Analytical Questions

Please complete problems 10.24, 10.32, 3.21 in Oppenheim and Schafer (3rd Edition). Justify your answers!

Computational Questions

The computational questions are worth 10 points in total.

1 The DFT and DCT Matrices (3.5 pts)

Motivation. Many popular frequency transforms, such as the length-$N$ DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp \left( -j\frac{2\pi kn}{N} \right)$$

are linear in the input $x[n]$, and are designed to operate on sequences of finite length. As we saw in lecture, the DFT can be modeled as a matrix vector operation: $X = F_N x$, where $F_N$ is the $N \times N$ DFT matrix.

When working with applications involving real signals – such as images or audio – it is often preferred to use a frequency transform that stays within $\mathbb{R}^N$. A popular choice is the length-$N$ Discrete Cosine Transform (DCT):

$$\text{DCT}_N \{x\}[k] = \sum_{n=0}^{N-1} x[n] \cdot \underbrace{w_k \cos \left( \frac{\pi}{N} (n + \frac{1}{2})k \right)}_{k\text{-th vector } d_k},$$

$$w_k = \begin{cases} \frac{1}{\sqrt{N}}, & \text{if } k = 0, \\ \sqrt{\frac{2}{N}}, & \text{otherwise}. \end{cases}$$
for \( k = 0, \ldots, N - 1 \). This is again linear with respect to \( x[n] \), and we can similarly find a matrix 
\[
D_N = [d_0, \ldots, d_{N-1}]^H
\]
represent the transform as a matrix vector product. Also, the transform given in (2) is orthonormal, i.e. 
\[
D_N^H D_N = I, \quad \text{or} \quad D_N^{-1} = D_N^H.
\]
You may verify this if you like.

In MATLAB, \( F_N \) and \( D_N \) can be created via the commands \texttt{dftmtx(N)} and \texttt{dctmtx(N)} respectively.

1.1 Connecting DCT to DFT (2.5 pts)

From the discussion above, and since \( \cos(\phi) = \frac{1}{2}(e^{j\phi} + e^{-j\phi}) \), one may wonder if there is a direct relationship between the two transforms between the DCT and DFT.

The length-\( N \) DCT is, in fact, not directly related to the length-\( N \) DFT, but rather there is a direct relationship to the length-\( 4N \) DFT. Specifically, there is a way to use the entries of \( x \) to construct a vector \( y = Ax \in \mathbb{R}^{4N} \) (and corresponding signal \( y[n] \)) such that

\[
\text{DCT}_N\{x\}[k] = w_k \cdot Y[k], \quad k = 0, \ldots, N - 1,
\]
or

\[
D_N X = W [F_{4N} y]_{i=1:N} = W B y,
\]

where \( W = \text{diag}(w_0, \ldots, w_{N-1}) \in \mathbb{R}^{N \times N} \) is a diagonal matrix, and \( B \in \mathbb{C}^{N \times 4N} \). Since \( y = Ax \), we can further express the DCT matrix \( D_N \) as

\[
D_N = WBA.
\]

To-do. Write the function

\[
[D] = mydctmtx(N)
\]
to return \( D = D_N \). Do this through the following procedure:

1. Rewrite the DCT formula (2)

\[
\text{DCT}_N\{x\}[k] = w_k \cdot \sum_{n=0}^{N-1} x[n] \cdot \cos\left(\frac{\pi}{N}(n + \frac{1}{2})k\right)
\]

into a DFT form. This will allow you to figure out a mapping \( n'(n) \) which gives

\[
y[n'(n)] = x[n]
\]

for all nonzero entries \( y[n'] \).

2. Use this information to figure out \( A \in \mathbb{R}^{4N \times N} \).

3. Create \( B \in \mathbb{C}^{N \times 4N} \) from the first \( N \) rows of \( F_{4N} \).

4. Set \( W = \text{diag}(w_0, \ldots, w_{N-1}) \), and get \( D_N = WBA \in \mathbb{R}^{N \times N} \).
Hint. \( \exp\left(j\frac{2\pi k(2n+1)}{4N}\right) = \exp\left(-j\frac{2\pi k(4N-2n-1)}{4N}\right) \).

