**TECHNICAL UNIVERSITY OF MOLDOVA**
**FACULTY OF COMPUTERS INFORMATICS AND MICROELECTRONICS**
**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION**

# PS Individual Work №2

## Convolution of two sets and its properties

Done by: **Cristian Brinza**
st. gr. FAF-212

Verified by: **S. RAILEAN**
dr., conf. univ.

**Chișinău, 2024**

**The purpose of work:** Study the notion of convolution, its properties and different approaches to the problem of its calculation.

## BRIEF THEORY

There are two main approaches to analyzing linear systems to determine their response to a given input signal. The first approach involves directly solving the system's input-output equation. The second approach entails breaking down the input signal into a combination of elementary signals. These elementary signals are selected such that the system's response to each one is straightforward to determine. By leveraging the system's linearity, we can then sum the responses to these elementary signals to obtain the overall response to the original input. This method involves decomposing any arbitrary signal x(n) into a series of scaled and shifted unit impulses, allowing us to calculate the system's response to any input signal.

The expression from above shows the response *y(n)* of the LTI system as a function of the input signal *x(n)* and the unit impulse response *h(n)* and this expression is called the convolution sum. The input *x(n)* is said to be convoluted with the response *h(n)* to find the response *y(n)*.

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

## IMPLEMENTATION

This chapter contains the listings of the code written and the visual representation of the results obtained during this individual work realization in the form of different graphs and diagrams.

The first way of computing the convolution sum is the direct computation solely based on the formula that was mentioned in the brief theory chapter (in the computational package used is implemented by the function - *conv()*).

Direct comp. of conv. of sets a and b

```
% Define arrays a and b
a = [-2, 0, 1, -1, 3];
b = [1, 2, 0, -1];

% Define n and l for indexing
n = 1:1:5;
l = 1:1:4;

% First figure: plotting sequences a and b
figure(1)
```

And here is the result of convolving the sets *a* and *b*.

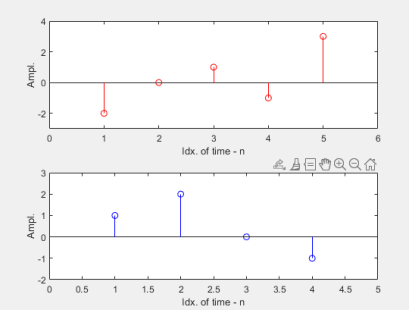



**Figure 1.** *Sets to be convolved*

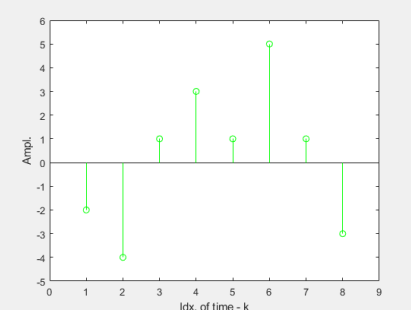



**Figure 2.** *Resulting convolution*

The second approach for calculating the convolution sum of two sequences utilizes the Fast Fourier Transform (FFT). This technique takes advantage of the FFT algorithm's efficiency to carry out the desired operation. The process involves converting both sequences *x(n)* and *h(n)* into the frequency domain via FFT, performing point-wise multiplication of the transformed sequences,

and then applying the inverse FFT (IFFT) to transform the result back into the time domain.And here are some results of using FFT for comp. convolution sum of the sets *a* and *b*.

```
AE = fft(a, m);
BE = fft(b, m);
p = AE .* BE;

stem(k, abs(p), 'LineStyle', '-', 'Marker', 'o', 'Color', 'm')
xlabel('Idx. of time - k'); ylabel('Ampl.')
axis([0, length(p) + 1, min(abs(p)) - 1, max(abs(p)) + 1])
min(c) - 1, max(c) + 1])
```

