Kacper Korta,

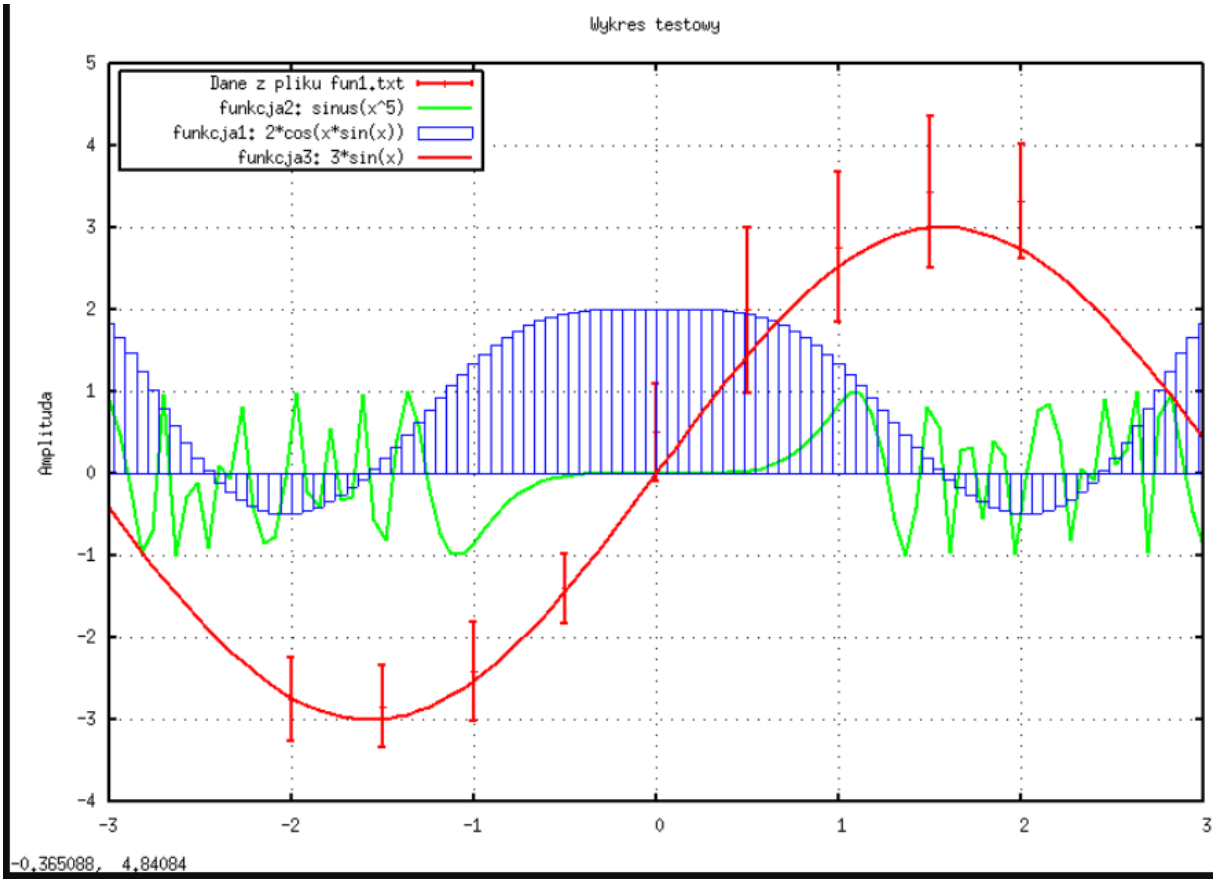
09.05.2023

Laboratorium 08

Wstęp do GSL i GNUPLOT

1. Treść zadań

1. Proszę skompilować i uruchomić program interpolacja.c. Korzystając z programu gnuplot narysować wykres. Narysować na jednym wykresie krzywe otrzymane różnymi metodami interpolacji.
2. Zadania gnuplot
   a) Przy pomocy gnuplot proszę narysować dane zgromadzone w pliku dane1.dat. Aby wykres był czytelny, jedna z osi musi mieć skale logarytmiczną. Proszę ustalić, która to os i narysować wykres.
   b) Proszę narysować wykres funkcji dwuwymiarowej, której punkty znajdują się w pliku dane2.dat. Proszę przeglądnąć plik i spróbować znaleźć w nim maksimum. Potem proszę zlokalizować maksimum wizualnie na wykresie. Proszę na wykresie zaznaczyć maksimum strzałką.
3. Proszę odtworzyć wykres znajdujący się na rysunku:



