25.04.2020

Sprawozdanie z LAB 06

Zadaniem zajęć było wykonanie cyfrowej demodulacji amplitudy, częstotliwości i fazy dla funkcji bazowej.

Oto kod źródłowy:

```
#define _USE_MATH_DEFINES
//#define _PIXEL_SIZE_X "7680"
//#define _PIXEL_SIZE_X "4320"
#define _PIXEL_SIZE_X "3840"
#define _PIXEL_SIZE_X "2160"
#include<sstream>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstddef>
#include <bitset>
#include <iostream>
#include "my_plot.h"
#include "dft_algorithm.h"

#define ARG1 " with linespoints pointtype 6 pointsize 1 lc rgb 'red' "
#define ARG2 " with linespoints lc rgb 'red'  "
#define ARG3 " with linespoints pointtype 6 pointsize 0.3 lc rgb 'black' "

const std::string file_name("zad ");
std::string path = "C:\\Users\\GSzwa\\source\\repos\\TD_2020_44522\\LAB_06";

namespace nosne {
	double fm = 2.0;
	double fi = 0;

	//-----------------------------------
	double A0 = 1.0;
	double za(double t) // kluczowanie amplitudy
	{
		return (A0 * sin((2 * M_PI * fm * t) + fi));
	}

	//-----------------------------------
	double fm1 = 4.0;
	double fm0 = 1.0;
	double zf0(double t)// kluczowanie czestotliowsci
	{
```

```
                return A0 * sin(2 * M_PI * fm0 * t + fi);
        }
        double zf1(double t)// kluczowanie czestotliowsci
        {
                return A0 * sin(2 * M_PI * fm1 * t + fi);
        }

        //-----------------------------------
        double fi1 = M_PI;
        double zp(double t) // kluczowanie fazy
        {
                return A0 * sin(2 * M_PI * fm * t + fi1);
        }
}

std::vector<double> generate(double (*fun) (double const), double const start, double const koniec, double dok)
{
        std::vector<double> new_tab;

        long long l_dok = (koniec - start) / dok;
        double actual = start;
        for (int i = 0; i <= l_dok; i++)
        {
                new_tab.push_back(fun(actual));
                actual += dok;
        }
        return new_tab;
}

std::string demodulate_ask_psk(std::vector<std::pair<double, double>> input, double Tb,double ask = true)
{
        std::string wynik;
        double temp = 0;
        std::vector<std::pair<double, double>>::const_iterator start = input.begin(), start2 = input.begin();
        std::vector<double> wyn;
        std::advance(start2, 1);

        double H = (start2->first) - (start->first);
        int liczba_probek = Tb / H;
        auto last = input.end();
        while (start2 != last)
        {
                for (int i = 0; i < liczba_probek; i++)
                {
                        if (((start2->second) - (start->second)) > (start2->second))
                        {
                                temp += abs((H/2) * ((start2->second) / 2));
                                temp += abs((H/2) * ((start->second) / 2));
```

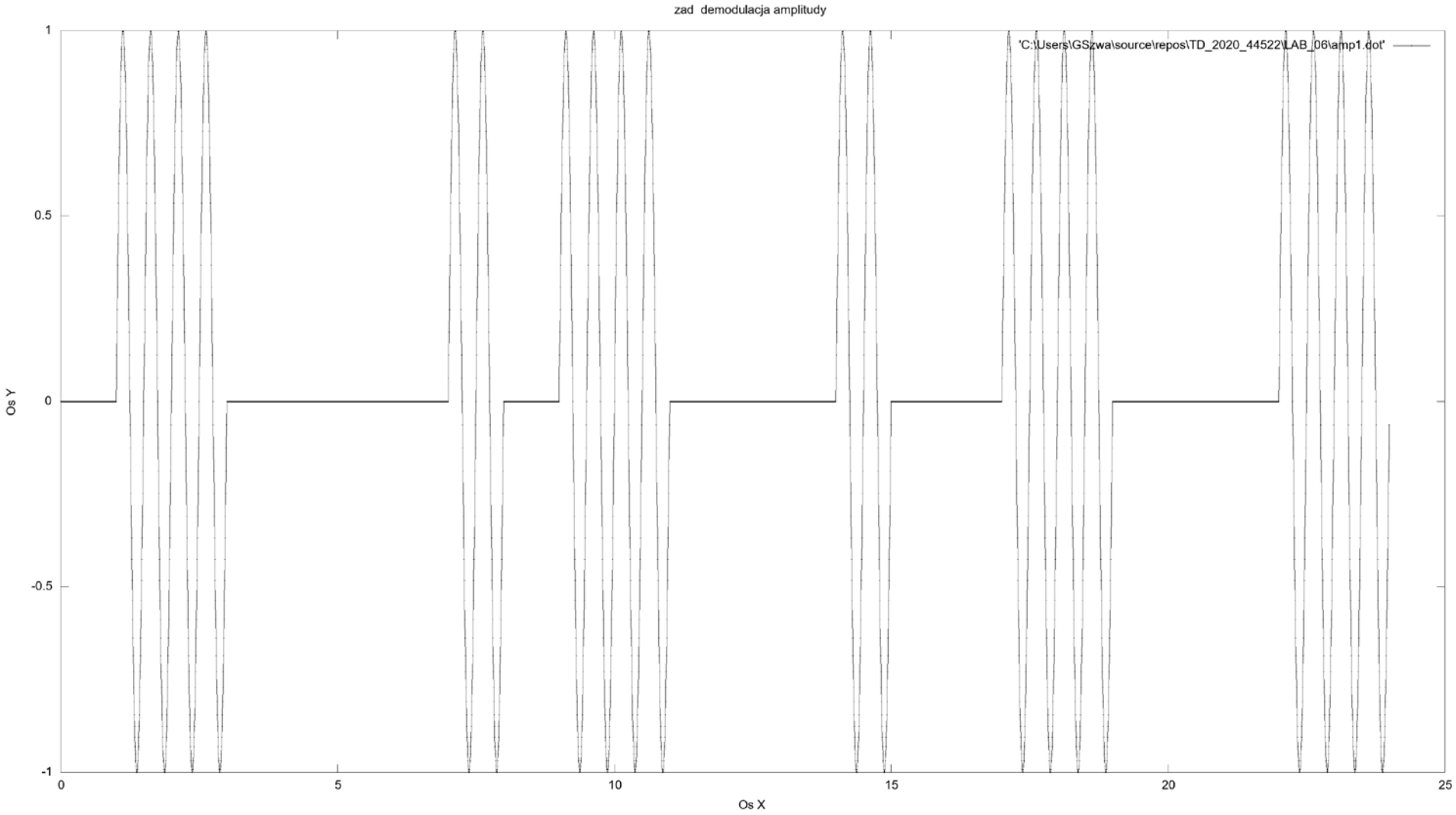
```
                        }
                        else
                        {
                                temp += abs(H * (((start2->second) + (start->second)) / 2));
                        }

                        start2++;
                        start++;
                }
                wyn.push_back(temp);

                temp = 0;
        }
        std::string ppp = dft::save_file_real(path + "\\calka.dot", wyn,H);

        if (ask)
        {
                my_plot wykres(path, file_name + " demodulacji amplitudy calka ");
                wykres.read_file(ppp, ARG3);
                wykres.print_plot();
        }
        else
        {
                my_plot wykres(path, file_name + " demodulacji fazy calka ");
                wykres.read_file(ppp, ARG3);
                wykres.print_plot();
        }


        auto minmax = std::minmax_element(wyn.begin(), wyn.end());
        double odciecie = (*minmax.first + *minmax.second) / 2;

        for (std::vector<double>::iterator it = wyn.begin(); it != wyn.end(); it++)
        {
                if (*it < odciecie) wynik.append("0");
                else wynik.append("1");
        }

        return wynik;
}

std::string demodulate_fsk(std::vector<std::pair<double, double>> input1, std::vector<std::pair<double, double>> input2, double Tb)
{
        std::string wynik;
        double temp1 = 0,temp2 = 0;
        std::vector<std::pair<double, double>>::const_iterator first1 = input1.begin(), second1 = input1.begin();
        std::vector<std::pair<double, double>>::const_iterator first2 = input2.begin(), second2 = input2.begin();
        std::vector<double> wyn;
```

