

One-sided Non-symmetric Matrix Decomposition by Means of Perturbed Rotations

Robert T. Wirski¹ and Aneta O. Hapka¹

¹ Faculty of Electronics and Computer Science, Koszalin University of Technology, Poland

Abstract. In the paper, a novel decomposition technique of a non-symmetric real square matrix is presented. The algorithm is a generalization of QR Givens rotations technique by means of matrices provisionally called perturbed rotations. It is described in detail and summarized in algorithmic steps. Its existence conditions are given. The orthogonal matrix case is discussed. To illustrate the algorithm, a simple exemplary matrix decomposition is included.

Key words: matrix diagonalization; QR; Givens rotation; orthogonal matrix; perturbed rotation

1. INTRODUCTION

Say, we have a real non-symmetric $N \times N$ matrix X . It is known, that by means of the QR algorithm [1, 2, 3], X can be represented as a product of orthogonal Q and an upper triangular R , i.e.

$$X = QR. \quad (1)$$

It can be obtained by means of $N \times N$ Givens rotation matrices [4, Sec. 5.1.8], given by

$$G_{m,n}(\phi_k) = \begin{matrix} & \begin{matrix} m & n \end{matrix} \\ \begin{matrix} m \\ n \end{matrix} & \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos \phi_k & & \sin \phi_k \\ & & & \ddots & \\ & 0 & -\sin \phi_k & & \cos \phi_k \\ & & & & 1 \\ & & & & & \ddots \end{bmatrix} \end{matrix}. \quad (2)$$

Using (2), it is possible to zero a selected element of a vector by multiplication

$$\begin{bmatrix} \cos \phi_k & \sin \phi_k \\ -\sin \phi_k & \cos \phi_k \end{bmatrix}^T \begin{bmatrix} x_m \\ x_n \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \quad (3)$$

where

$$\cos \phi_k = \frac{x_m}{\sqrt{x_m^2 + x_n^2}}, \quad \sin \phi_k = \frac{-x_n}{\sqrt{x_m^2 + x_n^2}}. \quad (4)$$

Then, by the appropriate repetition of (3), X can be converted to an upper triangular matrix. For details, see [4, Sec. 5.2.5]. If X is an orthogonal matrix, then QR algorithm becomes the decomposition technique, so that R becomes the diagonal matrix E with ± 1 on its main diagonal. If during the process, each $G_{m,n}(\phi_k)$ is recorded for $k = 1, 2, \dots$ then the QR algorithm for the orthogonal X will produce the decomposition

$$X = \left(\prod_{m=1}^{N-1} \prod_{n=N}^{m+1} G_{m,n}(\phi_k) \right) E. \quad (5)$$

The idea is to find out whether it is possible to decompose a general non-orthogonal matrix in the manner similar to (5). Motivation for such approach would be to develop novel computing algorithms for discrete state-space equations which, for a one-dimensional system, are given by

$$\begin{cases} x(n+1) = Ax(n) + Bp(n) \\ r(n) = Cx(n) + Dp(n) \end{cases}, \quad (6)$$

where $p(n)$ and $r(n)$ are input and output signals, respectively; $x(n)$ is a state-space vector representing internal system's values; A , B , C , and D are matrices of appropriate dimensions, constant for time-invariant systems. They can be combined into

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \quad (7)$$

Similarly, one can construct state-space equations for multi-dimensional signals. The main implementation problem of (6) is to effectively decompose (7) into a product of simple matrices which provide computational precision and easy implementation of (6). It is known, that one can use Givens rotations (2) which are efficient due to orthogonality. However, such decomposition requires X to be orthogonal, which is, in general, not easy to fulfill. For details on such implementations, see [5]. That is why we look for similar approaches which do not impose orthogonality on X . The authors believe that such techniques will also find applications in areas like automation and control or matrix algebra in general. The problem formulation and the obtained results are presented in the following sections.

In the paper, we use the following notation: $k : l$, where $l > k$, is a vector of the form $[k, k+1, \dots, l]$; $X_{k:l, m:n}$ is a submatrix obtained by forming the intersection of $k : l$ rows and $m : n$ columns of X ; X^T is a transposition of X .

