N-dimensional Rotation Matrix Generation Algorithm

Article · March 2018

CITATIONS
9

READS
1,133

1 author:

Ognyan Zhelezov
Nikola Vaptsarov Naval Academy
5 PUBLICATIONS **9** CITATIONS

SEE PROFILE

N-dimensional Rotation Matrix Generation Algorithm

Ognyan Ivanov Zhelezov[1]

[1] Dep. of Development, firm Data Consulting, Sofia, 1000, Bulgaria

Abstract This article presents a new algorithm for generation of N-dimensional rotation matrix M, which rotates given N-dimensional vector X to the direction of given vector Y which has the same dimension. Algorithm, named N-dimensional Rotation Matrix Generation Algorithm (NRMG) includes rotation of given vectors X and Y to the direction of coordinate axis x_1 using two-dimensional rotations. Matrix M is obtained as multiplication of matrix M_X and inverse of matrix M_Y, which rotates given vectors to the direction of axis x_1. Also examined is the possibility to perform parallel calculations of two-dimensional rotations.

Keywords Mathematics of computing , Mathematical analysis , Numerical analysis , Computations on matrices

1. Introduction

High-dimensional spaces frequently occur in mathematics and the sciences, in example N-dimensional feature space, which presents input signals of neural network or collection of N-dimensional parameters for multidimensional data analysis. Rotation is one of rigid transformations in geometrical space, which preserves length of vectors and can be presented using matrix operation like Y = M.X , where X and Y are input and output vector respectively, and M is rotation matrix. This article proposes an N-dimensional rotation matrix generation algorithm for given input and output vector.

2. Definition of the Task

Let's say that we have two N-dimensional vectors X and Y, having the same dimension, X, Y $\in R^N$. We want to obtain a rotation matrix M that satisfies the equation.

$$\tilde{Y} = M.X \tag{1}$$

where $\tilde{Y}$ has the same norm as X and the same direction as Y, i.e. $\|\tilde{Y}\| = \|X\|$ and $\cos(Y, \tilde{Y}) = 1$.

One of possibilities to generate rotation matrix M includes the following sequence of operations:

1) Obtain two orthogonal vectors in the plane, in which lies the two given vectors using Gramm-Schmidt procedure [2], [3], [16].
2) Enlarge this 2-dimensional basis to N- dimensional basis B2, in which given vectors X, Y are transformed to $\overline{X}$ and $\overline{Y}$.
3) Create matrix of Givens rotation G(1, 2, θ) for R^N, which makes rotation of vector $\overline{X}$ in $\overline{x}_1\overline{x}_2$ -plane to the direction of vector $\overline{Y}$.
4) Obtaining rotation matrix M as

$$M = P^{-1}.G(1,2,\theta).P \tag{2}$$

where P is matrix, which transforms initial basis B to basis B2.

Another way to generate rotation matrix M is to use Householder Reflection [4], [10], [17], [18]. If X and Y are vectors with the same norm $\|X\| = \|Y\|$, there exists an orthogonal symmetric matrix P such that $Y = P.X$ where $P = I - W.W^T$ and $W = (X - Y)/\|X - Y\|$. Matrix P is matrix of reflection (not a rotation) because det P = -1, (which gives the name to method) that's why to obtain matrix of rotation M, for which det M = 1, have to be performed two subsequent reflections. Matrix of rotation can be obtained as multiplication of two matrix of reflection P1 and P2 as M = P1.P2 .

This article presents a new algorithm for generation of N-dimensional rotation matrix, which rotates given vector X to the direction of given vector Y. Algorithm, named N-dimensional Rotation Matrix Generation Algorithm (NRMG) includes the following sequence of operations:

1) Obtaining rotation matrix M_X, which rotates given vector X to the direction of axis $\vec{x}_1$.
2) Obtaining rotation matrix M_Y, which rotates given vector Y to the direction of axis $\vec{x}_1$.
3) Obtaining rotation matrix M as multiplication of M_X by inverse matrix of M_Y given as $M = My^{-1}.Mx$.

