \int_{a_1}^{b1} \cdots \int_{a_N}^{b_N} H(\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^T A \mathbf{x}} dx_N \cdots dx_1 \tag{1}$$

using $O(N)$ operations where the prefactor depends on the rank of the off-diagonal blocks and dimension of the corresponding subspace of singular vectors. In this section, we explain different low-rank and low-dimensional structures and how they are utilized in the asymptotically optimal hierarchical algorithm.

The **first assumption** is that the function $H(\mathbf{x})$ has low-tensor-rank, i.e.

$$H(\mathbf{x}) = \sum_{k=1}^{K_H} u_{k,1}(x_1) u_{k,2}(x_2) \cdots u_{k,N}(x_N) = \sum_{k=1}^{K_H} \prod_{n=1}^{N} u_{k,n}(x_n), \tag{2}$$

where K_H is assumed to be a small constant independent of N, and each function $u_{k,n}$ is a univariate and not necessarily continuous function. Under this assumption, the evaluation of Eq. (1) becomes the evaluations of K_H integrals, each of the form

$$\int_{a_1}^{b1} \cdots \int_{a_N}^{b_N} u_{k,1}(x_1) u_{k,2}(x_2) \cdots u_{k,N}(x_N) e^{-\frac{1}{2}\mathbf{x}^T A \mathbf{x}} dx_N \cdots dx_1. \tag{3}$$

We further mention that all the moment functions $x_1^{k_1} \cdots x_N^{k_N}$ have low-tensor-rank (=1), and without such a low-rank assumption, the sampling of an arbitrary function $H(\mathbf{x})$ with N truly independent variables is computationally impossible for large N due to the curse of dimensionality.

Our **second assumption** is that when the variables $\mathbf{x}_n$ are properly clustered, the correlations between different clusters are low-rank. In [9], two specific examples are studied in detail. In the first example, the precision matrix A is tridiagonal and the quadratic form is given by

$$\mathbf{x}^T A \mathbf{x} = a_{1,1} x_1^2 + \cdots + a_{k,k} x_k^2 + 2 a_{k,k+1} x_k x_{k+1} + a_{k+1,k+1} x_{k+1}^2 + \cdots + a_{N,N} x_N^2.$$

Note that the two sets of variables $[x_1, \ldots, x_k]$ and $[x_{k+1}, \ldots, x_N]$ are only "weakly" coupled in the integrand through the term

$$(a_{k,k+1} + a_{k+1,k})x_k x_{k+1} = 2a_{k,k+1}x_k x_{k+1}.$$

In matrix language, each off-diagonal matrix block of the tridiagonal matrix A only contains one non-zero number, either at the lower-left or upper-right corner of the matrix block, therefore all the off-diagonal matrices have rank 1. In the second example, the matrix A is defined by the *exponential covariance function*

$$A_{i,j} = e^{-|z_i - z_j|/\beta}, \beta > 0 \tag{4}$$

where each variable $\mathbf{x}_i$ is observed at a location z_i (e.g., in temporal or spatial statistics) and β is a parameter controlling how the correlations between two different locations decay. Utilizing the separation of variables

$$e^{-|z-y|} = \begin{cases} e^{-z}e^{y}, & z \geq y \\ e^{z}e^{-y}, & z < y, \end{cases}$$

one can see that all the off-diagonal blocks of the exponential matrix A have rank 1. Similar to the tridiagonal case, the two sets of variables $\mathbf{x}_1 = [x_1, \ldots, x_k]^T$ and $\mathbf{x}_2 = [x_{k+1}, \ldots, x_N]^T$ are only "weakly" coupled through the term

$$\mathbf{x}_1^T \mathbf{v}\lambda \mathbf{u}^T \mathbf{x}_2 + \mathbf{x}_2^T \mathbf{u}\lambda \mathbf{v}^T \mathbf{x}_1 = 2\mathbf{x}_1^T \mathbf{v}\lambda \mathbf{u}^T \mathbf{x}_2$$

where λ is the (scalar) singular value, and $\mathbf{u}$ and $\mathbf{v}$ are respectively the left and right singular vectors of the rank-1 off-diagonal matrix block.

One key idea in the hierarchical algorithm is to introduce one new t-variable to decouple the weakly coupled x-variables using the formula

$$\sqrt{2\pi}e^{-\frac{1}{2}(x+y)^2} = \int_{-\infty}^{\infty} e^{it(x+y)}e^{-\frac{1}{2}t^2}\mathrm{d}t = \int_{-\infty}^{\infty} e^{itx}e^{ity}e^{-\frac{1}{2}t^2}\mathrm{d}t. \tag{5}$$

Note that the coupling term xy implicitly shown on the left side becomes decoupled $e^{itx}e^{ity}$ on the right side. We explain some details of this decoupling process in the hierarchical algorithm for the tridiagonal case. If we apply "completing the square" to rewrite the coupling term $2a_{k,k+1}x_k x_{k+1}$ as $(\frac{a_{k,k+1}}{\gamma}x_k + \gamma x_{k+1})^2$ in the integral, we get

$$\begin{aligned} \phi(\mathbf{a}, \mathbf{b}; A) &= \int_{\mathbf{a}}^{\mathbf{b}} e^{-\frac{1}{2}\left(a_{1,1}x_1^2 + \cdots + a_{k,k}x_k^2 + 2a_{k,k+1}x_k x_{k+1} + a_{k+1,k+1}x_{k+1}^2 + \cdots + a_{N,N}x_N^2\right)}\mathrm{d}\mathbf{x} \\ &= \int_{\mathbf{a}}^{\mathbf{b}} e^{-\frac{1}{2}\left(\cdots + (a_{k,k} - (\frac{a_{k,k+1}}{\gamma})^2)x_k^2 + (\frac{a_{k,k+1}}{\gamma}x_k + \gamma \cdot x_{k+1})^2 + (a_{k+1,k+1} - \gamma^2)x_{k+1}^2 + \cdots\right)}\mathrm{d}\mathbf{x} \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-t^2} h_{1,1}(t) h_{1,2}(t) \mathrm{d}t \end{aligned}$$

where we applied Eq. (5) in the last step, and $h_{1,1}(t)$ and $h_{1,2}(t)$ are single t-variable functions respectively given by

$$h_{1,1}(t) = \int_{a_1}^{b_1} \cdots \int_{a_k}^{b_k} e^{-\frac{1}{2}\left(a_{1,1}x_1^2+\cdots(a_{k,k}-(\frac{a_{k,k+1}}{\gamma})^2)x_k^2+2i\frac{a_{k,k+1}}{\gamma}x_k t\right)} \mathrm{d}x_k \cdots \mathrm{d}x_1,$$
$$h_{1,2}(t) = \int_{a_{k+1}}^{b_{k+1}} \cdots \int_{a_N}^{b_N} e^{-\frac{1}{2}(2i\gamma x_{k+1}t+(a_{k+1,k+1}-\gamma^2)x_{k+1}^2+\cdots+a_{N,N}x_N^2)} \mathrm{d}x_N \cdots \mathrm{d}x_{k+1}.$$

