The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science
Andre Milzarek · Fall Semester 2024-25

DDA 3005 — Numerical Methods

Exercise Sheet Nr.: 2

Name: 杨景兰 Jinglan Yang   Student ID: 121090699

In the creation of this solution sheet, I worked together with:

Name: ____________ Student ID: ____________

Name: ____________ Student ID: ____________

Name: ____________ Student ID: ____________

For correction:

Exercise						$\sum$
Grading						

Problem 1 (Evaluating Functions and Condition Number): (approx. 25 pts)
In this exercise, we consider the mathematical problem of evaluating functions $f : \mathbb{R} \to \mathbb{R}$.

a) Show that the problem $f(x) = \log(1 + x)$ is well-conditioned for $x \geq -\frac{1}{2}$.

Hint: The estimate $\log(1 + x) \leq x$ (for all $x > -1$) might be useful.

$f(x) = \log(1+x)$, $f'(x) = \frac{1}{1+x}$

$\text{cond} \approx \left| \frac{x f'(x)}{f(x)} \right| = \left| \frac{x \cdot \frac{1}{1+x}}{\log(1+x)} \right| = \left| \frac{x}{(1+x)\log(1+x)} \right|$

$\text{cond}'(x) = \frac{(1+x)\log(1+x) - x[\log(1+x) + 1]}{[(1+x)\log(1+x)]^2} = \frac{\log(1+x) - x}{[(1+x)\log(1+x)]^2}$

$\because \log(1+x) \leq x$ for $\forall x > -1$ $\therefore \text{cond}'(x) < 0$

$\therefore \text{cond}(x) \leq \text{cond}(-\frac{1}{2}) = \log 2$

Hence, $f(x) = \log(1+x)$ is well-conditioned for $x \geq -\frac{1}{2}$

b) Compute the relative condition number $\text{cond}_f(x)$ of $f(x) := \frac{\sin(x)}{x}$. Is the evaluation of f at $x = 0$ a well-conditioned problem or not? Is the evaluation of f generally well-conditioned (for all x)? Provide detailed explanations!

$f(x) = \frac{\sin(x)}{x}$, $f'(x) = \frac{x\cos(x) - \sin(x)}{x^2}$ for $x \neq 0$

① $x \neq 0$

$\text{cond}_f(x) = \left| \frac{x f'(x)}{f(x)} \right| = \left| \frac{x\cos x - \sin x}{\sin x} \right| = \left| \frac{x}{\tan x} - 1 \right| \gg 1$

② $x = 0$

$\text{cond}_f(x) = \lim_{\delta \to 0} \sup_{|\Delta x| \leq \delta} \frac{|f(x+\Delta x) - f(x)|}{|f(x)|} \Big/ \frac{|\Delta x|}{|x|}$

Taylor expansion : $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$

$f(x) = \frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \cdots$

$f'(x) = -\frac{2x}{3!} + \frac{4x^3}{5!} - \frac{6x^5}{7!} + \cdots$, $f'(0) = 0$

$\Rightarrow \text{cond}_f(x) = \left| \frac{x f'(x)}{f(x)} \right| = \left| \frac{x \cdot (-\frac{2x}{3!} + \frac{4x^3}{5!} - \frac{6x^5}{7!} + \cdots)}{1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \cdots} \right|$

$\Rightarrow \text{cond}_f(0) = 0$

Hence, $f(x)$ is well-conditioned at $x = 0$, but NOT well-conditioned for all x.

Problem 2 (Clustering and Flops): (approx. 25 pts)
Let $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m \in \mathbb{R}^n$ be a given point cloud of m different vectors in $\mathbb{R}^n$. In this exercise, we want to study an algorithm that allows to cluster the points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m$ into different groups or clusters. Here, clustering refers to assigning a given vector $\boldsymbol{x}^i$ to a specific group or cluster that $\boldsymbol{x}^i$ belongs to. We label the groups $1, \ldots, k$ and specify a clustering or assignment of the m vectors to groups using an m-dimensional vector $\boldsymbol{c}$, where c_i is the group number that $\boldsymbol{x}^i$ is assigned to. With each of the groups $1, \ldots, k$, we associate a group representative $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k \in \mathbb{R}^n$. Naturally, the (not necessarily known) group representative should be close to the vectors in its associated group, i.e., the terms $\|\boldsymbol{x}^i - \boldsymbol{z}^{c_i}\|$ should be small. Our goal is to find a choice of group assignments $c_1, \ldots, c_m$ and a choice of representatives $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k$ that minimize the clustering objective

$$f(\boldsymbol{c}, \boldsymbol{z}^1, \ldots, \boldsymbol{z}^k) := \frac{1}{m}\left(\|\boldsymbol{x}^1 - \boldsymbol{z}^{c_1}\|^2 + \cdots + \|\boldsymbol{x}^m - \boldsymbol{z}^{c_m}\|^2\right).$$

We want to use a heuristic algorithm to estimate $\boldsymbol{c}$ and the group representatives $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k$. The full procedure is shown below.

