Laboratorium 3 - aproksymacja

Tomasz Belczyk

24.04.2021

Metody Obliczeniowe w Nauce i Technice
Informatyka niestacjonarna 2020/2021
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

1. Treść zadań

1. Proszę zastosować aproksymację Czebyszewa dla funkcji:

- y = exp((x**2) w przedziale od -1 do 1
- y = abs(x+x**3) w przedziale od -1 do 1
- y = sign(x) w przedziale od -1 do 1

Do aproksymacji należy użyć biblioteki GSL.

Dla każdej funkcji proszę:

- narysować wykres funkcji aproksymowanej i aproksymującej (gnuplot)
- sprawdzić, jak wynik zależy od stopnia wielomianu aproksymującego - przedstawić odpowiednie wykresy

2. Podejście do rozwiązania zadań

Rozwiązując zadania wykorzystamy narzędzie VS Code jak i obraz docker build'a uruchomiony na dockerze pozwalający na kompilowanie, uruchamianie programów w ubuntu. Do pokazywania danych użyjemy gnuplot

Zadanie 1

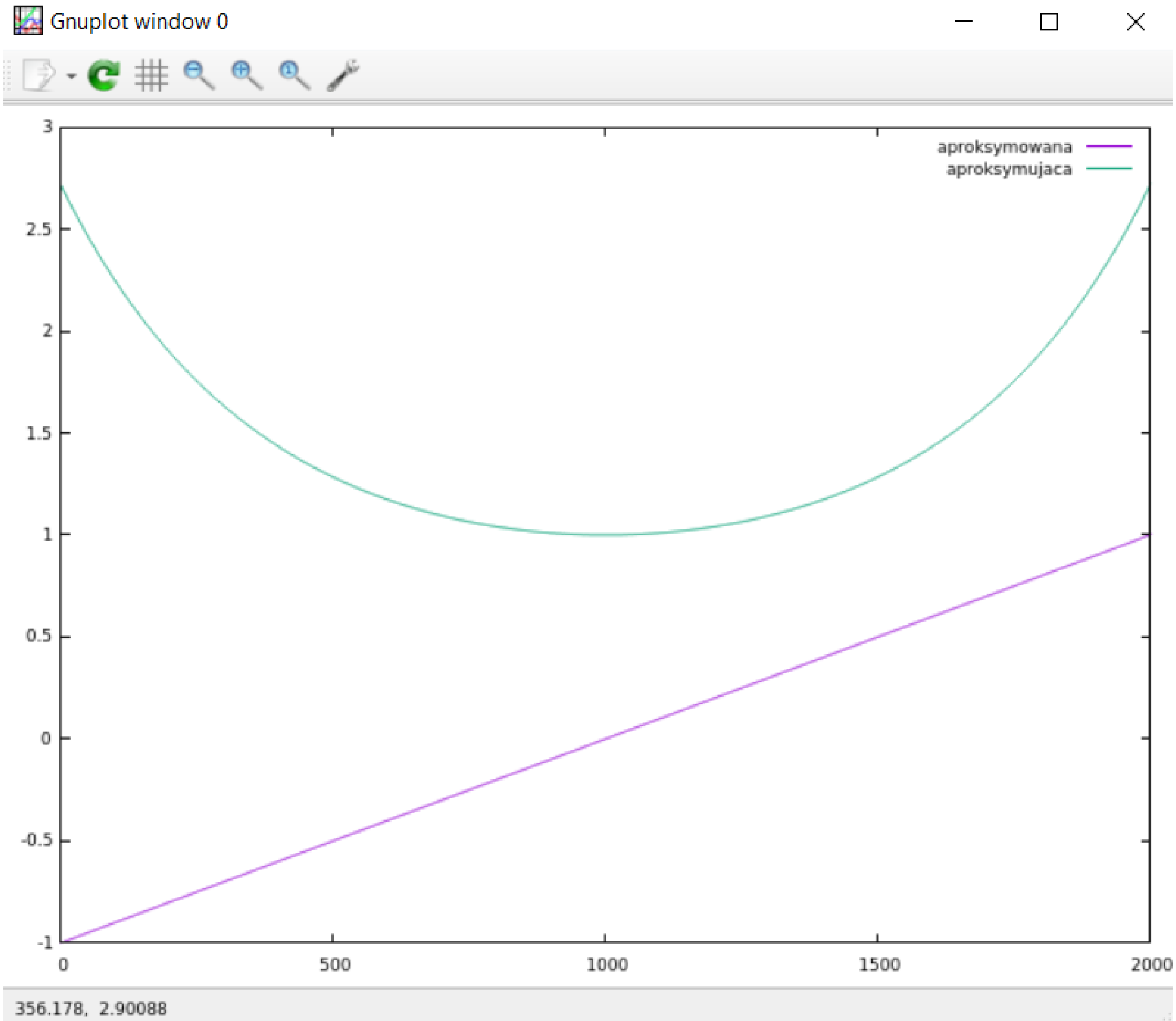
Implementujemy algorytm do wyliczenia aproksymacji który zapisze wyniki do pliku aproksymacja.txt

```c
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_chebyshev.h>

double f(double x, void *p)
{
  (void)(p); /* avoid unused parameter warning */

  return exp(x * x);
}

int main (void)
{
  FILE *output;
  int i, n = 500;
  output = fopen("aproksymacja.txt", "w");
  gsl_cheb_series *cs = gsl_cheb_alloc (40);

  gsl_function F;

  F.function = f;
  F.params = 0;

  gsl_cheb_init (cs, &F, -1.0, 1.0);

  for (i = -1000; i <= 1000; i++)
  {
    double x = i / (double) 1000;
    double r2 = gsl_cheb_eval_n (cs, 2, x);
    double r5 = gsl_cheb_eval_n (cs, 5, x);
    double def = gsl_cheb_eval (cs, x);
    fprintf (output,"%g %g %g %g %g\n", x, GSL_FN_EVAL(&F, x), def, r2, r5);
  }

  gsl_cheb_free (cs);

  return 0;
}
```

