Transmisja Danych – Lab 05

Krystian Bartosik 213A, nr 44266

Kod źródłowy:

```
// Krystian Bartosik
// bk44266@zut.edu.pl
//   FEDCBA
#define _USE_MATH_DEFINES
#include <iostream>
#include <string>
#include <fstream>
#include "math.h"
#include <complex>
#include <cstddef>
#include <bitset>

using namespace std;

string S2BS(const char * s, string Endian)
{
    string result = "";

    if (Endian == "BigEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = result + bitset<8>(s[j]).to_string();
        }
    }
    // "test" = 01110100011001010111001101110100

    if (Endian == "LittleEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = bitset<8>(s[j]).to_string() + result;
        }
    }
    // "test" = 01110100011100110110010101110100

    cout << result;
    return result;
}

double zA(double A1, double A2, double f, double Fi, char T, double t)
{
    double tt = (double)T - '0'; // Konwersja na liczbę
    if (tt == 0)
    {
        return A1 * sin(2.0 * M_PI * f * t + Fi);
    }

    if (tt == 1)
    {
        return A2 * sin(2.0 * M_PI * f * t + Fi);
    }
}
```

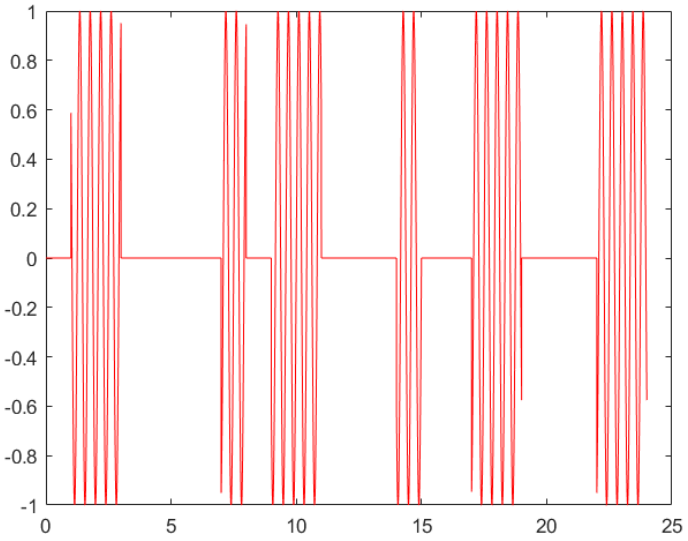
```
double zF(double A, long double f0, double f1, double Fi, char T, double t)
{
    double tt = (double)T - '0'; // Konwersja na liczbę
    if (tt == 0)
    {
        return A * sin(2.0 * M_PI * f0 * t + Fi);
    }

    if (tt == 1)
    {
        return A * sin(2.0 * M_PI * f1 * t + Fi);
    }
}

double zP(double A, double f, double Fi1, double Fi2, char T, double t)
{
    double tt = (double)T - '0'; // Konwersja na liczbę
    if (tt == 0)
    {
        return A * sin(2.0 * M_PI * f * t + Fi1);
    }

    if (tt == 1)
    {
        return A * sin(2.0 * M_PI * f * t + Fi2);
    }
}

complex<double>* DFT(double* Tab, int n)
{
    complex<double>* c = new complex<double>[n];
    complex<double> i = 0.0 + 1.0i;
    for (int k = 0; k < n; k++)
    {
        c[k] = 0.0 + 0.0i;

        for (int j = 0; j < n; j++)
        {
            c[k] = c[k] + (Tab[j] * exp(-2 * M_PI * i * (double)k * (double)j / double(n)));
        }
    }
    return c;
}

int main()
{
    fstream File;
    File.open("C:/Users/Qrystian/Desktop/results.txt", ios::out);
    string S = S2BS("abc", "BigEndian");

    for (double t = 0 ; t < S.length(); t=t+0.001)
    {
        //File << t << " " << zA(0.0, 1.0, ((double)S.length()/0.001) * pow(1000, -1), 2 * M_PI,
S[floor(t)], t) << endl;
        //File << t << " " << zF(1.0, 1.0, 5.0, 2 * M_PI, S[floor(t)], t) << endl;
        //File << t << " " << zP(1.0, ((double)S.length() / 0.01) * pow(1000, -1), 0.0, M_PI,
S[floor(t)], t) << endl;
    }
```

```
    for (double t = 0; t < 2; t = t + 0.001)
    {
        //File << t << " " << zA(0.0, 1.0, ((double)S.length() / 0.001) * pow(1000, -1), 2 * M_PI, S[floor(t*5.0)], t) << endl;
        //File << t << " " << zF(1.0, 1.0, 5.0, 2 * M_PI, S[floor(5*t)], t) << endl;
        //File << t << " " << zP(1.0, ((double)S.length() / 0.001) * pow(1000, -1), 0.0, M_PI, S[floor(5*t)], t) << endl;
    }

    // Widma

    double* Tab;
    double* M;
    complex<double>* X;
    int size = (unsigned int)(1.0 / 0.001);
    Tab = new double[size];
    M = new double[size];

    int j = 0;
    for (double t = 0.0; t <= 1; t = t + 0.001)
    {
        //Tab[j] = zA(0.0, 1.0, ((double)S.length()/0.001) * pow(1000, -1), 2 * M_PI, S[floor(24*t)], t);
        //Tab[j] = zF(1.0, 1.0, 5.0, 2 * M_PI, S[floor(24*t)], t);
        Tab[j] = zP(1.0, ((double)S.length() / 0.01) * pow(1000, -1), 0.0, M_PI, S[floor(24*t)], t);
        j++;
    }
    size = (unsigned int)(1.0 / 0.001);
    X = DFT(Tab, size);

    for (j = 0; j < size; j++)
    {
        M[j] = sqrt(pow(X[j].real(), 2) + pow(X[j].imag(), 2));
        M[j] = 10 * log10(M[j]);
    }

    j = 0;
    for (double t = 0.0; t <= 1; t = t + 0.001)
    {
        File << j * (0.001 / size) << " " << M[j] << endl;
        j++;
    }

    // Szerokość zA(t) -> W = 0.056
    // Szerokość zF(t) -> W = 0.83
    // Szerokość zP(t) -> W = 1.10

    File.close();
}
```