1.2 Separability of 2D discrete transforms (1 pt)

For images and other 2D signals, 2D equivalents of the DFT and DCT are needed, and frequencies have 2D interpretations based on the basis functions of the transform. That is to say, the 2D-DCT is a function of two frequency variables, which we will denote by \( k \) and \( l \) below. The size-\((N,N)\) 2D-DCT is given by the formula

\[
2\text{DCT}_N\{X\}[k,l] = (w_kw_l) \cdot \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X[m,n] \cos\left(\frac{\pi}{N}(n + \frac{1}{2})k\right) \cos\left(\frac{\pi}{N}(m + \frac{1}{2})l\right).
\]

(7)

\[\text{Figure 1.}\] Basis images for the size-(8,8) 2D-DCT. As \( k, l \) increases, oscillations increase along their respective directions.

In (7), we can move the terms around and observe that

\[
2\text{DCT}_N\{X\}[k,l] = \text{DCT}_N^{\{n\}} \{ \text{DCT}_N^{\{m\}} \{ X \} \}[k,l]
\]
\[
= \text{DCT}_N^{\{m\}} \{ \text{DCT}_N^{\{n\}} \{ X \} \}[k,l],
\]

That is, we can take the 2D-DCT by taking the DCT twice, once along the columns followed by once along the rows, or vice versa. Furthermore the order this is done doesn’t matter. This is called the separability property of the transform.

Given the DCT transform matrix \( D_N \), we should therefore be able to find a function \( f(X,D_N) \) that computes the size-\((N,N)\) 2D-DCT using matrix products alone.
To-do. Write the function

\[
\begin{bmatrix}
\mathcal{D}^2 X
\end{bmatrix} = \text{mydt2}(X, \mathbf{D}_N),
\]

using matrix products only, to return \( \mathcal{D}^2 X = f(X, \mathbf{D}_N) \), which contains the 2D-DCT of \( X \), i.e.

\[
\mathcal{D}^2 X(u,v) = 2\text{DCT}_N \{ X \} [u - 1, v - 1].
\]

This function should be a one-liner!

Hints.

- Let \( \mathbf{P}_1, \mathbf{P}_2 \in \mathbb{R}^{N \times N} \) represent linear operations on \( \mathbb{R}^N \). The product \( \mathbf{P}_1 \mathbf{M} \) applies a linear operation \( \mathbf{P}_1 \) independently to each column of \( \mathbf{M} \). Likewise, \( \mathbf{M} \mathbf{P}_2 \) applies a linear operation \( \mathbf{P}_2 \) independently to each row of \( \mathbf{M} \).
- If you haven’t completed \texttt{mydctmtx.m}, remember that you can still test \texttt{mydt2()} using \texttt{dctmtx()}.

Observations & discussion.

- Remember to test your functions using the built-in functions \texttt{dctmtx()}, \texttt{dct()}, \texttt{dct2()}, etc.
- How would you use \( \mathbf{D}_N \) to do the inverse DCT and inverse 2D-DCT?
- The size-\((N,N)\) 2D-DFT of an image \( X \) with \( \text{supp}\{X\} \subseteq \{0,\ldots,N-1\} \times \{0,\ldots,N-1\} \) is given by the formula

\[
2\text{DFT}_N \{ X \}[k,l] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X[m,n] \cdot e^{-j \frac{2\pi(mk+n)}{N}}, \quad (8)
\]

For \( k,l \in \{0,\ldots,N-1\} \). Is this transform separable? Can \texttt{mydt2()} be applied using \( \mathbf{F}_N \)?

- The 1D-DFT can be computed in time \( \mathcal{O}(N \log N) \). From this problem and the discussion, we may conclude that the DCT as well as the 2D-DFT and DCT (for a \( N \times N \) image) can also be computed in time \( \mathcal{O}(N \log N) \). On the other hand, a matrix computation between two \( N \times N \) square matrices requires \( \mathcal{O}(N^3) \) time to compute. In general, one would therefore not want to rely on matrix computations. However, matrix computations may prove to be faster over many repetitions of the 2D-DCT if \( N \) is small. Such a scenario arises in the next problem.

2 Lossy DCT Compression Encoding in JPEG (3.5 pts)

Motivation. For this problem we are going to explore the role of DCT in JPEG encoding. JPEG encoding is an extremely popular image compression technique due to its relative simplicity and large compression ratios: a 10 to 50 fold loss in image size can be achieved whilst retaining acceptable image quality. The JPEG pipeline is described in Figure 2.
The image is first split up into patches of size $8 \times 8$, thus standardizing the application of the DCT to the $(8,8)$ 2D-DCT for all incoming images. The DCT fulfills two functions. Firstly, the energy of natural images – or the squared magnitudes of its entries – mostly lie in low-frequency components, so DCT patches tend to aggregate signal energy to these entries.

