Instructor: Amy Orsborn — TA: Christopher Yin
aorsborn@uw.edu — c8yin@uw.com

LAB 3 - CONNECTIONS BETWEEN DTFT, DFT, AND FFT

Due: Sunday 05/22/2022 at 11:59pm

The goal of this lab is to explore the connections between the Discrete Time Fourier Transform (DTFT), the Discrete Fourier Transform (DFT), and the Fast Fourier Transform (FFT).

LAB INSTRUCTIONS

1. You might have to spend time outside the allocated lab hours to finish the lab. In doing so, you can approach any of the course instructors be it your lab TA, the other TA(s) or the main instructor.
2. You may work in teams or groups of 1-3 members. When it comes to collaborating with students in other groups, please do not share code but only discuss the relevant concepts and implementation methods. You may change group or team members for different labs but you cannot change group members for the given lab you are working on.
3. Please document your code well by using appropriate comments, variable names, spacing, indentation, etc.
4. The starter code is not binding on you. Feel free to modify it as you wish. Everything is fine so long as you're getting the right results.
5. Please upload the .ipynb file and the PDF version of the .ipynb file to canvas. One notebook per team is fine and any one team member can upload the file. The required file(s) must be uploaded by the deadline.

1 Discrete Time Fourier Transform (DTFT)

In the lectures you have learned that the DTFT can be used for analyzing signals and linear time-invariant systems. The DTFT and it inverse transforms are defined as

$$\text{(DTFT)} \qquad X\left(\mathrm{e}^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x[n]\mathrm{e}^{-j\omega n} \tag{1}$$

$$\text{(inverse DTFT)} \qquad x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\mathrm{e}^{j\omega})\mathrm{e}^{j\omega n}\mathrm{d}\omega \tag{2}$$

Since the DTFT is periodic with period 2π it is sufficient to focus on a single DTFT period, when analyzing $X\left(\mathrm{e}^{j\omega}\right)$. Typically, the DTFT is evaluated and displayed for a range of $[0,\ 2\pi]$ or $[-\pi,\ \pi]$.

If we are to evaluate the DTFT of a (not necessarily trivial) discrete time signal $x[n]$ using computational instead of analytical techniques, there are two things we have to be aware of:

1. Using computer software, we cannot compute an infinite sum as in the DTFT definition in (1). Luckily, in practice $x[n]$ has a finite length N. That is, for causal signals $x[n] = 0$ for $n < 0$ and $n \geq N$.
2. We can only evaluate the DTFT for a finite set of frequencies. That is, we have to chose some integer K and compute the DTFT according to (1) at a set of frequencies ω_k, where $k = 1 \ldots K$.

Truncating the sum to a finite length is equivalent to multiplying the signal $x[n]$ with a rectangular window defined as

$$w[n] = \begin{cases} 1, & 0 \leq n \leq N-1 \\ 0, & \text{else} \end{cases} . \tag{3}$$

Then we may define the truncated signal to be

$$x_{\text{tr}}[n] = w[n]x[n]. \tag{4}$$

The DTFT of $x_{\text{tr}}[n]$ is given by:

$$X_{\text{tr}}\left(\mathrm{e}^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x_{\text{tr}}[n]\mathrm{e}^{-j\omega n} = \sum_{n=0}^{N-1} x[n]\mathrm{e}^{-j\omega n} \tag{5}$$

We would like to compute $X\left(\mathrm{e}^{j\omega}\right)$, but the truncation window distorts the desired frequency characteristics. That is, $X\left(\mathrm{e}^{j\omega}\right)$ and $X\text{tr}\left(\mathrm{e}^{j\omega}\right)$ are generally not equal. To understand the relation between these two DTFT's we need to convolve in frequency domain:

$$X_{\text{tr}}\left(\mathrm{e}^{j\omega}\right) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(\mathrm{e}^{j\sigma}\right) W\left(\mathrm{e}^{j(\omega-\sigma)}\right) \mathrm{d}\sigma \tag{6}$$

where $W\left(\mathrm{e}^{j\omega}\right)$ is the DTFT of $w[n]$. Eq. (6) is the periodic convolution of $X\left(\mathrm{e}^{j\omega}\right)$ and $W\left(\mathrm{e}^{j\omega}\right)$. Hence, the true DTFT $X\left(\mathrm{e}^{j\omega}\right)$ is smoothed via convolution with $W\left(\mathrm{e}^{j\omega}\right)$ to produce the truncated DTFT, $X_{\text{tr}}\left(\mathrm{e}^{j\omega}\right)$.

