# Comparison of Wavelet Transform and Discrete Fourier Transform for 1D-Signals

1st Gökcer Sönmezocak
*Department of Computer Science*
*Fachhochschule Dortmund*
Dortmund, Germany
Matriculation Nr.: 7218785 goekcer.soenmezocak001@stud.fh-dortmund.de

***Abstract*—The Haar Wavelet Transform (HWT) and Discrete Fourier Transform (DFT) are two widely-used techniques in digital signal processing for analyzing one-dimensional (1D) signals, particularly in the context of audio processing. This paper presents a comparative analysis of these methods, highlighting the distinct advantages and limitations of each in terms of time-frequency localization, data compression, and signal reconstruction. While the DFT excels in providing a global frequency domain representation of a signal, it lacks the ability to provide time localization, making it less effective for analyzing non-stationary signals. Conversely, the Haar Wavelet Transform offers the unique capability of capturing both time and frequency information, making it ideal for applications requiring signal compression and reconstruction. The paper details the implementation of both transforms using Python, including the inverse Haar Wavelet Transform, which allows for perfect reconstruction of the original signal from its compressed form. Through practical examples, the paper demonstrates the superiority of the Haar Wavelet Transform for applications where maintaining the integrity and fidelity of the signal is critical, such as in audio signal processing. The results suggest that the Haar Wavelet Transform is a more versatile and efficient tool for such tasks compared to the DFT.**

***Index Terms*—Haar Wavelet Transform, Discrete Fourier Transform, 1D Signal Processing, Time-Frequency Analysis, Signal Compression, Signal Reconstruction, Wavelet Functions, Audio Processing, Digital Signal Processing**

## I. Introduction

In the field of digital signal processing, the analysis and transformation of one-dimensional (1D) audio signals are essential for various applications, including data compression, noise reduction, and feature extraction. Among the most prominent techniques used for signal analysis are the Discrete Fourier Transform (DFT) and the Wavelet Transform, specifically the Haar Wavelet Transform. Each of these methods offers unique advantages and limitations, making them suitable for different applications depending on the specific requirements of the signal processing task.

The Discrete Fourier Transform is a powerful tool that transforms a signal from the time domain to the frequency domain, providing valuable insights into the frequency components of the signal. However, while DFT is highly effective in frequency analysis, it lacks the ability to efficiently compress data or reconstruct signals with minimal loss. This limitation is particularly evident in applications where data compression and reconstruction are crucial, such as in audio signal processing, where preserving the integrity of the original signal after compression is essential.

In contrast, the Wavelet Transform, particularly the Haar Wavelet, has emerged as a versatile method capable of both analyzing and compressing 1D audio signals [5]. The Haar Wavelet Transform is known for its ability to decompose a signal into various frequency components while maintaining localized time information. This dual capability enables the efficient compression of audio signals by reducing the amount of data required to represent the signal without significant loss of quality. Furthermore, the Haar Wavelet Transform supports the reconstruction of the original signal from its compressed form, a feature that the DFT lacks. This reconstruction capability is vital for applications where data integrity and quality are critical after compression.

This paper aims to compare the DFT and the Haar Wavelet Transform, focusing on their respective strengths and weaknesses in the context of 1D audio signal processing. Specifically, the paper will explore how each method performs in terms of data compression, signal reconstruction, and the preservation of key signal characteristics. By highlighting the distinct advantages of the Haar Wavelet Transform in compressing and reconstructing audio signals, this comparison will underscore the suitability of wavelet-based methods for applications where data efficiency and signal fidelity are paramount.

## II. Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a mathematical technique used to convert a discrete time-domain signal into its corresponding frequency domain representation. [3] Essentially, DFT analyzes the input signal by breaking it down into its constituent frequency components, producing a list of frequencies present in the signal along with their respective amplitudes. However, one key limitation of DFT is that it provides no information about when these frequencies occur within the original signal. In other words, DFT lacks time localization, meaning it only tells us which frequencies are present but not at what time they appear in the signal. This makes DFT less effective for analyzing signals where

Identify applicable funding agency here. If none, delete this.

the timing of frequency components is important, such as in transient or non-stationary signals.

