J. Nonlinear Var. Anal. 6 (2022), No. 2, pp. 139-154
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.6.2022.2.10

A HOMOTOPY METHOD FOR MULTIKERNEL-BASED APPROXIMATION

YING LIN[1], YIMIN WEI[2], QI YE[3,4,*]

[1]*Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, China*
[2]*School of Mathematical Sciences, Fudan University, Shanghai, China*
[3]*School of Mathematical Sciences, South China Normal University, Guangzhou, Guangdong, China*
[4]*Pazhou Lab, Guangzhou, Guangdong, China*

Abstract. In this paper, we propose several numerical tricks and show some related propositions and theorems to deal with the considerable computation resources required to solve the multilinear system associated with the multikernel-based approximation method. Moreover, a homotopy method is proved to be suitable for the multikernel-based approximation method, including the global convergence and the locally superlinear convergence. We also discuss some problems behind the choice of the truncated length M and the low-rank approximation. Finally, we perform several numerical experiments to evaluate the benefits of the multikernel-based approximation method and the higher speed of the proposed numerical tricks.

Keywords. Homotopy method; Kernel-based approximation method; Multilinear system; Positive definite multikernel; Positive definite tensor.

1. Introduction

The central mathematical problem in many practical applications is reconstructing an unknown function f from the scattered data, which consist of high-dimension sampling points $\mathbf{x} := \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^N$ and the corresponding output values $\boldsymbol{y} := \{y_i \in \mathbb{R}\}_{i=1}^N$ such that $y_i = f(\boldsymbol{x}_i)$ for all $i = 1,2,\dots,N$. The reconstruction is to find an estimate function s to approximate f, aiming to interpolate the scattered data such that $s(\boldsymbol{x}_i) = y_i$ for all $i = 1,2,\dots,N$. It is well-known that the kernel-based approximation method is a fundamental approach of scattered data interpolations [1, 2]. Recently, the classical kernels were generalized to be multikernels with m variables, where m is a positive even integer; see, e.g., [3, 4]. Compared to the kernel matrix and the linear system generated by classical kernel-based approximation, the multikernel-based approximation method generates a kernel tensor $\mathscr{A}_m \in T_{m,N}$ and a corresponding multilinear system is as following

$$\mathscr{A}_m \boldsymbol{x}^{m-1} = \boldsymbol{y}. \tag{1.1}$$

*Corresponding author.

E-mail addresses: ying.lin@connect.polyu.hk (Y. Lin), ymwei@fudan.edu.cn (Y. Wei), yeqi@m.scnu.edu.cn (Q. Ye).

Received November 11, 2021; Accepted December 30, 2021.

Indeed, the multilinear systems of form (1.1) with different kinds of $\mathscr{A}_m$ arise in a number of applications, such as numerical partial differential equations, machine learning, and tensor complementarity problems, etc. [5, 6, 7]

In addition to their applications, the research on multilinear systems also fall in the existence and uniqueness of their solutions, as well as how to solve such a kind of nonlinear systems fast. Recently, several theoretical analysis and algorithms for solving multilinear systems of form (1.1) were presented. For example, Ding and Wei [5] showed the uniqueness of positive solution to multilinear system (1.1) if $\mathscr{A}_m$ is a nonsingular $\mathscr{M}$-tensor, and $\boldsymbol{y}$ is a positive vector. Hu [8] discussed the number of solutions to the general tensor equation, providing a more general condition of the leading tensor for a general tensor equation to have special number of solutions. Wang, Huang and Qi [9] deeply discussed about the global uniqueness and the solvability of tensor variational inequalities (TVI), which provides a new way to show the uniqueness of solutions to a positive definite multilinear system. For (1.1) with symmetric tensors, Li, Xie and Xu [10] proposed a Newton-Gauss-Seidel method, which converges faster than the methods presented in [5]. Xie, Jin and Wei proposed a new tensor method based on rank-1 approximation of the coefficient tensor to solve (1.1) with symmetric $\mathscr{M}$-tensors in [11]. Han [12] proposed a homotopy method to find the unique positive solution to (1.1) with nonsingular $\mathscr{M}$-tensors, and proved its convergence. In [13], the equivalence between solving (1.1) with $\mathscr{M}$-tensors and solving nonlinear systems of equations involving P-functions was presented, as well as a Newton-type method to solve (1.1) with $\mathscr{M}$-tensors. A Levenberg-Marquardt method was applied to solve (1.1) with semi-symmetric coefficient tensor in [14], the authors also presented the global convergence and local quadratic convergence under the local error bound condition. Liu, Du and Chen [15] showed a sufficient descent nonlinear conjugate gradient method for solving (1.1) with $\mathscr{M}$-tensors. More recently, Wang *et al.* developed a kind of accelerated dynamical approaches to find the unique positive solution of (1.1) with a new class of tensors called $\mathscr{K}\mathscr{S}$-tensors in [16]. Compared to traditional numerical methods, Wang, Che and Wei [17] provided a new approach based on continuous time neural networks to solve (1.1) with $\mathscr{M}$-tensors. Similarly, for the general nonsingular multilinear tensor systems, two gradient neural networks models were developed in [18]. For the so-called $\mathscr{S}\mathscr{C}\mathscr{P}$ tensors, Yang, Xu and Huang [19] provided a homotopy method to solve the associated multilinear systems superlinearly.

Nevertheless, most of methods proposed in the above literature are only suitable for $\mathscr{M}$-tensors, while we show the tensor associated with the multikernel-based approximation method is not an $\mathscr{M}$-tensor and cannot be directly transformed to be an $\mathscr{M}$-tensor in Subsection 3.1. Apart from that, those methods for solving general multilinear systems require considerable computation resources. We will show the associated numerical crisis in Subsection 3.1.

In this paper, we first discuss the numerical crisis arise in solving multilinear system related to the multikernel-based approximation method and present a series of numerical tricks to help accelerate the computation. Then, inspired by [19], we prove that the homotopy method therein is suitable for the multilinear system that we consider. We further discuss the choice of the truncation length since it plays a significant role in our numerical tricks and the homotopy method. Finally, several numerical experiments are performed to evaluate the numerical tricks and benefits of the multikernel-based approximation method.

The rest of this paper is organized as follows. In Section 2, we review some necessary concepts and theorems. The associated numerical crisis and main motivations of this paper are shown in Section 3.1. Then we present several numerical tricks and associated propositions and theorems in Section 3.2. The homotopy method and its suitability to solve (3.3) as well as the global convergence and locally superlinear convergence is shown in 3.3. We also dive into the choice of the important coefficient M and other implementation details in Section 3.4. In Section 4, several numerical experiments are performed to evaluate the numerical tricks and benefits of the multikernel-based approximation method.

2. PRELIMINARIES

In this section, we review several necessary concepts, and theorems, including those related to tensors, multilinear systems and the multikernel-based approximation method.

For convenience of readers, the notations and operators of tensors are defined as in the book [20], and without specification, every vector is supposed to be a column vector. We denote $T_{m,N}$ the collection of all m-th order N-th dimensional real tensors, for example, $T_{3,N} = \mathbb{R}^{N\times N\times N}, T_{2,N} = \mathbb{R}^{N\times N}$, and $T_{1,N} = \mathbb{R}^N$. For $\mathscr{A}_m \in T_{m,N}$, it is said to be *symmetric* if all entries $a_{i_1 i_2 \dots i_m}$ of $\mathscr{A}_m$ are invariant under any permutation of the indices. For $\boldsymbol{c} \in \mathbb{R}^N$, the tensor multiplications are defined as

$$\mathscr{A}_m \boldsymbol{c}^{m-2} := \Big(\sum_{i_3,\dots,i_m=1}^{N,N,\dots,N} a_{i_1 i_2 \dots i_m} c_{i_3} \dots c_{i_m} \Big)_{i_1,i_2=1}^{N,N} \in T_{2,N},$$

$$\mathscr{A}_m \boldsymbol{c}^{m-1} := \Big(\sum_{i_2,i_3,\dots,i_m=1}^{N,N,\dots,N} a_{i_1 i_2 \dots i_m} c_{i_2} c_{i_3} \dots c_{i_m} \Big)_{i_1=1}^{N} \in \mathbb{R}^N,$$

and

$$\mathscr{A}_m \boldsymbol{c}^{m} := \sum_{i_1,i_2,\dots,i_m=1}^{N,N,\dots,N} a_{i_1 i_2 \dots i_m} c_{i_1} c_{i_2} \dots c_{i_m} \in \mathbb{R}.$$

It is clear that $\mathscr{A}_m \boldsymbol{c}^m = (\mathscr{A}_m \boldsymbol{c}^{m-1})^\top \boldsymbol{c}$ and $\mathscr{A}_m \boldsymbol{c}^{m-1} = \boldsymbol{c}^\top (\mathscr{A}_m \boldsymbol{c}^{m-2})$. Similar with matrix, we can also define different norms for tensors. Here we introduce the commonly used infinite norm of tensors as $\|\mathscr{A}_m\|_{\max} := \|\mathrm{vec}(\mathscr{A}_m)\|_\infty = \max\{|a_{i_1 i_2 \dots i_m}| : i_1, i_2, \dots, i_m = 1, 2, \dots, N\}$, where $\mathrm{vec}(\mathscr{A}_m)$ is the operator that unfolds the tensor $\mathscr{A}_m$ to a vector.

