Michigan State University

CMSE 823 – Numerical Linear Algebra **Spring 2020**

Homework 6

Alexander Harnisch

1.

See the following two handwritten pages.

11.1. Suppose the $m \times n$ matrix A has the form

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where A_1 is a nonsingular matrix of dimension $n \times n$ and A_2 is an arbitrary matrix of dimension $(m-n) \times n$. Prove that $\|A^+\|_2 \le \|A_1^{-1}\|_2$.

When $m = n$ then $A = A_1$ and $A^+ = A_1^{-1} = A^{-1}$.

So we can assume $m > n$.

Let $A = QR = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} R$ be the reduced QR decomposition of A. Then:

$$A^+ = (A^* A)^{-1} A^* = (R^* Q^* Q R)^{-1} R^* Q^*$$
$$= R^{-1} Q^* = A_1^{-1} Q_1 Q^*$$

$$\Rightarrow \|A^+\|_2 \le \|A_1^{-1}\|_2 \|Q_1 Q^*\|_2$$

So we need to prove that $\|Q_1 Q^*\|_2 \le 1$

Recall:

These processes of multiplication by a unitary matrix or its adjoint preserve geometric structure in the Euclidean sense, because inner products are preserved. That is, for unitary Q,

$$(Qx)^*(Qy) = x^* y, \tag{2.9}$$

as is readily verified by (2.4). The invariance of inner products means that angles between vectors are preserved, and so are their lengths:

$$\|Qx\| = \|x\|. \tag{2.10}$$

In the real case, multiplication by an orthogonal matrix Q corresponds to a rigid rotation (if $\det Q = 1$) or reflection (if $\det Q = -1$) of the vector space.

Since the colums of Q are orthonormal.
So we can allways construct an orthonormal $m \times m$ matrix (Q_1, B), with B $m\times(m-n)$.
(Like for the full QR-decomposition).

Then for any $x \in \mathbb{C}^m$:

orthonormal, conserves norm

$$\|Q_1 Q_1^* x\|_2 \leq \|(Q_1, B)\begin{pmatrix} Q_1^* x \\ 0 \end{pmatrix}\|_2 = \|\begin{pmatrix} Q_1^* x \\ 0 \end{pmatrix}\|_2 \leq \|\begin{pmatrix} Q_1^* x \\ B^* x \end{pmatrix}\|_2$$

$$= \|(Q_1, B)^* x\|_2 = \|x\|_2$$

$$\Rightarrow \|Q_1 Q_1^*\|_2 \leq 1$$

$$\Rightarrow \|A^+\|_2 \leq \|A_1^{-1}\|_2 \|Q_1 Q_1^*\|_2 \leq \|A_1^{-1}\|_2 \quad \square$$

2.

12.3. The goal of this problem is to explore some properties of random matrices. Your job is to be a laboratory scientist, performing experiments that lead to conjectures and more refined experiments. Do not try to prove anything. Do produce well-designed plots, which are worth a thousand numbers.

Define a *random matrix* to be an $m \times m$ matrix whose entries are independent samples from the real normal distribution with mean zero and standard deviation $m^{-1/2}$. (In MATLAB, `A = randn(m,m)/sqrt(m)`.) The factor $\sqrt{m}$ is introduced to make the limiting behavior clean as $m \to \infty$.

(a) What do the eigenvalues of a random matrix look like? What happens, say, if you take 100 random matrices and superimpose all their eigenvalues in a single plot? If you do this for $m = 8, 16, 32, 64, \ldots$, what pattern is suggested? How does the spectral radius $\rho(A)$ (Exercise 3.2) behave as $m \to \infty$?

(b) What about norms? How does the 2-norm of a random matrix behave as $m \to \infty$? Of course, we must have $\rho(A) \leq \|A\|$ (Exercise 3.2). Does this inequality appear to approach an equality as $m \to \infty$?

(c) What about condition numbers—or more simply, the smallest singular value $\sigma_{\min}$? Even for fixed m this question is interesting. What proportions of random matrices in $\mathbb{R}^{m \times m}$ seem to have $\sigma_{\min} \leq 2^{-1}, 4^{-1}, 8^{-1}, \ldots$? In other words, what does the tail of the probability distribution of smallest singular values look like? How does the scale of all this change with m?

(d) How do the answers to (a)–(c) change if we consider random triangular instead of full matrices, i.e., upper-triangular matrices whose entries are samples from the same distribution as above?

For all of the sub-parts of this part I chose m to be all powers of 2 from 2 to 512. For each m I generate 10^4 matrices, which is by no means large enough to draw any statistically relevant conclusions. The results dependent on m for **(a)** to **(c)** are shown in Figure 1 and Figure 2.

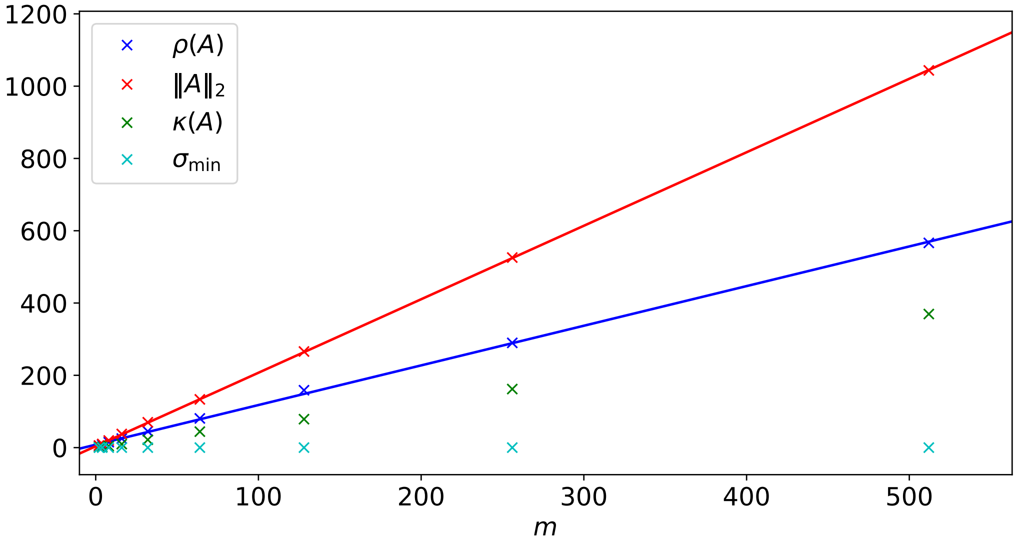

Figure 1: The norm, spectral radius, condition number and smallest singular value out of a sample of square Gaussian random matrices in dependence of their dimension m.

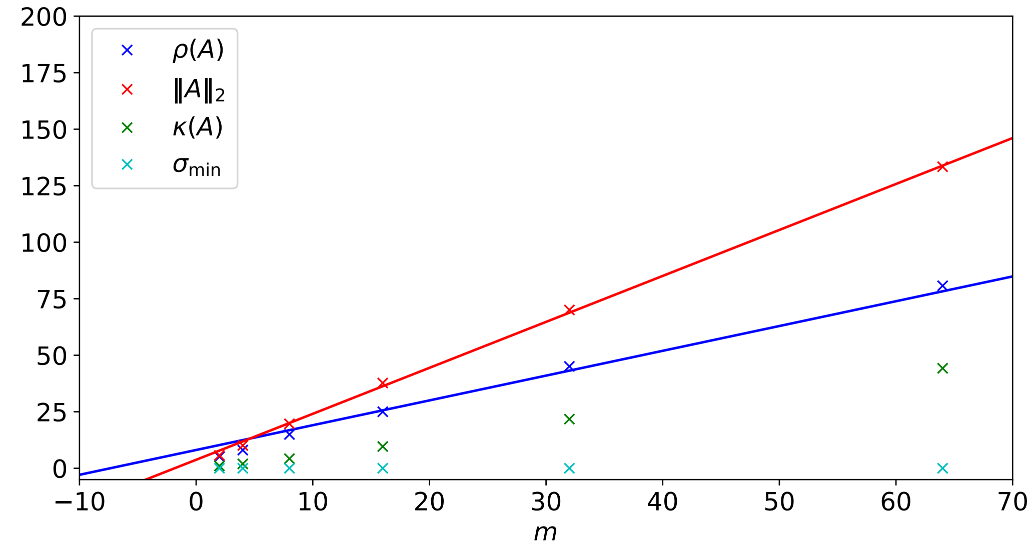

Figure 2: Zoomed in view for small m of Figure 1.

(a)

Plots of the complex plane with all eigenvalues for each m are shown in Figure 3 to Figure 11. It is obvious, that most of the eigenvalues seem to be almost uniformly distributed within a circle of radius m. Meaning that there absolute value is most likely to be smaller or equal than m and that the probability of getting an eigenvalue with absolute value $r \leq m$ increases with the square of r until $r = m$ where it suddenly falls off. However, real eigenvalues seem to be more likely than eigenvalues close to the real axis and they also seem to be more likely to fall outside of the circle (meaning their absolute value being larger than m). This effect seems to be more pronounced for small m and starts to asymptotically disappear for large m in relative terms.

If I had the time I would look at the PDFs of the absolute value of the eigenvalues and their complex phase. As mentioned before I would expect a square distribution for the absolute value with a steep cut-off at m. For the distribution of complex phase I would expect a mostly uniform distribution, with a slight peak at 0 and small dips close around it.

