SUBSPACE METHODS WITH LOCAL REFINEMENTS FOR EIGENVALUE COMPUTATION USING LOW-RANK TENSOR-TRAIN FORMAT

JUNYU ZHANG[†], ZAIWEN WEN[‡], AND YIN ZHANG[§]

Abstract. Computing a few eigenpairs from large-scale symmetric eigenvalue problems is far beyond the tractability of classic eigensolvers when the storage of the eigenvectors in the classical way is impossible. We consider a tractable case in which both the coefficient matrix and its eigenvectors can be represented in the low-rank tensor train formats. We propose a subspace optimization method combined with some suitable truncation steps to the given low-rank Tensor Train formats. Its performance can be further improved if the alternating minimization method is used to refine the intermediate solutions locally. Preliminary numerical experiments show that our algorithm is competitive to the state-of-the-art methods on problems arising from the discretization of the stationary Schrödinger equation.

Key words. high-dimensional eigenvalue problem, tensor-train format, alternating least square method, subspace optimization method

1. Introduction.

The need for computing a set of smallest or largest eigenvalues and their corresponding eigenvectors frequently arises from diverse fields in statistics, signal processing, data mining or compression, and from nonlinear eigenvalues problems in electronic structure calculations. Given a symmetric matrix $A \in \mathbb{R}^{N\times N}$ and an integer $p \ll N$, finding the p-smallest eigenpairs is equivalent to solving the trace-minimization problem

$$\min_{U\in\mathbb{R}^{N\times p}} \mathbf{Tr}(U^\top AU), \quad \text{s.t.} \quad U^\top U = I_p, \tag{1.1}$$

where $\mathbf{Tr}(A)$ is the summation of the diagonal elements of a square matrix A and I_p is the order p identity matrix. In this paper, we are interested in large-scale problems whose size N can be up to 10^{42}. For example, when solving the Schrödinger equation in quantum mechanics, the discretized matrix can be as large as $n^d \times n^d$, where n stands for the discretized point in one dimension and d represents the dimension. Even the storage of the matrix A or the eigenvectors U may be impossible and it is far beyond the tractability of classic eigensolvers.

The low-rank tensor representation enables us to handle the high-dimensional problem (1.1) efficiently. For instance, by viewing a vector in $\mathbb{R}^{n^d}$ as the vectorization of a tensor in $\mathbb{R}^{\overbrace{n\times\cdots\times n}^{d}}$, and viewing matrices in $\mathbb{R}^{n^d\times n^d}$ as a reshaped tensor in $\mathbb{R}^{\overbrace{n\times\cdots\times n}^{2d}}$, these vectors and matrices often possess good low-rank structures or allow low-rank approximations with a high accuracy, especially for discretized eigenvalue problems from partial differential equations (PDE). In fact, low-rank tensor representation is a promising tool in data compression and there have been a wide variety of low-rank tensor formats, including canonical polyadic (CP), Tucker, Hierarchical Tucker, tensor-train (TT) and quantized tensor-train (QTT) formats. The interested reader is referred to [10] and the references therein for an overview of CP and Tucker formats, and to [14, 4, 13] for the properties and operations on Hierarchical Tucker format. A relatively complete introduction of the basic properties of the TT format is provided in [19]. The recent development of the TT format and its various

[†]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, USA (zhan4393@umn.edu). Research supported in part by NSF Grant CMMI-1462408.

[‡]Beijing International Center for Mathematical Research, Peking University, Beijing, CHINA (wenzw@pku.edu.cn). Research supported in part by NSFC grant 11322109, and by the National Basic Research Project under the grant 2015CB856000.

[§]Department of Computational and Applied Mathematics, Rice University, Houston, UNITED STATES (yzhang@rice.edu). Research supported in part by NSF DMS-1115950 and NSF DMS-1418724.

applications can be found in the survey paper [5]. The QTT format [18] is a further compression of the TT format and the application of the QTT format can often speed up methods using the TT format.

In this paper, we choose the BTT format because it has a simple structure and convenient numerical operations. Our goal is to solve (1.1) under a low-rank BTT format $\mathbf{T}$, where the detailed definitions of the BTT format $\mathbf{T}$ will be introduced in section 2. Suppose that the matrix A in (1.1) itself can be expressed in the TT format. Let $\mathbf{U}$ be the BTT representation of U. Then the trace-minimization problem becomes

$$\min_{U\in\mathbb{R}^{N\times p}} \mathbf{Tr}(U^\top AU), \quad \text{s.t.} \quad U^\top U = I_p \text{ and } \mathbf{U}\in\mathbf{T}, \tag{1.2}$$

where $\mathbf{U}\in\mathbf{T}$ means that all operations are performed in the BTT format. Due to the tractability of the operations involving the BTT format, problem (1.2) is much more computational friendly than the original problem (1.1).

Algorithms for solving (1.2) highly rely on the representations of the BTT format. The standard iterative eigensolvers have been extended with low-rank tensor representation for matrices and vectors/block vectors in [1, 12, 15]. In particular, the so-called locally optimal preconditioned conjugate gradient (LOBPCG) method [9] is applied to the Hierarchical Tucker format in [12] for calculating only the smallest eigenvalue. Although these tensor formats can cover high dimensional data with a relatively low rank and the classic iterative methods can solve certain examples, they exhibit convergence and robustness issues. Another difficulty is that the rank of some intermediate vectors increases iteration by iteration and they have to be compressed to lower rank carefully. The most successful framework is based on the technique of alternating minimization (block coordinate descent) method, such as alternating linear scheme (ALS) and its variations [2, 8, 12]. Basically, the variables in the BTT format are expressed blocks by blocks naturally according to its structure. Then the objective function is minimized with respect to only one block of variables while all other blocks are fixed at each iteration. Applying the unfolding properties of the BTT format, this subproblem itself is a linear eigenvalue problem of a smaller size and can be solved efficiently by standard eigensolvers. An algorithm called EVAMEn is developed in [11]. It also uses the ALS framework but enriches each core locally based on preconditioned residual information. The subproblems are in the same fashion as ALS and they again are solved by the LOBPCG method. Numerical examples show that EVAMEn can be more efficient than ALS in a few cases.

In this paper, we propose a truncated subspace optimization method for solving (1.2) whose solutions are represented in the low-rank BTT formats. The main challenge of a direct extension of the subspace optimization method [16] is that the TT-ranks increase dramatically after several operations between tensor formats, such as the addition in the TT formats and matrix-vector multiplications in the TT format. Consequently, the computational cost of all subsequent operations becomes more and more expensive and eventually prohibitive as the TT-ranks increase. Our strategy is to add projections to the given TT formats at some suitable places so that the overall computational cost is still tractable while the accuracy is still maintained. This scheme itself can be interpreted as an inexact alternating minimization procedure between finding eigenpairs in a subspace and enforcing the given low-rank tensor representation. Its performance can be further improved if the ALS method is used to refine the intermediate solutions locally. Preliminary numerical experiments show that our algorithm is quite promising comparing to the ALS method and EVAMEn on problems arising from the discretization of the stationary Schrödinger equation. In particular, our method may has its advantage when solving the linear eigenvalue problem in the ALS method and EVAMEn is still time consuming.

Organization and notations. The rest of this paper is organized as follows. The low-rank TT and BTT formats as well as the operator TT format are introduced in section 2. Some properties of these TT formats are reviewed in section 3. In section 4, we propose the truncated subspace optimization method with local refinements. Numerical results are presented in section 5. Finally, we conclude the paper in section 6.

We adopt the following notation. The operations $x \otimes y$ and $x \odot y$ denote the Kronecker and Hadamard products between x and y, respectively. The ordinary upper case letters denote matrices, including a block of vectors; the ordinary lower case letters denote vectors; the bold-faced upper case letters represent tensors and the tensor representation for matrices and a block of vectors; the bold-faced lower case letters represent tensors and and the tensor representation for vectors. When a same letter appears in both ordinary and bold-faced letterforms, for example $\mathbf{u}$ and u, then u is a vector and $\mathbf{u}$ is the tensor representation of u.

2. The TT Formats for Vectors and Matrices. For completeness of this paper, we give a relatively detailed introduction on the TT formats and its variants. The interested reader is referred to [11] for further information. A vector $u \in \mathbb{R}^N$ with $N = n_1 n_2 \dots n_d$ can be reshaped as a tensor $\mathbf{u} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ whose entries $u_{i_1 i_2 \dots i_d}$ are aligned in reverse lexicographical order, $1 \le i_\mu \le n_\mu, \mu = 1, 2, \dots, d$. The tensor $\mathbf{u}$ can be written in the TT format if its entries can be expressed as

$$u_{i_1 i_2 \dots i_d} = U_1(i_1) U_2(i_2) \cdots U_d(i_d), \tag{2.1}$$

where $U_\mu(i_\mu) \in \mathbb{R}^{r_{\mu-1} \times r_\mu}, i_\mu = 1, 2, \dots, n_\mu$ and the dimensions r_μ, $\mu = 0, 1, \dots, d$, are fixed with $r_0 = r_d = 1$. The tuple $\{r_0, r_1, \dots, r_d\}$ is called the TT-rank. For each μ, the tensor $\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$ formed by stacking the matrices $U_\mu(i_\mu), i_\mu = 1, \dots, n_\mu$, is called the μth TT core of $\mathbf{u}$. Therefore, a vector u of size $\mathcal{O}(n^d)$ can be stored with $\mathcal{O}(dnr^2)$ elements, if the corresponding tensor $\mathbf{u}$ has a TT representation or it can be approximated by a TT format, with a maximal TT-rank r. A graphical representation of $\mathbf{u}$ is shown in Figure 2.1.

