1. SHORT INTRODUCTION TO MATLAB

1.1 THE AIM OF THE EXERCISE

MATLAB application is a software package that can be used to calculate both, advanced and simple technical calculations. The application is designed to calculate short command linage from the level of user's keyboard, as well as complex m-scripts. It facilitates adding new functions as well as those that already exist and are called toolbox.
The easiest way of using MATLAB is typing an equation and commands in the command window. Typing one, or several commands that are separated by comas or semicolons, then pressing ENTER, results in correct execution of commands, instructions or displaying a failure notice. More effective way of using MATLAB is by creating the ".m" file. This sort of file can be created in the menu File-New, or by the use of a corresponding icon on the icon toolbar (creating ".m" file is also allowed in other text editors). The content of the ".m" file is a set of commands and expressions that are separated by comas or semicolons. The created ".m" file is initiated by the command RUN, or by typing its name in the command window (without the amplification ".m") and pressing ENTER. All the instructions included in the file will be executed in the order they were saved. This mode of operation can be enhanced with the graphic mode where we can find dialog windows presented as sliders, edit boxes etc., the tools that make the use of MATLAB computing abilities more flexible. Another step is the use of SIMULINK. This mode allows running functional blocks with dynamic models, on their input we can place certain values of the variables, then we may observe the output signals and values.

Regardless of the manner of use of the MATLAB application, it is essential to know the basic syntactic elements of the commands that occur in the language used by the program, that is the set and the meaning of available key phrases, as well as the ranges of constants and variables used in those expressions. The expressions that are scalars use the following arithmetic operators:

Arithmetic Operators for Scalar	**Explanation**
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
^	Power

The list above is expanded when it comes to arrays and matrices:

Arithmetic Operators for Array and Matrices	**Explanation**
.*	Array multiplication. `A.*B` is the element-by-element product of the arrays `A` and `B`.
/	Slash or matrix right division. `B/A` is roughly the same as `B*inv(A)`.
./	Array right division. `A./B` is the matrix with elements `A(i,j)/B(i,j)`.
.\	Array left division. `A.\B` is the matrix with elements `B(i,j)/A(i,j)`
^	Matrix Power.
.^	Array power. `A.^B` is the matrix with elements `A(i,j)` to the `B(i,j)` power.
‘	Matrix transpose. `A'` is the linear algebraic transpose of `A`.
.’	Array transpose. `A.'` is the array transpose of `A`.

Another important group are relational operations presented in the list below:

Relational Operations	Explanation
<	Less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
= =	equality
~ =	inequality

Among numerous known mathematical functions implemented in MATLAB, it is possible to differentiate the following functions:

Mathematical Functions	Explanation
sin(x), cos(x), tan(x), cot(x)	Basic Trigonometric Functions
asin(x), acos(x), atan(x), acot(x)	Inverse Trigonometric Functions
sinh(x), cosh(x), tanh(x), coth(x)	Hyperbolic Functions
sqrt(x)	Square root
abs(x)	Absolute value of real or complex value
exp(x)	e to x
log(x)	Natural logarithm of entries of symbolic matrix
log2(x)	Logarithm base 2 of entries of symbolic matrix
log10(x)	Logarithm base 10 of entries of symbolic matrix
mod(x)	Modulo operator
div(x)	Integer part of a quotient
real(x)	Real part of complex number
imag(x)	Imaginary part of complex number
round(x)	Symbolic matrix element-wise round
ceil(x)	Round symbolic matrix toward positive infinity
fix(x)	Round toward zero

In order to work on the variables there are certain commands available. Several of them are presented in the table below:

Command	Description
who	List variables in workspace
whos	List variables in workspace, with sizes and types
clear	Remove device objects from MATLAB workspace
clear all	Remove items from MATLAB workspace
close	Close database connection
load	Load variables from file into workspace
save	Save workspace variables to file

Apart from these, MATLAB offers key phrases that are typical for high level programming languages. Selected key phrases and their meaning are shown below (next page):

Control word	Description	Control word	Description
break	Terminates execution of a loop	case	Provides alternate execution paths within switch structure
else	Delineates alternate block of statements	elseif	Conditionally executes statements
end	Terminates for, while, and if statements	for	Repeats statements a specific number of times
if	Executes statements conditionally	return	Return to the invoking function.
switch	Directs program execution by comparing point with case expressions	while	Repeats statements an indefinite number of times

1.2. NUMERAL COMPUTING IN MATLAB

MATLAB application offers a wide range of build-in functions that are available in the install version of the software, as well as the possibility of expanding the computing abilities by the use of extensions called toolboxes. Apart from this, every user is able to create his own functions on the basis of an instruction. In order to do so, having run the MATLAB application, the user needs to open in the menu File→New→M-file (the icon of white sheet of paper on the icon toolbar) the window of text editor, where the source code of the calculations needs to be placed. Having written or changed the source code, the file needs to be saved by its unique name (the first character in the name must be a letter – the name of the file cannot begin with a digit !). At this point, in order to run the saved program, the user needs to either type its name in the command window (only the name, without the amplification ".m") or click the "RUN" button in the editing window. With the following runs of the same file, the user can simplify the opening procedure by pressing the arrow-up button, the file will be opened from the "history".
Two types of the ".m" file can be differentiated:
- scripts that do not require parameters, they are run after typing their name in the command window,
- functions that include values, they usually require typing in the function arguments, that are fragments of instructions for other ".m" files.
For the purpose of clarity of the ".m" file code, it is possible to comment and explain the commands by preceding the commentary with the percent sign [%].

1.3. PRESENTATION OF THE RESULTS

The most significant issue in calculating mathematical problems by the means of MATLAB, is the way of presenting and recording the results. The easiest method of presenting data is to omit the semicolon ";" after commands whose results are to be displayed. The second method uses the function ***disp(x)***, where ***x*** is provided with certain value, either numeral or text. There are two more functions that offer even more possibilities ***fprintf(text)*** and ***fprintf('format', var)***, here *'format'* is a string of characters that may include

- Control signs – letters preceded by the sign "\":
  \a alarm,
  \b backspace,
  \f form feed,
  \n new line,
  \r carriage return,
  \t horizontal tab,
  \v vertical tab,
  \xN hexadecimal number N,
  \N octal number N.

- descriptions of formats – preceded by the sign %
  - following the sign, there is a number which specifies the length of the data field, for real numbers there is an additional full stop, after which another number determines the precision of displayed value.
  - the format is closed by a letter that specifies the type of displayed value,
  d – integers – e.g. %8d,
  e – rational numbers in exponential form,
  f – rational numbers in fixed-point form,
  g – rational numbers in **e** or **f** format,
  s – for text values (string).

- the remaining signs with spaces is a text that needs to be displayed.

Another, impressive way of presenting the results is by the means of graphs. The graphs are displayed in the graph windows (*Figure*). The user can display numerous windows, which means numerous graphs. The window is created automatically while performing graphic commands, or it may also be created by the means of the command *figure*. Selected functions operating with graphs are presented in the table below:

Command/Function	**Description**	**Command/Function**	**Description**
plot(x)	create a 2-D line plot of the data in x versus the index of each value.	loglog(x, y)	plots the graph using a log-log scale.
plot(x,y)	create a 2-D line plot of the data in y versus the corresponding values in x	subplot(a, b, n)	divides the current figure into an a-by-b grid and creates an axes in the grid position specified by n
plot(x1,y1, x2,y2, ...)	create a several graphs on the same figure	grid on	adds major grid lines to the current axes
bar(y)	Drawsdone bar for each element in y	grid off	removes all grid lines from the current axes
bar(x, y)	draws bars for each column in y at locations specified in x	title('*graph title*')	add title to current axes
plot3(x, y, z)	displays a three-dimensional plot of a set of data points	xlabel('string_x'); ylabel('string_y')	labels the *x*-axis and the y-axis of the current axes with the string
mesh(x, y, z)	draws a wireframe mesh with the colour determined by z, so the colour is proportional to surface height	legend(s1, s2, s3, ...)	makes a legend entry and activates the legend
surf(x, y, z)	creates a three-dimensional shaded surface	text(x, y, 'text')	represents the text of a text object

Additionally, each graph may be introduced with certain modifications by the tag type and line colour from the data, as well as by the special labelling of the characteristic points in the data. The list of signs defining the parameters of linear chart (plot function) is presented below (next page):

Char	Type of line	Char	Graph tag
–	continuous line	+	+
---	dashed line	*	*
….	dotted line	.	dot
-.-.	dash-dotted line	o	o
Char	**Colour of line**	x	x
y –	yellow	s	square
m –	magenta	d	diamond
c –	cyan	p	star with five arms
r –	red	h	star with six arms
g –	green	v	triangle with node up
b –	blue	^	triangle with node down
w –	white	<	triangle with node on the left
b –	black	>	triangle with node on the right

The graph window FIGURE has numerous parameters. The most often used ones are:
Name – the name is in the title bar of the window,
Position – **[*Xp, Yp, Dx, Dy*] –** position of the left bottom corner of the window *Xp, Yp,* and the dimensions of the window *Dx, Dy*, default in pixels,
Colour – the color of the background with the components [*red, green, blue*] that is specified with three digits in the range from 0 to 1.