2. THE ALGORITHM

$$\prod_{j=2}^N c_{1,j}^{-1}(l_k, u_k) = \frac{1}{\prod_{i=1}^{N-1} h_i} \begin{bmatrix} 1 & -u_{N-1}P_{N-1} & -u_{N-2}P_{N-2} & \cdots & -u_1P_1 \\ -l_{N-1}R_{N-1} & P_{N-1}R_{N-1} & l_{N-1}u_{N-2}P_{N-2}R_{N-1} & \cdots & l_{N-1}u_1P_1R_{N-1} \\ -l_{N-2}R_{N-2} & 0 & P_{N-2}R_{N-2} & \cdots & l_{N-2}u_1P_1R_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -l_2R_2 & 0 & 0 & \cdots & l_2u_1P_1R_2 \\ -l_1R_1 & 0 & 0 & \cdots & P_1R_1 \end{bmatrix} \quad (8)$$

2.1. Extension of the Givens Rotation Matrix

Consider (2) for $N = 2$:

$$G(\phi_k) = \begin{bmatrix} \cos \phi_k & \sin \phi_k \\ -\sin \phi_k & \cos \phi_k \end{bmatrix} = \cos \phi_k \begin{bmatrix} 1 & \tan \phi_k \\ -\tan \phi_k & 1 \end{bmatrix}. \quad (9)$$

Let us shift one of angles in (9) by δ , which leads to

$$\begin{aligned} G(\phi_k, \delta) &= \cos \phi \begin{bmatrix} 1 & \tan \phi_k \\ -\tan(\phi_k + \delta) & 1 \end{bmatrix} = \\ &= \cos \phi_k \begin{bmatrix} 1 & \tan \phi_k \\ -\frac{\tan \phi_k + \tan \delta}{1 - \tan \phi_k \tan \delta} & 1 \end{bmatrix}. \end{aligned} \quad (10)$$

For $|\tan \phi_k \tan \delta| \ll 1$ and $\cos \phi_k$ omitted, (10) can be interpreted as a perturbed rotation matrix

$$G(\phi_k, \delta) \approx \begin{bmatrix} 1 & \tan \phi_k \\ -(\tan \phi_k + \tan \delta) & 1 \end{bmatrix}. \quad (11)$$

Let $u_k = \tan \phi_k$ and $l_k = -(\tan \phi_k + \tan \delta)$. Then, (11) can be generalized to

$$c_{m,n}(l_k, u_k) = \begin{bmatrix} & m & n \\ & & \\ 1 & & & & \\ & \ddots & & & 0 \\ & & 1 & & u_k \\ & & & \ddots & \\ & & & & 1 \\ & 0 & & & 1 \\ & & & & & \ddots \end{bmatrix}. \quad (12)$$

Define,

$$h_k = 1 - l_k u_k. \quad (13)$$

For $h_k \neq 0$, (12) possesses its inverse, given by

$$c_{m,n}^{-1}(l_k, u_k) = \frac{1}{h_k} \begin{bmatrix} & m & n \\ & & \\ h_k & & & & \\ & \ddots & & & 0 \\ & & 1 & & -u_k \\ & & & h_k & \\ & & & & \ddots \\ & & -l_k & & 1 \\ & 0 & & & h_k \\ & & & & & \ddots \end{bmatrix}. \quad (14)$$

From (14), the perturbed rotation (12) where $l_k u_k = 1$ is the noninvertible matrix.

In the following sections, an algorithm is presented, similar to QR, which decompose X into the product of invertible perturbed rotations and the diagonal matrix.

