SIAM J. SCI. COMPUT. © 2012 Society for Industrial and Applied Mathematics
Vol. 34, No. 5, pp. A2718–A2739

SOLUTION OF LINEAR SYSTEMS AND MATRIX INVERSION IN THE TT-FORMAT[*]

I. V. OSELEDETS[†] AND S. V. DOLGOV[†]

Abstract. Tensors arise naturally in high-dimensional problems in chemistry, financial mathematics, and many other areas. The numerical treatment of such problems is difficult due to the curse of dimensionality: the number of unknowns and the computational complexity grow exponentially with the dimension of the problem. To break the curse of dimensionality, low-parametric representations, or *formats*, have to be used. In this paper we make use of the TT-format (tensor-train format) which is one of the most effective stable representations of high-dimensional tensors. Basic linear algebra operations in the TT-format are now well developed. Our goal is to provide a "black-box" type of solver for linear systems where both the matrix and the right-hand side are in the TT-format. An efficient DMRG (density matrix renormalization group) method is proposed, and several tricks are employed to make it work. The numerical experiments confirm the effectiveness of our approach.

Key words. TT-format, solution of linear systems, high-dimensional problems, density matrix renormalization group

AMS subject classifications. 65F30, 65F50, 65N35, 65F10

DOI. 10.1137/110833142

1. Introduction.

Tensors arise naturally in high-dimensional problems, for example, in quantum chemistry [27, 26], financial mathematics [38, 41], and many other areas. The treatment of d-dimensional tensors is notoriously difficult due to the *curse of dimensionality*: the number of elements of a tensor grows exponentially with the number of dimensions d, as does the complexity when working with fully populated tensors. As in the matrix case, one can rely on the sparsity of tensors [2]. However, this helps only for small dimensions: in typical examples of high-dimensional operators, the number of nonzero entries grows exponentially with d, and the curse of dimensionality is not avoided. For d of order tens or hundreds, other approaches are needed, and special low-parametric representations, or *formats*, are required. Several formats have been proposed to represent a tensor in a data-sparse way. They include canonical and Tucker formats, which are the two formats with well-established properties and application areas; see the review [25] for more details. However, they have known drawbacks. To avoid these drawbacks, the development of new tensor formats began. In 2009 Hackbusch and Kühn [18], and later Grasedyck [15] and Oseledets and Tyrtyshnikov [35], proposed two (slightly different) hierarchical schemes for the tensor approximation, $\mathcal{H}$-Tucker and tree Tucker formats. These formats depend on specially chosen *dimension trees* and require recursive procedures. To avoid the recursion, it was proposed to use a simple matrix product form of the decomposition [30, 32], the so-called *tensor-train format* or, simply, TT-format.

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section May 6, 2011; accepted for publication (in revised form) June 26, 2012; published electronically October 11, 2012. This work was supported in part by RFBR grants 09-01-12058, 10-01-00757, 12-01-00546-a, and 11-01-00549, RFBR/DFG grant 09-01-91332, Russian Federation Government contracts Π1112 and 14.740.11.0345, Russian President grant MK-140.2011.1, and Promotionsstipendium of Max-Planck Gesellschaft. Part of this work was done during the authors' stay at the Max-Plank Institute for Mathematics in Sciences, Leipzig, Germany.
http://www.siam.org/journals/sisc/34-5/83314.html

[†]Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street 8, 119333 Moscow, Russia (ivan.oseledets@gmail.com, sergey.v.dolgov@gmail.com).

This paper is concerned with the solution of linear systems $Ax = b$ which arise from discretizations or representations of problems in high dimensions. Consider, for example, the discretization of a two-dimensional elliptic PDE under a lexicographical ordering. The matrix A is then an $nm \times nm$ block matrix with n, $m \times m$ submatrices defining each block row. Thus, two equivalent ways of indexing into A are by i, j position $1 \leq i \leq nm$, $1 \leq j \leq nm$ or by using four indices, two of which (bounded by n) represent the specific submatrix block in the block matrix, and two of which represent the matrix indices (and are thus bounded by m) within each block. The latter representation is equivalent to representing A as a fourth-order tensor. Similarly, a matrix that comes from the discretization of a three-dimensional elliptic PDE would correspond to a sixth-order tensor, and its solution could be considered as a third-order tensor. In this paper, we focus on the tensor representations of the linear operator A, the output x, and the right-hand side f. Specifically, we consider the case when the tensor forms of A and f can be expressed in the TT-format, and we desire solutions that are also in the TT-format.

An $n_1 \times n_2 \times \cdots \times n_d$ tensor $\mathbf{G}$ is said to be in the TT-format if its elements are defined by formula

$$G(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d), \tag{1.1}$$

where $G_k(i_k)$ is an $r_{k-1} \times r_k$ matrix for each fixed $i_k, 1 \leq i_k \leq n_k$. To make the matrix-by-matrix product in (1.1) scalar, boundary conditions $r_0 = r_d = 1$ have to be imposed. The numbers r_k are called *TT-ranks*, and $G_k(i_k)$ are the cores of the TT-decomposition of a given tensor. If $r_k \leq r, n_k \leq n$, then the storage of the TT-representation requires $\leq dnr^2$ memory cells. If r is small, then this is much smaller than the storage of the full array, n^d.

The TT-format was introduced in [30, 32] as an alternative to two commonly used formats: the canonical format and the Tucker format. These two formats can be considered as different generalizations of the singular value decomposition (SVD) from matrices (i.e., $d = 2$) to higher order tensors. An $n_1 \times n_2 \times \cdots \times n_d$ tensor $\mathbf{G}$ is said to be in the canonical format if

$$G(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha). \tag{1.2}$$

This representation is often referred to as *CANDECOMP/PARAFAC* or, simply, the *CP model*. If r, the canonical rank, is small, then the number of parameters depends on d linearly. The Tucker model [39, 8] is the representation of the form

$$G(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_d} C(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d),$$

where U_k are called *Tucker factors*, and the tensor $G(\alpha_1, \ldots, \alpha_d)$ is the *Tucker core*. The canonical format is a good candidate for low-parametric representation of tensors. It can be used to approximate high-dimensional operators and their inverses (see, for example, [3, 4, 16, 17, 14]) using certain analytical expansions. However, the canonical format has a serious drawback: there are no robust algorithms to compute the canonical approximation numerically. There are efficient algorithms for computing a low-rank approximation of a given tensor [6, 19, 12, 5], but there is no general method for finding the best or quasi-best approximation. The best approximation problem in the canonical format can be ill-posed [10], and the best approximation may not even exist.

In two dimensions the CP decomposition reduces to the skeleton (dyadic) decomposition of a matrix, which can be computed via the SVD. The SVD is stable and efficient. These properties are lost for tensors of order $d \geq 3$. The Tucker decomposition is stable and an optimal approximation always exists, and a quasi-optimal approximation can be computed via the higher order SVD (HOSVD) [8, 9]. However, it has an exponential in d number of parameters, and thus it can be applied only for small d (for the three-dimensional case $d = 3$ (see the Cross-3D method in [33], and for application of the Tucker model to the solution of Hartree–Fock equation, see [23]).

The TT-format falls between the CP and Tucker formats: it does not have an intrinsic exponential dependence on d, but it is stable in the sense that the best approximation with bounded TT-ranks always exists, and a quasi-optimal approximation can be computed by a sequence of SVDs of auxiliary matrices [30, 32]. Thus it has a great potential for the stable and robust approximation of high-dimensional tensors.

The TT-format comes with all basic linear algebra operations. Addition, matrix-by-vector product, and elementwise multiplication can be computed in linear in d and polynomial in r complexity with the result also in the TT-format [30, 32]. The problem is that after such operations, TT-ranks begin to grow. For example, the TT-ranks of the sum of two tensors are equal (formally) to the sum of the TT-ranks of the addends. In the case of the matrix-by-vector product, the result is also in the TT-format, with the TT-ranks of matrix and vector multiplied. After several iterations, the TT-ranks will become too large, and thus *rounding* is needed: a given tensor $\mathbf{A}$ is approximated by another tensor $\mathbf{B}$ with minimal possible TT-ranks with a prescribed accuracy ε:

$$||\mathbf{A} - \mathbf{B}||_F \leq \varepsilon ||\mathbf{A}||_F.$$

The rounding procedure in the TT-format can be implemented in $\mathcal{O}(dnr^3)$ operations [30, 32].

Methods for solving "advanced" high-dimensional problems, such as eigenvalue problems and linear systems, can be constructed directly from the standard iterative methods using TT-arithmetic with rounding. However, they can be rather slow due to the large number of iterations, and thus a certain preconditioner is needed: they are not "black-box"-type solvers.

Our main goal in this paper is to present a black-box solver for the linear system with both the matrix and the right-hand side in the TT-format, and the approximate solution is sought in the TT-format directly. Such problems routinely appear in the solution of high-dimensional problems, for example, the Fokker–Planck equation in polymeric fluid modeling (see [13, 1] for the solution in the canonical format, and see [11] for the preliminary application of the TT-format).

