Artur Gęsiarz,
Kwiecień 13, 2024

Laboratorium nr 6
MOwNiT – Kwadratury

1. Treść zadania

1.1. Zadanie pierwsze

Wiadomo, że

$$\int_0^1 \frac{4}{1+x^2}\,dx = \pi$$

Powyższą równość można wykorzystać do obliczenia przybliżonej wartości π po- przez całkowanie numeryczne.

Obliczę wartość powyższej całki, korzystając ze złożonych kwadratur otwartej prostokątów (ang. mid-point rule), trapezów i Simpsona. Na przedziale całkowania rozmieszczę $2^m + 1$ równoodległych węzłów. W kolejnych próbach m wzrasta o 1, tzn. między każde dwa sąsiednie węzły dodawany jest nowy węzeł, a ich zagęszczenie zwiększa się dwukrotnie. Przyjmię zakres wartości m od 1 do 25.

Dla każdej metody narysuj wykres wartości bezwzględnej błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej, n + 1 (gdzie n = 1/h, z krokiem h). Wyniki przedstaw na wspólnym wykresie, używając skali logarytmicznej na obu osiach.

Czy istnieje pewna wartość, poniżej której zmniejszanie kroku h nie zmniejsza już błędu kwadratury? Porównaj wartość h_min, odpowiadającą minimum wartości bezwzględnej błędu względnego, z wartością wyznaczoną w laboratorium 1.

Dla każdej z użytych metod porównaj empiryczny rząd zbieżności z rząd zbieżności przewidywanym przez teorię. Aby wyniki miały sens, do obliczenia rzędu empirycznego użyj wartości h z zakresu, w którym błąd metody przeważa nad błędem numerycznym.

1.2. Zadanie drugie

Obliczę wartość całki

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

metodą Gaussa-Legendre'a. Narysuję wykres wartości bezwzględnej błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej, n + 1. Przyjmę na tyle duży zakres n, aby wykryć, kiedy błąd numeryczny zaczyna przeważać nad błędem metody.

2. Rozwiązanie zadań

2.1. Zadanie pierwsze

2.1.1. Funkcja do całkowania

```
def function_to_integrate(x):
    return 4 / (1 + x ** 2)
```

2.1.2. Funkcja obliczająca wartość pi za pomocą całki

```
def calculate_pi_integral(method, x, y):
    integral_value = method(y, x)
    return integral_value
```

2.1.3. Funkcja obliczająca błąd względny

```
def relative_error(excat, approx):
    return np.abs((excat - approx) / excat)
```

2.1.4. Lista do przechowyuwania błędów względnych dla każdej z metod

```
errors_trapezoidal = []
errors_simpson = []
```

2.1.5. Przedział całkowania

```
a = 0
b = 1
```

2.1.6. Zakres wartości m

```
m_values = np.arange(1, 26)
```

2.1.7. Pętla po wartościach m

```
for m in m_values:
    # Generowanie wezlów
    x = np.linspace(a, b, 2 ** m + 1)
    y = function_to_integrate(x)

    exact_value = np.pi

    integral_trapezoidal = calculate_pi_integral(trapz, x, y)
    integral_simpson = calculate_pi_integral(simps, x, y)

    error_trapezoidal = relative_error(exact_value, integral_trapezoidal)
    error_simpson = relative_error(exact_value, integral_simpson)

    errors_simpson.append(error_simpson)
    errors_trapezoidal.append(error_trapezoidal)
```

2.1.8. Tworzenie wykresu

```
plt.figure(figsize=(10,6))
plt.plot(m_values, errors_trapezoidal, label='Metoda trapezow', marker='o')
plt.plot(m_values, errors_simpson, label='Metoda Simpsona', marker='o',
 color='orange')
plt.xscale('log')
plt.yscale('log')
plt.title('Blad wzgledny w zależności od liczby ewaluacji funkcji')
plt.xlabel('Liczba ewaluacji funkcji')
plt.ylabel('Blad wzgledny')
plt.legend()
plt.grid(True)
plt.show()
```

2.1.9. Wyświetlanie danych

```
m = 1
for error_trapezional in errors_trapezoidal:
    print(f"Error trapezional for {m} -> equals: {error_trapezional}")
    m += 1

m = 1
for error_simpson in errors_simpson:
    print(f"Error Simpson for {m} -> equals: {error_simpson}")
    m += 1
```

2.1.10. Obliczanie h_min

```
def calculate_hmin(method):
    h = 1.0
    previous_error = 1.0

    while True:
        x = np.linspace(0, 1, int(1 / h) + 1)
        y = function_to_integrate(x)

        exact_value = np.pi
        integral_value = method(y, x)
        error = relative_error(exact_value, integral_value)

        if error >= previous_error or np.isnan(error):
            break

        previous_error = error
        h /= 2

    return h
```