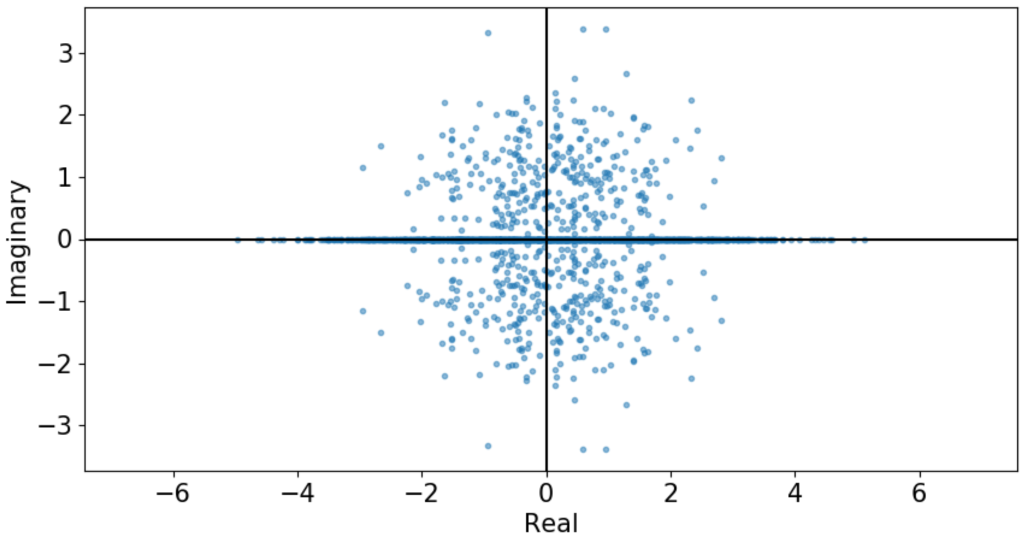

Figure 3: All eigenvalues of the sample with $m = 2$.

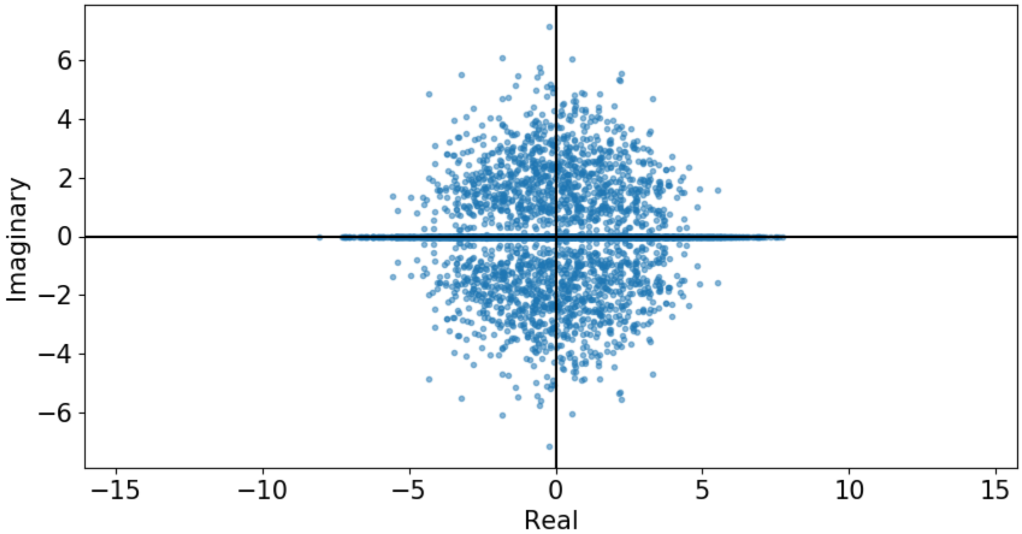

Figure 4: All eigenvalues of the sample with $m = 4$.

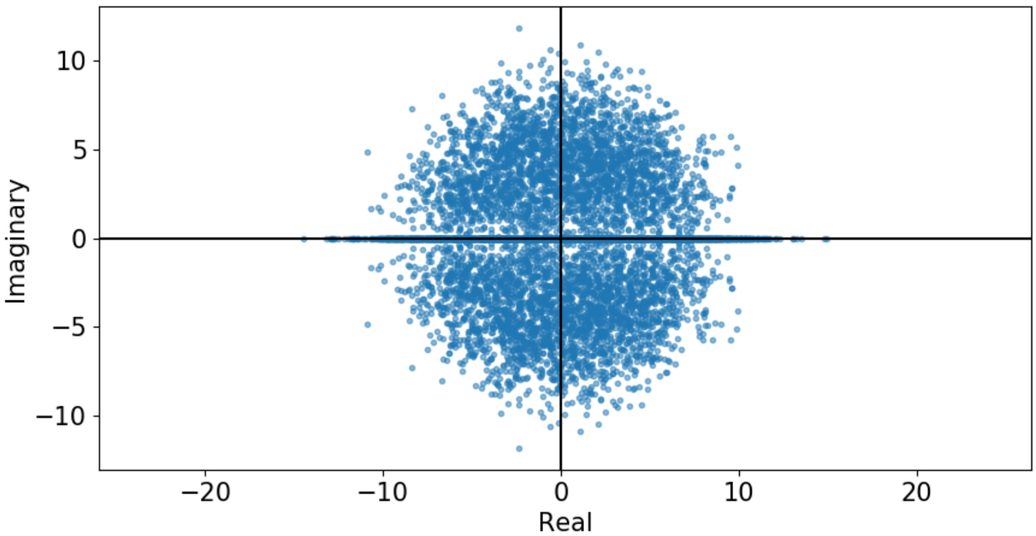

Figure 5: All eigenvalues of the sample with $m = 8$.

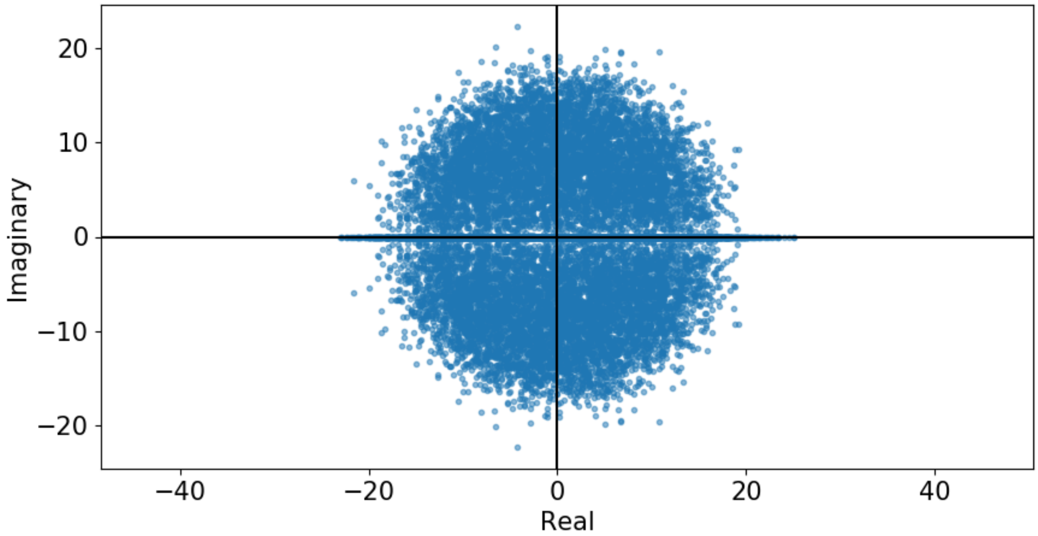

Figure 6: All eigenvalues of the sample with $m = 16$.

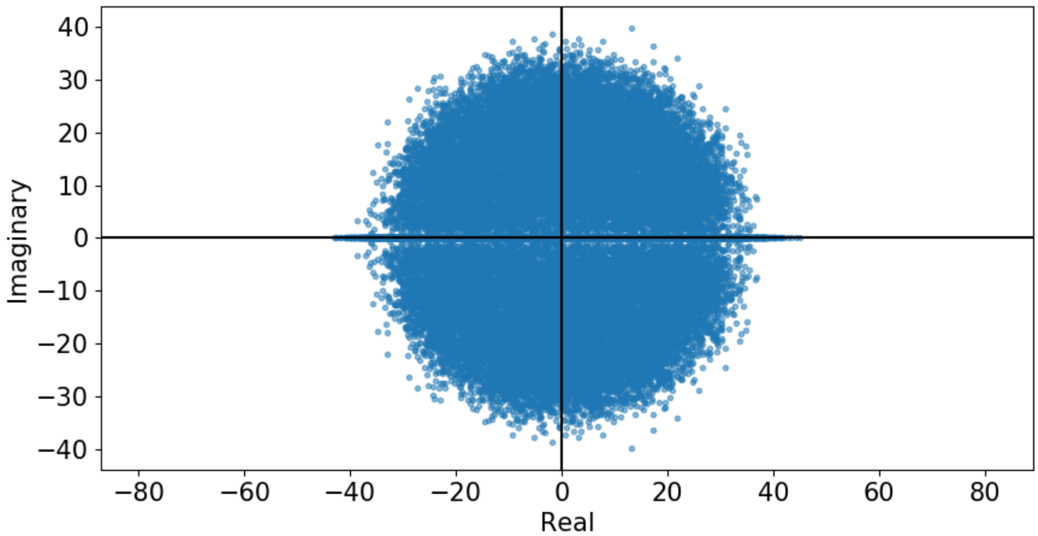

Figure 7: All eigenvalues of the sample with $m = 32$.

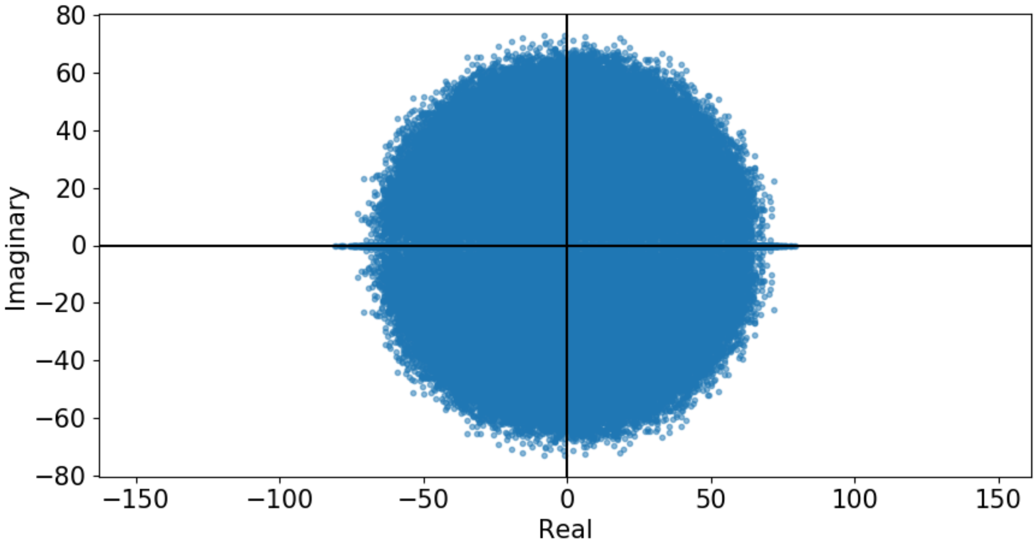

Figure 8: All eigenvalues of the sample with $m = 64$.