2.2. Problem Formulation

Let $X = [x_{i,j}]$ be the $N \times N$ real non-symmetric matrix and $m = 1$. Consider the decomposition

$$c_{m,2}^{-1}(l_{N-1}, u_{N-1}) c_{m,3}^{-1}(l_{N-2}, u_{N-2}) \cdots c_{m,N}^{-1}(l_1, u_1) X = Y, \quad (15)$$

where

$$Y = [y_{i,j}] = \begin{bmatrix} y_{1,1} & 0 & \cdots & 0 \\ 0 & y_{2,2} & \cdots & y_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & y_{2,N} & \cdots & y_{N,N} \end{bmatrix}. \quad (16)$$

The problem is to find l_k and u_k for $k = 1, \dots, N-1$, for which (15) is satisfied. Once found, the same technique can be applied to $X = Y$ for $m = 2$, etc. As the result, one obtains the decomposition of X into the product of perturbed rotations (12) and the diagonal matrix E , given by

$$X = \left(\prod_{m=1}^{N-1} \prod_{n=N}^{m+1} c_{m,n}(l_k, u_k) \right) E. \quad (17)$$

2.3. Solution to the Decomposition Problem

From (14) and (15), one can observe that the product of perturbed inverted rotations in the left hand side of (15) contains a multiplication of subsequent h_k for different set of k . Let us

define them as P_k and R_i , given by

$$P_k = \begin{cases} 1 & \text{for } k = 1 \\ \prod_{j=1}^{k-1} h_j & \text{for } k = 2, \dots, N-1 \end{cases} \quad (18)$$

and

$$R_i = \begin{cases} \prod_{j=i+1}^{N-1} h_j & \text{for } i = 1, \dots, N-2 \\ 1 & \text{for } i = N-1 \end{cases}. \quad (19)$$

By inspection one can verify, that the left hand side product of (15) is given by (8), where

$$k = N - j + 1. \quad (20)$$

Evaluating $y_{1,j}$ in (15) for $m = 1$, $j = 2, \dots, N$ and using (8), one gets

$$y_{i,j} = x_{1,j} - \sum_{i=2}^N u_{N-i+1} P_{N-i+1} x_{i,j} = 0 \text{ for } j = 2, \dots, N. \quad (21)$$

Let us construct the sub-vector $x_r = X_{1,2:N}$ and the sub-matrix $x_s = X_{2:N,2:N}$. Then, (21) can be represented as

$$x_s^T F = x_r^T, \quad (22)$$

where $F = [f_1 \dots f_{N-1}]^T$, such that

$$f_{N-k} = u_k P_k \text{ for } k = 1, \dots, N-1. \quad (23)$$

Using (23), one gets

$$u_k = \frac{f_{N-k}}{P_k}. \quad (24)$$

Similarly, solving $y_{j,1}$ in (15) for $j = 2, \dots, N$ and using (8), one gets

$$l_{N-j+1} \left(-x_{1,1} + \sum_{k=j+1}^N u_{N-k+1} P_{N-k+1} x_{k,1} \right) + P_{N-j+1} x_{j,1} = 0 \quad (25a)$$

$$-l_1 x_{1,1} + P_1 x_{N,1} = 0 \quad (25b)$$

for $j = 2, \dots, N-1$. From (25b),

$$l_1 = \frac{P_1 x_{N,1}}{x_{1,1}}. \quad (26)$$

Substituting (20) to (25a), the remaining l_k can be found using

$$l_k = \frac{P_k x_{N-k+1,1}}{D_k} \text{ for } k = 2, \dots, N-1, \quad (27)$$

where

$$D_k = x_{1,1} - \sum_{j=N-k+2}^N f_{j-1} x_{j,1}. \quad (28)$$

Using (24), (26), and (27), one can construct the inverted perturbed rotations (14) which diagonalize the first row and column of Y . Now, the similar procedure can be applied to the remaining submatrix $Y_{2:N,2:N}$, and so on, until Y becomes the diagonal one. Let us extend the obtained equations for $m = 1, \dots, N-1$. From (22), $x_r = X_{m+1:N,m+1:N}$ and

$x_s = X_{m,m+1:N}$. Eq. (28) can be generalized to

$$D_k = x_{m,m} - \sum_{j=N-k+m+1}^N f_{j-m} x_{j,m} \text{ for } k = m+1, \dots, N-1. \quad (29)$$

and can be further converted into a recurrent form, given by

$$\begin{cases} D_m = x_{m,m} \\ D_k = D_{k-1} - f_{N-k+1} x_{N-k+m+1,m} \text{ for } k = m+1, \dots, N-1 \end{cases}. \quad (30)$$

