Dominik Ciesiołkiewicz - Sprawozdanie Lab 6 – Transmisja Danych

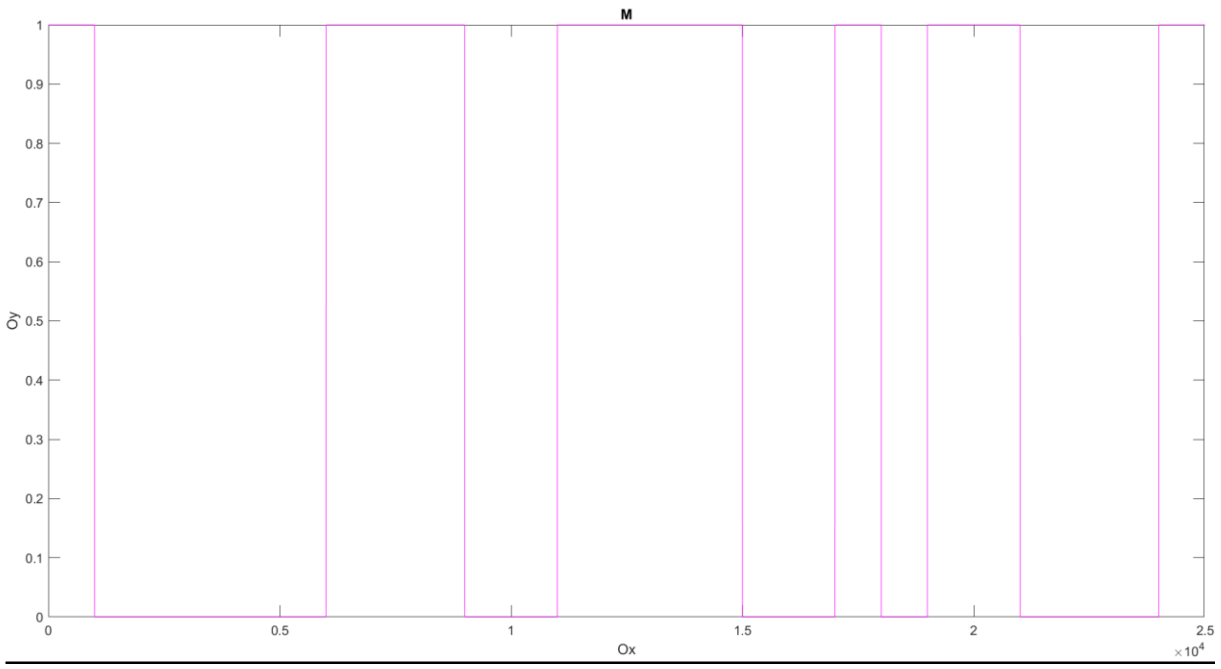
Kod:

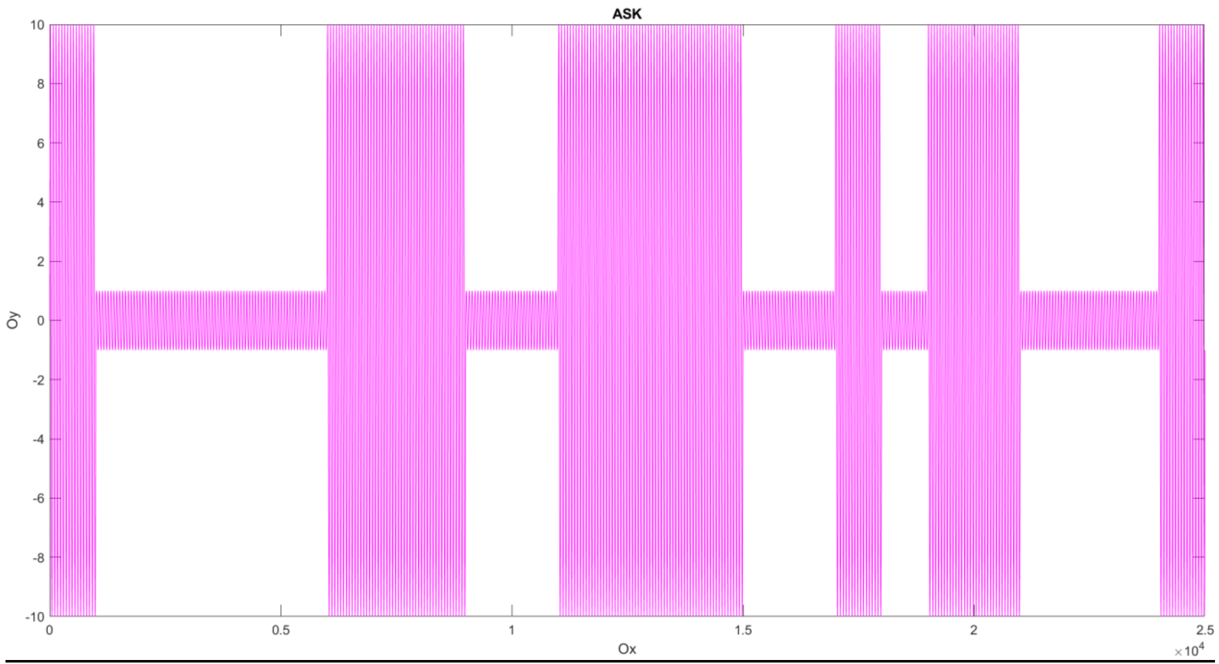
```
#include <iostream>
#include <fstream>
#include <complex>

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;
}

complex<double>* DFT(const double* tab, int N)
{
    complex<double>* tab2 = new complex<double>[N];

    for (int k = 0; k < N; k++)
    {
        tab2[k] = 0;
        complex<double> WN = cos(tab[k]) + 1i * sin(tab[k]);

        for (int n = 0; n < N; n++)
        {
            tab2[k] += tab[n] * pow(WN, -k * n);
        }
        //for (int n = 0; n < N; n++)
        //{
        //    tab2[k] += tab[n] * exp(-2 * pi * 1i * (double)k * (double)n / 
(double)N);
        //}

    }

    return tab2;
}

double ton_prosty(double A1, double F, double t)// czy jest w ogóle potrzebny?
{
    return A1 * sin(2 * pi * F * t);
}

double * ASK(int * m, int n, int A1, int A2, double f, double fs, double phi)
{
    double * zA = new double[n];
    for (int i = 0; i < n; i++)
    {
        if (m[i] == 0)
        {
            zA[i] = A1 * sin(2 * pi * f * i/fs + phi);
        }
        else
        {
            zA[i] = A2 * sin(2 * pi * f * i/fs + phi);
        }
    }
    ofstream saveASK("zad2ASK.txt");
    for (int i = 0; i < n; i++)
    {
        saveASK << zA[i] << endl;
    }
    saveASK.close();
    return zA;
}

double * FSK(int* m, int n, int A, int N, double fs, double Tb, double phi)
{
    double * zF = new double[n];
    double f0 = (N + 1) / Tb;
    double f1 = (N + 2) / Tb;
    for (int i = 0; i < n; i++)
    {
        if (m[i] == 0)
        {
            zF[i] = A * sin(2 * pi * f0 * i/fs + phi);