We can calculate $W\left(\mathrm{e}^{j\omega}\right)$:

$$W\left(\mathrm{e}^{j\omega}\right) = \sum_{n=0}^{N-1} \mathrm{e}^{-j\omega n} = \begin{cases} \frac{1-\mathrm{e}^{-j\omega N}}{1-\mathrm{e}^{-j\omega}} = \mathrm{e}^{-j\omega(N-1)/2}\,\frac{\sin(\omega N/2)}{\sin(\omega/2)}, & \text{for } \omega \neq 0, \pm 2\pi, \ldots \\ N, & \text{for } \omega = 0, \pm 2\pi, \ldots \end{cases} . \tag{7}$$

1.1 Assignment 1a: Compute and plot the DTFT of the rectangular window (15 %)

For this assignment you will compute the DTFT of the rectangular window according to Eq. (7) and then visualize it's magnitude in linear and decibel (dB) scale.

1. Complete the function `dtft_rect_win()` provided in the starter code. The function takes two input parameters `N` and `K`:
   - `N`: The number of samples of the window. This is the same `N` as in (3), (5), and (7).
   - `K`: The number of frequency points for evaluating the DTFT. `K` should be used to generate an array of frequencies ω_k for which we want to compute the DTFT of the rectangular window. For this lab, all ω_k should be equally spaced between $-\pi$ and π. That is if `K` is an odd number, ω_k can be computed in Python as `omega_k = np.linspace(-np.pi, np.pi, K)`.

   The function `dtft_rect_win()` should return an array of DTFT samples of the rectangular window with length `N` evaluated at the frequencies ω_k
2. Complete the function `plot_dtft()` that plots the magnitude of a given DTFT. (You will also use this function later in the lab.) The input to the function is a DTFT array `X` and a parameter `scale`.
   - `X`: `X[k]` corresponds to ω_k as defined above. That is `X` is the DTFT from $-\pi$ to π and evaluated at `K = len(X)` equally spaced frequencies.
   - `scale`: Indicates whether the DTFT is to be plotted in a linear (`scale = 'lin'`) or decibel (dB) (`scale = 'dB'`) scale. The conversion from linear to dB is $X_{\text{dB}} = 20\log_{10}(|X_{\text{lin}}|)$
3. Use the functions `dtft_rect_win()` and `plot_dtft()` to compute and plot the DTFT of a rectangular window with length `N = 20` for at least `K = 501` values. Create two separate subplots: One showing the DTFT in linear and the other in dB scale. Make sure to properly label the x- and y-axis.

Once you have completed this task, your plots should look like the ones in Figure 1. Note that if $X_{\text{lin}}(\omega_k) = 0 \rightarrow X_{\text{dB}} = -\infty$. Numerical approximations causes X_{dB} to assume very small values (e.g. -350 dB). For better visualization, limit your y-axis of the dB plot to a reasonable range such as shown in Figure 1 below.

1.2 Assignment 1b: Compute and plot the DTFT of 2 windowed sine waves (15 %)

For this assignment you will create two sinusoidal signals $x_1[n]$ and $x_2[n]$ with different frequencies and compute and plot their DTFTs. Later we will compare the DTFTs of $x_1[n]$ and $x_2[n]$ with their DFTs.