2. Rozwiązania

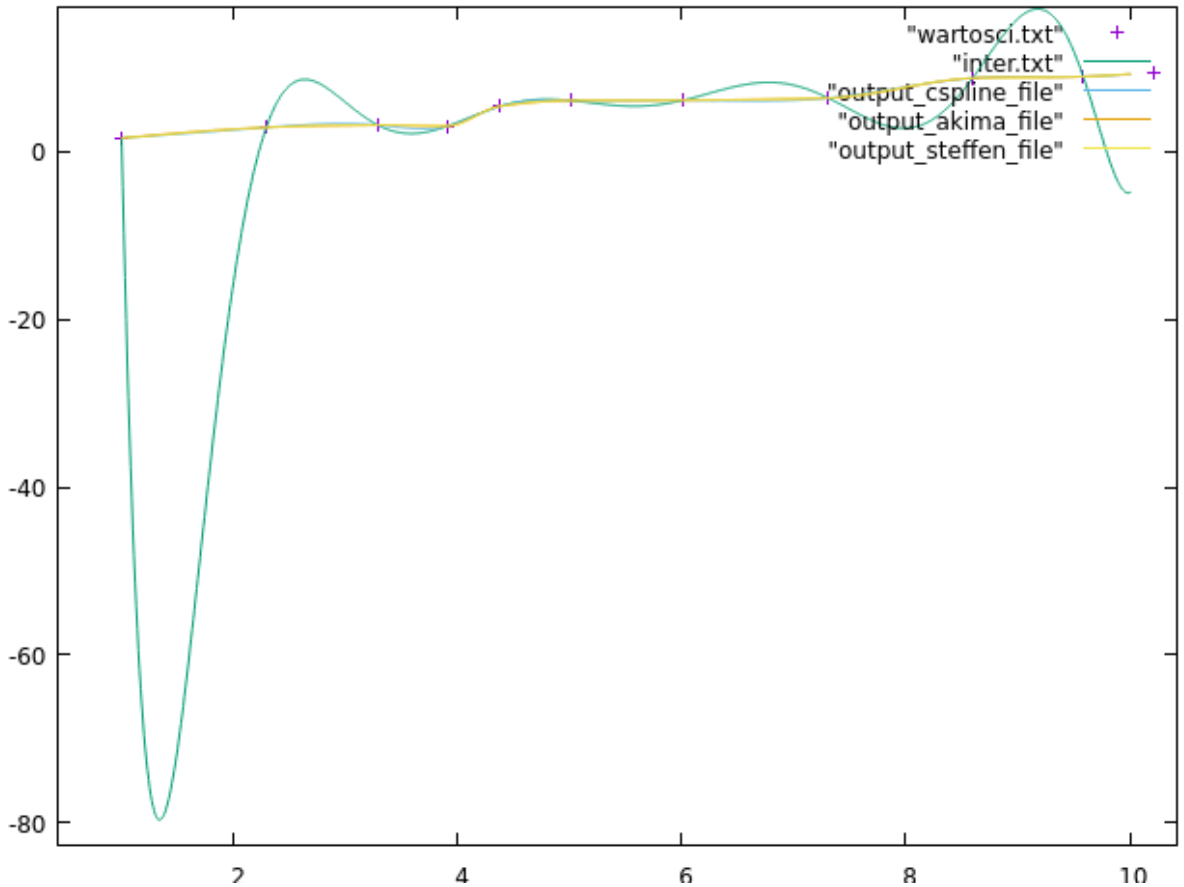
Zadanie 1.

W zadaniu tym skorzystałem z Interpolacji sześciennej (cubic interpolation), interpolacji Akimy oraz interpolacji Steffena.

Następnie w konsoli wpisuje takie polecenia:

gnuplot> set output "zadanie1.jpg"
gnuplot> plot "wartosci.txt", "inter.txt" with lines, "output_cspline_file" with lines, "output_akima_file" with lines, "output_steffen_file" with lines

Przy użyciu powyższych poleceń otrzymujemy taki wykres:



Krzywe znajdujące się na wykresie otrzymaliśmy przy użyciu interpolacji sześciennej (output_cspline_file) oraz interpolacji Akimy i Steffena.

