Institute of Numerical Mathematics
Russian Academy of Sciences
119333, Moscow, Gubkina 8
www.inm.ras.ru

P U B
I N M
R A S
pub.inm.ras.ru

I.V. Oseledets, A. Yu. Mikhalev

A $\mathcal{O}(nr)$ representation of quasiseparable matrices

Preprint 2009-03 2009-03-15

Moscow 2009

A $\mathcal{O}(nr)$ representation of quasiseparable matrices

I.V. Oseledets, A. Yu. Mikhalev

Institute of Numerical Mathematics, Russian Academy of Sciences,
Russia, 119333 Moscow, Gubkina 8,
ivan.oseledets@gmail.com

March 15, 2009

[$]This work was supported by RFBR grants 08-01-00115, 09-01-00565 and RFBR/DFG grant 09-01-91332

Abstract

A new parametric representation for the general quasiseparable matrix is derived, based on the ideas from the multipole method. It uses functional expansions and successive skeleton approximations, approximations, but finally is formulated in the matrix language. The number of parameters is linear in the dimension of the matrix and in the quasiseparable rank. Numerical examples illustrate the effectiveness of our approach.

1. Introduction

Quasiseparable have received a lot of attention in the recent years, see for example [1, 2, 3, 4, 5]. There are numerous applications as well as interesting theoretical issues. Quasiseparable matrices are structured, i.e. described by a small (compared to n^2 elements in an $n \times n$ matrix) number of parameters. This structure can be used to design fast solvers and fast eigensolvers [3, 2, 6, 7]. There are several choices for the parametrization of the quasiseparable matrices,among them generator representation [6, 7], Givens weight representation [4]. Special attention has to be paid to the stability issues especially when the matrix is not quasiseparable but approximately quasiseparable. In this paper we present a new parametrization of quasiseparable matrices which has $\mathcal{O}(nr)$ parameters for a general quasiseparable matrix. The known generator representation of a quasiseparable matrix needs $\mathcal{O}(nr^2)$ to store [6, 7]. Also, it is worth to mention the alternative approach based on the sequence of Givens rotations [4]. Also the fast matrix-by-vector product algorithm is presented (in fact, our parametric representation gives us the ability to compute the product with an arbitrary vector fast). We do not describe the fast solver for our form, it will be presented elsewhere. It is interesting to find the relation between our decomposition and the existing ones, since our method in its final form is a purely matrix method and is based on successive skeleton decompositions [8] of submatrices in the lower and upper triangular parts of the matrix.

The quasiseparability property of a matrix deals with the ranks of submatrices in lower and upper triangular parts of a matrix. The lower quasiseparable rank r_l is defined as the maximal rank among all submatrices that lie strictly below the main diagonal, and the upper quasiseparable rank r_u is the maximal rank among all submatrices that lie strictly above the main diagonal. The matrix is called (r_l, r_u)—quasiseparable in that case. Our decomposition algorithm treats the lower triangular part and the upper triangular part indepedently, so in what follows we will assume that our matrix is lower triangular with zeros on the main diagonal and every submatrix with elements in lower triangular part has rank at most r. In the beginning we assume that we know the elements of $A_{ij}, i < j$, and want to compute the reduced approximation to A. This is the first problem we are going to solve, and present a new representation for such matrix containing $\mathcal{O}(nr)$ parameters. The second problem is the fast matrix-by-vector product.