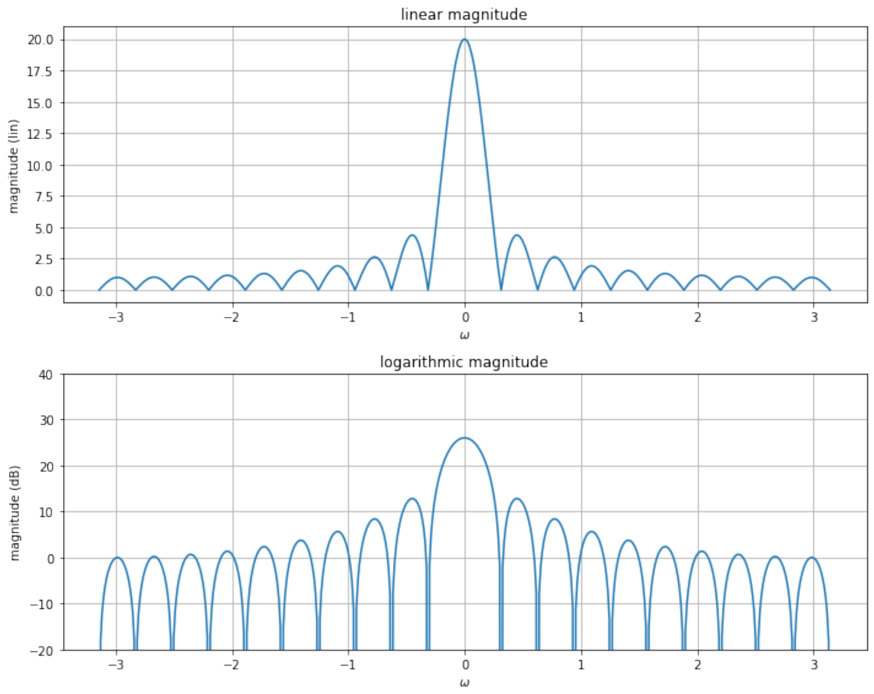

Figure 1: DTFT (magnitude) of rectangular window in linear and dB scale.

1. Complete the function `dtft()` provided in the starter code. The input parameters of the function are

   - `x`: A discrete time signal.
   - `K`: The number of frequency points for which we want to compute the DTFT.

   We want to use `dtft()` to compute the DTFT of an arbitrary discrete time signal over a range of $[-\pi, \pi]$. The parameter `K` is defined as in the `dtft_rect_win()` function and should be used to generated an array of equally spaced frequencies `omega_k` for which the DTFT is evaluated. To compute the DTFT of `x` use Eq. (5).

2. Now assume we have two sine waves

$$x_1(t) = \sin(2\pi f_1 t) \tag{8}$$

$$x_1(t) = \sin(2\pi f_2 t) \tag{9}$$

   with $f_1 = 200$ Hz and $f_2 = 375$ Hz. To obtain the discrete time signals $x_1[n]$ and $x_2[n]$, both signals are sampled using a sampling frequency of $f_s = 1$ kHz. Compute $x_1[n]$ and $x_2[n]$ for $n = 0, ..., N-1$, where $N = 20$, and compute and plot their DTFTs for at least 501 frequency values (i.e., $K \geq 501$). Recall that the relation between the normalized frequency ω and the natural frequency f (in Hz) is $\omega = 2\pi f / f_s$. For plotting the DTFTs, you should use the function `plot_dtft()` that you implemented in Assignment 1a and set the `scale` parameter to `'dB'`. You can plot both DTFTs in the same figure.

Discussion (5 %):

Determine expressions for $X_1\left(\mathrm{e}^{j\omega}\right)$ and $X_2\left(\mathrm{e}^{j\omega}\right)$ (the DTFTs of the non-truncated signals). Compare those expressions with the DTFTs of the truncated sines. Briefly describe the effects of the truncation (i.e., windowing with a rectangular window) on the DTFTs.

2 Discrete Fourier Transform (DFT)

In Section 1 we have illustrated how to compute the DTFT $X\left(\mathrm{e}^{j\omega}\right)$ of a discrete time signal $x[n]$ for a finite set of frequencies. The number of frequency points K determines how fine the DTFT is sampled, and should generally be chosen significantly larger than the length of the signal N to get a good approximation of the DTFT of $x[n]$.