        }
        else
        {
            zF[i] = A * sin(2 * pi * f1 * i/fs + phi);
        }
    }
    ofstream saveFSK("zad2FSK.txt");
    for (int i = 0; i < n; i++)
    {
        saveFSK << zF[i] << endl;
    }
    saveFSK.close();
    return zF;
}

double * PSK(int* m, int n, int A, double f, double fs, double Tb)
{
    double * zP = new double[n];
    for (int i = 0; i < n; i++)
    {
        if (m[i] == 0)
        {
            zP[i] = A * sin(2 * pi * f * i/fs + 0);
        }
        else
        {
            zP[i] = A * sin(2 * pi * f * i/fs + pi);
        }
    }
    ofstream savePSK("zad2PSK.txt");
    for (int i = 0; i < n; i++)
    {
        savePSK << zP[i] << endl;
    }
    savePSK.close();
    return zP;
}

void printOut(double* tab, int n, bool sw)
{
    if(sw==0)
    {
        for (int i = 0; i < n; i++)
        {
            cout << tab[i] << endl;
        }
    }
    else
    {
        ofstream saveASK("zad1ASK.txt");
        for (int i = 0; i < n; i++)
        {
            saveASK << tab[i] << endl;
        }
        saveASK.close();
    }
}

void widmoAmplitudowe(complex<double>* DFTvalues, int size)
{
    double* M = new double[22050];
    double* Mprim = new double[22050];

    ofstream saveM("zad3M.txt");
    ofstream saveMprim("zad3Mprim.txt");

    for (int i = 0; i < size; i++)
    {
        M[i] = sqrt(pow(real(DFTvalues[i]), 2) + pow(imag(DFTvalues[i]), 2));
        saveM << M[i] << endl;
        Mprim[i] = 10 * log10(M[i]);
        saveMprim << Mprim[i] << endl;
    }
```