The definition of (strictly) positive definite tensors is quite important for that of (strictly) positive definite multikernels. For $\mathscr{A}_m \in T_{m,N}$, it is called a *semi-positive definite tensor* if $\mathscr{A}_m \boldsymbol{c}^m \geq 0$, for all $\boldsymbol{c} \in \mathbb{R}^N$. If further $\mathscr{A}_m \boldsymbol{c}^m = 0$ if and only if $\boldsymbol{c} = 0$, then $\mathscr{A}_m$ is said to be a *positive defintie tensor*. Moreover, as same as in [9], we say that $\mathscr{A}_m$ is a *strictly positive definite tensor* if $(\boldsymbol{c} - \boldsymbol{d})^\top \big(\mathscr{A}_m \boldsymbol{c}^{m-1} - \mathscr{A}_m \boldsymbol{d}^{m-1}\big) > 0$ for all $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^N$ with $\boldsymbol{c} \neq \boldsymbol{d}$. The solution sets to the multilinear systems with (strictly) positive definite tensors were discussed in [9, Theorem 4.2] and [3, Proposition 2.1, Proposition 2.2].

We call a tensor $\mathscr{A}_m \in T_{m,N}$ a $\mathscr{Z}$-tensor if all of its off-diagonal entries are nonpositive. A pair $(\lambda, \boldsymbol{x}) \in \mathbb{C} \times (\mathbb{C}^N \backslash \{0\})$ is called an eigenvalue-eigenvector pair (simply eigenpair) of $\mathscr{A}_m \in T_{m,N}$ if it satisfies the system of equations $\mathscr{A}_m \boldsymbol{x}^{m-1} = \lambda \boldsymbol{x}^{[m-1]}$, where $\boldsymbol{x}^{[m-1]}$ means the element-wise $(m-1)$-th power of $\boldsymbol{x}$, that is, $(\boldsymbol{x}^{[m-1]})_i = \boldsymbol{x}_i^{m-1}$. The spectral radius of $\mathscr{A}_m$ is defined by $\rho(\mathscr{A}_m) := \max\{|\lambda| : \lambda \text{ is an eigenvalue of tensor } \mathscr{A}_m\}$. An $\mathscr{M}$-tensor $\mathscr{A}_m \in T_{m,N}$ is a $\mathscr{Z}$-tensor which can be written as $\mathscr{A}_m = t\mathscr{I}_m - \mathscr{B}_m$ with $t \geq \rho(\mathscr{B}_m)$, where $\mathscr{I}_m$ is the m-th order N-th dimensional tensor with all entries being 1, $\mathscr{B}_m$ is a nonnegative tensor. Furthermore, $\mathscr{A}_m$ is called a nonsingular $\mathscr{M}$-tensor if $t > \rho(\mathscr{B}_m)$. The equivalent conditions for a $\mathscr{Z}$-tensor to be a nonsingular $\mathscr{M}$-tensor can be found in [20, Page 10].

A tensor $\mathscr{A}_m \in T_{m,N}$ is called a *completely positive tensor ($\mathscr{CP}$ tensor)* if there exist a positive integer r and $\boldsymbol{u}^k \in \mathbb{R}^n_+$ for all $k = 1,2,\ldots,r$ such that $\mathscr{A}_m = \sum_{k=1}^r (\boldsymbol{u}^k)^{\otimes m}$. If further $\operatorname{span}\{\boldsymbol{u}^k\}_{k=1}^r = \mathbb{R}^N$, then $\mathscr{A}_m$ is called a *strong completely positive tensor ($\mathscr{SCP}$ tensor)*.

Let a domain $\Omega \subseteq \mathbb{R}^d$. By [3, Section 3], a multikernel of order $m \in \mathbb{N}$ is defined as $K_m : \otimes_{k=1}^m \Omega \to \mathbb{R}$ for $m \in \mathbb{R}$. It is called a *symmetric multikernel* if $K_m(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m) = K_m(\boldsymbol{z}_{i_1}, \boldsymbol{z}_{i_2}, \ldots, \boldsymbol{z}_{i_m})$ for all permutation of pairwise distinct indices $i_1, i_2, \ldots, i_m \in \{1,2,\ldots,m\}$. By [3, Definition 3.1], the *semi-positive definite multikernel* and the *(strictly) positive definite* multikernel can be similarly defined by the multiple product.

Given any pairwise distinct points $\{\boldsymbol{x}_i\}_{i=1}^N \subseteq \Omega$, we define

$$\mathscr{A}_m := (K_m(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_m}))_{i_1,i_2,\ldots,i_m=1}^{N,N,\ldots,N} \in T_{m,N}. \tag{2.1}$$

By [3, Proposition 3.2], if K_m is a symmetric (strictly) positive definite multikernel, then $\mathscr{A}_m$ in (2.1) is a symmetric (strictly) positive definite tensor.

To introduce the computation of multikernels based on traditional kernels, we now review the eigenpairs-based representation of positive definite kernels in [4]. As same as in [2, Section 10.4], we suppose that the domain Ω is compact and the symmetric positive definite kernel Φ_2 is continuous. Thus, the Mercer theorem assures that Φ_2 possesses the absolutely uniformly convergence representation

$$\Phi_2(z_1, z_2) = \sum_{k=1}^{\infty} \lambda_k e_k(z_1) e_k(z_2), \text{ for } z_1, z_2 \in \Omega,$$

where $\{\lambda_k : k \in \mathbb{N}\}$ and $e_k : k \in \mathbb{N}$ are the related positive eigenvalues and continuous eigenfunctions of Φ_2, respectively. Let $\phi_k := \lambda_k^{1/2} e_k$ for all $k \in \mathbb{N}$. As same as [4, Theorem 4.2], we further suppose that $\sum_{k=1}^{\infty} |\phi_k(\boldsymbol{x})| < \infty$, for all $\boldsymbol{x} \in \Omega$. Thus, by [4, Theorem 5.10], the above equation assures that the special multikernel

$$\Phi_m(z_1, z_2, \ldots, z_m) := \sum_{k=1}^{\infty} \prod_{i=1}^{m} \phi_k(z_i), \text{ for } z_1, z_2, \ldots, z_m \in \Omega, \tag{2.2}$$

is well-defined for all $m \geq 2$. By [3, Proposition 3.4], if m is a positive even integer, then Φ_m is a symmetric strictly positive definite multikernel. Thus, the formula (2.2) provides a way to compute the multikernel based on the eigenvalues and eigenfunctions of the corresponding standard kernel. For more information, we refer readers to [4].

3. Multikernel-based Approximation via Multilinear Systems

The multikernel-based approximation method can be finally formulated as a multilinear system. Solving this system always encounters numerical issues. In this section, we first discuss the multilinear system associated with the multikernel-based approximation method and its numerical crisis. The crisis are the main motivation for this article. Then we show some numerical tricks to significantly reduce the computation and the related propositions and theorems. Although we shall show the kernel tensors are not $\mathscr{SCP}$ tensors, we still want to apply the homotopy method in [19] owing to its locally superlinear convergence. In Subsection 3.3, we will show the homotopy method is also suitable for our problem. Furthermore, several advanced details about the numerical implementation will be shown in Subsection 3.4, including the low-rank approximation and the choice of M.

