Laboratorium 08

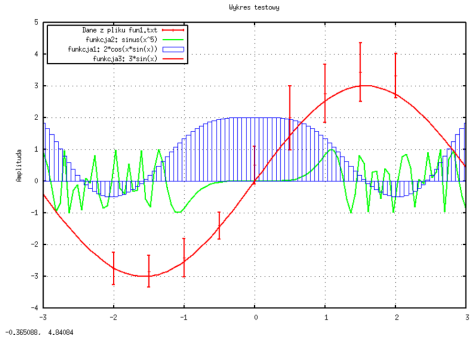
Wstęp do GSL i GNUPLOT

Adam Bińta, 05.05.2023

1 Treść zadań

Zadania

1. **Zadanie -gsl**
   (a) Proszę skompilować i uruchomić program interpolacja.c. Korzystając z programu gnuplot narysować wykres.
   (b) Narysować na jednym wykresie krzywe otrzymane różnymi metodami interpolacji.
2. **Zadania - gnuplot**
   (a) Przy pomocy gnuplot prosze narysowac dane zgromadzone w pliku dane1.dat. Aby wykres byl czytelny, jedna z osi musi miec skale logarytmiczna. Prosze ustalic, ktora to os i narysowac wykres.
   (b) Prosze narysowac wykres funkcji dwywymiarowej, ktorej punkty znajduja sie w pliku dane2.dat. Prosze przegladnac plik i sprobowac znalezc w nim maksimum. Potem prosze zlokalizowac maksimum wizualnie na wykresie. Prosze na wykresie zaznaczyc maksimum strzałką
   (c) Prosze odtworzyc wykres znajdujacy sie na rysunku



2 Rozwiązania zadań

1. **Zadanie -gsl (oba podpunkty)**
W kodzie źródłowym programu o nazwie interpolacja.c zastosowano cztery metody do interpolacji funkcji:

   - Interpolacja wielomianowa z wykorzystaniem metody Lagrange'a.
   - Interpolacja stosując podejście Steffena.
   - Interpolacja kubiczna sklejana, oparta na algorytmie de Boor'a.
   - Interpolacja według Akimy.

   Kod w pliku interpolacja.c

```c
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_spline.h>
#include <gsl/gsl_interp.h>

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;
    int i;

    FILE *input = fopen("data/wartosci.txt", "w");
    if (input == NULL) {
        perror("Nie udalo sie otworzyc pliku wartosci.txt");
        exit(EXIT_FAILURE);
    }
    FILE *output = fopen("data/inter.txt", "w");
    if (output == NULL) {
        perror("Nie udalo sie otworzyc pliku inter.txt");
        exit(EXIT_FAILURE);
    }
    FILE *output_steffen = fopen("data/inter_steffen.txt",
↪   "w");
    if (output_steffen == NULL) {
        perror("Nie udalo sie otworzyc pliku
↪   inter_steffen.txt");
        exit(EXIT_FAILURE);
```

```
    }
    FILE *output_cspline = fopen("data/inter_cspline.txt",
↪   "w");
    if (output_cspline == NULL) {
        perror("Nie udalo sie otworzyc pliku
↪   inter_cspline.txt");
        exit(EXIT_FAILURE);
    }
    FILE *output_akima = fopen("data/inter_akima.txt", "w");
    if (output_akima == NULL) {
        perror("Nie udalo sie otworzyc pliku
↪   inter_akima.txt");
        exit(EXIT_FAILURE);
    }
    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_cspline_method =
↪   gsl_spline_alloc(gsl_interp_cspline, steps + 1);
        gsl_spline *spline_steffen_method =
↪   gsl_spline_alloc(gsl_interp_steffen, steps + 1);
        gsl_spline *spline_akima_method =
↪   gsl_spline_alloc(gsl_interp_akima, steps + 1);
        gsl_spline *spline_poly_method =
↪   gsl_spline_alloc(gsl_interp_polynomial, steps + 1);

        gsl_spline_init(spline_cspline_method, x, y, steps +
↪   1);
        gsl_spline_init(spline_steffen_method, x, y, steps +
↪   1);
        gsl_spline_init(spline_akima_method, x, y, steps +
↪   1);
        gsl_spline_init(spline_poly_method, x, y, steps +
↪   1);

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

