Experimental Testing of Simple Rotations in QR Factorization

1st Agnes Verne Bergholt
Fakultät für Elektro- und Informationstechnik
Technische Universität München
Munich, Germany
ge92jil@mytum.de

***Abstract*—A QR factorization of a matrix creates an orthogonal matrix Q and an upper triangular matrix R and can be done for any matrix. This is very useful tool and there are several ways to determine the factors Q and R. MATLAB uses a method with Householder reflections, which is known to be effective and stable. In [1] Diepold, Kissel and Dewilde propose a similar method using a rotation instead of a reflection. This paper looks further into their method and the accuracy of QR factorization based on these rotations. This will be done in comparison to the MATLAB implemented QR factorization which, as mentioned uses Householder reflections.**

I. Introduction

The QR factorization is a well-known tool with many applications including linear equations and least squares. Common ways to compute the QR factorization is through Householder reflections, Givens rotations and the Gram-Schmidt method. Another, not as used method is through simple rotations. Like the Householder reflections, simple rotations eliminates n-1 entries of a vector, but also possess the properties from being a rotation.

In this research, the aim is to do experimental testing of the simple rotations in comparison to the Householder reflections. More precisely looking at the accuracy of the use of the rotation and reflection in the QR factorization.

To do this, an overview of the QR factorization and the Householder reflections will be given. Further, the construction of the simple rotations is shown. As the main focus experiments on the accuracy of simple rotations will be performed and compared to the results for the MATLAB implemented QR factorization using Householder reflections. This will be done for different test matrices in different sizes.

II. QR Factorization

The QR factorization of a Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, gives the two factors $\mathbf{Q}$ and $\mathbf{R}$

$$\mathbf{A} = \mathbf{QR} = \begin{bmatrix} \mathbf{Q_1} & \mathbf{Q_2} \end{bmatrix} \begin{bmatrix} \mathbf{R_1} \\ \mathbf{0} \end{bmatrix} = \mathbf{Q_1 R_1} \tag{1}$$

Where $\mathbf{Q}$ is orthogonal and $\mathbf{R_1}$ is upper triangular. In the special case where $\mathbf{A}$ is square, $\mathbf{R}$ would be square as well, and the two last equations of 1 can be ignored.

III. Householder Reflections

A householder reflection $\mathbf{H}$, is a tool for computing the QR factorization of a matrix $\mathbf{A}$. $\mathbf{H}$ is on the form

$$\mathbf{H} = \mathbf{I} - \frac{2}{v^T v} v v^T \tag{2}$$

As a reflection, the matrix $\mathbf{H}$ satisfies, as expected, the properties of being orthogonal and with determinant -1. Which would mean that applied to a vector x, you would get a new vector y of equal length $\|x\|_2 = \|y\|_2$.

$$y = \mathbf{H}x \tag{3}$$

The householder reflections is a common tool for computing the QR factorization because of its ability to produce a vector y, which we can choose to be on the form

$$y = \begin{bmatrix} \pm\sqrt{x^T x} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4}$$

To do this we have to find a reflector $\mathbf{H}$ that creates a given y from a given x. This can be done by setting $v = x - y$. You will then get a reflection visualized in 1

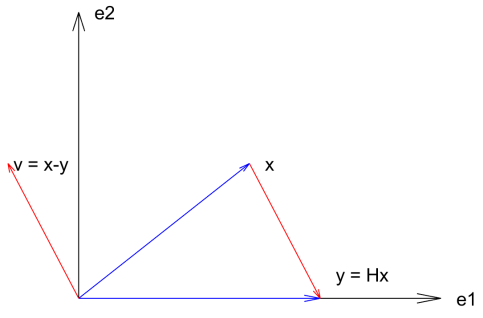

Fig. 1. Illustrating how the vector x is reflected onto a vector y of equal length along the first axis.

A. Householder reflections to compute QR factorization

This idea is used when computing the QR factorization of a matrix $\mathbf{A}$ with householder reflections. By premultiplying a householder matrix to $\mathbf{A}$, we can remove all the elements except the first in the first column of $\mathbf{A}$. This is done by using the first column as x and set y according to 4. By continuing to multiply by householder matrices, we can create an upper triangular matrix as depicted in 5, where the last matrix is the upper triangular one.

