CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.SO-2023-0045

Vol. **x**, No. x, pp. 1-25
xx 2024

A Scalable Optimization Approach for the Multilinear System Arising from Scattered Data Interpolation

Yannan Chen[1,*], Kaidong Fu[1], Can Li[1,2] and Qi Ye[1]

[1] *School of Mathematical Sciences, South China Normal University, Guangzhou 510631, P.R. China.*
[2] *School of Mathematics and Statistics, Honghe University, Mengzi 661199, P.R. China.*

Received 16 September 2023; Accepted 19 March 2024

Abstract. Scattered data interpolation aims to reconstruct a continuous (smooth) function that approximates the underlying function by fitting (meshless) data points. There are extensive applications of scattered data interpolation in computer graphics, fluid dynamics, inverse kinematics, machine learning, etc. In this paper, we consider a novel generalized Mercel kernel in the reproducing kernel Banach space for scattered data interpolation. The system of interpolation equations is formulated as a multilinear system with a structural tensor, which is an absolutely and uniformly convergent infinite series of symmetric rank-one tensors. Then we design a fast numerical method for computing the product of the structural tensor and any vector in arbitrary precision. Whereafter, a scalable optimization approach equipped with limited-memory BFGS and Wolfe line-search techniques is customized for solving these multilinear systems. Using the Łojasiewicz inequality, we prove that the proposed scalable optimization approach is a globally convergent algorithm and possesses a linear or sublinear convergence rate. Numerical experiments illustrate that the proposed scalable optimization approach can improve the accuracy of interpolation fitting and computational efficiency.

AMS subject classifications: 15A69, 65H10, 90C30, 90C90
Key words: Scattered data interpolation, generalized Mercel kernel, structural tensor, multilinear system, optimization, Łojasiewicz inequality.

1 Introduction

Scattered data consists of a set of data sites $X:=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}\subset\Omega$ and corresponding values $V:=\{f_1,\ldots,f_N\}\subset\mathbb{R}$, where $\Omega\subseteq\mathbb{R}^d$ is locally compact. Here, "scattered" means that

*Corresponding author. *Email addresses:* ynchen@scnu.edu.cn (Y. Chen), yeqi@m.scnu.edu.cn (Q. Ye), 2020021905@m.scnu.edu.cn (K. Fu), 2021010122@m.scnu.edu.cn (C. Li)

the data sites have no structure or order between their relative locations. The values come from an underlying (not necessarily known) function. Scattered data interpolation aims to reconstruct a (typically smooth) function $s(\mathbf{x})$ that approximates the underlying function and particularly satisfies

$$s(\mathbf{x}_i)=f_i, \quad \forall i=1,\dots,N. \tag{1.1}$$

Scattered data interpolation has extensive applications in computer graphics [2], fluid dynamics [14], inverse kinematics [17], machine learning [37], etc. When data sites enjoy a well mesh geometry, plenty of methods such as wavelets, multivariant splines, and finite elements have been used for the interpolation problem (1.1). However, in the context of scattered data interpolation, meshless methods including radial basis functions [9] and kernel-based approximations [37] are promising. The monograph [41] provides more details. We focus on kernel-based methods in this paper.

To separate the influence between data sites and values, functions $u_j:\mathbb{R}^d\to\mathbb{R}$ $(j=1,\dots,N)$ that depend only on X are chosen to form

$$s(\mathbf{x})=\sum_{j=1}^{N} f_j u_j(\mathbf{x}).$$

We note that the Lagrange interpolation is equipped with $u_j(\mathbf{x}_i)=\delta_{ij}$ for $i,j\in\{1,\dots,N\}$, where δ_{ij} stands for the Kronecker delta. However, due to the Mairhuber-Curtis theorem [41, Theorem 2.3], continuous functions $\{u_j\}$ satisfying $u_j(\mathbf{x}_i)=\delta_{ij}$ may not always exist and be unique when $d\geq 2$.

Kernel methods [37] are a class of simple and powerful approaches for solving the interpolation problem (1.1). The reproducing kernel Hilbert space (RKHS) provides a reproducing kernel $K:\Omega\times\Omega\to\mathbb{R}$ and the associated interpolation function is defined as

$$s(\mathbf{x})=\sum_{j=1}^{N} c_j K(\mathbf{x},\mathbf{x}_j), \tag{1.2}$$

where c_j for $j=1,\dots,N$ are undetermined real coefficients. The kernel K ensures that matrix $A:=[K(\mathbf{x}_i,\mathbf{x}_j)]\in\mathbb{R}^{N\times N}$ is positive definite or conditionally positive definite for any set $X\subset\Omega$ of data sites [41]. Fasshauer and Ye [16] gave a unified theory for conditionally positive definite kernels. Combining the interpolation condition (1.1) and the interpolation function (1.2), we get the following linear system:

$$A\mathbf{c}=\mathbf{b}, \tag{1.3}$$

where $\mathbf{c}:=(c_1,\dots,c_N)^T$ is an undetermined vector and $\mathbf{b}:=(f_1,\dots,f_N)^T$ is the value vector. By solving the linear system (1.3), we determine coefficients in (1.2) and hence obtain the interpolation function $s(\mathbf{x})$.

Using the novel generalized Mercel kernel in the reproducing kernel Banach space (RKBS) [15], Xu and Ye [43] proposed the following system of polynomial equations for

the interpolation problem (1.1):

$$\sum_{n\in\mathbb{N}}\left(\sum_{k=1}^{N}c_k\phi_n(\mathbf{x}_k)\right)^{2m-1}\phi_n(\mathbf{x}_i)=f_i,\quad\forall i=1,\ldots,N,\tag{1.4}$$

where m is a positive integer and c_k's are undetermined real coefficients. We will review the derivative process and the definition of ϕ_n in Section 2. From the viewpoint of multilinear algebra, there exists a $2m$-th order N dimensional tensor

$$\mathcal{A}=[a_{i_1\ldots i_{2m}}]:=\left(\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x}_{i_1})\cdots\phi_n(\mathbf{x}_{i_{2m}})\right)\in\mathbb{R}^{[2m,N]}.$$

Since all of the entries of $\mathcal{A}$ are invariant under any index permutation, $\mathcal{A}$ is called a symmetric tensor. Further, for $\mathbf{c}=(c_1,\ldots,c_N)^T\in\mathbb{R}^N$, we define

$$\mathcal{A}\mathbf{c}^{2m}:=\sum_{i_1,\ldots,i_{2m}}a_{i_1\cdots i_{2m}}c_{i_1}\cdots c_{i_{2m}}\in\mathbb{R}.$$

It is straightforward to see

$$\mathcal{A}\mathbf{c}^{2m}=\sum_{n\in\mathbb{N}}\left(\sum_{k=1}^{N}c_k\phi_n(\mathbf{x}_k)\right)^{2m}\geq 0.$$

Hence, $\mathcal{A}$ is a positive semidefinite tensor. The system of polynomial equations (1.4) is indeed a multilinear system

$$\mathcal{A}\mathbf{c}^{2m-1}=\mathbf{b},\tag{1.5}$$

where

$$\mathcal{A}\mathbf{c}^{2m-1}:=\left[\sum_{i_2,\cdots,i_{2m}}a_{ii_2\cdots i_{2m}}c_{i_2}\cdots c_{i_{2m}}\right]_{i=1}^{N}\in\mathbb{R}^N.$$

It is easy to see that the multilinear system (1.5) reduces to a linear system (1.3) when $m=1$. Recently, Ye [44] pointed out that the multilinear system (1.5) exists the unique solution.

Moreover, in numerical partial differential equations [4,13], data mining [26], and tensor complementarity problems [31], multilinear systems (1.5) play important roles [35]. To solve multilinear systems, there are roughly three kinds of methods.

First, multilinear systems are a special class of algebraic systems of polynomial equations, which can be solved by elimination theorem [38] and homotopy continuation [18,24].

Second, by exploiting structural tensors in multilinear systems, various tensor algorithms were studied extensively. Li and Ng [26] proposed Jacobi and Gauss-Seidel iterative methods for solving sparse nonnegative tensor equations. When the coefficient

tensor of a multilinear system is a nonsingular M-tensor, Ding and Wei [13] proved that the multilinear system has a unique positive solution if the right-hand side is a positive vector. Then, they utilized the Jacobi iteration, the Gauss-Seidel iteration, and Newton's method for solving multilinear systems with M-tensors. Liu *et al.* [29] designed a tensor splitting method for multilinear systems with strong M-tensors. Furthermore, Li *et al.* [25] proposed preconditioned tensor splitting methods. Xie *et al.* [42] solved the multilinear system with symmetric M-tensors by some tensor methods. He *et al.* [19] showed that solving multilinear systems with M-tensors is equivalent to solving nonlinear systems of equations where the involving functions are P-functions. Based on this result, they proposed a Newton-type method to solve multilinear systems with M-tensors. Wang *et al.* [39, 40] utilized continuous time neural network for solving multilinear systems with M-tensors.

Finally, for solving general multilinear systems, well-designed nonlinear optimization algorithms become an active research issue. Li *et al.* [23] extended Jacobi, Gauss-Seidel and successive over-relaxation iterative methods for system of linear equations to solve general multilinear equations. Under mild conditions, these tensor splitting methods were proved to be globally convergent and locally R-linearly convergent. Li *et al.* [27] developed a hybrid alternating projection algorithm for solving three-order tensor equations. Lv and Ma [32] used a Levenberg-Marquardt method for solving multilinear systems and proved its global convergence and locally quadratic convergence under the local error bound condition.

For solving large-scale multilinear systems, it is a time-consuming and difficult calculation problem to compute the tensor-vector product $\mathcal{A}\mathbf{c}^{2m-1}$, where $\mathcal{A} \in \mathbb{R}^{[2m,N]}$ and $\mathbf{c} \in \mathbb{R}^N$. Many existing algorithms were interested in dense tensors (e.g. M-tensors and nonnegative tensors) that require about $\mathcal{O}(N^{2m})$ flops to compute the tensor-vector product. In this paper, by exploiting the special structure of an absolutely and uniformly convergent infinite series of symmetric rank-one tensors, we propose a fast computational method for the tensor-vector product, which only costs about $\mathcal{O}(NP)$ flops. Numerical experiments confirm this advantage. For related works on fast computations of the products between structural tensors and vectors, we refer to [11, 12].

