Digital Signal Processing Lab (ECL333)

Experiment 3: Echo Cancellation and DFT Properties Verification

Department of Electronics and Communication Engineering

Faculty: Jinu Jayachandran, Viji R

Student Name: _____ **Roll Number:** _____

Instructions

1. Complete the Pre-lab work before the lab session.
2. Write all answers neatly within the space provided. Do not attach additional sheets.
3. Submit your lab sheet before leaving the lab. Late submissions receive zero marks.
4. Ensure to import Python libraries 'numpy', 'matplotlib.pyplot', 'scipy.io.wavfile', 'scipy.signal', and 'pandas'.
5. The post-lab questions are indicated by **PTQ**.

Objective

- To implement an echo cancellation system using autocorrelation for delay estimation and projection for attenuation estimation.
- To verify fundamental properties of the Discrete Fourier Transform (DFT): Time Reversal, Circular Time Shift, Frequency Shift, and Circular Convolution.
- To understand the practical implications of these properties in signal processing.

1 Theoretical Background

1.1 Echo Model and Cancellation

An echo in a signal can be modeled as a delayed and attenuated version of the original signal added to itself. If $x[n]$ is the original signal, the observed signal $y[n]$ with a single echo can be expressed as:

$$y[n] = x[n] + \alpha x[n - D]$$

where α is the attenuation factor ($0 < \alpha < 1$) and D is the delay in samples.

The receiver receives only $y[n]$ and it doesn't know the values of attenuation factor (α), delay (D) and transmitted signal ($x[n]$). So by looking at $y[n]$ the receiver should estimate α, D and $x[n]$.

Autocorrelation for Delay Estimation

The autocorrelation function of a signal $y[n]$ is given by

$$R_{yy}[m] = \sum_{n} y[n]y[n-m]$$

. For an echo signal, the autocorrelation will exhibit a peak at $m = 0$ and a secondary peak at $m = D$ (and $m = -D$), which can be used to estimate the delay D.

Properties of Autocorrelation

Autocorrelation possesses several important properties:

1. **Symmetry:** The autocorrelation function is symmetric about the origin:

   $$R_{xx}(\tau) = R_{xx}(-\tau) \quad \text{or} \quad R_{xx}[m] = R_{xx}[-m]$$

   This means we only need to compute it for non-negative lags.

2. **Maximum at Zero Lag:** The autocorrelation function is maximum at zero lag ($\tau = 0$ or $m = 0$):

   $$|R_{xx}(\tau)| \leq R_{xx}(0) \quad \text{or} \quad |R_{xx}[m]| \leq R_{xx}[0]$$

   This is intuitive because a signal is perfectly correlated with itself when there is no shift. $R_{xx}(0)$ represents the average power of the signal.

3. **Fourier Transform Pair:** The autocorrelation function and the Power Spectral Density (PSD) form a Fourier Transform pair (Wiener-Khinchin Theorem):

   $$R_{xx}(\tau) \leftrightarrow \mathcal{F}S_{xx}(\omega)$$

   $$R_{xx}[m] \leftrightarrow \text{DFT}S_{xx}(k)$$

   where $S_{xx}(\omega)$ or $S_{xx}(k)$ is the Power Spectral Density. This property highlights the direct relationship between the time-domain correlation and the frequency-domain power distribution.

4. **Even Function:** For real signals, $R_{xx}(\tau)$ and $R_{xx}[m]$ are even functions.

5. **Periodic Signals:** If a signal $x(t)$ is periodic with period T_0, then its autocorrelation $R_{xx}(\tau)$ will also be periodic with the same period T_0. Similarly for discrete-time signals. This is crucial for detecting periodicities.

Interpretation of Autocorrelation Peaks

The shape and peaks of the autocorrelation function provide valuable information about the signal:

- **Peak at $\tau = 0$ (or $m = 0$):** Always the global maximum, representing the signal's power.
- **Secondary Peaks:**

  - If a signal contains a strong periodic component (e.g., a sine wave or a human voice producing a sustained vowel), the autocorrelation function will show strong peaks at lags corresponding to multiples of the signal's period. The first significant peak (after the zero-lag peak) indicates the fundamental period of the signal.
  - For signals with echoes, the autocorrelation will show a strong secondary peak at a lag equal to the echo delay.

