EECS 16B Designing Information Devices and Systems II
Spring 2016 Anant Sahai and Michel Maharbiz **Homework 2**

This homework is due Monday February 8, 2016, at Noon.

1. Homework process and study group

Who else did you work with on this homework? List names and student ID's. (In case of hw party, you can also just describe the group.) How did you work on this homework?

2. Amplitude Modulation (AM) In electronic communication, transmission is achieved by varying *some aspect* of a *higher frequency signal*, ("carrier signal)", with some information-bearing waveform ("base signal") to be sent, such as an audio or video signal, a process known as **"modulation"**. Amplitude modulation is a specific scheme of modulation, where the amplitude of the carrier oscillations is varied. In this question, you will be led through the mixing of some simple base and carrier signals so you know the magic behind your traditional AM radio.

Recap of DFT: Consider a sampled signal that is a function of discrete time $x[t]$. We can represent it as a vector of discrete samples over time $\vec{x}$, of length n.

$$\vec{x} = \begin{bmatrix} x[0] & \dots & x[n-1] \end{bmatrix}^T \tag{1}$$

Let $\vec{X} = \begin{bmatrix} X[0] & \dots & X[n-1] \end{bmatrix}^T$ be the signal $\vec{x}$ represented in the frequency domain, that is

$$\vec{X} = U^{-1}\vec{x} = U^*\vec{x} \tag{2}$$

where U is a matrix of the DFT basis vectors.

$$U = \begin{bmatrix} | & & | \\ \vec{u}_0 & \cdots & \vec{u}_{n-1} \\ | & & | \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{2\pi i}{n}} & e^{\frac{2\pi i(2)}{n}} & \cdots & e^{\frac{2\pi i(n-1)}{n}} \\ 1 & e^{\frac{2\pi i(2)}{n}} & e^{\frac{2\pi i(4)}{n}} & \cdots & e^{\frac{2\pi i2(n-1)}{n}} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{\frac{2\pi i(n-1)}{n}} & e^{\frac{2\pi i2(n-1)}{n}} & \cdots & e^{\frac{2\pi i(n-1)(n-1)}{n}} \end{bmatrix} \tag{3}$$

Alternatively, we have that $\vec{x} = U\vec{X}$ or more explicitly

$$\vec{x} = X[0]\vec{u}_0 + \cdots + X[n-1]\vec{u}_{n-1} \tag{4}$$

In other words, $\vec{x}$ *can be always represented as* a linear combination of the DFT basis signals $\vec{u}_m$ with coefficients $X[m]$.

(a) Let n be the number of samples in a vector. Suppose we have a signal that is a complex exponential, $s[t] = e^{i\frac{2\pi Kt}{n}}$, which has a frequency of $\frac{K}{n}$ for some integer K and the number of samples n. In vector form it is given by:

$$e^{i\frac{2\pi Kt}{n}} \Rightarrow \begin{bmatrix} 1 & e^{i\frac{2\pi K}{n}} & e^{i\frac{2\pi K(2)}{n}} & \cdots & e^{i\frac{2\pi K(n-1)}{n}} \end{bmatrix}^T = \sqrt{n}\vec{u}_K \tag{5}$$

Note, that this is a scaled version of the Kth DFT basis vector.

Mixing two signals is defined to be *the process of element-wise multiplying them together*, i.e. computing $e^{i\frac{2\pi Kt}{n}}x[t]$. This amounts to *element-wise multiplication* $\odot$:

$$e^{i\frac{2\pi Kt}{n}}x[t] = (\sqrt{n}\vec{u}_K \odot \vec{x})[t] \tag{6}$$

The idea is, instead of transmitting the signal $\vec{x}$ directly, we would transmit the mixed signal. The mixing process is known as modulation. There are two questions you probably have are: how can we demodulate a signal? and, what does amplitude modulation do for us? You will know both answers by the end of this problem. But you don't have to answer them now.

