

The Action of Householder Reflectors

Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$. You may recall that the Householder QR factorization can be written as

$$H_p \cdots H_2 H_1 A = R,$$

where $p = \min(n, m - 1)$ and $H_k = I - (\frac{2}{u_k^T u_k}) u_k u_k^T$ is a Householder reflector which introduces zeros into positions $k + 1$ to m of the k^{th} column of the matrix $A^{(k-1)} = H_{k-1} \cdots H_2 H_1 A$. This gives $A = QR$, where $Q \equiv H_1 H_2 \cdots H_p$.

The H_k are not explicitly formed, since that is very inefficient, and since they are completely determined by the u vectors. The matrix Q is rarely formed, either. What we usually do is save the u_k 's. If we let Q_u be the array whose k^{th} column is u_k , it could be called the "factored form" of Q , but Q_u is *not* Q (maybe you can think of Q_u as the 'dna' for Q). In Matlab, if u is our variable name for u_k , then we might write $Q_u(:, k) = u$. In a memory efficient code, we would only save entries $k : m$ of u_k ($\tilde{u}_k = u_k(j : m)$), and would use the lower triangle of A (plus an n -vector) to store \tilde{u}_k , $k = 1, \dots, p$. To be explicit: if Q_u and $u1$ are given as the output of `houseqr`, then $Q_u(k+1:m, k)$ contains the "decapitated" \tilde{u}_k and $u1(k)$ contains its "head" $\tilde{u}_k(1)$, so it can be reconstructed as $\tilde{u}_k = [u1(k); Q_u(k+1:m, k)]$.

If later we will need to find the vector $c = Q^T b$, then instead of explicitly forming Q , we use the routine `HOUSEQACT`: Notice that

$$C = Q^T B = H_p \cdots H_2 H_1 B,$$

so we write `HOUSEQACT` to form this product

```
C = B;
for k=1:p,
    C = Hk * C; %but you code this line the "right way"...
end
```

The "right way" (above), in Matlab with $v = \tilde{u}_k$ and $\beta = 2/(\tilde{u}_k^T \tilde{u}_k)$, might be something like

$$C(k:m, :) = C(k:m, :) + (\beta * v) * (v' * C(k:m, :));$$

Of course, it would be just as easy to write code to implement

$$C = QB = H_1 H_2 \cdots H_p B,$$

or BQ^T or BQ . In fact, while it is rarely needed explicitly, we could compute Q^T with the loop above by taking $B = I$.

We discuss how to find the u_j 's and how to code the algorithm above efficiently for speed and memory on other pages, but a good implementation of the loop above should be all the convincing you need to understand that Householder reflectors are never explicitly computed. If $B \in \mathbb{R}^{m \times r}$, a good implementation of the pseudocode above requires $r \sum_{j=0}^n 4(m - j) \approx 2mnr(2 - \frac{n}{m})$ flops, and even if we knew $Q \in \mathbb{R}^{m \times m}$ explicitly, that matrix product would require $2m^2r$ flops (which is more costly than `HOUSEQACT` since $n \leq m$).