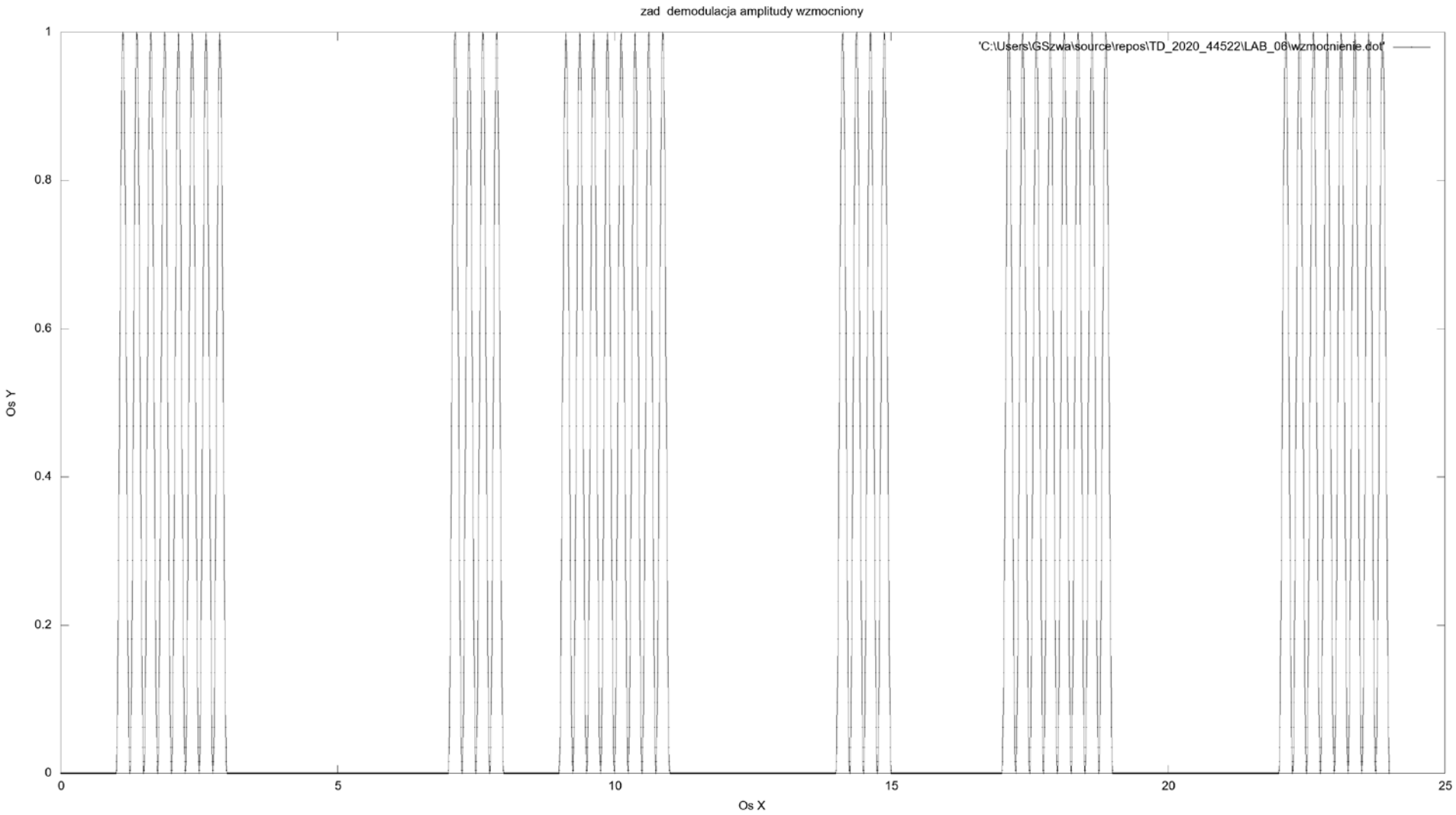
```
	std::advance(second1, 1);
	std::advance(second2, 1);

	double H = (second1->first) - (first1->first);
	int liczba_probek = Tb / H;
	auto last = input1.end();
	while (second1 != last)
	{
		for (int i = 0; i < liczba_probek; i++)
		{
			if (((second1->second) - (first1->second)) > (second1->second))
			{
				temp1 += abs((H / 2) * ((second1->second) / 2));
				temp1 += abs((H / 2) * ((first1->second) / 2));
			}
			else
			{
				temp1 += abs(H * (((second1->second) + (first1->second)) / 2));
			}

			if (((second2->second) - (first2->second)) > (second2->second))
			{
				temp2 += abs((H / 2) * ((second2->second) / 2));
				temp2 += abs((H / 2) * ((first2->second) / 2));
			}
			else
			{
				temp2 += abs(H * (((second2->second) + (first2->second)) / 2));
			}

			second1++;
			first1++;
			second2++;
			first2++;
		}
		wyn.push_back(temp2-temp1);

		temp1 = 0;
		temp2 = 0;
	}

	std::string ppp1 = dft::save_file_real(path + "\\calka.dot", wyn, H);
	my_plot wykres1(path, file_name + " demodulacji amplitudy calka ");
	wykres1.read_file(ppp1, ARG3);
	wykres1.print_plot();

	auto minmax = std::minmax_element(wyn.begin(), wyn.end());
	double odciecie = (*minmax.first + *minmax.second) / 2;
```

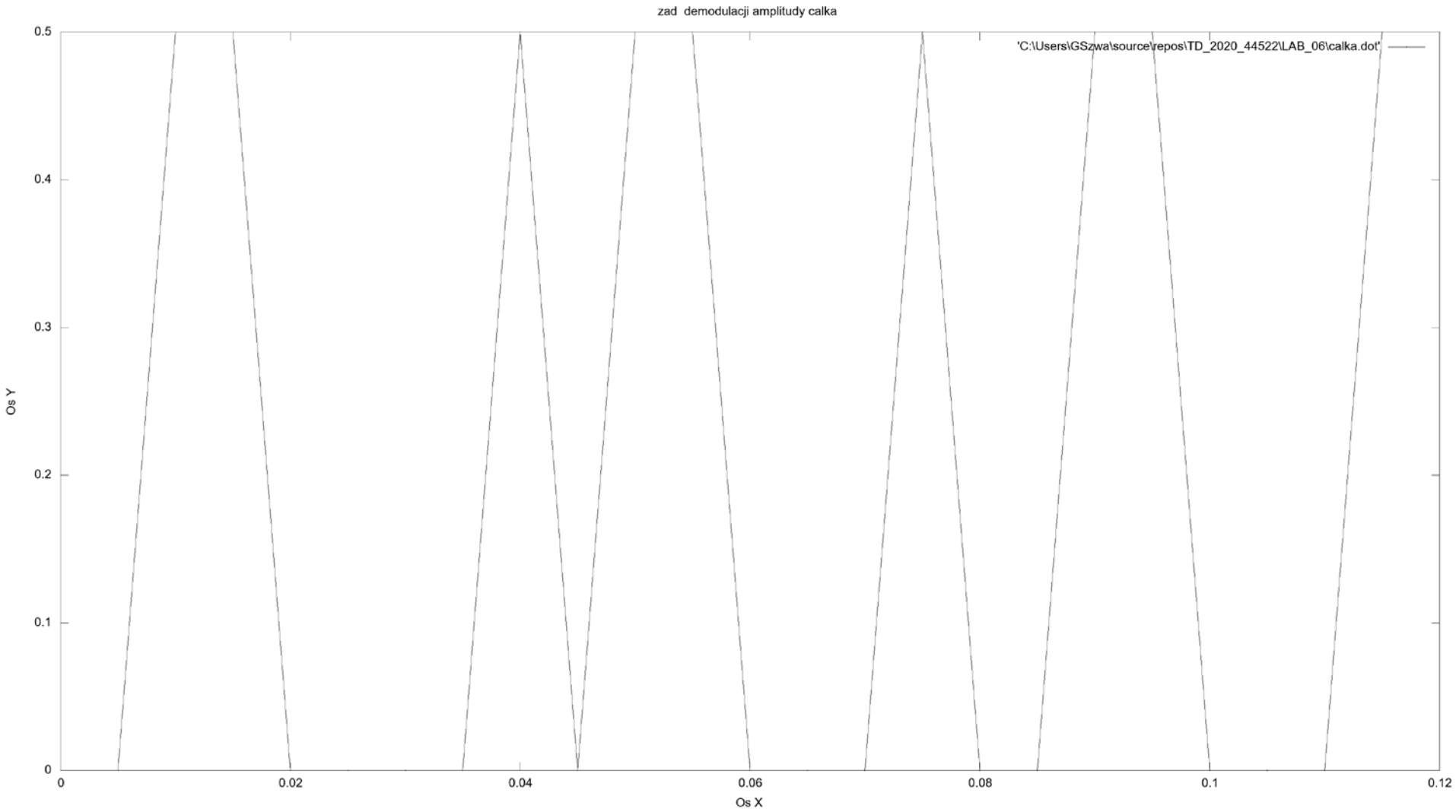
```
	for (std::vector<double>::iterator it = wyn.begin(); it != wyn.end(); it++)
	{
		if (*it < odciecie) wynik.append("0");
		else wynik.append("1");
	}

	return wynik;
}

double informacyjny(double t, int const data)
{
	return data == 0 ? 0 : 1;
}

int main()
{
	std::string path_wykres1;
	//std::cout << sygnal[4800].first << " " << sygnal[4800].second << " * " << nosny[4800] << " --- " << wynik[4800].first << " " <<
wynik[4800].second << std::endl;

	double Tb = 1.0; //zad2
	double Ts = 0.005;
	int rozmiar = 24;
	int N = rozmiar * (1 / Tb);
	nosne::fm0 = 1;
	nosne::fm1 = 4;

	path_wykres1 = path + "\\amp1.dot";
	my_plot wykresA(path, file_name + " demodulacja amplitudy ");
	wykresA.read_file(path_wykres1, ARG3);
	wykresA.print_plot();

	std::vector<std::pair<double, double>> sygnal = dft::load_file_real(path_wykres1);
	std::vector<double> nosny = generate(nosne::za, 0, Tb * rozmiar, Ts);
	std::vector<std::pair<double, double>> wynik = dft::multiply(sygnal, nosny);
	path_wykres1 = dft::save_file_real(path + "\\wzmocnienie.dot", wynik);

	my_plot wykres1(path, file_name + " demodulacja amplitudy wzmocniony");
	wykres1.read_file(path_wykres1, ARG3);
	wykres1.print_plot();
	std::string binary = demodulate_ask_psk(wynik, 1.0);
	my_plot wykres1B(path, file_name + " demodulacja amplitudy wynik");
	wykres1B.function_plot(informacyjny, ARG1, binary, Ts, Tb);
	wykres1B.print_plot();


	//------------------------------------------------------------------
```