Let's start simple. Suppose our signal $x(t)$ is a simple complex exponential with frequency $\frac{k}{n}$, that is $\vec{x} = \vec{u}_k$.

What is $\vec{x}$ written in the frequency domain? Now mix $\vec{x}$ with the complex signal $\sqrt{n}\vec{u}_K$. What is $\sqrt{n}\vec{u}_K \odot \vec{x}$? What is this new signal written in the frequency domain? What does mixing with a complex exponential do to the frequency of the signal?

(b) Now suppose that $\vec{x}$ is a cosine signal, that is $x[t] = \cos(2\pi\frac{k}{n}t)$. Mix this new $\vec{x}$ with the complex exponential $\vec{s} = \sqrt{n}\vec{u}_K$. What is the frequency domain representation of this new signal and how does it differ from $\vec{x}$?

(c) With the above $x[t] = \cos(2\pi\frac{k}{n}t)$, let's go one step further by mixing it with a carrier signal $\vec{s}$, where $s[t] = \cos(2\pi\frac{K}{n}t)$.

This is practically important because complex-exponentials need to be realized in a physical way that is real to build real-world AM radios.

What is the frequency domain representation of this new signal and how does it differ from $\vec{x}$?

Assume here that K is much bigger than k and in-turn, n is much bigger than K.

(d) Now that you see the effect of mixing the base signal with a cosine wave as the carrier signal. It is natural to wonder how can we undo this process. The most obvious answer — divide through by the cosine — suffers from the problem of having to divide by numbers that might be close to zero. Dividing by very small numbers is generally a bad idea in engineering contexts.

The technique that is used is called **de-modulation**. Now, mix $\cos\left(\frac{2\pi Kt}{n}\right)$ with the signal you get from the previous question. How many resulting terms do you get in the frequency domain? Considering the fact that $K \gg k$, what are the frequencies relative positions on the frequency spectrum? Have we recovered the original signal?

If not, what additional steps would we have to take? *(Hint: is there a subspace in which we have correctly recovered the original signal? How can we just keep that subspace?)*

(e) Refer to the iPython notebook for the implementations. Comment on what you see.

(f) Now Assume that we want to send two signals, $x_1(t) = \cos(2\pi\frac{k_1}{n}t)$ and $x_2(t) = \cos(2\pi\frac{k_2}{n}t)$. Assume that the carrier frequency $K \gg k_1$ and $K \gg k_2$. If we modulate the first signal by $s_1(t) = \cos(2\pi\frac{K}{n}t)$ and the second signal by $s_2(t) = \cos(2\pi\frac{2K}{n}t)$ and add them together, can we recover both signals?

3. Cosine Transform

We have seen in lecture and discussion that it can be convenient to write real signals as linear combinations of sines and cosines. Furthermore, because of the conjugate-symmetry property of the DFT coefficients for real vectors, we can also write them as weighted sums of phase-shifted cosines.

A key advantage of writing signals as linear combinations of phase-shifted cosines had to do with the special propety between cosines and LTI sytems.

(a) The first thing we want to do is to establish that a pure cosine input was transformed to a cosine output *at the same frequency*, while the magnitude and phase changes. First assume that n is odd.

More specifically, let C be some real circulant matrix with eigenvalues $\lambda_0, \ldots, \lambda_{n-1}$. If the input is

$$x[t] = \cos(\frac{2\pi p}{n}t + \phi_p),$$

then please show that the ouput is

$$y[t] = |\lambda_p| \cos(\frac{2\pi p}{n}t + \phi_p + \angle\lambda_p).$$

The change in phase $\angle\lambda_p$ to a cosine of frequency $\frac{2\pi p}{n}$ is called the phase response at p.

It will be useful to use the property that you showed in the last homework, namely that the λ_p exhibit conjugate symmetry if C is real — namely $\lambda_p = \bar{\lambda}_{-p}$ where $\bar{a}$ denotes the complex conjugate of a. (We also use a^* for this purpose, but that means conjugate transpose in general.)