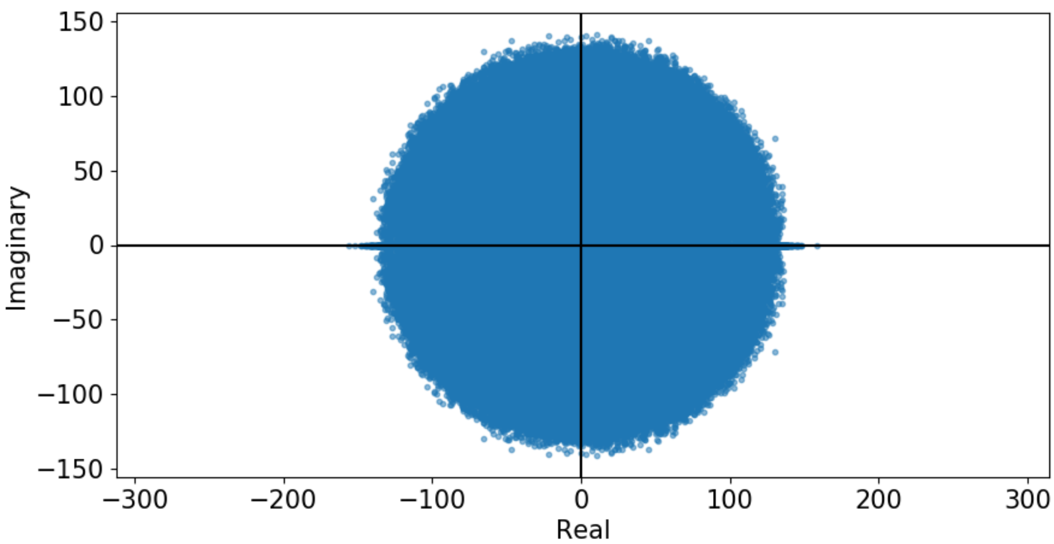

Figure 9: All eigenvalues of the sample with $m = 128$.

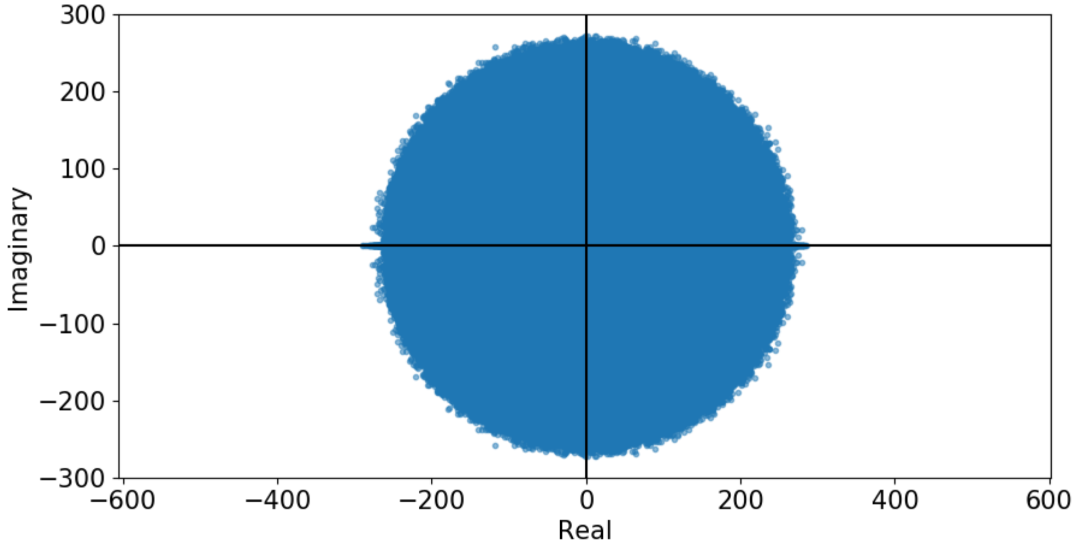

Figure 10: All eigenvalues of the sample with $m = 256$.

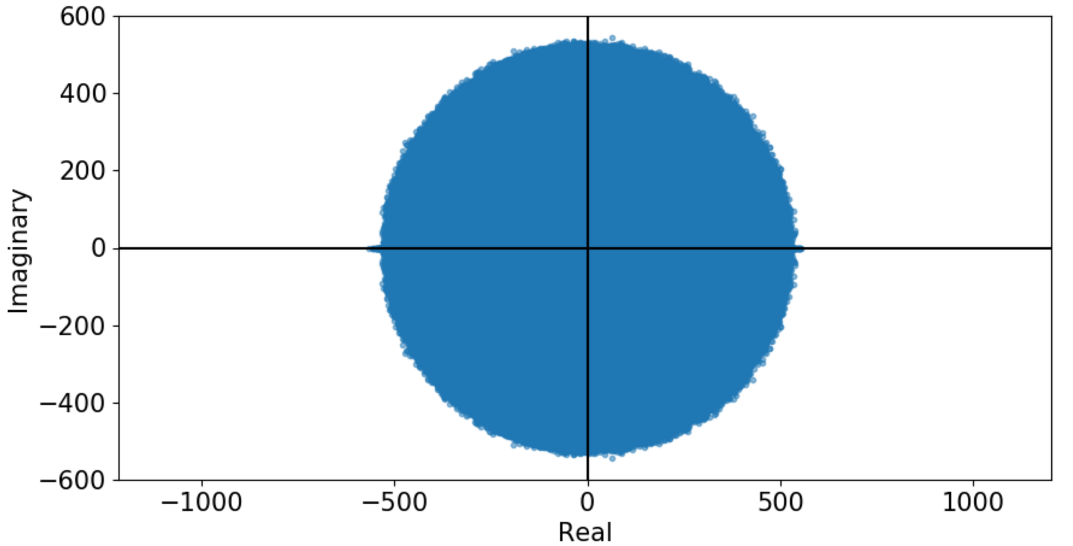

Figure 11: All eigenvalues of the sample with $m = 512$.

b)

I know we are not supposed to prove anything, but the inequality

$$\rho(A) \leq \|A\| \tag{1}$$

could not possibly approach an equality for $m \to \infty$ because it in fact holds true for the case $m = 1$ (which is immediately obvious). The inequality grows larger with m as it can be seen in Figure 1.

Both the L2 norm and the spectral radius appear to grow proportionally with m, as the linear regressions shown in Figure 1 suggest. I find

$$\rho(A) \approx 1.10 \cdot m + 8.10 \tag{2}$$

with a standard error of 10^{-2} and

$$\|A\| \approx 2.033 \cdot m + 3.762 \tag{3}$$

with a standard error of $3 \cdot 10^{-3}$.

(c)

The minimal condition numbers $\kappa(A)$ and smallest singular values $\sigma_{\min}$ for the different samples with varied m are also shown in Figure 1. They both do not seem to follow any obvious trend. However, especially for the smallest singular values the result is extremely volatile between runs of the script. A zoomed in view is given by Figure 12. This figure can not be used to draw any conclusions as the statistics are not sufficient.

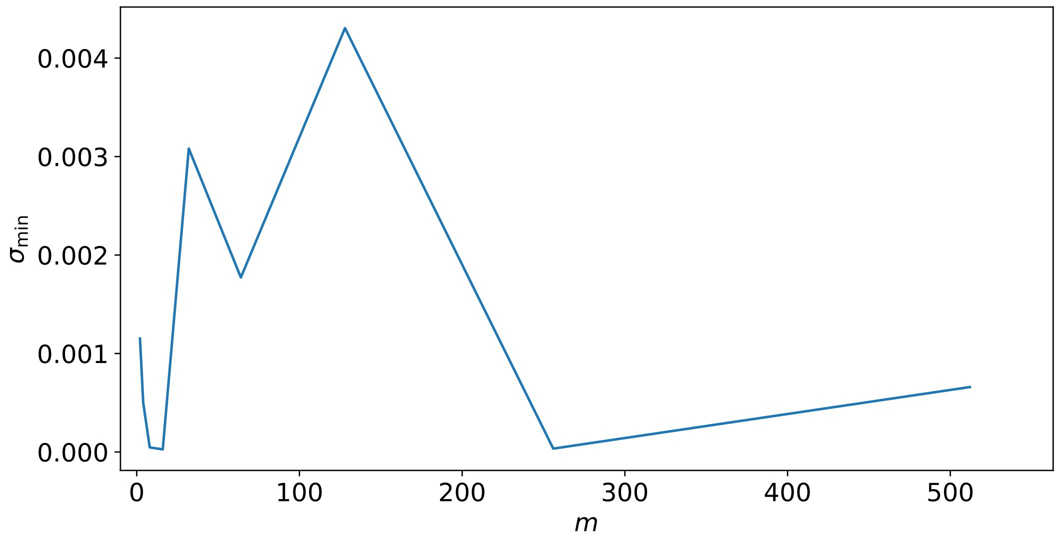

Figure 12: The minimal singular values of all samples. This figure is just an example and looks very different for every run.

To learn about how the minimal singular values are effected by m, it makes a lot more sense to look at the distributions as suggested by the assignment. Figure 13 to Figure 21 show the empirical cumulative density functions of the smallest singular values for all matrices in the respective sample. The distribution does not seem to be effected by m.