In this paper, we concentrate on the multilinear system with a novel class of structural tensors, which arise from the generalized Mercel kernel for scattered data interpolation. The structural tensor is an absolutely and uniformly convergent infinite series of symmetric rank-one tensors. Since the sum tensor of the infinite series is unavailable, we turn to an approximate finite sum of symmetric rank-one tensors, i.e. a symmetric canonical polyadic tensor. By exploiting the canonical polyadic tensor, a fast computation method is proposed for computing the product of the novel structural tensor and any vector in arbitrary precision. Whereafter, using this fast computation, a first-order optimization approach equipped with limited-memory BFGS and Wolfe line search techniques is customized for solving multilinear systems with canonical polyadic tensors. Using the Łojasiewicz inequality, the proposed optimization approach is globally convergent with a locally linear or sublinear convergent rate. Numerical experiments on

one and two dimensional scattered data interpolation problems illustrate that the tensor-based multilinear system and the customized optimization approach can improve the accuracy of interpolation fitting and computational efficiency.

The remainder of this paper is organized as follows. We give in Section 2 some preliminary knowledge on the generalized Mercel kernel for scattered data interpolation and the derivation of structural multilinear systems. In Section 3, a fast numerical approach for computing the product of the structural tensor and any vector is proposed. Then a scalable optimization algorithm is customized for solving these multilinear systems. Convergence analysis of the proposed scalable optimization algorithm is presented in Section 4. In Section 5, preliminary numerical experiments on one and two dimensional interpolation problems are conducted to evaluate the efficiency of the proposed method. Finally, we give a brief conclusion in Section 6.

2 Preliminary

At the beginning, we review classical Mercer kernels in RKHS. Then, generalized Mercer kernels are introduced in RKBS. As a byproduct of generalized Mercer kernels, the multilinear system, which is of most interest to us, is derived.

2.1 Mercer kernels

Let Ω be a locally compact Housdorff space equipped with regular Borel measure μ and let $\mathrm{C}(\Omega\times\Omega)$ be the set of continuous functions on $\Omega\times\Omega$. The Mercer theorem says that a continuous symmetric positive definite kernel $K\in\mathrm{C}(\Omega\times\Omega)$ has countable positive eigenvalues $\{\lambda_n\}$ and associated continuous eigenfunctions $\{e_n\}$, that is,

$$\int_\Omega K(\mathbf{x},\mathbf{y})e_n(\mathbf{x})\mu(\mathrm{d}\mathbf{x})=\lambda_n e_n(\mathbf{y}),\quad \forall n\in\mathbb{N}.$$

Furthermore, the kernel K possesses the absolutely and uniformly convergent representation

$$K(\mathbf{x},\mathbf{y})=\sum_{n\in\mathbb{N}}\lambda_n e_n(\mathbf{x})e_n(\mathbf{y}),\quad \forall \mathbf{x},\mathbf{y}\in\Omega. \tag{2.1}$$

The kernel satisfying (2.1) is called a Mercer kernel. For convenience, we define

$$\phi_n:=\lambda_n^{\frac{1}{2}}e_n,\quad \forall n\in\mathbb{N}, \tag{2.2}$$

and rewrite the Mercer kernel as

$$K(\mathbf{x},\mathbf{y})=\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x})\phi_n(\mathbf{y}),\quad \forall \mathbf{x},\mathbf{y}\in\Omega.$$

The classical Mercer kernel is always a reproducing kernel of some separable RKHS and the Mercer kernel representation of this RKHS guarantees the isometrical isomorphism

onto the standard 2-norm space of countable sequences [41, Theorem 10.29]. Some frequently used Mercer kernels are as follows.

The univariate min kernel with the homogeneous boundary condition on $[0,1]$ is defined by

$$\Psi_1(x,y) := \min\{x,y\} - xy, \quad 0 \le x,y \le 1,$$

which is a special case of the Brownian bridge kernels [10,36,43]. Eigenvalues and eigenfunctions of the univariate min kernel are

$$\rho_n := \frac{1}{n^2\pi^2}, \quad \varphi_n(x) := \sqrt{2}\sin(n\pi x), \quad \forall n \in \mathbb{N},$$

respectively. Then, the associated integral-type min kernel is defined by

$$K_1(x,y) := \int_0^1 \Psi_1(x,z)\Psi_1(z,y)\mathrm{d}z = \begin{cases} -\frac{1}{6}(x^3 - x^3y - xy^3 + 3xy^2 - 2xy), & 0 \le x \le y \le 1, \\ -\frac{1}{6}(y^3 - xy^3 - x^3y + 3x^2y - 2xy), & 0 \le y \le x \le 1. \end{cases}$$

The eigenvalues and eigenfunctions of K_1 are respectively

$$\lambda_{1,n} := \rho_n^2, \quad e_{1,n} := \varphi_n, \quad \forall n \in \mathbb{N}.$$

Next, on the d-dimensional cubic $[0,1]^d$, we define the product kernel

$$\Psi_d(\mathbf{x},\mathbf{y}) = \prod_{k=1}^d \Psi_1(x_k,y_k),$$

where

$$\mathbf{x} := [x_k]_{k=1}^d, \quad \mathbf{y} := [y_k]_{k=1}^d \in [0,1]^d.$$

The d-dimensional integral-type kernel is represented as

$$K_d(\mathbf{x},\mathbf{y}) = \int_{[0,1]^d} \Psi_d(\mathbf{x},\mathbf{z})\Psi_d(\mathbf{z},\mathbf{y})\mathrm{d}\mathbf{z} = \prod_{k=1}^d K_1(x_k,y_k), \quad \forall \mathbf{x},\mathbf{y} \in [0,1]^d.$$

Eigenvalues and eigenfunctions of K_d are respectively

$$\lambda_{d,\mathbf{n}} := \prod_{k=1}^d \lambda_{1,n_k} = \prod_{k=1}^d \rho_{n_k}^2, \quad e_{d,\mathbf{n}} := \prod_{k=1}^d e_{1,n_k}(x_k) = \prod_{k=1}^d \varphi_{n_k}(x_k)$$

for $\mathbf{n} \in \mathbb{N}^d$. Then, functions $\phi_{d,\mathbf{n}}$'s can be defined by (2.2) similarly. Interested readers can refer to Xu and Ye [43] for more kinds of Mercer kernels.

2.2 Generalized Mercer kernel

Let Ω and Ω' be locally compact Housdorff spaces equipped with regular Borel measures μ and μ', respectively. We consider the kernel $K\in \mathrm{L}_0(\Omega\times\Omega')$, where $\mathrm{L}_0(\Omega\times\Omega')$ is the collection of all real measurable functions defined on $\Omega\times\Omega'$.

Definition 2.1 ([43, Definition 3.1]). *A kernel $K\in \mathrm{L}_0(\Omega\times\Omega')$ is called a generalized Mercer kernel induced by the left-sided and right-sided expansion sets*

$$\mathcal{S}_K:=\{\phi_n\,|\,n\in\mathbb{N}\}\subseteq \mathrm{L}_0(\Omega),\quad \mathcal{S}'_K:=\{\phi'_n\,|\,n\in\mathbb{N}\}\subseteq \mathrm{L}_0(\Omega'),$$

if the kernel K can be written as the pointwise convergent representation

$$K(\mathbf{x},\mathbf{y})=\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x})\phi'_n(\mathbf{y}),\quad \forall \mathbf{x}\in\Omega,\quad \forall \mathbf{y}\in\Omega'.$$

It is easy to see that classical Mercer kernels are symmetric, i.e. $K(\mathbf{x},\mathbf{y})=K(\mathbf{y},\mathbf{x})$, which are special cases of generalized Mercel kernels. Motivated by the relationship between RKHS with the classical Mercer kernel and the 2-norm space, Xu and Ye [43] studied the isometrical isomorphisms about p-norm and q-norm spaces.

Let $1<p,q<\infty$ such that $p^{-1}+q^{-1}=1$. The space

$$\mathcal{B}_K^p(\Omega):=\left\{s:=\sum_{n\in\mathbb{N}}a_n\phi_n\,\middle|\,\sum_{n\in\mathbb{N}}|a_n|^p<\infty\right\}$$

composed of functions defined on Ω is equipped with the semi-norm

$$\|s\|_{\mathcal{B}_K^p(\Omega)}:=\left(\sum_{n\in\mathbb{N}}|a_n|^p\right)^{\frac{1}{p}}.$$

Similarly, define the space $\mathcal{B}_{K'}^q(\Omega')$ on Ω' as

$$\mathcal{B}_{K'}^q(\Omega'):=\left\{\sum_{n\in\mathbb{N}}b_n\phi'_n\,\middle|\,\sum_{n\in\mathbb{N}}|b_n|^q<\infty\right\},$$

where K' is the adjoint kernel of K. Xu and Ye [43] pointed out that $\mathcal{B}_K^p(\Omega)$ and $\mathcal{B}_{K'}^q(\Omega')$ are isometrical isomorphisms of p-norm and q-norm spaces, respectively. Hence, the Gâteaux derivative of $s\in\mathcal{B}_K^p(\Omega)$ can be computed by the Gâteaux derivative of $\|\cdot\|_p$, i.e. the Gâteaux derivative of $s\in\mathcal{B}_K^p(\Omega)$ is

$$\sum_{n\in\mathbb{N}}\frac{a_n|a_n|^{p-2}}{\|\mathbf{a}\|_p^{p-1}}\phi'_n\in\mathcal{B}_{K'}^q(\Omega'),\tag{2.3}$$

where $\|\mathbf{a}\|_p:=(\sum_{n\in\mathbb{N}}|a_n|^p)^{1/p}$.

