# Dominik Ciesiołkiewicz 44289 – Sprawozdanie Lab 3 & 4

Zadanie 1, 2 i 4 uważam za zakończone. W dalszej części sprawozdania załączam kod i wykres do zadania 2. Nie jestem jednak w stanie wykonać zadania nr 3. Pomimo implementacji (zdawać by się mogło poprawnej) program wykonuje się za długo. Po 25 minutach stwierdziłem, iż gdzieś musi być błąd. Jakiś problem musi być w linijkach 201-205. To na tych częściach program za długo pracuje. Załączam kod łączny który powinien generować wyniki na wszystkie zadane pytania. Jeżeli zachce Pan, bym wydzielił każde zadanie do osobnego pliku poproszę o taką informację. Niżej znajduje się wydzielony kod zadania nr 1.

**Kod:**

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

#define _USE_MATH_DEFINES

using namespace std;

double pi = 3.14159265359;

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);
        }
    }

    return tab2;
}

complex<double>* IDFT(complex<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] * exp(2 * pi * 1i * (double)k * (double)n / (double)N);
        }
        tab2[k] = 1. / N * tab2[k];
        cout << tab2[k] << endl;
```

```
    }

    return tab2;
}

double ton_prosty(double a, double F, double phi, double t)
{
    double s = a * sin(2 * pi * F * t + phi);
    return s;
}

int kwantyzacja(double wartosc, int q)
{
    double quantum = wartosc * pow(2, q - 1);

    if (quantum > 0)
        quantum = ceil(quantum);
    else
        quantum = floor(quantum);
    //cout << quantum << endl;
    return quantum;
}

int main()
{
    //definicja zmiennych
    double ilosc = 982;
    double a = 1;//Volt
    double A = 9;//z numeru albumu
    double F = 8;//Hz
    double C = 2;
    double phi = C * pi;//rad
    double fs = 250;
    double Ts = 1 / fs;

    double q = 16;

    ofstream saveOX("zad2OX.txt");
    ofstream savesigquant("zad2sigquant.txt");
    ofstream saveOXk("zad2OXk.txt");

    //obliczanie wartosci sygnalow

    double * sig = new double[ilosc];
    double * sigquant = new double[ilosc];
    int iterator = 0;

    //zad2
    for (int i = 0; i < ilosc; i++)
    {
        double freq = i / fs;
        double freqk = i * fs / ilosc;
        double sig = ton_prosty(a, F, phi, freq);

        saveOX << freq << endl;
        saveOXk << freqk << endl;

        sigquant[i] = kwantyzacja(sig, q);
        savesigquant << sigquant[i] << endl;
    }

    saveOXk.close();
```

```
saveOX.close();
savesigquant.close();

//zad3
//Funkcja p

ofstream savep("data_p.txt");

int N = 98;
double ptab[22050];
for (int i = 0; i < 22050; i++)
{
    ptab[i] = 0;
}

int count = 0;

for (double i = 0; i <= 1; i = i + 1. / 22050)
{
    float p = 0;
    for (int n = 1; n < N; n++)
    {
        ptab[count] += (cos(12 * i * n * n) + cos(16 * i * n)) / (n * n);
    }
    savep << ptab[count] << endl;
    count++;
}

savep.close();

//Funkcja v

ofstream savev("data_v.txt");

double vtab[22050];
count = 0;

for (double i = 0; i <= 1; i = i + 1. / 22050)
{
    float v;
    if (i < 0.22)
        vtab[count] = (1 - 7 * i) * sin((2 * pi * i * 10) / (i + 0.04));
    else if (i < 0.7)
        vtab[count] = 0.63 * i * sin(125 * i);
    else
        vtab[count] = pow(i, -0.662) + 0.77 * sin(8 * i);

    savev << vtab[count] << endl;
    count++;
}

savev.close();