Example:
set(F1,'Position',[2 35 630 290], 'Name', 'TEST'... , 'Colour', [0.9, 0.9, 0.9], 'MenuBar', 'none');

1.4. FILE OPERATIONS

MATLAB can use numerous different data types embedded in files: text, binary, graphic, audio and video. In order to perform simple calculations, the user needs to know only a few functions that facilitate saving and loading variables from the workspace. The variables are stored in a so called *Matlab workspace*, and may be saved by the command *save(filename)*, which will save all the variables included in the workspace in a file with the amplification **.mat**. If the user would like to save only certain variables, the command *save(filename,variables)* needs to be applied. It is possible to load the variables saved in the file by the use of a command *load(filename)* or, if the user requires only certain variables, by the command *load(filename,variables)*. If the variables are to be saved in a text file, the user needs to type the file name with the amplification **.txt**. A similar case appears with loading the data from a text file.

2. CREATING THE OUTPUT SIGNAL OF CHOSEN FUNCTIONS IN MATLAB

MATLAB application allows to create output signal in a very simple way. The program creates a table consisting of subsequent signal samples for 1-D signals. Let us for example, create a linear signal *x(t)* consisting of 1001 signal samples that have the following parameters

$$x(t) = t, \quad t \in [0,1] \tag{1}$$

the source code of the program that is used to estimate signal relations (1) may by as the following:

```
t=0:0.001:1;
x=t;
plot(t,x);
```

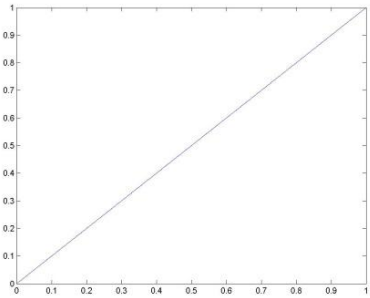


Fig. 1. The source code for a linear function and the corresponding graph

The sine wave can be created by the following code

```
clear all
t=-1:0.001:1;
x=sin(pi*t);
plot(t,x)
title('x=sin(pi*t)')
grid('on')
```

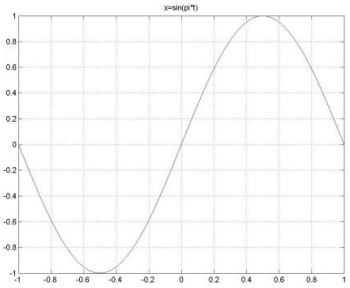


Fig. 2. The source code for sinus function and the corresponding graph

The range of the sine wave can also be defined by other means, for example, by the function 'LINSPACE'.

```
clc
clear all
t=linspace(-pi,pi,1000);
x=sin(t);
plot(t,x); axis([-pi,pi,-1.1,1.1]);
title('x=sin(pi*t)')
grid('on')
axis([-pi,pi,-1.1,1.1]);
```

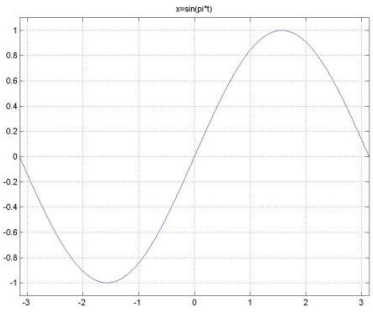


Fig. 3. The source code with 'linspace' for the sine wave and the corresponding graph

The similar case is with the dual function that is the cosine. The source code and the corresponding graph is presented below.

```
clc
clear all
t=linspace(-pi,pi,1000);
x=cos(t);
plot(t,x) axis([-pi,pi,-1.1,1.1]);
title('x=sin(pi*t)')
grid('on')
axis([-pi,pi,-1.1,1.1]);
```

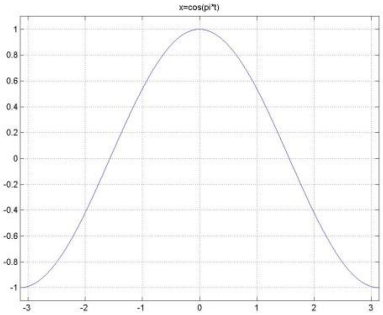


Fig. 4. The source code with ‘linspace’ for the cosine and the corresponding graph

Another example concerns multiplication of two sine waves

```
clear all
t=-1:0.001:1;
x=sin(2*pi*t).*sin(5*pi*t);
plot(t,x)
title(' sin(2*pi*t).*sin(5*pi*t)')
grid('on')
```

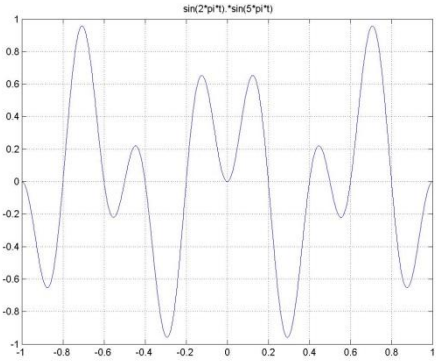


Fig. 5. The source code and the graph representing the multiplication of the sine wave of 1 Hz frequency with another one of 2.5 Hz frequency. The amplitude equals 1

The following example is an assemblage of two sine waves of the same amplitude

```
clear all
t=-1:0.001:1;
x=sin(2*pi*t)+sin(5*pi*t);
plot(t,x)
title(' sin(2*pi*t)+sin(5*pi*t)')
grid('on')
```

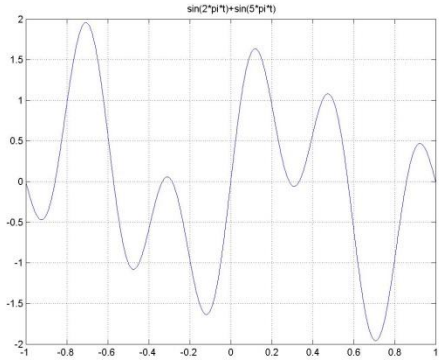


Fig. 6. The source code and the result of addition a sine wave of 1 Hz frequency with another one of 2.5 Hz frequency. The amplitude equals 1

This example presents a source code for three sine waves of different frequencies and amplitudes. Change the code in such a way that it would result in more accurate output signal on the graph (it is sinusoidal). What are the component frequencies of the assembled signal?

```
clear all
t=-1:0.05:1;
x=sin(2*pi*50*t)+2*sin(2*pi*120*t)+sin(5*pi*80*t);
plot(t,x)
title('
sin(2*pi*50*t)+2sin(2*pi*120*t)+sin(5*pi*80*t)')
grid('on')
```

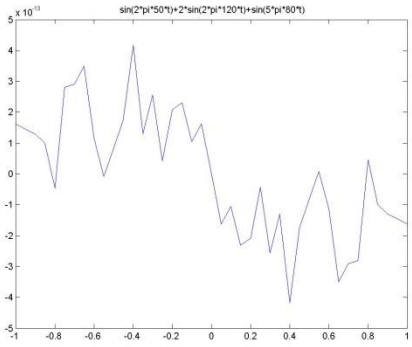


Fig. 7. The source code and the result of addition of three sine waves of different frequencies and amplitudes with mismatched processing step

Another example presents sinusoidal damped vibrations:

```
clear all
t=0:0.001:1;
x=sin(20*pi*t).*exp(-5*t);
plot(t,x);
title('sin(20*pi*t).*exp(-5*t)')
grid('on')
```

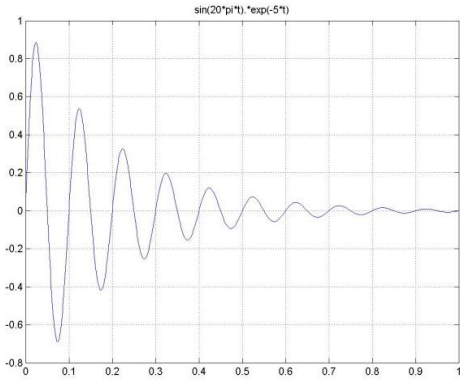


Fig. 8. The source code and the graph presenting sinusoidal damped vibrations

The next example is a rectangular wave.

```
clear all
N=1000;
t=(0:N-1)/N;
i=ones(1,N/4);
k=zeros(1,N/4);
x=[i k i k];
plot(t,x);
axis([0,1,-.1,1.1]);
title(‘rectangular signal’)
grid('on');
```

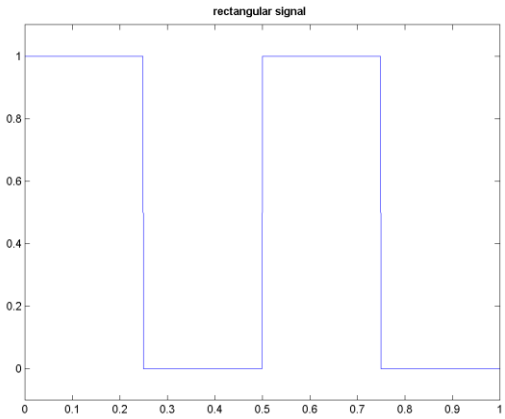


Fig. 9. The source code and the graph presenting the rectangular wave

Exercise: Try to modify the above MATLAB code in order to create (a) asymmetric wave (b) a wave of smaller fulfilment – e.g. 10% (c) a wave of greater fulfilment – e.g. 80%.