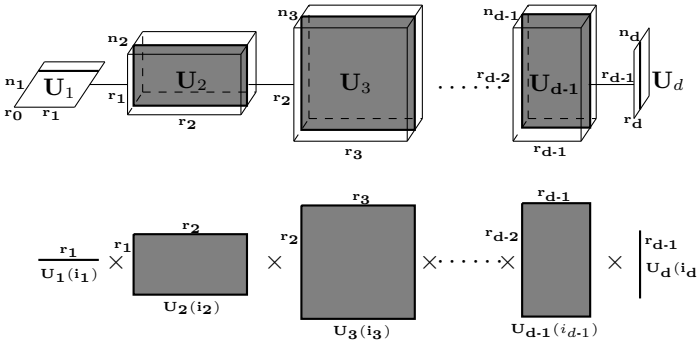

FIG. 2.1. *Graphical representation of a TT tensor of order d with cores $\mathbf{U}_\mu$, $\mu = 1, 2, \dots, d$. The first row is for $\mathbf{u}$ and the second row is for its entry $u_{i_1 i_2 \dots i_d}$.*

Similarly, a matrix $U \in \mathbb{R}^{N \times p}$ with $p \ll N$ can be expressed in the TT format. A simple way is to derive tensors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p$ in the TT format for the first to the last columns of U, respectively. A more compact form is to share all but one core of these p tensors. Let the TT cores of $\mathbf{u}_i$ be $\mathbf{U}_{1,i}, \mathbf{U}_{2,i}, \dots, \mathbf{U}_{d,i}$, for $i = 1, 2, \dots, p$. Then the p tensors form a block-μ TT (μ-BTT) format if

$$\mathbf{U}_{i,1} = \mathbf{U}_{i,2} = \dots = \mathbf{U}_{i,p} = \mathbf{U}_i, \quad i = 1, 2, \dots, \mu - 1, \mu + 1, \dots, d.$$

Then the $i_1 i_2 \cdots i_d$ entry of the jth tensor is

$$U(i_1, \ldots, i_\mu, \ldots, i_d; j) = U_1(i_1) \cdots U_\mu^j(i_\mu) \cdots U_d(i_d). \tag{2.2}$$

A graphical representation of the BTT format of $\mathbf{U}$ is depicted in Figure 2.2.

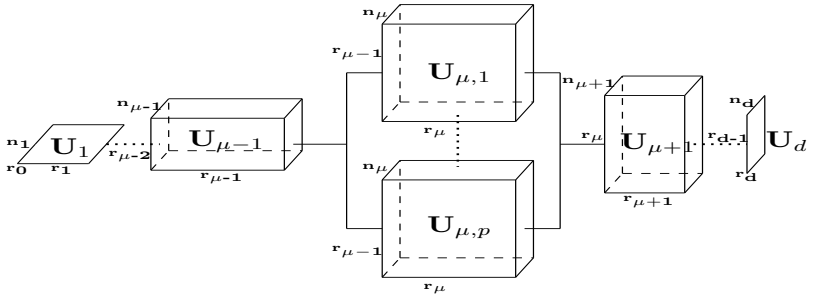

FIG. 2.2. *Graphical representation of a μ-BTT format.*

Consider a matrix $A \in \mathbb{R}^{m_1 m_2 \cdots m_d \times n_1 n_2 \cdots n_d}$ whose rows and columns are both indexed by multi-indices in reverse lexicographical order. If the entries of A can be written as

$$A_{i_1 i_2 \cdots i_d, j_1 j_2 \cdots j_d} = A_1(i_1, j_1) A_2(i_2, j_2) \cdots A_d(i_d, j_d), \tag{2.3}$$

where $A_\mu(i_\mu, j_\mu) \in \mathbb{R}^{r_{\mu-1} \times r_\mu}, i_\mu = 1, \ldots, m_\mu, j_\mu = 1, \ldots, n_\mu$, then the representation is called the operator TT format $\mathbf{A}$. Similarly, for each μ, the tensor $\mathbf{A}_\mu \in \mathbb{R}^{r_{\mu-1} \times m_\mu \times n_\mu \times r_\mu}$ constructed by stacking all matrices $A_\mu(i_\mu, j_\mu)$ is called the μth core of $\mathbf{A}$. The tuple $\{r_0, r_1, \ldots, r_d\}$ with $r_0 = r_d = 1$ is called the TT-rank of $\mathbf{A}$.

3. Properties of the TT formats.

3.1. Unfoldings of the TT formats. For a TT core $\mathbf{U}_\mu$, the left and right unfoldings are given by

$$U_\mu^L = \begin{pmatrix} U_\mu(1) \\ \vdots \\ U_\mu(n_\mu) \end{pmatrix} \in \mathbb{R}^{r_{\mu-1} n_\mu \times r_\mu}, U_\mu^R = \begin{pmatrix} V_\mu(1), & \cdots, & V_\mu(r_\mu) \end{pmatrix} \in \mathbb{R}^{r_{\mu-1} \times n_\mu r_\mu},$$

where $V_\mu(k) = \begin{pmatrix} U_\mu^{(k)}(1), & \cdots, & U_\mu^{(k)}(n_\mu) \end{pmatrix}$ and $U_\mu^{(k)}(i)$ is the kth column of matrix $U_\mu(i)$. Another pair of frequently used matrices are the interface matrices defined as

$$\begin{aligned} U_{\le \mu} &= [U_1(i_1) U_2(i_2) \cdots U_\mu(i_\mu)] \in \mathbb{R}^{n_1 n_2 \cdots n_\mu \times r_\mu}, \\ U_{\ge \mu} &= [U_\mu(i_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d)]^T \in \mathbb{R}^{n_\mu n_{\mu+1} \cdots n_d \times r_{\mu-1}}, \end{aligned}$$

where the rows and columns are aligned in reverse lexicographical order. It can be verified that

$$U_{\le \mu} = (I_{n_\mu} \otimes U_{\le \mu-1}) U_\mu^L \text{ and } U_{\ge \mu}^T = U_\mu^R (U_{\ge \mu+1} \otimes I_{n_\mu}). \tag{3.1}$$

Alternatively, the TT format can be expressed as a matrix-vector product. The properties of the Kronecker products and (3.1) lead to

$$u = \mathrm{vec}(\mathbf{U}_{\le \mu-1} \mathbf{U}_{\ge \mu}^T) = \mathrm{vec}(\mathbf{U}_{\le \mu-1} \mathbf{U}_\mu^R (\mathbf{U}_{\ge \mu+1}^T \otimes I_{n_\mu})) = \mathcal{U}_{\neq \mu} \mathrm{vec}(\mathbf{U}_\mu^R), \tag{3.2}$$

where

$$\mathcal{U}_{\neq\mu} := (\mathbf{U}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{U}_{\leq\mu-1}). \tag{3.3}$$

By substituting (3.1) into (3.2), we can extract two cores and build

$$u = \mathcal{U}_{\neq\mu,\mu+1}\mathrm{vec}(\mathbf{U}_\mu^L\mathbf{U}_{\mu+1}^R), \tag{3.4}$$

where

$$\mathcal{U}_{\neq\mu,\mu+1} = \mathbf{U}_{\geq\mu+2} \otimes I_{n_{\mu+1}} \otimes I_{n_\mu} \otimes \mathbf{U}_{\leq\mu-1}.$$

For a matrix $U \in \mathbb{R}^{n_1n_2\ldots n_d\times p}$, the unfolding properties (3.2) and (3.4) can be extended to its μ-BTT tensor $\mathbf{U}$. Applying them to each column of U gives

$$U = \mathcal{U}_{\neq\mu}\begin{pmatrix}\mathrm{vec}(\mathbf{U}_{\mu,1}^R) & \cdots & \mathrm{vec}(\mathbf{U}_{\mu,p}^R)\end{pmatrix} \tag{3.5}$$

$$= \mathcal{U}_{\neq\mu,\mu+1}\begin{pmatrix}\mathrm{vec}(\mathbf{U}_{\mu,1}^L\mathbf{U}_{\mu+1}^R) & \cdots & \mathrm{vec}(\mathbf{U}_{\mu,p}^L\mathbf{U}_{\mu+1}^R)\end{pmatrix}. \tag{3.6}$$

3.2. Operations Related to the TT and BTT Formats. For simplicity, we will call the vectorization of a TT tensor as a TT tensor if no confusion can arise. Almost all results of the basic numerical linear algebraic operations, including vector additions, matrix by vector products, the multiplication between a matrix in the BTT format and an ordinary matrix, are still in the TT formats. We briefly describe a few of these operations as follows.

Consider two vectors $u, v \in \mathbb{R}^N$ and their corresponding TT formats $\mathbf{u}, \mathbf{v}$ with TT cores $\mathbf{U}_\mu$ and $\mathbf{V}_\mu$, for $\mu = 1, \ldots, d$. Let $\mathbf{w} = \mathbf{u} + \mathbf{v}$ denote the addition of $u + v$. Then $\mathbf{w}$ is again a TT tensor with TT cores $\mathbf{W}_\mu$ given by

$$W_1(i_1) = \begin{pmatrix}U_1(i_1) & V_1(i_1)\end{pmatrix}, \quad W_d(i_d) = \begin{pmatrix}U_d(i_d) & V_d(i_d)\end{pmatrix}^\top,$$
$$W_k(i_k) = \begin{pmatrix}U_k(i_k) & \\ & V_k(i_k)\end{pmatrix}, \quad \forall 2 \leq k \leq d-1.$$

It is easy to check that $w_{i_1i_2\cdots i_d} = W_1(i_1)W_2(i_2)\cdots W_d(i_d)$. The TT-ranks of $\mathbf{w}$ is the addition of the TT-ranks of $\mathbf{u}$ and $\mathbf{v}$. The addition between two μ-BTT formats can be defined in the same fashion and the details are omitted here.

