Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Implementation of Enhanced Multivariance Products Representation for Multiway Arrays

C. Gözükırmızı*[1]

[1]Computer Engineering Department, İstanbul Beykent University, İstanbul, Türkiye

ABSTRACT

Multiway arrays (also called multilinear arrays, tensors, folded arrays) can be decomposed by Enhanced Multivariance Products Representation (EMPR). In this work, Enhanced Multivariance Products Representation for multiway arrays is implemented in Julia programming language. A program for computing all the terms of the EMPR for multiway arrays is written. We have utilized the ITensor library which makes the multiway array operations easier to understand and implement.

Keyword: high dimensional model representation, multidimensional array decomposition, tensor network diagram.

ARTICLE INFO

Article history:
Research paper
Received 25 April 2025
Accepted 8 June 2025
Available online 8 June 2025

AMS subject Classification: 15C04.

1 Introduction

High Dimensional Model Representation (HDMR) is a representation for functions as sum of terms of increasing number of variables. HDMR has been adapted for the discrete case to decompose multiway arrays. Therefore, there is HDMR for functions and also HDMR for multiway arrays. Enhanced Multivariance Products Representation (EMPR) is a generalization of HDMR where there are univariate supports (univariate functions for representing functions and vectors for representing multiway arrays). Introducing these parameters into the finite expansion provides the ability for the truncations to better represent the original function or multiway array.

*Email: cosargozukirmizi@beykent.edu.tr

Within this paper, we limit ourselves to the decomposition of multiway arrays. Therefore, we use HDMR and EMPR to mean HDMR for multiway arrays, and EMPR for multiway arrays respectively. A multiway array is an array and its number of indices can also be called ways. A 2-way array is a matrix, a 1-way array with more than one element is a vector and a 1-way array with one element is a scalar. As an alternative interpretation, a scalar can also be considered as a 0-way array. 3-way arrays and 4-way arrays are also widely used in applications. In general for an n-way array, n can be any natural number. For an n-way array, n represents the number of ways (indices). The range of each index should also be stated. The range of indices should be independent. Triangular indexing where the range of one index depends on another is not within the scope of this paper. Each index may go up to a different number, therefore any hyperprism geometry is allowed. We do not limit ourselves to hypercube.
The motivation of decomposing multiway arrays with EMPR can be various. EMPR has proven useful for machine learning applications to provide measurements to learn from. Also, EMPR truncations are used for lossy compression of data. EMPR has also been used with success to replace missing or corrupt data. In general, EMPR may be considered as a replacement in many situations where other multiway array decomposition algorithms are utilized.

1.1 Literature review

[69] focuses on computational complexity of HDMR for function interpolation. [59] combines plain and logarithmic HDMR. [65] tries to better interpolate functions of multiplicative nature from the data points of the function. [70] is about approximating differential operators by HDMR for numerical solution of ODEs. [68] is important to show the link between the continuous HDMR and discrete HDMR and how discrete HDMR may be used effectively for interpolation problems. [67] combines HDMR and Lagrange interpolation. [15] focuses on fixing missing data. [34] details the decomposition of integral kernels by an EMPR based representation. [20] focuses on decomposing three-way arrays recursively by EMPR truncations. [35] is also on decomposition of integral operators. [66] puts forward an implementation of Hybrid HDMR which is one of the early varieties of HDMR for function representation. [21] proposes matrix supports instead of vector supports for decomposing multiway arrays. [60] details how the support functions change the quality of EMPR truncations. [38] shows how EMPR may be used recursively to obtain a structure similar to singular value decomposition. [39] explains the importance of weight matrices in such decompositions. [37] shows how HDMR for functions behave in infinitely small intervals. [62] combines HDMR with fluctuationlessness theorem so that HDMR components can be approximated easily. [63] approximates functions so that both functions of additive and multiplicative nature are approximated well by their representation truncations. [61] tries to make the truncation at the constancy level, a better representative of the original function. [57] uses fluctuationlessness theorem for component determination and also piecewise approach to better represent the original function in expense of performing HDMR more than one time. [14] focuses on matrix represen-