An example of a triangle wave.

```
clear all
t=-1:0.001:1;
x=2*t-floor(2*t);
plot(t,x);
title('saw tooth signal’);
grid('on');
```

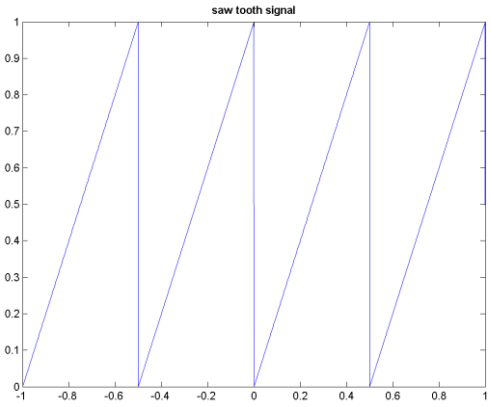


Fig. 10. The source code and the graph presenting a triangle wave.

Examine what is going to change in the wave if the data in the third line of the code will be altered. How to increase the number of "bumps" in the same interval?
Now, we have a signal consisting of two sine waves with random distribution of noise.

```
clc
clear all
t=0:0.001:1;
y=sin(2*pi*50*t)+2*sin(2*pi*12*t);
randn('state',0);
yn=y+0.5*rand(size(t));
figure
subplot(3,1,1);
plot(y(1:300));
grid('on');
title('original signal with random
signal );
subplot(3,1,2);
plot(rand(1,300));
grid('on');
subplot(3,1,3);
plot(yn(1:300));
grid('on');
```

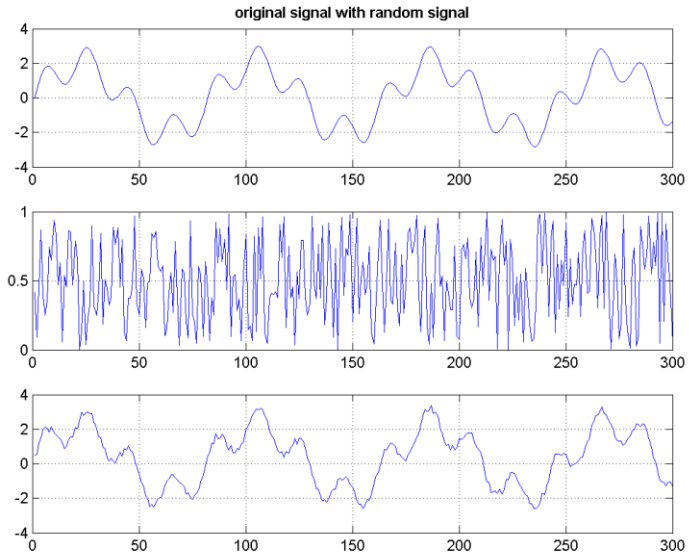


Fig. 11. The source code and the graphs for a signal consisting of the sum of two sine waves and a noise of random distribution

Try to alter the input signal and increase/decrease the influence of noise that is of a random distribution.
The waves need to be created:
Generate the following signals for n in the range from <0…999> for

a) (2)
b) (3)
c) (4)
d) (5)
e) (6)

3. QUANTIZATION OF SIGNALS

Essential information:

3.1 Shannon-Kotielnikov sampling theorem

The discreet signal of specified shape can arise from different continuous signals, hence the reconstruction of discreet continuous signal is ambiguous. In order to be unequivocal, the following condition needs to be met:
A signal that has a spectrum limited to the band ($-\omega_g$; ω_g), can be reconstructed on the basis of samples that are collected with the frequency of ω_g, that is no less than .

3.2 Aliasing

The phenomenon of aliasing occurs when the speed of the sampling ω_g is too low when compared with the width of frequency band which is used by the sampling continuous signal

($-\omega_g$; ω_g). As a result of sampling, the spectrum of continuous signal is duplicated in the field of frequency with the period ω_g.

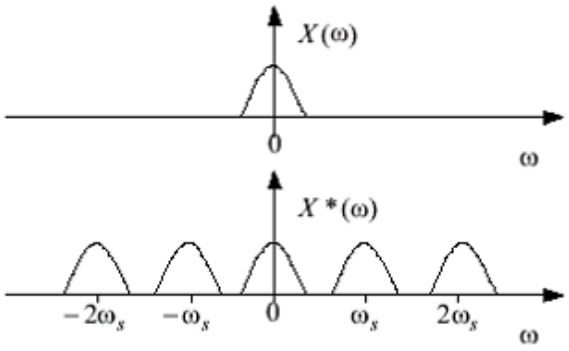


Fig. 12. The original spectrum of the continuous signal and the spectrum after sampling

If the frequency of sampling is too low, then, as a result of adding two overlapping spectrums, a new one occurs with no possibility of reconstructing the original continuous signal.

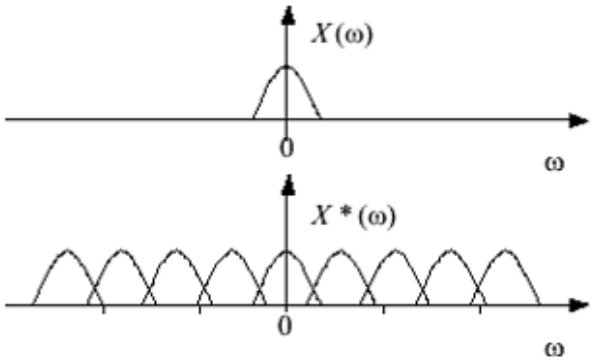


Fig. 13. The graph presenting an aliasing effect

In practice, in order to eliminate the phenomenon of aliasing, it is often necessary to limit the band of the continuous signal before sampling, by the use of low-pass analogue filter

3.3 Quantization of signal, noise of quantization

Modern appliances, most often, change the discreet signal into a digital signal – the value of a sample is presented as a number. The change of a certain value, for example electrical voltage into a number, is performed by the device called analogue-to-digital converter (A/D).
Practically, the number can be processed with certain accuracy – in A/D converter it is usually specified by the amount of binaries that are used to save it. Consequently, a difference between the real value of the signal and the number value occurs for each sample. It can be treated as additional disruptive signal – the noise of quantization. The more precise the save of a number value is the lower the quantization noise.

Exercise 1. Write an .m script that can generate a sine wave and performs sampling of the signal. Observe what continuous signal can be reconstructed on the basis of the samples with different frequency of sampling of the original signal.

Help: exemplary MATLAB code (next page)

```
clear all
close all

T=1;
f=1/T;
fs=100;
t=0:(1/(1000*f)):(20*T);
u=sin(2*pi*f*t);
tp=0:(1/fs*f):(20*T);
up=sin(2*pi*f*tp);
plot(t,u,tp,up,'x',tp,up,'r');
ax=axis
```

Exercise 2. Write a script that will perform a spectral analysis of different signals. Observe in what way the spectrum of different signal changes for different sampling frequencies. Compare the original signal with the one reconstructed from the scope of a signal for different sampling frequencies.

```
clear all
close all

N=2^10;
n=1:N;
k=0;
u=0.2+sin(2*pi*n/N+1);
for i=2:150
   u=u+sin(i*2*pi*n/N+1)/i;
end

fu=fft(u);
fup=fftshift(fu);
np=n-N/2;
bar(np,abs(fup));
mask=[ones(1,151),zeros(1,N-301), ones(1,150)];
fureg=fu.*mask;
ureg=real(ifft(fureg));
plot(n,u,n,ureg);
```

Exercise 3. Estimate the ratio S/N for discreet signal of 4 binary A/D converter.

a) Generate the discreet sine wave
b) Designate the quantization noise by graph
c) Perform an analysis of amplitude spectrum of noise
d) Calculate S/N

Repeat the calculations for two 8, 12 and 16 binary A/D converter. Compare S/N for each case

```
clear all
close all
clc
N=2^10;
n=0:N-1;
x=sin(2*pi*n/N);
M=4;
xc=(1/(2^(M-1)))*round((2^(M-1)-1)*x);
subplot(3,1,1);
plot(n,x,n,xc)
title('original signal after quantization');

xs=x-xc;
```

```
subplot(3,1,2);
plot(xs)
title('quantization noise');

fxs=fft(xs);
subplot(3,1,3);
plot(abs(fxs));

es=sum(x.^2);
en=sum(xs.^2);
SN=10*log10(es/en);
opis=[' amplitude spectrum of noise of A/D converter resolution'  ,num2str(M),'binaries;
S/N= ',... num2str(SN),'dB'];
title(description);
```

4. SIGNAL MODULATION BY THE USE OF CARRIER WAVE

4.1 Signal modulation by the use of carrier wave

The amplitude modulation of the sine wave is often used in practice. This action facilitates the creation of signal which has a spectrum that can be placed in a specified band, required, for example, by a specific way of transmission, simultaneous transmission in one transmission channel for several signals, with a so called division in the field of frequency etc. An output signal y(t) is equal to:

(7)

where $u(t)$ is a modulating signal, – pulsation of the carrier wave, $0<m<1$ coefficient called the depth of modulation.