The Hadamard product $\mathbf{w} = \mathbf{u} \odot \mathbf{v}$ between $\mathbf{u}$ and $\mathbf{v}$ is also a tensor in the TT format with cores given by $W_k(i_k) = (U_k(i_k) \otimes (V_k(i_k))$ since it can be verified that

$$\begin{aligned}W_{i_1i_2\cdots i_d} &= (U_1(i_1)U_2(i_2)\cdots U_d(i_d)) \otimes (V_1(i_1)V_2(i_2)\cdots V_d(i_d)) \\ &= (U_1(i_1) \otimes V_1(i_1))(U_2(i_2) \otimes V_2(i_2))\cdots(U_d(i_d) \otimes V_d(i_d)).\end{aligned}$$

Consider two matrices $U \in \mathbb{R}^{N\times p}$ and $M \in \mathbb{R}^{p\times k}$. Let $\mathbf{U}$ be the BTT format of U. Then $\mathbf{U}M$ denotes the matrix multiplication UM, which can still be written in the BTT format. It follows from the equation (3.5) that

$$UM = \mathcal{U}_{\neq\mu}\begin{pmatrix}\mathrm{vec}(\mathbf{U}_{\mu,1}^R)\cdots\mathrm{vec}(\mathbf{U}_{\mu,p}^R)\end{pmatrix} M.$$

Hence, the $1, \ldots, \mu-1, \mu+1, \ldots, d$th cores of UM are the same as $\mathbf{U}$ while the μth cores are the reshapes of $\begin{pmatrix}\mathrm{vec}(\mathbf{U}_{\mu,1}^R)\cdots\mathrm{vec}(\mathbf{U}_{\mu,p}^R)\end{pmatrix} M$.

When U_μ^L or $(U_\mu^R)^T$ is orthonormal, the TT core $\mathbf{U}_\mu$ is called left or right orthonormal, respectively. If U_μ^L is not orthonormal, one can compute the QR decomposition of U_μ^L as $U_\mu^L = QR$ and substitute

$$U_\mu^L \leftarrow Q, U_{\mu+1}^R \leftarrow RU_{\mu+1}^R. \tag{3.7}$$

The unfolding property (3.4) implies that u is not changed under this new representation. Similar substitutions apply to U_μ^R as well. Hence, $U_{\le\mu-1}$ and $U_{\ge\mu+1}$ can be made to have orthonormal columns and rows, respectively, which implies that $U_{\neq\mu}$ and $U_{\neq\mu,\mu+1}$ are orthogonal matrices. Consequently, a μ-BTT format can also be orthogonalized due to (3.5).

A μ-BTT format can be transformed to a $(\mu+1)$-BTT format. Perform a minimal-rank decomposition

$$[U_{\mu,1}^L,\ldots,U_{\mu,p}^L] = Q[P_1,\ldots,P_p], \quad Q\in\mathbb{R}^{r_{\mu-1}n_\mu\times s}, P_i\in\mathbb{R}^{s\times r_\mu},$$

where the minimal-rank decomposition can be obtained by, e.g., a QR decomposition with column pivoting or an SVD. Then the cores are updated as

$$U_\mu^L \leftarrow Q \text{ and } U_{\mu+1,i}^R \leftarrow P_i U_{\mu+1}^R, \tag{3.8}$$

and the rank is changed as $r_\mu = s$.

3.3. Operations Related to the Operator TT Formats. A summation of the Kronecker products can be expressed in the operator TT format in (2.3). Suppose that

$$A = \sum_{k=1}^{R} L_d^{(k)} \otimes \ldots \otimes L_2^{(k)} \otimes L_1^{(k)}, \tag{3.9}$$

where $L_i^{(k)}$, $k=1,\ldots,R$ and $i=1,\ldots,d$, are matrices of suitable sizes. Then it has an operator TT format with rank at most $r_1=\ldots=r_{d-1}=R$ and the corresponding matrices in (2.3) are

$$A_1(i_1,j_1) = [L_1^{(1)}(i_1,j_1),\ldots,L_1^{(R)}(i_1,j_1)], \quad A_d(i_d,j_d) = \begin{pmatrix} L_d^{(1)}(i_d,j_d) \\ \vdots \\ L_d^{(R)}(i_d,j_d)\end{pmatrix},$$

$$A_\mu(i_\mu,j_\mu) = \begin{pmatrix} L_\mu^{(1)}(i_\mu,j_\mu) & & & \\ & L_\mu^{(2)}(i_\mu,j_\mu) & & \\ & & \ddots & \\ & & & L_\mu^{(R)}(i_\mu,j_\mu)\end{pmatrix}, \mu=2,\ldots,d-1.$$

A special form of (3.9) is

$$A = \sum_{\mu=1}^{d} M_d \otimes\cdots\otimes M_{\mu-1}\otimes L_\mu \otimes M_{\mu+1}\otimes\cdots\otimes M_1. \tag{3.10}$$

It has a rank-2 operator TT representation as

$$A_1(i_1,j_1) = (L_1(i_1,j_1),\ M_1(i_1,j_1)), \quad A_d(i_d,j_d) = (M_d(i_d,j_d),\ L_d(i_d,j_d))^T,$$
$$A_\mu(i_\mu,j_\mu) = \begin{pmatrix} M_\mu(i_\mu,j_\mu) & 0 \\ L_\mu(i_\mu,j_\mu) & M_\mu(i_\mu,j_\mu)\end{pmatrix}, \mu=2,\ldots,d-1.$$

Another specific matrix is

$$A = \sum_{\mu=1}^{d-1} M_d\otimes\cdots\otimes M_{\mu+2}\otimes C_{\mu+1}\otimes B_\mu\otimes M_{\mu-1}\otimes\cdots\otimes M_1, \tag{3.11}$$

whose operator TT format is

$$A_1(i_1,j_1) = \begin{pmatrix} 0, & B_1(i_1,j_1), & M_1(i_1,j_1) \end{pmatrix}, \quad A_d(i_d,j_d) = \begin{pmatrix} M_d(i_d,j_d) \\ C_d(i_d,j_d) \\ 0 \end{pmatrix},$$

$$A_\mu(i_\mu,j_\mu) = \begin{pmatrix} M_\mu(i_\mu,j_\mu) & 0 & 0 \\ C_\mu(i_\mu,j_\mu) & 0 & 0 \\ 0 & B_\mu(i_\mu,j_\mu) & M_\mu(i_\mu,j_\mu) \end{pmatrix}, \quad \mu = 2,\ldots,d-1.$$

The multiplication between a matrix in the operator TT format and a vector in the TT format also gives a TT format. Consider a matrix $\mathbf{A}$ and a vector $\mathbf{u}$ defined by (2.1) and (2.3), respectively. Let $\mathbf{v} = \mathbf{A}\mathbf{u}$. Then we have

$$\begin{aligned} v_{i_1 i_2 \cdots i_d} &= \sum_{j_1 j_2 \cdots j_d} (A_1(i_1,j_1)A_2(i_2,j_2)\cdots A_d(i_d,j_d)) \otimes (U_1(j_1)U_2(j_2)\cdots U_d(j_d)) \\ &= \sum_{j_1 j_2 \cdots j_d} (A_1(i_1,j_1)\otimes U_1(j_1))(A_2(i_2,j_2)\otimes U_2(j_2))\cdots(A_d(i_d,j_d)\otimes U_d(j_d)) \\ &= \sum_{j_1}(A_1(i_1,j_1)\otimes U_1(j_1)) \sum_{j_2}(A_2(i_2,j_2)\otimes U_2(j_2)) \ldots \sum_{j_d}(A_d(i_d,j_d)\otimes U_d(j_d)). \end{aligned}$$

Hence, the TT cores of $\mathbf{v}$ are $V_k(i_k) = \sum_{j_k} A_k(i_k,j_k)\otimes U_k(j_k), k \in \{1,2,\ldots,d\}$. The TT ranks of $\mathbf{v}$ are bounded by the multiplications of the TT ranks of $\mathbf{A}$ and $\mathbf{u}$.

3.4. Truncation of the BTT Formats. We now describe the truncation of the TT and BTT formats by using a sequence of singular value decompositions (SVDs). Assume that the SVD of $\mathbf{U}_\mu^L$ is $\mathbf{U}_\mu^L = USV^\top$. The unfolding property (3.4) leads to

$$\begin{aligned} u &= \mathcal{U}_{\neq\mu,\mu+1}\mathrm{vec}(\mathbf{U}_\mu^L\mathbf{U}_{\mu+1}^R) = \mathcal{U}_{\neq\mu,\mu+1}\mathrm{vec}(USV^\top\mathbf{U}_{\mu+1}^R) \\ &= \mathcal{U}_{\neq\mu,\mu+1}\mathrm{vec}(\widetilde{\mathbf{U}}_\mu^L\widetilde{\mathbf{U}}_{\mu+1}^R), \end{aligned} \tag{3.12}$$

where $\widetilde{\mathbf{U}}_\mu^L = U$ and $\widetilde{\mathbf{U}}_{\mu+1}^R = SV^\top\mathbf{U}_{\mu+1}^R$. Given a TT tensor $\mathbf{u} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ with TT-rank r and a maximal TT-rank R, we set $R_\mu = \min(R_\mu, r_\mu)$, $\mu = 0,1,\ldots,d$. Therefore, $\mathbf{u}$ can be truncated by applying the step (3.12) such that the rank of the μth core is smaller than R_μ. A description of this truncation procedure based on SVD is presented in Algorithm 1.

Algorithm 1: SVD Truncation of a vector in the TT Format

1. Given a TT tensor $\mathbf{u} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ with TT-rank r and a maximal TT-rank R.
2. Compute $R \leftarrow \min(R, r)$, where the minimum is taken componentwisely.
3. **for** $i = 1 : d-1$ **do**
4. Compute the SVD of $\mathbf{U}_i^L$ as $\mathbf{U}_i^L = USV^\top$.
5. Set $U \leftarrow U(:, 1:R_{i+1})$, $S \leftarrow S(1:R_{i+1}, 1:R_{i+1})$ and $V \leftarrow V(:, 1:R_{i+1})$.
6. Set $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$ by properly reshape U and $SV^\top\mathbf{U}_{i+1}^R$, respectively.