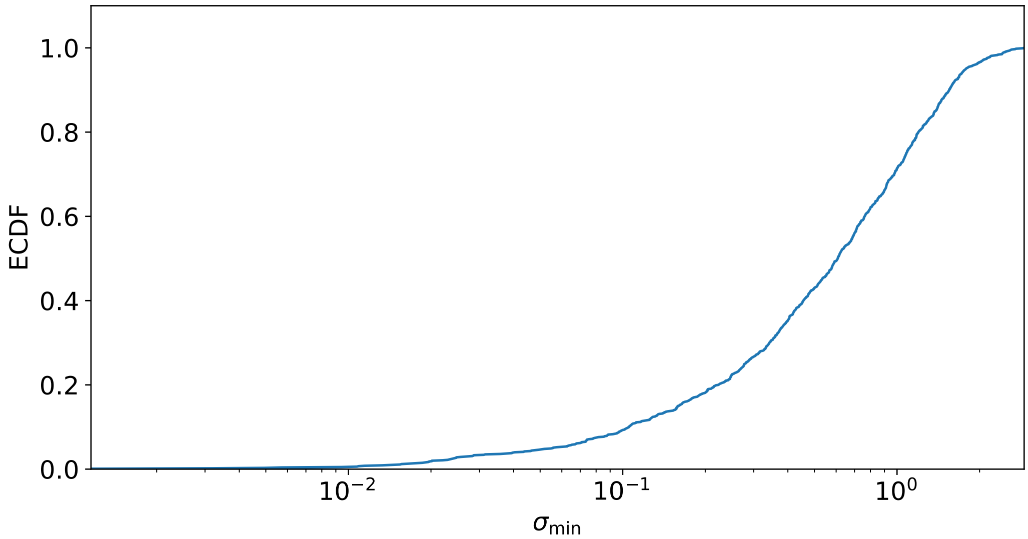

Figure 13: ECDF of the smallest singular values for all matrices in the sample with $m = 2$.

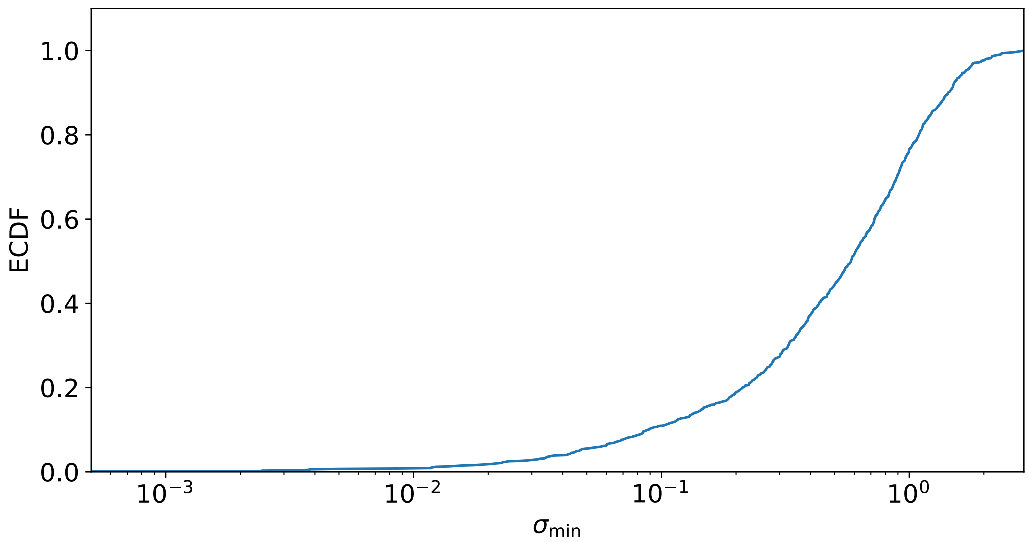

Figure 14: ECDF of the smallest singular values for all matrices in the sample with $m = 4$.

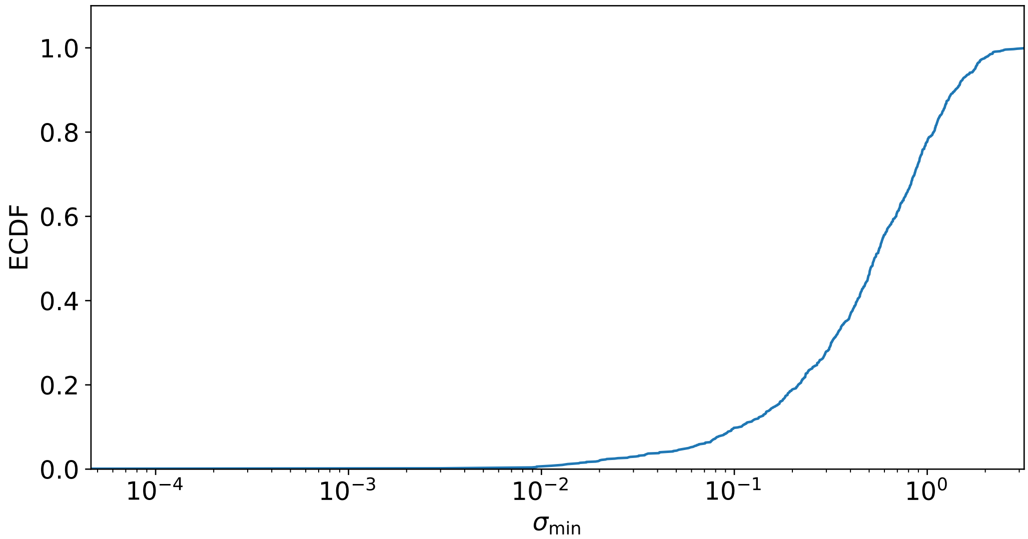

Figure 15: ECDF of the smallest singular values for all matrices in the sample with $m = 8$.

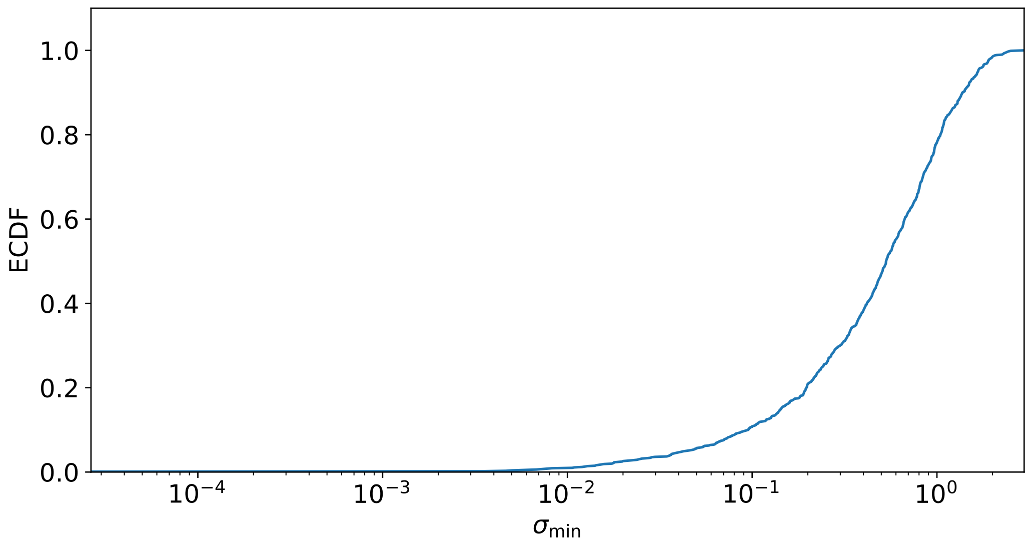

Figure 16: ECDF of the smallest singular values for all matrices in the sample with $m = 16$.

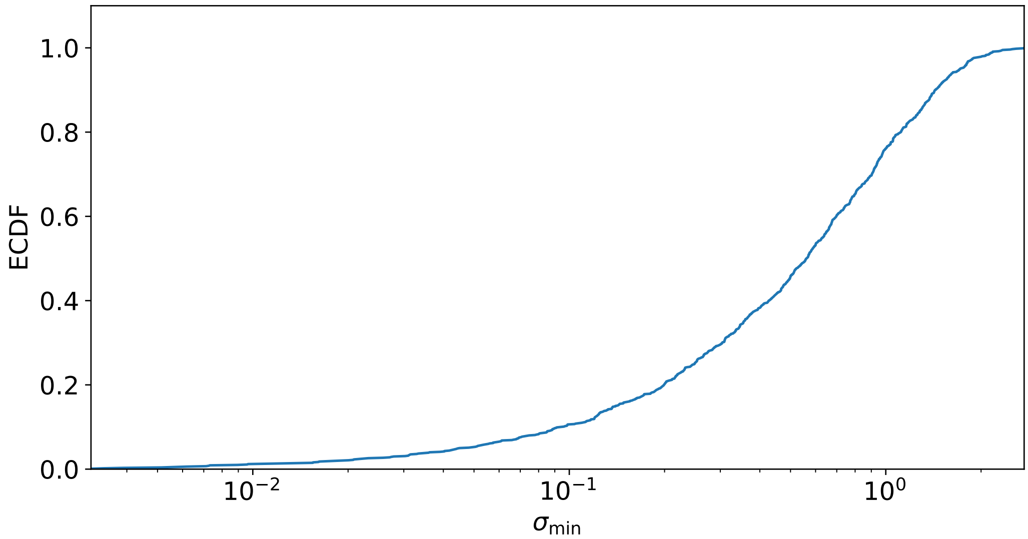

Figure 17: ECDF of the smallest singular values for all matrices in the sample with $m = 32$.

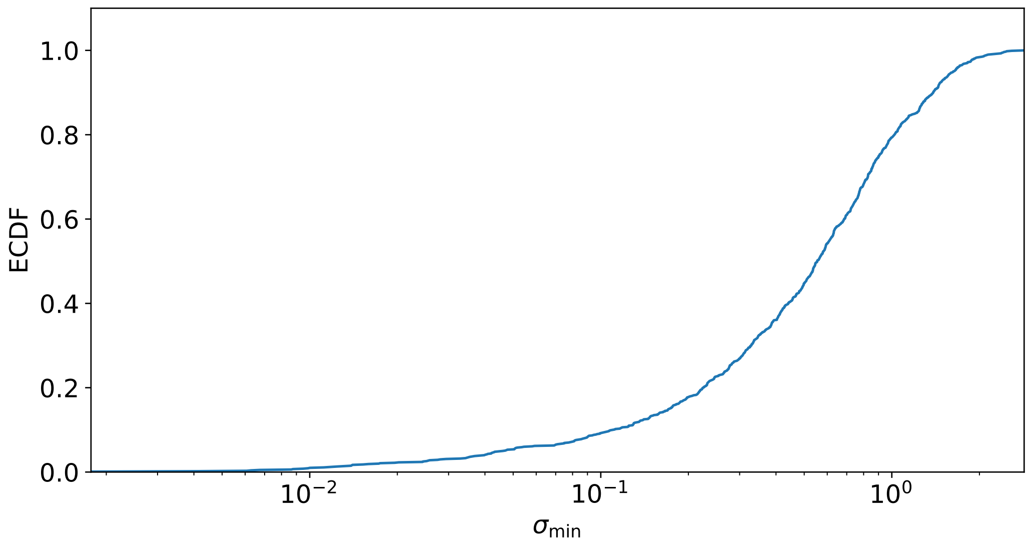

Figure 18: ECDF of the smallest singular values for all matrices in the sample with $m = 64$.