3.1. **Numerical crisis and motivations.** Given the scattered dataset $\mathbf{x}$ and corresponding output values $\boldsymbol{y}$, let $\mathscr{A}_m$ be a tensor defined by (2.1) with $K_m = \Phi_m$ in (2.2), and let $B_m(\boldsymbol{x})$ be a tensor function defined as

$$B_m(\boldsymbol{x}) := (\Phi_m(\boldsymbol{x}, \boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \dots, \boldsymbol{x}_{i_{m-1}}))_{i_1, i_2, \dots, i_{m-1}=1}^{N,N,\dots,N} \in T_{m-1,N}, \text{ for } \boldsymbol{x} \in \Omega.$$

Then, $\mathscr{A}_m = (B_m(\boldsymbol{x}_i))_{i=1}^N$.

As in [3, Theorem 4.1], for a positive even integer m, the estimate function $s_m(\boldsymbol{x})$ has the form $s_m(\boldsymbol{x}) := B_m(\boldsymbol{x})\boldsymbol{c}^{m-1}$, for $\boldsymbol{x} \in \Omega$, where the coefficients $\boldsymbol{c} \in \mathbb{R}^N$ are uniquely solved by multilinear system (1.1). By [3, (4.14)], it is believed that as m increases, the performance of the interpolance also improves. In this way, we can interpolant the scattered dataset by the m-th order multikernel-based approximation method via solving a multilinear system.

Generally, we prefer the regularization networks performed in reproducing kernel Hilbert spaces (RKHSs) [21] or reproducing kernel Banach spaces(RKBSs) [4, 22] to help prevent algorithms from over-fitting caused by the noise in real-world applications. The regularization networks in RKBSs corresponding to multikernels are of form

$$\min_{\boldsymbol{c} \in \mathbb{R}^N} \sum_{i=1}^N L(y_i, s_m(\boldsymbol{x}_i)) + \lambda \mathscr{A}_m \boldsymbol{c}^m,$$

However, since it involves tensor computation in both error term and regularization term, it is so far a challenge to solve it directly. Therefore, we need to firstly consider the simpler case, that is the multilinear system without the regularization term.

Pursuing better performance compared to the standard kernel-based approximation method, it is natural and crucial to use multikernels with larger m's to interpolant the scattered dataset. However, the numerical crisis cannot be ignored when m increases. Suppose given m pairwise distinct scattered data points and a vector $\boldsymbol{c}$, we can calculate the value of Φ_m in constant time. Then the computation and storage of $\mathscr{A}_m$, as well as the computation of tensor multiplications, such as $\mathscr{A}_m \boldsymbol{c}^m$ and $\mathscr{A}_m \boldsymbol{c}^{m-1}$, are challenges. Under the constant-time computation assumption, it is easy to induce that the space and computation complexity of $\mathscr{A}_m$ are both $O(N^m)$, and those of $\mathscr{A}_m \boldsymbol{x}^m$ and $\mathscr{A}_m \boldsymbol{x}^{m-1}$ are $O(mN^m)$, which are tremendous numbers for large N and m. While in practice, Φ_m cannot be computed in constant time since it is an infinite sum. Thus, the computation complexity will be much higher. Not to mention the subsequent computation needed to solve the multilinear system, especially the computation of tensor multiplications in each iteration for iterative methods.

As $\mathscr{M}$-tensors are more prevalent, there are more studies focusing on fast methods solving multilinear systems associated with them. But we shall show that the tensor generated by a multikernel is not an $\mathscr{M}$-tensor and cannot be transferred to be an $\mathscr{M}$-tensor directly. Generally, positive definite multikernels are nonnegative functions, namely for any input, the output is nonnegative. So, $\mathscr{A}_m$ is not a $\mathscr{Z}$-tensor and so not an $\mathscr{M}$-tensor. A simple idea is to consider whether $-\mathscr{A}_m$, whose all off-diagonal entries are nonpositive, is an $\mathscr{M}$-tensor or not. If so, we can solve the modified multilinear system $-\mathscr{A}_m \boldsymbol{x}^{m-1} = -\boldsymbol{y}$ rather than the original system. For any positive vector $\boldsymbol{x}$, we have $-\mathscr{A}_m \boldsymbol{x}^{m-1} < 0$ by the definition of tensor multiplications since all entries of $-\mathscr{A}_m$ are nonpositive. By equivalent condition for a $\mathscr{Z}$-tensor to be a nonsingular $\mathscr{M}$-tensor in [20, Page 10], $-\mathscr{A}_m$ is not a nonsingular $\mathscr{M}$-tensor. Although it can still be possible to be an $\mathscr{M}$-tensor which is not nonsingular, it would be undoubtedly difficult to find the significant decomposition. The above statement implies that we cannot solve (1.1) by using those methods for $\mathscr{M}$-tensor equations.

By contrast, for current methods designed for general multilinear systems, in each iteration, either an inverse of a squared matrix of size N is required to compute, or a linear system of size N is needed to solve. For example, in [10], based on a decomposition $\mathscr{A}_m = \mathscr{M}_m - \mathscr{U}_m$, in each iteration, the inverse of a matrix of form $\mathscr{M}_m \boldsymbol{x}^{m-2}$ needs to be calculated. In the case of the multikernel-based approximation method, $\mathscr{A}_m$ is an m-th order N-th dimensional tensor, which means that $\mathscr{M}_m \boldsymbol{x}^{m-2}$ is an N-th order squared matrix, where N is usually a huge number in machine learning. Again for instance, in each iteration of the method proposed in [14], a linear system of size N is required to be solved, which also needs significant computation. Such situations make it difficult to solve (1.1) associated with the multikernel-based approximation method in machine learning using these kinds of iterative methods. Therefore, solving (1.1) may encounter difficulties.

Among those general methods, the homotopy method in [19] is special. Similarly, it also needs to solve a linear system of size N in each iteration, but as a trade-off, some theorems guarantee its locally superlinear convergence, which attracts our interests. The numerical experiments in [19] show that the stop criterion can be reached in only a small number of iterations. However, the tensor generated by a multikernel is not a $\mathscr{SCP}$ tensor. By (2.2), letting $\boldsymbol{u}^k = (\phi_k(\boldsymbol{x}_i))_{i=1}^N$, then $\mathscr{A}_m = \sum_{k=1}^{\infty} (\boldsymbol{u}^k)^{\otimes m}$. But in general, we cannot guarantee the positiveness of $\boldsymbol{u}^k$ for any k. Fortunately, we can use this form of $\mathscr{A}_m$ to apply some numerical tricks and under some mild conditions we can still apply the homotopy method to solve our problem.

3.2. **Numerical tricks.** To reduce the requirements of computation resources of $\mathscr{A}_m$ and the tensor multiplications, we propose a series of numerical tricks. Recall that $\mathscr{A}_m$ is obtained by (2.1) with $K_m := \Phi_m$ defined in (2.2), which is a sum of infinite terms and is absolutely uniformly convergent. Thus, for any $\varepsilon_{i_1 i_2 \ldots i_m} > 0$, there exists $M_{i_1 i_2 \ldots i_m} \in \mathbb{N}$ such that, for any $n \geq M_{i_1 i_2 \ldots i_m}$,

$$\left| \Phi_m(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m) - \sum_{k=1}^{n} \prod_{i=1}^{m} \phi_k(\boldsymbol{z}_i) \right| < \varepsilon_{i_1 i_2 \ldots i_m}, \text{ for } \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m \in \Omega. \tag{3.1}$$

Let $\mathscr{A}_m^M$ be the tensor whose entries are truncated by $M := \max\{M_{i_1 i_2 \ldots i_m}\}$, that is,

$$\mathscr{A}_m^M = \Big(\sum_{k=1}^{M} \prod_{j=1}^{m} \phi_k(\boldsymbol{x}_{i_j})\Big)_{i_1, i_2, \ldots, i_m = 1}^{N, N, \ldots, N}. \tag{3.2}$$

Then, we have $\|\mathscr{A}_m - \mathscr{A}_m^M\|_{\max} < \varepsilon := \min\{\varepsilon_{i_1 i_2 \ldots i_m}\}$ and $\lim_{M \to +\infty} \mathscr{A}_m^M = \mathscr{A}_m$. Therefore, we can use $\mathscr{A}_m^M$ instead of $\mathscr{A}_m$ and consider the multilinear system

$$\mathscr{A}_m^M \boldsymbol{c}^{m-1} = \boldsymbol{y} \tag{3.3}$$

other than (1.1) to deal with the infinite sum and reduce the computation.

