Computational Methods in Applied Mathematics
Vol. 11 (2011), No. 3, pp. 382–393

DMRG Approach to Fast Linear Algebra in the TT-Format

Ivan Oseledets

Abstract — In this paper, the concept of the DMRG minimization scheme is extended to several important operations in the TT-format, like the matrix-by-vector product and the conversion from the canonical format to the TT-format. Fast algorithms are implemented and a stabilization scheme based on randomization is proposed. The comparison with the direct method is performed on a sequence of matrices and vectors coming as approximate solutions of linear systems in the TT-format. A generated example is provided to show that randomization is really needed in some cases. The matrices and vectors used are available from the author or at http://spring.inm.ras.ru/osel

2010 Mathematical subject classification: 15A23; 15A69; 65F99.

Keywords: Tensors; high-dimensional problem; SVD; TT-format.

1. Introduction

Tensors arise naturally in high-dimensional problems, for example, in quantum chemistry [1, 2], financial mathematics [3, 4], and in 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, and so does the complexity of working with fully populated tensors. Thus, in order to solve high-dimensional problems (where even the storage of the full array is prohibitive), one has to use certain low-parametric representations, or *formats*. These formats are related to the so-called tensor decompositions. Tensor decompositions in multilinear algebra and numerical analysis were originally considered in the data analysis, in chemometrics and physometrics communities, and decompositions were related to certain features of the datasets. Since the dataset has to be stored, this restricts the application of many approaches to problems of small sizes. The review [5] contains a lot of information and references on this aspect of tensor decompositions. The first successful attempt to use tensor decompositions (separated representation, or *canonical decomposition*) was made by Beylkin and Mohlenkamp [6, 7], where they showed how to perform basic operations with tensors in such format. The tensor $\mathbf{A}$ is said to be in the canonical format if it is represented as

$$A(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \ldots U_d(i_d, \alpha).$$

Part of this work was supported by RFBR grants 09-01-12058, 10-01-00757, 11-01-00549, RFBR/DFG grant 09-01-91332, Russian Federation Gov. contracts No. П1178, П1112 and П940, 14.740.11.0345 Part of this work was done by the author during his stay at the Max-Plank Institute for Mathematics in Sciences, Leipzig, Germany and Hausdorff Institute of Mathematics, Bonn, Germany

Ivan Oseledets
Institute of Numerical Mathematics, Russian Academy of Sciences, 8 Gubkin Street, Moscow, Russia
E-mail: ivan.oseledets@gmail.com, http://spring.inm.ras.ru/osel

The number r is called *the canonical rank*, and $n_k \times r$ matrices U_k are called *canonical factors*.

The canonical format is simple, and if r is small it has a small number of parameters. However, after each operation (such as addition, matrix-by-vector product, elementwise product) the ranks increase, and one has to reapproximate the result back to a tensor of a smaller rank. Such an approximation problem can be ill-posed for $d \geqslant 3$, and, moreover, there are no algorithms that are guaranteed to compute the approximation even if it is known a priori that it exists. Usually, approaches like alternating least squares (ALS) [8, 9] or special minimization methods [10] are used, but none of them is absolutely robust. Thus, some other formats may be helpful.

In this paper, the tensor train format, or, in short, the *TT-format* is utilized. It can be represented in a simple form and is closed under basic linear algebra operations. Moreover, there exists a stable *rounding procedure* that approximates a given tensor in the TT-format with a prescribed accuracy, using singular value decompositions (SVDs) of certain small auxiliary matrices. This format is stable and the best approximation always exists [11]. For more details on the tensor rounding and basic linear algebra operations, see [12, 13]. A tensor $\mathbf{A}$ is said to be in the TT-format if its elements are defined by the formula

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

where $G_k(i_k)$ is an $r_{k-1} \times r_k$ matrix for each fixed $i_k, 1 \leqslant i_k \leqslant n_k$. To make the matrix-by-matrix product in (1.1) a 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 $r_k \leqslant r, n_k \leqslant n$, then the storage of the TT-representation requires $\leqslant dnr^2$ memory cells. If r is small, then the storage is much smaller than the storage of the full array, n^d. In terms of the TT-ranks (if $r_k \approx r$) , the storage of an array in the TT-format is $\mathcal{O}(dnr^2)$ and the complexity of tensor rounding is $\mathcal{O}(dnr^3)$ operations, i.e., it is linear in the dimension. The TT-format has been known in other areas as the MPS (matrix product states) representation for quite a long time [14, 15]. However, its importance in numerical analysis and scientific computing together with several important mathematical properties was realized only recently.