Given: $k \in \mathbb{N}$; a list of m vectors $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m$; and an initial list of k group representative vectors $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^k$. Repeat until STOP:

1. Partition the vectors into k groups. For each vector $i = 1, \ldots, m$ assign $\boldsymbol{x}^i$ to the group associated with the nearest representative $\boldsymbol{z}^j$, i.e., find j such that $\|\boldsymbol{x}^i - \boldsymbol{z}^j\|$ is minimal and set $c_i = j$.
2. Update representatives. For each group $j = 1, \ldots, k$, set $\boldsymbol{z}^j$ to be the mean of the vectors in the group j.

a) Estimate the total number of flops this algorithm requires per iteration. Express your final result in terms of asymptotic big-$\mathcal{O}$ notation.

Hint: You can assume that the Euclidean norm is used when calculating distances. The square root operation $\sqrt{\cdot}$ can be realized using 1 flop per application. Furthermore, finding the minimum element $\min\{a_1, \ldots, a_n\}$ of an array of n numbers costs $\mathcal{O}(n)$ flops (the total costs are bounded by $2n - 2$ flops).

(1) partitioning :

$\|x^i - z^j\|$: $O(n)$

$\forall x_i, z_j$, compute $\|x^i - z^j\|$: $O(mkn)$

$\min\{\|x^i - z^j\|, \cdots\}$: $O(mk)$

$\Rightarrow O(mkn + mk) = O(mkn)$

(2) Updating : $O(mk)$

$O(mkn + mk) = O(mkn)$

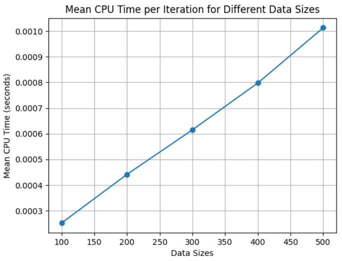
Hence, total number of flops per iteration: $O(mkn)$

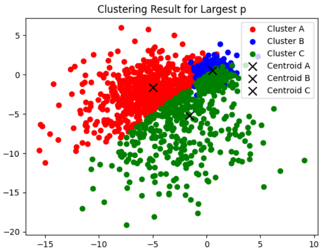
b) The codes `clustering.m` and `clustering.py` implement the clustering algorithm in `MATLAB` and `Python`. Download the code from BB and write a test program that allows to verify your result from part b) experimentally. Specifically, generate three random data clouds $\boldsymbol{A} = [\boldsymbol{a}^1, \ldots, \boldsymbol{a}^p]$, $\boldsymbol{B} = [\boldsymbol{b}^1, \ldots, \boldsymbol{b}^p]$, and $\boldsymbol{C} = [\boldsymbol{c}^1, \ldots, \boldsymbol{c}^p]$ where

$$\boldsymbol{a}^i = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \quad \boldsymbol{b}^i = 4\begin{bmatrix} r_3 \\ r_4 \end{bmatrix} - \begin{bmatrix} 4 \\ 6 \end{bmatrix}, \quad \boldsymbol{c}^i = 2\begin{bmatrix} r_5 \\ r_6 \end{bmatrix} - \begin{bmatrix} 5 \\ 2 \end{bmatrix}, \quad r_j \sim \mathcal{N}(0,1)$$

for all $j = 1, \ldots, 6$ and $i = 1, \ldots, p$ and set $\boldsymbol{X} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m] = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ as test set. Run the code for different data sizes $p = 100 \cdot \texttt{1:10}$ (or $p = 100 \cdot \texttt{1:5}$) and report the mean cpu-time per iteration for each run (e.g., using a plot). Do the observed results match your expectation? Visualize the obtained clustering results for the largest choice of p.