For implementing DFT in python language, it has been benefited from the numpy library as below.

```
# Define DFT function

def just_dft(signal, sample_rate):
    # Perform the Discrete Fourier Transform
        (DFT)
    dft = np.fft.fft(signal)
    frequencies = np.fft.fftfreq(len(signal),
        d=1/sample_rate)
    return dft, frequencies
```

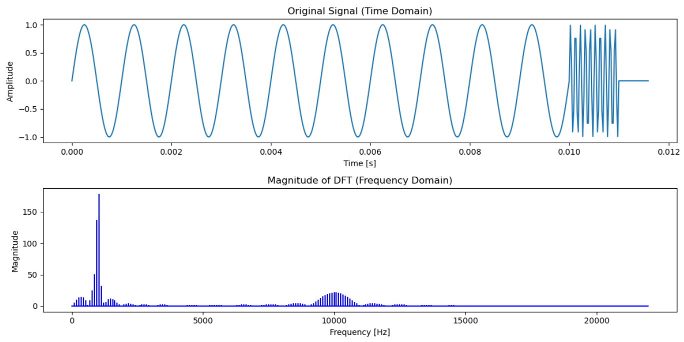

Fig. 1. 1D audio signal which composed of 1kHz and 10kHz components and its frequency domain representation after DFT implementation.

As it can be seen from the Fig. 1., DFT results in just the frequency components of the provided time domain signal without providing any time localization.

## III. Wavelet Transform

The Wavelet Transform is a powerful signal processing technique that simultaneously provides both frequency and time localization information about the input signal. Unlike the Discrete Fourier Transform (DFT), which only reveals the frequencies present in a signal without indicating when they occur, the Wavelet Transform decomposes a signal into components that reveal how its frequency content evolves over time. This dual capability allows for a more detailed and precise analysis of signals, especially those that are non-stationary or contain transient features.

Because the Wavelet Transform captures both time and frequency information [4], it is particularly well-suited for signal compression. It enables the efficient representation of a signal by focusing on the most significant components, thereby reducing the amount of data required. Additionally, the Wavelet Transform supports the reconstructability of the original signal from its compressed form, ensuring that the essential characteristics of the signal can be preserved and restored with minimal loss. This makes it an ideal tool for applications where maintaining signal integrity after compression is critical, such as in audio processing.

### A. Wavelet Functions

Wavelet functions, such as Haar and Daubechies wavelets, are fundamental building blocks of the Wavelet Transform and serve as the basis for analyzing signals in both the time and frequency domains. These functions differ significantly from the sinusoidal basis functions used in the Discrete Fourier Transform (DFT), leading to distinct advantages in various signal processing applications.

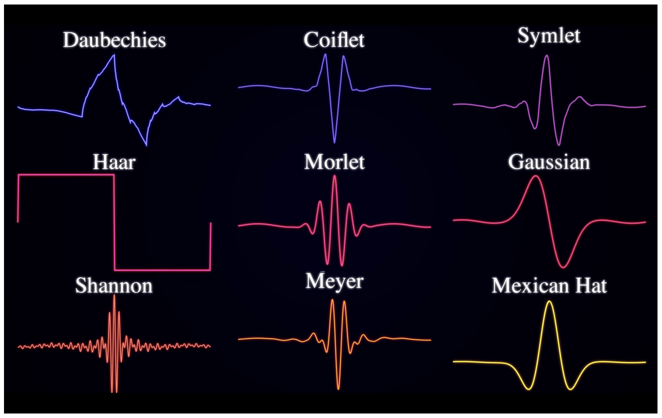

Fig. 2. Different wavelet functions used in the wavelet transform [1]

The Haar wavelet is one of the simplest and most intuitive wavelet functions. It represents the signal as a series of step functions, which makes it particularly useful for capturing sharp changes or discontinuities in the signal. The Haar wavelet provides excellent time localization but offers less precision in frequency resolution compared to more complex wavelets.

