Corrections series 2

Exercise 1

1. What is the primary purpose of using a filter in signal processing?
A) To remove noise and unwanted components
B) To increase signal amplitude
C) To convert an analog signal to a digital signal
D) To change the signal's sampling rate

2 How does a high-pass filter differ from a low-pass filter?
A) A high-pass filter removes high frequencies
B) A low-pass filter removes low frequencies
C) A high-pass filter allows high frequencies to pass through
D) Both (B) and (C)

3 What type of filter is best for removing low-frequency baseline wander in ECG signals?
A) Notch Filter
B) High-pass Filter
C) Low-pass Filter
D) Band-stop Filter

4. What is the main difference between FIR and IIR filters?
A) FIR filters have feedback, IIR filters do not
B) IIR filters have feedback, FIR filters do not
C) FIR filters use recursion, IIR filters do not
D) FIR filters require fewer coefficients than IIR filters

5. What is an advantage of FIR filters over IIR filters?
A) Always more efficient than IIR filters
B) Always require fewer computations
C) Always have a linear phase response
D) Can approximate analog filters exactly

6. How do you determine the order of a digital filter?
A) By the highest power of the denominator polynomial
B) By the number of taps in an FIR filter
C) By the number of poles and zeros in the transfer function
D) All of the above

7. How does the wavelet transform differ from the Fourier transform?
A) Wavelet transform provides both time and frequency information
B) Fourier transform provides time information only
C) Wavelet transform does not decompose signals into frequencies
D) Both are the same

8. What is a major application of wavelet transform?
A) Image compression
B) DC voltage regulation
C) Digital modulation
D) Time-domain convolution

9. Why is multi-resolution analysis useful in wavelet transform?
A) It allows for zooming into different frequency components
B) It improves the speed of the Fourier Transform
C) It enhances the power spectrum
D) It only works for stationary signals

10. What is the delta function $\delta[n]$ in discrete-time signals?
A) A unit step function
B) A function that is 1 for n=0 and 0 otherwise

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

C) A function used only in Fourier analysis
D) A function that is 1 for all n

11. How is the delta transform used in signal processing?
A) To transform discrete signals into the frequency domain
B) To simplify difference equations
C) To represent impulse responses
D) All of the above

12. What is the region of convergence (ROC) in Z-transform analysis?
A) The region where the Fourier series converges
B) The set of values for which the Z-transform is finite
C) The area in a Fourier transform plot
D) The phase response of a signal

13. What is a notch filter commonly used for?
A) Removing high-frequency noise
B) Removing a specific frequency band
C) Amplifying low frequencies
D) Removing low-frequency noise

14. What type of noise does a **60 Hz Notch Filter** remove from an EEG signal?
A) Muscle artifact noise
B) Power line interference
C) Electrode movement noise
D) High-frequency noise

15. How does a notch filter affect the frequency response of a signal?
A) It amplifies all frequencies equally
B) It removes a specific frequency while preserving others
C) It increases the overall power of the signal
D) It applies a low-pass filtering effect

16. What does the Fourier Transform do?
A) Converts a signal into the time domain
B) Converts a time-domain signal into its frequency components
C) Increases the amplitude of a signal
D) Reduces the length of a signal

17. What is the difference between the Fourier Transform and the Laplace Transform?
A) Fourier Transform is used only for continuous signals
B) Laplace Transform includes complex frequency analysis
C) Fourier Transform cannot be applied to biomedical signals
D) There is no difference

18. How does the DTFT differ from the DFT?
A) DTFT is continuous, DFT is discrete
B) DTFT is computed only for finite-length signals
C) DTFT is periodic in time
D) DFT and DTFT are identical

19. How does the sampling frequency affect the DTFT spectrum?
A) Higher sampling frequency reduces aliasing
B) Lower sampling frequency improves resolution
C) Sampling frequency has no effect on DTFT
D) DTFT works only at Nyquist rate

20. What is the advantage of using FFT instead of DFT?
A) FFT is faster and computationally efficient
B) FFT provides more accurate results
C) FFT requires fewer memory resources
D) Both (A) and (C)

21. How is the DFT matrix constructed?
A) Using exponential functions
B) Using sine and cosine functions
C) Using convolution operations
D) By applying Laplace Transform

22. What are the two main properties of a Linear Time-Invariant (LTI) system?
A) Linearity and Stability
B) Time-invariance and Causality
C) Linearity and Time-invariance
D) Causality and Boundedness

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

23. How can convolution be used to analyze LTI systems?
A) It determines the system's frequency response
B) It finds the system's impulse response
C) It calculates the system's stability
D) It is not used for LTI analysis

24. Why is the impulse response important in LTI systems?
A) It determines how the system behaves for all inputs
B) It is only useful for continuous-time signals
C) It cannot be used for FIR filters
D) It is only used in electrical circuits

Exercise2 Quiz Questions:

1. The primary purpose is to **remove unwanted noise or frequency components** while preserving the desired signal.

2. A **high-pass filter** allows **high frequencies** to pass and attenuates low frequencies.

A **low-pass filter** allows **low frequencies** to pass and attenuates high frequencies.

3. **Low-pass filter** (LPF), **High-pass filter** (HPF), **Band-pass filter** (BPF), **Band-stop filter** (Notch filter), **All-pass filter**

4.

Feature	FIR	IIR
Feedback	No	Yes
Stability	Always stable	Can be unstable
Phase Response	Linear (predictable)	Nonlinear (distorted)
Order	Higher order required	Lower order sufficient

5. FIR filters have a **linear phase response**, meaning they do not introduce phase distortion, which is crucial for some applications like image and audio processing.
6. The order of a filter is determined by:

The number of **past input (FIR) or past output (IIR) terms** in the difference equation.

The **steepness** of the transition band (higher order = sharper cutoff).

Design criteria like **stopband attenuation and passband ripple**.

7. The key differences between the **Wavelet Transform (WT)** and the **Fourier Transform (FT)** are:

Feature	**Fourier Transform (FT)**	**Wavelet Transform (WT)**
Representation	Uses **sinusoids** (global basis functions)	Uses **wavelets** (localized basis functions)

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Feature	Fourier Transform (FT)	Wavelet Transform (WT)
Time-Frequency Resolution	Provides **only frequency information**, no time localization	Provides **both time and frequency** information
Best for	Analyzing **stationary signals** (where frequency content does not change over time)	Analyzing **non-stationary signals** (where frequency content changes over time)
Drawback	Loses **time-domain information**	Requires **more computation**

- **FT** is useful for signals with **consistent frequency content**, such as pure tones.
- **WT** is better for analyzing **transients, sharp changes, and irregular patterns**, making it ideal for real-world signals like EEG and ECG.