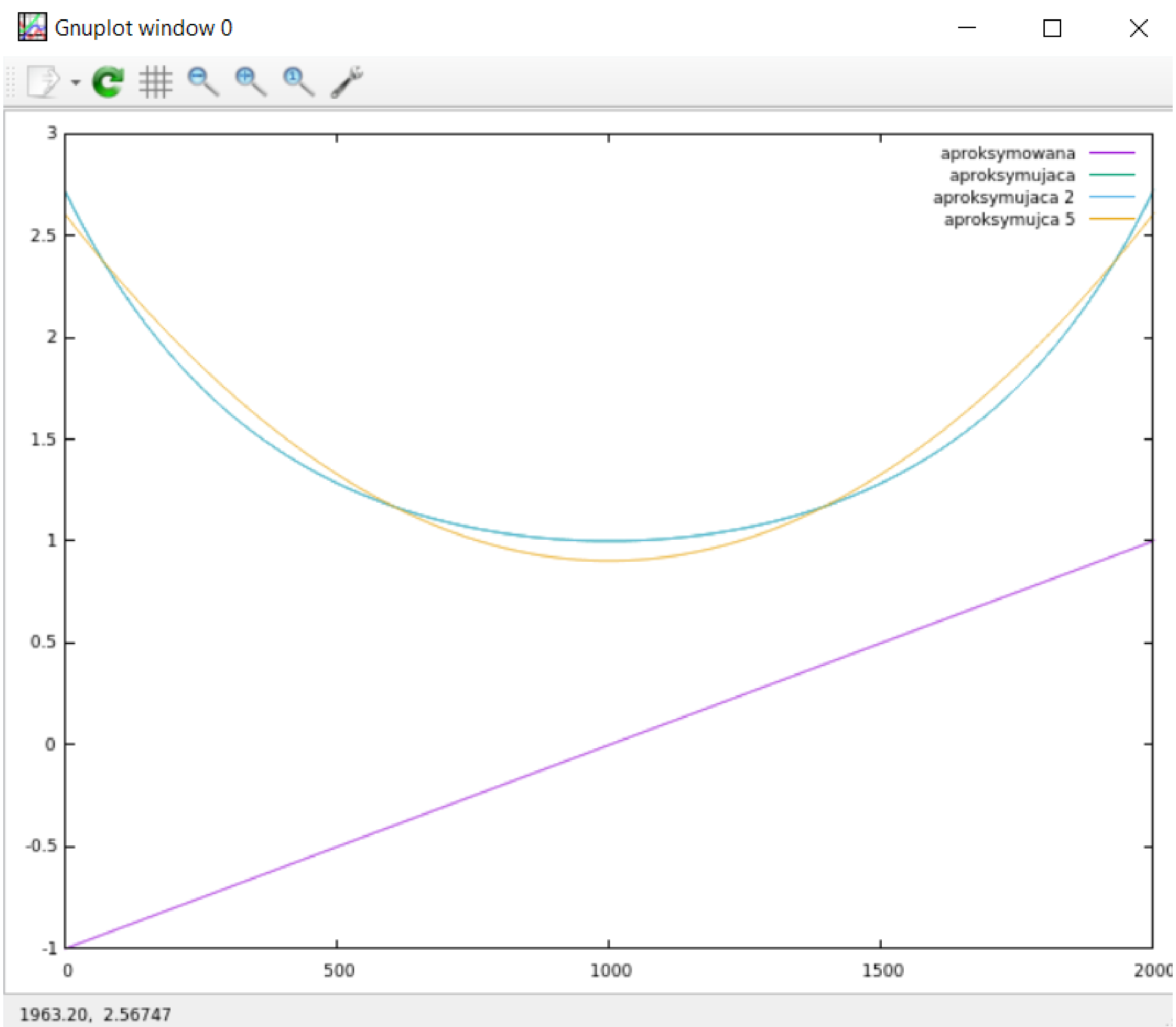
Zaimplementowała została funkcja pierwsza exp(x**2). Jako rezultat otrzymujemy funkcje aproksymowaną, aproksymująca, aproksymująca wielomian 2, aproksymująca wielomian 5. Kompilujemy i uruchamiamy program a następnie używając gnuplot wpisujemy:

plot "aproksymacja.txt" using 0:1 title 'aproksymowana' with lines, "aproksymacja.txt" using 0:2 title 'aproksymujaca' with lines

W wyniku otrzymujemy wykres:



Prezentuje się on następująco dla wielomianów 2 i 5:

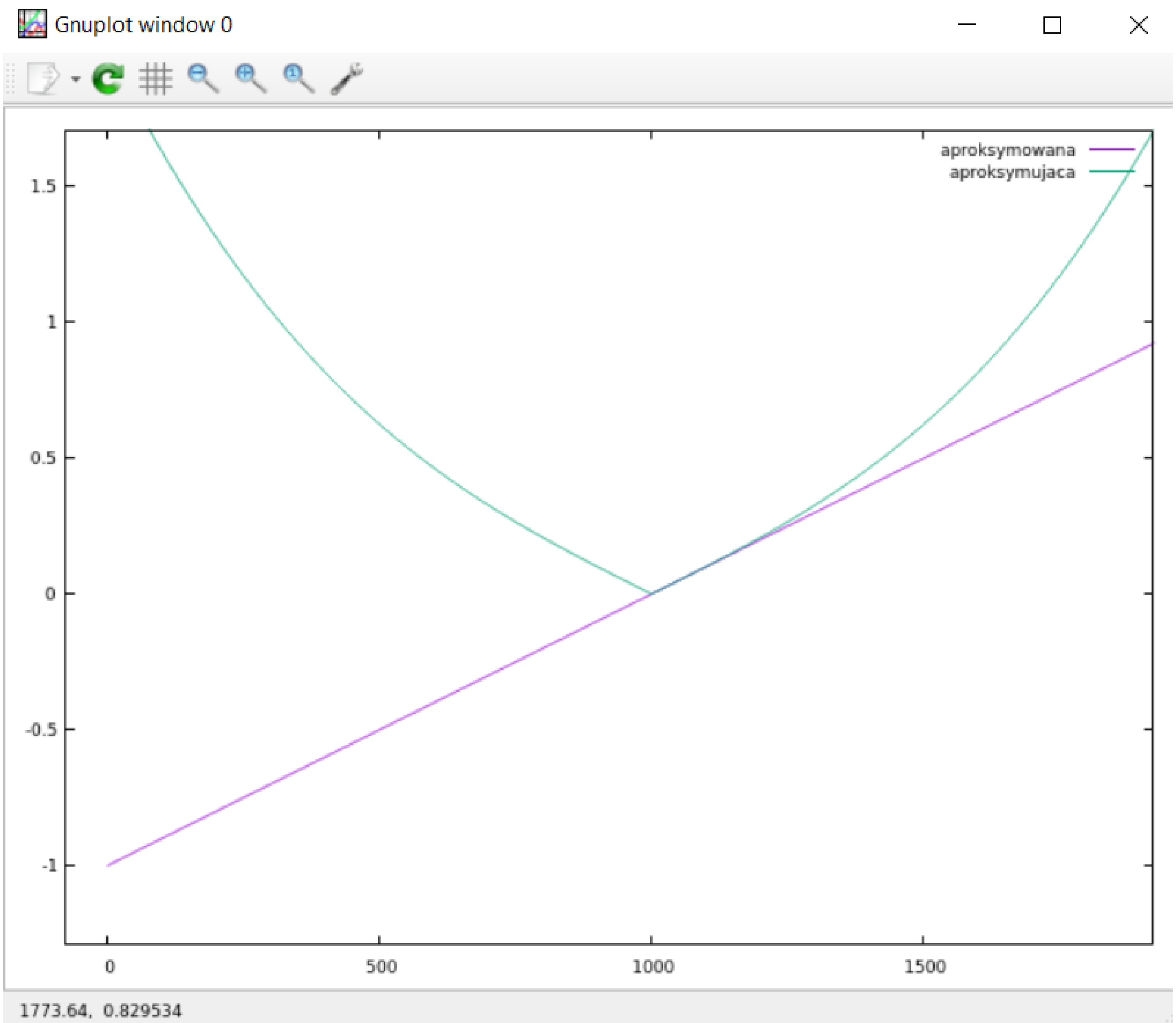


Jak widać przy funkcji exp(x**2) nie ma różnicy w stopniu wielomianu funkcji aproksymującej, niebieska linia i zielona nachodzą na siebie

