Laboratorium 8

Artem Buhera
GĆ01 135678

03.05.2021

Dany laboratorium polega na zaimplementowaniu programu do obliczania przybliżonych wartości pierwszych pochodnych funkcji $f(x)=\sin(x)$ w punktach końcowych i środkowym przedziału [0, π/2], czyli: 0, π/4 oraz π/2. Poza tym, sporządzimy wykresy zależności błędów bezwzględnych przybliżeń różnicowych od kroku siatki *h* dla zmiennych typu *float* oraz *double*.

Zastosujemy pięć metod:

- różnica progresywna dwupunktowa $f'(x_n) \approx \frac{f(x_{n+1}) - f(x_n)}{h}$
- różnica progresywna trzypunktowa $f'(x_n) \approx \frac{-\frac{3}{2}f(x_n) + 2f(x_{n+1}) - \frac{1}{2}f(x_{n+2})}{h}$
- różnica centralna dwupunktowa $f'(x_n) \approx \frac{f(x_{n+1}) - f(x_{n-1})}{2h}$
- różnica wsteczna dwupunktowa $f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{h}$
- różnica wsteczna trzypunktowa $f'(x_n) \approx \frac{\frac{1}{2}f(x_{n-2}) - 2f(x_{n-1}) + \frac{3}{2}f(x_n)}{h}$

Zaczynamy od *h* = 0.1 i w każdej iteracji zmniejszamy 1.2 razy aż *h* nie będzie na poziomie epsilonu maszynowego odpowiednio dla *float* i *double*.

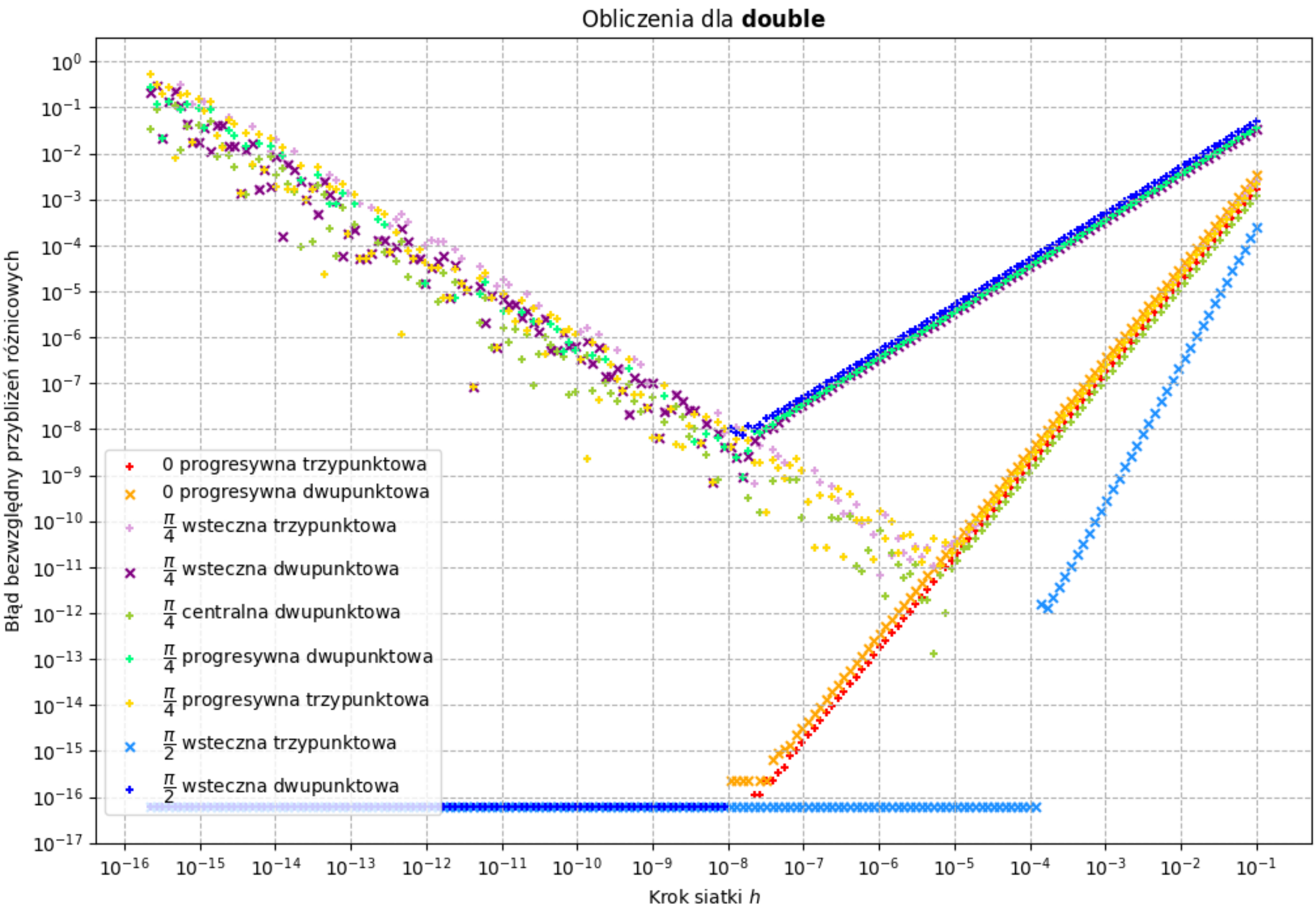
Kod programu:

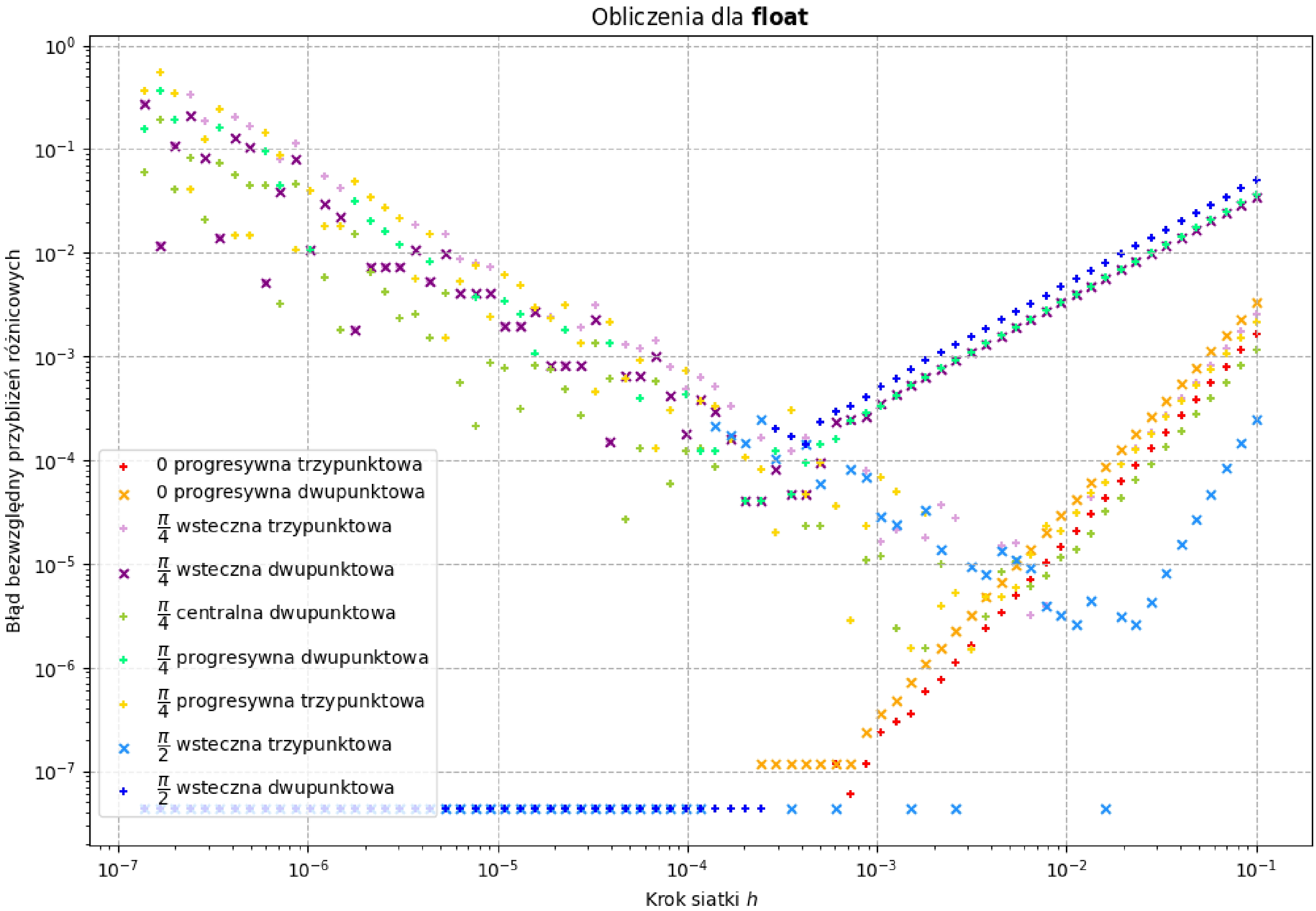
```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <float.h>
#include <fstream>

using namespace std;

template <class T>
double f(T x) {
    return sin(x);
}

template <class T>
double f_derivative(T x) {
    return cos(x);
}
```

```cpp
template <class T>
T backward_difference_3p(T xn_prev_prev, T xn_prev, T xn, T h) {
    return (static_cast<T>(0.5)*f(xn_prev_prev) - static_cast<T>(2.0)*f(xn_prev) +
            static_cast<T>(1.5)*f(xn)) / h;
}

template <class T>
T backward_difference_2p(T xn_prev, T xn, T h) {
    return (f(xn) - f(xn_prev)) / h;
}

template <class T>
T central_difference(T xn_prev, T xn_next, T h) {
    return (f(xn_next) - f(xn_prev)) / (static_cast<T>(2.0)*h);
}

template <class T>
T forward_difference_2p(T xn, T xn_next, T h) {
    return (f(xn_next) - f(xn)) / h;
}

template <class T>
T forward_difference_3p(T xn, T xn_next, T xn_next_next, T h) {
    return (-static_cast<T>(1.5)*f(xn) + static_cast<T>(2.0)*f(xn_next) -
             static_cast<T>(0.5)*f(xn_next_next)) / h;
}

template <class T>
void calculate(string filename) {
    T limit;
    if (is_same<T, double>::value)
        limit = DBL_EPSILON;
    else if (is_same<T, float>::value)
        limit = FLT_EPSILON;
    else
        exit(1);

    T a = static_cast<T>(0.0), b = static_cast<T>(M_PI_4), c = static_cast<T>(M_PI_2);
    T h = static_cast<T>(0.1);

    T derivative_a = f_derivative(a);
    T derivative_b = f_derivative(b);