The form of $\mathscr{A}_m^M$ can help us accelerate the computation of tensor multiplications. Let H_N^M be a matrix of shape $N \times M$ such that, for any $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$, the (i, j)-th entry of H_N^M is defined by $\phi_j(\boldsymbol{x}_i)$. Denote the k-th column and i-th row of H_N^M by $H_N^{(k)}$ and $H_{(i)}^M$, respectively. Combining with the form of $\mathscr{A}_m^M$ and the definition of tensor multiplications, for any $i_1, i_2 = 1, 2, \dots, N$, the (i_1, i_2)-th entry of $\mathscr{A}_m^M \boldsymbol{c}^{m-2}$ has the form

$$\begin{aligned}
&\sum_{i_3,i_4,\dots,i_m=1}^{N,N,\dots,N} \Big(c_{i_3} c_{i_4} \dots c_{i_m} \big(\sum_{k=1}^{M} \prod_{j=1}^{m} \phi_k(\boldsymbol{x}_{i_j}) \big) \Big) \\
&= \sum_{k=1}^{M} \Big(\sum_{i_3,i_4,\dots,i_m=1}^{N,N,\dots,N} \big(c_{i_3} c_{i_4} \dots c_{i_m} \prod_{j=1}^{m} \phi_k(\boldsymbol{x}_{i_j}) \big) \Big) \\
&= \sum_{k=1}^{M} \Big(\phi_k(\boldsymbol{x}_{i_1}) \phi_k(\boldsymbol{x}_{i_2}) \sum_{i_3,i_4,\dots,i_m=1}^{N,N,\dots,N} \big(\prod_{j=3}^{m} c_{i_j} \phi_k(\boldsymbol{x}_{i_j}) \big) \Big) \\
&= \sum_{k=1}^{M} \Big(\phi_k(\boldsymbol{x}_{i_1}) \phi_k(\boldsymbol{x}_{i_2}) \big(\sum_{i=1}^{N} c_i \phi_k(\boldsymbol{x}_i) \big)^{m-2} \Big) \\
&= \sum_{k=1}^{M} \Big(\phi_k(\boldsymbol{x}_{i_1}) \phi_k(\boldsymbol{x}_{i_2}) (\boldsymbol{c}^\top H_N^{(k)})^{(m-2)} \Big) \\
&= \big(c^\top H_N^M \big)^{[m-2]} \big(H_{(i_1)}^M \odot H_{(i_2)}^M \big)^\top,
\end{aligned}$$

where for two vector $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{x} \odot \boldsymbol{y}$ means the element-wise multiplications, and $\boldsymbol{x}^{[m-2]}$ means the $(m-2)$-th order element-wise power of $\boldsymbol{x}$. Then, letting $\hat{H}_N^M$ be a matrix of shape $N \times M$ such that its ith row is $(c^\top H_N^M)^{[m-2]} \odot H_{(i)}^M$, one can see $\mathscr{A}_m^M \boldsymbol{c}^{m-2} = \hat{H}_N^M (H_N^M)^\top$. Similarly, for any $i = 1, 2, \dots, N$, the i-th entry of $\mathscr{A}_m^M \boldsymbol{c}^{m-1}$ has the form $(\boldsymbol{c}^\top H_N^M)^{[m-1]} \big(H_{(i)}^M \big)^\top$, so we have $\mathscr{A}_m^M \boldsymbol{c}^{m-1} = H_N^M \big((\boldsymbol{c}^\top H_N^M)^{[m-1]} \big)^\top$. Also, $\mathscr{A}_m^M \boldsymbol{c}^m = \big((\boldsymbol{c}^\top H_N^M)^{[m]} \big) \mathbf{1}$, where $\mathbf{1}$ refers to an all-one vector.

We can see that, in this way, the calculation of tensor multiplications only involves matrix-vector and vector-vector multiplications, and we only need to store H_N^M rather than the whole tensor $\mathscr{A}_m^M$, which reduces the space complexity from $O(N^m)$ to $O(MN)$. Moreover, by this formula, the price of increasing m reduces dramatically to nearly 0. That is because m only appears in the power, while computing different powers in computer takes the same price.

So far we still do not discuss the existence and uniqueness of solutions to (3.3), which are actually strongly related to the choice of M and the rank of H_N^M. Recall that Proposition 2.1 and Proposition 2.2 in [3] indicate that $\mathscr{A}_m^M$ is required to be at least positive definite to guarantee the existence of solutions to (3.3), and strictly positive definite to make sure the solution is unique. However, $\mathscr{A}_m^M$ is not always positive definite, not to mention being strictly positive definite. To exploit the conditions of (3.3) to have (unique) solutions, we first introduce the following lemma considering the rank of H_N^M.

Lemma 3.1. *There exists a $j_N \geq N$ such that, for any $M \geq j_N$, rank$(H_N^M) = N$, namely H_N^M has full (row) rank.*

Proof. Since all cases share the same H_N^M, we can only consider the case $m = 2$.

Consider the case that $m = 2$, where $\mathscr{A}_2$ is a positive definite matrix and so $\operatorname{rank}(\mathscr{A}_2) = N$. By (2.2), $\mathscr{A}_2 = H_N^\infty (H_N^\infty)^\top$. Here, we abuse the notation H_N^∞ to represent an infinite matrix. Then, because $N = \operatorname{rank}(\mathscr{A}_2) = \operatorname{rank}(H_N^\infty (H_N^\infty)^\top) \leq \operatorname{rank}(H_N^\infty) \leq N$, one can see $\operatorname{rank}(H_N^\infty) = N$. That means that there are N linearly independent columns of H_N^∞. Let j_N be the largest column index of these N linearly independent columns. Then, for any $M \geq j_N$, we have $\operatorname{rank}(H_N^M) = N$. $\square$

It is worthy pointing out that H_N^M has full (row) rank means that the linear system $(H_N^M)^\top \boldsymbol{x} = \mathbf{0}$ for $\boldsymbol{x} \in \mathbb{R}^N$ with N unknown variables and M equations has only zero solution. Conversely, it cannot conclude that the linear system $H_N^M \boldsymbol{x} = \mathbf{0}$ for $\boldsymbol{x} \in \mathbb{R}^M$ with M unknown variables and N equations has only zero solution, which holds when $M \leq N$ with $\operatorname{rank}(H_N^M) = M$. Based on such a statement, we can obtain the following proposition.

Proposition 3.1. *(1) For $M > 0$, $\mathscr{A}_m^M$ is semi-positive definite.*

(2) If further $M \geq N$ is such that $\operatorname{rank}(H_N^M) = N$, $\mathscr{A}_m^M$ is strictly positive definite.

(3) If, in addition, $M = N$ and $\operatorname{rank}(H_N^M) = N = M$, which means that H_N^M is a full-rank squared matrix, then $\mathscr{A}_m^M$ is nonsingular, say $\{\boldsymbol{x} \in \mathbb{R}^N \backslash \{0\} | \mathscr{A}_m^M \boldsymbol{x}^{m-1} = 0\} = \emptyset$ [23].

Proof. (1) Combining the formula $\mathscr{A}_m^M \boldsymbol{c}^m = \big((\boldsymbol{c}^\top H_N^M)^{[m]}\big)\mathbf{1}$ with the fact that m is a positive even integer, it is easy to see $\mathscr{A}_m^M$ is semi-positive definite.

(2) If further $M \geq N$ is such that $\operatorname{rank}(H_N^M) = N$, then the linear system $(H_N^M)^\top \boldsymbol{x} = \mathbf{0}$ only has zero solution. Then, for any $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^N$, we have

$$\begin{aligned}
&(\boldsymbol{c}-\boldsymbol{d})^\top (\mathscr{A}_m^M \boldsymbol{c}^{m-1} - \mathscr{A}_m^M \boldsymbol{d}^{m-1}) \\
&= (\boldsymbol{c}-\boldsymbol{d})^\top \big(H_N^M \big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top - H_N^M \big((\boldsymbol{d}^\top H_N^M)^{[m-1]}\big)^\top\big) \\
&= (\boldsymbol{c}-\boldsymbol{d})^\top \Big(H_N^M \big((\boldsymbol{c}^\top H_N^M)^{[m-1]} - (\boldsymbol{d}^\top H_N^M)^{[m-1]}\big)^\top\Big) \\
&= (\boldsymbol{c}^\top H_N^M - \boldsymbol{d}^\top H_N^M)\big((\boldsymbol{c}^\top H_N^M)^{[m-1]} - (\boldsymbol{d}^\top H_N^M)^{[m-1]}\big)^\top .
\end{aligned}$$

Since the elements of $\boldsymbol{c}^\top H_N^M - \boldsymbol{d}^\top H_N^M$ and $\big((\boldsymbol{c}^\top H_N^M)^{[m-1]} - (\boldsymbol{d}^\top H_N^M)^{[m-1]}\big)^\top$ have the same sign, then $(\boldsymbol{c}-\boldsymbol{d})^\top (\mathscr{A}_m^M \boldsymbol{c}^{m-1} - \mathscr{A}_m^M \boldsymbol{d}^{m-1}) \geq 0$. Moreover, $(\boldsymbol{c}-\boldsymbol{d})^\top (\mathscr{A}_m^M \boldsymbol{c}^{m-1} - \mathscr{A}_m^M \boldsymbol{d}^{m-1})$=0 if and only if $\boldsymbol{c}^\top H_N^M - \boldsymbol{d}^\top H_N^M = \mathbf{0}^\top$. That is, $(H_N^M)^\top(\boldsymbol{c}-\boldsymbol{d}) = \mathbf{0}$, solving which we can obtain $\boldsymbol{c} = \boldsymbol{d}$. Thus, $\mathscr{A}_m^M$ is strictly positive definite.

