Module 1, B-set 1:
Signals and the Frequency Domain

The whole world is sigsys, really! Or maybe it's linear algebra...

November 2, 2016

Goals

By the end of this building block, you should be able to:

1. Identify examples of both continuous time (CT) and discrete time (DT) signals
2. Explain how to convert from CT to DT and vice-versa.
3. Qualitatively explain how the frequency content of an audio signal relates to how the signal sounds.
4. Calculate the frequency content of a discrete time signal.
5. Manipulate the frequency content of signals through frequency domain techniques.

What you might (will) have difficulty with:

1. Remembering Matlab syntax
2. Remembering linear algebra concepts from last semester
3. Manipulating complex numbers

Overview and Orientation

In class on Thursday last week, you started working on building an acoustic modem. You probably rediscovered Matlab and figured out how to turn a word into a binary number, and might have even built a transmitter to play that number as a sound, and decode the recorded sound wave into the original bits. Receiving is generally harder than transmitting, and a lot of the tools that you'll need to properly process the signal are concepts from iSIM. This Thursday, you started working qualitatively with the frequency content of signals. Here, we are going to dive deeper into going back and forth between discrete-time (DT) and continuous-time (CT), and to get a quantitative understanding of how we can analyze the frequency components in a DT signal. In developing quantitative tools to analyze the frequency content of signals, we will draw heavily on linear algebra material from last semester, as well as use complex numbers.

WARNING: By the time you're 4 or 5 hours into this, you should feel confident that you can complete the B-set in another 4-5 hours. If not, this is when you should **ask for help.** This means **talk to a colleague**, or **talk to a ninja**, or **track down an instructor**, or **send an email to an instructor**.

About Signals

Types of Signals

In class, we saw examples of continuous and discrete signals. Examples of continuous-time (CT) signals include the temperature in Boston as a function of time. Discrete-time (DT) signals are defined only at integer values of the argument. For instance, the daily closing value of the New York Stock Exchange. Note that we use the term time here to represent the free variable in the signal, as is commonly done in the signal processing world. In many situations "time" may actually refer to a different variable such as spatial position.

Audio samples that you generate in your computer are another example of a DT signal and the waveform coming out of the speakers in your computer is an example of a CT signal, which is the result of converting a DT signal The process of converting from CT to DT is called sampling and the process of going from DT to CT is called interpolation.

Interpolation and Sampling

In most signal processing systems (e.g. the Analog-Discovery modules, the audio processing hardware on your computer), conversion from CT to DT is done by sampling the CT signals at regular intervals. For instance Figure 1 illustrates a waveform $x(t)$ which is sampled every T_s time units to generate a DT signal, $x[n]$ where

$$x[n] = x(nT_s) \tag{1}$$

T_s here is the sampling period, and n is an integer. Hence $f_s = 1/T_s$ is the sampling frequency. The sampling operation can be realized using a number of circuit approaches.

Figure 2 illustrates a sine wave at a frequency of f_s which is sampled at a rate of 10 samples per cycle of the sine wave.

1. In this exercise, you will explore the effects of different sampling rates on a sinusoidal signal. You will see how sampling at too low a rate could result in ambiguity in the sampled signal. We shall use sine waves as they are relatively easy to visualize, and you can actually hear them if they are at appropriate frequencies. Run

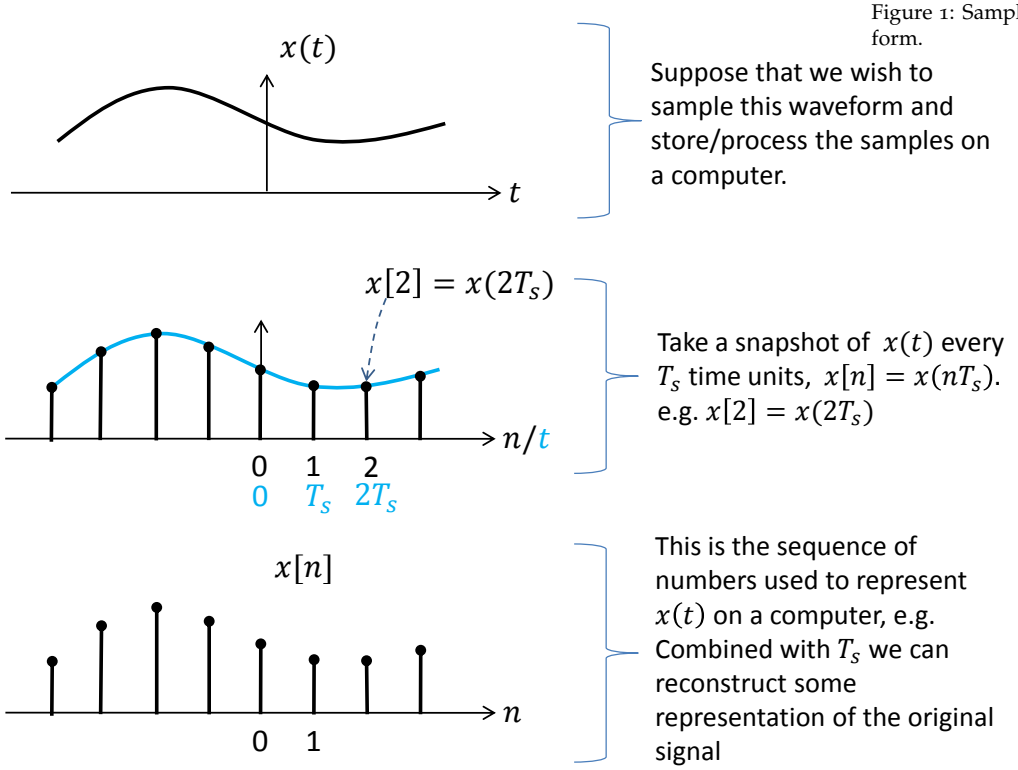


Figure 1: Sampling a continuous waveform.

