Assignment 6

Due Friday 20 October 2023, at the start of class

Please read Lectures 8,9,10,11,12 in the textbook *Numerical Linear Algebra* by Trefethen and Bau. This Assignment mostly covers Gram-Schmidt QR, Householder reflectors and QR, and least squares.

DO THE FOLLOWING EXERCISES from Lecture 8:

- **Exercise 8.3**

DO THE FOLLOWING EXERCISES from Lecture 9:

- **Exercise 9.3**

DO THE FOLLOWING EXERCISES from Lecture 10:

- **Exercise 10.2** *Implicit in this and similar questions, which ask for Matlab codes, is to show that they work! Please show a brief command line session where you generate a generic matrix, run the code(s) on it, and then verify that the outputs have the required properties. Use* `norm()` *to avoid spewing too many numbers at me.*
- **Exercise 10.3**

DO THE FOLLOWING ADDITIONAL EXERCISES.

The Matlab built-in `qr()` *computes the QR factorization using Householder reflectors (Lecture 10). In the next two problems, go ahead and use it.*

P11. By applying Matlab's backslash command, reproduce Figure 11.1. By applying Algorithm 11.2, using the `qr` and backslash commands, reproduce Figure 11.2. Please make at least a modest effort to duplicate the appearance of these Figures. (*Hints.* Note `axis off` creates a clean picture without ticks and axes labels. Then you can put back the axes themselves using `plot([-6 6],[0 0],'k')` and similar.)

P12. *While we have used QR to solve linear systems, here we see that QR factorization has a completely different application. For more, see Lectures 24–29.*

(a) By googling for "unsolvable quintic polynomials" or similar, confirm that there is a theorem which shows that fifth and higher-degree polynomials cannot be solved using finitely-many operations (including roots, a.k.a. "radicals"). In other words, there is no finite formula for the solutions ("roots") of such polynomial equations. Who proved this theorem? When? Show a quintic polynomial for which it is known that there is no finite formula. (*You do* not *need to prove your claim!*)

(b) At the Matlab command line, try the following:

```
>> A = randn(5,5);  A = A' * A;     % create a random 5x5 symmetric matrix
>> A0 = A;                          % save a copy of the original A
>> [Q, R] = qr(A);  A = R * Q
...                                 % repeat about 10 times
>> [Q, R] = qr(A);  A = R * Q
```

We start with a random, symmetric 5×5 matrix A_0 and then generate a sequence of new matrices A_i. Specifically, each matrix A_i is factored

$$A_i = Q_i R_i$$

and then the next matrix A_{i+1} is generated by multiplying-back in reversed order:

$$A_{i+1} = R_i Q_i.$$

What happens to the matrix entries when you iterate at least 10 times? (*Perhaps also use a* `for` *loop to see a strong effect from e.g. 100 iterations.*) What do you observe about this sequence of A_i? Now compare `sort(diag(A))` to `sort(eig(A0))`.

(c) To see a bit more of what is going on in part **(b)**, show that

$$A_{i+1} = Q_i^* A_i Q_i.$$

This shows A_{i+1} has exactly the same eigenvalues as A_i ; explain why.

(d) Write a few sentences which relate part **(a)** to what happens in parts **(b)** and **(c)**. (*Hint. Try to relate the two parts by yourself first. Then read "A Fundamental Difficulty" in Lecture 25 to confirm your understanding.*)