    T derivative_c = f_derivative(c);

    ofstream out;
    out.open(filename);

    out << "h,0 progresywna trzypunktowa,0 progresywna dwupunktowa,"
       "$\\dfrac{\\pi}{4}$ wsteczna trzypunktowa,"
       "$\\dfrac{\\pi}{4}$ wsteczna dwupunktowa,"
       "$\\dfrac{\\pi}{4}$ centralna dwupunktowa,"
       "$\\dfrac{\\pi}{4}$ progresywna dwupunktowa,"
       "$\\dfrac{\\pi}{4}$ progresywna trzypunktowa,"
       "$\\dfrac{\\pi}{2}$ wsteczna trzypunktowa,"
       "$\\dfrac{\\pi}{2}$ wsteczna dwupunktowa" << endl;
```

```
    while (h > limit) {
        char delim = ',';
        out << h << delim
            << abs(derivative_a - forward_difference_2p(a, a+h, h)) << delim
            << abs(derivative_a - forward_difference_3p(a, a+h, a+h+h, h)) << delim

            << abs(derivative_b - backward_difference_3p(b-h-h, b-h, b, h)) << delim
            << abs(derivative_b - backward_difference_2p(b-h, b, h)) << delim
            << abs(derivative_b - central_difference(b-h, b+h, h)) << delim
            << abs(derivative_b - forward_difference_2p(b, b+h, h)) << delim
            << abs(derivative_b - forward_difference_3p(b, b+h, b+h+h, h)) << delim

            << abs(derivative_c - backward_difference_3p(c-h-h, c-h, c, h)) << delim
            << abs(derivative_c - backward_difference_2p(c-h, c, h)) << endl;

        h /= static_cast<T>(1.2);
    }

    out.close();
}


int main() {
    calculate<double>("../lab8/double.csv");
    calculate<float>("../lab8/float.csv");
}
```