On the other hand, Daubechies wavelets are a family of wavelets known for their ability to provide a better balance between time and frequency localization. They are more sophisticated than the Haar wavelet and are characterized by smoother, more oscillatory waveforms, which allow for better frequency resolution while still maintaining good time localization. This makes Daubechies wavelets particularly useful for analyzing signals with more complex structures, where both time and frequency details are important.

The key difference between wavelet functions like Haar or Daubechies and the sinusoidal basis functions of DFT lies in their ability to localize information in both time and frequency domains. While DFT uses sine and cosine functions to represent the entire signal, assuming it to be periodic and extending over all time, wavelet functions are localized both in time and frequency. This means wavelets can capture transient, non-stationary, or localized events within a signal more effectively than DFT, which only provides global frequency information without any time context.

This ability of wavelet functions to provide detailed time-frequency analysis is what makes them especially powerful for applications such as data compression. By focusing on significant wavelet coefficients and discarding less important ones, wavelet-based methods can compress data efficiently while still enabling the reconstruction of the original signal. This reconstructability feature ensures that the essential characteristics of the signal are preserved, a capability that is limited in DFT due to its lack of time localization.

## *B. Discrete Wavelet Transform*

The Discrete Wavelet Transform (DWT) is a signal processing technique that decomposes a signal into a set of wavelet coefficients, capturing both time and frequency information in a hierarchical manner. Unlike the Discrete Fourier Transform (DFT), which only provides global frequency information, the DWT analyzes the signal at multiple scales or resolutions, allowing for the detection of both high-frequency details and low-frequency trends within the signal.

In digital signal processing, the DWT is used because of its ability to represent signals with varying degrees of detail. This makes it particularly effective for tasks such as signal compression, noise reduction, and feature extraction. The multi-resolution analysis provided by the DWT is ideal for efficiently capturing important signal characteristics while discarding redundant or insignificant information, leading to more compact signal representations that can still be accurately reconstructed. This efficiency, combined with the ability to focus on both time and frequency aspects of a signal, is why the DWT is widely used in applications like audio and image compression, where preserving essential features while reducing data size is crucial.

In this paper, Discrete Wavelet Transform is investigated by especially benefiting from Haar Wavelet applications.

## *C. Uncertainty Principle*

In fact, there is a fundamental limit to how much a function and its DFT can both be localized. The *uncertainty principle* states that [2]

$$\|f\|_0 \|\mathcal{D}f\|_0 \geq n, \tag{1}$$

where $\|f\|_0$ is the number of nonzero values of $f$.

- This says that it is impossible for both $f$ and $\mathcal{D}f$ to both be localized (i.e., have mostly zero entries).
- This bound is saturated by the Fourier basis functions $u_\ell$, which satisfy $\|u_\ell\|_0 = n$ and $\|\mathcal{D}u_\ell\|_0 = 1$.

The Wavelet Transformation is a principled approach to finding a decomposition of a signal or image into frequency components where the basis functions are localized in time/space and frequency, as much as is possible.

## *D. The 1D Haar Wavelet Transformation*

The Wavelet Transform is based on repeatedly decomposing a signal into a low frequency part, called the approximation coefficients, and a high frequency part, called the detail coefficients. This is best illustrated at first with an example. Consider the following length n = 8 signal. [2] Signal: (7,5,6,3,2,5,4,1)