(3) If H_N^M is a full-rank squared matrix, then $H_N^M \boldsymbol{x} = \mathbf{0}$ only exists zero solution. Using $\mathbf{0} = \mathscr{A}_m^M \boldsymbol{c}^{m-1} = H_N^M \big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top$, and letting $\hat{\boldsymbol{x}} = \big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top$, one can obtain $\hat{\boldsymbol{x}} = \mathbf{0}$ by solving $H_N^M \hat{\boldsymbol{x}} = \mathbf{0}$. We can simply drop the $(m-1)$-power and consider $(H_N^M)^\top \boldsymbol{c} = \mathbf{0}$, while one should pay attention to the fact that H_N^M is not symmetric. Again, we can easily obtain that $\boldsymbol{c} = \mathbf{0}$. Thus, $\mathscr{A}_m^M$ is nonsingular.

$\square$

The global convergence or the convergence rates of many algorithms solving (1.1) require the positive definiteness of $\mathscr{A}_m \boldsymbol{c}^{m-2}$. For example, the methods in [10] and the homotopy method in [19]. We provide Propositions about it here.

Proposition 3.2. *(1) For $M > 0$ and any $\boldsymbol{c} \in \mathbb{R}^N$, $\mathscr{A}_m^M \boldsymbol{c}^{m-2}$ is a semi-positive definite matrix.*

(2) If further $M \geq N$ is such that $\operatorname{rank}(H_N^M) = N$, then $\mathscr{A}_m^M \boldsymbol{c}^{m-2}$ is positive definite for any non-zero $\boldsymbol{c}$.

Proof. (1) Given any $\boldsymbol{c} \in \mathbb{R}^N$, by the formula of the element of $\mathscr{A}_m^M \boldsymbol{c}^{m-2}$, we have the formula of $\boldsymbol{y}^\top (\mathscr{A}_m^M \boldsymbol{c}^{m-2}) \boldsymbol{y}$ for any $\boldsymbol{y} = (y_i)_{i=1}^N \in \mathbb{R}^N$.

$$\begin{aligned}
\boldsymbol{y}^\top(\mathscr{A}_m^M\boldsymbol{c}^{m-2})\boldsymbol{y} &= \sum_{i_1=1}^N\sum_{i_2=1}^N\sum_{k=1}^M y_{i_1}y_{i_2}\phi_k(\boldsymbol{x}_{i_1})\phi_k(\boldsymbol{x}_{i_2})(\boldsymbol{c}^\top H_N^{(k)})^{m-2}\\
&= \sum_{k=1}^M\sum_{i_1=1}^N\sum_{i_2=1}^N \phi_k(\boldsymbol{x}_{i_1})y_{i_1}\phi_k(\boldsymbol{x}_{i_2})y_{i_2}(\boldsymbol{c}^\top H_N^{(k)})^{m-2}\\
&= \sum_{k=1}^M(\boldsymbol{c}^\top H_N^{(k)})^{m-2}\Big(\sum_{i_1=1}^N\phi_k(\boldsymbol{x}_{i_1})y_{i_1}\Big)\Big(\sum_{i_2=1}^N\phi_k(\boldsymbol{x}_{i_2})y_{i_2}\Big)\\
&= \sum_{k=1}^M(\boldsymbol{c}^\top H_N^{(k)})^{m-2}\big(\boldsymbol{y}^\top H_N^{(k)}\big)^2\\
&= (\boldsymbol{c}^\top H_N^M)^{[m-2]}\big((H_N^M)^\top\boldsymbol{y}\big)^{[2]}\geq 0.
\end{aligned}$$

The final inequality comes from the fact that m is a positive even integer. This shows that $\mathscr{A}_m^M\boldsymbol{c}^{m-2}$ is a semi-positive definite matrix.

(2) If further $M\geq N$ is such that $\operatorname{rank}(H_N^M)=N$, then two linear systems $\boldsymbol{c}^\top H_N^M=\boldsymbol{0}$ and $(H_N^M)^\top\boldsymbol{y}=\boldsymbol{0}$ in the inequality only have zero solution. That means that, for any non-zero $\boldsymbol{c}$, $\boldsymbol{y}^\top(\mathscr{A}_m^M\boldsymbol{c}^{m-2})\boldsymbol{y}=0$ if and only if $\boldsymbol{y}=\boldsymbol{0}$. Thus, $\mathscr{A}_m^M\boldsymbol{c}^{m-2}$ is positive definite.

□

By Proposition 3.1 and [3, Proposition 2.2], we have that, for $M\geq N$ such that $\operatorname{rank}(H_N^M)=N$, (3.3) exists a unique solution. The next theorem shows that when M is selected large enough, we can get an approximated solution to the original problem (1.1) by solving (3.3) with acceptable error.

Theorem 3.1. *For $M\geq N$ such that $\operatorname{rank}(H_N^M)=N$, let $\boldsymbol{c}_*$ and $\boldsymbol{c}_*^M$ be solutions to (1.1) and (3.3), respectively. Then*

$$\lim_{M\to+\infty}\boldsymbol{c}_*^M=\boldsymbol{c}_*.$$

Proof. By [3, Proposition 2.2], $\boldsymbol{c}_*$ is also unique. So, it is equivalent to show that

$$\lim_{M\to+\infty}\mathscr{A}_m\boldsymbol{c}_*^M=b.$$

We have $\mathscr{A}_m^M(\boldsymbol{c}_*^M)^{m-1}=b$ and $\lim_{M\to+\infty}\mathscr{A}_m^M-\mathscr{A}_m=0$ by (3.1). Hence

$$\lim_{M\to+\infty}\mathscr{A}_m(\boldsymbol{c}_*^M)^{m-1}-\mathscr{A}_m^M(\boldsymbol{c}_*^M)^{m-1}=\lim_{M\to+\infty}(\mathscr{A}_m-\mathscr{A}_m^M)(\boldsymbol{c}_*^M)^{m-1}=0.$$

It follows that $\lim_{M\to+\infty}\mathscr{A}_m(\boldsymbol{c}_*^M)^{m-1}=\mathscr{A}_m^M(\boldsymbol{c}_*^M)^{m-1}=b$. The proof is completed. □

Given a new data point $\boldsymbol{x}$, we can use the formula to predict the corresponding output by $s_m(\boldsymbol{x})=B_m(\boldsymbol{x})\boldsymbol{c}^{m-1}$. Let $\varphi(\boldsymbol{x}):=(\phi_k(\boldsymbol{x}))_{k=1}^M$. Then

$$\begin{aligned}
B_m(\boldsymbol{x})\boldsymbol{c}^{m-1} &= \sum_{i_1,i_2,\ldots,i_{m-1}=1}^{N,N,\ldots,N}\Big(c_{i_1}c_{i_2}\ldots c_{i_{m-1}}\Big(\sum_{k=1}^M\phi_k(\boldsymbol{x})\prod_{j=1}^{m-1}\phi_k(\boldsymbol{x}_{i_j})\Big)\Big)\\
&= \sum_{k=1}^M\phi_k(\boldsymbol{x})\Big(\sum_{i_1,i_2,\ldots,i_{m-1}=1}^{N,N,\ldots,N}\Big(c_{i_1}c_{i_2}\ldots c_{i_{m-1}}\prod_{j=1}^{m-1}\phi_k(\boldsymbol{x}_{i_j})\Big)\Big)\\
&= \sum_{k=1}^M\Big(\phi_k(\boldsymbol{x})\Big(\sum_{i=1}^N c_i\phi_k(\boldsymbol{x}_i)\Big)^{m-1}\Big)\\
&= \varphi(\boldsymbol{x})^\top\big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top.
\end{aligned}$$

3.3. **A homotopy method.** In this section, inspired by [19], we try to apply the homotopy method therein to solve (3.3).

