Dominik Szot, 20.04.2023

Laboratorium 06

Kwadratury

Zadanie 1 - obliczanie przybliżonej wartości π poprzez całkowanie numeryczne.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.linalg as scp
import scipy.integrate as integrate
import matplotlib.ticker
```

Funkcja którą będziemy wykorzystywać do obliczenia przybliżonej wartości π.

In [ ]:
```
# funkcja całkowana
f_x = lambda x : 4./(1 + x**2)
```

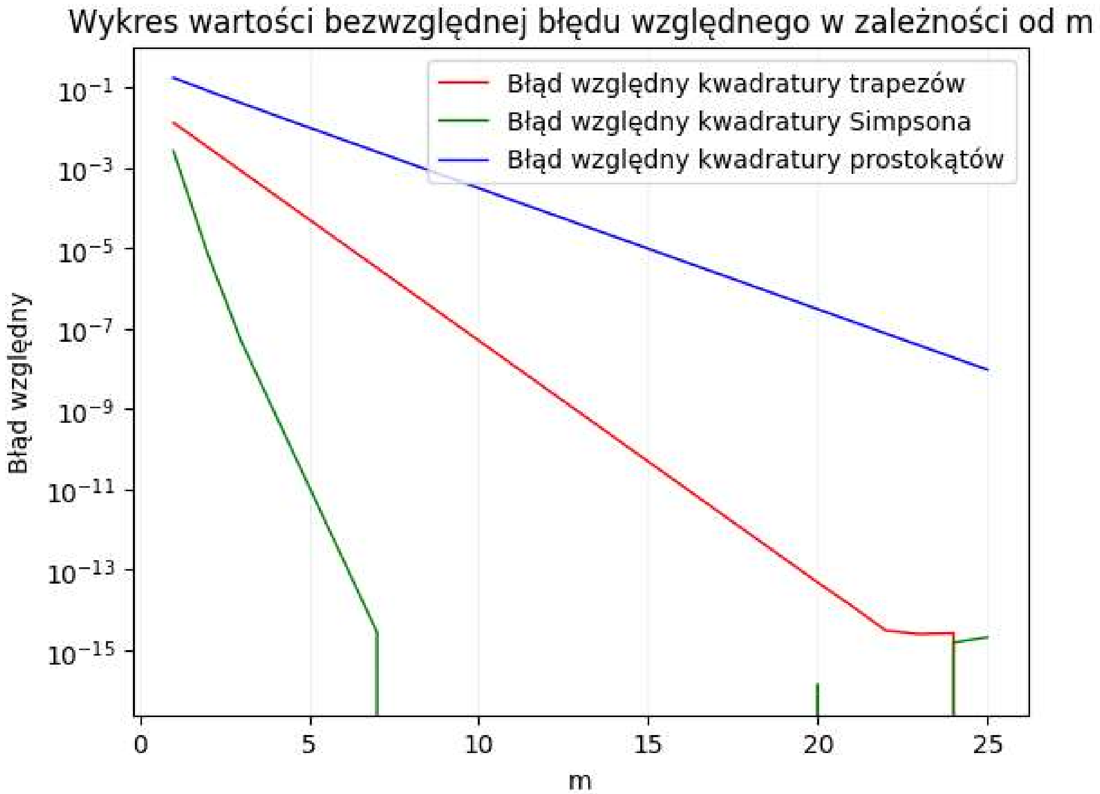
Metoda prostokątów do obliczenia wartości całki.

In [ ]:
```
def rectangular_method(nodes, values):
    cumulative_area=0

    for i in range(1, len(nodes)):
       cumulative_area += (nodes[i] - nodes[i-1])*values[i]

    return cumulative_area
```

Funkcja odpowiadająca za obliczanie wartości całki oraz wartości bezwzględnej błędu względnego w zależności od liczby równoodległych węzłów. Do obliczania wartości wykorzystałem funkcje `scipy.integrate.trapezoid()`, `scipy.integrate.simpson()` oraz wcześniej zdefiniowaną funkcję obliczająca wartości metodą prostokątów.

In [ ]:
```
m, a, b = 26, 0, 1

error_trapz = np.zeros(m-1, dtype=np.double)
error_simps = np.zeros(m-1, dtype=np.double)
error_rectangle = np.zeros(m-1, dtype=np.double)

for i in range(1, m):
    no_nodes = 2**i + 1

    # węzły kwadratury z przedziału [a, b]
    quadrature_nodes = np.array([np.double(a) + np.double(i) *
                                 np.double((b-a))/(no_nodes-1) for i in range(no_nodes)])