In order to do this, the basic problem,

$$Ax = f,$$

has to be reformulated as a minimization problem. The simplest functional is

$$||Ax - f|| \to \min. \tag{1.3}$$

Then, the minimization over $\mathbb{R}^N, N = n^d$, is reduced to the minimization over a certain class of structured solutions $\mathcal{S}$. In our case, x is associated with a tensor $X(i_1, \ldots, i_d)$ with small TT-ranks. Then, $\mathcal{S}$ is the so-called *TT-manifold* [20].

Then, (1.3) becomes a nonquadratic minimization problem, but over a much smaller parameter space.

For simplicity, in the symmetric positive definite case a functional

$$J(x) := (Ax, x) - 2(f, x) \tag{1.4}$$

will be considered.

The minimization problem (1.4) can be solved by any suitable general-purpose minimization method (Newton method, conjugate gradient). However, this approach avoids the very specific structure of the solution and of the functional.

The simplest method which makes use of this structure is an *ALS (alternating least squares)* method.[1] If all cores except one are fixed, the problem becomes a quadratic problem and can be reduced to a small linear system. Then one can minimize over the next core and so on, up to d. This approach guarantees that the value of the functional will not increase in each iteration step. This method, however, requires the knowledge of all TT-ranks. There are $d-1$ TT-ranks, and if they are underestimated, the solution will not be close to the true solution. If they are overestimated, then the complexity may be too high. Usually, one can specify the accuracy ε, to which the solution is sought, and the ALS method is nonadaptive in the sense that it requires all TT-ranks to be known in advance. The second problem is that the convergence of the ALS approach may be too slow. How do we modify the ALS method to avoid these problems?

The idea of modifying the ALS scheme was brought to the field of numerical analysis in the paper [21] by Holtz, Rohwedder, and Schneider, and it was also used to construct efficient eigenvalue solvers for Hamiltonians in quantum molecular dynamics [24].

The idea of the modified ALS comes from the solid state physics for the computation of the minimal eigenvalue of quantum spin Hamiltonians. It is the so-called DMRG (density matrix renormalization group) method of White [42]. A quantum spin system consists of L *spins*, and such a system is defined by a wavefunction with S^L parameters, where typically $S = 2$ or $S = 4$. This wavefunction can be considered as a L-dimensional tensor. The computation of the minimal eigenvalue can be reformulated as a problem of the minimization of the Rayleigh quotient

$$(H\psi, \psi), \quad ||\psi|| = 1$$

(compare with (1.4), which is very similar), and the solution is sought as a matrix product state [36], which is equivalent to the TT-format. The DMRG is an ALS method, but with one minor modification. Instead of minimizing over one core, the functional is minimized over a pair of cores. The special structure of the TT-format still allows us to reduce this minimization problem to the quadratic one, which is not possible for the canonical format.

If the matrix of the problem A comes from the discretization of a high-dimensional equation, and the one-dimensional mode size n is usually not small, the so-called QTT-format (*quantized TT-format*[2]) can be very helpful. The idea behind the QTT-format is the following. Consider a vector of values of a univariate function on a grid with 2^L points, i.e., $n = 2^L$. By the binary coding, this vector can be considered

[1]Note, that the ALS method in this paper is not the well-known method of tensor approximation in the canonical format. However, they have the same ideology, and thus the same term is used.

[2]Originally the name was *quantics TT-format*, but now it is clear that "quantized" is a more appropriate term.

as an L-dimensional tensor with all mode sizes equal to 2. In combination with the TT-format, this yields the *QTT-format* for representing vectors [31, 22]. This transformation of vectors into tensors, "tensorization," can be generalized to an arbitrary dimension. If a function of d variables is discretized on a tensor grid with 2^L points in each direction, then the corresponding d-dimensional tensor can be reshaped into a dL-dimensional tensor with small mode sizes. This leads to a problem with small mode sizes that is algebraically similar to the problem for the quantum spin Hamiltonian, and the size of the local DMRG problem is only two times larger than that for the traditional ALS step.

In this paper we take one step further (especially compared to the work [21]) toward the efficient realistic black-box solver of linear systems in the TT-format. A direct implementation encounters serious difficulties, which have to be tackled. Several "tricks" will be proposed, which can be considered as technical, but they make the method work.

The paper is organized as follows. In section 2 basic facts about TT-format and notation used are described. In section 3 optimization problems for ALS/DMRG schemes are posed, the explicit formulae for the local problems are derived, and the complexity is estimated. Section 4 contains the description of tricks that make the method more robust. In section 5 the developed algorithms are adapted to perform fast matrix inversion. In section 6 the numerical experiments on several model problems are presented, confirming the efficiency of the proposed solver.

2. Notation and basic facts.

2.1. Parameter-dependent matrices.

In this section several basic facts and definitions are recollected. Throughout the paper, any multi-index object, i.e., $G_k(i_k)$, is a matrix depending on parameters, and these parameters are listed in brackets. In other words, a parameter-dependent matrix is a matrix-valued function of integer arguments. The size of the parameter-dependent matrix should be clear from the context (i.e., $r_{k-1} \times r_k$). The number of parameters can be arbitrary. This convention allows us to replace certain summations by matrix products, simplifying the presentation. For example, multiplication of two parameter-dependent matrices yields a new parameter-dependent matrix with a larger number of parameters:

$$W(i_k, i_{k+1}) = G_k(i_k)G_{k+1}(i_{k+1}).$$

Individual elements of the parameter-dependent matrices are referred to as $G_k(i_k)_{(\alpha_{k-1},\alpha_k)}$. This operation will be used often, and we will use a special notation for it.

DEFINITION 2.1. *For a given parameter-dependent matrix $G_k(i_k)$, ${}^{<}G_k^{>}(\alpha_{k-1}, i_k, \alpha_k)$ is a parameter-dependent scalar function of arguments $\alpha_{k-1}, i_k, \alpha_k$ (i.e., a matrix of size 1×1),*

$$\begin{gathered} {}^{<}G_k^{>}(\alpha_{k-1}, i_k, \alpha_k) = G_k(i_k)_{(\alpha_{k-1},\alpha_k)}, \\ \alpha_{k-1} = 1, \dots, r_{k-1}, \quad \alpha_k = 1, \dots, r_k, \quad i_k = 1, \dots, n_k. \end{gathered} \tag{2.1}$$

The notation (2.1) can be extended to extract either a row or a column index of a parameter-dependent matrix.

DEFINITION 2.2. *Given a parameter-dependent matrix $G_k(i_k)$, by ${}^{<}G_k(\alpha_{k-1}, i_k)$ and $G_k^{>}(i_k, \alpha_k)$ we denote parameter-dependent matrices of sizes $1 \times r_k$ and $r_{k-1} \times 1$, respectively, with elements*

$${}^{<}G_k(\alpha_{k-1}, i_k) = G_k(i_k)_{(\alpha_{k-1},:)}, \quad G_k^{>}(i_k, \alpha_k) = G_k(i_k)_{(:,\alpha_k)}.$$

One should be very careful, since the matrix-by-matrix product is noncommutative, and the order is very important with this notation.

Using the above conventions, the TT-format can be written in a simple "separable" form (1.1). Note that, in the definition of the canonical format (1.2), $U_k(i_k, \alpha)$ are scalars (1×1 matrices), and thus (1.2) also fits well into the notation.

The set of tensors with bounded TT-ranks forms a manifold [20].

Definition 2.3 (TT-manifold). *Let* $\mathbf{r} = (r_1, \ldots, r_{d-1})$ *be a given* rank parameter. *Then* $\mathbb{TT}_{\mathbf{r}}$ *is the set of tensors that can be represented in the form* (1.1).

2.2. Vectors and matrices in the TT-format. The functions and their discretizations in this paper are directly related to d-dimensional tensors. For brevity, we will use the term "vector in the TT-format." By that we implicitly assume that the vector in question has length $N = n_1 n_2 \ldots n_d$, and it can be considered as an $n_1 \times \cdots \times n_d$ tensor which has low TT-ranks. Square matrices acting on such vectors have size $N \times N$, and their elements can be naturally indexed by a $2d$-tuple $(i_1, \ldots, i_d, j_1, \ldots, j_d)$, $1 \leq i_k, j_k \leq n_k$, i.e., $M(i_1, \ldots, i_d, j_1, \ldots, j_d)$. A matrix M is said to be in the TT-format if

$$M(i_1, \ldots, i_d, j_1, \ldots, j_d) = M_1(i_1, j_1) M_2(i_2, j_2) \ldots M_d(i_d, j_d), \tag{2.2}$$

where $M_k(i_k, j_k)$ is an $r_{k-1} \times r_k$ matrix, and $r_0 = r_d = 1$. If all TT-ranks are equal to 1, then (2.2) reduces to

$$M = M_1 \otimes \cdots \otimes M_d,$$

i.e., it has Kronecker rank 1. Thus, (2.2) is a generalization of a standard low-rank approximation of high-dimensional operators [40, 3].