The signal can be portioned as 2-tuples in the form of (A,B). Then the approximation coefficients and the detail coefficients can be defined as follows,

approximation coefficient = A + B

detail coefficient = A - B

Then, according to the input signal

approximation coefficients = {12, 9, 7, 5}

detail coefficients = {-2, -3, 3, -3}

Finally, 1-level Haar Transform results in the following

1-level Haar = {12, 9, 7, 5, -2, -3, 3, -3}

## *E. Python Code Implementation of the 1D Haar Wavelet Transform*

```
# Define the Haar Wavelet Transform
    function

def haar_wavelet(f,depth):

   if len(f) % 2 == 1:
      f = f[:-1]

   g = np.zeros_like(f)
   n2 = len(f)>>1

   first_half = f[::2]
   second_half = f[1::2]

   approx_coeff = []
   detail_coeff = []

   for i in range(len(first_half)):
      approx_index_result = first_half[i]
          + second_half[i]
      approx_coeff.append(approx_index_result)

      detail_index_result =
          second_half[i] - first_half[i]
      detail_coeff.append(detail_index_result)

   g[:n2] = approx_coeff

   g[n2:] = detail_coeff

   if depth >= 2:
      g[:n2] =
          haar_wavelet(g[:n2],depth-1)
   return g
```

The provided code implements the Haar Wavelet Transform (HWT) for a one-dimensional signal f and is designed to recursively apply the transformation up to a specified depth, allowing for multiple levels of decomposition. This implementation begins by handling signals of odd length by discarding the last element if necessary, ensuring that the signal length is even. This step is crucial because the Haar Wavelet Transform operates by pairing adjacent elements in the signal.

The transformation process initializes a new array g that has the same length as the input signal f . This array will store the transformed coefficients, with the variable n2 representing half the length of f , indicating the point at which the approximation and detail coefficients are separated within the array. The input signal is then divided into two parts: first_half, consisting of the even-indexed elements, and second_half, consisting of the odd-indexed elements.

The function proceeds to compute the approximation and detail coefficients. The approximation coefficients are calculated by summing each pair of elements from first_half and second_half, while the detail coefficients are determined by taking the difference between the same pairs of elements. These coefficients are then stored in the array g , with the approximation coefficients populating the first half of g and the detail coefficients filling the second half.

To achieve a multi-level decomposition, the function recursively applies the Haar Wavelet Transform to the approximation coefficients (the first half of g ) if the specified depth is greater than or equal to 2. This recursive approach enables the extraction of finer details from the signal at each successive level of decomposition.

Finally, the function returns the array g , which contains the Haar Wavelet coefficients, representing both the approximation and detail information of the input signal at the specified depth. This recursive method of decomposition makes the Haar Wavelet Transform particularly effective for analyzing signals that require both time and frequency localization, as well as for applications such as data compression, where the ability to reconstruct the original signal from its compressed form is essential.

This implementation is observed to be worked for the signals that they have number of samples that are the powers of 2. Therefore, the input signals are needed to be 0-padded to the next nearest power of 2 before provided to this function for multi-dimensional analysis purpose.

### *F. Results of the Haar Wavelet Transform*

The generated sample signal which is demonstrated on Fig. 3. has been used to investigate its Haar Wavelet Transform analysis up to level 3. The signal is generated by the following code in python language considering 0-padding that was mentioned earlier on this paper as it can be observed as constant 0 values at the most right hand-side of the signal representations.

```
# Generate a sample 1D audio signal
    composed of 1kHz and 10kHz components

# Parameters
fs = 44100 # Sampling frequency
f1 = 1000 # Frequency 1 (1 kHz)
f2 = 10000 # Frequency 2 (10 kHz)

# Number of samples for 10 full waves
n1 = int(fs * 10 / f1) # For 1 kHz
n2 = int(fs * 10 / f2) # For 10 kHz

# Total number of samples (next power of
    2)
total_samples = n1 + n2
next_power_of_2 = int(pow(2,
    math.ceil(math.log(total_samples, 2))))

# Time vectors
t1 = np.linspace(0, 10/f1, n1,
    endpoint=False)
t2 = np.linspace(0, 10/f2, n2,
    endpoint=False)

# Sine waves
sine_wave_1 = np.sin(2 * np.pi * f1 * t1)
sine_wave_2 = np.sin(2 * np.pi * f2 * t2)

# Combine the signals and pad with zeros
    if necessary
signal = np.concatenate((sine_wave_1,
    sine_wave_2))
signal = np.pad(signal, (0,
    next_power_of_2 - len(signal)),
    'constant')

# Plot the signal
plt.figure(figsize=(10, 4))
plt.plot(signal)
plt.xlabel('Sample Number')
plt.ylabel('Amplitude')
plt.title('Sine Signal with 1 kHz and 10
    kHz Components')
plt.show()

# Plot the signal in the Time Domain
plt.figure(figsize=(10, 4))
plt.plot(np.arange(len(signal)) / fs,
    signal)
plt.title("Original Signal (Time Domain)")
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.show()
```