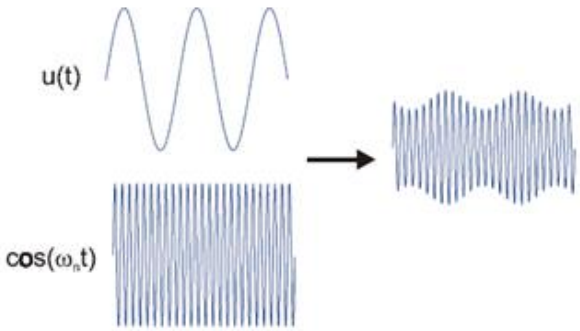


Fig. 14. The graph presenting an amplitude modulation

The mathematical analysis of the modulated signal shows that the spectrum of the modulated signal shifts according to the value.

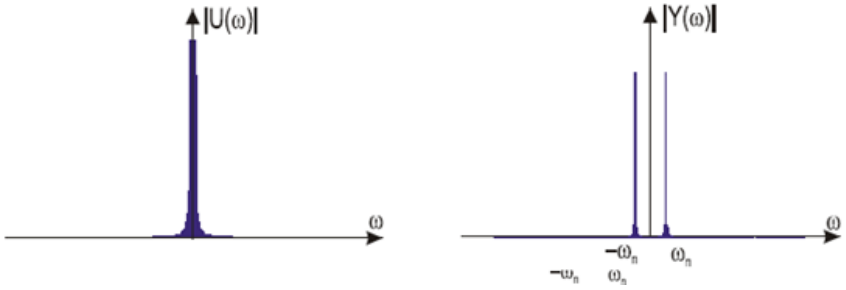


Fig. 15. The influence of the amplitude modulation on the signal spectrum

An additional component appears in the modulated signal and it is called the carrier wave. The reconstruction of $u(t)$ signal is performed in the device called amplitude demodulator. The simplest example of demodulator is the peak detector.

If the output signal of the modulator is specified by the following presumption:

(8)

it turns out that its spectrum does not include the carrier wave.

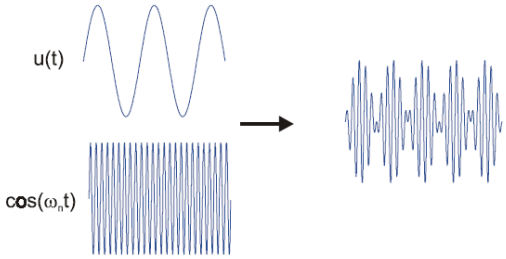


Fig. 16. The graph presenting an amplitude modulation without carrier

The simplest way of reconstructing $u(t)$ signal from $y(t)$ is another signal modulation, this time of $y_{rec}(t)$:

(9)

The spectrums of the original signal U(ω) with modulated Y(ω) and demodulated $Y_r(\omega)$ are presented in the picture below.

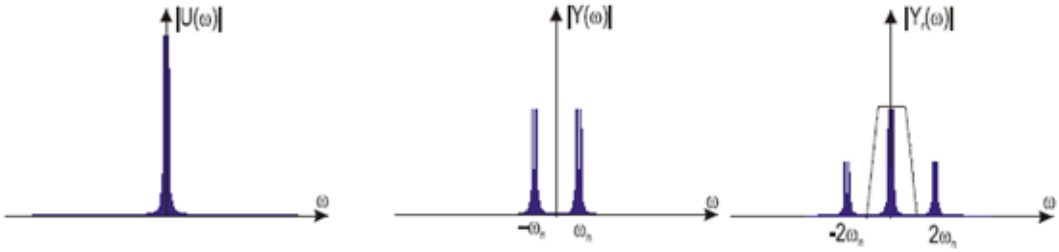


Fig. 16. The spectrums of the original signal U(ω) with modulated Y(ω) and demodulated $Y_r(\omega)$

Example 1. Generate a signal that is described by the equation $x(n) = \sin(\frac{2\pi}{N}n) + 3\sin(\frac{12\pi}{N}n)$ for 1023 samples of the signal, calculate DFT, designate the amplitude spectrum and comment on the result.

Help: exemplary MATLAB code below.

```
clear all
close all
N=2^10;
n=0:N-1;
x=sin(2*pi*n/N)+3*sin(2*pi*12*n/N);
fx=fft(x);
ffx=fftshift(fx);
np=n-N/2;
subplot(2,1,1);
plot(x);
title('sine wave);
subplot(2,1,2)
bar(abs(ffx));
title('and its amplitude spectrum');
```

Example 2.

Generate signals that are described by the equation $x1(n) = \sin(\frac{2\pi}{N}n)$, $x2(n) = \sin(\frac{150\pi}{N}n)$ for 4095 samples of the signal, signal *xl* is a modulated signal of *x2*, calculate DTF, designate the amplitude spectrum and comment on the result.

Help: exemplary MATLAB code below.

```
clear all
close all
N=2^12;
n=0:N-1;
x1=sin(2*pi*n/N);
x2=sin(150*pi*n/N);
fx1=fft(x1);
fx2=fft(x2);
x3=sin(2*pi*n/N).*sin(150*pi*n/N);
ffx1=fftshift(fx1);
ffx2=fftshift(fx2);
fx3=fft(x3);
ffx3=fftshift(fx3);
np=n-N/2;
subplot(2,2,1);
plot(x1);
title('Sine wave);
subplot(2,2,3)
plot(np,abs(fx1));
title('and its amplitude spectrum');
subplot(2,2,2);
plot(x2);
title('Modulating signal');
subplot(2,2,4)
plot(np,abs(fx2));
title('and its amplitude spectrum');
figure
subplot(2,1,1);
plot(x3);
title('Modulated signal');
```

```
subplot(2,1,2);
plot(np,abs(fx3));
title('and its amplitude spectrum');
```

5. FM MODULATION

FM modulation is a process where the carrier wave is coded by the changes in its instantaneous frequency, depending on the input signal. The frequency f_n of the carrier wave changes in the range from $f_n - \Delta_f$ to $f_n + \Delta_f$. Δ_f is called **frequency deviation**, while the ratio $m_f = \Delta_f/f_n$ **frequency deviation index**. FM modulation facilitates filtering much more noise on the collector's side than in the AM system. An example of the use of the FM is a radio on UKF band.

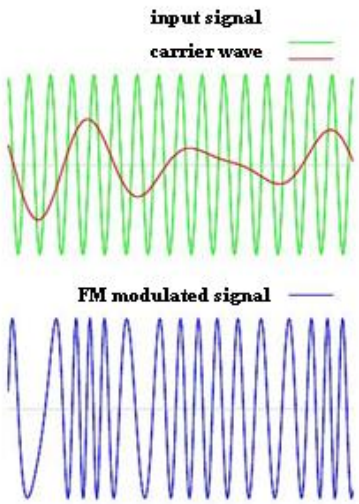


Fig. 17. The graph presenting FM modulation

Basic equations describing this modulation are presented below:

$$y(t) = Y \sin(2\pi \cdot F(t) \cdot t) \qquad F(t) = f_0 + A_M \sin(2\pi \cdot f_M \cdot t) \tag{10}$$

where:
$y(t)$ – output signal (Y - amplitude)
f_0 – carrier frequency
A_M – the amplitude of input signal (responsible for frequency modulation)
f_M – the frequency of input signal

Frequency deviation is proportional to the amplitude of modulating signal, and is not conditioned in any way by its frequency. If we adopt large – when compared with modulating frequency – value of the parameter (subsequently, the modulation index is large as well), it will allow to restrict the frequency noise that appears during the transmission through the radio channel, because of the fact that both, atmospheric noise and the radio noise influence mainly the amplitude of the signal that has a high frequency (not the frequency). However, this is the reason for the growth of occupancy of the band for FM signal. On the basis of the value of the modulation index, it is possible to differentiate two types of FM modulation:
· narrowband – when the ***modulation index*** is lower or equal to 1 [rad],
· broadband – when the ***modulation index*** is > 1 [rad].
When the modulation index has a great value, the width of transmission band approaches the full range of frequency alterations $f_{M,}$ however, it remains only slightly bigger. On the other hand, when the modulation index is of a small value, the FM signal spectrum is effectively restricted to the carrier frequency f_0 and one couple of side frequency $fo \pm fM$, so that the width of the band is close to $2\ f_{M,.}$

The modulation index implies that the signal of modulated frequency is a non-linear function of the modulating signal, regardless of the fact that the instantaneous frequency changes in a linear way along with the modulating signal. It means that frequency modulation is a non-linear wave. This is why the spectrum of FM signal is not connected with the spectrum of the modulating signal in a simple manner. The wave that is modulated in respect of frequency, consists of the carrier frequency and the sideband, that in turn consists of symmetrical bands adjacent to the carrier frequency and recurring in intervals equal to the modulating frequency. The amplitude and the amount of occurring couples of bands, and what is more the width of the occupied band, depends on the modulation index thus it changes according to the value of deviation or modulating frequency.

