**Obraz zawierający logo, tekst, Czcionka, Grafika

Opis wygenerowany automatycznie**

**Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,**

**INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Raport

**Podstawy telekomunikacji**

Autor: Grzegorz Lis, Karolina Sawosz

Kierunek studiów: Mikroelektronika w Technice i Medycynie

# Kraków, 2024

**Laboratorium 5**

***Opis problemu***

Celem laboratoriów było dokonanie modulacji oraz demodulacji kwadraturowej sygnału, zarówno analogowej, jak i cyfrowej. Należało sprawdzić ortogonalność funkcji (analitycznie oraz iloczynem skalarnym). Ostatnim etapem było przeprowadzenie modulacji i demodulacji  
z nałożonymi szumami.

***Podstawy matematyczne rozwiązania***

Ortogonalność niezerowych funkcji:

Modulacja kwadraturowa analogowa:

i(t) – składowa synfazowa,

q(t) – składowa kwadraturowa,

x(t) – sygnał zmodulowany transmitowany w kanale.

Modulacja kwadraturowa cyfrowa:

N – dyskretna chwila czasu,

ωn = ωc/Fp – znormalizowana względem częstotliwości próbkowania Fp pulsacja sygnału nośnego.

***Symulacje i obserwacje***

Zadanie 1

Ortogonalność funkcji policzona analitycznie:

Jak widać na powyższych obliczeniach, sinus oraz cosinus są funkcjami ortogonalnymi, jeżeli mają taką samą fazę.

Listing 1. Sprawdzenie ortogonalności funkcji sinus i cosinus.

orthogonal(sin(t),cos(t))

orthogonal(sin(2\*t),cos(2\*t))

orthogonal(sin(2\*t),cos(543\*t))

orthogonal(sin(t+pi/100),cos(t+2\*pi/100))

orthogonal(sin(t+pi/100),cos(t+pi/100))

function y = orthogonal(a, b)

if length(a) == length(b)

c = a.\*b;

c = sum(c);

if round(c,10) == 0

y=1;

else

y=0;

end

else

y=0;

end

end

Obraz zawierający zrzut ekranu, tekst, linia, Czcionka

Opis wygenerowany automatycznie

Figure . Wynik funkcji orthogonal.

Wyniki potwierdzają obserwację z obliczeń analitycznych – sinus oraz cosinus o takim samym przesunięciu fazowym są ortogonalne. Sygnały są ortogonalne niezależnie od częstotliwości podanej w argumencie funkcji.

Zadanie 2

Listing 2. Modulacja oraz demodulacja QAM analogowo.

%% Signal modulation

signal1 = A\*sin(t\*f1);

signal2 = A\*sin(t\*f1\*5);

carrier\_sin = A\*sin(t\*f2);

carrier\_cos = A\*cos(t\*f2);

modulated = (signal1.\*carrier\_sin) - (signal2.\*carrier\_cos);

%% Demodulation

modulated\_sin = modulated .\* carrier\_sin;

modulated\_cos = -modulated .\* carrier\_cos;

demodulated\_sin = lowpass(modulated\_sin,f2,fs);

demodulated\_cos = lowpass(modulated\_cos,f2,fs);

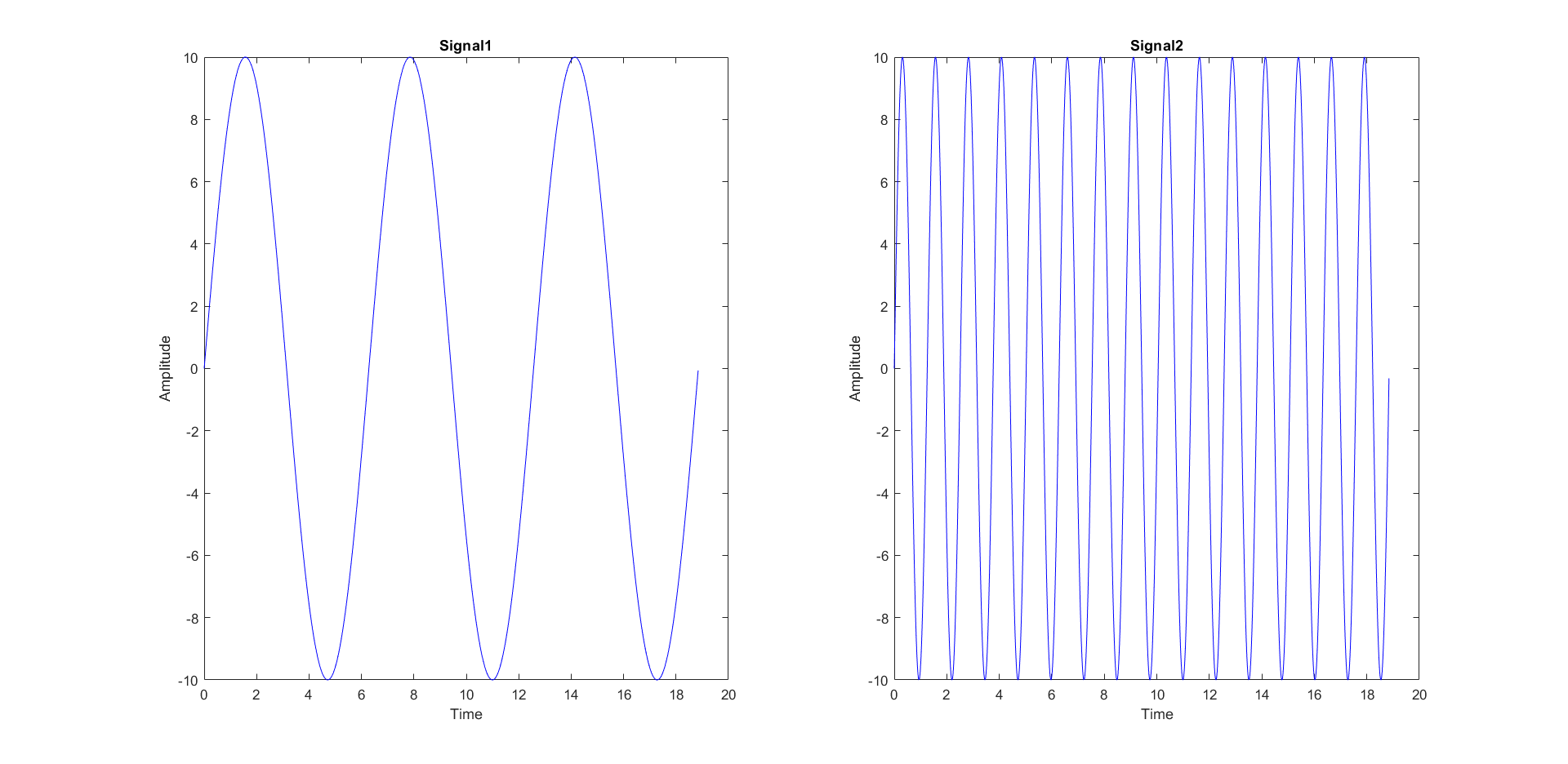


Figure . Sygnały do modulacji.

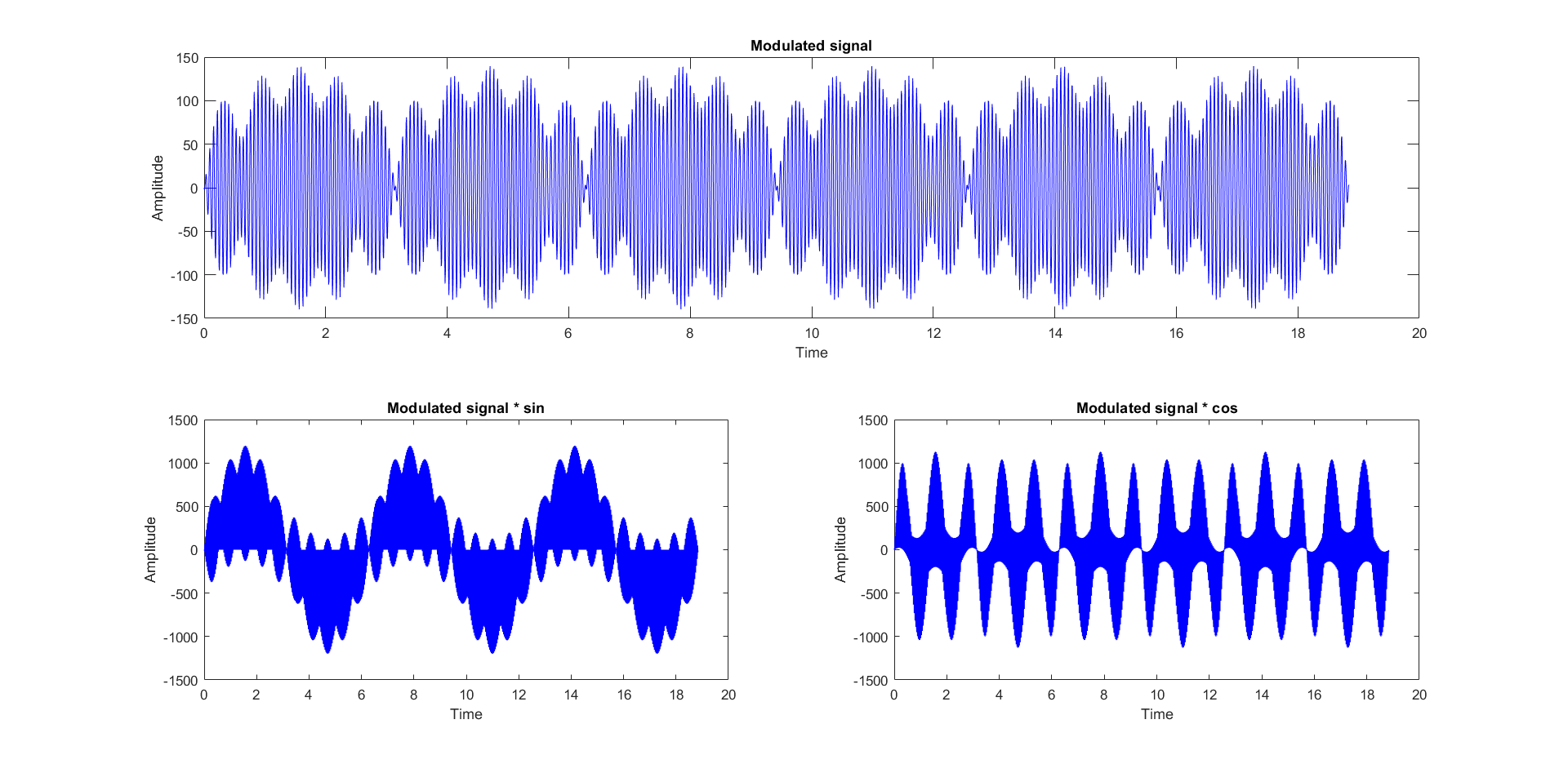


Figure . Sygnały zmodulowane i przemnożone przez nośną.