Similarly, (18) can be evaluated using

$$\begin{cases} P_m = 1 \\ P_k = P_{k-1} h_{k-1} \text{ for } k = m+1, \dots, N-1 \end{cases}. \quad (31)$$

Eq. (27) extends to

$$l_k = \frac{P_k x_{N-k+m,m}}{D_k} \text{ for } k = m+1, \dots, N-1. \quad (32)$$

Consider (13) from the previous iteration of k , which is

$$h_{k-1} = 1 - u_{k-1} l_{k-1}. \quad (33)$$

Substituting (32) and (24) into (33), one gets

$$h_{k-1} = \frac{D_{k-1} - f_{N-k+1} x_{N-k+m+1,m}}{D_{k-1}}. \quad (34)$$

By (30), the numerator of (34) is D_k for $k = m+1, \dots, N-1$. One can use (34) to evaluate D_k using the simpler formula, quite similar to (31), which is

$$\begin{cases} D_m = x_{m,m} \\ D_k = D_{k-1} h_{k-1} \text{ for } k = m+1, \dots, N-1 \end{cases}. \quad (35)$$

The results above can be arranged into the iterative algorithm which decompose X :

1. Let $m = 1$.
2. Solve (22) to get f_i .
3. Let C^{-1} be the $N \times N$ identity matrix and $k = m$.
4. Evaluate P_k, D_k , using (31), (35).
5. Evaluate l_k and u_k , using (24), (32).
6. Construct and record $C_{m,N-k+m}$ as well as $c_{m,N-k+m}^{-1}$, given by (12) and (14), respectively. Accumulate $C^{-1} \leftarrow c_{m,n}^{-1} C^{-1}$.
7. Repeat steps 4-6 for $k = m+1, \dots, N-1$.
8. $X \leftarrow C^{-1} X$.
9. Repeat steps 2-8 for $m = 2, \dots, N-1$.
10. $E = X$.

3. EXISTENCE OF THE SOLUTION TO THE DECOMPOSITION PROBLEM

Lemma 1. The algorithm given in Sec. 2.3 for $m = 1$ applied to the $N \times N$ real matrix X , or for $m > 1$ applied to $X = Y$, obtained from the previous $m-1$ step is feasible if and only if

$$|X_{m+1:N,m+1:N}| \neq 0 \quad (36)$$

$$x_{m,m} \neq 0 \quad (37a)$$

and

$$h_k \neq 0 \text{ for } k = m, \dots, N-1. \quad (37b)$$

Proof. The algorithm consists of final steps. So, it will finish if all computations are feasible. i.e. the linear system (22) and three divisions which may break computations, namely by h_k in (14), by D_k in (32), and by P_k in (24). To guarantee the feasibility of computations they all must be nonzero. $P_1 \neq 0$ by (31) as well as $D_1 \neq 0$ by (35), (37a). Clearly, (36) guarantees that (22) can be solved. (37b) guarantees that $P_k \neq 0$ and $D_k \neq 0$ for $k = m, \dots, N-1$ due to, respectively, (31) and (35).

For the "only if" part assume, that the algorithm is computationally solvable and some or all conditions (36), (37) do not hold. However, when $|X_{m+1:N, m+1:N}| = 0$, (22) does not have a finite solution.

From (31), (35), one gets

$$h_{k-1} = \frac{P_k}{P_{k-1}} \quad h_{k-1} = \frac{D_k}{D_{k-1}}, \quad (38)$$

for $k = m+1, \dots, N-1$, starting for $P_m = 1, D_m = x_{m,m}$. Now, assume that there are some k for which $h_{k-1} = 0$. Then, by (38), $P_k = 0$ and $D_k = 0$, which brakes (24) and (32). So the statement, that the algorithm is computationally solvable and some or all conditions (36), (37) do not hold is false. $\square$

The condition (37b) refers directly to the form of (12) and ensures reversibility of (15). In other words, (15) does not exist if (37b) does not hold for all its perturbed rotations. So, if the decomposition (15) exists, it will be found by the algorithm presented in Sec. 2.3.