8. Wavelet Transform is widely used in **biomedical signal processing** due to its ability to analyze signals with time-varying frequency content. Some key applications include:

ECG (Electrocardiogram) Analysis

- Detecting **R-peaks** and **arrhythmias**
- Removing **baseline wander and noise**

EEG (Electroencephalogram) Processing

- Identifying **epileptic seizures**
- Detecting **brain wave abnormalities**

EMG (Electromyogram) Analysis

- Identifying **muscle fatigue**
- Detecting **neuromuscular disorders**

Medical Imaging

- **Wavelet-based image compression** (JPEG 2000)
- Noise reduction in **MRI & CT scans**

9. **Multi-resolution analysis (MRA)** is one of the biggest advantages of the Wavelet Transform. It allows us to analyze a signal at **different levels of detail** by decomposing it into:

- **Low-frequency components** (approximation coefficients) → Capture **global trends**
- **High-frequency components** (detail coefficients) → Capture **sharp changes**

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Key Advantages of Multi-Resolution Analysis:

Better Feature Extraction

- Helps detect both **slow and fast** changes in the signal.

Improved Noise Removal

- Can selectively remove unwanted noise while keeping important features.

Efficient Data Compression

- Reduces data size without losing important information (used in **JPEG 2000**).

Example in ECG Processing:

- Low-frequency wavelets capture **baseline drift** (slow changes).
- High-frequency wavelets detect **QRS complexes and arrhythmias** (fast changes).

10. The **unit impulse function**, denoted as δ[n], is a discrete-time signal that is **zero** for all values of n except for n=0 **1**. Mathematically:

$$\delta[n] = \begin{cases} 1 \; if \; n = 0 \\ 0 \; otherwise \end{cases}$$

It is often used in signal processing to model sudden or instantaneous changes and is the discrete equivalent of the **Dirac delta function** in continuous time.

11. The **Delta Transform** is closely related to the **Z-Transform** and is used to analyze signals by converting them from the time domain to the frequency domain. It helps represent discrete-time signals in terms of their **impulse responses** and allows for the analysis of system behaviors (such as stability and frequency response) by examining their **Z-domain representations**.

12. The **Delta Transform** (also known as the Z-Transform) and the **Fourier Transform** are both used for signal analysis, but they differ in the following ways:

- **Delta Transform (Z-Transform)**:
  The Z-Transform is a **generalized form** of the Fourier Transform that uses a complex variable $z = re^{j\omega}$ It is used to analyze discrete-time signals with respect to both time and frequency, and can handle **non-periodic signals** as well as **causal and stable systems**.
- **Fourier Transform**:
  The **Fourier Transform** is a **special case** of the Z-Transform with $= re^{j\omega}$, and it is used primarily to analyze **periodic signals** in terms of their frequency content. It only works for signals that are **periodic** and can be applied in both continuous-time and discrete-time domains.

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

In summary:

- Z-Transform: **More general, handles all types of signals**.
- Fourier Transform: **Special case of Z-Transform, focused on periodic signals**.

13. The **DTFT (Discrete-Time Fourier Transform)** and the **DFT (Discrete Fourier Transform)** are related in that both transform a discrete-time signal from the time domain to the frequency domain. The key differences between the two are:

- **DTFT**: The DTFT of a discrete-time signal is continuous and periodic in the frequency domain. It provides a frequency spectrum for an infinite sequence (or a long sequence, if we consider a periodic extension), and the result is a continuous function of frequency.
- **DFT**: The DFT is a **discrete approximation** of the DTFT, used to compute the frequency spectrum for a finite sequence. It is a sampled version of the DTFT, where the signal is discretized in both time and frequency. The DFT is periodic, and the frequency resolution is limited by the length of the signal.

In short, **the DFT is a sampled version of the DTFT**. When the length of the signal increases and the sampling rate is high, the DFT approaches the continuous DTFT.

14. The **sampling frequency** affects the DTFT spectrum by determining the **frequency resolution** of the transformed signal. When sampling a continuous-time signal, the DTFT captures the frequency content over a continuous range of frequencies. The sampling frequency dictates how finely this spectrum is "sampled."

- **Higher sampling frequency**: Results in a more finely sampled DTFT spectrum, allowing for better frequency resolution and a more detailed representation of the signal's frequency components.
- **Lower sampling frequency**: Reduces the frequency resolution and can lead to **aliasing**, where high-frequency components of the signal fold back into the lower frequency range, distorting the DTFT.

In summary, the sampling frequency impacts the spacing between the frequency components in the DTFT spectrum, and it must be sufficiently high to avoid aliasing and capture all relevant frequency components.

15. The **periodicity** of the DTFT is one of its fundamental properties. Specifically, the DTFT of any discrete-time signal is periodic with a period of 2π. This means that after every 2π radians, the frequency content repeats itself.

This periodicity arises because the underlying discrete-time signal is periodic in nature (due to sampling), and thus, its frequency spectrum must also repeat at regular intervals. The periodicity of the DTFT allows us to analyze the signal over a single interval (usually from $-\pi$-\pi$-\pi$ to π\pi$\pi$) and understand its frequency content fully without needing to analyze the entire spectrum.

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

16. A **Notch Filter** is primarily used to remove specific narrow-band interference in biomedical signal processing, such as **power line noise** (50/60 Hz), **electromyographic (EMG) artifacts**, or any other frequency components that can distort the physiological signal. In EEG and ECG, the Notch Filter removes unwanted interference without significantly affecting the signal of interest.

17. A **Notch Filter** is designed to **attenuate** a narrow range of frequencies around a central notch frequency, leaving the other frequencies largely unaffected. It works by creating a **deep dip** in the frequency response at the specified frequency (e.g., 60 Hz) and **passing** all other frequencies. This characteristic allows it to effectively remove noise at a specific frequency while preserving the rest of the signal.

18. Notch Filters have several practical applications, including:

- **Power line interference removal** in biomedical signals, such as EEG, ECG, and EMG.
- **Audio processing** to remove hum noise at specific frequencies (e.g., 50 Hz or 60 Hz).
- **Communication systems**, where specific interference frequencies need to be filtered out.
- **Signal processing in electrical systems** to remove undesirable noise or harmonics at specific frequencies.

19. **Fourier Transform** is typically used to analyze **periodic** signals and represents a signal as a sum of **sinusoidal components** with specific frequencies. It works only for signals that are stable and decay to zero as $t \to \infty$ (i.e., they are absolutely integrable). The Fourier Transform maps a signal from the time domain to the frequency domain, providing information about the signal's frequency content.

- **Laplace Transform** is more general and can be used for a broader range of signals, including those that are **non-periodic** and have transient behaviors (e.g., decaying or growing exponentially). It works in both the **s-domain** (complex frequency) and can describe signals that are not absolutely integrable. The Laplace Transform provides both **magnitude** and **phase** information of the signal, and it is particularly useful for analyzing **systems** and their stability.

20. The **Fourier Transform** represents a signal as a **sum of sinusoidal waves** (sines and cosines), each corresponding to a particular frequency. The result is a spectrum where the **frequency** (or angular frequency) corresponds to the components of the signal, and the **amplitude** of each frequency component represents how much of that frequency is present in the original signal. This transforms a time-domain signal into a frequency-domain signal, showing the distribution of energy or signal strength at different frequencies.

In other words, it breaks down the signal into its constituent sinusoidal components, allowing us to see how much of each frequency is contributing to the overall signal. This is particularly useful for analyzing periodic behavior and identifying frequency components like noise or harmonics.

21. **Magnitude Spectrum**: This tells you how much of each frequency is present in the signal. It provides information about the **strength** or **amplitude** of the frequency components. The magnitude spectrum is often used to understand the power distribution across frequencies in a signal.