We should point out that Algorithm 1 can not guarantee a small truncation error. It should only be used when a low-rank approximation exists. There is a version in the reference [19] with guarantees on the accuracy, but the low-rank properties may not be preserved.

The truncation of a collection of p vectors in the BTT format can be performed in the same fashion as Algorithm 1. One only needs to truncate these tensors in one block and break

it down into the BTT format again. Suppose that we have a μ-BTT format $\mathbf{U}$ with cores $\mathbf{U}_1, \ldots, \mathbf{U}_{\mu-1}, \mathbf{U}_{\mu+1}, \ldots, \mathbf{U}_d$, and $\mathbf{U}_\mu^j$, for $j = 1, 2, \ldots, p$. By treating the block $\mathbf{U}$ as one tensor in $\mathbb{R}^{n_1 \times \cdots \times n_{\mu-1} \times (n_\mu p) \times n_{\mu+1} \times \cdots \times n_d}$ and (i_μ, j) as a long index in the lexicographical order, then (2.2) can be rewritten in the form as

$$U(i_1, \ldots, \widetilde{i}_\mu, \ldots., i_d) = U_1(i_1) \cdots U_{\mu-1}(i_{\mu-1}) \widetilde{U}_\mu(\widetilde{i}_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d),$$

where $\widetilde{i}_\mu = (i_\mu, j)$ and $\widetilde{U}_\mu$ is a properly reshaped combination of the p cores. Hence, the truncation procedure for a single tensor can be applied, then we break the truncated core $\widetilde{U}_\mu$ into p cores and reconstruct the BTT format into $\mathbf{U}$.

4. Truncated Subspace Optimization Methods. Let $U \in \mathbb{R}^{N \times p}$ and $\mathbf{T}_{\mathbf{n},r,p}$ be the set of the BTT formats for a collection of p tensors such that the TT-ranks are no more than r and the size is $\mathbf{n} = (n_1, \ldots, n_d)$. Then the eigenvalue problem (1.2) is specified as

$$\min_{U \in \mathbb{R}^{N \times p}} \mathbf{Tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U}), \quad \text{s.t.} \quad \mathbf{U}^\top \mathbf{U} = I_p \text{ and } \mathbf{U} \in \mathbf{T}_{\mathbf{n},r,p}. \tag{4.1}$$

When the TT-rank of $\mathbf{A}$ is relatively low, problem (4.1) becomes tractable in the BTT format. We next briefly review the subspace methods for solving the standard eigenvalue problem (1.1), including the limited memory block Krylov subspace optimization method (LMSVD) developed in [16] and the LOBPCG method [9]. Then we extend these algorithms to solve the eigenvalue problem in the BTT formats.

4.1. Subspace Optimization Methods for Standard Eigenvalue Problems. Let $(U^{(k)}, \Lambda^{(k)})$ be a Ritz-pair, i.e., the pair of approximated eigenvectors and their corresponding eigenvalues, and $\mathcal{S}^{(k)}$ be a chosen subspace with a block Krylov subspace structure at the kth iteration. One solves a subspace trace minimization problem

$$U^{(k+1)} := \underset{U \in \mathbb{R}^{N \times p}}{\arg\min} \ \mathbf{Tr}(U^\top A U), \ \text{s.t.} \ U^\top U = I_p, \ U \in \mathcal{S}^{(k)}, \tag{4.2}$$

where $U \in \mathcal{S}^{(k)}$ means that all columns of U are from the subspace $\mathcal{S}^{(k)}$. The LOBPCG method constructs $\mathcal{S}^{(k)}$ as the span of the current iterate $U^{(i)}$, the conjugate gradient direction $P^{(k)} = U^{(k)} - U^{(k-1)}$ and the residual vectors $R^{(k)} = AU^{(k)} - U^{(k)}\Lambda^{(k)}$ as

$$\mathcal{S}^{(k)} = \text{span}\{U^{(k)}, R^{(k)}, P^{(k)}\}. \tag{4.3}$$

The term $R^{(k)}$ may be pre-multiplied by a pre-conditioning matrix. The subspace (4.3) is essentially equivalent to

$$\mathcal{S}^{(k)} \in \text{span}\left\{U^{(k-1)}, U^{(k)}, AU^{(k)}\right\}, \tag{4.4}$$

In the LMSVD method, the subspace $\mathcal{S}^{(k)}$ is spanned by the current i-th iterate and the previous p iterates; i.e.,

$$\mathcal{S}^{(k)} := \text{span}\left\{U^{(k)}, U^{(k-1)}, ..., U^{(k-q)}\right\}, \tag{4.5}$$

where q is a suitable integer. In general, the subspace $\mathcal{S}^{(k)}$ should be constructed such that the cost of solving (4.2) can be kept relatively low.

Suppose that S is the collection of all blocks in $\mathcal{S}^{(k)}$, for example, $S = [U^{(k)}, R^{(k)}, P^{(k)}]$ in (4.3). It is clear that $U \in \mathcal{S}^{(k)}$ if and only if $U = SV$ for some $V \in \mathbb{R}^{q \times p}$ with $q =$

$3p$. Then the subspace optimization problem (4.2) is equivalent to a generalized eigenvalue decomposition problem:

$$\min_{V\in\mathbb{R}^{q\times p}} \mathbf{Tr}(V^\top (S^\top AS)V),\ \text{s.t. } V^\top S^\top SV = I_p. \tag{4.6}$$

Numerical difficulty may arise in solving (4.6) when the matrix $S^\top S$ is numerically rank deficient. A more stable approach is to find an orthonormal basis for $\mathcal{S}^{(k)}$, say

$$Q \in \mathbf{orth}\left(\mathcal{S}^{(k)}\right),$$

and to express a matrix $U \in \mathcal{S}^{(k)}$ as $U = QV$ for some $V \in \mathbb{R}^{q\times p}$. Here we assume that $\mathcal{S}^{(k)}$ has a full rank. The rank deficient case can be handled similarly. We now convert the generalized eigenvalue problem (4.6) into an equivalent eigenvalue problem

$$\min_{V\in\mathbb{R}^{q\times p}} \mathbf{Tr}(V^\top (Q^\top AQ)V),\ \text{s.t. } V^\top V = I_p. \tag{4.7}$$

A description of the subspace optimization algorithm is summarized in Algorithm 2.

Algorithm 2: A Subspace Optimization Framework

1 Given an initial guess $U^{(0)} \in \mathbb{R}^{N\times p}$ and a matrix $A \in \mathbb{R}^{N\times N}$. Set $k = 0$.
2 **while** *not converge* **do**
3 Construct a suitable subspace $\mathcal{S}^{(k)}$.
4 Find an orthonormal basis Q of $\mathcal{S}^{(k)}$.
5 Compute the eigenvalue decomposition $V\Lambda V^T = Q^T AQ$.
6 Assemble $U^{(k+1)} = QV$ and set $\Lambda^{(k+1)} = \Lambda$.

4.2. Truncated Subspace Optimization Methods Under the BTT Formats. Unfortunately, Algorithm 2 cannot be used directly to solve the eigenvalue problems in the BTT formats. The main challenge is that the TT-ranks increase dramatically after several operations between tensor formats, such as the addition in the TT formats and matrix-vector multiplications in the TT format. Consequently, the computational cost of all subsequent operations becomes more and more expensive as the TT-ranks increase. Hence, projections to $\mathbf{T}_{\mathbf{n},r,p}$ at some suitable places are necessary so that the overall computational cost is still tractable.

Each iteration (4.2) of our truncated subspace optimization methods is split into two steps. First, the subspace $\mathcal{S}^{(k)}$ is modified so that the computation of the coefficient matrix $Q^T AQ$ in the RR procedure is affordable. Let $\mathcal{P}_{\mathbf{T}}(\mathbf{U})$ be the truncation of $\mathbf{U}$ to the BTT format $\mathbf{T}_{\mathbf{n},r,p}$. One choice of the subspace is

$$\mathcal{S}^{(k)}_{\mathbf{T}} = \mathrm{span}\{\mathcal{P}_{\mathbf{T}}(\mathbf{A}\mathbf{U}^{(k)}), \mathbf{U}^{(k)}, \mathbf{U}^{(k-1)}\}. \tag{4.8}$$

The subspace (4.4) can also be used but it requires two truncations as

$$\mathcal{S}^{(k)}_{\mathbf{T}} = \mathrm{span}\{\mathbf{U}^{(k)}, \mathcal{P}_{\mathbf{T}}(\mathbf{R}^{(k)}), \mathcal{P}_{\mathbf{T}}(\mathbf{P}^{(k)})\}. \tag{4.9}$$

It is important to to use the projection of $\mathbf{R}^{(k)}$ directly rather than $\mathcal{P}_{\mathbf{T}}(\mathcal{P}_{\mathbf{T}}(\mathbf{A}\mathbf{U}^{(k)}) - \mathbf{U}^{(k)}\Lambda^{(k)})$, although the latter may be cheaper. The reason is that both $\mathbf{A}\mathbf{U}^{(k)}$ and $\mathbf{U}^{(k)}\Lambda^{(k)}$ can be large while its subtraction is small.