tation of functions. [2] shows how HDMR can used to solve initial value problems. [19] uses Kronecker products and foldings, unfoldings in the representation. [5] is about how functions can be decomposed using nonproduct type weight functions. [3] focuses on weight, geometry and transformations in HDMR for functions: representing the image of the original function may be preferred in certain situations. [58] proposes a method of optimizing the weight functions. [71] shows how HDMR can be used for rational approximation of functions. [64] shows an application of HDMR for digital image enhancement. [9] is a very early survey of HDMR based representations produced by the group of Metin Demiralp. [4] shows how coordinate transformations can be utilized to easily compute the integrals necessary for computing the HDMR components. [11] shows the rationale behind logarithmic HDMR and how this is related to univariance. [10] shows certain examples of logarithmic HDMR. [12] compares (plain) HDMR and logarithmic HDMR. [13] is an early work focusing on weight optimization in HDMR for functions. [8] is one of the first applications of HDMR to computer vision. [18] details TMEMPR (one of the recursive applications of EMPR) application to vision and how the initial support vector may be improved to get better representation by truncations. [23] is one of the works that focus on application of HDMR to multispectral imaging. [56] uses HDMR and wavelet transform for hyperspectral imaging. [36] is one of the works that lay the ground for EMPR for multiway array decomposition. [24] focuses on image denoising by EMPR.
[26] proposes the use of random sampling for the generation of HDMR terms. [40] details the rationale behind HDMR. [1] focuses on its efficient implementation. [28] proposes an alternative relaxed formulation that improves the efficiency for computing the terms. [31] explains why HDMR is important for experiment design. [32] shows the relation between HDMR and Fourier series. [30] details the HDMR varieties formed by Rabitz group. [29] provides a general framework for ranking the importance of system inputs by HDMR. [27] details the application of HDMR for experiment design. [72] is about a software for finding and visualizing the HDMR for functions. [22] proposes a modified method that is parallelizable by MPI and CUDA.
[44] is the main article for HDMR. HDMR was put forward by Sobol. [48] makes a connection between sensitivity indices and a derivative based measure. [45] discusses the use of Monte Carlo and Quasi Monte Carlo methods for computing the multiple integrals in HDMR. [46] discusses if HDMR truncations are representative of the original function using certain theorems and examples. [42] suggest a rank based expansion. [49] focuses on the relation between Monte Carlo integration and HDMR and uses HDMR for computing an integral. [25] is a review on HDMR. [47] is on derivative based criteria for HDMR. Also, [43, 50, 41, 53, 52, 51, 54] build upon HDMR, providing many examples and explaining the basic philosophy.
[6, 7] are extensive articles on tensor network diagrams. Tensor network diagrams makes diagrammatic representation of multiway array decompositions. [16] explain the ITensor library which is a software that builds upon the idea of tensor network diagrams. [55, 33] show certain applications of ITensor library.

1.2 Novelty of the work

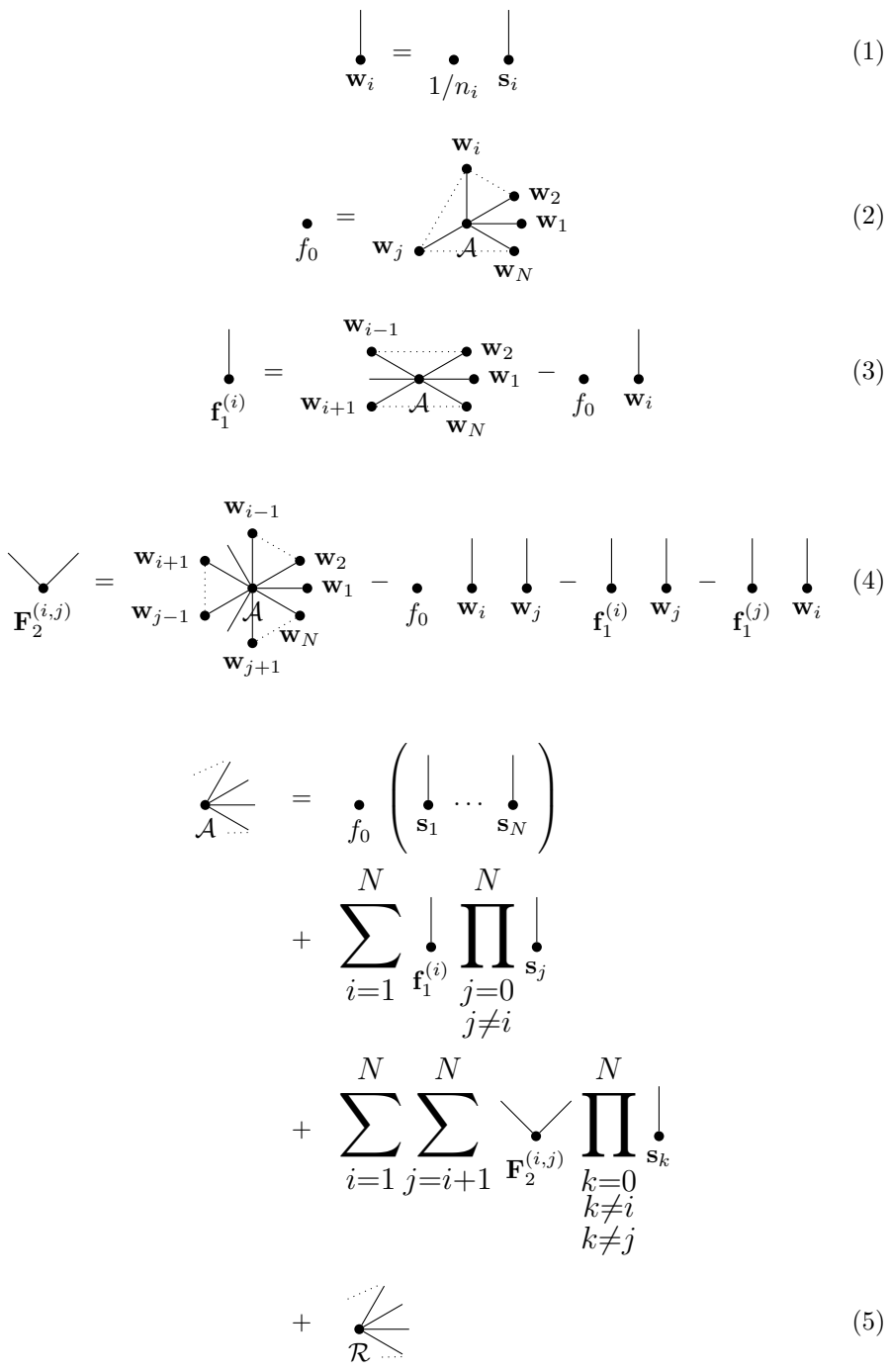
This work looks at Enhanced Multivariance Products Representation (certain generalization of HDMR) from tensor network perspective. The work provides the equalities for computing the components and also the representation itself, using tensor network diagram notation. Within this work, a computer program for Enhanced Multivariance Products Representation (EMPR) for multiway arrays, is written in Julia using ITensor library. This is the first publicly available code for EMPR for multiway arrays [17]. The code is easy to understand and easy to change [17]. The EMPR components are stored only through two indices. Such an approach has made it possible to put a general formula for computing any component.
The algorithm is quite general and is able to compute all EMPR components, truncations, remainders and quality measurers for any multiway array as long as the computational resources allow it.

