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

DDA 3005 — Numerical Methods

Solutions 2

Problem 1 (Evaluating Functions and Condition Number): (approx. *25* points)
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.

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!

Solution :

a) The condition number is given by

$$\text{cond}(x) = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x}{(1+x)\log(1+x)}\right| =: g(x)$$

The function g can be shown to be differentiable (for $x > -1$) and we have

$$g(x) = \begin{cases} \frac{x}{(1+x)\log(1+x)} & \text{if } x \neq 0, \\ 0 & \text{if } x = 0, \end{cases} \quad \text{and} \quad g'(x) = \begin{cases} \frac{\log(1+x)-x}{(1+x)^2\log(1+x)^2} & \text{if } x \neq 0, \\ -\frac{1}{2} & \text{if } x = 0. \end{cases}$$

(The limits for $x = 0$ can be verified by, e.g., applying L'Hôpital). Using the hint, this implies $g'(x) \leq 0$ and hence, g is monotonically decreasing. This yields $\text{cond}(x) \leq g(-\frac{1}{2}) \approx 1.44 \leq 2$. Thus, evaluating the function f is well-conditioned.

b) The condition number for $f(x) = \frac{\sin(x)}{x}$ is given by

$$\text{cond}(x) = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x^2 \cdot \frac{\cos(x)x-\sin(x)}{x^2}}{\sin(x)}\right| = \left|\frac{\cos(x)x}{\sin(x)} - 1\right| =: g(x)$$

Using L'Hôpital, we have $\lim_{x\to 0} \frac{\cos(x)x}{\sin(x)} = \lim_{x\to 0} \frac{-\sin(x)x+\cos(x)}{\cos(x)} = 1$ and hence, it follows $g(0) = 0$. Consequently, evaluating f at $x = 0$ is "perfectly conditioned". The evaluation of f is generally not well-conditioned for all x.

For $x \to k\pi$, $k \in \mathbb{Z}\backslash\{0\}$, we obtain $g(x) \to \infty$. Thus, evaluating f is not well-conditioned for such x.

Problem 2 (Clustering and Flops): (approx. *25* points)

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).

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 $p = 100 \cdot$ `1:10` (or $p = 100 \cdot$ `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.

Solution :

a) For each $i = 1, \ldots, m$, we first compute the distances:

$$\|\boldsymbol{x}^i - \boldsymbol{z}^1\|, \quad \ldots, \quad \|\boldsymbol{x}^i - \boldsymbol{z}^k\|.$$

Calculating $\|\boldsymbol{x}^i - \boldsymbol{z}^j\|$ costs $n + (2n-1) + 1 = 3n$ flops (n flops for computing $\boldsymbol{\zeta}^j = \boldsymbol{x}^i - \boldsymbol{z}^j$; $2n-1$ flops for the inner product $(\boldsymbol{\zeta}^j)^\top \boldsymbol{\zeta}^j$; 1 flop for taking the square root. Consequently, this first step requires $3nk$ flops in total. As mentioned in the hint, the operation $c_i = \arg\min_{j=1,\ldots,k} \|\boldsymbol{x}^i - \boldsymbol{z}^j\|$ costs additional $2k-2$ flops. Hence, in order to determine the full group assignment $\boldsymbol{c}$, we require $(3nk + 2k - 2) \cdot m$ flops.

For all $j = 1, \ldots, k$, we now set $\mathcal{I}_j := \{i : c_i = j\}$. The new group representative $\boldsymbol{z}^j$ is then calculated via $\boldsymbol{z}^j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \boldsymbol{x}^{c_i}$. This operation costs $(|\mathcal{I}_j| - 1) \cdot n + 1$ flops (for the summations and final multiplication). Hence, in total, the second step of the algorithm requires $\sum_{j=1}^{k} (|\mathcal{I}_j| - 1) \cdot n + 1 = (m-k)n + k$ flops.

In total this yields $3kmn + 2km + mn - kn + k - 2m = \mathcal{O}(kmn)$ flops.