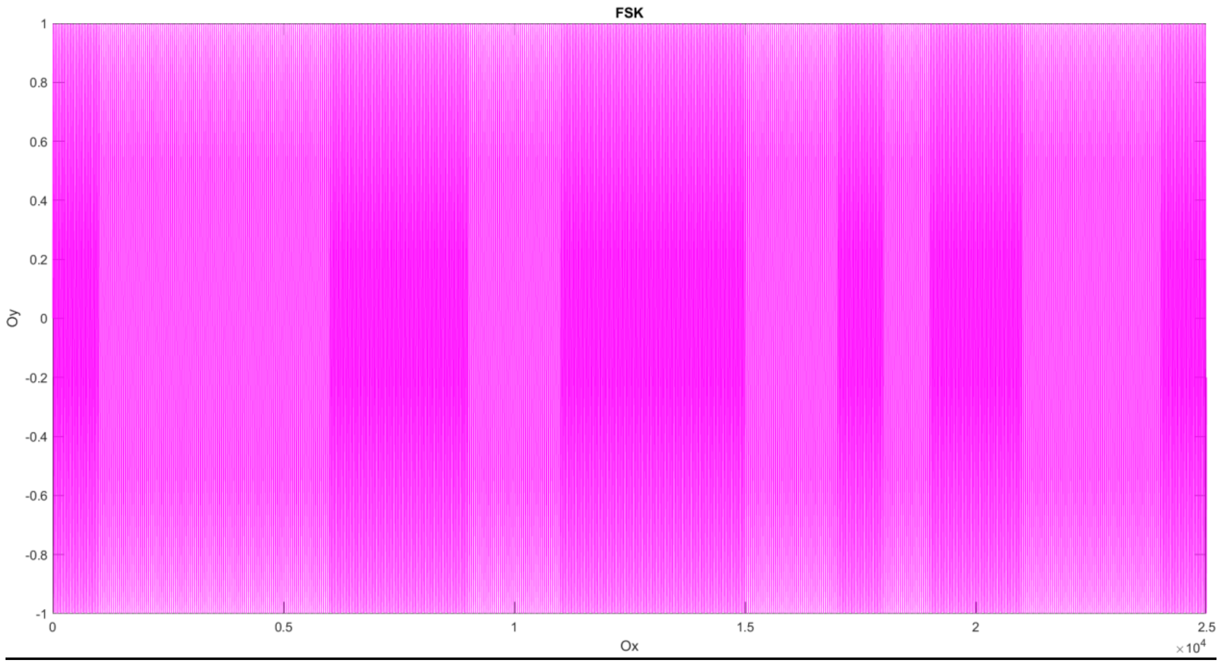
```cpp
    saveM.close();
    saveMprim.close();
}

void szerokoscPasma(double* pasmo, int n) {
    double max = pasmo[0];
    double min = pasmo[0];

    for (int i = 1; i < n; i++)
    {
        if (pasmo[i] < min)
        {
            min = pasmo[i];
        }
        if (pasmo[i] > max)
        {
            max = pasmo[i];
        }
    }

    double szerokosc = max - min;
    cout << szerokosc << endl;
}


int main()
{
    string str=S2BS("123A", 1);
    //100000111001111001011001 - Little Endian
    //S2BS("123A", 0);
    //110001110010110011100001 - Big Endian

    double phi = 0;
    double Tb = 0.1;//sekundy
    int A1 = 1;
    int A = A1;
    int A2 = 10;
    int N = 2;
    int fs = 250;
    double f = N*pow(Tb,-1);
    int n = lengthOfString(str);
    int probki = fs * Tb;
    int msize = n * probki * 8;
    int* m = Mgenerator(str, n, Tb, fs);

    /*//Zad2:
    double * asktab = ASK(m, msize, A1, A2, f, fs, phi);
    double * fsktab = FSK(m, msize, A, N, fs, Tb, phi);
    double * psktab = PSK(m, msize, A, f, fs, Tb);
    */

    //Zad 3 i 4:
    double* asktab = ASK(m, 10 * probki * 8, A1, A2, f, fs, phi);
    //complex<double>* askWidmo = DFT(asktab, 10 * probki * 8);
    //widmoAmplitudowe(askWidmo, 10 * probki * 8);

    double* fsktab = FSK(m, 10 * probki * 8, A, N, fs, Tb, phi);
    //complex<double>* fskWidmo = DFT(fsktab, 10 * probki * 8);
    //widmoAmplitudowe(fskWidmo, 10 * probki * 8);

    double* psktab = PSK(m, 10 * probki * 8, A, f, fs, Tb);
    //complex<double>* pskWidmo = DFT(psktab, 10 * probki * 8);
    //widmoAmplitudowe(pskWidmo, 10 * probki * 8);

    //Zad 5:
    szerokoscPasma(asktab, 10 * probki * 8);
    szerokoscPasma(fsktab, 10 * probki * 8);
    szerokoscPasma(psktab, 10 * probki * 8);

    //Szerokość pasma ASK: 19.9605
    //Szerokość pasma PSK: 1.99605
    //Szerokość pasma FSK: 1.99605

        return 1;
}
```