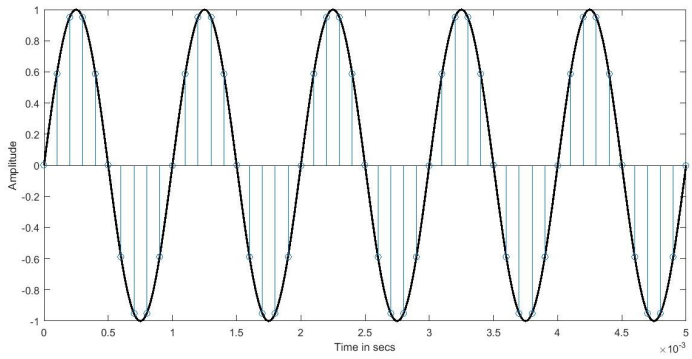


Figure 2: Sampling of a sinusoidal waveform.

the MATLAB code in the script `sine_sampling.m`. You should see two graphs of a sine wave, $x(t)$ of 1kHz, sampled at two different rates. We denote the sample rate by $f_s = 1/T_s$. Note that it is customary to represent sampled signals using a stem plot, i.e. `stem` in MATLAB. Of course, since your computer is a discrete-time device, the "continuous" waveform $x(t)$ is only an approximation, as it really is a DT signal with a high sample rate.

(a) How many samples are generated per cycle of $x(t)$ when it is sampled every $1/10000$ seconds and $1/5000$ seconds?

(b) What happens to the number of samples and the representation of the waveform if the sampling period is increased?

(c) In `sine_sampling.m`, change the dt1 and dt2 variables to `1/2000` and `1/1000` respectively, and run the code.

i. How do your graphs compare with those generated using the original sample rates?

ii. How many samples per cycle of the sine wave are generated in each case?

iii. What are the magnitudes of each sample?

(d) In `sine_sampling.m`, keeping the `dt1` and `dt2` variables at `1/2000` and `1/1000` respectively, change the code to plot a cosine wave instead of the sine wave. How do the resulting graphs compare with the previous part, and explain any differences you see.

It is evident that if you sample a CT signal too slowly, (i.e. if T_s is too large), you cannot obtain an accurate representation of the continuous waveform. In fact, if sampled at too low a rate, CT signals of different frequencies can be indistinguishable in DT. Consider the signal shown in Figure 3, where two cosine waves at frequencies of 440Hz and 2560Hz are shown. The samples are taken at a rate of 3000Hz. The samples, illustrated by the stems in the plot, are identical for both signals.

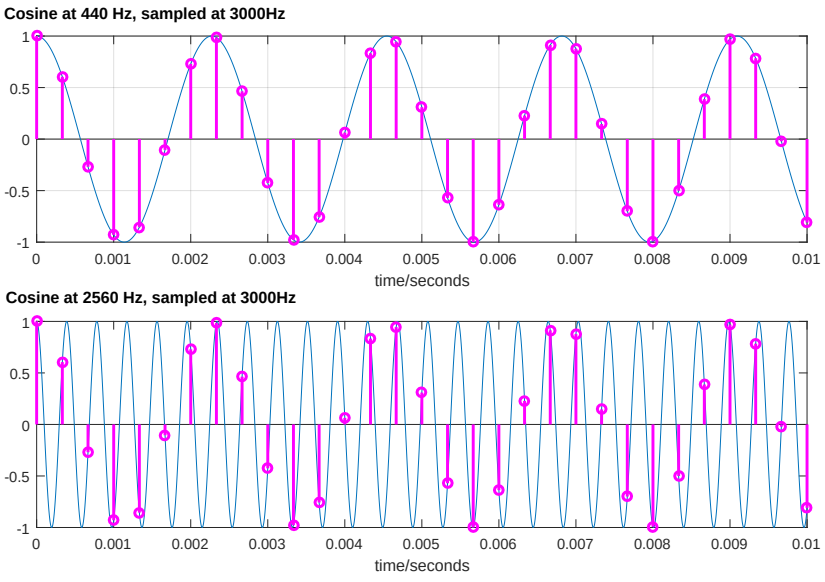


Figure 3: Cosine waves at 440Hz and 2560Hz, sampled at 3000Hz. DT samples for both Cosine waves are identical. Figure adapted from ThinkDSP, by Allen Downey.