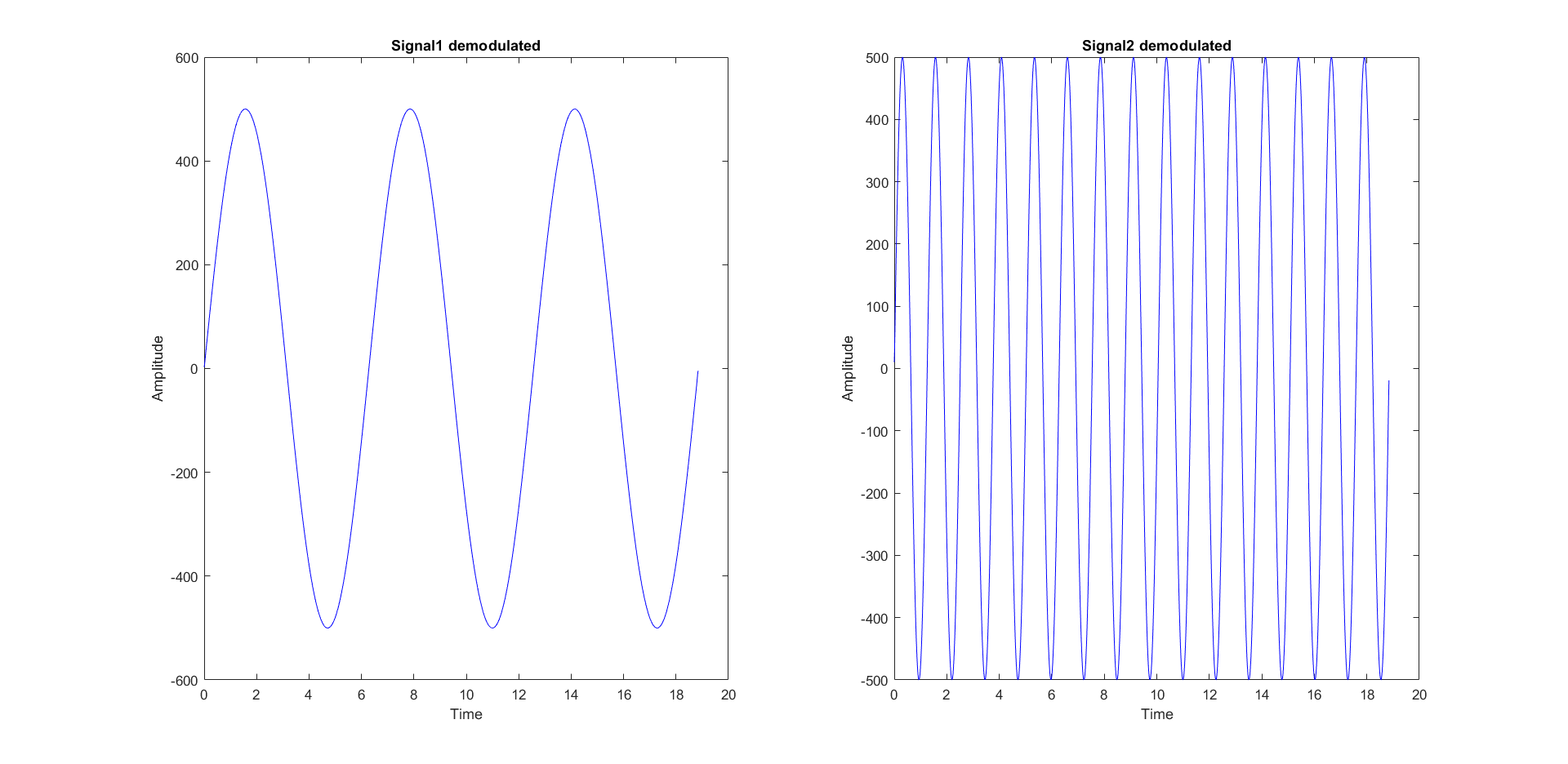


Figure 4. Sygnały po demodulacji

Listing 3. Modulacja oraz demodulacja QAM analogowo z przesunięciem fazowym.

%% Signal modulation with phase shift

signal1\_phase = A\*sin(t\*f1+phase);

signal2\_phase = A\*sin(t\*f1\*5+phase);

carrier\_sin\_phase = A\*sin(t\*f2+phase);

carrier\_cos\_phase = A\*cos(t\*f2+phase);

modulated\_phase = (signal1\_phase.\*carrier\_sin\_phase) - (signal2\_phase.\*carrier\_cos\_phase);

%% Demodulation

modulated\_sin = modulated\_phase .\* carrier\_sin;

modulated\_cos = -modulated\_phase .\* carrier\_cos;

demodulated\_sin = lowpass(modulated\_sin,f2,fs);

demodulated\_cos = lowpass(modulated\_cos,f2,fs);

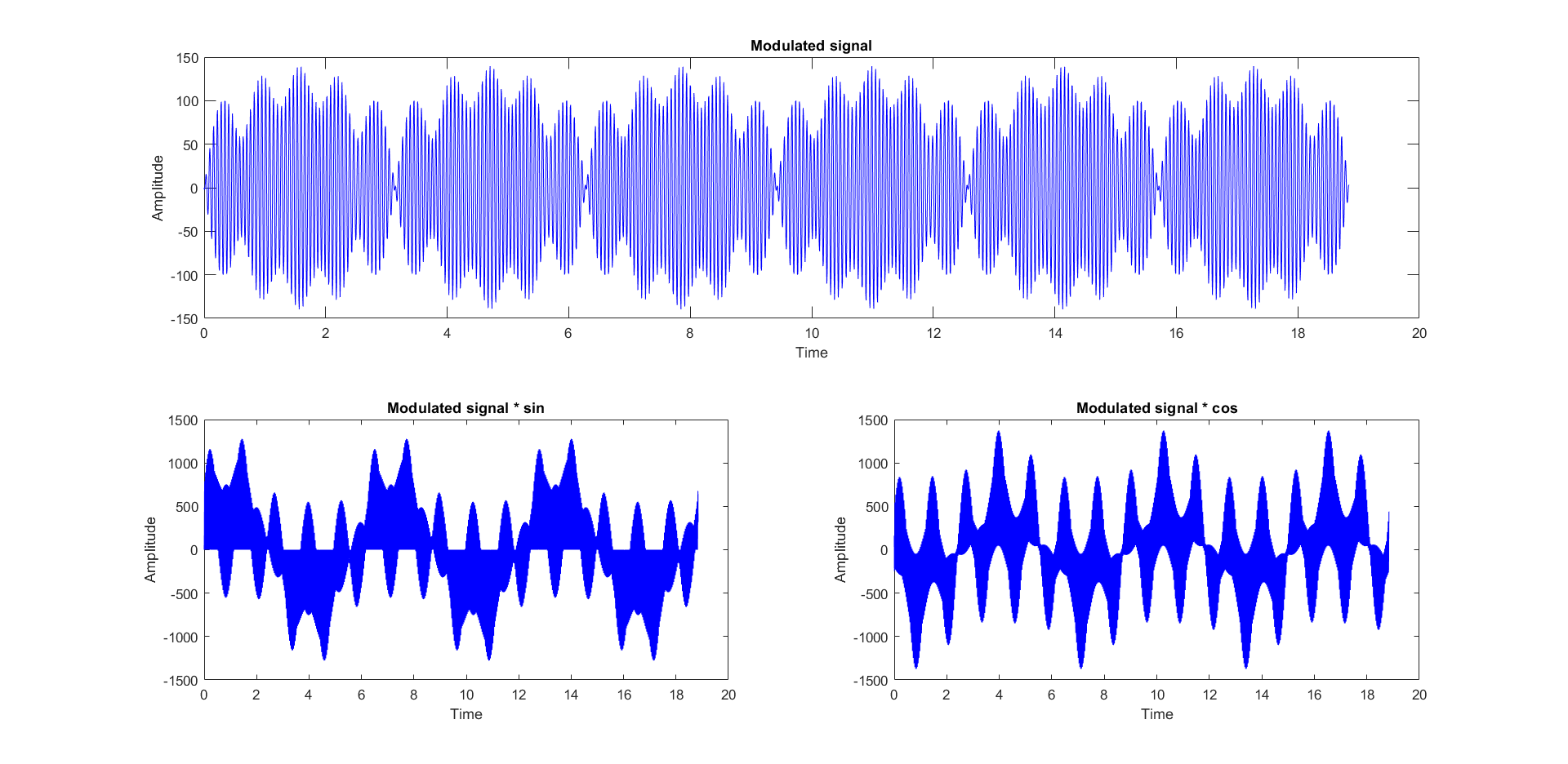


Figure . Modulacja z przesunięciem fazy na kanale.

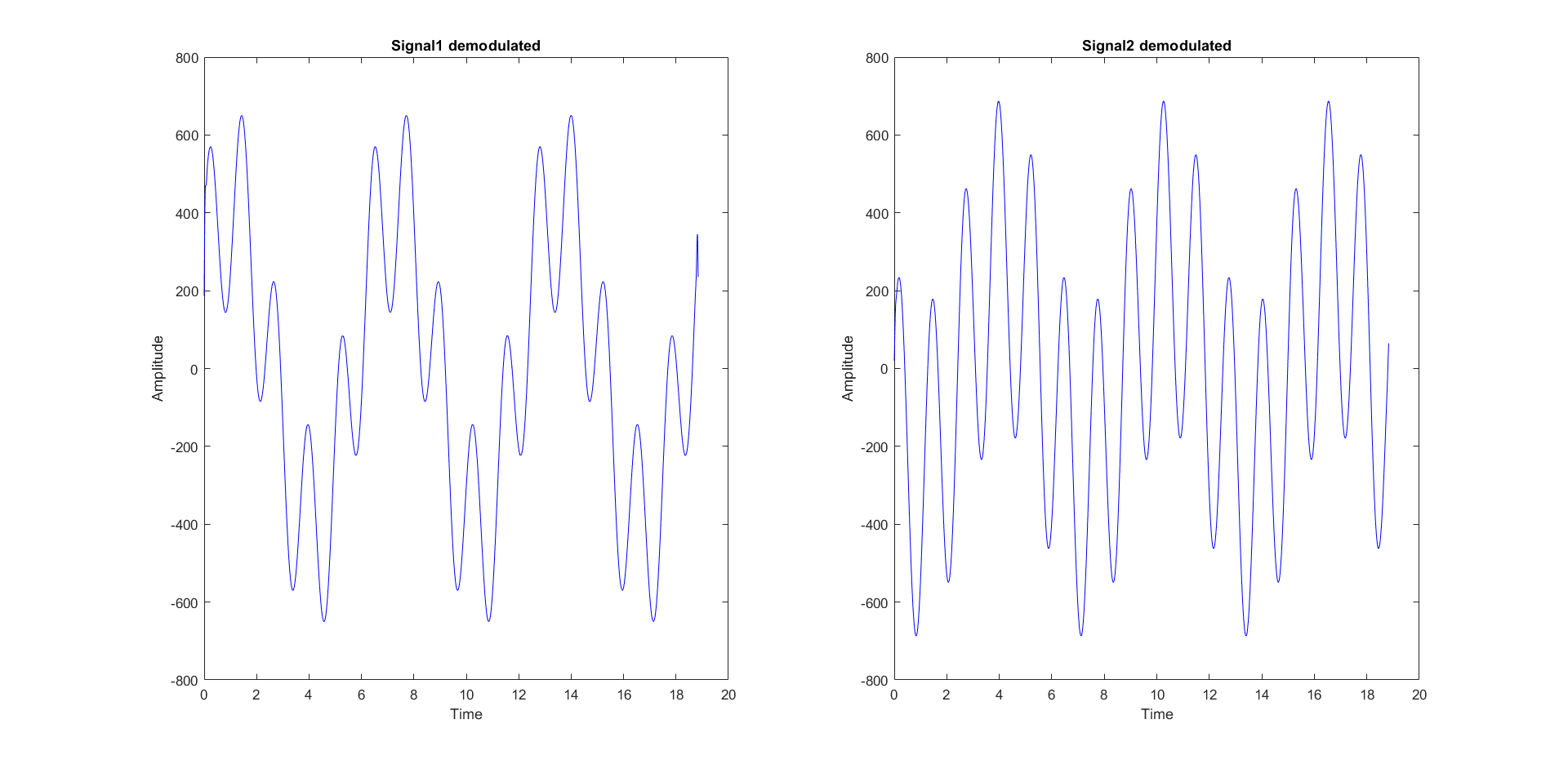


Figure . Sygnały po zdemodulowaniu.