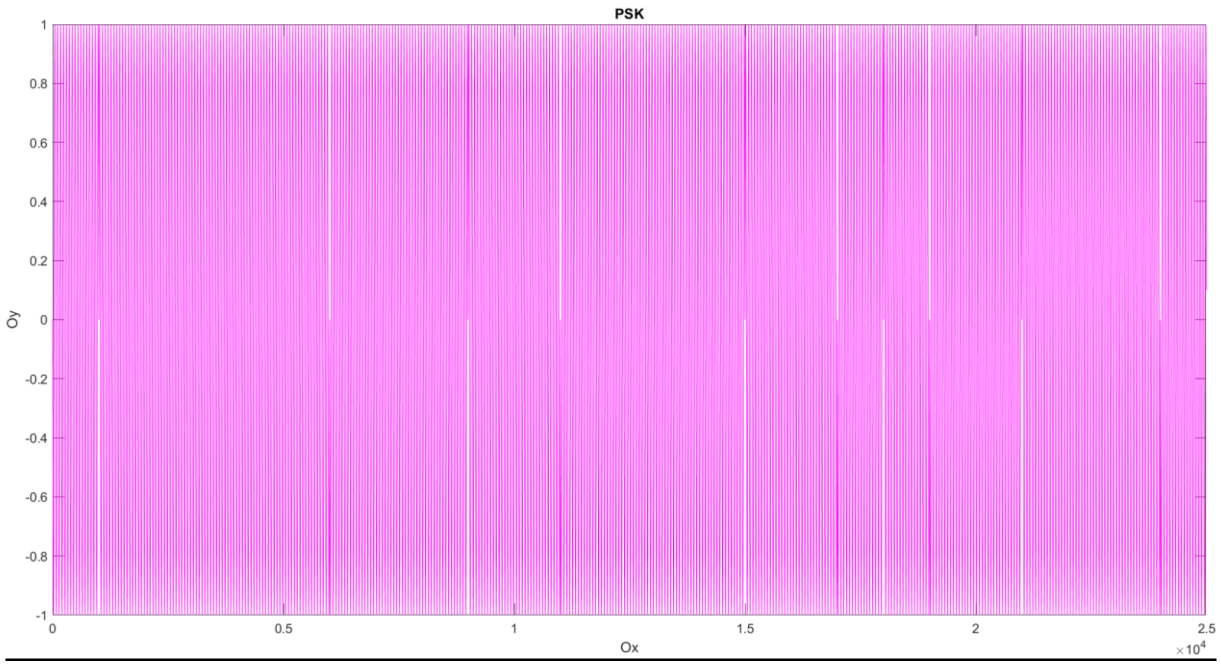
Wykresy:

Zad2:

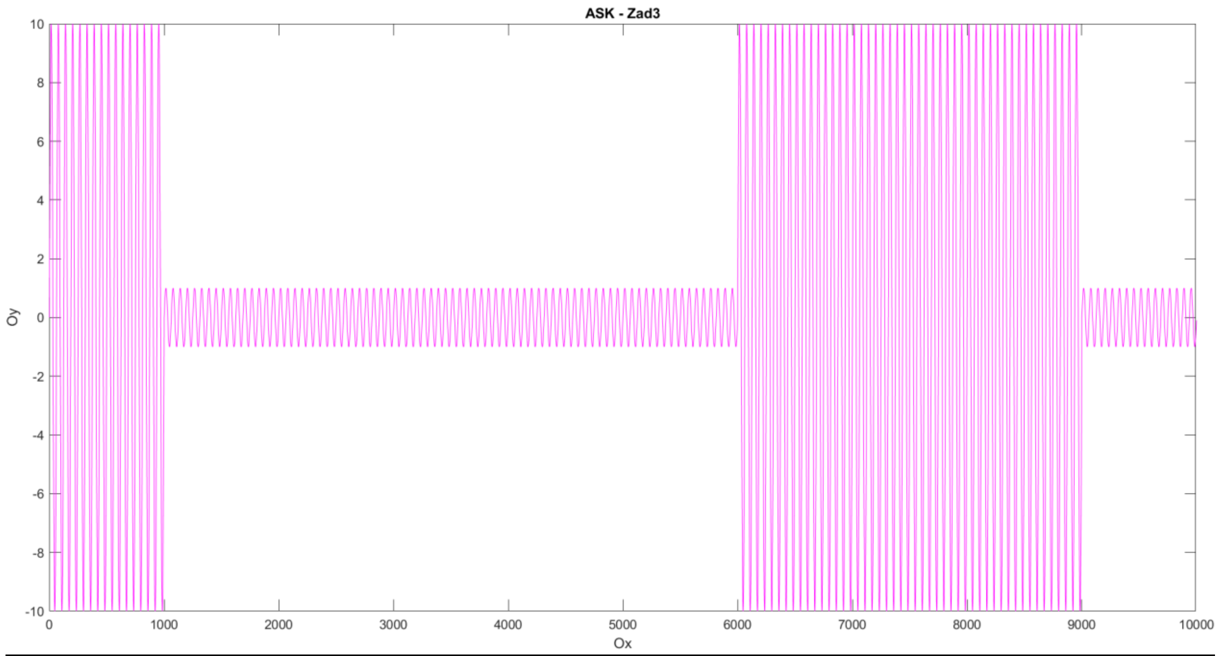


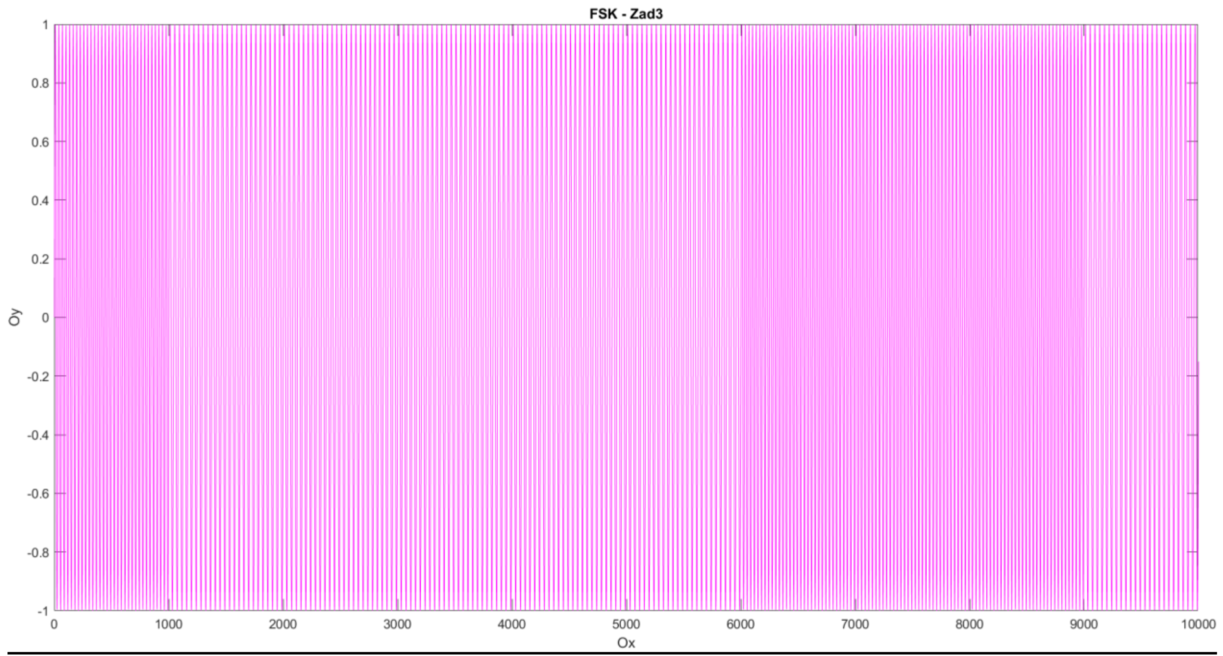

ASK
Oy
Ox
×10^4
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
0.5
1
1.5
2
2.5