2.3 Derivation of the multilinear system

Now, we consider the problem of scattered data interpolation with a regularized empirical risk [43]

$$\mathcal{T}_p(s) := \frac{1}{N}\sum_{k=1}^{N} L(\mathbf{x}_k, y_k, s) + R\big(\|s\|_{\mathcal{B}_K^p(\Omega)}\big), \tag{2.4}$$

where the regular function $R:[0,\infty)\to[0,\infty)$ is convex and strictly increasing, and the loss function $L:\Omega\times\mathbb{R}\times\mathbb{R}\to[0,\infty)$ is defined such that $s\mapsto L(\mathbf{x},y,s)$ is a convex map for each fixed $\mathbf{x}\in\Omega$ and each fixed $y\in\mathbb{R}$. Suppose that the right-sided kernel set

$$\mathcal{K}_K' := \{K(\mathbf{x},\cdot)\,|\,\mathbf{x}\in\Omega\} \subseteq \mathcal{B}_{K'}^q(\Omega')$$

is linearly independent. Then, the global solution of (2.4) has the closed-form formula

$$s = \sum_{n\in\mathbb{N}} a_n\phi_n \in \mathcal{B}_K^p(\Omega),$$

the Gâteaux derivative of which is a linear combination

$$\sum_{k=1}^{N}\beta_k K(\mathbf{x}_k,\cdot) = \sum_{k=1}^{N}\beta_k\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x}_k)\phi_n' \in \mathcal{B}_{K'}^q(\Omega'). \tag{2.5}$$

Comparing coefficients between (2.3) and (2.5), we have

$$\frac{a_n|a_n|^{p-2}}{\|\mathbf{a}\|_p^{p-1}} = \sum_{k=1}^{N}\beta_k\phi_n(\mathbf{x}_k), \quad \forall n\in\mathbb{N}.$$

By taking $c_k := \|\mathbf{a}\|_p^{p-1}\beta_k = \|\mathbf{a}\|_p^{1/(q-1)}\beta_k$ for all $k=1,\ldots,N$, it yields

$$a_n|a_n|^{p-2} = \sum_{k=1}^{N} c_k\phi_n(\mathbf{x}_k), \quad \forall n\in\mathbb{N}.$$

Solving the above equation, we obtain

$$a_n = \sum_{j=1}^{N} c_j\phi_n(\mathbf{x}_j)\left|\sum_{k=1}^{N} c_k\phi_n(\mathbf{x}_k)\right|^{q-2}, \quad \forall n\in\mathbb{N}.$$

That is to say, the global solution of (2.4) is

$$s = \sum_{n\in\mathbb{N}}\left(\sum_{j=1}^{N} c_j\phi_n(\mathbf{x}_j)\left|\sum_{k=1}^{N} c_k\phi_n(\mathbf{x}_k)\right|^{q-2}\right)\phi_n.$$

Then the following theorem holds.

Theorem 2.1 ([43, Theorem 5.10]). *Consider the optimization problem* (2.4) *with* $q=2m$ *and* $p=2m/(2m-1)$. *Then the unique global solution of*

$$\min_{s\in\mathcal{B}_K^p(\Omega)} \mathcal{T}_p(s)$$

has the finite dimensional representation

$$s(\mathbf{x})=\sum_{n\in\mathbb{N}}\left(\sum_{k=1}^{N}c_k\phi_n(\mathbf{x}_k)\right)^{2m-1}\phi_n(\mathbf{x}),\quad \forall \mathbf{x}\in\Omega,$$

where coefficients $c_1,\ldots,c_N$ *are undetermined. The norm of* $s(\mathbf{x})$ *is*

$$\|s(\mathbf{x})\|_{\mathcal{B}_K^p(\Omega)}=\left(\sum_{n\in\mathbb{N}}\left(\sum_{k=1}^{N}c_k\phi_n(\mathbf{x}_k)\right)^{2m}\right)^{1-\frac{1}{2m}}.$$

Now, we are ready to derive a mathematical model, which is a multilinear system, for the application of scattered data interpolation. Given a set of scattered data sites $X:=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}\subseteq\Omega\subset\mathbb{R}^d$ and corresponding values $V:=\{f_1,\ldots,f_N\}\subset\mathbb{R}$, we try to find an interpolating function $s(\mathbf{x})\in\mathcal{B}_K^p(\Omega)$ such that $s(\mathbf{x}_i)=f_i$ for $i=1,\ldots,N$. According to Theorem 2.1, we can rewrite the interpolation condition as

$$\begin{aligned} f_i=s(\mathbf{x}_i)&=\sum_{n\in\mathbb{N}}\left(\sum_{k=1}^{N}c_k\phi_n(\mathbf{x}_k)\right)^{2m-1}\phi_n(\mathbf{x}_i)\\ &=\sum_{k_2=1}^{N}\cdots\sum_{k_{2m}=1}^{N}c_{k_2}\cdots c_{k_{2m}}\left(\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x}_i)\phi_n(\mathbf{x}_{k_2})\cdots\phi_n(\mathbf{x}_{k_{2m}})\right) \end{aligned} \tag{2.6}$$

for all $i=1,\ldots,N$. Let $\mathbf{u}_n=(\phi_n(\mathbf{x}_1),\ldots,\phi_n(\mathbf{x}_N))^T$ for $n\in\mathbb{N}$. We introduce a symmetric tensor

$$\mathcal{A}:=\sum_{n\in\mathbb{N}}\underbrace{\mathbf{u}_n\circ\cdots\circ\mathbf{u}_n}_{2m\text{ times}}\in\mathbb{R}^{[2m,N]}, \tag{2.7}$$

every element of which is an absolutely and uniformly convergent infinite series

$$[\mathcal{A}]_{i_1\ldots i_{2m}}=\sum_{n\in\mathbb{N}}[\mathbf{u}_n]_{i_1}\cdots[\mathbf{u}_n]_{i_{2m}}=\sum_{n\in\mathbb{N}}\phi_n(\mathbf{x}_{i_1})\cdots\phi_n(\mathbf{x}_{i_{2m}}),\quad \forall i_1,\ldots,i_{2m}\in\{1,\ldots,N\}.$$

Let $\mathbf{c}:=(c_1,\ldots,c_N)^T$. The system (2.6) is indeed a multilinear system

$$\mathcal{A}\mathbf{c}^{2m-1}=\mathbf{b}. \tag{2.8}$$

For example, when we take the min kernel on $[0,1]$, elements of the associated coefficient tensor $\mathcal{A}$ are

$$[\mathcal{A}]_{i_1\cdots i_{2m}}=\sum_{n\in\mathbb{N}}\frac{2^m}{n^{2m}\pi^{2m}}\sin(n\pi x_{i_1})\cdots\sin(n\pi x_{i_{2m}}),\quad \forall i_1,\ldots,i_{2m}\in\{1,\ldots,N\}. \tag{2.9}$$

When the integral-type min kernel on $[0,1]$ is used, elements of $\mathcal{A}$ are

$$[\mathcal{A}]_{i_1\cdots i_{2m}} = \sum_{n\in\mathbb{N}} \frac{2^m}{n^{4m}\pi^{4m}} \sin(n\pi x_{i_1})\cdots\sin(n\pi x_{i_{2m}}), \quad \forall i_1,\ldots,i_{2m}\in\{1,\ldots,N\}. \tag{2.10}$$

3 A scalable optimization approach

Since the coefficient tensor $\mathcal{A}$ of the multilinear system (2.8) is an absolutely and uniformly convergent infinite series, the exact tensor is unavailable at the numerical viewpoint. In this section, we propose to approximate the infinite series (2.7) by its partial sum

$$\widetilde{\mathcal{A}}_P = \sum_{n=1}^{P} \underbrace{\mathbf{u}_n\circ\cdots\circ\mathbf{u}_n}_{2m \text{ times}} \in \mathbb{R}^{[2m,N]}, \tag{3.1}$$

where the finite integer P is controlled by a predetermined tolerance ϵ.

Theorem 3.1. *Let $\epsilon>0$ be a small number. Then the error of elements* (2.9) *corresponding to the min kernel satisfies*

$$\left|\left[\widetilde{\mathcal{A}}_P-\mathcal{A}\right]_{i_1\cdots i_{2m}}\right| \le \epsilon,$$

if

$$P \ge \left\lceil [\epsilon(2m-1)]^{-\frac{1}{2m-1}} \left(\frac{\sqrt{2}}{\pi}\right)^{\frac{2m}{2m-1}} \right\rceil,$$

where $\lceil\alpha\rceil$ stands for the smallest integer that is not less than α.

Proof. For any $(x_{i_1},\ldots,x_{i_{2m}})$, the difference between $[\mathcal{A}]_{i_1\ldots i_{2m}}$ and its partial sum approximation $[\widetilde{\mathcal{A}}_P]_{i_1\cdots i_{2m}}$ is

$$\begin{aligned}
\left|\left[\widetilde{\mathcal{A}}_P-\mathcal{A}\right]_{i_1,\ldots,i_{2m}}\right| &= \left|\sum_{n=P+1}^{\infty} \frac{2^m}{n^{2m}\pi^{2m}} \sin(n\pi x_{i_1})\cdots\sin(n\pi x_{i_{2m}})\right| \\
&\le \left(\frac{\sqrt{2}}{\pi}\right)^{2m} \sum_{n=P+1}^{\infty} \frac{1}{n^{2m}} \left|\sin(n\pi x_{i_1})\cdots\sin(n\pi x_{i_{2m}})\right| \\
&\le \left(\frac{\sqrt{2}}{\pi}\right)^{2m} \sum_{n=P+1}^{\infty} \int_{n-1}^{n} \frac{1}{n^{2m}}\mathrm{d}t \le \left(\frac{\sqrt{2}}{\pi}\right)^{2m} \sum_{n=P+1}^{\infty} \int_{n-1}^{n} \frac{1}{t^{2m}}\mathrm{d}t \\
&= \left(\frac{\sqrt{2}}{\pi}\right)^{2m} \int_{P}^{+\infty} \frac{1}{t^{2m}}\mathrm{d}t \\
&= \left(\frac{\sqrt{2}}{\pi}\right)^{2m} \frac{1}{2m-1} P^{1-2m} \le \epsilon.
\end{aligned}$$

Hence, the theorem is valid. □

Using a similar discussion, we get the following theorem.