2. From matrices to functions

Our initial motivation for studying quasiseparable matrices came from the N-body problem. As a simple example, consider the evaluation of sums of form

$$F_i = \sum_{j<i} \frac{q_j}{x_i - y_j}. \tag{1}$$

This is a multiplication of a lower triangular dense matrix by a vector. A naive implementation requires, of course, $\mathcal{O}(n^2)$ operations and we seek for something better. If

the nodes x_i, y_j of a Cauchy matrix $A_{ij} = \frac{1}{x_i - y_j}$ are properly sorted (as an example, consider the uniform grid, the matrix will be $A_{ij} = \frac{1}{i-j+\frac{1}{2}}$). A appears to be approximately quasiseparable — i.e. the ε-*rank* of all submatrices strictly under and below the main diagonal is small. We will illustrate our approach on the evaluation of the expression (1), but then it will be shown that it does not depend on the actual "interaction", and $\frac{1}{x_j - y_i}$ can be replaced by an element A_{ij} of an arbitrary quasiseparable matrix A. However, the electrostatic analogy appears to be very useful and natural, so we will call q_j *charges*, and the element of the matrix A *the interaction between particle* i *and particle* j.

There are many approaches for the evaluation of (1). One of the most effective is the *fast multipole method* [9, 10], which separates the interaction into "far" and "near" interactions, and approximates the interaction between separated clusters of particles by some expansions. In a matrix language it is nothing more than a decomposition of a matrix into non-intersectiong low rank blocks. However, such approach requires hierarchical splitting of points and it is not always easy to implement multipole and local expansions. It works fine, but in one-dimensional case we can expect something simpler and more effective. There is no need in the computation of the analytical decompositions, and everything can be done on the matrix level [11, 12, 13], but the dependence from N is $\mathcal{O}(N \log^{\alpha} N)$ and is not optimal (we aim at linear complexity in N without the logarithmic factor).

An interesting idea was proposed in [14] for the computation of so called excluded sums and functional expansions. It was suggested in [14] to replace the evaluation of sums in (1) by the evaluation of *functional sums*

$$f_i(x) = \sum_{i<j} \frac{q_j}{x - y_j}, \tag{2}$$

and F_i are just the values of f_i in the nodes:

$$F_i = f_i(x_i).$$

What is the benefit? The benefit is that the functions f_i satisfy short *recurrence relation* of form

$$f_{i+1} = f_i + \frac{q_i}{x - y_i}. \tag{3}$$

If we for a second imagine, that f_i were not functions but numbers, then (3) would give us a fast algorithm. However, f_i are still functions, and to store a function we have to store some discrete representation of it. In our case, the most obvious way to represent a function is to represent it as a set of charges $\widehat{q}_k$ and a set of charges positions $\widehat{y}_k$. The function is described as

$$f(x) = \sum_{k=1}^{r} \frac{\widehat{q}_k}{x - \widehat{y}_k}. \tag{4}$$

In this representation the recurrence relation (3) can be used to "compute" f_i. However as everyone can notice, there is no gain: at the end the function is represented by n

charges, and we have $\mathcal{O}(n^2)$ complexity. The number of parameters has to be reduced. So here comes the main idea: *after each iteration step, approximate* f_i *represented by* r *charges, with a* $\widehat{f}_i$ *represented by* $\widehat{r} \leqslant r$ *charges, such that*

$$\widehat{f}_i \approx f_i. \tag{5}$$

This is quite informal, so let us be more specific. From a function $f_i(x)$ we need only its value at one point x_i. However, the function f_i influences $f_{i+1}, \ldots, f_n$ by the recurrence relations (3), so (5) has to hold at also at points $x_{i+1}, \ldots, x_n$. So that is the set, where we approximate the function. Now we can specify what is the analogy between functional terms and matrix terms. The "points" x_i correspond to the rows of the matrix A, the sources — to the columns of the matrix A. The function f_i, specified by equivalent charges located at $\widehat{y}_j$ and defined in the nodes $x_{i+1}, \ldots, x_n$ corresponds to the *submatrix*. If we make no approximation, then the function corresponds to the leading submatrix U_i of our matrix which, in Matlab notation is given as

$$U_i = A((i+1):n, 1:i).$$

The evaluation of a function specified by charges $q_1, \ldots, q_i$ in the nodes is just matrix by vector product,

$$U_i q.$$

The matrix is quasiseparable (approximately) so the main assumption is that the number of *equivalent charges* required to represent each f_i remains small. In that case the algorithm is fast. However, there are two questions:

1. When it is possible to approximate a function?
2. How to approximate it?

A very simple idea can be adopted. Suppose at some step we have an approximation of f_i with r equivalent charges (q_k, y_k), and we want to compute f_{i+1}. f_{i+1} can be trivially represented by $r+1$ equivalent charges, and there are $r+2$ possibilities for the approximation: we can leave either r charges, or leave $(r+1)$ charges. For each of $(r+1)$ subset of cardinality r we can calculate the best possible approximation (we will show how later on) and choose the best one among them. If it does not satisfy our accuracy requirements, then we leave everything as it is, storing $(r+1)$ equivalent charges as the representation of the function f_i. In order to find out when it is possible and how the number of the equivalent charges can be estimated, we have to go back from functions to matrices.

3. Back to matrices

The approximation condition (5) has to hold in the nodes $x_{i+1}, \ldots, x_n$. A function f_i is specified by its equivalent charges, $\widehat{q}_k$ and their positions $\widehat{y}_k$. The positions of the equivalent charges are just the *column indices*, and the approximation nodes are just

the row indices, and the function f_i corresponds to some *submatrix* of a matrix A. Initially, it was specified by i columns, (i.e. equivalent charges) and the evaluation of a function, but it is now replaced by a smaller number of columns. When f_{i+1} is approximated and f_i given one column to a matrix is added, but it now should be evaluated in a smaller number of points, i.e. one row is removed. These two submatrices lie strictly in the lower triangular part of the matrix, and for our case of quasiseparable matrices they both have rank not higher that r. Denote this "active submatrix" on the i-th step by U_i. U_i has dimension $(n-i) \times i$, and the submatrix corresponding to the equivalent charges has size $(n-i) \times r_i$, $r_i \leqslant i$. Denote it by $\widehat{U}_i$. Suppose from now that the matrix is exactly quasiseparable, i.e. $r_i \leqslant r$. Then $\widehat{U}_i$ form the basis of the column space of U_i and

$$U_i = \widehat{U}_i S_i,$$

where S_i is $r_i \times i$. When we go from i to $(i+1)$, we can form the column space basis of U_{i+1} from $\widehat{U}_i$ with the first row excluded and the newly added vector a_{i+1}, the dimension of that space would be $(r+1)$, but the rank of this submatrix is not higher than r, therefore we can take r vectors out of $(r+1)$ that span the whole column subspace of U_{i+1}:

$$\widehat{U}_{i+1} \approx \begin{bmatrix} \widehat{U}'_i & u_i \end{bmatrix} Q_i,$$

where a matrix Q_i has size $(r+1) \times r$. There are $r+1$ possible choices fo $\widehat{U}_{i+1}$. For each possible choice we can compute Q_i by solving an overdetermined linear system for Q_i and finding the actual residue

$$\|\widehat{U}_{i+1} - \begin{bmatrix} \widehat{U}'_i & u_i \end{bmatrix} Q_i\|,$$

The matrix Q_i has an obvious special structure. If we select every column except the column with the number k, the matrix Q will differ from the identity matrix only in one column, since this excluded column has to be a linear combination of columns that are left. Only the coefficients of this linear combination are required, and the storage for the matrix Q_i is only r floats plus one integer.

The matrix Q_i is the only matrix that is needed. It recalculates the equivalent charges. If a function f_i is represented by a set of charges $\widehat{q} = [\widehat{q}_k]$, a new charge q_{i+1} is added then the new charges of a new function are

$$q' = Q_i^\top \begin{bmatrix} \widehat{q} \\ q_{i+1} \end{bmatrix}. \tag{6}$$

So, if we know the "transition matrices" Q_i, then the equivalent charges can be easily computed for all i. If the ranks of the submatrices are bounded by r, then each matrix Q_i would require at most r floats to store and $\mathcal{O}(r)$ operations to apply. For each i we have to store r equivalent charges. After the charges were computed we have to evaluate

each function $f_i(x)$ in one node. This is equivalent to the computation of

$$\varphi_i = \sum_{k=1}^{r} \frac{\widehat{q}_k}{x_i - \widehat{y}_k},$$

i.e. for each i we have to store additionally r matrix elements $\frac{1}{x_i - \widehat{y}_k}$. In total, for the storage of the new representation we have $2nr$ parameters and the number of operations required to apply the representation is $\mathcal{O}(rn)$.

