Fall 2022: Numerical Analysis
Assignment 3 (due Oct 18, 2022 at 11:59pm ET)

1. **Matrix condition numbers, [2+1+2pt+2pt (extra credit)]** Let us explore matrix norms and condition numbers.

   (a) For the following matrix given by
   $$A = \begin{bmatrix} 1 & -2 \\ 6 & -2 \end{bmatrix},$$
   calculate $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$ as well as the condition numbers for each norm by hand. Is A well or poorly conditioned?[1]

   (b) Recall the formulas from Theorems 2.7 and 2.8 in the text book. If you assume that taking the absolute value and determining the maximum does not contribute to the overall computational cost, how many *flops* (floating point operations) are needed to calculate $\|A\|_1$ and $\|A\|_\infty$ for $A \in \mathbb{R}^{n\times n}$? By what factor will the calculation time increase when you double the matrix size?

   (c) Now implement a simple code that calculates $\|A\|_1$ and $\|A\|_\infty$ for a matrix of any size $n \geq 1$. Try to do this without using loops[2]! Using system sizes of $n_1 = 100$, $n_{k+1} = 2n_k, k = 1, \ldots, 7$, determine how long your code takes[3] to calculate $\|A\|_1$ and $\|A\|_\infty$ for a matrix $A \in \mathbb{R}^{n_i \times n_i}$ with random entries and report the results. Can you confirm the estimate from (b)?

   (d) **(extra credit)** MATLAB/Python have the build-in function `norm` and `np.linalg.norm` to calculate matrix norms. Calculate for the system sizes in (c) $\|A\|_1$ and $\|A\|_\infty$ using both your implementation and the built-in `norm` function, determine for each n_i how long each code takes and plot the results in one graph. On average, by what factor is MATLAB or Python's implementation faster than yours?

   Please also hand in your code.

2. **Induced matrix norms, [2+2+1+1pt]** Let $A, B \in \mathbb{R}^{n\times n}$ and let the matrix norm $\|\cdot\|$ be induced by/subordinate of a vector norm $\|\cdot\|$.

   (a) Show that $\|AB\| \leq \|A\|\|B\|$.

   (b) For the identity matrix $I \in \mathbb{R}^{n\times n}$, show that $\|I\| = 1$.

   (c) For A invertible, show that $\kappa(A) \geq 1$, where $\kappa(A)$ is the condition number of that matrix A corresponding to the norm $\|\cdot\|$. Use the above two properties with $B := A^{-1}$ for your argument.

   (d) Argue that the Frobenius matrix norm $\|A\|_F := \left(\sum_{i,j=1}^n a_{ij}^2\right)^{1/2}$ cannot be induced by a suitable vector norm. *Hint*: Use one of the points above.

[1] There is no strict bound above which a system becomes poorly conditioned–but one usually considers conditions numbers large when they are $> 10^6, 10^7, \ldots$ or even larger.

[2] The commands needed in MATLAB are `abs` and `sum`, and in Python `np.abs` and `np.sum`. Most commands can not only applied to numbers, but also to vectors, where they apply to each component; this is typically much faster.

[3] In MATLAB use the *stop watch* commands `tic` and `toc`. In Python, import the `time` package and use the command `time.time()` to get the current time. Taking the difference of two times gives you the amount of time that has passed in seconds.

3. **Condition numbers and sharpness of estimate [2+4pt]** Let $A \in \mathbb{R}^{n\times n}$ be invertible.

   (a) Let $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum eigenvalues of $A^{\mathsf{T}}A$, respectively. Show that
   $$\kappa_2(A) = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}.$$

   (b) Let $\boldsymbol{b} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$, and $A\boldsymbol{x} = \boldsymbol{b}$, $A\boldsymbol{x}' = \boldsymbol{b}'$ and denote the perturbations by $\Delta\boldsymbol{b} = \boldsymbol{b}' - \boldsymbol{b}$ and $\Delta\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{x}$. Show that the inequality obtained in Theorem 2.11 is *sharp*. That is, find vectors $\boldsymbol{b}, \Delta\boldsymbol{b}$ for which
   $$\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \kappa_2(A)\frac{\|\Delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}\ .$$
   (Hint: consider the eigenvectors of $A^{\mathsf{T}}A$.)

4. **An analytical estimate [4pt]** Let $A \in \mathbb{R}^{n\times n}$ be defined by
   $$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$
   Compute the matrix $A^{\mathsf{T}}A$. Show that for any $\boldsymbol{x} \neq 0 \in \mathbb{R}^n$, $\boldsymbol{x}$ will be an eigenvector if $x_1 = 0$ and $x_2 + x_3 + \ldots + x_n = 0$. Show that there are two more eigenvectors with $x_2 = x_3 = \ldots = x_n$ and find the corresponding eigenvalues. Deduce that
   $$\kappa_2(A) = \frac{1}{2}(n+1)\left(1 + \sqrt{1 - \frac{4}{(n+1)^2}}\right)$$
   after application of the result from problem 3(a). Would you trust the result of a linear system solve for large n?

   (Hint: write the matrix A in block form with blocks of size 1×1, $1\times(n-1)$, $(n-1)\times 1$ and $(n-1)\times(n-1)$.)

5. **Approximate intersection of 3 lines [2pt]** Using the `qr` function in MATLAB, the `np.linalg.qr` (or however else you like), find the QR factorization of
   $$A = \begin{bmatrix} 9 & -6 \\ 12 & -8 \\ 0 & 20 \end{bmatrix}.$$
   Use it to find the least squares solution to the system of linear equations
   $$\begin{aligned} 9x - 6y &= 300 \\ 12x - 8y &= 600 \\ 20y &= 900\ . \end{aligned}$$
   Plot the three lines above and indicate the location of the least squares solution.

6. **Least squares data fitting [3pt]**. We believe that a real number Y is approximately determined by X with the model function

$$Y = a\exp(X) + bX^2 + cX + d\ .$$

We are given the following table of data for the values of X and Y:[4]

X	0.0	0.5	1.0	1.5	2.0	2.0	2.5
Y	0.0	0.20	0.27	0.30	0.32	0.35	0.27

Using the above data points, write down 7 equations in the four unknowns a, b, c, d. The least squares solution to this system is the best fit function. Write down the normal equations for this system, and solve them with MATLAB/Python/Julia. Plot the data points (X, Y) as points[5] and the best fit function.

[4] Note that you have two measurements at the same point $X = 2.0$. That is not uncommon in practice, and since measurements can contain noise it is possible that data at the same point are different.

[5] Do not connect the points; in MATLAB you can do that using `plot(X,Y,'ro')`. In Python, you can use `plt.scatter(X, Y)`.