Spring 2022: Numerical Analysis
Assignment 3 (due March 8, 2022 at 11:59pm ET)

Gradescope. When you upload your solution to Gradescope, please mark the problems to help us with grading.

Midterm preparation. Note that working on the homework, the worksheets and quizzed by yourself are a very good preparation for exams.

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

   (a) For the following matrix given by
   $$A = \begin{bmatrix} 1 & -2 \\ 3 & -2 \end{bmatrix},$$
   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 $n_1 = 100$, $n_{k+1} = 2n_k, k = 1, \dots, 7$, determine how long your code takes[3] to calculate $\|A\|_1$ and $\|A\|_\infty$ for a matrix $A \in \mathbb{R}^{n_k\times n_k}$ with random entries and report the results. Can you confirm the estimate from (b)? 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.

3. **Sharpness of condition number estimates [4pt]** Let $A \in \mathbb{R}^{n\times n}$ be invertible. Let $\boldsymbol{b} \in \mathbb{R}^n \backslash \{\boldsymbol{0}\}$, 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
$$\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \kappa_2(A)\frac{\|\Delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}.$$

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

[2] Most commands in MATLAB/Python/Julia can not only applied to numbers, but also to vectors, where they apply to each component.

[3] In MATLAB use the *stop watch* commands `tic` and `toc`. For Python, see e.g., here: https://realpython.com/python-timer/.

(Hint: consider the eigenvectors of A^TA.)

4. **Least squares [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.

5. **Floating point arithmetic, [2+2pts]** Consider the Harmonic sum

$$H(N) := \sum_{i=1}^{N} \frac{1}{i}, \tag{1}$$

which satisfies

$$H(N) = \Psi(N+1) - \Psi(1) \tag{2}$$

where Ψ is the digamma special function.[6]

(a) While software uses double precision by default, you can force it to use single precision using the `single` function. To compute $H(3)$, you could write (in MATLAB)

```
result = single(0.);
result = result + single(1.)/single(1.);
result = result + single(1.)/single(2.);
result = result + single(1.)/single(3.);
```

Implement a function that computes the sum $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$ for arbitrary N (in single precision). Now run this function for $N \in \{2^1, 2^2, \ldots, 2^{30}\}$. Plot, using a double-logarithmic plot, the relative error

$$\frac{|H(N) - H_{\text{ex}}(N)|}{|H_{\text{ex}}(N)|}$$

as a function of N, where $H_{\text{ex}}(N)$ is given by (2)

(b) For large N, we add a small number $(\frac{1}{N})$ to a large number $(H(N-1))$, which can lead to numerical error. Rearrange the sum to avoid this issue (i.e., start summing from the smallest element) and repeat the computation from the previous task. Plot the error again and compare to the error when using forward summation.

[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')`.

[6] You can access this function using `psi` in MATLAB or `digamma` in Python's scipy.