TECHNICAL UNIVERSITY OF MOLDOVA

FACULTY OF COMPUTERS INFORMATICS

ANDMICROELECTRONICS

DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION

Laboratory work 2

Subject: The convolution of two sequences and its properties

Done by: **Konjevic Alexandra,**
st. gr. FAF-213

Verified by: **Railean Serghei,**
university lecturer

CHIŞINĂU, 2024

TASK OF THE LABORATORY WORK

The scope of this lab work encompasses a comprehensive exploration of signal processing concepts, convolution operations, Fourier transforms, and system modeling using block diagrams. It involves a series of tasks designed to deepen understanding and practical application of these fundamental concepts.

Initially, the lab begins with the generation of two finite sequences (a(n)) and (b(n)) with specific lengths, typically 5 and 4, respectively. These sequences are then visualized using stem plots, providing a tangible representation of the data. Subsequently, the lab delves into the convolution operation, a fundamental mathematical operation in signal processing. Utilizing the convolve function, the convolution of the generated sequences is computed. The resulting discrete signal is displayed using stem plots, facilitating visual analysis.

Moving forward, the lab explores the Fourier transform of the sequences (a(n)) and (b(n)). By calculating the product of the obtained transforms, the convolution of the signals is determined. This convolutional signal is visualized in discrete form, enabling examination of its frequency domain characteristics. Furthermore, the lab involves the calculation of the inverse Fourier transform of the product of the Fourier transforms of (a(n)) and (b(n)).

Error analysis is conducted to compare the obtained convolution with the initial convolution, facilitating the assessment of accuracy and identifying discrepancies. This involves calculating the error between the two convolution signals and visualizing them alongside the original convolution signal.

The lab extends its scope to evaluate the performance of convolution operations for larger signals. This entails assessing the difference in time between direct convolution and convolution using the Fast Fourier Transform (FFT) algorithm. Additionally, the time taken for block-wise convolution is calculated to understand its computational efficiency. Further evaluation is conducted by repeating the performance assessment for even larger signals, providing insights into the scalability and efficiency of convolution algorithms.

Block-wise convolution is explored as an alternative approach, involving the division of the input signal into two blocks. The convolution for each block is then computed separately to assess the efficacy of this method.

Finally, the lab culminates with the determination of the final convolution by combining the convolutions of the two blocks as specified. The resulting convolution signal is visualized in discrete form, facilitating a comprehensive understanding of its characteristics.

THEORETICAL CONSIDERATIONS

Linear System Analysis Techniques

Linear systems can be analyzed using two main techniques. The first method involves solving the system's input-output equation directly, while the second method decomposes the input signal into a sum of elementary signals to determine the system's response.

In the second method, the input signal is expressed as a weighted sum of unit sample sequences. These sequences are chosen so that the system's response to each is easily determined. By summing the system's responses to these elementary signals, the total response of the system to the input signal can be found.

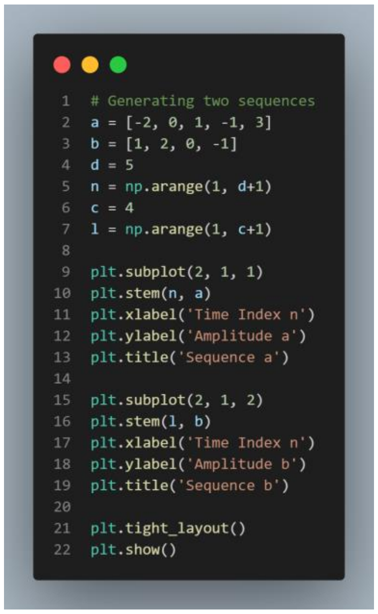
Signal Decomposition into Elementary Signals

To decompose an arbitrary signal into a sum of unit sample sequences, each unit sample is multiplied by the corresponding value of the signal. This places the value of the signal at the location where the unit sample is nonzero. By repeating this process for all possible shifts, the signal can be decomposed into a sum of sequences, allowing for analysis of the system's response to each component.

These techniques provide insights into how linear systems respond to different input signals, aiding in understanding their behavior and properties.

IMPLEMENTATIONS AND DIAGRAMS

1 Generate two finite sequences a(n) and b(n) of lengths 4 and 5.



```
# Generating two sequences
a = [-2, 0, 1, -1, 3]
b = [1, 2, 0, -1]
d = 5
n = np.arange(1, d+1)
c = 4
l = np.arange(1, c+1)

plt.subplot(2, 1, 1)
plt.stem(n, a)
plt.xlabel('Time Index n')
plt.ylabel('Amplitude a')
plt.title('Sequence a')

plt.subplot(2, 1, 2)
plt.stem(l, b)
plt.xlabel('Time Index n')
plt.ylabel('Amplitude b')
plt.title('Sequence b')

plt.tight_layout()
plt.show()
```