We are interested in computing certain basic linear algebra operations in the TT-format, with a special focus on the matrix-by-vector product. Here comes the main problem. If the results of [12, 13] are used, then the TT-ranks of the matrix-by-vector product are the product of those for the matrix and for the vector. Suppose they are approximately equal: $R_k \approx r_k \approx r$. Then the product has TT-ranks r^2. The complexity of rounding is then $\mathcal{O}(dnr^6)$. This complexity becomes prohibitive for $r \sim 30$. There are typical cases where TT-ranks are of the order of hundreds, for example in the solution of stochastic PDEs [16] and in the solution of high-dimensional eigenvalue problems [17]. In the general case, the exponent cannot be reduced. However, we assume that the ranks of the product are also close to r. Can we reduce the number of arithmetic operations to compute the product *approximately*? The answer is definitely yes, and it is possible to reduce the complexity to $\mathcal{O}(dnr^4)$. Note, that analogous results were first obtained in the series of papers by D.V. Savostyanov et. al [18, 19, 20, 21, 22] and in the works of V. Khoromskaia and B. Khoromskij [23, 24, 25, 26, 27] for the Tucker format, which also has nice stability properties, but cannot be used in high dimensions due to the exponential dependence on the number of dimensions. However, in this paper another approach employing special nice properties of the TT-format is used. The algorithms are based on the so-called *DMRG* (Density Matrix Renormalization Group) scheme, which was proposed in the solid state physics by White [15] for the solution

of eigenvalue problems and has only recently attracted the attention of the mathematical community [28, 29, 17]. The DMRG was found to be a special Block-Gauss-Seidel scheme for the minimization in the TT-format, which can be applied to different problems that can be formulated as minimization problems. Moreover, its convergence properties were found experimentally to be much better than for the standard ALS-type approaches. It is especially effective for small mode sizes, since it requires handling of vectors of size n^2. Quite surprisingly, such tensors appear routinely in the solution of high-dimensional problems with the help of *quantization* (more generally, tensorization) of the solution. The QTT-format (Quantized TT-format) is defined in the simplest case as follows. Consider a univariate function f on a uniform grid with $n = 2^d$ points. Then the vector of values of this function on this grid can be considered as a $2 \times 2 \times \ldots \times 2$ d-dimensional tensor. Its TT-decomposition gives the QTT-representation of f. This concept is naturally generalized to high-order problems (see [30, 31, 32, 33, 31, 16, 34]). Thus, all of our numerical examples are for the operations in the QTT-format, but the algorithms can be used for arbitrary mode sizes. For large mode sizes, special modifications are possible, but they are not discussed here.

We focus on the approximate matrix-by-vector product, where the matrix and the vector are in the TT-format, and the product is also sought in the TT-format. This problem can be formulated as a minimization problem

$$||y - Ax|| \to \min \tag{1.2}$$

for $y \in \mathcal{S}$ where $\mathcal{S}$ is a certain class of structured solutions. In our case, y is associated with a tensor $Y(i_1, \ldots, i_d)$ with small TT-ranks. Then, (1.2) becomes a nonquadratic minimization problem, which has to be solved by a certain minimization method. In fact, any $\widehat{y}$ that satisfies

$$||\widehat{y} - Ax|| \leqslant \varepsilon ||Ax||,$$

is good, but among these the solution with the smallest ranks is required. Why (1.2) can be solved faster than the direct product combined with the tensor rounding? To see that, it is sufficient to reformulate (1.2) as an equivalent minimization problem

$$||y||^2 - 2(y, Ax) \to \min\,. \tag{1.3}$$

It appears that if TT-ranks of A, y, x are bounded by r, then the scalar product (Ax, y) can be computed exactly in $\mathcal{O}(dnr^4)$ operations. Thus, the functional to be minimized can be computed cheaply, and it is an indicator that a fast method is possible.