    # wartości węzłów
    quadrature_points = [f_x(i) for i in quadrature_nodes]

    result_trapz = integrate.trapezoid(quadrature_points, quadrature_nodes, 1)
    result_simps = integrate.simpson(quadrature_points, quadrature_nodes, 1)
    result_rectangle = rectangular_method(quadrature_nodes, quadrature_points);

    error_trapz[i-1]=((np.pi - result_trapz)/np.pi)
    error_simps[i-1]=((np.pi - result_simps)/np.pi)
    error_rectangle[i-1]=((np.pi - result_rectangle)/np.pi)
```

In [ ]:
```
x_no_points = [i for i in range(1, m)]

plt.semilogy(x_no_points, error_trapz, linewidth=1,  color="red",
             label="Błąd względny kwadratury trapezów")
plt.gca().yaxis.set_major_formatter(matplotlib.ticker.LogFormatterSciNotation())

plt.semilogy(x_no_points, error_simps, linewidth=1,  color="green",
             label="Błąd względny kwadratury Simpsona")
plt.gca().yaxis.set_major_formatter(matplotlib.ticker.LogFormatterSciNotation())

plt.semilogy(x_no_points, error_rectangle, linewidth=1,  color="blue",
             label="Błąd względny kwadratury prostokątów")
plt.gca().yaxis.set_major_formatter(matplotlib.ticker.LogFormatterSciNotation())
plt.grid(axis='x', color='0.95')

plt.title("Wykres wartości bezwzględnej błędu względnego w zależności od m")
plt.xlabel("m")
plt.ylabel("Błąd względny")
plt.legend()
plt.show()
```



Następnie wyznaczam najmniejszą wartość h_{min} - wartość poniżej której zmniejszanie kroku nie zmniejsza błędu kwadratury.

In [ ]:
```
# Funkcja szukająca minimalnej wartości

h_min_t, h_min_r , h_min_s  = 0, 0, 0

for i in range(1, len(error_trapz)):
    if error_trapz[i] < error_trapz[i-1] and error_trapz[i-1] > 0:
        h_min_t = error_trapz[i-1]
    else:
        break

for i in range(1, len(error_trapz)):
    if error_trapz[i] < error_rectangle[i-1] and error_rectangle[i-1] > 0:
        h_min_r = error_rectangle[i-1]
    else:
```

```
        break

for i in range(1, len(error_trapz)):
    if error_simps[i] < error_simps[i-1] and error_simps[i-1] > 0:
        h_min_s = error_simps[i-1]
    else:
        break
```

In [ ]:

```
print(f"Wartość minimalna dla kwadratury prostokątów: {h_min_r}")
print(f"Wartość minimalna dla kwadratury trapezów: {h_min_t}")
print(f"Wartość minimalna dla kwadratury Simpsona: {h_min_s}")
```

```
Wartość minimalna dla kwadratury prostokątów: 1.8972566473542012e-08
Wartość minimalna dla kwadratury trapezów: 3.1098756885421053e-15
Wartość minimalna dla kwadratury Simpsona: 2.8271597168564595e-15
```

Wartość minimalna dla kwadratury prostokątów: $1.8972566473542012e - 08$

Wartość minimalna dla kwadratury trapezów: $3.1098756885421053e - 15$

Wartość minimalna dla kwadratury Simpsona: $2.8271597168564595e - 15$

Wartość h_{min} wyznaczona w laboratorium 1. wynosi około $1.0 * 10^{-8}$

In [ ]:

```
def machineEpsilon(func=float):
    machine_epsilon = func(1)
    while func(1)+func(machine_epsilon) != func(1):
        machine_epsilon_last = machine_epsilon
        machine_epsilon = func(machine_epsilon) / func(2)
    return machine_epsilon_last

print("Eps. maszynowy dla np.double: ", machineEpsilon(np.double))
```

```
Eps. maszynowy dla np.double:  2.220446049250313e-16
```

In [ ]:

```
v1 = np.log(np.abs(error_rectangle[4]/error_rectangle[3]))/np.log((2**3 + 1)/(2**4 + 1))
v2 = np.log(np.abs(error_trapz[4]/error_trapz[3]))/np.log((2**3 + 1)/(2**4 + 1))
v3 = np.log(np.abs(error_simps[4]/error_simps[3]))/np.log((2**3 + 1)/(2**4 + 1))

print(f"Empiryczny rząd zbieżności dla kwadratury prostokątów: {v1}")
print(f"Empiryczny rząd zbieżności dla kwadratury trapezów: {v2}")
print(f"Empiryczny rząd zbieżności dla kwadratury Simpsona: {v3}")
```