2.1.11. Obliczanie wartości poniżej której zmniejszenie h nie zmniejsza już błędu kwadratury dla metody trapezów

```
h_min_trapezoidal = calculate_hmin(trapz)
print("H_min dla metody trapezow wynosi:",h_min_trapezoidal)
```

2.1.12. Obliczanie wartości poniżej której zmniejszenie h nie zmniejsza już błędu kwadratury dla metody Simpsona

```
h_min_Simpson = calculate_hmin(simps)
print("H_min dla metody Simpsona wynosi:",h_min_Simpson)
```

2.1.13. Obliczenie błędu numerycznego dla danej metody i wartości h

```
def calculate_error(method, h):
    x = np.linspace(0, 1, int(1 / h) + 1)
    y = function_to_integrate(x)

    exact_value = np.pi
    integral_value = method(y, x)

    error = np.abs((exact_value - integral_value) / exact_value)

    return error
```

2.1.14. Obliczanie rzędu zbieżności

```
def calculate_convergence_order(errors, hs):
    p_values = []

    for i in range(len(errors) - 1):
        if errors[i] == 0 or errors[i+1] == 0:
            continue
        p = np.log(errors[i+1] / errors[i]) / np.log(hs[i+1] / hs[i])
        p_values.append(p)

    return p_values
```

2.1.15. Zakres wartości h

```
hs = np.logspace(-5, -1, 100)
```

2.1.16. Obliczanie błędów numerycznech dla każdej metody

```
errors_trapezoidal_empi = [calculate_error(trapz, h) for h in hs]
errors_simpson_empi = [calculate_error(simps, h) for h in hs]
```

2.1.17. Obliczanie rzędu zbieżności dla każdej metody

```
p_values_trapezoidal = calculate_convergence_order(errors_trapezoidal_empi, hs)
p_values_Simpson = calculate_convergence_order(errors_simpson_empi, hs)
```

2.1.18. Wyświetlanie wyników

```
print("Rząd zbieżności dla metody trapezów: ",np.mean(p_values_trapezoidal))
print("Rząd zbieżności dla metody Simpsona: ",np.mean(p_values_Simpson))
```

2.2. Zadanie drugie

2.2.1. Implementacja metody Gaussa-Lengendre'a

```
def gauss_legendre_integration(n):
    nodes, weights = roots_legendre(n)

    x = 0.5 * (nodes + 1)
    w = 0.5 * weights

    integral_value = np.sum(w * function_to_integrate(x))

    return integral_value
```

2.2.2. Obliczanie wartości dokładniej całki

```
exact_value = np.pi
```

2.2.3. Lista przechowująca wartości bezwzględnych błędów względnych

```
errors = []
```

2.2.4. Obliczanie wartości całki i błędów dla różnych wartości n

```
n_values = range(1, 100)
```

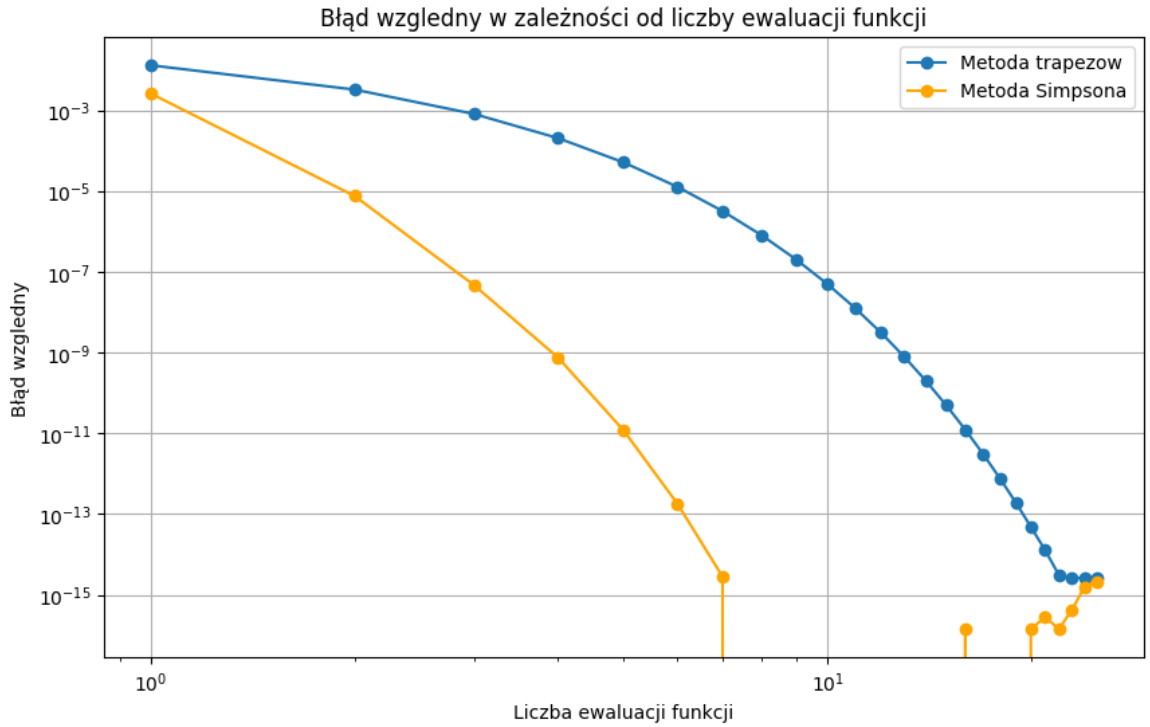
```
for n in n_values:
    integral_value = gauss_legendre_integration(n)
    error = np.abs((exact_value - integral_value) / exact_value)
    errors.append(error)
```