The simplest method that makes use of the TT-structure is ALS-type methods. If all cores except one are fixed, then it is easy to compute the remaining one. The value of the functional will not increase at 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 non-adaptive in the sense that it requires all TT-ranks to be known in advance. To avoid this problem, the DMRG method is used. The DMRG method is an alternating least square method, but with one minor modification. Instead of minimizing over one core, the functional is minimized over a pair of cores $G_k(i_k), G_{k+1}(i_{k+1})$. The resulting problem still has to be solved. However, if a *supercore* W is introduced by contracting over

α_k:

$$W(i_k, i_{k+1}) = G_k(i_k)G_{k+1}(i_{k+1}), \tag{1.4}$$

then this supercore can be found in a cheap way. The factors G_k, G_{k+1} are then recovered by the SVD or any suitable rank-revealing decomposition. The algorithm then proceeds with the second pair of cores and so on.

The convergence of the DMRG algorithm (compute the supercore, decompose it and go further) is typically very fast, and only a few sweeps (a sweep is the sequence of iterations where all pairs $(k, k+1)$ were optimized) are required. However, there are examples, where it converges not to the right solution. The problem is that the functional (1.3) gives the same stationary points as the functional (1.2), but the residual $||y - Ax||$ cannot be computed fast. Thus, certain random checking is required. Randomization was first used in [29] in the context of linear system solutions. Here we propose a much simpler randomization scheme that is incorporated into the DMRG scheme directly.

2. Introduction to the notation used

In this section, several basic facts and definitions are recollected. When dealing with the TT-representations, it is convenient to work with parameter-dependent matrices, which are denoted by, for example, $G_k(i_k)$. The size of these matrices should be clear from the context (i.e, $r_{k-1} \times r_k$). The parameter-dependent matrices can be multiplied:

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

yielding a new matrix depending on a larger number of parameters. In this representation, it is important to look at the order of the elements of the product. The row and column indices of a parameter-dependent matrix can be considered as parameters. For example, for an $r_{k-1} \times r_k$ parameter-dependent matrix $G_k(i_k)$ its elements (being scalars) are denoted by $G_k(\alpha_{k-1}, i_k, \alpha_k)$. Again, the actual size of the parameter-dependent matrix is defined by the context. For example, if $G_k(i_k)$ is an $r_{k-1} \times r_k$ matrix, then $G_k(\alpha_{k-1}, i_k)$ defines a row-vector of length r_k for each fixed α_{k-1} and i_k. In this form, the TT-format is represented as

The TT-representation (1.1) is non-unique, since it is invariant under the 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. A core $G_k(i_k)$ is said to be left-orthogonal if

$$\sum_{i_k} G_k(i_k)G_k^\top(i_k) = I_{r_{k-1}},$$

and right-orthogonal if

$$\sum_{i_k} G_k^\top(i_k)G_k(i_k) = I_{r_k}.$$

Another interpretation of the left-orthogonality is that the tensor $G_k(\alpha_{k-1}, i_k, \alpha_k)$, considered as a matrix of size $(r_{k-1}n_k) \times r_k$, has orthonormal columns. The right-orthogonality of the core means that G_k, considered as a matrix of size $r_{k-1} \times (n_k r_k)$ has orthonormal rows. The

cores $G_k(i_k), \dots G_p(i_p)$, can be made left-orthogonal by equivalent transformations. Indeed, $G_k(i_k)$ can be written as

$$G_k(i_k) = Q_k(i_k)R,$$

where $Q_k(i_k)$ is left-orthogonal by applying the QR-decomposition to a $(r_{k-1}n_k) \times r_k$ matrix obtained from $G_k(i_k)$ by reshaping. The corresponding matrix R is constant and can be incorporated into the next core. Good news is that if the cores $G_1(i_1) \dots G_k(i_k)$ are left-orthogonal, then their product

$$Q(i_1, \dots, i_k) = G_1(i_1) \dots G_k(i_k),$$

considered as an $N_k \times r_k$ matrix has orthonormal columns ($N_k = \prod_{s=1}^{k} n_s$). The proof can be found in [13].

3. DMRG algorithm for the matrix-by-vector product

3.1. Basic idea

In this paper, the DMRG algorithm is considered for the approximation of the matrix-by-vector product $y = Ax$ with both A and x in the TT-format. This is just a minimization of the functional

$$z = \arg\min_{z \in \mathcal{S}} ||y - z|| \tag{3.1}$$

over all tensors z with "small" TT-ranks. The tensor y in this scheme is given *implicitly*. Thus, first the equations are derived for some TT-tensor y with large ranks, and they are then simplified for a special case.