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Phase Spectrum: This describes the phase shift (or timing) of each frequency component relative to a reference point. It provides insight into how the signal's frequency components are aligned with respect to one another in time. The phase spectrum is essential for reconstructing the original signal from its frequency components and is particularly important in signal transmission and modulation.

Together, the **magnitude and phase spectra** completely characterize the frequency content of the signal. The magnitude spectrum gives information about the signal's **intensity** across frequencies, while the phase spectrum provides information about the **temporal alignment** of those frequencies.

22. The primary difference between the **Discrete Time Fourier Transform (DTFT)** and the **Discrete Fourier Transform (DFT)** is:

- **DTFT** is a continuous frequency transform that applies to **discrete-time signals** of infinite length, and it produces a **continuous spectrum**. It is defined for all real-valued frequencies and is typically used for theoretical analysis.
- **DFT**, on the other hand, is a **finite-sampled version** of the DTFT and is defined for **periodic signals**. The DFT computes the frequency content of a **discrete-time signal** that is sampled over a finite time duration, producing a **finite set of frequency bins**. It is used for practical signal processing and analysis with a finite set of data points.

23. **Computational Efficiency:** FFT significantly reduces the computational complexity of DFT. A direct DFT computation requires O(NXN), while FFT reduces it to O(NlogN), making it much faster for large datasets.

- **Real-time Processing:** Due to its faster computation, FFT is ideal for real-time signal processing, especially when handling large datasets or signals.

23. How is the **DFT matrix** constructed?

The **DFT matrix** is a complex-valued matrix that is used to transform a sequence from the time domain to the frequency domain. The elements of the matrix are based on the complex exponential function, which encodes the frequency components of the signal. Each element of the matrix is a complex exponential, where:

$$W_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & e^{-j\frac{2\pi}{N}2} & \cdots & e^{-j\frac{2\pi}{N}(N-1)} \\ 1 & e^{-j\frac{2\pi}{N}2} & e^{-j\frac{2\pi}{N}4} & \cdots & e^{-j\frac{2\pi}{N}(N-1)2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j\frac{2\pi}{N}(N-1)} & e^{-j\frac{2\pi}{N}(N-1)2} & \cdots & e^{-j\frac{2\pi}{N}(N-1)(N-1)} \end{bmatrix}$$

24.

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

The two main properties of a Linear Time-Invariant (LTI) system are:

- **Linearity:** An LTI system is linear if it satisfies the principles of **additivity** (the system's response to the sum of two inputs is the sum of the responses to each input) and **homogeneity** (the system's response to a scaled input is the scaled response). Mathematically, this means that for any two input signals $x_1[n]$ and $x_2[n]$, and constants a and b, the system output for $a \cdot x_1[n] + b \cdot x_2[n]$ is $a \cdot y_1[n] + b \cdot y_2[n]$, where $y_1[n]$ and $y_2[n]$ are the responses to $x_1[n]$ and $x_2[n]$, respectively.
- **Time-Invariance:** An LTI system is time-invariant if its behavior and characteristics do not change over time. That is, if the input signal is shifted by k, the output will also be shifted by k. In other words, if $x[n]$ produces output $y[n]$, then $x[n-k]$ will produce output $y[n-k]$.

25. Convolution is a mathematical operation used to analyze LTI systems by determining the output of the system when a given input signal is passed through it. The output of an LTI system can be calculated by convolving the input signal with the system's **impulse response** h[n]. The process of convolution for discrete-time signals is given by:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Where:

- $x[n]$ is the input signal,
- $h[n]$ is the impulse response of the system,
- $y[n]$ is the output signal.

This operation essentially sums the weighted contributions of past inputs to the current output, with the weights being determined by the impulse response of the system.

- What is the significance of the **impulse response**?

The **impulse response** h[n] of an LTI system is the system's output when the input is a unit impulse δ[n]. It is a key characteristic of the system because:

- The impulse response completely characterizes the behavior of an LTI system. Once the impulse response is known, the system's output for any input can be determined using convolution.
- It provides insight into how the system responds to different frequencies, as it encapsulates the system's dynamics.
- The impulse response is essential for system analysis in both the time domain and the frequency domain, as it can be related to the system's transfer function or frequency response through the Fourier Transform.

Exercise 3:

1. **Analog filters** operate on **continuous-time** signals using electronic components (resistors, capacitors).

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Digital filters operate on **discrete-time** signals using **mathematical computations** (FIR/IIR filters).

2. **IIR filters use past outputs** (feedback) to create an infinite-duration response.

   FIR filters only use past inputs, so they do not need feedback, making them inherently stable.

   Describe one practical application of **Wavelet Transform** in **bio signal**.

The ECG signal is **non-stationary**, meaning its frequency content changes over time. Traditional methods like **Fourier Transform (FT)** fail to analyze such signals effectively because FT provides **only frequency information but no time localization**.

Wavelet Transform solves this problem by providing **both time and frequency resolution**. It allows **multi-resolution analysis (MRA)**, which helps in detecting:

Baseline Wander Removal

- Low-frequency noise caused by respiration and electrode movement can be removed using **low-pass wavelet coefficients**.

QRS Complex Detection

- The **QRS complex** (the sharp peak in an ECG) contains high-frequency components, which can be identified using **high-pass wavelet coefficients**.

Arrhythmia & Abnormality Detection

- Different heart conditions (e.g., **atrial fibrillation, tachycardia**) cause **specific changes** in ECG waveforms, which can be detected through wavelet-based feature extraction.

Example of Wavelet-Based ECG Processing

✓ **Denoising:** Removing motion artifacts and power-line interference.
✓ **Feature Extraction:** Identifying P-waves, QRS complexes, and T-waves.
✓ **Classification:** Using wavelet coefficients for diagnosing heart diseases.

Why Use Wavelet Transform for ECG Analysis?

Feature	Fourier Transform (FT)	Wavelet Transform (WT)
Time-Frequency Resolution	Only frequency, no time information	Both time and frequency

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Feature	Fourier Transform (FT)	Wavelet Transform (WT)
Best for	Stationary signals	Non-stationary signals like ECG
Noise Removal	Less effective	Highly effective
Feature Extraction	Limited	Superior for detecting cardiac events

Conclusion

Wavelet Transform is a powerful tool in **ECG analysis**, enabling accurate detection of heart abnormalities, noise removal, and feature extraction. It is widely used in **wearable heart monitors, hospital ECG machines, and AI-driven heart disease detection systems**.

3. Explain the significance of the unit impulse function in system analysis.

The **unit impulse function δ[n]\delta[n]δ[n]** plays a crucial role in system analysis because it is used to determine the **system's response** to an instantaneous input. The response of a system to δ[n]\delta[n]δ[n] is called the **impulse response**, denoted as h[n]h[n]h[n]. By knowing the impulse response of a system, we can predict how the system will respond to any arbitrary input using **convolution**. The impulse function is fundamental in **linear time-invariant (LTI) system analysis**, as it serves as a building block for analyzing and understanding the behavior of complex systems. The **impulse response** can be used to characterize system properties such as **stability, causality, and frequency response**.