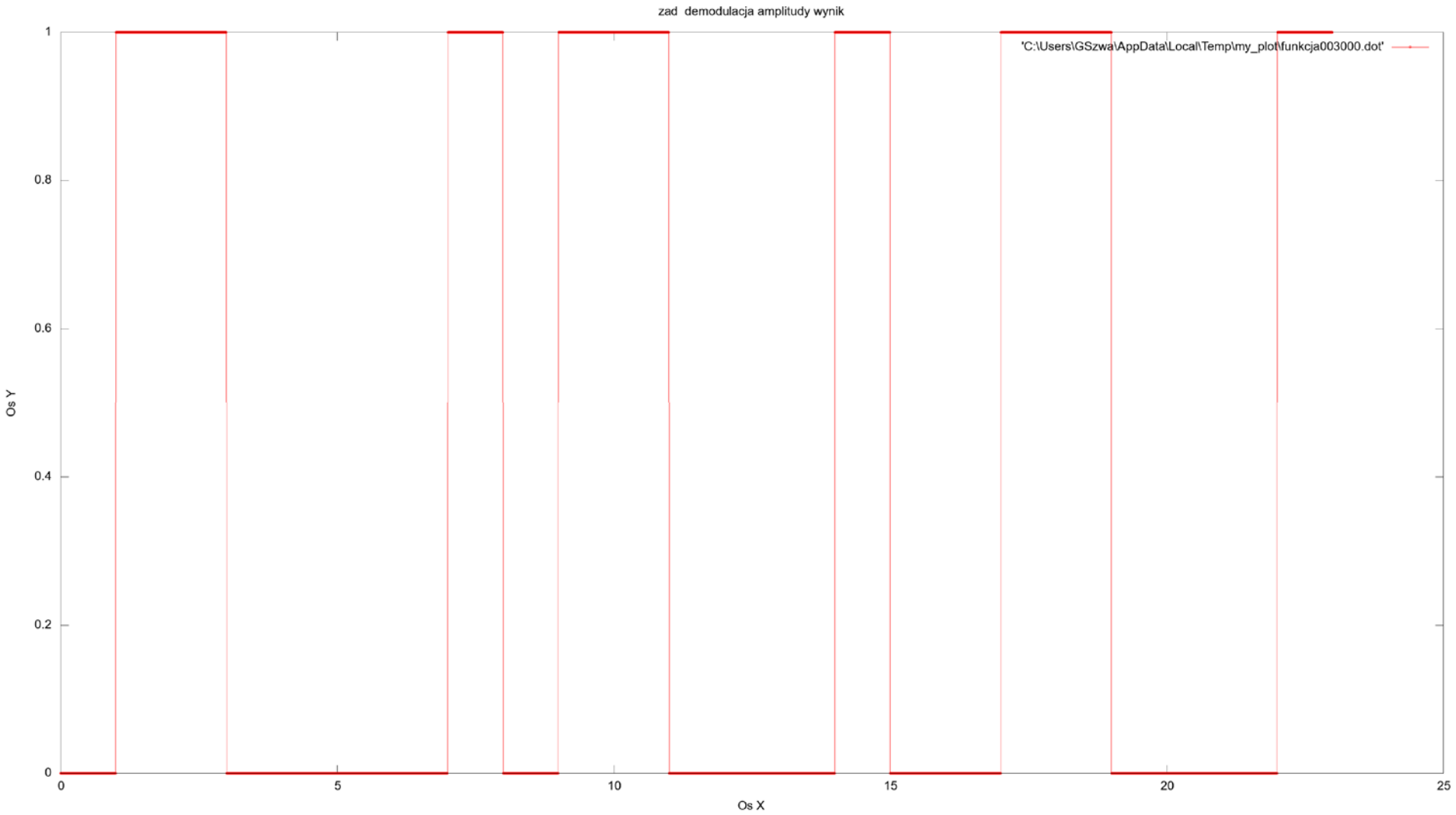
```
        path_wykres1 = path + "\\faz1.dot";
        my_plot wykresB(path, file_name + " demodulacja fazy ");
        wykresB.read_file(path_wykres1, ARG3);
        wykresB.print_plot();
        sygnal = dft::load_file_real(path_wykres1);
        nosny = generate(nosne::zp, 0, Tb * rozmiar, Ts);
        wynik = dft::multiply(sygnal, nosny);
        path_wykres1 = dft::save_file_real(path + "\\wzmocnienie.dot", wynik);

        my_plot wykres2(path, file_name + " demodulacja fazy wzmocniony");
        wykres2.read_file(path_wykres1, ARG3);
        wykres2.print_plot();
        binary = demodulate_ask_psk(wynik, 1.0, false);
        my_plot wykres2B(path, file_name + " demodulacja fazy wynik");
        wykres2B.function_plot(informacyjny, ARG1, binary, Ts, Tb);
        wykres2B.print_plot();

        //-------------------------------------------------------------
        path_wykres1 = path + "\\czes1.dot";
        my_plot wykresC(path, file_name + " demodulacja czestotliwosci ");
        wykresC.read_file(path_wykres1, ARG3);
        wykresC.print_plot();
        sygnal = dft::load_file_real(path_wykres1);
        std::vector<double> nosny0 = generate(nosne::zf0, 0, Tb * rozmiar, Ts);
        std::vector<std::pair<double, double>> wynik0 = dft::multiply(sygnal, nosny0);
        path_wykres1 = dft::save_file_real(path + "\\wzmocnienie.dot", wynik);
        my_plot wykres3_A(path, file_name + " demodulacja czestotliwosci wzmocniony pierwszy");
        wykres3_A.read_file(path_wykres1, ARG3);
        wykres3_A.print_plot();

        std::vector<double> nosny1 = generate(nosne::zf1, 0, Tb * rozmiar, Ts);
        std::vector<std::pair<double, double>> wynik1 = dft::multiply(sygnal, nosny1);
        my_plot wykres3_B(path, file_name + " demodulacja czestotliwosci wzmocniony drugi");
        wykres3_B.read_file(path_wykres1, ARG3);
        wykres3_B.print_plot();
        binary = demodulate_fsk(wynik0, wynik1, 1.0);
        my_plot wykres3(path, file_name + " demodulacja czestotliwosci wynik");
        wykres3.function_plot(informacyjny, ARG1, binary, Ts, Tb);
        wykres3.print_plot();

}
```