The DFT $X[k]$ of the signal $x[n]$ of length N is also a sampled version of the DTFT $X\left(\mathrm{e}^{j\omega}\right)$. However, instead of sampling at K frequency points, we sample $X\left(\mathrm{e}^{j\omega}\right)$ at N frequency points uniformly spaced in the range $[0,\,2\pi)$. That is, we can substitute

$$\omega = 2\pi k/N \tag{10}$$

for $k = 0, 1, ...(N-1)$ in (5) to obtain the DFT of $x[n]$:

$$X[k] = X_{\mathrm{tr}}\left(\mathrm{e}^{j\omega}\right)\Big|_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x[n]\mathrm{e}^{-j\omega n}\Bigg|_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x[n]\mathrm{e}^{-j2\pi\frac{nk}{N}}. \tag{11}$$

Eq. (10) shows the relation between the frequency index k and the normalized frequency ω. If our discrete time signal $x[n]$ was obtained by sampling a continuous time process $x(t)$ with sampling frequency f_{s} (in Hz), the corresponding unnormalized frequency f_k (in Hz) is given by

$$f_k = \frac{k}{N} f_{\mathrm{s}}. \tag{12}$$

Matrix Representation of the DFT

The DFT Eq. (11) can be implemented as a matrix-vector product

$$X = \boldsymbol{W} x, \tag{13}$$

where $\boldsymbol{W}$ is an $N \times N$ matrix, and both X and x are $N \times 1$ column vectors. The elements of $\boldsymbol{W}$ are

$$\boldsymbol{W}(k,n) = \mathrm{e}^{-j2\pi\frac{nk}{N}} \tag{14}$$

2.1 Assignment 2: Compare DFT and DTFT (15 %)

1. Complete the function `dft()` provided in the starter code. The function takes a discrete time signal $x[n]$ as an input and returns its DFT $X[k]$. We recommend implementing the DFT according to Eq. (13). You might find the function `scipy.linalg.dft()` helpful. Note that you are not allowed to use the function `np.fft.fft()` here.
2. Complete the function `plot_dft()`. The function should plot the DFT $X[k]$ for frequencies $[-\pi,\pi)$ or $[0,2\pi)$ (see parameter `shift`). That is, you have to convert from the frequency index k to the normalized frequency ω. The function takes three input parameters:
   - `X`: DFT array
   - `shift`: If `shift = True` (default), the DFT should be plotted for a range $[-\pi,\pi)$. If `shift = False`, the DFT should be plotted for a range $[0,2\pi)$. Note that `X` corresponds to a range of $[0,2\pi)$. That is, if `shift = True`, you have to shift the DFT array by π. You might find the function `np.fft.fftshift()` useful.
   - `scale`: Indicates whether to plot the DFT in linear (`scale = 'lin'`) or decibel (dB) (`scale = 'dB'`, default) scale.
3. Compute the DFTs of $x_1[n]$ and $x_2[n]$ from Assignment 1b and plot them on top of their DTFTs. Do not plot them in the same figures you created for Assignment 1b, but instead make a new figure where you plot the DFTs on top of the corresponding DTFTs.

Discussion (5 %):

Briefly describe how the DFT of $x_1[n]$ differs from the DFT of $x_2[n]$. How do the DFTs compare to the expression you got for $X_1\left(\mathrm{e}^{j\omega}\right)$ and $X_2\left(\mathrm{e}^{j\omega}\right)$, i.e., the DTFTs of the non-truncated signals?