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & \cdot & \cdot \end{bmatrix} \xrightarrow{H_2} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \\ 0 & 0 & \cdot \end{bmatrix} \xrightarrow{H_3} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

From this, we get that:

$$H_3H_2H_1A = R \tag{6}$$

So that

$$(H_3H_2H_1)^T = Q \tag{7}$$

IV. Simple rotations

Another way to perform QR factorization is through Givens rotations. However, this only creates one zero element in the matrix per rotation. Therefore, by using simple rotations as proposed in [1] you get a method where you with rotations create zeros in the same manner as with Householder reflections. We denote the simple rotation by S. As a rotation we require orthogonality and that the determinant is equal to 1. Since we in a QR factorization need to induce zeros to create the upper triangular matrix R. We enforce the resulting vector $y = Sx$ to be on the form 4. This can be done through the rotation

$$S = \begin{bmatrix} x_1 & x_2^T \\ -x_2 & I - \frac{x_2 x_2^T}{1+x_1} \end{bmatrix} \tag{8}$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^n, x_1 \in \mathbb{R}, x_2 \in \mathbb{R}^{n-1} \tag{9}$$

A. Simple rotations to compute QR factorization

The algorithm for a QR factorization with use of simple rotation is similar to the one using Householder reflections. the only difference being that we need to build a new simple rotation instead of a Householder reflection. Again we see that

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \xrightarrow{S_1} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & \cdot & \cdot \end{bmatrix} \xrightarrow{S_2} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \\ 0 & 0 & \cdot \end{bmatrix} \xrightarrow{S_3} \begin{bmatrix} \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 0 & 0 & \cdot \\ 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Which results in the matrices R and Q

$$S_3S_2S_1A = R \tag{11}$$

$$(G_3G_2G_1)^T = Q \tag{12}$$

V. Experminetal testing

The following sections are based on experiments with random matrices induced with the MATLAB function $randn$, in dimensions $10x5$, $100x50$ and $300x70$. This gives matrices with elements drawn from the standard normal distribution.

In addition $10x10$, $50x50$ and $100x100$ dimensional matrices where the columns repeat cyclically were also used in the testing. The cyclic matrix is created through the *Matrix Computation Toolbox for MATLAB* [5] created by N. Higham. One cycle consist of the matrix $randn(N, k)$, where N is the dimension of the full matrix and k is $round(N/4)$. These columns are repeated until the matrix has the desired size. The cyclic matrices were used in order to see if the results were different for a specific structure.

The Hilbert matrix was also considered as a test matrix like in the QR algorithm comparison [3] done by Laurent Hoeltgen. Even though the Hilbert matrix is a famous test matrix, Higham [4] argued that it is not really suited as a test matrix because it is too ill-conditioned and not all elements can be stored accurately in floating-point arithmetic. Therefore, the Hilbert matrix was not used in this experimental testing.

For testing, the QR factorization was implemented with simple rotations as proposed in [1] and using the MATLAB QR implementation qr with Householder reflections. For each size of the test matrices, both the random and the cyclic, the experiments were executed 100 times. The results are given through the mean and variance of the error for these 100 iterations.

A. Orthogonality of Q

A first test on how well the simple rotation works in comparison to the householder reflection in creating a QR factorization, is exploiting the fact that the Q matrix should be orthogonal.

An orthogonal matrix times its transpose is equal to the identity matrix. Therefore, testing is done by taking the difference between the identity matrix and QQ^T. Since this should ideally be the zero matrix, the elementwise l2-norm of the difference is used as a measure on the accuracy.