4. Formal part

Now when the answer is known, the results can be formulated more rigorously and formally in the matrix language. We want to multiply a lower triangular quasiseparable matrix A by a vector q:

$$z = Aq.$$

The elements z_i of the vector z are computed successively:

$$z_i = a_i^\top q,$$

where $a_i^\top$ is the i-th row of the matrix A. It is equal to

$$a_i = A^\top e_i,$$

where e_i is the i-th column of the identity matrix, and since the matrix is lower triangular,

$$a_i = A_i^\top e_1,$$

where

$$A_i = A((i+1):n, 1:i)$$

is the submatrix of the matrix A. For brevity, denote by $m_i = n - i$ the number of columns in A_i. So,

$$z_i = e_1^\top A_i q_i,$$

where $q_i = q(1:i)$. Matrices A_i are located strictly in the lower triangular part of a quasiseparable matrix A, i.e. their rank is bounded from above by r. Therefore, there exists a dyadic (skeleton decomposition) of A_i:

$$A_i = U_i V_i^\top, \tag{7}$$

where U_i is $m_i \times r_i$, V_i is $i \times r_i$, $r_i = \operatorname{rank} A_i$, and

$$z_i = u_i^\top s_i,$$

where

$$s_i = V_i^\top q_i, u_i = U_i^\top e_1,$$

and s_i, u_i both have length r_i. Now we have to find the way to recompute u_i, s_i when going from i to $(i+1)$. Matrices A_i and A_{i+1} have a $(m_i - 1) \times i$ submatrix in common, and A_{i+1} is a special rank-1 correction of $\widehat{A}_i = A_i(2 : m_i, 1 : i)$:

$$A_{i+1} = \begin{bmatrix} \widehat{A}_i & a_{i+1} \end{bmatrix},$$

where a_{i+1} is a column that is added to our submatrix. From the skeleton decomposition of A_i we now have the skeleton decomposition of $\widehat{A}_i$ just by throwing away one row of U_i and hence the skeleton decomposition of A_{i+1} is

$$A_{i+1} = \begin{bmatrix} \widehat{U}_i V_i^\top & a_{i+1} \end{bmatrix} = \begin{bmatrix} \widehat{U}_i & a_{i+1} \end{bmatrix} \begin{bmatrix} V_i^\top & 0 \\ 0 & 1 \end{bmatrix}.$$

The matrix A_{i+1} has rank not higher than $r_{i+1} \leqslant r$, and the matrix V_i has full column rank, therefore

$$U'_{i+1} = \begin{bmatrix} \widehat{U}_i & a_{i+1} \end{bmatrix} = U_{i+1} Q_{i+1}^\top, \tag{8}$$

where Q_i is $(r_i + 1) \times r_{i+1}$, and the skeleton decomposition of A_{i+1} reads

$$A_{i+1} = \widehat{U}_{i+1} \widehat{V}_{i+1}^\top,$$

where

$$\widehat{V}_{i+1} = \begin{bmatrix} V_i & 0 \\ 0 & 1 \end{bmatrix} Q_{i+1}.$$

The matrix A_{i+1} is a special rank-2 correction of the matrix A_i, and the rank of A_{i+1} can not be smaller than $r_i - 1$ and can not be higher than $r_i + 1$:

$$r_i - 1 \leqslant r_{i+1} \leqslant r_i + 1.$$

The case when the rank drops by 1 is possible, but in the generic case it stays the same: $r_{i+1} = r_i = r$. The rank of A_{i+1} can be found by inspecting the rank of $\widehat{U}_{i+1}$. For the selection of Q_{i+1} the low-rank approximation of $\widehat{U}_{i+1}$ is needed. We favour the selection based on the skeleton decomposition, i.e. on the selection of certain columns and rows in a low-rank matrix. The skeleton decomposition of a matrix B of rank r has form