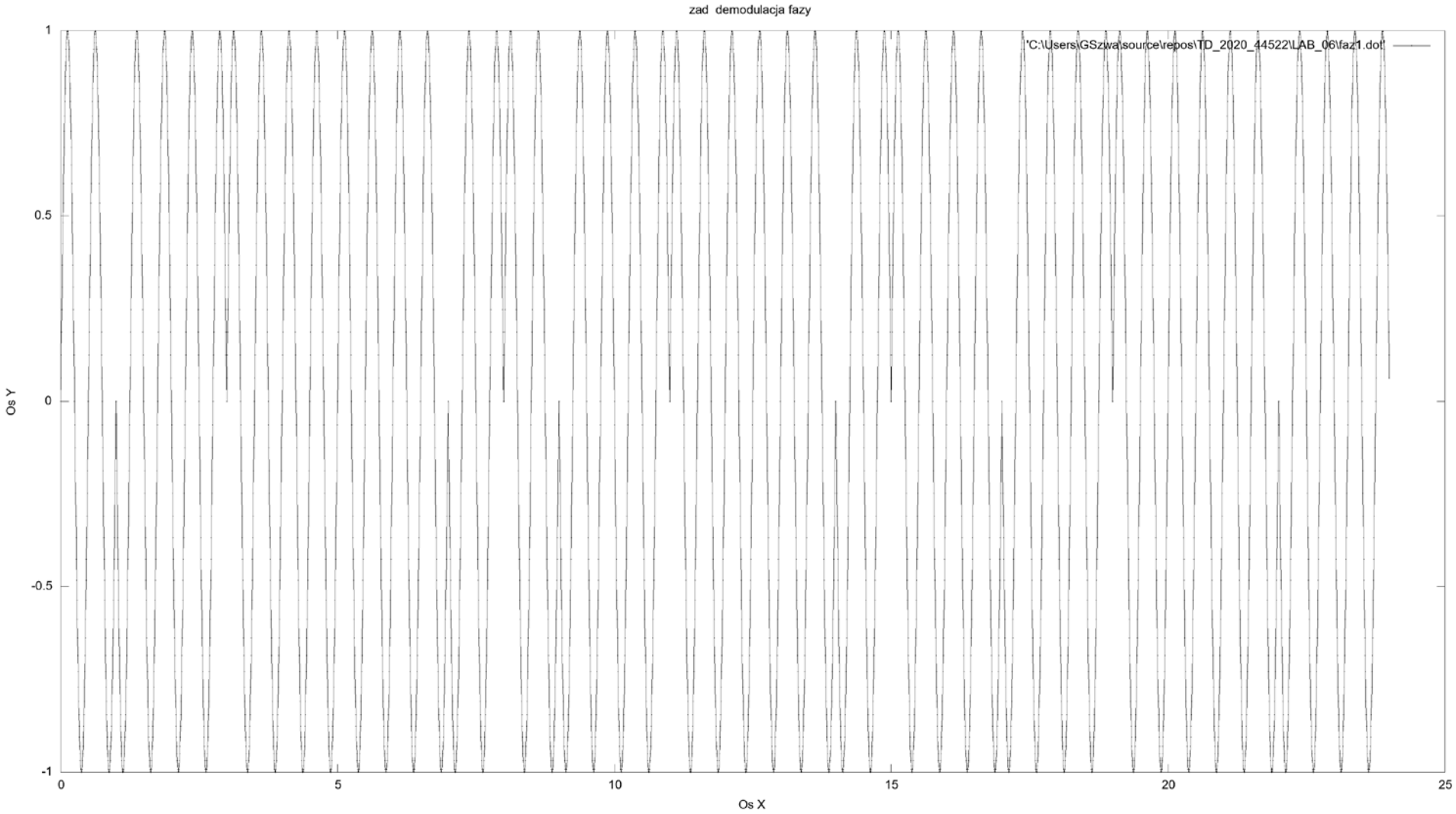
Podczas zajęć sporządziłem 13 wykresów (wraz z sygnałami informacyjnymi).