2.3. Left and right orthogonalization of the TT-format. The TT-representation (1.1) is nonunique, since it is invariant under a transformation

$$G'_k(i_k) := G_k(i_k) S, \quad G'_{k+1}(i_{k+1}) := S^{-1} G_{k+1}(i_{k+1})$$

for any nonsingular matrix S. Using such transformations, certain cores of the TT-representation can be made *orthogonal*. There are two types of orthogonality of cores: left-orthogonality and right-orthogonality. First, introduce a notion of left and right unfolding of the core.

Definition 2.4 (left and right unfolding). *For a given parameter-dependent matrix* $G_k(i_k)$*, its left unfolding, denoted by* $L(G_k(i_k))$ *is defined as*

$$L(G_k(i_k)) = \begin{pmatrix} G_k(1) \\ G_k(2) \\ \vdots \\ G_k(n_k) \end{pmatrix}. \tag{2.3}$$

The matrix $L(G_k(i_k))$ *has size* $(r_{k-1} n_k) \times r_k$. *The right unfolding* $R(G_k(i_k))$ *is defined as*

$$R(G_k(i_k)) = \begin{pmatrix} G_k(1) & G_k(2) & \cdots & G_k(n_k) \end{pmatrix}. \tag{2.4}$$

The matrix $R(G_k(i_k))$ *is of size* $r_{k-1} \times (n_k r_k)$.

Using $L(\cdot)$ and $R(\cdot)$ operators, it is easy to define left- and right-orthogonality of cores.

DEFINITION 2.5 (left- and right-orthogonality). *The core $G_k(i_k)$ is said to be left-orthogonal if $L(G_k(i_k))$ has orthonormal columns and to be right-orthogonal if $R(G_k(i_k))$ has orthonormal rows.*

The cores $G_1(i_1), \dots, G_k(i_k)$ can be made left-orthogonal by equivalent transformations, starting from the first core. Indeed, $G_1(i_1)$ can be written as

$$G_1(i_1) = Q_1(i_1)R,$$

where $Q_1(i_1)$ is left-orthogonal by applying the QR-decomposition to a $L(G_1)$. Then, the matrix R is incorporated into the second core:

$$G'_2(i_2) = RG_2(i_2),$$

and the same procedure is applied to the modified core $G'_2(i_2)$, and the second and the third cores will be modified, and so on. This procedure is called *orthogonalization* of cores [32] and can be implemented in $\mathcal{O}(dnr^3)$ complexity.

3. DMRG scheme for linear systems.

3.1. Basic idea. First, let us focus on the DMRG scheme applied to the minimization of the functional

$$J(x) := (Ax, x) - 2(f, x), \tag{3.1}$$

which is equivalent to the solution of a linear system

$$Ax = f$$

for the symmetric and positive definite matrix A. We suppose that the following assumptions hold:

1. The matrix A can be represented (or approximated) in the TT-format:
$$A(i_1, \dots, i_d, j_1, \dots, j_d) = A_1(i_1, j_1)A_2(i_2, j_2)\dots A_d(i_d, j_d). \tag{3.2}$$
2. The right-hand side f can be represented (or approximated) in the TT-format:
$$F(i_1, \dots, i_d) = F_1(i_1)F_2(i_2)\dots F_d(i_d). \tag{3.3}$$
3. The solution x can be well-approximated in the TT-format:
$$X(i_1, \dots, i_d) \approx X_1(i_1)X_2(i_2)\dots X_d(i_d). \tag{3.4}$$

The validity of the assumptions (3.2), (3.3), and (3.4) has to be verified for each particular problem. For example, the solution of Laplace-like equations can be well-approximated in the canonical format [16, 17] and hence in the TT-format as well. However, for many practically interesting cases, the existence of the structured solution can be established only numerically, and thus the black-box solver for the linear system has great practical importance in the high-dimensional applications as an experimental tool.

Algorithm 1 ALS sweep.

```
Require: Initial cores $X_1(i_1), \dots, X_d(i_d)$
Ensure: Improved approximation $\widehat{X}_1(i_1), \dots, \widehat{X}_d(i_d)$.
1: Set $\widehat{X}_k(i_k) := X_k(i_k)$
2: for $k = 1, \dots, d, d-1, \dots, 1$ do
3:     $\widehat{X}_k(i_k) := \arg\min_{\widetilde{X}_k(i_k)} J(\widehat{X}), \quad \widehat{X} = \widehat{X}_1(i_1) \dots \widetilde{X}_k(i_k) \dots \widehat{X}_d(i_d)$.
4: end for
```

3.2. Formulation of the optimization problem. In this subsection we formulate the optimization problems that we are going to solve. For the ALS method the formulation is rather simple. The manifold of the structured solutions is the manifold of tensors with bounded TT-ranks, $\mathbb{TT}_{\mathbf{r}}$, where $\mathbf{r} = (r_1, \dots, r_{d-1})$ is the rank parameter.

PROBLEM 3.1 (ALS optimization problem).

$$X_{\mathrm{ALS}} = \arg \min_{X \in \mathbb{TT}_{\mathbf{r}}} J(X). \tag{3.5}$$

The ALS algorithm is then obtained by the sequential minimization of (3.5) over the cores $X_1(i_1), \dots, X_d(i_d)$. It consists of elementary sweeps through the cores. Algorithm 1 describes the principal scheme of one sweep. Each local problem is quadratic in the unknowns and can be reduced to a small linear system. Later on we will derive efficient methods to obtain its solution.

The DMRG algorithm is obtained by minimizing (3.1) over a supercore $W(i_k, i_{k+1}) = X_k(i_k)X_{k+1}(i_{k+1})$. The local problem is the solution of the following optimization problem.

PROBLEM 3.2 (DMRG local problem). *Given an initial approximation* $X = X_1(i_1) \dots X_{k-1}(i_{k-1}) W_k(i_k, i_{k+1}) X_{k+2}(i_{k+2}) \dots X_d(i_d)$, *find an improved supercore*

$$\begin{aligned} &\widehat{W}_k(i_k, i_{k+1}) := \arg \min_{\widetilde{W}_k(i_k, i_{k+1})} J(X), \\ &X = X_1(i_1) \dots X_{k-1}(i_{k-1}) \widetilde{W}_k(i_k, i_{k+1}) X_{k+2}(i_{k+2}) \dots X_d(i_d). \end{aligned} \tag{3.6}$$

After $W_k(i_k, i_{k+1})$ has been computed, the new cores $X_k(i_k), X_{k+1}(i_{k+1})$ are computed with the help of the SVD:

$$X_k(i_k) X_{k+1}(i_{k+1}) \approx W_k(i_k, i_{k+1}).$$

This is called the *decimation step.* An important problem is that in this step the rank r_k is likely to change! Thus, the manifold is updated during each local step. The goal is now not only to compute the optimal approximation on the fixed manifold, but also to compute a rank parameter $\mathbf{r}$ and the corresponding manifold $\mathbb{TT}_{\mathbf{r}}$ such that the solution of the ALS problem (3.5) approximates in a certain sense the true solution. There are several possible ways to rigorously formulate this problem. Denote by X^* the minimum over the whole space:

$$X^* = \arg \min_{X} J(X), \quad X \in \otimes_{i=1}^{d} \mathbb{R}^{n_i}.$$

Then, we can seek to approximate the solution by a minimal number of parameters.

Problem 3.3 (DMRG solution approximation). *For a given accuracy ε find a rank parameter* $\mathbf{r}$ *such that the set*

$$\mathbb{TT}_{\mathbf{r}} \cap \{X : ||X - X^*|| \leq \varepsilon\} \neq \emptyset,$$

and the number of parameters describing an element of $\mathbb{TT}_{\mathbf{r}}$ *is minimal.*

Unfortunately, X^* is usually not easy to compute, and thus it is convenient to replace Problem 3.3 by the approximation of the functional itself.

Problem 3.4 (DMRG solution via approximation of the functional). *For a given accuracy ε find a rank parameter* $\mathbf{r}$ *such that the set*

$$\mathbb{TT}_{\mathbf{r}} \cap \{X : 0 \leq J(X) - J(X^*) \leq \varepsilon\} \neq \emptyset,$$

and the number of parameters describing an element of $\mathbb{TT}_{\mathbf{r}}$ *is minimal.*

The decimation step provides a simple way to increase the rank if the approximation is found to be inadequate. It is not our goal to find the exact minimum: we are also satisfied with quasi-optimal solutions to Problems 3.3 and 3.4, where the number of parameters is sufficiently small.

3.3. Derivation of the main equations.

There are several ways to derive an expression for the local subproblem; for example, in [24] compact representations of matrix-by-vector and scalar products were used, and in [21] a diagrammatic representation was utilized.