First, we consider the most beautiful case. If, fortunately, H_N^M is a full-rank squared matrix, we can solve (3.3) by simply calculating $(H_N^M)^{-1}$ and some matrix-vector multiplications. By $\mathscr{A}_m^M \boldsymbol{c}^{m-1} = H_N^M\big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top = \boldsymbol{y}$, we have $\big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top = (H_N^M)^{-1}\boldsymbol{y}$. Then $(H_N^M)^\top \boldsymbol{c} = ((((H_N^M)^{-1}\boldsymbol{y})^\top)^{[\frac{1}{m-1}]})^\top$. Finally, we have,

$$\boldsymbol{c} = ((H_N^M)^{-1})^\top ((((H_N^M)^{-1}\boldsymbol{y})^\top)^{[\frac{1}{m-1}]})^\top.$$

However, in general, H_N^M is not always a full-rank squared matrix.

Given $\mathscr{A}_m^M$ defined by (3.2), we define a function by

$$\mathscr{H}(t,\boldsymbol{c}) := \begin{bmatrix} t \\ (1-t)\mathscr{A}_m^M \boldsymbol{c}^{m-1} + t\boldsymbol{c} - \boldsymbol{y} \end{bmatrix}, \text{ for any } t \in [0,1], \boldsymbol{c} \in \mathbb{R}^N.$$

By Proposition 3.2, we can obtain a result similar with [19, Theorem 3]. That is, $\mathscr{H}$ is continuously differentiable at any $(t,\boldsymbol{c}) \in (0,1)\times\mathbb{R}^N$, and its Jacobian matrix is given by

$$\mathscr{H}'(t,\boldsymbol{c}) = \begin{bmatrix} 1 & \boldsymbol{0} \\ -\mathscr{A}_m^M \boldsymbol{c}^{m-1} + \boldsymbol{c} & (m-1)(1-t)\mathscr{A}_m^M \boldsymbol{c}^{m-2} + tI \end{bmatrix},$$

which is invertible for any $(t,\boldsymbol{c}) \in (0,1)\times\mathbb{R}^N$.

This construction of $\mathscr{H}$ is motivated by that $\boldsymbol{c}$ solves (3.3) if and only if $\mathscr{H}(t,\boldsymbol{c}) = \boldsymbol{0}$. Then we can employ a Newton-type methods to solve $\mathscr{H}(t,\boldsymbol{c}) = \boldsymbol{0}$.

The algorithm is totally the same as that in [19, Algorithm 1]. For convenience, we shall write it down here as a reference.

Algorithm 1: A Homotopy Method

```
Input: Constants $\delta \in (0,1), \sigma \in (0,\frac{1}{2}), \bar{t} > 0, \tau \in (0,1)$ such that $\tau\bar{t} < 1$. Take $t_0 := \bar{t}$
       and an initial point $\boldsymbol{c}_0 \in \mathbb{R}^N$. Let $e^0 := (1,0,\ldots,0) \in \mathbb{R}^{N+1}$.
Output: An approximated solution $\boldsymbol{c}_*$.
1 while $\|\mathscr{H}(t_k,\boldsymbol{c}_k)\| > 0$ do
2     Compute $\beta_k = \tau\min\{1, \|\mathscr{H}(t_k,\boldsymbol{c}_k)\|^2\}, (\Delta t_k, \Delta\boldsymbol{c}_k) \in (0,1)\times\mathbb{R}^N$ by
          $\mathscr{H}'(t_k,\boldsymbol{c}_k)(\Delta t_k,\Delta\boldsymbol{c}_k) + \mathscr{H}(t_k,\boldsymbol{c}_k) = \bar{t}\beta_k e^0.$
3     Let $\lambda_k$ be the maximum of the values $1, \delta, \delta^2, \ldots$ such that
          $\|\mathscr{H}(t_k + \lambda_k\Delta t_k, \boldsymbol{c}_k + \lambda_k\Delta\boldsymbol{c}_k)\|^2 \le [1 - 2\sigma(1-\tau\bar{t})\lambda_k]\|\mathscr{H}(t_k,\boldsymbol{c}_k)\|^2.$
4     Set $(t_{k+1},\boldsymbol{c}_{k+1}) = (t_k,\boldsymbol{c}_k) + (\Delta t_k,\Delta\boldsymbol{c}_k)$ and $k := k+1$.
```

Next, we will show that the global convergence and the locally superlinear convergence of this algorithm still hold for our problem.

Lemma 3.2. *For $M > 0$, $\lim_{\|\boldsymbol{c}\|\to+\infty} \|\mathscr{A}_m^M \boldsymbol{c}^{m-1}\| = +\infty$.*

Proof. Recall that we have $\mathscr{A}_m^M \boldsymbol{c}^{m-1} = H_N^M\big((\boldsymbol{c}^\top H_N^M)^{[m-1]}\big)^\top$. By the definitions of H_N^M and the eigenfunctions ϕ_k, all rows of H_N^M are not all-zero. Thus, it is easy to conclude that $\lim_{\|\boldsymbol{c}\|\to+\infty} \|\mathscr{A}_m^M \boldsymbol{c}^{m-1}\| = +\infty$. □

By Lemma 3.2, $\mathscr{A}_m^M(\boldsymbol{c}^*)^{m-2}$ is positive definite and so nonsingular if $M \geq N$ is such that $\mathrm{rank}(H_N^M) = N$. Similar to Theorems 4, 5, 6 in [19], we can obtain the global convergence and locally superlinear convergence of Algorithm 1.

Theorem 3.2. *Let $M \geq N$ be such that $\mathrm{rank}(H_N^M) = N$, and let $\{(t_k, \boldsymbol{c}_k)\}_{k=1}^{\infty}$ be the infinite sequence generated by Algorithm 1.*

(1) $\{(t_k, \boldsymbol{c}_k)\}_{k=1}^{\infty}$ is bounded.
(2) Any accumulation point of $\{\boldsymbol{c}_k\}_{k=1}^{\infty}$ is a solution to (3.3).
(3) If $(t_*, \boldsymbol{c}_*)$ be an accumulation point of $\{(t_k, \boldsymbol{c}_k)\}_{k=1}^{\infty}$, then the whole sequence $\{(t_k, \boldsymbol{c}_k)\}_{k=1}^{\infty}$ converges to $(t_*, \boldsymbol{c}_*)$ and

$$\limsup_{k\to\infty} \frac{\|(t_{k+1}, \boldsymbol{c}_{k+1}) - (t_*, \boldsymbol{c}_*)\|}{\|(t_k, \boldsymbol{c}_k) - (t_*, \boldsymbol{c}_*)\|} = 0, \qquad \limsup_{k\to\infty} \frac{t_{k+1}}{t_k} = 0.$$

3.4. **The choice of M.** In this subsection, we shall focus on some details when performing the numerical tricks proposed in 3.2.

A problem cannot be ignored is that how large the M should be? Or equivalently, what M should we choose? Indeed, the choice of M involves twofold considerations. On the one hand, M should be selected to satisfy the significant condition that $M \geq N$ such that $\mathrm{rank}(H_N^M) = N$ to make sure the existence of solutions to (3.3). However, no matter how it seems natural, it is not as easy as what we imagine to find an appropriate M. The construction of H_N^M implies that all of its columns are pairwise linearly independent because of the property of eigenfunctions. But this cannot ensure that H_N^M is full-rank for any M, say, if $M \leq N, \mathrm{rank}(H_N^M) = M$, otherwise $\mathrm{rank}(H_N^M) = N$. A worse case is that although we can theoretically guarantee that existence of M such that $\mathrm{rank}(H_N^M) = N$ by Lemma 3.1, the appropriate M may be impossible to find when implementing the numerical tricks because of the limit of machine precision. For example, we shall consider the well-known Gaussian kernel function. For $m = 2$, the classical Gaussian kernel is defined by $K_{\mathrm{gau}}(x, y) = e^{-\sigma^2\|x-y\|^2}$. By [4, Subsection 4.4], we can write down the eigenpair-based representation of Gaussian kernel for $m > 2$ by

$$K_{\mathrm{gau}}(x_1, x_2, \ldots, x_m) = \sum_{k=1}^{\infty} \prod_{i=1}^{m} \rho_{\sigma,k}^{1/2} \phi_{\sigma,k}(x_i),$$

where

$$\rho_{\sigma,k} := (1 - \omega_\sigma)\omega_\sigma^{k-1}, \qquad \omega_\sigma := \frac{2\sigma^2}{1 + (1 + 4\sigma^2)^{1/2} + 2\sigma^2},$$

and

$$\phi_{\sigma,k}(x) := \left(\frac{(1 + 4\sigma^2)^{1/4}}{2^{k-1}(k-1)!} \right)^{1/2} e^{-u_\sigma x^2} H_{k-1}\left((1 + 4\sigma^2)^{1/4} x \right),$$

where

$$u_\sigma := \frac{2\sigma^2}{1 + (1 + 4\sigma^2)^{1/2}},$$

and H_{k-1} is the Hermite polynomial of degree $k - 1$, that is,

$$H_{k-1}(x) := (-1)^{k-1} e^{x^2} \frac{\mathrm{d}^{k-1}}{\mathrm{d}x^{k-1}} (e^{-x^2}).$$