Then we compute $\mathbf{Y}^{(k+1)}$ in the BTT format as

$$\mathbf{Y}^{(k+1)} := \underset{U\in\mathbb{R}^{N\times p}}{\arg\min}\ \mathbf{Tr}(\mathbf{U}^\top\mathbf{A}\mathbf{U}),\ \text{s.t. } \mathbf{U}^\top\mathbf{U} = I_p,\ \mathbf{U}\in\mathcal{S}_{\mathbf{T}}^{(k)}. \tag{4.10}$$

In general, we have that $\mathbf{Y}^{(k+1)} \notin \mathbf{T}_{\mathbf{n},r,p}$ since the rank of $\mathbf{Y}^{(k+1)}$ is larger than r due to the addition of several BTT formats. However, because $\mathbf{Y}^{(k+1)}$ is a linear combination of the BTT formats $\mathcal{P}_{\mathbf{T}}(\mathbf{A}\mathbf{U}^{(k)}), \mathbf{U}^{(k)}$ and $\mathbf{U}^{(k-1)}$, problem (4.10) can be solved and $\mathbf{Y}^{(k+1)}$ is stored implicitly.

The next step is to truncate $\mathbf{Y}^{(k+1)}$ to the required space $\mathbf{T}_{\mathbf{n},r,p}$ by using an orthogonal projection as

$$\mathbf{U}^{(k+1)} = \arg\min_{\mathbf{U}\in\mathbb{R}^{N\times p}} \|\mathbf{U}-\mathbf{Y}^{(k+1)}\|_F^2, \quad \text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = I_p, \quad \mathbf{U}\in\mathbf{T}_{\mathbf{n},r,p}. \tag{4.11}$$

The orthogonal projection (4.11) again can be solved by using the alternating minimization scheme in the same fashion as the projection (4.16). The only difference between them is the orthogonality constraints in (4.11). For the μth core, the new subproblem is

$$\min_V \quad \|\mathcal{U}_{\neq\mu}V - \mathrm{vec}(\mathbf{Y}^{(k+1)})\|_F^2,\ \text{s.t. } V^\top\mathcal{U}_{\neq\mu}^\top\mathcal{U}_{\neq\mu}V = I_p. \tag{4.12}$$

By imposing orthogonality on $\mathcal{U}_{\neq\mu}$, (4.12) is reduced to

$$\min_V \quad \|V - \mathcal{U}_{\neq\mu}^\top\mathrm{vec}(\mathbf{Y}^{(k+1)})\|_F^2,\ \text{s.t. } V^\top V = I_p, \tag{4.13}$$

whose optimal solution is determined by the p-dominant SVD of $\mathcal{U}_{\neq\mu}^\top\mathrm{vec}(\mathbf{Y}^{(k+1)})$.

An outline of the above method is presented in Algorithm 3.

Algorithm 3: Truncated Subspace Optimization Methods (SPB)

1 Input $\mathbf{A}$, initialize $\mathbf{U}^{(0)}\in\mathbf{T}_r$ and set $k=0$.
2 **while** *"not converged"* **do**
3 Construct a suitable subspace $\mathcal{S}_{\mathbf{T}}^{(k)}$.
4 Compute $\mathbf{Y}^{(k+1)}$ implicitly from the subspace problem (4.10).
5 Compute $\mathbf{U}^{(k+1)}$ from (4.11).
6 Increment k and continue.

Similar to EVAMEn [11], our termination criteria is based on $||\mathcal{P}_{\mathbf{T}}(\mathbf{R}^{(k)})||_F \le \mathrm{tol}$, where tol is a given tolerance. The projection of the residual to $\mathbf{T}_{\mathbf{n},r,p}$ is used because of the requirement $\mathbf{U}\in\mathbf{T}_{\mathbf{n},r,p}$. Note that Algorithm 3 is faster per iteration when the TT rank r is small. Hence, we also adopt a continuation strategy on the TT rank r in Algorithm 3 by starting from a small TT rank in our implementation. Then the TT rank is increased by a given number when Algorithm 3 stagnates at the current rank. The continuation process is terminated when Algorithm 3 stops with the given maximal TT rank.

4.3. Subspace Optimization with Local Refinements. Inspired by the block algorithms with augmented Rayleigh-Ritz projections for large-scale eigenpair computation proposed in [20], we propose to improve the convergence of Algorithm 3 by an iterative two-step framework. At each iteration, the first step computes a better point U so that its column space is a good approximation to the p-dimensional eigenspace spanned by p desired eigenvectors. Once U is obtained, the subspace optimization step (4.10)-(4.11) extracts from a set of approximate eigenpairs that are optimal in certain sense.

We next present the basic concept of the alternating minimization method. Given a block-μ BTT $\mathbf{U}$ and a TT operator $\mathbf{A}$, it follows from (3.5) that

$$U^T A U = V^T \mathcal{A}_{\neq\mu} V, \tag{4.14}$$

where $\mathcal{A}_{\neq\mu} = \mathcal{U}_{\neq\mu}^T A \mathcal{U}_{\neq\mu}$ and $V = \big(\mathrm{vec}(\mathbf{U}_{\mu,1}^R), ..., \mathrm{vec}(\mathbf{U}_{\mu,p}^R)\big)$. Since the columns of $\mathcal{U}_{\neq\mu}$ can be easily orthogonalized, the matrix U is orthonormal if and only if V is orthonormal. For the μth cores of $\mathbf{U}$, the alternating minimization scheme first calculates

$$\min_{V^T V = I} \mathbf{Tr}(V^T \mathcal{A}_{\neq\mu} V). \tag{4.15}$$

Then we replace the μth cores of $\mathbf{U}$ by the optimal solution V^* of (4.15) and shift $\mathbf{U}$ to be a block-$(\mu+1)$ BTT block by using (3.8). A similar procedure leads to a block-$(\mu-1)$ BTT block. Finally, we repeat this scheme core by core until the algorithm converges. A single left to right operation of the ALS method is described in Algorithm 4 and its counterpart for the right to left operation can be performed similarly. Several approaches can be used to accelerate this basic version of alternating minimization. One can construct a good preconditioner for solving the subproblem (4.15) and add subspace enrichment by putting the gradient information into the subspace $\mathrm{ran}(\mathcal{U}_{\neq\mu})$. The detailed construction of the preconditioners and local subspace correction are referred to [11].

Algorithm 4: A single left to right operation in ALS

1 Given $\mathbf{A}$ and a block-μ BTT block $\mathbf{U} \in \mathbf{T}_r$.
2 Orthonormalize $\mathcal{U}_{\neq\mu}$ by (3.7) or its counterpart for $\mathbf{U}_i^R, i \neq \mu$, if $\mathcal{U}_{\neq\mu}^T \mathcal{U}_{\neq\mu} \neq I$.
3 Replace $\mathbf{U}_{\mu,\alpha}$ by the optimal solution V of subproblem (4.15).
4 Shift $\mathbf{U}$ to be a block-$(\mu+1)$ BTT block using (3.8).

A complete subspace optimization with local refinements is presented in Algorithm 5. The main difference to Algorithm 3 is the construction of the subspace $\mathcal{S}_{\mathbf{T}}^{(k)}$, which is generated from a point $\widetilde{\mathbf{U}}^{(k)}$ by performing t steps of the left to right or the right to left ALS operations by using Algorithm 4. The direction of applying these operations are reversed when the index μ reaches the right or left end of the tensor. For convenience, the number t can be a half of the dimension of $\mathbf{U}$ but other values can be used as well. Our numerical results show that the combination of the truncated subspace optimization and ALS often yields much better solutions than performing only one of them.

4.4. Truncation of the BTT Formats Using the ALS Method. The truncation procedure Algorithm 1 is based on a sequence of SVDs whose the numerical cost increase cubicly with respect to the increase of the TT-ranks. In our framework, we need truncations of the BTT block $\mathbf{AU}$, where $\mathbf{A}$ is a TT matrix and $\mathbf{U}$ is a BTT block. For example, the TT-rank of the Newton potential in our numerical experiment is $(1, 12, \ldots, 12, 1)$ and the numerical cost roughly increases 12^3 times for $\mathbf{AU}$. Therefore, more efficient approaches for truncation are needed.

Another type of methods is the ALS-type truncation. For a given tensor $\mathbf{y}$, the projection of $\mathbf{y}$ to the space $\mathbf{T}_{\mathbf{n},r,p}$ is defined as

$$\mathcal{P}_{\mathbf{T}}(\mathbf{y}) = \arg\min_u \ \|\mathbf{u} - \mathbf{y}\|_2^2, \ \text{s.t. } \mathbf{u} \in \mathbf{T}_{\mathbf{n},r,p}. \tag{4.16}$$

From the property (3.2), the matrix $\mathcal{U}_{\neq\mu}$ is orthonormal if all cores to the left of the μth core are left-orthonormal and all cores to the right of core μ are right-orthonormal. Therefore,

Algorithm 5: Truncated subspace optimization with ALS refinement (ALSPB)

Input $\mathbf{A}$, initialize $\mathbf{U}^{(0)} \in \mathbf{T}_r$. Set $k = 0$, $t > 1$ and the position $\mu = 1$.
while *"not converged"* **do**
  Perform t steps of the left to right or the right to left ALS operations by using Algorithm 4, where the direction of applying these operations are reversed when μ reaches the right or left end of the tensor. The outputed solution is denoted by $\widetilde{\mathbf{U}}^{(k)}$. `/* Local refinement by ALS */`
  Construct a suitable subspace $\mathcal{S}_{\mathbf{T}}^{(k)}$ based on $\widetilde{\mathbf{U}}^{(k)}$.
  Compute $\mathbf{Y}^{(k+1)}$ implicitly from the subspace problem (4.10).
  Compute $\mathbf{U}^{(k+1)}$ from (4.11).
  Increment k and continue.