Wykresy na podstawie otrzymanych danych:





Ewidentnie, obliczenia przy użyciu *double* pozwalają uzyskać dużo większą dokładność obliczeń przy użyciu odpowiednio mniejszego kroku *h*.

Z wykresów widać, że na początku zmniejszenie kroku *h* powoduje zwiększenie precyzji obliczeń, ale na poziomie około $10^{-3.9}$ i $10^{-7.9}$ dla odpowiednich różnic przy obliczeniach na *double* oraz $10^{-1.9}$ i $10^{-3.7}$ na *float* błąd bezwzględny liniowo zaczyna wzrastać - powodują to błędy maszynowe, które powstają podczas odejmowania coraz mniejszych liczb.

Na podstawie otrzymanych obliczeń, obliczyliśmy przybliżenia rzędów dokładności, które odpowiadają współczynnikom nachylenia części liniowej wykresów:

$$rząd_i \approx \frac{\log_{10} błąd_{i_5} - \log_{10} błąd_{i_0}}{\log_{10} h_5 - \log_{10} h_0}$$

Obliczenia rzędów teoretycznych dla różnic dwupunktowych w konkretnych punktach dla funkcji $f(x)=\sin(x)$:

Progresywna dwupunktowa w punkcie $x_n=0$:

$$\frac{f(x_{n+1})-f(x_n)}{h}=\frac{-f(x_n)+f(x_n)+f'(x_n)h+\frac{1}{2}f''(x_n)h^2+\frac{1}{6}f'''(x_n)h^3+\ldots}{h}=$$

$$=f'(x_n)+\frac{1}{2}f''(x_n)h^1+\frac{1}{6}f'''(x_n)h^2+\ldots=\cos 0-\frac{1}{2}\sin 0\cdot h^1-\frac{1}{6}\cos 0\cdot h^2+\ldots=$$

$$=1-\frac{1}{2}\cdot 0\cdot h^{1}-\frac{1}{6}\cdot 1\cdot h^2+\ldots=1-\frac{1}{6}h^{2}+\ldots$$

Wsteczna dwupunktowa w punkcie $x_n=\pi/4$:

$$\frac{f(x_n)-f(x_{n-1})}{h}=f'(x_n)-\frac{1}{2}f''(x_n)h^1+\frac{1}{6}f'''(x_n)h^2-\ldots=$$

$$\cos(\frac{pi}{4})+\frac{1}{2}\sin(\frac{pi}{4})h^1-\frac{1}{6}\sin(\frac{pi}{4})h^2+\ldots=\frac{1}{\sqrt{2}}+\frac{1}{2}\cdot\frac{1}{\sqrt{2}}\cdot h^{1}-\ldots$$

Centralna dwupunktowa w punkcie $x_n=\pi/4$:

$$\frac{f(x_{n+1})-f(x_{n-1})}{2h}=$$

$$=\frac{f(x_n)-f(x_n)+f'(x_n)h+f'(x_n)h+\frac{1}{2}f''(x_n)h^2-\frac{1}{2}f''(x_n)h^2+\frac{1}{6}f'''(x_n)h^3+\frac{1}{6}f'''(x_n)h^3+\ldots}{2h}=$$

$$=f'(x_n)+\frac{1}{6}f'''(x_n)h^2+\ldots=\cos(\frac{\pi}{4})-\frac{1}{6}\cos(\frac{\pi}{4})h^2+\ldots=\frac{1}{\sqrt{2}}-\frac{1}{2\sqrt{6}}h^{2}+\ldots$$

Progresywna dwupunktowa w punkcie $x_n=\pi/4$:

$$\frac{f(x_{n+1})-f(x_n)}{h}=f'(x_n)+\frac{1}{2}f''(x_n)h^1+\frac{1}{6}f'''(x_n)h^2+\ldots=$$

$$=\cos(\frac{\pi}{4})-\frac{1}{2}\sin(\frac{\pi}{4})\cdot h^1-\frac{1}{6}\cos(\frac{\pi}{4})\cdot h^2+\ldots=\frac{1}{\sqrt{2}}-\frac{1}{2}\cdot\frac{1}{\sqrt{2}}\cdot h^1-\frac{1}{6}\cdot\frac{1}{\sqrt{2}}\cdot h^2+\ldots=\frac{1}{\sqrt{2}}-\frac{1}{2}\sqrt{2}h^{1}-\ldots$$