In comparison to the amplitude modulation, frequency modulation has two main advantages that are: constant power of the modulated wave that does not depend on the modulation index, and lower susceptibility for atmospheric noise. Whereas the drawback of FM modulation when compared to AM, is that it has lower modulation range, what is more, it occupies broader band and requires the application of more complicated electrical systems.

The power of FM signal is not altered in the process of modulation, and is equal to non-modulated carrier signal, hence all the power is used to transmit the information. The FM transmitters have very high energy efficiency. In order to improve the overall signal-to-noise ratio while transmitting, the collector needs to apply the process of increasing the magnitude of high frequencies (pre-emphasis), as well as the appropriate symmetrical suppression of the frequencies (de-emphasis). The outcome of these processes is that the resultant characteristic of the signal frequency does not change, however, it causes the noise to decrease.

Two known methods of frequency modulation:

- direct
- indirect

In the direct method, most commonly used in CB radiophones, a change in the frequency of the carrier wave is delivered by connecting a capacitive diode to the generator circuit. The capacity of the diode changes according to the time of shifts of the modulating voltage.

The indirect method of frequency modulation obtains required deviation in an indirect manner, through the modulation of the carrier wave's phase. This results in a change of resistance or reactance of the circuit on the next level in the generator, according to the time of shifts of the modulating voltage. In order to restrict the maximal deviation of a set value (2,5 kHz) modulation restrictors are applied – this action prevents the occurrence of transmission noise and other noise for neighbouring channels.

Exemplary FM modulation MATLAB code:

```
close all;
clear all;
clc;

t1 = -0.02:1.e-4:0;
t2 = 0:1.e-4:0.02;
Ta = 0.01;
m1 = 1 - abs((t1+Ta)/Ta);
m1 = [zeros([1 200]),m1,zeros([1 400])];
m2 = 1 - abs((t2-Ta)/Ta);
m2 = [zeros([1 400]),m2,zeros([1 200])];
m = m1 - m2;
m = 5*m;

t = -0.04:1.e-4:0.04;                    % period
```

```
fc = 400;                                   % carrier wave frequency

kf = 160*pi;                                % FM modulation block
m_int = kf*1.e-4*cumsum(m);
fm = cos(2*fc*pi*t + m_int);


dem = diff(fm);                             % FM demodulation
dem = [0,dem];
rect_dem = abs(dem);

a = fir1(80,100*1.e-4);                     % Signal filtering
b = 1;
rec = filter(a,b,rect_dem);


fl = length(t);
fl = 2^ceil(log2(fl));
f = (-fl/2:fl/2-1)/(fl*1.e-4);
mF = fftshift(fft(m,fl));                   % input signal spectrum
fmF =  fftshift(fft(fm,fl));                % FM modulated signal spectrum
rect_demF = fftshift(fft(rect_dem,fl));     % FM demodulated signal
recF = fftshift(fft(rec,fl));               % output signal spectrum

                                            %Graphs in the time domain
figure(1);
subplot(2,1,1);
plot(t,m);
title(;input signal');
xlabel('{\ t} (s)');
ylabel('m(t)');
grid;
subplot(2,1,2);
plot(t,fm);
title('FM');
xlabel('{\t} (s)');
ylabel('FM(t)');
grid;

figure(2);
subplot(2,1,1);
plot(t,rect_dem);
title('FM reconstruction');
xlabel('{\ t} (s)');
ylabel('FM modulated signal (t)')
grid;
subplot(2,1,2);
plot(t,rec);
title('reconstructed signal');
xlabel('{\t} (s)');
ylabel('m(t)');
grid;

figure(3);                  %Graphs in the frequency domain
subplot(2,2,1);
plot(f,abs(mF));
title('spectrum input signal');
xlabel('f(Hz)');
ylabel('M(f)');
grid;
```

```
axis([-600 600 0 200]);
subplot(2,2,2);
plot(f,abs(fmF));
title('FM spectrum');
grid;
xlabel('f(Hz)');
ylabel('C(f)');
axis([-600 600 0 200]);
subplot(2,2,3);
plot(f,abs(rect_demF));
title('spectrum FM demodulated signal');
xlabel('f(Hz)');
ylabel(' FM demodulation ');
grid;
axis([-600 600 0 150]);
subplot(2,2,4);
plot(f,abs(recF));
title('output signal spectrum');
xlabel('f(Hz)');
ylabel('M(f)');
grid;
axis([-600 600 0 150]);
```

6. ASK_FSK_PSK MODULATION

Analog modulation. It occurs when a certain parameter (amplitude, frequency or phase) of the carrier wave changes according to the shifts in the values of the amplitude of the input signal.
Digital modulation is a method allowing to change the rectangular digital signal wave into a sine carrier wave transmitted in a certain band.
As a result of changing the value of one of the three parameters describing the sine carrier wave (A – amplitude, F – frequency, P – the phase of the carrier signal), we obtain appropriately three types of modulation:

- **Amplitude modification AM** – applied for analogue signals, the value of the carrier wave amplitude changes according to the shifts in the value of the input signal amplitude. During modulation of the **digital signals**, the switch arises between two levels of the amplitude, moreover, this way of modulation is called amplitude shift keying **ASK**.
  Digitally modulated acoustic signal sounds like one tone of fast shifts of volume, however, these shifts appear too quickly to be differentiated by a human ear. In this way, transmitted modulated signal is prone to suppressions, thus the received signal can be different from the emitted one, and this is the reason why the AKS modulation is not suitable for the transmission of data.
- **Frequency modification FM** – the value of the frequency changes according to the shifts of the input signal amplitude.
  Frequency modulation that is used with digital transmissions is called **Frequency Shift Keying FSK**. In the simplest case, only two frequencies are used: fL (*low*) – usually to represent the binary "1", and fH (*high*) – to represent the binary "0" of the output modulated signal.
- **Phase modification PM** – it is a change of the phase of the modulated carrier signal, according to the shifts of the output signal amplitude – for example digital signal.
  Phase Shift Keying PFK is applied to the modulation of digital waves. For example, if a wave, in a particular moment, goes down and the digital signal shifts, the direction of the wave is altered in such a way that it goes up.

  There are numerous varieties of the basic PSK modulation form. Each type is characterized by advantages as well as disadvantages, which are used in order to determine which kind is the most suitable in a given application. The basic form of the phase modulation is expressed by bivalent BPSK modulation (Binary PSK). Along with the shift of the logic state of the digital modulating signal, the phase of BPSK signal is switched between two values, usually distanced by 180^{o}. This

situation indicates that the phase modulation, in its simplest form, causes a shift of the carrier phase by 0^o or 180^o, according to the state of the output signal.

In order to avoid problems with estimating the collectors phase, a method where a shift of the phase by 180^o corresponds to the binary "1" and the lack of shift means the binary "0" is used very often. It is called Differential PSK.
Upgraded, differential method of this modulation is also used very often, and is called Differential Phase-Shift Keying. This modulation allows the binary value to be estimated by the level of the phase shift according to the current binary, which means that the characteristic states of the carrier wave are representing the shift of the value of the output signal in the relation to the previous one, not the direct binary value of the modulated digital signal as in the ordinary PSK modulation. For example, a shift of a phase by 90^o can represent the binary "0", and a shift by 270^o can represent the binary "1".
Apart from binary modulation, polyvalent varieties of the PKS modulation are widespread. A good example would be Quadrature PSK modulation. In this case, the signal phase can take 4 values that are shifted by 90^o. In 8-PSK modulation, 8 phase values of modulated signal occur.