Modulacja sygnału polega na różnicy sygnałów pomnożonych przez nośne sinusa oraz cosinusa. Ważne jest, żeby nośne były ortogonalne względem siebie i miały dużo większą częstotliwość niż sygnał. Demodulacja polega na pomnożeniu otrzymanego sygnału przez odpowiednie nośne. Następnie należy odciąć wyższe częstotliwości, aby odciąć nośną (zastosowanie filtru dolnoprzepustowego). Sygnał zdemodulowany za pomocą QAM analogowo jest bardzo dokładny – przebieg sygnału jest identyczny. Niestety, dla przebiegu z symulacją opóźnienia kanału demodulacja jest niedokładna.

Zadanie 3

Obraz zawierający diagram, Plan, Rysunek techniczny, wykres

Opis wygenerowany automatycznie

Figure . Źródło: MATLAB i podstawy telekomunikacji, Jacek Izydorczyk i inni, 2017-06-26

Poniższe wykresy przedstawiają modulację 16QAM.

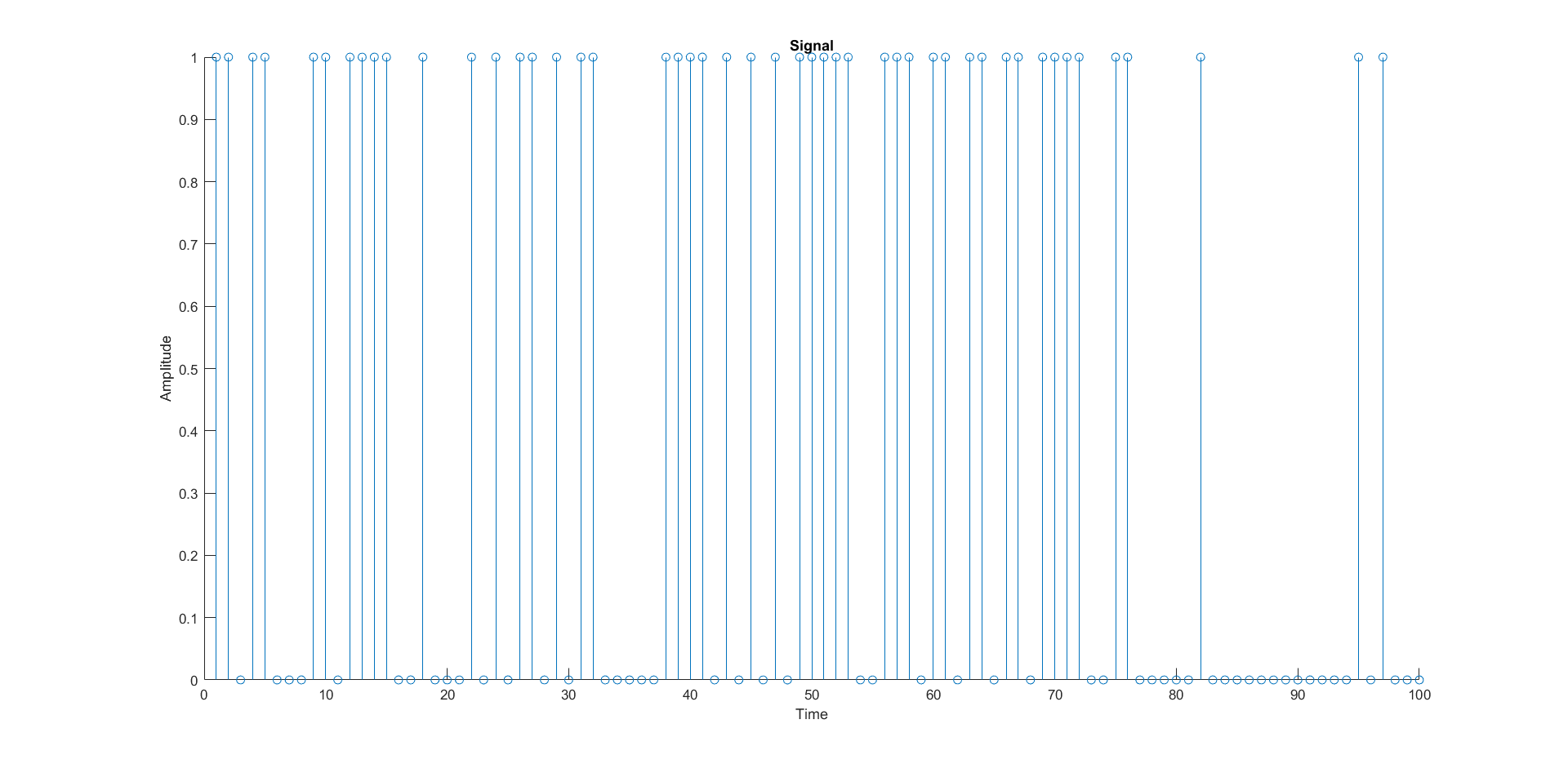


Figure . Sygnał wejściowy.

Listing 4. Implementacja kodera.

dataIn = randi([0 1],number\_of\_samples,1);

dataSymbolsIn = bit2int(dataIn,log2(qam\_number));

data = qammod(dataSymbolsIn,qam\_number,'bin');

real\_part = real(data);

imag\_part = imag(data);

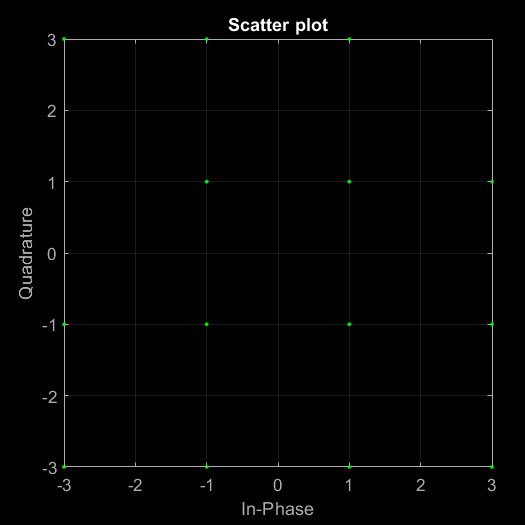


Figure . Wynik kodera.

Listing 5. Interpolacja części rzeczywistej i urojonej.

real\_interpolation = zeros(1,(length(data)\*zeros\_between\_data));

imag\_interpolation = zeros(1,(length(data)\*zeros\_between\_data));

for i=1:length(data)

real\_interpolation(i\*zeros\_between\_data) = real\_part(i);

imag\_interpolation(i\*zeros\_between\_data) = imag\_part(i);

end

real\_interpolation = [real\_interpolation(zeros\_between\_data:length(real\_interpolation)),zeros(1,zeros\_between\_data-1)];

imag\_interpolation = [imag\_interpolation(zeros\_between\_data:length(imag\_interpolation)),zeros(1,zeros\_between\_data-1)];

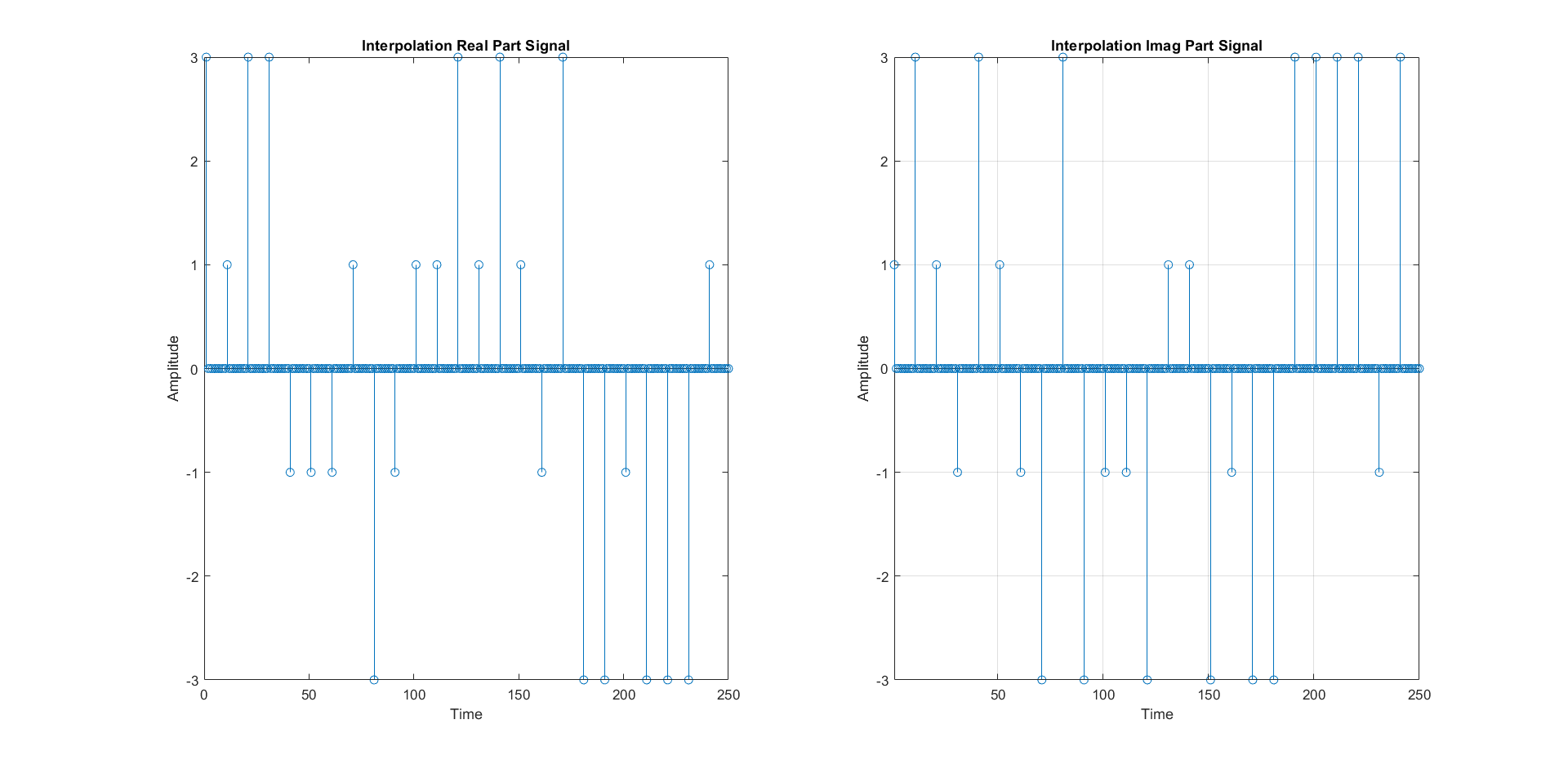


Figure 10. Wynik interpolacji.