(b) Repeat the previous part for n even — what, if anything, changes?

(c) To convince yourself that the cosine representation actually works, complete the corresponding ipython notebook. Please complete the code to compute the representation in terms of phase-shifted cosines and verify that this is indeed a correct representation.

(d) Calculate the phase response of systems with the following impulse response. Remember that the eigenvalues of C can be found by writing the impulse response in the DFT basis: $\vec{\lambda} = \sqrt{n}U^*\vec{h}$.

Sometimes, in order to better visualize the symmetry of the signals (vectors) that we are talking about, we will write them using an unusual ordering of their entries. We will place the 0th entry (ordinarily viewed as coming first) in the middle of the vector and count forward from there down and backwards from their upwards. This is done only for ease of visualization of the symmetries involved.

Let $n = 5$

$$\vec{h} = \begin{bmatrix} h[-2] \\ h[-1] \\ h[0] \\ h[1] \\ h[2] \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 2 \\ 1 \\ -1 \end{bmatrix}. \tag{7}$$

(e) Repeat the previous part for an even $n = 4$

$$\vec{h} = \begin{bmatrix} h[-1] \\ h[0] \\ h[1] \\ h[2] \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 2 \\ 1 \end{bmatrix}. \tag{8}$$

Be careful about including the $H[2]$ term.

Do you think that this example should be considered symmetric like the previous one was?

(f) Repeat the previous part with $n = 5$ and

$$\vec{h} = \begin{bmatrix} h[-2] \\ h[-1] \\ h[0] \\ h[1] \\ h[2] \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{9}$$

(g) The examples above are not accidents! Prove, for an arbitrary n odd:

- If the impulse response is symmetric about $t = 0$, then the phase response is always 0 or π. This means our λ_p values are always real.
- If the impulse response is symmetric around a point $t = m$, then the phase response is the same as it would be for a pure delay of m (i.e. the phase response is linear in the complex frequency k).

(Hint: for the second part, can you express the resulting matrix C as the cascade of a pure delay and something that looks like it came from the first part?)

(h) When n is even, we can actually come up with a different way of grouping terms. Using Euler's identity, prove the identity

$$e^{-i\frac{2\pi}{n}k} + e^{i\frac{2\pi}{n}(k+1)} = 2e^{i\frac{\pi}{n}} \cos\left(\frac{2\pi}{n}(k+\frac{1}{2})\right). \tag{10}$$

(i) Let $n = 4$. In the spirit of the identity above, compute the frequency-domain representation of $\vec{x}$

$$\vec{x} = \begin{bmatrix} x[-1] \\ x[0] \\ x[1] \\ x[2] \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 2 \end{bmatrix} \tag{11}$$

as a sum of cosines. Comment on the phases here. Do they depend on the actual vector given the symmetric structure?

(j) (optional) In the spirit of the identity earlier, see if you can come up with a representation for a real signal $\vec{x}$ for when n is even and the signal has the symmetry $x[0] = x[1], x[-1] = x[2], \cdots, x[-\frac{n}{2}+1] = x[\frac{n}{2}]$. This should represent an arbitrary such signal in terms of $\frac{n}{2}$ basis elements.

4. MIMO wireless signals

Ever wonder why newer wifi routers and cellular base stations have 4 or sometimes even more antennas on them? New wireless technologies actually use multiple antennas that each send their own signal on the same frequency band. The key here is not only do we encode signals in frequency bands, but also in spatial ones using a technique known as "Spatial Multiplexing".

We call this idea "MIMO" wireless, which stands for "multiple input multiple output". This technique is used in many standards including 802.11n/ac, 4G LTE, and WiMAX.

In this problem, we will explore how signals are decoded on the output end.