4. Why is a Notch Filter commonly used in ECG and EEG signal processing?

A **Notch Filter** is commonly used in **ECG** (electrocardiogram) and **EEG** (electroencephalogram) signal processing to remove **power line interference** (typically at 50 Hz or 60 Hz). This interference, often caused by the electrical mains, can obscure the physiological signals of interest. A Notch Filter effectively removes this interference while leaving the underlying heart or brain activity largely unaffected, thereby improving the signal quality and making it easier to analyze.

5. Why is the **Fourier Transform** useful in signal processing?

   The **Fourier Transform** is essential in signal processing because it allows us to analyze a signal in the frequency domain, which is crucial for understanding and manipulating the signal's frequency content. By transforming a signal from the time domain to the frequency domain, we can easily identify and filter out noise, detect periodic components, and understand the signal's characteristics in terms of its frequency components. It also plays a key role in compression, modulation, and signal analysis, making it a foundational tool in many areas of engineering, communications, and physics.

6. How does the DTFT differ from the Fourier Transform?

The **DTFT** (Discrete-Time Fourier Transform) and the **Fourier Transform** both decompose signals into their frequency components, but they apply to different types of signals:

- **DTFT**: The DTFT is used for discrete-time signals (sequences). It transforms a sequence into a continuous frequency spectrum. The result is a periodic spectrum with infinite resolution, which is typically represented in the range from −π-\pi−π to π\piπ. The DTFT is especially useful for analyzing signals that are already sampled or inherently discrete.
- **Fourier Transform (FT)**: The Fourier Transform is generally used for continuous-time signals. It converts a time-domain signal into a continuous spectrum, which represents the signal's frequency content over all possible frequencies.

In essence, the **DTFT** is a special case of the **Fourier Transform** applied to discrete-time signals, where the frequency spectrum is periodic and continuous, whereas the **Fourier Transform** applies to continuous-time signals and produces a continuous, non-periodic spectrum.

7. Why is the **DFT** always computed using the **FFT algorithm**?

   The **DFT** is often computed using the **FFT** algorithm because the FFT provides a much more **efficient** method for calculating the DFT. The FFT reduces the time complexity from O(NXN) (for the direct DFT calculation) to O(NlogN), making it significantly faster, especially for large datasets. This efficiency gain allows for real-time processing of signals and large-scale computations, which would be computationally expensive with the direct DFT method.

8. How does **superposition** help determine if a system is **linear**?

   The principle of **superposition** helps determine if a system is linear by verifying whether the system satisfies the additivity and homogeneity properties. In simple terms, a system is linear if the response to a combination of inputs is the same as the combination of the responses to each input individually.

   - **Additivity:** If a system responds to two signals x1[n] and x2[n], then the system's response to x1[n]+x2[n] should be the sum of the individual responses.
   - **Homogeneity:** If the system responds to a signal x[n], then for any constant a, the response to a·x[n] should be a times the original response.

By testing these properties, superposition ensures that the system is linear. If a system violates either property, it is not linear.

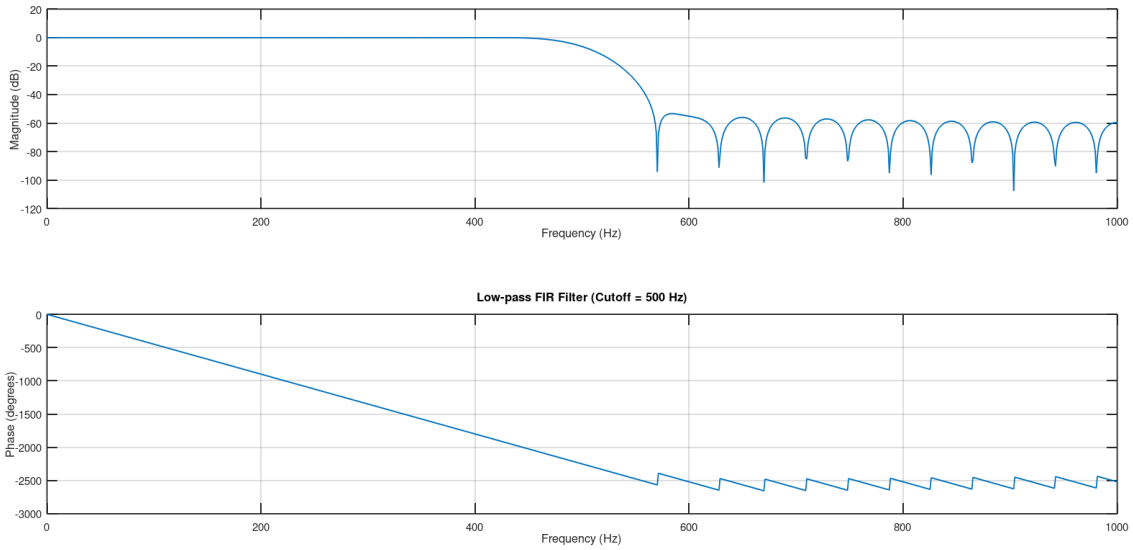
Exercise 4

We can use a **Finite Impulse Response (FIR) filter** using the **window method** or an **Infinite Impulse Response (IIR) filter** like a **Butterworth filter**.

Using this code (Hamming windows) is a FIR filter

```
pkg load signal  % Load the signal processing package
```

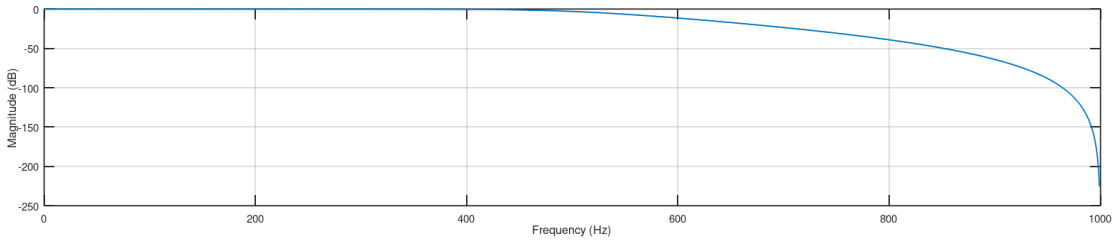
```
fs = 2000;          % Sampling frequency in Hz
fc = 500;           % Cutoff frequency in Hz
N = 50;             % Filter order
wn = fc / (fs / 2); % Normalized cutoff frequency (0 to 1)

% Design low-pass FIR filter using Hamming window
b = fir1(N, wn, 'low', hamming(N+1));

% Frequency response
freqz(b, 1, 1024, fs);  % Plot frequency response
title('Low-pass FIR Filter (Cutoff = 500 Hz)');
```

