Problem 1 (3pts):

(a) Assuming that the set $\{1, \cos k\omega t, \sin k\omega t\}_{k=1}^{\infty}$ where $\omega = \frac{\pi}{T}$ spans the space of all square integrable functions $f(t)$ defined on the domain $t \in [-T, T)$, show that the set $\{1, \cos k\omega t, \sin k\omega t\}_{k=1}^{\infty}$ also forms a basis for any $f(t) \in L^2([-T, T))$.

(b) Show that the Fourier series

$$f(x) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} a_k \cos kx + b_k \sin kx$$

can be written as the complex series expansion

$$f(x) = \sum_{k\in\mathbb{Z}} c_k e^{ikx}.$$

where c_k's for $k \in \mathbb{Z}$ are constants that may be expressed in terms of a_k, b_k for $k = 0, \ldots, \infty$.

(c) Show that the Discrete Fourier Transform $\hat{\mathbf{f}}$ of a signal vector $\mathbf{f}$ preserves its energy, that is, $\|\hat{\mathbf{f}}\|^2 = \|\mathbf{f}\|^2$. [1]

Problem 2 (3pts):

(a) Active noise cancellation (used in headphones) typically work by generating anti-noise signals. The file `hwk1_p2a.mat` (attached with this assignment in the zipped folder `hwk1_files`) is a noisy audio signal. (After downloading the file, load the variables `piano_noisy` and the sample rate `Fs`. Listen to the audio using the MATLAB command `sound(piano_noisy,Fs)`). Generate an anti-noise signal, which when added to the original signal, eliminates the noise. Listen to verify that the anti-noise signal does indeed eliminate the noise. Submit **the anti-noise signal**.

(b) 2D convolution of an image with a Gaussian (or Gaussian-like) kernel is often used for blurring images. The variable `Xblurred`, in the file `hwk1_p2b.mat`, is the image of a dog (in floating points; use `imshow(uint8(Xblurred))` to display image) that was blurred using the filter

$$\frac{1}{100}\begin{bmatrix} 0 & 2 & 4 & 2 & 0 \\ 2 & 4 & 6 & 4 & 2 \\ 4 & 6 & 8 & 6 & 4 \\ 2 & 4 & 6 & 4 & 2 \\ 0 & 2 & 4 & 2 & 0 \end{bmatrix}.$$

Recover the original image. Submit a side-by-side image of the blurred and recovered image. (For example, assuming `X` is the recovered image, in floating points, you may use the command `imshow([uint8(Xblurred),uint8(X)])` to display both images).

[1] It is also okay to show that $\|\hat{\mathbf{f}}\|^2 = C\|\mathbf{f}\|^2$ where C is some constant.

Problem 3 (8pts):

For this question, the goal is to build a facial expression recognition system that can distinguish between *happy*, *sad*, *disgusted*, *fearful*, *angry*, *surprised*, and *neutral* faces. The principles underlying such pattern recognition are valuable in computer vision systems and fault detection systems. Here we will use the open-source facial expression datasets from the KDEF-dyn database. The datasets have been pre-processed for you based on [1]. You can download the processed data from the subfolder `KDEF_wavelet_data` in the `hwk1_files` zipped folder attached with this assignment. For each face category, or class, you are to select 75% of the data for *training*, and the rest will be used to *test* your classification/recognition algorithm. (You can make use of the code `get_train_test_data.m` to implement this step). Your homework should answer the following.

(a) Perform a principal component analysis (PCA) on the training data to get the mean face and the first five (5) *eigen-faces* (principal components). Plot all six (6) images (mean and eigenfaces).

(b) Plot the reconstruction error against the number of principal components and determine how many principal components c are needed to achieve a reconstruction error of 2% or less. Explain how to reconstruct any random image from your data and reconstruct a random image from your data set using 1, 10 and c principal components. Plot the original image and the three reconstructions.

(c) Project the training data to the first c *eigen-faces* to get the first c PCA coefficients. Plot a scatter diagram of the first two PCA coefficients for the entire training dataset. Make a color legend on the scatter plot to distinguish the classes (in MATLAB, you may use the `gscatter` function for such scatter plot).

(d) Classify the test data into one of the seven categories based on the PCA coefficients (however many that gives optimal results). Plot a confusion chart to show your classification accuracy for each of the seven classes (in MATLAB, you may use the `confusionchart` function for this).

(e) Now perform a linear discriminant analysis (LDA) on the training data. Plot a scatter diagram (similar to (c)) of the first two linear discriminant coefficients (obtained from the projection of your data to the first two *fisher-faces*).

(f) Classify the test data into one of the seven categories based on the LDA. Plot a confusion chart to show your classification accuracy for each of the seven classes.

(g) Compare your results in (c) with that in (e); and compare your results in (d) with that in (f).

You are welcome to use one or more of the methods of classification discussed in class after performing PCA and LDA, but you are expected to achieve an average classification accuracy of at least 75% in (f) on the test data. Do not use any inbuilt toolbox functions (such as MATLAB's `pca`) for the principal component and linear discriminant analyses. You may use inbuilt functions for the classification after performing PCA and LDA provided you clearly state what the function does in your submission. Include your well-commented codes in your submission.

References

[1] Frank Y. Shih, Chao-Fa Chuang, and Patrick S. P. Wang. Performance comparisons of facial expression recognition in jaffe database. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(03):445–459, 2008.