# Dominik Ciesiołkiewicz 44289 – Sprawozdanie Lab 7

# Kodowanie transmisyjne

**Kod:**

#include <iostream>

#include <fstream>

#include <complex>

using namespace std;

double pi = 3.14159265359;

int lengthOfString(string str)

{

return str.length();

}

string S2BS(string in, bool choice) //String To Binary Stream

{

string out = "";

int n = in.length();

string bity = "";

if (choice == 1)//LittleEndian

{

for (int i = 0; i < n; i++)

{

int wartosc = int(in[i]);

bity = "";

while (wartosc > 0)

{

if (wartosc % 2)

{

bity += '1';

}

else

{

bity += '0';

}

wartosc = wartosc / 2;

}

out += bity;

}

reverse(out.begin(), out.end());

//cout << out << endl;

return out;

}

else {//BigEndian

for (int i = 0; i < n; i++)

{

int wartosc = int(in[i]);

bity = "";

while (wartosc > 0)

{

if (wartosc % 2)

{

bity += '1';

}

else

{

bity += '0';

}

wartosc = wartosc / 2;

}

reverse(bity.begin(), bity.end());

out += bity;

}

//cout << out << endl;

return out;

}

}

int\* Mgenerator(string tab, int size, double Tb, double fs)

{

ofstream saveM("M.txt");

int probki = fs \* Tb;

int\* m = new int[size \* probki \* 8];

int index = 0;

for (int i = 0; i < size; i++)

{

if (tab[i] == '1')

{

for (int j = 0; j < 8 \* probki; j++)

{

m[index] = 1;

saveM << m[index] << endl;

index++;

}

}

else

{

for (int j = 0; j < 8 \* probki; j++)

{

m[index] = 0;

saveM << m[index] << endl;

index++;

}

}

}

saveM.close();

return m;

}

int\* clock(double f, int size, double Tb, double fs)

{

ofstream saveClock("Clock.txt");

int probki = fs \* Tb;

int\* clock = new int[size \* probki \* 8];

double phase = 0;

for (int i = 0; i < size \* probki \* 8; i++) {

if (phase < 0.5)

{

clock[i] = 1;

}

else

{

clock[i] = 0;

}

phase += f / (probki\*8);

if (phase >= 1)

{

phase -= 1;

}

saveClock << clock[i] << endl;

}

saveClock.close();

return clock;

}

double\* timeSpan(double f, int size, double Tb, double fs)

{

ofstream saveTimeSpan("Time.txt");

int probki = fs \* Tb;

double\* time = new double[size \* probki \* 8];

double timeStamp = 0;

for (int i = 0; i < size \* probki \* 8; i++) {

//timeStamp = double(double(i) / double(fs));

time[i] = timeStamp;

saveTimeSpan << time[i] << endl;

timeStamp += double(1 / (double(probki)\*8));

}

saveTimeSpan.close();

return time;

}

double\* TTLCoder(int size, double Tb, double fs, int\* m, int\* clock)

{

ofstream saveTTL("TTL.txt");

int probki = fs \* Tb;

double\* TTL = new double[size \* probki \* 8];

TTL[0] = m[0];

saveTTL << TTL[0] << endl;

for (int i = 1; i < size \* probki \* 8; i++)

{

if (clock[i] == 1 && clock[i] != clock[i - 1])

{

if (m[i] == 1)

{

TTL[i] = 1;

}

else

{

TTL[i] = 0;

}

}

else

{

TTL[i] = TTL[i - 1];

}

saveTTL << TTL[i] << endl;

}

saveTTL.close();

return TTL;

}

double\* ManchesterCoder(int size, double Tb, double fs, int\* m, int\* clock)

{

ofstream saveManc("Manchester.txt");

int probki = fs \* Tb;

double\* Manc = new double[size \* probki \* 8];

Manc[0] = m[0];

saveManc << Manc[0] << endl;

int prevclock = 0;

for (int i = 1; i < size \* probki \* 8; i++)

{

if (clock[i] == 0 && clock[i] != clock[i - 1])

{

if (m[i] == 1)

Manc[i] = -1;

else

Manc[i] = 1;

}

else if ((clock[i] == 1 && clock[i] != clock[i - 1]) && (m[i - 1] == m[i]))