The problem occurs in the first term of $\phi_{\sigma,k}(x)$, that is, $(\frac{(1+4\sigma^2)^{1/4}}{2^{k-1}(k-1)!})^{1/2}$. One can observe that this term reduces to 0 at a crazy speed. Then, for any input x, $\phi_{\sigma,k}(x)$ also decreases to 0 incredibly fast. Although for any $x \in \mathbb{R}$ and $k \in \mathbb{N}_+$, $\phi_{\sigma,k}(x) \neq 0$ theoretically and then by Lemma 3.1, we can find a suitable M, unfortunately, we cannot do so in practice. Owing to the limit of machine precision, in Matlab or other kinds of scientific software, for any $x \in \mathbb{R}, \sigma \in \mathbb{R}$ and $k > 151$, $\phi_{\sigma,k}(x)$ will be treated as 0. That means, no matter what σ we use, how does the input data points distribute, and how large M we choose, $\text{rank}(H_N^M) \leq 151$ in practice.

Moreover, the case may be even worse. We shall take several numerical examples to show the problem behind that. First, we generate 200 Halton points $\mathbf{x}'$ and consider the dataset $\mathbf{x} = \mathbf{x}' - 0.5$. Then, we use different σ's $(0.1, 1, 2, 3, 5, 9)$ to illustrate the change of rank of H_N^M when M increases, see Figure 1. As we can see, for any σ, at the beginning, $\text{rank}(H_N^M)$ increases nearly linearly as M increases. After a certain M_σ dependent on σ, the increasing speed of $\text{rank}(H_N^M)$ slows down gradually and more cost is needed to increase the rank by 1. Finally, the rank will stop at highest 45.

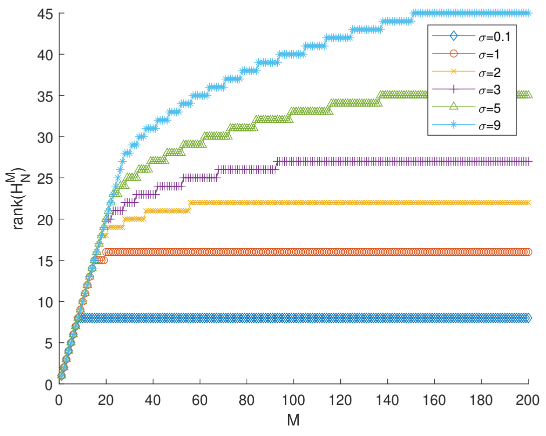


FIGURE 1. The change of rank of H_N^M when M increases

To deal with this problem, we should use the low-rank approximation trick. That is, the H_N^M actually need not to be full-rank in practice to obtain the approximated results. This trick is discussed and well-studied in [24, Subsection 6.1].

On the other hand, to obtain better approximation performance, M should be chosen to achieve as high precision as possible, generally the machine precision is preferred. Actually, in this sense, in practice, M needs not to be chosen as large as we imagine. As m increases, given a preferred precision $\varepsilon > 0$, the corresponding M to achieve this precision decreases significantly. By $\sum_{k=1}^{\infty} |\phi_k(\boldsymbol{x})| < \infty$, we have $\lim_{k\to+\infty} \phi_k(\boldsymbol{x}) = 0$. Then, in (2.2), as m increases, for each k, the speed at which $\prod_{i=1}^{m} \phi_k(\boldsymbol{z}_i)$ decreases to 0 increases. It leads to for larger m, we need smaller M. To give an illustrative example, consider the min kernel $K_2(z_1, z_2) := \min\{z_1, z_2\} - z_1 z_2$ for $z_1, z_2 \in [0,1]$ and the corresponding $\phi_k(z)$ in [4, Section 4.3] defined by $\phi_k(z) := \frac{\sqrt{2}}{k\pi} \sin(k\pi z)$. If the scale of each single term in the infinite sum is set to $O(10^{-6})$, for $m = 2$, M has to be 1000, for $m = 4$, M has to be 32, for $m = 6$, M only has to be 10.

Combining the above discussion, we can observe that M needs not to be chosen as high as we imagine. Then, we can move to the numerical experiments.

4. Numerical Experiments

In this section, we shall perform simple numerical experiments on artificial datasets to evaluate the benefits of the multikernel-based approximation method, namely the improvement from $m = 2$ to $m > 2$, and how the choice of M influences the performance. The reasons for experimenting only on synthesis dataset are twofold. On the one hand, although the numerical tricks can significantly reduce the computation complexity, due to the limit of computation resources, the real-world dataset is still too large-scale to handle. On the other hand, the power of the multikernel-based approximation method lies more in the charm of the regularization network. That means that the multilinear system considered in this paper cannot deal with real-world datasets with noise well.

We shall provide several details of the numerical experiments firstly. We generate two sets of sampling points $\mathbf{x}_1 := \{0.0625, 0.125, 0.25, 0.375, 0.5, 0.5625, 0.625, 0.75, 0.875\}$ and $\mathbf{x}_2 := \{-0.4375, -0.375, -0.25, -0.125, 0, 0.0625, 0.125, 0.25, 0.375\}$. Actually, $\mathbf{x}_1$ is the Halton points, while $\mathbf{x}_2 = \mathbf{x}_1 - 0.5$. This is based on the experiences of numerical experiments from [25]. Two functions $f_1(x) := \sin(9x)\cos(9x) + x, x \in (0,1)$ and $f_2(x) := x\sin(18x), x \in [-0.5, 0.5]$ are used to generate the output values without noise, i.e. $\boldsymbol{y}_1 := \{f_1(x) : x \in \mathbf{x}_1\}$ and $\boldsymbol{y}_2 := \{f_2(x) : x \in \mathbf{x}_2\}$. Thus, we have two datasets $(\mathbf{x}_1, \boldsymbol{y}_1), (\mathbf{x}_2, \boldsymbol{y}_2)$. To compare the performance of interpolance, we use root mean square errors of grid points [25, Page 10, (1.5)] as the indicator. The grid points vary from the first dataset to the second dataset. For the first dataset, the grid points are set to be the 999 grid points in $[0.001, 9.999]$ with the gap 0.001. For the second dataset, the grid points are set to be the 1000 grid points in $[-0.5, 0.5]$ with the gap being 0.001. Specifically, we use the min kernel for $m = 2$ and $m = 4$ on $(\mathbf{x}_1, \boldsymbol{y}_1)$, the Gaussian kernel for $m = 2$ and $m = 4$ on $(\mathbf{x}_2, \boldsymbol{y}_2)$, then compare their performance according to the root mean square errors of the corresponding grid points.

We shall use the min kernel and Gaussian kernel, which have been discussed in Subsection 3.4. Here, based on experiences from [25] and the results of numerical examples shown in Subsection 3.4 about the rank of H_N^M, we choose $\sigma = 9$. Moreover, since we want to know how the choice of M affects the performance, for each experiment, we compare different M's performance. To be more precise, for the experiment using min kernel, we use $M = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10000$, and, for the one using Gaussian kernel, we use $M = 1, 2, \ldots, 100$. This is because that min kernel has a precious property that select $M = N$ then $\operatorname{rank}(H_N^M) = N$. According to the experience in [19], we choose $\delta = 0.75, \sigma = 10^{-4}, \tau = 10^{-5}, t_0 = 0.01, \boldsymbol{c}_0 = \mathbf{1}$, and the stop criterion is set as $\|\mathscr{H}(t_k, \boldsymbol{c}_k)\| < 10^{-7}$.

