DMRG APPROACH TO FAST LINEAR ALGEBRA IN THE TT-FORMAT

I. V. Oseledets

Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow, Russia

ivan.oseledets@gmail.com, http://spring.inm.ras.ru/osel

Abstract. In this paper the concept of DMRG minimization scheme is extended for important operations in the TT-format, like matrix-by-vector product and conversion from the canonical format to the TT-format. Fast algorithms are implemented in the TT-Toolbox stabilization scheme based on randomization is proposed, and 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. An 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

1. Introduction

Tensors arise naturally in high-dimensional problems, for example, in quantum chemistry [1, 2], financial mathematics [3, 4] and many others. 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 to work with fully populated tensors. Thus, in order to solve high-dimensional problems (where even the storage of the full array is prohibitive) certain low-parametric representations, or *formats*, have to be used. These formats are related to so-called tensor decompositions. Tensor decompositions in multilinear algebra and numerical analysis originally where considered in data analysis, in chemometrics and physometrics communities, and the decompositions where 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 size. A review [5] contains a lot of information and references on this aspect of tensor decompositions. The first successful

[$]This work was supported in part 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 during the stay of author in Max-Plank Institute for Mathematics in Sciences, Leipzig, Germany and Hausdorff Institute of Mathematics, Bonn, Germany

attempt to use tensor decompositions (the separated representation, or *the canonical decomposition*), was made by Beylkin and Mohlenkampf [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

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

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

The canonical format is simple and if r is small has low amount of parameters. However, after each operation (addition, matrix-by-vector product, elementwise product) the ranks are increasing, and one has to reapproximate the result back to a tensor of smaller rank. Such 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 its known apriori 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 simply *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 formula

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

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 $G_k(i_k)$ — 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 this 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 the tensor rounding is $\mathcal{O}(dnr^3)$ operations, i.e. it is linear in the dimension.

We are interested in computing basic linear algebra operations in the TT-format, with an especial 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 result 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 order hundreds. In the general case, the exponent can not be reduced. However, we assume that the ranks of the product are also close to r. Can we reduced 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 and others [14, 15, 16, 17, 18] and in the works of V. Khoromskaia and B. Khoromskij [19, 20, 21, 22, 23] for the Tucker format, that also has nice stability properties, but can not be used in high dimensions due to the exponential dependence on the number of dimensions. However, in this paper another approach is used, which uses special nice properties of the TT-format. The algorithms are based on the so-called *DMRG* (Density Matrix Renormalization Group) scheme, which was proposed in the solid state physics by White [24] for the solution of the eigenvalue problems and only recently attracted the attention of the mathematical community [25, 26, 27]. 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 which can be formulated as minimization problems. Moreover, its convergence properties were experimentally found to be much better, than for 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 routinely appear 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 [28, 29, 30, 31, 29, 32, 33]. 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 sought also in the TT-format. This problem can be formulated as a minimization problem

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

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 smallest ranks is required. Why (1.2) can be solved faster than direct product and rounding? To see that, it is sufficient to reformulate (1.2) as an equivalent minimization problem

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

It appears, that if TT-ranks of A, y, x are bounded by r, 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 which makes use of the TT-structure is are ALS-type methods. If all cores except one are fixed, it is simple 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 methods is an alternating least squares 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:

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

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 convergences 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 $||y - Ax||$ can not be computed fast. Thus, certain random checking is required. Randomization was first used in [26] 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 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 $G_k(i_k), \ldots G_p(i_p)$, 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 are that if the cores $G_1(i_1) \ldots G_k(i_k)$ are left-orthogonal, their product

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

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 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 minimization of a 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 $n_1 \times \ldots \times n_{k-1} \times (n_k n_{k+1}) \times \ldots \times n_d$, i.e. with modes $k, (k+1)$ treated as a 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 k-th core is varied (and all 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 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

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

If the cores $G_s(i_s), s = 1, \ldots, k-1$ are orthogonalized from the left, and the cores $G_s(i_s), s = k+1, \ldots, d$ 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 the new core Z_k. Substituting 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 a simple formula

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

with a starting condition $\Psi_0 = 1$. 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 more simple, requires only one sweep through all the cores, and is guaranteed to compute the result. If the cores $Y_k(i_k)$ possess 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, evaluation of (3.4) for a cores of form (3.5) requires n_k multiplications of an Kronecker rank-1 matrix by a matrix from the left and a matrix from the right. It is simple to implement these operations using standard matrix-by-matrix multiplications, reshape and permute operations. For more details on such kind of operations, see [26] where the case of linear system solution is considered in details. Here we give only the final estimate to avoid technical details. The cost of computing Ψ_k can be estimated as