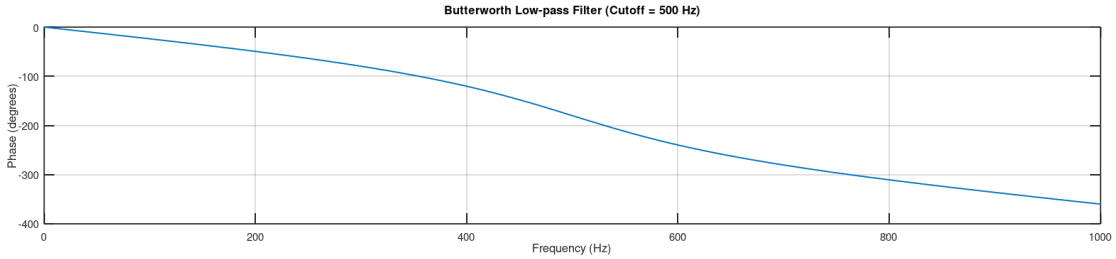


Using this code (Butterworth) is an IIR filter

```
pkg load signal  % Load the signal package

fs = 2000;          % Sampling frequency (Hz)
fc = 500;           % Cutoff frequency (Hz)
N = 4;              % Filter order (adjustable)

% Normalize the cutoff frequency
wn = fc / (fs / 2);  % Normalize with respect to Nyquist freq

% Design Butterworth filter
[b, a] = butter(N, wn, 'low');

% Plot frequency response
freqz(b, a, 1024, fs);
title('Butterworth Low-pass Filter (Cutoff = 500 Hz)');
```


O. ELOUTASSI: o.eloutassi@vac.ueuromed.org





- **IIR filters** are more efficient because they require fewer coefficients.
- The **Butterworth filter** provides a maximally flat response.

Justification:

- **FIR** is used when **linear phase** is required.
- **IIR** is used when computational efficiency is preferred.

Exercise 5:

This is an **IIR (Infinite Impulse Response) filter** because:

The equation contains a **recursive term** (**0.5y[n-1]**), meaning the output depends on past outputs. **IIR filters always have feedback**, while **FIR filters do not**.

Compute the output for x[n]={1,2,3,4}, assuming y[−1]=0

We compute the output step by step:

n	y[n] Calculation	Result
0	y[0]=0.5·y[−1]+x[0]−x[−1]=0.5·0+1−0=1	1
1	y[1]=0.5·y[0]+x[1]−x[0]=0.5·1+2−1=1.5	1.5
2	y[2]=0.5·y[1]+x[2]−x[1]=0.5·1.5+3−2=1.75	1.75
3	y[3]=0.5·y[2]+x[3]−x[2]=0.5·1.75+4−3=1.875	1.875

use this code to compute the results

```
pkg load signal  % Load signal package

% Input signal
```


O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

```
x = [1, 2, 3, 4];

% Filter coefficients

b = [1, -1];      % Numerator (x[n] - x[n-1])

a = [1, -0.5];    % Denominator (y[n] - 0.5*y[n-1])

% Compute output

y = filter(b, a, x);

% Display result

disp('Output y[n]:');

disp(y);
```

Exercises 6. Wavelet Transform

The **Haar wavelet transform** is the simplest wavelet transform, often used in signal processing. It is particularly useful for analyzing signals with **sharp changes** because of its step-like nature.

Wavelet transforms decompose a signal into different **frequency components** while preserving **time-domain information**, unlike the Fourier Transform, which only provides frequency information.

Wavelet transforms analyze signals **at multiple scales** by breaking them down into:

- **Low-frequency components (Approximation coefficients)** → Capture the general trend of the signal.
- **High-frequency components (Detail coefficients)** → Capture rapid changes in the signal.

Unlike the Fourier Transform, which represents signals as sums of sinusoids, wavelets are **localized in both time and frequency**, allowing better detection of transient features.

Octave Code for Haar Wavelet Transform

Let's generate a signal, apply the Haar wavelet transform, and visualize the results.

```
% Define a sample signal (e.g., step-like signal)
x = [1 2 3 4 4 3 2 1];
N = length(x);

% Initialize coefficient vectors
cA = zeros(1, N/2);  % Approximation coefficients (low frequencies)
cD = zeros(1, N/2);  % Detail coefficients (high frequencies)
```

```
% Perform 1-level Haar wavelet transform
for k = 1:2:N
    i = (k+1)/2;
    cA(i) = (x(k) + x(k+1)) / sqrt(2);  % Average
    cD(i) = (x(k) - x(k+1)) / sqrt(2);  % Difference
end

% Plotting results
subplot(3,1,1);
stem(x, 'b', 'filled');
title('Original Signal');
xlabel('Sample');
ylabel('Amplitude');

subplot(3,1,2);
stem(cA, 'r', 'filled');
title('Approximation Coefficients (cA)');
xlabel('Index');
ylabel('Amplitude');

subplot(3,1,3);
stem(cD, 'g', 'filled');
title('Detail Coefficients (cD)');
xlabel('Index');
ylabel('Amplitude');
```
Explanation of Results

1. **Original Signal:** The input signal is displayed as a sequence of discrete points.
2. **Approximation Coefficients:** Represent the low-frequency part, capturing the **smooth structure** of the signal.
3. **Detail Coefficients:** Represent the high-frequency part, detecting **sharp transitions** in the signal.

Wavelet transforms are useful in **denoising, feature extraction, and image compression** (like JPEG 2000).

Exercise 7: Z-Transform of the Given Sequence

The **Z-Transform** of a sequence x[n] is defined as: $X(z) = \sum_{n=-\infty}^{\infty} x[n]Z^{-n}$

Since the sequence x[n]=δ[n]+2δ[n−1]−δ[n−2] consists of impulse functions, we can compute the Z-Transform of each term separately:

1. **Z-Transform of δ[n]**: $Z\{\delta[n]\} = 1$
2. **Z-Transform of δ[n−1]**: $Z\{\delta[n-1]\} = z^{-1}$
3. **Z-Transform of δ[n−2]]**:$\{\delta[n-2]\} = z^{-2}$

Thus, the Z-Transform of x[n] is: $X(z) = 1 + 2z^{-1} - z^{-2}$

The **Region of Convergence (ROC)** for the Z-Transform of a signal depends on its causality and stability. For this sequence, it is composed of a finite number of impulses, which means the ROC is typically the entire **complex plane**, excluding any singularities of X(z).

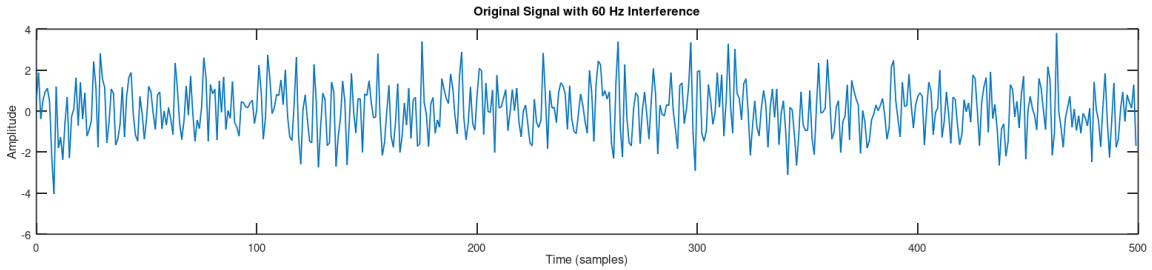
Since X(z) polynomial in z^{-1}, it has no poles, so the ROC is: |z|>0