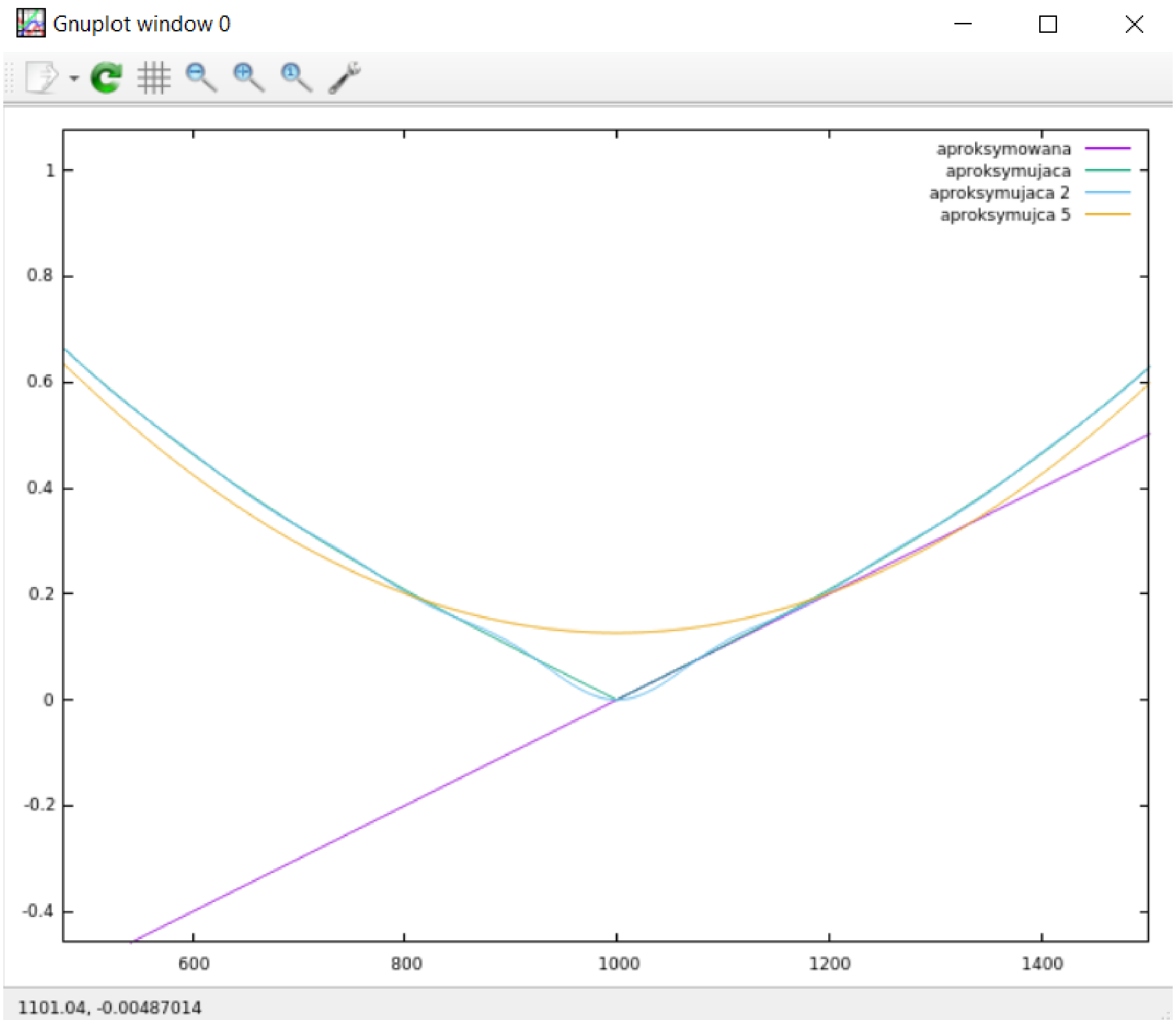
Implementujemy teraz drugi algorytm

```
double f(double x, void *p)
{
  (void)(p); /* avoid unused parameter warning */

  // return exp(x * x);
  return fabs(x + x * x * x);
}
```

Kompilujemy i uruchamiamy program a następnie przedstawiamy dane z użyciem gnuplot



Prezentuje się on następująco dla wielomianów 2 i 5:

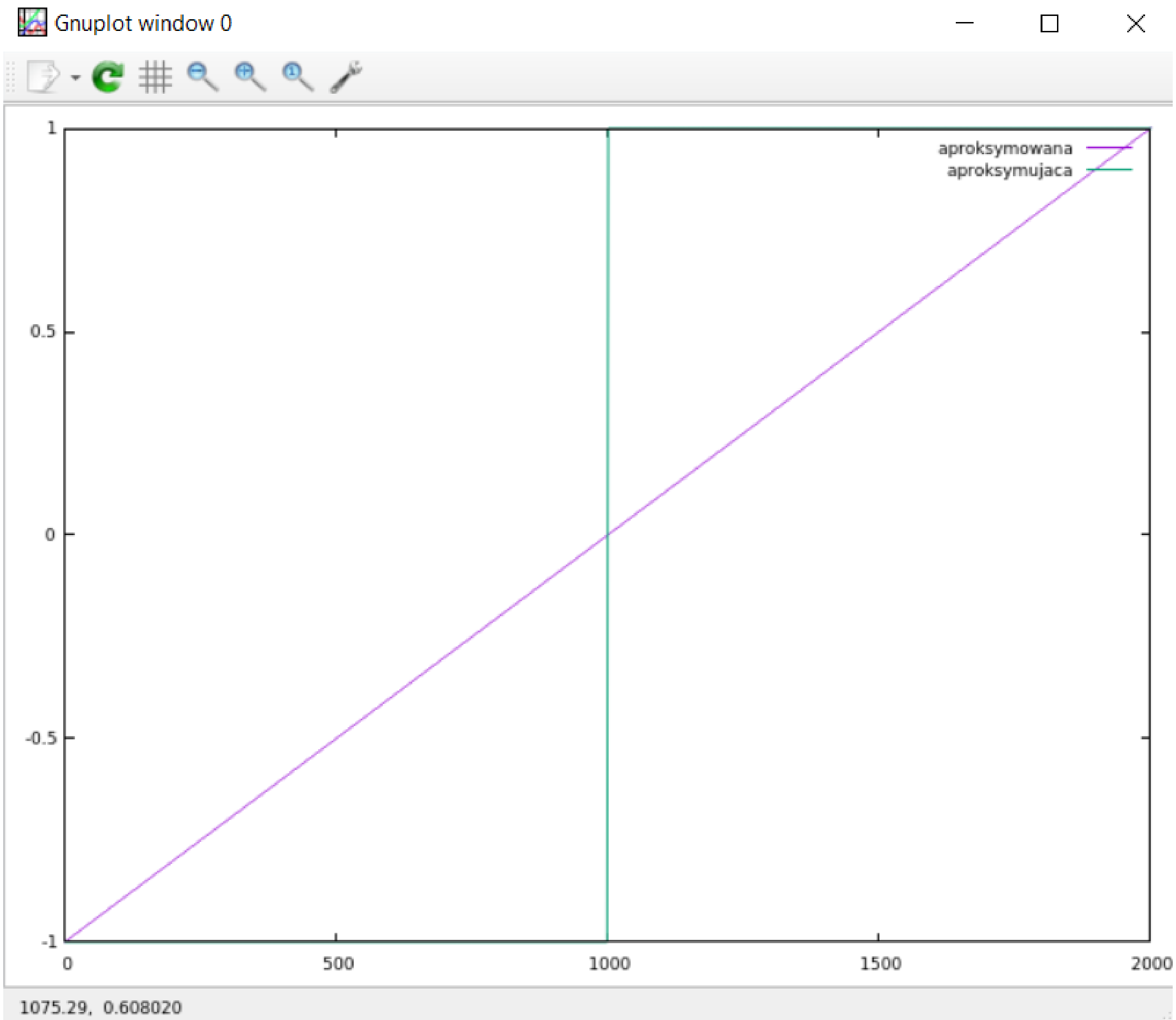


Jak widzimy tutaj wielomian aproksymacji ma duże znaczenie. Funkcje znacząco różnią się od siebie wartościami. Z definicji wynika, że dla k parzystego wielomian Czebyszewa k-tego stopnia jest parzysty, dla nieparzystego k – nieparzysty

