1. (Is Matlab's operator `*` "smart" in the sense that it is able to compute matrix products faster than the nominal cost of $O(n^3)$ in special cases, like for instance `A\b` is faster when A is triangular?)

   No, it is not (as far as I know, at least — it's closed-source, so there could be anything in it...). It always computes the product with the general method: the product between $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times p}$ is computed by doing mp scalar products between vectors of length n, and costs $O(mnp)$.

   In the cases where there is a smarter strategy to compute a product (e.g., A is an identity-plus-rank-1, A is diagonal, A has zero rows...), you will have to implement it yourself.

   There is one exception: Matlab has a type `sparse` for sparse matrices (that is, matrices with lots of zeros), although we haven't discussed it in detail in the course, which is implemented by storing only the triples (i, j, A_{ij}) with $A_{ij} \neq 0$. Products with a matrix of type `sparse` iterate only on the nonzero values.

2. (In the end of the November 29 lecture, why do we choose shift $\sigma_k = (A_k)_{nn}$ in shifted QR, and why does this choice give local convergence with $\|E_k\| = O(\|E_k\|^2)$)

   We did not prove this statement – I just claimed it. A proof for it will not be required. If you are curious, there is an 'informal analysis' in Section 4.4.4 of Demmel's book. This analysis captures the typical behavior of the method. A general statement that works for all matrices is more difficult to obtain. There are convergence proofs under special assumptions, but convergence of the QR iteration in full generality (with the various shifting strategies that are used in practice, repeated eigenvalues, and keeping track also of errors in computer arithmetic) is still a research topic.

3. (Again in the November 29 lecture, we quickly describe how Hessenberg matrices can be used to reduce the complexity of a method to compute eigenvalues; does this apply to the QR iteration only, or also to the other methods?)

   We just outlined this method, and only for the QR iteration.

   There might be some minor advantages in reducing A to Hessenberg form before starting one of the other methods (we have more zeros in the matrix, so the products are cheaper), but since the cost of the procedure is $O(n^3)$ I don't think it is worth it in general. It is not used in practice.

   In particular, I don't see a simple way to exploit the Hessenberg form to reduce the complexity of orthogonal iteration with $p = n$ directly, without transforming it into QR iteration.

   For subspace iteration with $p \ll n$ and the power method (which is its $p = 1$ variant), the methods cost $O(n^2 p)$ per iteration, so applying a $O(n^3)$ precomputation is not worth it, unless we have a very large number of iterations. But at that point full QR iteration starts to become competitive: one typically uses power iteration and its variants to compute eigenvalues only when either the number of iterations p is small, or we have some particular structure that allows one to perform matrix products with costs less than $O(n^2)$ (such as sparsity).

4. (At the top of page 4 of the November 29 lecture, shouldn't $|\lambda_p| > |\lambda_{p+1}|$ be $|\lambda_{p'}| > |\lambda_{p'+1}|$ instead?) Yes, you are correct. My bad. I will add a correction to the slides.