Here slightly different derivation will be given, which will show that the DMRG (or ALS) is in fact a *projection method* on adaptively chosen subspaces. Moreover, an orthogonal basis can be selected in these subspaces. In the rest of this subsection we focus on the ALS scheme, since the local problems on a certain core (or supercore) are very similar in the ALS and DMRG methods. Indeed, the supercore can be considered as one block $W(\widetilde{i}_k) \equiv W(i_k, i_{k+1})$, and the DMRG scheme reduces to the ALS, applied to the $(d-1)$-dimensional problem.

Suppose that the kth core is varied (and all others are fixed). The linear system

$$Ax = f$$

can be treated as an overdetermined linear system in the unknown core. Introduce auxiliary parameter-dependent matrices $\widehat{A}_k(i_k, j_k)$ of size $N_k \times N_k$, $N_k = \prod_{s \neq k} n_s$, defined as

$$\widehat{A}_k(i_k, j_k)_{((i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_d),(j_1,\ldots,j_{k-1},j_{k+1},\ldots,j_d))} = A(i_1, \ldots, i_d,\ j_1, \ldots, j_d),$$

where the rows of $\widehat{A}_k(i_k, j_k)$ are indexed by a multi-index $(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d)$, and the columns are defined by a multi-index $(j_1, \ldots, j_{k-1}, j_{k+1}, \ldots, j_d)$, i.e., only with the kth index omitted. In the same fashion, parameter-dependent vectors $\widehat{x}_k(j_k)$ and $\widehat{f}_k(i_k)$ are defined, both having sizes $N_k \times 1$. Then, the linear system can be written as

$$\sum_{j=1}^{n_k} \widehat{A}_k(i_k, j_k)\widehat{x}_k(j_k) = \widehat{f}_k(i_k), \quad 1 \leq i_k, j_k \leq n_k.$$

The TT-representation of X leads to the representation of form

$$\widehat{x}_k(j_k) = Q_k w_k(j_k), \tag{3.7}$$

where Q_k is an $N_k \times r_{k-1}r_k$ matrix, and $w_k(j_k)$ is an $(r_{k-1}r_k)\times 1$ vector. The elements of Q_k are given as

$$Q_{k\ ((i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_d),(\alpha_{k-1},\alpha_k))} \\ = X_1(i_1)\ldots X^{>}_{k-1}(i_k,\alpha_{k-1})^{<}X_{k+1}(\alpha_k,i_{k+1})\ldots X_d(i_d), \tag{3.8}$$

where the rows of Q_k are indexed by the multi-index $(i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_d)$ and the columns by the multi-index (α_{k-1},α_k). By equivalent transformations of the TT-representation of x, the matrix Q_k can be made orthogonal [32].

For a symmetric positive definite case, the minimization of the functional

$$(Ax,x)-2(f,x)=\sum_{i_k,j_k}\widehat{x}_k^{\top}(i_k)\widehat{A}_k(i_k,j_k)\widehat{x}_k(j_k)-2\sum_{i_k}\widehat{x}_k^{\top}(i_k)\widehat{f}_k(i_k),$$

under the restriction (3.7), leads to a smaller linear system of form

$$\sum_{j_k}(Q_k^{\top}\widehat{A}(i_k,j_k)Q_k)w_k(j_k)=Q_k^{\top}\widehat{f}_k(i_k). \tag{3.9}$$

The matrices $B_k(i_k,j_k)=Q_k^{\top}\widehat{A}(i_k,j_k)Q_k$ are of size $(r_{k-1}r_k)\times(r_{k-1}r_k)$ for each fixed pair i_k,j_k. Note that (3.9) is a linear system with $(r_{k-1}r_kn_k)$ unknowns.

An important observation is that even if A is indefinite, or even nonsymmetric, (3.9) can be used, but now it is not a minimization method but a projection method on a subspace defined by the orthogonal matrix Q_k. In this setting it is unclear why this method should converge, but our numerical experiments show that in several cases it is a good choice. The matrices $B_k(i_k,j_k)$ can be computed from the TT-representations of A and x. In each step of the sweep one should solve a linear system. The matrix of this linear system is small, compared to the full matrix; however, its actual size can be quite large and thus special techniques are required to form this matrix and to solve the linear system with it. It appears that it might be more economical to solve even the local linear system with the help of some iterative solver.

3.4. Fast computation in local problems. We have to derive explicit formulae and computational algorithms to solve (3.9). In order to do that, expressions for the elements of

$$\widehat{B}_k((\beta_{k-1},\beta_k),i_k,j_k,(\gamma_{k-1},\gamma_k))=\ ^{<}B_k^{>}((\beta_{k-1},\beta_k),i_k,j_k,(\gamma_{k-1},\gamma_k))$$

have to be obtained (here we used the fact that the rows of the parameter-dependent matrix $B_k(i_k,j_k)$ can be indexed by a pair of indices (β_{k-1},β_k) and the columns by a pair (γ_{k-1},γ_k)).

By the definition of $B_k(i_k,j_k)$,

$$\begin{aligned}&\widehat{B}_k((\beta_{k-1},\beta_k),i_k,j_k,(\gamma_{k-1},\gamma_k))\\ &\quad=\sum_{i_s,s\neq k}\sum_{j_l,l\neq k}A(i_1,\ldots,i_d,j_1,\ldots,j_d)^{<}Q_k^{>}((i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_d),(\beta_{k-1},\beta_k))\\ &\qquad\times^{<}Q_k^{>}((j_1,\ldots,j_{k-1},j_{k+1},\ldots,j_d),(\gamma_{k-1},\gamma_k)),\end{aligned}$$

where the summation goes through all the indices $i_1,\ldots,i_d$, except i_k, and all the indices $j_1,\ldots,j_d$, except j_k. Using the representations (3.2), (3.8), the following formula is obtained:

$$\widehat{B}((\beta_{k-1},\beta_k),i_k,j_k,(\gamma_{k-1},\gamma_k))=\Psi_{k-1}(\beta_{k-1},\gamma_{k-1})A_k(i_k,j_k)\Phi_k(\beta_k,\gamma_k), \tag{3.10}$$

where

$$\tag{3.11} \begin{aligned} \Psi_{k-1}(\beta_{k-1},\gamma_{k-1}) &= \sum_{i_1,\ldots,i_{k-1},j_1,\ldots,j_{k-1}} \Big[A_1(i_1,j_1)\ldots A_{k-1}(i_{k-1},j_{k-1})\Big] \\ &\quad\times \Big[X_1(i_1)\ldots X_{k-1}^{>}(i_{k-1},\beta_{k-1})\Big]\Big[X_1(j_1)\ldots X_{k-1}^{>}(j_{k-1},\gamma_{k-1})\Big], \\ \Phi_k(\beta_k,\gamma_k) &= \sum_{i_{k+1},\ldots,i_d,j_{k+1},\ldots,j_d} \Big[A_{k+1}(i_{k+1},j_{k+1})\ldots A_d(i_d,j_d)\Big] \\ &\quad\times \Big[{}^{<}X_{k+1}(\beta_k,i_{k+1})\ldots X_d(i_d)\Big]\Big[{}^{<}X_{k+1}(\gamma_k,j_{k+1})\ldots X_d(j_d)\Big]. \end{aligned}$$

Equations (3.11) can be written as a simple recursion formula,

$$\tag{3.12} \begin{aligned} \Psi_k(\beta_k,\gamma_k) = \sum_{i_k,j_k,\beta_{k-1},\gamma_{k-1}} &\Psi_{k-1}(\beta_{k-1},\gamma_{k-1})A_k(i_k,j_k) \\ &\times {}^{<}X_k^{>}(\beta_{k-1},i_k,\beta_k){}^{<}X_k^{>}(\gamma_{k-1},j_k,\gamma_k), \end{aligned}$$

where, by convention, $\Psi_0 = 1$. An analogous representation can be obtained for Φ_k. $\Psi_k(\beta_k,\gamma_k)$ is a parameter-dependent matrix of size $1 \times R_k$, where R_k are the TT-ranks of the matrix A (i.e., the size of the parameter-dependent matrix $A_k(i_k,j_k)$ is $R_{k-1} \times R_k$). It can be shown that the complexity of computing Φ_k is

$$\mathcal{O}(dn^2r^2R^2 + dn^2r^3R).$$

Often $r \gg R$, and in this case the complexity is linear in the matrix rank and is cubic in the solution rank.