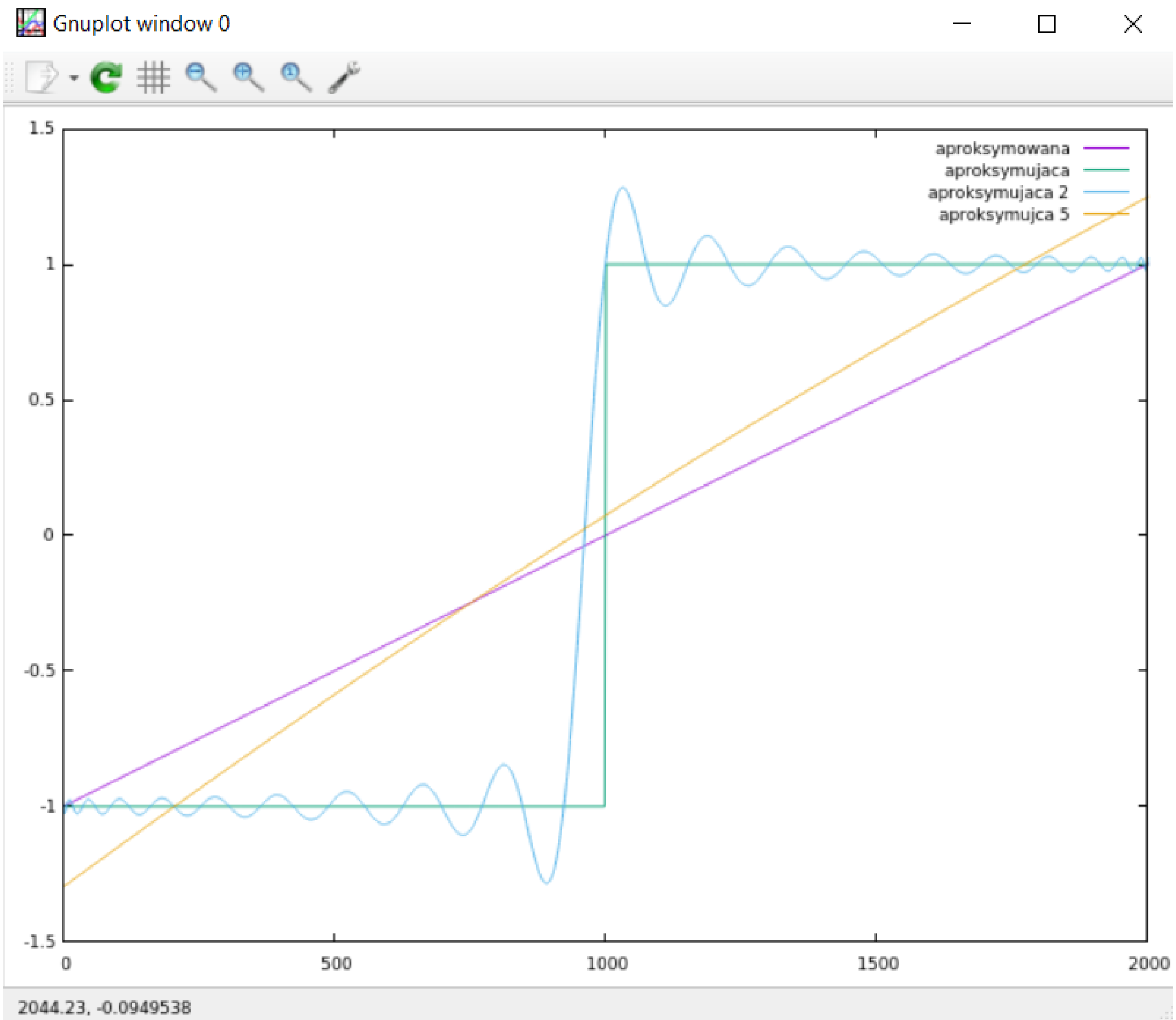
Teraz implementujemy trzecią funkcje

```
double f(double x, void *p)
{
  (void)(p); /* avoid unused parameter warning */

  // return exp(x * x);
  // return fabs(x + x * x * x);
  return (x > 0) - (x < 0); // sign(x)
}
```

Kompilujemy, uruchamiamy i przedstawiamy wykres:



Jak widać wszystkie trzy przykłady znacząco różnią się od siebie. Jest to spowodowane inną funkcją aproksymującą która zwraca wartości. Wykres przedstawia się następująco dla wielomianów 2 i 5



4. Wnioski

Przy obliczaniu wartości wielomianu interpolacyjnego w jednym lub kilku punktach problem wyboru postaci wzoru interpolacyjnego nie jest istotny. Rodzaj wybranego wzoru i rozmieszczenie węzłów ma wpływ jedynie na błąd obliczeń. O czasochłonności obliczeń decyduje liczba mnożeń i dzieleń.