lab6

May 16, 2024

1 Laboratorium 06

1.1 Kwadratury

Iga Antonik, Helena Szczepanowska

2 Zadanie 1

Wiadomo, że

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

Powyższą równość można wykorzystać do obliczenia przybliżonej wartości poprzez całkowanie numeryczne. - (a) Oblicz wartość powyższej całki, korzystając ze złożonych kwadratur otwatej prostokątów (ang. mid-point rule), trapezów i Simpsona. Można wykorzystać funkcje integrate.trapz i integrate.simps z biblioteki scipy. Na przedziale całkowania rozmieść 2m+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. Przyjmij 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.

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

2.1 Rozwiązanie

2.1.1 Biblioteki

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import trapz, simps
```

2.1.2 Definicja funkcji podcałkowej

[ ]:
```
def f(x):
    return 4 / (1 + x**2)
```

2.1.3 Funkcja obliczająca wartość błędu względnego

[ ]:
```
def relative_error(exact, approx):
    return np.abs((exact - approx) / exact)
```

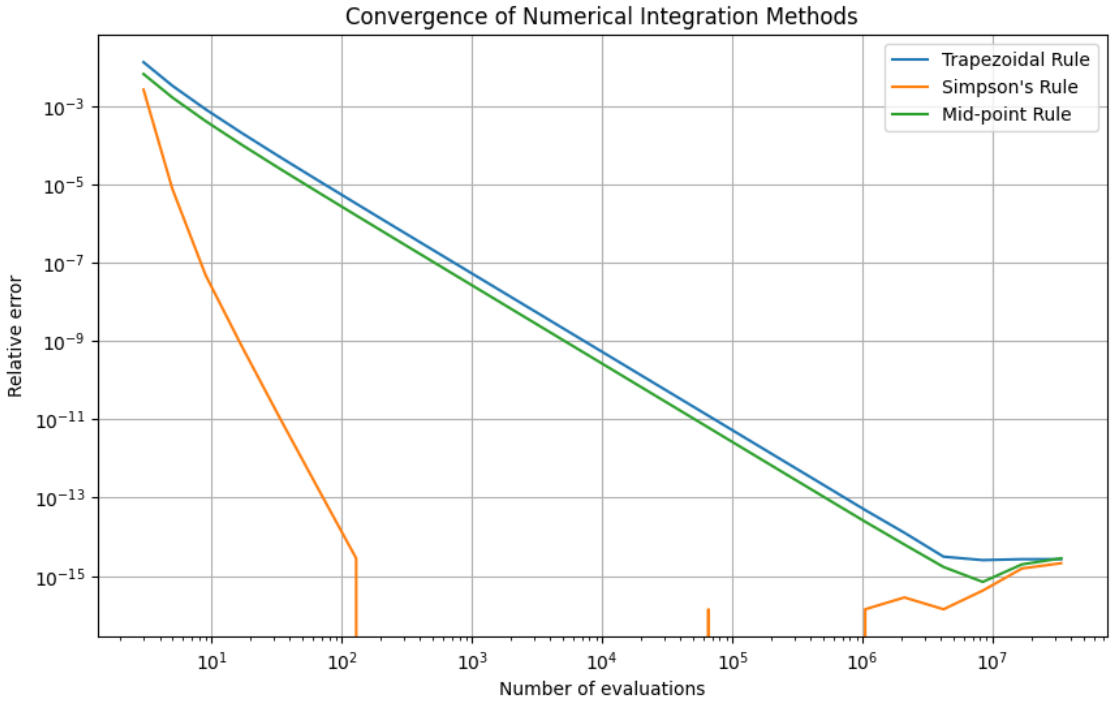
2.1.4 Obliczanie wartości całek i błędów dla różnych wartości m

[ ]:
```
# Zakres wartości m
m_values = np.arange(1, 26)

# Listy przechowujące błędy dla każdej metody
errors_trapezoidal = []
errors_simpson = []
errors_midpoint = []

for m in m_values:
    n = 2 ** m + 1
    x = np.linspace(0, 1, n)
    y = f(x)