Exemplary ASK, FSK and PSK modulation MATLAB code:

```
clear all;
close all;
clc;

n=1000;
b=randint(1,n);
f1=1;
f2=2;
t=0:1/30:1-1/30;
%ASK
sa1=sin(2*pi*f1*t);
E1=sum(sa1.^2);
sa1=sa1/sqrt(E1);
sa0=0*sin(2*pi*f1*t);
%FSK
sf0=sin(2*pi*f1*t);
E=sum(sf0.^2);
sf0=sf0/sqrt(E);
sf1=sin(2*pi*f2*t);
E=sum(sf1.^2);
sf1=sf1/sqrt(E);
%PSK
sp0=-sin(2*pi*f1*t)/sqrt(E1);
sp1=sin(2*pi*f1*t)/sqrt(E1);

ask=[];psk=[];fsk=[];      %Modulation block
for i=1:n
   if b(i)==1
      ask=[ask sa1];
      psk=[psk sp1];
      fsk=[fsk sf1];
   else
      ask=[ask sa0];
      psk=[psk sp0];
      fsk=[fsk sf0];
   end
end
figure(1)
subplot(411)
```

```
stairs(0:10,[b(1:10) b(10)])
axis([0 10 -0.5 1.5])
title(‘bit data’);grid on
subplot(412)
tb=0:1/30:10-1/30;
plot(tb, ask(1:10*30),'b','linewidth',1.5)
title('ASK Modulation');grid on
subplot(413)
plot(tb, fsk(1:10*30),'r','linewidth',1.5)
title('FSK Modulation');grid on
subplot(414)
plot(tb, psk(1:10*30),'k','linewidth',1.5)
title('PSK Modulaion');grid on
xlabel('Time');ylabel('Amplitude')

for snr=0:20                    %AWGN
   askn=awgn(ask,snr);
   pskn=awgn(psk,snr);
   fskn=awgn(fsk,snr);

   A=[];F=[];P=[];              % Noise block
   for i=1:n
                                %ASK Demodulation
      if sum(sa1.*askn(1+30*(i-1):30*i))>0.5
         A=[A 1];
      else
         A=[A 0];
      end
                                %FSK Demodulation
      if sum(sf1.*fskn(1+30*(i-1):30*i))>0.5
         F=[F 1];
      else
         F=[F 0];
      end
                                 %PSK Demodulation
      if sum(sp1.*pskn(1+30*(i-1):30*i))>0
         P=[P 1];
      else
         P=[P 0];
      end
   end
end

figure(2)
subplot(411)
stairs(0:10,[b(1:10) b(10)])
axis([0 10 -0.5 1.5]);grid on
title(‘Reconstructed signal with AWGN ')
subplot(412)
tb=0:1/30:10-1/30;
plot(tb, askn(1:10*30),'b','linewidth',1.5)
title(‘ASK signal’);grid on
subplot(413)
plot(tb, fskn(1:10*30),'r','linewidth',1.5)
title(‘FSk signal’);grid on
subplot(414)
plot(tb, pskn(1:10*30),'k','linewidth',1.5)
title(‘PSK signal’);grid on
```

7. QAM MODULATION

QAM – *Quadrature Amplitude Modulation* is a method that links two, modulated by the amplitude, AM signals in one band, which results in doubling of the effective bandwidth information.
In QAM signal two carriers exist, each of the same frequency however, different in a phase by 90 degrees – which is a quarter of a period, hence the term quadrature. The first signal is called the signal I, and the other one the signal Q. Mathematically speaking, one of the signals can be represented by a sine wave, and the other one by a cosine wave. Those two modulated carriers are directed into a transmission channel. Carriers are separated in the collector, the data are extracted from them and then joined in. This is the way in which they create an original signal. Picture 18 presents the QAM modulator, and picture 19 presents a schema of creating constellation points.

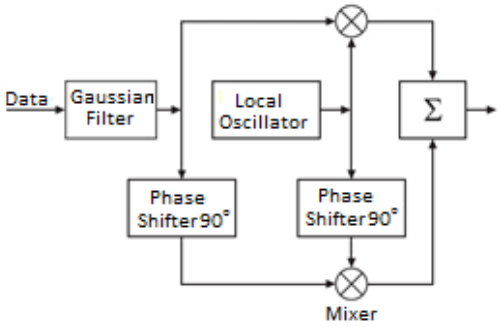


Fig. 17. The QAM modulator

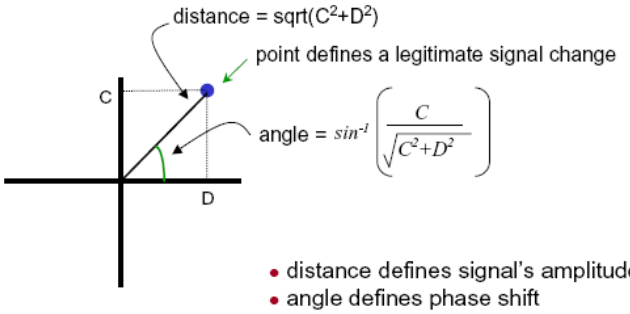


Fig. 18. A schema of creating constellation points

Examples of the constellations that are used most frequently:

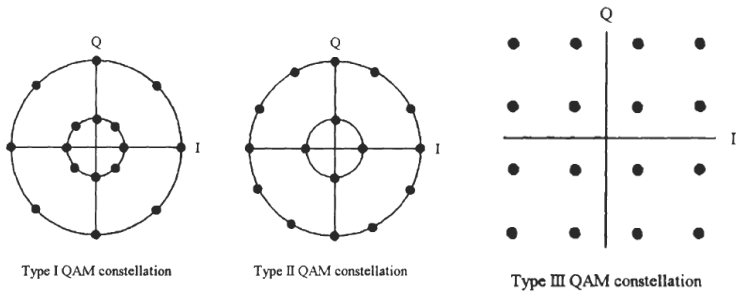
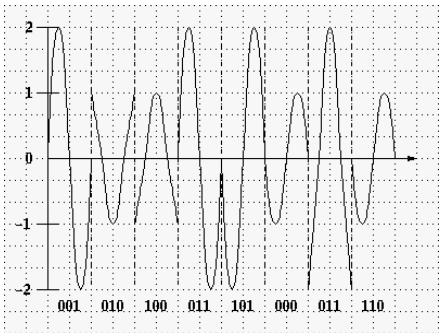


Fig. 19. The constellation in QAM modulation used most frequently

The pictures below present a table with all the possible changes of the amplitude (0 or 1) and the phase (change of the phase by 90^{o}). Next to the table there is a graphic representation of the modulated QAM signal wave.



$Q_2Q_1Q_0$	A	φ
000	1	0°
001	2	0°
010	1	90°
011	2	90°
100	1	180°
101	2	180°
110	1	270°
111	2	270°

Fig. 20. The graph presenting the table with all the possible changes of the amplitude and the phase signal wave

An exemplary source code for QAM modulation in MATLAB is presented below:

```
clear all;
close all
clc
disp(' 4-QAM Modulation');
disp('Exampl of input bit string  [0 0 1 1 0 1 1 0 1 0 1 1],10)');
bin = [0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1];f=5;
L=length(bin);
k=100;
bit1=ones(1,k);
bit0=0*bit1;
symbol=ones(1,2*k);
mbit=[];mx=[];my=[];

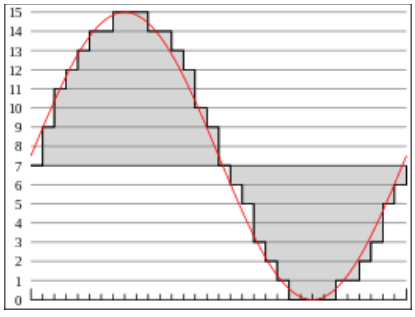
if 2*round(L/2)~=L
   error('Not appropriate bit number ');
end
                                                    % The constellation
for n=1:2:L
   if bin(n)==0 & bin(n+1)==0
      x=symbol;y=0*symbol; bit=[bit0 bit0];
  elseif bin(n)==0 & bin(n+1)==1
      x=-1*symbol;y=0*symbol; bit=[bit0 bit1];
  elseif bin(n)==1 & bin(n+1)==0
     x=2*symbol;y=0*symbol; bit=[bit1 bit0];
  elseif bin(n)==1 & bin(n+1)==1
      x=-2*symbol;y=0*symbol; bit=[bit1 bit1];
   end
   mbit=[mbit bit];mx=[mx x];my=[my y];
end
v=0:2*pi/k:2*pi*L-2*pi/k;
msync = mx+my*j;
                                                  %  QAM signal-  QAM modulation

qam =  real(msync).*cos(f*v)+imag(msync).*sin(f*v);
                                                  % Signal with noise
Vn=awgn(qam,10,'measured');
Vnx=Vn.*cos(f*v);
Vny=Vn.*sin(f*v);
```

```
[b,a]=butter(2,0.04);
                                        % QAM demodulation
Hx=filter(b,a,Vnx);
Hy=filter(b,a,Vny);
M=length(Hx);
mdeb=[];
for m=k:2*k:M
   if 0<Hx(m)& Hx(m)<0.7
      deb=[bit0 bit0];
   elseif 0.7<Hx(m) & Hx(m) <1.5
      deb=[bit1 bit0];
   elseif -0.6<Hx(m) & Hx(m) <0
      deb=[bit0 bit1];
   elseif -1.5<Hx(m) & Hx(m) <-0.7
      deb=[bit1 bit1];
end
mdeb=[mdeb deb];
end
                                        % QAM figure
figure(1);
subplot(4,1,1);plot(mbit,'r','linewidth',2);axis([0  k*L -0.5 1.5]);grid on;legend('Input data');
subplot(4,1,2);plot(qam,'m','linewidth',1.5);axis([0  k*L -2.5 2.5]);grid on;legend('QAM
modulation' );
subplot(4,1,3);plot(Vn,'g','linewidth',1.5);axis([0  k*L -2.5 2.5]);grid on;legend('QAM  signal with
noise');
subplot(4,1,4);plot(mdeb,'k','linewidth',1.5);axis([0  k*L -0.5 1.5]);grid on;legend(' QAM
demodulation signal');
figure(2)
subplot(2,1,1);plot(Hx,'g','linewidth',1.5);axis([0  k*L -2.5 2.5]);grid on;
legend('In-phase (I) component);
subplot(2,1,2);plot(Hy,'m','linewidth',1.5);axis([0  k*L -2.5 2.5]);grid on; legend('Quadrature-
phase (Q) component');
```

8. DPCM MODULATION

PCM modulation is a Pulse Code modulation which assigns string codes (usually of 8 binaries) to every code sample. This solution has two drawbacks that are an increase of the digital signal band when compared to the analogue signal band, and the necessity to extend the bandwidth of information. Picture 21 presents the period of the original signal and the PCM signal, along with the structure of 8 binary PCM coding word.