Listing 6. Filtr kształtujący.

real\_pulsed = real\_interpolation;

imag\_pulsed = imag\_interpolation;

for i=2:length(real\_interpolation)

if ((real\_pulsed(i)==0) && (imag\_pulsed(i)==0))

real\_pulsed(i) = real\_pulsed(i-1);

imag\_pulsed(i) = imag\_pulsed(i-1);

end

end

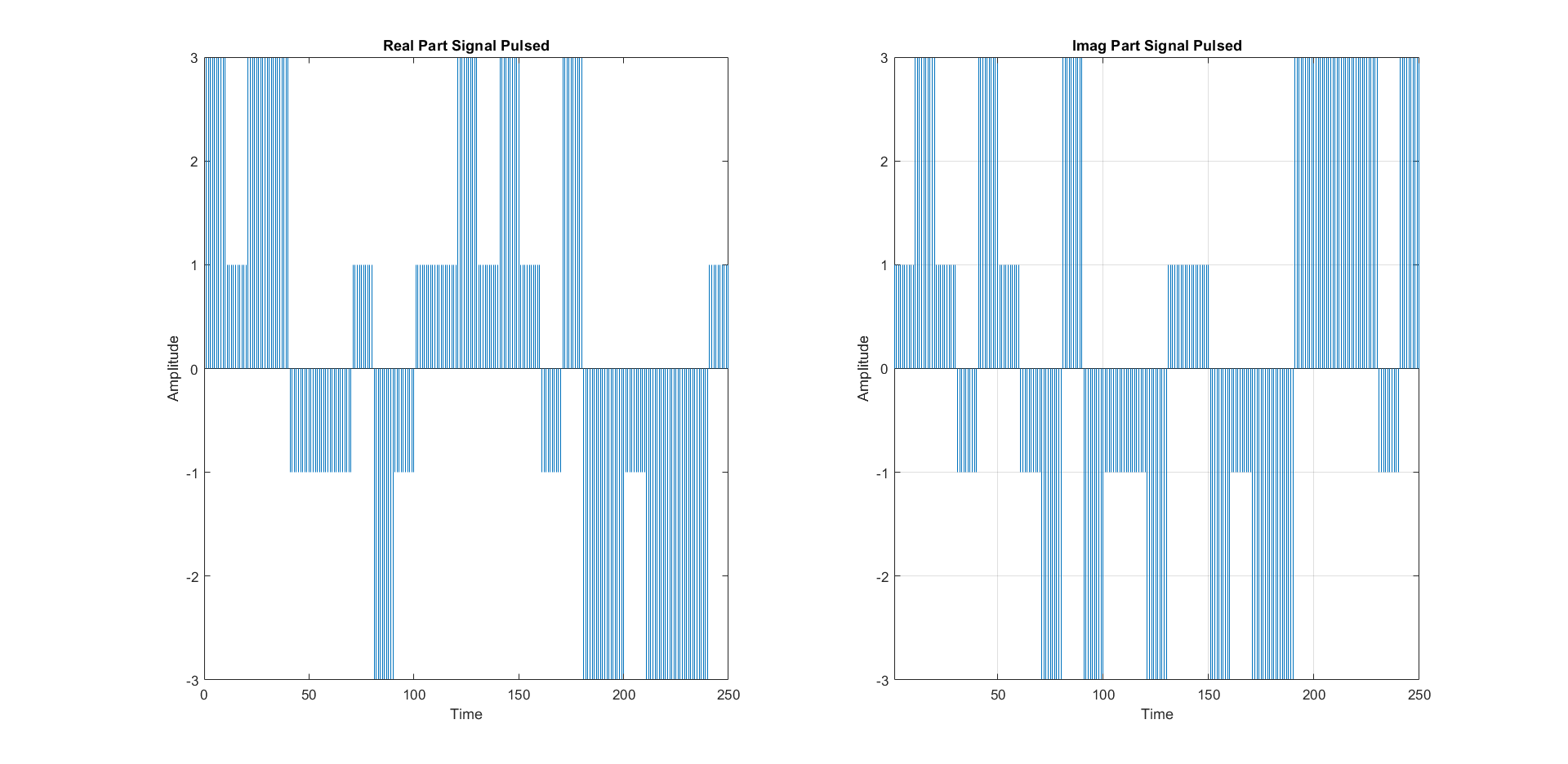


Figure 11. Wynik filtru kształtującego.

Listing 7. Pomnożenie przez nośne.

t\_mod = linspace(0,1,length(real\_pulsed));

carrier\_freq = 50;

real\_mod = 2\*cos(2\*pi\*carrier\_freq\*t\_mod).\*real\_pulsed;

imag\_mod = 2\*sin(2\*pi\*carrier\_freq\*t\_mod).\*imag\_pulsed;

modulated\_signal = real\_mod + imag\_mod;

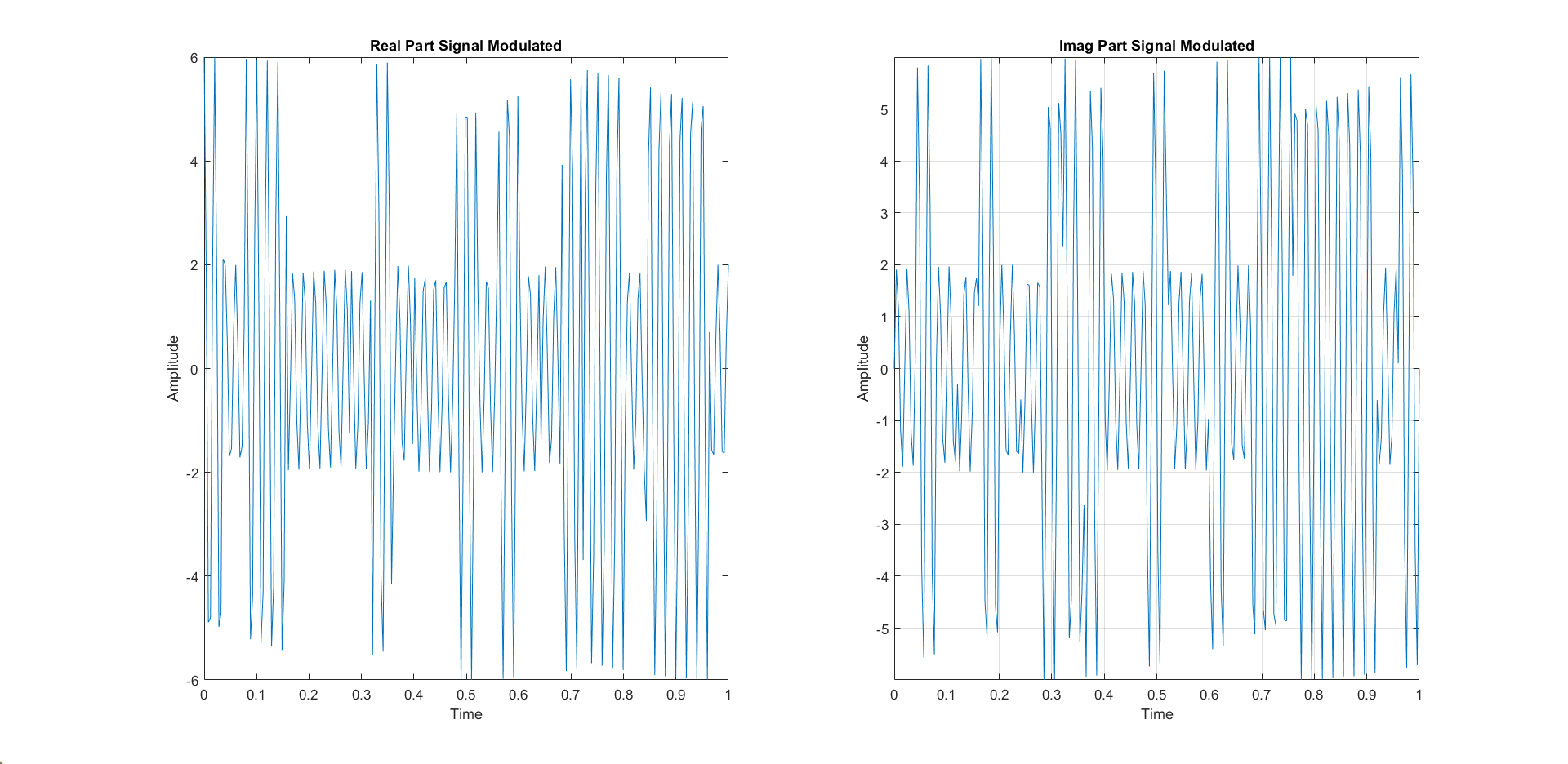


Figure . Wynik pomnożenia przez nośne.

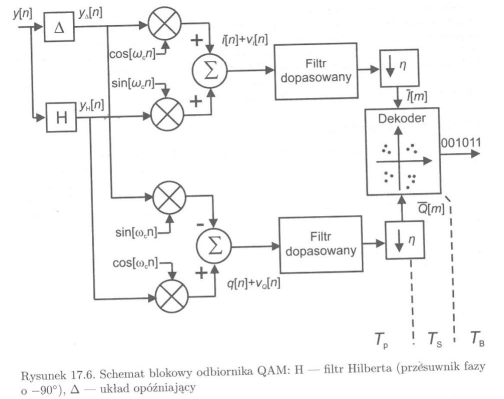


Figure . Źródło: MATLAB i podstawy telekomunikacji, Jacek Izydorczyk i inni, 2017-06-26.

Listing 8. Filtr Hilberta, układ opóźniający, filtr dopasowany.

