BELMONT UNIVERSITY, CURB COLLEGE OF ENTERTAINMENT AND MUSIC BUSINESS

AET 5410 Homework 4

Digital Audio, Computer Programming, Signal Analysis

Due: December 12, 2023

1 DISCRETE FOURIER TRANSFORM

An important method to analyze the spectral content of an audio signal is the Discrete Fourier Transform (DFT). This analysis projects an audio signal, $x[n]$, onto a set of complex exponential periodic signals. The following is an equation which can be used to calculate the DFT:

$$X[\omega_k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\omega_k t_n} \tag{1.1}$$

$$\text{for } k = 0, 1, 2, ..., N-1 \tag{1.2}$$

By expanding the complex exponential function, the DFT can be written:

$$X[\omega_k] = \sum_{n=0}^{N-1} x[n] \cdot (cos(\omega_k t_n) - j \cdot sin(\omega_k t_n)) \tag{1.3}$$

$$\text{for } \omega_k = k \cdot \frac{2\pi}{N} \cdot Fs \tag{1.4}$$

The result of the DFT calculation is a vector, $X(\omega_k)$ of length N, the same length as the input audio signal, $x[n]$. Each sample of $X[\omega_k]$ corresponds with a frequency, $f = \frac{k \cdot Fs}{N}$, in Hz.

The values of $X[\omega_k]$ will be complex numbers with a real ($Re\{\cdot\}$) and imaginary ($Im\{\cdot\}$) part. For each frequency bin, ω_k, the **amplitude** ($|X[\omega_k]|$) and **phase** ($\angle X[\omega_k]$) can be calculated using the following:

$$|X[\omega_k]| = \sqrt{Re\{X[\omega_k]\} + Im\{X[\omega_k]\}} \tag{1.5}$$

$$\angle X(\omega_k) = tan^{-1}\Big(\frac{Im\{X(\omega_k\}}{Re\{X(\omega_k\}}\Big) \tag{1.6}$$

Note: In MATLAB, the inverse tangent, tan^{-1}, can be calculated using the function **atan2**.

1.1 Problem

For this problem you will write a **script** (m-file) in MATLAB which calculates the DFT based on the following specifications:

- Name the script - discreteFourierTransform.m
- The goal is to calculate column vector variable representing $X[\omega_k]$.
- Initially, read in the file: **sawtoothwave.wav**
  - The input signal: column vector representing the signal to be analyzed $x[n]$.
  - Fs : sampling rate
- At the start of the function, determine the length of the input signal, N.
- Create a time vector, t, used for creating the periodic signals equal the length of the input signal.
  - Units of t should be in seconds
- Within a loop, create a matrix, p, to contain the set of analyzing periodic functions
  - Declare the loop for $k = 0 : N-1$
  - Create a variable $\omega = k * \frac{2*pi}{N} * Fs$
  - Each row of the matrix, p, should be a different periodic function based on k.
- After the loop, create the output variable, X, by calculating the DFT: $X = p \cdot x$.
- Finally, compare your result to the built-in MATLAB function: `fft`
  - Create a plot showing the the "null test" between your result and the MATLAB function

Remember to add comments to your code to explain what each command is accomplishing. For this problem you will submit the script and the sound file.

2 Inverse Discrete Fourier Transform

The Discrete Fourier Transform (DFT) has a complementary transform called the Inverse Discrete Fourier Transform (IDFT). This transform converts a complex-valued, frequency-domain signal, $X[k]$, into a real-valued, time-domain signal $x[n]$. The following is an equation which can be used to calculate the IDFT:

$$x[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot e^{+j\omega_k t_n} \tag{2.1}$$

$$\text{for } n = 1, 2, ..., N \tag{2.2}$$

By expanding the complex exponential function, the DFT can be written:

$$x[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot (cos(\omega_k t_n) + j \cdot sin(\omega_k t_n)) \tag{2.3}$$

$$\text{for } t_n = (n-1) * T_s \tag{2.4}$$

The result of the IDFT calculation is a vector, $x[n]$ of length N, the same length as the input signal, $X[k]$. Each sample of $X[k]$ corresponds with a frequency, $f = \frac{k \cdot Fs}{N}$, in Hz.

In our computation of the DFT, the array t is typically pre-allocated, `t = [0:N-1]*Ts`, before the loop. The scalar k is calculated once per loop. In the case of the IDFT, the scalar t is calculated once per loop, and the array ω_k is typically pre-allocated before the loop, `wk=k*2*pi*Fs/N` with $k = 0, 1, 2, ..., N-1$.