In the formulas, γ is an "optimized" algorithm parameter so that the children's matrices representing the quadratic forms of

$$a_{1,1}x_1^2 + \cdots (a_{k,k} - (\frac{a_{k,k+1}}{\gamma})^2)x_k^2 \text{ and } (a_{k+1,k+1} - \gamma^2)x_{k+1}^2 + \cdots + a_{N,N}x_N^2$$

are also positive definite, respectively. To evaluate $\int_{-\infty}^{\infty} e^{-t^2} h_{1,1}(t)h_{1,2}(t)\mathrm{d}t$, because of the rapid decay of the weight function e^{-t^2}, one only needs to accurately approximate the functions $h_{1,1}(t)$ and $h_{1,2}(t)$ in a finite interval. Similar to [9], a filter function *filter*(t) (e.g., "top-hat" or "bell" functions) is introduced such that *filter*$(t)h(t) = h(t)$ when $t \in [-7, 7]$ and *filter*$(t)h(t)$ decays smoothly to zero when $|t| \to 14$. The Fast Fourier Transform (FFT) is then applied to derive the Fourier series expansion of *filter*$(t)h(t)$ for $t \in [-14, 14]$. The integral

$$\int_{-\infty}^{\infty} e^{-t^2} h_{1,1}(t)h_{1,2}(t)\mathrm{d}t \approx \int_{-14}^{14} e^{-t^2} \mathit{filter}(t)h_{1,1}(t)h_{1,2}(t)\mathrm{d}t$$

is then evaluated analytically. The same divide-and-conquer process, filter function, and integral evaluation strategy can be applied to the case when A is given by the exponential matrix. We refer interested readers to [9] for the potential theory based analysis and Fourier series based implementation details.

When the rank of the off-diagonal matrix block is K ($K = 1$ for the tridiagonal and exponential cases), a total number of K t-variables are required to decouple the x-variables in the original system. When this strategy is applied recursively in a hierarchical divide-and-conquer setting, the number of t-variables grows rapidly when the number of levels in the hierarchical tree structure increases. To control the number of "effective variables", the **third assumption** in the algorithm is the "low-dimensional" structure of the subspace formed by all the singular vectors from the singular value decompositions of the off-diagonal low-rank matrix blocks. For the two specific examples, the singular vectors are either $\mathbf{u}_i = [1, 0, 0, \ldots, 0]^T$ or $\mathbf{u}_i = [0, 0, \ldots, 0, 1]^T$ for the tridiagonal case, or are combinations of the discretized basis functions e^z and e^{-z} for the exponential case. Therefore the dimension of the subspace is bounded by $P = 2$. More generally, when the off-diagonal covariance function can be well-approximated by a low degree polynomial expansion using the separation of variables, the singular vectors are just the discretized versions of these polynomials, therefore the rank of all the old and new singular vectors (from the decomposition of the off-diagonal matrix blocks) cannot be higher than the number of the polynomial basis functions. Note that the t-variables introduced to decouple the x-variables are always in the form of $\left(\sum_i t_i \mathbf{v}_i\right)^T \mathbf{x}$ or $\left(\sum_i t_i \mathbf{u}_i\right)^T \mathbf{x}$. Using the P independent basis $\boldsymbol{\Phi}_p$, $p = 1, \cdots, P$ for the subspace of singular vectors, we have $\left(\sum_i t_i \mathbf{v}_i\right)^T \mathbf{x} = \left(\sum_{p=1}^{P} w_p \boldsymbol{\Phi}_p\right)^T \mathbf{x}$, i.e., instead of considering the t-variables, one can

consider the effective w-variables and the number of such variables for each function associated with a hierarchical tree node is bounded by P.

The hierarchical algorithm in [9] can be applied to general hierarchical matrices with low-rank off-diagonal structures. Following the Fast Multipole Method terminology, we refer to the coarsest level (with one tree "root" box/node containing all the x-variables) as level 0, and assume each childless tree "leaf" box/node at the finest level l only contains one x-variable. We summarize the algorithm as the following two sweeps. In the downward sweep, by utilizing the low-rank structures in the off-diagonal matrix blocks of the precision matrix A, new t-variables are introduced to decouple the parent node's function/integral as the integral of two child nodes' functions/integrals. The relations between the parent's effective variables $\mathbf{w}_P$, children's effective variables $\mathbf{w}_1$, $\mathbf{w}_2$, and new t-variables t_{new} are computed in the decoupling process and stored for each tree node. At the finest level, for each childless tree node, the single x-variable function is integrated either analytically or numerically in order to construct the approximating function associated with each childless node, each with no more than P effective w-variables. In the upward sweep, an interpolating function $h_p(\mathbf{w}_p)$ is constructed for each parent tree node using its two children's interpolating functions $h_1(\mathbf{w}_1)$ and $h_2(\mathbf{w}_2)$, by integrating the t_{new} variables and using the relations between the parent's and children's effective variables. The expectation in Eq. (1) is simply the function value at the root node. We leave the discussions of additional algorithm details for general matrix A with low-rank off-diagonals and low-dimensional features to Section 4. We demonstrate the algorithm structure for the specific tridiagonal case with $N = 8$ in Fig. 1 (also see [9]). We assume $H(\mathbf{x}) = 1$, all the diagonal entries of the precision matrix A are $a_{i,i} = 4$, and all the lower/upper off-diagonal entries are given by -2.

For the N-dimensional integration problem in Eq. (1), when the divide-and-conquer strategy is applied, the number of levels in the hierarchical tree structure is approximately $O(\log(N))$ and number of tree nodes is $O(N)$. Under the three low-rank and low-dimension assumptions, the total amount of work for each tree node is a constant which depends on the rank K of the off-diagonal matrix block and number P of the effective variables. Therefore the overall algorithm complexity for computing the N-dimensional integral in Eq. (1) is asymptotically optimal $O(N)$. In the

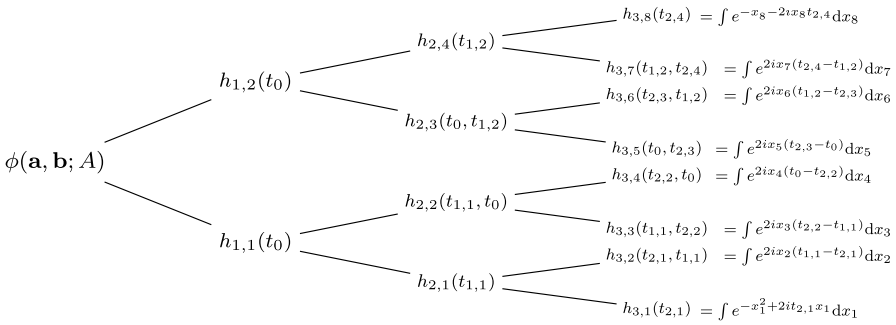

Fig. 1 A three-level hierarchical decomposition of the 8-dimensional integral

numerical results presented in [9], when the rank K of the off-diagonal blocks is one and dimension P of the singular vector space is bounded by two, extremely accurate results (allowed by the problem condition number) were derived for dimension N as large as 2048 on a personal desktop computer.