%% Demodulation

shifted\_90=imag(hilbert(modulated\_signal-mean(modulated\_signal)));

demod1 = (modulated\_signal.\*cos(2\*pi\*carrier\_freq\*t\_mod))+(shifted\_90.\*sin(2\*pi\*carrier\_freq\*t\_mod));

demod2 = (-modulated\_signal.\*sin(2\*pi\*carrier\_freq\*t\_mod))+(shifted\_90.\*cos(2\*pi\*carrier\_freq\*t\_mod));

filtered1 = lowpass(demod1,carrier\_freq,1000);

filtered2 = lowpass(demod2,carrier\_freq,1000);

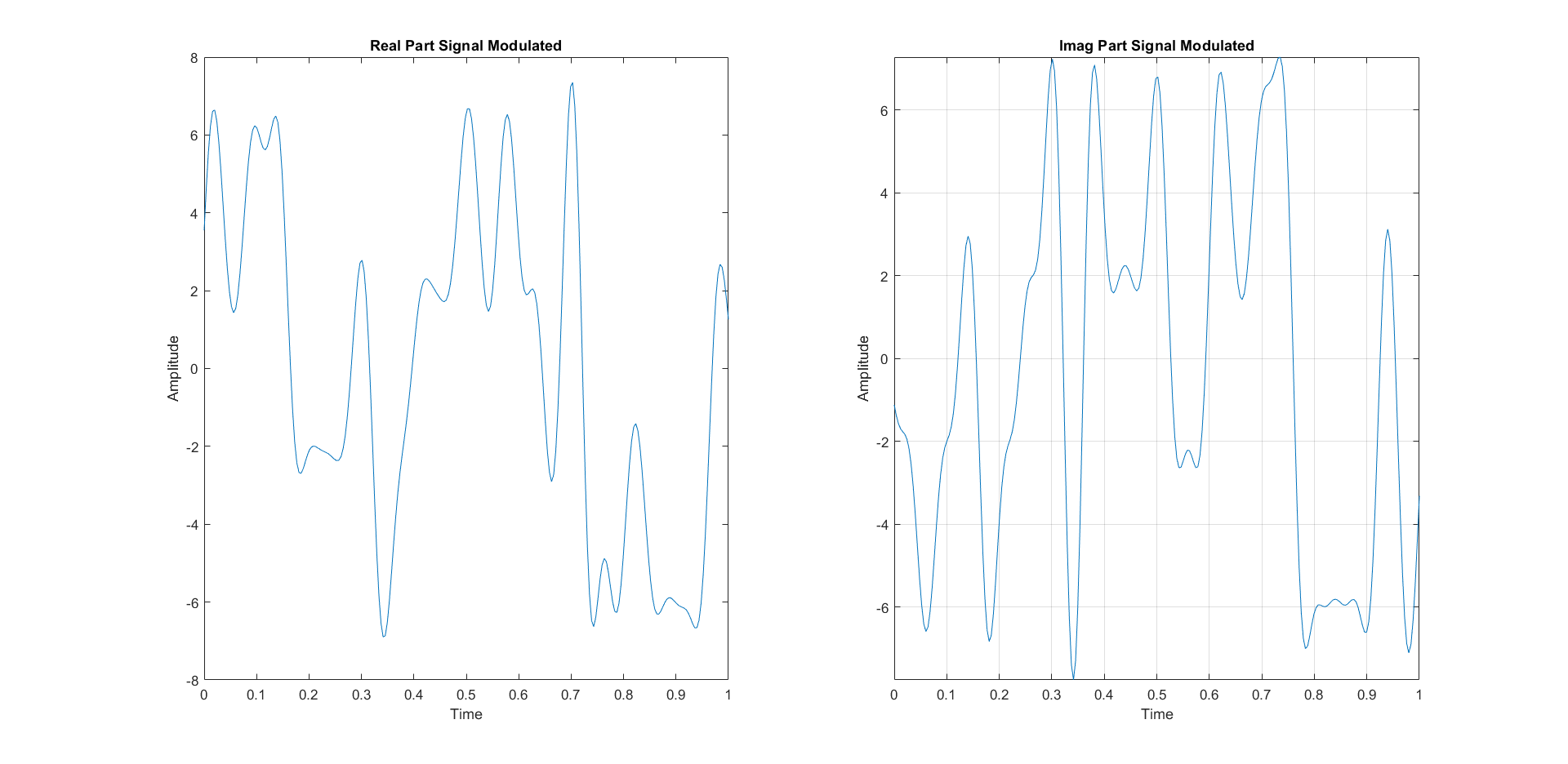


Figure . Wynik po filtrze dopasowującym.

Listing 9. Downsampling.

filtered1\_down = downsample(filtered1./2,zeros\_between\_data,2);

filtered2\_down = downsample(filtered2./2,zeros\_between\_data,2);

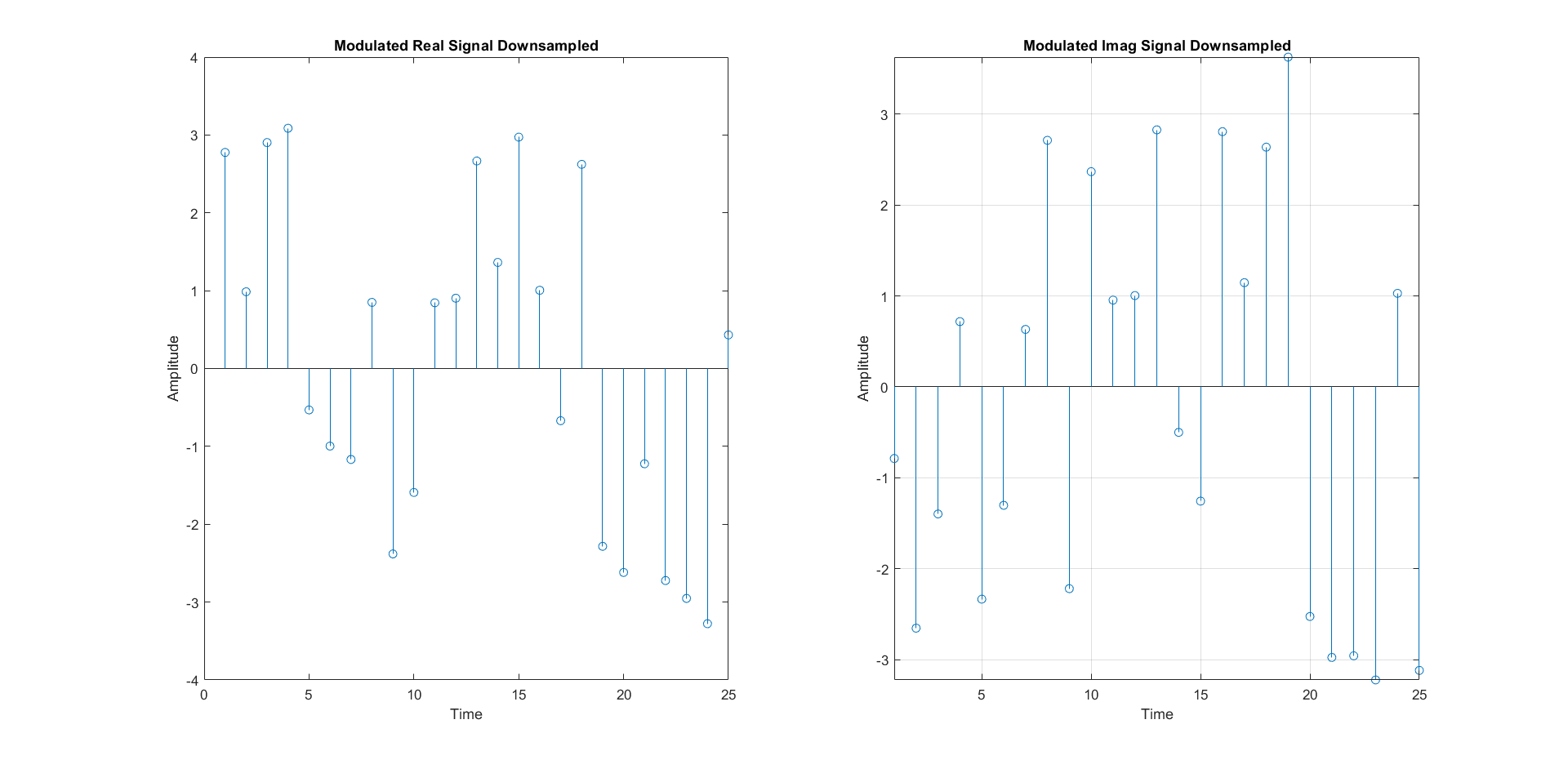


Figure 15. Wynik downsamplingu.

Listing 10. Implementacja dekodera.

demod = reshape(qamdemod(filtered1\_down-1i\*filtered2\_down,qam\_number,'bin',OutputType = 'bit'),1,[]);

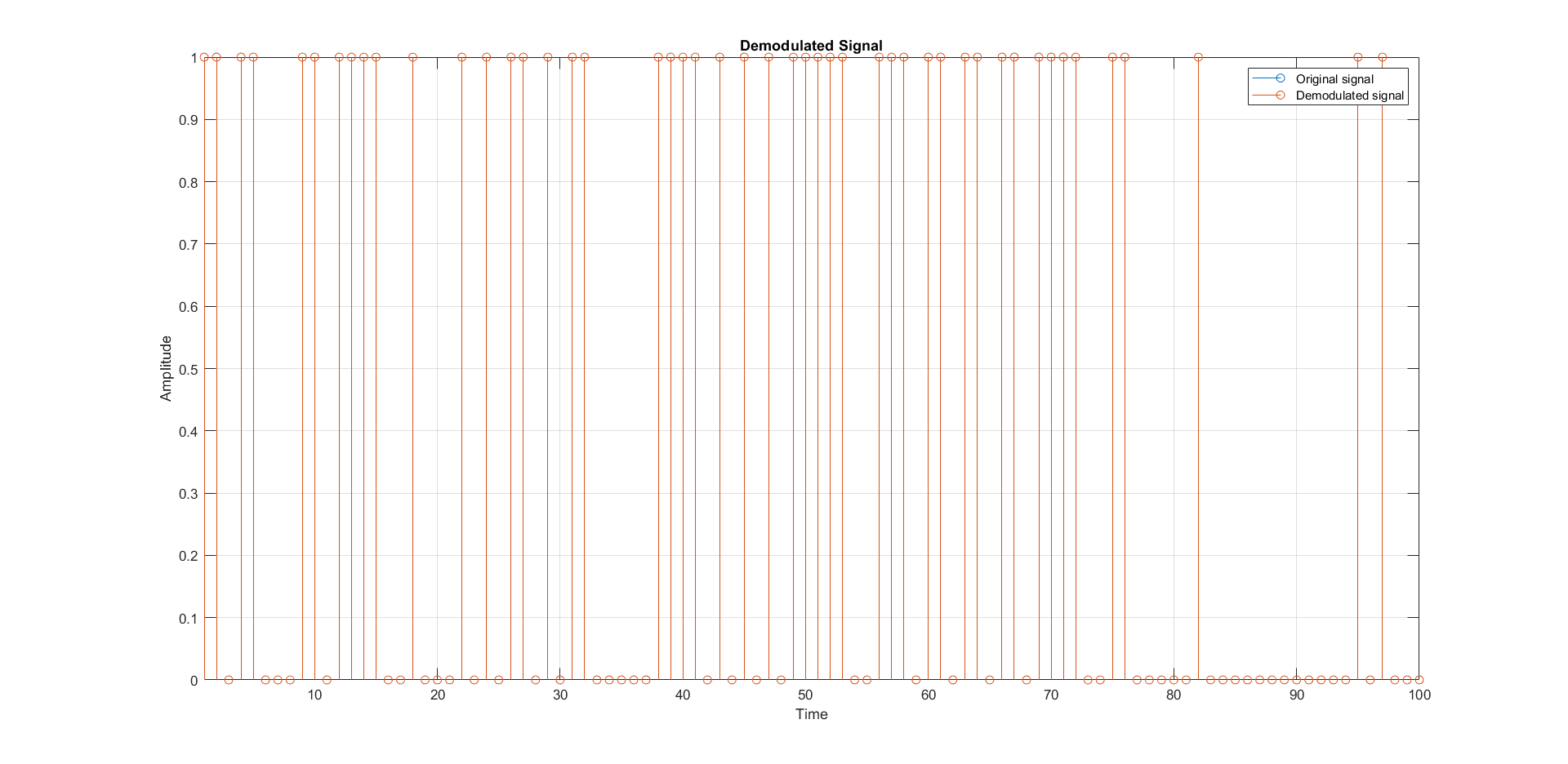


Figure . Wynik dekodera.