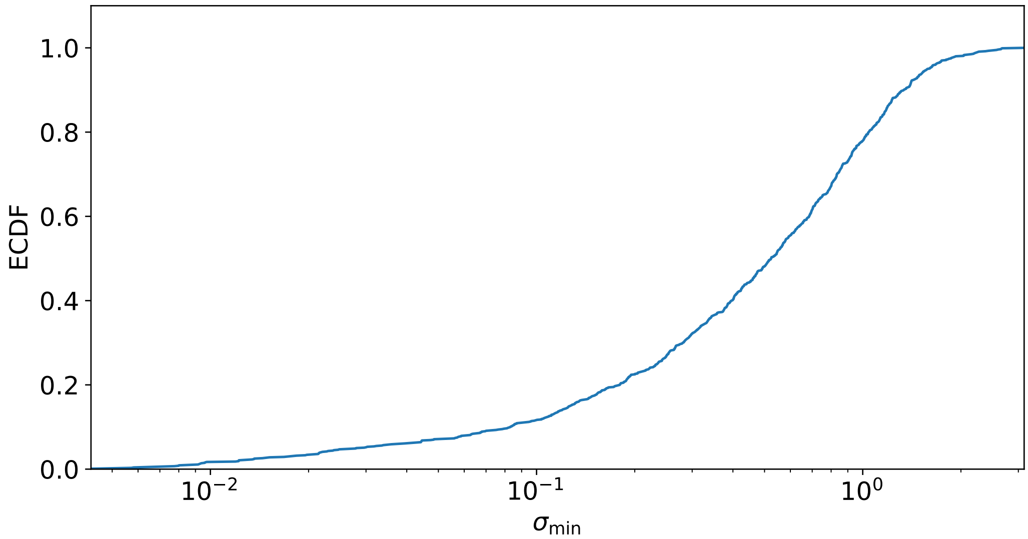

Figure 19: ECDF of the smallest singular values for all matrices in the sample with $m = 128$.

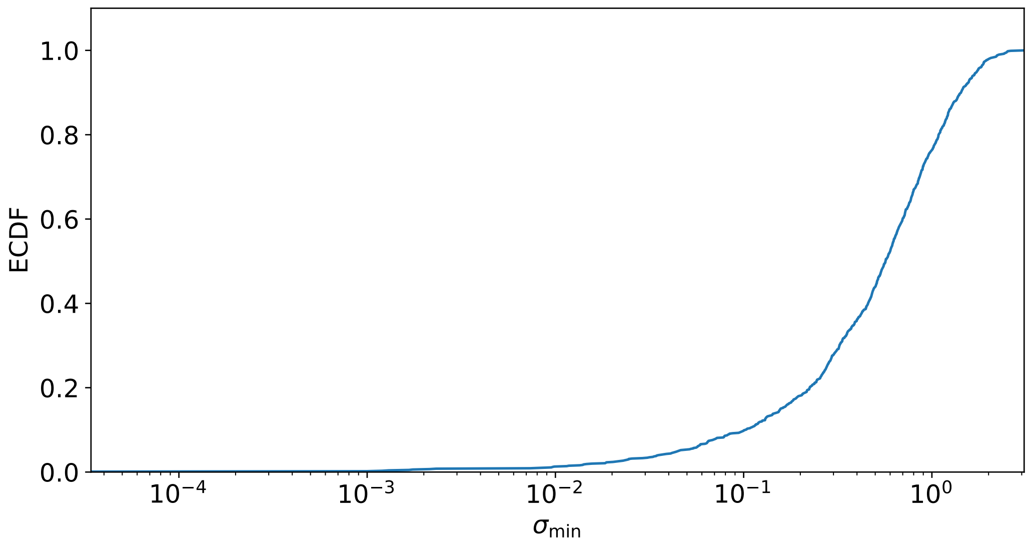

Figure 20: ECDF of the smallest singular values for all matrices in the sample with $m = 256$.

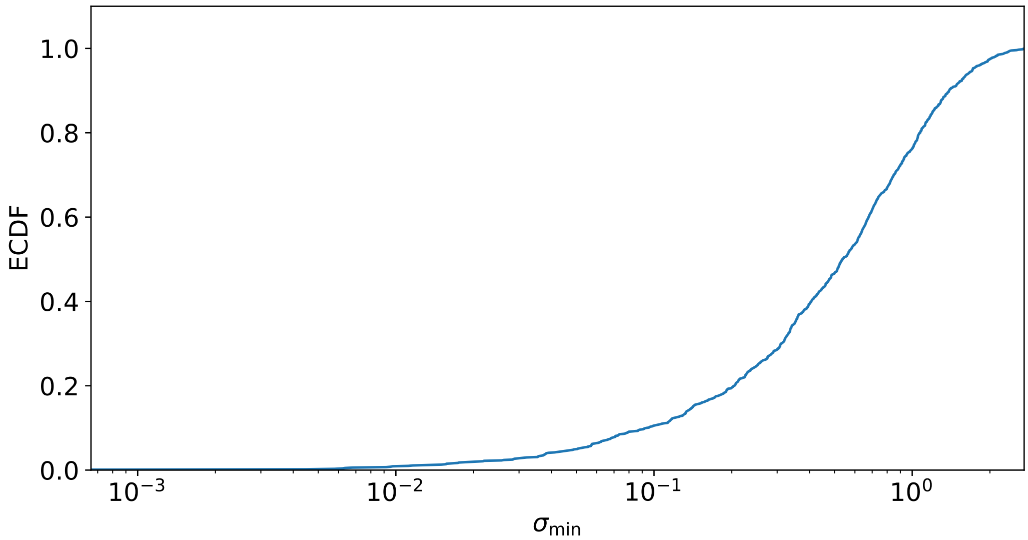

Figure 21: ECDF of the smallest singular values for all matrices in the sample with $m = 512$.

(d)

Okay now let's see what changes for upper triangular Guassian random matrices. Obviously now we only get real eigenvalues. So the plots from **(a)** are a waist of space and I won't include them here for that reason. However, the behavior for eigenvalues on the real axis (so all of them here) seems the same as for the square matrices.

The same plots as before are provided in Figure 22 and Figure 23 except that the condition numbers are plotted seperately in Figure 24, as they become to large to be shown in the same plots.

Again, both the L2 norm and the spectral radius appear to grow proportionally with m, as the linear regressions shown in Figure 1 suggest. Here I find

$$\rho(A) \approx 0.22 \cdot m + 13.11 \tag{4}$$

with a standard error of $2 \cdot 10^{-2}$ and

$$\|A\| \approx 1.680 \cdot m + 4.479 \tag{5}$$

with a standard error of $4 \cdot 10^{-3}$. The general behavior is similar to the square matrix case. It just appears as the spectral radius is now growing significantly slower with m.

Where we do see a significant change is in the distribution of minimal singular values. Again the empirical CDFs are shown in Figure 25 to Figure 33. The tail of the distribution for smallest singular values seems to extend and approach 0 as m grows. We can also see that by looking at Figure 24: The triangular matrices for large m are significantly closer to being singular compared to their square counterparts.

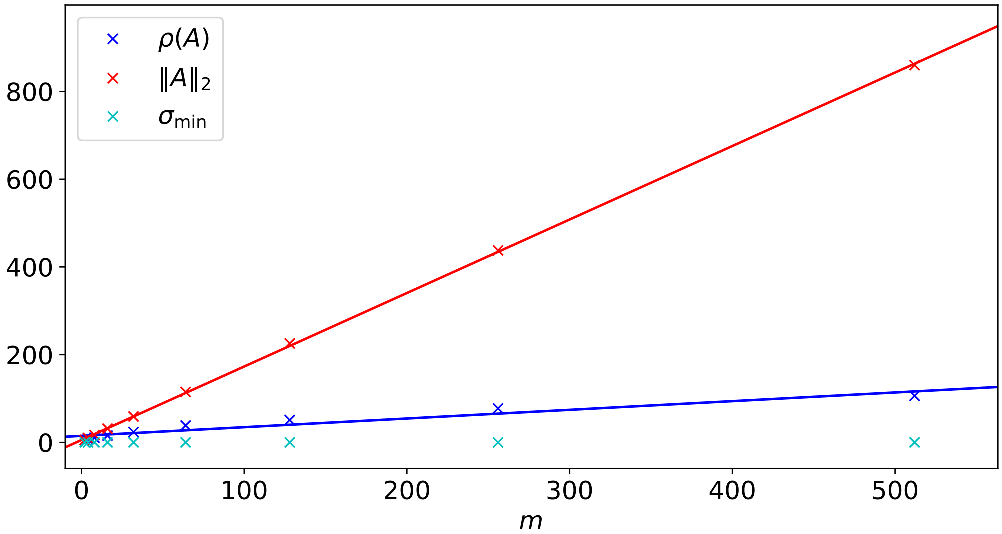

Figure 22: The norm and spectral index for triangular Gaussian random matrices with different m.