The phenomenon that is illustrated in Figure 3 is known as *aliasing*. The cosine at 2560Hz is aliased such that it looks like a 440Hz cosine wave. The word alias comes from the Latin root *alius* meaning

other. Remarkably, if a CT signal is sampled at greater than twice the maximum frequency present in the signal, it can be *perfectly* reconstructed from its DT samples. Conversely, if we are trying to construct a CT signal from a DT signal at a sample frequency of F_s, then the highest frequency we could reconstruct perfectly is just less than $F_s/2$. We will discuss this more as the course progresses, including deriving this result. Rest assured.

What most of you did in your acoustic modem exercise was to generate a tone in DT and play it through your speaker in CT. Your computer hardware translated the DT samples into a CT waveform. A number of interpolation techniques are used in order to "fill in the gaps" between samples. Three canonical methods are

- Linear interpolation - connect a straight line between samples
- Zero-order hold - hold the value of the current sample until the next sample time
- Sinc/ideal interpolation - replace each sample value with the function $\text{sinc}(t)$. But this function extends from $t = -\infty$ to $t = \infty$, and has zero-crossings at regular intervals except at $t = 0$. We will get into this one in detail later in this module. The sinc function is plotted in Figure 4.

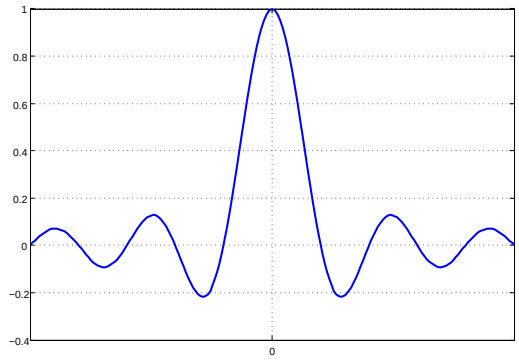
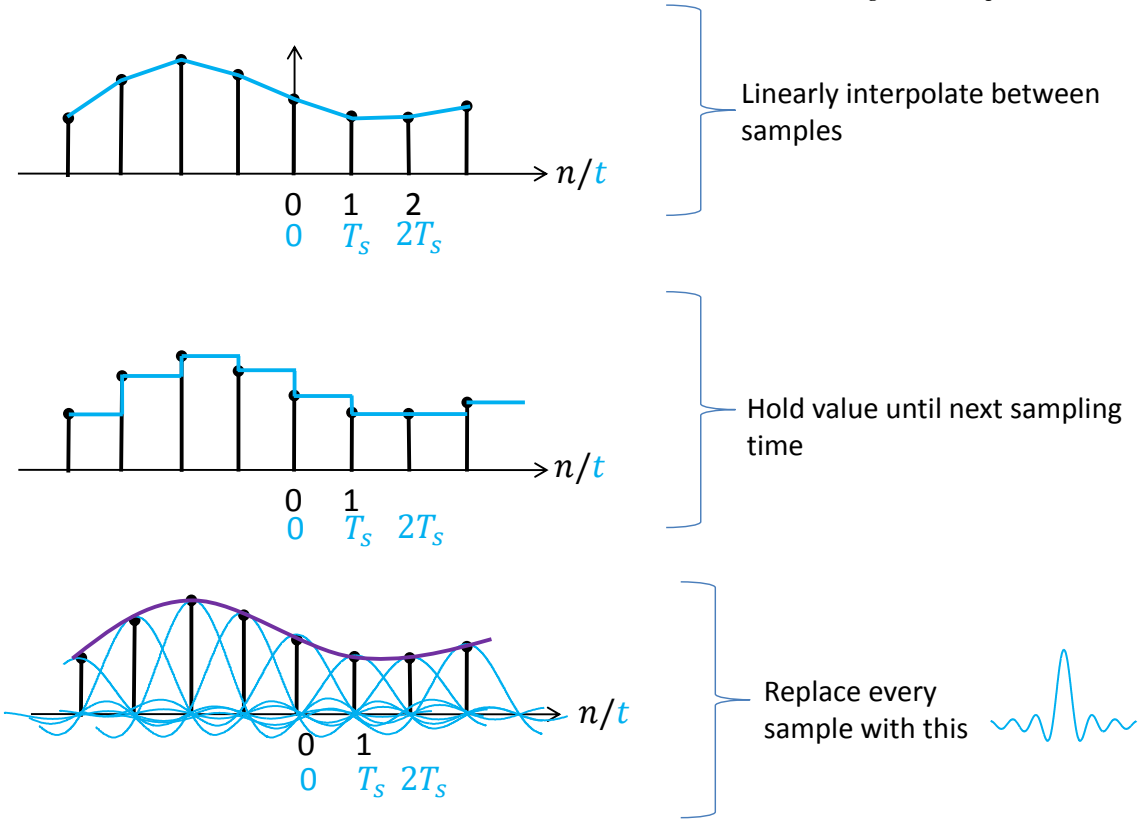


Figure 4: Sinc Function

The three interpolation methods give above are illustrated in Figure 5.

2. Next, we illustrate the aliasing effect using an audible tone (cosine wave). We shall simulate the effect of sampling a CT sine wave in MATLAB by calculating what the samples would be, if we were to sample a CT cosine wave. This can be calculated in MATLAB using the `cos` function. When you play the samples that you calculated through your speaker, the hardware on your computer converts the samples into a continuous waveform.

Figure 5: Interpolation from DT to CT



You may wish to turn down the volume on your computer for these exercises.