an alternating minimization can be derived in a simple form for solving (4.16). Similar to the above procedure, we treat the block $\mathbf{y}$ as one tensor and reshape it back when we finish truncation. Hence, we only consider the case where $p = 1$ here. Starting from an initial approximation $\mathbf{u}$, we fix all cores of $\mathbf{u}$ except the μth core and solve the subproblem

$$\min_{V} \quad \|\mathcal{U}_{\neq\mu} V - \mathrm{vec}(\mathbf{y})\|_F^2, \tag{4.17}$$

where $\mathcal{U}_{\neq\mu}$ is defined in (3.2). The optimal solution of (4.17) is simply

$$V^* = \mathcal{U}_{\neq\mu}^{\dagger} \mathrm{vec}(\mathbf{y}),$$

where $\mathcal{U}_{\neq\mu}^{\dagger}$ represents the generalized inverse of $\mathcal{U}_{\neq\mu}$. By enforcing the orthogonality of the initial guess $\mathbf{u}$, the matrix $\mathcal{U}_{\neq\mu}$ is orthogonal and we have $\mathcal{U}_{\neq\mu}^{\dagger} = \mathcal{U}_{\neq\mu}^{T}$. Once the optimal V^* is calculated, we can reconstruct $\mathbf{U}_\mu$ by reshaping V^* to the size of the μth core of $\mathbf{u}$. Then we shift to the next core and update it similarly. In this way, we move from dth core to 1st core and switch back again. This procedure is repeated until it converges. To keep the orthonormality when shifting cores from μ to $\mu+1$, one can perform orthogonalization steps as (3.7). Therefore, a TT tensor $\mathbf{u}$ with $\mathcal{U}_{\neq\mu}$ orthonormal is converted to a TT tensor $\mathbf{u}$ with $\mathcal{U}_{\neq\mu+1}$ orthonormal while keeping the entries of $\mathbf{u}$ unchanged. The shift from core $\mu+1$ to μ is similar and we omit it here. The ALS-type truncation is presented in Algorithm 6.

Algorithm 6: ALS Truncation of the BTT format

Given a TT or d-BTT tensor $\mathbf{y}$ and a maximal TT-rank $\mathbf{R}$.
Convert $\mathbf{y}$ to a tensor in $\mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_d p}$ if $\mathbf{y}$ is a collection of p tenors.
Construct an initial point $\mathbf{u}$ with TT cores $\mathbf{U}_i, i = 1, \ldots, d$, with a TT-rank at most $\mathbf{R}$.
while *not converge* **do**
  for $\mu = d : 2$ **do**
    $\mathbf{U}_\mu = \mathrm{reshape}(\mathcal{U}_{\neq\mu}^{T} \mathrm{vec}(\mathbf{y}))$ and shift the core from μ to $\mu - 1$.
  for $\mu = 1 : d-1$ **do**
    $\mathbf{U}_\mu = \mathrm{reshape}(\mathcal{U}_{\neq\mu}^{T}(\mathrm{vec}(\mathbf{y}))$ and shift the core from μ to $\mu + 1$
Reshape the dth core of $\mathbf{u}$ back to original size to obtain a TT or d-BTT format.

The cost of one sweep of Algorithm 6 includes a few block inner products and $d-1$ QR decompositions for shifting cores of $\mathbf{u}$. This feature enables a faster algorithm than SVD-

type truncation. The derivation of DMRG-type truncation [17] is also possible. It is still in the alternating minimization framework but finds two neighbouring cores at each time. The ALS procedure has a smaller computational cost each iteration while it converges slower than the DMRG-type procedure.

The truncation error is a by-product for the SVD-type truncation. For the ALS truncation, the approximation error of $\mathbf{y}-\mathbf{u}$ can be estimated by applying itself to $\mathbf{y}-\mathbf{u}$ for a few sweeps. Numerical experiments show that one or two sweeps are often enough to estimate an error with the correct order of magnitude.

4.5. Separability of A. The structure of the operator $\mathbf{A}$ is critical for speeding up our algorithm in that many operations including truncation procedure can be performed much more efficiently when the operator $\mathbf{A}$ is separable.

Suppose that $\mathbf{U}$ has a TT-rank r and that $\mathbf{A} = \sum_{i=1}^{k} \mathbf{P}_i$, where each $\mathbf{P}_i$ has a TT-rank R. Then the computational cost of $\mathbf{AU}$ directly and the storage are both $\mathcal{O}(nd(kRr)^2)$. On the other hand, the computational cost of $\sum_{i=1}^{k} \mathbf{P}_i\mathbf{U}$ and its storage are reduced by k times, i.e., they are $\mathcal{O}(ndk(Rr)^2)$. Then in the subsequent ALS-type truncation procedures, the cost of the direct truncation of $\mathbf{AU}$ becomes $\mathcal{O}(nd(kRr)^3)$ flops while the cost of truncating $\sum_{i=1}^{k} \mathbf{P}_i\mathbf{U}$ is reduced to $\mathcal{O}(ndk(Rr)^3)$ by k^2 times.

Consider a tensor of the form $\mathbf{y} = \mathbf{y}_1 + \ldots + \mathbf{y}_k$, where each $\mathbf{y}_i$ has a low TT-rank. This form is ubiquitous in the multiplication of a matrix in the TT format of separable structure with a BTT block. For example, the residual $\mathbf{R} = \mathbf{AU} - \mathbf{U}\Lambda$ is of this form when $\mathbf{A}$ is separable. Suppose that all tensors $\mathbf{y}_i, i = 1, ..., k$, have the same TT-ranks. Then in the SVD Algorithm 1 the cost of truncation for $\mathbf{y}$ grows cubicly with respect to k. One possible strategy is to execute truncation after adding each $\mathbf{y}_i$. Although its numerical cost grows linearly with respect to k, it may lead to a large error when each $\mathbf{y}_i$ has a large norm while the summation $\mathbf{y}$ has a rather small norm (see the case $\mathbf{R}$). However, the numerical cost of the ALS Algorithm 6 grows linearly with respect to k when truncating the full vector $\mathbf{y}$ if we calculate $\mathcal{U}_{\neq\mu}^T\text{vec}(\mathbf{y})$ in algorithm 6 by summing up $\mathcal{U}_{\neq\mu}^T\text{vec}(\mathbf{y}_i), i = 1, \ldots, k$. In this case, the ALS truncation procedure can take advantage of the structure while maintaining a reasonably high accuracy.

In fact, for many functions such as the trigonometric functions and polynomials, their TT formats have an explicit low-rank representation. In addition, when approximating the potential function in $\mathbf{A}$ using rank-1 tensors with exponential sums or various kinds of polynomial interpolations, we also obtain this favorable structure. For more general functions on which we cannot analytically find a low-rank approximation, there are many ways to compute a low-rank TT or CP approximation, see [7, 6, 5] and the references therein, for a more detailed review of tensor approximation on multi-dimensional functions.

When a function itself is already a compact and higher-rank tensor, it is inconvenient to approximate the function using exponential sums or interpolation. Then we can numerically approximate it by a sum of low-rank tensors. In particular, the CP approximation can be applied. A rank-R CP tensor $\mathbf{t}$ is defined as $\mathbf{t} = \sum_{k=1}^{r} t_d^k \otimes \cdots \otimes t_1^k$, where $t_1^k \in \mathbb{R}^{n1}, ..., t_d^k \in \mathbb{R}^{n_d}, k = 1, ..., r$. This expression implies that a rank-r CP tensor is a summation of r rank-1 TT tensors and it is therefore a special case in the TT format. Finding a CP approximation for a given tensor $\mathbf{u}$ can be formulated as solving the optimization problem

$$\mathbf{y} = \underset{\mathbf{v}\in\text{CP}(r)}{\arg\min} ||\mathbf{v} - \mathbf{u}||_F^2,$$

where $\text{CP}(r)$ stands for the set of CP tensors with ranks no more than R. The above problem has been solved by using the ALS method, see section 3.4 in [10] and the references therein.

Since the convergence of ALS method is highly dependent on the initial guess, we propose a method to construct some initial points. Note that the tensor $\mathbf{u}$ is already in TT format. The SVD Algorithm 1 usually provides a robust and reliable way for low-rank approximations. Suppose that $\mathcal{P}_{\mathbf{T}}(\cdot)$ denotes the projection to rank-1 TT set $\mathbf{T}_{\mathbf{n},1,1}$ by the SVD approach. Then we calculate

$$\mathbf{P}_1 = \mathcal{P}_{\mathbf{T}}(\mathbf{u}),\ \mathbf{P}_k = \mathcal{P}_{\mathbf{T}}(\mathbf{u} - \sum_{i=1}^{k-1} \mathbf{P}_i),\ k = 2, ..., r.$$

One can prove that $\langle \mathbf{P}_i, \mathbf{P}_j \rangle = 0, i \neq j$, which implies that $||\sum_{k=1}^{r} \mathbf{P}_k||_F^2 = \sum_{k=1}^{r} ||\mathbf{P}_k||_F^2$ is monotone increasing in r. Combining another fact that $||\sum_{k=1}^{R} \mathbf{P}_k||_F^2 \leq ||\mathbf{u}||_F^2$, we know that this sequential projection method constantly improves approximation accuracy. Numerical experiment shows that it takes a very small r to reach a relative error of order 10^{-2}. Of course, the convergence may still slow down rapidly with the growth of r.

5. Numerical Results. In this section, we evaluate the numerical performance of the proposed method for finding the p smallest eigenvalues of the PDE eigenvalue problem:

$$\begin{aligned} -\Delta u(x) + V(x)u(x) &= \lambda u(x), \qquad \text{for } x \in \Omega = (a,b)^d, \\ u(x) &= 0, \qquad \text{for } x \in \partial\Omega, \end{aligned} \tag{5.1}$$

where Δ is the d-dimensional Laplace operator and the function $V(x)$ denotes potential energy. Problem (5.1) becomes the eigenvalue problem (4.1) after a suitable discretization of the system. The coefficient matrix has the form $\mathbf{A} = \mathbf{L} + \mathbf{V}$, where $\mathbf{L}$ is the discretization of the negative Laplace operator as

$$\mathbf{L} = \sum_{\mu=1}^{d} I_n \otimes \cdots \otimes I_n \otimes L \otimes I_n \otimes \cdots \otimes I_n, \tag{5.2}$$

which is naturally in the operator TT format given in section 3.3, and $\mathbf{V}$ is the discretized potential whose specific form will be specified later.