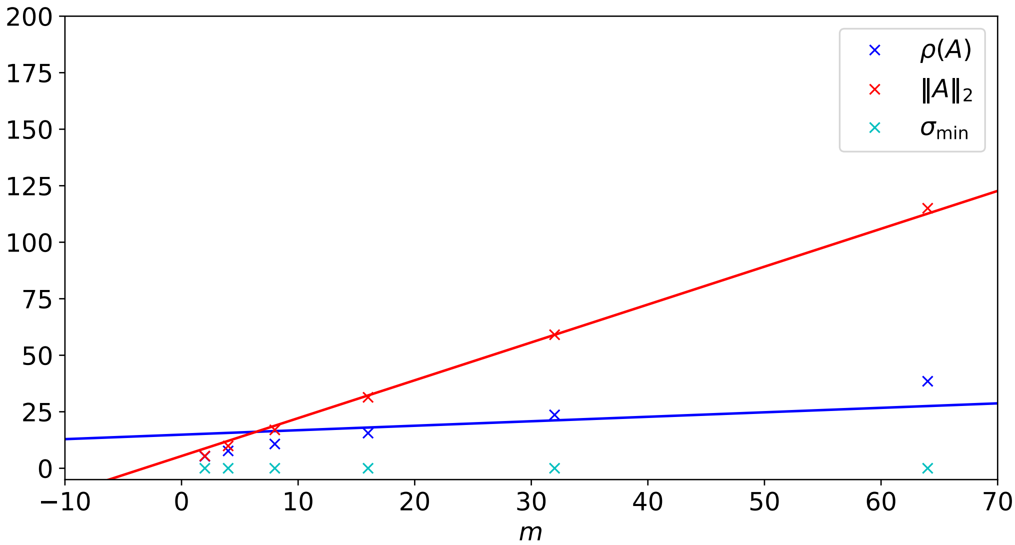

Figure 23: Zoomed in view of Figure 22.

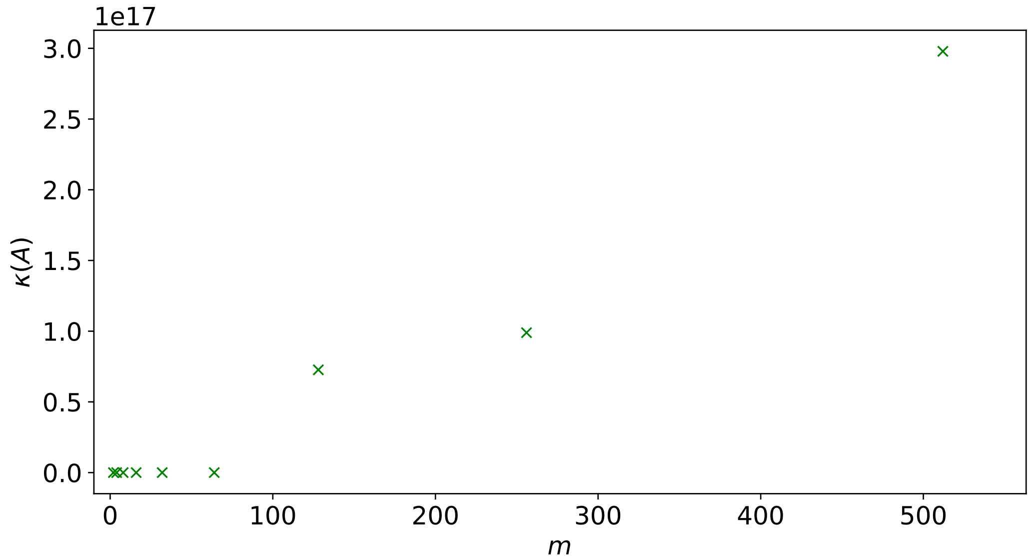

Figure 24: The condition numbers for the same triangular Gaussian random matrices.

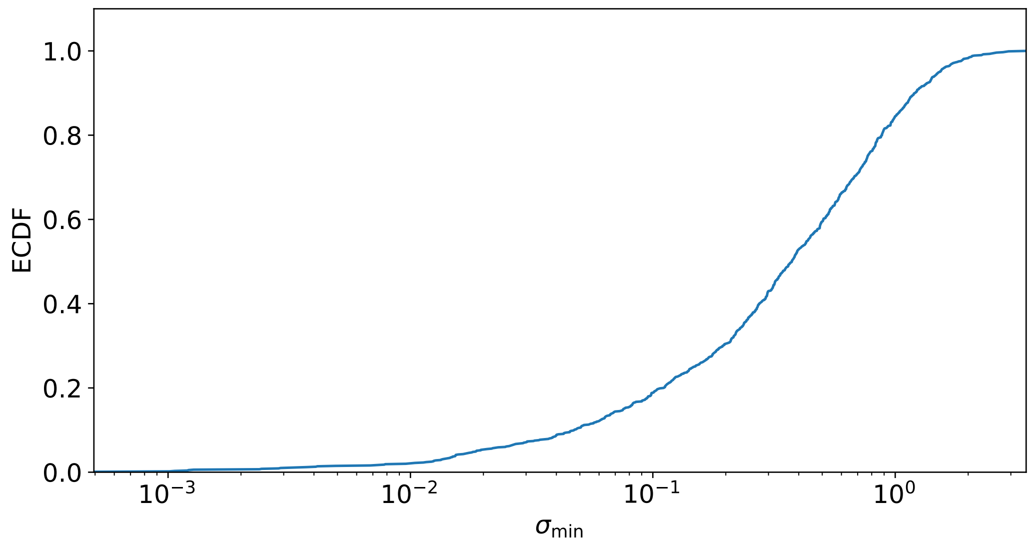

Figure 25: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 2$.

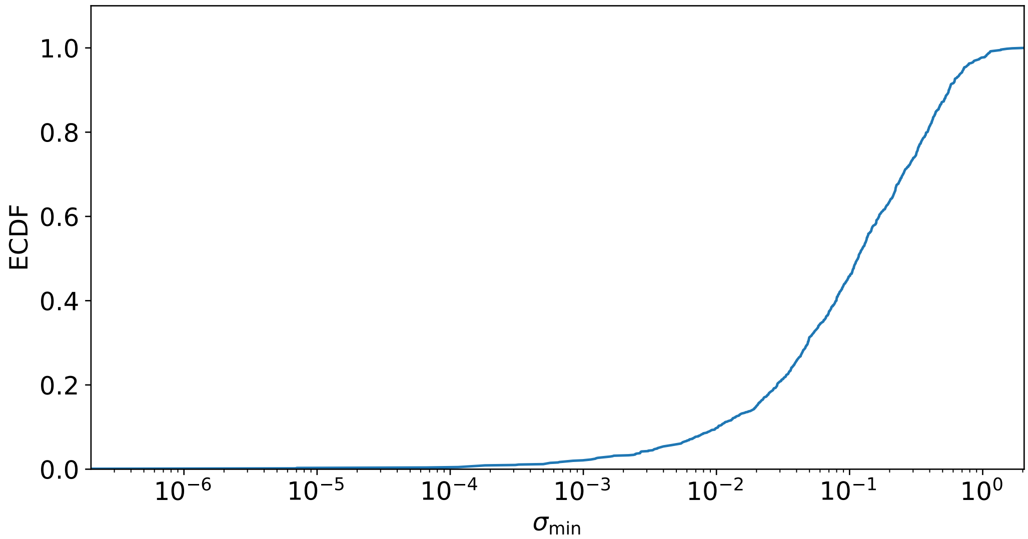

Figure 26: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 4$.

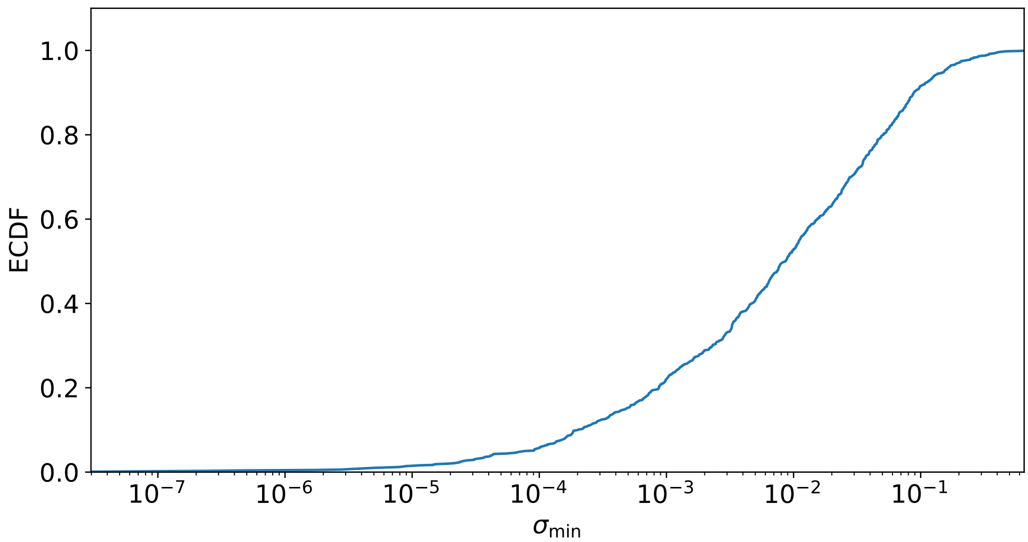

Figure 27: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 8$.

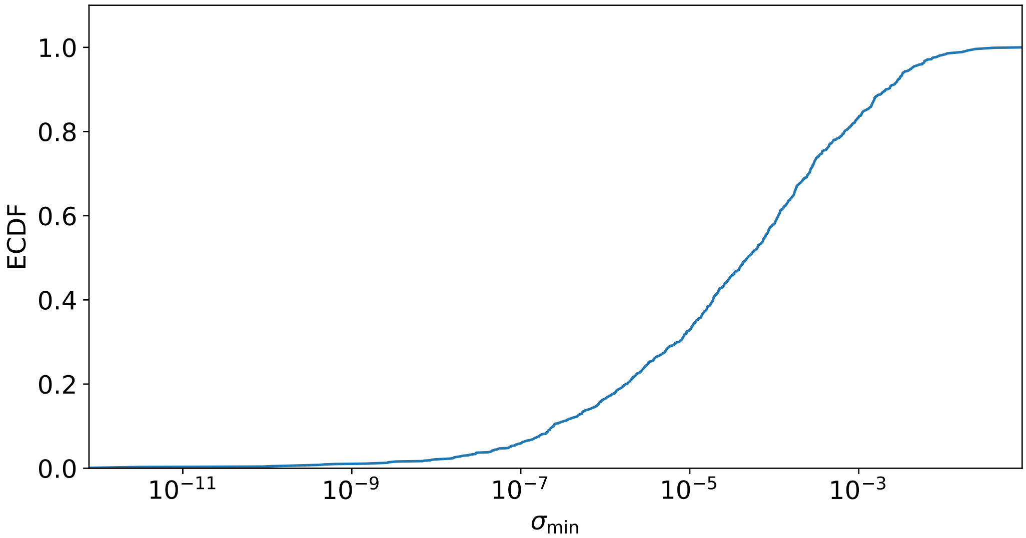

Figure 28: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 16$.