```
            yi = gsl_spline_eval(spline_steffen_method, xi,
↪   acc);
            fprintf(output_steffen, "%g %g\n", xi, yi);
            yi = gsl_spline_eval(spline_akima_method, xi,
↪   acc);
            fprintf(output_akima, "%g %g\n", xi, yi);
            yi = gsl_spline_eval(spline_poly_method, xi,
↪   acc);
            fprintf(output, "%g %g\n", xi, yi);
        }
        gsl_spline_free(spline_poly_method);
        gsl_spline_free(spline_cspline_method);
        gsl_spline_free(spline_akima_method);
        gsl_spline_free(spline_steffen_method);
        gsl_interp_accel_free(acc);
    }
    return 0;
}
```

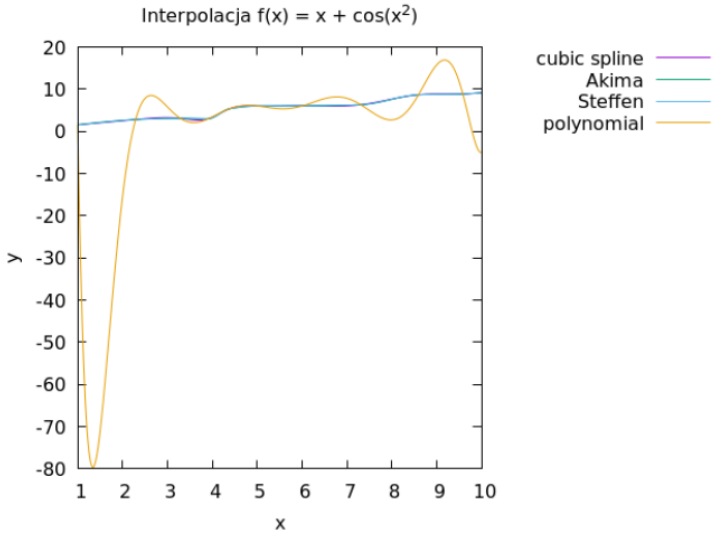
Uwaga! Powyższy plik znajduje się w folderze gsl. Należy Wpisać **make main** aby skompilować a następnie **make run** aby program załączyć. Aby wykonać wszystkie czynności na raz wystarczy wpisać **make** lub **make all**
Następnie wpisujemy w terminalu: **gnuplot**
Później wpisujemy następujące komendy:

```
set terminal pngcairo
set output 'interpol.png'
set title 'Interpolacja f(x) = x + cos(x^2)'
set xlabel 'x'
set ylabel 'y'
set key outside

file1 = 'data/inter_cspline.txt'
file2 = 'data/inter_akima.txt'
file3 = 'data/inter_steffen.txt'
file4 = 'data/inter.txt'