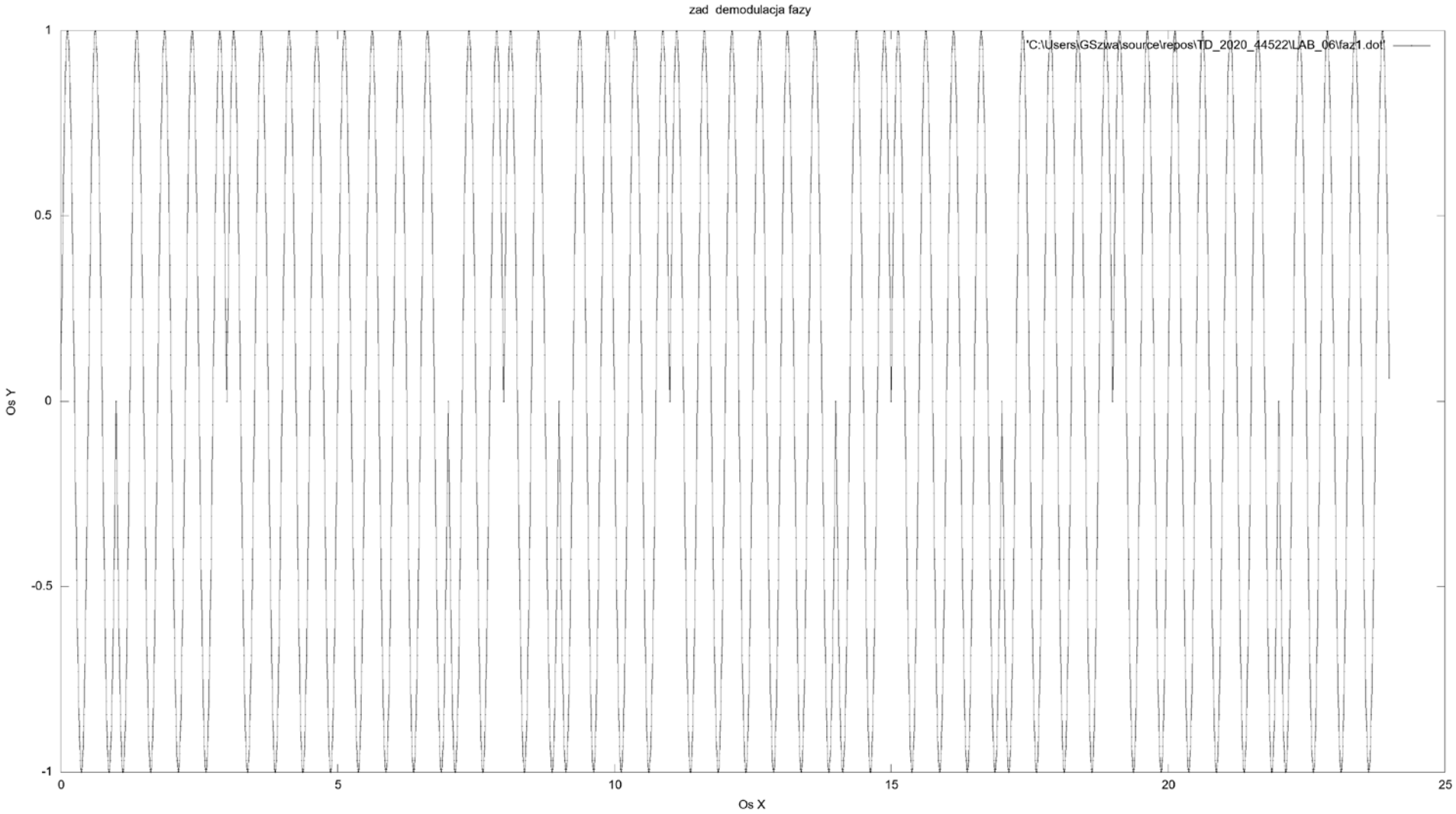


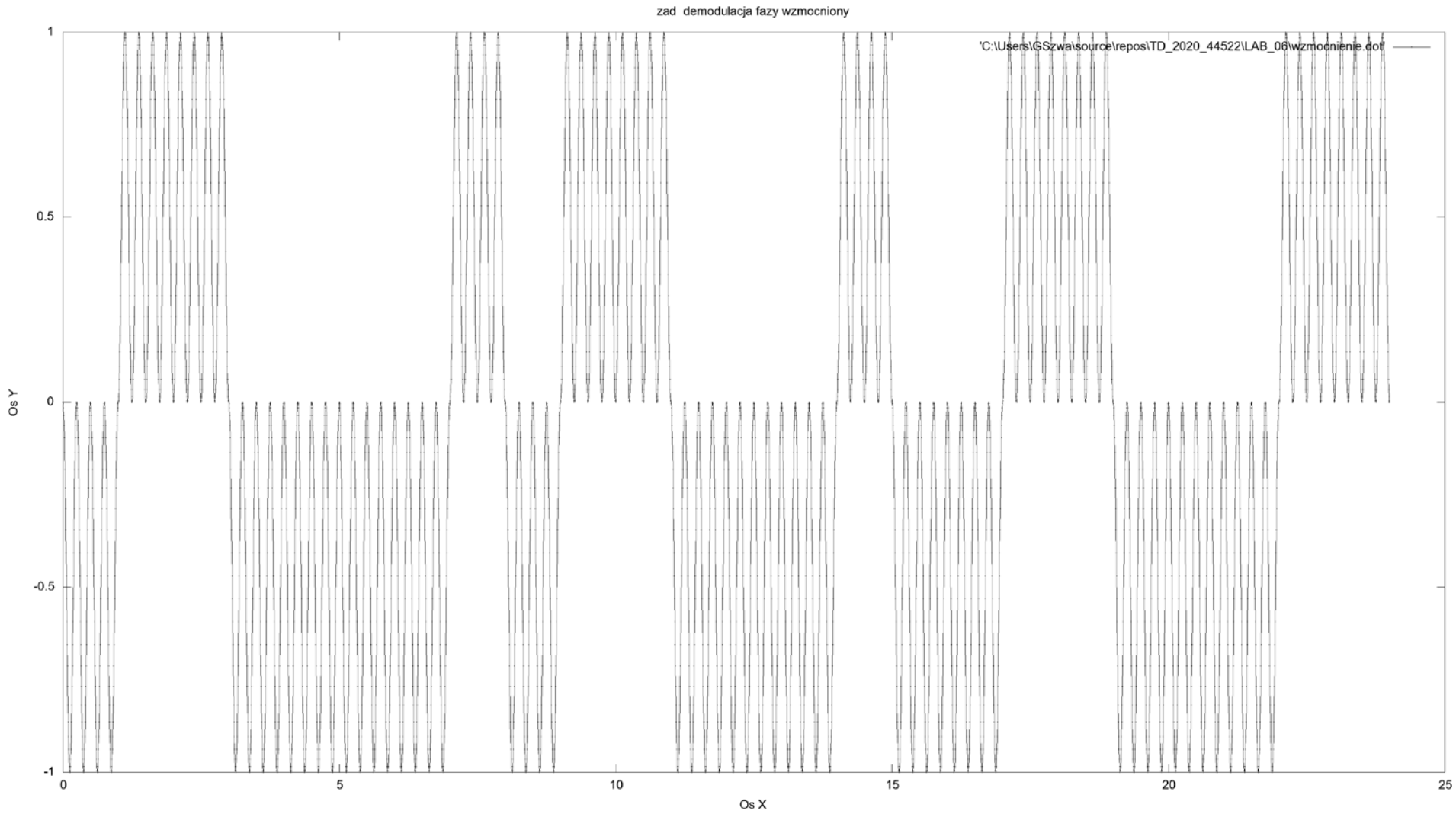

zad demodulacja amplitudy wzmocniony
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\wzmocnienie.dat'
Os Y
Os X
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25

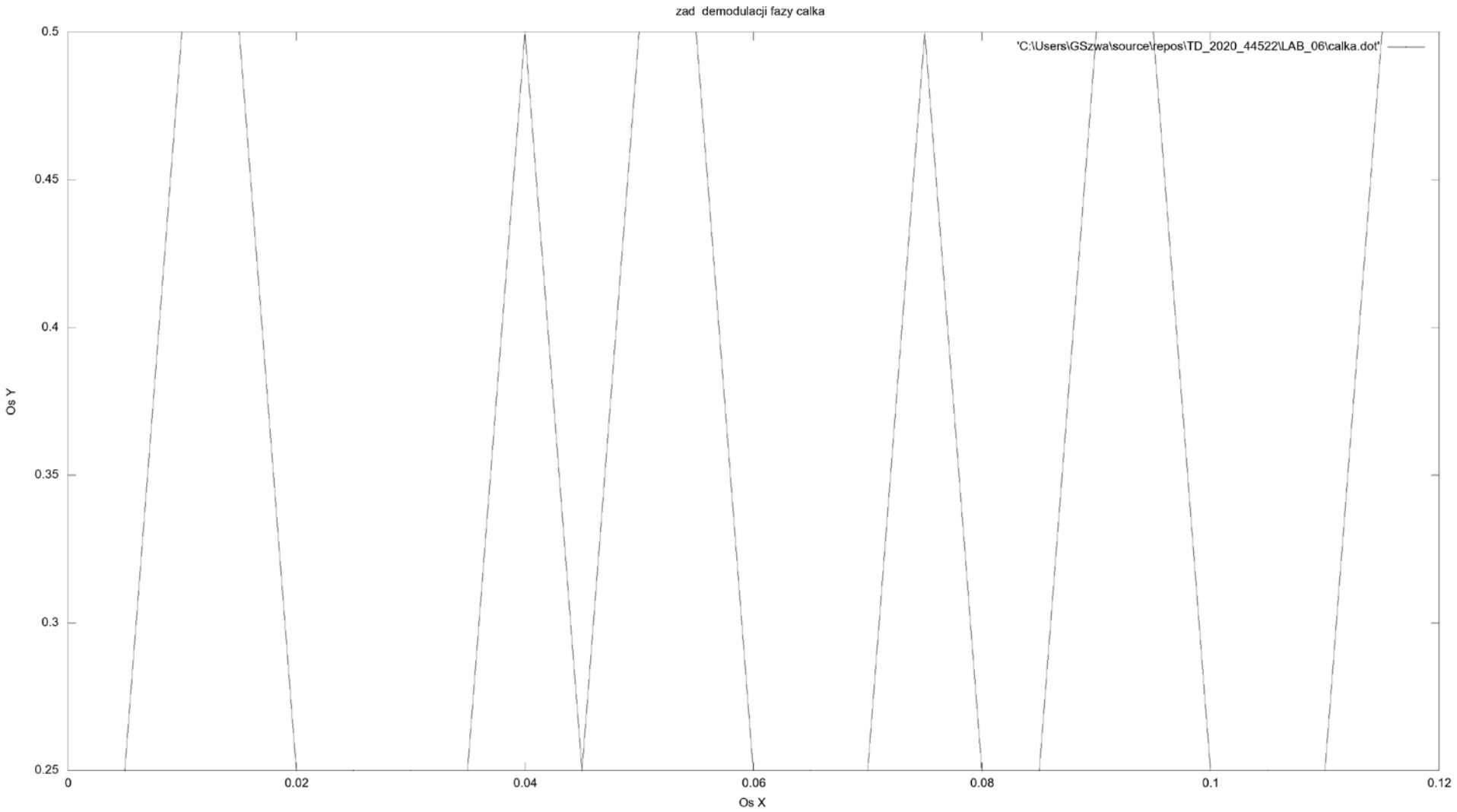

zad demodulacji amplitudy calka
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\calka.dot'
0.5
0.4
0.3
0.2
0.1
0
Os Y
0
0.02
0.04
0.06
0.08
0.1
0.12
Os X