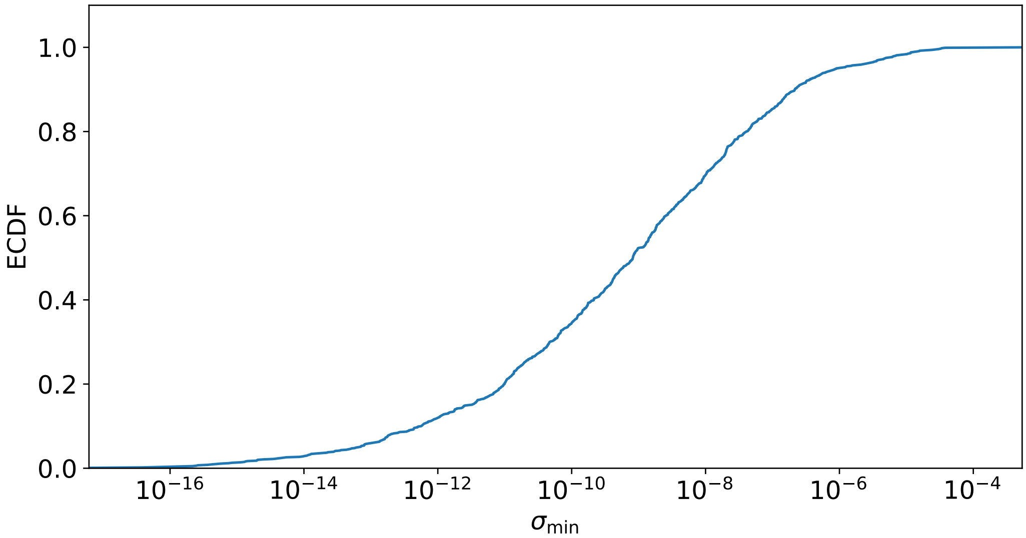

Figure 29: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 32$.

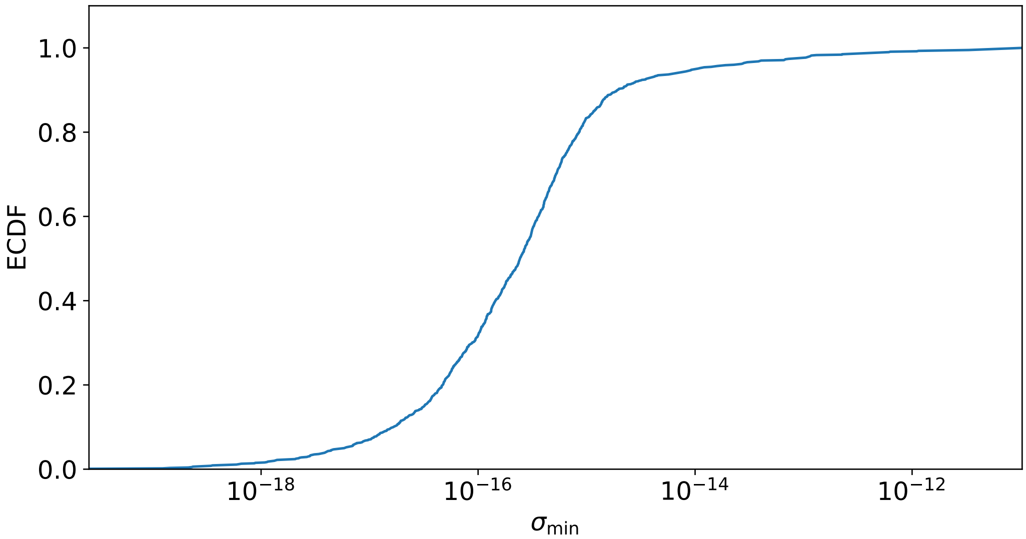

Figure 30: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 64$.

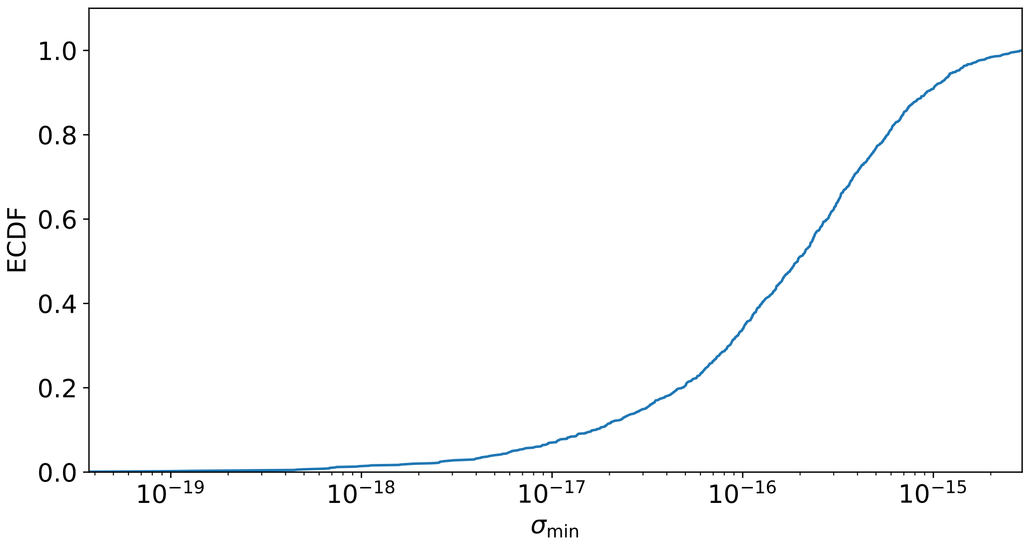

Figure 31: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 128$.

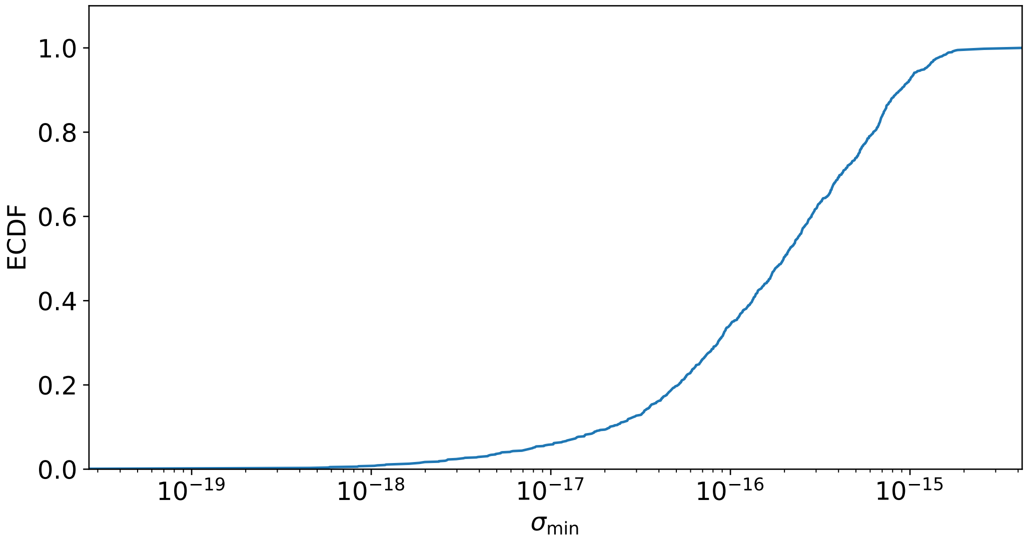

Figure 32: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 256$.

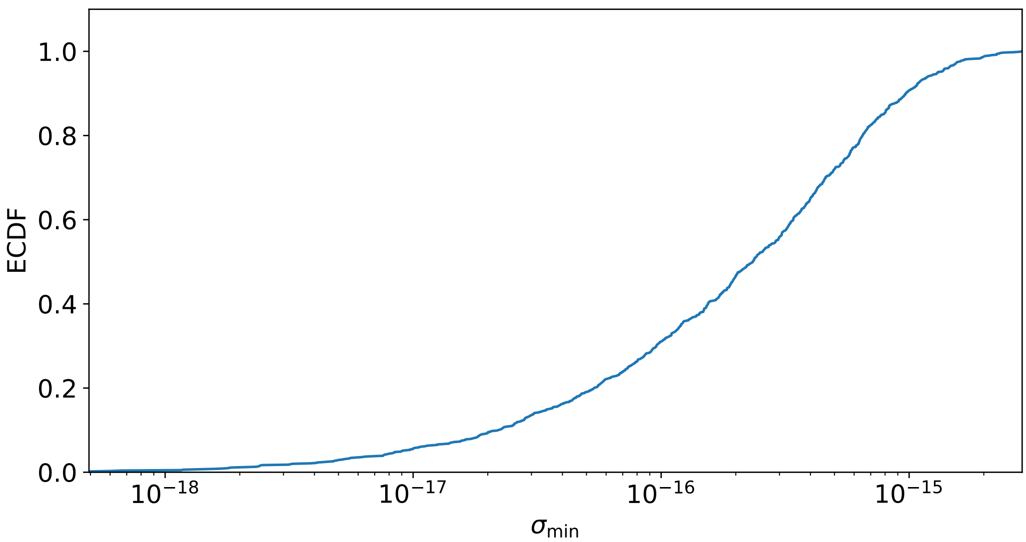

Figure 33: ECDF of the smallest singular values for all triangular matrices in the sample with $m = 512$.

3.

13.3. **Consider the polynomial $p(x) = (x-2)^9 = x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$.**

(a) Plot $p(x)$ for $x = -1.920, -1.919, -1.918, \ldots, 2.080$, evaluating p via its coefficients 1, -18, 144, ...

(b) Produce the same plot again, now evaluating p via the expression $(x-2)^9$.