plot file1 using 1:2 with lines title 'cubic spline', \
     file2 using 1:2 with lines title 'Akima', \
     file3 using 1:2 with lines title 'Steffen', \
     file4 using 1:2 with lines title 'polynomial'
```

Wygenerowany wykres wygląda następująco:



Wnioski

(a) wszystkie zastosowane metody interpolacji, z wyjątkiem interpolacji wielomianowej, dają podobne rezultaty

(b) Najbardziej precyzyjne wyniki osiągnięto, korzystając z interpolacji kubicznej sklejanej.

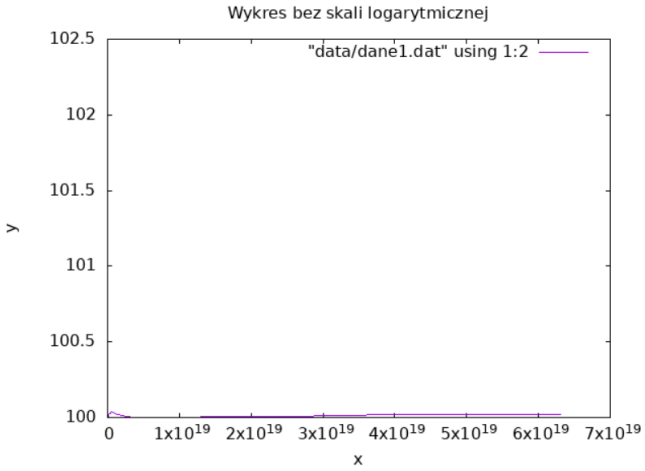
(c) Interpolacja wielomianowa jest czuła na liczbę węzłów interpolacyjnych (może skutkować skutkować wystąpieniem zjawiska Rungego)

2. **Zadania - gnuplot**

podpunkt a)

Najpierw upewniamy się, czy w folderze data znajduje się plik dane1.dat. i wpisujemy polecenie **gnuplot**.
Zaczynamy od wygenerowania wykresu bez skali logarytmicznej, wpisujemy następujące komendy :

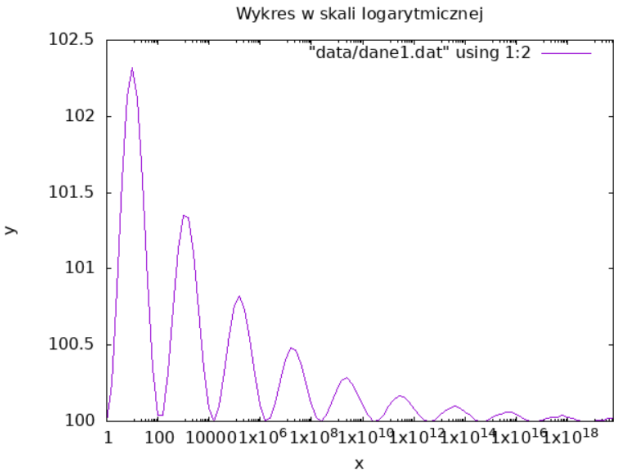
```
set terminal png
set output "zadanie2a-bezLog.png"
set title "Wykres bez skali logarytmicznej"
set xlabel "x"
set ylabel "y"
plot "data/dane1.dat" using 1:2 with lines
```



Aby wygeenrować wykres w skali logarytmicznej, przyjmijmy, że oś X będzie w skali logarytmicznej.
Wykonujemy następujące komendy:

```
set output "zadanie2aXlog.png"
set title "Wykres z osią X w skali logarytmicznej"
set xlabel "x"
set ylabel "y"
set logscale x
plot "data/dane1.dat" using 1:2 with lines
```

Powstały wykres:



Wnioski

- **Koncentracja punktów:** Na pierwszym wykresie (bez skali logarytmicznej na osi X) punkty są gęsto skupione w rejonie niskich wartości X, co sprawia, że linie łączące punkty są bardzo blisko siebie, utrudniając zauważenie różnic w gęstości punktów dla większych wartości X.
- **Czytelność wykresów:** Stosowanie skali logarytmicznej na osi X może zwiększyć czytelność wykresów, zwłaszcza gdy dane są nierównomiernie rozłożone wzdłuż tej osi.

podpunkt b)

(a) Upewniamy się, że w folderze data występuje plik **dane2.dat**

(b) musimy wcześniej wpisać polecenie **gnuplot**

(c) odczytujemy informacje statystyczne z pliku dane2.dat:
stats "data/dane2.dat"using 3 nooutput

(d) wskazujemy indeks wiersza, w którym znajduje się największa wartość w trzeciej kolumnie:

```
max_row_idx = system(sprintf("awk '{if ($3==%f) print
↪  NR}' data/dane2.dat", STATS_max))
```

(e) odczytujemy wartości x, y i z z wiersza o największej wartości f(x, y):

```
x_max = system(sprintf("awk 'NR==%s {print $1}'
↪  data/dane2.dat", max_row_idx))
y_max = system(sprintf("awk 'NR==%s {print $2}'
↪  data/dane2.dat", max_row_idx))
z_max = STATS_max
```

(f) teraz wyswietlimy wwartości x, y i z dla punktu, w którym funkcja osiąga wartość maksymalną:

```
print "Wartosc maksymalna funkcji z(x,y) znajduje sie w
↪  punkcie (", x_max, ", ", y_max, ", ", z_max, ")"
```

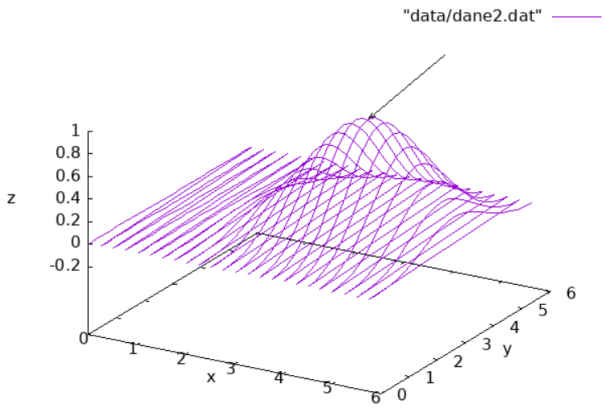
(g) Wynik wywolania:
Wartosc maksymalna funkcji z(x,y) znajduje sie w punkcie (4, 3, 1.0)

Z tego wynika, że maksymalna wartość wynosi 1.

Aby wygenerować wykres, należy zapisać następujące komendy:

```
set terminal png
set output "zadanie2b-wykres3d.png"
set xlabel "x"
set ylabel "y"
set zlabel "z"
set arrow from 5, 4, 1.5 to 4, 3, 1
splot "data/dane2.dat" with lines
```

Wynik:



Wnioski:

(a) Gnuplot to potężne i wszechstronne narzędzie do wizualizacji danych, które oferuje mnóstwo funkcji użytecznych do analizowania i prezentowania wyników.

(b) Umożliwia przeprowadzanie operacji na danych zapisanych w plikach, co pozwala na wykonanie bardziej zaawansowanych analiz

(c) Niezmiernie przydatne w wielu dziedzinach, takich jak analiza danych czy badania naukowe.

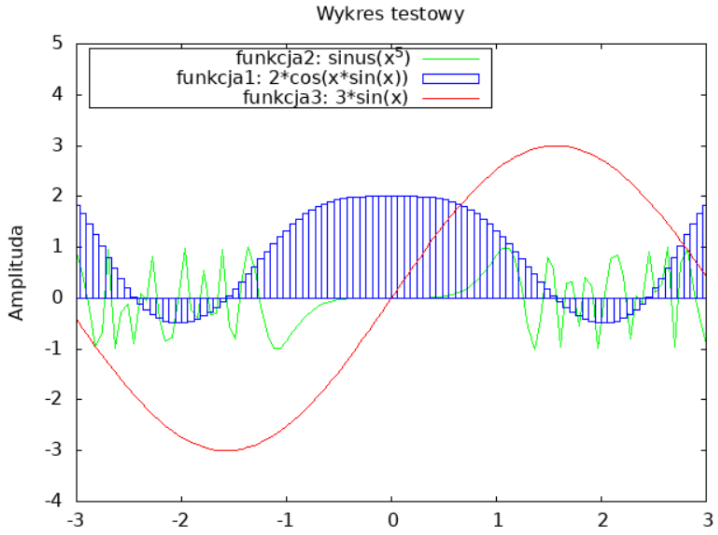
podpunkt c)

Aby wygenerować wykres przedstawiony na rysunku, wpisujemy poniższe komendy w Gnuplot:

```
set terminal png
set output "zadanie2c.png"
set xrange [-3:3]
```

```
set yrange [-4:5]
set title "Wykres testowy"
set ylabel "Amplituda"
set boxwidth
set key left top box
plot 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)"
```

Wynik:



Wnioski

(a) Zastosowanie różnych kolorów i legendy ułatwia odczytanie, której funkcji odpowiada dany element.

(b) Taki wykres może być przydatny przy analizie skomplikowanych funkcji matematycznych, gdzie istnieje potrzeba porównania wpływu poszczególnych składników na ogólny wynik.

3 Bibliografia

- Katarzyna Rycerz: "Wykład z przedmiotu Metody Obliczeniowe w Nauce i Technice"

- home.agh.edu.pl/ funika/mownit/lab8/
- www.gnu.org/software/gsl/doc/html/interp.html
- www.gnuplot.info