- **Decay Rate:** How quickly the autocorrelation decays from its peak at zero lag gives an indication of how fast the signal loses its "memory" or how quickly its values become uncorrelated over time. For example, white noise has an autocorrelation that is zero for all non-zero lags (a single peak at zero), indicating no correlation between samples.

Projection for Attenuation Estimation

Once the delay D is known, the attenuation factor α can be estimated by projecting the delayed signal onto the original. If we assume $y[n] \approx x[n] + \alpha x[n-D]$, and we want to find α such that $y[n] - \alpha x[n-D]$ is minimized (ideally $x[n]$), we can derive α using a least squares approach:

$$\hat{\alpha} = \frac{\sum_n y[n]x[n-D]}{\sum_n y^2[n-D]}$$

In practice, since $x[n]$ is unknown, we approximate $x[n]$ with $y[n]$ for the projection, leading to:

$$\hat{\alpha} \approx \frac{\sum_n y[n]y[n-D]}{\sum_n y^2[n-D]} \tag{1}$$

Once α and D are estimated, the original signal $\hat{x}[n]$ can be recovered by subtracting the estimated echo:

$$\hat{x}[n] = y[n] - \hat{\alpha} y[n - \hat{D}]$$

1.2 DFT Properties

For a discrete-time signal $x[n]$ of length N with DFT $X[k]$, the following properties hold:

1. Time Reversal Property

If $x[n] \xrightarrow{\text{DFT}} X[k]$, then $x[(-n)_N] \xrightarrow{\text{DFT}} X[(-k)_N]$. Here, $(-n)_N$ denotes $N - n$ for $n \neq 0$ and 0 for $n = 0$ (circular reversal).

2. Circular Time Shift Property

If $x[n] \xrightarrow{\text{DFT}} X[k]$, then $x[(n - n_0)_N] \xrightarrow{\text{DFT}} X[k]e^{-j2\pi k n_0/N}$. A circular shift in the time domain results in a linear phase shift in the frequency domain.

3. Frequency Shift Property

If $x[n] \xrightarrow{\text{DFT}} X[k]$, then $x[n]e^{j2\pi k_0 n/N} \xrightarrow{\text{DFT}} X[(k - k_0)_N]$. Multiplication by a complex exponential in the time domain results in a circular shift in the frequency domain.

4. Circular Convolution Property

If $x[n] \xrightarrow{\text{DFT}} X[k]$ and $h[n] \xrightarrow{\text{DFT}} H[k]$, then the circular convolution $y[n] = x[n] \circledast h[n]$ is given by:

$$y[n] \xrightarrow{\text{DFT}} Y[k] = X[k]H[k]$$

And conversely, $x[n] \circledast h[n] = \text{IDFT}(X[k]H[k])$.

Concept: Circular convolution can be expressed as a matrix-vector multiplication. One of the sequences is arranged into a special matrix called a **circulant matrix**, which then multiplies the other sequence (represented as a column vector).

Constructing the Circulant Matrix: Let's say we want to compute $y[n] = x[n] \circledast h[n]$. We can form an $N \times N$ circulant matrix $\mathbf{H}$ from the sequence $h[n]$:

$$\mathbf{H} = \begin{pmatrix} h[0] & h[N-1] & h[N-2] & \dots & h[1] \\ h[1] & h[0] & h[N-1] & \dots & h[2] \\ h[2] & h[1] & h[0] & \dots & h[3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \dots & h[0] \end{pmatrix}$$

- The first row is $h[0], h[N-1], h[N-2], \dots, h[1]$.
- Each subsequent row is a circular right shift of the row above it.

The circular convolution $y[n]$ is then obtained by multiplying this circulant matrix $\mathbf{H}$ by the column vector $\mathbf{x}$ (formed from $x[n]$): $\mathbf{y} = \mathbf{Hx}$ where $\mathbf{y} = [y[0], y[1], \dots, y[N-1]]^T$ and $\mathbf{x} = [x[0], x[1], \dots, x[N-1]]^T$.