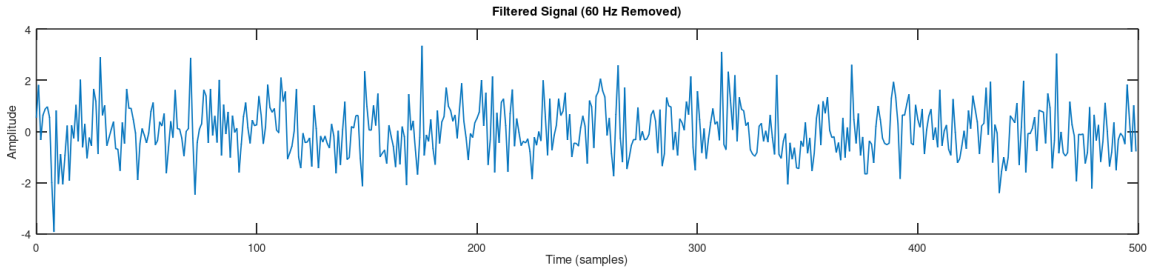
This indicates that the Z-Transform converges for all values of z except at z=0

Exercise 8 Notch filters:

Use this code:

```
pkg load signal  % Load the signal processing package
% Parameters
fs = 250;        % Sampling frequency in Hz
f_notch = 60;    % Notch frequency (power line frequency) in Hz
Q = 30;          % Quality factor (higher Q means a narrower notch)
% Normalize the notch frequency to Nyquist frequency
Wn = [f_notch - 1, f_notch + 1] / (fs / 2);  % Create a frequency band around 60 Hz
% Design a band-stop (Notch) filter using butterworth
[b, a] = butter(2, Wn, 'stop'); % Band-stop filter
% Display the filter coefficients
disp('Filter Coefficients:');
disp('Numerator (b):');
disp(b);
disp('Denominator (a):');
disp(a);
% Generate a test EEG signal with 60 Hz interference
n = 0:499;  % Create 500 samples
x = sin(2 * pi * 60 * n / fs) + randn(size(n));  % 60 Hz signal with noise
% Apply the band-stop (Notch) filter to remove 60 Hz interference
y = filter(b, a, x);
% Plot the original and filtered signal
subplot(2, 1, 1);
plot(n, x);
title('Original Signal with 60 Hz Interference');
xlabel('Time (samples)');
ylabel('Amplitude');
subplot(2, 1, 2);
plot(n, y);
title('Filtered Signal (60 Hz Removed)');
xlabel('Time (samples)');
ylabel('Amplitude');
```





To remove the 60 Hz interference, we will design a **Notch Filter** (a type of band-stop filter) with a narrow bandwidth around 60 Hz. The goal of this filter is to attenuate frequencies around 60 Hz without significantly affecting the rest of the signal spectrum.

1. **Cutoff Frequency**: The notch filter should be centered around 60 Hz, i.e., the notch filter's frequency is at **60 Hz**.
2. **Bandwidth**: The bandwidth of the notch filter determines how wide the attenuation region is around 60 Hz. A narrower bandwidth provides more precise removal of the interference. Typically, a bandwidth of 1-10 Hz is used for this application to ensure minimal signal distortion.
3. **Sampling Frequency**: The signal is sampled at fs=250. To design the filter, we need to convert the frequency specifications to the normalized digital frequency (between 0 and 1) by dividing by the sampling frequency fs.

   Normalized frequency $fnotch = 60/250 = 0.24$ f (in cycles per sample).

4. **Filter Type**: A **second-order notch filter** is typically used to remove narrow-band interference like power line noise. The filter can be implemented using a **IIR (Infinite Impulse Response) filter** or **FIR (Finite Impulse Response) filter**, but IIR filters are more efficient for this type of design.

Exrcise 9: Fourier Transform

Since u(t) is the unit step function, x(t) $= \begin{cases} 0, t < 0 \\ e^{-2t}, t \geq 0 \end{cases}$

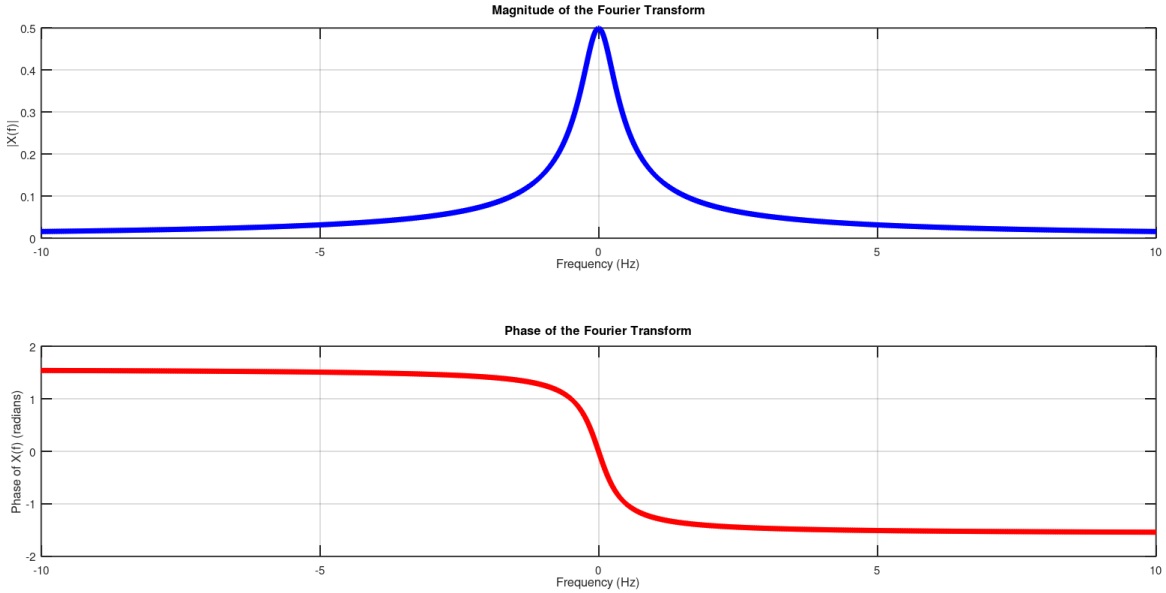
Apply the Fourier Transform: $X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt = \int_{0}^{\infty} e^{-2t} e^{-j2\pi ft}\, dt$

Simplify the Integral Combine the exponentials: $X(f) = \int_{0}^{\infty} e^{-(2+j2\pi f)t}\, dt = \frac{1}{2+j2\pi f}$

This represents the frequency-domain representation of the signal x(t).

