Householder QR

Algorithmically, this method is very close to G.E. with no pivoting. There, a Gauss transform $M_k = I + m_k e_k^t$ was used to introduce zeros below the (k, k) element of $A^{(k-1)}$, giving $A_G^{(k)} = M_k A_G^{(k-1)}$. Here, a Householder reflector $H_k = I - \beta u_k u_k^t$ replaces the Gauss transform as the operator that introduces zeros below the (k, k) element: $A_H^{(k)} = H_k A_H^{(k-1)}$.

Let $A \in \mathbb{R}^{m \times n}$, with $m \geq n$, and let $p = \min(n, m-1)$. The Householder QR factorization of A can be coarsely described as

$A^{(0)} = A$
For k = 1:p
  Compute u so that $(I - \beta uu^t)A^{(k-1)}$ has zeros below its (k, k) entry
  Compute $A^{(k)} = H_k A^{(k-1)}$
End

There are some important details to consider yet, but it is essentially this simple.

We know that if u is a Householder vector for x, then it is a multiple of $x \pm \|x\|_2\, e_1$, and that $Hx = (I - \beta uu^t)x = \pm\|x\|_2\, e_1$, with $\beta = 2/(u^t u)$. As with G.E. we can view the k^{th} step as

$$A^{(k)} = \begin{bmatrix} R^{(k)} & X^{(k)} \\ 0 & \tilde{A}^{(k)} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & \tilde{H}_k \end{bmatrix} \begin{bmatrix} R^{(k-1)} & X^{(k-1)} \\ 0 & \tilde{A}^{(k-1)} \end{bmatrix} = H_k A^{(k-1)},$$

where $R^{(k)}$ is $k \times k$ upper triangular, and $\tilde{A}^{(k)}$ is $(m-k) \times (n-k)$. Here $u_k^t = (0^t, \tilde{u}_k^t)$, where $\tilde{u}_k$ is a Householder vector associated with $x = \tilde{A}^{(k-1)} e_1$.

We *never* form the Householder reflectors. We simply save the $\tilde{u}_k$ vectors. When we want to compute $C = (I - \beta uu^t)B$ for some matrix B (as in the loop above), we simply note that

$$C = B - ((\beta u)(u^t B)).$$

The parenthesis are purposefully placed to suggest the implementation. If H is $n \times n$ and B is $n \times p$, then the cost of this this implementation is about $4np$ flops. In the loop above, we use $\tilde{u}_k$ in place of u, and $\tilde{A}^{(k-1)}$ in place of B.

When the loop terminates, we have $H_p \cdots H_2 H_1 A = R$, and defining $Q^t = H_p \cdots H_2 H_1$ gives $A = QR$. We do not explicitly have the matrix Q, but by saving the $\tilde{u}_k$'s, we have the "factored form" of Q: all the information needed to construct it, or to compute its action.

The cost of this factorization is about $\sum_{k=0}^{p-1} 4(m-k)(n-k) = 2mn^2 - 2n^3/3 + O(mn)$ flops. If we overwrite the upper triangle of the array A by R, the lower triangular part can be used to store all but one element of each of the $\tilde{u}_k$. Typically an extra n-vector is used to store the first element of each $\tilde{u}_k$.