Finally, we comment on the generality of the three assumptions. It is interesting to first cite the result in [17] which states that "matrices of (approximate) low rank are pervasive in data science." Indeed, one particular purpose of classification, categorization, or defining pseudo-location and pseudo-distance is to reveal the compressible features between different groups, and the low-rank structure is one of the most well-studied compressible features. In order to be asymptotically optimal $O(N)$, the minimum requirements of the hierarchical algorithm in [9] are (a) the low-rank off-diagonal blocks in matrix A and (b) low-dimensional properties of the singular vector space. The discussions in [9] were restricted to the two specific cases when $K = 1$ and $P = 2$ and to the computation of one particular integral of a given low-rank function $H(\mathbf{x})$. In this paper, we demonstrate how to generalize the hierarchical algorithm to efficiently compute all the order$\leq M$ moments, and to cases when $K + P \approx 5 \sim 20$. As far as we know, there exist no practical tools to handle functions with more than 20 truly independent (cannot be further compressed) variables due to the curse of dimensionality.

3 Computing moments simultaneously

To illustrate how the 0^{th}, 1^{st}, and 2^{nd} order moments can be computed simultaneously, we follow the examples in [9] and consider the cases when A is either tridiagonal or has the same form as the exponential covariance matrix. We consider the computations of the N-dimensional integrals

$$\phi(\mathbf{a}, \mathbf{b}; A) = \int_{a_1}^{b1} \cdots \int_{a_N}^{b_N} H(\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^T A \mathbf{x}} dx_N \cdots dx_1. \tag{6}$$

for $H(\mathbf{x}) = 1$ (0^{th} order), $H(\mathbf{x}) = x_i$ (1^{st} order), and $H(\mathbf{x}) = x_i \cdot x_j$ (2^{nd} order) for $i, j = 1, \cdots, N$. Therefore there are a total number of $1 + N + \frac{N(N+1)}{2}$ different moments. It is always possible to apply the $O(N)$ hierarchical algorithm to each moment individually, and the total number of operations to compute all the $O(N^2)$ moments becomes $O(N^3)$. By avoiding any redundant computations involved in the hierarchical algorithm framework, in the following, we introduce an algorithm which computes all the moments using asymptotically optimal $O(N^2)$ operations.

Notice that the purpose of the downward pass of the hierarchical algorithm is to separate the x-variables and derive the relations between the parent node's h function and effective variables with those of its two children,

$$h_p(\mathbf{w}_p) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-t^2} h_1(\mathbf{w}_1) h_2(\mathbf{w}_2) \mathrm{d}t, \tag{7}$$

where $\mathbf{w}_p$, $\mathbf{w}_1$, and $\mathbf{w}_2$ are the effective variables of the parent, child 1, and child 2, respectively; and their relations are given by

$$\begin{cases} \mathbf{w}_1 &= R_1 \mathbf{w}_p + S_1 t \\ \mathbf{w}_2 &= R_2 \mathbf{w}_p + S_2 t \end{cases} \tag{8}$$

where the transformation matrices R_1, R_2, S_1, and S_2 are constructed so that the children matrices are still positive definite and the algorithm has good stability properties. For the tridiagonal and exponential cases, as the rank of each off-diagonal matrix is 1, only one t-variable is introduced; and as the rank of the singular vectors are no more than 2, the number of effective variables for each function is therefore bounded by 2. For more general problems, the number of the required t-variables to separate the parent's problem is bounded by the rank of the off-diagonal matrices, and the number of effective variables for each function is bounded by the dimension of the subspace formed by all the left (or right) singular vectors. As this downward decomposition process doesn't depend on the function $H(\mathbf{x})$ and is the same for all the moments, therefore the relations in Eqs. (7) and (8) only need to be computed once for all the moment computations.

In the upward pass when computing all the 0^{th}, 1^{st}, and 2^{nd} order moments, if each moment is computed separately, there exist many redundant calculations. For example, consider two moments $H_1(\mathbf{x}) = x_1 x_3$ and $H_2(\mathbf{x}) = x_2 x_3$. Note that only the h-function in the leaf nodes corresponding to the variable x_1 and x_2 and all their ancestors are different in the upward pass, and all the other nodes share the same h functions on the hierarchical tree structure. Therefore a new strategy for the upward pass is to compute each required h function only once and store it until its contributions to parent's functions are all accounted for. The new upward pass for computing all the 0^{th}, 1^{st}, and 2^{nd} order moments is as follows: As the moment function $H(\mathbf{x})$ is always in the form of $H(\mathbf{x}) = x_1^{k_1} \cdots x_N^{k_N}$ and the integrand only contains product of the x-variables, for a leaf node containing x-variable x_i, the integrand's dependency on x_i can only take the form of either 1 (constant function), x_i (linear function), or x_i^2 (quadratic function). Therefore instead of forming one single $h_{leaf}(w^{leaf})$ function in the original hierarchical algorithm, 3 functions can be formed for 1, x_i, and x_i^2, respectively, as shown in Fig. 2, where a 3-level tree with 8 x-variables are presented. For a parent node containing a number j of the x-variables, the total number of functions becomes $1 + j + \frac{j(j+1)}{2}$. Note that each function calculation only requires approximately a constant number of operations using the 2 child functions. For example, to compute the function $h_{2,3}\{x_6^2\}$, one takes two child functions $h_{3,6}\{x_6^2\}$ and $h_{3,5}\{1\}$, and evaluate Eq. (7) using approximately a constant number of operations. Similarly, to compute $h_{1,2}\{x_5 x_8\}$, one uses the child functions $h_{2,4}\{x_8\}$ and $h_{2,3}\{x_5\}$.