2.2.5. Narysowanie wykresu

```
plt.figure(figsize=(10,6))
plt.plot(np.array(n_values) + 1, errors, label='Metoda Gaussa-Legendre'a',marker='o'
plt.xscale('log')
plt.yscale('log')
plt.title('Błąd względny w zależności od liczby ewaluacji funkcji')
plt.xlabel('Liczba ewaluacji funkcji')
plt.ylabel('Bezwzględny błąd względny')
plt.legend()
plt.grid(True)
plt.show()
```

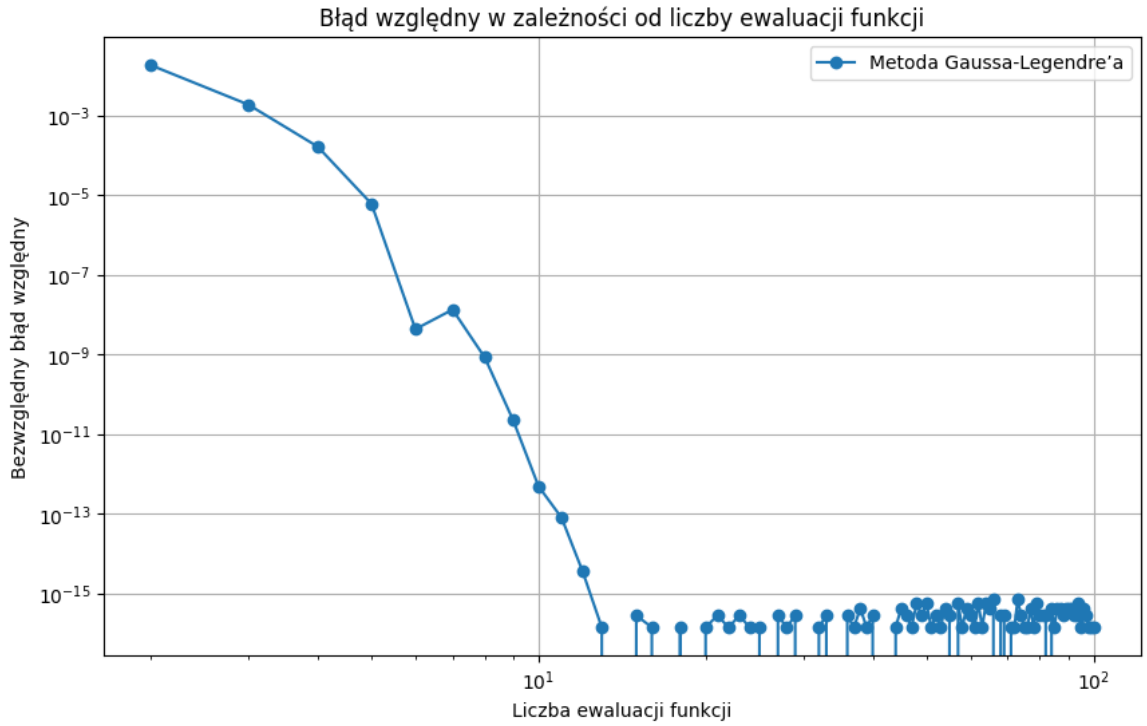
3. Wykresy

3.1. Wykres błędu względnego w zależności od liczby ewaluacji dla metody trapezów oraz metody Simpsona



Wykres 1. Błąd względny w zależności od liczby ewaluacji dla metody trapezów oraz metody Simpsona