Furthermore, the human eye, on average, also tends to be less sensitive to the differences in the high-frequency content in an image. This is conveniently exploited in JPEG by aggressively quantizing high-frequency entries in each DCT patch. Quantization is achieved dividing by each entry of the DCT patch entriwise by a quantization matrix $Q$, rounding afterwards so the result can be stored digitally. In other words, for an $8 \times 8$ image patch $X$

$$B_{ij} = \text{round}(G_{ij}/Q_{ij}),$$

where $G = \text{dct2}(X)$ and $B$ are $8 \times 8$ patches after DCT on the image patch $X$ (after normalization, see To-do) and quantization of $G$, respectively.

A common choice for the quantization matrix is

$$Q = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 19 & 26 & 60 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 69 & 69 & 56 \\
14 & 17 & 22 & 29 & 61 & 80 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 103 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 113 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 120 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 103 & 103 & 99 \\
\end{bmatrix},$$

the entries were determined subjectively based on human feedback. As a result the high-frequency entries being aggressively quantized, $B$ tends to be sparse, i.e. most of its entries are zero.

Sparsity is a very attractive quality for signal compression. On an $8 \times 8$ patch with $s$ nonzero entries. The contents of an image patch can in principle be stored using at most $2s$ rather than the full 64 values (since the location also needs to be stored).
For this problem, we will explore the DCT and quantization process by ignoring any further lossless compression techniques and other extraneous factors, and define the compression ratio achieved as

\[
\text{compression ratio} = \frac{\text{total pixels in image}}{2 \cdot \text{total number of nonzeros}}.
\]

(11)

**To-do.** Please work with the skeleton file `lossy.m` and make sure you have the file for the function `image2patches()` and `DCTQ.mat`, all of which are provided to you. The function `image2patches()` reads the specified image file, converts it to greyscale, then breaks it up into 8 × 8 patches. This process is taken care of for you in `lossy.m`, although you may try using different images.

The patches returned by `image2patches()` are stored in a cell array (you may need to look up how to use these); each cell contains an array representing a patch. Process each patch \(X^{(i)}\) by doing the following:

1. Apply the (8, 8) 2D DCT to \((X^{(i)} - 128)\) to get \(G^{(i)}\). You may use either matrix multiplications or `dct2()` to achieve this. The subtraction shifts the dynamic range from \([0, 255]\) to \([-128, 127]\), decreasing the magnitude of DCT values, and preventing the negative portion of the DCT from being wasted.

2. We will look at the average signal energy of each DCT coefficient over patches. Maintain an 8 × 8 array `avgenergy` by incrementing its entries so we end up with the following

\[
\text{avgenergy}(j, k) = \frac{1}{\# \text{ of patches}} \sum_{i=1}^{\# \text{ of patches}} \left( G^{(i)}_{jk} \right)^2.
\]

(12)

3. Quantize \(G^{(i)}\) by applying (9) to get \(B^{(i)}\). The quantization matrix should already be loaded for you from `DCTQ.mat` in the skeleton file.

4. We will keep track of the frequency that the entries \(B^{(i)}_{jk}\) are nonzero after quantization. Maintain the 8 × 8 array `nzeros` so we end up with the following

\[
\text{nzeros}(j, k) = \sum_{i=1}^{\# \text{ of patches}} 1\{B^{(i)}_{jk} \neq 0\}.
\]

(13)

Here \(1\{B^{(i)}_{jk} \neq 0\}\) just returns 1 if the entry is nonzero, and zero otherwise.

5. \(B^{(i)}\) now represents the encoded patch! The “size” of the resulting patch would be

\[
2 \cdot \sum_{j,k} 1\{B^{(i)}_{jk} \neq 0\}.
\]

Decode \(B^{(i)}\) by multiplying it entrywise with \(Q\), then applying the inverse 2D DCT and adding 128. Again you may use matrix multiplication or `idct2()`.

Put the reconstructed patch in the corresponding element in a cell array `pats_rec`.

The rest of the skeleton code will reconstruct the image from the patches, and display a set of statistics:
• A subplot display of the original and reconstructed images.
• The original image size and nonzero entries after quantization,
• The compression ratio defined in (11),
• The root mean squared error (RMSE) between the original and reconstructed images,
• The mean absolute deviation, or the average absolute difference between the pixels in the original and reconstructed images,
• The log average energy of the entries of $G^{(i)}$, and
• The log frequency that the entries of $B^{(i)}$ are nonzero.