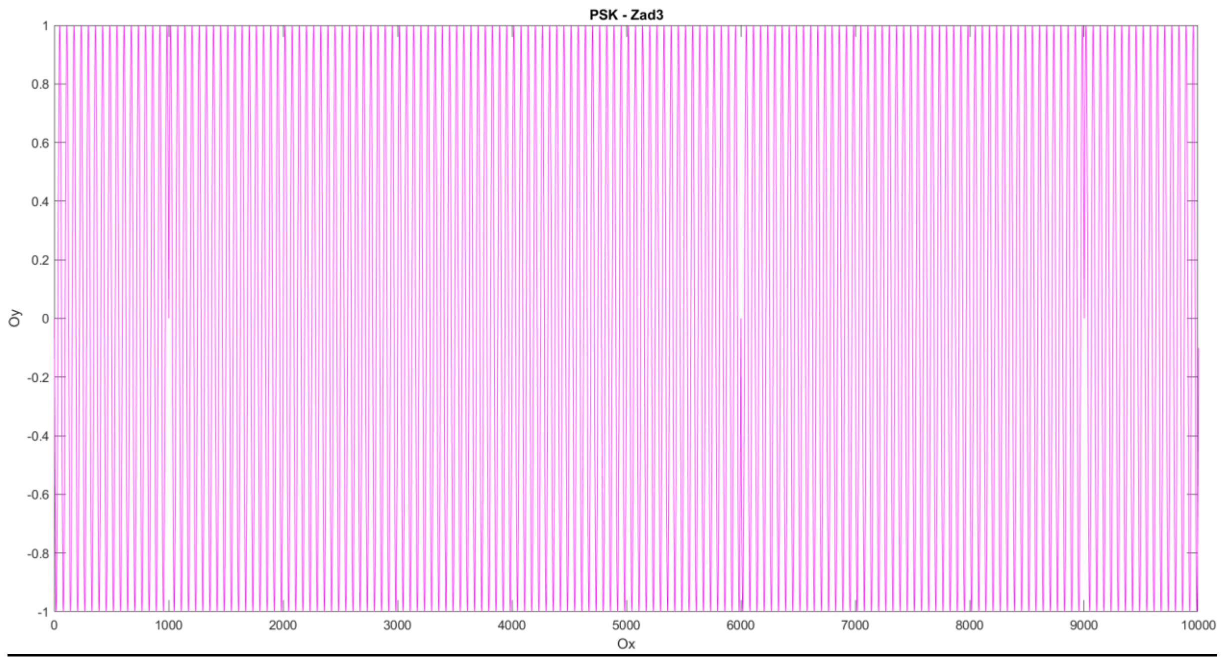

FSK
Oy
Ox
×10^4
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5


PSK
Oy
Ox
×10^4
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5

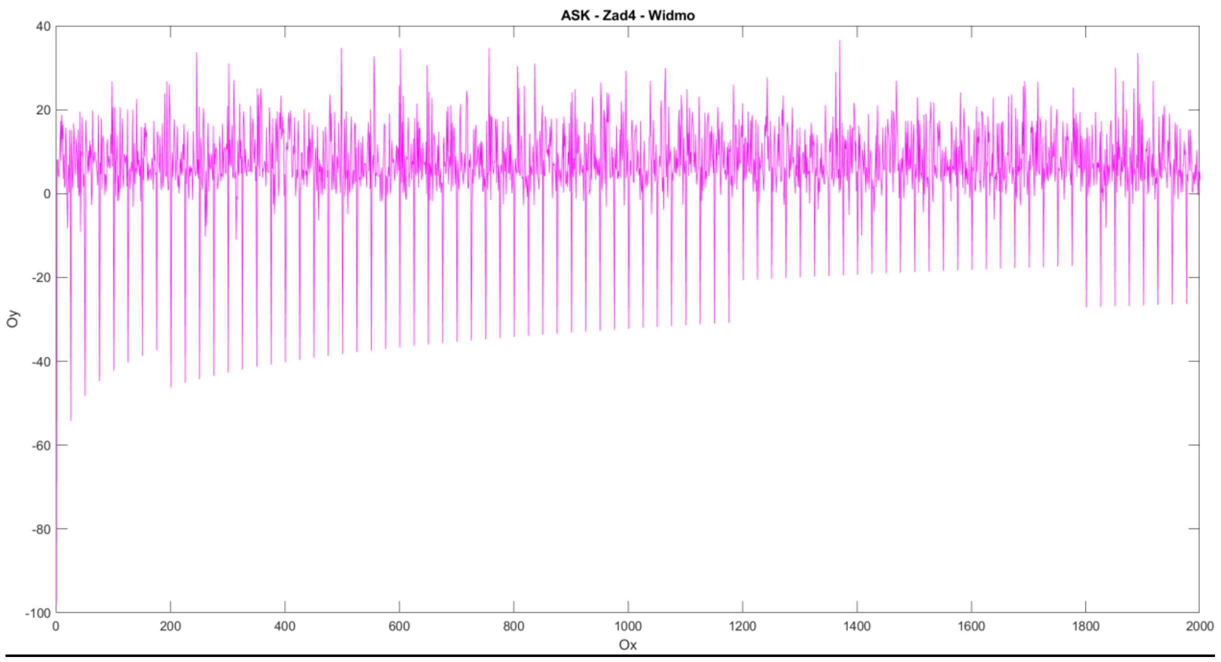
Zad3:


ASK - Zad3
Oy
Ox
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

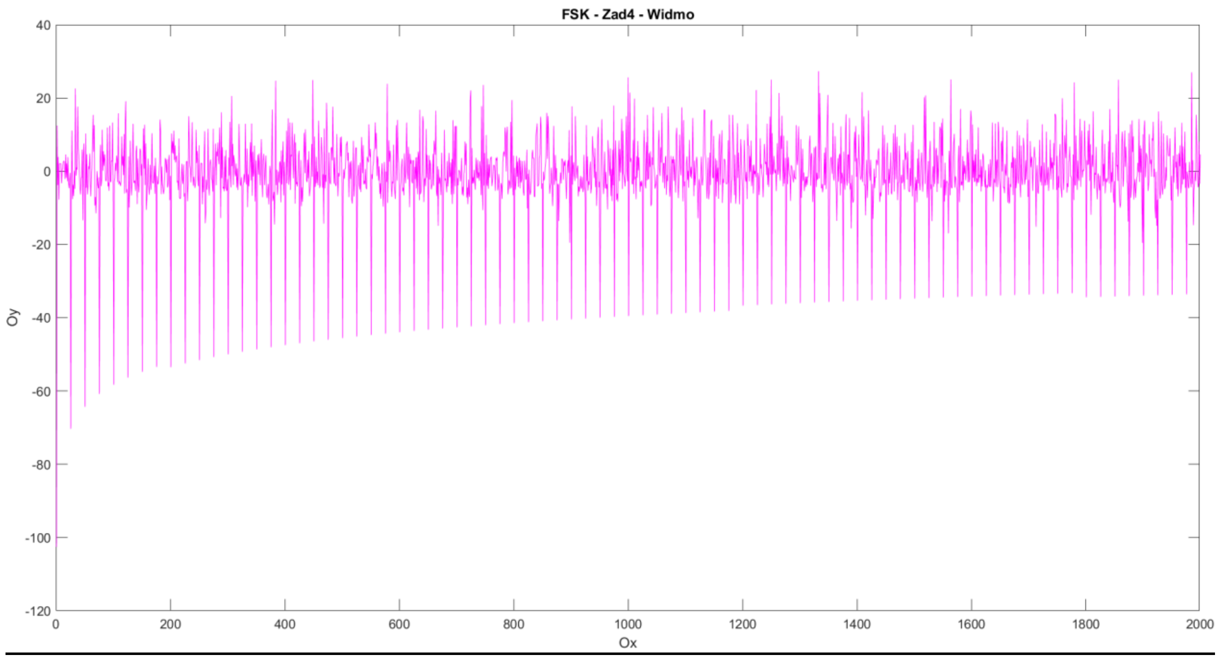

FSK - Zad3
Oy
Ox
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


PSK - Zad3
Oy
Ox
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

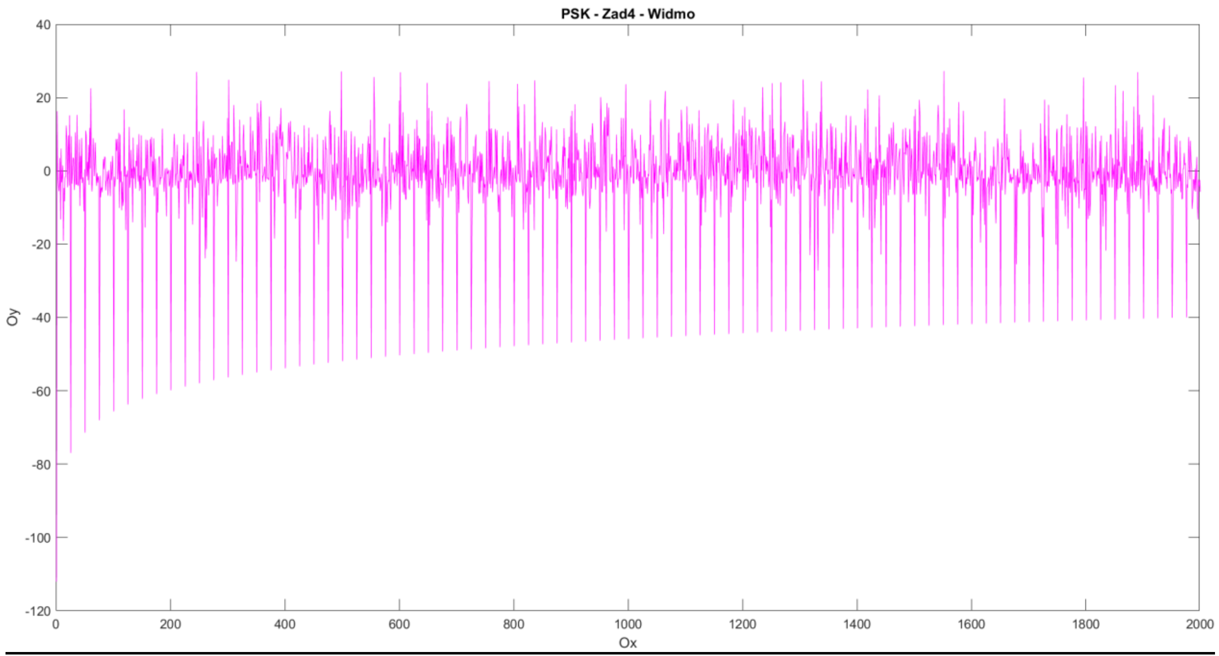
Zad4:



ASK - Zad4 - Widmo
Oy
Ox
40
20
0
-20
-40
-60
-80
-100
0
200
400
600
800
1000
1200
1400
1600
1800
2000



FSK - Zad4 - Widmo
Oy
Ox
40
20
0
-20
-40
-60
-80
-100
-120
0
200
400
600
800
1000
1200
1400
1600
1800
2000


PSK - Zad4 - Widmo
Oy
Ox
40
20
0
-20
-40
-60
-80
-100
-120
0
200
400
600
800
1000
1200
1400
1600
1800
2000