The linear system (3.9) has n^2r^2 unknowns. Even for $n = 2$, its size becomes large for $r \geq 25$. However, it is completely defined by $\Psi_{k-1}(\beta_{k-1},\gamma_{k-1})$, $A_k(i_k,j_k)$, and $\Phi_{k-1}(\beta_k,\gamma_k)$, which require $r_{k-1}^2R_{k-1} + r_k^2R_k + n^2R_{k-1}R_k = \mathcal{O}(n^2R^2 + r^2R)$ parameters to store. Moreover, using this additional structure, the matrix of linear system (3.9) can be multiplied by an arbitrary vector quickly. Indeed, the matrix-by-vector product in this case reduces to the computation of the sum of form

$$W(\beta_{k-1},i_k,\beta_k) = \sum_{\gamma_{k-1},j_k,\gamma_k} \widehat{B}((\beta_{k-1},\beta_k),i_k,j_k,(\gamma_{k-1},\gamma_k))Y(\gamma_{k-1},j_k,\gamma_k).$$

Using the expression (3.10) for the elements of $\widehat{B}$, we get

$$W(\beta_{k-1},i_k,\beta_k) = \sum_{\gamma_{k-1},j_k,\gamma_k} \Psi_{k-1}(\beta_{k-1},\gamma_{k-1})A_k(i_k,j_k)\Phi_k(\beta_k,\gamma_k)Y(\gamma_{k-1},j_k,\gamma_k).$$

This sum can be computed in three steps. First, the sum over γ_k yields

$$\sum_{\gamma_k} \Phi_k(\beta_k,\gamma_k)Y(\gamma_{k-1},j_k,\gamma_k) = Y'(\beta_k,\gamma_{k-1},j_k),$$

where $Y'(\beta_k,\gamma_{k-1},j_k) \in \mathbb{R}^{R_k\times 1}$, and it can be realized as a product of a matrix of size $R_kr_k \times r_k$ by a matrix of size $r_k \times (n_kr_{k-1})$. The cost is $\mathcal{O}(nRr^3)$ operations. The contraction over j_k gives

$$Y''(\beta_k,i_k,\gamma_{k-1}) = \sum_{j_k} A_k(i_k,j_k)Y'(\beta_k,\gamma_{k-1},j_k),$$

where $Y''(\beta_k, i_k, \gamma_{k-1}) \in \mathbb{R}^{R_{k-1}\times 1}$. It can be realized as a product of a matrix of size $(R_{k-1}n_k)\times(n_kR_k)$ by a matrix of size $(n_kR_k)\times(r_{k-1}r_k)$, and the cost is $\mathcal{O}(n^2R^2r^2)$. Finally, the summation over γ_{k-1} gives the result

$$W(\beta_{k-1}, i_k, \beta_k) = \sum_{\gamma_{k-1}} \Psi_{k-1}(\beta_{k-1}, \gamma_{k-1}) Y''(\beta_k, i_k, \gamma_{k-1}),$$

which can be implemented via a product of a matrix of size $r_{k-1}\times(r_{k-1}R_{k-1})$ by a matrix of size $(r_{k-1}R_{k-1})\times(n_kr_k)$. The cost of computing such product is $\mathcal{O}(nRr^3)$. The total cost of computing W (i.e., the local matrix-by-vector product) is $\mathcal{O}(nRr^3 + n^2R^2r^2)$. If the matrix B is stored as a full matrix, the cost of its matrix-by-vector product is $\mathcal{O}(n^2r^4)$ in the ALS scheme and $\mathcal{O}(n^4r^4)$ in the DMRG.

4. Tricks that make it work.

4.1. Trick 1: Solve large systems iteratively; solve small exactly.

Let us investigate the influence of the residual in a local problem on the "global" accuracy.

From (3.7) it is easy to see that the local system (3.9) can be written as

$$Q_k^\top Ax = Q_k^\top f,$$

where x is the current approximation to the solution in the TT-format, and Q_k is an orthoprojector. Now, suppose the local residual $||Q_k^\top Ax - Q_k^\top f|| = \varepsilon$. Then

$$\varepsilon \leq ||Q_k^\top||\ ||Ax - f||,$$

and, as the norm of a orthoprojector is equal to 1, we have

$$||Ax - f|| \geq \varepsilon.$$

Thus, the global residual cannot be less than the local one. So, local systems are to be solved at least with the same accuracy as expected for the global problem. Despite the fast matrix-by-vector procedure described in the previous section, unpreconditioned iterative solvers may require a lot of iterations to converge to a desired residual. If the TT-ranks of the solution are small, it is better to assemble the full matrix B in (3.10) of size $n^2r^2 \times n^2r^2$ and solve the local system via the direct solver, thus providing the machine precision in the local residual. So, the following solution strategy for the local systems is proposed:

- First, when the TT-ranks are small, assemble the full local matrix and solve the local system directly.
- When the TT-ranks become large (in practice, a criteria of form

  $$n^2r^2 \gtrsim 1000 \qquad (4.1)$$

  is used), run some iterative solver with the fast matrix-by-vector procedure. Usually when this second step starts, the residual is already small enough (e.g., 10^{-2}) to provide sufficiently fast convergence of the iterative solver.

This trick allows us to maintain a good local residual, and hence the rapid convergence of the whole DMRG scheme, without the time-consuming cases (the last steps with large ranks for the direct solution; the first steps with slow convergence for the iterative solver). Nevertheless, preconditioning techniques have to be considered in a future work.

4.2. Trick 2: Truncation based on the residual. The ALS scheme is good, but it is inapplicable as a general-purpose solver, since it requires the knowledge of the solution ranks, and, moreover, even if the TT-ranks are set correctly, the convergence may be quite poor. The DMRG scheme is different. Since the supercore is optimized, the kth TT-rank can be determined adaptively from the SVD of the supercore. However, the approximation is accurate only in the discrete analogue of the L_2-norm. If A corresponds to the discretization of a differential operator, it inevitably has a large condition number, and if $\widetilde{x}$ approximates x in the Frobenius norm well, i.e.,

$$||x - \widetilde{x}|| \leq \varepsilon ||x||,$$

the residue (which is the only available accuracy measure) $||A\widetilde{x} - f||$ can be large. Indeed,

$$||A\widetilde{x} - f|| = ||A(x - \widetilde{x})|| \leq (||A||\ ||x||)\varepsilon \leq (||A||\ ||A^{-1}||\ ||f||)\varepsilon,$$

and thus the relative residual can be bounded only by

$$\frac{||A\widetilde{x} - f||}{||f||} \leq \operatorname{cond}(A)\varepsilon.$$

Thus, the approximation by the SVD in the decimation step of the DMRG,

$$W_k(i_k, i_{k+1}) \approx \widehat{W}_k(i_k, i_{k+1}) = X_k(i_k)X_{k+1}(i_{k+1}),$$

can make the local residue too large. To avoid this, a simple heuristics is proposed. It is based not on the Frobenius norm of the error $||W - \widehat{W}||$ but on the approximation with rank r, which provides the local residue $B\widehat{w} - f$ not worse than the true one up to some constant:

$$||B\widehat{w} - f|| \leq c||Bw - f||.$$

The number $c > 1$ is the parameter of the algorithm. If it is set to 1, no compression is possible, and the ranks will grow. The larger c is, the smaller the ranks, but we need to maintain the accuracy. Numerical experiments show that values of c from 2 to 5 are reasonable (but the optimal value, of course, depends on the particular problem). An optimal low-rank approximation for the solution is a difficult optimization problem, and it is replaced by a simpler constructive procedure: the truncation to rank r is performed by setting singular values starting from $r + 1$ to zero, but the value of the rank is chosen by checking all values of r until the local residue is small enough. This requires several additional matrix-by-vector products, but this additional cost is fully compensated by much improved convergence. Now the accuracy parameter in the DMRG procedure influences only the accuracy of the local solves.

4.3. Trick 3: Random restart. There is no guarantee that the proposed algorithm will converge to the local minimum of the functional. Suppose that the situation is as follows: all local residues are reported to be small, but the obtained solution is not an adequate solution to the full system. This can be checked, in principle, by computing the matrix-by-vector product Ax in the TT-format and computing the residue directly. However, a cheaper way is the following randomized check. Take a random vector z with fixed TT-ranks (say, 2) and compare the scalar products (Ax, z) and (f, z). If the difference is large, then the convergence is not obtained. In the program implementation, this check can be incorporated into the "restart scheme," which proceeds as follows:

1. Generate a random tensor $\mathbf{Z}$ with cores $Z_k(i_k)$ and TT-ranks r_0 (say, $r_0 = 2$).
2. Add $Z_k(i_k)$ to the basis, i.e., set

$$X_k(i_k) := \begin{pmatrix} X_k(i_k) & 0 \\ 0 & Z_k(i_k) \end{pmatrix}.$$

3. Orthogonalize $X_k(i_k)$ from right to left.
4. Start left-to-right DMRG sweep.

It is not difficult to see that if $Ax \approx f$, then such a modification will give the same local residues. If it is not true and the solution has nonzero components in the direction of the randomly generated tensor $\mathbf{Z}$, then the local system will reflect this and the initial local residue will be not small (and this will serve as an indicator for the restart of the method). Note that the theoretical estimates on this random restart approach should be obtained, and we plan it for future research.

The algorithms described above are available as a part of the TT-Toolbox (http://github.com/oseledets/TT-Toolbox).