3 Fast Fourier Transform (FFT)

In practice, we do not typically compute the DFT via Eq. (13). Instead, we use a fast implementation of the DFT that is called Fast Fourier Transform (FFT). One of the main properties of the DFT implementation that can be exploited to speed up computation is that the computation can be divided into two parts as shown below:

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \\ &= \sum_{m=0}^{N/2-1} x_[2m]e^{-j2\pi(2m)/N} + \sum_{m=0}^{N/2-1} x_[2m+1]e^{-j2\pi(2m+1)/N} \\ &= \sum_{m=0}^{N/2-1} x_[2m]e^{-j2\pi m/(N/2)} + e^{-j2\pi k/N}\sum_{m=0}^{N/2-1} x_[2m+1]e^{-j2\pi m/(N/2)} \end{aligned} \tag{15}$$

So, we see that it can be divided such that the portion of the time domain signal with even indices can be considered separately from the odd portion. This lets us effectively divide the current problem at hand into "even" and "odd" parts. Cooley and Tukey further applied the divide and conquer approach to continue dividing the divided problem into sub problems which would be divided even further. In this fashion, a divide and conquer algorithm was formulated that could solve the problem of computing the DFT in $O(N \log N)$ time as opposed to $O(N^2)$. Consistent with the above factorization the FFT algorithm can be implemented as follows:

1. Check if the length of the signal is a power of 2 and print out error message if not.
2. Define the base case for a particular value of N (e.g., $N = 32$) where the regular DFT is computed and returned if the length of the signal is equal to that lower threshold for N.
3. Use Algorithm 1 below to recursively call the FFT.

```
Data: $x[n]$
$x_e \leftarrow$ all values of $x[n]$ with even indices starting from zero
$x_o \leftarrow$ all values of $x[n]$ with odd indices starting from one
$X_e \leftarrow$ FFT($x_e$)
$X_o \leftarrow$ FFT($x_o$)
factor $\leftarrow e^{-j2\pi k/N}$
return $X_e + factor \cdot X_o$
```

Algorithm 1: FFT

3.1 Assignment 3: Implement FFT algorithm (15 %)

1. Complete the function `fft()` provided in the starter code by implementing the above FFT algorithm.
2. Test it out on the test signal defined in the starter code.
3. In a new block use the `time.time()` function to find how long it takes to find the DFT using your implementation of `dft()` and your implementation of `fft()` and compare the two.

4 Effect of Window Size on DFT

In this last section, we study the effect that the window size N has on the DFT and how this effects the ability to detect two sinusoidal components with different frequencies and amplitudes in a signal. To do so, let's assume we have the following continuous time signal

$$x(t) = x_A(t) + x_B(t), \tag{16}$$

with

$$\begin{aligned} x_A(t) &= 10\cos(2\pi f_A t), \quad f_A = 17\,\text{Hz} \\ x_B(t) &= \frac{1}{10}\cos(2\pi f_B t), \quad f_B = 34\,\text{Hz} \end{aligned}. \tag{17}$$

Assume the sampling frequency is 100 Hz. For all plots you should use the 'dB' scale.

4.1 Assignment 4a: FFT of x for two different window length (10 %)

1. Compute the normalized frequencies ω_A and ω_B corresponding to the frequencies f_A and f_B.
2. Compute and plot the FFT of $x[n] = x_A[n]+x_B[n]$ for two different window length $N_1 = 256$ and $N_2 = 200$. Note that for N_1 you can use the FFT function you implemented earlier. However, since N_2 is not a power of 2 you have to either use the DFT function or make use of numpy's FFT function `np.fft.fft()`. For plotting the FFTs use the `plot_dft()` function. Scale the x-axis to a range of $[\omega_A - 0.2, \omega_B + 0.2]$

Discussion (5 %):

Why do the FFTs for the two different window length look so different? Hint: The plots you have generated for Assignment 2 might be helpful for answering this question.

4.2 Assignment 4b: Compare FFTs and DTFTs (10 %)

1. Create a plot that contains
   - the DTFT of $x_A[n]$ using window length N_1,
   - the DTFT of $x_B[n]$ using window length N_1,
   - the FFT of $x[n]$ using window length N_1,

   When computing the DTFTs, chose a K of at least 4001. Scale the x-axis to a range of $[\omega_A - 0.2, \omega_B + 0.2]$
2. Repeat the above with N_2 instead of N_1.

Discussion (5 %):

Based on the plots, which window length is better suited for detecting the two components $x_A[n]$ and $x_B[n]$ from the FFT of the signal $x[n]$?