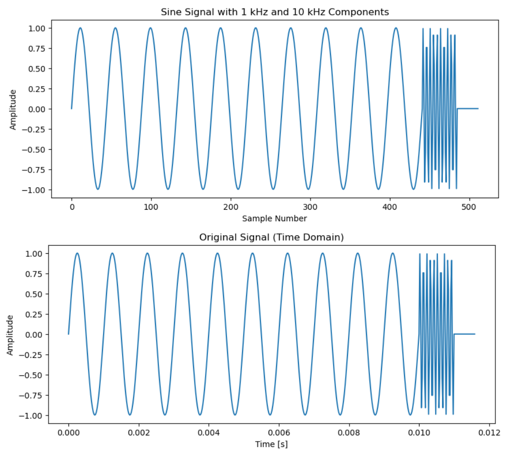

Fig. 3. Input signal for the multi-dimensional Haar Wavelet Transform analysis both with its sample number and time based representations

The corresponding multi-dimensional Haar Wavelet Transform has been conducted by the previously defined haar_wavelet() function as below.

```
level_1_result = haar_wavelet(signal , 1)
level_2_result = haar_wavelet(signal , 2)
level_3_result = haar_wavelet(signal , 3)

# Plot 1-Level Haar Transform
plt.figure(figsize=(12, 6))

# Plot 3-Level Haar Transform
plt.subplot(3, 1, 1)
plt.plot(level_1_result)

# Adding labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Value of Level 1 Haar Wavelet
    (Wavelet Domain)')

# Plot 2-Level Haar Transform
plt.subplot(3, 1, 2)
plt.plot(level_2_result)

# Adding labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Value of Level 2 Haar Wavelet
    (Wavelet Domain)')

# Plot 3-Level Haar Transform
plt.subplot(3, 1, 3)
plt.plot(level_3_result)

# Adding labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Value of Level 3 Haar Wavelet
    (Wavelet Domain)')


# Show the plots
plt.tight_layout()
plt.show()
```

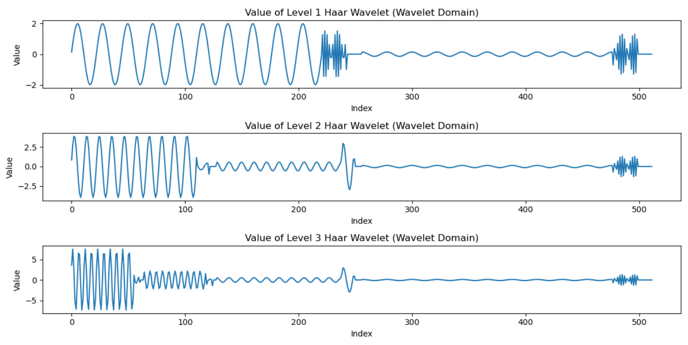

Fig. 4. Multi-level Haar Wavelet Transform analysis

The figure above illustrates the results of a multi-level Haar Wavelet Transform analysis applied to a one-dimensional signal. This analysis is represented in three distinct levels, with each subplot corresponding to a different level of decomposition, offering a progressively refined view of the signal in the wavelet domain.

