Householder vs. Modified Gram-Schmidt

Mose Wintner

Fall 2015

The QR factorization is a useful tool in matrix and numerical analysis. Here we evaluate two different algorithms for calculating the QR factorization: Householder and Gram-Schmidt. Gram-Schmidt QR on an $m \times n$ matrix A factors in about $2mn^2$ flops, and Householder slightly fewer at $2(mn^2 - n^3/3)$ flops. Each method has a backward error bound. Let u be the least significant digit of the machine precision. Then the 2-norm of the error in Gram-Schmidt QR is bounded "in practice" (I assume this means away from limitations of the algorithm) by $2(mn)^2 u||A||_2$. The backwards error for Householder is given by Theorem 2.3.2 in Björck: the 2-norm of the error in the j^{th} column of the reconstructed QR factorization of A is bounded by $cmnu||a_j||^2$, where c is a small constant.

Note that if Q is orthogonal,

$$||Qy||_2^2 = \langle Qy, Qy\rangle = \langle Q^T Qy, y\rangle = \langle y, y\rangle = ||y||_2^2.$$

Using QR to solve an eigenvalue problem: To use a QR factorization to compute the eigenvalues of a matrix, an iterative procedure is used. First, set $A_1 = A = Q_1 R_1$. The eigenvalues of $A = A_1$ are the same as the similar matrix $Q_1^T Q_1 R_1 Q_1 = R_1 Q_1 =: A_2$. The QR factorization of A_2 is computed as $Q_2 R_2$ and we set $A_3 = R_2 Q_2$ and continue until the difference between A_n and A_{n-1} is sufficiently small. This takes quite a long time, since QR factorization is roughly $O(n^3)$ for square matrices. To reduce the runtime, a series of similarity transformations involving Householder matrices are applied to transform A into a similar Hessenberg matrix, i.e. a matrix with lower bandwidth 1. The iterative procedure converges quickly for such matrices. There is a built-in MATLAB procedure to find a Hessenberg matrix similar to a given matrix: `hess()`. The iterative process preserves the Hessenberg structure, according to Björck. It will converge if the eigenvalues' absolute values are distinct, again according to Theorem 3.4.1 in Björck. If they aren't, we can shift the algorithm to work on $A_k - \tau_k I = Q_k R_k$, then setting $A_{k+1} = R_k Q_k + \tau_k I$ for all k, where τ_k is chosen so that the magnitudes of the eigenvalues of the resulting matrix are distinct. Theorem 3.4.2 describes how this algorithm works:

$$(A - \tau_k I) \cdots (A - \tau_1 I) = Q_1 \cdots Q_k R_k \cdots R_1.$$

The shifts also preserve the Hessenberg structure, according to lemma 3.4.1 in Björck.

An example: First, I pre-and post-multiply a diagonal matrix `diag([5, 4, -2, 10])` by a random unitary matrix. This way, I know the eigenvalues, and I can ensure that they are real. Then, I run `hess(A)`, and then the above procedure. The tolerance for my while loop was set at

10^{-10}. The results:

$$A = \begin{pmatrix} 1.5169 & 3.2594 & 0.2549 & -1.1769 \\ 3.2594 & 1.6545 & 0.18334 & 1.4801 \\ 0.2549 & 0.18334 & 4.7004 & -1.7244 \\ -1.1769 & 1.4801 & -1.7244 & 9.1282 \end{pmatrix}$$

$$H = \begin{pmatrix} 1.5169 & -3.4748 & -2.2496\text{e}-16 & -1.7525\text{e}-16 \\ -3.4748 & 1.6986 & 1.6754 & -4.6297\text{e}-16 \\ 0 & 1.6754 & 9.0164 & 1.7353 \\ 0 & 0 & 1.7353 & 4.7681 \end{pmatrix}$$

$$D_{HH} = \begin{pmatrix} 10 & 1.547\text{e}-07 & -5.8463\text{e}-10 & 2.9303\text{e}-16 \\ 1.547\text{e}-07 & 5 & 7.7453\text{e}-08 & 1.2146\text{e}-15 \\ -5.8463\text{e}-10 & 7.7453\text{e}-08 & 4 & 3.757\text{e}-08 \\ -3.795\text{e}-18 & 4.8499\text{e}-16 & 3.757\text{e}-08 & -2 \end{pmatrix}$$

$$D_{GS} = \begin{pmatrix} 10 & 1.4502\text{e}-15 & -2.607\text{e}-15 & 1.3465\text{e}-15 \\ 3.2744\text{e}-29 & 5 & -3.7307\text{e}-10 & -2.7417\text{e}-16 \\ 0 & -3.7307\text{e}-10 & 4 & 3.4876\text{e}-15 \\ 0 & 0 & 1.57\text{e}-29 & -2 \end{pmatrix}$$

We see that Gram-Schmidt is more accurate, though it takes more iterations to reach the loop condition (Gram-Schmidt took 98 iterations, Householder took 75). The algorithms perform roughly the same for $A = URU^T$, where R is a random upper triangular matrix and U is a random unitary matrix, though they are both slightly more accurate in this case.

