## Experiment 4 Predictive and Transform coding

## Purpose:

1. Understanding the basic concepts of predictive and transform coding.
2. Understanding why predictive and transform coding can benefit data compression.
3. Understanding the AR model and DCT transform.

## Principle:

The following formulas may be useful during the experiment.

* 1. AR model



2.2 Transfer function



## Procedure:

1. **Preprocess of the audio data**

We first create the audio data with a sampling frequency of 8000Hz and time duration 5s. One can use the file “audio.wav” directly or record the audio by yourself with function “Audio.m”. The data provided in “audio.wav” is a vector, i.e., 8000 samples/s times 5s.

1. **Predictive coding for linear prediction coding**

We first load the audio data by using the matlab function “audioread()” to read the file “audio.wav”. We can obtain the audio data ***x*** ( vector) and the sampling frequency. Plot the original signal based on the audio data.

What is the approximate entropy of the original signal? (See the hints below)

1. We now consider the LPC of the AR model to predict the audio signal, where the critical part is to calculate the optimal coefficients such that the mean squared error between the original signal and predicted signal is minimized. To achieve this, we set the order of the AR model =12, and use the Matlab function “lpc()” to obtain the optimal coefficients, where the input for the function are the training audio data ***x*** ( vector) and the order . The output of the lpc() is the optimal coefficients in equation 2.1 ( vector with ).
2. To obtain the error signal , i.e., the differences between the original signal and predicted signal, we can treat the AR model as a filter with the transfer function given in 2.2. By using the Matlab function “filter(***B***,***A***,***x***)”, we can obtain the error signal based on the original signal ***x***, where ***B*** is the coefficients in the numerator of the transfer function ( vector representing ), ***A*** is the denominator and equal to 1. (Note that the original signal passes the filter with transfer function 2.2 generates error signal)

Plot the error signal based on the output of the “filter(***B***,***A***,***x***)” .

What is the approximate entropy of the error signal? (See the hints below)

Plot the “PDF” of original and error signals.

Discuss on why prediction can benefit data compression.

Hint: One can use the Matlab function “histcounts(***x***,128,'Normalization','probability')” to calculate the “PDF” of the original signal, where the input ***x*** is the signal data, and the outputs are ***v*** and ***d***. ***v*** is a vector denotes the probability that the data samples fall within the 128 bins. ***d*** represents the boundary of the 128 bins and hence a vector. Using the Matlab function “histcounts(***x***, ***d***,'Normalization','probability')” to calculate the “PDF” of the error signal, where the input ***d*** is the ***d*** of original signal.

1. **Transform coding based on DCT for sub-image**

We consider the DCT transform of an sub-image provided in the file “subimage.mat”.

Let ***X*** denote the original sub-image data, and ***Y*** denote the data after DCT transform.

1. . Using the matlab function “dct2()” for DCT transform to obtain ***Y***.
2. . Using the matlab function “idct2()” to do inverse DCT transform for the matrix .

Comment on whether you can reconstruct the original matrix based on the matrix and why.

1. **Transform coding based on DCT for normal image**
2. We first load the image file “Mona Lisa.jpg” by using the Matlab function “imread()”. We then use the Matlab function “rgb2gray()” to change the image data into grayscale data, and the data is a matrix. To do DCT transform, we divide the whole image into sub-images with size .
3. Using the Matlab function “dct2()” to do DCT transform for each sub-image.
4. We now consider two different methods for data compression.

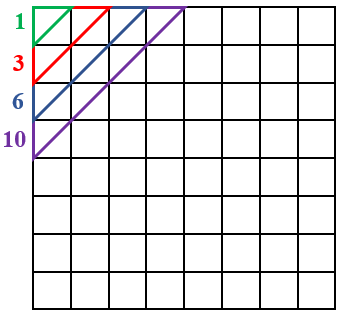
In the first method, we consider a threshold *k* to compress the data after DCT transform, where we treat the elements to be 0 if it is smaller than the threshold.

To recover the image, we use the Matlab function “idct2()” to do inverse DCT transform for each sub-images and merge the sub-images to form the original image.

After forming the original image matrix, use Matlab function mat2gray() to normalize the matrix, and use the Matlab function imshow() to depict the recovered image with different threshold values *k = 20, 50, 100, 300*, respectively. Comment on the quality of the recovered images and the data compression rate (i.e., the percentage of elements that are treated as 0).

In the second method, we only reserve the top left triangles with elements 1, 3, 6, 10, respectively and treat all the other elements to be 0 (see the figure below).

Depict the recovered images, comment on the quality of the recovered images and the data compression rate.



## Report Requirements:

Answer the questions given in the above procedure.