Middle East Technical University - Department of Computer Engineering

CENG 371

Scientific Computing

Fall' 2024-2025

Homework 3

Due Date: 13 December 2024, Thursday, 23:55
Late Submission Policy will be explained below

Question 1 (45 points)

1. **(10 pts)** Implement the power method.
(Signature: `[eVal, eVec] = power_method(A, V)`; where `A` is the matrix and `V` is an optional starting vector.)
2. **(10 pts)** Implement the shifted inverse power method.
(Signature: `[[eVal, eVec] = inverse_power(A, alpha)`, where `A` is the matrix `alpha` is the shift value. `eVal, eVec` is the eigenvalue/vector that is closest to `alpha`)
3. **(10 pts)** Find the largest and smallest (in magnitude) eigenvalues and the corresponding eigenvectors of matrix `A` where;

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

4. **(15 pts)** Find the largest eigenvalue eigenvector pair **by hand** of matrix `B`. where;

$$A = \begin{bmatrix} 0.2 & 0.3 & -0.5 \\ 0.6 & -0.8 & 0.2 \\ -1.0 & 0.1 & 0.9 \end{bmatrix}$$

(You can use the identity $Av = \lambda v$). Do the same using the power method. Use starting vector v where $v = [1, 1, 1]^T$. Reflect on your findings.

Question 2 (55 points)

One can use the power method to find the largest k eigenvalues of a matrix A iteratively by subtracting the value $\lambda_i \frac{v_i v_i^T}{v_i^T v_i}$ from the current matrix at each step.

1. **(10 pts)** Show how the above idea works.
2. **(15 pts)** Implement the above idea. (Signature: `[eVals, eVecs] = power_k(A, k)` where `k` indicates the largest k eigenvalue, `eVals` is the list of k-largest eigenvalues and `eVecs` shows the corresponding eigenvectors (a.k.a $k \times n$ matrix, where n is the height of matrix A).

3. **(15 pts)** Implement subspace iteration method. (Signature: `[eVals, eVecs] = power_k( A, k)`, same argument definitions as stated above.)

    > You can use the built-in QR Factorization routine of Matlab (link).

4. **(15 pts)** Compare the performances of these two methods on the matrix can_{229} of University of Florida Sparse Matrix Collection.

Regulations and Submission

- **Programming Language:** You can use any programming language, **however Matlab is recommended**. Other good choices are Python (via Numpy package), and Octave (open source alternative to Matlab). Students can download Matlab (please refer to this link).
- Most of the points will be granted to the **explanation/discussion parts** of the questions. Make sure you **reflect your reasoning** cleanly and concisely.
- Most of your points will come from the PDF text, however; you should submit your code as well.
- Please make sure that your reports are readable, clean, and concise. **Note that the organization of your PDF will also be subject to grading**. You can get bonus/penalty points based on it.
- Uploaded codes should be clean and understandable similar to the PDFs. The codes will not be graded rigorously (such as black-box testing) since there aren't standard language or script arguments. However, these will be visually inspected.
- **Late Submission Policy:** Accepted with a deduction of $5 \times d^2$; where d is the number of late days submitted.
- Submission will be done via Odtuclass, (odtuclass.metu.edu.tr).
- Please upload both your code and your findings (as a PDF) to the system in a zip file.