3.2. Wykres błędu względnego w zależności od liczby ewaluacji dla metody Gaussa-Legendre'a



Wykres 2. Błąd względny w zależności od liczby ewaluacji dla metody Gaussa-Legendre'a

4. Tabele

4.1. Tabela błędów względnych metody trapezów

Wartość m	Wartość błędu względnego metody trapezów
1	0.0132
2	0.0033
3	0.0008
4	0.0002
5	$5.1808 * 10^{-5}$
6	$1.2952 * 10^{-5}$
7	$3.2380 * 10^{-6}$
8	$8.0950 * 10^{-7}$
9	$2.0237 * 10^{-7}$
10	$5.0593 * 10^{-8}$
11	$1.2648 * 10^{-8}$
12	$3.16 * 10^{-9}$
13	$7.90 * 10^{-10}$
14	$1.97 * 10^{-10}$
15	$4.94 * 10^{-11}$
16	$1.235 * 10^{-11}$
17	$3.08 * 10^{-13}$
18	$7.719 * 10^{-13}$
19	$1.936 * 10^{-14}$
20	$4.834 * 10^{-14}$
21	$1.272 * 10^{-15}$
22	$3.109 * 10^{-15}$
23	$2.544 * 10^{-15}$
24	$2.685 * 10^{-15}$
25	$2.685 * 10^{-15}$

Tabela 1. Tabela błędów względnych metody trapezów

4.2. Tabela błędów względnych metody Simpsona

Wartość m	Wartość błędu względnego metody trapezów
1	0.0026
2	$7.647 * 10^{-6}$
3	$4.810 * 10^{-8}$
4	$7.527 * 10^{-10}$
5	$1.17 * 10^{-11}$
6	$1.83 * 10^{-13}$
7	$2.82 * 10^{-15}$
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	$1.41 * 10^{-16}$
17	0
18	0
19	0
20	$1.41 * 10^{-16}$
21	$2.82 * 10^{-16}$
22	$1.41 * 10^{-16}$
23	$4.24 * 10^{-15}$
24	$1.41 * 10^{-16}$
25	$4.24 * 10^{-16}$

Tabela 2. Tabela błędów względnych metody Simpsona

4.3. Tabela rzędu zbieżności metody trapezów oraz Simpsona

Metoda	Rząd zbieżności
Trapezów	1.999
Simpsona	3.07

Tabela 3. Tabela rzędu zbieżności metody trapezów oraz Simpsona

4.4. Tabela wartości, poniżej której zmniejszanie kroku h nie zmniejsza już błędu kwadratury dla metody trapezów oraz Simpsona

Metoda	H_min
Trapezów	$5.9604 * 10^{-8}$
Simpsona	0.001

Tabela 4. Tabela wartości h_min dla metody trapezów oraz Simpsona

5. Wnioski

   Empiryczny rząd zbieżności dla metody trapezów odczytana z **Tabela 3.** wynosi 1.99, co jest bardzo blisko oczekiwanego rzędu zbieżności równego 2. Wartość ta potwierdza teoretyczne założenia dotyczące rzędu zbieżności tej metody.

   Empiryczny rząd zbieżności dla metody Simpsona odczytana z z **Tabela 3.** wynosi 3.07. Wynik ten wydaje się nieco niższy od oczekiwanego rzędu zbieżności równego 4. Może to wynikać z niedokładności obliczeń numerycznych lub innych czynników wpływających na dokładność wyniku.

   Zarówno metoda trapezów, jak i metoda Simpsona są skutecznymi metodami całkowania numerycznego. Empiryczne rządy zbieżności dla obu metod są zgodne z teoretycznymi oczekiwaniami, co potwierdza ich poprawność i skuteczność. Minimalne wartości kroku h_min dla obu metod są na akceptowalnym poziomie, co oznacza, że metody te są w stanie osiągnąć wysoką dokładność wyników dla dostatecznie małych wartości kroku h.

   Wykres wartości bezwzględnej błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej pokazuje, że błąd względny maleje wraz ze wzrostem liczby węzłów, co sugeruje, że metoda Gaussa-Legendre'a jest skuteczną metodą całkowania numerycznego.

6. Bibliografia
   Wykład MOwNiT - prowadzony przez dr. Inż. K. Rycerz
   Prezentacje – dr. Inż. M. Kuta

7. Dodatkowe informacje
   Rozwiązanie obu zadań znajduje się odpowiednio w plikach ex1.ipynb oraz ex2.ipynb.