Lab 4

Noah Prentice

22 November 2024

Note.

Code for this assignment was done in Python. Only important code snippets and outputs are shown here, but complete files can be found at this GitHub repository.[a] All code for Exercise N is in `ExN.py`.

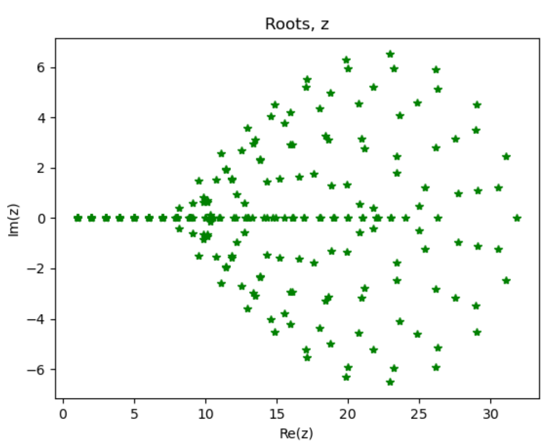
[a]URL: https://github.com/NoahPrentice/Numerical-Linear-Algebra-MTH551-F24/tree/main/Lab4

Exercise 1. For various n values, construct a monic polynomial with roots at each of the integers 1 through n. Then attempt to solve the for the roots of these polynomials. Plot the estimated roots in the complex plane. Just by looking at the plot, and assuming that the relative errors in the coefficients of the polynomial are close to machine epsilon, can you estimate the relative condition number of the problem of finding say the 20^{th} root of the degree 30 Wilkinson polynomial?

Solution. The plotting code was written as follows:

```
plt.figure()
for n in range(21, 31):
    roots = np.roots(np.poly(np.arange(1, n+1)))
    plt.plot(np.real(roots), np.imag(roots), "g*")
plt.title("Roots, z")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
plt.show()
```

This produced the following plot:



Estimating from the plot, the total error between the computed 20^{th} root of the degree 30 Wilkinson polynomial and the true root is about 6. Assuming that the relative errors in the coefficients of the polynomial are close to machine epsilon, this yields a relative condition number of

$$\left(\frac{6}{20} \middle/ \varepsilon_{machine} \right) \approx 3 \cdot 10^{15}.$$

□

Exercise 3.

(a) Write a program that constructs a 50×50 matrix `A = U * S * V'`, where `U` and `V` are random orthogonal matrices and `S` is a diagonal matrix whose diagonal entries are random uniformly distributed numbers in $[0, 1]$, sorted into nonincreasing order. Have your program compute `[U2, S2, V2] = svd(A)` and the norms of `U - U2`, `V - V2`, `S - S2`, and `A - U2 * S2 * V2'`. Do this for five matrices `A` and comment on the results.

(b) Fix the signs in your computed SVD so that the difficulties of (a) go away. Run the program again for five random matrices and comment on the various norms. Do they have a connection with `cond(A)`?

(c) Replace the diagonal entries of S by their sixth powers and repeat (b). Do you see significant differences between the results of this exercise and those of the experiment for QR factorization?

Solution.

(a) The code for this problem is as follows:

```
U, X = np.linalg.qr(np.random.randn(n, n))
V, X = np.linalg.qr(np.random.randn(n, n))
S = np.diag(np.flip(np.sort(np.random.rand(n, 1), None)))
A = U @ S @ V.T

U2, S2, V2h = np.linalg.svd(A)
S2 = np.diag(S2)
V2 = V2h.T
```

This yielded, on average, an error of about 2 for U and V and an error of about 10^{-15} for S and A.

(b) Fixing the issues with (a) involves insuring that the columns of $U, U2$ and $V, V2$ have the same sign. This results in the following code:

```
U, X = np.linalg.qr(np.random.randn(n, n))
V, X = np.linalg.qr(np.random.randn(n, n))
S = np.diag(np.flip(np.sort(np.random.rand(n, 1), None)))
A = U @ S @ V.T

U2, S2, V2h = np.linalg.svd(A)
S2 = np.diag(S2)
V2 = V2h.T

flip = np.sign(np.diag(U2.T @ U))
U2 = U2 @ np.diag(flip)

flip = np.sign(np.diag(V2.T @ V))
V2 = V2 @ np.diag(flip)
```

This caused the error for U and V to decrease to about 10^{-12}. There is some variance though: in general, it seems as though the condition number of A is correlated to the error in U and V.

(c) Modifying the code for (b) yields the following:

```
U, X = np.linalg.qr(np.random.randn(n, n))
V, X = np.linalg.qr(np.random.randn(n, n))
S = np.power(np.diag(np.flip(np.sort(np.random.rand(n, 1), None))), 6)
A = U @ S @ V.T

U2, S2, V2h = np.linalg.svd(A)
S2 = np.power(np.diag(S2), 6)
V2 = V2h.T

flip = np.sign(np.diag(U2.T @ U))
U2 = U2 @ np.diag(flip)

flip = np.sign(np.diag(V2.T @ V))
V2 = V2 @ np.diag(flip)
```

This results in an error of about 10^{-7} for U and V, with larger condition numbers for A on average. This yields similar results to those in the experiment for QR factorization. $\square$