Figure 1a. Generation of two sequences

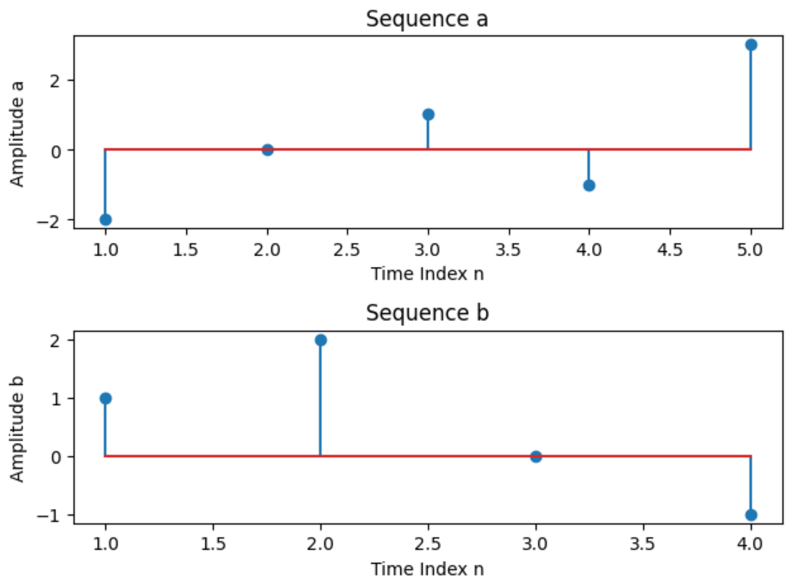
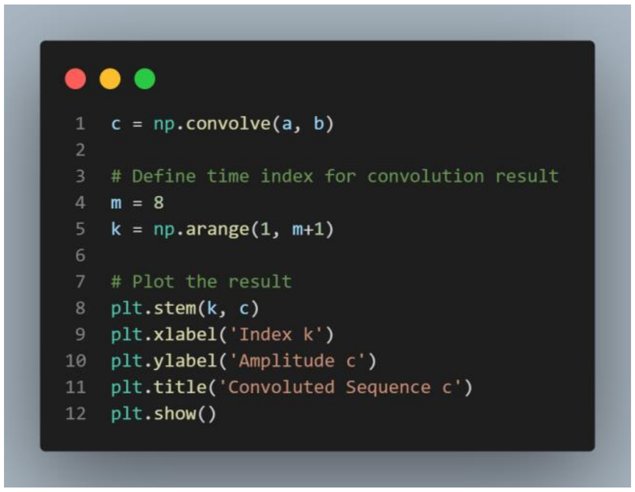


Figure 1b. Sequencs a and b diagrams

2 Convolution of a(n) and b(n).



```
c = np.convolve(a, b)

# Define time index for convolution result
m = 8
k = np.arange(1, m+1)

# Plot the result
plt.stem(k, c)
plt.xlabel('Index k')
plt.ylabel('Amplitude c')
plt.title('Convoluted Sequence c')
plt.show()
```

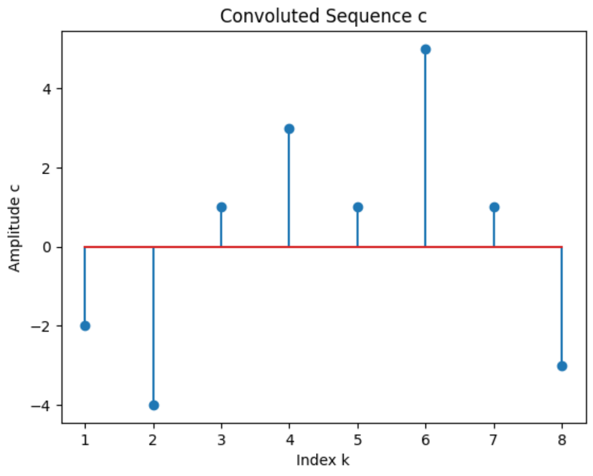
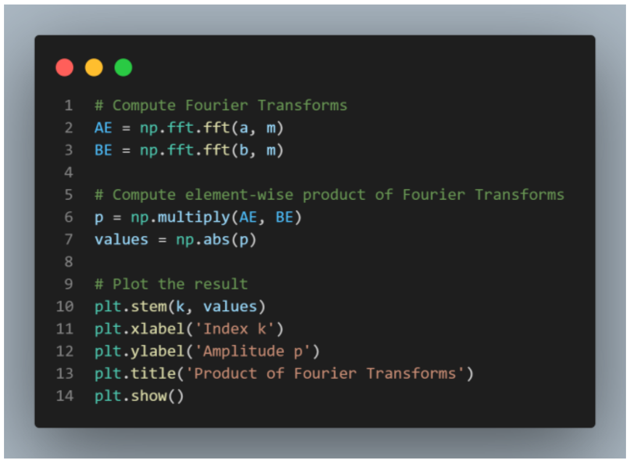


Figure 2a. Convolution of a(n) and b(n) ***Figure 2b. Diagram of convolution***

3 Fourier Transform of the convolution of a(n) and b(n).



```
# Compute Fourier Transforms
AE = np.fft.fft(a, m)
BE = np.fft.fft(b, m)

# Compute element-wise product of Fourier Transforms
p = np.multiply(AE, BE)
values = np.abs(p)

# Plot the result
plt.stem(k, values)
plt.xlabel('Index k')
plt.ylabel('Amplitude p')
plt.title('Product of Fourier Transforms')
plt.show()
```