//Funkcje y, z, u

ofstream savey("data_y.txt");
ofstream savez("data_z.txt");
ofstream saveu("data_u.txt");
ofstream saveOXzad3("data_OX_zad3.txt");
ofstream saveOXkzad3("data_OX_zad3k.txt");

double y[22050];
```

```
    double z[22050];
    double u[22050];

    count = 0;
    for (double i = 0; i <= 1; i = i + 1. / 22050)
    {
        //cout << i << endl;
        double x = 9 * i * i + 8 * i + 2;
        y[count] = 2 * x * x + 12 * cos(i);
        savey << y[count] << endl;
        z[count] = sin(2 * pi * 7 * i) * x - 0.2 * log10(abs(y[count]) + pi);
        savez << z[count] << endl;
        u[count] = sqrt(abs(y[count] * y[count] * z[count])) - 1.8 * sin(0.4 * i *
z[count] * x);
        saveu << u[count] << endl;
        saveOXzad3 << i << endl;
        double freqk3 = i * fs / ilosc;
        saveOXkzad3 << freqk3 << endl;
    }

    savey.close();
    savez.close();
    saveu.close();
    saveOXzad3.close();
    saveOXkzad3.close();

    //DFT

    complex<double>* DFTvalues2 = DFT(sig, ilosc);
    //dla testow:
    IDFT(DFTvalues2, ilosc);

    //kłopotliwe linie:
    complex<double>* DFTvalues3y = DFT(y, 22050);
    complex<double>* DFTvalues3z = DFT(z, 22050);
    complex<double>* DFTvalues3u = DFT(u, 22050);
    complex<double>* DFTvalues3v = DFT(vtab, 22050);
    complex<double>* DFTvalues3p = DFT(ptab, 22050);

    //M i Mprim

    ofstream saveRealDFT("zad2realDFT.txt");
    ofstream saveImagDFT("zad2imagDFT.txt");
    ofstream saveM("zad2M.txt");
    ofstream saveMprim("zad2Mprim.txt");

    double* M = new double[ilosc];
    double* Mprim = new double[ilosc];

    //zad2
    for (int i = 0; i < ilosc; i++)
    {
        //cout << DFTvalues[i] << endl;
        saveRealDFT << real(DFTvalues2[i]) << endl;
        saveImagDFT << imag(DFTvalues2[i]) << endl;
        M[i] = sqrt(pow(real( DFTvalues2[i] ), 2) + pow(imag( DFTvalues2[i] ), 2));
        saveM << M[i] << endl;
        Mprim[i] = 10 * log10(M[i]);
        saveMprim << Mprim[i] << endl;
    }

    //zamkniecie strumieni
```

```
saveM.close();
saveMprim.close();
saveRealDFT.close();
saveImagDFT.close();

//zad3

double* M3 = new double[22050];
double* M3prim = new double[22050];

ofstream save3yM("zad3yM.txt");
ofstream save3yMprim("zad3yMprim.txt");

for (int i = 0; i < 22050; i++)
{
    M3[i] = sqrt(pow(real(DFTvalues3y[i]), 2) + pow(imag(DFTvalues3y[i]), 2));
    save3yM << M3[i] << endl;
    M3prim[i] = 10 * log10(M[i]);
    save3yMprim << M3prim[i] << endl;
}

save3yM.close();
save3yMprim.close();

ofstream save3uM("zad3uM.txt");
ofstream save3uMprim("zad3uMprim.txt");

for (int i = 0; i < 22050; i++)
{
    M3[i] = sqrt(pow(real(DFTvalues3u[i]), 2) + pow(imag(DFTvalues3u[i]), 2));
    save3uM << M3[i] << endl;
    M3prim[i] = 10 * log10(M[i]);
    save3uMprim << M3prim[i] << endl;
}

save3uM.close();
save3uMprim.close();

ofstream save3vM("zad3vM.txt");
ofstream save3vMprim("zad3vMprim.txt");

for (int i = 0; i < 22050; i++)
{
    M3[i] = sqrt(pow(real(DFTvalues3v[i]), 2) + pow(imag(DFTvalues3v[i]), 2));
    save3vM << M3[i] << endl;
    M3prim[i] = 10 * log10(M[i]);
    save3vMprim << M3prim[i] << endl;
}

save3vM.close();
save3vMprim.close();

ofstream save3pM("zad3pM.txt");
ofstream save3pMprim("zad3pMprim.txt");