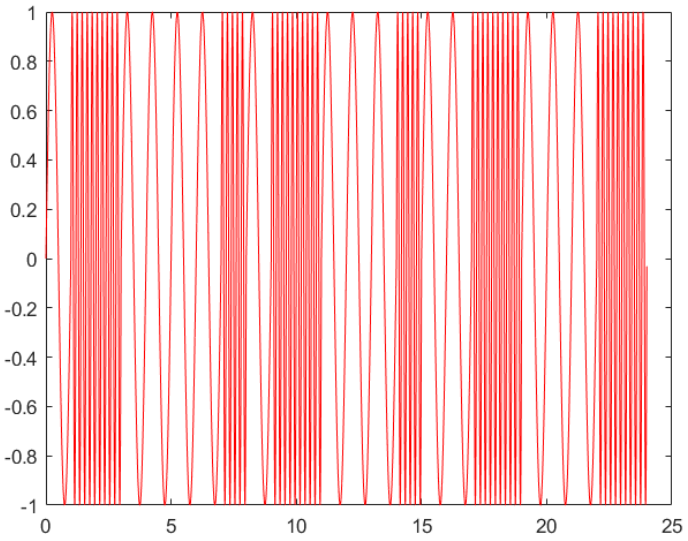
Opis kodu:

- Zadanie 1
- Zadanie 2a
- Zadanie 2b
- Zadanie 2c
- Zadanie 3
- Zadanie 4

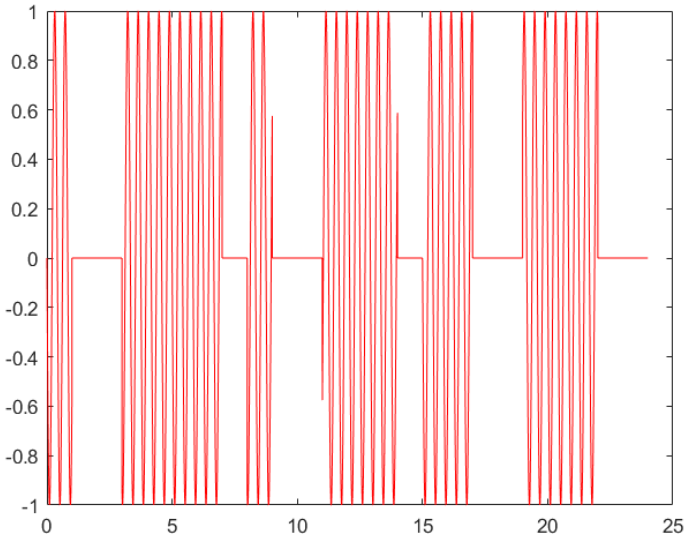
Wygenerowane wykresy:



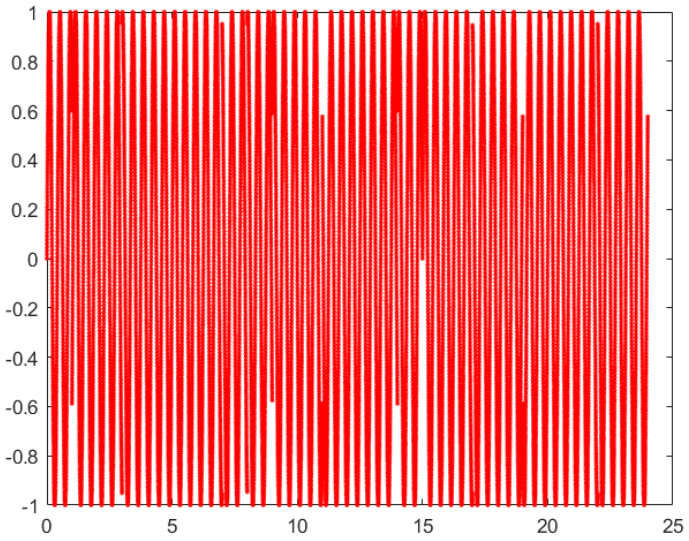
Wykres 1
Zadanie 2a



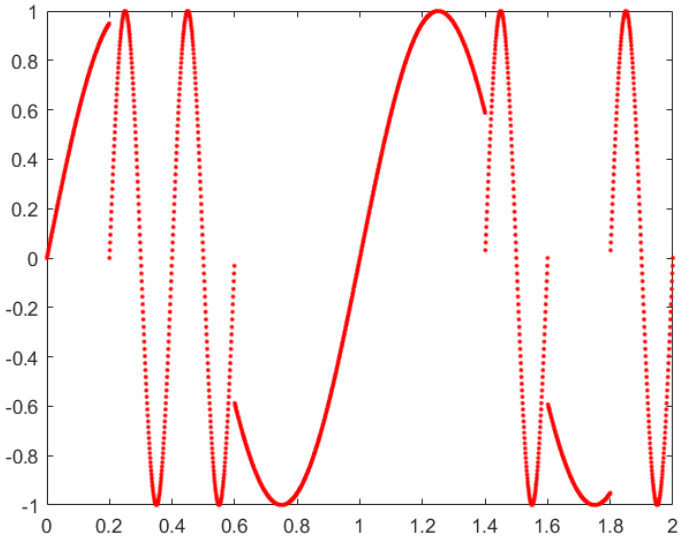
Wykres 2
Zadanie 2b



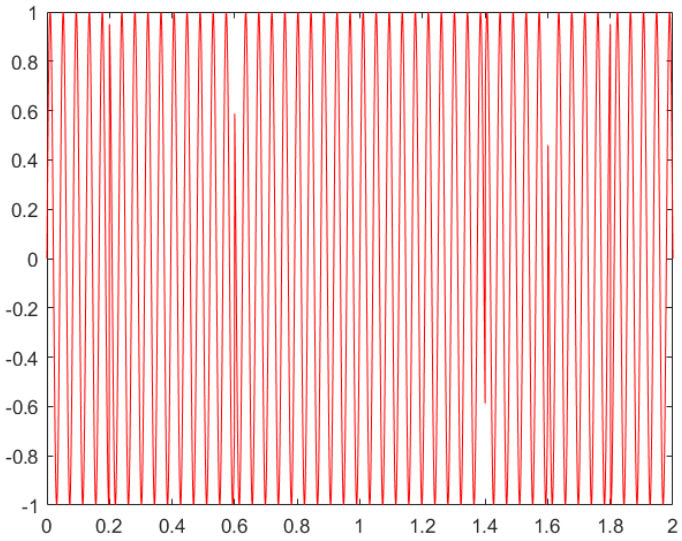
Wykres 3
Zadanie 2c



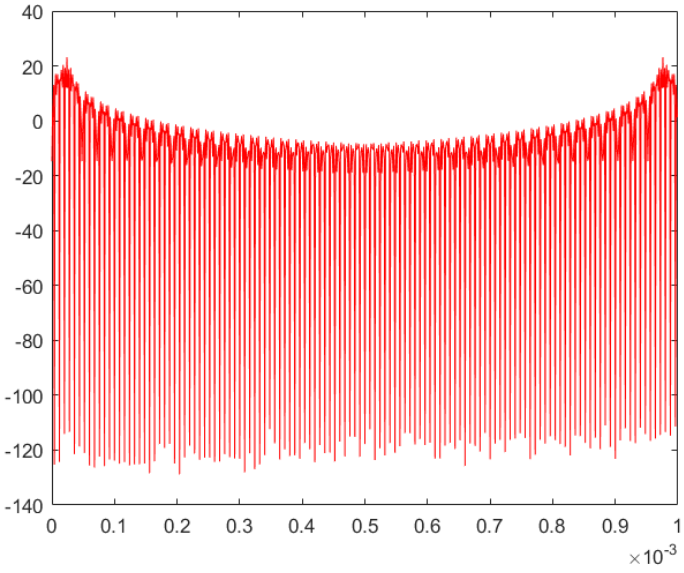