Using QR to solve linear least squares: In everything that follows, P is chosen so that the diagonal elements of R appear in nonincreasing order; this improves numerical accuracy, but is not necessary. If the rank of A is less than or equal to the number of linearly independent columns of A, then there are infinitely many exact solutions to our minimization problem. To use the QR factorization to solve a linear least squares problem with $m \geq n$ (the case where $m < n$ is very similar), first assume that the coefficient matrix A has rank n. Then there is a permutation matrix P giving the QR factorization as follows:

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Then

$$\begin{aligned} ||Ax - b||_2^2 &= ||Q^T(Ax - b)||_2^2 \\ &= ||Q^T Ax - Q^T b||_2^2 \\ &= ||(Q^T AP)P^T x - Q^T b||_2^2 \\ &= \left|\left| \begin{pmatrix} R \\ 0 \end{pmatrix} P^T x - Q^T b \right|\right|_2^2 \end{aligned}$$

Partitioning $Q^T b = (c\ d)^T$, where the length of c is n, we have

$$||Ax - b||_2^2 = \left|\left| \begin{pmatrix} RP^T x - c \\ -d \end{pmatrix} \right|\right|_2^2 = ||RP^T x - c||_2^2 + ||d||_2^2,$$

which is minimized in terms of x when $P^T x = R^{-1}c$, which is defined because A and hence R were assumed nonsingular. Thus the solution is $x = PR^{-1}c$.

Now, if $\text{rank}(A) = r < n$, then we still have a permutation matrix P so that $Q^T AP = R$, but now we write

$$R = \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix} \qquad Q^T b = \begin{pmatrix} c \\ d \\ e \end{pmatrix}, \qquad P^T x = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

where R_1 is $r \times r$ and invertible, the lengths of c and y_1 are each r, and the length of d is $n - r$. Then

$$||Ax - b||_2^2 = ||(Q^T AP)P^T x - Q^T b||_2^2 = \left|\left|\begin{pmatrix} R_1 R_2 \\ 0 \end{pmatrix} - Q^T b\right|\right|_2^2 = \left|\left|\begin{pmatrix} R_1 y_1 + R_2 y_2 - c \\ -d \\ -e \end{pmatrix}\right|\right|_2^2.$$

This is minimized when $\hat{y}_1 = R_1^{-1}(c - R_2 y_2)$, so that the x minimizing this quantity is

$$\hat{x} = P\hat{y} = P\begin{pmatrix} R_1^{-1}(c - R_2 y_2) \\ y_2 \end{pmatrix}.$$

Since y_2 is arbitrary, we can just choose $y_2 = 0$ to get a choice of

$$\hat{x} = P\begin{pmatrix} R_1^{-1} c \\ 0 \end{pmatrix}.$$

If $m < n$, the proof is carried out similarly, except by horizontally partitioning vectors.

Backwards error in solving least-squares by QR differs in the two algorithms. Let

$$\gamma_n = nu/(1 - nu)$$

and consider a perturbed least squares problem. Then the Householder QR algorithm for a least squares problem is backwards stable, meaning the computed solution and residual are an exact solution and residual of a slightly perturbed least squares problem with

$$||\delta A||_F \leq c\gamma_{mn}||A||_F, \qquad ||\delta b||_2 \leq c\gamma_{mn}||b||_2$$

for some constant c depending on m and n. The Gram-Schmidt QR least squares algorithm for computing the solution and residual is also backwards stable, according to Björck. Also, if $m >> n$, Householder is twice as slow as Gram-Schmidt in solving least squares problems.

An example: We don't need to run a test of least squares on a specific matrix away from the poorly conditioned ones because the formula above shows only stability in the reconstruction of R_1 is needed to ensure accuracy in least squares. This is covered below.