Terrible assignment, probably due to the book being so old. I'll just do it and it is worth full points, obviously not what the assignment is aiming for. It is not saying what data types to use and it is also not asking for any discussion. The assignment is literally just asking to create the two plots using the different functions. Using any modern language the two plots will look exactly the same. Obviously it is aiming at showing that the coefficient form will be less accurate than the factorize form because of numerical instability for the values close to zero. However, without using some ancient float data type and/or a terrible language/library, that's simply not the case anymore today. *Numpy* handles such scenarios with ease (I would assume primarily because of proper use of guard digits, but I don't really know the details of *Numpy*'s implementation), even when using single precision floats (which I did even though the assignment does not ask for it). To actually see a difference one has to use half precision, however in that case we actually get overflows and the range that is asked for by the assignment can't even be represented. So here you go, two identical plots in Figure 34 and Figure 35 :D

(a)

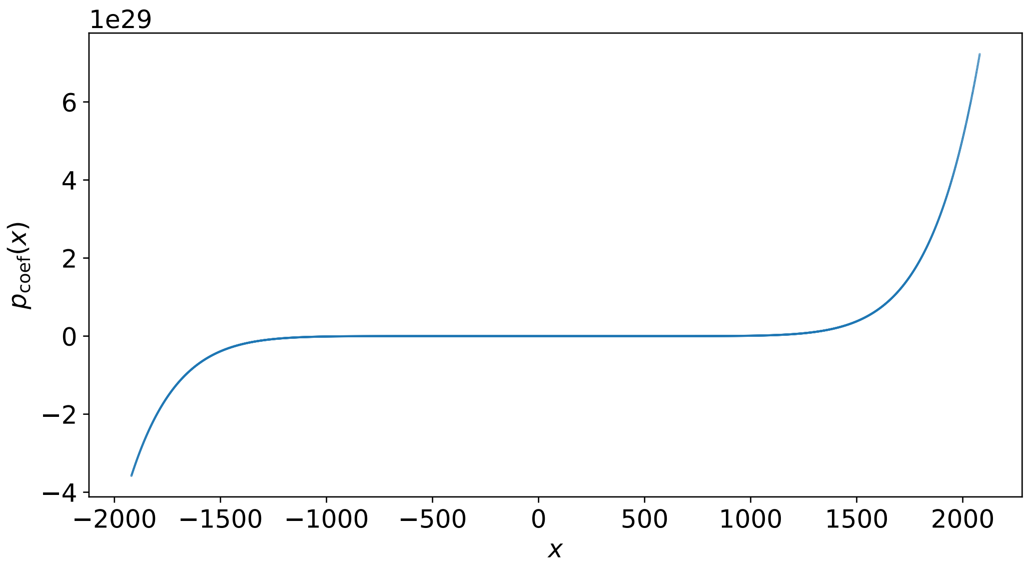

Figure 34: Plot of the polynomial using the coefficient form for evaluation and single precision floating point numbers.

(b)

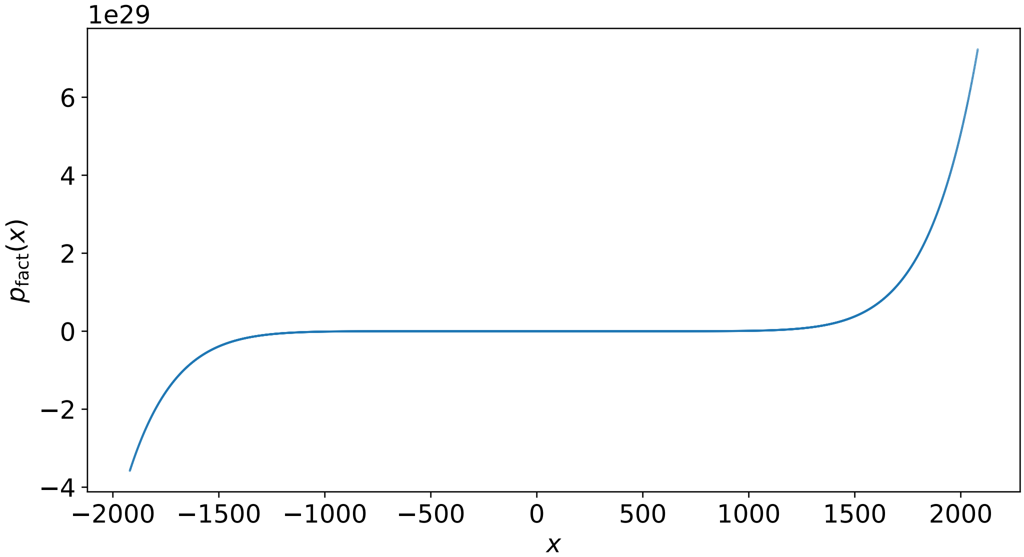

Figure 35: Plot of the polynomial using the factorized form for evaluation and single precision floating point numbers.

For good measure (even though this terribly designed assignment does not ask for it) I include a plot showing the difference between the two forms using single precision in Figure 36. We can see the usual pattern arising due to numerical instability when subtracting equally large numbers. However, the range where the polynomial is zero is totally fine, even when subtracting the two representations which are both numerically zero from each other. Trivial scenarios like this are simply not an issue any more for any modern computing system.

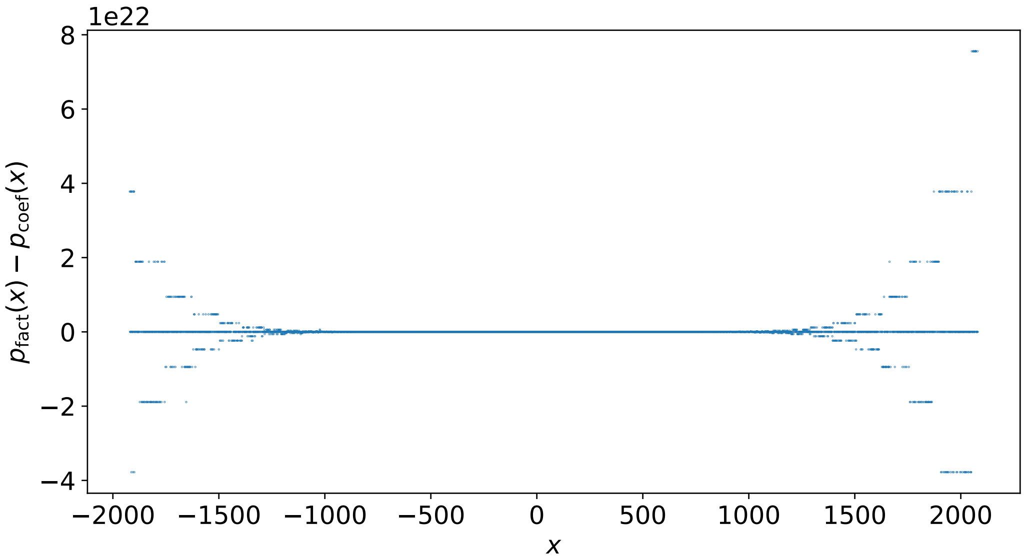

Figure 36: Difference between the coefficient and factorized form of the polynomial using single precision.

4.

None of this has been taught in class in time to finish the assignment! See the following handwritten pages.

4. Using single precision, evaluate the expression by hand only,

$$a = 1000\left(\frac{c}{\sqrt{b^2+c}-b} - 2b\right)$$

when $b = 1$ and $c = 0.004004$. Compare the computed value of a with the exact value $a = 2$. Show that a can be written

$$a = \frac{1000c}{\sqrt{b^2+c}+b}.$$

Now evaluate a again when $b = 1$ and $c = 0.004004$. Explain why this second expression is more accurate.

Since the floating point standard is not specified, I'll simply use

For all $x \in \mathbb{R}$, there exists ϵ with $|\epsilon| \le \epsilon_{\text{machine}}$ such that $\text{fl}(x) = x(1+\epsilon)$. **(13.5)**

and

Fundamental Axiom of Floating Point Arithmetic

For all $x, y \in \mathbf{F}$, there exists ϵ with $|\epsilon| \le \epsilon_{\text{machine}}$ such that

$$x \circledast y = (x * y)(1+\epsilon). \quad \textbf{(13.7)}$$

With a typical $\varepsilon_{\text{macine}} =: \varepsilon_0 = 2^{-24}$ for single precision.

I'll do a worst case scenario analysis when $\varepsilon = \varepsilon_0$ in every calculation. It is easy to see then why the second expression is more accurate.

So far: Let's call $1+\varepsilon = \varepsilon_r$

$$a = 1000\left(\frac{c}{\sqrt{b^2+c}-b} - 2b\right) \quad , \quad b = 1 \text{ and } c = 0.004004$$

$b \to b\,\varepsilon_r \quad , \quad c \to c\,\varepsilon_r$

$2b \to (2\varepsilon_r \cdot b\,\varepsilon_r)\,\varepsilon_r$

$\quad 2b\,\varepsilon_r^3$

$b^2 \to b\,\varepsilon_r^3 \quad , \quad b^2 + c \to (b\,\varepsilon_r^3 + c\,\varepsilon_r)\,\varepsilon_r$

$\quad = b\,\varepsilon_r^4 + c\,\varepsilon_r^2$

For simplicity let's assume $\sqrt{\cdot}$ is also a fundamental operation. Then:

$\sqrt{b^2+c} \to \sqrt{b\,\varepsilon_r^4 + c\,\varepsilon_r^2}\;\varepsilon_r$ and so on.

In the end we get:

$$a \to \varepsilon_r^4\, 1000\left(\frac{c\,\varepsilon_r^{\cancel{2}}}{\sqrt{b\,\varepsilon_r^4 + c\,\varepsilon_r^2}\;\cancel{\varepsilon_r}} - 2b\,\varepsilon_r^3\right)$$

So a worst case rounding error of $O(\varepsilon_r^7)$

For

$$a = \frac{1000c}{\sqrt{b^2+c}+b}.$$

we get

$$a \to \frac{1000\, c\, \varepsilon_r^{3\,2}}{\Big(\underbrace{\sqrt{b\varepsilon_r^4 + c\varepsilon_r^2}}_{O(\varepsilon_r^2)} + \underbrace{b\varepsilon_r}_{O(\varepsilon_r)}\Big)\cancel{\varepsilon_r}}$$

$$\to \frac{O(\varepsilon_r^2)}{O(\varepsilon_r)} = O(\varepsilon_r)$$

So $O(\varepsilon_r)$ here, which is a lot better. This is simply due to less operations in a way that the rounding errors can cancel.

In numbers using Numpy's single precision:

$$a = 1000\left(\frac{c}{\sqrt{b^2+c}-b} - 2b\right) \quad = 2.0000001100\boxed{59709}$$

$$a = \frac{1000c}{\sqrt{b^2+c}+b}. \quad = 2.0000001100\boxed{038816}$$