The computational complexity can be easily derived using the recursive relation

$$T(l+1) = 2T(l) + c(1 + N + \frac{N(N+1)}{2}) \tag{9}$$

where $N = 2^{l+1}$ is the total number of x-variables and $T(l+1)$ is the total amount of work for a hierarchical tree with $l+1$ levels. In the formula, $2T(l)$ is the amount of work for computing all the order≤ 2 moments for each child box at level 1. Each

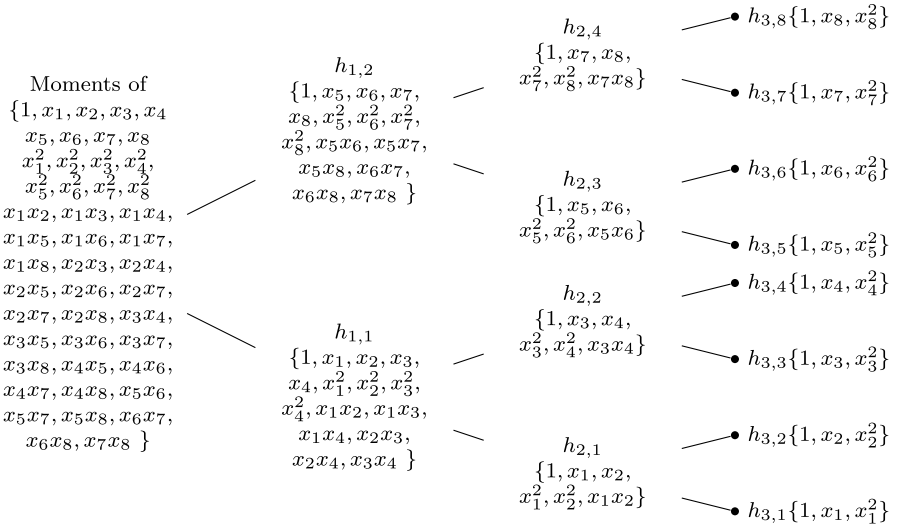

Fig. 2 A three-level hierarchical partition ($N = 8$) to compute all order $M = 2$ or less moments

child box contains $\frac{N}{2} = 2^l$ x-variables and the corresponding sub-tree only has l levels. We also assume each evaluation of Eq. (7) takes approximately a constant number c of operations. Solving the difference equation in Eq. (9), we get the algorithm complexity $O(N^2)$, which is asymptotically optimal to compute the total number of $1 + N + \frac{N(N+1)}{2}$ moments. Note that the algorithm can be generalized to compute all the order$\leq M$ moments in a straightforward way. As there are a total number of $O(N^M)$ moments, the difference equation becomes

$$T(l + 1) = 2T(l) + c(N^M). \tag{10}$$

The algorithm complexity becomes $O(N^M)$ when $M > 1$ and $O(N \log N)$ when $M = 1$.

4 Generalization for larger *K* and *P* values

In this section, we discuss the generalization of our algorithm for problems with larger K and P values. We first go over the original algorithm for computing one expectation to check where the challenges are and then introduce the sparse grid techniques to address some of these challenges.

4.1 Downward pass and numerical linear algebra tools

The downward pass only involved linear algebra operations which recursively divide the precision matrix A into two child matrices problems. Following the notations in [9] and consider a parent's function

$$h_p(\mathbf{w}_p) = \int_{\mathbf{a}_p}^{\mathbf{b}_p} H_p(\mathbf{x}_p) e^{-\frac{1}{2}\mathbf{x}_p^T A_p \mathbf{x}_p} e^{i\mathbf{w}_p^T E_p \mathbf{x}_p} \mathrm{d}\mathbf{x}_p \tag{11}$$

where $H_p(\mathbf{x}_p)$ is the parent's tensor-rank 1 function (at the root level, $H_p(\mathbf{x}_p) = H(\mathbf{x}) = \prod_{n=1}^{N} u_n(x_n)$), $\mathbf{w}_p$ is the compressed vector of effective variables satisfying the relation $\mathbf{w}_p^T E_p \mathbf{x}_p = \mathbf{t}_p D_p \mathbf{x}_p$, $\mathbf{t}_p$ contains the t-variables inherited from ancestors, D_p describes how $\mathbf{t}_p$ interacts with the $\mathbf{x}_p$-variables, the size P of $\mathbf{w}_p$ is the same as the rank of D_p, A_p has low-rank off-diagonals

$$A_p = \begin{bmatrix} A_{l,1} & A_{l,2} = \mathbf{V}\Lambda\mathbf{U}^T \\ A_{l,3} = \mathbf{U}\Lambda\mathbf{V}^T & A_{l,4} \end{bmatrix}, \mathbf{x}_p = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix},$$

the first index l is the current level of the block matrix, and we assume $rank(\Lambda) = K$. Completing the squares for the quadratic form

$$\begin{aligned} \mathbf{x}^T A_p \mathbf{x} =& \mathbf{x}_1^T (A_{l,1} - \mathbf{V}B^{-1}\Lambda B^{-T}\mathbf{V}^T)\mathbf{x}_1 + \\ & \mathbf{x}_2^T (A_{l,4} - \mathbf{U}B^T\Lambda B\mathbf{U}^T)\mathbf{x}_2 + \\ & (B\mathbf{U}^T\mathbf{x}_2 + B^{-T}\mathbf{V}^T\mathbf{x}_1)^T \Lambda (B\mathbf{U}^T\mathbf{x}_2 + B^{-T}\mathbf{V}^T\mathbf{x}_1), \end{aligned}$$

the **first major computation in the downward pass** is to find a $K \times K$ matrix B such that both the matrices $A_1 = A_{l,1} - \mathbf{V}B^{-1}\Lambda B^{-T}\mathbf{V}^T$ and $A_2 = A_{l,4} - \mathbf{U}B^T\Lambda B\mathbf{U}^T$ in the child problems are positive definite. We learned from the tridiagonal and exponential cases that the choice of B is not unique, therefore the optimal B should "balance" two child problems (i.e., for the exponential A, the midpoint is chosen in the potential theory based analysis). Next, introducing a new vector $\mathbf{t}_{new}$ of dimension K and applying the vector form of the formula in Eq. (5), the parent's function can be decomposed into two child problems as

$$h_p(\mathbf{t}_p) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\mathbf{t}_{new}^T \mathbf{t}_{new}} h_1(\mathbf{t}_1) h_2(\mathbf{t}_2) \mathrm{d}\mathbf{t}_{new},$$

where

$$\begin{aligned} h_1(\mathbf{t}_1) &= \textstyle\int_{\mathbf{a}_1}^{\mathbf{b}_1} H_1(\mathbf{x}_1) e^{-\frac{1}{2}\mathbf{x}_1^T A_1 \mathbf{x}_1} e^{i\mathbf{t}_1^T D_1 \mathbf{x}_1} \mathrm{d}\mathbf{x}_1, \\ h_2(\mathbf{t}_2) &= \textstyle\int_{\mathbf{a}_2}^{\mathbf{b}_2} H_2(\mathbf{x}_2) e^{-\frac{1}{2}\mathbf{x}_2^T A_2 \mathbf{x}_2} e^{i\mathbf{t}_2^T D_2 \mathbf{x}_2} \mathrm{d}\mathbf{x}_2. \end{aligned}$$