Suppose z is in the TT-format with cores Z_k. Then the DMRG algorithm is obtained by minimizing (3.1) over a supercore $W(i_k, i_{k+1}) = G_k(i_k)G_{k+1}(i_{k+1})$. It is not difficult to see, that this step is equivalent to an ALS step applied to a $(d-1)$-dimensional tensor of size $n_1 \times \dots \times n_{k-1} \times (n_k n_{k+1}) \times \dots \times n_d$, i.e., with modes $k, (k+1)$ treated as one long mode of size $n_k n_{k+1}$. Thus, it is sufficient to derive a formula for one iteration step of the ALS method (and use it for the modified tensor).

Suppose that the kth core is varied (and all the others are fixed). This means that the vector y is then restricted to a subspace

$$z(i_k) = Qw(i_k), \tag{3.2}$$

where $y(i_k)$ is a vector of length N_k, $N_k = \prod_{s=1, s\neq k} n_s$, and $w(i_k)$ is a vector of length $r_{k-1}r_k$ (just a part of the new core $Y_k(i_k)$) The matrix Q is an $N_k \times r_{k-1}r_k$ matrix. By equivalent transformations of the TT-representation of $\mathbf{X}$ the matrix Q can be made orthogonal. Indeed, it can be treated as a tensor with elements

$$Q(i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_d, \alpha_{k-1}, \alpha_k) = G_1(i_1) \dots G_{k-1}(i_k, \alpha_{k-1})G_{k+1}(\alpha_{k+1}, i_{k+1}) \dots G_d(i_d).$$

If the cores $G_s(i_s), s = 1, \dots, k-1$ are orthogonalized from the left, and the cores $G_s(i_s), s = k+1, \dots, d$ are orthogonalized from the right, then the matrix Q will have orthonormal columns.

Under the restriction (3.2) the minimization of the functional

$$\sum_{i_k} ||y(i_k) - Qw(i_k)||$$

leads to a simple solution

$$w(i_k) = Q^\top y(i_k). \tag{3.3}$$

The vectors $w(i_k)$ form a new core Z_k. Substituting the TT-representations for Q and y, we obtain a simple formula

$$Z_k(i_k) = \Psi_k Y_k(i_k) \Phi_k, \tag{3.4}$$

where Ψ_{k-1} is an $r_{k-1} \times R_{k-1}$ matrix, and Φ_k is an $R_k \times r_k$ matrix (R_k are the TT-ranks of the given representation of y, whereas r_k are the TT-ranks of the current approximation to the solution). Matrices Ψ_k are computed recursively via the simple formula

$$\Psi_k = \sum_{i_{k-1}} Z_{k-1}(i_{k-1}) \Psi_{k-1},$$

with a starting condition $\Psi_0 = 1$. A similar representation holds for the right matrices Φ_k. Using (3.4) is not very useful if the tensor is only given in some TT-representation. The TT-rounding procedure is much simpler, requires only one sweep through all the cores, and is guaranteed to compute the result. If the cores $Y_k(i_k)$ possess an additional structure, then the DMRG/ALS algorithm may become very useful, as it will be shown in the next subsection.

3.2. Compact expression for an ALS step

Now, suppose that $y = Ax$, with both A and x in the TT-format. Then, recall the representation of the matrix-by-vector product in the TT-format [13]. Let

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

be a TT-representation of A, and

$$X(j_1, \ldots, j_d) = X_1(j_1) \ldots X_d(J_d)$$

be a TT-representation of X. Then, the cores Y_k of the product are given as

$$Y_k(i_k) = \sum_{j_k} A_k(i_k, j_k) \otimes X_k(j_k). \tag{3.5}$$

Thus, the evaluation of (3.4) for a core of form (3.5) requires n_k multiplications of a Kronecker rank-1 matrix by a matrix from the left and a matrix from the right. It is easy to implement these operations using standard matrix-by-matrix multiplications, reshape and permute operations. For more details on such a kind of operations, see [29] where the case of the linear system solution is considered in detail. Here we give only the final estimate to avoid technical details. The cost of computing Ψ_k can be estimated as

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

and the total sum in (3.4) can be computed in $\mathcal{O}(nRr^3 + n^2 R^2 r^2)$ operations.