Wyniki dla modulacji 4QAM – bez pokazania poszczególnych etapów, ponieważ są analogiczne do 16QAM.

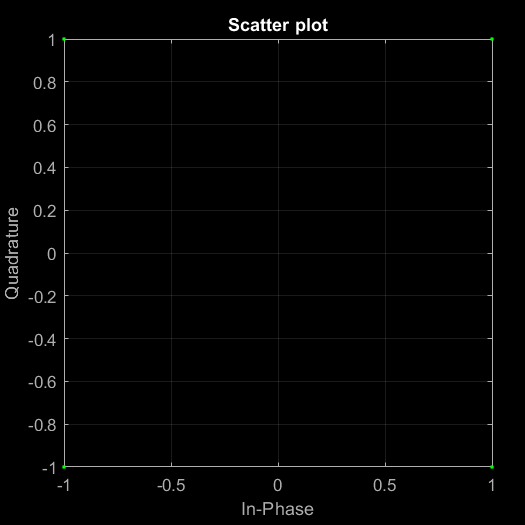


Figure . Wynik koder.

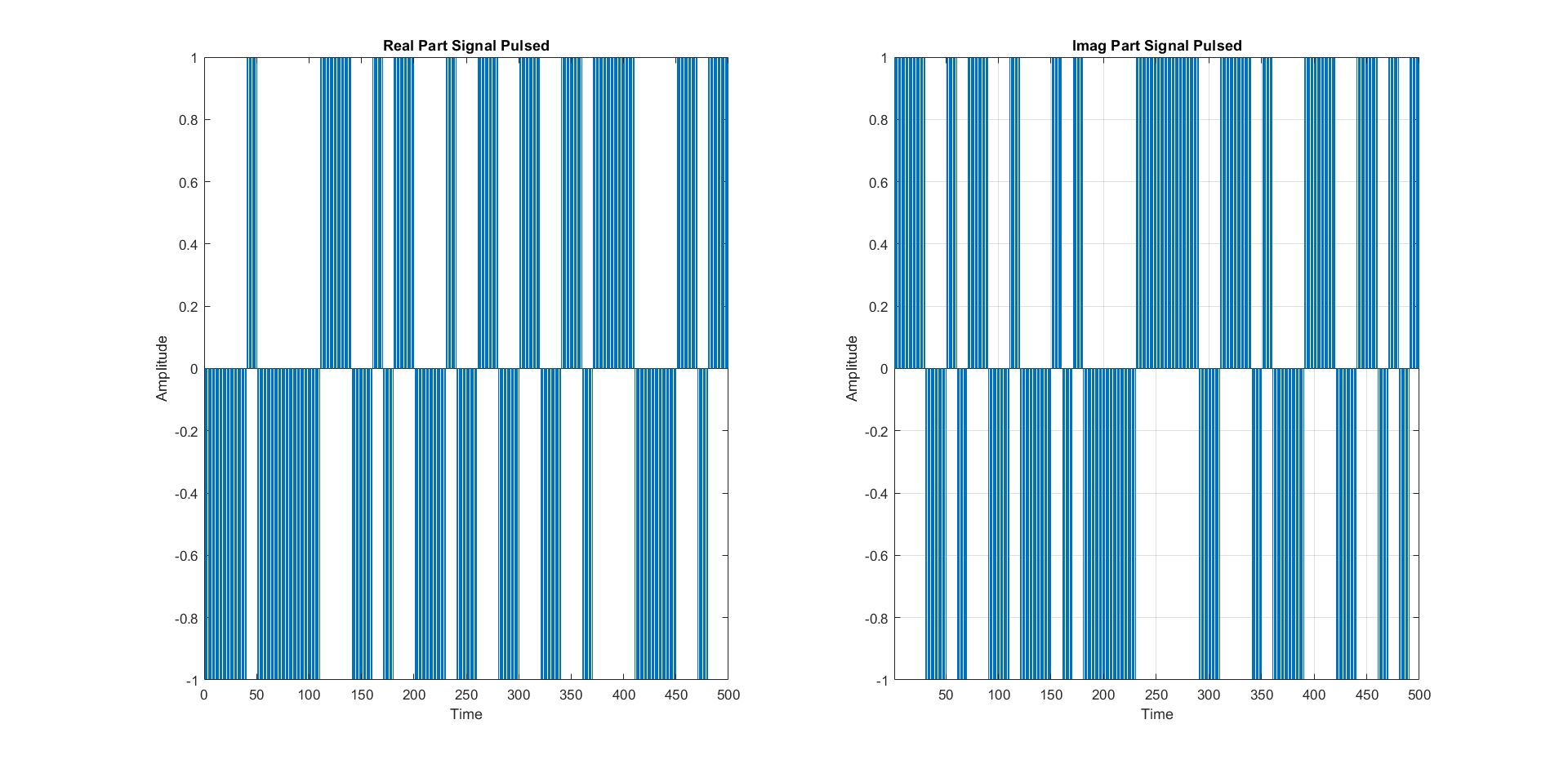


Figure . Filtr kształtujący.

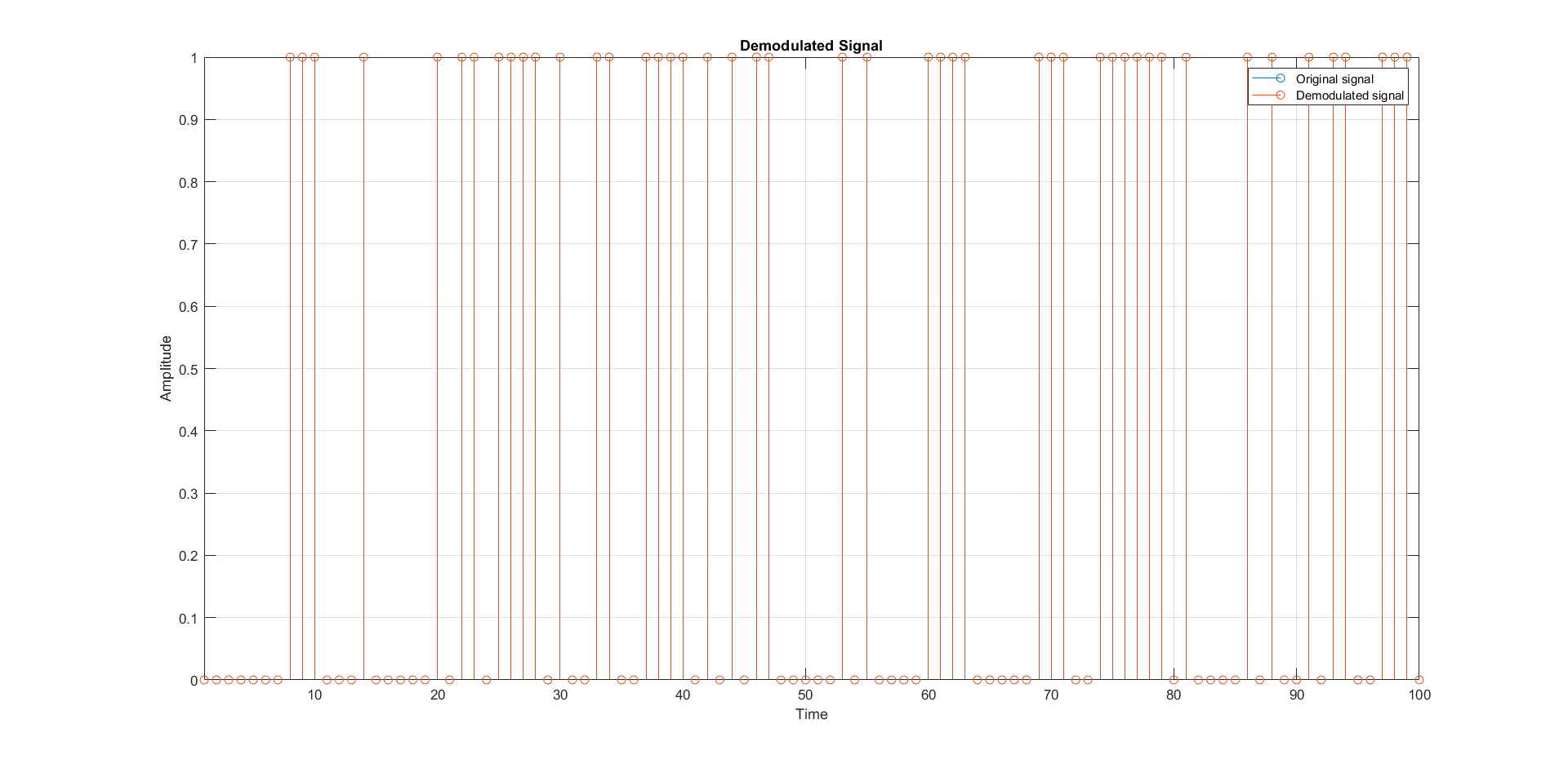


Figure . Wynik dekodera.

W modulacji cyfrowej QAM na wejście kodera podajemy liczby o rozmiarze log2(numer\_modulacji) bitów. Następnie koder rozmieszcza je w odpowiedniej części konstelacji, która odpowiada za liczby urojone i rzeczywiste. Kolejnym etapem jest interpolacja sygnału, aby przenieść go w niższe częstotliwości, poszerzenie prążków, aby ułatwić późniejsze odfiltrowanie danej. Tak przekształcony sygnał jest nakładany na nośne sinusoidalną oraz cosinusoidalną. Zmodulowane sygnały odejmujemy od siebie i uzyskujemy sygnał gotowy do wysłania. W celu demodulacji przesuwamy fazę o 90 stopni filtrem Hilberta i opóźniamy oryginalny sygnał, by skompensować opóźnienie filtra. Następnie modulujemy sygnał poprzez:

,

Oba otrzymane sygnały przepuszczamy przez filtr dolnoprzepustowy, wykonujemy downsampling oraz podajemy na wejście dekodera.