Pre-lab Work

Q1. If $x[n]$ is a real-valued signal, what is the relationship between $X[k]$ and $X[N-k]$ for its DFT $X[k]$? How does this relate to the time reversal property? [2 Marks]

Q2. For a signal $x[n]$ of length N, if its DFT is $X[k]$, what is the DFT of $x[(n-3)_N]$? Express your answer in terms of $X[k]$. [1 Marks]

Q3. If $X[k]$ is the DFT of $x[n]$, what is the DFT of $x[n] \cdot e^{j2\pi \cdot 2n/N}$? [2 Marks]

Q4. Write a python function `circular_convolution(x, h)` which accepts two arguments `x` and `h`, and returns the circular convolution of the arguments? [5 Marks]

2 Procedure

In this experiment, you will first implement an echo cancellation system and then verify several fundamental DFT properties.

2.1 Load the audio file

1. You have been provided with a '.wav' audio file. This file contains an white noise audio sample with an echo. Since the audio sample of very short duration you will not be able hear the audio. The following code will help you in loading the file.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.signal import correlate

# Read the wav file
rate, y_in = wavfile.read("echo_signal_random.wav")
y_in = y_in.astype(float) / 32767 # Convert to float for processing.
```

PTQ1: The maximum value of the audio sample present in the file is _____

PTQ2: The minimum value of the audio sample present in the file is _____

PTQ3: The total length of the audio sample is _____

2.2 Echo Delay using Autocorrelation:

Here, we are going to compute the autocorrelation of the received signal `y_in` and find the location of second peak.

1. **Compute Autocorrelation:** You can use the `scipy` function `correlate()` to find the autocorrelation.

```
# Recompute autocorrelation
autocorr = correlate(y_in, y_in, mode='full')
lags = np.arange(-len(y_in)+1, len(y_in))
# Plot autocorrelation with lags in x-axis and autocorr in y-axis
```

PTQ4: The total length of `autocorr` is _____

PTQ5: Express the length of `autocorr` in terms of length of `y_in`.

PTQ6: The location of the highest peak in the plot is _____

PTQ7: Draw the autocorrelation plot here clearly marking the axis values corresponding to the first two peaks.

2. **Estimate the Delay:** To compute the delay we are going to find the location of second peak along the positive x-axis.

```
# Find the peak in the positive lags side (excluding lag 0)

d_delay = # The delay is the value of lags at the location of second peak
```

PTQ8: The location of the second peak is at `lags=`_____ and the value of estimated delay `d_delay=` _____

2.3 Estimation of Attenuation:

We are going to implement equation 1 to estimate the attenuation. The numerator of the equation is $\sum_n y[n]y[n-D]$ which is the dot product of $y[n]$ and $y[n-D]$ (compare equation 1 with equation of dot product). We need to first generate $y[n-D]$, which is the D delayed version of $y[n]$ before finding the dot product. Similarly, the denominator of the equation is dot product of $y[n-D]$ with itself.

1. Let us define a function `estimate_attenuation` to find the estimate. To create the delay $y[n-D]$ we use the function `np.roll(signal, num_delay)` which delays the `signal` by `num_delay` number of samples. The dot product is calculated using `np.dot()`

```
# Function to estimate attenuation using projection
def estimate_attenuation(y_signal, d_delay):

        y_delayed = # Use np.roll to produce d_delay for y_signal
        numerator = # Dot product of y_signal and y_delayed
        denominator = # Dot product of y_delayed and y_delayed
        if denominator == 0:
        return 0
        return numerator / denominator

# Estimate attenuation
estimated_alpha = estimate_attenuation(y_in, estimated_delay)
# Print the estimated delay
```

PTQ9: The estimated α is _____

2.4 Echo Cancellation

To cancel the echo we recreate the echo at the receiver using the estimated delay and estimated attenuation and then subtract this echo from the received signal.