3.3. Randomization

After the supercore $W(i_k, i_{k+1})$ is computed at the step k, it is approximated back

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

This can be done by the truncated SVD of a matrix P of size $(r_{k-1} n_k) \times (n_{k+1} r_{k+1})$. For small mode sizes (for the QTT-format $n_k = 2$):

$$P \approx U V^\top, \tag{3.6}$$

which gives a new (approximate) rank r_k and a new left basis U that turns into a kth core. Note that the $k+1$th core $G_{k+1}(i_{k+1})$ (obtained from V) will be recalculated immediately at the next step of the sweep, so it may not even be stored. However, it is useful to keep it to monitor the convergence of the process. The stabilization of the supercore (compared to the previous approximant) is the only viable stopping criterion in the method. This stopping criterion, however, does not guarantee anything in terms of the global residue. The method, in principle, can get stuck in the local minima. And this fact is not very easy to detect! However, simple randomization can serve as the stabilization of the method. In order to check the appropriate equality $Ax \approx f$ without computing the product Ax directly, we select some random tensor q and compare scalar proucts (Ax, q) and (f, q). We propose a similar but a different approach that fits into the iteration scheme directly. After decomposition (3.6) has been computed, the basis spanned by the columns of U is randomly enlarged by k columns. This number is an heuristic parameter; however, we found out that even $k = 1$ is enough (at least in our examples). The matrix $U_{\mathtt{add}}$ is generated at random, and then its columns are orthogonalized to the columns of U by the Golub-Kahan reorthogonalization. In the end, the new U matrix is (we use the MATLAB notation for block matrices)

$$U := [U, U_{\mathtt{add}}],$$

it still has orthonormal columns, and the new matrix V is just

$$V := [V, N],$$

where N is a zero matrix with k columns. Such an operation, of course, does not change P and the value of the functional. Why do we do that? The explanation is that on the next step of the sweep the projection will also be performed on the space spanned by those randomly added columns, and this is equivalent to performing a randomized check of the current approximant by taking scalar product with a low-rank random tensor. Unfortunately, we have no theoretical justification for this approach. The following Hypothesis may be very helpful to estimate the probability of detecting a "good" approximation in the case where the approximation is far from the true solution.

Hypothesis 3.1. *Suppose Y, Z are d-dimensional tensors such that*

$$||Y - Z||_F > \varepsilon ||Y||_F.$$

Take a tensor Q at random with TT-ranks k and $||Q||_F = 1$. Then, with a high probability

$$|\langle Y - Z, Q \rangle|_F > c_1 \varepsilon ||Z||_F,$$

with a constant c_1.

The point of the hypothesis is as follows. The residue $||Ax - f||$ cannot be computed cheaply. However, one can compute scalar products (Ax, z) and (f, z) cheaply for any low-rank tensor z. The randomization criterion is based on the computation of such scalar products for a certain low-tensor z. The hypothesis postulates that if the residue is large, then, with a high probability for a general low-rank tensor, the scalar products will differ, which is good justification for the convergence of the method.

4. Extension to other TT-approximation problems

The approach proposed in this paper can be extended to other approximate operations in the TT-format, like the elementwise (Hadamard) product and the matrix-by-matrix product. This is in fact a DMRG scheme for the approximation of a given tensor, but the computation of the supercores, i.e., products $Q^\top y(i_k)$, is performed efficiently utilizing the structure. Among the important problems where a DMRG-type scheme is very helpful is the reduction of a given suboptimal canonical representation to the TT-format. It is known that the canonical rank-R tensor is also in the TT-format with ranks $\leqslant R$, just by forming cores G_k as diagonal matrices from the columns of the factors of the canonical decomposition (see [13]). However, this is not a practical approach for the case of large ranks, say thousands: the storage even of a single core with such a rank is large. A simple approach is the add-and-compress algorithm proposed in [31]. Each individual rank-1 component is transformed into the TT-format, and they are added using the TT-arithmetic, and compression is performed after each addition. If, hopefully, during the whole process the TT-ranks stay bounded by r, the complexity is then $\mathcal{O}(dnr^3R)$.

