HOMEWORK SET #9

EE 510: Linear Algebra for Engineering

Assigned: 26 October 2024
Due: 3 November 2024

Directions: Please show all work and box answers when appropriate.

1. Introduction to Linear Algebra by Gilbert Strang (5th Edition):

   a) Problem Set 9.2: #17, #25, #26

2. Find a Schur decomposition for:

$$A = \begin{bmatrix} 3 & 0 & 0 & -1 \\ 1 & 2 & 0 & 1 \\ 2 & 0 & 4 & 2 \\ -1 & 0 & 0 & 3 \end{bmatrix} .$$

3. Show that if U is unitary and $A = U^H BU$, then B is normal if and only if A is normal.

4. Show that if x is an eigenvector of a normal matrix A corresponding to eigenvalue λ, then x is an eigenvector of A^H corresponding to the conjugate λ^*.

5. **Image Compression with Discrete Fourier Transform (DFT)** (60 *Points*): An RGB image is made up of three color channels (red, green, and blue). Each channel can be represented as a 2D matrix of intensity values. The 2D DFT of an RGB image maps each color channel from the spatial domain to the frequency domain. The transform applies independently to each of the color channels. We have

$$X[k_1, k_2] = \sum_{m=0}^{H-1} \sum_{n=0}^{W-1} x[m, n] \; e^{-j2\pi\left(\frac{mk_1}{H} + \frac{nk_2}{W}\right)} \qquad 0 \leq k_1 \leq W-1, \quad 0 \leq k_2 \leq H-1 \tag{1}$$

$x[m, n]$ is the spatial representation, $X[k_1, k_2]$ is the frequency representation, W is the image width, H is the image height.

For this task, you will need to take the DFT of an RGB image (baboon.png), remove some components (compression), take the inverse DFT of the remaining components, and display the compressed image. Follow the following steps:

   a) Load the 2D image using a suitable library like matplotlib.pyplot.imread

   b) Set the pixel values of the image to a range of [0, 1].

c) Compute the 2D DFT of the image using the numpy.fft.fft2 function.

d) Shift the zero frequency component of the DFT to the center of the array using numpy.fft.fftshift.

e) Calculate the magnitude spectrum of the shifted DFT using numpy.abs.

f) Determine a compression ratio r. This ratio represents the percentage of the largest DFT coefficients that will be kept after compression.

g) Calculate the threshold value based on the compression ratio. This threshold will be used to determine which DFT coefficients to keep and which to remove.

h) Apply compression by keeping only the DFT coefficients with magnitudes larger than the threshold. Set the remaining coefficients to zero.

i) Shift the zero frequency component back to its original position using numpy.fft.ifftshift. This step reverses the shift performed in step (d).

j) Compute the inverse DFT of the compressed DFT using np.fft.ifft2. This will reconstruct the compressed image in the spatial domain.

k) Extract the real part of the reconstructed image using .real.

l) Display the original and compressed images using matplotlib.pyplot.imshow.

m) Compute the mean squared error between the original and compressed image.

Repeat Steps (a) - (m) for compression ratio $r \in \{0.001, 0.003, 0.01, 0.03\}$. Submit a short report that includes your code, the compressed images, and their corresponding mean squared error. *NOTE: You can use MATLAB or Python for Q5.*