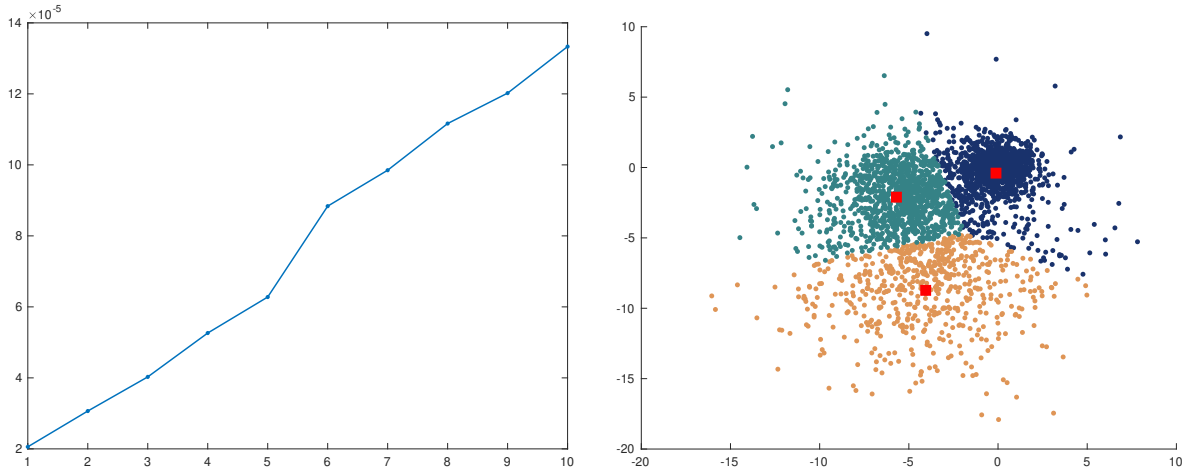
b) The `MATLAB` code is shown below (to save space, the `Python` code is provided in the corresponding zip-file on BB).

```
% demo_clustering
rng(20240929)
tests   = 10;
t       = zeros(tests,1);

for fac = 1:tests
    A       = norminv(rand(fac*100,2), 0, 1)';
    B       = 4*norminv(rand(fac*100,2), 0, 1)'-[4;6];
    C       = 2*norminv(rand(fac*100,2), 0, 1)'-[5;2];

    X       = [A,B,C];
    Z0      = [A(:,1),B(:,1),C(:,1)];

    opts.iter = 1000;

    tic
    [c,Z]   = clustering(X,Z0,opts);
    t(fac)  = toc/opts.iter;
end

figure
plot(1:tests,t,'.-','MarkerSize',10,'LineWidth',1.2);
set(gcf,'Renderer', 'painters');
saveas(gcf,strcat('exercise_24_01'),'epsc');

figure
hold on
plot(X(1,c==1),X(2,c==1),'.','Color',[25,50,108]/255,'MarkerSize',12);
plot(X(1,c==2),X(2,c==2),'.','Color',[223,149,86]/255,'MarkerSize',12);
plot(X(1,c==3),X(2,c==3),'.','Color',[53,130,134]/255,'MarkerSize',12);

plot(Z(1,:),Z(2,:),'s','MarkerSize',10,'MarkerFaceColor',[1,0,0],'MarkerEdgeColor'
    ,[1,0,0]);
hold off
```

```
34 set(gcf,'Renderer', 'painters');
35 saveas(gcf,strcat('exercise_24_02'),'epsc');
```

The runtime and clustering result are shown below in Figure 1. Based on the complexity result derived in part a), we expect the runtime to grow linearly with the number of datapoints m (or with p). This is also reflected in Figure 1 (a).

Remark: The algorithm described and analyzed in Problem 2 is known as **k-means**. k-means is of fundamental importance in machine learning and clustering.



(a) Mean runtime for $p \in \{100, \ldots, 1000\}$ (b) Final clustering results for $p = 1000$

Figure 1: Results and visualization for Problem 2.

Problem 3 (Big-$\mathcal{O}$ Calculus): (approx. *10* points)
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 $x \to 0$.

Solution : Per definition of the asymptotic big-$\mathcal{O}$ notation, there exist functions $f(x) = \mathcal{O}(x)$ and $g(x) = \mathcal{O}(x)$ and constants C_f, C_g, and ε such that $|f(x)| \leq C_f \cdot |x|$ and $|g(x)| \leq C_g \cdot |x|$ for all x with $|x| \leq \varepsilon$. We then have

$$(1 + f(x)) \cdot (1 + g(x)) = 1 + \underbrace{f(x) + g(x) + f(x) \cdot g(x)}_{=:h(x)}.$$

To establish the desired result, we need to show $h(x) = \mathcal{O}(x)$ for $x \to 0$. For all x with $|x| \leq \varepsilon$, we have

$$\begin{aligned} h(x) = f(x) + g(x) + f(x) \cdot g(x) &\leq |f(x)| + |g(x)| + |f(x)| \cdot |g(x)| \\ &\leq C_f|x| + C_g|x| + C_fC_g|x|^2 \leq (C_f + C_g + C_fC_g\varepsilon) \cdot |x|. \end{aligned}$$

Hence, it follows $h(x) = \mathcal{O}(x)$ as $x \to 0$ which verifies $(1 + \mathcal{O}(x))(1 + \mathcal{O}(x)) = 1 + \mathcal{O}(x)$ for $x \to 0$.