Below this three operations will be described subsequently

3. Rotation of Given Vector X to the Direction of Axis x_1

Rotation of given vector X to the direction of one of coordinate axes (e.g. axis $\vec{x}_1$) can be performed by subsequent multiplications by Givens matrices [1] as follow:

$$X_{(N-1)} = \prod_{k=N-1}^{1} G(k, k+1, \theta_k).X = M_X.X = [r_X, 0, \ldots 0, 0]^T \quad (3)$$

Givens matrices G(k, k+1, θk), k = N-1, n-2, … 1 are defined as follows [1], [2]:

G(k, k+1, θk) =

$$\begin{vmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & 0 & \cdots & C_k & -S_k & \cdots & 0 & 0 \\ 0 & 0 & \cdots & S_k & C_k & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \iddots & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \end{vmatrix} \quad (4)$$

where θk is the angle of rotation and coefficients Ck=cos(θk) and Sk=sin(θk) appears at the intersections of k-th and k+1 -th rows and columns.

Every one multiplication of vector by Givens matrix G(k, k+1, θk) performs rotation of its projection in coordinate plane (x_k, x_{k+1}) which changes values only of vector's coordinates x_k, x_{k+1} to $\bar{x}_k, \bar{x}_{k+1}$ .This multiplication can be presented as multiplication of coordinates by sub matrix A(k, k+1, θk) as follows:

$$\begin{vmatrix} \bar{x}_k \\ \bar{x}_{k+1} \end{vmatrix} = A(k, k+1, \theta_k). \begin{vmatrix} x_k \\ x_{k+1} \end{vmatrix} = \begin{vmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{vmatrix} . \begin{vmatrix} x_k \\ x_{k+1} \end{vmatrix} \quad (5)$$

Taking in consideration that multiplication with Givens matrix G(k, k+1, θk) changes only values of vector's coordinates x_k, x_{k+1} (to $\bar{x}_k, \bar{x}_{k+1}$), schema of multiplication by Givens matrix G(k, k+1, θk) can be presented as operator for two-dimensional rotation as follows:

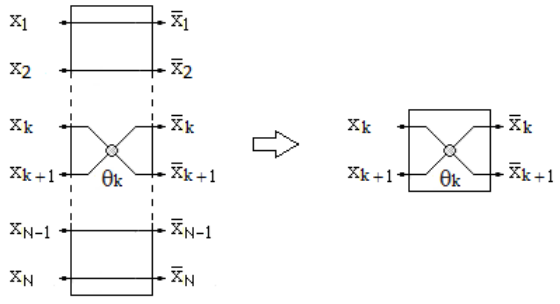

Figure 1 Schema of two-dimensional rotation, performed by Givens matrix G(k, k+1, θk)

The target of multiplication by Givens matrix G(k, k+1, θk) is to set to zero coordinate $\bar{x}_{k+1}$. It is easy to find that equation $\bar{x}_{k+1} = 0$ and equations (5) are satisfied simultaneously when $\sin(\theta_k)$ and $\cos(\theta_k)$ are calculated using formulas:

$$\begin{cases} \sin(\theta_k) = -\dfrac{x_{k+1}}{\sqrt{x_k^2 + x_{k=1}^2}}, \\ \cos(\theta_k) = \dfrac{x_k}{\sqrt{x_k^2 + x_{k=1}^2}} \text{ if } x_k^2 + x_{k=1}^2 > 0 \\ \sin(\theta_k) = 0, \cos(\theta_k) = 1 \text{ if } x_k^2 + x_{k=1}^2 = 0 \end{cases} \quad (6)$$

Thus searched matrix M_X can be calculated as multiplication of Givens matrices $G(N-1, N-2, \theta_{N-1}), G(N-2, N-3, \theta_{N-2}), \ldots, G(1, 2, \theta_1)$ as follows:

$$M_X = \prod_{k=1}^{N-1} G(N-k, N-k+1, \theta_{N-k}) \quad (7)$$

where coefficients of rotation $\sin(\theta_k)$ and $\cos(\theta_k)$ are calculated using formulas (6). Calculation of angles of two-dimensional rotations θ_k, k = N-1, N-2, .., 1 really is not needed. If $x_k^2 + x_{k+1}^2 = 0$ then corresponding Givens matrix is equal to Identity matrix G(k, k+1, θk) = I – no rotation.