In the formulas, $\{\mathbf{a}_1,\mathbf{b}_1\}$ and $\{\mathbf{a}_2,\mathbf{b}_2\}$ are respectively the lower and upper integration bounds of $\mathbf{x}_1$ and $\mathbf{x}_2$. By separating the x-variables, the term $\mathbf{t}_p^T D_p \mathbf{x}_p$ is written as

$$\mathbf{t}_p^T D_p \mathbf{x}_p = \mathbf{t}_p^T D_{p,1} \mathbf{x}_1 + \mathbf{t}_p^T D_{p,2} \mathbf{x}_2,$$

$\mathbf{t}_1 = [\mathbf{t}_p; t_{new}]$ and $\mathbf{t}_2 = [\mathbf{t}_p; t_{new}]$ are respectively the new t-variable sets for child 1 and child 2,

$$\begin{cases} \mathbf{t}_1^T D_1 \mathbf{x}_1 = \mathbf{t}_p^T D_{p,1} \mathbf{x}_1 + \mathbf{t}_{new} \Lambda^{\frac{1}{2}} B^{-T} V^T \cdot \mathbf{x}_1, \\ \mathbf{t}_2^T D_2 \mathbf{x}_2 = \mathbf{t}_p^T D_{p,2} \mathbf{x}_2 + \mathbf{t}_{new} \Lambda^{\frac{1}{2}} B U^T \cdot \mathbf{x}_2, \end{cases} \tag{12}$$

and the tensor-rank 1 functions H_1 and H_2 satisfy the relation $H_p(\mathbf{x}_p) = H_1(\mathbf{x}_1) \cdot H_2(\mathbf{x}_2)$. The **second major computation in the downward pass** is to identify the effective variables of the child problems $\mathbf{w}_1^T E_1 \mathbf{x}_1 = \mathbf{t}_1 D_1 \mathbf{x}_1$ and $\mathbf{w}_2^T E_2 \mathbf{x}_2 = \mathbf{t}_2 D_2 \mathbf{x}_2$, and find the relations between the parent's and children's effective variables in the form

$$\begin{cases} \mathbf{w}_1 = R_1 \mathbf{w}_p + S_1 \mathbf{t}_{new}, \\ \mathbf{w}_2 = R_2 \mathbf{w}_p + S_2 \mathbf{t}_{new}. \end{cases} \tag{13}$$

For both the tridiagonal and exponential cases, the matrix B and relations in Eq. (13) are derived using analysis tools, e.g., the corresponding Green's functions and potential theory based analysis for the exponential case. For more general problems, numerical linear algebra tools have to be applied to find B and the relations numerically. When the problem size N is in the range of thousands or tens of thousands, existing $O(N^3)$ SVD based algorithms are sufficient for all the computations. For extremely large N values, recent low-rank linear algebra tools are available to reduce the total amount of work to $O(N)$, interested readers are referred to some recent results in [7, 8, 18, 19]. Notice that the problem itself becomes ill-conditioned for extremely large N values, we conclude that existing numerical linear algebra tools are sufficient for the downward pass.

4.2 Upward pass and sparse grid method

The **most time-consuming operation** in the algorithm is the construction of the parent's function $h_p(\mathbf{w}_p)$ recursively on the hierarchical tree from its child functions $h_1(\mathbf{w}_1)$ and $h_2(\mathbf{w}_2)$ in the upward pass using the relation

$$\begin{aligned} h_p(\mathbf{w}_p) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\mathbf{t}_{new}^T \mathbf{t}_{new}} h_1(\mathbf{w}_1) h_2(\mathbf{w}_2) \mathrm{d}\mathbf{t}_{new} \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\mathbf{t}_{new}^T \mathbf{t}_{new}} h_1(R_1 \mathbf{w}_p + S_1 \mathbf{t}_{new}) h_2(R_2 \mathbf{w}_p + S_2 \mathbf{t}_{new}) \mathrm{d}\mathbf{t}_{new}. \end{aligned} \tag{14}$$

As all the functions h_p, h_1, and h_2 are nonlinear functions, the fundamental building blocks required are (1) the representations and interpolations (evaluations) of the functions $h_p(\mathbf{w}_p)$, $h_1(\mathbf{w}_1)$, and $h_2(\mathbf{w}_2)$, each with no more than P effective variables, and (2) numerical integration quadrature rules to integrate the product function $h_1(R_1 \mathbf{w}_p + S_1 \mathbf{t}_{new}) h_2(R_2 \mathbf{w}_p + S_2 \mathbf{t}_{new})$ with $K + P$ variables with respect to $\mathbf{t}_{new}$ in order to get the function $h_p(\mathbf{w}_p)$ with P effective variables. For the tridiagonal and exponential cases, as $K = 1$ and $P = 2$, the "windowed" Fourier series representations can be utilized and accelerated by uniform FFT algorithm for the tridiagonal case and by NuFFT for the exponential case [9]. Although higher dimensional (i.e., dimension> 4) FFT solvers can be developed, due to its uniform sampling nature, for a prescribed accuracy requirement, the algorithm complexity increases exponentially as the dimension increases due to the "curse of dimensionality." Therefore the

capability of our hierarchical algorithm is limited by existing numerical tools to handle nonlinear functions with $K + P$ independent variables. We introduce the sparse grid technique to address this bottleneck. We have tested the packages SG^{++} and Matlab based *Sparse Grids Matlab Kit*. In the following discussion, we present the algorithm details using the modules from the *Sparse Grids Matlab Kit*. Interested readers are referred to [1, 11–13, 16] for more references of the sparse grid method and existing packages.

In the algorithm implementation, one starts from the finest level to first construct the sparse grid representation of the function associated with each childless node i, by calling the module

$$adapt = adapt\_sparse\_grid(@(\mathbf{w})leaf_i(\mathbf{w}), other\ parameters)$$

where the structure *adapt* stores the compressed adaptive sparse grid representation of the h function associated with the childless node i, $leaf_i(\mathbf{w})$ is a user provided function for sampling node i's h function (an integral with respect to the x-variable(s)) for the given effective variable(s) $\mathbf{w}$, and *other parameters* include the number of effective variables in the function h and parameters required by the sampling function $leaf_i$ and *adapt_sparse_grid* module. When there is only one x-variable in the childless node, the number of effective variables is one (e.g., see Fig. 1) and the sparse grid representation can be efficiently constructed either numerically or analytically by evaluating the childless node's integral. It is possible to allow more x-variables for a childless node, which results in a function with larger number of effective variables but the number of tree levels (and hence the number of tree nodes) will be reduced.