1.3 Structure of the article

The next section details EMPR from tensor network perspective. Tensor networks are widely used to model decomposition algorithms. In this work, EMPR is explained by this powerful tool. This is a novel look at EMPR. The third section provides implementation details. The difficulties and how it was to possible overcome them are explained. The paper wraps up with the fourth section which emphasizes the important points.

2 Enhanced multivariance products representation for multiway arrays using tensor network diagram notation

Tensor network diagrams provides a way to show multiway array operations in a compact manner [6, 7]. In this work, tensor network diagram notation is combined with mathematical notation to show EMPR. It is also possible to use the tensor network diagram notation by itself, but the combination with mathematical notation becomes more understandable. In our explanation of EMPR, we have used sums. The sums can also be shown within product of multiway arrays just like the fact that an eigendecomposition can both be shown as sum of outer products and product of three matrices.
The operations on multiway arrays are defined on contraction operations. Sums over matching indices determine the result. If there are no matching indices, an outer product is under consideration. A multiway array is shown by a node with edges coming out of it. A node without any edge is a scalar, a node with a single edge is a vector, a node with two edges is a matrix, and so on. Contraction operations are shown by connecting the edges of two multiway arrays. If the multiway arrays are shown next to each other without any edge connected, it is an outer product. Each edge knows its direction, therefore, for a matrix, one of them is for the rows, and the other one is for the columns. Similarly, in



(1)

(2)

(3)

(4)

$$
\begin{aligned}
\mathcal{A} \quad = \quad & f_0 \left(\mathbf{s}_1 \ \cdots \ \mathbf{s}_N \right) \\
& + \sum_{i=1}^{N} \mathbf{f}_1^{(i)} \prod_{\substack{j=0 \\ j \neq i}}^{N} \mathbf{s}_j \\
& + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbf{F}_2^{(i,j)} \prod_{\substack{k=0 \\ k \neq i \\ k \neq j}}^{N} \mathbf{s}_k \\
& + \mathcal{R}
\end{aligned}
\tag{5}
$$

Figure 1: Tensor diagrammatic equalities for enhanced multivariance products representation

