1 Introduction

The Discrete Cosine Transform is a variant of the Fourier Transform where a signal vector is expressed in terms of an orthonormal basis made up of sinusoids. The DCT is widely used in signal compression. In this project you will implement the DCT and use it to compress some audio signals and images.

2 Audio Coding [40 pts]

The whole secret to transform coding is to take your original signal, which you can think of as a vector in $\mathbb{R}^N$, and express it in terms of an orthonormal basis where most of the coefficients are close to zero. The Discrete Cosine Transform begins with a collection of N orthonormal vectors $\{q_1, q_2, \ldots, q_N\}$ that span $\mathbb{R}^N$ these vectors can be constructed from the following equation which gives us the ith element of vector q_k.

$$q_k^i = s_k \cos\left(\left(\frac{\pi}{2N}\right)(k-1)(2i-1)\right) \tag{1}$$

where

$$\mathbf{q}_k = \begin{pmatrix} q_k^1 \\ q_k^2 \\ \vdots \\ q_k^N \end{pmatrix} \tag{2}$$

The scalar s_k is just the normalization factor which is required to make sure that each of the vectors q_k has unit norm. In these equations the parameter $k \in \{1 \ldots N\}$ acts as a frequency index, higher values of k correspond to sinusoids with more cycles.

Your first task is to write a Matlab function with the following signature that you can use to generate each vector $q_k \in \mathbb{R}^N$. Note that this function takes k and N as arguments and returns a vector $q \in \mathbb{R}^N$.

```
function q = DCTvector(N, k).
```

Make sure to normalize your output vectors to unit length. You can use Matlab's norm function to help with this.

You should test your function by generating an 8 by 8 matrix Q whose columns correspond to the 8 DCT vectors in $\mathbb{R}^8$. You should plot all of these vectors to see what they look like. You may find the subplot function useful since it will allow you to plot all the vectors on one figure so that you can compare them. You should also verify that the resulting matrix Q is orthogonal by checking that $Q^T Q = I_8$.

Your next job is to write two functions, the first of which computes the Discrete Cosine Transform and the second of which computes the inverse Discrete Cosine Transform.

```
function x = DCT(y).
```

Takes a vector $y \in \mathbb{R}^N$ and produce as output a vector $x \in \mathbb{R}^n$ where the kth element of the vector can be computed from $x_k = q_k \cdot y$. Note that you can figure out the dimension of the input vector y using the size or length function. Matlab has a built in function called dot which is one of the many ways of computing the dot product between two vectors.

```
function y = InverseDCT(x).
```

Takes a vector $x \in \mathbb{R}^N$ and produce as output a vector $y \in \mathbb{R}^n$ where $y = \sum_{k=1}^{n} x_k \mathbf{q}_k$.

After you have written your DCT and InverseDCT functions you will use them to compress some audio signals. Specifically you are being asked to write a Matlab function with the following signature:

```
function y_compressed = DCTCompress1DSignal(y, pct).
```

Which takes as input two parameters, a signal vector $y \in \mathbb{R}^n$ and a parameter called pct which indicates the compression ratio. Setting pct to 20 would imply that the system would compress the input by computing the DCT transform and keeping the 20 percent of the coefficients with the largest absolute magnitude. For this function you will use the `DCT` and `InverseDCT` functions that you wrote earlier and in the middle you would identify the components that you are going to keep and zero the rest. You will find Matlabs `abs` and `sort` functions useful for this task. If one were using this procedure to transmit the signal you would only send the DCT components you are using resulting in a five fold reduction in data. Obviously the more coefficients you keep the closer the compressed signal should be to the original but the more data you need to store.

3 Image Coding [40 pts]

Just as the DCT can be used to compress 1D signals it can also be used to compress 2D signals like images, in fact it's the basis for the JPEG compression standard. We begin by consider images blocks that are 8 pixels on side. We can form an orthonormal basis for the set of 8 by 8 images by simply taking outer products of the 1D DCT vectors. That is if $\{q_1, q_2, \dots, q_8\} \in \mathbb{R}^8$ denote the 8 DCT vectors that span $\mathbb{R}^8$ then we can construct 64 8 by 8 matrices by taking all possible outer products $q_i q_j^T$. These 64 matrices happen to form an orthonormal basis for the set of all 8 by 8 matrices. You should check this out for yourself.

Note that if A is an 8 by 8 matrix in Matlab then $v = A(:)$ produces a 64 element vector by stacking up all the columns of A. So you can take the inner product of two matrices A and B as follows $dot(A(:), B(:))$. Similarly if v is a vector with 64 elements you can produce an 8 by 8 matrix A from its elements as follows $A = reshape(v, 8, 8)$.

You should write a function with the following signature

```
function B = CompressBlock (A, n)
```

which takes an 8 by 8 image block, A, computes the 64 DCT coefficients and then produces a new 8 by 8 block from the n DCT components with the largest magnitude coefficients.

Using this function as a building block you should write the following function

```
function out = CompressImage (I, n)
```

which compresses an input image, I, into an output image, out, by compressing every 8 by 8 block using the CompressBlock function with the parameter, n. You may need to trim the input image I to ensure that the number of rows and number of columns are both multiples of 8.

You are being provided with a few test images, cameraman.tif, moon.tif and mandi.tif which you should test your routines on. You are also provided with a short Matlab script that shows how to read and display images using built in functions.

4 Extra Credit [20 pts]

In class we briefly discussed the Haar wavelet which provides another approach to constructing an orthonormal basis for $\mathbb{R}^N$ by scaling and translating a basic on off pattern. Try repeating the image coding experiments using this basis instead of the DCT basis using two new functions

```
function B = CompressBlockHaar (A, n)
```

and

```
function out = CompressImageHaar (I, n).
```

5 Least Squares [20 pts]

In this section you are asked to compute a least squares fit to a set of data. You are provided with a Matlab data file entitled DataFit.mat that contains two vectors x and y. You should plot y vs x to see what this data looks like. Your job is to fit the following model to the data you are given.

$$y(x) = c_1 \sin x + c_2 \cos x + c_3 \log x + c_4 \tag{3}$$

Write a function with the following signature

```
function [c1, c2, c3, c4] = FitModel(x,y).
```

Which computes the best fit coefficients (c_1, c_2, c_3, c_4) in a least squares sense. As a test, you should plot your best fit curve through the data. You may find the 'fplot' function useful for this task. You should use the pseudoinverse formulation to solve this problem.