Signals and Systems project:
Signal Processing Project

Task 1 – Generating and Analyzing Sinusoidal Waves with Different Frequencies

A. Generate three sinusoidal signals with the following specifications:
   - Sine Wave 1: Frequency = 200 Hz, Amplitude = 1
   - Sine Wave 2: Frequency = 400 Hz, Amplitude = 0.5
   - Sine Wave 3: Frequency = 600 Hz, Amplitude = 0.2

   How do different frequencies and amplitudes affect the shape of the waveform?

B. Playback of Sinusoids:
   Use MATLAB to convert each sinusoid into audible sound using sound().
   Describe the perceptual (auditory) differences between the three sine waves.

C. Plotting in Time Domain:
   Plot all three sine waves on a single graph for visual comparison.
   How does the time-domain representation reflect differences in frequency and amplitude?

D. Fourier Series Representation:
   Use Fourier series to represent and visualize the frequency content of each signal. (Note: You have to find the coefficients first)
   How do the Fourier series plots compare to the original sine waveforms?

Task 2 – Sampling Theory and Frequency Analysis of a Music File

A. Load the provided music file in MATLAB and play it using sound().
Describe the audio quality. Is the sound clear and continuous?

B. Theoretical Sampling Analysis:
The music is sampled at 44.1 kHz (CD quality).
Explain why 44.1 kHz is a standard sampling rate for audio. What is its relationship to the Nyquist Theorem?

C. Plot Waveform in Time Domain:
Plot the waveform of the audio signal in the time domain.
How does the time-domain waveform reflect properties of the audio signal (e.g., loudness, pitch)?

D. Fourier Transform and Plot:
Apply the Fourier Transform to the music file and plot the frequency spectrum.
What insights about the music's frequency content does the Fourier analysis provide? Compare this with the time-domain view.

E. Effects of Different Sampling Rates:
Downsample the audio to 22.05 kHz and 11.025 kHz, and observe changes.
How does reducing the sampling rate affect the audio quality and Fourier analysis results?

💡You can explore an audio file by multiplying it with a exponential function. See what happens!

Task 3 – Adding Noise to Specific Frequencies

A. Add noise to the music file at specific frequencies, such as 500 Hz and 1000 Hz.
How do these specific noise frequencies affect the audio?

B. Fourier Analysis of Noisy Audio:
Apply the Fourier Transform to the noisy audio for visualization.
Can you identify the added noise frequencies in the frequency spectrum?

C. Filter Design:
Design a filter (e.g., notch filter) based on the Fourier analysis to remove the added noise.
Which type of filter did you use, and why is it suitable for this task?

🔥 Read more about filters here:

https://vru.vibrationresearch.com/lesson/filtering/

D. Apply Filter and Listen Again:
Apply the designed filter to the noisy audio. Use MATLAB to play the filtered audio.
Compare the filtered audio to the original and noisy versions. How effective was your filter in improving sound quality?

E. Frequency Spectrum of Filtered Audio:
Perform Fourier Transform on the filtered audio.
How did the frequency spectrum change after applying the filter?

Task 4 – Spectrogram and STFT Analysis

Overview of spectrogram() Function:
The spectrogram function in MATLAB computes the Short-Time Fourier Transform (STFT) of a signal. It shows how a signal's frequency content evolves over time. STFT is useful for analyzing time-varying signals like audio and radio waves.

Windowing:
What is a window?
A window breaks the long signal into shorter, manageable segments for frequency analysis. Since frequency content can change over time, we analyze smaller pieces one at a time.

- A shorter window gives better time resolution, but worse frequency resolution.
- A longer window gives better frequency resolution, but worse time resolution.

Overlap (Noverlap):
Noverlap is the number of samples that overlap between consecutive windows.

- Higher overlap → smoother spectrogram with better time continuity.
- Lower overlap → less computation, but rougher time-frequency view.

A. Use MATLAB's spectrogram() function on the provided music file. Use default settings for window size, Noverlap, and NFFT to keep it simple.
   Analyze the spectrogram and describe any observable changes in the frequency content over time. Can you identify sections where high or low frequency content suddenly changes? Do these correspond to different instruments or melody changes?

Task 5 – Generating and Analyzing a Sawtooth Wave

Generate a sawtooth wave, another fundamental periodic signal with a rich harmonic structure different from a sine or square wave.

A. Generate a 1-second sawtooth wave with a fundamental frequency of 440 Hz (the musical note 'A').

B. Listen to the wave and compare its timbre (sound quality) to that of a pure sine wave.

C. Apply the Fourier Transform and plot the frequency spectrum. A sawtooth wave contains both even and odd harmonics. Identify the first few harmonics in your plot.

D. **Analysis**: How does the presence of all integer harmonics in the sawtooth wave's spectrum explain its "buzzy" or "bright" sound compared to a sine wave?

Task 6 – Designing a Two-Way Audio Crossover

Design a simple two-way audio crossover system. A crossover uses filters to split an audio signal into different frequency bands to send to different speakers (e.g., a woofer for low frequencies and a tweeter for high frequencies).

A. Filter Design: Choose a crossover frequency (e.g., 2 kHz). Design two filters:
    a. A low-pass filter that passes frequencies below the crossover point.
    b. A high-pass filter that passes frequencies above the crossover point.

B. Apply both filters to the original music file to create two new audio signals: a "low frequency" signal and a "high frequency" signal.

C. **Analysis:** Listen to each of the two output signals. Does the low-frequency signal contain mostly bass and the high-frequency signal contain mostly cymbals and "sizzle"? Plot the frequency response of both filters on the same graph to visualize the crossover point.

🔑 The whole point of the crossover system you've designed is that it's very difficult for a single speaker to accurately reproduce the entire spectrum of audible sound. By splitting the signal, you send the right job to the right tool: the bass goes to the big woofer, and the treble goes to the small, fast-moving tweeter. This results in a much clearer and more balanced sound.

🛑 Resources on the filters mentioned:

https://en.wikipedia.org/wiki/Audio_crossover

https://www.psaudio.com

Task 7 – Verifying the Convolution Theorem

Numerically verify the Convolution Theorem, which states that convolution in the time domain is equivalent to multiplication in the frequency domain. This is a cornerstone of digital signal processing.

A. Time-Domain Convolution:
   a. Load the music file (or a short segment of it).
   b. Create a simple filter impulse response, h (e.g., a simple averaging filter like h = [0.25, 0.25, 0.25, 0.25]).
   c. Perform the convolution directly using y_conv = conv(y, h).

B. Frequency-Domain Multiplication:
   a. Compute the FFT of the music signal (Y = fft(y)) and the impulse response (H = fft(h)). Make sure both inputs to the FFT have the same length by zero-padding the shorter one.
   b. Multiply the two spectra element-by-element: G = Y. * H.
   c. Compute the inverse FFT of the result: g = ifft(G).

C. Analysis: Plot y_conv and g on the same graph. Are they identical (within a small numerical error)? Explain why this theorem is so important for applying filters efficiently to long signals.

About submission:

✓ File's naming format:
   ❖ For MATLAB Project: "S&S_PRJ_StudentNo." E.g.: PRJ_401613198.rar
   ❖ Make sure to include all your m-files in the rar file.

✓ Your submission should include a report file and the rar file.

✓ Late submissions will result in a 10% daily reduction from your Project mark.

✓ Clean codes will receive a slight extra mark.