{

Manc[i] = -Manc[i-1];

}

else

{

Manc[i] = Manc[i - 1];

}

saveManc << Manc[i] << endl;

}

saveManc.close();

return Manc;

}

double\* NRZICoder(int size, double Tb, double fs, int\* m, int\* clock)

{

ofstream saveNRZI("NRZI.txt");

int probki = fs \* Tb;

double\* NRZI = new double[size \* probki \* 8];

NRZI[0] = 1;

saveNRZI << NRZI[0] << endl;

int counter = 0;

double NRZISignal = 1;

cout << size \* probki \* 8 << endl;

for (int i = 1; i < size \* probki \* 8; i++)

{

if (clock[i]==0 && clock[i] != clock[i - 1])

{

if (m[i] == 0 )

{

NRZI[i] = NRZI[i - 1];

}

else

{

NRZI[i] = -NRZI[i - 1];

}

//counter++;

}

else

{

NRZI[i] = NRZI[i-1];

}

saveNRZI << NRZI[i] << endl;

}

saveNRZI.close();

return NRZI;

}

double\* BAMICoder(int size, double Tb, double fs, int\* m)

{

ofstream saveBAMI("BAMI.txt");

int probki = fs \* Tb;

double\* BAMI = new double[size \* probki \* 8];

int counter = 0;

int value = 1;

for (int i = 0; i < size \* probki \* 8; i++) {

if (m[i] == 0)

{

BAMI[i] = 0;

counter = probki\*8;

}

else

{

if (counter == probki \* 8)

{

counter = 0;

value = -value;

}

BAMI[i] = value;

counter++;

}

saveBAMI << BAMI[i] << endl;

}

saveBAMI.close();

return BAMI;

}

int\* TTLDecoder(int size, double Tb, double fs, double\* m, int\* clock)

{

ofstream saveDecTTL("DecTTL.txt");

int probki = fs \* Tb;

int\* decoded = new int[size \* probki \* 8];

decoded[0] = 1;

saveDecTTL << decoded[0] << endl;

for (int i = 1; i < size \* probki \* 8; i++)

{

if (clock[i] == 0 && clock[i] != clock[i - 1])

{

decoded[i] = m[i];

}

else

{

decoded[i] = decoded[i - 1];

}

saveDecTTL << decoded[i] << endl;

}

saveDecTTL.close();

return decoded;

}

int\* ManchesterDecoder(int size, double Tb, double fs, double\* Man, int\* clock)

{

ofstream saveDecManc("DecManchester.txt");

int probki = fs \* Tb;

int\* decoded = new int[size \* probki \* 8];

decoded[0] = 1;

saveDecManc << decoded[0] << endl;

for (int i = 1; i < size \* probki \* 8; i++) {

if (clock[i] == 0 && clock[i] != clock[i - 1]) {

if (Man[i] > 0)

decoded[i] = 0;

else

decoded[i] = 1;

}

else

{

decoded[i] = decoded[i - 1];

}

saveDecManc << decoded[i] << endl;

}

saveDecManc.close();

return decoded;

}

int\* NRZIDecoder(int size, double Tb, double fs, double\* NRZI, int\* clock)

{

ofstream saveDecNRZI("DecNRZI.txt");

int probki = fs \* Tb;

int\* decoded = new int[size \* probki \* 8];

decoded[0] = 1;

int prevclock = 0;

saveDecNRZI << decoded[0] << endl;

for (int i = 1; i < size \* probki \* 8; i++) {

if (clock[i] == 0 && clock[i] != clock[i - 1]) {

if (NRZI[prevclock] != NRZI[i])

decoded[i] = 1;

else

decoded[i] = 0;

}

else

{

decoded[i] = decoded[i - 1];

}

saveDecNRZI << decoded[i] << endl;

prevclock = i;

}

saveDecNRZI.close();

return decoded;

}

int\* BAMIDecoder(int size, double Tb, double fs, double\* BAMI)

{

ofstream saveDecBAMI("DecBAMI.txt");

int probki = fs \* Tb;

int\* decoded = new int[size \* probki \* 8];

int counter = 0;

int value = 1;

for (int i = 0; i < size \* probki \* 8; i++) {

if (BAMI[i] == 0)