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

and the full sum in (3.4) can be computed in $\mathcal{O}(nRr^3 + n^2R^2r^2)$ 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 truncated SVD of the matrix P of size $(r_{k-1}n_k) \times (n_{k+1}r_{k+1})$. For small modes sizes (for the QTT-format $n_k = 2$):

$$P \approx UV^\top, \tag{3.6}$$

which gives the new (approximate) rank r_k and the new left basis U, which turns into a k-th core. Note, that the $(k+1)$-th core $G_{k+1}(i_{k+1})$ core (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 criteria in the method. This stopping criteria, however, does not guarantee anything in terms of global residue. The method, in principle, can get stuck in the local minima. And this fact is not very easy to detect! However, a simple randomization serves as a stabilization of the method. An idea to check if

$$y \approx Ax$$

fast is to create some random TT-tensor q with small ranks, and compute scalar products (y, q) and (Ax, q) and compare. We propose another approach. After decomposition (3.6) has been computed, the basis spanned by columns of U is randomly enlarged by some number of columns k. This number is an heuristic parameter, however we found out that even $k = 1$ is enough (at least in our examples). The matrix U_{add} is generated at random, and then its columns are orthogonalized to the columns of U by Golub-Kahan reorthogonalization. In the end, the new U matrix is (we use MATLAB notation for block matrices);

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

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 operation, of course, does not change P and the value of the functional. Why would we do that? The explanation is that on the next step of the sweep the projection will be performed also 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 when the approximation is far from the true solution.

Hypothesis 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 high probability

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

with a constant c_1.

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 elementwise (Hadamard) product and 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 important problems where DMRG-type scheme is very helpful, is the reduction of a given suboptimal canonical representation to the TT-format. It is known, that canonical rank-R tensor is also in the TT-format with ranks $\leqslant R$, just by making cores G_k diagonal matrices from the columns of 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 rank is large. A simple approach is add-and-compress algorithm, proposed in [29]. 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 $\mathcal{O}(dnr^3R)$. The computation of the 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). The 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, an 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 [26]. 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 the 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 therefore limited by 6. It appears, that DMRG produces a quite good approximation, however it is not always possible to design a good stopping criteria for the method. The timings for different matrices are given in the 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

The Table 5.2 shows, that usually mvk2 method is faster on these 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 $\mathtt{diag}(x) * x$. 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 the Table 5.2, `tt-solve_ex6`. For the accuracy 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 at 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 order 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 it is contained in the file big_mat.mat. It is a 51-dimensional matrix. If we run mvk2 without the random stabilization with a high accuracy parameter:

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

the program reports 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 minima! If the kick rank is set to 1, then everything is fine, but the convergence requires much 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 rank.

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 where already used, for example, in a recent paper [33] to compute the convolutions of two vectors in the QTT-format. From algorithmic point of view, there are still places where the algorithm can be improved, at least in two parts. 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 Wedderburn-type techniques [17] to compute approximations by reusing the information from previous sweeps and computing only low-rank updates.

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 not yet fully established (however, the recent work [25] 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 the least) objective for the future work are the applications of the TT-format. They include Fokker-Planck and multiparametric equations, quantum chemistry, where the TT-format has good perspectives.

References

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

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

[3] *Sloan I., Wozniakowski H.* When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals // *J. of Complexity*. 1998. V. 14, № 1. P. 1–33.

[4] *Wang X., Sloan I.* Why are high-dimensional finance problems often of low effective dimension? // *SIAM J. Sci. Comp.* 2006. V. 27, № 1. P. 159–183.

[5] *Kolda T. G., Bader B. W.* Tensor Decompositions and Applications // *SIAM Review*. 2009. V. 51, № 3. P. 455–500. doi: 10.1137/07070111X.

[6] *Beylkin G., Mohlenkamp M. J.* Numerical operator calculus in higher dimensions // *Proc. Nat. Acad. Sci. USA*. 2002. V. 99, № 16. P. 10246-10251. doi: 10.1073/pnas.112329799.

[7] *Beylkin G., Mohlenkamp M. J.* Algorithms for numerical analysis in high dimensions // *SIAM J. Sci. Comput.* 2005. V. 26, № 6. P. 2133-2159.