The conditions (37a) and (37b), in Lemma 1, can be translated on elements of X which is gathered as the following

Theorem 1. A real $N \times N$ matrix X can be decomposed into (17) by the algorithm presented in Sec. 2.3 if and only if for each step m the following conditions hold

$$|X_{m+1:N, m+1:N}| \neq 0, \quad (40)$$

$$\begin{vmatrix} x_{m,m} & x_{m,m+1} & \cdots & x_{m,N} \\ 0 & x_{m+1,m+1} & \cdots & x_{m+1,N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & x_{i-1,m+1} & \cdots & x_{i-1,N} \\ x_{i,m} & x_{i,m+1} & \cdots & x_{i,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,m} & x_{N,m+1} & \cdots & x_{N,N} \end{vmatrix} \neq 0 \quad (41a)$$

for $i = m+2, \dots, N-1$ and

$$\begin{vmatrix} x_{m,m} & x_{m,m+1} & \cdots & x_{m,N} \\ 0 & x_{m+1,m+1} & \cdots & x_{m+1,N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & x_{N-1,m+1} & \cdots & x_{N-1,N} \\ x_{N,m} & x_{N,m+1} & \cdots & x_{N,N} \end{vmatrix} \neq 0. \quad (41b)$$

Proof. From (35), it follows that the conditions (37) are equivalent to

$$D_k \neq 0 \text{ for } k = m, \dots, N-1. \quad (42)$$

By Cramer's rule, (22) can be solved by

$$f_i = \frac{|x_i^T|}{|x_r^T|}, \quad (43)$$

where x_i^T is the matrix formed by replacing the i -th column of x_s^T by x_r^T in (22). Eq. (43) can be also represented using $|x_i^T| = -(-1)^{i+2m} M_{i+m,m}$ and $|x_r^T| = M_{m,m}$, where $M_{i+m,m}$ and $M_{m,m}$ are minors of $X_{m:N, m:N}$. As the result, (29) is given by

$$D_k = \frac{x_{m,m}(-1)^{2m} M_{m,m} + \sum_{j=N-k+m+1}^N x_{j,m}(-1)^{j+m} M_{j,m}}{M_{m,m}} \quad (44)$$

for $k = m+1, \dots, N-1$. $M_{m,m} \neq 0$ by (40). The numerator of (44) is the Laplace expansion of the submatrix $X_{m:N, m:N}$ with $x_{l,m}$ replaced by zeros for $l = m+1, \dots, N-k+m$, which are given in explicit form by (41). $\square$

4. ORTHOGNAL MATRIX CASE

It is possible to transform Givens rotations (2) to perturbed rotations (12) and the diagonal matrix using (39). By (39), starting from the first left Givens rotation in (5) with the identity matrix on the left, one can convert it into the perturbed rotation followed by the diagonal matrix. In the similar way, one can apply iteratively (39) to subsequent Givens rotations taking into account the previously obtained diagonal matrix. As the result, one obtains (17). In conclusion, we can formulate the following

Proposition 1. Every orthogonal matrix can be represented as a product of the perturbed rotations and a diagonal matrix.

5. NUMERICAL EXAMPLE

Suppose

$$X = \begin{bmatrix} 2 & 4 & 8 & 2 \\ 10 & 5 & 3 & 1 \\ 7 & 5 & 9 & 3 \\ 1 & 5 & 7 & 1 \end{bmatrix}. \quad (45)$$

Let us apply the algorithm presented in Sec. 2.3 to (45).