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Plot the frequency domain as magnitude and phase since $X(f)$ it is a complex result



We did use this code

```
% Define frequency range

f = linspace(-10, 10, 1000);

% Analytical Fourier Transform of x(t) = e^(-2t) * u(t)

Xf = 1 ./ (2 + j*2*pi*f);

% Plot magnitude and phase

figure;

subplot(2,1,1);

plot(f, abs(Xf), 'b', 'LineWidth', 2);

xlabel('Frequency (Hz)');

ylabel('|X(f)|');

title('Magnitude of the Fourier Transform');

grid on;

subplot(2,1,2);

plot(f, angle(Xf), 'r', 'LineWidth', 2);

xlabel('Frequency (Hz)');

ylabel('Phase of X(f) (radians)');
```


O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

```
title('Phase of the Fourier Transform');

grid on;
```

Exercise 10. DTFT (Discrete-Time Fourier Transform)

Since the sequence is finite (x[n]={1,2,3,4} for n=0,1,2,3), the DTFT simplifies to:

$$X(e^{j\omega}) = x[0] + x[1]e^{-j\omega} + x[2]e^{-2j\omega} + x[3]e^{-3j\omega} = 1 + 2e^{-j\omega} + 3e^{-2j\omega} + 4e^{-3j\omega}$$

Magnitude Spectrum:

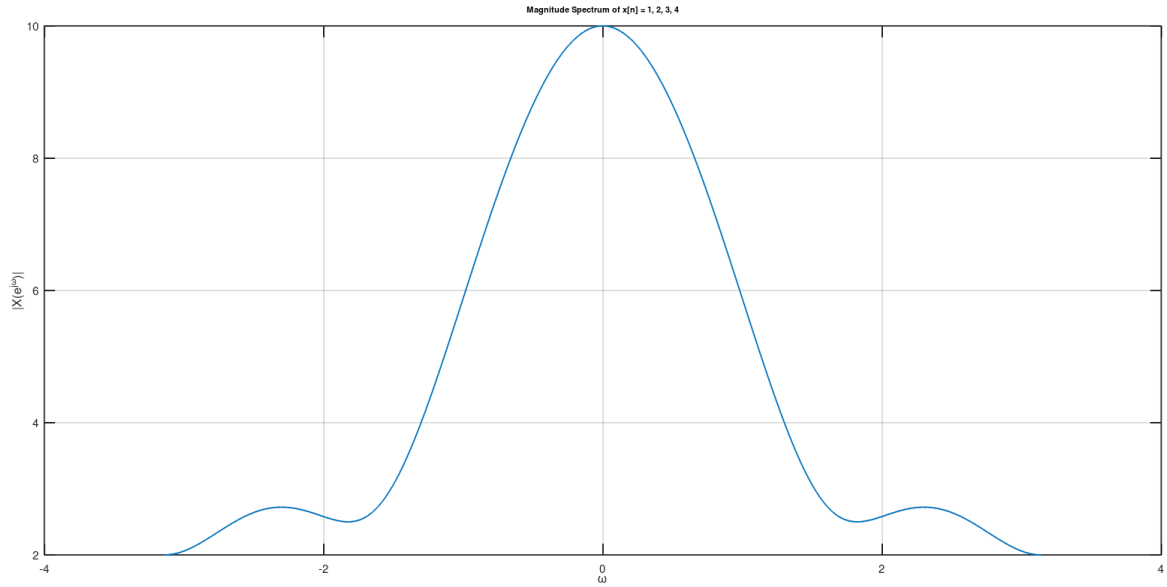
The magnitude spectrum $|X(e^{j\omega})| = |1 + 2e^{-j\omega} + 3e^{-2j\omega} + 4e^{-3j\omega}|$ is simply the magnitude of the DTFT:

To sketch the magnitude spectrum, you would compute this expression numerically for values of ω ranging from $-\pi$ to π. In practice, this is often done by sampling ω at different points and plotting the resulting magnitude.

Octave Code for DTFT Calculation and Plot:

You can use Octave (or MATLAB) to compute and plot the DTFT and its magnitude spectrum. Here's a simple implementation:

```
% Define the sequence x[n]
x = [1, 2, 3, 4];

% Define the frequency range for DTFT
omega = linspace(-pi, pi, 1000); % 1000 points from -pi to pi

% Initialize the DTFT result
X_omega = zeros(1, length(omega));

% Compute the DTFT using the formula
for k = 1:length(omega)
    X_omega(k) = sum(x .* exp(-1j * omega(k) * (0:length(x)-1)));
end

% Compute the magnitude spectrum
magnitude_spectrum = abs(X_omega);

% Plot the magnitude spectrum
figure;
plot(omega, magnitude_spectrum);
title('Magnitude Spectrum of x[n] = {1, 2, 3, 4}');
xlabel('\omega');
ylabel('|X(e^{j\omega})|');
grid on;
```



Explanation of the Plot:

The plot you generate with this code will show how the magnitude of the DTFT varies with frequency. The frequency variable ω\omegaω ranges from −π-\pi−π to π\piπ, and the magnitude spectrum shows the strength of each frequency component in the signal.

- The **peaks** in the magnitude spectrum correspond to the frequency components where the signal has the most energy.
- The shape of the magnitude spectrum is influenced by the values of the sequence and their relative positions in time.

This visualization helps to analyze the frequency content of the sequence x[n].

Exercise 11. DFT (Discrete Fourier Transform)

Compute the **DFT** of the sequence: x[n]={0.5,0,1.0,1.5} using **matrix multiplication** and d**efinition of DFT (**Discrete Fourier Transform). For the sequence x[n] of length N is given by the formula

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

Where: X[k] is the k-th component of the DFT (for k=0,1,2,...,N−1), x[n] is the input sequence (for n=0,1,2,...,N−1), N is the length of the sequence.

The DFT as a matrix multiplication: $X = W_N \cdot x$

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

Where: X is the vector of DFT coefficients, x is the input signal vector, W_N is the **DFT matrix** of size N×N, defined as:

$$W_N = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & e^{-j\frac{2\pi}{N}2} & e^{-j\frac{2\pi}{N}3} \\ 1 & e^{-j\frac{2\pi}{N}2} & e^{-j\frac{2\pi}{N}4} & e^{-j\frac{2\pi}{N}6} \\ 1 & e^{-j\frac{2\pi}{N}3} & e^{-j\frac{2\pi}{N}6} & e^{-j\frac{2\pi}{N}9} \end{bmatrix}$$

For N=4 is: W_4:

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

The input sequence is:

$$x[n] = \begin{bmatrix} 0.5 \\ 0 \\ 1.0 \\ 1.5 \end{bmatrix}$$

we perform the matrix multiplication $X = W_4.x$ to compute the DFT

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0 \\ 1.0 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.5 + j1.5 \\ 0.0 \\ -0.5 - j1.5 \end{bmatrix}$$

To plot the **magnitude spectrum**, we calculate the magnitude of each DFT component:

$$|X[0]| = |3.0| = 3.0$$
$$|X[1]| = \sqrt{(-0.5)^2 + (1.5)^2} = \sqrt{0.25 + 2.25} = \sqrt{2.5} \approx 1.58$$
$$|X[2]| = |0.0| = 0.0$$
$$|X[3]| = \sqrt{(-0.5)^2 + (-1.5)^2} = \sqrt{0.25 + 2.25} = \sqrt{2.5} \approx 1.58$$