As seen in the plots 2 and 3, both the Householder reflections and the simple rotations produce Q matrices which are very close to orthogonal. However, we can see that the standard deviation for the $100x100$ cyclic matrix is significantly larger than for the other test matrices even though the mean error is still small. The relfections and the rotations both work better for smaller matrices, both for cyclic matrices and random matrices. We can also observe that the gap between

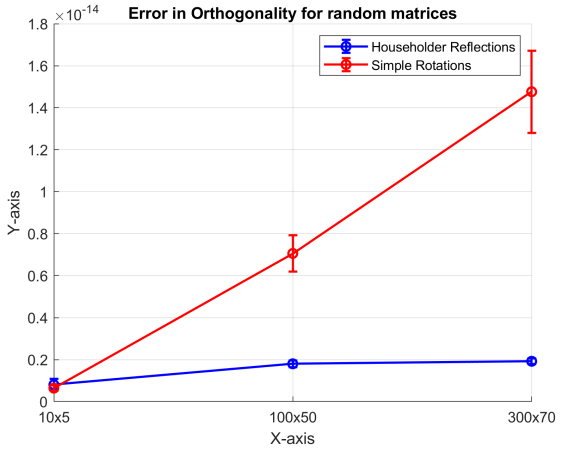

Fig. 2. Plotting the mean error and standard deviation in orthogonality of Q for randomly induced matrices

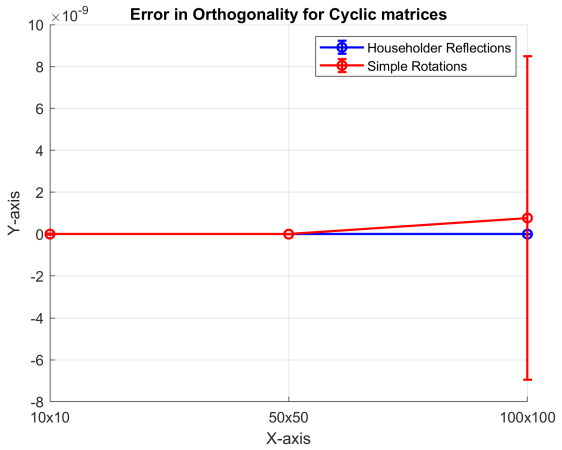

Fig. 3. Plotting the mean error and standard deviation in orthogonality of Q for cyclic matrices

accuracy increases with increasing dimensions. The simple rotation becomes worse in comparison to householder reflections for bigger dimension both for the mean and variance and both for the random matrices and the cyclic matrices.

B. Comparing the column space of Q and A

In this section and the following, A will be used as notation for the test matrix.

Another interesting property to check is how similar the column space of Q and A are. In theory they should be equal.

This can easily be shown:

y is a part of the column space of A if and only if there exists an x such that $Ax = y$.

If we factorize A, we can rewrite this as: $QRx = y$. Substituting w for Rx, we get that y is in the column space of Q if w exists with this property. From this we can conclude that the column space of A is a subspace of the column space of Q. In a similar fashion we can show that the column space of Q is a subspace of the column space of R. Therefore the column space of Q and R are equivalent.

To test this the MATLAB build-in function $orth$ was used to calculate the column space of A and of Q. Then to look at how similar the column spaces are, the angle between them is evaluated. This is done through the MATLAB build-in function $subspace$.

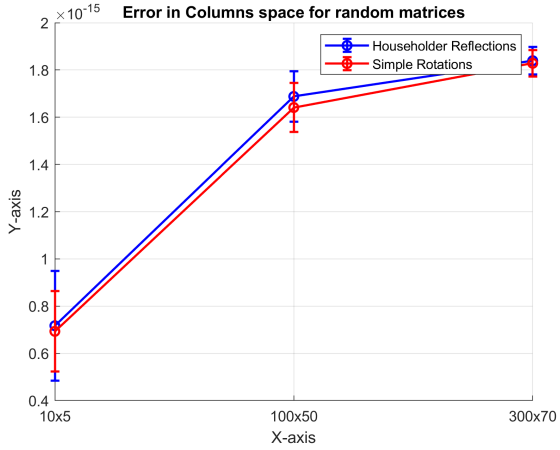

Fig. 4. Plotting the mean error and standard deviation for the column space of Q for randomly induced matrices