{

decoded[i] = 0;

}

else

{

decoded[i] = 1;

}

saveDecBAMI << decoded[i] << endl;

}

saveDecBAMI.close();

return decoded;

}

int main()

{

//Zad 1 i 2:

string str = S2BS("1A", 1);

int n = lengthOfString(str);

cout << n << endl;

int fs = 250;

double Tb = 0.1;//sekundy

double\* time = timeSpan(2, n, Tb, fs);

int\* m = Mgenerator(str, n, Tb, fs);

int\* clockSig = clock(2, n, Tb, fs);

//Zad 3:

double\* BAMI = BAMICoder(n, Tb, fs, m);

double\* NRZI = NRZICoder(n, Tb, fs, m, clockSig);

double\* Manchester = ManchesterCoder(n, Tb, fs, m, clockSig);

double\* TTL = TTLCoder(n, Tb, fs, m, clockSig);

//Zad 4:

BAMIDecoder(n, Tb, fs, BAMI);

NRZIDecoder(n, Tb, fs, NRZI, clockSig);

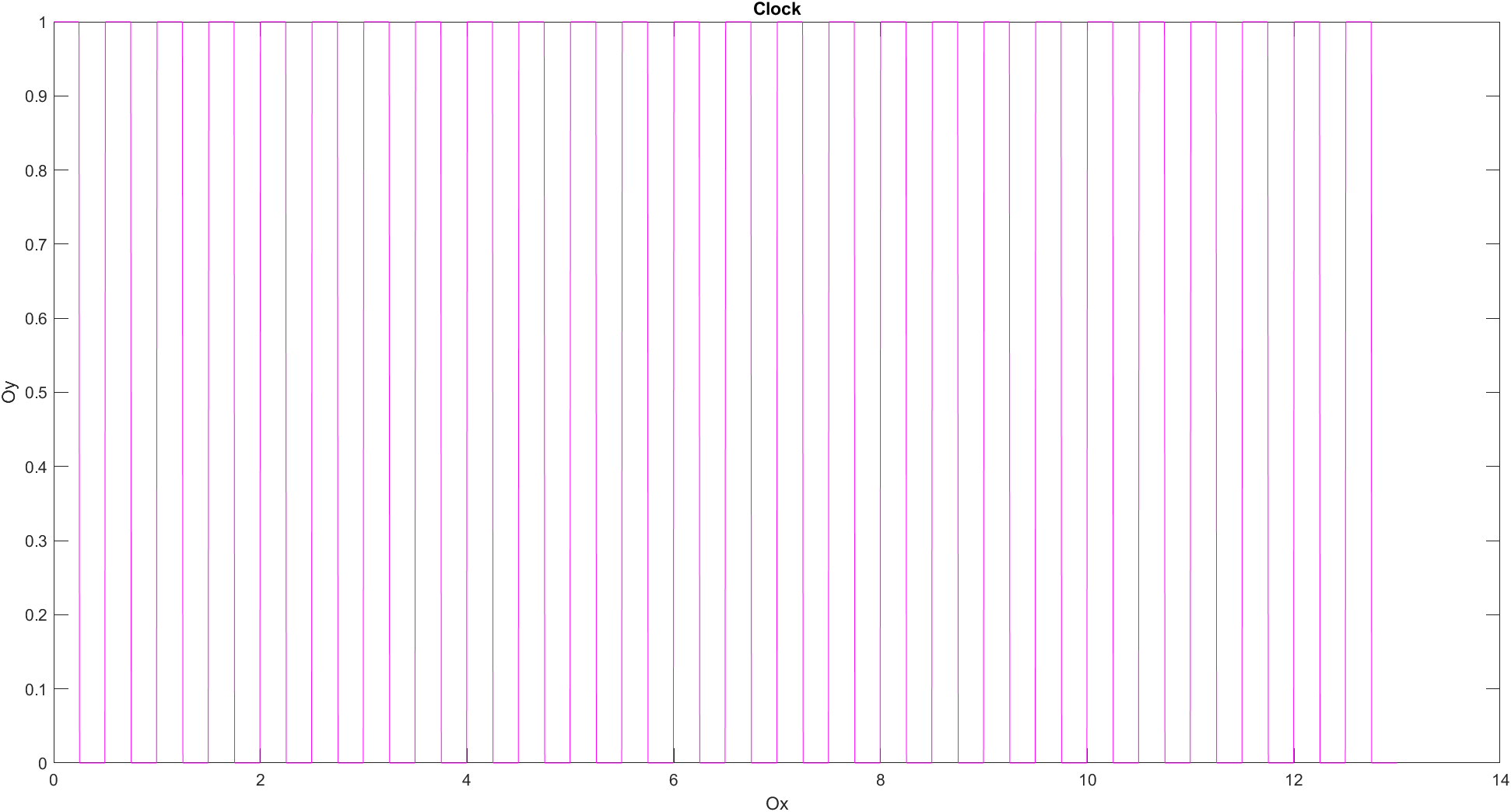
ManchesterDecoder(n, Tb, fs, Manchester, clockSig);

