Householder QR-factorization

We have constructed a method for obtaining a QR factorization and for using this to solve the least squares problem. However, to obtain numerically orthonormal columns required additional computation.

Householder's method produces a QR factorization through a sequence of orthogonal transformations that reduce the matrix to upper triangular form. The basic transformation we use is called a Householder transformation or an elementary reflector of the form

\[ Q = I - 2uu^* \quad \text{with} \quad uu^* = 1. \]

There are two things to note:

1) \( Q^*Q = I \) and this form is the only possibility for an orthogonal matrix that is a rank-one modification of \( I \).

2) \( Q \) can be represented by a single vector \( u \). Even though \( Q \) is \( m \times n \), it can be represented with a vector of length \( m \).
We shall derive the factorization in real arithmetic first and then discuss the changes necessary to obtain a complex form.

**Key Operation**

Given a vector \( a \in \mathbb{R}^m \), we want to construct \( Q \) such that \( Q^T a = e_1 \rho \) with \( Q^T Q = I \), \( \rho \) is unitary. The last condition must hold if \( Q \) is to be orthogonal.

\[(I - 2uu^T) a = e_1 \rho, \quad \|u\|_2 = 1\]

implies

\[a - 2u(u^T a) = e_1 \rho, \quad \text{ie} \quad \eta \theta = a - e_1 \rho, \quad \text{some } \theta.

\[\|u\|_2 = 1 \Rightarrow u = \frac{a - e_1 \rho}{\|a - e_1 \rho\|_2} \quad \text{must hold.}

Check:

\[(I - 2\frac{(a - e_1 \rho)(a - e_1 \rho)^T}{\|a - e_1 \rho\|^2}) a\]

\[= a - 2\frac{(a - e_1 \rho)(a^T a - \rho e_1^T a)}{a^T a - 2 e_1^T a + \rho^2}
= a - 2\frac{(a - e_1 \eta)(\rho (\rho - e_1^T a))}{2 \rho (\rho - e_1^T a)}
= a - (a - e_1 \rho) = e_1 \rho
\]

since \( a^T a = \rho^2 \).
In real arithmetic there are only two choices for $\rho$, $\rho = \pm \| a \|_2$.

Which sign should we take?

We choose the sign that avoids cancellation

$$\rho = -\text{sign}(e_{TA}) \| a \|_2$$

Why? With this choice $\rho$ and $-e_{TA}$ have the same algebraic sign and we can compute $\rho - e_{TA}$ without any loss of significant digits. Cancellation might happen otherwise. We’ll learn about this in great detail, but for the moment here is an example:

Suppose we work with a 4 digit decimal arithmetic

$\begin{align*}
\text{.1235} - \text{.1234} &= .0001
\end{align*}$

Which is completely accurate, but there is only one significant digit in the result. Whereas

$\begin{align*}
\text{.1235} + \text{.1234} &= .2469
\end{align*}$

has all four significant digits.
Note that $u$ is easily obtained from $a$

1) $p = -\text{sign}(a(1)) \cdot ||a||_2$;
2) $a(1) \leftarrow a(1) - p$;
3) $a \leftarrow a ||a||_2$ ($= u$) \quad (Note: $a$ has changed)

We have overwritten $a$ with $u$.

How do we use this to triangularize $A$?

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$,

1) $a \leftarrow A(:,1)$

2) construct $Q_1^T a = e_1 \tilde{p}$, $||\tilde{p}||_2 = ||a||_2$

3) $A \leftarrow Q_1^T A = \begin{bmatrix} p_1 & r_1^T \\ 0 & A_2 \end{bmatrix}$

Now repeat

1) $a \leftarrow A(2:m,1)$

2) construct $Q_2^T a = e_1 \tilde{p}$, $||\tilde{p}||_2 = ||a||_2$

3) $A \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & Q_2^T A \end{bmatrix} = \begin{bmatrix} p_1 & r_1^T & r_1^T \\ 0 & p_2 & r_2^T \\ 0 & 0 & A_3 \end{bmatrix}$
Continuing in this way, we can completely reduce $A$ to upper triangular form

$$Q_n^T A Q_j^T = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where $Q_j = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & \hat{Q}_j \end{bmatrix}$, $\hat{Q}_j = I - 2 y_j y_j^T$

and $R \in \mathbb{R}^{m \times n}$ is upper triangular.