For the DMRG algorithm, the computation of a new core is reduced to

$$W_k(i_k) = \Psi_k Y_k(i_k) \Phi_k, \tag{4.1}$$

where now Ψ_k is an $r_{k-1} \times R$ matrix and Φ_k is an $R \times r_k$ matrix. The core $Y_k(i_k)$ corresponds to the formal conversion of a rank-R tensor to the TT-format:

$$Y_k(i_k) = \mathtt{diag}(\Lambda_k(i_k)),$$

and is of size $R \times R$ (for each fixed i_k). A new core can be then computed in $\mathcal{O}(nRr^2)$ operations, and the total complexity of one sweep is $\mathcal{O}(dnRr^2 + dnr^3)$ (provided that n is small).

5. Numerical experiments

5.1. Comparison with the direct method

Our implementation is included in the TT-Toolbox (for the recent version see http://spring.inm.ras.ru/osel or contact the author) as a subroutine `mvk2`. The minimal input is a matrix A in the TT-format, a vector x in the TT-format, and a required relative accuracy ε. Optionally, the initial guess can be specified, as well as the maximal TT-rank during the iterations and the maximum number of sweeps. As benchmarking matrices, we took examples from [29]. The matrices are available at the webpage http://spring.inm.ras.ru/osel in the Section "Benchmarks and Data". They are provided as .mat files with the matrix A in the TT-format and the right-hand size and the approximate solution in the TT-format. There are six example matrices and vectors there now. Some statistics is given in Table 5.1

Matrix	N	rank(A)	rank(x)
`tt-solve_ex2.1_6-8`	2^{24}	3.7	23.3
`tt-solve_ex2_6-10`	2^{20}	3.7	19.0
`tt-solve_ex3`	2^{64}	5.1	40.5
`tt-solve_ex4`	2^{512}	3.7	27.4
`tt-solve_ex5`	2^{48}	4.6	48.0
`tt-solve_ex6`	2^{20}	5.1	24.8

Table 5.1. Different test matrices

The following simple MATLAB code was used for testing (A stores the matrix, x the vector)

```
eps=1e-6; %Set up the accuracy
tic; x1=a*x; x1=round(x1,eps); toc;
tic; x2=mvk2(a,x,eps,6); toc;
fprintf('Error: %3.2e \n',norm(x2-x1)/norm(x1));
```

The number of DMRG sweeps is limited by 6. It appears that DMRG produces a fairly good approximation; however, it is not always possible to design a good stopping criterion for the method. The timings for different matrices are given in Table 5.2. The accuracy is set to 10^{-6}.

Matrix	a*x	mvk2	Relative error
`tt-solve_ex2.1_6-8`	0.19	0.1	2.39e-06
`tt-solve_ex2_6-10`	0.08	0.06	1.49e-06
`tt-solve_ex3`	4.72	0.40	8.15e-06
`tt-solve_ex4`	6.99	6.19	5.22e-05
`tt-solve_ex5`	5.08	0.28	1.32e-05
`tt-solve_ex6`	1.17	4.67	7.59e-07

Table 5.2. Comparison of timings (in seconds) of direct and mvk matrix-by-vector products

Table 5.2 shows, that usually the `mvk2` method is faster on this type of problems, but there are two cases where it is not: `tt-solve_ex4` and `tt-solve_ex6`. What is the reason? First of all, these problems are rather "hard", since the rank of the matrix is small. `mvk2` is especially suited for the case where both the rank of the matrix and the rank of the vector are big. For example, if in the example `tt-solve_ex6` an artificial computation of $x.^2$ (i.e., elementwise square) is considered, then mvk2 can be applied to the computation of the product $\texttt{diag}(x) * x$. It finishes in approximately 10 seconds, whereas the full computation was not able to finish due to the lack of memory on the used machine (4 Gb). The second reason is that the TT-ranks of the product are not very small. Consider again the worst example in Table 5.2, `tt-solve_ex6`. For an accuracy of 10^{-6} the maximal TT-rank of the product is 165. That is too much and comes from the fact that x has already incorporated some "noise". Indeed, it is a solution of a linear system with the matrix A, and the accuracy of the solution can be computed:

```
>> norm(A*x-rhs)/norm(rhs)
ans = 3.1202e-04
```