Hint: You can set the number of groups to 3 and initialize $\boldsymbol{z}^1, \boldsymbol{z}^2, \boldsymbol{z}^3$ randomly. The number of iterations per run can be controlled via the `options`; you can use 100 or 1000.





Match my expectation.

Problem 3 (Big-$\mathcal{O}$ Calculus): (approx. *10* pts)
In this exercise, we want to verify computational rules involving the asymptotic big-$\mathcal{O}$ notation.

- Show that $(1 + \mathcal{O}(x))(1 + \mathcal{O}(x)) = 1 + \mathcal{O}(x)$ for $x \to 0$.

The precise meaning of this statement is that if f is a function satisfying $f(x) = (1+\mathcal{O}(x))(1+\mathcal{O}(x))$ as $x \to 0$, then f also satisfies $f(x) = 1 + \mathcal{O}(x)$ as $x \to 0$.

$f(x) = O(x)$ for $x \to 0 \Leftrightarrow \exists\, \varepsilon > 0,\ C \geq 0$, s.t. $|f(x)| \leq C \cdot |x|$ for $\forall\, x$ satisfying $|x| < \varepsilon$.

$$(1+O(x))(1+O(x)) = 1 + 2O(x) + O^2(x)$$
$$\leq 1 + 2C\cdot|x| + (C\cdot|x|)^2$$
$$= 1 + 2C\cdot|x| + C^2\cdot|x^2|$$
$$\leq 1 + C\cdot|x| \quad \text{for } x \to 0$$

$\therefore (1+O(x))(1+O(x)) = 1 + O(x)$ for $x \to 0$

Problem 4 (Error Analysis): (approx. *30* pts)
In this exercise, we consider the problem of computing the function value:

$$F(\boldsymbol{z}) = \begin{bmatrix} x^2 - y^2 \\ 2xy \end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2, \quad \boldsymbol{z} \neq \boldsymbol{0}. \tag{1}$$

The following algorithm is used to evaluate F:

$$\hat{F}(\boldsymbol{z}) = \begin{bmatrix} (\mathrm{fl}(x) \odot \mathrm{fl}(x)) \ominus (\mathrm{fl}(y) \odot \mathrm{fl}(y)) \\ 2 \odot (\mathrm{fl}(x) \odot \mathrm{fl}(y)) \end{bmatrix}.$$

Here, we use a base 2 floating-point system with machine precision $\varepsilon_{\mathrm{mach}}$ which satisfies:

- $|\mathrm{fl}(x) - x| \leq \varepsilon_{\mathrm{mach}}|x|$ for all $x \in \mathbb{R}$.
- The `IEEE`-Standard 754 holds, i.e., for any machine numbers x, y, we have $x \circledast y = \mathrm{fl}(x * y)$, where $*$ can represent the operations $\{+, -, \cdot, /\}$.

The goal of this problem is to investigate accuracy of the algorithm $\hat{F}$.

a) Show that the algorithm $\hat{F}$ is accurate for problem (1). You can use the following steps:

   i) First express $\hat{F}(\boldsymbol{z})$ in terms of the true inputs $x, y \in \mathbb{R}$ and $\varepsilon_{\mathrm{mach}}$ (using the asymptotic Landau big-$\mathcal{O}$ notation).

   Hint: The following fact can be useful. Let ε_i, $i = 1, \ldots, m$ be a collection of numbers satisfying $|\varepsilon_i| \leq \varepsilon_{\mathrm{mach}}$. Then, it holds that $\prod_{i=1}^m (1 + \varepsilon_i) = 1 + \mathcal{O}(\varepsilon_{\mathrm{mach}})$.

   ii) Estimate the relative forward error $\|\hat{F}(\boldsymbol{z}) - F(\boldsymbol{z})\| / \|F(\boldsymbol{z})\|$.

