Numerical analysis: Homework 1

Instructor: Anil Damle
Due: February 5, 2025

Policies

You may discuss the homework problems freely with other students, but please refrain from looking at their code or writeups (or sharing your own). Ultimately, you must implement your own code and write up your own solution to be turned in. Your solution, including plots and requested output from your code should be typeset and submitted via the Gradescope as a pdf file. This file must be self contained for grading. Additionally, please submit any code written for the assignment as zip file to the separate Gradescope assignment for code.

Question 1:

Suppose $A, B \in \mathbb{R}^{n\times n}$ are general square matrices, $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix, and $u, v \in \mathbb{R}^n$ are vectors of length n. For the following mathematical expressions: determine an optimal way to compute them (in terms of complexity in n, i.e., the number of required basic arithmetic operations for a given n). Give the complexity and explain your reasoning (you need not prove "lower bounds"), implement your procedure, demonstrate that your code scales correctly by timing your code as n is increased and providing corroborating evidence (a sufficiently clear plot will suffice). You may reorder and modify the expressions in any way you choose, so long as the result remains mathematically equivalent to the given statement.

(a) $\mathrm{Tr}(D + uv^T)$

(b) $uv^T A$

(c) $(I + uu^T)v$

(d) $v^T ABu$

In the preceding parts we considered "reordering" computations to achieve an asymptotically different runtime. Now we will consider something more practical, how various optimizations in numerical codes (and their interface with hardware) impact the practical runtime of a common operation: matrix multiplication. Write code that implements computing the product of two matrices in the following ways:

(e) $C(i, j) = \sum_k A(i, k)B(k, j)$; for this part you can only use built in scalar multiplication

(f) $C(:, i) = A(B(:, i))$; you may now leverage your chosen languages calls to compute matrix-vector products.

(g) As a point of comparison we will also use the "built in" routine for computing matrix-matrix multiplication (e.g., simply writing $C = A^*B$ in Matlab), this is our way of accessing the routine for matrix-matrix multiplication from BLAS (`http://www.netlib.org/blas/`).

For all the above algorithms clearly illustrate that your implementation is $\mathcal{O}\left(n^3\right)$, compare and contrast their performance (again, a clear plot will help here). Argue about why you believe you might be seeing such differences.

Remarks: Up to constants, we expect the arithmetic complexity of the above computations, and hence the time taken for sufficiently large n, to behave like n^q for some non-negative q. When you are construing your plots think carefully about how you can generate a plot where the slope of the corresponding line can be used to determine/estimate q. Here is a quick hint: simply plotting $t(n)$, the time taken, vs n does not have this property and is not an easy way to distinguish between various complexities. Is something like $\log(t(n))$ vs $\log(n)$ a better choice? How would different exponents manifest in this case? Lastly, for a given n run your timing experiment multiple times—how consistent are the measurements? What are the potential implications of this and are there ways to mitigate them?

Question 2:

For this problem, let $A \in \mathbb{R}^{n\times n}$ be a square matrix and $x \in \mathbb{R}^n$ be a vector of length n. Prove the following:

1. $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$
2. $\|A\|_2 \leq \sqrt{n}\|A\|_\infty$
3. For any orthogonal matrix $Q \in \mathbb{R}^{n\times n}$: $\|Qx\|_2 = \|x\|_2$.
4. $\|A\|_2 = \sigma_{\max}$, where $\sigma_{\max}$ is the largest singular value of A.
5. For any orthogonal matrix $Q \in \mathbb{R}^{n\times n}$: $\|QA\|_2 = \|A\|_2$.

Question 3:

For this problem, let $V \in \mathbb{R}^{m\times n}$ with $m > n$ be a matrix with linearly independent columns, prove that

1. V^TV is positive definite
2. VV^T is positive semi-definite but not positive definite

Question 4:

For differentiable functions $f(x)$, where the input x may be a vector $x = (x_1, x_2, \ldots, x_n)$, we define the relative condition number $\kappa_2(x)$ of computing $f(x)$ at x as

$$\kappa_2(x) \equiv \frac{\|J(x)\|_2}{\|f(x)\|_2/\|x\|_2},$$

where J is the Jacobian of f.

1. Compute $\kappa_2(x)$ for subtraction, *i.e.* $f(x) = x_1 - x_2$. When, if ever, is this an ill-conditioned problem?
2. Compute $\kappa_2(x)$ for multiplication, *i.e.* $f(x) = x_1x_2$. When, if ever, is this an ill-conditioned problem?