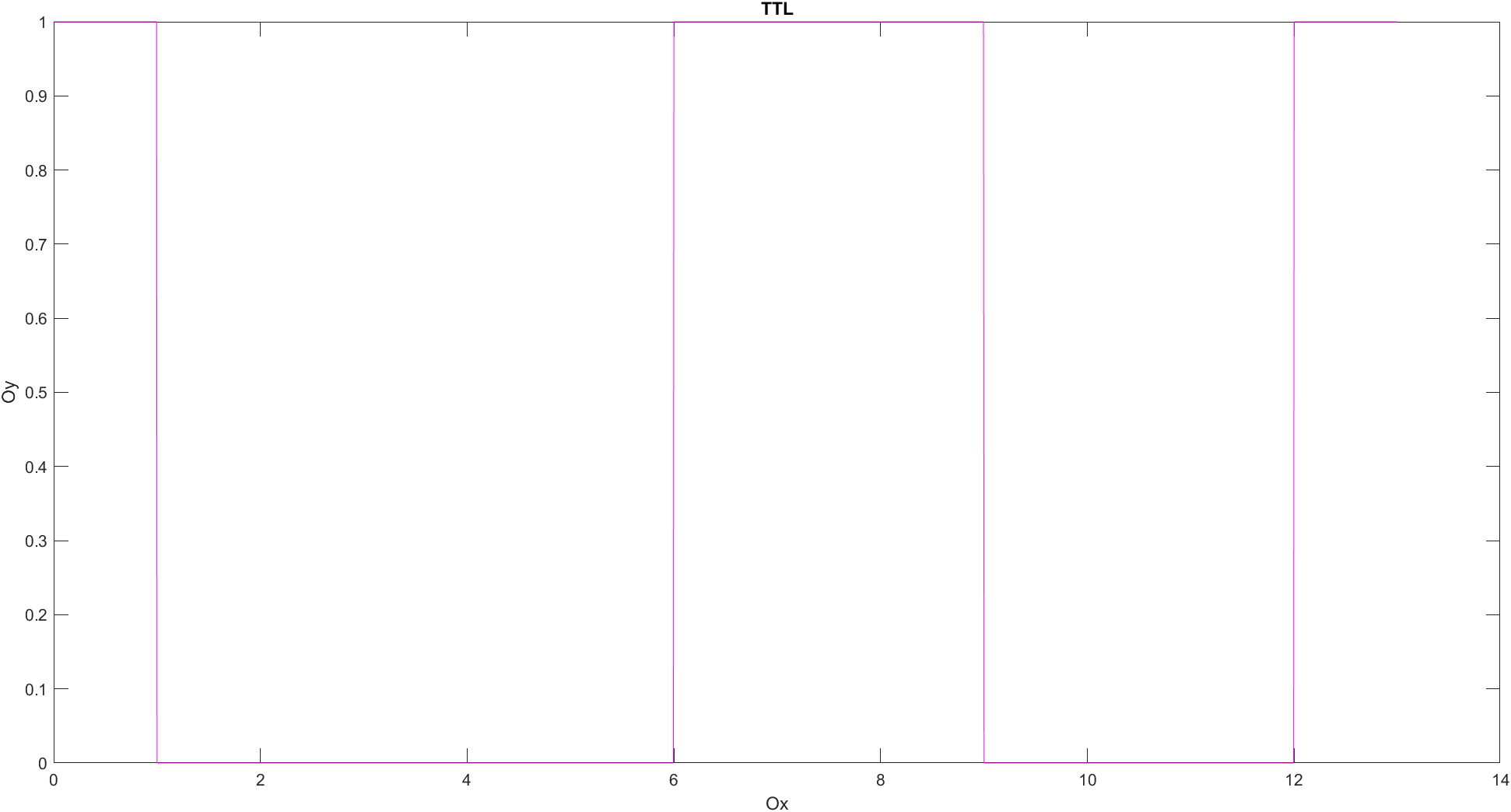
TTLDecoder(n, Tb, fs, TTL, clockSig);