[8] *Caroll J. D., Chang J. J.* Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition // *Psychometrika*. 1970. V. 35. P. 283-319. doi: 10.1007/BF02310791.

[9] *Harshman R. A.* Foundations of the Parafac procedure: models and conditions for an explanatory multimodal factor analysis // *UCLA Working Papers in Phonetics*. 1970. V. 16. P. 1-84.

[10] *Espig M., Hackbusch W.* 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] *Oseledets I. V., Tyrtyshnikov E. E.* TT-cross algorithm for the approximation of multidimensional arrays // *Linear Algebra Appl.* 2010. V. 432, № 1. P. 70-88. doi: 10.1016/j.laa.2009.07.024.

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

[13] *Oseledets I. V.* Tensor train decomposition // *SIAM J. Sci. Comp.* 2011.

[14] *Oseledets I. V., Savostyanov D. V., Tyrtyshnikov E. E.* Linear algebra for tensor problems // *Computing.* 2009. V. 85, № 3. P. 169-188. doi: 10.1007/s00607-009-0047-6.

[15] *Savostyanov D. V., Tyrtyshnikov E. E.* Approximate multiplication of tensor matrices based on the individual filtering of factors // *J. Comp. Math. Math. Phys.* 2009. V. 49, № 10. P. 1662-1677. doi: 10.1134/s0965542509100029.

[16] *Savostyanov D. V.* Fast revealing of mode ranks of tensor in canonical format // *Numer. Math. Theor. Meth. Appl.* 2009. V. 2, № 4. P. 439-444. doi: 10.4208/nmtma.2009.m9006s.

[17] *Goreinov S. A., Oseledets I. V., Savostyanov D. V.* 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.

[18] *Savostyanov D. V., Tyrtyshnikov E. E., Zamarashkin N. L.* Fast truncation of mode ranks for bilinear tensor operations // *Numer. Lin. Alg. Appl.* 2011. doi: 10.1002/nla.765, Early View.

[19] *Khoromskaia V.* Computation of the Hartree-Fock exchange by tensor-structured methods // *Comp. Methd. Appl. Math.* 2008. V. 10, № 2.

[20] *Khoromskij B. N., Khoromskaia V.* Multigrid accelerated tensor approximation of function related multidimensional arrays // *SIAM J. Sci. Comp.* 2009. V. 31, № 4. P. 3002-3026. doi: 10.1137/080730408.

[21] *Chinnamsetty S. R., Flad H.-J., Khoromskaia V., Khoromskij B. N.* Tensor decomposition in electronic structure calculations on 3D Cartesian grids // *J. Comp. Phys.* 2009. V. 228, № 16. P. 5749-5762. doi: 10.1016/j.jcp.2009.04.043.

[22] *Khoromskaia V.* 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/.

[23] *Khoromskij B. N., Khoromskaia V., Flad. H.-J.* Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format // *SIAM J. Sci. Comp.* 2011. V. 33, № 1. P. 45-65. doi: 10.1137/090777372.

[24] *White S. R.* Density-matrix algorithms for quantum renormalization groups // *Phys. Rev. B*. 1993. V. 48, № 14. P. 10345–10356. doi: 10.1103/PhysRevB.48.10345.

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

[26] *Dolgov S., Oseledets I.* Solution of linear systems and matrix inversion in the TT-format: Preprint 19. — Leipzig: MPI MIS, 2011.

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

[28] *Khoromskij B. N.* $\mathcal{O}(d \log N)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling // *Constr. Appr.* 2011. doi: 10.1007/s00365-011-9131-1.

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

[30] *Oseledets I. V.* Approximation of $2^d \times 2^d$ matrices using tensor decomposition // *SIAM J. Matrix Anal. Appl.* 2010. V. 31, № 4. P. 2130-2145. doi: 10.1137/090757861.

[31] *Kazeev V., Khoromskij B. N.* 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`.

[32] *Khoromskij B. N., Oseledets I. V.* Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs // *Comp Meth. Appl. Math.* 2010. V. 10, № 4. P. 376-394.

[33] *Kazeev V., Khoromskij B. N., Tyrtyshnikov E. E.* Tensor structure of multilevel Toeplitz–circulant matrices leads to convolution with logarithmic complexity: Preprint 32. — Leipzig: MPI MIS, 2011.