Consider the following:

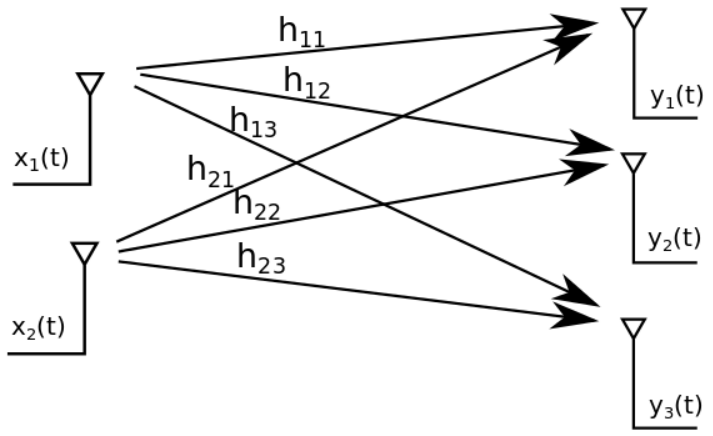
We have 2 transmit antennas and 3 receive antennas, each receive antenna gets some signal from each of the receive antennas. We can model the input output relation of the system as follows:

$$\begin{bmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \\ h_{13} & h_{23} \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

or

$$H\vec{x}(t) = \vec{y}(t)$$

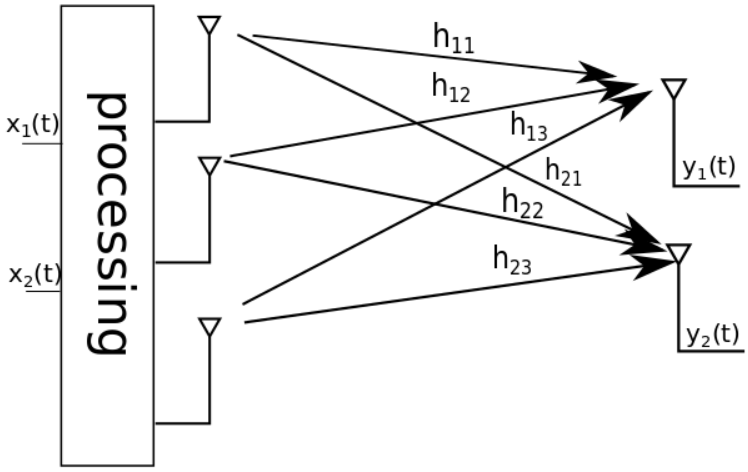
Here, H is the spatial-response matrix and it acts on the signals instantaneously at each time. For the purpose of this problem, we are going to pretend that there are no echoes across time.



(a) With our new MIMO wireless system, we want to recover the original $\vec{x}(t)$ signal after receiving the $\vec{y}(t)$. In order to do this, we will left multiply $\vec{y}(t)$ by some matrix A; ideally we should then exactly recover $\vec{x}(t)$ ($A\vec{y}(t) = \vec{x}(t)$). Using the SVD to decompose H, analytically write down what this matrix A should be.

(b) How is the solution you found in part (a) related to the least squares solution of this problem?

(c) What we just did is referred to as "post processing" or "post coding", and involves the receive end having more antennas than the send side. Many times this is not the case (eg. a wireless cell tower having many more antennas than a phone). What if we wanted to send 2 streams on 3 antennas and receive precisely those 2 streams back on the other end?



The channel is very similar to the one we had made in part (a). In fact, the original channel modelled with spatial response matrix H is precisely the transpose of this channel! Thus, we can say the spatial response matrix for this channel, lets call it H' is simply the following:

$$H' = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix} = H^T$$

Using the SVD of H and its relation to H', show how you can pre-process $x_1(t)$ and $x_2(t)$ so that you recover them precisely after they have been transmitted across the channel. To be more explicit, after the processing and transmission has been done $y_1(t) = x_1(t), y_2(t) = x_2(t)$.

(d) (Optional) Why do you think that we are using the SVD here? Is there a unique solution of what to transmit that will achieve the desired goal in the previous part? Why choose this approach?