$$B = C \widehat{B}^{-1} R,$$

where C contains some r rows of the matrix B, R contains some r rows of the matrix B and $\widehat{B}$ is the submatrix on their intersection. If $\widehat{B}$ is the submatrix of maximal volume (i.e., determinant in modulus) then the error of the skeleton decomposition can be estimated elementwise by $(r+1)\sigma_{r+1}(B)$ [1][15]. For the matrix U_{i+1} we take r_i columns of $\widehat{U}_{i+1}$ that are linearly indepedent. When U_{i+1} is a submatrix of $\widehat{U}_{i+1}$ with only one

[1] By $\sigma_k(B)$ we denote the k-th singular value of the matrix B, $\sigma_1 \geqslant sigma_2 \geqslant \ldots$

column omitted, Q_i has form

$$Q_i = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & p_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & p_s & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & p_{s+1} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & p_m & 0 & \cdots & 1 \end{bmatrix},$$

i.e. it is defined by r parameters. The matrix Q_i is all we need to evaluate s_{i+1}, since

$$s_{i+1} = V_{i+1}^\top q_{i+1} = Q_i^\top \begin{bmatrix} s_i \\ \widehat{q}_{i\cdot} \end{bmatrix}.$$

The case when $r_{i+1} = r_i - 1$ (i.e., the rank falls) can be considered as a part of the case $r_{i+1} = r_i$ since it is clear from the proof that only V_{i+1} has to have full column rank and the matrix U_{i+1} can be rank-deficient.

From now for simplicity we assume that $r_{i+1} = r_i = r$. It does not always hold, for example for small $i < r$ and for i that are sufficiently close to n. However, these "border" cases do no not influence the algorithms and complexity estimates much.

To represent a matrix, additionally to the matrices Q_i the vectors u_i have to be stored. These vectors also contain r parameters each.

The order-r lower-triangular quasiseparable matrix is stored as transition matrices Q_i vectors u_i requiring $\mathcal{O}(nr)$ parameters. It is indeed a matrix representation, since using these parameters the product with an arbitrary vector can be computed. Such product requires $\mathcal{O}(nr)$ operations due to the special structure of Q_i. In the computer implementation, the matrix A is stored by a sequence of quadruples. Each quadruple $P = P_i$ has form $P = (s, p, r, u)$, where s is then number of column that is thrown out at the i-th step, if the update is does not increase the rank (i.e. $r_{i+1} \leqslant r_i$), and $s = -1$ otherwise, p is an s-th column of Q_i and r is the number of columns in U_i, and u is the first row of U_i (required to compute the result of matrix-by-vector product). The greedy approximation algorithm (it needs $\mathcal{O}(n^2 r)$) operations is presented in Algorithm 1, and the matrix-by-vector product is described in Algorithm 2.

Algorithm 1: Greedy approximation algorithm

Input: $n \times n$ lower triangular matrix A, required accuracy ε.

Output: Parameters P_k.

{Initialization}
$I = \{\}, r = 1$.
for $k = 1$ to n **do**
  $J := (k+1) : n$.
  $I := I \cup k$.
  $U := A(I, J)$
  $u = U(1, :)$.
  {Check if the rank reduction is possible}
  if $\sigma_r(U) < \varepsilon\sigma_1(U)$ **then**
    {Throw out one column}
    Find s that minimizes $f(s) = \min_p \|U(:, S)p - U(:, s)\|$, where $S = (1 : r) \setminus s$, i.e. find $(r-1)$ columns that span the best basis in the column space of U.
    {Find p}
    $p = \arg\min_p \|U(:, S)p - U(:, s)\|$, where $S = (1 : r) \setminus s$
    Set transition parameters: $P_k := \{s, p, r, u\}$.
    New basis columns: $S = (1 : r) \setminus s, I := I(S)$.
  else
    {Rank has to be increased}
    $r = r + 1$.
    Set transition parameters: $P_k := \{-1, 0, r+1, u\}$.
  end if
end for

Algorithm 2: Fast matrix-by-vector product

Input: $n \times n$ lower triangular quasiseparable matrix A represented by parameters P_k, $k = 1, \ldots, n$, vector x.