Wykres 4
Zadanie 3a



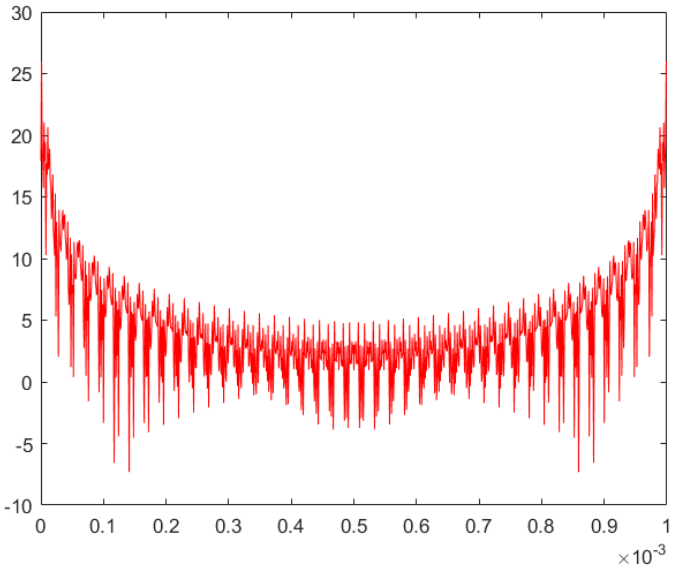
Wykres 5
Zadanie 3b



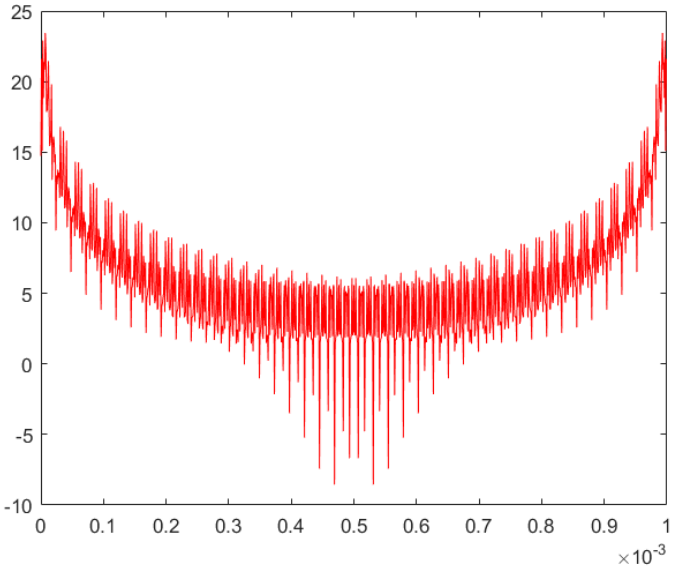
Wykres 6
Zadanie 3c



Wykres 7
Zadanie 4a



Wykres 8
Zadanie 4b



Wykres 9
Zadanie 4c