    # Metoda trapezów
    integral_trapezoidal = trapz(y, x)
    error_trapezoidal = relative_error(np.pi, integral_trapezoidal)
    errors_trapezoidal.append(error_trapezoidal)

    # Metoda Simpsona
    integral_simpson = simps(y, x)
    error_simpson = relative_error(np.pi, integral_simpson)
    errors_simpson.append(error_simpson)

    # Metoda punktu środkowego
    x_mid = (x[1:] + x[:-1]) / 2
    y_mid = f(x_mid)
    integral_midpoint = np.sum(y_mid * (x[1:] - x[:-1]))
    error_midpoint = relative_error(np.pi, integral_midpoint)
    errors_midpoint.append(error_midpoint)
```

2.2 Wykres

[ ]:
```
n_values = 2**m_values +1

plt.figure(figsize=(10, 6))
```

```
plt.plot(n_values, errors_trapezoidal, label='Trapezoidal Rule')
plt.plot(n_values, errors_simpson, label='Simpson\'s Rule')
plt.plot(n_values, errors_midpoint, label='Mid-point Rule')
plt.yscale('log')
plt.xscale('log')
plt.xlabel('Number of evaluations')
plt.ylabel('Relative error')
plt.title('Convergence of Numerical Integration Methods')
plt.legend()
plt.grid(True)
plt.show()
```



2.2.1 Obliczenie h_{min}

[ ]:
```
midpoint = lambda y_mid, x : np.sum(y_mid * (x[1:] - x[:-1]))

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

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

h_min_trapezoidal = calculate_hmin(trapz)
print("H_min dla metody trapezow wynosi:",round(h_min_trapezoidal,7))
h_min_Simpson = calculate_hmin(simps)
print("H_min dla metody Simpsona wynosi:",round(h_min_Simpson,3))
```

```
H_min dla metody trapezow wynosi: 1e-07
H_min dla metody Simpsona wynosi: 0.002
```

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

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

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

    return error


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

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

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

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

print("Rząd zbieznosci dla metody trapezów: ",round(np.
 ↪mean(p_values_trapezoidal),2))
print("Rząd zbieznosci dla metody Simpsona: ",round(np.
 ↪mean(p_values_Simpson),2))
```

```
Rząd zbieznosci dla metody trapezów:  2.0
Rząd zbieznosci dla metody Simpsona:  3.07
```

2.3 Wnioski

Po wykonaniu obliczeń dla trzech różnych metod całkowania oraz narysowaniu wykresu wartości bezwzględnej błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej, można zaobserwować, że wszystkie trzy metody zblizają się do dokładnej wartości wraz ze zwiększaniem liczby ewaluacji funkcji podcałkowej. Błąd względny najszybciej maleje dla metody Simpsona, ale tez najszybciej zwiększa się błąd numeryczny.

Porównując empiryczny rząd zbieżności z rzędem zbieżności przewidywanym przez teorię, możemy zauważyć, że po zaokrągleniu dla metody trapezów empiryczny rząd zbieżności około 2 , podczas gdy dla metody Simpsona jest to 3.07. Otrzymane wyniki są bliskie z teoretycznymi przewidywaniami, ponieważ rząd zbieżności dla metody trapezów wynosi 2, a dla metody Simpsona wynosi 4, co potwierdza skuteczność tych metod w przypadku tej konkretnej funkcji.

3 Zadanie 2

Oblicz wartość całki

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

metodą Gaussa-Legendre'a. Narysuj wykres wartości bezwzględnej błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej, n + 1. Przyjmij na tyle duży zakres n, aby wykryć, kiedy błąd numeryczny zaczyna przeważać nad błędem metody. Postaraj się umiejscowić otrzymane wyniki na wykresie stworzonym w podpunkcie (a).