**Listing 2.** *FFT comp. of conv. of sets a and b*

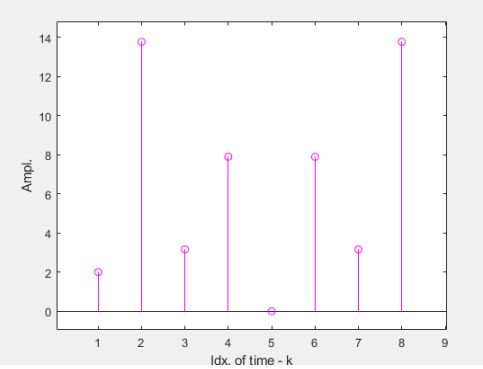
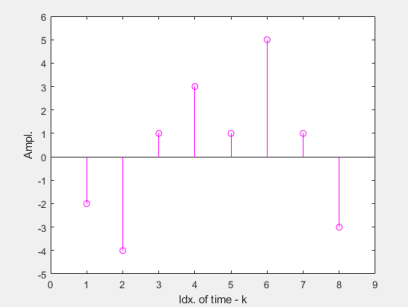



**Figure 3.** The *product of FF-Transformed a and b*

This set of values happens to be equal to the Fourier Transform of the convolution of signals *a(n)* and *b(n)*.

So by determining the Inverse Fourier Transform of the product of the Fourier Transforms of the sequences *a(n)* and *b(n)*, we obtain the convolution of these signals.

Below, you may find a comparison of the results and possible error estimations.



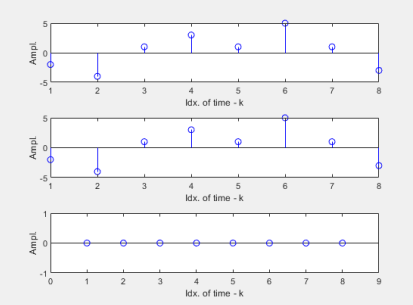




**Figure 4.** *FFT computed convolution of signals a(n), b(n)*

**Figure 5. a)** *Directly comp. conv.* **b)** *FFT comp. conv.* **c)** *Difference*

At this scale the effectiveness of the FFT approach is not that visible so let's increase the size of original signals up to 65536 samples. The first signal will be a square curve and the second - a sawtooth wave. For measuring the elapsed time the *tic* and *toc* function will be used.

```
n = 1:1:65536;
a = 2 * square(20 * pi * n + 1);
b = 3 * sawtooth(20 * pi * n + 1);
m = length(a) + length(b) - 1;

% Time convolution and FFT operations
tic
dc = conv(a, b);
toc

tic
AE = fft(a, m);
BE = fft(b, m);
p = AE .* BE;
fftc = ifft(p);
toc
```

**Listing 3.** *Comparison of approaches on a larger scale*

Naturally, the first approach (direct) did it in 9.3 seconds and the second (FFT) was almost 20 times faster with a time of 0.4 seconds. Here are some plots of the data considered during this "test" (not very depictive because of the scale, but it is what it is).

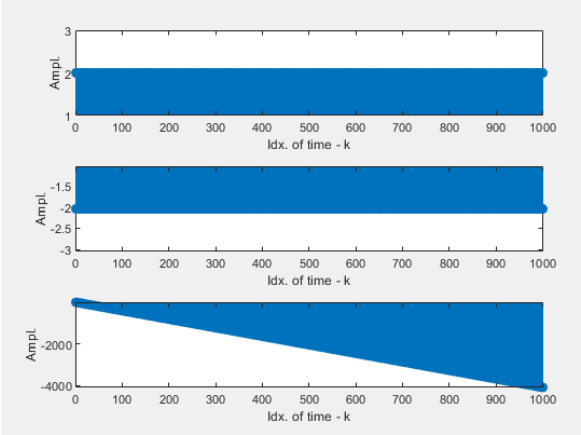



**Figure 6.** *FFT convolution of larger inputs*

For two inputs of size 131072 (convolution length equals to 262143), the growth in efficiency is even higher: 32 seconds shown by the trivial solution and 320 times higher performance of 0.1 seconds by the FFT method.