It is only 10^{-4}. What happens if the truncation is performed at the right level?

```
>> tic; x1=A*x; x1=round(x1,1e-4); toc;
Elapsed time is 0.655725 seconds.
>> tic; x2=mvk2(A,x,1e-4); toc;
Elapsed time is 0.227166 seconds.
>> norm(x1-x2)/norm(x1)
ans =  1.5845e-04
```

Now the situation is clear: the approximate multiplication should be done only on the right threshold, adapted for the problem. In this case, `mvk2` is significantly faster. Moreover, it uses much less memory, since for the direct method one has to store $dn(Rr)^2$ memory cells, and even for moderate values of R and r (around 20-30) this can be of the order of several gigabytes.

5.2. Effect of randomization

In the end, let us study the effect of randomization. It rarely appears that DMRG without random kicks do not work well. However, in our experiments we were able to find such an example. This matrix came from the stochastic PDE and is contained in the file `big_mat.mat`. It is a 51-dimensional matrix. If we run `mvk2` without random stabilization with a high accuracy parameter:

```
>> load big_mat;  w=mvk2(A,x,1e-9);
```

the program reports a good "local" convergence:

```
swp=10 er=3.39e-10
```

However, the real error is not that small:

```
>> norm(A*x-w)/norm(w)
ans =   5.7768e-04
```

Thus, this is an important example of a local minimum If the kick rank is set to 1, then everything is fine, but the convergence requires many more sweeps:

```
swp=28 er=5.63e-10
```

and

```
>> norm(A*x-w)/norm(w)
ans = 3.6758e-09
```

Thus, to avoid such pathological cases, one has to use the random stabilization all the time, especially for the cases of large ranks.

6. Conclusion and future work

In this paper, a DMRG-based algorithm for the computation of the matrix-by-vector product is proposed. It is compared to the direct method (multiply and compress) on a set of test matrices. These subroutines are used in the TT-Toolbox 2.1 for the computation of matrix-by-vector products, and have already used, for example, in a recent paper [34] to compute the convolutions of two vectors in the QTT-format. From the algorithmic point of view, there are still places where the algorithm can be improved, at least in two parts. The first part is the selection of a cheap initial guess for the method to decrease the number of sweeps. The second part is to make use of the Wedderburn-type techniques [21] to compute approximations by reusing the information from previous sweeps and computing only low-rank updates.

The experimental facts presented in this paper require theoretical investigation. Even the local convergence properties of the DMRG scheme (it is believed to converge fast) are still not clearly understood (however, the recent work [28] is a huge step forward in this direction). The randomization scheme is a separate topic and is related, to our belief, to the geometry of low-rank tensors. It has to be investigated. And the last (but not least) objective for the future work are the applications of the TT-format. They include Fokker-Planck and multiparametric equations, and quantum chemistry, where the TT-format has good perspectives.

References

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

[2] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, — Zurich: EMS, 2008.

[3] I. Sloan and H. Wozniakowski, *When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals*, J. of Complexity, **14** (1998), no. 1, pp. 1–33.

[4] X. Wang and I. Sloan, *Why are high-dimensional finance problems often of low effective dimension?*, SIAM J. Sci. Comp., **27** (2006), no. 1, pp. 159–183.

[5] T. G. Kolda and B. W. Bader, *Tensor Decompositions and Applications*, SIAM Review, **51** (2009), no. 3, pp. 455–500.

[6] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Nat. Acad. Sci. USA, **99** (2002), no. 16, pp. 10246–10251.

[7] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., **26** (2005), no.6, pp. 2133–2159.

[8] 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.

[9] 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.

[10] M. Espig and M. Hackbusch, *A regularized Newton method for the efficient approximation of tensors represented in the canonical format: Preprint 78*, — Leipzig: MPI MIS, 2010, www.mis.mpg.de/preprints/2010/preprint2009_78.pdf.

[11] I. V. Oseledets and E. E. Tyrtyshnikov, *TT-cross algorithm for the approximation of multidimensional arrays*, Linear Algebra Appl., **432** (2010), no. 1., pp. 70–88.

[12] I. V. Oseledets, *Compact matrix form of the d-dimensional tensor decomposition: Preprint 2009-01*, — Moscow: INM RAS, 2009, http://pub.inm.ras.ru.