5. Inversion via DMRG.

5.1. Reduction to a linear system. The proposed technique can be used not only to solve linear systems in the TT-format but also to compute approximate inverses. Usually, the inversion of a matrix is not performed, since it is much more expensive than the solution of a linear system. For some classes of structured matrices it might not be the case, and the computation of the approximate inverse is "cheap." Such an approximate inverse can be used as a preconditioner. This of course requires that the inverse matrix indeed be approximated by a structured matrix. The computation of approximate inverses via different approaches was performed for Toeplitz-like matrices [37], for computing sparse approximate inverses [7], and for the inversion of low Kronecker-rank matrices [34, 29]. In all these cases, the Newton method for the matrix inversion was used. It has the form

$$X_{k+1} = 2X_k - X_k A X_k, \quad k = 0, \dots,$$

and if $||AX_0 - I|| < 1$, for some matrix norm, it converges to the inverse quadratically. The only problem is that it requires two matrix-by-matrix products in each step. If such multiplication can be implemented quickly for a given class of structured matrices, then the Newton method can often provide good approximate inverses. The disadvantage in the TT-case is that the TT-ranks of the product $X_k A X_k$ can be quite large, and the complexity is polynomial in the solution rank but with a quite large exponent. In this paper we propose an alternative approach, which is much simpler to implement, given an algorithm that solves linear systems. The inverse matrix X to a given matrix A satisfies

$$AX = I. \tag{5.1}$$

Equation (5.1) is in fact a linear system:

$$(A \otimes I)x = \mathrm{vec}(I),$$

where $x = \mathrm{vec}(X)$ is a vectorized form of X. Now suppose that A comes with the TT-structure; i.e., it can be associated with a $2d$-dimensional array with elements

$$A(i_1, \dots, i_d, j_1, \dots, j_d) = A_1(i_1, j_1) \dots A_d(i_d, j_d).$$

For simplicity, assume that all indices involved vary from 1 to 2, i.e., the QTT case. In the multi-index notation, (5.1) is formulated as

$$\sum_{j_1,\ldots,j_d} A(j'_1,\ldots,j'_d,j_1,\ldots,j_d)X(j_1,\ldots,j_d,k_1,\ldots,k_d) = I(j'_1,\ldots,j'_d,k_1,\ldots,k_d).$$

For the solution, the same "matrix ordering" of dimensions is used:

$$X(j_1,k_1,\ldots,j_d,k_d) \approx X_1(j_1,k_1)\ldots X_d(j_d,k_d).$$

The matrix of this large linear system (corresponding to the matrix $A \otimes I$) then depends on the $4d$ indices, $\widehat{A}(j'_1,\ldots,j'_d,k'_1,\ldots,k'_d,j_1,\ldots,j_d,k_1,\ldots,k_d)$. The modes of the solution are ordered as $(j_1,k_1),(j_2,k_2),\ldots,(j_d,k_d)$, and thus the indices for the matrix of the linear system are naturally ordered as $(j'_1,k'_1,j_1,k_1),\ldots,(j'_d,k'_d,j_d,k_d)$, i.e., corresponding to a $16 \times 16 \times \cdots \times 16$ d-dimensional tensor. Its elements are defined as

(5.2)
$$\widehat{A}(j'_1,k'_1,j_1,k_1,\ldots,j'_d,k'_d,j_d,k_d) = A(j'_1,\ldots,j'_d,j_1,\ldots,j_d)I(k'_1,\ldots,k'_d,k_1,\ldots,k_d),$$

where $I(k'_1,\ldots,k'_d,k_1,\ldots,k_d) = \delta(k'_1,k_1)\delta(k'_2,k_2)\ldots\delta(k'_d,k_d)$ corresponds to the identity matrix which has TT-ranks equal to 1. From (5.2) it is simple to find that the cores of $\widehat{A}$ in the required permutation are defined as

$$\widehat{A}_p(j'_p,k'_p,j_p,k_p) = A_p(j_p,j'_p)\delta(k_p,k'_p);$$

i.e., the TT-ranks of the $\widehat{A}$ are not larger than the TT-ranks of A. The only difference is that the mode size of $\widehat{A}$ is four times larger. Thus, the TT-Solve algorithm can be applied to the solution of the linear system with the matrix $\widehat{A}$ directly. Moreover, the right-hand side here is of a very simple structure. It has TT-ranks equal to 1. The complexity is quadratic in the mode size, and thus the constant is only four times larger. The dependence in the grid size is the same; the crucial parameter that influences the complexity is the solution rank, which is always larger for the inverse than for the particular solution. However, it is important to note that when computing an approximate inverse, high accuracy is not often needed: even an approximation with small ranks may lead to an excellent preconditioner. Thus, the linear system approach for the inversion looks promising.

The approach described above requires a small modification to make it more robust. Suppose that X is an approximate solution of $AX \approx I$, i.e., $||AX-I||$ is small. It does not mean that the right residue $||XA-I||$ is small, even for a symmetric A, which means that the approximate inverse for a symmetric matrix, computed by such a method, can be nonsymmetric. It is very simple to avoid this. Instead of solving (5.1), let us solve

$$AX + XA = 2I.$$

This is the Lyapunov equation with the symmetric right-hand side, being known to have a symmetric solution, and the TT-ranks for the corresponding matrix are only two times larger than that for the initial one. The solution obtained is now much better: the right and left residue are of the same order.

6. Numerical experiments.

6.1. Data and benchmarks. All computations were performed using the TT-Toolbox (available at http://github.com/oseledets/TT-Toolbox). The TT-Toolbox contains an object-oriented implementation of the TT-format and operations with tensors in such a format. Our solver is included in the TT-Toolbox 2.1 (procedure `dmrg_solve2`). A minimal input is the matrix A in the TT-format (i.e., `tt_matrix` object), the right-hand side which is also in the TT-format (i.e., `tt_tensor` object) and the required relative accuracy ε. There are also several tuning parameters. All results of the computations in all considered examples can be found at http://spring.inm.ras.ru/osel in the section "Benchmarks and Data." They are provided as .mat files with the matrix A, the right-hand side, and the approximate solution all in the TT-format. We hope that these data may be used as benchmarks in future research.

6.2. Solution of linear systems; low-dimensional problems.

6.2.1. Small three-dimensional example. Our solver is intended for the solution of high-dimensional problems. However, it may be useful even for some "small"-dimensional problems as well. We have chosen the following model problem:

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u_{\partial\Omega} &= 0, \end{aligned}$$

where Ω is a three-dimensional L-shaped domain; see Figure 6.1.

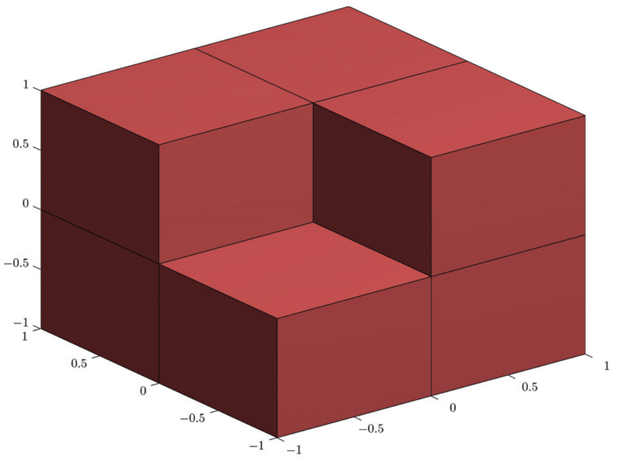

FIG. 6.1. *Three-dimensional L-shaped domain.*

The problem is discretized as follows: the domain is split on seven cubes of size one, intersecting by faces (as in Figure 6.1), and in each cube, the finite difference scheme with $2^L \times 2^L \times 2^L$ points is applied. The solution is thus a four-dimensional tensor with mode sizes $2^L \times 2^L \times 2^L \times 7$. For the first three "physical" modes, the QTT-format is applied.

The stiffness matrix consists of blocks on the diagonal, corresponding to the inner points of each subdomain, and the off-diagonal blocks, representing the interfaces between cubes. The former part is just a Kronecker product of the three-dimensional finite difference Laplacian and the identity matrix of size seven, i.e., $\Delta_3 \otimes I$. The latter

Table 6.1
Comparison of TT-Solve and AGMG methods for the three-dimensional L-shaped problem.

L	CPU time (TT-Solve)	CPU time (AGMG)	$\frac{\|\|u_L-u^*\|\|_F}{\|\|u^*\|\|_F}$	h^2	erank(u_L)
6	24.754078	6.86384	1.7111e-06	2.4336e-04	46.7176
7	55.770516	61.594702	2.3069e-06	6.0840e-05	59.2265
8	**133.381089**	592.626734	3.5154e-06	1.5210e-05	73.3350
9	**356.307145**	$\sim$ 5300	-	4.0000e-06	88.4789

also has the Kronecker rank-1 form $\left(-e_1 e_{2^L}^\top\right) \otimes I \otimes I \otimes \left(e_i e_j^\top\right)$, where $i, j = 1, \dots, 7$. Thus, the matrix possesses a good tensor structure.