The most time-consuming operation in the sparse grid accelerated algorithm is the construction of parent's function's sparse grid representation from its children's representations. Assuming two child nodes' sparse grid representations $adapt_1$ and $adapt_2$ of $h_1(\mathbf{w}_1)$ and $h_2(\mathbf{w}_2)$ are available, respectively. To construct the "compressed" sparse grid representation $adapt_p$ of the parent's function $h_p(\mathbf{w}_p)$ with P effective variables, the same sparse grid module

$$adapt_p = adapt\_sparse\_grid(@(\mathbf{w}_p)h_p(\mathbf{w}_p), other\ parameters)$$

is called. In the user provided sampling function $h_p(\mathbf{w}_p)$, one calls the adaptive quadrature integration module

$$y = quadrature\_on\_sparse\_grid(@(\mathbf{t}_{new})f(\mathbf{t}_{new}, \mathbf{w}_p), other\ parameters)$$

for any sampling point $\mathbf{w}_p$. Note that the sparse grid integration module provides the weights and nodes to integrate Eq. (14) with respect to $\mathbf{t}_{new}$. The user provided integrand function f is given by

$$f(\mathbf{t}_{new}, \mathbf{w}_p) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mathbf{t}_{new}^T \mathbf{t}_{new}} h_1(R_1\mathbf{w}_p + S_1\mathbf{t}_{new}) h_2(R_2\mathbf{w}_p + S_2\mathbf{t}_{new})$$

where one applies the sparse grid interpolation module to evaluate the sparse grid representations of the functions $h_1(\mathbf{w}_1)$ and $h_2(\mathbf{w}_2)$ using

$$interpolate\_on\_sparse\_grid(adapt_1, R_1\mathbf{w}_p + S_1\mathbf{t}_{new}, other\ parameters),$$

$$interpolate\_on\_sparse\_grid(adapt_2, R_2\mathbf{w}_p + S_2\mathbf{t}_{new}, other\ parameters)$$

at the interpolation points $\mathbf{w}_1 = R_1\mathbf{w}_p + S_1\mathbf{t}_{new}$ for h_1 and $\mathbf{w}_2 = R_2\mathbf{w}_p + S_2\mathbf{t}_{new}$ for h_2, respectively. As there are a total number of $K + P$ variables in the integrand function f, we say that our algorithm requires numerical tools capable of handling functions with $K + P$ variables.

The sparse grid method [2, 4, 10, 14] utilizes an "optimal" number of interpolation points to sample a multi-variable function. By introducing a new hierarchy of basis functions, the contribution/importance of each basis function or corresponding interpolation points at a certain level is measured. For a prescribed accuracy requirement, the non-important basis (and corresponding interpolation points) are removed from the hierarchical basis set therefore the "sparse grid" strategy can significantly reduce the number of grid points in the representation. For a P–dimensional function $h(w)$ with reasonable smoothness, when n discretization levels are used in each dimension and define the stepsize $h = 2^{-n}$, the conventional full grid methods normally require $O(\frac{1}{h^P})$ grid points for an accuracy of $O(h^2)$. The sparse grid method, by taking advantage of the hierarchical basis, only requires $O(h^{-1} \cdot \log(h^{-1})^{P-1})$ grid points for an approximation error of $O(h^2 \log(h^{-1})^{P-1})$.

Note that the number of operations to construct each $h_p(\mathbf{w}_p)$ is independent of the total number of variables N. Therefore our algorithm is still $O(N)$ to compute one expectation, and $O(N^M)$ to compute all order$\leq M$ moments. However, the prefactor depends on the prescribed accuracy requirement, the K and P values, and existing computational tools for handling multi-variable functions. When $K + P < 5$, existing FFT and other "full-grid" algorithm implementations usually outperform current sparse grid toolboxes for higher accuracy requirements. When $K + P$ is in the range of $5 \sim 20$, as we will demonstrate in Section 5, the sparse grid technique should have the ability to handle these problems where higher dimensional FFT solvers are either unavailable or too expensive. When $K + P > 20$ and for high accuracy requirement, although the sparse grid based hierarchical algorithm is still asymptotically optimal $O(N)$, the prefactor becomes so large that simulation becomes impractical on existing computing platforms for reasonable accuracy requirements. We are currently studying other possible strategies to overcome some of these hurdles, e.g., by studying smaller matrix blocks so the $K + P$ can be lowered, and more promisingly, by coupling the Monte Carlo approach with the divide-and-conquer strategy [3]. Results along these directions will be reported later.

5 Preliminary numerical results

We present the accuracy and efficiency results of the moment computation algorithm and sparse grid accelerated expectation computation algorithm for higher rank and number of effective variable cases in this section.

5.1 Computing all order$\leq M$ moments

We consider the tridiagonal matrix case with $a_{i,i} = 4$ and off-diagonal entries $a_{i-1,i} = a_{i+1,i} = -2$ to demonstrate the performance of the new moment algorithm. Similar to the numerical experiments in [9], we set the integration intervals to $[-1, 1]$ except for $b_1 = 0.5$ and $b_2 = 1$. To demonstrate the accuracy of the algorithm, we present the results for $N = 4$ using different numbers of Fourier expansion terms $2M_f$ for all order $M \leq 2$ moments in Table 1. We only show a few selected moment results for the case $N = 8$ in Table 2. As analytical solutions are not available, reference solutions are computed using Mathematica with settings $PrecisionGoal \rightarrow 30$ and $WorkingPrecision \rightarrow 60$. For the case $N = 4$, the Mathematica computed reference solutions are -0.2858091904005895, 0.3590755261636870, and 0.1604892154110472 for the moments x_1, x_1^2, and x_1x_2, respectively. There were no error messages in the computation. For $N = 8$, the Mathematica computed reference solutions are -0.8268288904371333, 1.6044375355684, and 0.0294672112998 for the moments x_1, x_5^2, and x_2x_7, with the estimated errors of $2.49e-7$, $9.65e-6$, and 0.029, respectively.

For both cases, the moment algorithm results converge as the number of Fourier expansion terms $2M_f$ increases. When $N = 4$, the converged moment algorithm results match the Mathematica results to machine precision. When $N = 8$, the converged moment algorithm results match the Mathematica results in the first 10, 7 and 3 digits, respectively, which agree with the estimated errors. We strongly believe the errors from our moment algorithm are close to machine precision.

Next we present the CPU times (in seconds) for different M_f and N values to demonstrate the algorithm efficiency. In Table 3, we show the CPU time to compute all the order 0 and 1 moments. In Table 4, we present the timing results when all the order≤ 2 moments are computed. The numerical tests are performed on a laptop with Intel i5-8265U CPU @1.8GHz and 8.00GB RAM. The numerical results approximately agree with the theoretical complexity results in Eq. (9). We believe the fluctuations are due to the memory/storage hierarchy in the laptop computer system.