Theorem 3.2. *Let $\epsilon>0$ be a small number. Then the error of elements* (2.10) *corresponding to the integral-type min kernel satisfies*

$$\left|[\widetilde{\mathcal{A}}_P-\mathcal{A}]_{i_1\dots i_{2m}}\right|\leq\epsilon,$$

if

$$P\geq\left\lceil[\epsilon(4m-1)]^{\frac{-1}{4m-1}}\left(\frac{\sqrt[4]{2}}{\pi}\right)^{\frac{4m}{4m-1}}\right\rceil.$$

Theorems 3.1 and 3.2 mean that tensors (2.9) and (2.10) could be approximated by their finite partial sums in arbitrary precision. Some truncated parameter P's corresponding to various tolerance ϵ's with $m=2$ are listed in Table 1. We note that Theorems 3.1 and 3.2 for min kernels can be extended to other kinds of Mercer kernels straightforwardly.

Table 1: Truncated parameter P corresponds to tolerance ϵ.

ϵ ($m=2$)	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}	10^{-16}
P corresponds to (2.9)	24	112	516	2393	11104	51537
P corresponds to (2.10)	2	4	7	13	25	49

Next, we focus on the slightly inexact tensor $\widetilde{\mathcal{A}}_P$, which is a canonical polyadic tensor and hence enjoys some fast computation approaches [6,7].

Lemma 3.1. *The tensor $\widetilde{\mathcal{A}}_P$ is a canonical polyadic tensor which is symmetric and positive semidefinite. Define $U:=[\mathbf{u}_1,\dots,\mathbf{u}_P]\in\mathbb{R}^{N\times P}$. Then we have*

$$\widetilde{\mathcal{A}}_P\mathbf{c}^{2m}=\|U^T\mathbf{c}\|_{2m}^{2m},\quad \widetilde{\mathcal{A}}_P\mathbf{c}^{2m-1}=U\mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1},$$

where $\mathbf{1}\in\mathbb{R}^P$ is an all-one vector and $\|\mathbf{v}\|_{2m}:=(\sum_{n=1}^P|v_n|^{2m})^{1/(2m)}$ is a norm for $\mathbf{v}\in\mathbb{R}^P$.

Proof. It is straightforward to say that $\widetilde{\mathcal{A}}_P$ is a symmetric canonical polyadic tensor. From (3.1), the i-th element of $\widetilde{\mathcal{A}}_P\mathbf{c}^{2m-1}$ is

$$\begin{aligned}\left[\widetilde{\mathcal{A}}_P\mathbf{c}^{2m-1}\right]_i&=\sum_{k_2=1}^N\cdots\sum_{k_{2m}=1}^N\sum_{n=1}^P[\mathbf{u}_n]_i[\mathbf{u}_n]_{k_2}\cdots[\mathbf{u}_n]_{k_{2m}}\mathbf{c}_{k_2}\dots\mathbf{c}_{k_{2m}}\\&=\sum_{n=1}^P[\mathbf{u}_n]_i\left(\sum_{k_2=1}^N[\mathbf{u}_n]_{k_2}\mathbf{c}_{k_2}\right)\cdots\left(\sum_{k_{2m}}^N[\mathbf{u}_n]_{k_{2m}}\mathbf{c}_{k_{2m}}\right)\\&=\sum_{n=1}^P[\mathbf{u}_n]_i\left([U^T\mathbf{c}]_n\right)^{2m-1}\\&=\mathbf{e}_i^TU\mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1},\end{aligned}$$

where $\mathbf{e}_i \in \mathbb{R}^N$ is the ith column of an identity matrix. Thus, we say

$$\widetilde{\mathcal{A}}_P \mathbf{c}^{2m-1} = U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1}.$$

Moreover, it holds that

$$\widetilde{\mathcal{A}}_P \mathbf{c}^{2m} = \mathbf{c}^T \left(\widetilde{\mathcal{A}}_P \mathbf{c}^{2m-1}\right) = \mathbf{1}^T \mathrm{diag}(U^T \mathbf{c})^{2m} \mathbf{1} = \|U^T \mathbf{c}\|_{2m}^{2m}.$$

Hence, tensor $\widetilde{\mathcal{A}}_P$ is positive semidefinite. □

Lemma 3.1 reveals a fast computation method for calculating the product of the tensor $\widetilde{\mathcal{A}}_P \in \mathbb{R}^{[2m,N]}$ and any vector $\mathbf{c} \in \mathbb{R}^N$. We compare two approaches. (I) When the dense tensor $\widetilde{\mathcal{A}}_P \in \mathbb{R}^{[2m,N]}$ is stored explicitly, we need a lot of memory and $\mathcal{O}(N^{2m})$ flops for the tensor-vector product. (II) We prefer to store a matrix $U \in \mathbb{R}^{N \times P}$. By Lemma 3.1, the computational cost for $\widetilde{\mathcal{A}}_P \mathbf{c}^{2m-1}$ is only about $\mathcal{O}(NP)$ flops. We will give a numerical examination in Fig. 2.

By substituting the interpolation equation (2.8) into the optimization model (2.4), we get the following approximate model:

$$\min_{\mathbf{c} \in \mathbb{R}^N} f(\mathbf{c}) = \left\| \widetilde{\mathcal{A}}_P \mathbf{c}^{2m-1} - \mathbf{b} \right\|_2^2 + \sigma \widetilde{\mathcal{A}}_P \mathbf{c}^{2m}, \tag{3.2}$$

where $\widetilde{\mathcal{A}}_P \in \mathbb{R}^{[2m,N]}$ is defined in (3.1), $\mathbf{c}$ is the unknown vector, σ is a positive parameter, and m is a positive integer. Model (3.2) is a nonlinear nonconvex optimization. Since the objective function of (3.2) is smooth, we customize an efficient gradient-based iterative algorithm. Now, a fast computational method for computing function values and gradient vectors of the objective function of (3.2) is presented in the following theorem.

Theorem 3.3. *Let $U = [\mathbf{u}_1, \ldots, \mathbf{u}_P] \in \mathbb{R}^{N \times P}$. Then we have*

$$f(\mathbf{c}) = \|U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} - \mathbf{b}\|_2^2 + \sigma \|U^T \mathbf{c}\|_{2m}^{2m}. \tag{3.3}$$

The gradient of $f(\mathbf{c})$ is

$$\begin{aligned} \nabla f(\mathbf{c}) = {} & (4m-2) U \mathrm{diag}(U^T \mathbf{c})^{2m-2} U^T [U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} - \mathbf{b}] \\ & + \sigma 2m U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1}. \end{aligned}$$

Proof. According to Lemma 3.1, we immediately get (3.3). By computing the total derivative of $f(\mathbf{c})$, we have

$$\begin{aligned} \mathrm{d} f(\mathbf{c}) = {} & \mathrm{d} \left\langle U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} - \mathbf{b}, U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} - \mathbf{b} \right\rangle \\ & + \sigma \mathrm{d} \left\langle U^T \mathbf{c}, \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} \right\rangle \\ = {} & 2 \left\langle \mathrm{d}[U \mathrm{diag}(U^T \mathbf{c})^{2m-2} U^T \mathbf{c}], U \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} - \mathbf{b} \right\rangle \\ & + \sigma 2m \left\langle \mathrm{d}[U^T \mathbf{c}], \mathrm{diag}(U^T \mathbf{c})^{2m-1} \mathbf{1} \right\rangle \end{aligned}$$

$$
\begin{aligned}
&= 2\big\langle (2m-1)U\mathrm{diag}(U^T\mathbf{c})^{2m-2}U^T\mathrm{d}\mathbf{c}, U\mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1}-\mathbf{b}\big\rangle \\
&\quad + \sigma 2m\big\langle U^T\mathrm{d}\mathbf{c}, \mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1}\big\rangle \\
&= (4m-2)\big\langle \mathrm{d}\mathbf{c}, U\mathrm{diag}(U^T\mathbf{c})^{2m-2}U^T[U\mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1}-\mathbf{b}]\big\rangle \\
&\quad + \sigma 2m\big\langle \mathrm{d}\mathbf{c}, U\mathrm{diag}(U^T\mathbf{c})^{2m-1}\mathbf{1}\big\rangle \\
&= \langle \mathrm{d}\mathbf{c}, \nabla f(\mathbf{c})\rangle,
\end{aligned}
$$

which verifies the gradient presented in the lemma. □

We note that the computational cost for evaluating a function value $f(\mathbf{c})$ and a gradient vector $\nabla f(\mathbf{c})$ is about $\mathcal{O}(NP)$ flops. This motivates us to develop a scalable optimization algorithm.

Algorithm 1: A Scalable Optimization Algorithm.