the representation of an outer product, the order in which the multiway arrays is written do not make a difference because each edge knows its direction.
The equalities of Enhanced Multivariance Products Representation is given in Figure 1. The right hand side of the first equality of Figure 1 shows the multiplication of a scalar with a vector. The scalar $1/n_i$ is multiplied with the vector $\mathbf{s}_i$. This can also be viewed as the outer product of a scalar with a vector. Here, within the framework of EMPR, $\mathbf{s}_i$ is the support vector with index i. Also, n_i is the number of elements of $\mathbf{s}_i$.
The support vectors play an important role in EMPR. For the rest of the article, we call the multiway array whose EMPR is to be found, as $\mathcal{A}$. For each way of $\mathcal{A}$, there is a corresponding support vector. The number of elements of the support vector is equal to the dimension of the corresponding way of $\mathcal{A}$. For example, if $\mathcal{A}$ is a 2×3 matrix, there are two support vectors: one with 2 elements and the other with 3 elements.
For each support vector, the sum of the squares of the elements should be 1. There are two main approaches for computing the support vectors. One of them, is the High Dimensional Model Representation based support vectors. In this choice, the support vectors are found by normalizing vectors with all elements as 1. EMPR with HDMR based support vectors is not the same representation as HDMR. In EMPR with HDMR based support vectors, the support vectors are found by normalizing vectors with all elements as 1. HDMR can be thought of as relaxation of the unit norm restriction of EMPR and using support vectors with all elements as 1 (without normalizing).
The other approach is to use averaged directional support vectors. In this approach, the way to compute $\mathbf{s}_i$ is as follows. $\mathcal{A}$ is contracted in all directions except i, with vectors whose elements are $\frac{1}{n_i}$ where n_i is the number of elements of the vector. The resulting vector is normalized to yield $\mathbf{s}_i$.
Equation 2 of Figure 1 shows the way to compute f_0. f_0 is the 0-index component of EMPR. It is computed by contracting $\mathcal{A}$ with all of the $\mathbf{w}$ vectors. Contracting all the ways of a multiway array gives a scalar value. If the multiway array to be decomposed is the matrix $\mathbf{A}$, then the way two compute f_0 would be in the form $\mathbf{u}^T\mathbf{A}\mathbf{v}$ where $\mathbf{u}$ and $\mathbf{v}$ are the corresponding $\mathbf{w}$ vectors.
Equation 3 of Figure 1 shows the way to compute $\mathbf{f}_1$ vectors. $\mathbf{f}_1$ are the 1-index components of EMPR. $\mathbf{f}_1^{(i)}$ is computed by contracting $\mathcal{A}$ with all of the $\mathbf{w}$ vectors except $\mathbf{w}_i$ and subtracting the product of f_0 with $\mathbf{w}_i$.
Similarly, Equation 4 of Figure 1 shows the way to compute the $\mathbf{F}_2$ matrices. $\mathbf{F}_2$ are the two-index components of EMPR. If all the representation is under consideration, the last component would be the N-way component of EMPR, where N is the number of ways of the original multiway array. In many applications, EMPR is truncated at 1-index or 2-index level.
Equation 5 of Figure 1 shows how to obtain the original multiway array from the components. The right hand side of Equation 5 is the Enhanced Multivariance Products Representation truncation of $\mathcal{A}$. The terms up to and including the two-index components are taken. The remainder multiway array is shown by $\mathcal{R}$. It is also possible to obtain all of the terms of the EMPR, therefore, not performing any kind of truncation. If that is the case, the last term of the representation is simply the representation so far

subtracted from the original multiway array.

3 Implementation details

Enhanced Multivariance Products Representation for multiway arrays is implemented in Julia programming language using the ITensor library. The pseudocode of the algorithm is given in Algorithm 1. In this section, the pseudocode is explained in detail.
The components of the representation is structured as an *array of array of ITensors*. The outer array is the subscript of f, therefore, 0 is the scalar component, 1 are the vector components and 2 are the matrix components, and so on. The outer array starts with index 0. The inner array starts with index 1. The inner array orders the components for a certain value of the outer array. For example, each matrix component is ordered. The multi-index used for components is made into a single index by this ordering. The multi-index that only uses increasing values of indices as shown in Figure 1, is put into a single index. The increasing indices is shown in Equation 5 within Figure 1, by the lower bounds of the sums.
The way to put the increasing multi-indices into a single index, is performed by the help of the Combinatorics library of Julia. N stores the index information of the multiway array. *length(N)* is the number of ways of the multiway array. The *collectedCombinations* is filled by using *combinations* function from the Combinatorics library. It is an array of arrays. *collectedCombinations[i]* shows all the ways to select i elements from the array [1 ... *length(N)*]. For example, if *length(N)* is 3, *collectedCombinations[2]* is an array with elements as arrays with values [1,2],[1,3],[2,3].
The array *supportVectors* stores the **s** vectors of Figure 1, whereas, the array *supportVectorsDividedByDim* stores the **w** vectors of Figure 1. *supVecProd* is a programmer-defined function that finds the outer product of the **s** vectors whose indices are given in the vector argument of the function.
Line 1 shows the computation of f_0. It is found by the contraction of $\mathcal{A}$ with all the **w**. On the right hand side of the assignment, within the array, the multiplication operation for the contraction is used as a function. The first argument is the multiway array to be decomposed. The ellipsis after the second argument shows that the elements of the array that is before the ellipsis, are made into function arguments in their given order. Therefore, the number of arguments of the multiplication function in Line 1 is 1+*length(N)*. The result of the multiplication is put into an array and assigned to $f[0]$.
Line 2 is a *for* header. The variable *iter1* is for the computing of each subindex of f. For example, when *iter1* is 1, all the vector components are computed. When *iter1* is 2, all the matrix components are computed, and so on. Line 3 makes an assignment to *f[iter1]*. *f[iter1]* is an array that holds the vector components. The right hand side of the assignment in Line 3, uses the comprehension notation. Each iteration of the *for* loop inside the comprehension, makes a new element for the array. Also, the InvertedIndices library is used in order to exclude certain w vectors from the multiplication. When *iter1* is 1, each v would be array with a single element from 1 to *length(N)*. $\mathcal{A}$ is contracted