$$\hat{F}(z) = \begin{bmatrix} (fl(x) \odot fl(x)) \ominus (fl(y) \odot fl(y)) \\ 2 \odot (fl(x) \odot fl(y)) \end{bmatrix}$$

$$= \begin{bmatrix} (1+\varepsilon_3)\left[[(1+\varepsilon_1)x]^2 - [(1+\varepsilon_2)y]^2\right] \\ 2(1+\varepsilon_4)(1+\varepsilon_1)x \cdot (1+\varepsilon_2)y \end{bmatrix}, \quad |\varepsilon_1|, |\varepsilon_2|, |\varepsilon_3|, |\varepsilon_4| \leq \varepsilon_{mach}$$

$$= \begin{bmatrix} (1+\varepsilon_3)(1+2\varepsilon_1+\varepsilon_1^2)x^2 - (1+\varepsilon_3)(1+2\varepsilon_2+\varepsilon_2^2)y^2 \\ 2(1+\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2+\varepsilon_4+\varepsilon_1\varepsilon_4+\varepsilon_2\varepsilon_4+\varepsilon_1\varepsilon_2\varepsilon_4)xy \end{bmatrix},$$

$$= \begin{bmatrix} [1+O(\varepsilon_{mach})](x^2-y^2) \\ 2[1+O(\varepsilon_{mach})]xy \end{bmatrix}$$

Forward error:

$$\frac{\|\hat{F}(z) - F(z)\|}{\|F(z)\|} = \frac{\left\| \begin{matrix} [1+O(\varepsilon_{mach})](x^2-y^2) - [x^2-y^2] \\ 2[1+O(\varepsilon_{mach})]xy - 2xy \end{matrix} \right\|}{\left\| \begin{matrix} x^2-y^2 \\ 2xy \end{matrix} \right\|}$$

$$= \frac{\left\| \begin{matrix} [O(\varepsilon_{mach})](x^2-y^2) \\ 2[O(\varepsilon_{mach})]xy \end{matrix} \right\|}{\left\| \begin{matrix} x^2-y^2 \\ 2xy \end{matrix} \right\|}$$

$$= O(\varepsilon_{mach}) \frac{\left\| \cancel{\begin{matrix} x^2-y^2 \\ 2xy \end{matrix}} \right\|}{\left\| \cancel{\begin{matrix} x^2-y^2 \\ 2xy \end{matrix}} \right\|}$$

$$= O(\varepsilon_{mach})$$

Hence, $\hat{F}(z)$ is accurate for problem (1).

b) Is $\hat{F}$ also a stable algorithm for evaluating F? Explain your answer briefly!

$$\hat{F}(z) = \begin{bmatrix} [1+O(\varepsilon_{mach})](x^2-y^2) \\ 2[1+O(\varepsilon_{mach})]xy \end{bmatrix} = \begin{bmatrix} [1+O(\varepsilon_{mach})]x^2 - [1+O(\varepsilon_{mach})]y^2 \\ 2[1+O(\varepsilon_{mach})][1+O(\varepsilon_{mach})]xy \end{bmatrix}$$

plus, $[1+O(\varepsilon_{mach})][1+O(\varepsilon_{mach})] = 1+O(\varepsilon_{mach})$.

Choose $\hat{x} = \sqrt{1+O(\varepsilon_{mach})}\, x$, $\hat{y} = \sqrt{1+O(\varepsilon_{mach})}\, y$.

This implies: $\dfrac{\|\hat{F}(z) - F(\hat{z})\|}{\|F(z)\|} = 0$

we only need to show $\dfrac{\|z-\hat{z}\|}{\|z\|} = O(\varepsilon_{mach})$

$$\frac{\|z-\hat{z}\|}{\|z\|} = \frac{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \sqrt{1+O(\varepsilon_{mach})}\, x \\ \sqrt{1+O(\varepsilon_{mach})}\, y \end{pmatrix} \right\|}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} \right\|} = \sqrt{2+O(\varepsilon_{mach}) - 2\sqrt{1+O(\varepsilon_{mach})}} \; \frac{\cancel{\sqrt{x^2+y^2}}}{\cancel{\sqrt{x^2+y^2}}} = O(\varepsilon_{mach})$$

Hence, $\hat{F}$ is a stable algorithm for evaluating F.

Problem 5 (Nonexistent LU Factorization): (approx. *10* pts)

Prove that the matrix

$$A = \begin{array}{c c} & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \end{array} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

is nonsingular and that it does not possess a (naïve) LU factorization (without pivoting).

① $\det(A) = 0 \times (2\times1 - 1\times1) - 1 \times (1\times1 - 0\times1) + 0 \times (1\times1 - 0\times2) = -1 \neq 0$

Hence, A is nonsingular

② We cannot perform Gaussian elimination on A since $A[0,0] = 0$ (first pivot is 0).

Plus, since $\det(A) = -1 < 0$, A is not positive definite $\Rightarrow$ can not perform Cholesky factorization.

Hence, A does not possess a naïve LU factorization without pivoting.