- $m = 1$. $F = \begin{bmatrix} -0.3 & 0.6 & 0.5 \end{bmatrix}^T$, $D_1 = 2$, $D_2 = 1.5$, $D_3 = -2.7$, $P_1 = 1$, $P_2 = 0.75$, $P_3 = -1.35$,

$$c_{1,4} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0 & 1 \end{bmatrix} \quad c_{1,3} = \begin{bmatrix} 1 & 0 & 0.8 & 0 \\ 0 & 1 & 0 & 0 \\ 3.5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$c_{1,2} = \begin{bmatrix} 1 & 0.2 & 0 & 0 \\ 5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (46a)$$

$$\begin{bmatrix} \ddots & & & \\ & d_m & & 0 \\ & & \ddots & \\ 0 & & & d_n \\ & & & & \ddots \end{bmatrix} G_{m,n}(\phi) = c_{m,n} \begin{pmatrix} -d_n \sin(\phi) & d_m \sin(\phi) \\ d_m \cos(\phi) & d_n \cos(\phi) \end{pmatrix} \begin{bmatrix} \ddots & & & \\ & d_m \cos(\phi) & & 0 \\ & & \ddots & \\ 0 & & & d_n \cos(\phi) \\ & & & & \ddots \end{bmatrix} \quad (39)$$

- $m = 2$. $F = \begin{bmatrix} 0.45 & 0 \end{bmatrix}^T$, $D_2 = 5$, $D_3 = 5$, $P_2 = 1$, $P_3 = 1$,

$$c_{2,4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0.8 & 0 & 1 \end{bmatrix} c_{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.45 & 0 \\ 0 & 0.2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (46b)$$

- $m = 3$. $F = -2.(7)$, $D_3 = 6.(6)$, $P_3 = 1$,

$$c_{3,4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2.(7) \\ 0 & 0 & 0.24 & 1 \end{bmatrix} E = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 6.(6) & 0 \\ 0 & 0 & 0 & -0.8 \end{bmatrix} \quad (46c)$$

The obtained $c_{m,n}$ and E in (46) are the solution to the decomposition of (45) into (17), i.e.

$$X = c_{1,4} c_{1,3} c_{1,2} c_{2,4} c_{2,3} c_{3,4} E \quad (47)$$

6. BACKWARD ERROR ANALYSIS

The algorithm presented in Sec. 2.3 requires solving of the system of linear equations (22). So, the sensitivity of the algorithm depends on a condition number $\kappa(F)$, see [4, Sec. 2.6.2]. Moreover, the subsequent steps of the algorithm will compute using previously contaminated results. So, the sensitivity analysis here is far more complicated than in the case of original Givens QR [6, Sec. 19.6]. To get an estimate of round-off errors for the algorithm, a series of simulations have been performed. A set of random square matrices $X = \{x_{i,j}\}$ of size $N \times N$ has been generated with numbers from a Gaussian distribution: 10.000 for each $N = 5, 10, 20, 30$ and 1000 for $N = 40$. All obtained matrices have been normalized to $|X|_1 = 1$. The algorithm in Sec. 2.3 has been applied to each X using finite double floating precision. Then, the results of the decomposition, namely $c_{m,n}$ and E in the right hand-side of (17), have been multiplied back to form a backward estimate $\hat{X} = \{\hat{x}_{i,j}\}$. During that step, computations were conducted with increased quadruple floating point precision in Scilab [7] using modified DD_QD MuPAT toolbox [8]. For each decomposition, the relative backward error ε_{rel} has been computed, given by

$$\varepsilon_{rel} = \max_{\forall x_{i,j} \neq 0} \frac{|x_{i,j} - \hat{x}_{i,j}|}{|x_{i,j}|}. \quad (48)$$

Histograms of ε_{rel} for each N are presented in Fig. 1-4.

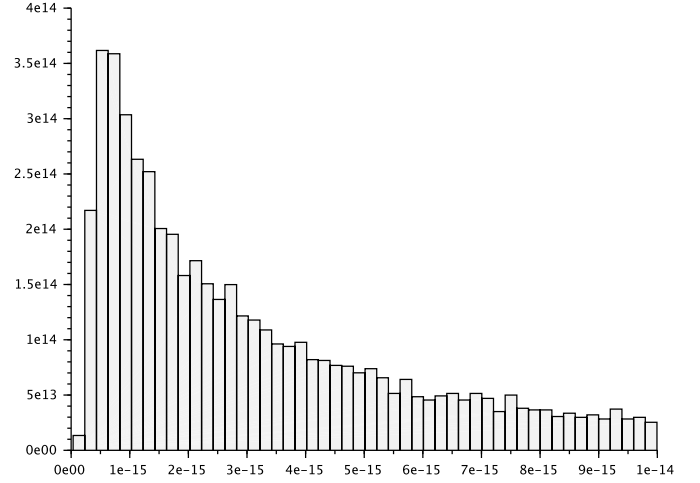


Fig. 1. Histogram of relative backward errors for X of size 5×5 .