(a) Run the file `cos_aliasing.m` (After changing `Fs` to 3000). You will hear a cosine wave at 440Hz, sampled at 3000Hz. Modify the code to change the frequency of the cosine to 2560Hz. Run the modified code and describe what it sounds like relative to the 440Hz sine wave. Can you explain what you hear, based on Figure 3.

(b) Change the frequency of the tone back to 440Hz, but now change the sampling frequency to 8000Hz, and run the code. You should hear a tone at 440Hz. What do you expect to hear if you changed the frequency of the tone to 2560Hz? Try to answer this question without actually running the code first. Then, verify that your answer is accurate by making the appropriate change to the code and running it.

Frequency Content: Qualitative

Next, we shall do an exercise to get a qualitative understanding of the frequency content of an audio signal.

3. Load the following two wave files using the following MATLAB code snippets

```
[wb, Fs] = audioread('wideband_noise.wav');
[lf, Fs] = audioread('lowfrequency_noise.wav');
```

Play these two signals on your computer. You can use the following code to do so, but you should play the second, only after the first has stopped playing.

```
sound(wb, Fs);
sound(lf, Fs);
```

Plot the frequency content of the two signals using the following MATLAB code snippet

```
% these are the frequency indices to be plotted.  Don't
% worry about this or the fftshift below yet
fs = linspace(-Fs/2, -Fs/2*(length(wb)-1)/length(wb),
length(wb));
plot(fs,abs(fftshift(fft(wb))), 'b');
hold on
plot(fs,abs(fftshift(fft(lf))), 'm');
```

Explain what you hear based on the frequency content depicted in the plots.

Frequency Content: Quantitative

Earlier in this BSet, and in class, you performed some qualitative analysis and manipulation of the frequency content of signals. In this section, we shall develop some *quantitative* tools to analyze and manipulate the frequency content of signals. In particular, we will work with the Discrete-Fourier Transform (DFT) which is an operation that takes a finite-length discrete-time signal and decomposes it into its constituent frequency components. The well-known Fast-Fourier Transform (FFT) algorithm is a numerical algorithm that is used to efficiently compute the DFT, by exploiting certain symmetries in the computation of the DFT.

When we think of "pure" frequency components, we often think of sine waves at a particular frequency. A more powerful view of frequency, would be to think of a frequency in terms of a complex exponential of the form $e^{j\omega t}$, where ω would be the frequency of this complex exponential. From Euler's formula, we have

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t), \tag{2}$$

so the complex exponential is itself made up of the sum of a cosine and a sine with frequency ω.

The general strategy we are going to use is to project a finite-length DT signal represented as a vector, onto a set of orthonormal vectors which are vector representations of complex exponential functions. These projections will thus tell us "how much" of each complex exponential (hence frequency) is present in the signal. Since we will be using complex numbers, and some concepts from linear algebra that you encountered last semester, we shall do a brief review of these topics next.

More Review!

First let's review complex numbers. Define a complex number $z = x + jy$, where $j = \sqrt{-1}$. As you have seen before, we can express this complex number in terms of a complex exponential with an amplitude/magnitude and phase as follows

$$x + jy = Ae^{j\theta}. \tag{3}$$

The magnitude (or amplitude) is given by

$$A = |x + iy| = \sqrt{x^2 + y^2}$$

and the phase (or angle) is given by

$$\theta = \tan^{-1}\frac{y}{x}$$

The complex conjugate of $x + jy$ is defined as $x - jy$.

Since

$$\tan^{-1}\left(-\frac{y}{x}\right) = -\tan^{-1}\left(\frac{y}{x}\right)$$

the conjugate of $Ae^{j\theta}$ is $Ae^{-j\theta}$.

4. The following exercises are to help you get an understanding of some properties of complex numbers which will be useful in the subsequent development. Simplify and write the following numbers in the form of $a + jb$.

   (a) $Ae^{j\theta} \cdot Be^{j\phi}$

   (b) $|e^{j\theta}|$

   (c) $e^{j\pi}$

   (d) $e^{j2\pi}$

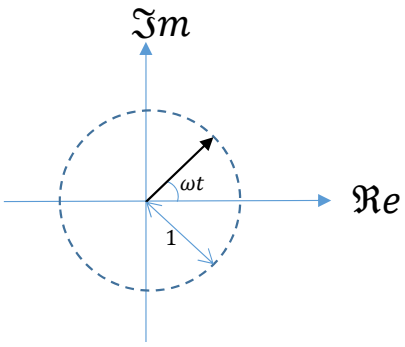


Figure 6: Complex exponential on the complex plane.

If you plot $e^{j\omega t}$ on the complex plane, you will get Figure 6, whose real part equals $\cos(\omega t)$ and whose imaginary part equals $\sin(\omega t)$.

For a positive ω, as time t increases, the complex exponential as depicted in Figure 6 spins around the unit circle counter clockwise, covering ω radians per unit time. ω here is the *angular frequency* of the complex exponential $e^{j\omega t}$. The frequency in Hertz (Hz), is the number of times the complex exponential goes around the unit circle per second. Suppose that t is in units of seconds.

Hence, if the angular frequency is ω radians per second, the frequency in units of Hz is $\frac{\omega}{2\pi}$ Hz. For negative frequencies ($\omega < 0$), the complex exponential spins clockwise as t increases.