5. Two-dimensional DFT

In lecture, we have seen that the concept of DFT can be extended to 2-dimensional signals. For example, images are considered as 2D signals in the "image domain." We can model a 2D signal as an $n \times n$ array M_f where

$$M_f = \begin{bmatrix} M_f[0,0] & M_f[0,1] & \cdots & M_f[0,n-1] \\ M_f[1,0] & M_f[1,1] & \cdots & M_f[1,n-1] \\ \vdots & \vdots & \cdots & \vdots \\ M_f[n-1,0] & M_f[n-1,1] & \cdots & M_f[n-1,n-1] \end{bmatrix} \tag{12}$$

Let the matrix M_F be the 2D DFT coefficients of M_f. The 2D DFT coefficients are given by

$$M_F[k,l] = \frac{1}{n} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} M_f[u,v] e^{-i\frac{2\pi}{n}(ku+lv)} \tag{13}$$

Conversely, an image-domain signal can be expressed by the linear combinations of its 2D DFT coefficients.

$$M_f[u,v] = \frac{1}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} M_F[k,l] e^{i\frac{2\pi}{n}(ku+lv)} \tag{14}$$

In this problem, we will play with images with 2D DFT using the IPython notebook. Use the provided IPython notebook file accompanying this homework.

(a) The equation (13) is the result of doing DFT on the column vectors of M_f first (let the result be M'_f), and then applying DFT again to the row vectors of M'_f. Demonstrate this using the IPython notebook with the image `campus_256.jpg`.

(b) The equation (14) is the result of doing inverse DFT on the row vectors of M_F first (let the result be M'_F), and then applying inverse DFT again to the column vectors of M'_F. Demonstrate this fact using the IPython notebook with the image `campus_256.jpg`.

(c) We can modify the 2D DFT coefficients of an image by zeroing out its high- or low-frequency coefficients. Use the IPython notebook to demonstrate the effects of modifying the 2D DFT coefficients of the image `campus_256.jpg`.

(d) Look at the provided image `einstein_monroe_256.jpg` and comment about what is special of this image. Use the IPython notebook to separate the hybrid image into one with a low-frequency Marilyn Monroe and one with high-frequency Albert Einstein.

(e) The discrete cosine transform (DCT) is another useful way of transforming an image-domain signal into a different 2d-frequency domain. Use the IPython notebook to comment and compare the results of using DFT and DCT to compress information in the frequency domain.

(f) Similar to the DFT, the DCT can be extended to 2 dimensions. This is done by first doing DCT on the column vectors of M_f (let the result be M'_f) and then applying DCT to the rows vectors of M'_f. Use the IPython notebook to process the image `tower.jpg`. Comment and compare the results of using 2D DFT and 2D DCT to compress information in the frequency domain.

(g) (Optional) How would you show that the 2D DFT is an orthonormal transformation into a new basis?

6. Haar Basis In class, you have been introduced to the DFT basis which has many useful properties. Another orthonormal basis with useful properties is the Haar Basis. Before showing you these properties, we will first show you how to construct the unnormalized Haar Basis, hierarchically.

Let N be an integer. Assume that we are working in an 2^N-dimensional real vector space (this Basis can be built for any arbitrally vector as long as we zero-pad to the next power of 2).

The first vector is $\vec{v_0} = [1,1,\ldots,1,1]^T = [(+1)_{2^n}]^T$. This is the DC term.

We will be illustrating with rows because we write from top to bottom in English and this makes it easier to visually see the hierarchical structure of this basis.