Schema for rotation of vector X to the direction of axis $\vec{x}_1$ using two-dimensional rotations can be presented as follows:

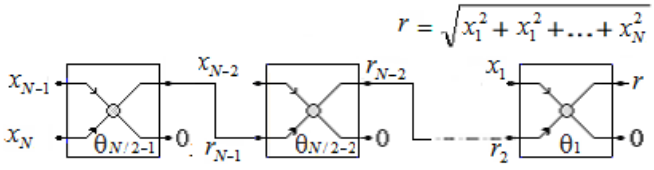

Figure 2 Schema for rotation of vector X to the direction of axis $\vec{x}_1$

Every one block for two-dimensional rotation (as it was given on Fig. 3) presents rotation of two-dimensional vector $[x_k, x_{k+1}]^T$ to the direction of axis $\vec{x}_k$ k=1,2,…N-1. This two-dimensional rotation is the base operation of proposed algorithm.

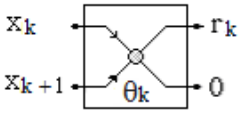

Figure 3 Block of base operation - rotation of two-dimensional vector $[x_k, x_{k+1}]^T$ to the direction of axis $\vec{x}_k$

As it can be seen from (3) and Fig. 2, rotation of N-dimensional vector to the direction of axis $\vec{x}_1$ needs

execution of N-1 subsequent base operations. Thus if one base operation has execution time Tb , then rotation of given N-dimensional vector X to the direction of axis $\vec{x}_1$ will take execution time (N-1).Tb . As it is described below, this time can be reduced using parallel execution of base operations.

4. Description of NRMG Algorithm

NRMG algorithm for generation of N-dimensional rotation matrix M, which rotates given vector X to the direction of given vector Y consist the following operations:

1) Obtaining rotation matrix M_X, which rotates given vector X to the direction of axis $\vec{x}_1$
2) Obtaining rotation matrix M_Y, which rotates given vector Y to the direction of axis $\vec{x}_1$.
3) Obtaining rotation matrix M, which rotates given vector X to the direction of given vector Y as multiplication of matrix M_X and inverse matrix of M_Y , as follows:

$$M = M_Y^{-1}.M_X \tag{8}$$

It is important to note that NRMG algorithm do not need vectors X and Y to have the same norm (as it is needed for Householder Reflection i.e.). Multiplication of given vector X by matrix of rotation M will give resultant vector $\tilde{Y}$, which will have norm of vector X, but direction of vector Y.

Time complexity of proposed algorithm depends of algorithm, used for calculation of coefficients $\sin(\theta_k)$ and $\cos(\theta_k)$ of two-dimensional base operations. This calculation depends of practical realization and is not a subject of this article.

NRMG algorithm can be used for different reversible calculations. An example that uses NRMG algorithm for transformation of images appears below (6).

5. Increasing Performance Using Parallel Execution of Two-Dimensional Rotations

Parallel execution of two-dimensional rotations (base operations of algorithm) is one way to increase calculation performance. This possibility is proposed in [7],[8] in relation of matrix decomposition. As this parallelism gives significant decrease of subsequent calculations, below will be presented one realization of this approach for proposed algorithm without pretensions of novelty.

Let's have N-dimensional vector $X = [x_1, x_2, \ldots, x_n]^T$. Vector X can be presented as a sum of two vectors X1 and X2, every one of which has half coordinates of value zero and half coordinates, that have the same values as the coordinates of given vector X, as follows:

$$\begin{aligned} X1 &= [x_1, x_2, \ldots, x_{N/2}, 0, \ldots, 0]^T \\ X2 &= [0, 0, \ldots 0, x_{N/2+1}, x_{N/2+2}, \ldots, x_N]^T \\ X &= X1 + X2 \end{aligned} \tag{9}$$

Rotation of vector X to the direction of axis $\vec{x}_1$ by multiplication with matrix of rotation M will give resultant vector $X12_{(N/2)}$ which is the sum of rotated to the direction of axis $\vec{x}_1$ vectors X1 and X2 as follows:

$$\begin{aligned} X12_{(N/2)} &= M.X = M.(X1 + X2) = X1_{(N/2)} + X2_{(N/2)} \\ &= \prod_{k=1}^{N-1} G(N-k, N-k+1, \theta_{N-k}).X1 \\ &+ \prod_{k=1}^{N-1} G(N-k, N-k+1, \varphi_{N-k}).X2 \end{aligned} \tag{10}$$

For zero coordinates of X1 and X2 corresponding matrices of base operations G(k, k+1, θk) are equal to identity matrix I (i.e. no rotation). This shows that coordinate planes, in which are rotated projections of X1 are different from those, in which are rotated X2. As a resultant vector we get:

$$\begin{aligned} X12_{(N/2)} &= \prod_{k=1}^{N/2-1} G(N/2-k, N/2-k+1, \theta_{N/2-k}).X1 \\ &+ \prod_{k=1}^{N/2-1} G(N-k, N-k+1, \theta_{N-k}).X2 \end{aligned} \tag{11}$$

where in second sum symbol for angle or rotation φ has been replaced with θ for convenience.