For the first dataset, only 8 iterations are taken to achieve the stop criterion. Since the min kernel for $m = 2$ is linear, it cannot construct a nonlinear function, and its performance is bad except for the known sampling points with root mean square error is 0.0096. By contrast, the multikernel version of min kernel can approximate better with the minimal root mean square error is only 0.0039. The root mean square errors for different M's are shown in Table 1. We can notice that the result obtained by $M = 10000$ is nearly the same as the one obtained by $M = 9$, which implies that the choice of M needs not to be as large as we imagine.

For the second dataset, both $m = 2$ and $m = 4$ can obtain great results, while only 10 iterations are needed to reach the stop criterion. But the curves in $[0.375, 0.5]$ show the remarkable difference in performance. Although there are no more sampling points, the approximated function for $m = 4$ is close to the true function, while that for $m = 2$ cannot. The root mean square error

1	2	3	4	5	6	7	8	9	10000
0.1087	0.0834	0.0675	0.0567	0.0360	0.0040	0.0041	0.0042	0.0042	0.0039

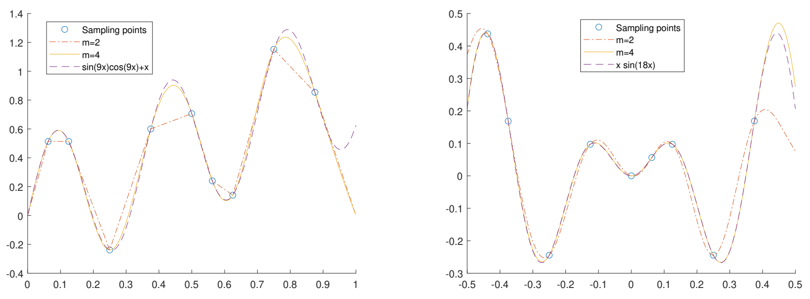
TABLE 1. Root mean square errors for different M's for the first dataset using min kernel. The first row is the different choices of M, the second row is the corresponding root mean square errors.

for $m = 2$ is 0.0034, and the best root mean square error is $8.4927e-05$ for $m = 4$. The root mean square errors for $M = 1, 2, \ldots, 30$ are shown in Table 2, while for $30 \leq M \leq 100$ the errors are nearly the same value. This also verifies our statement that M needs not choose as large as we imagine.

1	2	3	4	5	6
0.0232	0.0240	0.0224	0.0256	0.0063	0.0064
7	8	9	10	11	12
0.0283	0.3323	0.0013	5.9171e-04	8.4927e-05	4.1073e-04
13	14	15	16	17	18
5.9683e-04	4.5063e-04	8.1266e-04	6.8779e-04	7.5755e-04	6.5180e-04
19	20	21	22	23	24
6.5499e-04	5.4742e-04	5.4740e-04	4.9099e-04	4.9018e-04	4.8815e-04
25	26	27	28	29	30
4.8752e-04	5.0588e-04	5.0579e-04	5.1964e-04	5.1964e-04	5.2474e-04

TABLE 2. Root mean square errors for $M = 1, 2, \ldots, 30$ for the second dataset using Gaussian kernel. The odd rows are the different choices of M, the even rows are the corresponding root mean square errors.

We pick the best results among all results using different M's as the approximated functions. The approximated functions and the true functions are displayed in Figure 2. We can see that in both two numerical experiments the results of $m = 4$ are better than those of $m = 2$.



(A) Results of the first dataset using min kernel.

(B) Results of the second dataset using Gaussian kernel.

FIGURE 2. Numerical results of multilinear system (3.3) for $m = 2$ and $m = 4$.

Acknowledgments
The second author was supported by the National Natural Science Foundation of China under grant 11771099 and Shanghai Municipal Science and Technology Commission under grant 22WZ2501900. The third author was supported by the National Natural Science Foundation of China #12071157 and #12026602, and the Natural Science Foundation of Guangdong #2019A1515011995 and #2020B1515310013.

References

[1] G.E. Fasshauer, M.J. McCourt, Kernel-based Approximation Methods Using MATLAB, vol. 19, World Scientific Publishing Company, Singapore, 2015.

[2] H. Wendland, Scattered Data Approximation, vol. 17, Cambridge University Press, Cambridge, 2004.

[3] Q. Ye, Positive definite multi-kernels for scattered data interpolations, arXiv preprint arXiv:2111.03490, 2021.

[4] Y. Xu, Q. Ye, Generalized Mercer Kernels and Reproducing Kernel Banach Spaces, Memoirs Amer. Math. Soc. vol. 258, no. 1243 vi+122, 2019. doi: 10.1090/memo/1243.

[5] W. Ding, Y. Wei, Solving multi-linear systems with $\mathscr{M}$-tensors, J. Sci. Computing, 68 (2016), 689-715.

[6] X. Li, M.K. Ng, Solving sparse non-negative tensor equations: Algorithms and applications, Front. Math. China, 10 (2015), 649-680.

[7] Z. Luo, L. Qi, N. Xiu, The sparsest solutions to $\mathscr{Z}$-tensor complementarity problems, Optim. Lett. 11 (2017), 471-482.

[8] S. Hu, A note on the solvability of a tensor equation, J. Ind. Manag. Optim. (2021), doi:10.3934/jimo.2021146.

[9] Y. Wang, Z.H. Huang, L. Qi, Global uniqueness and solvability of tensor variational inequalities, J. Optim. Theory Appl. 177 (2018), 137-152.

[10] D.H. Li, S. Xie, H.R. Xu, Splitting methods for tensor equations, Numer. Linear Algebra Appl. 24 (2017), e2102. doi: 10.1002/nla.2102.

[11] Z.J. Xie, X.Q. Jin, Y.M. Wei, Tensor methods for solving symmetric $\mathscr{M}$-tensor systems, J. Sci. Comput. 74 (2017), 412–425.

[12] L. Han, A homotopy method for solving multilinear systems with $\mathscr{M}$-tensors, Appl. Math. Lett. 69 (2017), 49-54.

[13] H. He, C. Ling, L. Qi, G. Zhou, A globally and quadratically convergent algorithm for solving multilinear systems with $\mathscr{M}$-tensors, J. Sci. Computing, 76 (2018), 1718-1741.

[14] C.Q. Lv, C.F. Ma, A levenberg–marquardt method for solving semi-symmetric tensor equations, J. Comput. Appl. Math. 332 (2018), 13-25.

[15] J. Liu, S. Du, Y. Chen, A sufficient descent nonlinear conjugate gradient method for solving $\mathscr{M}$-tensor equations, J. Comput. Appl. Math. 371 (2020), 112709.

[16] X. Wang, C. Mo, M. Che, Y. Wei, Accelerated dynamical approaches for finding the unique positive solution of $\mathscr{K}\mathscr{S}$-tensor equations, Nume. Algorithms, 88 (2021), 1787-1810.

[17] X. Wang, M. Che, Y. Wei, Neural networks based approach solving multi-linear systems with $\mathscr{M}$-tensors, Neurocomputing, 351 (2019), 33-42.

[18] X. Wang, M. Che, Y. Wei, Neural network approach for solving nonsingular multi-linear tensor systems, J. Comput. Appl. Math. 368 (2020), 112569.

[19] J.C. Yan, Y. Xu, Z.H. Huang, A homotopy method for solving multilinear systems with strong completely positive tensors, Appl. Math. Lett. 124 (2022), 107636.

[20] L. Qi, H. Chen, Y. Chen, Tensor Eigenvalues and Their Applications, vol. 39, Springer, Singapore, 2018.

[21] I. Steinwart, A. Christmann, Support Vector Machines, Springer, New York, 2008.

[22] H. Zhang, Y. Xu, J. Zhang, Reproducing kernel Banach spaces for machine learning, 2009 International Joint Conference on Neural Networks, pp. 3520-3527, 2009. doi: 10.1109/IJCNN.2009.5179093.

[23] W. Yan, C. Ling, L. Ling, H. He, Generalized tensor equations with leading structured tensors, Appl. Math. Comput. 361 (2019), 311–324.

[24] G.E. Fasshauer, M.J. McCourt, Stable evaluation of gaussian radial basis function interpolants, SIAM J. Sci. Computing, 34 (2012), A737–A762.

[25] G.E. Fasshauer, Meshfree approximation methods with MATLAB, vol. 6, World Scientific, Singapore, 2007.