1. (Why does my QR implementation in Matlab get some entries correct and some wrong — sometimes only by a sign?)

   The QR factorization is not unique. For instance, if D is any diagonal matrix with ± 1 on the diagonal, one can replace Q and R with (QD), $(D^{-1}R)$. So it is possible that your implementation just returns a different factorization. To check the result, you can verify that `norm(A-Q*R) / norm(A)` is small. For instance, a good test is

```
assert(norm(Q*R-A)/norm(A) < sqrt(eps));
assert(norm(Q*Q' - eye(size(Q))) < sqrt(eps));
assert(all(all(R == triu(R))));
```

   Many of you wrote your for loop as

```
n = size(A, 1);
for i = 1:n
       (compute HH reflector that maps A(k,k:end) to a multiple of e1)
       (apply reflector to the last n-i+1 rows of the matrix)
end
```

   Note that the last iteration of the loop works on a 1×1 matrix, so it can be omitted (a 1×1 matrix is already upper triangular!) and the for loop can stop at $n - 1$. If you check what your code does for a 1×1 matrix, it turns out that $H = -1$, so this last step does nothing but changing signs.

2. (Why does `M=qr(A)` return a different result than `[Q,R] = qr(A)`)

   Matlab is a weird language, and functions can have a different behavior according to the number of return values they are called with.

   In particular, for a dense matrix, the one-output version of QR returns a matrix `M` such that its upper triangular part `triu(M)` is R, and its lower triangular part contains a compressed representation of Q (its jth column contains a compressed representation of the vector u_j that defines H_j). All of this is described in the docs (see `doc qr`).

3. (In conjugate gradient, why do we claim that d_{k+1} is orthogonal to d_j for $j = 1, 2, \dots, k$ if in the algorithm we enforce only orthogonality to d_k)?

   It is a consequence of the symmetry/Hermitianity of A that d_{k+1} is always orthogonal also to $d_1, d_2, \dots, d_{k-1}$, even if we do not enforce it explicitly in the algorithm.

   Formally, one proves simultaneously by induction (we did not see the details during the course).

   Lemma 1. *Let d_k, r_k be the sequences of search directions and residuals produced in conjugate gradient. Suppose that no breakdown happens in the process. Then,*

   (a) $r_k \in K_{k+1}(A, b)$, i.e., $r_k = \alpha_{k0}b + \alpha_{k1}Ab + \cdots + \alpha_{kk}A^k b$ for some choice of the coefficients α_{kj}. Moreover, $\alpha_{kk} \neq 0$.

   (b) $r_0, r_1, \dots, r_{k-1}, r_k$ are a basis of $K_{k+1}(A, b)$;

   (c) $d_k \in K_{k+1}(A, b)$, i.e., $d_k = \gamma_{k0}b + \gamma_{k1}Ab + \cdots + \gamma_{kk}A^k b$ for some choice of the coefficients γ_{kj}. Moreover, $\gamma_{kk} \neq 0$.

   (d) $d_0, d_1, \dots, d_{k-1}, d_k$ are a basis of $K_{k+1}(A, b)$;

   (e) $r_j^* r_k = 0$ for each $j < k$;

   (f) $d_j^* A d_k = 0$ for each $j < k$.

   Proof (sketch). Let us focus on the induction step $k \to k + 1$, i.e., we assume that the result holds already for a certain value of k and prove it for $k + 1$.

(a)

$$r_{k+1} = r_k + t_k A d_k = (\alpha_{k0} b + \alpha_{k1} Ab + \cdots + \alpha_{kk} A^k b) + t_k A(\gamma_{k0} b + \gamma_{k1} Ab + \cdots + \gamma_{kk} A^k b),$$

so r_{k+1} is a linear combination of $b, Ab, \ldots, A^{k+1}b$. The coefficient in front of $A^{k+1}b$ is $t_k \gamma_{kk}$; t_k can't be zero otherwise there would be breakdown, and γ_{k-1} can't be zero by induction.

(b) $r_0, r_1, \ldots, r_{k-1}, r_k$ are a basis of $K_{k+1}(A, b)$, and r_{k+1} is in $K_{k+2}(A, b)$ but not in $K_{k+1}(A, b)$, so it is independent from them.

(c) Analogous to 1.

(d) Analogous to 2.

(e)

$$r_j^* r_{k+1} = r_j^* (r_k - t_k A d_k). \tag{1}$$

If $j = k$, orthogonality is enforced in the algorithm by the choice of t_k. If $j < k$, then $r_j^* r_k = 0$ by induction hypothesis, so we only need to prove that $r_j^* A d_k = 0$. We have $r_j \in K_{j+1}(A, b)$, so $r_j = \delta_0 d_0 + \delta_1 d_1 + \cdots + \delta_j d_j$, hence if $j < k$ $r_j^* A d_k = 0$ by induction hypothesis.

(f) Similarly to 5,

$$d_j^* A d_{k+1} = d_j^* A (r_{k+1} + \beta_{k+1} d_k). \tag{2}$$

If $j = k$, $d_k^* A d_{k+1} = 0$ follows by the choice of β_{k+1}. If $j < k$, we have $d_j^* A d_k = 0$ by induction, so we only need to show that $d_j^* A r_{k+1} = 0$. The vector $A d_j$ is in the Krylov space $K_k(A, b)$, hence it is a linear combination of $r_0, r_1, \ldots, r_k$, so $(A d_j)^* r_{k+1} = 0$.

□

4. (How does one get the formula $\beta_k = \frac{r_k^* r_k}{r_{k-1}^* r_{k-1}}$?)

We choose the value β_k so that $d_k = r_k + \beta_k d_{k-1}$ satisfies $d_{k-1}^* A d_k = 0$. Substituting and expanding one gets

$$0 = d_{k-1}^* A d_k = d_{k-1}^* A (r_k + \beta_k d_{k-1}) = d_{k-1}^* A r_k + \beta_k (d_{k-1}^* A d_{k-1}) \implies \beta_k = -\frac{d_{k-1}^* A r_k}{d_{k-1}^* A d_{k-1}}.$$

This gives already an expression for β_k; now we prove that it is also equal to $\frac{r_k^* r_k}{r_{k-1}^* r_{k-1}}$.

We manipulate the numerator using the other formula that defines the CG iteration, that is, $r_{k+1} = r_k - t_k A d_k$. We have

$$r_{k+1}^* r_{k+1} = (r_k - t_k A d_k)^* r_{k+1} = 0 - \overline{t_k} (d_k^* A r_{k+1})$$

(here the bar denotes a complex conjugate), so, shifting indices, $r_k^* r_k = -\overline{t_{k-1}} (d_{k-1}^* A r_k)$.

For the denominator, we have similarly

$$r_{k-1}^* r_{k-1} = (r_k + t_{k-1} A d_{k-1})^* r_{k-1} = 0 + \overline{t_{k-1}} d_{k-1}^* A r_{k-1} = \overline{t_{k-1}} d_{k-1}^* A (d_{k-1} - \beta_{k-1} d_{k-2}) = \overline{t_{k-1}} d_{k-1}^* A d_{k-1}.$$

Combining these two last formulas one gets the equivalent expression for β_k.