Output: $y = Ax$.

{Initialization}
$q = [\,]$.
for $k = 1$ to n **do**
  $s = P_k(1), p = P_k(2), r = P_k(3), u = P_k(4)$.
  $q := [\; q \quad x_k \;]$.
  {Compute element}
  $y_k = u^\top q$.
  if $s \neq -1$ **then**
    {Rank is not increasing}
    for $i = 1$ to r **do**
      $q(i) := q(i) - p(s)q(s)$
      $q := q(S), S = (1 : r) \setminus s$.
    end for
  end if
end for

5. Numerical examples

We have tested the factorization algorithm and the matrix-by-vector algorithm for $n \times n$ matrices. As a test we took

$$A_{ij} = \frac{1}{i - j + \frac{1}{2}},$$

and approximated the lower triangular and the upper triangular parts separately In the Tables r the maximum rank reached in decomposition for lower or upper part of matrix Ftime is a time of factorization and Mtime is a time for matrix-by-vector multiplication in seconds. Also the "true" approximation error in the Frobenius norm was computed and compared to the accuracy parameter ε that is specified in the program. In Table 1 the results are given for a fixed matrix order $n = 1000$ and ε is varying, in the Table 2 ε is set to 10^{-16} and n changes from 1000 to 10000, and Table 3 contains a similar experiment but with $\varepsilon = 10^{-6}$.

r	Approximation error	Ftime	Mtime
9	8.5e-3	4.52	6.4e-4
12	2.4e-5	9.75	9.2e-4
16	5.7e-8	17.86	11.1e-4
19	8.7e-10	26.64	12e-4
21	1.1e-11	35.2	13.1e-4
24	8.5e-14	48.73	15.2e-4
27	9.2e-16	66	16e-4

Table 1. Dependence of the approximation time and matrix-by-vector time from ε, $n = 1000$

n	r	Approximation error	Ftime	Mtime
1000	27	9.2e-16	66	16e-4
2000	30	11.9e-16	313.43	44.5e-4
3000	31	13.3e-16	772.51	70e-4
4000	32	14.3e-16	1464.53	96.2e-4
5000	33	15e-16	2474.42	122.3e-4
6000	34	16.1e-16	3847.75	149.5e-4
7000	34	16.4e-16	5301.87	175e-4
8000	35	16.8e-16	7282.94	203.1e-4
9000	35	17e-16	9494.13	230e-4
10000	36	16.9e-16	12166.73	257.4e-4

Table 2. Dependence of the approximation time and matrix-by-vector time from n, accuracy parameter $\varepsilon = 10^{-16}$

n	r	Approximation error	Ftime	Mtime
1000	17	9.25e-9	21.92	10.7e-4
2000	19	5.6e-9	94.86	33.75e-4
3000	19	6.45e-9	226.8	56.5e-4
4000	20	5e-9	435.52	77.1e-4
5000	20	4.64e-9	713.57	97.1e-4
6000	21	4.76e-9	1056.5	117.7e-4
7000	21	4.4e-9	1519.9	138.5e-4
8000	21	4.1e-9	2014.5	158.8e-4
9000	21	3.8e-9	2623.5	178.6e-4
10000	22	4.3e-9	3313.5	200e-4

Table 3. Dependence of the approximation time and matrix-by-vector time from n, accuracy parameter $\varepsilon = 10^{-9}$

The numerical experiments confirm the theoretical investigations. For the example considered, even to achieve the machine precision the rank $r = 36$ is sufficient, and the matrix-by-vector product is very fast. The approximation algorithm is "naive" and scales quadratically with the matrix dimensions, so for really large n other approaches, maybe using some cross approximation techniques are required, but this is a subject of the ongoing research.