[13] I. V. Oseledets, *Tensor train decomposition*, SIAM J. Sci. Comp, **33** (2011), no. 5, pp. 2295-2317.

[14] S. Östlund and S. Rommer, *Thermodynamic Limit of Density Matrix Renormalization*, Phys. Rev. Lett., **75** (1995), no. 19, pp. 3537–3540, http://link.aps.org/doi/10.1103/PhysRevLett.75.3537.

[15] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, **48** (1993), no. 14, pp. 10345–10356.

[16] B. N. Khoromskij and I. V. Oseledets, *Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs*, Comp Meth. Appl. Math., **10** (2010), no 4, pp. 376–394.

[17] B. N. Khoromskij and I. V. Oseledets, *DMRG + QTT approach to high-dimensional quantum molecular dynamics: Preprint 68*, — Leipzig: MPI MIS, 2010, www.mis.mpg.de/preprints/2010/preprint2010_68.pdf.

[18] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Linear algebra for tensor problems*, Computing, **85** (2009), no. 3, pp. 169–188.

[19] D. V. Savostyanov and E. E. Tyrtyshnikov, *Approximate multiplication of tensor matrices based on the individual filtering of factors*, J. Comp. Math. Math. Phys., **49** (2009), no. 10, pp. 1662–1677.

[20] D. V. Savostyanov, *Fast revealing of mode ranks of tensor in canonical format*, Numer. Math. Theor. Meth. Appl., **2** (2009), no 4., pp. 439–444.

[21] S. A. Goreinov, I. V. Oseledets, and D. V. Savostyanov, *Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case: Preprint 2010-01*, — Moscow: INM RAS, 2010 — arXiv:1004.1986, http://pub.inm.ras.ru.

[22] D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *Fast truncation of mode ranks for bilinear tensor operations*, Numer. Lin. Alg. Appl., 2011.

[23] V. Khoromskaia, *Computation of the Hartree-Fock exchange by tensor-structured methods*, Comp. Methd. Appl. Math., **10** (2008), no. 2.

[24] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comp., **31** 2009, no. 4., pp. 3002–3026.

[25] S. R. Chinnamsetty, H. J. Flad, V. Khoromskaia, and B. N. Khoromskij, *Tensor decomposition in electronic structure calculations on 3D Cartesian grids*, J. Comp. Phys., **228** (2009), no. 16, pp. 5749–5762.

[26] V. Khoromskaia, *Numerical Solution of the Hartree-Fock Equation by Multilevel Tensor-structured methods: Ph.D. thesis / TU Berlin*, — 2010. http://opus.kobv.de/tuberlin/volltexte/2011/2948/.

[27] B. N. Khoromskij, V. Khoromskaia, and H.-J. Flad, *Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format*, SIAM J. Sci. Comp., **33** (2011), no. 1, pp. 45–65.

[28] T. Rohwedder, S. Holtz, and R. Schneider R, *The alternation least squares scheme for tensor optimisation in the TT-format*: Preprint DFG-Schwerpunktprogramm 1234 71: 2010.

[29] S. V. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*: Preprint 19 (Submitted to SIAM J. of Sci. Comp.), — Leipzig: MIS MPI, 2011, http://www.mis.mpg.de/preprints/2011/preprint2011_19.pdf.

[30] B. N. Khoromskij, $\mathcal{O}(d \log N)$–*Quantics approximation of* N–d *tensors in high-dimensional numerical modeling*, Constr. Appr., 2011.

[31] B. N. Khoromskij and I. V. Oseledets, *QTT-approximation of elliptic solution operators in high dimensions*, Russian J. Numer. Anal. Math. Modelling, (2011), To appear.

[32] I. V. Oseledets, *Approximation of* $2^d \times 2^d$ *matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., **31** (2010), no. 4, pp. 2130–2145.

[33] V. Kazeev and B. N. Khoromskij, *Explicit low-rank QTT representation of Laplace operator and its inverse: Preprint 75*, — Leipzig: MPI MIS, 2010, www.mis.mpg.de/preprints/2010/preprint2010_75.pdf.

[34] V. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Multilevel Toeplitz matrices generated by QTT tensor-structured vectors and convolution with logarithmic complexity: Preprint 36*, — Leipzig: MPI MIS, 2011 http://www.mis.mpg.de/publications/preprints/2011/prepr2011-36.html.