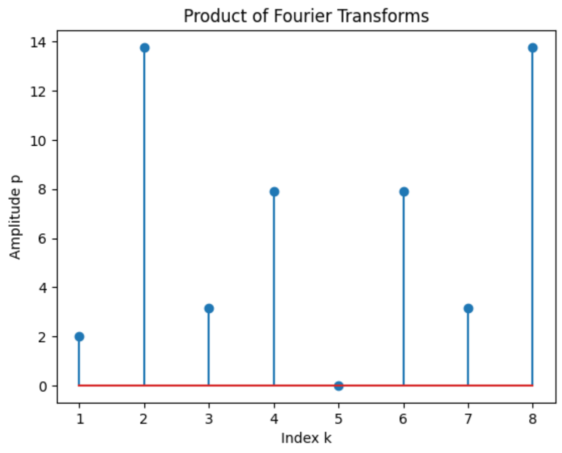
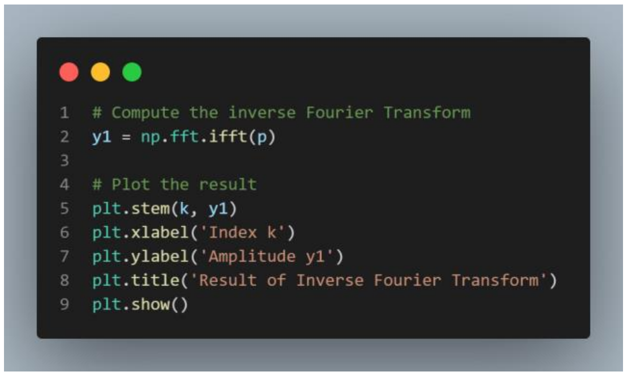


Figure 3a. Fourier Transform of the convolution ***Figure 3b. Diagram of Fourier tr. of the conv.***

4 Inverse Fourier transform of the product from the previous point



```
# Compute the inverse Fourier Transform
y1 = np.fft.ifft(p)

# Plot the result
plt.stem(k, y1)
plt.xlabel('Index k')
plt.ylabel('Amplitude y1')
plt.title('Result of Inverse Fourier Transform')
plt.show()
```

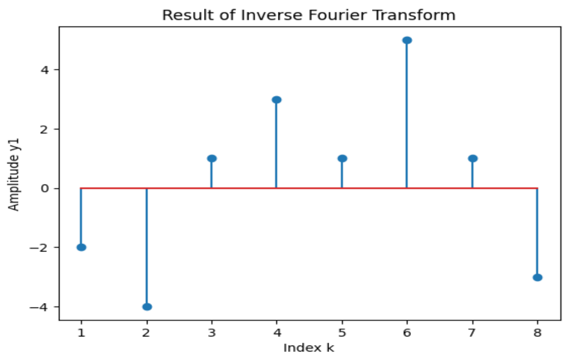
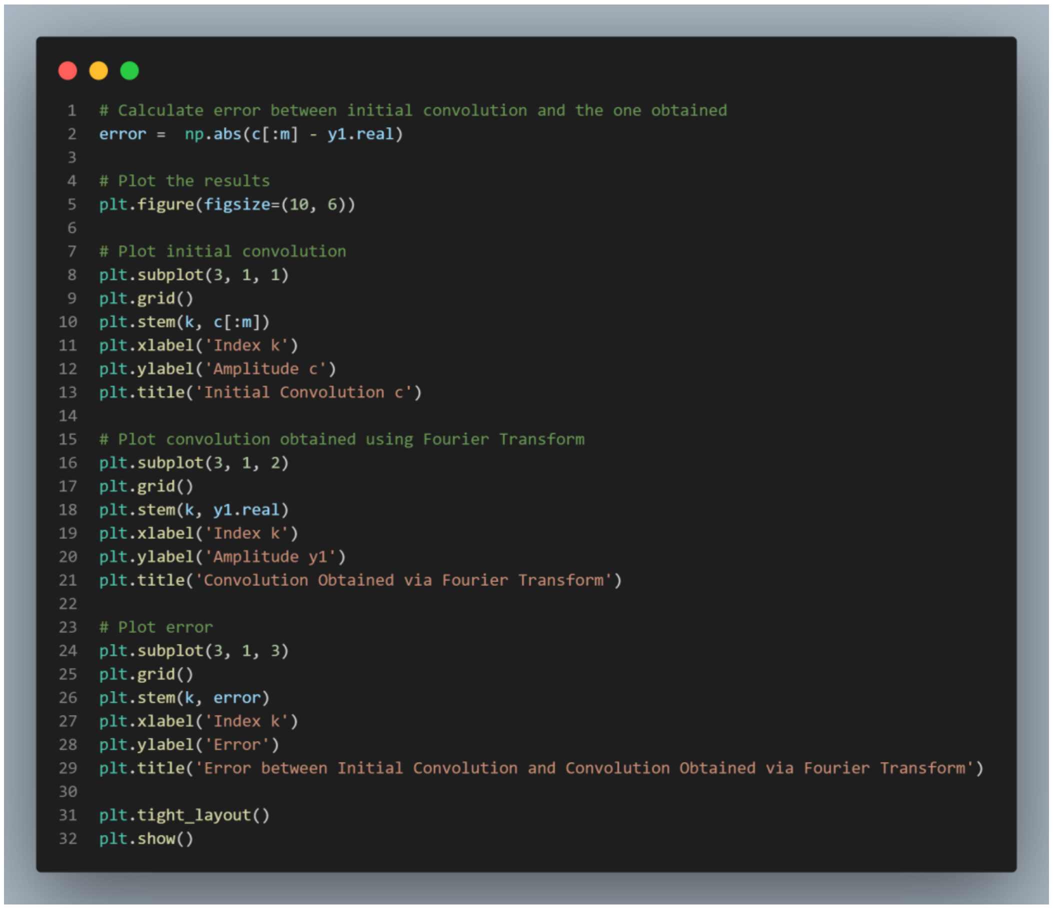