Conceptually we have the following QR factorization algorithm

**Householder QR ($A \in \mathbb{R}^{m \times n}, m \geq n$)**

for $j = 1 \rightarrow n$

1) $a \in A(j; m, j)$

2) Construct $Q_j = I - 2 y_j y_j^T$ s.t. $Q_j^T a = e_j$

3) $A(j; m, j:n) \leftarrow Q_j^T A(j; m, j:n)$

end

On completion $A$ contains $(R, 0)$. Where should we put the $y_j$? Well, we could put $y_j$ in place of the components $A(j; m, j)$ that were annihilated.
However, there is a slight problem because we would like to have $R(i,j)$ in place of $A(i,j)$ on completion. Here is a convenient way to do that.

For a given vector $a$, $\|a\|_2 = \|\rho\|_2$, let us return to the formula

$$Q = I - \frac{(a - \rho)(a - \rho)^T}{\rho(\rho - e^Ta)}$$

Let $T = \frac{\rho - e^Ta}{\rho}, \quad (a - \rho)^T(e^Ta - \rho) = (\hat{a})$

Then

$$Q = I - T (\hat{a}) (1, \hat{a}^T)$$

Gives us a way to represent $Q$ with an $m-1$ vector $\hat{a}$ together with the scalar $T$.

Note $T = 1 + \frac{|e^Ta|}{\|a\|_2}$

So $1 \leq T \leq 1 + \frac{\|a\|_2}{\|a\|_2} = 2$, and
Since \( \hat{\alpha}(j) = \frac{\alpha(j)}{\rho - \eta \alpha} \),
we have \( |\hat{\alpha}(j)| = \frac{|\alpha(j)|}{|\rho - \eta \alpha|} = \frac{\|\alpha\|}{\|\alpha\|} \leq 1 \),
so there is no chance for overflow or growth in the components. We have the following algorithm:

```python
function [u, T] = House (a);
    n = length(a);
    \rho = -\text{sign}(a(1)) \cdot \|a\|_2;
    T = \rho - a(1);
    u = -a(2:n)/T;
    T += T/\rho;
```

Now, to apply the transformation \( QT \) to a vector \( c \in \mathbb{R}^m \), we use

\[
 QTc = \left[ I - T (u)(u^T) \right] c
 = c - (u)(u^T) T (u) c
\]
This leads to the following algorithm to apply a Householder transformation represented in this form:

\[
\text{function } [C] = \text{ApplyHouse}(C, U, E) \\
m = \text{length}(U) + 1; \\
\theta = C(1) + U^T C(2:m); \quad \theta < \theta E; \\
C = C - (U)\theta
\]

The following two routines may be found on the CMMUS1 website under matlab codes:

`QRfac.m`  `QRfac2.m`

The first uses an auxiliary vector to compute the vector of the Householder transformation, while the second just works with the array A. They both compute the same thing.

Note: I have change slightly the way a certain sign is handled. It should be easy enough to see that these compute the factorization as we developed it.
function [QR,tau] = qrfac(A);
%
% Input: A - an m by n matrix with m .ge. n
% Output: QR - The Householder QR factorization
% in product form. QR can overwrite A.
% tau - A vector of length n containing
% "pivot" information
%
Factors A = QR , where A is m by n
Q'Q = I_m, R is m by n upper triangular
The leading n by n block of R is an
n by n upper triangular matrix R_n
In practice A is overwritten by the
Q in product form and R_n in the
upper triangle
D.C. Sorensen
3 Oct 00

[m,n] = size(A);
tau = zeros(n,1);

for k = 1:n,

Compute the Householder vector v

I - tau(k)*v*v' is the transformation

v = A(k:m,k);
rho = sign(v(1))*norm(v);
if (abs(rho) > 0),
v(1) = v(1) + rho;
tau(k) = v(1)/rho;
v = v/v(1);

for j = k+1:n

Apply the Householder transformation to the j-th column of A

alpha = tau(k)*(v'*A(k:m,j));
A(k:m,j) = A(k:m,j) - v*alpha;
end
A(k:m,k) = v;
A(k,k) = -rho;
end
QR = A;

qrfac.m
function [QR,tau] = qrfact(A);