3.1 Rozwiązanie

3.1.1 Biblioteki

[ ]:
```
import numpy as np
from scipy.special import roots_legendre
import matplotlib.pyplot as plt
```

3.1.2 Funkcja podcałkowa

[ ]:
```
def f(x):
    return 4 / (1 + x**2)
```

3.1.3 Metoda Gaussa-Legendre'a

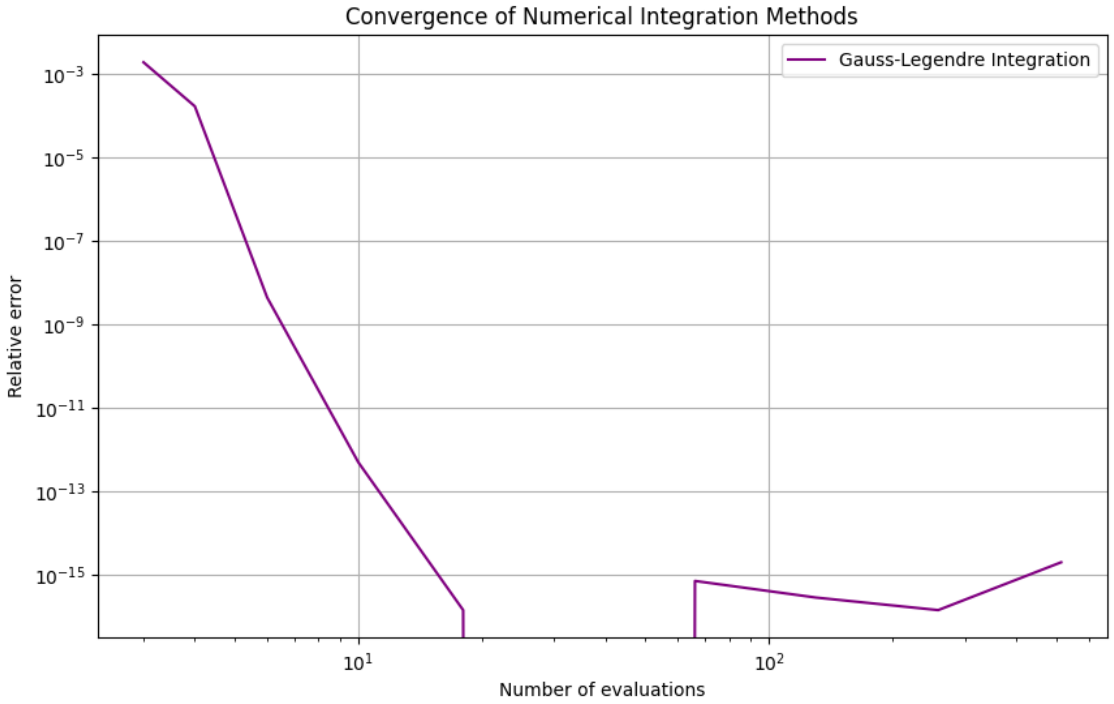
[ ]:
```
def gauss_legendre_integration(n):
    # Węzły i wagi Legendre'a dla n punktów
    x, w = roots_legendre(n)

    # Skalowanie węzłów do przedziału [0, 1]
    x_scaled = 0.5 * (x + 1)
    w_scaled = 0.5 * w

    integral = np.sum(w_scaled * f(x_scaled))
    return integral
```