Figure 4a. Inverse Fourier transform ***Figure 4b. Inverse Fourier transform diagram***

5 Error between the original convolution and the inverse Fourier transform.



```python
# Calculate error between initial convolution and the one obtained
error =  np.abs(c[:m] - y1.real)

# Plot the results
plt.figure(figsize=(10, 6))

# Plot initial convolution
plt.subplot(3, 1, 1)
plt.grid()
plt.stem(k, c[:m])
plt.xlabel('Index k')
plt.ylabel('Amplitude c')
plt.title('Initial Convolution c')

# Plot convolution obtained using Fourier Transform
plt.subplot(3, 1, 2)
plt.grid()
plt.stem(k, y1.real)
plt.xlabel('Index k')
plt.ylabel('Amplitude y1')
plt.title('Convolution Obtained via Fourier Transform')

# Plot error
plt.subplot(3, 1, 3)
plt.grid()
plt.stem(k, error)
plt.xlabel('Index k')
plt.ylabel('Error')
plt.title('Error between Initial Convolution and Convolution Obtained via Fourier Transform')

plt.tight_layout()
plt.show()
```

Figure 5a. Determining the error and drawing the three diagrams

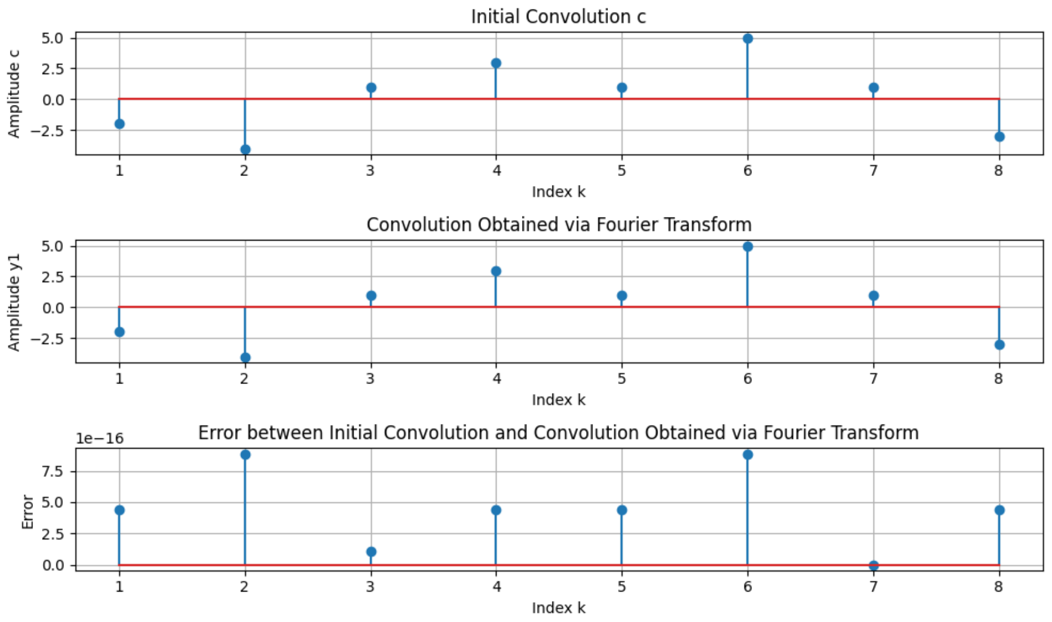
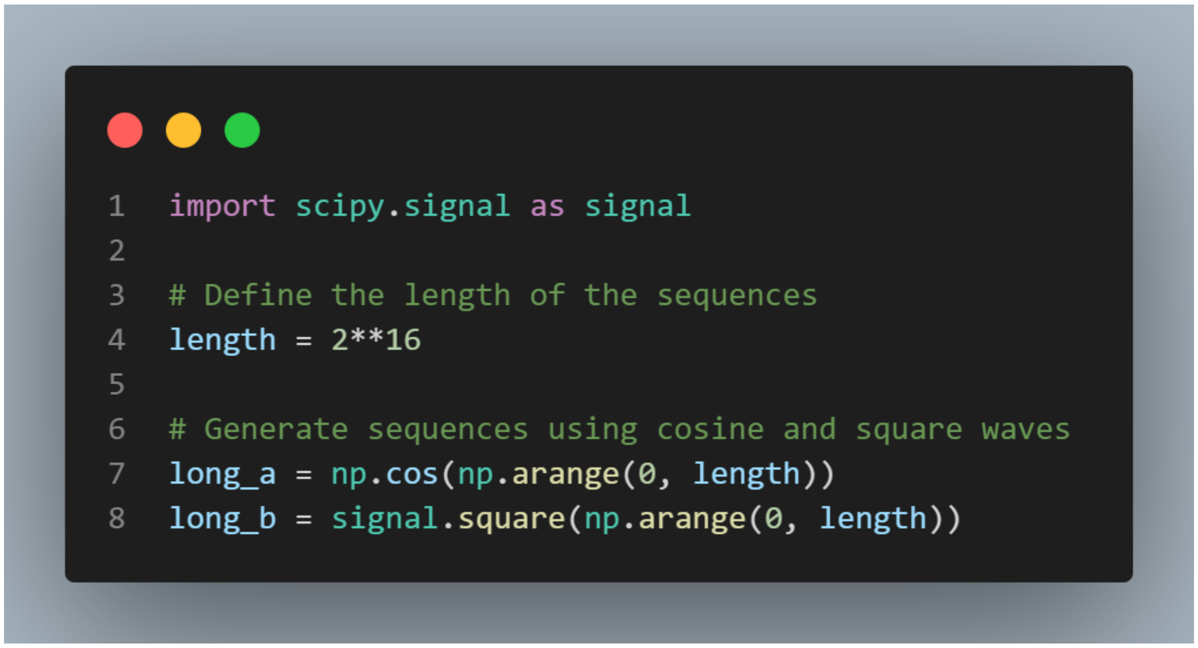