Kod użyty do zadania

```
//#include <config.h>
#include <stdlib.h>
#include <stdio.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_spline.h>
#include <gsl/gsl_interp.h>
#include <math.h>
```

```c
static double fun(double x)
{
  return x + cos(x*x);
}

int main (void)
{
  const double a = 1.0;
  const double b = 10.0;
  const int steps = 10;
  double xi, yi, x[100], y[100], dx;
  FILE *input, *output, *output_cspline, *output_akima, *output_steffen;
  int i;

  input = fopen("wartosci.txt", "w");
  output = fopen("inter.txt", "w");
  output_cspline = fopen("output_cspline_file", "w");
  output_akima = fopen("output_akima_file", "w");
  output_steffen = fopen("output_steffen_file", "w");

  dx = (b-a) / (double) steps;

  for (i = 0; i <= steps; ++i) {
    x[i] = a + (double)i * dx + 0.5 * sin((double)i * dx);
      y[i] = fun(x[i]);
      fprintf (input,"%g %g\n", x[i], y[i]);
  }

    {
        gsl_interp_accel *acc = gsl_interp_accel_alloc();
        gsl_spline *spline_poly = gsl_spline_alloc(gsl_interp_polynomial,
steps + 1);
        gsl_spline *spline_cspline = gsl_spline_alloc(gsl_interp_cspline,
steps + 1);
        gsl_spline *spline_akima = gsl_spline_alloc(gsl_interp_akima, steps +
1);
        gsl_spline *spline_steffen = gsl_spline_alloc(gsl_interp_steffen,
steps + 1);
        gsl_spline_init(spline_poly, x, y, steps + 1);
        gsl_spline_init(spline_cspline, x, y, steps + 1);
        gsl_spline_init(spline_akima, x, y, steps + 1);
        gsl_spline_init(spline_steffen, x, y, steps + 1);

        for (xi = a; xi <= b; xi += 0.01) {
            yi = gsl_spline_eval(spline_poly, xi, acc);
            fprintf(output, "%g %g\n", xi, yi);
            yi = gsl_spline_eval(spline_cspline, xi, acc);
            fprintf(output_cspline, "%g %g\n", xi, yi);
```

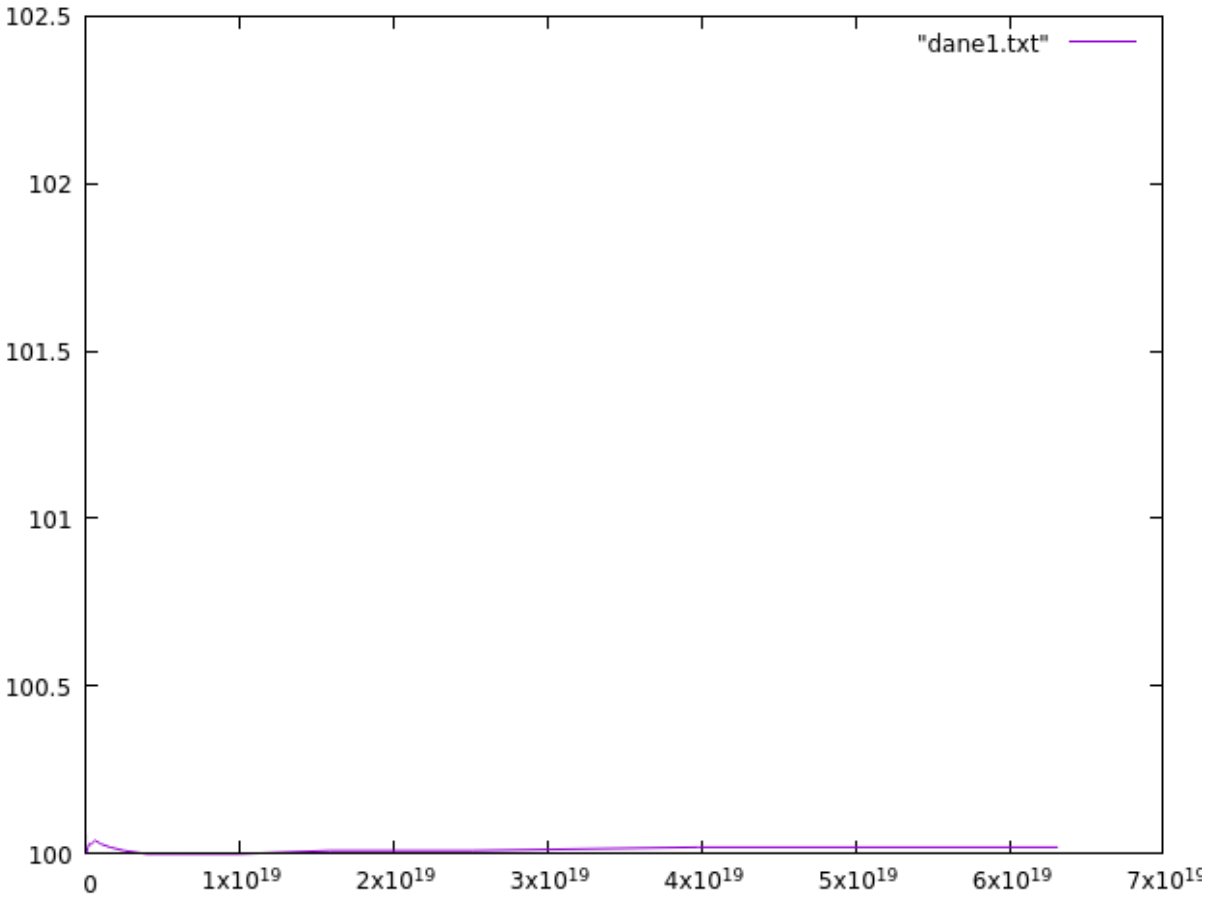
```
            yi = gsl_spline_eval(spline_akima, xi, acc);
            fprintf(output_akima, "%g %g\n", xi, yi);
            yi = gsl_spline_eval(spline_steffen, xi, acc);
            fprintf(output_steffen, "%g %g\n", xi, yi);
        }
        gsl_spline_free(spline_poly);
        gsl_spline_free(spline_cspline);
        gsl_spline_free(spline_akima);
        gsl_spline_free(spline_steffen);
        gsl_interp_accel_free(acc);
    }

  return 0;
}
```