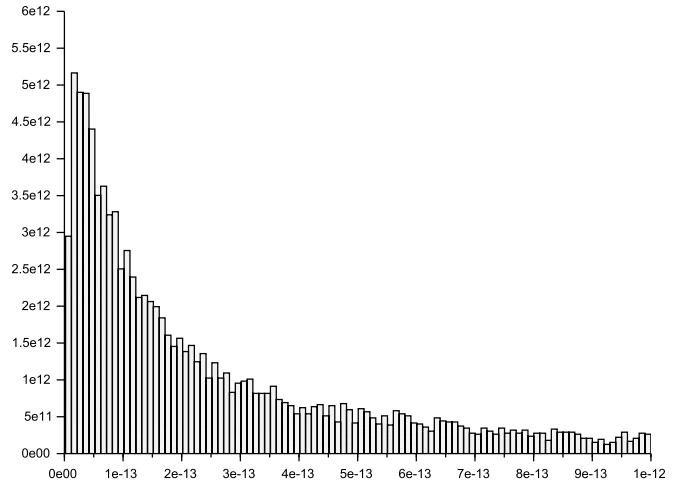


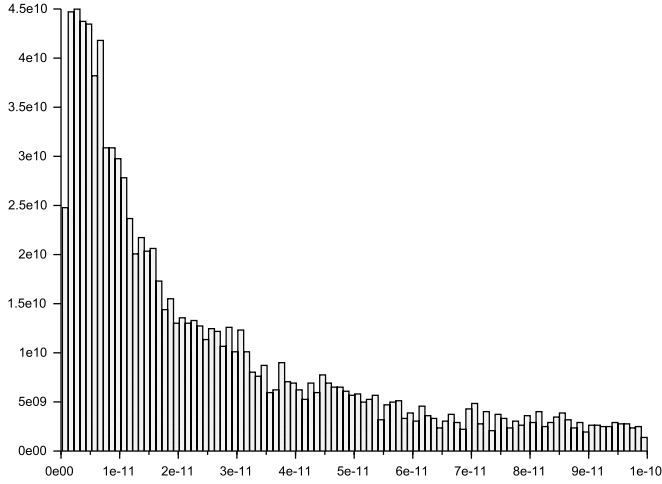
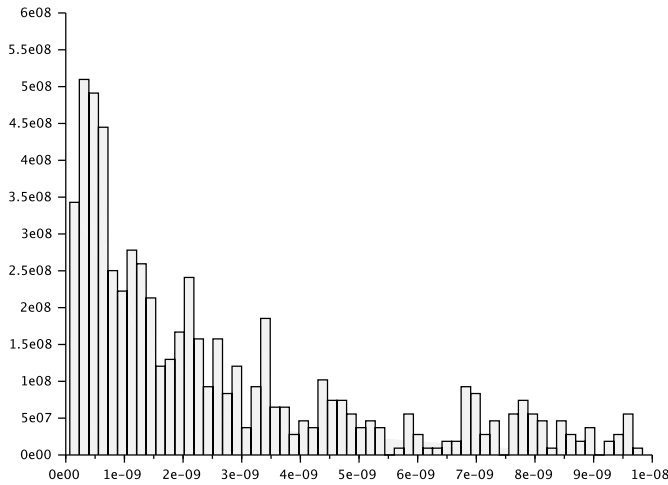
Fig. 2. Histogram of relative backward errors for X of size 10×10 .

