The various QR factorizations in Matlab

These notes explain what Matlab's various QR factorization functions do, in the terms introduced in Lecture 4. Of course you can also type the command `help qr` in Matlab to get more details.

Let's suppose that $A \in \mathbf{R}^{n\times k}$, with rank r.

The Matlab command `[Q,R]=qr(A)` returns the 'full' QR factorization, with square, orthogonal $Q \in \mathbf{R}^{n\times n}$, and $R \in \mathbf{R}^{n\times k}$ upper staircase, with its bottom $(n-r)\times k$ block zero.

```
[Q,R]=qr(A); % compute full QR factorization
```

When A is skinny (or square), *i.e.*, $n \geq k$, a variation on this call,

```
[Q,R]=qr(A,0); % compute economy QR factorization
```

generates a QR factorization with $Q \in \mathbf{R}^{n\times k}$, with $Q^TQ = I_k$, and $R \in \mathbf{R}^{k\times k}$ upper triangular, with its bottom $(k-r)\times k$ block zero.

The general Gram-Schmidt procedure in Lecture 4, *i.e.*, with $Q \in \mathbf{R}^{n\times r}$ and $R \in \mathbf{R}^{r\times k}$ upper staircase, is not the same as what Matlab refers to as the 'economy' QR factorization. This 'economy' routine in Matlab returns $R \in \mathbf{R}^{k\times k}$ which can be singular (non-invertible), *i.e.*, $\mathbf{Rank}(R) < k$, if A is not full-rank.

One way to do QR factorization in the same form of Lecture 4-17 is:

```
r = rank(A);
[Q,R]=qr(A); % compute full QR factorization
Q = Q(:,1:r);
R = R(1:r,:);
```

This routine returns a $Q \in \mathbf{R}^{n\times r}$, with $Q^TQ = I_r$, and $R \in \mathbf{R}^{r\times k}$ upper staircase and full rank.

Matlab can permute the columns of A in order to sort the diagonal elements of R:

```
[Q,R,E]=qr(A); % compute full QR factorization such that A*E = Q*R
```

This command generates a QR factorization with $Q \in \mathbf{R}^{n\times n}$, with $Q^TQ = I_n$, and $R \in \mathbf{R}^{n\times k}$ upper staircase, with its bottom $(n-r)\times k$ block zero. If we divide R into $[\tilde{R}\ S]$, then upper staircase $\tilde{R} \in \mathbf{R}^{n\times r}$ has non-zero values in its diagonal entries in decreasing order.

The 'economy permutation' QR factorization command in Matlab, *i.e.*, `[Q,R,E]=qr(A,0)`, also can generate singular $R \in \mathbf{R}^{k\times k}$ as the 'economy' QR factorization does. One way to do QR factorization in the same form of Lecture 4-18 is:

```
r = rank(A);
[Q,R,E]=qr(A); % compute full QR factorization such that A*E == Q*R
Q = Q(:,1:r);
R = R(1:r,:);
R_tilde = R(:,1:r);
S = R(:,r+1:end);
P = E' % note that E' == inv(E)
```

This routine returns $Q \in \mathbf{R}^{n\times r}$ with $Q^TQ = I_r$, full rank upper staircase $R \in \mathbf{R}^{r\times k}$, and a permutation matrix $P \in \mathbf{R}^{k\times k}$ holding $A = Q[\tilde{R}\,S]P$ where $R = [\tilde{R}\ S]$ and $\tilde{R} \in \mathbf{R}^{r\times r}$ is upper triangular and invertible as Lecture 4-18.