Wnioski: Możemy zauważyć, że interpolacje (sześcienna, Steffena i Akimy) dają bardzo zbliżone wyniki. Natomiast interpolacja wielomianowa dużo różni się od pozostałych. Powodem takiego zachowania jest efekt Rungego.

Zadanie 2.

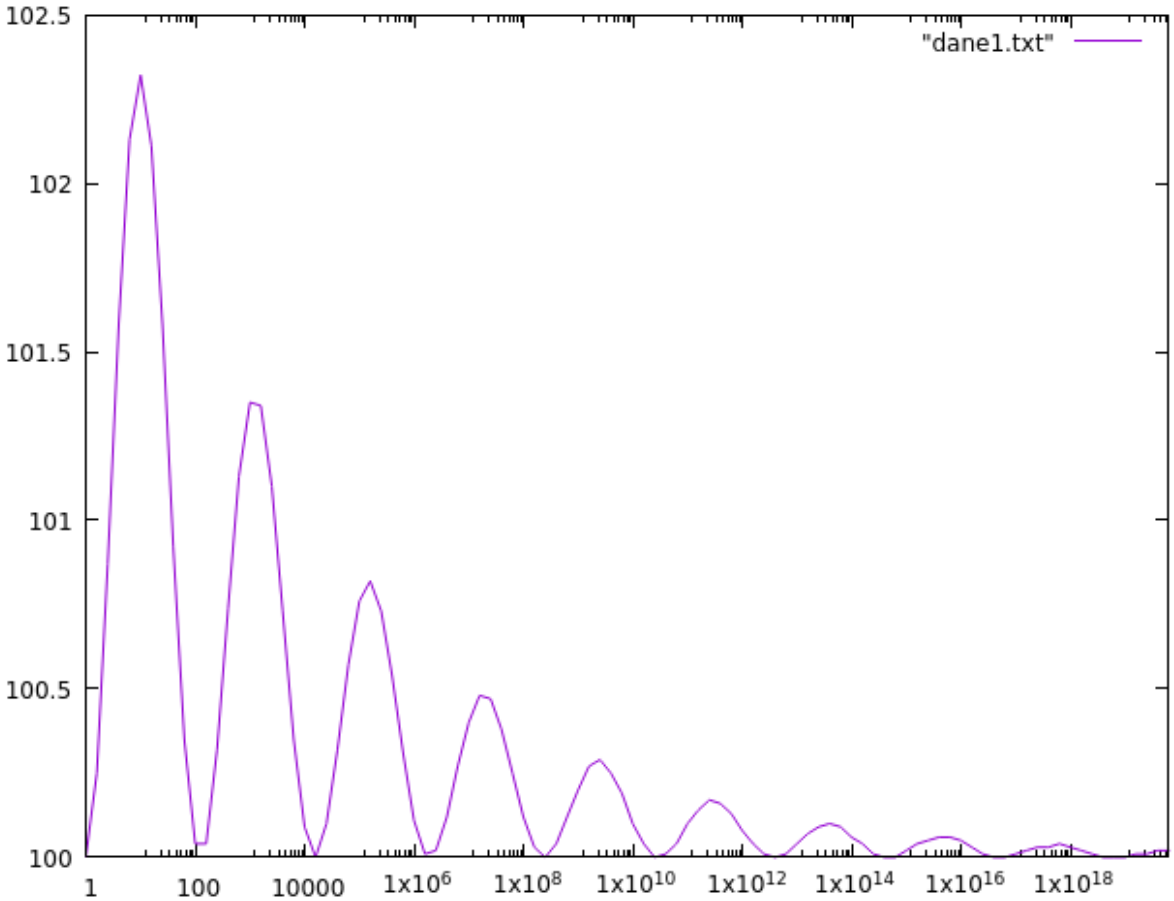
a) Wykres danych przed ustawieniem osi X w skali logarytmicznej przy użyciu poleceń **Gnuplot**> plot „dane1.txt” with lines