```
# Perform echo removal
estimated_echo = # Use np.roll to get the delayed version of y_in which is
    delayed by d_delay
 estimated_echo = # Multiply the delayed version of y_in by the
     estimated_alpha

x_hat = # Subtract the original received signal (y_in) and estimated_echo to
    recover the signal

# Plot original, echoed, and recovered signals in a single figure
```

PTQ10: Draw the plot comparing original, echoed, and recovered signals in a single figure.

2.5 Part 2: DFT Properties Verification

In this part, you will verify the properties of DFT using the original signal `x` generated in Part 1. For taking mode you can use the `numpy` function `np.mod()`.

1. **Time Reversal Property:** Verify that $x[(-n)_N] \xrightarrow{\text{DFT}} X[(-k)_N] = X[N-k]$.

```
N = len(y_in)
y_in_rev = # Time reverse 'y_in'. When you do the reversal keep the
    sample at n=0 as it is and reverse the others, i.e swap y_in[1] and
    y_in[N-1], swap y_in[2] and y_in[N-2] and so on .

Y = # Compute DFT of original signal y_in
```

```
Y_N_refl_shift = # Find out Y[N-k] or Y[-k (mod N)] for each value of k.
Y_rev = # Compute DFT of time reversed signal y_in_rev
verifcation_err = # Find the sum of square of abs(Y_N_refl_shift-y_in_rev
    )

# print verifcation_err
```

PT_Q11: The value of the `verifcation_err`=_____.

PT_Q12: The minimum value of `N` we should take for DFT is _____.

2. **Circular Time Shift Property:** Verify that $x[(n-n_0)_N] \xrightarrow{\text{DFT}} X[k]e^{-j2\pi kn_0/N}$.

```
# 1. Circular Time Shift of x[n]
y_in_shifted = # Delay 'y_in' by 12. You can use np.roll()
Y_shifted_dft = #Compute DFT of y_in_shifted
Y_expected_shift_dft = #Multiply Y with exp(-1j * 2 * np.pi * k * 12 / N)
     where k is from 0 to N-1.
verifcation_err = # Find the sum of square of abs(Y_expected_shift_dft-
    Y_shifted_dft)
```

PT_Q13: The value of the `verifcation_err`=_____.

3. **Frequency Shift Property:** Verify that $x[n]e^{j2\pi k_0 n/N} \xrightarrow{\text{DFT}} X[(k-k_0)_N]$.

```
k0 = #Set the frequency shift to 3
y_in_freq_shifted = #y_in multiplied by np.exp(1j * 2 * np.pi * k0 * n /
    N) where n is from 0 to N-1.
Y_freq_shifted = # Find DFT of y_in_freq_shifted
Y_expected_freq_shift = # Delay 'Y' by k0. Use np.roll()'
verifcation_err = # Find the sum of square of abs(Y_freq_shifted-
    Y_expected_freq_shift
)
print(verifcation_err)
```

PT_Q14: The value of the `verifcation_err`=_____.

4. **Circular Convolution Property:** Verify that $x[n] \circledast h[n] = \text{IDFT}(X[k]H[k])$. Refer Section 1.2, point 4 for explanation on circular convolution. You should use the function `circular_convolution(x, h)` that you have written in pre-lab.

   Here we are going to generate a random signal `h` of length `N` to circularly convolute with `y_in`. Then we find the DFTs of `h` and `y_in`. The IDFT of the product of DFT(`h`) and DFT(`y_in`) should be equal to the circular convolution of `y_in` and `h`.

```
h = # Generate N Gaussian distributed random samples. Use randn from
    numpy.
y_circ = # Find IDFT(DFT(y_in)*DFT(h))
y_circ_direct = circular_convolution(y_in,h)
```

```
verifcation_err = # Find the sum of square of abs(y_circ - y_circ_direct)
print(verifcation_err)
```

PT_Q15: The value of the `verifcation_err`=_____.

Work Done During Lab

(Record your major observations, plots, and comparisons here.)

Analysis and Inference

(Discuss the accuracy and limitations of the echo cancellation method. Analyze the verification results for each DFT property, explaining any observed numerical errors. Discuss the significance of these properties in practical DSP applications.)

Code Attachment

(Attach your final Python code here.)

Student Signature: _____ **Date:** _____