In order to construct a good preconditioner, we approximate $\mathbf{A}$ by the matrix of the form

$$\hat{\mathbf{L}} = \sum_{\mu=1}^{d} I_n \otimes \cdots \otimes I_n \otimes L_\mu \otimes I_n \otimes \cdots \otimes I_n. \tag{5.3}$$

It follows from [3] that, for symmetric definite matrices $L_\mu,\ \mu = 1, \ldots, d$, the inverse of $\hat{\mathbf{L}}$ is

$$\hat{\mathbf{L}}^{-1} \approx \sum_{i=1}^{M} \frac{\nu_i}{\lambda_{\min}} \exp(-\frac{\gamma_i}{\lambda_{\min}} L_d) \otimes \cdots \otimes \exp(-\frac{\gamma_i}{\lambda_{\min}} L_1),$$

where γ_i and $\nu_i,\ i = 1, \ldots, M$ are given parameters, and $\lambda_{\min} = \lambda_{\min}(\hat{\mathbf{L}}) = \sum_{i=1}^{d} \lambda_{\min}(L_i)$. The inverse matrix $\hat{\mathbf{L}}^{-1}$ is a TT operator with a maximal TT-rank M and we set M to be 3 in our numerical experiments. The choices of the parameters ν_k, γ_k and M with respect to a given error tolerance can be found in [6].

We have implemented a practical version of Algorithm 3, denoted as SPB, and Algorithm 5, denoted as ALSSPB, based on the TT/MPS tensor toolbox TTeMPS[1]. We compare

[1]Downloadable from `http://anchp.epfl.ch/TTeMPS`

SPB and ALSSPB with the state-of-the-art method EVAMEn [11]. We have also examined the block ALS method [2, 11]. Since its overall performance is less comparable with EVAMEn, we decided not to use it in this paper. Instead of starting from initial points generated randomly, we use EVAMEn with a maximal rank p and one sweep to produce a better point. When the Ritz pair $(\mathbf{X}, \Lambda)$ is a good approximation to the true eigenpair, i.e., $\mathbf{AX} \approx \mathbf{X}\Lambda$, it means that $\mathbf{AX}$ can be compressed with a TT-rank which is approximately equal to that of $\mathbf{X}$. Therefore the truncation error of $\mathbf{R} = \mathbf{AX} - \mathbf{X}\Lambda$ is relatively small and the convergence of numerical algorithms usually is faster. The parameters of EVAMEn are set to the same as these in [11]. Specifically, a locally preconditioned augmentation is applied in EVAMEn for acceleration. The maximal rank r in EVAMEn is set to 40. The maximal rank R of SPB and ALSSPB is taken so that the total number of variables matches that of EVAMEn. In fact, the variable is switched back and forth between the 1-BTT to d-BTT formats in EVAMEn while the variable is fixed as the 1-BTT format in ALSSPB. Hence, the total number of variables in EVAMEn and ALSSPB is $O(n((d+p-3)r^2+2r))$ and $O(n((d-2)R^2+(p+1)R))$, respectively.

All of our numerical experiments are preformed on a workstation with two twelve-core Intel Xeon E5-2697 CPUs and 128GB of memory running Ubuntu 12.04 and MATLAB 2013b.

5.1. Experiments on Newton Potential. We first test the Newton potential of the form

$$V(x) = \frac{1}{\|x\|}.$$

The parameters of the PDE problem (5.1) are set to $d = 10$ and $\Omega = (-1, 1)^d$. It is discretized with $n = 128$ grid points in each dimension which generates a discretized eigenvalue problem of size 128^{10}. The potential is approximated by a summation of ten rank-one TT tensors by using the technique of exponential sum (see [7] on the introduction to exponential sum and see [6] on the examples including $f(x) = \frac{1}{x}$). The approximation accuracy is 3.6×10^{-5} [11]. For preconditioning, the matrix L_μ in (5.3) is taken the same as L in (5.2).

We compare SPB and ALSSPB with EVAMEn for $p = 1$ and 11 eigenpairs. The numerical results are presented in Figures 5.1 and 5.2, respectively. In the figures, the solid lines correspond to residual errors measured by $||\mathcal{P}_{\mathbf{T}}(\mathbf{AU} - \mathbf{U}\Lambda)||_F$, where $(\mathbf{U}, \Lambda)$ is an approximated eigenpair. The dashed lines show the eigenvalue error $\sum_{i=1}^{p} |\Lambda_{ii} - \widehat{\Lambda}_{ii}|$, where $\widehat{\Lambda}$ are the best eigenvalues returned by all solvers. The residual and eigenvalue errors versus the iteration history in SPB and ALSSPB and the microiteration history of EVAMEn are shown in the left side of each figure. The errors with respect to the CPU time measured in seconds are presented in the right side of each figure. Each marker of the lines of SPB and ALSSPB indicates an iteration and each marker of the lines of EVAMEn indicates the completion of a half sweep.

Figure 5.1 shows that all algorithms achieve a similar accuracy in terms of both residual and eigenvalue errors in almost the same amount of time. We can observe from Figure 5.2 that ALSSPB is significantly faster than EVAMEn with a smaller accuracy. EVAMEn converges faster in one or two sweeps and then it tends to stagnate. The residual and eigenvalue errors of EVAMEn going up and down in the late microiterations on the case of $p = 11$ due to the variation in the number of parameters during shifts between block-d tensors and block-μ tensors, $\mu \neq d$. We can observe that SPB can also stagnate after reaching a certain accuracy. The jump of the residual in ALSSPB is due to our continuation strategy on increasing the maximal rank gradually. When the TT-rank is sufficient, the improvement in both residual and eigenvalue errors usually sharply slows down.

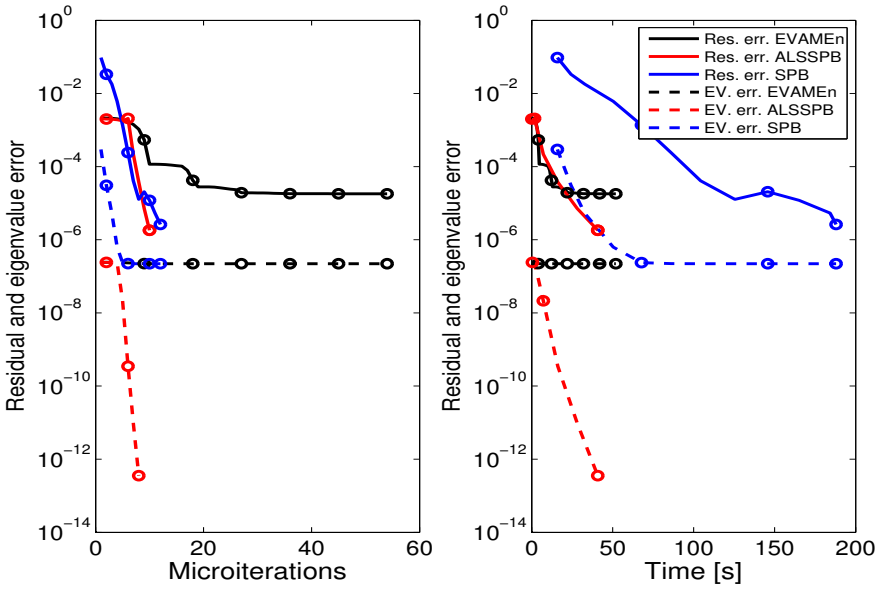

FIG. 5.1. *Numerical results on Newton potential with* $n = 128$, $d = 10$, *and* $p = 1$.

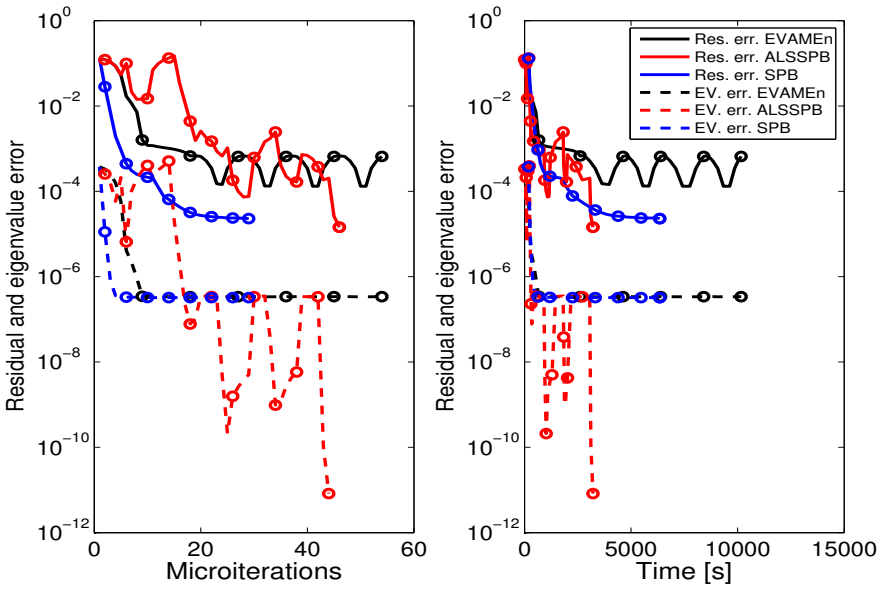

FIG. 5.2. *Numerical results on Newton potential with* $n = 128$, $d = 10$, *and* $p = 11$

Each microiteration of EVAMEn needs to compute p eigenpairs from a matrix of size nr^2 for p vectors while each iteration of SPB only need to compute a dense eigen-decomposition of size $3p$ by $3p$ and a few truncations. Hence, SPB and ALSSPB may be competitive to EVAMEn when n and r are not too small. We also should point out that EVAMEn is less sensitive to the initial points than SPB.