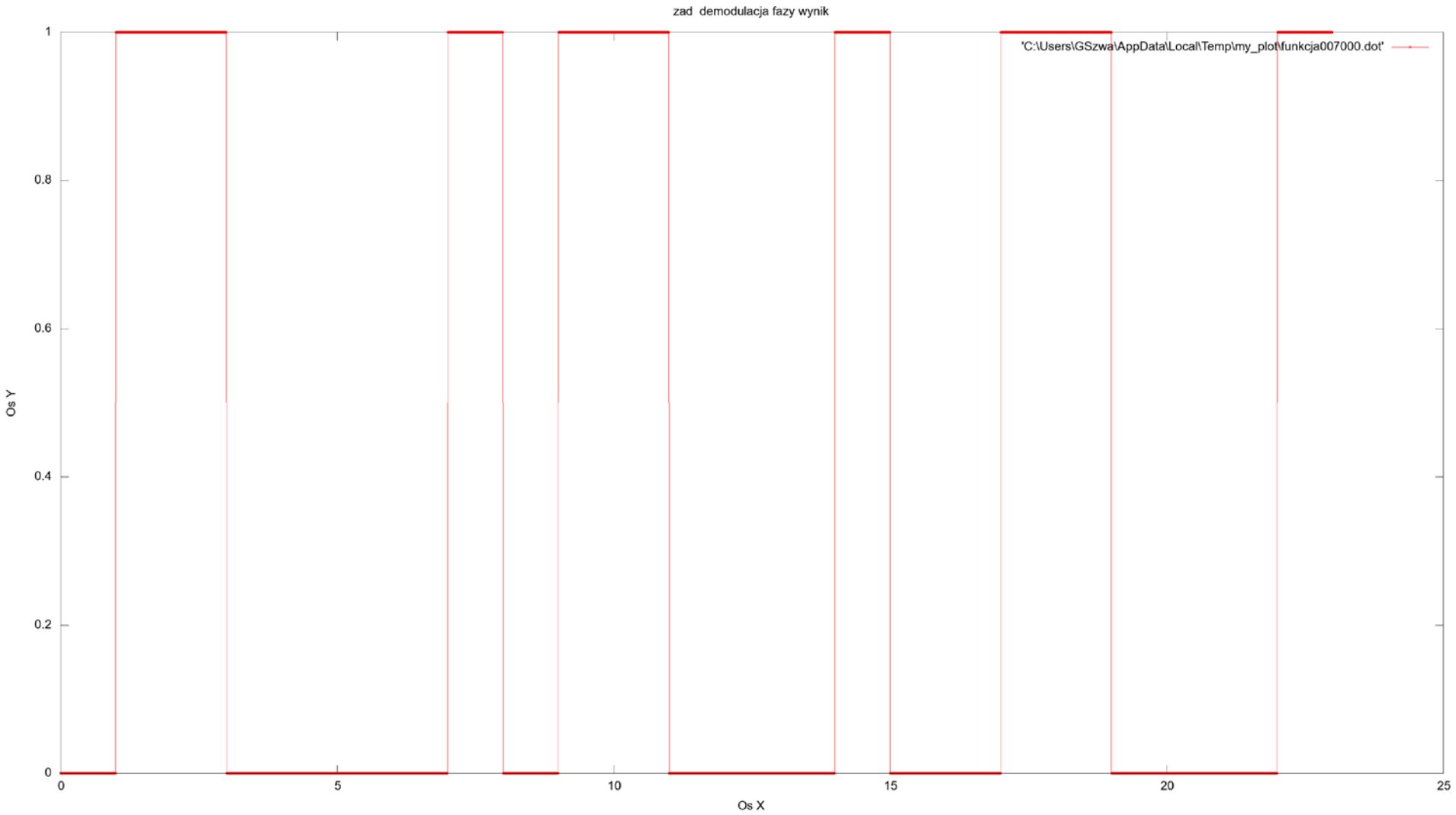

zad demodulacja amplitudy wynik
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja003000.dot'
Os Y
Os X
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25

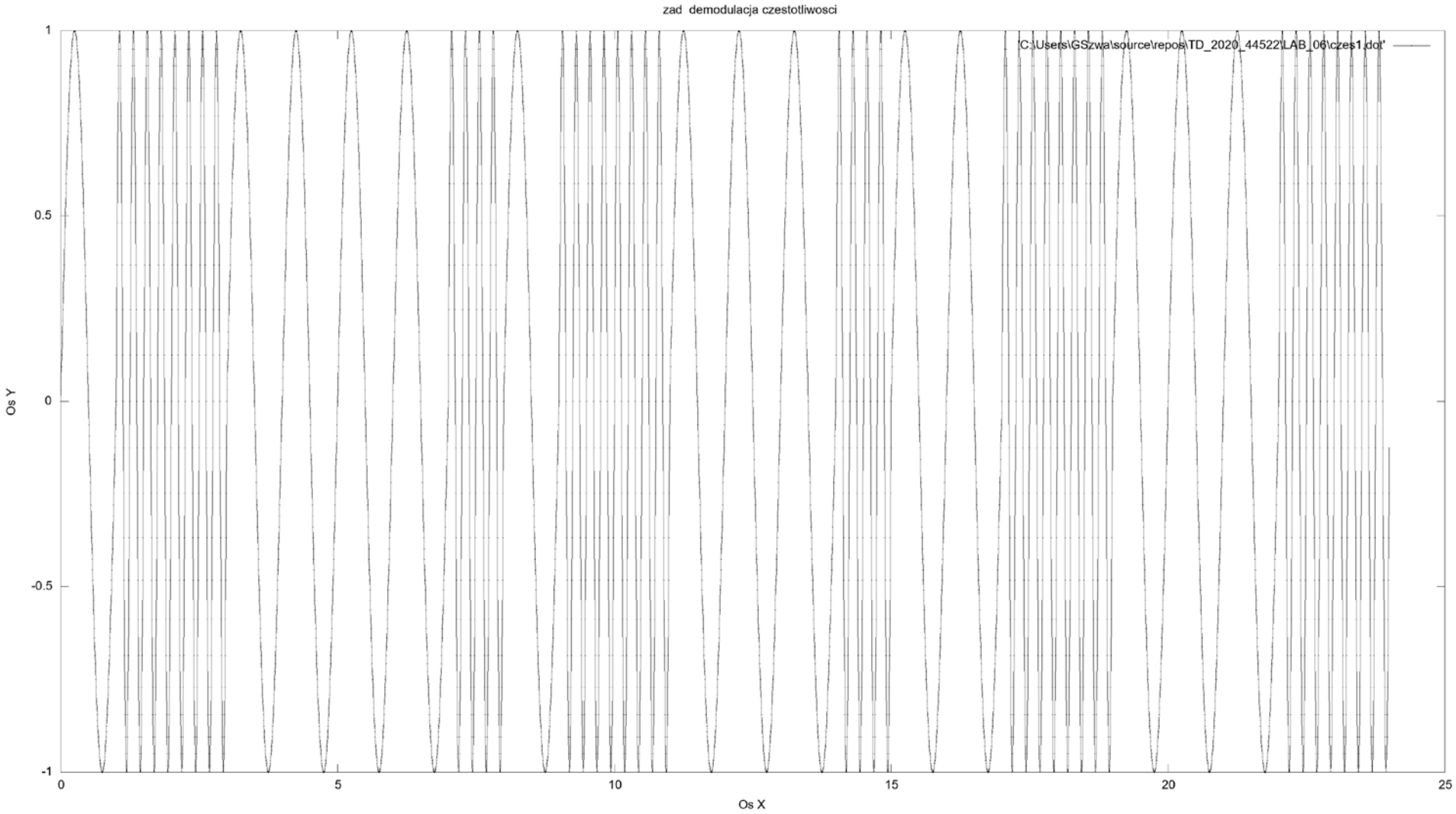

zad demodulacja fazy
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\faz1.dat'
Os Y
Os X
1
0.5
0
-0.5
-1
0
5
10
15
20
25