Select $\mathbf{c}_0 \in \mathbb{R}^N$, choose positive numbers $\kappa_3 < \kappa_4 < 1, \gamma_{lbd} < \gamma_{ubd}, L_{ubd}, \epsilon$ and ε.
Evaluate $f(\mathbf{c}_0)$ and $\nabla f(\mathbf{c}_0)$ by Theorem 3.3, choose $\mathbf{d}_0 = -\nabla f(\mathbf{c}_0)$, and set $t \leftarrow 0$.
while $\|\nabla f(\mathbf{c}_t)\|_\infty > \varepsilon$ **do**
  Find a stepsize α_t satisfying (3.5) and (3.6) by the Wolfe line search.
  Update an iterate $\mathbf{c}_{t+1} = \mathbf{c}_t + \alpha_t \mathbf{d}_t$.
  Evaluate $f(\mathbf{c}_{t+1})$ and $\nabla f(\mathbf{c}_{t+1})$ by the fast computation in Theorem 3.3.
  Calculate and save $\mathbf{s}_t = \mathbf{c}_{t+1} - \mathbf{c}_t$ and $\mathbf{y}_t = \nabla f(\mathbf{c}_{t+1}) - \nabla f(\mathbf{c}_t)$.
  Initialize $\mathbf{q} \leftarrow -\nabla f(\mathbf{c}_t)$ and $L \leftarrow \min(t, L_{ubd})$.
  for $i = t, t-1, \ldots, t+1-L$ **do**
    Calculate and save $\beta_i \leftarrow \rho_i \mathbf{s}_i^T \mathbf{q}$.
    Update $\mathbf{q} \leftarrow \mathbf{q} - \beta_i \mathbf{y}_i$.
  end for
  Select $\gamma_t \in [\gamma_{lbd}, \gamma_{ubd}]$ and set $\mathbf{d} \leftarrow \gamma_t \mathbf{q}$.
  for $i = t+1-L, t-L, \ldots, t$ **do**
    Calculate $\zeta \leftarrow \rho_i \mathbf{y}_i^T \mathbf{d}$.
    Update $\mathbf{d} \leftarrow \mathbf{d} + (\beta_i - \zeta)\mathbf{s}_i$.
  end for
  Update a descent direction $\mathbf{d}_{t+1} = \mathbf{d}$.
  Set $t \leftarrow t+1$.
end while

To solve the unconstrained minimization (3.2), we utilize the limited memory BFGS (L-BFGS) technique [33, 34], which is an efficient quasi-Newton algorithm for large-scale optimization. Given an initial iterate $\mathbf{c}_0 \in \mathbb{R}^N$ and the approximate inverse $H_0 \in \mathbb{R}^{N\times N}$ of an initial Hessian that is symmetric and positive definite, we set $t \leftarrow 0$ and compute a descent direction

$$\mathbf{d}_t = -H_t \nabla f(\mathbf{c}_t),$$

which satisfies

$$\mathbf{d}_t^T\nabla f(\mathbf{c}_t)\leq-\kappa_1\|\nabla f(\mathbf{c}_t)\|_2^2,\quad \|\mathbf{d}_t\|_2\leq\kappa_2\|\nabla f(\mathbf{c}_t)\|_2, \tag{3.4}$$

where $0<\kappa_1\leq 1\leq\kappa_2$. Whereafter, the Wolfe inexact line search is performed along the descent direction $\mathbf{d}_t$ to find a stepsize α_t such that

$$f(\mathbf{c}_t+\alpha_t\mathbf{d}_t)\leq f(\mathbf{c}_t)+\kappa_3\alpha_t\mathbf{d}_t^T\nabla f(\mathbf{c}_t), \tag{3.5}$$
$$\mathbf{d}_t^T\nabla f(\mathbf{c}_t+\alpha_t\mathbf{d}_t)\geq\kappa_4\mathbf{d}_t^T\nabla f(\mathbf{c}_t), \tag{3.6}$$

where $0<\kappa_3<\kappa_4<1$. Then the new iterate is defined as

$$\mathbf{c}_{t+1}=\mathbf{c}_t+\alpha_t\mathbf{d}_t. \tag{3.7}$$

Define $\mathbf{s}_t:=\mathbf{c}_{t+1}-\mathbf{c}_t$ and $\mathbf{y}_t:=\nabla f(\mathbf{c}_{t+1})-\nabla f(\mathbf{c}_t)$. Owing to (3.6), (3.7) and (3.4), it holds that

$$\begin{aligned}\mathbf{s}_t^T\mathbf{y}_t&=\alpha_t\mathbf{d}_t^T\big(\nabla f(\mathbf{c}_t+\alpha_t\mathbf{d}_t)-\nabla f(\mathbf{c}_t)\big)\geq\alpha_t(\kappa_4-1)\mathbf{d}_t^T\nabla f(\mathbf{c}_t)\\&\geq\alpha_t\kappa_1(1-\kappa_4)\|\nabla f(\mathbf{c}_t)\|_2^2>0,\end{aligned}$$

when $\mathbf{c}_t$ is not a stationary point. Once H_t is symmetric positive definite, BFGS generates a new symmetric positive definite matrix

$$H_{t+1}=V_t^TH_tV_t+\rho_t\mathbf{s}_t\mathbf{s}_t^T,\quad V_t:=I-\rho_t\mathbf{y}_t\mathbf{s}_t^T,\quad \rho_t:=\frac{1}{\mathbf{s}_t^T\mathbf{y}_t}.$$

Hence, the vector pair $\{\mathbf{s}_t,\mathbf{y}_t\}$ captures Hessian information at iteration t.

To solve large scale optimization problems, L-BFGS only adopts $L:=\min(t,L_{ubd})$ latest vector pairs $\{\mathbf{s}_t,\mathbf{y}_t\}$. Starting from a simple matrix

$$H_t^{(0)}=\gamma_tI$$

with $\gamma_t\in[\gamma_{lbd},\gamma_{ubd}]\subset(0,\infty)$, L-BFGS updates $H_t^{(\ell)}$ recursively

$$H_t^{(\ell)}=V_{t+\ell-L}^TH_t^{(\ell-1)}V_{t+\ell-L}+\rho_{t+\ell-L}\mathbf{s}_{t+\ell-L}\mathbf{s}_{t+\ell-L}^T,\quad\forall\ell=1,\ldots,L.$$

The positive parameter γ_t could be chosen from Barzilai-Borwein stepsizes [8] and improved variants [21]. In this way, there exist positive constants $0<\kappa_1\leq 1\leq\kappa_2$ such that the new descent direction $\mathbf{d}_{t+1}=-H_t^{(L)}\nabla f(\mathbf{c}_{t+1})$ satisfies (3.4). Then we use $H_{t+1}=H_t^{(L)}$ and set $t\leftarrow t+1$.

In numerical computation, L-BFGS can be implemented by a fast two-loop recursion which costs about $\mathcal{O}(LN)$ flops. Finally, a complete algorithm is presented in Algorithm 1 formally.

4 Convergence analysis

If Algorithm 1 terminates finitely, i.e. there exists an iterate $\mathbf{c}_t$ such that $\nabla f(\mathbf{c}_t)=0$, we immediately know that $\mathbf{c}_t$ is a first-order stationary point. So, in the remainder of this section, we assume that Algorithm 1 generates an infinite sequence of iterates $\{\mathbf{c}_t\}$.

Theorem 4.1. *Assume that the gradient ∇f is Lipschitz continuous, i.e. there exists a constant $L_1>0$ such that*

$$\|\nabla f(\mathbf{c})-\nabla f(\tilde{\mathbf{c}})\|_2\le L_1\|\mathbf{c}-\tilde{\mathbf{c}}\|_2,\quad \forall \mathbf{c},\tilde{\mathbf{c}}\in\mathbb{R}^N. \tag{4.1}$$

Then,

$$\lim_{t\to\infty}\|\nabla f(\mathbf{c}_t)\|_2=0.$$

Proof. From the second Wolfe condition (3.6) and the Lipschitz continuity (4.1) of ∇f , we have

$$(\kappa_4-1)\mathbf{d}_t^T\nabla f(\mathbf{c}_t)\le \mathbf{d}_t^T\big(\nabla f(\mathbf{c}_{t+1})-\nabla f(\mathbf{c}_t)\big)\le \alpha_t L_1\|\mathbf{d}_t\|_2^2,$$

which means

$$\alpha_t\ge -\frac{1-\kappa_4}{L_1}\frac{\mathbf{d}_t^T\nabla f(\mathbf{c}_t)}{\|\mathbf{d}_t\|_2^2}. \tag{4.2}$$

By substituting this inequality into the first Wolfe condition (3.5), we obtain

$$f(\mathbf{c}_t)-f(\mathbf{c}_{t+1})\ge \kappa_3\frac{(1-\kappa_4)}{L_1}\frac{\big(\mathbf{d}_t^T\nabla f(\mathbf{c}_t)\big)^2}{\|\mathbf{d}_t\|_2^2}\ge\frac{\kappa_1^2\kappa_3(1-\kappa_4)}{\kappa_2^2L_1}\|\nabla f(\mathbf{c}_t)\|_2^2,$$

where the last inequality holds owing to (3.4). By summing this expression over all indices $t=0,1,\ldots,T$ with T being an iterative number, we obtain

$$f(\mathbf{c}_0)\ge f(\mathbf{c}_0)-f(\mathbf{c}_{T+1})=\sum_{t=0}^{T}f(\mathbf{c}_t)-f(\mathbf{c}_{t+1})\ge\frac{\kappa_1^2\kappa_3(1-\kappa_4)}{\kappa_2^2L_1}\sum_{t=0}^{T}\|\nabla f(\mathbf{c}_t)\|_2^2,\quad\forall T.$$

Hence, by letting $T\to\infty$, it holds that

$$\sum_{t=0}^{\infty}\|\nabla f(\mathbf{c}_t)\|_2^2\le\frac{\kappa_2^2L_1f(\mathbf{c}_0)}{\kappa_1^2\kappa_3(1-\kappa_4)}<+\infty,$$

That is to say $\|\nabla f(\mathbf{c}_t)\|_2\to0$ as $t\to\infty$. Every accumulation point of iterates $\{\mathbf{c}_t\}$ generated by Algorithm 1 is a stationary point. □

Since the objective function $f(\mathbf{c})$ defined in (3.2) is a polynomial, the following Łojasiewicz property holds.

Theorem 4.2 (Łojasiewicz property [30])**.** *Suppose that $\mathbf{c}_*$ is a stationary point of $f(\mathbf{c})$. Then there exists a neighborhood $\mathbb{U}$ of $\mathbf{c}_*$, an exponent $\theta\in[0,1)$, and a positive constant κ_K such that*

$$|f(\mathbf{c})-f(\mathbf{c}_*)|^\theta\le\kappa_K\|\nabla f(\mathbf{c})\|_2$$

for all $\mathbf{c}\in\mathbb{U}$. Here, we define $0^0\equiv0$.

Using the Łojasiewicz property, we show that the sequence of iterates generated by Algorithm 1 converges to a unique accumulation point.