Modulacja 4QAM ma inną amplitudę niż 16QAM. Jak widać, bez ingerencji szumów sygnał po demodulacji jest identyczny jak przed.

Zadanie 4

Listing 11. Szum nałożony na kanał.

%% Add noise

snr=3;

modulated\_signal = awgn(modulated\_signal, snr, 'measured');

error\_number=0;

for i=1:length(demod)

if demod(i) ~= dataIn(i)

error\_number = error\_number+1;

end

end

error\_number

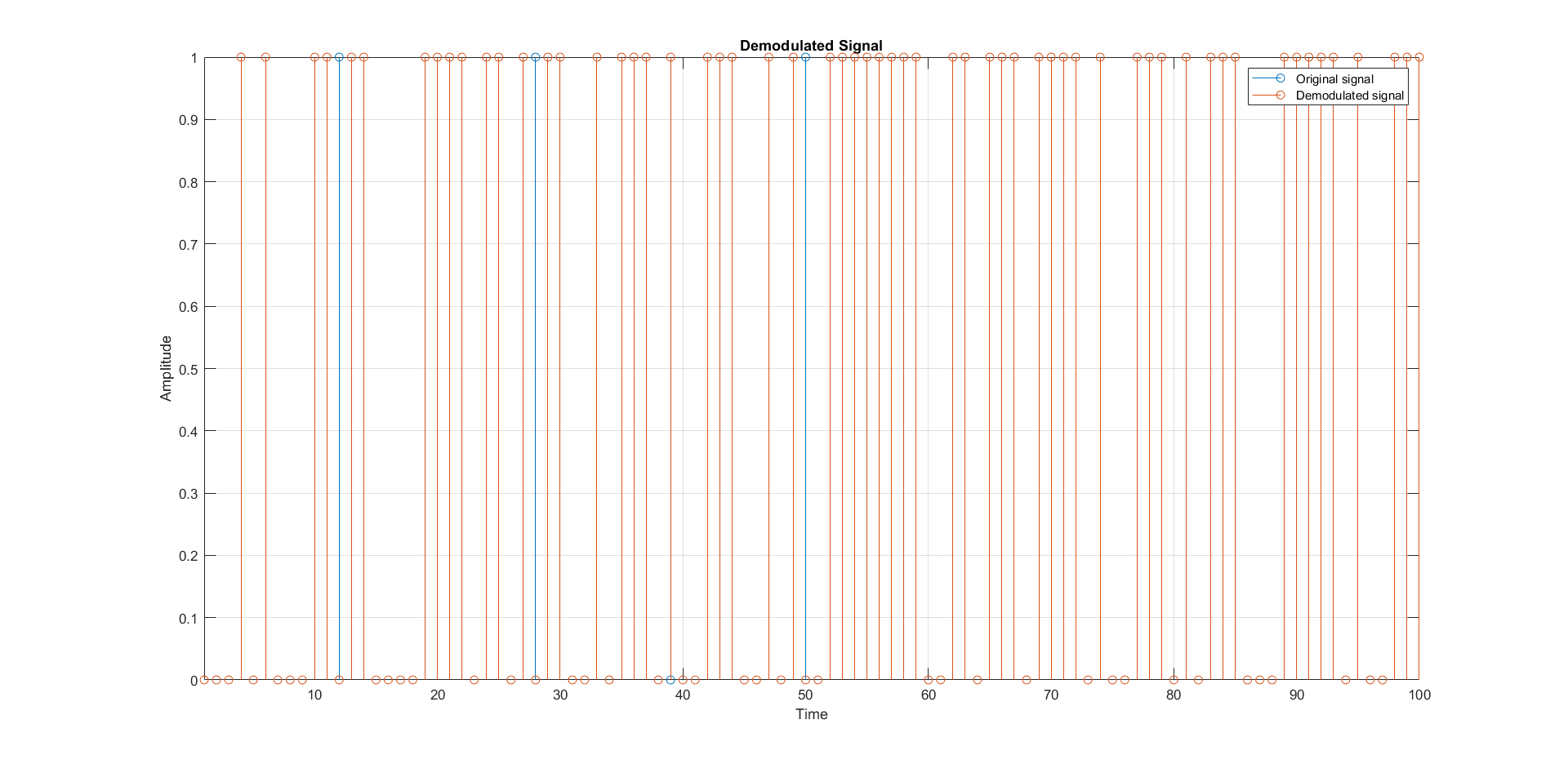


Figure . Sygnał po zdemodulowaniu dla 4QAM.

Obraz zawierający Czcionka, biały, tekst, zrzut ekranu

Opis wygenerowany automatycznie

Figure . Ilość odstępstw od oryginalnego sygnału dla 4QAM.

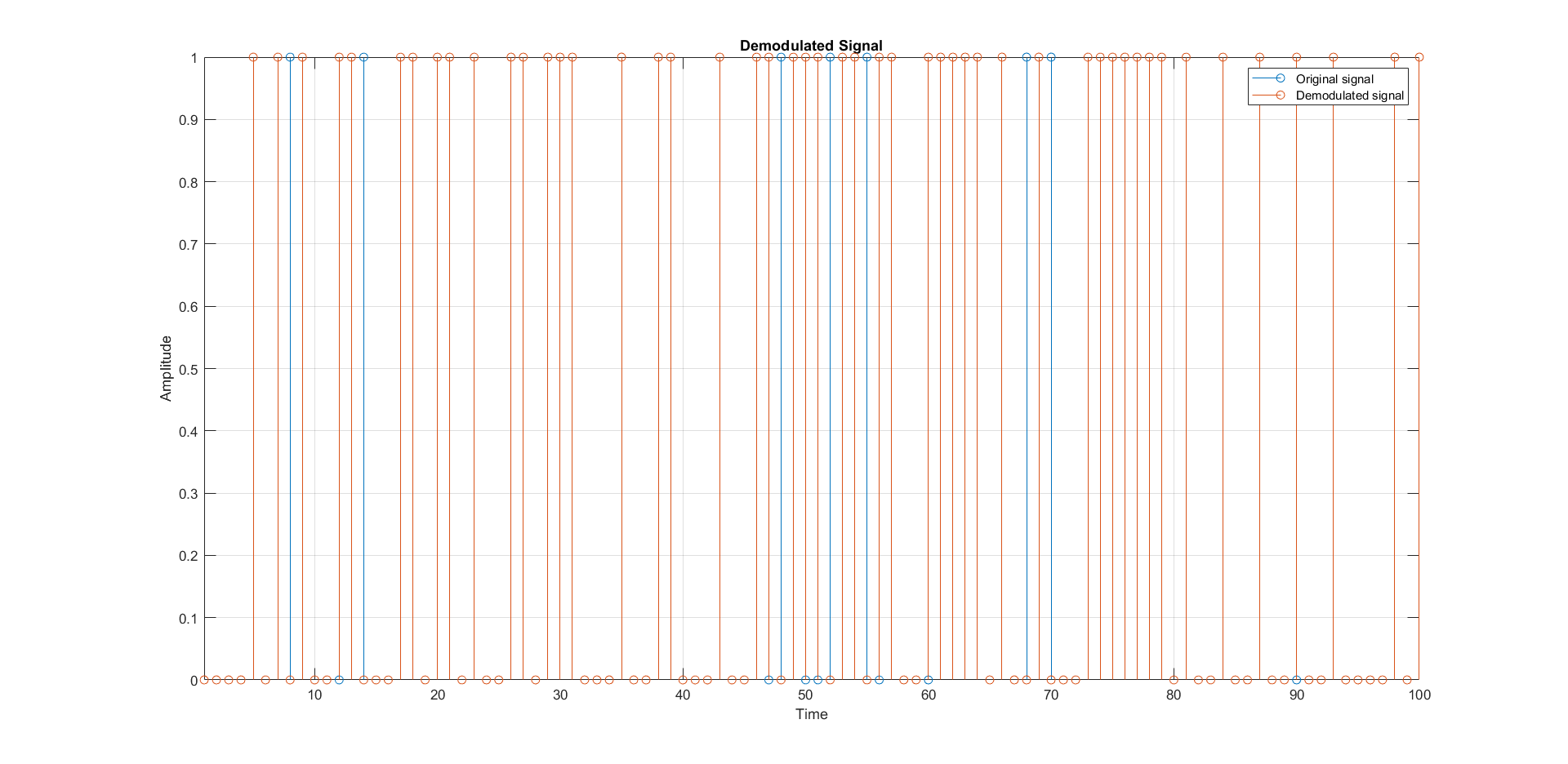


Figure . Sygnał po zdemodulowaniu dla 16QAM.

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Figure . Ilość odstępstw od oryginalnego sygnału dla 16QAM.

Szumy są dodawane po modulacji sygnału, w miejscu, w którym dane powinny być wysłane (kanał). W listingu 11. dodany został kod, którego wynik obrazuje ile odstępstw od oryginalnego sygnału nastąpiło po zdemodulowaniu.

Listing 12. Dodanie przesunięcia fazowego na kanał.

%% Add phase shift

phase\_shift = pi/2;

modulated\_signal = real(modulated\_signal .\* exp(1j \* phase\_shift));