At the next level, we split the vector in the previous level where the left 2^{N-1} points are 1 and the right 2^{N-1} points are -1.

$$\vec{v_1} = [1,1,\ldots,1,1,-1,-1,\ldots,-1,-1]^T = [(+1)_{2^{n-1}},(-1)_{2^{n-1}}]^T$$

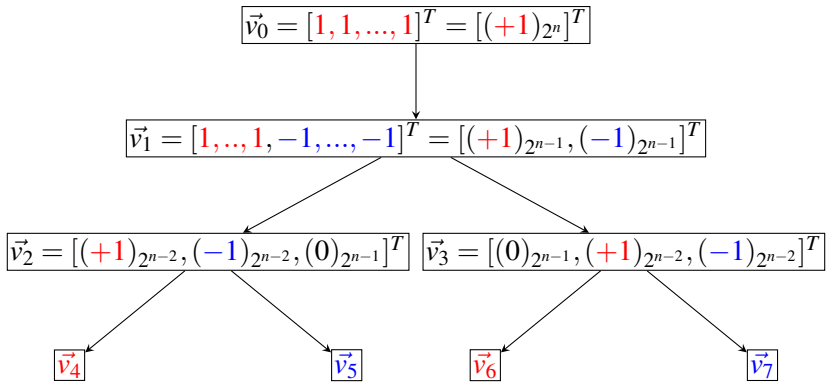
From this point forward, each vector creates two child vectors where the left child keeps the points where there parent is 1 and the right child keeps the points where the parent is -1 and the rest become 0. Then each child splits the vectors the same way as above. As such we get,

$$\vec{v_2} = [1,1,\ldots,1,1,-1,-1,\ldots,-1,-1,0,0,\ldots,0,0]^T = [(+1)_{2^{n-2}},(-1)_{2^{n-2}},(0)_{2^{n-1}}]^T$$

$$\vec{v_3} = [0,0,\ldots,0,0,1,1,\ldots,1,1,-1,-1,\ldots,-1,-1]^T = [(0)_{2^{n-1}},(+1)_{2^{n-2}},(-1)_{2^{n-2}}]^T$$

As such, this next level of child vectors each have 2^{N-2} points that are 1 and -1 and 2^{N-1} points that are 0.

You can view this as a tree:



This continues until there are 2^N vectors total. The final "leaves" of this tree have vectors that have exactly one +1 and exactly one -1 and the rest zeros. There are 2^{N-1} such leaf vectors. As is common in a binary tree, most of the nodes are leaves.

An example of a 4-dimensional unnormalized Haar Basis:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

More examples can be seen in the ipython code.

(a) Show that these vectors form an orthogonal basis. *(Hint: Try to understand how a vector is orthogonal to its descendents, siblings, and siblings' descendents. Then think about whether the two descendents of orthogonal vectors in this scheme can be anything other than orthogonal.)*

(b) In the ipython code, complete the normalize function so that we can use the normalized Haar Matrix to do orthonormal transformations. This is important because we currently have only orthogonal row vectors.

(c) Now, create the pulse function and plot the pulse function in the notebook. Then transform this vector to the Haar Basis and plot its new representation. Comment on the difference between the two plots.

(d) The Haar Basis has a very useful structure. Using this structure, we can actually perform the Haar Transform much faster then doing naive Matrix Multiplication with the Haar Transform Matrix. Given a vector of size N, the Fast Haar Transform can be computed in around $O(N)$ steps while the Matrix Multiplication method is completed using $O(N^2)$ steps. (The $O(\cdot)$ notation means that we are ignoring constant factors here.) The recipe is shown in the ipython notebook. For vectors of size 128, how much faster is the Fast Haar Transform compared to the Mattrix Multiplication approach?

(e) The Haar Basis can also be used to transform images using the 2D-Haar Transform. The approach is shown in the Ipython Notebook.

Comment on the structure of the image in the 2D-Haar Basis.

(f) (Optional) Combine with the code from the 2D-DFT problem to see if you can use the Haar Basis for compression. What happens when you throw away small coefficients?

7. Your Own Problem

Write your own problem related to this week's material and solve it. You may still work in groups to brainstorm problems, but each student must submit a unique problem. What is the problem? How to formulate it? How to solve it? What is the solution?

Contributors:

- Ming Jin.
- Alan Malek.
- Saavan Patel.
- Yen-Sheng Ho.
- Lev Tauz.