Input:  A - an m by n matrix with m .ge. n

Output: QR - The Householder QR factorization
         in product form. QR can overwrite A.
         tau - A vector of length n containing
         "pivot" information

Factors A = QR , where A is m by n
Q'Q = I_m, R is m by n upper triangular

The leading n by n block of R is an
n by n upper triangular matrix R_n

In practice A is overwritten by the
Q in product form and R_n in the
upper triangle

D.C. Sorensen
3 Oct 00

[m,n] = size(A);
tau = zeros(n,1);

for k = 1:n,

    Compute the Householder vector v
    I - tau(k)*v*v' is the transformation
    v = [1 ; A(k+1:m,k)];

    rho = sign(A(k,k))*norm(A(k:m,k));
    if (abs(rho) > 0),
        A(k,k) = A(k,k) + rho;
        tau(k) = A(k,k)/rho;
        A(k:m,k) = A(k:m,k)/A(k,k);

        for j = k+1:n

            Apply the Householder transformation to the j-th column of A
            alpha = tau(k)*(A(k:m,k)'*A(k:m,j));
            A(k:m,j) = A(k:m,j) - A(k:m,k)*alpha;

        end
    end
    A(k,k) = -rho;

end

QR = A;

QRfact2.m
This variant does not use any auxiliary
storage.
Solving \( \min \| b - Ax \|_2 \) via Householder’s method

Conceptually, we have shown how to compute

\[
A = QR, \quad \hat{A} = Q_R, \hat{A}_2, \ldots, \hat{A}_n
\]

\[
Q^TQ = I_m, \quad R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}, \quad Q = [\hat{Q}, \hat{Q}]
\]

with \( \hat{Q}^T\hat{Q} = I_m \), \( A = \hat{A}R \). The “short” form QR factorization. Recall: This \( \hat{A}R \)

is mathematically equivalent (up to signs) to the QR that CGS or MQS would give.

However, the computed factorizations obtained

via these three methods are very different.

How can we use the Householder factorization to solve this? Well, we could form \( \hat{Q} \)

\[
\hat{Q} = (Q_1 b_2 \ldots a_n) \begin{bmatrix} I_n \\ 0 \end{bmatrix}
\]

and then

\[
x_\ell = R^{-1} \hat{Q}^T b \quad \text{We've shown}
\]

this already. Another way to proceed

is to observe that for any \( x \),

\[
\| b - Ax \|_2^2 = \| Q^T [b - Ax] \|_2^2 = \| Q^T b - (\hat{R}) x \|_2^2
\]
Let \( (\tilde{c}) = (\tilde{a}^T \tilde{b}) = Q^T b \).

Then
\[
\|b-Ax\|_2^2 = \|(\tilde{c} - \tilde{R}x)\|_2^2
\]
\[
= \|\tilde{c} - \tilde{R}x\|_2^2 + \|\tilde{R}x\|_2^2
\]
\[
\geq \|\tilde{b}\|_2^2
\]

with equality if and only if \( \tilde{R}x = \tilde{b} \).

We can solve the \( n \times n \) triangular system by back substitution.

Key Observation:

Don't form \( Qx = Q_1 Q_2 \ldots Q_n \) and then form \( (\tilde{c}) = b \).

Instead note

\[
(\tilde{c}) = Q^T b = Q_n^T Q_{n-1}^T \ldots Q_1^T b
\]

ie for \( j = 1 : n \)

\[
b_{j} = Q_j^T b
\]

and

solves \( \tilde{R}x = \tilde{b}(1:n) \)

Furthermore:

\[
Q_j^T b = (I - y_j y_j^T) b = b - y_j (y_j^T b).
\]
We never need to form $Q$ to solve $\mathbf{L}$. This is not only more efficient, it is much better numerically.

So far, we have seen SVD, CGS, HGS, and Householder for solving $\mathbf{L}$. You will derive a fifth way in problem set 4.

The final method we will look at for the full rank least squares problem is the normal equations.