See if you can make any observations from the statistics displayed (no need to submit observations).

Observations & discussion.

• (Optional) So far lossy compression and sparsity has only been achieved through the quantizer, which rounds small DCT values to zero. Modify your code so that it retains at most $s$ nonzero entries for each DCT patch by keeping the entries with largest magnitude. This lower-bounds compression ratio:
\[
\text{compression ratio} \geq \frac{\text{total number of pixels}}{2s}.
\]
Vary your choice of $s$, how does the quality of the recovered image and the compression statistics change?

• (Optional) Another way to modify the compression ratio is to change the quantization matrix: try scaling $Q$ with a factor $\alpha > 0$ and observe any changes.

• In practice, the locations of the nonzero entries are not stored directly. Rather entropy encoding techniques provide more efficient ways to encode the information obtained from $B$, and can significantly increase the compression ratio.

3 The Uncertainty Principle and STFT (3 pts)

Here we will visualize the STFT of a signal via its spectrogram. The STFT can be expressed via the following equation,
\[
\text{STFT}\{x\}[k,m] \triangleq \text{DFT}_N\{D_{m,N_0}x \cdot w\}[k],
\]
where $w$ is a window with $|\text{supp} (w)| = N$. The window is used to obtain a small chunk of the signal $x$ near the sample $m \cdot N_0$, and the DFT analyzes the frequency content of the signal at that chunk.

One important insight from Fourier analysis regarding the STFT is the uncertainty principle, which suggests that the spectral resolution of the STFT trades off with its temporal resolution. Enlarging the size of the window provides better accuracy with respect to frequency information, but precision with respect to time of occurrence is lost. On the other hand, shrinking the size of the window gives you better temporal precision, but resolution of the frequency content is lost.
A common technique to is to apply an overlapping window by choosing \( N_0 < N \). This does not circumvent the uncertainty principle, but allows one to get frequency information more smoothly with respect to time when using a large window. Of course there is no free lunch, as computational costs increase proportionally with overlap.

**Preparation.** In a script file `uncertainty.m`, sample over the following signals over \( t \in [0, 2] \) seconds at a sampling frequency of \( f_s = 1kHz \)

\[
\begin{align*}
x_1(t) &= \begin{cases} 
\cos(2\pi \cdot 250 \cdot t), & t \leq 1 \\
\sin(2\pi \cdot 250 \cdot t), & t > 1
\end{cases} \\
x_2(t) &= \text{chirp}(t, 0, 1, 100, 'Quadratic'),
\end{align*}
\]

to produce the arrays \( x_1 \) and \( x_2 \). The first signal \( x_1 \) is a pure tone of 250 Hz with a discontinuity at \( t = 1s \), and \( x_2 \) is a quadratic chirp - a signal whose frequency increases quadratically over time. As preparation, please briefly familiarize yourself with \( \text{chirp}() \) in the MATLAB documentation, and play back the tones using \( \text{sound}() \).

We will use the \( \text{spectrogram}() \) function on \( x_1 \) and \( x_2 \) to explore the uncertainty principle with respect to the STFT.

**To-do.** Write a script `uncertainty.m` to produce two separate figures for \( x_1 \) and \( x_2 \), with \( 2 \times 2 \) subplots for each figure. Each subplot should contain a spectrogram created by

\[
\text{spectrogram}(X, \text{WINDOW}, \text{NOVERLAP}, \text{NFFT}, F_s)
\]

(please look up how to use this!) using FFT size 512, plus the following parameters:

3. Window size: 512 points. Overlap: 0 points.

For each subplot, append an appropriate title, e.g. “Quadratic chirp - 512 pt window - No overlap”.

**Observations & discussion.**

- What happens to the frequency localization of the STFT on \( x_1 \), outside of the discontinuity, as the window size is increased?
- Now look at the discontinuity, what happens to the time localization as the window size is increased?
- What about the effect of changing the window size on \( x_2 \)?
- How does overlapping change the STFTs with the different window sizes for \( x_1 \) and \( x_2 \)?
Submission

Submission Instructions. Place the following – and only the following – files into a single .zip file with your UNI and homework number as the filename, e.g. ‘y13027_hw5.zip’.

mydftmtx.m, mydt2.m, lossy.m, uncertainty.m.

Please upload and submit to Courseworks before the beginning of class on Monday, Nov. 21st. You do not need to submit answers to the observation questions.

If you are submitting the analytical questions electronically, please name it in the form of ‘y13027_hw5_writeup.pdf’ and submit it separately from the .zip file.