for (int i = 0; i < 22050; i++)
{
    M3[i] = sqrt(pow(real(DFTvalues3p[i]), 2) + pow(imag(DFTvalues3p[i]), 2));
    save3pM << M3[i] << endl;
    M3prim[i] = 10 * log10(M[i]);
    save3pMprim << M3prim[i] << endl;
```

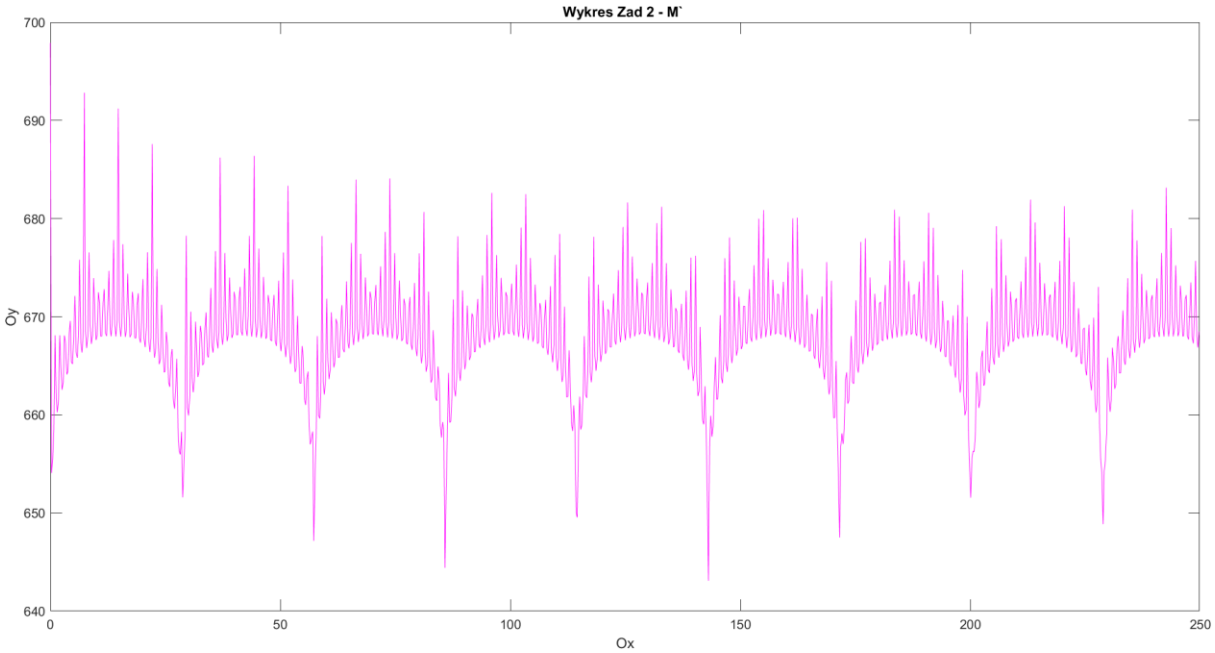
```
    }

    save3pM.close();
    save3pMprim.close();

    ofstream save3zM("zad3zM.txt");
    ofstream save3zMprim("zad3zMprim.txt");

    for (int i = 0; i < 22050; i++)
    {
        M3[i] = sqrt(pow(real(DFTvalues3z[i]), 2) + pow(imag(DFTvalues3z[i]), 2));
        save3zM << M3[i] << endl;
        M3prim[i] = 10 * log10(M[i]);
        save3zMprim << M3prim[i] << endl;
    }

    save3zM.close();
    save3zMprim.close();

    return 0;
}
```

**Kod zawierający tylko zadanie nr 1:**

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

#define _USE_MATH_DEFINES

using namespace std;

double pi = 3.14159265359;

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

    for (int k = 0; k < N; k++)
    {
        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);
        }
    }

    return tab2;
}


int main()
{
    double tab[5] = { 0.5,0.3,-0.2,0.7,0.99 };
    double ilosc = 5;
    double A = 1;//Volt
    double f = 8;//Hz
    double C = 2;
    double phi = C * pi;//rad

    complex<double>* DFTvalues = DFT(tab, ilosc);
```

```
    for (int i = 0; i < 5; i++)
    {
        cout << DFTvalues[i] << endl;
    }


    return 0;
}
```

**Wykres zadania 2:**




Wszystkie pliki z kodami w formacie „.cpp” są dostępne na Githubie.