Other potential issues: Computational limitations come from 1) the condition of the input matrices, 2) numerical singularity of the input matrices, 3) catastrophic cancellation error in an uncareful implementation of the Householder method, and 4) instability coming from loss of orthogonality of the columns of computed columns of Q in Gram-Schmidt.

First, according to Björck, the file HouseholderQR.m is subject to cancellation error. Since the Householder vector is only important insofar that it specifies a normal vector to a hyperplane which the Householder matrix reflects across, we have a choice of sign in specifying u_k in the k^{th} iteration. To avoid making the difference close to 0 and getting catastrophic cancellation error, we should choose α to have a sign opposite to $R(k,k)$.

To test efficiency we can use the fact that Q_1 and R_1 are uniquely determined if A has full rank n and the diagonal of R is positive. By Remark 2.3.1. in Björck, r_{kk} is positive if and only if $a_{kk}^{(k)}$ is negative, where the latter denotes the k^{th} diagonal element of the k^{th} step of the Householder algorithm. To make all diagonal elements positive, we can multiply the rows and columns of Q and R corresponding to negative diagonal elements by -1. Thus it is sufficient in the Householder algorithm to consider test matrices with positive singular values. For Gram-Schmidt, we need only notice that the algorithm is invariant under column scaling, so we can write $\overline{A} = AD$, where D is a diagonal matrix consisting only of ± 1 on the diagonal, and where $\overline{A}$ has only positive singular values, apply Gram-Schmidt to that, then postmultiply by $D^{-1} = D$ to get back to the original A. Finally, it's important to know that the singular values are precisely the diagonal of R.

A test of accuracy in reconstruction of both Q and R in a QR factorization subsumes a test of accuracy in using the QR factorization to solve eigenvalue and least square problems for nice matrices. This also subsumes a discussion of accuracy of the reconstructed matrix A.

We proceed thusly, first testing each algorithm's stability by looking at accuracy in reconstruction of R for several test matrices. We'll try test matrices where the singular values are known. We test performance on a matrix with singular values smaller than the machine precision of MATLAB, which is `eps`$= 2^{-52}$. Since R_1 and Q_1 are only uniquely determined if A is (numerically) nonsingular, problems may happen for matrices which are almost numerically nonsingular.

We also test discrepancy in the computed value of Q, i.e. loss of orthogonality. The computed columns of Q in the Gram-Schmidt algorithm may not remain orthogonal on a finite precision machine if the matrix A is close to singular or numerically singular; here too is where the theoretical Gram-Schmidt factorization regime breaks down: it requires the matrix A to have full rank. Of course, it Suppose one of the methods produces a matrix $\overline{Q}$ for the QR factorization of a matrix A. Then we can test loss of orthogonality by measuring $||I - \overline{Q}^H \overline{Q}||_2$.

The test matrices: First, we will need one or more good control test matrices. We use a matrix C whose singular values are normally distributed with small variance around a sufficiently positive mean, to ensure uniqueness of Q_1 and R_1. To ensure the matrix is (sufficiently) nonsingular, we require its determinant to be sufficiently large. A matrix with negative singular values or one which is numerically singular are not good control matrices since then we have no control over the uniqueness of the QR factorization and hence no good way to test accuracy.

We also use a (nonsingular) control matrix D whose singular values are *uniformly* distributed over a range of positive numbers to test the case where the $\log(\sigma_j)$ are not approximately equal. The results show that the algorithms perform (slightly) worse in this case, so rescaling the columns as discussed above is a good idea.

Next, we want some matrices which test the algorithms under potentially unfavorable conditions. One good test matrix is a random matrix J with geometrically decreasing, positive singular values extending below the precision of the machine. This is a good matrix to use to test the case when the input matrix is ill-conditioned; all such cases are precisely the ones where the ratio of greatest to smallest singular value is large, and this matrix allows us to observe what happens in this case, since the singular values are prescribed. We construct the $m \times n$ matrix with singular values `2.^(-1:-1:-80)` on the diagonal, then pre- and post-multiply by random unitary matrices. This also reveals another way in which these test matrices subsume tests of matrices with prescribed eigenvalues: our matrices are of the form $U\Sigma V^H$ for unitary U and V; matrices with prescribed eigenvalues are of the form $U\Sigma U^H$.