Thinking of frequencies in terms of complex exponentials is more powerful than thinking in terms of cosines and sines in a number of ways, in particular, it enables us to compactly represent both frequency and phase. E.g. $e^{j\omega t+j\theta}$ is a complex exponential of frequency ω and phase θ. Moreover, it allows us to very directly connect frequency analysis of signals to fundamental concepts in linear algebra. We shall later see that the complex exponential is an *eigenfunction* for an important class of systems, with properties that are direct analogs of eigenvectors that you saw in linear algebra. It should be noted here however, that if we limit ourselves to real signals, the following analysis can be done using sines and cosines, but the notation and development of key ideas will be more cumbersome, and believe it or not, more confusing if done with sines and cosines.

In DT, the complex exponential is written as $e^{j\Omega n}$, for integer values of n. If the DT signal is the result of sampling a continuous-time signal at frequency of f_s samples per second, then the sampling period (time between consecutive samples) is $\frac{1}{f_s}$ seconds. Hence, an angular frequency of Ω radians per unit time in DT, corresponds to an angular frequency of Ωf_s radians per second in CT, and $\frac{\Omega f_s}{2\pi}$ Hz.

Next, let's review what projections and bases are about. Consider

vectors with real entries in 2-dimensions, and define the following

$$\mathbf{c}_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{4}$$

$$\mathbf{c}_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{5}$$

These vectors are orthogonal to each other since $\mathbf{c}_1 \cdot \mathbf{c}_2 = \mathbf{c}_2^T \mathbf{c}_1 = 0$. Moreover, these vectors are unit length. $\mathbf{c}_1$ and$\mathbf{c}_2$ therefore form an orthonormal basis for the set of all 2-dimensional vectors with real entries. Thus, any 2-dimensional vector with real entries $\mathbf{q}$, can be written as a linear combination of $\mathbf{c}_1$ and $\mathbf{c}_2$, with the projection (dot-product) of $\mathbf{q}$ onto $\mathbf{c}_1$ and$\mathbf{c}_2$ representing their respective weights. In other words,

$$\mathbf{q} = (\mathbf{c}_1^T \mathbf{q})\mathbf{c}_1 + (\mathbf{c}_2^T \mathbf{q})\mathbf{c}_2 \,. \tag{6}$$

The weights, $\mathbf{c}_1^T \mathbf{q}$ and $\mathbf{c}_2^T \mathbf{q}$ represent "how much" of the components $\mathbf{c}_1$ and $\mathbf{c}_1$ respectively are present in the vector $\mathbf{x}$.

5. Express $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ as a linear combination of $\mathbf{c}_1$ and $\mathbf{c}_2$.

We are soon going to use this idea to analyze the frequency content of signals, by taking an N-dimensional vector $\mathbf{x}$ which represents an N-sample long DT signal, and projecting it onto orthonormal vectors that are N-sample long complex exponentials. This is the fundamental idea behind the DFT.

Discrete Fourier Transform

In this section, we shall develop the DFT. We will start with theoretical development, followed by some applications of the DFT. We shall draw connections to the work you did last semester in the face-recognition module.

Theory of the DFT

First, we need a way to represent DT signals as vectors. Consider discrete-time signal $x[n]$ from time $n = 0, 1, 2, \cdots, N-1$. We can write this signal as an $N \times 1$ vector as follows

$$\mathbf{x} = \begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{pmatrix} \tag{7}$$

We are going to project this vector onto a set of orthonormal vectors which are vector representations of complex exponentials. (Recall that you did a similar operation last semester when projecting

vectorized pictures of faces onto a set of orthonormal vectors, i.e. the eigenfaces).

Before we project the vectors onto each other, we will have to define projections of vectors with complex entries. For vectors $\mathbf{a}$ and $\mathbf{b}$ whose entries are all real numbers, the projection of $\mathbf{a}$ onto a unit-length vector $\mathbf{b}$ is $\mathbf{b}^T\mathbf{a}$. For complex numbers, the "length" of the projection of $\mathbf{a}$ onto a unit-length vector $\mathbf{b}$ is defined as $\mathbf{b}^H\mathbf{a}$ where $\mathbf{b}^H$ is known as the conjugate transpose, or Hermetian transpose. The conjugate transpose of a vector or a matrix is the transpose of the vector or matrix, with each entry replaced by its complex conjugate.

We shall project $\mathbf{x}$ onto a set of orthonormal vectors $\mathbf{v}_i$, where i can be $i = 0, 1, \cdots, N-1$. Let's define $\mathbf{v}_i$ as follows

$$\mathbf{v}_i = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ e^{j2\pi \frac{i}{N} \cdot 1} \\ e^{j2\pi \frac{i}{N} \cdot 2} \\ e^{j2\pi \frac{i}{N} \cdot 3} \\ \vdots \\ e^{j2\pi \frac{i}{N} \cdot (N-1)} \end{pmatrix} \tag{8}$$

The vector $\mathbf{v}_i$ represents the signal $\frac{1}{\sqrt{N}} e^{j\omega n}$ for $n = 0, 1, \cdots N-1$, where the frequency of the complex exponential is $\omega = 2\pi \frac{i}{N}$.