Problem 4 (Error Analysis): (approx. *30* points)

In this exercise, we consider the problem of computing the function value:

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

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 $\|\hat{F}(\boldsymbol{z}) - F(\boldsymbol{z})\|/\|F(\boldsymbol{z})\|$.

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

Solution :

a) We first start with part i). According to the assumption, there are $|\varepsilon_i| \leq \varepsilon_{\mathrm{mach}}$, $i = 1, 2, \ldots, 7$, such that $\mathrm{fl}(x) = (1 + \varepsilon_1)x$, $\mathrm{fl}(y) = (1 + \varepsilon_3)y$,

$$\begin{aligned} &(\mathrm{fl}(x) \odot \mathrm{fl}(x)) \ominus (\mathrm{fl}(y) \odot \mathrm{fl}(y)) \\ &\qquad = \{[x(1 + \varepsilon_1)]^2(1 + \varepsilon_2) - [y(1 + \varepsilon_3)]^2(1 + \varepsilon_4)\}(1 + \varepsilon_5) \\ &\qquad = x^2(1 + \varepsilon_1)^2(1 + \varepsilon_2)(1 + \varepsilon_5) - y^2(1 + \varepsilon_3)^2(1 + \varepsilon_4)(1 + \varepsilon_5) \end{aligned}$$

and

$$2 \odot (\mathrm{fl}(x) \odot \mathrm{fl}(y)) = 2xy(1 + \varepsilon_1)(1 + \epsilon_3)(1 + \varepsilon_6)(1 + \varepsilon_7).$$

Thus, using the hint, we have

$$\hat{F}(\boldsymbol{z}) = \begin{pmatrix} x^2(1 + \mathcal{O}(\varepsilon_{\mathrm{mach}})) - y^2(1 + \mathcal{O}(\varepsilon_{\mathrm{mach}})) \\ 2xy(1 + \mathcal{O}(\varepsilon_{\mathrm{mach}})) \end{pmatrix}$$

We now show that this algorithm is accurate (part ii)):

$$\begin{aligned} \frac{\|\hat{F}(\boldsymbol{z}) - F(\boldsymbol{z})\|^2}{\|F(\boldsymbol{z})\|^2} &= \frac{(x^2 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}) - y^2 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}))^2 + 4x^2y^2 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}})^2}{(x^2 - y^2)^2 + 4x^2y^2} \\ &\leq \frac{x^4 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}^2) - 2x^2y^2 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}})^2 + y^4 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}^2)}{(x^2 + y^2)^2} + \mathcal{O}(\varepsilon_{\mathrm{mach}}^2) \\ &\leq \mathcal{O}(\varepsilon_{\mathrm{mach}}^2), \end{aligned}$$

where we used $(x^2 - y^2)^2 + 4x^2y^2 \geq 4x^2y^2$ and $x^4 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}^2) - 2x^2y^2 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}})^2 + y^4 \cdot \mathcal{O}(\varepsilon_{\mathrm{mach}}^2) \leq (x^2 + y^2)^2 \cdot O(\varepsilon_{\mathrm{mach}}^2)$. Taking the square root, this implies that $\hat{F}$ is accurate.

b) Notice that accuracy implies stability (we can simply choose $\hat{\boldsymbol{z}} = \boldsymbol{z}$). Hence, $\hat{F}$ is also a stable algorithm!

Problem 5 (Nonexistent LU Factorization): (approx. *10* points)
Prove that the matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

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

Solution : Using the cofactor expansion of the determinant, we can calculate

$$\det(\boldsymbol{A}) = 0 \cdot \det\left(\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}\right) - 1 \cdot \det\left(\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\right) + 0 \cdot \det\left(\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}\right) = -1 \neq 0.$$

Hence, $\boldsymbol{A}$ is nonsingular. To show that $\boldsymbol{A}$ has no (naïve) LU factorization, we can use the ansatz

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U} = \begin{bmatrix} 1 & & \\ \ell_{21} & 1 & \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ & u_{22} & u_{23} \\ & & u_{33} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ \ell_{21}u_{11} & \ell_{21}u_{12} + u_{22} & \ell_{21}u_{13} + u_{23} \\ * & * & * \end{bmatrix}.$$

This readily implies $u_{11} = 0$, $u_{12} = 1$, and $u_{13} = 0$ which leads to the contradiction $0 = \ell_{21}u_{11} = 1$. Consequently, $\boldsymbol{A}$ cannot have a classical LU factorization (without pivoting).