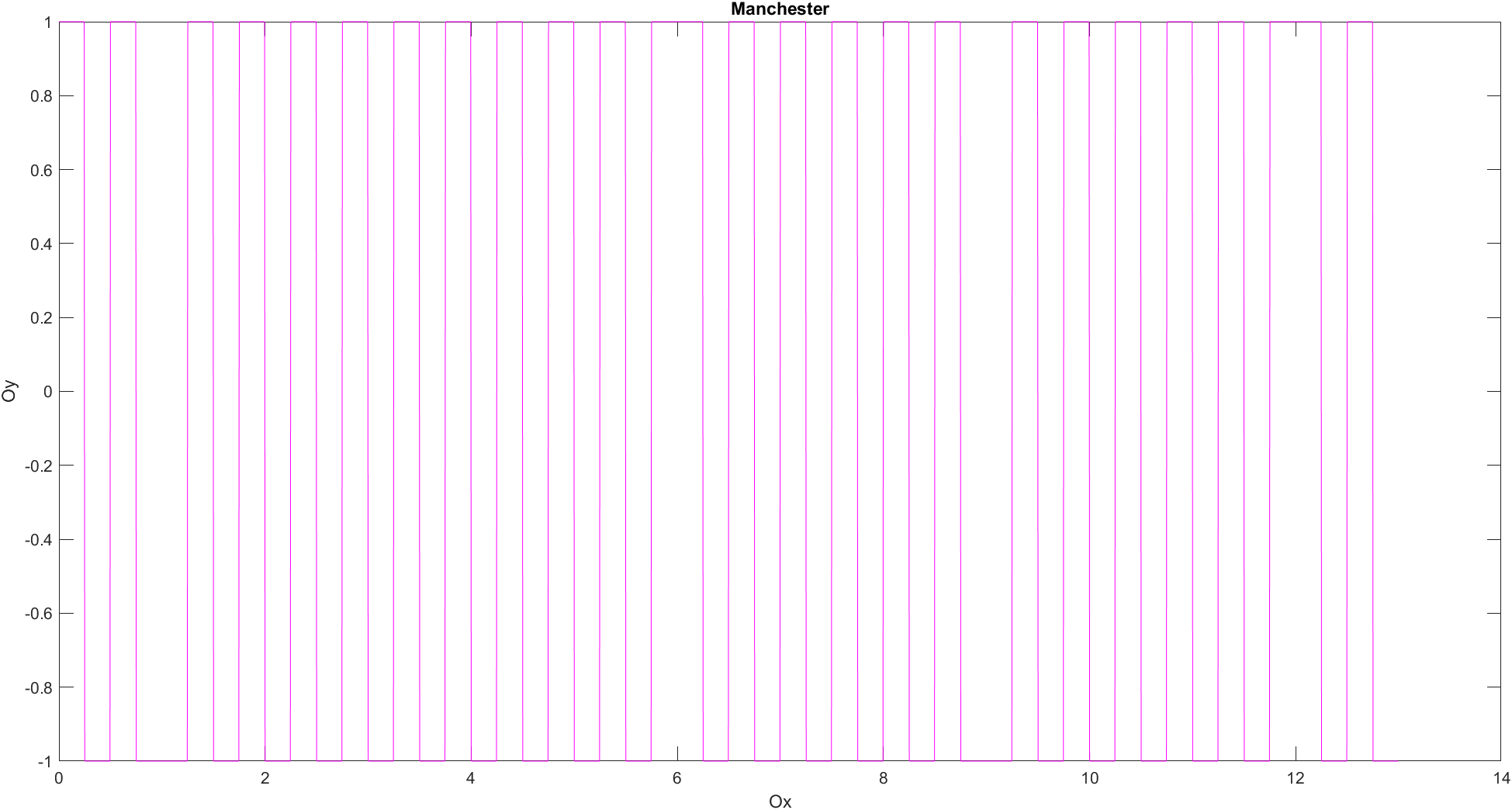
return 1;