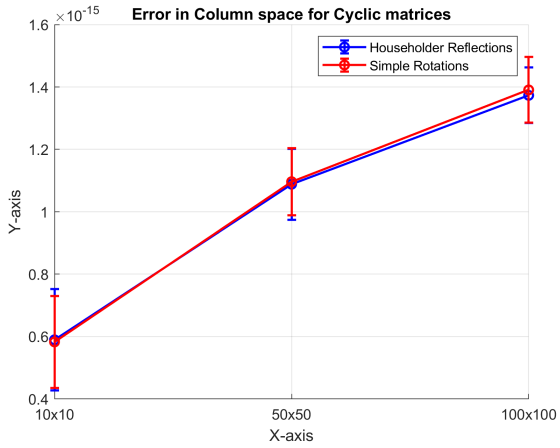

Fig. 5. Plotting the mean error and standard deviation for the column space of Q for cyclic matrices

Looking at plots 4 and 5, we observe that that both the simple rotations and the Householder reflections create Q-matrices with very similar column space as the original matrix A. This is true for both the random matrices and the cyclic matrices. The two plots show that whether the simple rotations or the Householder reflections perform better seems to vary with size. Since the errors also are very small, it is hard to draw any conclusions from this. To look further into this, more test matrices of different sizes and structures could be evaluated.

C. Testing the projections of A onto col(Q) and null(Q)

As a final test, we evaluate the projections of A onto $col(Q)$ and $null(Q)$. As discussed, Q theoretically has the same

column space as A. Therefore, projecting A onto the column space of Q should just give A. Similarly, the nullspace of Q should be the nullspace of A, making the projection of A onto the nullspace of Q zero.

To test these properties, the projections were executed by taking $Q(Q^TQ)^{-1}Q^TA$ and $(I - Q^T(QQ^T)^{-1}Q)A$ respectively. As in the first experiment, the l2-norm was taken of the difference between the theoretical answer and the computed result. Which means the difference between identity and $Q(Q^TQ)^{-1}Q^TA$ and the difference between the zero matrix and $(I - Q^T(QQ^T)^{-1}Q)A$.

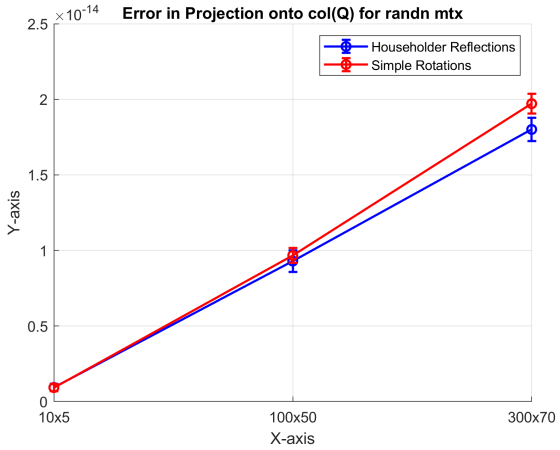

Fig. 6. Plotting the mean error and standard deviation in the projection of A onto col(Q) for randomly induced matrices

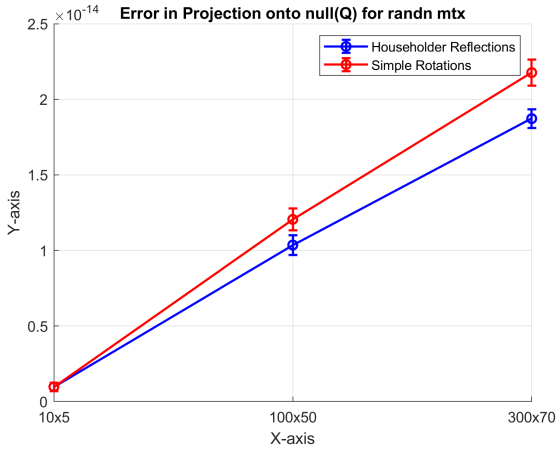

Fig. 7. Plotting the mean error and standard deviation in the projection of A onto null(Q) for randomly induced matrices

Looking first at the projections onto the column space of Q, we have the plots 6 and 8. Both the Householder reflections and the simple rotations are very accurate, with just a small difference between them. However, for larger test matrices this difference becomes larger, with Householder being the more accurate. The results for the random matrices and the cyclic matrices almost equal. In general, both the Householder reflections and the simple rotations perform better for smaller dimensions.