The next few of questions are designed as a guided derivation of the steps required to write $\mathbf{x}$ as a sum of scaled complex exponentials.

The first step is to show that the vectors $\mathbf{v}_i$ form an orthonormal basis. We start by showing that $\mathbf{v}_i$ and $\mathbf{v}_k$ are orthogonal for $i \neq k$. Note that

$$\begin{aligned} \mathbf{v}_i^H \mathbf{v}_k &= \frac{1}{N} \left(1 \; e^{-j2\pi \frac{i}{N} \cdot 1} \; e^{-j2\pi \frac{i}{N} \cdot 2} \; \cdots \; e^{-j2\pi \frac{i}{N} \cdot (N-1)} \right) \begin{pmatrix} 1 \\ e^{j2\pi \frac{k}{N} \cdot 1} \\ e^{j2\pi \frac{k}{N} \cdot 2} \\ \vdots \\ e^{j2\pi \frac{k}{N} \cdot (N-1)} \end{pmatrix} \\ &= \frac{1}{N} \sum_{\ell=0}^{N-1} e^{j2\pi \frac{k-i}{N} \cdot \ell} = \frac{1}{N} \sum_{\ell=0}^{N-1} \left(e^{j2\pi \frac{k-i}{N}} \right)^{\ell} \end{aligned} \tag{9}$$

6. First we shall consider the specific case of $N = 8$, $i = 0$ and $k = 1$, to illustrate that $\mathbf{v}_i^H \mathbf{v}_k = 0$.

   (a) Write the summation in (9) out for this special case.

   (b) On the complex plane, plot each term of the summation in (9).

   (c) Based on your plot, explain why $\mathbf{v}_i^H \mathbf{v}_k = 0$.

7. Now we prove the same property for a general N, using a different approach, (although you can use an approach similar to the one above here as well). Observe that the right-hand-side of (9) is a geometric series with N terms, where the ratio of consecutive terms is $e^{j2\pi\frac{k-i}{N}}$. Use the geometric series summation formula to show that for $i \neq k$,

$$\mathbf{v}_i^H \mathbf{v}_k = 0, \tag{10}$$

   Hint: $e^{j2\pi m} = 1$ for all non-zero integers m.

8. If $i = k$, show that

$$\mathbf{v}_i^H \mathbf{v}_k = 1. \tag{11}$$

Since there are N vectors $\mathbf{v}_i$, each of which has N-elements, the vectors $\mathbf{v}_i$ form an N-dimensional, orthonormal basis for the space of N-dimensional complex vectors. In other words, any N dimensional complex vector can be written as a linear combination of the $\mathbf{v}_i$ vectors.

Since $\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_{N-1}$ form an orthonormal basis, we can find coefficients $a_0, a_1, \cdots, a_{N-1}$ such that any N-dimensional vector $\mathbf{x}$ can be expressed as

$$\mathbf{x} = a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_{N-1}\mathbf{v}_{N-1}. \tag{12}$$

Multiplying both sides of the equation above with $\mathbf{v}_i^H$, results in

$$\mathbf{v}_i^H \mathbf{x} = \mathbf{v}_i^H \left(a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_{N-1}\mathbf{v}_{N-1}\right) \tag{13}$$

9. By simplifying the right-hand side of the previous equation, find a_i in terms of the $\mathbf{v}$ vectors and $\mathbf{x}$.

Your answer to the previous part should indicate that you can express $\mathbf{x}$ as follows:

$$\mathbf{x} = (\mathbf{v}_0^H\mathbf{x})\mathbf{v}_0 + (\mathbf{v}_1^H\mathbf{x})\mathbf{v}_1 + (\mathbf{v}_2^H\mathbf{x})\mathbf{v}_2 + \cdots + (\mathbf{v}_{N-1}^H\mathbf{x})\mathbf{v}_{N-1} \tag{14}$$

Since the $\mathbf{v}_i$ are complex exponentials of different frequencies, we have decomposed the signal $\mathbf{x}$ into a sum of scaled complex exponentials at different frequencies. The projection of $\mathbf{x}$ onto $\mathbf{v}_i$, i.e. the term $\mathbf{v}_i^H\mathbf{x}$ thus tells us "how much" (magnitude and phase) of the complex exponential $e^{j2\pi\frac{i}{N}n}$ (which is represented by $\mathbf{v}_i$) is present in the signal $\mathbf{x}$.

The operations of computing the coefficients of the $\mathbf{v}_i$ terms in (14) can be accomplished by a matrix multiplication. Consider an $N \times N$ matrix $\mathbf{W}$ whose i-th row equals $\mathbf{v}_i^H$. In other words

$$\mathbf{W} = \begin{bmatrix} -\mathbf{v}_0^H - \\ -\mathbf{v}_1^H - \\ -\mathbf{v}_2^H - \\ \vdots \\ -\mathbf{v}_{N-1}^H - \end{bmatrix} \tag{15}$$