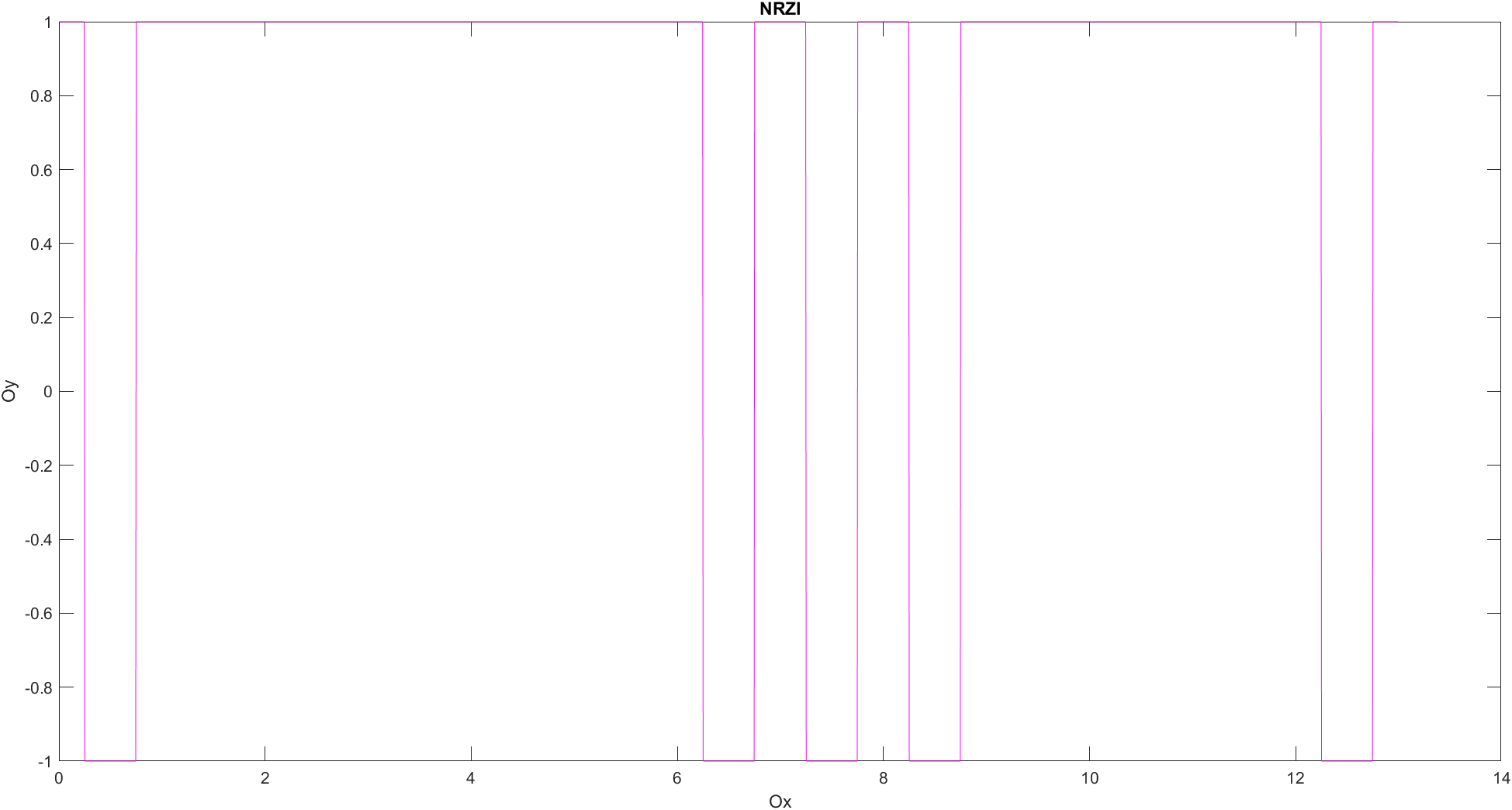
}