Theorem 4.3. *Suppose that the sequence of iterates* $\{\mathbf{c}_t\} \subset \mathbb{R}^N$ *generated by Algorithm 1 is bounded. Then there exists a single point* $\mathbf{c}_* \in \mathbb{R}^N$ *such that*

$$\lim_{t\to\infty} \mathbf{c}_t = \mathbf{c}_*.$$

Further, under the assumption of Theorem 4.1, *we say that* $\mathbf{c}_*$ *is a stationary point.*

Proof. According to [1, Theorem 3.2], we only need to prove two strong descent conditions

$$\begin{aligned}&[f(\mathbf{c}_{t+1}) = f(\mathbf{c}_t)] \;\Rightarrow\; [\mathbf{c}_{t+1} = \mathbf{c}_t],\\ &f(\mathbf{c}_t) - f(\mathbf{c}_{t+1}) \geq \kappa_5 \|\nabla f(\mathbf{c}_t)\|_2 \|\mathbf{c}_t - \mathbf{c}_{t+1}\|_2\end{aligned}$$

for all t and for some $\kappa_5 > 0$. The first condition holds since Algorithm 1 is a descent algorithm.

From the Wolfe condition (3.5), the descent condition (3.4), and (3.7), we have

$$\begin{aligned}f(\mathbf{c}_t) - f(\mathbf{c}_{t+1}) &\geq -\kappa_3 \alpha_t \mathbf{d}_t^T \nabla f(\mathbf{c}_t)\\ &\geq \kappa_1 \kappa_3 \alpha_t \|\nabla f(\mathbf{c}_t)\|_2^2\\ &\geq \frac{\kappa_1 \kappa_3}{\kappa_2} \alpha_t \|\nabla f(\mathbf{c}_t)\|_2 \|\mathbf{d}_t\|_2\\ &= \frac{\kappa_1 \kappa_3}{\kappa_2} \|\nabla f(\mathbf{c}_t)\|_2 \|\mathbf{c}_t - \mathbf{c}_{t+1}\|_2.\end{aligned}$$

Let $\kappa_5 := \kappa_1 \kappa_3 / \kappa_2$. This theorem is proved. □

To estimate the convergence rate of Algorithm 1, the following lemma is valuable.

Lemma 4.1. *Assume that the gradient* ∇f *is Lipschitz continuous, i.e.* ∇f *satisfies* (4.1). *Then there exists a positive constant* κ_6 *such that*

$$\|\mathbf{c}_{t+1} - \mathbf{c}_t\|_2 \geq \kappa_6 \|\nabla f(\mathbf{c}_t)\|_2.$$

Proof. According to (4.2) and (3.4), we obtain

$$\|\mathbf{c}_{t+1} - \mathbf{c}_t\|_2 = \alpha_t \|\mathbf{d}_t\|_2 \geq \frac{1-\kappa_4}{L_1} \frac{-\mathbf{d}_t^T \nabla f(\mathbf{c}_t)}{\|\mathbf{d}_t\|_2} \geq \frac{\kappa_1 (1-\kappa_4)}{\kappa_2 L_1} \|\nabla f(\mathbf{c}_t)\|_2.$$

Let $\kappa_6 := \kappa_1 (1-\kappa_4) / (\kappa_2 L_1)$, we get this lemma. □

Using Lemma 4.1, the following theorem can be proved by a similar discussion in [3, Theorem 2] and [20, Theorem 3.2].

Theorem 4.4. *Under assumptions of Theorems* 4.1 *and* 4.3*, we have the following estimations of convergence rate.*

- *If* $\theta \in (0,1/2]$*, there exist* $\kappa_7 > 0$ *and* $\varrho \in (0,1)$ *such that*

$$\|\mathbf{c}_t - \mathbf{c}_*\|_2 \leq \kappa_7 \varrho^t.$$

- *If* $\theta \in (1/2,1)$*, there exist* $\kappa_8 > 0$ *and* $\kappa_9 > 0$ *such that*

$$\|\mathbf{c}_t - \mathbf{c}_*\|_2 \leq \kappa_8 t^{-\frac{1-\theta}{2\theta-1}}, \quad |f(\mathbf{c}_t) - f(\mathbf{c}_*)| \leq \kappa_9 t^{-\frac{1}{2\theta-1}}.$$

Because the objective function $f:\mathbb{R}^N \to \mathbb{R}$ defined in (3.2) is a polynomial with degree $4m-2$, an estimator [20] of the Łojasiewicz exponent is

$$\theta = 1 - \frac{1}{(4m-2)(12m-9)^{N-1}},$$

which gets close to one. Hence, $(1-\theta)/(2\theta-1) \to 0$ and $1/(2\theta-1) \to 1$ as $m \to \infty$ or $N \to \infty$. That is to say, the objective function value generated by Algorithm 1 at least exhibits a sublinear convergence rate of $\mathcal{O}(1/t)$ without assuming strong convexity.

5 Numerical experiments

We are going to illustrate the effectiveness and the efficiency of the proposed scalable optimization algorithm equipped with the L-BFGS quasi-Newton and the Wolfe inexact line search techniques. On parameters of L-BFGS, we use at most $L_{ubd} = 5$ latest $\{\mathbf{s}_t, \mathbf{y}_t\}$ vector pairs to construct an approximate inverse of Hessian and choose the parameter $\gamma_t = \mathbf{s}_t^T \mathbf{y}_t / (\mathbf{y}_t^T \mathbf{y}_t)$ at each iteration. For the Wolfe inexact line search, parameters κ_3 and κ_4 are set to 0.1 and 0.5, respectively. The proposed algorithm terminates if $\|\nabla f(\mathbf{c}_t)\|_\infty < 10^{-5}\|\nabla f(\mathbf{c}_0)\|_\infty$ or the number of iterations exceeds one thousand.

5.1 One-dimensional interpolation

For the purpose of choosing a proper truncated parameter P, we consider a univariate function

$$f(x) = x\sin(20\pi x), \quad x \in [0,1]. \tag{5.1}$$

One hundred random points are sampled uniformly from $[0,1]$ and associated function values are evaluated to form the scattered data. Let $m = 2$. The tensor $\mathcal{A}$ is a fourth order 100 dimensional symmetric tensor. Using the min kernel (2.9), we test truncated error ϵ ranging from 10^{-6} to 10^{-12}. The corresponding P's by Theorem 3.1 are calculated. For each ϵ, we test one thousand pieces of random scattered data and compute the average absolute error

$$\text{AAE} := \frac{1}{1000}\sum_{k=1}^{1000} \max_{x\in[0,1]} \left|\hat{f}_k(x) - f(x)\right|,$$

where $\hat{f}_k(x)$ is the interpolation function generated from the k-th piece of random scattered data. Numerical results are reported in Table 2. It is easy to see that moderate truncated error $\epsilon = 10^{-7}$ produces the best AAE. Small ϵ can not improve the quality of the interpolation function but costs expensive computing resources. Thus, we fix the truncated error $\epsilon = 10^{-7}$ in the remainder experiments.

Next, we compare the novel tensor method, which uses the tensor kernel (2.9), with the classical matrix method with a Mercer kernel. For a typical piece of random scatted data, curves of interpolation functions generated by tensor and matrix kernels are illustrated in the Fig. 1. The thick green curve denotes the truth graph of the function (5.1) and magenta circles stand for positions of scattered sites. The matrix method seems to use blue line segments to connect adjacent points. When the original function oscillates frequently, the interpolation error of matrix method is significant. The function generated by the tensor method illustrated by a red dotted curve seems smooth and captures peaks and valleys of the original function.

To evaluate the efficiency of the fast computation approach proposed in Lemma 3.1, we compare two approaches for computing the product of the coefficient tensor and a vector. One approach deals with a dense tensor $\widetilde{\mathcal{A}}_P$. The other one exploits the canonical polyadic structure and works with U. When the number of data sites varies from fifty to twenty thousand, the scalable optimization algorithm with these two approaches cost

Table 2: AAE corresponds to truncated error ϵ with the min kernel.

ϵ	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
P	24	52	112	240	516	1111	2393
AAE	0.0283	0.0151	0.0168	0.0170	0.0170	0.0170	0.0169

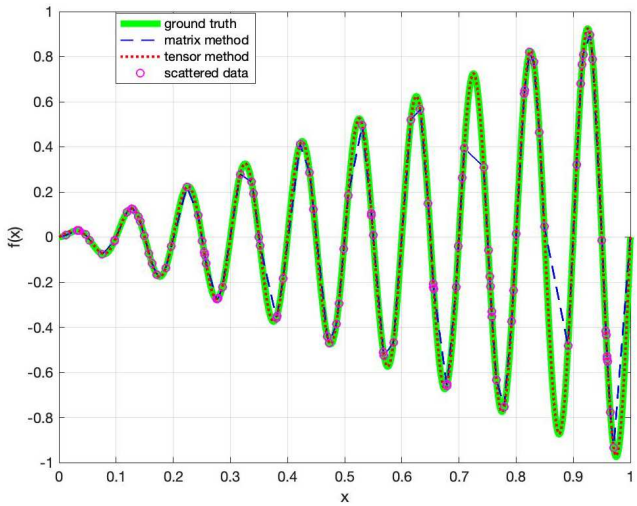

Figure 1: Curves of interpolation functions within one dimension.

different CPU times, which are listed in Fig. 2. Obviously, the proposed fast computation method is rather efficient.

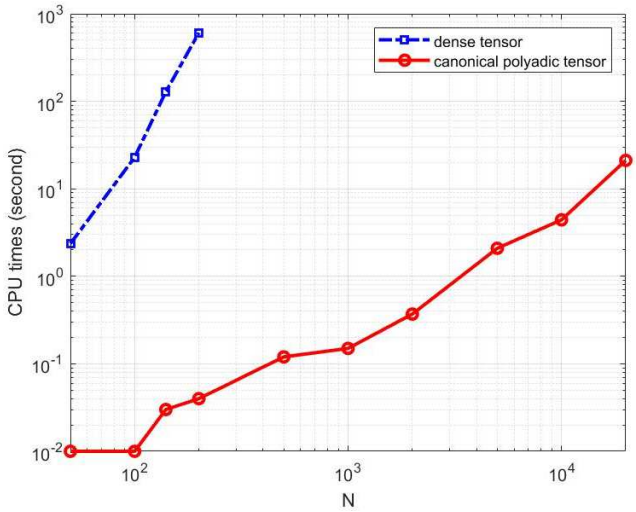

Figure 2: Comparison of CPU times.