7. CONCLUSIONS

In the paper, the decomposition algorithm of the real non-symmetric matrix X is presented by means of matrices similar to the Givens rotations, provisionally called perturbed rotations. We are not aware of any similar technique published in the matrix algebra books or papers. Possible applications of the method would cover matrix algebra (properties, equations solving, matrix inversion) or linear systems (synthesis, implementation, identification). Not every matrix can be decomposed in that way. Conditions for the existence of the decomposition for each m -step of the algorithm are presented as

Table 2. Histogram peaks $\tilde{\epsilon}$ generated by polynomials.

N	5	10	20	40
$\tilde{\epsilon}_2(N)$	7.2608e-12	-1.2688e-11	9.0527e-12	2.9909e-10
$\tilde{\epsilon}_3(N)$	5.3569e-16	1.6989e-14	2.7e-12	3e-10

**Fig. 3.** Histogram of relative backward errors for X of size 20×20 .**Fig. 4.** Histogram of relative backward errors for X of size 40×40 .**Table 1.** Histogram peaks $\tilde{\epsilon}$ of relative backward errors.

N	5	10	20	40
$\tilde{\epsilon}$	5.3e-16	1.7e-14	2.7e-12	3e-10

Thrm. 1. The backward errors histograms, in Fig. 1-4, have peaks $\tilde{\epsilon}$ whose values are collected in Tab. 1. For subsequent

N , $\tilde{\epsilon}$ clearly increases. To estimate the rate, *datafit* function from Scilab has been used to fit quadratic and cubic functions to data in Tab. 1, resulting in the following polynomials:

- $\tilde{\epsilon}_2(N) = 5.3e-16(775.34N^2 - 19158N + 90106)$
- $\tilde{\epsilon}_3(N) = 5.3e-16(25.277N^3 - 851.36N^2 + 8353.1N - 23640)$.

Evaluating $\tilde{\epsilon}_2(N)$ and $\tilde{\epsilon}_3(N)$ for N in Tab. 1, one gets results collected in Tab. 2. Comparing values of $\tilde{\epsilon}$ in Tab. 1 and Tab. 2 one can see that $\tilde{\epsilon}_2(N)$ generates incorrect values but $\tilde{\epsilon}_3(N)$ recreates closely the peaks in the histograms. The conclusion is that $\epsilon_{rel} \approx O(N^2)$ but it might be $\epsilon_{rel} \approx O(N^3)$. However, it is only a presumption which requires algebraic backward error analysis of the algorithm planned as a future work. It is also interesting if the existence of the decomposition would be determined directly from X , instead of at the beginning of each m -step of the algorithm, like in Thrm. 1. Another problem arises how to effectively implement multiplications by the perturbed rotations using finite precision computations, which would be at a service in digital signal processing.

REFERENCES

- [1] J. G. F. Francis, “The QR Transformation A Unitary Analogue to the LR Transformation—Part 1,” *Comput. J.*, vol. 4, no. 3, pp. 265–271, 01 1961, doi: 10.1093/comjnl/4.3.265.
- [2] J. G. F. Francis, “The QR Transformation—Part 2,” *Comput. J.*, vol. 4, no. 4, pp. 332–345, 01 1962, doi: 10.1093/comjnl/4.4.332.
- [3] V. Kublanovskaya, “On some algorithms for the solution of the complete eigenvalue problem,” *USSR Comput. Math. Math. Phys.*, vol. 1, no. 3, pp. 637–657, 1962, doi: 10.1016/0041-5553(63)90168-X.
- [4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, Maryland 21218: The Johns Hopkins Univ. Press, 2013.
- [5] R. Wirski, “Synthesis and realization of two-dimensional separable denominator orthogonal systems via decomposition into 1-D systems,” *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 66, no. 11, pp. 4309–4322, 2019, doi: 10.1109/TCSI.2019.2927673.
- [6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.
- [7] “The SCILAB homepage,” [Online]. Available: <https://www.scilab.org>.
- [8] “Multiple precision arithmetic ATOMS toolbox,” [Online]. Available: https://atoms.scilab.org/toolboxes/DD_QD.