Algorithm 1: Enhanced Multivariance Products Representation

f[0]←[*(A,supportVectorsDividedByDim...)]
for *iter1←1* **to** *length(N)*
  f[iter1]←[*(A,supportVectorsDividedByDim[Not(v)]...)-
   *(f[0][1],supVecProd(v)) **for** $v \in$ *collectedCombinations[iter1]*];
  for *iter2←1* **to** *(iter1-1)*
    foreach *temp1* $\in$ *allCombs(iter1,iter2)*
      f[iter1]←f[iter1] - [f[iter2][linearizer(iter2, v[temp1])] *
       supVecProd(v[excluder(iter1,temp1)]) **for** $v \in$
       collectedCombinations[iter1]];
    end
  end
end
t[0] ← *(f[0][1],supportVectors...)
for *ind←1* **to** *length(N)*
  t[ind]←t[ind-1]
  $+ \sum_{v \in collectedCombinations[ind]} \Big(*(f[ind][linearizer(ind,v)],$
   supportVectors(Not(v))...) $\Big)$
end
for *i←0* **to** *length(N)*
  qm[i]← $\frac{\text{sum of squares of elements of t[i]}}{\text{sum of squares of elements of A}}$
  rest[i]← A-t[i]
end

with $\mathbf{w}$ excluding the $\mathbf{w}$ with indices shown in v. For the situation when *iter1* is 1, the contribution in a given $\mathbf{f}_1$ produced by f_0 is computed by the outer product of f_0 and the $\mathbf{w}$ whose index is in v and subtracted from the given $\mathbf{f}_1$.
For Line 3, consider the case when *iter1* is 2. Then, the computation of the matrix components are under consideration. $f[2]$ is an array that will hold all the matrix components. Then, v is all the ways to select 2 items from $[1 \ldots length(N)]$. It will start with [1,2]. For this case, $\mathcal{A}$ is contracted with all $\mathbf{w}$ vectors except $\mathbf{w}_1$ and $\mathbf{w}_2$. From the result of the contraction, the outer product of f_0 with $\mathbf{w}_1$ and $\mathbf{w}_2$ is subtracted. The result becomes the first array within $f[1]$.
Line 4 is a *for* header. Line 3 only has the subtraction of subspace contributions resulting from f_0. For $\mathcal{F}_i$, where i is also the number of indices of i, it is necessary to subtract all subspace contributions from 0 to (i-1). The *for* header in Line 4 is for subtracting the subspace contributions from 1 to (i-1), since the contribution from 0 is already subtracted in Line 3. When *iter1* is 1, the *for* body belonging to the header in Line 4 will not execute. When *iter1* is 2, it will execute in order to subtract the contributions in each $\mathbf{F}_2$ resulting from the corresponding $\mathbf{f}_i$ vectors.
The *foreach* header in Line 5, includes a programmer-defined function *allCombs*. The function takes two arguments in the form *allCombs(n,k)*. It shows, in order, all the ways to select k objects from n objects where the objects are numbered from 1 to n. For example, if n is 3 and k is 2, then the result would be an array of arrays with [1,2],[1,3],[2,3].
In Line 6, indexing an array with an array gives an array where the inner array elements become indices to extract elements from the outer array. To see Line 6 in action, assume that the multiway array to be decomposed is a 4-way array. Assume *iter1* is 3 and *iter* 2 is 2. Then v are [1,2,3],[1,2,4],[1,3,4],[2,3,4]. $f[3]$ is an array of ITensors. *allCombs* are [1,2],[1,3],[2,3]. From $f[3]$, an array of ITensors is subtracted.
The *for* loop from Line 10 to Line 14 shows the computation of the representation from the components. The *for* loop from Line 15 to Line 18 shows the computation of quality measurers and the remainders. The quality measurers form an ordered sequence of real nonnegative numbers up to and including 1. The remainders shown by the array *rest*, are found by subtracting EMPR truncations from the original multiway array.

4 Conclusion

A computer program is implemented for decomposing multiway arrays using Enhanced Multivariance Products Representation. The program is written in Julia programming language. The program uses the ITensor library. This is the first publicly available program for Enhanced Multivariance Products Representation for multiway arrays. The use of Julia with ITensor library has made the code easier to implement and more maintainable.
The function in the program is able to decompose any multiway array of floating point numbers. The number of ways of the original array can be any positive integer. However, large multiway arrays would cause problems related to the fact that the multiway arrays

may not fit on computer's memory.
The purpose here is not comparing the EMPR with other commonly used multiway array decomposition algorithms. The purpose is to implement EMPR for multiway arrays as a maintainable code so that its popularity can increase.
The code can also be used in order to test different support vector choices. In the future, different decompositions which are based on EMPR for multiway arrays will also be implemented.