This matrix is known as the N-point DFT matrix. If we write out its entries, we have

$$\mathbf{W} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{2\pi j}{N}} & e^{-\frac{2\pi j}{N}2} & e^{-\frac{2\pi j}{N}3} & \cdots & e^{-\frac{2\pi j}{N}(N-1)} \\ 1 & e^{-\frac{2\cdot 2\pi j}{N}} & e^{-\frac{2\cdot 2\pi j}{N}2} & e^{-\frac{2\cdot 2\pi j}{N}3} & \cdots & e^{-\frac{2\cdot 2\pi j}{N}(N-1)} \\ 1 & e^{-\frac{3\cdot 2\pi j}{N}} & e^{-\frac{3\cdot 2\pi j}{N}2} & e^{-\frac{3\cdot 2\pi j}{N}3} & \cdots & e^{-\frac{3\cdot 2\pi j}{N}(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{(N-1)\cdot 2\pi j}{N}} & e^{-\frac{(N-1)\cdot 2\pi j}{N}2} & e^{-\frac{(N-1)\cdot 2\pi j}{N}3} & \cdots & e^{-\frac{(N-1)\cdot 2\pi j}{N}(N-1)} \end{bmatrix} \tag{16}$$

Suppose that

$$\tilde{\mathbf{x}} = \mathbf{W}\mathbf{x}\,. \tag{17}$$

By recalling the operations involved in matrix multiplication, we can see that

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_0^H \mathbf{x} \\ \mathbf{v}_1^H \mathbf{x} \\ \vdots \\ \mathbf{v}_{N-1}^H \mathbf{x} \end{bmatrix} \tag{18}$$

$\tilde{\mathbf{x}}$ is thus a vector that has the weights of the frequency components in $\mathbf{x}$ as its entries. This vector is known as the DFT of $\mathbf{x}$, and this is what `fft(x)` computes in MATLAB. Computing $\tilde{\mathbf{x}}$ is called taking the DFT (or FFT) of $\mathbf{x}$.

The inverse of the DFT matrix is known not surprisingly as the Inverse DFT matrix. It turns out that $\mathbf{W}^{-1} = \mathbf{W}^H$ which is because $\mathbf{W}$ is made up of orthonormal entries. (If you are motivated to do so, you can look up your notes from last semester on orthogonal matrices). It turns out that

$$\mathbf{x} = \mathbf{W}^H \tilde{\mathbf{x}} \tag{19}$$

This is what `ifft(x)` computes in MATLAB. Computing $\mathbf{x}$ from $\tilde{\mathbf{x}}$ is called taking the IDFT (or IFFT) of $\tilde{\mathbf{x}}$. Writing out the entries of

$\mathbf{W}^H = \mathbf{W}^{-1}$, we have

$$\mathbf{W}^{-1} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{2\pi j}{N}} & e^{\frac{2\pi j}{N}2} & e^{\frac{2\pi j}{N}3} & \cdots & e^{\frac{2\pi j}{N}(N-1)} \\ 1 & e^{\frac{2\cdot 2\pi j}{N}} & e^{\frac{2\cdot 2\pi j}{N}2} & e^{\frac{2\cdot 2\pi j}{N}3} & \cdots & e^{\frac{2\cdot 2\pi j}{N}(N-1)} \\ 1 & e^{\frac{3\cdot 2\pi j}{N}} & e^{\frac{3\cdot 2\pi j}{N}2} & e^{\frac{3\cdot 2\pi j}{N}3} & \cdots & e^{\frac{3\cdot 2\pi j}{N}(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{(N-1)\cdot 2\pi j}{N}} & e^{\frac{(N-1)\cdot 2\pi j}{N}2} & e^{\frac{(N-1)\cdot 2\pi j}{N}3} & \cdots & e^{\frac{(N-1)\cdot 2\pi j}{N}(N-1)} \end{bmatrix}, \tag{20}$$

The N-point DFT matrix can be generated in MATLAB using `dftmtx(N)`.

Properties and applications of the DFT

10. By hand, find the 64-point DFT of the vector $\mathbf{x}$ whose n-th entry is $\cos\left(\frac{\pi}{4}(n-1)\right)$. Hint: use the fact that $\cos\theta = \frac{1}{2}e^{j\theta} + \frac{1}{2}e^{-j\theta}$. Another hint: We will not be so cruel as to make you explicitly multiply a 64×64 matrix with a 64×1 vector, so think about ways in which this can be simplified.

11. Take a 64 point DFT of a cosine wave of frequency $\frac{\pi}{4}$ radians per unit time in MATLAB. You can do this by first generating a vector `x = cos(pi/4 * [0:1:63]);` and multiplying it with the DFT matrix. Plot the magnitude of the DFT of the cosine to verify your answer to the previous part.

In signal processing, the DFT is often written in terms of a summation, and there is some variation on the scale factor used (i.e. the $\sqrt{N}$ term in the definition of $\mathbf{W}$). When written in terms of a summation, the N-point DFT can be expressed as

$$X_i = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x[k] e^{-j\frac{2\pi}{N}ik}, \tag{21}$$

where X_i is the i-th entry of the vector $\tilde{\mathbf{x}}$. Recall here that X_i is "how much" (both magnitude and phase) of the frequency $e^{j2\pi\frac{i}{N}}$ is present in the signal $x[n]$.

The IDFT is expressed as

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_i e^{j\frac{2\pi}{N}nk}, \tag{22}$$

Writing it in this form enables us to more directly understand certain properties of the DFT, but on the other hand, writing it in matrix-vector form enables us to directly connect the DFT to concepts in linear algebra such as projection and bases which we used last semester.