Następnie wygenerowano kolejny wykres przy zmianie osi X na skalę logarytmiczną.

Wykres uzyskałem przy użyciu poleceń:

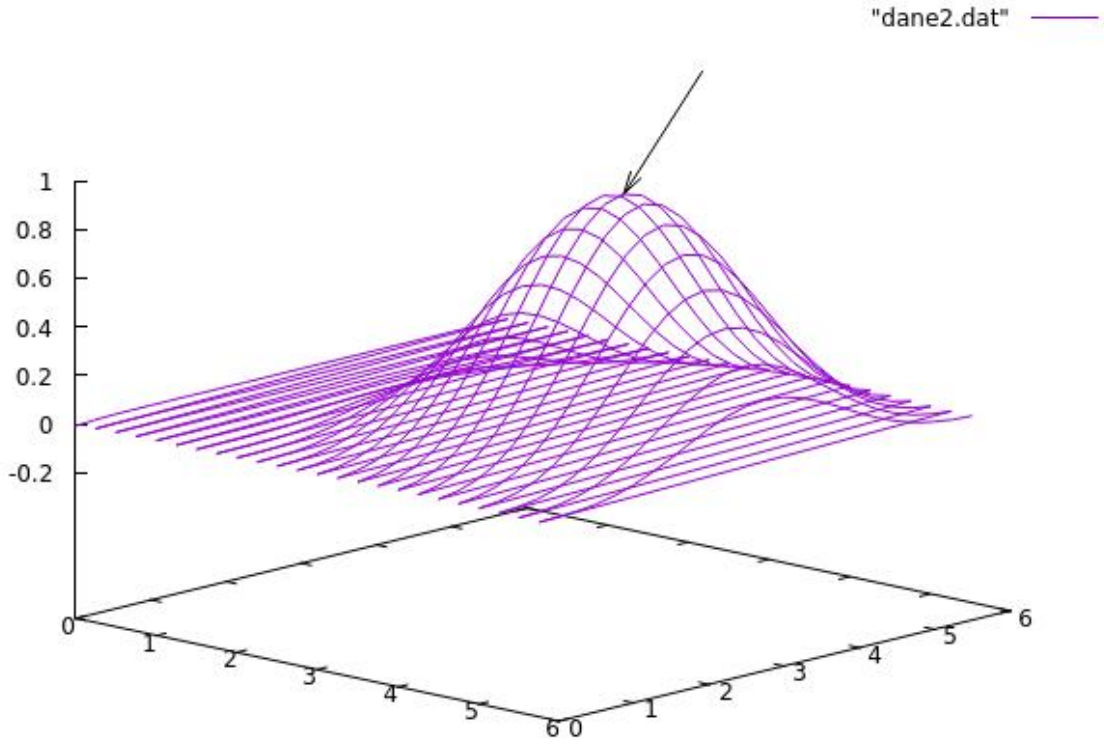
gnuplot> set logscale x
gnuplot> plot "dane1.txt" with lines



Wnioski: Można zauważyć, że zmiana skali na logarytmiczną znacząco poprawia jakość wykresu. W przypadku pierwszego wykresu (przed ustawieniem osi X w skali logarytmicznej) gęste punkty koncentrują się w okolicach małych wartości X, co powoduje, że linie łączące punkty są bardzo blisko siebie i ciężko zauważyć jakieś różnice w gęstości punktów dla większych wartości X.