References

[1] Ömer F Alış and Herschel Rabitz. Efficient implementation of high dimensional model representations. *Journal of Mathematical Chemistry*, 29:127–142, 2001.

[2] Nejla Altay and Metin Demiralp. A high dimensional model representation based numerical method for solving ordinary differential equations. *Journal of Mathematical Chemistry*, 49:687–710, 3 2011.

[3] Nasır Abdülbaki Baykara. Weight, geometry and transformation dependence of high dimensional model representation (hdmr). In *MINO'08 Proceedings of the 7th WSEAS International Conference on Microelectronics, Nanoelectronics, Optoelectronics, 29-31 May 2008*, page 17, 2008.

[4] Nasır Abdülbaki Baykara and Metin Demiralp. Hyperspherical or hyperellipsoidal coordinates in the evaluation of high dimensional model representation approximants. In *The Fourth International Conference on Tools For Mathematical Modelling, 23-28 June 2003*, pages 48–62, 2003.

[5] Derya Bodur. High dimensional model representation (hdmr) under nonproduct type weight. *Mathematics in Engineering, Science and Aerospace (MESA)*, 9:141–151, 5 2018.

[6] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5):249–429, 2016.

[7] Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan Oseledets, Masashi Sugiyama, and Danilo P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends in Machine Learning*, 9(6):431–673, 2017.

[8] Emre Demiralp. Applications of high dimensional model representations to computer vision. *WSEAS Transactions on Mathematics*, 8:184–192, 1 2009.

[9] Metin Demiralp. High dimensional model representation and its application varieties. In *The Fourth International Conference on Tools For Mathematical Modelling, 23-28 June 2003*, pages 146–159, 2003.

[10] Metin Demiralp. Illustrative implementations to show how logarithm based high dimensional model representation works for various function structures. *WSEAS Transactions on Computers*, 5:1139–1344, 6 2006.

[11] Metin Demiralp. Importance and measurement of univariance in high dimensional model representation for multivariate analysis. *WSEAS Transactions on Computers*, 5:1738–1744, 9 2006.

[12] Metin Demiralp. Plain and logarithmic high dimensional model representation and the effect of their types on univariance level. *WSEAS Transactions on Mathematics*, 5:582–588, 5 2006.

[13] Metin Demiralp. Weight parameters optimization to get maximum constancy in high dimensional model representation. *WSEAS Transactions on Mathematics*, 5:1117–1181, 5 2006.

[14] Metin Demiralp. No fluctuation approximation in any desired precision for univariate function matrix representations. *Journal of Mathematical Chemistry*, 47:99–110, 1 2009.

[15] Metin Demiralp. Data production for a multivariate function on an orthogonal hyperprismatic grid via fluctuation free matrix representation: Completely filled grid case. *International Journal of Electronics, Electrical and Communication Engineering*, 1:61–76, 2010.

[16] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor Software Library for Tensor Network Calculations. *SciPost Phys. Codebases*, page 4, 2022.

[17] Coşar Gözükırmızı. Enhanced multivariance products representation for multiway arrays using julia.

[18] Coşar Gözükırmızı and Metin Demiralp. The influence of initial vector selection on tridiagonal matrix enhanced multivariance products representation. In *2014 International Conference on Mathematics and Computers in Sciences and in Industry*, pages 182–188, 2014.

[19] Zeynep Gündoğar. Tridiagonal vector enhanced multivariance products representation. *Mathematics in Engineering, Science and Aerospace (MESA)*, 9:153–163, 5 2018.

[20] Zeynep Gündoğar and Metin Demiralp. Certain illustrative numerical implementations of tridiagonal folmat enhanced multivariance products representation (tfempr) for 3-way arrays. *International Journal of Signal Processing*, 1:108–113, 2016.

[21] Zeynep Gündoğar and Metin Demiralp. Block tridiagonal matrix enhanced multivariance products representation (btmempr). *Journal of Mathematical Chemistry*, 56:747–769, 3 2018.

[22] Mehmet Engin Kanal and Metin Demiralp. A modified method of calculating high dimensional model representation (hdmr) terms for parallelization with mpi and cuda. *The Journal of Supercomputing*, 62:199–213, 2012.

[23] Evrim Korkmaz Özay and Burcu Tunga. A novel method for multispectral image pansharpening based on high dimensional model representation. *Expert Systems with Applications*, 170:114512, 2021.

[24] Evrim Korkmaz Özay and Burcu Tunga. Hyperspectral image denoising with enhanced multivariance product representation. *Signal, Image and Video Processing*, 16(4):1127–1133, 2022.

[25] S S Kucherenko and I M Sobol. Global sensitivity indices for nonlinear mathematical models, review. *Wilmott Mag*, 1:56–61, 2005.