5.2. Experiments on Henon-Heiles Potential. Our second example uses the Henon-Heiles potential as

$$V(\mathbf{x}) = \frac{1}{2}\sum_{\mu=1}^{d} x_\mu^2 + \sum_{\mu=1}^{d-1}\left(\sigma_*\left(x_\mu x_{\mu+1}^2 - \frac{1}{3}x_\mu^3\right) + \frac{\sigma_*^2}{16}\left(x_\mu^2 + x_{\mu+1}^2\right)\right),$$

where σ_* is a constant and is taken to be 0.11 in the numerical experiment. We choose an unsymmetric domain $\Omega = (-10, 2)^{10}$ and the same discretization scheme as in [11] with $d = 10$ and $n = 128$. Let $D = (\zeta_1, \zeta_2, \cdots, \zeta_n)$ contain the discretized equidistant mesh points for one dimension and set $B_\mu = \sigma_* D + \frac{\sigma_*^2}{8} D^2$, $C_\mu = D^2$, and

$$L_\mu = \begin{cases} L + \frac{1}{2}D^2 - \frac{\sigma_*}{3}D^3 + \frac{\sigma_*^3}{16}D^4, & \mu = 1, \\ L + \frac{1}{2}D^2 - \frac{\sigma_*}{3}D^3 + \frac{\sigma_*^3}{8}D^4, & 2 \le \mu \le d-1, \\ L + \frac{1}{2}D^2 + \frac{\sigma_*^3}{16}D^4, & \mu = d. \end{cases} \tag{5.4}$$

Then the discretized operator $\mathbf{A}$ is represented by

$$\mathbf{A} = \sum_{\mu=1}^{d} I_n \otimes \cdots \otimes I_n \otimes L_\mu \otimes I_n \otimes \cdots \otimes I_n + \sum_{\mu=1}^{d-1} I_n \otimes \cdots \otimes I_n \otimes C_\mu \otimes B_\mu \otimes I_n \otimes \cdots \otimes I_n.$$

Section 3.3 shows that the above $\mathbf{A}$ has a TT operator representation with all ranks equal to 3. The matrix in (5.4) is used for preconditioning in the form of (5.3).

Similar to the case of Newton potential, we compare SPB and ALSSPB with EVAMEn for $p = 1$ and 11 eigenpairs and present in Figures 5.3 and 5.4, respectively. EVAMEn significantly outperforms ALSSPB in terms of the computational time on $p = 1$. They are comparable on $p = 11$. The convergence rate of the residual norms begins to slow down when the TT-rank is not sufficiently large enough. The accuracy of both methods can be improved by increasing the maximal rank.

6. Conclusion. The goal of this paper is to compute a few eigenpairs when the storage of the eigenvectors in the classical way is impossible. The main difference comparing to the classical eigensolver is expressing the coefficient matrix and its eigenvectors in the low-rank tensor train formats. Since the TT-ranks increase dramatically after several operations between tensor formats, such as the addition in the TT formats and matrix-vector multiplications in the TT format, the computational cost of many existing algorithms becomes eventually prohibitive. We propose a subspace optimization method combined with some suitable truncation steps to the given low-rank Tensor Train formats. Numerical results show that our algorithm is competitive to the ALS type methods on problems arising from the discretization of the stationary Schrödinger equation when the linear eigenvalue problem in each step of the ALS methods is still expensive.

Acknowledgment. We thank D. Kressner, M. Steinlechner and A. Uschmajew for sharing online their matlab codes on EVAMEn and the TT/MPS tensor toolbox TTeMPS.

REFERENCES

[1] JONAS BALLANI1 AND LARS GRASEDYCK, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43.

[2] S. V. DOLGOV, B. N. KHOROMSKIJ, I. V. OSELEDETS, AND D. V. SAVOSTYANOV, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Comput. Phys. Commun., 185 (2013), pp. 1207–C1216.

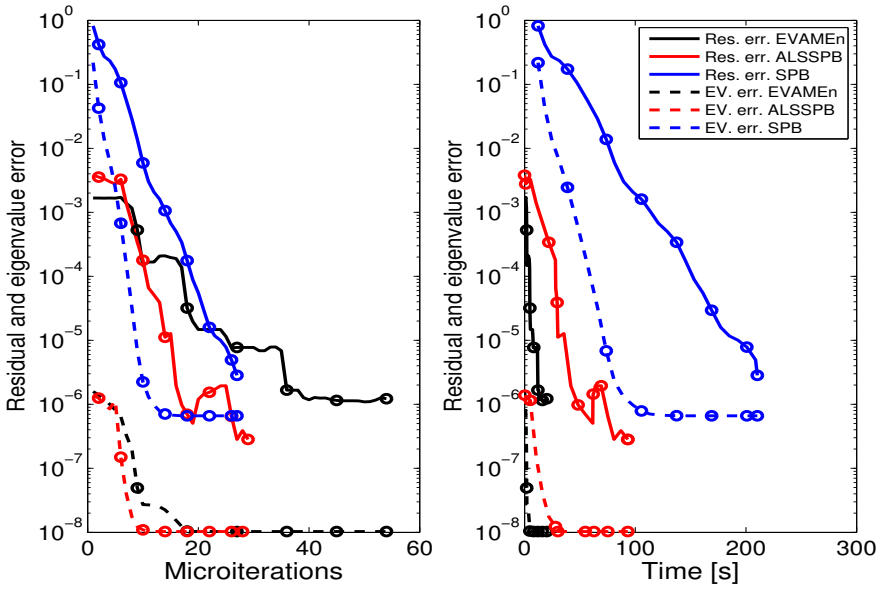

FIG. 5.3. *Numerical results on Henon-Heiles Potential with* $n = 128$, $d = 10$, *and* $p = 1$.

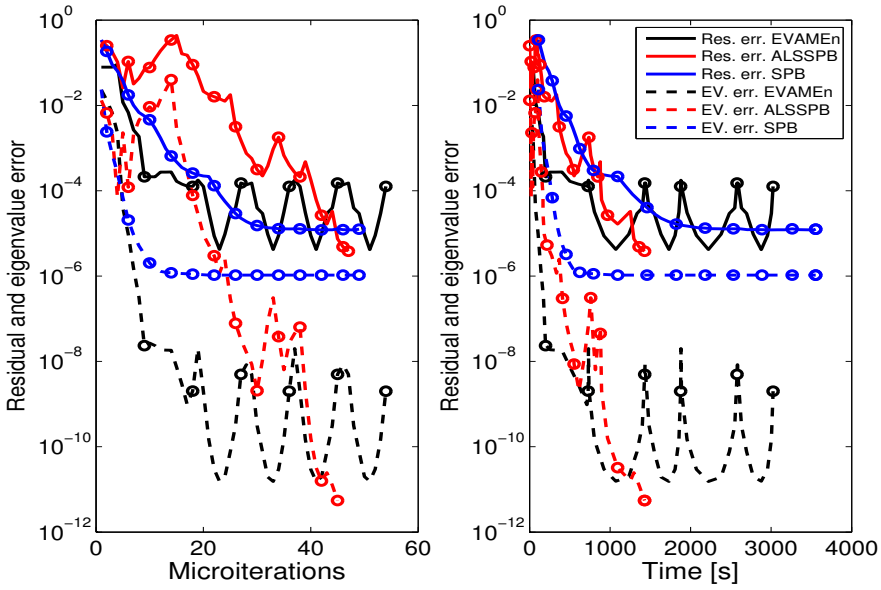

FIG. 5.4. *Numerical results on Henon-Heiles Potential with* $n = 128$, $d = 10$, *and* $p = 11$.

[3] L. GRASEDYCK, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.

[4] ———, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2009/10), pp. 2029–2054.

[5] L. GRASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.

[6] W. HACKBUSCH, *Entwicklungen nach exponentialsummen*, Technical Report 4, Max Planck Institute for Mathematics in the Sciences,/2005, MPI MIS Leipzig, 2010. Revised version, September.

[7] WOLFGANG HACKBUSCH, *Tensor Spaces and Numerical Tensor Calculus*, Springer, Heidelberg, 2012.
[8] S. HOLTZ, T. ROHWEDDER, AND R. SCHNEIDER, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.
[9] A. V. KNYAZEV, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[10] TAMARA G. KOLDA AND BRETT W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[11] D. KRESSNER, M. STEINLECHNER, AND A. USCHMAJEW, *Low-rank tensor methods with subspace correction for symmetric eigenvalue problems*, SIAM J. Sci. Comput., 36 (2014), pp. A2346–CA2368.
[12] D. KRESSNER AND C. TOBLER, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Comput. Methods Appl. Math., 11 (2011), pp. 363–381.
[13] ———, *htucker, a MATLAB toolbox for tensors in hierarchical Tucker format*, Tech. rep, 2012.
[14] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[15] O. S. LEBEDEVA, *Tensor conjugate-gradient-type method for Rayleigh quotient minimization in block QTT-format*, Russian J. Numer. Anal. Math. Modelling, 26 (2011), pp. 465–489.
[16] X. LIU, Z. WEN, AND Y. ZHANG, *Limited memory block krylov subspace optimization for computing dominant singular value decompositions*, SIAM Journal on Scientific Computing, 35-3 (2013), pp. A1641–A1668.
[17] IVAN OSELEDETS, *DMRG approach to fast linear algebra in the TT-format*, Comput. Methods Appl. Math., 11 (2011), pp. 382–393.
[18] I. V. OSELEDETS, *Approximation of* $2^d \times 2^d$ *matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl, 31 (2010), pp. 2130–2145.
[19] ———, *Tensor train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
[20] Z. WEN AND Y. ZHANG, *Block algorithms with augmented rayleigh-ritz projections for large-scale eigenpair computation*, Arxiv: 1507.06078, (2015).