Wsteczna dwupunktowa w punkcie $x_n=\pi/2$:

$$\frac{f(x_n)-f(x_{n-1})}{h}=\frac{f(x_n)-(f(x_n)-f'(x_n)h+\frac{1}{2}f''(x_n)h^2-\frac{1}{6}f'''(x_n)h^3+\ldots)}{h}=$$

$$=f'(x_n)-\frac{1}{2}f''(x_n)h^1+\frac{1}{6}f'''(x_n)h^2-\ldots=\cos(\frac{pi}{2})+\frac{1}{2}\sin(\frac{pi}{2})h^1-\frac{1}{6}\sin(\frac{pi}{2})h^2+\ldots=$$

$$=0+\frac{1}{2}\cdot 1\cdot h^1-\frac{1}{6}\cdot 1\cdot h^2+\ldots=\frac{1}{2}h^{1}-\ldots$$

Różnica	Rząd *double*	Rząd *float*	Rząd teoretyczny
0 progresywna trzypunktowa	2.000	1.999	2
0 progresywna dwupunktowa	1.997	1.997	1
π/4 wsteczna trzypunktowa	2.044	2.043	2
π/4 wsteczna dwupunktowa	0.977	0.977	1
π/4 centralna dwupunktowa	2.000	2.001	2
π/4 progresywna dwupunktowa	1.021	1.021	1
π/4 progresywna trzypunktowa	1.945	1.945	2
π/2 wsteczna trzypunktowa	2.998	3.040	2
π/2 wsteczna dwupunktowa	0.999	0.999	1

Rozbieżność w oczekiwanych rzędach dla różnicy progresywnej dwupunktowej w punkcie 0 oraz wstecznej trzypunktowej w π/2 możemy wytłumaczyć tym, że pochodne funkcji *sin(x)* w odpowiednich punktach są równe zeru, co powoduje "wyzerowanie" *h* o oczekiwanym stopniu:
$\sin(0)=0, \cos\left(\frac{\pi}{2}\right)=0$

Dlatego właściwym rzędem staje się następny - co widać przy obliczeniach rzędu dokładności dla progresywnej różnicy dwupunktowej powyżej.

Analiza otrzymanych wyników (wykres oraz obliczanie rzędów dokładności) w środowisku Python przy użyciu bibliotek matplotlib oraz pandas:

```
import matplotlib.pyplot as plt
from matplotlib import ticker
import pandas as pd
import math


def rzad(df, i):
    calc = df.columns[i], ((math.log10(df[df.columns[i]][0])) -
math.log10(df[df.columns[i]][5])) / (
            math.log10(df[df.columns[0]][0]) - math.log10(df[df.columns[0]][5]))
    print(f'{calc[0]} {calc[1]:.3f}')


def make_plot(name):
    df = pd.read_csv(f'{name}.csv')

    fig, ax = plt.subplots(figsize=(12, 8), dpi=100)

    column_colors = {
        df.columns[1]: ('red', '+'),
        df.columns[2]: ('orange', 'x'),
        df.columns[3]: ('plum', '+'),
        df.columns[4]: ('purple', 'x'),
        df.columns[5]: ('yellowgreen', '+'),
        df.columns[6]: ('springgreen', '+'),
        df.columns[7]: ('gold', '+'),
        df.columns[8]: ('dodgerblue', 'x'),
        df.columns[9]: ('blue', '+')
    }

    for column, (color, marker) in column_colors.items():
        df.plot(kind='scatter', x='h', y=column, color=color, marker=marker, s=25,
label=column, ax=ax)
    ax.set_yscale('log')
    ax.set_xscale('log')

    ax.yaxis.set_major_locator(ticker.LogLocator(numticks=20))
    ax.xaxis.set_major_locator(ticker.LogLocator(numticks=20))

    ax.set_title(rf'Obliczenia dla $\bf{name}$')
    ax.set_xlabel(r'Krok siatki $h$')
    ax.set_ylabel(r'Błąd bezwzględny przybliżeń różnicowych')

    ax.grid(linestyle='--')

    plt.legend(loc='best', bbox_to_anchor=(0., 0., 0.5, 0.5))

    for i in range(1, 10):
        rzad(df, i)

    plt.show()


print('double')
make_plot('double')

print('\nfloat')
make_plot('float')
```