In MATLAB, the `fft` command returns the DFT as computed using the matrix product given in (19). The frequencies that are computed by the `fft` operation range from 0 to $\frac{N-1}{N}2\pi$ radians per second, since the frequencies are evaluated for $\frac{2\pi i}{N}$ for $i = 0, 1, \cdots, N-1$.

12. In this problem, we shall show that the N point DFT is periodic with period N, and study some of the implications of this fact.

    (a) Show that the N-point DFT is periodic with period N, in other words, using (22), show that

    $$X_{i+N} = X_i \tag{23}$$

    (b) Assuming that N is even, find the smallest positive value of k for which $X_k = X_{-1}$.

    (c) Assuming that N is even, find the smallest positive value of k for which $X_k = X_{-N/2}$.

    Your answers to the previous parts should indicate that a high positive frequency can appear as a low negative frequency. If we wish to use the DFT to analyze the frequency content of signals, it is helpful to know what we should consider to be the highest frequency that the DFT computes. On first pass, it may seem that the highest frequency that the DFT computes is $2\pi\frac{N-1}{N}$ radians per unit time. But on the other hand, note that

    $$\begin{aligned} e^{-j\frac{2\pi}{N}n} &= 1 \cdot e^{-j\frac{2\pi}{N}n} \\ &= e^{j2\pi} \cdot e^{-j\frac{2\pi}{N}n} = e^{j\frac{2\pi(N-1)}{N}n}. \end{aligned} \tag{24}$$

    Hence, a DT complex exponential at frequency $-\frac{2\pi}{N}$ is equal to a DT complex exponential at a frequency $\frac{2\pi(N-1)}{N}$. So its completely legitimate to consider $\frac{2\pi(N-1)}{N}$ to be a small, negative frequency component. In fact, this interpretation is the one that makes the most sense if we consider a DT signal that originates from a CT signal. It turns out that if a DT signal is the result of sampling a CT signal at a rate high enough that there is no aliasing, any frequency content that shows up in the DFT in the frequency range of π to 2π radians per unit time, will have originated from lower negative frequencies in the CT signal. We shall prove and analyze this graphically next week when we revisit sampling, so don't despair if this is confusing.

    For this reason, it is often helpful to visualize frequencies ranging from $-\pi$ radians per unit time to $\frac{N-1}{N}\pi$ radians per unit time. In other words, it is helpful to plot the frequency content of signals centered around 0 radians per second, as the low frequency

components will then be close to zero and large high positive and negative frequency components will be at the two edges. For this reason, we typically run the results of the `fft` command through `fftshift` in MATLAB, before plotting. `fftshift` reorders the entries of the result of the FFT so that the frequencies range from $-\pi$ to $\frac{N-1}{N}\pi$ radians per unit time.

Applications of the DFT.

In these next two exercises, we use the DFT to perform some filtering. The general idea is to take the DFT of a signal to compute the signal as a function of frequency. The result of the DFT is then manipulated (e.g. by attenuating some frequencies relative to others), before an IFFT is taken to convert it back to a function of time.

13. In MATLAB type `load handel`. This command will load a data file provided with MATLAB that contains an audio segment `y` sampled at `Fs` samples per second.

    (a) Type `sound(y/max(abs(y)), Fs)` to hear this audio signal. We divide `y` with the maximum of the absolute value in order to normalize it.

    (b) What are the highest and lowest frequencies that this signal can have, given the sample rate, and assuming that it was sampled at a rate such that there was no aliasing?

    (c) Using the `fft` and `ifft` commands, implement a low-pass filter which suppresses the frequencies in the following ranges $[-Fs/2, -Fs/4]$ and $[Fs/4, Fs/2]$ Hz, by a factor of 10. You will have to figure out how to go from Hz frequencies to radians/unit time given a sampling frequency here. Plot the absolute values of the FFTs of the signal before and after you do this processing.

    You should play the filtered signal on your computer to check if it sounds as you expected. In order not to play too large an amplitude through your speaker, you should normalize the amplitude of the signal by dividing by the largest value in your signal. If your processed signal is contained in `yfilt`, you can play it through your speakers using `sound(real(yfilt)/max(abs(real(yfilt))), Fs)`. Note that, filtering the signal in this manner may cause your signal to have imaginary components. We shall see why this is the case later in the class.

14. Load the wav file guitar_riff_hum.wav in MATLAB using
    `[x, Fs] = audioread('guitar_riff_hum.wav');`
    This file contains a guitar riff played through an amplifier that has a low-frequency hum in the background. This hum is well

approximated by a cosine at a particular frequency. This hum is meant to simulate 60 Hz noise that you may encounter in building electronic circuits, however since your speakers will not reproduce anything at 60Hz, we injected a signal at > 250Hz into this sample. You can play this audio clip in MATLAB using `sound(x, Fs);`. If you have headphones (which should be able to reproduce 60Hz noise), and would like to try this example out with real 60Hz noise, you can try this with the file 'guitar_60Hz_hum.wav'.

(a) By analyzing the FFT of `x`, find what frequency this hum is at. Please upload any plots you may have generated to do this.

(b) By manipulating the FFT, of `x`, and then taking the IFFT, reduce the effects of the hum. Please upload any code you used to do this, and any relevant plots if you used them to verify what you did.