The top subplot shows the wavelet coefficients after the first level of Haar Wavelet Transform. At this initial stage, the signal is decomposed into its approximation and detail components. The sharp transitions and high-frequency components of the signal are captured, especially visible in regions where the signal changes rapidly. This level provides a broad overview of the signal's structure, highlighting the most prominent features, including sharp peaks and sudden changes.

The middle subplot represents the signal after the second level of decomposition. Here, the approximation component from the first level is further decomposed, revealing more detailed information about the signal. The wavelet coefficients at this level show a smoother version of the original signal, with some of the finer details from the first level being more pronounced. This level of analysis begins to separate the signal into components that reveal underlying patterns and trends that were less obvious at the first level.

The bottom subplot depicts the third level of the Haar Wavelet Transform. At this stage, the analysis delves even deeper, focusing on the residual low-frequency components that characterize the overall trend of the signal. The result is a much smoother and more abstracted representation of the signal, where high-frequency details are largely suppressed, and the global structure is emphasized. This level is particularly useful for identifying long-term trends and patterns within the signal, providing a clearer view of the signal's behavior over time.

## IV. Comparison of Haar Wavelet Transform (HWT) and Discrete Fourier Transform (DFT)

In order to observe the main differences of the HWT and DFT, both analysis have been implemented for the same input signal and demonstrated by the code script below.

```
sample_rate = 44100

result = haar_wavelet(signal , 3)
dft, frequencies = just_dft(signal,
    sample_rate)

# Plot the original signal
plt.figure(figsize=(12, 6))

plt.subplot(3, 1, 1)
plt.plot(np.arange(len(signal)) /
    sample_rate, signal)
plt.title("Original Signal (Time Domain)")
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
```

```
# Plot the magnitude of the DFT
    (Frequency Spectrum)
plt.subplot(3, 1, 2)
plt.stem(frequencies[:len(signal) // 2],
    np.abs(dft)[:len(signal) // 2], 'b',
    markerfmt=" ", basefmt="-b")
plt.title("Magnitude of DFT (Frequency
    Domain)")
plt.xlabel("Frequency [Hz]")
plt.ylabel("Magnitude")

# Plot the value of the Haar Wavelet
    (Wavelet Spectrum)
# Plotting the data
plt.subplot(3, 1, 3)
plt.plot(result)

# Adding labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Value of Haar Wavelet (Wavelet
    Domain)')


# Show the plots
plt.tight_layout()
plt.show()
```

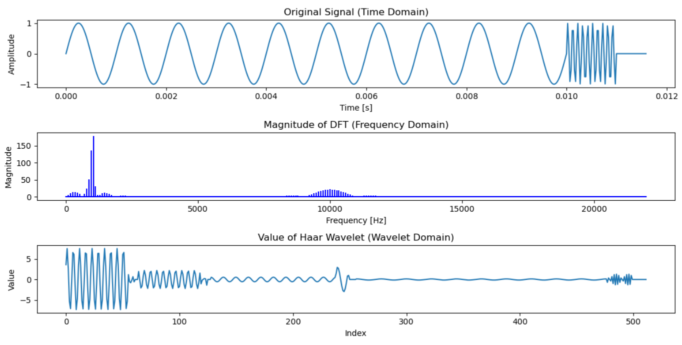

Fig. 5. Demonstration of the input signal and its corresponding DFT and HWT representations