5.2 Two-dimensional interpolation

Now, we turn to two-dimensional scattered data interpolation. Six binary functions will be tested, viz.

$$f_1(x,y)=\frac{1}{3}\exp\left(-\frac{81}{4}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right),$$

$$f_2(x,y)=\frac{1.25+\cos(5.4y)}{6+6(3x-1)^2},$$

$$f_3(x,y)=\frac{1}{9}\left(\tanh(9-9x-9y)+1\right),$$

$$f_4(x,y)=2\exp\left(-30\left(\left(x-\frac{1}{3}\right)^2+\left(y-\frac{1}{3}\right)^2\right)\right)-\exp\left(-20\left(\left(x-\frac{2}{3}\right)^2+\left(y-\frac{2}{3}\right)^2\right)\right),$$

$$f_5(x,y)=-\exp\left(-\frac{1}{5}\sqrt{\frac{x^2+y^2}{2}}\right)+\frac{1}{20}\left[e-\exp\left(\frac{\cos(2\pi x)+\cos(2\pi y)}{2}\right)\right]+1,$$

$$f_6(r,\theta)=\begin{cases}\exp\left(-\dfrac{1}{1-r^2}\right)\left[1-\dfrac{4r^4}{4r^4+(1-r^2)^4}\sin\left(\theta-\dfrac{1}{1-r^2}\right)\right], & \text{if} \quad r<1,\\ 0, & \text{if} \quad r\geq 1,\end{cases}$$

where f_1, f_2, f_3 are from [22] and f_5 is the Ackley function [5], and f_6 is a smooth "Mexican hat" function [1] with (r,θ) standing for polar coordinates in $\mathbb{R}^2$.

For each function, we randomly select $N=500$ data sites and evaluate associated function values to perform scattered data interpolation. For comparison, we test a MATLAB built-in routine "scatteredInterpolant", the matrix kernel model (1.3) in RKHS, and the tensor kernel model (3.2) in RKBS. Surfaces of interpolation functions generated by these three methods are illustrated in Fig. 3. The first column draws surfaces of original functions. Interpolation function images generated by a MATLAB built-in routine, the matrix kernel approach, and the tensor kernel approach are illustrated in the second to the last columns, respectively. In Table 3, we report the maximal errors and average errors between original functions and corresponding interpolation functions produced by these

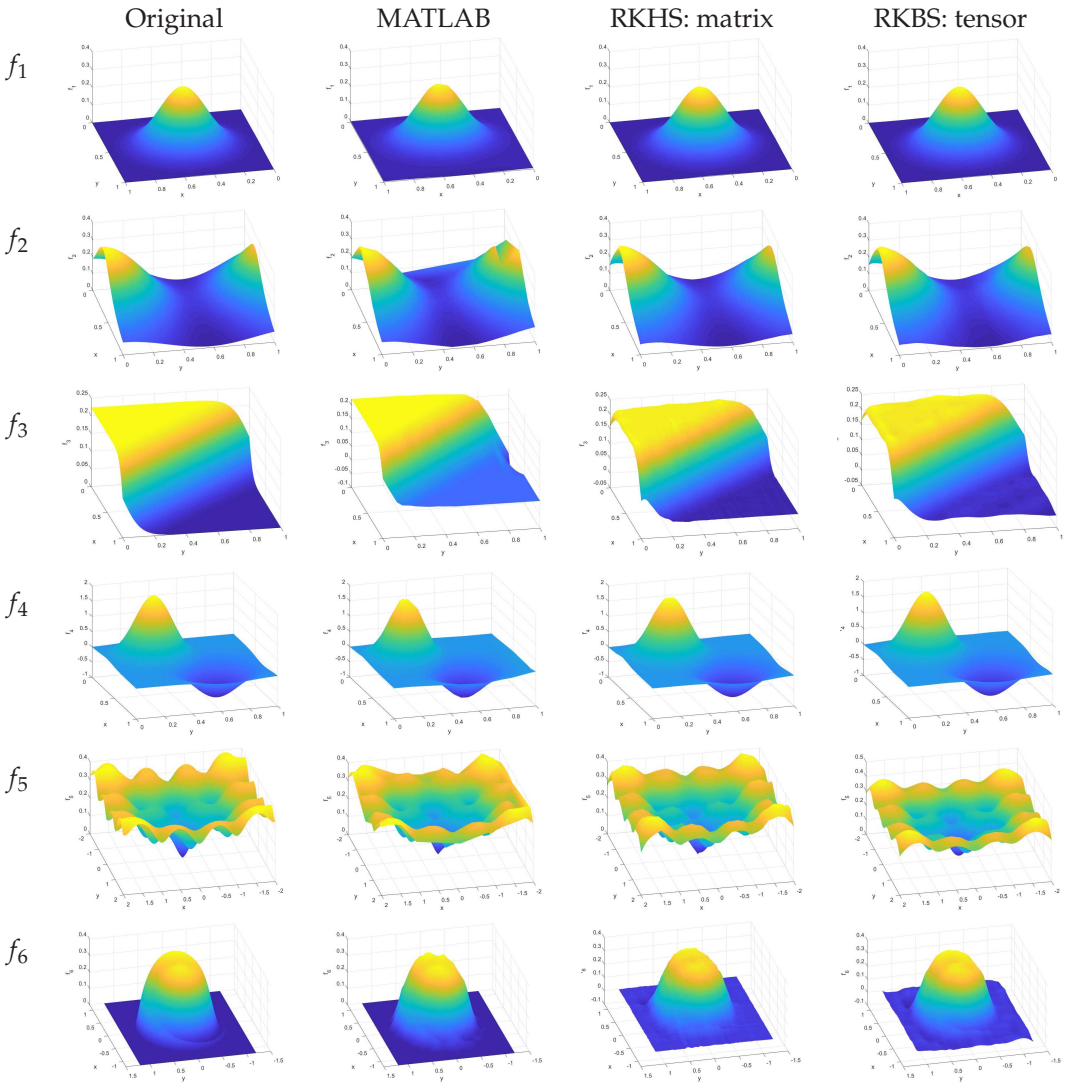

Figure 3: Surfaces of interpolation functions within two dimension.

Table 3: Errors of interpolation surfaces.

	Maximal errors					
Methods	f_1	f_2	f_3	f_4	f_5	f_6
MATLAB	0.0991	0.0927	0.1223	0.5459	0.1035	0.0548
RKHS: matrix	0.0090	0.0390	0.0659	0.0902	0.0939	0.0772
RKBS: tensor	0.0014	0.0334	0.0586	0.0161	0.0610	0.0538
	Average errors					
Methods	f_1	f_2	f_3	f_4	f_5	f_6
MATLAB	0.0143	0.0240	0.0127	0.1014	0.0067	0.0047
RKHS: matrix	0.0005	0.0007	0.0024	0.0050	0.0064	0.0072
RKBS: tensor	0.0002	0.0017	0.0033	0.0029	0.0047	0.0070

three interpolation methods. It can be seen that function images generated by the MATLAB built-in routine and the matrix kernel approach are not smooth enough compared to the original image, and the images generated by the tensor method seem better. This means that our proposed algorithm recovers original functions better.

5.3 Numerical partial differential equations

The last experiment concentrates on numerical partial differential equations. It is well-known that some finite element methods for solving numerical partial differential equations have the superconvergence property [28], i.e. the numerical solution at some points is higher precision than that of the overall domain. Then, a postprocessing, e.g. an interpolation, of these high-precision points can produce a better numerical solution.

In this subsection, we consider the following one-dimensional heat equation with a nonlinear term

$$\begin{cases} \pi^2 \dfrac{\partial u}{\partial t} = \dfrac{\partial}{\partial x}\left(\dfrac{\partial u}{\partial x}\right) + \dfrac{1}{2}u^2, \quad 0 \le x \le 1, \quad t \ge 0, \\ u(x,0) = \sin(\pi x), \quad u(0,t) = u(1,t) = 0. \end{cases} \tag{5.2}$$

Since there exists a nonlinear term in the heat equation, the closed-form solution of (5.2) is unavailable. Now, we use a MATLAB built-in routine "pdepe" to produce a numerical solution illustrated in Fig. 4(a).

Using the MATLAB built-in routine "scatteredInterpolant", the matrix kernel based interpolation in RKHS and the tensor kernel based interpolation in RKBS, we obtain three surfaces illustrated in Figs. 4(b)-4(d), respectively. To examine in details, we plot curves at times $t = 0, 0.5, 1, 1.5, 2$ in Fig. 5. Obviously, the MATLAB built-in routine "scatteredInterpolant" and the matrix kernel based interpolation only connect data points with line segments at $t = 0$, while the tensor kernel based interpolation produces a smooth curve. For $t = 0.5$, the matrix kernel based interpolation still uses line segments, the MATLAB built-in routine "scatteredInterpolant" and the tensor kernel based interpolation produce

curves. By enlarging the picture, we see that the curve produced by tensor kernel based interpolation is smoother than that of the matrix kernel.