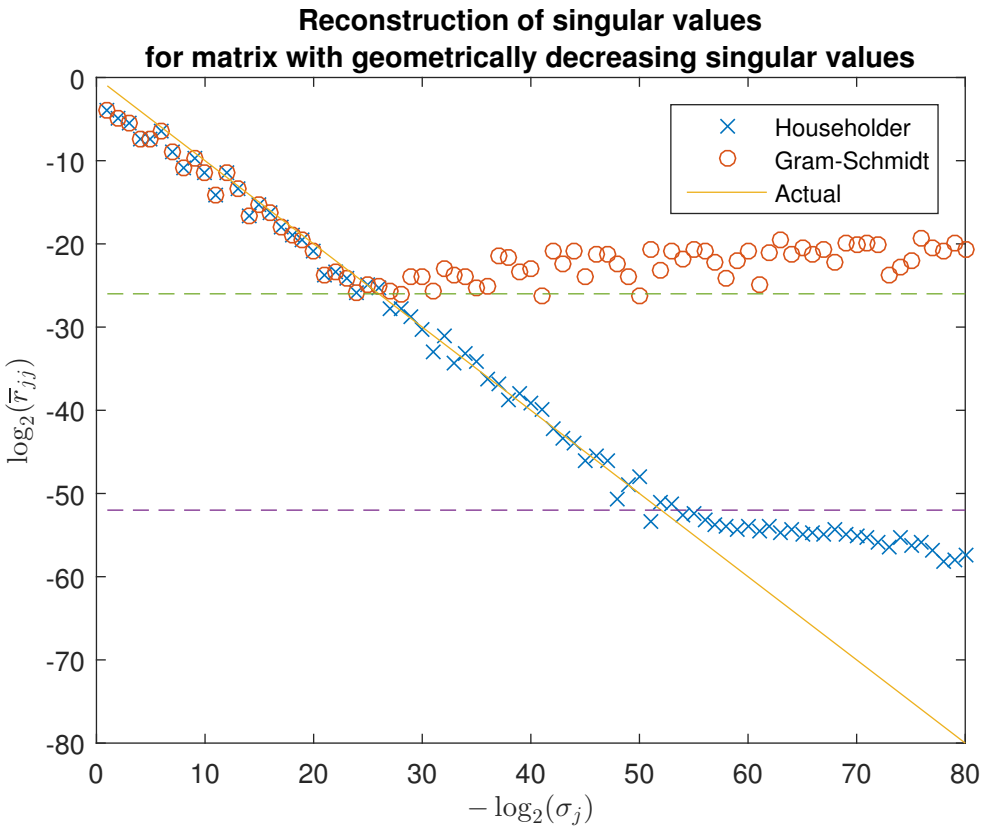
We also want to test a numerically singular matrix N, which is easy to construct: just construct a random $(n-1)$-vector of positive integers and let 0 be the last singular value, then pre- and post-multiply by random unitary matrices as above.

Finally, we test a matrix U which is *almost* orthogonal, for which we just subtract a small amount from a random entry of a unitary matrix. Turns out there are no problems with such a matrix, though.

The results: For the control matrices and the almost orthogonal matrix, Gram-Schmidt seems to consistently edge out Householder in terms of accuracy, but only by a very small amount. In particular, the log accuracies are approximately equal. However, Householder dramatically outperforms Gram-Schmidt for matrices which have small singular values or which are numerically singular. Therefore, before using either algorithm, one should decide which to use based on the determinant of the matrix and its condition number. If the matrix is ill-conditioned or has small determinant, one should use the Householder algorithm. If the matrix is well-conditioned and sufficiently non-singular, one should use the Gram-Schmidt algorithm, since it is generally more accurate, albeit slightly, than Householder. Despite this, the Householder algorithm is far more numerically stable. Below is a table comparing loss of orthogonality in the two algorithms for one run of our test matrices.

$\| I - \overline{Q}^H \overline{Q} \|_2$	C (control)	D (control)	J	N	U
Householder	6.383e-15	6.0863e-15	6.3193e-15	6.3853e-15	6.5992e-15
Gram-Schmidt	1.0923e-15	3.4508e-14	50.634	0.99978	1.1042e-15

Below is a reconstruction of the singular values of J. The dotted lines are drawn at ϵ and $\sqrt{\epsilon}$.



In cases where the Gram-Schmidt algorithm is subject to loss of orthogonality, i.e. cases where the angle between two columns is small (cf. p241) and the system $Ax = b$ needs to be transformed into $Rx = c$, c should be computed directly as

$$c = (1 - q_n q_n^T) \cdots (1 - q_1 q_1^T) b$$

instead of computing the inaccurate matrix Q (p.240). The q_k are the columns. If we just need an accurate QR factorization, we can use "selective factorization," cf. p244 in Björck.

All citations above are from our "textbook," the preprint by Björck.