3.1.4 Obliczanie błędów dla różnych wartości n

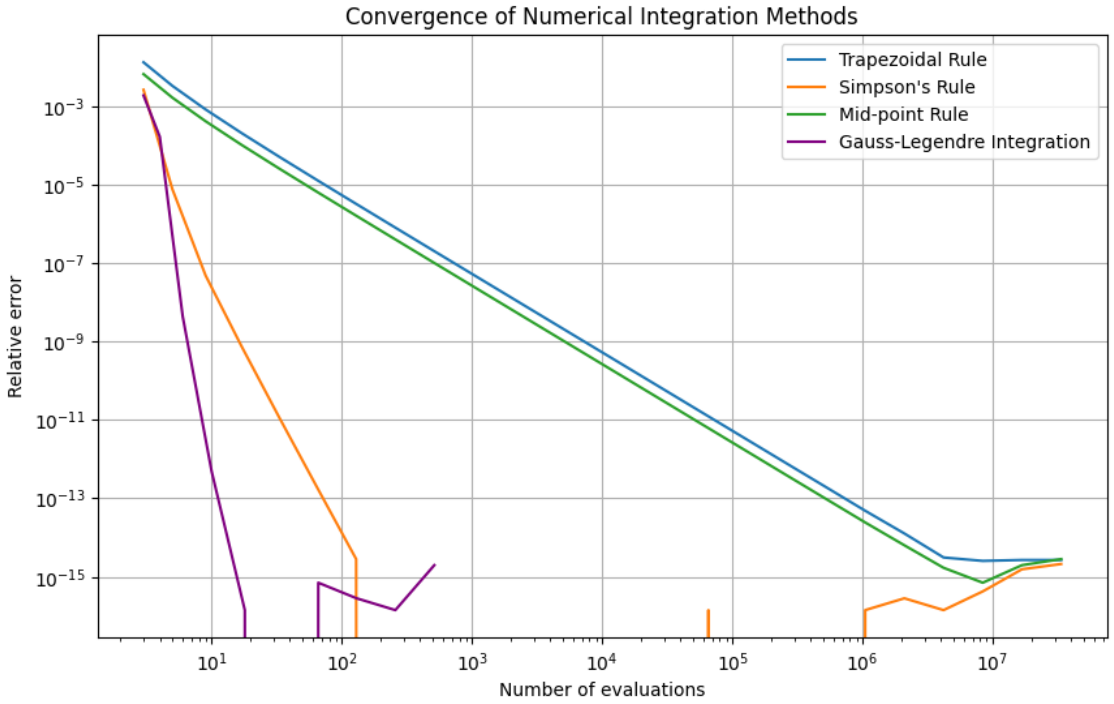
[ ]:
```
exact_integral = np.pi
errors_gauss_legendre = []
m_values = np.arange(0, 10)

for m in m_values:
    n = 2**m+1
    integral_approx = gauss_legendre_integration(n)
    error = np.abs((exact_integral - integral_approx) / exact_integral)
    errors_gauss_legendre.append(error)
```

3.2 Wykres

[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot((2**m_values+1) + 1, errors_gauss_legendre, color='purple',␣
 ↪label='Gauss-Legendre Integration')
plt.yscale('log')
plt.xscale('log')
plt.xlabel('Number of evaluations')
plt.ylabel('Relative error')
plt.title('Convergence of Numerical Integration Methods')
plt.legend()
plt.grid(True)
plt.show()
```



[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(n_values, errors_trapezoidal, label='Trapezoidal Rule')
plt.plot(n_values, errors_simpson, label='Simpson\'s Rule')
plt.plot(n_values, errors_midpoint, label='Mid-point Rule')
plt.plot((2**m_values+1) + 1, errors_gauss_legendre, color='purple',␣
 ↪label='Gauss-Legendre Integration')
plt.yscale('log')
plt.xscale('log')
plt.xlabel('Number of evaluations')
plt.ylabel('Relative error')
plt.title('Convergence of Numerical Integration Methods')
plt.legend()
plt.grid(True)
plt.show()
```



3.3 Wnioski

Jak mozna zaobserwować na wykresie błąd numeryczny zaczyna przewazać nad błędem metody dla liczby ewaluacji n trochę większym niz 10. Jest to punkt, w którym dalsze zwiększanie liczby ewaluacji funkcji podcałkowej nie przynosi już znaczącej poprawy w precyzji wyniku. W przypadku metody Gaussa-Legendre'a ten moment wydaje się być stosunkowo wcześnie, co wskazuje na skuteczność tej metody dla niewielkiej liczby węzłów.

Po umieszczeniu wartości błędów obliczonych za pomocą metody Gaussa-Legendre'a na wspólnym wykresie z z wynikami uzyskanymi dla innych metod całkowania (jak metoda prostokątów, trapezów i Simpsona) mozemy zauwazyc, ze błąd względny maleje najszybciej dla metody Gaussa-Legendre'a, a następnie dla metody Simpsona, ale tez ich błędy numeryczne najszybciej zaczynają przewazac nad błędami metody.

3.4 Bibliografia

prezentacja Quadratures - Marcin Kuta

https://home.agh.edu.pl/~funika/mownit/lab5/calkowanie.pdf