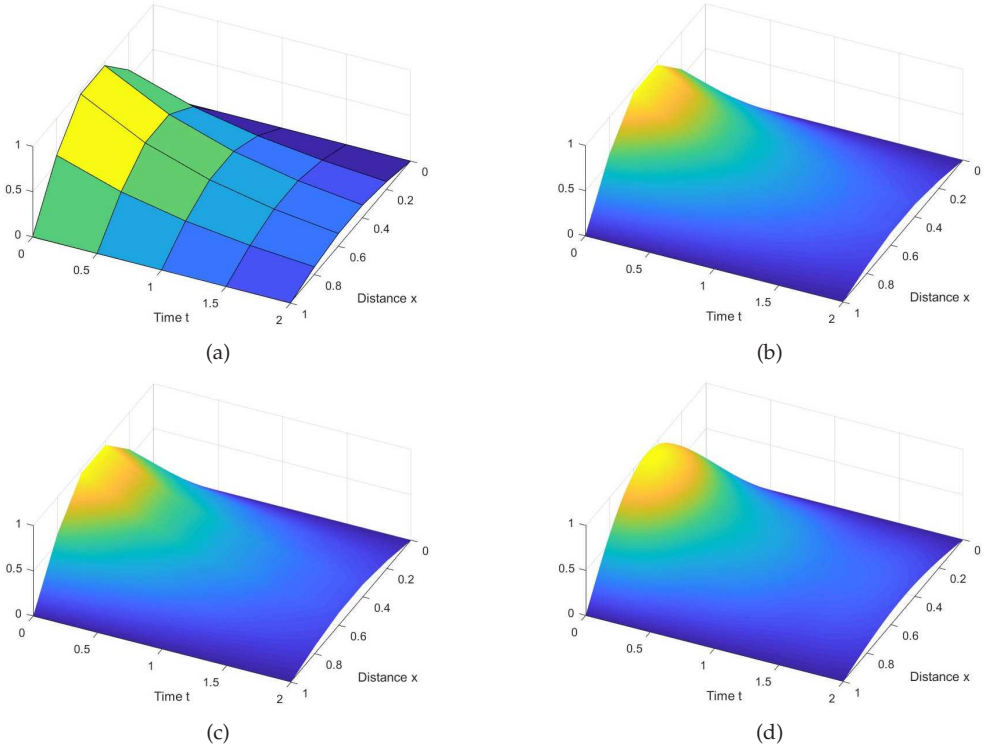

Figure 4: Interpolation surfaces of a numerical solution of (5.2).

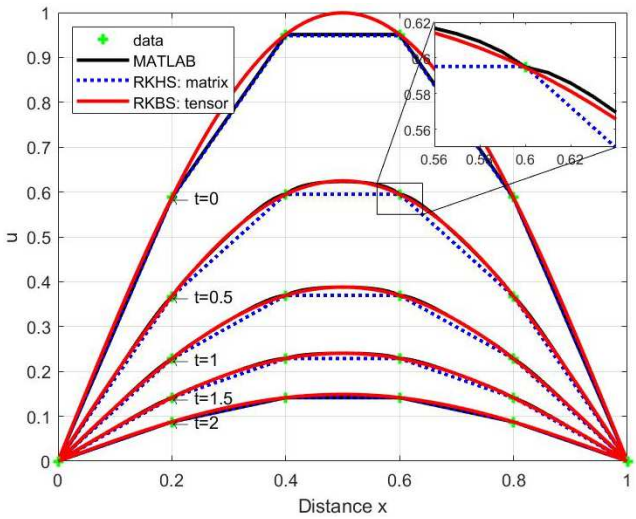

Figure 5: Curves at various times.

6 Conclusion

By exploiting the low rank structure of canonical polyadic tensors, we designed a fast computation method for computing the product of the tensor and any vector in arbitrary precision. Then the fast approach for computing the objective function and the associated gradient vector were derived straightforwardly. Using nonlinear optimization methods, we customized a scalable optimization algorithm equipped with the L-BFGS quasi-Newton and the Wolfe inexact line search techniques for solving scattered data interpolation problems. Numerical experiments illustrated the effectiveness and efficiency of our algorithm. For further works, we may consider other kernel functions besides the min kernel and effective algorithms to solve problem (3.2) directly.

Acknowledgments

The authors are grateful to the associate editor and three anonymous referees for their comments which helped us to improve our manuscript essentially. We thank Dr. Rongrong Lin for the valuable discussion on Mercer kernels.

This research was supported by the National Natural Science Foundation of China (Grant Nos. 12171168, 12071159 and 12071157), and by the Yunnan Natural Science Foundation (Grant No. 202101BA070001-047).

References

[1] P. A. Absil, R. Mahony, and B. Andrews, *Convergence of the iterates of descent methods for analytic cost functions*, SIAM J. Optim., 16:531–547, 2005.
[2] K. Anjyo, J. P. Lewis, and F. Pighin, *Scattered data interpolation for computer graphics*, in: Proceeding SIGGRAPH '14 ACM SIGGRAPH 2014 Courses, 27:1–69, 2014.
[3] H. Attouch and J. Bolte, *On the convergence of the proximal algorithm for nonsmooth functions involving analytic features*, Math. Program., 116:5–16, 2009.
[4] P. Azimzadeh and E. Bayraktar, *High order Bellman equations and weakly chained diagonally dominant tensors*, SIAM J. Matrix Anal. Appl., 40:276–298, 2019.
[5] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, 1996.
[6] B. W. Bader and T. G. Kolda, *Algorithm 862: MATLAB tensor classes for fast algorithm prototyping*, ACM Trans. Math. Softw., 32:635–653, 2006.
[7] B. W. Bader and T. G. Kolda, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30:205–231, 2008.
[8] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8:141–148, 1988.
[9] M. D. Buhmann, *Radial basis functions*, Acta Numer., 9:1–38, 2000.
[10] R. Cavoretto, G. E. Fasshauer, and M. McCourt, *An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels*, Numer. Algorithms, 68:393–422, 2015.
[11] J. Chang, Y. Chen, and L. Qi, *Computing eigenvalues of large scale sparse tensors arising from a hypergraph*, SIAM J. Sci. Comput., 38:A3618–A3643, 2016.

[12] W. Ding, L. Qi, and Y. Wei, *Fast Hankel tensor-vector product and its application to exponential data fitting*, Numer. Linear Algebra Appl., 22:814–832, 2015.
[13] W. Ding and Y. Wei, *Solving multi-linear systems with $\mathcal{M}$-tensors*, J. Sci. Comput., 68:689–715, 2016.
[14] G. E. Fasshauer, *Solving differential equations with radial basis functions: Multilevel methods and smoothing*, Adv. Comput. Math., 11:139–159, 1999.
[15] G. E. Fasshauer, F. J. Hickernell, and Q. Ye, *Solving support vector machines in reproducing kernel Banach spaces with positive definite functions*, Appl. Comput. Harmon. Anal., 38:115–139, 2015.
[16] G. E. Fasshauer and Q. Ye, *Reproducing kernels of generalized Sobolev spaces via a Green function approach with distributional operators*, Numer. Math., 119:585–611, 2011.
[17] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, *Style-based inverse kinematics*, ACM Trans. Graph., 23(3):522–531, 2004.
[18] L. Han, *A homotopy method for solving multilinear systems with M-tensors*, Appl. Math. Lett., 69:49–54, 2017.
[19] H. He, C. Ling, L. Qi, and G. Zhou, *A globally and quadratically convergent algorithm for solving multilinear systems with $\mathcal{M}$-tensors*, J. Sci. Comput., 76:1718–1741, 2018.
[20] S. Hu and G. Li, *Convergence rate analysis for the higher order power method in best rank one approximations of tensors*, Numer. Math., 140:993–1031, 2018.
[21] Y. Huang, Y. Dai, and X. Liu, *Equipping the Barzilai-Borwein method with the two dimensional quadratic termination property*, SIAM J. Optim., 31:3068–3096, 2021.
[22] S. Lee, G. Wolberg, and S. Y. Shin, *Scattered data interpolation with multilevel B-splines*, IEEE Trans. Vis. Comput. Graph., 3(3):228–244, 1997.
[23] D. Li, S. Xie, and H. Xu, *Splitting methods for tensor equations*, Numer. Linear Algebra Appl., 24(5):e2102, 2017.
[24] T. Y. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Acta Numer., 6:399–436, 1997.
[25] W. Li, D. Liu, and S. W. Vong, *Comparison results for splitting iterations for solving multi-linear systems*, Appl. Numer. Math., 134:105–121, 2018.
[26] X. Li and M. K. Ng, *Solving sparse non-negative tensor equations: Algorithms and applications*, Front. Math. China, 10:649–680, 2015.
[27] Z. Li, Y. Dai, and H. Gao, *Alternating projection method for a class of tensor equations*, J. Comput. Appl. Math., 346:490–504, 2019.
[28] Q. Lin, L. Tobiska, and A. Zhou, *Superconvergence and extrapolation of non-conforming low order finite elements applied to the Poisson equation*, IMA J. Numer. Anal., 25:160–181, 2005.
[29] D. Liu, W. Li, and S. W. Vong, *The tensor splitting with application to solve multi-linear systems*, J. Comput. Appl. Math., 330:75–94, 2018.
[30] S. Łojasiewicz, *Une propriété topologique des sous-ensembles analytiques réels*, in: Les Équations aux Dérivées Partielles, Éditions du centre National de la Recherche Scientifique, 87–89, 1963.
[31] Z. Luo, L. Qi, and N. Xiu, *The sparsest solutions to Z-tensor complementarity problems*, Optim. Lett., 11:471–482, 2017.
[32] C. Lv and C. Ma, *A Levenberg-Marquardt method for solving semi-symmetric tensor equations*, J. Comput. Appl. Math., 332:13–25, 2018.
[33] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comp., 35:773–782, 1980.
[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.
[35] L. Qi, H. Chen, and Y. Chen, *Tensor Eigenvalues and Their Applications*, in: Advances in Me-

chanics and Mathematics, Vol. 39, Springer, 2018.
[36] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005.
[37] R. Schaback and H. Wendland, *Kernel techniques: From machine learning to meshless methods*, Acta Numer., 15:543–639, 2006.
[38] B. Sturmfels, *Solving Systems of Polynomial Equations*, AMS Chelsea Pub./AMS, 2002.
[39] X. Wang, M. Che, and Y. Wei, *Neural networks based approach solving multi-linear systems with M-tensors*, Neurocomputing, 351:33–42, 2019.
[40] X. Wang, M. Che, and Y. Wei, *Neural network approach for solving nonsingular multi-linear tensor systems*, J. Comput. Appl. Math., 368:112569, 2020.
[41] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, 2004.
[42] Z. Xie, X. Jin, and Y. Wei, *Tensor methods for solving symmetric $\mathcal{M}$-tensor systems*, J. Sci. Comput., 74:412–425, 2018.
[43] Y. Xu and Q. Ye, *Generalized Mercer kernels and reproducing kernel Banach spaces*, Mem. Amer. Math. Soc., 258:1243, 2019.
[44] Q. Ye, *Positive definite multi-kernels for scattered data interpolations*, Appl. Comput. Harmon. Anal., 62:251–260, 2023.