5.2 Sparse grid acceleration for large $K + P$ values

As most existing full-grid algorithm implementations (e.g., FFT) are limited to dimensions ≤ 4 and the complexity grows exponentially as the dimension (number of variables) increases, for problems with higher rank and larger number of effective variables, we adopt the sparse grid (SG) Matlab Kit [1, 11, 13, 16] to handle the

Table 1 Moment algorithm convergence test: $N = 4$

Moment	x_1	x_2	x_3
$M_f = 16$	–0.3041533749565543	0.0462052307715464	–0.0141130411535048
$M_f = 32$	–0.2858091904148540	–0.0159045800148543	–0.0103271430324009
$M_f = 64$	–0.2858091904005896	–0.0159045800045992	–0.0103271430492453
$M_f = 128$	–0.2858091904005894	–0.0159045800045996	–0.0103271430492452
$M_f = 256$	–0.2858091904005894	–0.0159045800045993	–0.0103271430492453
$M_f = 512$	–0.2858091904005894	–0.0159045800045993	–0.0103271430492453
Moment	x_4	x_1^2	x_2^2
$M_f = 16$	–0.0052681407108029	0.3747819132588349	0.7545841147234842
$M_f = 32$	–0.0038865286568660	0.3590755261680768	0.6470624696800552
$M_f = 64$	–0.0038865286629617	0.3590755261636870	0.6470624696728164
$M_f = 128$	–0.0038865286629618	0.3590755261636869	0.6470624696728147
$M_f = 256$	–0.0038865286629618	0.3590755261636869	0.6470624696728169
$M_f = 512$	–0.0038865286629618	0.3590755261636869	0.6470624696728160
Moment	x_3^2	x_4^2	x_1x_2
$M_f = 16$	0.5740966493995562	0.5047959618778686	0.1296632727469494
$M_f = 32$	0.5560872797206770	0.4957514695322755	0.1604892154356083
$M_f = 64$	0.5560872796840923	0.4957514695054667	0.1604892154110471
$M_f = 128$	0.5560872796840937	0.4957514695054663	0.1604892154110471
$M_f = 256$	0.5560872796840932	0.4957514695054664	0.1604892154110470
$M_f = 512$	0.5560872796840937	0.4957514695054664	0.1604892154110471
Moment	x_1x_4	x_2x_3	x_3x_4
$M_f = 16$	0.0230434527037889	0.2508953622362097	0.2130009249755526
$M_f = 32$	0.0240731556685396	0.2581815531307147	0.2066885705421552
$M_f = 64$	0.0240731556700224	0.2581815531349327	0.2066885705556584
$M_f = 128$	0.0240731556700224	0.2581815531349327	0.2066885705556583
$M_f = 256$	0.0240731556700224	0.2581815531349326	0.2066885705556583
$M_f = 512$	0.0240731556700224	0.2581815531349327	0.2066885705556583

Table 2 Moment algorithm convergence test: $N = 8$

Moment	x_1	x_5^2	x_2x_7
$M_f = 16$	–0.8807821578989413	1.6332145157428453	0.0206071438781314
$M_f = 32$	–0.8268288906575754	1.6044370774787666	0.0212923079285230
$M_f = 64$	–0.8268288905688056	1.6044370772116807	0.0212923079369516
$M_f = 128$	–0.8268288905688050	1.6044370772116832	0.0212923079369516
$M_f = 256$	–0.8268288905688048	1.6044370772116823	0.0212923079369516
$M_f = 512$	–0.8268288905688050	1.6044370772116832	0.0212923079369516

Table 3 Moment algorithm efficiency test: *order* $M \leq 1$

N	4	8	16	32
N	4	8	16	32
$M_f = 32$	0.012s	0.096s	0.368s	1.984s
$M_f = 64$	0.017s	0.378s	1.681s	5.368s
$M_f = 128$	0.070s	7.310s	49.783s	203.756s
N	64	128	256	512
$M_f = 32$	3.476s	8.341s	22.007s	72.382s
$M_f = 64$	23.898s	42.072s	154.072s	342.061s
$M_f = 128$	324.646s	786.801s	2518.458s	5259.002s

Table 4 Moment algorithm efficiency test: *order* $M \leq 2$

N	4	8	16	32
$M_f = 32$	0.015s	0.212s	0.876s	3.351s
$M_f = 64$	0.015s	0.724s	3.653s	17.090s
$M_f = 128$	0.106s	12.726s	93.328s	427.916s
N	64	128	256	512
$M_f = 32$	14.456s	53.367s	376.201s	4891.492s
$M_f = 64$	65.927s	275.311s	1971.194s	27025.872s
$M_f = 128$	1374.638s	5670.928s	19130.150s	251696.373s

Table 5 Accuracy test when $K = 1$ and $P = 2$ using FFT/NuFFT and sparse grid

	N	2	4
Tridiagonal case	FFT	1.330383072420054e+00	2.289334215088778e+00
	SG	1.330383072420054e+00	2.289334215088776e+00
Exponential case	NuFFT	2.943767426876201e+00	9.631287915311061e+00
	SG	2.943767426876196e+00	9.631287915313981e+00
8		16	32
6.624246691490003e+00		5.544625397830172e+01	3.884575991340498e+03
6.624246691489994e+00		5.544625397830164e+01	3.884575991340494e+03
1.167505122544801e+02		1.759175095515877e+04	5.404567374129064e+08
1.167505122546464e+02		1.759175095519246e+04	5.404567374135844e+08

functions associated with each tree node. Other toolboxes are also being tested, i.e., the Sparse Grid Toolbox SG^{++} in [12]. We present preliminary results to demonstrate the SG-based algorithm accuracy and efficiency. The goal is to find the limitations of existing sparse grid toolbox accelerated algorithm on a desktop with 24G RAM and Intel Xeon CPU.

In the accuracy test, we compare the SG-based algorithm results with those from the FFT-based algorithms for both the tridiagonal and exponential cases with rank $K = 1$. The parameters are chosen to be the same as the settings in the examples of

[9] (See Tables 3 and 5). We use the adaptive SG algorithm with the max number of sparse grid points $M_s = 50,000$ and error tolerance 10^{-10}. In Table 5, the SG-based results agree with the FFT-based results in at least the first 10 digits. We want to mention that for the low rank and number of effective variables settings $K = 1$ and $P = 2$, FFT and NuFFT algorithms are far more efficient than any sparse grid packages for the same accuracy requirements.

To find the limit of the SG-based algorithm in the efficiency test on today's commonly used desktop computers, we present the CPU time for different K (rank of off-diagonal submatrices), P (number of effective variables), N (dimension of the integral), and M_s (max number of SG grid points) values. The results are shown in Tables 6, 7, and 8. We neglect any results (*N/A*) when the CPU time exceeds 7 days in the tables.

For problem size $N = 32$, when $P = 16$, the hierarchical tree only has two levels, the root level 0 and level 1 after one division, therefore the module to construct parent's sparse grid representation from its children's is only called once; when $P = 8$, the hierarchical tree contains 3 levels and the module is called 3 times; and when $P = 4$, there are 4 levels in the tree structure and the module is called 7 times. The CPU time results in Table 6 approximately show the linear dependency on the number of such module calls. When the max number of sparse grid points M_s increases, the CPU times of most existing sparse grid interpolation and integration algorithms increase rapidly. Our current settings of the M_s values only allow very low accuracy results. How to further improve the efficiency and accuracy of existing sparse grid algorithms is still an ongoing research topic. When the K and P values increase, the computational complexity also increases. Based on the results from Tables 6, 7, and 8, we conclude that for reasonable accuracy requirements on our current desktop computer, existing SG-based algorithm can handle problem sizes of no more than $K = P = 16$.