Vector $X12_{(N/2)}$ has only two non-zero coordinates - x_1 and $x_{N/2+1}$, so it lie in coordinate plane $(x_1, x_{N/2+1})$. Vector $X_{(N/2)}$ to the direction of axis $\vec{x}_1$ can be obtained by only one two-dimensional rotation of vector $X12_{(N/2)}$ in this coordinate plane as follows:

$$X_{(N/2)} = G(1, N/2+1, \theta_{N/2}).X12_{(N/2)} = [r_X, 0, \ldots, 0, \ldots, 0] \tag{12}$$

Schema below presents obtaining of vector $X_{(N/2)} = [r_X, 0, \ldots, 0]^T$. using rotation of vector $X12_{(N/2)}$ to the direction of axis $\vec{x}_1$.

As it can be seen from (8)÷(10) and from schema on Fig. 4, after one division of given vector X the number of subsequent base operations is decreased to N/2 regardless that the number of base operations remain the same – N-1. It is easy to find that separation of given vector X to more then two parts will give more decreasing of subsequent base operations.

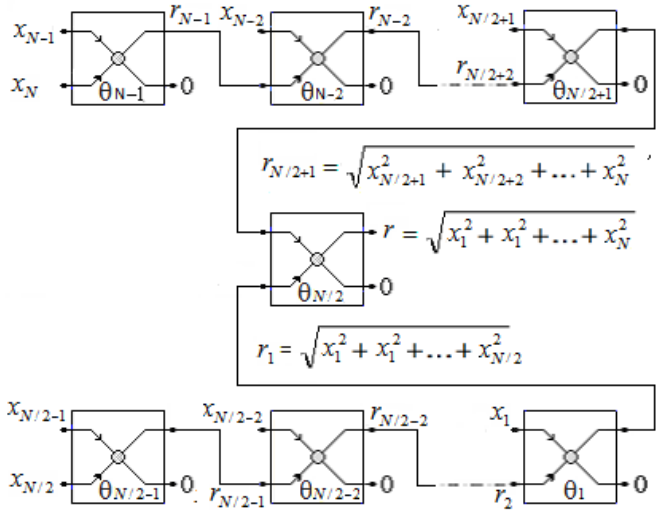

Figure 4 Schema for rotation of vector X, separated into two parts, to the direction of axis $\vec{x}_1$

The limit of subsequent divisions is reached when every one part has only two elements. In example if given vector has dimension N=8 it can be separated to 4 parts of two elements. Rotation of every one part of vector (as two-dimensional vector) to the direction of one of coordinates gives resultant vector, that has N/2 non-zero elements. The same way this sub-vector of dimension N/2 can be separated to N/4 parts of two elements and rotation of every one of this parts gives resultant vector with N/4 non-zero elements. It's easy to find that if $N=2^P$, $P \in \mathbb{N}$ (set of all natural numbers) then the number of subsequent rotations, needed to obtain vector with only one non-zero element, is equal to $\log_2 N$.

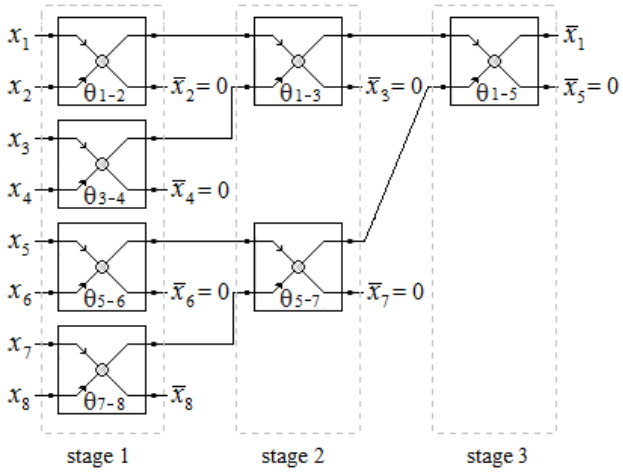

Figure 5 Schema for accelerated rotation of 8-dimensional vector to the direction of axis $\vec{x}_1$