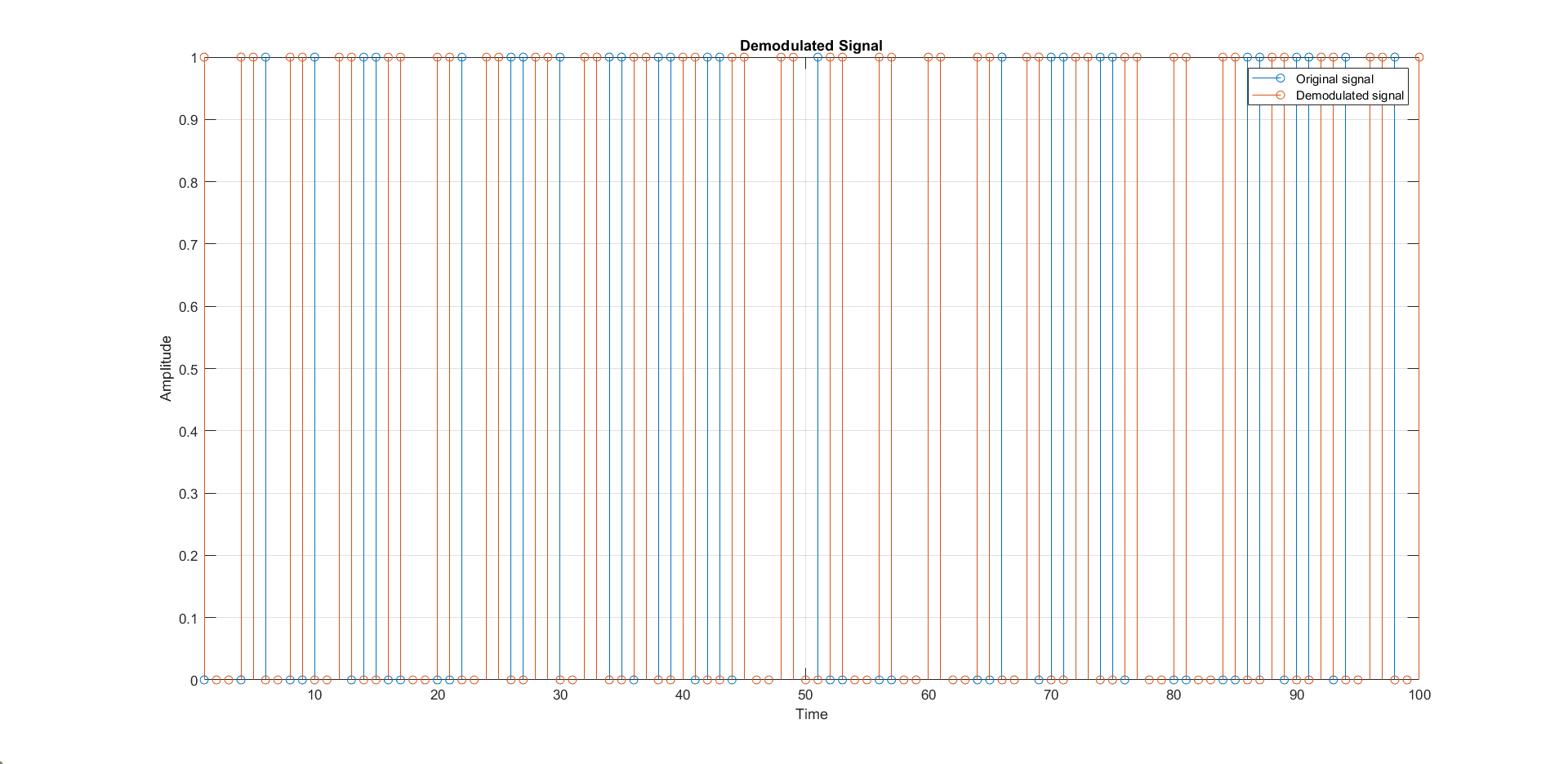


Figure . Sygnał po zdemodulowaniu dla 16QAM.

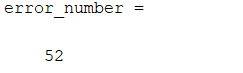


Figure . Ilość odstępstw od oryginalnego sygnału dla 16QAM.

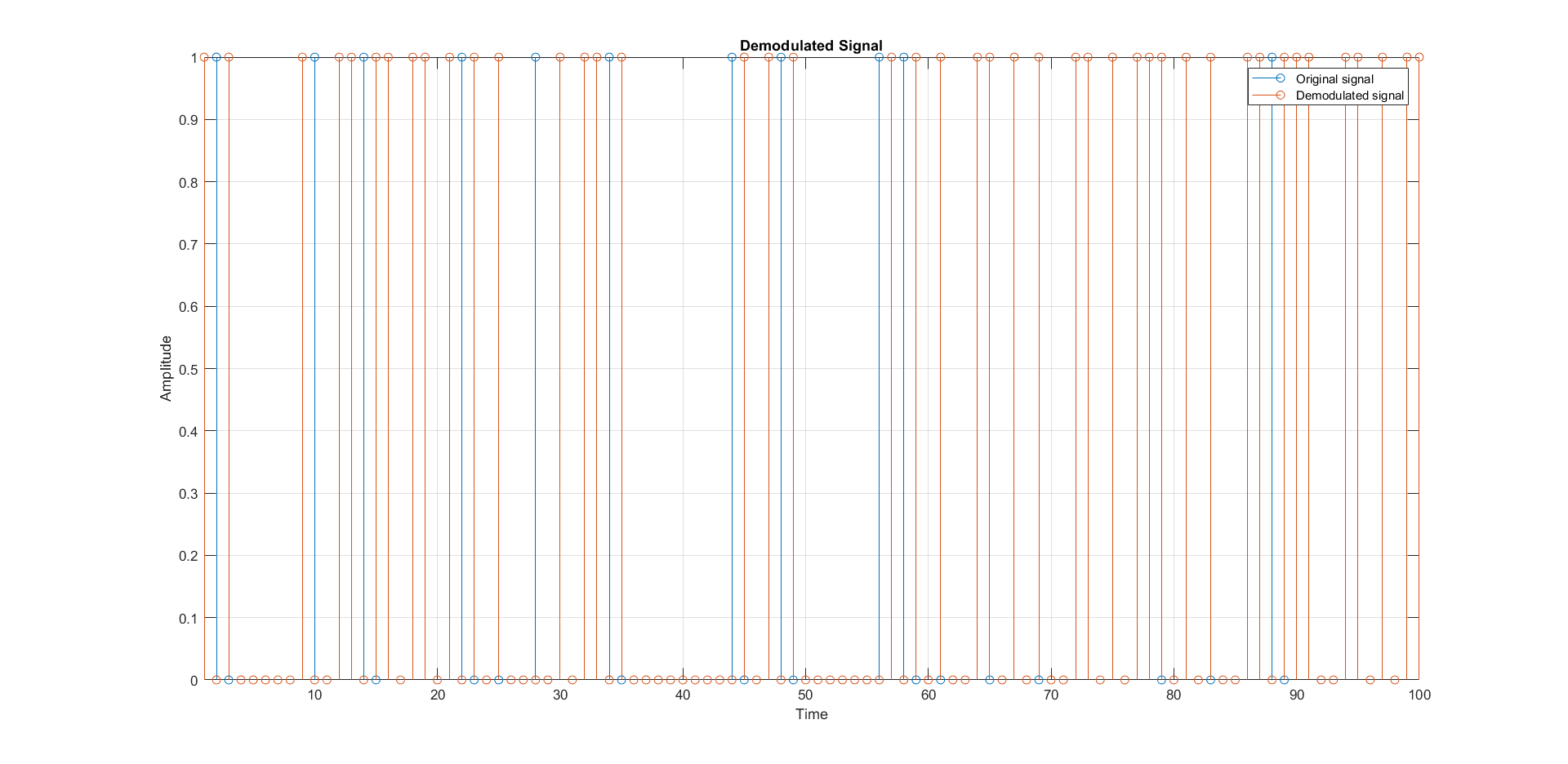


Figure . Sygnał po zdemodulowaniu dla 4QAM.

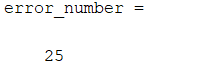


Figure . Ilość odstępstw od oryginalnego sygnału dla 4QAM.

Modulacja nie jest wrażliwa na małe zmiany w fazie, natomiast jej wynik zaburza duże przesunięcie fazowe (od π/2 do 2π).

***Wnioski***

Zadanie 1

Ortogonalne funkcje sinus oraz cosinus są wykorzystywane do modulacji QAM, aby było możliwe przemnożenie przez nie sygnału. Ważne jest, aby zwrócić uwagę na takie same przesunięcie fazowe tych funkcji, gdyż to determinuje ich ortogonalność.

Zadanie 2

Modulacja QAM analogowo jest bardzo wrażliwa na zmiany w fazie. Korekcja tego zjawiska jest możliwa za pomocą układu PLL po otrzymaniu sygnału przez odbiornik.

Zadanie 3

16QAM pozwala na szybsze przesłanie danych ze względu na większą liczbę danych przesyłaną w jednym pakiecie.

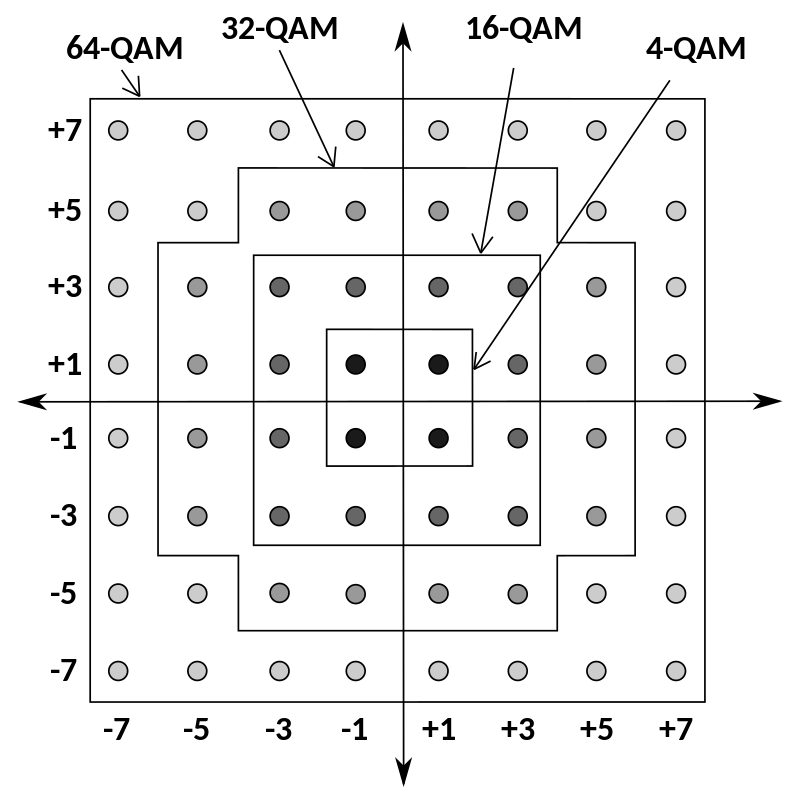


Figure 28. Źródło: https://en.wikipedia.org/wiki/Quadrature\_amplitude\_modulation#/media/File:Rectangular\_constellation\_for\_QAM.svg

Zadanie 4

Modulacja QAM jest wysoce odporna na szumy. Przy SNR = 3 (szumy stanowią około 33% sygnału) dokładność dekodowania wynosi około 86% dla modulacji 16QAM oraz aż 96% dla modulacji 4QAM. Czym mniejsza kwadratura modulacji, tym bardziej jest ona odporna na zniekształcenia (wynika to z rys. 14, na którym widać możliwe zakresy).

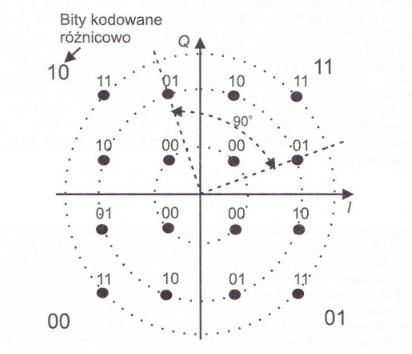


Figure . Źródło: MATLAB i podstawy telekomunikacji, Jacek Izydorczyk i inni, 2017-06-26.

Na podstawie rys. 29 można stwierdzić, że przesunięcie fazowe bardzo wpływa na modulację, ponieważ przesuwany jest zakres osi (o kąt przesunięcia fazowego).