Z	A1	A2	A3	B1	B2	B3	B4

Fig. 21. The graph presenting the period of the original signal and the PCM signal

Instead of transferring the amplitude, it is possible to send the difference of the particular signal samples. If the signal is undergoing the process of sampling with a high frequency, the neighbouring samples will differ only slightly. Additionally, one may be tempted to predict, on the basis of the last

several samples, what the next sample would be like – this forecast is referred to as the estimation by the signal prediction. The function above is conducted by the DPCM modulation method (Differential Pulse Code Modulation). The evaluation of the samples is subjected to any possible mistakes and a greater modulator/demodulator system complication. On the other hand, it facilitates an increase of the information flow. In most of the real signals, this solution is the most suitable because of the strong correlation between the neighbouring signals. If such samples are coded in PCM standard, the output digital signal will include the information redundancy. This means that the PMC modulation process includes symbols that are of no use for the transferred information. The DPCM modulator/demodulator schema in presented in the picture below:

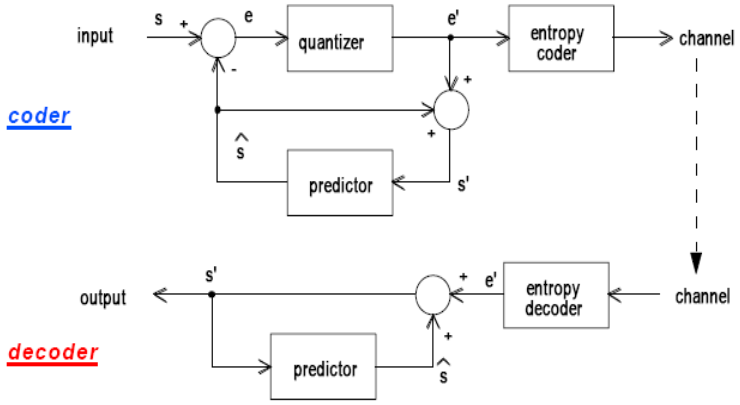


Fig. 21. The block diagram of DPCM modulator/demodulator

On the basis of the figure above, we are able to define the differentiating signal e as a difference between the input signal s and its prediction . It is a prediction error, what is more, it shows the value of the correction of the actual difference of noise and its prognosis. After the process of quantification of the differentiating signal, its e' value goes into the signal prediction adder and then, as s' to the prediction system, which estimates the following input signal sample s, and simultaneously it goes to the output coder of the modulator. Due to the fact that with a great sample correlation of the s signal, the dynamics of the error is of a less value when compared to the dynamics of the input signal, a smaller number of binaries is required in the coding word representing the differential code. In the speech systems the advantage of DPCM in the OCM context is estimated for about 4-11 dB. Because of the fact that 6 dB is a gain of 1 binary (prolonging of the coded word of 1 binary in the relation to the original word), the advantage of DPCM system can be expressed in the category of the ratio of transmission. For the fixed sampling of the 8kHz frequency, the implementation of DPCM modulation allows to increase the ratio of transmission by approximately 8-16 kb/s (1-2 binaries per sample) in the relation to the standard PCM modulation.

An exemplary MATLAB code describing the working of the DPCM coder is presented below:

a)

```
function [Q,B] = pcm_kwant_koder(e,rozm_bit)

if rozm_bit<4
  slen = length(e);
  D = 2^(-rozm_bit);
  switch rozm_bit
  case 3,
     for i=1:slen
       if e(i) < -3*D
                 Q(i) = -(7/2)*D;
              b(i,:) = [ 0 0 0 ];
          elseif e(i) >= -3*D & e(i) < -2*D
```

```
            Q(i) = -(5/2)*D;
            b(i,:) = [ 0 0 1 ];
            elseif e(i) >= -2*D & e(i) < -D
          Q(i) = -(3/2)*D;
                b(i,:) = [ 0 1 0 ];
                                elseif e(i) >= -D & e(i) < 0
             Q(i) = -(1/2)*D;
                b(i,:) = [ 0 1 1 ];
             elseif e(i) >= 0 & e(i) < D
             Q(i) = (1/2)*D;
           b(i,:) = [ 1 0 0 ];
               elseif e(i) >= D & e(i) < 2*D
               Q(i) = (3/2)*D;
             b(i,:) = [ 1 0 1 ];
                                elseif e(i) >= 2*D & e(i) < 3*D
               Q(i) = (5/2)*D;
             b(i,:) = [ 1 1 0 ];
               elseif e(i) >= 3*D
               Q(i) = (7/2)*D;
             b(i,:) = [ 1 1 1 ];
        end
      end

    case 2,
      for i=1:slen
        if e(i) < -D
          Q(i) = -(3/2)*D;
                b(i,:) = [ 0 0 ];
                                elseif e(i) >= -D & e(i) < 0
             Q(i) = -(1/2)*D;
          b(i,:) = [ 0 1 ];
        elseif e(i) >= 0 & e(i) < D
          Q(i) = (1/2)*D;
                b(i,:) = [ 1 0 ];
                                elseif e(i) >= D
             Q(i) = (3/2)*D;
                b(i,:) = [ 1 1 ];
        end
      end

    case 1,
      for i=1:slen
        if e(i) < 0
          Q(i) = -(1/2)*D;
                b(i,:) = [ 0 ];
                                elseif e(i) >= 0
             Q(i) = (1/2)*D;
          b(i,:) = [ 1 ];
        end
      end
    otherwise
      fprintf('Choose the length of code word 1, 2 or 3');
    end
    b=b';
    B=b(:)';
end
```

b) main MATLAB source

```
close all
clear all
```

```
clc
                                        % sampling
fs=1/2000;
tn=0:fs:1/25;
                                        %choose a signal
s=1*sin(2*pi*50*tn);
%[s,fs]=wavread('bye441');          % read .wav file

lpclen=20;
rozm_bit=input('Length the filter =');

                                        %coder
slen = length(s);
e(1) = s(1);
[Q(1),b(1,:)] = pcm_kwant_koder(e(1), rozm_bit);
st(1) = Q(1);

for i=2:slen
  if i<=lpclen
    e(i) = s(i)-st(i-1);
    [Q(i),b(i,:)] = pcm_kwant_koder(e(i), rozm_bit);
    st(i) = st(i-1) + Q(i);
  else
    m=0;
    [a,G]=lpc(s(i-lpclen:i-1),lpclen);
    a = a*G;
    m = 1:lpclen;
    j=2:lpclen + 1;
    sth = sum(a(j).*st(i-m));
    ai(i,:)=a;
    e(i) = s(i)-sth;
    [Q(i),b(i,:)] = pcm_kwant_koder(e(i), rozm_bit);
    st(i) = sth + Q(i);
  end
end
b=b';
b=b(:)';

figure
subplot(3,1,1):plot(s,'r');
ylabel('amplitude');
title('DPCM with predictor: (red-input signal), (green-predictor), (blue- output signal DPCM)');
subplot(3,1,2):stairs(Q,'g');
ylabel('amplitude');
subplot(3,1,3);bar(b,'b');
ylabel('amplitude');
xlabel('index, n');
grid on;
```

9. THE DELTA MODULATION

The basic drawback of the DPCM differential modulation is a general complexity of the coder/decoder, and the necessity to increase the length of the coding word (the number of binaries per sample) with the increase in resolution. The delta modulation is a simplified digital method based on forecasting – it is a special case of the differential modulation. This type of modulation has the frequency of sampling that is significantly higher than the frequency resulting from the law on sampling. This is the reason why the subsequent samples of the signal differ insignificantly and are closely correlated. The increase in the correlation is automatically reflected by the increase of the prediction error. Two-level quantizer is applied when the

samples differ only slightly. The difference between the output signal value and its estimated value is a constant which is equal to one unit – usually referred to as delta.
The approximating signal resembles a "stepped" signal that is alternating in all of the sampling points. For the fact that it requires having a binary signal characteristic, its amplitude cannot be constant – in one moment it increases and in another it decreases. It has been systematically presented in the picture below:

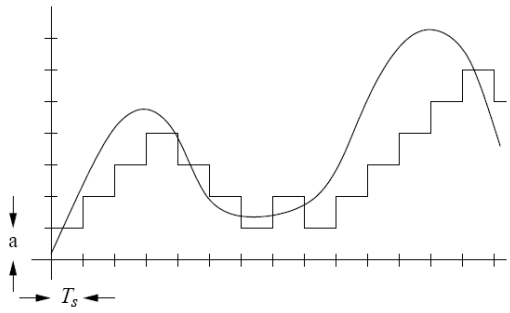


Fig. 22. The original signal and the stepped signal

The output signal of the modulator is solely a difference between the processed signal and the approximating stepped signal. The input signal is processed into a string of zeros and ones, where 1 means an increase of the amplitude, and 0 a decrease of the amplitude. On the modulator's output delta, a string of bipolar impulses is present – positive is represented by 1 (increase of the amplitude in the decoder), negative by 0.
The delta modulator is simple to create – a threshold system detects a token of the error signal. The oscillator of the approximating signal can be represented by an integrator or other, more complex electronic system.
In numerous systems using the delta modulation, a typical distortion of a signal can occur. In the literature, the distortion is referred to as lag error (slope overload). This error results from too quick rising or descending of the sampling signal in the relation to rising/descending approximating signal of the delta coder.
Another source of errors, which occur during recreating of the modulated signal, is the modulation of the input signal of the amplitude which is smaller than a singular positive or negative delta growth – it is then called the granulation noise. The correct selection of a quantum leap and the period of sampling is an essential issue in the delta modulation. Graphically indicated mistakes are presented in the picture below

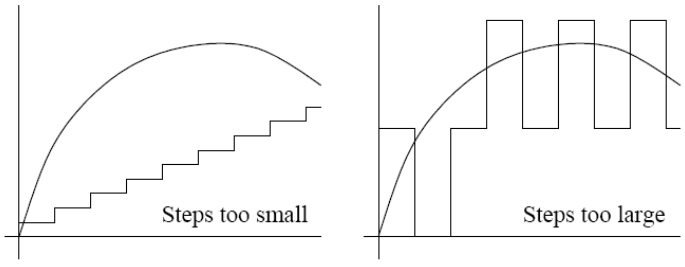


Fig. 23. The source of error in the delta modulation

An exemplary MATLAB code describing the rule of delta modulation is presented below:

```
clear all
close all
clc
```

```
fprintf('Wait…- it is a real signal… ');
[s,fs]=wavread('notify');                    %choose a signal
%[s,fs]=wavread('bye441');

krok=1/25;
slen=length(s);
sapro(1)=0;
for i=1:slen
   if(s(i)>=sapro(i))
      delta(i)=1;
      sapro(i+1) = sapro(i) + delta(i) * krok;
  else
     delta(i)=-1;
     sapro(i+1) = sapro(i) + delta(i) * krok;
     tx(i)=0;
 end;
end;

cn = delta< 0;

slen = length(cn);
%Ts=na/N;
Ts=1/fs;
n=0:Ts:Ts*(slen-1);

slen = length(cn);
sapro(1) = 0;
for i=1:slen
  if(cn(i)==0)
     sapro(i+1) = sapro(i)+krok;
  else
     sapro(i+1) = sapro(i)-krok;
  end
end

 [xx,yy]=stairs(n,sapro(2:slen+1));
Sn=sapro(2:slen+1);

                                    % low-pass filter
b=fir1(100,.1);
Sa = conv2(Sn,b,'same');

subplot(3,1,1):plot(s,'r');
grid on
title('Delta Modulation (red:input signal), (green:output signal from dekoder), (blue: filter
output)');
ylabel('amplitude');
subplot(3,1,2):plot(Sn,'g'); grid on
ylabel('amplituda');
subplot(3,1,3):plot(Sa,'b'); grid on
xlabel('index, n');
ylabel('amplitude');
figure
plot(xx,yy);
figure
bar(delta);
figure
bar(cn);
```

10.MODULATIONS WITH THE SPREAD SPECTRUM

The essence of **the transmission methods using the spread spectrum** is the broadband transmission, with the level of the signal that is significantly **below the level of noises.**
The combination of these two signal properties allows to reach the assumed value of the channel bandwidth, and it also causes the signal to be difficult to trace. Hence, the primary application of the spread spectrum systems was mainly in the military communication. However, over time the broadband transmission was applied in the public communication as well, for example, in mobile telephony UMTS, GPS systems, and in the Bluetooth standard.
Several types of the transmission methods which are using the spread spectrum can be differentiated.
The most commonly used are only two of them. The first one is:
DSSS (Direct Sequence Spread Spectrum) – the technique that uses the pseudorandom string directly with the spread spectrum.
The second method is **FHSS (Frequency Hopping Spread Spectrum)** that involves systems with a pseudorandom sequence of impulses over time.
The examples of a useful spread sequences:
Pseudorandom sequence
Gold sequence
Barker sequence

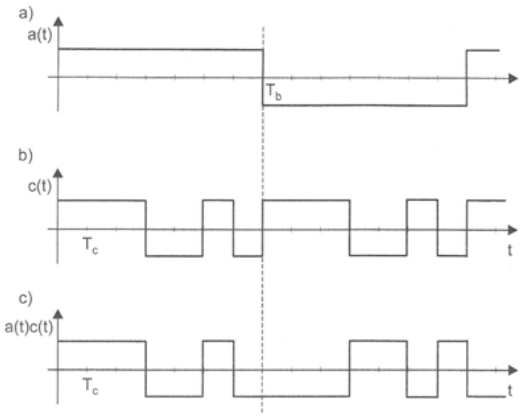


Fig. 24. A generation of the signal with a spread spectrum by the means of: a) sequence of data, b) spread sequence, c) the outcome

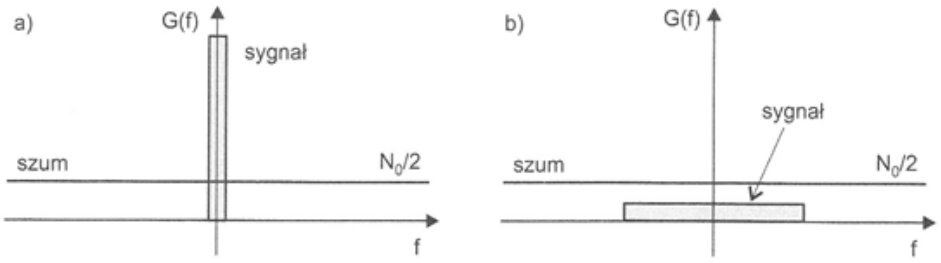


Fig. 25. A picture presenting the influence of the spread spectrum on the spectral properties of the signal, in the presence of additive white gaussian noise, on the basis of the power density spectrum and the noise for the system: a) narrowband, b) spread spectrum with the same signal power

An exemplary MATLAB code describing the modulation rule by the spread spectrum is presented below:

```
clc
close all
```

```
clear all

R = 20;
S = 21;
bit_ciag = round(rand(1,R));
PAM = zeros(1,S*R);
                                        % DSSS modulation
for i = 1:R
   if bit_ciag(1,i)== 0
      PAM(1+(i-1)*S : i*S) = -1;
   else
      PAM(1+(i-1)*S : i*S) =  1;
   end
end
figure
subplot(3,2,1)
stairs(PAM);
axis([-1 S*R+10 -1.2 1.2]);
grid on
title('PAM input signal ');
                                        %Spread sequence
pn_seq = round(rand(1,60));
t = 0:2*pi/6:2*pi;
c = sin(t);
carrier=[];
pn_sig_txr1 = zeros(1,S*R);         % Noise signal
for i = 1:60
   if pn_seq(1,i)==0
      pn_sig_trx1(1+(i-1)*7:i*7) = -1;
   else
      pn_sig_trx1(1+(i-1)*7:i*7) =  1;
   end
   carrier=[carrier c];
end

K = round(rand(1,1)*420);
pn_sig_trx = [pn_sig_trx1(K:end) pn_sig_trx1(1:K-1)];
                                        % Spreading of sequence
spreaded_sig = PAM .* pn_sig_trx;
subplot(3,2,3);
stairs(spreaded_sig);
axis([-1 S*R+10 -1.2 1.2]);
title('Spread signal');

z = abs(fft(xcorr(spreaded_sig)));   % Spectrum spread sequence
subplot(3,2,4);
plot(z);
axis([0 840 0 1.1*max(z)]);
title('Spectrum spread sequence');

                                        % BPSK modulation
bpsk_sig = spreaded_sig .* carrier;
subplot(3,2,5);
stairs(bpsk_sig)
axis([-1 150 -1.2 1.2]);
title('BPSK modulation');

                                        %Spectrum BPSK signal.
subplot(3,2,6);
y = abs(fft(xcorr(bpsk_sig)));
```

```
plot(y)
xlabel('Frequency (Hz)')
ylabel('Spectrum power')
title(' Spectrum BPSK signal')
```

REFERENCES:

1. Simon Haykin, „Systemy telekomunikacyjne 1", WKŁ Warszawa 2004, ISBN 83-206-1273-X
2. Krzysztof Wesołowski, "Podstawy cyfrowych systemów telekomunikacyjnych", WKŁ Warszawa 2006, ISBN 978-83-206-1508-1
3. Richard G. Lyons, "Wprowadzenie do cyfrowego przetwarzania sygnałów", WKŁ Warszawa 2003, ISBN 83-206-1318-3
4. Wiera Dziech, „ Telekomunikacja analogowa i cyfrowa-ćwiczenia z wykorzystaniem programu Mathcad", Wydawnictwo PŚk 2009,ISSN 0239-6386
5. http://www.mathworks.com/