Figure 5b. Initial convolution, Fourier transform convolution and error

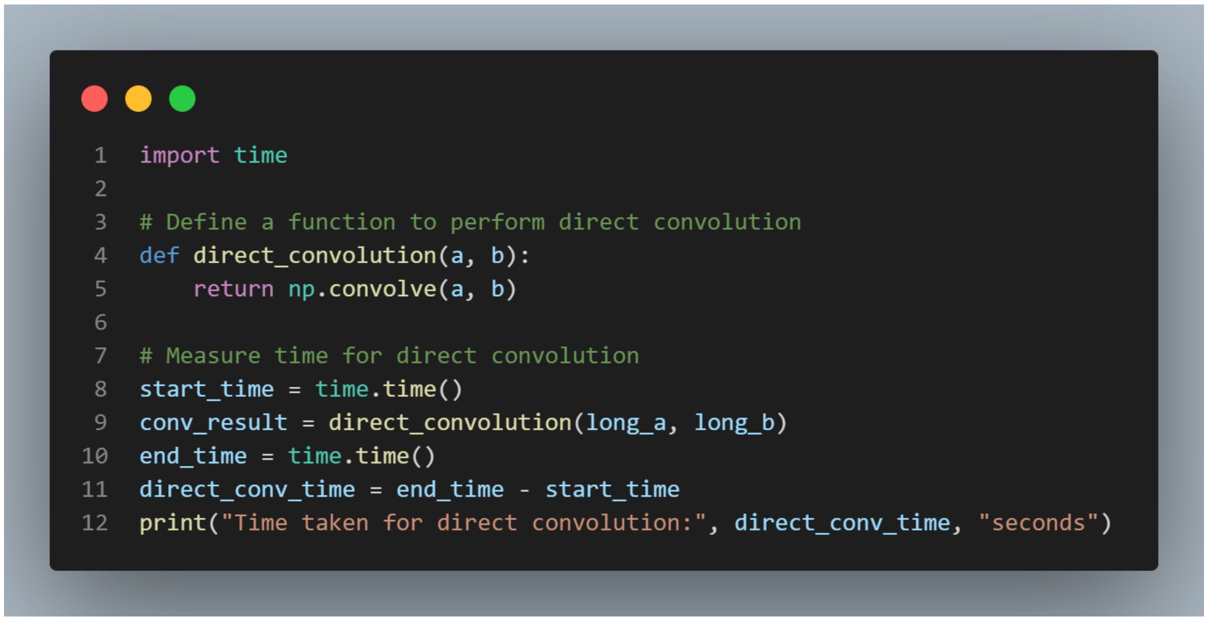
6.1 Sequences of large lengths.



```python
import scipy.signal as signal

# Define the length of the sequences
length = 2**16

# Generate sequences using cosine and square waves
long_a = np.cos(np.arange(0, length))
long_b = signal.square(np.arange(0, length))
```

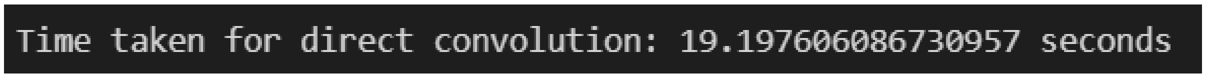
Figure 6. Initializing two sequences of large lengths

6.2 Execution time of the direct convolution.



```python
import time

# Define a function to perform direct convolution
def direct_convolution(a, b):
    return np.convolve(a, b)

# Measure time for direct convolution
start_time = time.time()
conv_result = direct_convolution(long_a, long_b)
end_time = time.time()
direct_conv_time = end_time - start_time
print("Time taken for direct convolution:", direct_conv_time, "seconds")
```

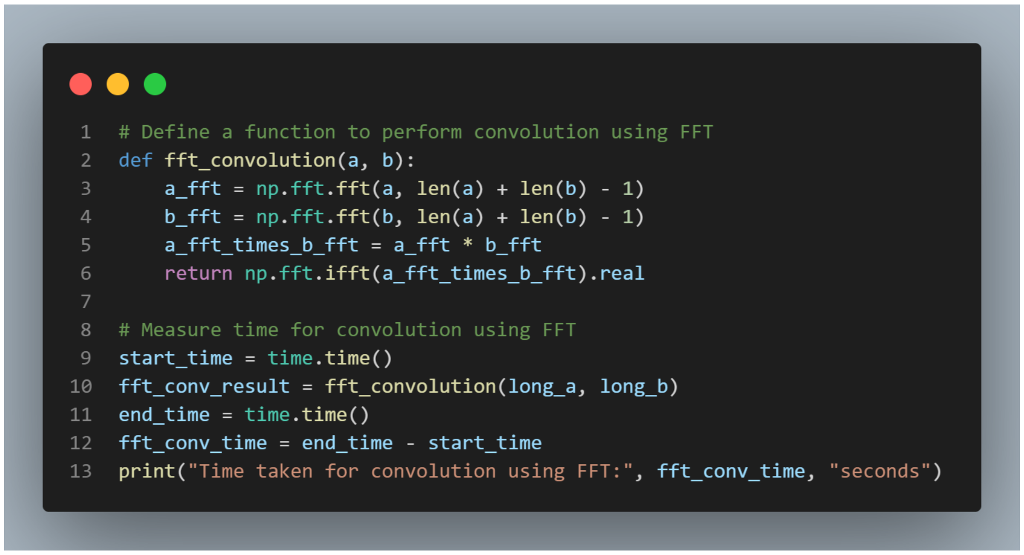
Figure 7a. Program to determine the execution time of direct convolution



```
Time taken for direct convolution: 19.197606086730957 seconds
```

