Signals and Systems 2025
Programming Assignment 2

Deadline: 5/26 1:19 pm

Discrete Fourier Transform

The objective of this section is to learn how to use Python's NumPy `` `fft` `` function.

1. Background

To analyze the frequency domain of a finite-duration and discrete-time signal $x[n], n = 1, 2, \ldots, N$, its discrete Fourier transform (DFT) is defined as:

$$X_k = \sum_{n=1}^{N} x[n] e^{-j\frac{2\pi}{N}(n-1)(k-1)}, k = 1, 2, \ldots, N \tag{1}$$

The inverse DFT (IDFT) of X_k is defined as:

$$x[n] = \frac{1}{N} \sum_{k=1}^{N} X_k e^{j\frac{2\pi}{N}(n-1)(k-1)}, n = 1, 2, \ldots, N. \tag{2}$$

To calculate the DFT of the signal $x[n]$ in Python, you may use:

```python
import numpy as np
X = np.fft.fft(x)
```

If you want to explicitly specify the length M, then you can use:

```python
X = np.fft.fft(x, n=M)
```

Additionally, Python's `` `np.fft.fftshift` `` command swaps the first and second half of the vector X so that the frequency range is in $[-N/2, \ N/2]$ (assuming N is even).

2. Questions

Please write a Python script (saved as `` `fftsinc.py` ``) to implement problems (a) to (f).

Part I

Let $x(t)$ be a sinc function written as

$$x(t) = \frac{\sin(2\pi t)}{2\pi t}$$

Now, $x(t)$ is sampled at a rate $T_s = \frac{T}{N_1}$ so that $x[n] = x(nT_s), n \in \{-N_1, -N_1 + 1, \dots, 0, \dots, N_1 - 1, N_1\}$ and $N = 2N_1 + 1$.
Let $N = 1001$ and $T = 100$.

(a) (10%) Use the Python function `plot` from matplotlib to plot $x[n]$ vs n.
(b) (20%) Use the NumPy function `fft` directly to compute DFT of $x[n]$, and use the `plot` function to plot the magnitude of the `fft` output vs frequency ω. Center the zero frequency in your plot. Observe the *Gibbs phenomenon* in (b) and give some explanation for it in your report.
(c) (20%) Create a Python program by yourself to compute $X_k(e^{j\omega})$ of equation (1) and use the `plot` function to plot the magnitude of $X_k(e^{j\omega})$ vs frequency ω. You also need to rearrange $X_k(e^{j\omega})$ so that the zero frequency is centered in your plot. Verify whether the answer is the same as Problem (b).

Part II

A way of mitigating *Gibbs phenomenon* is to multiply $x(t)$ by a finite-duration signal $w(t)$, i.e., $y(t) = x(t)w(t)$. The signal $w(t)$ is called the window function. A famous one is the *Hanning* window, which is specifically written as:

$$w(t) = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi|t|}{T_w}\right)\right], |t| \leq \frac{T_w}{2} \\ 0, \qquad\qquad\qquad\qquad\quad, \text{esle} \end{cases}$$

Suppose $w(t)$ is also sampled at a rate $T_s = \frac{T}{N_1}$ so that $w[n] = w(nT_s), n \in \{-N_1, -N_1 + 1, \dots, 0, \dots, N_1 - 1, N_1\}, N = 2N_1 + 1$.

Let $N = 1001, T = 100$, and $T_w = \frac{T}{2}$.

(d) (15%) Use the Python function `plot` to plot $w[n]$ vs n.
(e) (15%) Use the Python function `plot` to plot $y[n]$ vs n, where $y[n] = x[n]w[n]$, and $x[n]$ is the signal plotted in (a).
(f) (20%) Use the NumPy function `fft` directly to compute DFT of $y[n]$ in (e), and use the `plot` function to plot the magnitude of the `fft` output vs frequency ω. The zero frequency should also be centered in your plot. Observe the Gibbs phenomenon here and give some explanation for comparison with (b) in your report.

Note: We expect that executing your `fftsinc.py` file will output 6 figures in order.

(One figure for each of Problems (a) to (f))

3. E3 Submission Instructions

- Please upload a **Python script** (saved as `` `fftsinc.py` ``) and a **report** (saved as `` `report.pdf` ``). Include the figures mentioned above in the report and provide explanations as needed. **Do not zip them into a compressed file!**