As for the full linear solver for comparison, we use the aggregation-based algebraic multigrid (AGMG) algorithm [28] from http://homepages.ulb.ac.be/∼ynotay/AGMG/.

In Table 6.1 we present the CPU times in seconds for our TT-Solve and the AGMG methods to compute the solution with the residual tolerance 10^{-4}, the Frobenius-norm distance between the TT solution (u_L), and the full AGMG solution, providing the residual 10^{-10} (u^*), the grid approximation accuracy h^2, and the average rank of the TT solution versus the grid level L. We see that the TT-Solve method overcomes the full solver on the grid level $L = 7$. As for $L = 9$, we were not able to compute the full solution due to memory limitations, but the extrapolation gives the estimate 5300 seconds, which is 15 times larger than for the TT-Solve. Also, the residual tolerance 10^{-4} is chosen so that the Frobenius (L_2) error is not larger than the grid approximation error h^2.

Remark. It is interesting to compare our approach to other approaches (adaptive grids, hp-methods, adaptive wavelet schemes) in terms of complexity. This will be a subject of future work. However, if our assumptions are valid (matrix, solution, and right-hand side can be approximated in the TT-format), our solver is a black-box one; i.e., it does not require any additional information from the problem, such as high-order discretization and adaptive grids.

6.3. Solution of linear systems; multidimensional elliptic problems.

6.3.1. Eight-dimensional reaction-diffusion example (Table 6.2; Figures 6.2 and 6.3). Now the algorithm is tested on a model high-dimensional problem. Consider an eight-dimensional reaction-diffusion equation

$$(-\Delta + 100 \exp(-r^2))u = 1, \quad r^2 = \sum_{i=1}^{8} (x_i - 0.5)^2$$

with a positive definite operator. 256 grid nodes are taken in each direction, so the Q-dimension is $L = 8$ and the full dimension is $d = 64$. The residual tolerance was set to $\varepsilon = 10^{-6}$.

The matrices of form $-\Delta + V$ appear in the quantum chemistry in the solution of the molecular Schrödinger equation (see, for example, [24]).

Table 6.2

Sweep	Relative residual	Maximal QTT rank	Time of the current sweep, sec.
1	4.090e+00	4	0.173487
5	1.842e-02	42	32.8344
10	1.064e-05	63	28.3208
15	8.551e-06	62	27.0177

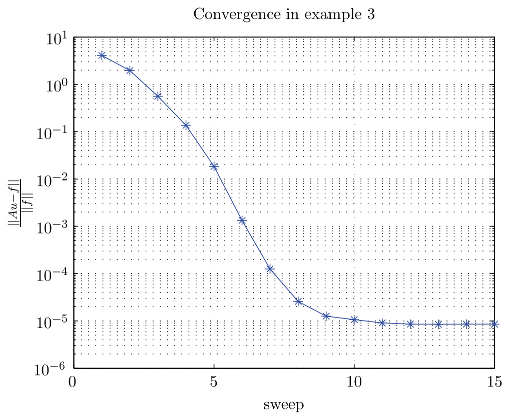

Fig. 6.2. *Relative residual versus the sweep number, eight-dimensional example.*

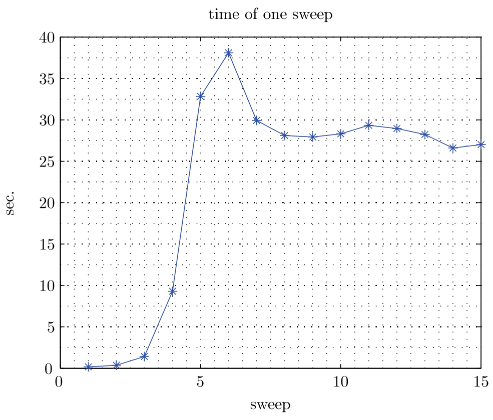

Fig. 6.3. *Time of one sweep versus the sweep number, eight-dimensional example.*

Table 6.3

Sweep	Relative residual	Maximal QTT rank	Time of the current sweep, sec.
1	1.399e+00	4	0.754949
5	1.410e-01	25	39.0297
10	3.973e-04	38	208.696
15	6.403e-05	39	166.427

6.3.2. 64-dimensional reaction-diffusion example (Table 6.3). Consider the following essentially high-dimensional problem: a 64-dimensional reaction-diffusion problem

$$(-\Delta + 100\exp(-r^2))u = 1, \quad r^2 = \sum_{i=1}^{64}(x_i - 0.5)^2,$$

discretized on a grid with 256 points in each direction, thus providing a 512-dimensional tensor as a solution. Here, 15 sweeps and 1845 seconds are required to achieve an $\mathcal{O}(10^{-5})$ accuracy. Notice that the full representation of such tensor requires $2^{512} = 10^{154}$ data points.

6.3.3. Parametric example (Table 6.4; Figures 6.4 and 6.5). In this example we solve a parameter-dependent two-dimensional Poisson equation with parametric boundary conditions:

$$\begin{aligned} -\Delta u(\mathbf{x}, \alpha) = f = 1 \quad \text{in } \Omega = [0,1]^2, \\ \mathbf{x} = [x_1, x_2], \quad \alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4], \\ \alpha_1 \in [0.5, 1], \quad \alpha_2, \dots, \alpha_4 \in [0,1], \\ \alpha_1 u + (1-\alpha_1)\partial u/\partial n = 0 \text{ at } x_1 = 0, \\ \alpha_2 u + (1-\alpha_2)\partial u/\partial n = 0 \text{ at } x_2 = 0, \\ \alpha_3 u + (1-\alpha_3)\partial u/\partial n = 0 \text{ at } x_1 = 1, \\ \alpha_4 u + (1-\alpha_4)\partial u/\partial n = 0 \text{ at } x_2 = 1. \end{aligned}$$

Hence we have a six-dimensional problem. We discretize it using a uniform grid with 256 points in both physical and parametric spaces, so the solution is a 48-dimensional

Table 6.4

Sweep	Relative residual	Maximal QTT rank	Time of the current sweep, sec.
1	1.006e+01	4	0.180521
5	2.181e+00	36	8.34796
9	1.082e-04	100	40.3669
10	2.236e-05	94	37.6331

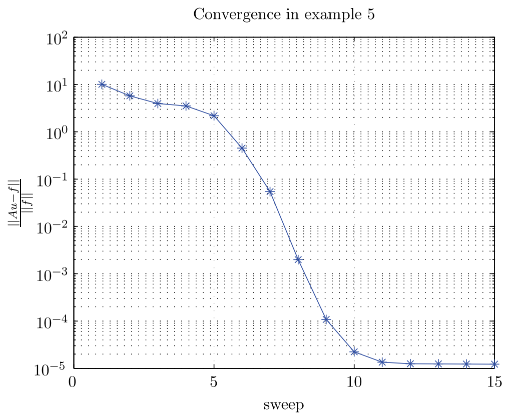

Fig. 6.4. *Relative residual versus the sweep number, parametric example.*

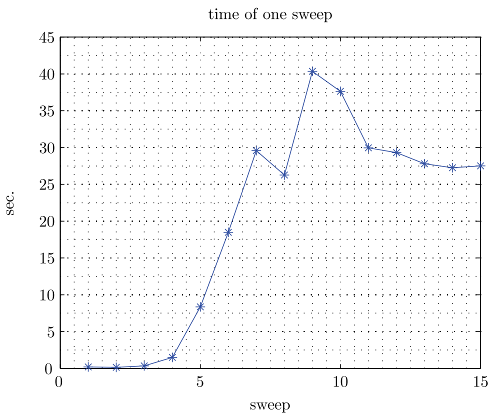

Fig. 6.5. *Time of one sweep versus the sweep number, parametric example.*

tensor. The residual tolerance is $\varepsilon = 10^{-6}$. For the comparison, the QTT-ranks of a solution of a “single” two-dimensional Poisson equation with the Dirichlet boundary conditions are about 20. It is interesting that for such a parametrized boundary condition the ranks are only four times larger. Another interesting feature of this example is that the approximate solution has maximal ranks in the middle sweeps, but then they stabilize with the convergence of the method.

6.4. Matrix inversion.

6.4.1. Heat equation (Table 6.5; Figures 6.6 and 6.7). In this last example, the application of the TT-Solve to the approximate inversion is presented. One of the most promising applications for the approximate inversion is the solution of the parabolic problems via implicit schemes. For the simplest implicit Euler scheme, applied to the heat equation, one has to invert a matrix of form $A = I - \alpha\Delta$, $\alpha = 10^{-3}$. The symmetrizing Lyapunov equation $AX + XA = 2I$ is solved. A two-dimensional problem is considered, discretized on a 1024×1024 grid so that the solution is a 20-dimensional tensor with mode sizes 4. The residual tolerance is set to $\varepsilon = 10^{-4}$. As an initial guess, the identity matrix was used to ensure the symmetry of the inverse matrix. After the solution, we checked the matrix related properties.

