Division of Applied Mathematics

Brown University

Subject : APMA 2070-Deep Learning for Scientists and Engineers Year : 2025–2026

Instructors : G. E. Karniadakis and K. Shukla

Homework : 2 Due Date : 02/28/2026

Instructions

1. This homework emphasizes *numerical understanding*, not just correct code. You may use AI tools for syntax or debugging, but all explanations, diagnostics, and interpretations must be your own. You are responsible for defending every result you submit.
2. Unless stated otherwise, submit : (i) well-documented Python code, (ii) a short write-up (PDF) with the requested numerical results, and (iii) brief explanations where prompted.

Task 1 : Modified Gram–Schmidt (MGS)

a. Implement the modified Gram–Schmidt algorithm for orthonormalizing N vectors. Your function should return Q and R (thin QR), and should gracefully handle (near) linear dependence.

b. Apply your implementation to the set

$$v_1 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \qquad v_2 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}, \qquad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Let $V = [v_1 \ v_2 \ v_3]$ and compute $V \approx QR$.

c. Report the following diagnostics :

— orthogonality error : $\|Q^T Q - I\|_2$,

— reconstruction error : $\|V - QR\|_2$.

d. Perturb v_3 by $v_3^{(\varepsilon)} = v_3 + \varepsilon v_1$ with $\varepsilon = 10^{-6}$ and rerun MGS. Report the new orthogonality error and explain *why* it changes (connect your explanation to conditioning / near dependence).

e. Implement classical Gram–Schmidt (CGS) and compare CGS vs. MGS on the original and perturbed inputs. Summarize the comparison using the same two diagnostics above and a short interpretation.

Task 2 : Singular Value Decomposition (SVD) without `numpy.linalg.svd`

a) Implement the (thin) SVD of a real matrix using only NumPy. You may use eigendecompositions (e.g. `numpy.linalg.eigh`/`numpy.linalg.eig`) but *must not* call built-in SVD routines such as `numpy.linalg.svd`.

b) Test your code on the matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 5}.$$

c) Report the singular values $\{\sigma_i\}$ in descending order and determine the *numerical rank* of M (state the tolerance you use and justify it briefly).

d) Identify which rows and/or columns are responsible for rank deficiency, and explain your reasoning (do not just state "there is a zero row").

e) Perturb one *zero* entry of M by 10^{-8} (you choose which entry) to form $\widetilde{M}$. Report how the smallest singular values change and comment on the stability (or instability) of the corresponding singular vectors.

Task 3 : L^2 Norm and Floating-Point Stability

a) Write a subroutine to compute the L^2 norm of a vector of length N :

$$\|x\|_2 = \left(\sum_{i=1}^{N} x_i^2 \right)^{1/2}.$$

b) Test it on the vector

$$x = \begin{pmatrix} \cos(\pi/4)\ \sin(\pi/8) \\ \sin(\pi/4)\ \sin(\pi/8) \\ \sin(\pi/4)\ \sin(\pi/8) \\ \cos(\pi/8) \end{pmatrix}.$$

Report your computed value and compare it to the expected value (briefly explain *why* that expected value holds).

c) Implement a rescaled (overflow/underflow-safe) algorithm for $\|x\|_2$ (e.g. "scaled sum of squares") and compare it to the naive implementation on at least one extreme test vector.

d) Construct an example where naive summation is inaccurate (overflow, underflow, or catastrophic cancellation). Explain the failure mechanism and show that your stable method fixes it.

Reflection (Mandatory)

— Identify one numerical result in this homework that surprised you. Explain what mathematical or floating-point mechanism caused it.