While working with the signal in the frequency domain, it is common to have the signal represented as **amplitude** ($|X[k]|$) and **phase** ($\angle X[k]$). Prior to performing the IDFT, the amplitude and phase must be converted into the complex-valued signal $X[k]$:

$$X_r[k] = |X[k]| * cos(\angle X[k]) \tag{2.5}$$

$$X_i[k] = |X[k]| * sin(\angle X[k]) \tag{2.6}$$

$$X[k] = X_r[k] + 1j * X_i[k] \tag{2.7}$$

2.1 Problem

For this problem you will write a **script** (m-file) in MATLAB which calculates the IDFT based on the following specifications:

- Name the script - `inverseDFT.m`
- The goal is to calculate column vector variable representing $x[n]$.
- Initially, read in the two files: **Xamp.wav** and **Xphase.wav**
  - Xamp.wav: amplitude in the frequency-domain

  - Xphase.wav : phase in the frequency-domain

- These signals have a length of $N = 4096$
- Create a frequency vector, ω_k, used for creating the periodic signals equal the length of the input signal.
  - `wk=k*2*pi*Fs/N`
- Create a complex-valued vector, $X[k]$, following Equations 1.5-1.7.
- Within a loop, create a matrix, p, to contain the set of analyzing periodic functions
  - Declare the loop for $n = 1 : N$
  - Create a scalar variable: `t = (n-1) * Ts`
  - You can choose to perform the IDFT using matrix or vector multiplication
  - Synthesize the periodic signals, p based on t and ω_k
- Create the output variable, x, by calculating the IDFT: $x = \frac{1}{N} \cdot p \cdot X$.
- In order to make sure the results is purely real-valued, perform the command: `x=real(x)`.
- Plot the time-domain signal, x
  - This should be a triangle wave with full-scale amplitude

Remember to add comments to your code to explain what each command is accomplishing. For this problem you will submit the script and the sound file.

3 Creating a Spectrogram

A spectrogram is a three-dimensional visualization of a signal. It displays a signal's amplitude versus time and frequency. It is a way to view how a signal spectrally varies over time (i.e. how frequency information changes over time).

One way to create a spectrogram uses the Discrete Fourier Transfrom. Specifically, the DFT is performed on short time frames, or buffers, of the signal. This is called the *Short-Time Fourier Transfrom* (STFT). If the DFT is performed on the entire signal, then the result contains the spectral information of the entire signal, not how it changes over time.

Previously, you calculated the DFT for a stationary signal (i.e. stays the same the entire duration). Therefore, it was not necessary to use the STFT.

For audio signals which are non-stationary (i.e. they do not stay the same the whole time), it is helpful to use a spectrogram to visualize how the signal changes.

3.1 PROBLEM: SPECTROGRAM SCRIPT

For this problem, you will write a script which calculates the Fourier Transform of a signal over short time frames. Each time frame should overlap by specified number of samples. Additionally, you will apply a windowing function to each frame as a way to cross-fade between frames. A sound file is provided which alternates between white noise, a 440 Hz sine wave, and a 500 Hz square wave. Create and save a **script** (m-file) in MATLAB that performs the following steps:

- Name the script: specGram.m
- Specify the following variables at the beginning of the script:
  - *x* : input signal to be analyzed (testSignal.wav)
  - *Fs* : sampling rate
  - *bufferSize* : number of samples to be included in each segment/window
  - *overlap* : number of samples for the overlap
  - *window* : array of the windowing function equal to the length of *bufferSize*
- At the start of the script, set up a loop to index segments of the input signal based on the *bufferSize* and *overlap*
- For each segment, apply the windowing function, then find the Fourier Transform
  - Use the built-in function *fft(x)* for speed
- Copy the result of the DFT into a matrix
  - Each row is used to store a different frequency bin.
  - Use one column for each frame
- Create a 3-D plot of the matrix using the function *surf(T,F,abs(S))*
  - Use a time vector, *T*, in seconds
  - Use a frequency vector, *F*, in Hz based on the DFT
  - Label the axes accordingly
  - For audio signals, the matrix will have redundant information from the Nyquist frequency to Fs. Therefore, it is only necessary to plot the frequencies from 0 to Nyquist.

I'd recommend experimenting with the MATLAB spectrogram function to get familiar with the plotted result. Remember to add comments to your code to explain what each command is accomplishing. For this problem you will submit the script and the sound file.

4 SUBMISSION

To submit your homework, create a single zip file that contains the MATLAB function you created and the provided test scripts. Additionally, include the test sound files in the folder. Name the zip file: xxxxx_AET5410_HW4.zip, where *xxxxx* is your last name. Email the zip file to: eric.tarr@belmont.edu by the start of exam on December 12th.