[26] Genyuan Li, Maxim Artamonov, Herschel Rabitz, Sheng-Wei Wang, Panos G Georgopoulos, and Metin Demiralp. High-dimensional model representations generated from low order terms–lp-rs-hdmr. *Journal of computational chemistry*, 24:647–656, 4 2003.

[27] Genyuan Li, Caleb Bastian, William Welsh, and Herschel Rabitz. Experimental design of formulations utilizing high dimensional model representation. *The Journal of Physical Chemistry A*, 119(29):8237–8249, 2015.

[28] Genyuan Li and Herschel Rabitz. General formulation of hdmr component functions with independent and correlated variables. *Journal of mathematical chemistry*, 50:99–130, 2012.

[29] Genyuan Li, Herschel Rabitz, Paul E Yelvington, Oluwayemisi O Oluwole, Fred Bacon, Charles E Kolb, and Jacqueline Schoendorf. Global sensitivity analysis for systems with independent and/or correlated inputs. *The journal of physical chemistry A*, 114(19):6022–6032, 2010.

[30] Genyuan Li, Carey Rosenthal, and Herschel Rabitz. High dimensional model representations. *The Journal of Physical Chemistry A*, 105(33):7765–7777, 2001.

[31] Genyuan Li, Sheng-Wei Wang, and Herschel Rabitz. High dimensional model representations (hdmr): Concepts and applications. In *Proceedings of the Institute of*

Mathematics and Its Applicatons Workshop on Atmospheric Modeling, pages 15–19. Citeseer, 2000.

[32] Xiaopeng Luo, Xin Xu, and Herschel Rabitz. On the fundamental conjecture of hdmr: a fourier analysis approach. *Journal of Mathematical Chemistry*, 55:632–660, 2017.

[33] Linjian Ma, Matthew Fishman, Miles Stoudenmire, and Edgar Solomonik. Approximate contraction of arbitrary tensor networks with a flexible and efficient density matrix algorithm. *Quantum*, 8:1580, 2024.

[34] Ayla Okan and Metin Demiralp. Numerical implementations for tridiagonal kernel enhanced multivariance products representation (tkempr) method: Bivariate case. *International Journal of Signal Processing*, 1:102–107, 2016.

[35] Ayla Okan and Metin Demiralp. A self-consistent high dimensional modelling based decomposition approach for univariate linear integral operators: Tridiagonal kernel enhanced multivariance products representation (tkempr). *Journal of Computational and Applied Mathematics*, 326:99–115, 12 2017.

[36] Evrim Korkmaz Özay. A new multiway array decomposition via enhanced multivariance product representation. *AIP Conference Proceedings*, 1479(1):2015–2018, 09 2012.

[37] Evrim Korkmaz Özay and Metin Demiralp. Combined small scale high dimensional model representation. *Journal of Mathematical Chemistry*, 50:2023–2042, 5 2012.

[38] Evrim Korkmaz Özay and Metin Demiralp. Reductive enhanced multivariance product representation for multi-way arrays. *Journal of Mathematical Chemistry*, 52:2546–2558, 11 2014.

[39] Evrim Korkmaz Özay and Metin Demiralp. Weighted tridiagonal matrix enhanced multivariance products representation (wtmempr) for decomposition of multiway arrays: applications on certain chemical system data sets. *Journal of Mathematical Chemistry*, 55:455–476, 2 2017.

[40] Herschel Rabitz and Ömer F Alış. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2):197–233, 1999.

[41] Igor Radović, Ilya M. Sobol, and Robert F. Tichy. Quasi-monte carlo methods for numerical integration: Comparison of different low discrepancy sequences. *Monte Carlo Methods and Applications*, 2(1):1–14, 1996.

[42] Andrea Saltelli and Ilya M Sobol. About the use of rank transformation in sensitivity analysis of model output. *Reliability Engineering and System Safety*, 50(3):225–239, 1995.

[43] I. M. Sobol. A multicriterial interpretation of the method of regularizing ill-posed problems. *USSR Computational Mathematics and Mathematical Physics*, 26(3):100–104, 1986.

[44] I M Sobol. Sensitivity estimates for nonlinear mathematical models. *MMCE*, 1, 1993.

[45] I. M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001. The Second IMACS Seminar on Monte Carlo Methods.

[46] I. M. Sobol. Theorems and examples on high dimensional model representation. *Reliability Engineering and System Safety*, 79(2):187–193, 2003. SAMO 2001: Methodological advances and innovative applications of sensitivity analysis.

[47] I. M. Sobol. On derivative-based global sensitivity criteria. *Mathematical Models and Computer Simulations*, 3:419–423, 2011.

[48] I M Sobol and S. Kucherenko. A new derivative based importance criterion for groups of variables and its link with the global sensitivity indices. *Computer Physics Communications*, 181:1212–1217, 2010.