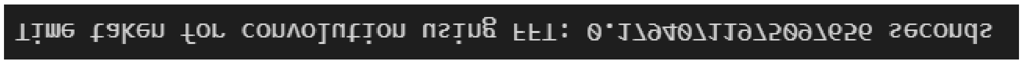
Figure 7b. Result of the program above

6.3 Execution time of the Fourier Transform convolution.



```python
# Define a function to perform convolution using FFT
def fft_convolution(a, b):
    a_fft = np.fft.fft(a, len(a) + len(b) - 1)
    b_fft = np.fft.fft(b, len(a) + len(b) - 1)
    a_fft_times_b_fft = a_fft * b_fft
    return np.fft.ifft(a_fft_times_b_fft).real

# Measure time for convolution using FFT
start_time = time.time()
fft_conv_result = fft_convolution(long_a, long_b)
end_time = time.time()
fft_conv_time = end_time - start_time
print("Time taken for convolution using FFT:", fft_conv_time, "seconds")
```

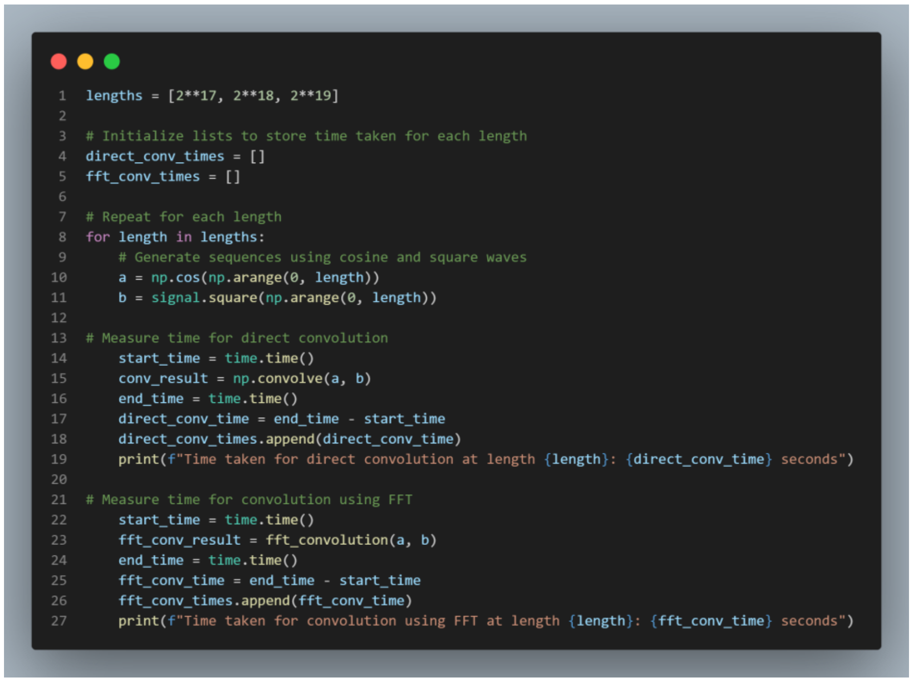
Figure 8a. Program to determine the execution time of Fourier Transform convolution



```
Time taken for convolution using FFT: 0.126402116125062626 seconds
```

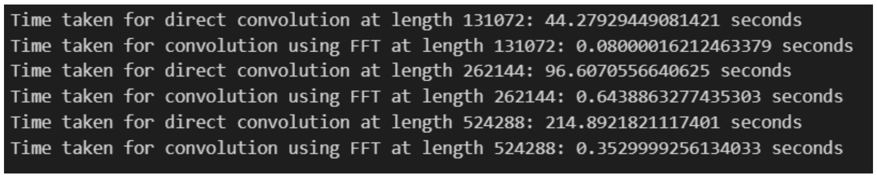
Figure 8b. Result of the program above

7 Execution times of the direct convolution and the Fourier Transform for sequences of lengths 2^17, 2^18, 2^19



```python
lengths = [2**17, 2**18, 2**19]

# Initialize lists to store time taken for each length
direct_conv_times = []
fft_conv_times = []

# Repeat for each length
for length in lengths:
    # Generate sequences using cosine and square waves
    a = np.cos(np.arange(0, length))
    b = signal.square(np.arange(0, length))

# Measure time for direct convolution
    start_time = time.time()
    conv_result = np.convolve(a, b)
    end_time = time.time()
    direct_conv_time = end_time - start_time
    direct_conv_times.append(direct_conv_time)
    print(f"Time taken for direct convolution at length {length}: {direct_conv_time} seconds")

# Measure time for convolution using FFT
    start_time = time.time()
    fft_conv_result = fft_convolution(a, b)
    end_time = time.time()
    fft_conv_time = end_time - start_time
    fft_conv_times.append(fft_conv_time)
    print(f"Time taken for convolution using FFT at length {length}: {fft_conv_time} seconds")
```