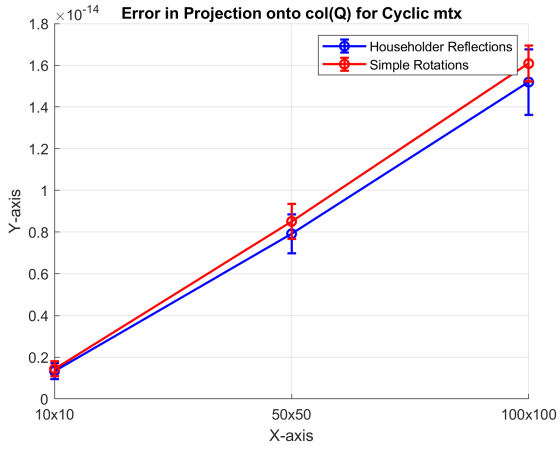

Fig. 8. Plotting the mean error and standard deviation in the projection of A onto col(Q) for cyclic matrices

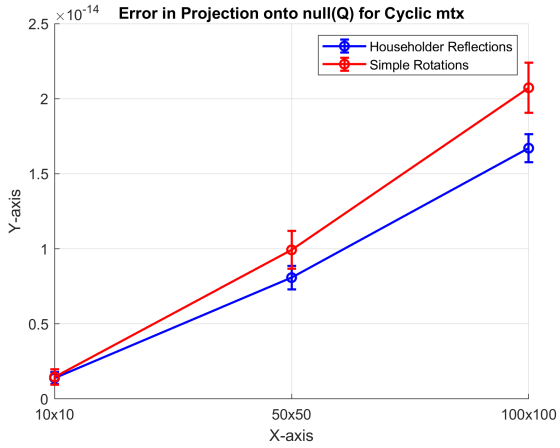

Fig. 9. Plotting the mean error and standard deviation in the projection of A onto null(Q) for cyclic matrices

The results for the projection onto the nullspace of Q are very similar to the ones for the column space of Q. This makes sense intuitively as the experiments are very similar. However, we can see that the error is a bit larger. This could be because there are more operations in the calculation, which could lead to more rounding errors.

D. Notes

These results show that the Householder reflections seem to be more accurate in the QR factorization than the simple rotations. However it may be that for some cases the performance of the simple rotations will be sufficient.

To gain a deeper insight to the accuracy of the simple rotations, it would be useful to use other types of test matrices in more dimensions in these comparisons.

Further it would be interesting to look at the stability and error bounds on the simple rotations in comparison to Householder reflections and perhaps other methods like Givens rotations or Gram-Schmitt-orthogonalization. This could then be done in a similar fashion as done by N. Higham in [2].

VI. GitHub Implementation

The MATLAB file $SimpleRotations.m$ includes a function implementing the QR factorization with simple rotations. A function comparing the results of the QR factorization with simple rotations with the results from MATLAB QR factorization with Householder reflections. In this function you can choose what dimensions you want to use, and also for how many iterations you want to run the test matrices. Finally, there is a function plotting the results.

References

[1] Klaus Diepold, Matthias Kissel, Patrick Dewilde , "Generalized Rotations",May, 2023.

[2] Nicholas J Higham, "Accuracy and Stability of numerical Algorithms"United States of America, Society for Industrial and Applied Mathematics, pp. 361–387, 1996.

[3] Laurent Hoeltgen, "QR Comparison with other Implementations", (https://laurenthoeltgen.name/post/qr-benchmark/) , December, 2020

[4] Nick Higham, (https://nhigham.com/2020/06/30/what-is-the-hilbert-matrix/), "What Is the Hilbert Matrix?", June, 2020

[5] Nick Higham, "The Matrix Computation Toolbox", (https://www.mathworks.com/MATLABcentral/fileexchange/2360-the-matrix-computation-toolbox), MATLAB Central File Exchange,version 1.0.0.0, 2002.