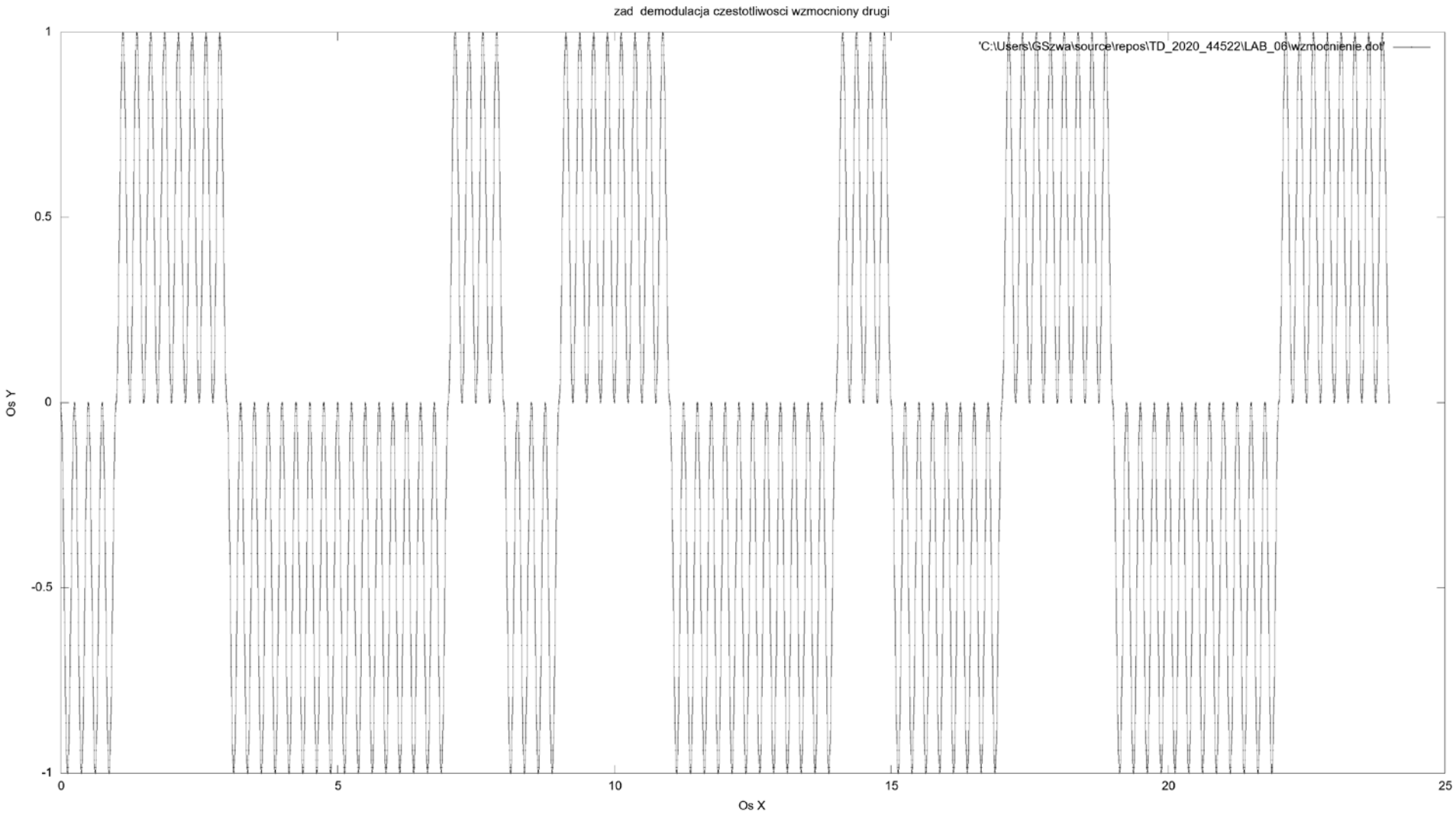

zad demodulacja fazy
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\faz1.dat'
Os Y
Os X
1
0.5
0
-0.5
-1
0
5
10
15
20
25

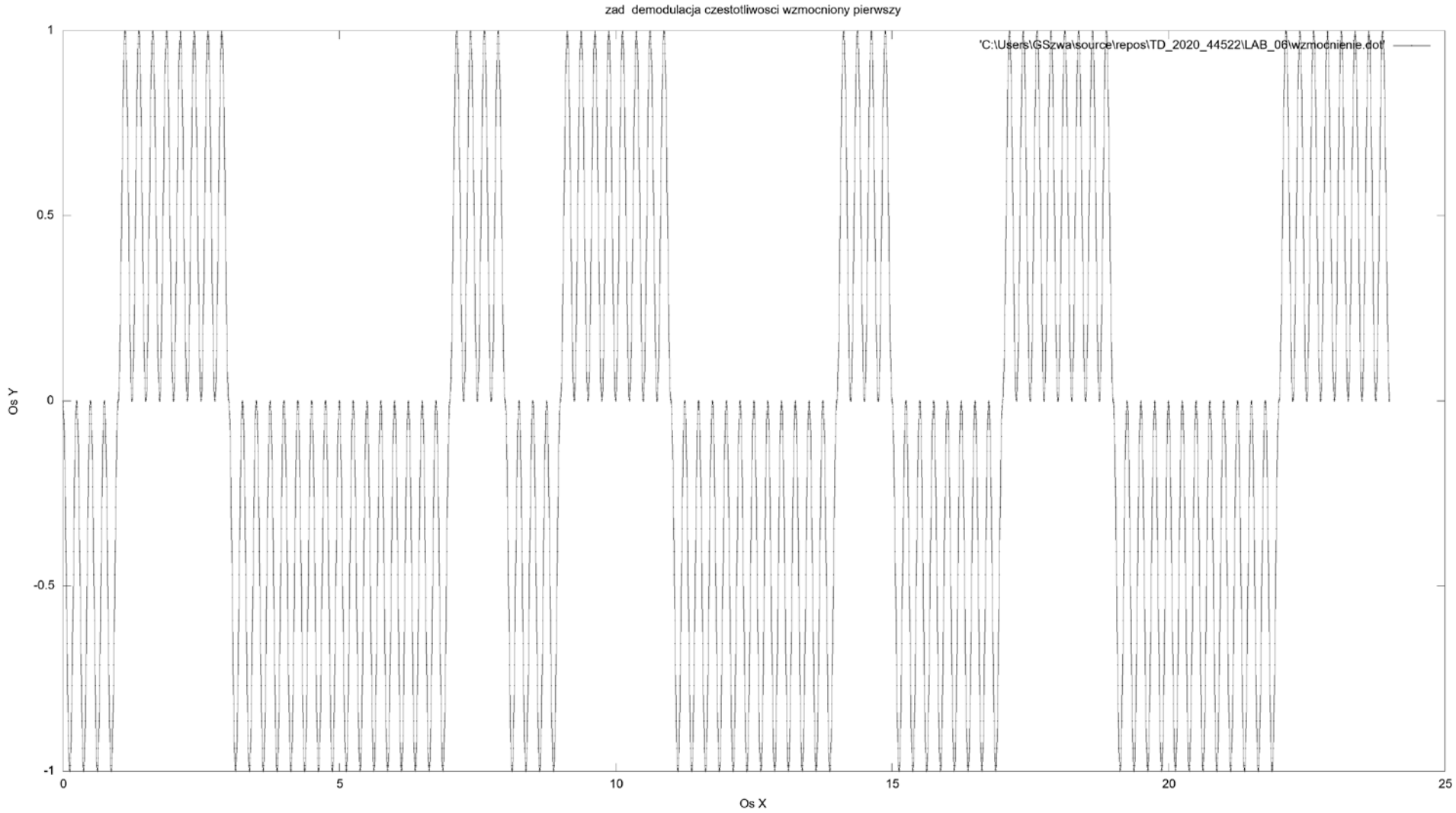

zad demodulacja fazy wzmocniony
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\wzmocnienie.dat'
Os Y
Os X
1
0.5
0
-0.5
-1
0
5
10
15
20
25