In the Fig. 5. the Haar Wavelet Transformation is applied in the level of 3.
The top plot represents the original signal in the time domain. It displays the amplitude of the signal as it varies over time. The signal appears to be a combination of low-frequency oscillations with a section of higher frequency components towards the end of the time window. This time-domain representation allows us to see the overall structure of the signal, including its amplitude and temporal variations, but does not provide any direct information about the frequencies present in the signal.
The middle plot shows the magnitude spectrum of the signal as obtained through the Discrete Fourier Transform (DFT). This representation transforms the time-domain signal into its constituent frequencies, with the x-axis representing frequency in Hertz (Hz) and the y-axis representing the magnitude of these frequencies. The plot reveals two distinct frequency components, which are the 1kHz and 10kHz. The DFT provides a clear picture of the frequency content of the signal, but it lacks information about when these frequencies occur within the original signal. This makes it difficult to correlate the frequency components directly with specific events in the time domain.
The bottom plot presents the signal after applying the Haar Wavelet Transform, offering a view in the wavelet domain. The Haar Wavelet analysis reveals both the frequency and temporal localization of the signal's components.

### A. Inverse Haar Wavelet Transform

Unlike the Discrete Fourier Transform, the time-domain signal can be reconstructed from its wavelet transform by knowing the level of provided wavelet analysis. The idea is based on the simple mathematical definition based on the definition of the approximation and the detail coefficients that were given earlier int this paper.

approximation coefficient = A + B

detail coefficient = A - B

Then the reconstruction can be conducted as the following

A =

$$\frac{(approx.coefficient + detailcoefficient)}{2}$$

B =

$$\frac{(approx.coefficient - detailcoefficient)}{2}$$

### B. Python Code Implementation of the Reverse 1D Haar Wavelet Transform

As it was discussed before about the capability of reconstruction of the input signal from its corresponding wavelet transform, the python code script is implemented as the following.

```
# Define the Inverse Haar Wavelet
    Transform function for 1D Signals
def inverse_haar_wavelet(f,depth):
   if depth == 0:
      return f
   else:
      n2 = len(f)>>1
      h =
          inverse_haar_wavelet(f[:n2],depth-1)
      g = np.zeros_like(f)
      g[1::2] = (h + f[n2:])/2
      g[::2] = (h - f[n2:])/2
      return g
```

The function begins by checking the base case: if the depth is 0, the function simply returns the input signal f , as no further inverse transformation is necessary. This base case serves as the termination condition for the recursion. If the depth is greater than 0, the function proceeds with the inverse transformation process.
In the recursive decomposition, the signal f is split into two parts. The first half of the signal, f[:n2] , represents the approximation coefficients from the most recent level of the Haar Wavelet Transform. The second half, f[n2:] , contains the corresponding detail coefficients. The function then recursively calls itself on the approximation coefficients f[:n2] , decreasing the depth by 1 with each call. This recursive approach ensures that the function progressively reconstructs the signal from the highest level of detail down to the original signal.
After obtaining the approximation coefficients from the recursive call, the function initializes a new array g to store the reconstructed signal. The reconstruction is performed by combining the approximation and detail coefficients. Specifically, the odd-indexed elements of g are calculated by adding the detail coefficients to the approximation coefficients and then dividing by 2. Conversely, the even-indexed elements are computed by subtracting the detail coefficients from the approximation coefficients, also followed by division by 2. This process effectively reverses the Haar Wavelet Transform, reconstructing the signal in its original time-domain form.
Once the reconstruction is complete, the function returns the fully reconstructed signal g . The recursive structure of the function ensures that the signal is accurately reconstructed from its wavelet coefficients, level by level, thereby restoring the original signal's form.

The reconstructed signal from its corresponding Haar Wavelet Transform is demonstrated by the following script.

```
reconstructed_signal =
    inverse_haar_wavelet(result, 3)

# Plot the reconstructed signal
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 2)
plt.plot(np.arange(len(reconstructed_signal))
    / sample_rate, reconstructed_signal)
plt.title("Reconstructed Signal (Time
    Domain)")
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")

# Plot the value of the Haar Wavelet
    (Wavelet Spectrum)
# Plotting the data
plt.subplot(2, 1, 1)
plt.plot(result)

# Adding labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Value of Haar Wavelet (Wavelet
    Domain)')


# Show the plots
plt.tight_layout()
plt.show()
```