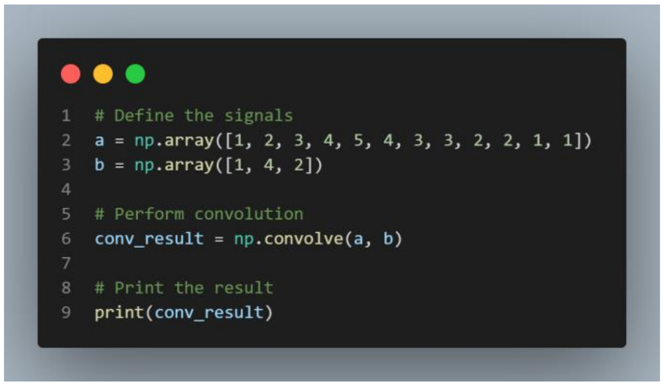
Figure 9a. Program to measure the execution time for larger lengths



```
Time taken for direct convolution at length 131072: 44.27929449081421 seconds
Time taken for convolution using FFT at length 131072: 0.08000016212463379 seconds
Time taken for direct convolution at length 262144: 96.6070556640625 seconds
Time taken for convolution using FFT at length 262144: 0.6438863277435303 seconds
Time taken for direct convolution at length 524288: 214.8921821117401 seconds
Time taken for convolution using FFT at length 524288: 0.3529999256134033 seconds
```

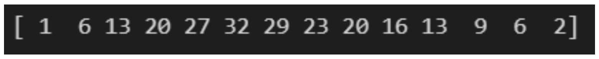
Figure 9b. Execution times for three different sequences

8 Convolution of two signals a and b.



```python
# Define the signals
a = np.array([1, 2, 3, 4, 5, 4, 3, 3, 2, 2, 1, 1])
b = np.array([1, 4, 2])

# Perform convolution
conv_result = np.convolve(a, b)

# Print the result
print(conv_result)
```

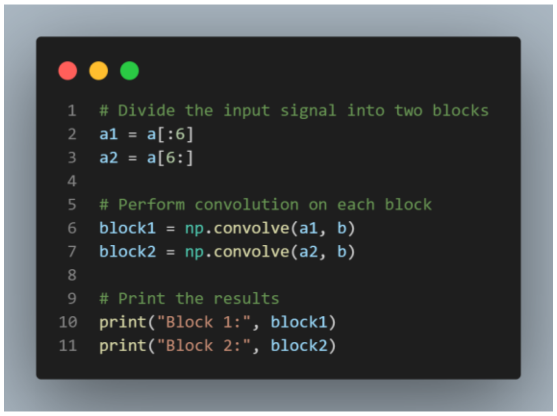
Figure 10a. Convolution of a and b



```
[ 1  6 13 20 27 32 29 23 20 16 13  9  6  2]
```

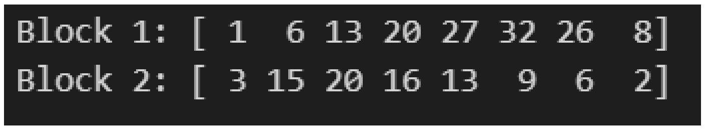
Figure 10b. Result of the convolution

9 Convolution of signals divided in blocks.



```python
# Divide the input signal into two blocks
a1 = a[:6]
a2 = a[6:]

# Perform convolution on each block
block1 = np.convolve(a1, b)
block2 = np.convolve(a2, b)

# Print the results
print("Block 1:", block1)
print("Block 2:", block2)
```

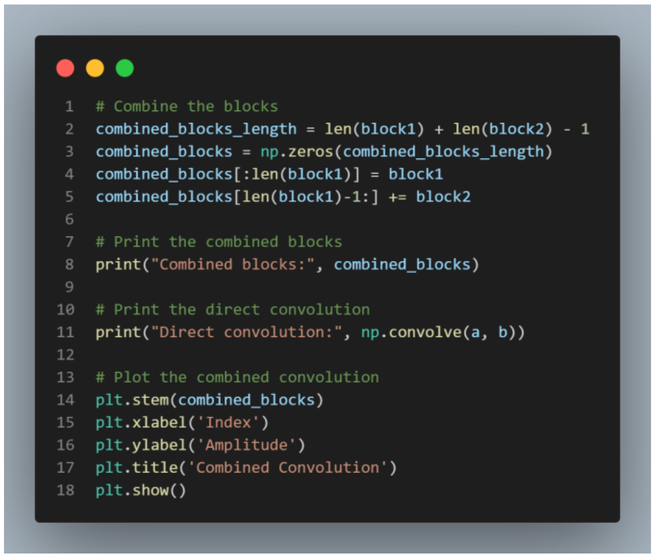
Figure 11a. Program for the convolution of each block



```
Block 1: [ 1  6 13 20 27 32 26  8]
Block 2: [ 3 15 20 16 13  9  6  2]
```