In example if vector X has dimension N=8, the number of needed subsequent rotations will be $\log_2 8 = 3$. Schema for accelerated rotation of 8-dimensional vector to the direction of axis $\vec{x}_1$ will look as on figure above

It is important to note the following:

1) Accelerated rotation (AR) of given vector to the direction of axis $\vec{x}_1$ can be applied not only for vectors, that have dimension $N=2^P$, $P \in N$, but for every one vector with dimension greater then 2. In example if N=7 then difference between schema of AR and those, given above, will be only this, that in first stage will miss the last base operation and x7 will go straight to base operation on second stage.
2) The number NR of stages (in which parallel execution of base operations of AR have to be performed) is equal to $\lceil \log_2 N \rceil$ whereas number of all base operations is N-1 – the same as when all rotations are subsequent.
3) AR really increases calculation performance in case of parallel execution of base operations in stages. If base operations in stages are executed consecutively, the only advantage of AR is decreasing of accumulation of calculation errors (due to rounding e.g).

$$MS_1=\begin{vmatrix} C_{1,1} & -S_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{1,1} & C_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{1,2} & -S_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{1,2} & C_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{1,3} & -S_{1,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{1,3} & C_{1,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{1,4} & -S_{1,4} \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{1,4} & C_{1,4} \end{vmatrix}$$

$$MS_2=\begin{vmatrix} C_{2,1} & 0 & -S_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{2,1} & 0 & C_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{2,2} & 0 & -S_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{2,2} & 0 & C_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

$$MS_3=\begin{vmatrix} C_{3,1} & 0 & 0 & 0 & -S_{3,1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ S_{3,1} & 0 & 0 & 0 & C_{3,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Figure 6 Matrices of stages for AR of 8-dimensional vector

For high dimensional vectors AR proposes significant decreasing of subsequently executed base operations. In example if dimension of vectors is N=1024, the number of subsequently executed base operations is 1024-1, whereas

AR algorithm uses number of stages (which defines the number of subsequent base operations) NStages = $\log_2 1024 = 10$.

It is important to note, that rotation is reversible linear operation, which mean that AR is reversible linear operation too. Reversibility of rotation, in particularly AR, is the "key stone" of proposed NRMG algorithm.

When AR is used, rotation matrix M_X and M_Y can be presented as a multiplication of matrices MS_1, MS_2, … MS_{NR}, which correspond to the stages of rotation, shown in Fig. 5. In example for vector with dimension N=8 matrices of stages $MS_3.MS_2.MS_1$ will look as in Fig. 6. where $C_{1,1}$, $S_{1,1}$, $C_{1,2}$, $S_{1,2}$, $C_{1,3}$, $S_{1,3}$, $C_{2,1}$, $S_{2,1}$,$C_{2,2}$, $S_{2,2}$, $C_{3,1}$, $S_{1,3}$ are denotation of $\cos(\theta_{i,j})$ and $\sin(\theta_{i,j})$, index ***i*** is the number of stage, and ***j*** is the number of base operation.

As it can be seen, matrices of stages MSk, k=1, 2, …,$\lceil \log_2 N \rceil$ (except the last one) are not Givens matrices. They are matrices for parallel execution of two-dimensional rotations in corresponding coordinate planes as follows:

$$MS_k = \prod_{n=1}^{N/2^k} G\left(1+(n-1)2^k, 1+(2.n-1)2^{k-1}, \theta_{n,k}\right) \quad (13)$$

In NRMG that uses AR (which is not obligate), matrices of stages denote parallel execution of base operations. Schema, given below shows rotation of given 8-dimensional vector X to the direction of vector Y and obtaining rotation matrix M, which performs this rotation. Every one of matrices M_X and M_Y is calculated as multiplication of $\lceil \log_2 N \rceil$ matrices of stages for which coefficients Ci-j and Si-j are calculated using (6).