[49] I. M. Sobol and S. S. Kucherenko. On global sensitivity analysis of quasi-monte carlo algorithms. *Monte Carlo Methods and Applications*, 11(1):83–92, 2005.

[50] I. M. Sobol and E. E. Myshetskaya. Monte carlo estimators for small sensitivity indices. *Monte Carlo Methods and Applications*, 13(5-6):455–465, 2008.

[51] I. M. Sobol, B. V. Shukhman, and A. Guinzbourg. On the distribution of random ranges. *Monte Carlo Methods and Applications*, 5(2):113–134, 1999.

[52] Ilya M. Sobol and Boris V. Shukhman. On global sensitivity indices: Monte carlo estimates affected by random errors. *Monte Carlo Methods and Applications*, 13(1):89–97, 2007.

[53] Ilya M. Sobol and Boris V. Shukhman. Quasi-monte carlo: A high-dimensional experiment. *Monte Carlo Methods and Applications*, 20(3):167–171, 2014.

[54] Ilya M. Sobol and Boris V. Shukhman. Qmc integration errors and quasi-asymptotics. *Monte Carlo Methods and Applications*, 26(3):171–176, 2020.

[55] Joseph Tindall, Matthew Fishman, E. Miles Stoudenmire, and Dries Sels. Efficient tensor network simulation of ibm's eagle kicked ising experiment. *PRX Quantum*, 5:010308, Jan 2024.

[56] Süha Tuna, Evrim Korkmaz Özay, Burcu Tunga, Ercan Gürvit, and M. Alper Tunga. An efficient feature extraction approach for hyperspectral images using wavelet high dimensional model representation. *International Journal of Remote Sensing*, 43(19-24):6899–6920, 2022.

[57] Süha Tuna and Burcu Tunga. A novel piecewise multivariate function approximation method via universal matrix representation. *Journal of Mathematical Chemistry*, 51:1784–1801, 4 2013.

[58] Burcu Tunga and Metin Demiralp. Constancy maximization based weight optimization in high dimensional model representation. *Numerical Algorithms*, 52:435–459, 11 2009.

[59] Burcu Tunga and Metin Demiralp. *A novel hybrid high-dimensional model representation (HDMR) based on the combination of plain and logarithmic high-dimensional model representations*, volume 11, pages 101–111. Springer, 2009.

[60] Burcu Tunga and Metin Demiralp. The influence of the support functions on the quality of enhanced multivariance product representation. *Journal of Mathematical Chemistry*, 48:827–840, 10 2010.

[61] Burcu Tunga and Metin Demiralp. Constancy maximization based weight optimization in high dimensional model representation for multivariate functions. *Journal of Mathematical Chemistry*, 49:1996–2012, 10 2011.

[62] Burcu Tunga and Metin Demiralp. Fluctuation free multivariate integration based logarithmic hdmr in multivariate function representation. *Journal of Mathematical Chemistry*, 49:894–909, 4 2011.

[63] Burcu Tunga and Metin Demiralp. Hybrid hdmr method with an optimized hybridity parameter in multivariate function representation. *Journal of Mathematical Chemistry*, 50:2223–2238, 9 2012.

[64] Burcu Tunga and Aziz Koçanaoğulları. Digital image decomposition and contrast enhancement using high-dimensional model representation. *Signal, Image and Video Processing*, 2017.

[65] Mehmet Alper Tunga and Metin Demiralp. A factorized high dimensional model representation on the nodes of a finite hyperprismatic regular grid. *Applied Mathematics and Computation*, 164:865–883, 5 2005.

[66] Mehmet Alper Tunga and Metin Demiralp. Hybrid high dimensional model representation (hhdmr) on the partitioned data. *Journal of Computational and Applied Mathematics*, 185:107–132, 1 2006.

[67] Mehmet Alper Tunga and Metin Demiralp. A new approach for data partitioning through high dimensional model representation. *International Journal of Computer Mathematics*, 85:1779–1792, 12 2008.

[68] Mehmet Alper Tunga and Metin Demiralp. Bound analysis in univariately truncated generalized high dimensional model representation for random-data partitioning: Interval ghdmr. *Applied Numerical Mathematics*, 59:1431–1448, 6 2009.

[69] Mehmet Alper Tunga and Metin Demiralp. *Computational complexity investigations for high-dimensional model representation algorithms used in multivariate interpolation problems*, volume 11, pages 15–29. Springer, 2009.

[70] İrem Yaman and Metin Demiralp. High dimensional model representation approximation of an evolution operator with a first order partial differential operator argument. *Applied Numerical Analysis and Computational Mathematics*, 1:280–289, 3 2004.

[71] İrem Yaman and Metin Demiralp. A new rational approximation technique based on transformational high dimensional model representation. *Numerical Algorithms*, 52:385–407, 11 2009.

[72] T. Ziehn and A.S. Tomlin. Gui–hdmr – a software tool for global sensitivity analysis of complex models. *Environmental Modelling and Software*, 24(7):775–785, 2009.