b) Za pomocą polecenia:
gnuplot> set arrow from 4.5,3.5,1.5 to 4,3,1
gnuplot> splot "dane2.dat" with lines

Otrzymujemy taki wykres:

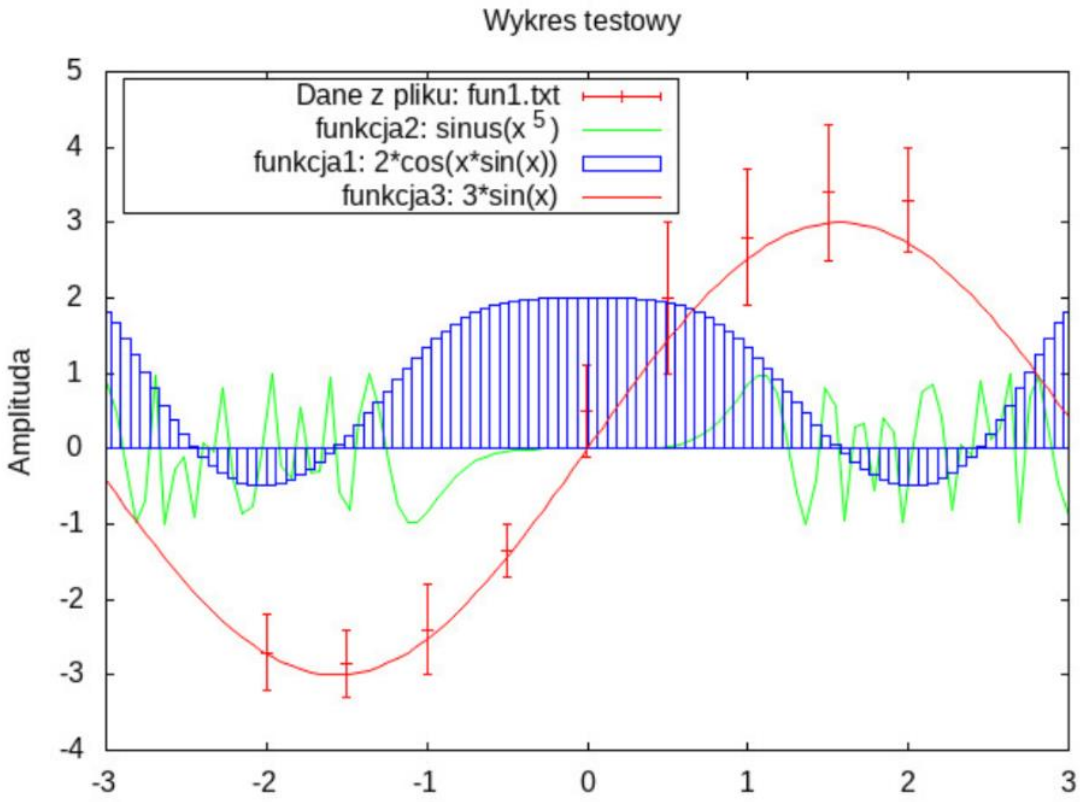


Mając wykres łatwo można znaleźć maksimum które leży w **punkcie (4, 3, 1).**
Natomiast bez wykresu nie byłoby to takie łatwe

Wnioski: Dzięki programowi GNUPLOT z łatwością możemy odczytywać wykresy, kiedy szukając tych punktów odczytując dane byłoby bardzo trudnym zadaniem

Zadanie 3.
Aby wygenerować wykres taki jak na rysunku musiałem wpisać następujące polecenia:

gnuplot> set output "zadanie3.png"
gnuplot> set xrange[-3:3]
gnuplot> set yrange[-4:5]
gnuplot> set title "Wykres_testowy"
gnuplot> set ylabel "Amplituda"
gnuplot> set boxwidth
gnuplot> set key left top box
gnuplot> plot "fun1.txt" lt rgb "red" with yerrorbars title "Dane z pliku: fun1.txt", sin(x**5) lt rgb "green" with lines title "funkcja2: sinus(x^5)", 2*cos(x*sin(x)) lt rgb "blue" with boxes title "funkcja1: 2*cos(x*sin(x))", 3*sin(x) lt rgb "red" with lines title "funkcja3: 3*sin(x)"



Wnioski: Dzięki programowi można łatwo tworzyć przejrzyste wykresy z legendami.

3. Bibliografia

Wykład
https://www.gnu.org/software/gsl/gsl.html
https://www.gnu.org/software/gsl/doc/html/
https://www.gnu.org/software/gsl/doc/html/interp.html