Table 6.5

Sweep	Relative residual	Maximal QTT rank	Time of the current sweep, sec.
1	3.849e-01	4	0.027729
5	5.062e-01	5	0.25945
10	7.574e-02	41	4.86231
15	3.379e-04	49	9.63131

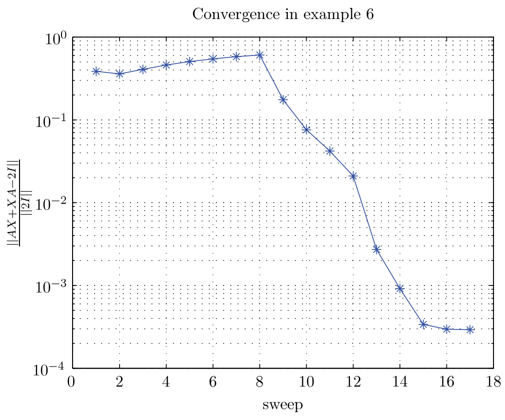

FIG. 6.6. *Relative residual versus the sweep number, matrix inversion example.*

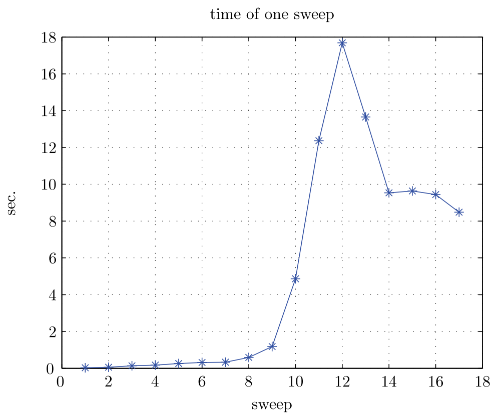

FIG. 6.7. *Time of one sweep versus the sweep number, matrix inversion example.*

The error in symmetry is $\frac{\|X-X^\top\|}{\|X\|} = 2.8872\text{e-}06$, even lower than ε. The right residual is $\frac{\|AX-I\|}{\|I\|} = 2.7156\text{e-}04$ and the left residual is $\frac{\|XA-I\|}{\|I\|} = 2.7365\text{e-}04$. Since we started from the identity matrix, the method required more "warming" iterations in comparison with a random initial guess. Fortunately these additional iterations are significantly cheaper than the "main" iterations, and we can admit them in order to obtain a symmetric matrix with a high accuracy.

7. Conclusion and future work.

In this paper we described how to solve linear systems in the TT-format directly, using the DMRG approach. The idea with preliminary experiments was already described in the paper [21]; however, an efficient practical implementation requires several "tricks." The proposed solver, as confirmed by numerical experiments, is able to solve problems in a black-box fashion (i.e., only the matrix in the TT-format is provided) with 2^d unknowns, where d is of order several hundreds, and, even more important, with TT-ranks up to hundreds even when using a prototype MATLAB implementation.

The main trick is to truncate the local solution on the basis of the local residue, not on the relative Frobenius-norm. Our method can handle nonconstant coefficients, where approaches based on sinc-type quadratures cannot be used. In principle, the TT-Solve can be used for solving any linear system if the matrix is provided in the TT-format. It can be generalized effortlessly to compute approximate inverses to serve as preconditioners. To compute such preconditioners to higher accuracy, we propose to solve the Sylvester equation by the TT-Solve directly. With small mode sizes (for QTT they are usually equal to 2) the inversion is as cheap as the solution itself, making the TT-Solve a very powerful approach for computing preconditioners for matrices corresponding to the discretizations of operators on structured tensor grids and even for computing the Green functions of certain operators.

There is no theory available yet for the convergence of the DMRG-type methods. In many cases the numerical experiments confirm that the DMRG-type methods converge in a number of sweeps independent of the grid size. This experimental fact deserves to be investigated and proved. From the computational point of view, the most time-consuming part is the solution of local linear systems, which may require a lot of iterations (the single matrix-by-vector product is cheap). A certain preconditioner for the local problem is required, and we will continue to test different approaches which can be used for the preconditioning of the local systems.

We plan to apply this linear system solver to certain high-dimensional problems, for example, implicit schemes for parabolic high-dimensional problems, and in particular, to the Fokker–Planck equation. This will be described in future work.

Acknowledgments. The authors thank the anonymous referees for their useful comments which helped to improve the paper.

REFERENCES

[1] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings, *A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids*, J. Non-Newtonian Fluid Mech., 139 (2006), pp. 153–176.

[2] B. W. Bader and T. G. Kolda, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30 (2007), pp. 205–231.

[3] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Nat. Acad. Sci. USA, 99 (2002), pp. 10246–10251.

[4] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[5] D. Braess and W. Hackbusch, *On the efficient computation of high-dimensional integrals and the approximation by exponential sums*, in Multiscale, Nonlinear and Adaptive Approximation, R. A. DeVore, ed., Springer, Berlin, 2009, pp. 39–74.

[6] J. D. Caroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[7] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[8] L. de Lathauwer, B. de Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[9] L. de Lathauwer, B. de Moor, and J. Vandewalle, *On the best rank-*1 *and rank-*$(R_1, R_2, \ldots, R_N)$ *approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[10] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.

[11] S. Dolgov, B. N. Khoromskij, and I. V. Oseledets, *Fast Solution of Multi-Dimensional Parabolic Problems in the TT/QTT–Format with Initial Application to the Fokker-Planck Equation*, Preprint 80, MPI MIS, Leipzig, 2011.

[12] M. Espig and W. Hackbusch, *A Regularized Newton Method for the Efficient Approximation of Tensors Represented in the Canonical Format*, Preprint 78, MPI MIS, Leipzig, 2010.

[13] L. E. Figueroa and E. Süli, *Greedy Approximation of High-Dimensional Ornstein-Uhlenbeck Operators*, preprint, arXiv:1103.0726v2 [math.NA], 2012.

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

[15] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.

[16] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators.* I. *Separable approximation of multi-variate functions*, Computing, 76 (2006), pp. 177–202.

[17] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators.* II. *HKT representation of certain operators*, Computing, 76 (2006), pp. 203–225.

[18] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[19] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[20] S. Holtz, T. Rohwedder, and R. Schneider, *On manifolds of tensors of fixed TT-rank*, Numer. Math., 120 (2012), pp. 701–731.

[21] 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.

[22] B. N. Khoromskij, *$\mathcal{O}(d \log n)$-quantics approximation of N-d tensors in high-dimensional numerical modeling*, Constr. Appr., 34 (2011), pp. 257–280.

[23] B. N. Khoromskij, V. Khoromskaia, and H.-J. Flad, *Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format*, SIAM J. Sci. Comput., 33 (2011), pp. 45–65.

[24] B. N. Khoromskij and I. V. Oseledets, *DMRG+QTT Approach to Computation of the Ground State for the Molecular Schrödinger Operator*, Preprint 69, MPI MIS, Leipzig, 2010.

[25] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[26] C. Lubich, *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*, EMS, Zürich, 2008.

[27] H.-D. Meyer, F. Gatti, and G. A. Worth, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, Wiley-VCH, Weinheim, 2009.

[28] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.

[29] V. Olshevsky, I. V. Oseledets, and E. E. Tyrtyshnikov, *Superfast inversion of two-level Toeplitz matrices using Newton iteration and tensor-displacement structure*, in Recent Advances in Matrix and Operator Theory, Oper. Theory Adv. Appl. 179, Birkhäuser, Basel, 2008, pp. 229–240.

[30] I. V. Oseledets, *Compact Matrix Form of the d-Dimensional Tensor Decomposition*, Preprint 2009-01, INM RAS, Moscow, 2009.

[31] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.

[32] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[33] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 939–956.

[34] I. V. Oseledets and E. E. Tyrtyshnikov, *Approximate inversion of matrices in the process of solving a hypersingular integral equation*, Comput. Math. Math. Phys., 45 (2005), pp. 302–313.

[35] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.

[36] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett., 75 (1995), pp. 3537–3540.

[37] V. Y. Pan, Y. Rami, and X. Wang, *Structure matrices and Newton's iteration: Unified approach*, Linear Algebra Appl., 343/344 (2002), pp. 232–265.

[38] I. H. Sloan and H. Wozniakowski, *When are quasi-Monte Carlo algorithms efficient for high dimensional integrals*? J. Complexity, 14 (1998), pp. 1–33.

[39] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[40] C. F. van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Linear Algebra for Large Scale and Real-Time Applications (Leuven, 1992), NATO Adv. Sci. Inst. Ser. E Appl. Sci. 232, Kluwer, Dordrecht, 1993, pp. 293–314.

[41] X. Wang and I. H. Sloan, *Why are high-dimensional finance problems often of low effective dimension?*, SIAM J. Sci. Comput., 27 (2006), pp. 159–183.

[42] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.