In some applications it is necessary to perform convolution of the signal as it forms. In this case, the signal is divided into blocks and block convolution is performed.

```
n = 1:1:131072;
l = 1:1:131072;
a = 2 * square(20 * pi * n + 1);
b = 3 * sawtooth(20 * pi * l + 1);
m = length(a) + length(b) - 1;
tic
dc = conv(a, b);
toc

tic
AE = fft(a, m);
BE = fft(b, m);
p = AE .* BE;
fftc = ifft(p);
toc
```

**Listing 4.** *Block convolution*

From the results you can see that the method is absolutely valid.

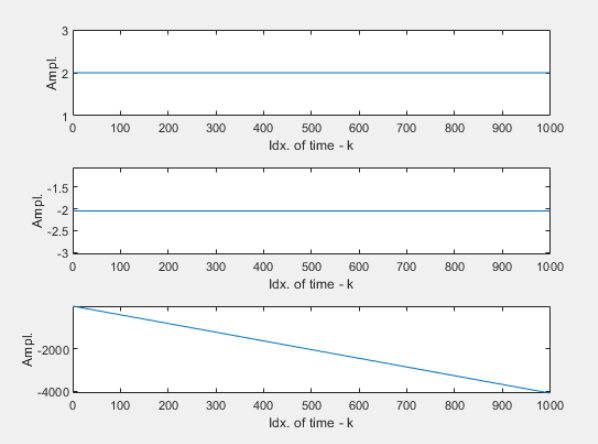



**Figure 7.**

**a)** *Directly comp. conv.*

**b)** *Block comp. conv.*

**c)** *Difference*

Let's apply this knowledge to design two systems that will receive two signals - *a* & *b* and return the convolution of these signals.

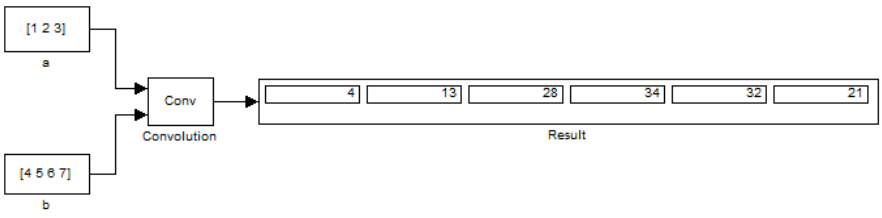



**Figure 8.** *Direct computation system*

For a system using FFT, for the algorithm itself to work, it is necessary to expand the input sequence to a size that is a multiple of 8.

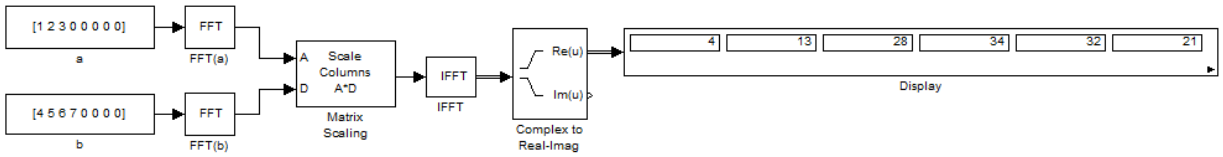



**Figure 9.** *FFT computation system*

# CONCLUSION

During this lab session, we examined several techniques for convolving two signals, emphasizing direct computation, FFT-based convolution, and block convolution. Notably, the FFT method improves computational efficiency, with performance gains that scale linearly with the size of the input data, making it superior to simpler methods. Block convolution, on the other hand, is advantageous for real-time signal processing as it allows convolution to be performed concurrently with signal generation. Engaging in the design and implementation of systems utilizing these algorithms provided hands-on experience, deepening my knowledge and skills in signal processing.