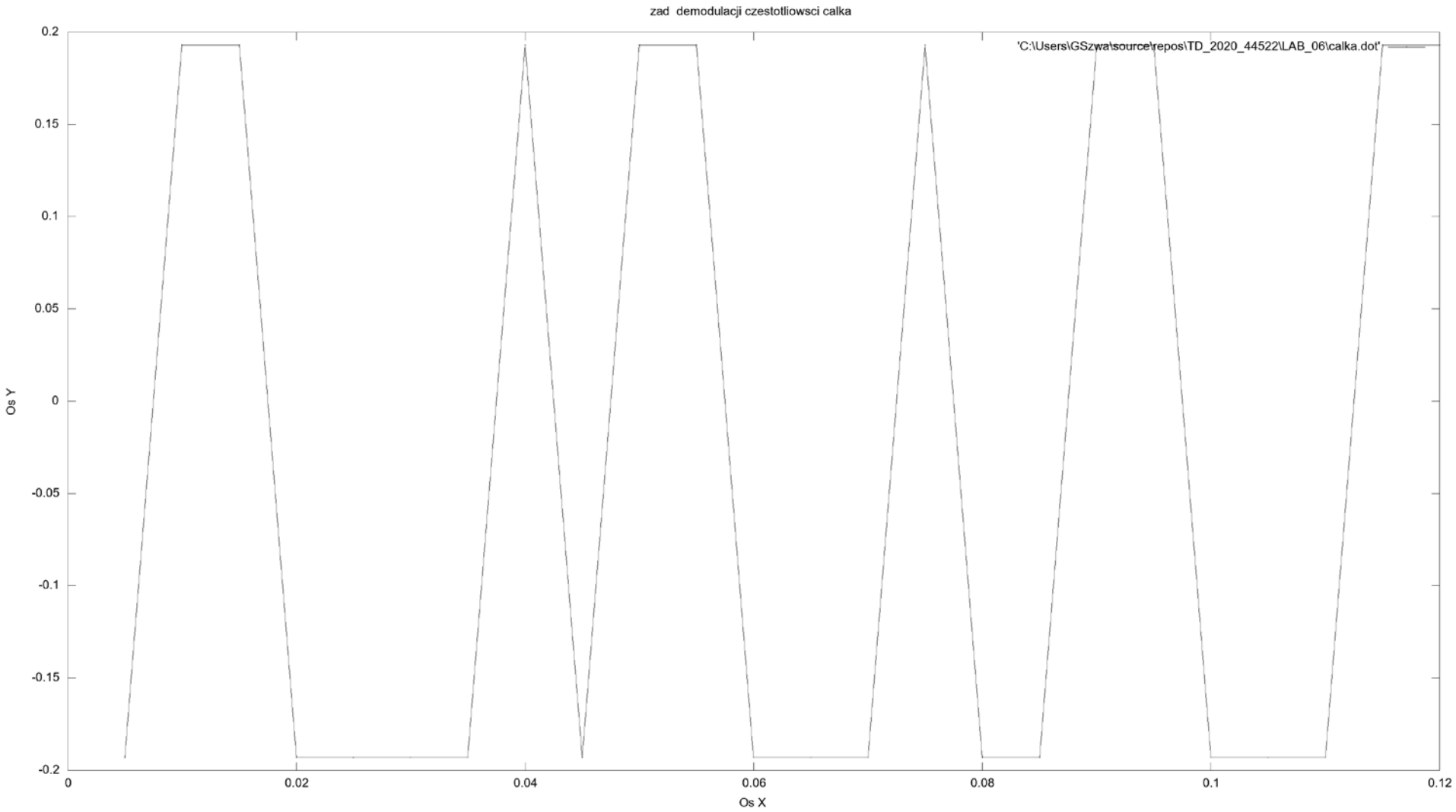

zad demodulacji fazy calka
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\calka.dot'
0.5
0.45
0.4
0.35
0.3
0.25
Os Y
0
0.02
0.04
0.06
0.08
0.1
0.12
Os X

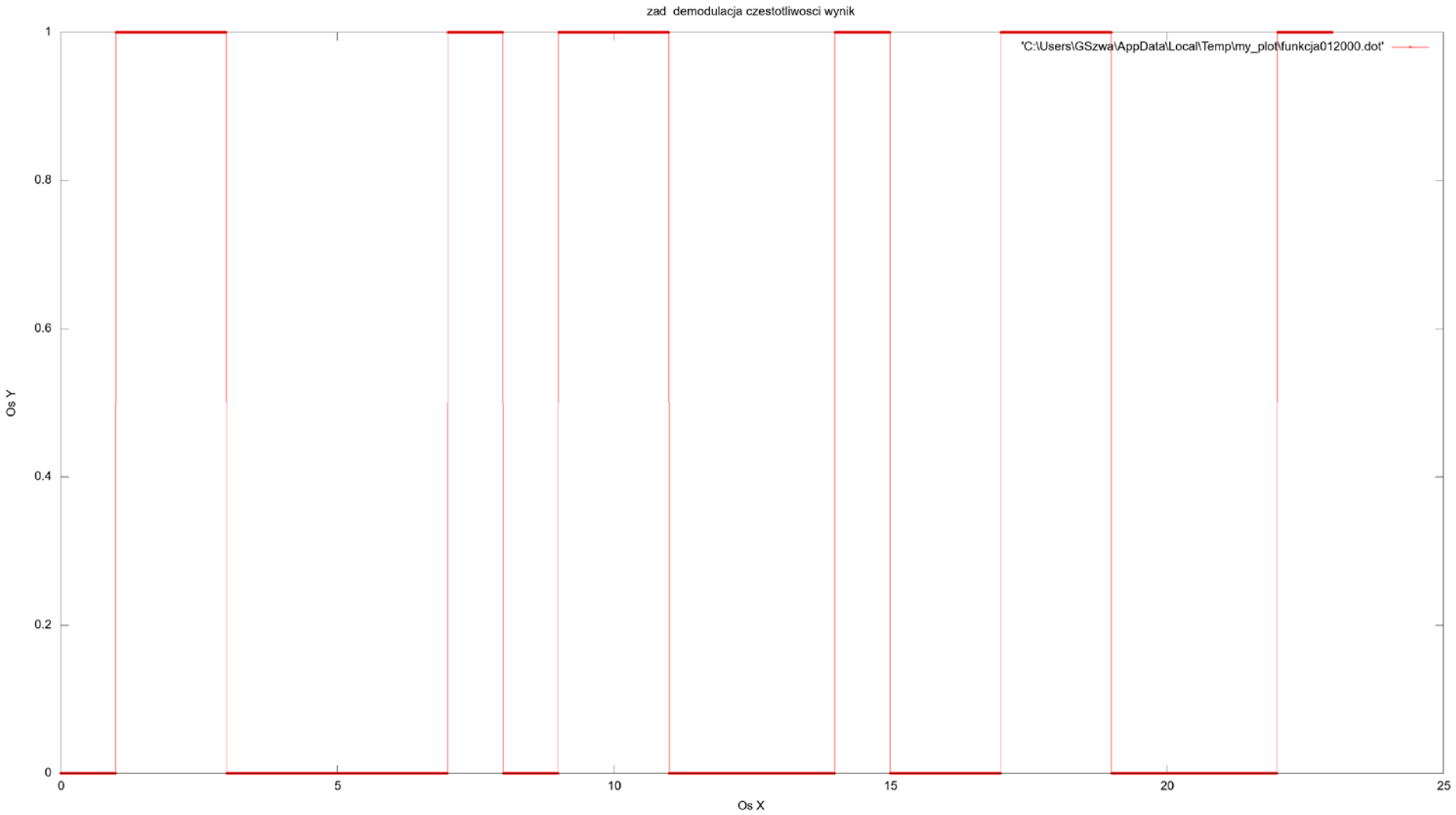

zad demodulacja fazy wynik
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja007000.dot'
Os Y
Os X
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25


zad demodulacja czestotliwosci
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\czes1.dat'
Os Y
Os X
1
0.5
0
-0.5
-1
0
5
10
15
20
25


zad demodulacja czestotliwosci wzmocniony drugi
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\wzmocnienie.dat'
Os Y
Os X
1
0.5
0
-0.5
-1
0
5
10
15
20
25


zad demodulacja czestotliwosci wzmocniony pierwszy
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\wzmocnienie.dat'
1
0.5
0
-0.5
-1
Os Y
0
5
10
15
20
25
Os X


zad demodulacji czestotliowsci calka
'C:\Users\GSzwa\source\repos\TD_2020_44522\LAB_06\calka.dot'
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
Os Y
0
0.02
0.04
0.06
0.08
0.1
0.12
Os X


zad demodulacja czestotliwosci wynik
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja012000.dot'
Os Y
Os X
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25

Podsumowanie

Dzięki tym laboratoriom nauczyłem się cyfrowej demodulacji sygnału fazy, częstotliwości i amplitudy, poznałem jak odbierany jest sygnał cyfrowy aby zakłócenia na niego nie wpływały, co pozwala na zrozumienie w jaki sposób jest wysyłany sygnał za pomocą analogowego ośrodka. Wiedza to jest wykorzystywana w transmisji danych stanowiących rdzeń komunikacji pomiędzy urządzeniami cyfrowymi.

Wykonał Szwarc Grzegorz