6. Conclusion and future work

We have presented another view at quasiseparable matrices, based on functional expansions and short recurrence relations. Our main interest comes from integral equations where approximately quasiseparable matrices arise naturally from the discretization of one-dimensional integral equations. The electrostatic interpretation with equivalent charges proved to be very useful and lead to a simple algorithm. It admits a nice matrix representation with $\mathcal{O}(nr)$ parameters, compared to $\mathcal{O}(nr^2)$ parameters in the generator representation for a quasiseparable matrix. The functional interpretation allows generalization to higher dimensions, i.e. to the discretization of integral equations in plane and volume regions, however in this case the situation is much more delicate and interesting. On the matrix language is the representation of a matrix as a collection of intersecting low-rank matrices. In 1D case, this is just a splitting of a matrix into a lower triangular and upper triangular and covering them by submatrices of form $A((i+1):n, 1:i)$ for the lower triangular part and analogously for the upper triangular part. It is interesting to do the same for multidimensional case, and that is the subject of the ongoing work. Also basic operations in our representation have to be performed efficiently, and that will be reported in future papers.

References

[1] *Dewilde P., van der Veen A.-J.* Time-varying systems and computations. — Boston, MA: Kluwer Academic Publishers, 1998. — P. xiv+459.

[2] *Eidelman Y., Gohberg I., Olshevsky V.* The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order // *Linear Algebra Appl.* 2005. V. 404. P. 305–324.

[3] *Eidelman Y., Gohberg I., Olshevsky V.* Eigenstructure of order-one-quasiseparable matrices. Three-term and two-term recurrence relations // *Linear Algebra Appl.* 2005. V. 405. P. 1–40.

[4] *Vandebril R., Van Barel M., Mastronardi N.* A note on the representation and definition of semiseparable matrices // *Numer. Linear Algebra Appl.* 2005. V. 12, № 8. P. 839–858.

[5] *Mastronardi N., Van Barel M., Vandebril R.* Matrix computations with semiseparable matrices. — John Hopkins University Press, 2007.

[6] *Eidelman Y., Gohberg I.* Fast inversion algorithms for diagonal plus semiseparable matrices // *Integral Equations Operator Theory*. 1997. V. 27, № 2. P. 165–183.

[7] *Eidelman Y., Gohberg I.* On a new class of structured matrices // *Integral Equations Operator Theory*. 1999. V. 34, № 3. P. 293–324.

[8] *Goreinov S. A., Tyrtyshikov E. E., Zamarashkin N. L.* A theory of pseudo–skeleton approximations // *Lin. Algebra Appl.* 1997. V. 261. P. 1-21.

[9] *Rokhlin V.* Rapid solution of integral equations of classical potential theory // *Journal of Computational Physics*. 1985. V. 60, № 2. P. 187-207.

[10] *Rokhlin V., Greengrad L.* A fast algorithm for particle simulations // *Journal of Computational Physics*. 1987. V. 135. P. 280-292.

[11] *Tyrtyshnikov E. E.* Mosaic-skeleton approximations // *Calcolo*. 1997. V. 33. P. 47-58.

[12] *Tyrtyshnikov E. E.* Piecewise separable matrices // *Calcolo*. 2005. V. 42, № 3. P. 243-248.

[13] *Tyrtyshnikov E. E.* Mosaic ranks for weakly semiseparable matrices // *Notes on Numerical Fluid Mechanics*. 2000. V. 73. P. 36-41.

[14] *Demaine E. D., Demaine M. L., Edelman A. et al.* Building Blocks and excluded sums // *SIAM News*. 2005. — Jan.

[15] *Goreinov S. A., Tyrtyshnikov E. E.* The maximal-volume concept in approximation by low-rank matrices // *Contemporary Mathematics*. 2001. V. 208. P. 47-51.