**NORMAL EQUATIONS:**

We shall use the key fact we have already developed: $\mathbf{x}$ solves $\mathbf{L}$ iff

$$(*) \quad \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}) = \mathbf{0}$$

**Remember the Picture!**

$b - \mathbf{A}\mathbf{x}$ must be orthogonal to $\text{Range}(\mathbf{A})$

This immediately gives $(*)$ which is generally referred to as the normal equations. In another form

$$(\mathbf{A}^T \mathbf{A})\mathbf{x} = \mathbf{A}^T \mathbf{b}$$
Normal Equations Algorithm (NEQ)

1) Form $M = A^T A$

2) Form $C = A^T b$

3) Solve $M x = c$

Formally, we see that

$$x = M^{-1} c = (A^T A)^{-1} A^T b$$

How does this relate to our other solutions?

(NEQ) vs (SVD)

$A = U S V^T$ Short Form SVD

$A^T A = VSU^T S V^T = V S^2 V^T$

$A^T b = VSU^T b$

$(A^T A)^{-1} A^T b = (V S^{-2} V^T) V S U^T b = V S^{-1} U^T b$

So, we get the same solution (mathematically) as we would have gotten with SVD.
\[(\text{NEQ}) \; vs \; (\text{QR})\]

\[A = QR\quad (\text{Short Form - I've dropped the hats})\]

\[Q^TQ = I, \quad R_{n \times n}, \text{upper triangular, biosing} \]

\[A^T A = R^TQ^TQR = R^T R\]

\[A^T b = R^T Q^T b\]

\[(A^T A)^{-1}A^T b = (R^{-1}R^{-T})R^TQ^T b = R^{-1}Q^T b\]

Again, the same solution (mathematically) as we would have gotten with QR.

Notice: NEQ involves a lot of "doing + undoing" with respect to SVD, QR.

The operators:

1) \((A^T A)^{-1}A^T\) \hspace{2cm} (NEQ)
2) \(R^{-1}Q^T\) \hspace{2cm} (QR)
3) \(USU^T\) \hspace{2cm} (SVD)

Are three equivalent expressions for...
a left inverse \( A^L \) for \( A \)
\[
A^2 A = I_n
\]

\( A^2 \) is (mathematically) unique if \( \text{Rank}(A) = n \)

Of the three expressions (8) - The SVD is the only one that holds up when \( A \) becomes rank deficient. However, it is still tricky to use. Mathematically

\[
A = U_r S_r V_r^T \\
U_r^T U_r = V_r^T V_r = I_r
\]

\( r = \text{rank}(A) \)

\[
S_r = \begin{pmatrix} 0_r & \cdot \cdot \cdot & 0_r \\ \cdot \cdot \cdot & \cdot \cdot \cdot & \cdot \cdot \cdot \\ 0_r & \cdot \cdot \cdot & 0_r \end{pmatrix}
\]

\[
X_0 = V_r S_r^{-1} U_r^T b
\]

is still a solution to \( \min_{x} \| b - Ax \|_2 \) (65)

In fact, \( X_0 \) is a solution and \( \| x \|_2 \leq \| x \|_2 \)

for any other solution to \( X \)

We'd see this in more detail when we study the rank deficient case.
More on Normal Equations

We have seen that

\[ M = AA^T = RTQR = RTR = (D)(D) \]

\( AA^T \) is the product of a lower triangular matrix times its transpose, an upper triangular matrix.

This is called the Cholesky factorization of \( M \).

**Key Properties of \( M \)**

1) \( M = MT \)

\[ (AA^T)^T = (A^T)^T (A^T)^T = AA^T \]

2) \( x \neq 0 \Rightarrow x^TMx > 0 \iff \text{rank}(A) = n \)

Why: \( x^TMx = (x^TAR)(Ax) = \|Ax\|^2 \geq 0 \)

with equality \( \iff Ax = 0 \).
Any matrix \( M \) that satisfies
\[
M = M^T, \quad x \neq 0 \Rightarrow x^T M x > 0
\]
is symmetric positive definite. (pos. def.)

If \( M = A^T A \), \( \text{Rank}(A) = n \) then \( M \) is automatically symmetric and positive definite. Later we shall see how to factor any symmetric pos. def. matrix \( M = R^T R \) into it's Cholesky factorization directly. In the special case \( M = A^T A \) we get this as a by-product of the QR factorization.