Figure 11b. Result of the convolution of each block

10 Combined convolution of signals divided in blocks.



```python
# Combine the blocks
combined_blocks_length = len(block1) + len(block2) - 1
combined_blocks = np.zeros(combined_blocks_length)
combined_blocks[:len(block1)] = block1
combined_blocks[len(block1)-1:] += block2

# Print the combined blocks
print("Combined blocks:", combined_blocks)

# Print the direct convolution
print("Direct convolution:", np.convolve(a, b))

# Plot the combined convolution
plt.stem(combined_blocks)
plt.xlabel('Index')
plt.ylabel('Amplitude')
plt.title('Combined Convolution')
plt.show()
```

Figure 12a. Program to combine the convolutions

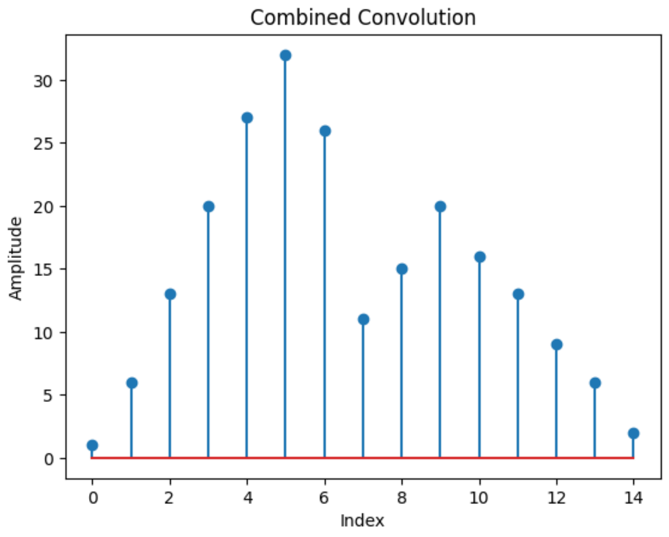


Figure 12b. Diagram of combined blocks

CONTROL QUESTIONS

1. Discrete signals are represented as a sum of elementary signals by decomposing the input signal into a weighted sum of elementary signals, each scaled by a coefficient. These elementary signals are chosen such that the system's response to each elementary signal is easily determined. Utilizing the linearity property of the system, the responses to individual elementary signals are then summed to find the total system response to the input signal. This process allows for a systematic analysis of the system's behavior and its response to complex input signals.
2. The elementary signal involved in the calculation of convolution is the unit impulse signal, also known as the Dirac delta function. It is represented by δ(n), and its shifted versions δ(n - k) are used in the convolution sum to compute the response of the system to arbitrary input signals.
3. Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Using the strategy of impulse decomposition, systems are described by a signal called the impulse response. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.
4. The steps in computing convolution are as follows:
   - Flip: One of the signals involved in convolution is flipped. Typically, this signal is the impulse response.

- Shift: The flipped signal is shifted left or right, corresponding to each sample of the other signal.
- Multiply: Each sample of the unflipped signal is multiplied with the corresponding sample of the flipped and shifted signal.
- Summation: The results of the multiplications are summed to obtain the output samples, which represent the convolution of the two signals.

5. The correlation between two signals is a measure of their similarity or relationship. It involves sliding one signal over the other and computing the integral of their product at each shift. The correlation function indicates how much the two signals resemble each other at different shifts. If the signals are similar, the correlation will be large; if they are dissimilar, the correlation will be small.
6. The main difference between convolution and correlation lies in their fundamental operations and the interpretations of their results:
   - Operation:
     - Convolution: In convolution, one signal (referred to as the input or impulse response) is flipped and shifted over another signal (referred to as the input or stimulus), and the overlapping areas are multiplied and summed.
     - Correlation: In correlation, one signal is slid over another, and at each shift, the product of the overlapping areas of the two signals is summed. The second signal is not flipped as in convolution.
   - Interpretation:
     - Convolution: Convolution is commonly used in signal processing to model the response of linear time-invariant systems to inputs. It represents the output of a system given an input signal and the system's impulse response.
     - Correlation: Correlation measures the similarity between two signals. Positive correlation indicates similarity, negative correlation indicates dissimilarity, and zero correlation indicates no relationship between the signals.

   In summary, while both convolution and correlation involve sliding one signal over another and computing the sum of the overlapping areas, convolution is primarily used to model system responses, while correlation is used to measure similarity between signals. Additionally, the operation of convolution involves flipping one of the signals, while correlation does not.

CONCLUSION

In conclusion, this lab work provided a comprehensive exploration of fundamental signal processing concepts, convolution operations, Fourier transforms, and system modeling using block diagrams. Through a series of tasks, we deepened our understanding and practical application of these concepts, starting with the generation and visualization of finite sequences (a(n)) and (b(n)).

The convolution operation, a cornerstone of signal processing, was thoroughly investigated using the convolve function. We computed convolutions of generated sequences and visualized the resulting discrete signals, enabling a detailed analysis. Fourier transforms of the sequences were then explored, leading to the determination of the convolution of the signals in the frequency domain.

Error analysis was employed to assess the accuracy of our convolutional results, revealing valuable insights into potential discrepancies. Furthermore, the performance of convolution operations was evaluated for larger signals, comparing direct convolution with FFT-based approaches. Block-wise convolution was also investigated, offering an alternative method for computation.

The final convolution, obtained through a combination of convolutions from divided blocks, provided a conclusive understanding of the convolutional process. Through visualizations and analysis, we gained insights into the scalability and efficiency of convolution algorithms.

Overall, this laboratory work not only enhanced our theoretical knowledge but also provided practical experience in signal processing techniques. It underscored the importance of convolution operations in various applications and laid a solid foundation for further exploration in the field of digital signal processing.