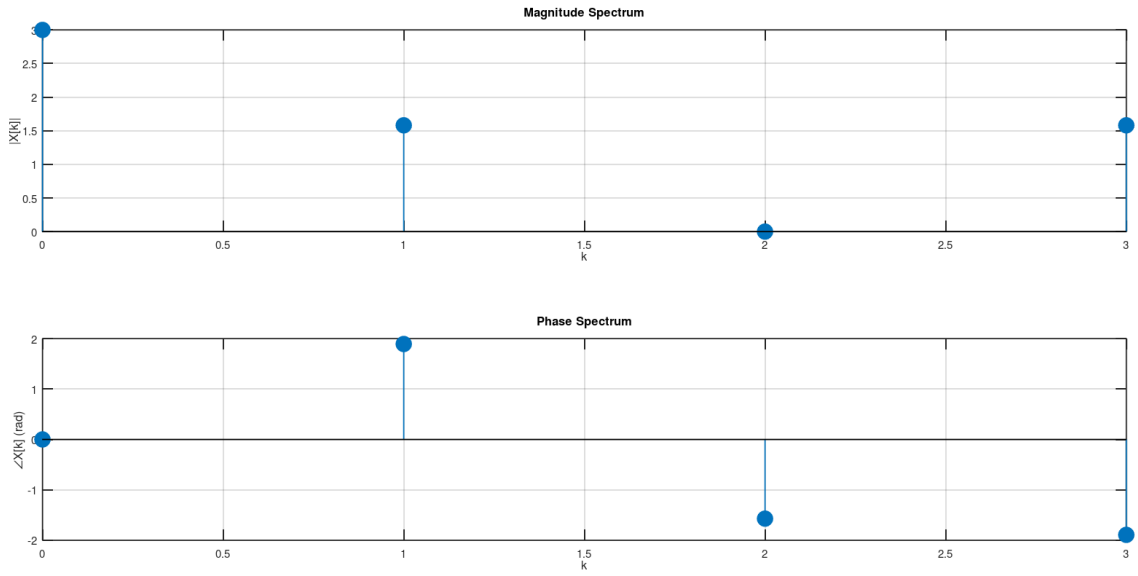
We get then the magnitude spectrum as |X|={3.0,1.58,0.0,1.58}

To plot the result we can use octave code

```
% Define the input sequence

x = [0.5; 0; 1.0; 1.5];  % Column vector

N = length(x);

% Construct the DFT matrix

W = zeros(N, N);

for k = 0:N-1

   for n = 0:N-1
```


O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

```
        W(k+1, n+1) = exp(-j*2*pi*k*n/N);
    end
end
% Compute the DFT using matrix multiplication
X = W * x;
% Display the result
disp('DFT Coefficients X[k]:');
disp(X);
% Plot the magnitude and phase
figure;
subplot(2,1,1);
stem(0:N-1, abs(X), 'filled');
xlabel('k'); ylabel('|X[k]|');
title('Magnitude Spectrum');
grid on;
subplot(2,1,2);
stem(0:N-1, angle(X), 'filled');
xlabel('k'); ylabel('∠X[k] (rad)');
title('Phase Spectrum');
grid on;
```



Exercise 12 Linear Systems

A system is **linear** if it satisfies: **Additivity**: T{x1[n]+x2[n]}=T{x1[n]}+T{x2[n]}and **Homogeneity (scaling)**: T{a·x[n]}=a·T{x[n]}

Let's consider: x1[n]: a signal with output y1[n] and : x2[n]: a signal with output y2[n]

Now apply input x[n]=a·x1[n]+b·x2 , the output becomes:

y[n] = 0.5·y[n−1]+a·x1[n]+b·x2[n] = 0.5·y[n−1]+a·x1[n]+b·x2[n]

A system is **time-invariant** if delaying the input delays the output by the same amount. We suppose:

Input: x[n]→y[n] then the Delayed input: x[n−n0]→y′[n]=0.5·y′[n−1]+x[n−n0]]. However, if we assume the system is **initially at rest** (zero initial state), the output is simply a **shifted version** of the original output.

To find the **impulse response**, apply x[n]=δ[n], the unit impulse:

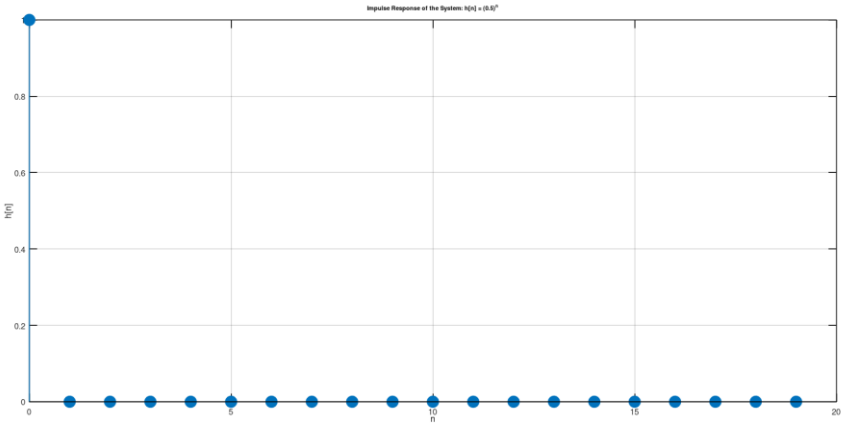
Assume initial rest: y[−1]=0, we compute:

h[0] = δ[0]+0.5·y[−1]=1+0 =1
h[1] = δ[1]+0.5·h[0] = 0+0.5·1= 0.5
h[2] = 0.5·h[1] = 0.25
h[3]=0.5·h[2] = 0.125
.
.
.
$h[n]=(0.5)^n$ for n≥0

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

We can use the Octave code

```
% Define system parameters

N = 20;                % Length of response

x = zeros(1, N);       % Input: Impulse signal

x(1) = 1;              % Delta[n] = 1 at n = 0

y = zeros(1, N);       % Output (impulse response)

% Apply the difference equation: y[n] = 0.5*y[n-1] + x[n]

for n = 2:N

   y(n) = 0.5 * y(n-1) + x(n);

end

y(1) = x(1);  % Initial condition (y[0] = x[0])

% Display and plot the impulse response

disp('Impulse response h[n]:');

disp(y);

stem(0:N-1, y, 'filled');

xlabel('n'); ylabel('h[n]');

title('Impulse Response of the System');

grid on;
```



Exercise 13

Step 1: Compute the Convolution y[n]

The convolution sum is:

O. ELOUTASSI: o.eloutassi@vac.ueuromed.org

$$y[n] = \sum_k x[k]h[n-k]$$]

Writing the sequences explicitly: x[n]={1,2,3,4},h[n]={1,−1,2} Perform the convolution manually:

n	Computation	y[n]
0	1(1)	1
1	1(−1)+2(1)	1
2	1(2)+2(−1)+3(1)	3
3	2(2)+3(−1)+4(1)	5
4	3(2)+4(−1)	2
5	4(2)	8

Thus, the output sequence is: y[n]={1,1,3,5,2,8}

Step 2: Check Symmetry of h[n]h[n]h[n]

A kernel function h[n] is symmetric if: h[n]=h[−n]

Given: h[n]={1,−1,2} reversing the sequence: h[−n]={2,−1,1}

Since h[n]≠h[−n], the kernel is not symmetric.

Step 3: Modification for a Smoothing Filter

To use the kernel as a smoothing filter, it should have **equal-weighted or low-pass characteristics** (e.g., an averaging filter). A good modification would be:

$$h[n] = \frac{1}{3}\{1,1,1\}$$

This modified kernel will **smooth** the input signal rather than introducing large variations.