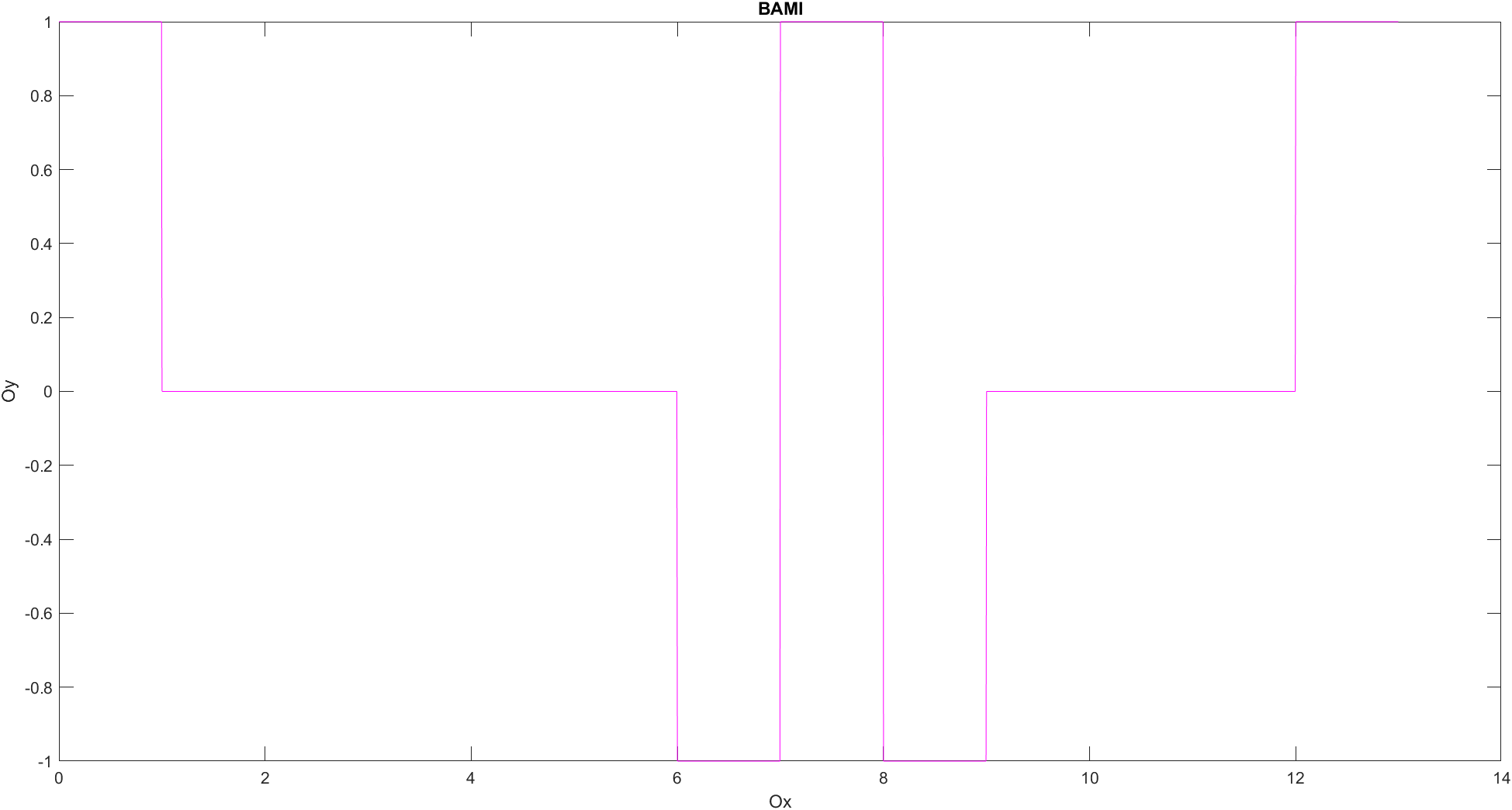
**Wykresy:**

****

****

****

****

****