Table 6 Efficiency test when $K = 4$ and $P = 4$ using sparse grid

M_s	100	500	1000	1500
$N = 8$	68.68s	1956.11s	7764.75s	17970.30s
$N = 16$	370.99s	6170.89s	23358.21s	56564.68s
$N = 32$	931.41s	13876.40s	55285.69s	112808.97s

Table 7 Efficiency test when $K = 8$ and $P = 8$ using sparse grid

M_s	100	500	1000	1500
$N = 16$	148.14s	5009.55s	19249.73s	47800.90s
$N = 32$	1275.78s	20284.20s	80097.09s	193414.454s
$N = 64$	4638.90s	72213.63s	318472.38s	N/A

Table 8 Efficiency test when $K = 16$ and $P = 16$ using sparse grid

M_s	100	500	1000	1500
$N = 32$	252.07s	9510.86s	44462.56s	112095.01s
$N = 64$	10401.89s	196132.82s	N/A	N/A
$N = 128$	35701.34s	N/A	N/A	N/A

6 Summary

In this paper, we demonstrate how a previously developed hierarchical algorithm can compute all the order $M(> 1)$ or less moments using asymptotically optimal $O(N^M)$ operations. We also show how the sparse grid technique can be combined with the hierarchical algorithm to allow the computation of problems with larger rank K and number of effective variables P. Our algorithm can be generalized to cases when the precision matrix A belongs to a class of symmetric positive definite hierarchical matrices ($\mathcal{H}$-matrices) [5, 6] with "low-rank" and "low-dimensional" properties, by numerically finding the relations between the effective variables in the downward pass, and finding the parent's function using its children's in the upward pass. We present preliminary numerical results to demonstrate the accuracy and efficiency properties of our algorithms. Extremely accurate results (allowed by the problem condition number) were derived for dimension N as large as 2048 on a personal desktop computer. These high accuracy results will provide valuable data for validating other high dimensional data analysis tools.

Although our algorithm is asymptotically optimal, its application is still limited by the capabilities of existing tools for handling high dimensional problems. Most existing tools are still facing numerical challenges when a function has more than 20 truly independent variables without any compressible features. We are considering possible strategies to generalize our algorithms, by considering smaller submatrix blocks and "sibling" interactions to reduce the matrix rank and number of effective variables, and by coupling with Monte Carlo simulations. Results along these directions will be reported in the future.

Funding The work of J. Huang was supported by the NSF grant DMS-1821093 and DMS-2012451. Y. Wu was partially supported by the NSF grant DMS-1821171.

Data availability Open source software packages and simulation results are available upon request.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Bäck, J., Nobile, F., Tamellini, L., Tempone, R.: Stochastic spectral galerkin and collocation methods for pdes with random coefficients: a numerical comparison. In: Spectral and high order methods for partial differential equations, pp. 43–62. Springer (2011)
2. Barthelmann, V., Novak, E., Ritter, K.: High dimensional polynomial interpolation on sparse grids. Adv. Comput. Math. **12**(4), 273–288 (2000)
3. Genton, M.G., Keyes, D.E., Turkiyyah, G.: Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities. J. Comput. Graph. Stat. **27**(2), 268–277 (2018)
4. Gerstner, T., Griebel, M.: Numerical integration using sparse grids. Numer Algorithms **18**(3), 209–232 (1998)
5. Hackbusch, W.: A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices. Computing **62**(2), 89–108 (1999)
6. Hackbusch, W., Khoromskij, B.N.: A sparse H-matrix arithmetic. Computing **64**(1), 21–47 (2000)
7. Ho, K.L., Greengard, L.: A fast direct solver for structured linear systems by recursive skeletonization. SIAM J. Sci. Comput. **34**(5), A2507–A2532 (2012)

8. Ho, K.L., Ying, L.: Hierarchical interpolative factorization for elliptic operators: differential equations. Commun. Pure Appl. Math. **69**(8), 1415–1451 (2016)
9. Huang, J., Cao, J., Fang, F., Genton, M.G., Keyes, D.E., Turkiyyah, G.: An o(n) algorithm for computing expectation of n-dimensional truncated multi-variate normal distribution i: fundamentals. Adv. Comput. Math. **47**(5), 1–34 (2021)
10. Nobile, F., Tempone, R., Webster, C.G.: A sparse grid stochastic collocation method for partial differential equations with random input data. SIAM J. Numer. Anal. **46**(5), 2309–2345 (2008)
11. Pflüger, D.M.: Spatially adaptive sparse grids for high-dimensional problems. Journal of Complexity **26**(5), 505–522 (2010)
12. Pflüger, D.M.: Sparse Grid Toolbox *SG*$^{++}$. https://sgpp.sparsegrids.org/ (2022). Accessed 24 Oct 2022
13. Piazzola, C., Tamellini, L.: The Sparse Grids Matlab kit - a Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification. ArXiv. (2022). https://doi.org/10.48550/arXiv.2203.09314
14. Shen, J., Yu, H.: Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems. SIAM J. Sci. Comput. **32**(6), 3228–3250 (2010)
15. Smolyak, S.A.: Quadrature and interpolation formulas for tensor products of certain classes of functions. In: Doklady Akademii Nauk, vol. 148, pp. 1042–1045. Russian Academy of Sciences (1963)
16. Tamellini, L., Piazzola, C., Nobile, F., Sprungk, B., Porta, G., Guignard, D., Tesei, F.: Sparse Grid Matlab Kit. https://sites.google.com/view/sparse-grids-kit (2022). Accessed 24 Oct 2022
17. Udell, M., Townsend, A.: Why are big data matrices approximately low rank? SIAM J. Math. Data Sci **1**(1), 144–160 (2019)
18. Xi, Y., Xia, J., Chan, R.: A fast randomized eigensolver with structured ldl factorization update. SIAM J. Matrix Anal. Appl. **35**(3), 974–996 (2014)
19. Xia, J., Gu, M.: Robust approximate cholesky factorization of rank-structured symmetric positive definite matrices. SIAM J. Matrix Anal. Appl. **31**(5), 2899–2920 (2010)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Chaowen Zheng[1] · Zhuochao Tang[2] · Jingfang Huang[2] · Yichao Wu[3]

✉ Jingfang Huang
huang@email.unc.edu

Chaowen Zheng
czheng6@ncsu.edu

Zhuochao Tang
zctang@email.unc.edu

Yichao Wu
yichaowu@uic.edu

[1] North Carolina State University, Raleigh, USA

[2] University of North Carolina at Chapel Hill, Chapel Hill, USA

[3] University of Illinois at Chicago, Chicago, USA