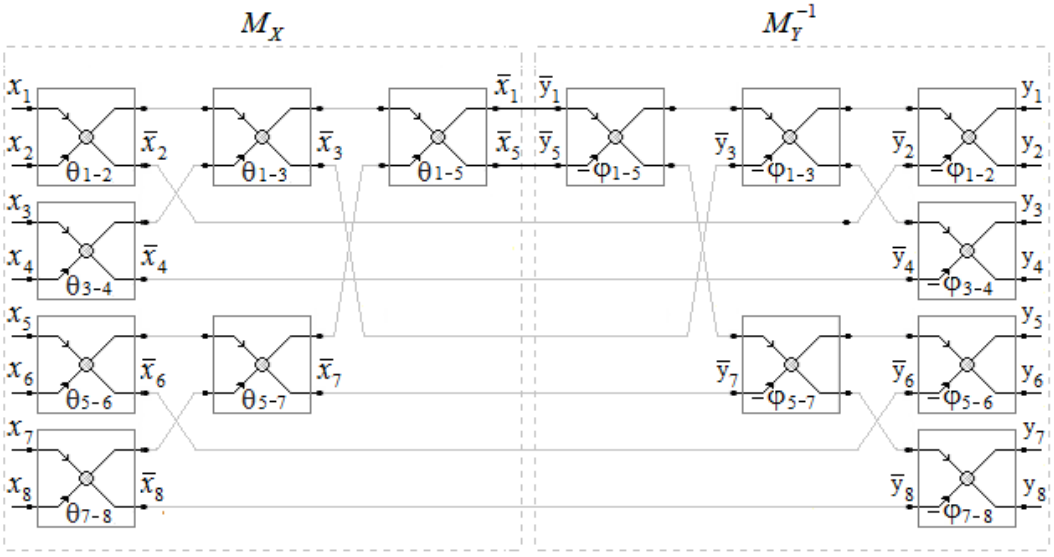

Figure 7 Schema for rotation of 8-dimensional vector X to the direction of vector Y using NRMG algorithm and AR.

6. Reversible Transformation of Images Using NRMG Algorithm

Down we will give one example of using NRMG algorithm for reversible image transformation.

Let's have two monochrome raster images I1 and I2 which have the same number of pixels N. As it is known [14], raster images are presented as dot matrix data structure, so the two given images can be presented as two N-dimensional vectors X and Y, every element of which presents brightness of one of pixels. Using NMRG algorithm can be obtained rotation matrix M, which rotates vector X to the direction of vector Y. Since rotation do not change the norm of the vector, resultant vector $\tilde{Y}$ will have the same norm as X. As a result, rotation will transform image I1 to one presentation of image I2 , brightness of every one pixel of which is equal to the brightness of corresponding pixel of image I2, scaled with coefficient $\|Y\|/\|X\|$. If X and Y have the same norm, vectors $\tilde{Y}$ and Y will be identical.

Has been created Matlab program to test proposed NRMG algorithm. Program consist function for accelerated rotation fnAR and code, which uses this function to obtain matrix M, which rotates given vector X to the direction of given vector Y.

```
function R = fnAR(X)
N = length(X);      %X have to be row vector (transposed)
R= eye(N);          %Initial rotation matrix = Identity matrix
step = 1;           %Initial step
while(step<N)       %Loop to create matrices of stages
   A= eye(N);
      n=1;
      while(n<=N-step)
            r2 = X(n)*X(n) + X(n+step)*X(n+step);
            if r2 > 0
```

```
            r = sqrt(r2);
            pcos = X(n)/r;
            psin = -X(n+step)/r;
            % Base 2-dimensional rotation
            A(n, n) = pcos;
            A(n, n+step) = -psin;
            A(n+step, n) = psin;
            A(n+step, n+step) =    pcos;
        end
        n=n+2*step;    % Move to the next base operation
    end;
    step = step*2;
    X=(A*X')';
    R= A*R;    % Multiply R by current matrix of stage A
end;
end;
```

Example 1 Code of Matlab function for accelerated rotation of vector X to the direction of axis x_1. Function returns matrix of rotation M_X

Code, that uses this function to obtain matrix M, which rotates given vector X to the direction of given vector Y is given below:

```
Mx = fnAR(X);
My = fnAR(Y);
M = My'*Mx;          %Obtaining rotation matrix M
Z = M*X';            %Obtain vector Z to the direction of Y
if round(Z'*100000)/100000==Y
  disp('Z and Y are identical');
end
```

Example 2 Code of Matlab, which creates matrix of rotation using fnAR function

For test data has been used the following two images:

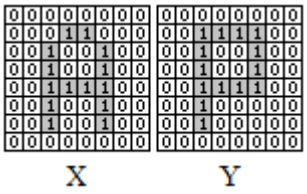

Figure 8 Test images

Images can be presented as 64-dimensional vectors as follows (written using Matlab Language syntax for row vectors):

X =[0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0];
Y =[0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

Test program, given above, obtain vector Z to the direction of vector Y using NRMG algorithm, and compare vectors Z and Y. Program displays message "Z and Y are identical", which shows that two vectors (and corresponding images) are identical at least with precision of 10^{-6}. It's important to note, that vectors X and Y have the same norm.