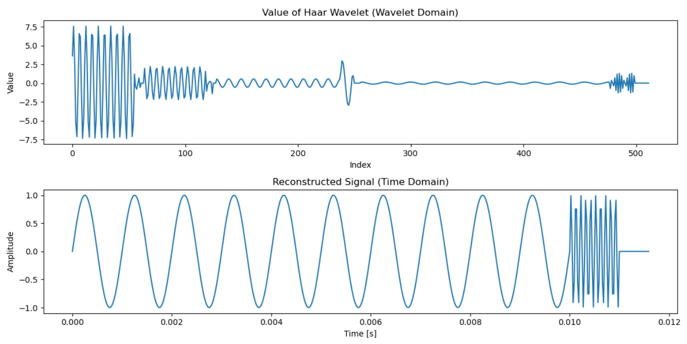

Fig. 6. Reconstructed signal from its corresponding Haar Wavelet Transform

As it can be seen from the Fig. 6., the reconstructed time-domain signal is a perfect copy of the initial input signal.

## V. Conclusion

In this paper, comprehensive analysis of the Discrete Fourier Transform (DFT) and the Haar Wavelet Transform (HWT), particularly focusing on their application to one-dimensional (1D) audio signal processing are conducted. The DFT, while powerful for frequency analysis, was shown to have limitations due to its lack of time localization, which restricts its effectiveness in handling non-stationary or transient signals. In contrast, the Haar Wavelet Transform demonstrated its superiority by providing both time and frequency localization, making it an ideal tool for applications where the timing of frequency components is critical.
Through the implementation and comparison of these two techniques, it is highlighted that the advantages of the Haar Wavelet Transform to reconstruct the original signal. This reconstructability, combined with its multi-level decomposition capability, underscores the HWT's suitability for advanced signal processing tasks such as audio compression, noise reduction, and feature extraction.
The obtained results suggest that the Haar Wavelet Transform is a more versatile and effective method than the Discrete Fourier Transform for many real-world signal processing applications, particularly those involving signals with varying frequency components over time. The Python implementations provided in this paper offer a practical guide for researchers and engineers to apply these techniques in their own work, further demonstrating

the utility of the HWT in modern digital signal processing.

Future work could explore the application of other wavelet functions, such as Daubechies or Coiflets, in similar contexts, as well as the potential for real-time processing of signals using wavelet transforms. Additionally, extending this analysis to two-dimensional signals, such as images, could provide further insights into the broader applicability of wavelet-based methods.

## APPENDIX

[ 1 ] https://github.com/gokcerSnmzck/G-Wavelet-Transform

## REFERENCES

[1] Kirsanov, A. (2022, August 15). Artem Kirsanov. https://www.youtube.com/watch?v=jnxqHcObNK4

[2] Calder, J. (2020). The Discrete Wavelet Transform. In Math 5467 – Intro to the Mathematics of Image and Data Analysis. Minnesota; Jeff Calder. Retrieved August 30, 2024, from https://www-users.cse.umn.edu/˜jwcalder/5467/lec_dwt.pdf.

[3] Wikimedia Foundation. (2024, August 27). Discrete Fourier Transform. Wikipedia. https://en.wikipedia.org/wiki/Discrete_Fourier_transform

[4] I. Ram, M. Elad and I. Cohen, ”Generalized Tree-Based Wavelet Transform,” in IEEE Transactions on Signal Processing, vol. 59, no. 9, pp. 4199-4209, Sept. 2011, doi: 10.1109/TSP.2011.2158428.

[5] T. V. Pham and G. Kubin, ”Comparison between DFT- and DWT-based speech/non-speech detection for adverse environments,” The 2011 International Conference on Advanced Technologies for Communications (ATC 2011), Da Nang, Vietnam, 2011, pp. 299-302, doi: 10.1109/ATC.2011.6027490.