7. Conclusion

In this article, we presents a new algorithm for generation of N-dimensional rotation matrix M, which rotates given N-dimensional vector X to the direction of given vector Y which has the same dimension. Also examined is the possibility to perform parallel calculation of base operations - two-dimensional rotations. Algorithm can be used for different reversible calculations. It is included one example of image transformation that uses NRMG algorithm.

Proposed NRMG algorithm proves that

1) Every one matrix of rotation M, which rotates given vector X to the direction of given vector Y can be presented as $M=M_Y^{-1}.M_X$ where M_X rotates vector X to the direction of one of coordinate axes (e.g. axis x_1) and M_Y rotates vector Y to the direction of the same coordinate axis.
2) Every one rotation of N-dimensional vector can be performed by 2(N-1) two-dimensional rotations in N-1 coordinate planes.

As an advantage of NRMG algorithm to known solutions, (shortly described in p. 2), can be pointed a possibility to realise it as Discrete Linear System, elements in which performs base operations. This possibility is a result of the locality of data, used for base operations and independence of base operations inside one stage of rotation.

REFERENCES

[1] H. G. Golub, J. M. Ortega, 1993, Scientific Computing and Introduction with Parallel Computing, Academic Press, Inc., San Diego.

[2] G. A. Korn, T. M. Korn, 1961, Mathematical Handbook for Scientists and Engineers (1st ed.), New York: McGraw-Hill. pp. 55–79.

[3] H. Friedberg, A. Insel, L. Spence, 1997, Linear Algebra (3rd ed), Prentice Hall.

[4] D. A. Harville, 1997, Matrix Algebra from Statistician's Perspective, Softcover.

[5] B. Buchberger, 1985, Multidimensional Systems Theory - Progress Directions and Open Problems in Multidimensional Systems, Reidel Publishing Company.

[6] G. H. Golub, C. F. Van Loan, , 1996, Matrix Computations, 4rd edition. Johns Horkins University Press, Baltimore

[7] M. Cosnard, Y. Robert. Complexity of parallel QR factorization. J. ACM 33, 4 (August 1986), 712-723. DOI=10.1145/6490.214102, 1986.

[8] A. H. Sameh, D. J. Kuck. On Stable Parallel Linear System Solvers. J. ACM 25, 1 (January 1978), 81-91. DOI= http://dx.doi.org/10.1145/322047.322054, 1978.

[9] N. J. Higham, 1996, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelfia.

[10] G. W.Steward, 1976, The economical storage of plane rotations, Numer. Math, 25, 2 1976, 137-139

[11] Matlock, H., and Reese, L.C., 1960, Generalized solutions for laterally loaded piles., Journal of Soil Mechanics and Foundation, 86(5), 63–91.

[12] N. K. Bose, 1985, Multidimensional Systems Theory: Progress, Directions, and Open Problems. D.Reidel Publishing Co., Dordrecht, The Netherland.

[13] A. S. Householder, 1958, "Unitary triangularization of a nonsimetric matrix", J. ACM 5, 339-342, 1958.

[14] E. Anderson, 2000, "Discontinuous Plane Rotations and the Symmetric Eigenvalue Problem" LAPACK Working Note 150, University of Tennessee, UT-CS-00-454, December 4, 2000. page 2.

[15] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, 1995, "Computer Graphics, Principles and Practice", 2nd edition in C, Addison-Wesley, ISBN 0-201-84840-6.

[16] G. Strang, 2006, Linear Algebra and its Applications, Thomson Learning Ink, pages 69-135, ISBN 0-03-010567.

[17] St. Roman, Advanced Linear Algebra, second ed., 2005 Springer-Verlag, New York. pages 59-85, ISBN: 978-1-4757-2180-5

[18] J. E. Gentle, 2007, Matrix Algebra: Theory, Computations, and Applications in Statistics, Springer, page 180, ISBN 978-0-387-70872-0

[19] I. R. Shafarevich, A. Remizov, 2013, Linear Algebra and Geometry, Springer, pages 133-160, ISBN 978-3-642-30993-9