```
Empiryczny rząd zbieżności dla kwadratury prostokątów: 1.0979991871648629
Empiryczny rząd zbieżności dla kwadratury trapezów: 2.1797463467357736
Empiryczny rząd zbieżności dla kwadratury Simpsona: 6.539051483127724
```

Empiryczny rząd zbieżności dla kwadratury prostokątów: 2.0858673100879317
Empiryczny rząd zbieżności dla kwadratury trapezów: 2.1797463467357736
Empiryczny rząd zbieżności dla kwadratury Simpsona: 6.539051483127724

Dla metod użytych w zadaniu kwadratura Simpsona posiada największy empiryczny rząd zbieżności. Oznacza to że metoda będzie najszybciej zbieżna do rozwiązania dla tych samych wartości h.

Teoretyczny rząd zbieżności trochę odbiega od wartości empirycznych, jednak
zarówno teoretyczny jak i empiryczny rząd największy jest dla kwadratury Simpsona.

Wartość h_{min} zależy w dużym stopniu od precyzji obliczeń. Dla metody kwadratury prostokątów błąd metody przważa nad błędem numerycznym dla m z przedziału [1, 25], natomiast dla kwadratury trapezów i Simpsona
wartości h_{min} wynoszą odpowiednio:

- Kwadratury Simpsona: $2.8271597168564595e-15$
- Kwadratury trapezów: $3.1098756885421053e-15$

Zadanie 2 - Obliczanie całki metodą Gaussa-Legendre'a

In [ ]:
```
f_x_2 = lambda x : np.double(4)/(1 + x**2)
```

Aby obliczyć wartości całki metodą Gaussa-Legendre'a skorzystałem z funkcji `np.polynomial.legendre.leggauss()` z biblioteki numpy.

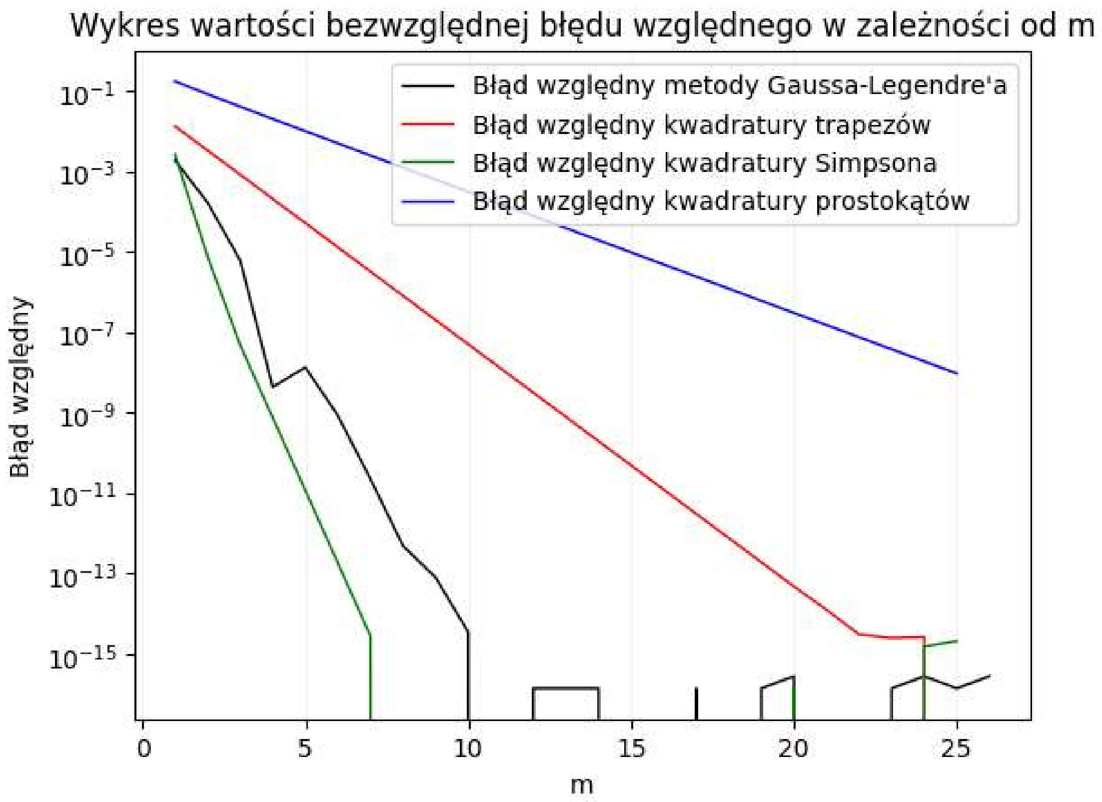
In [ ]:
```
def gauss_l(f_x_2, a, b, n):
    nodes = np.empty(n, dtype=np.double)
    weights = np.empty(n, dtype=np.double)
    mapped_nodes = np.empty(n, dtype=np.double)
    mapped_weights = np.empty(n, dtype=np.double)

    nodes, weights = np.polynomial.legendre.leggauss(n)

    mapped_nodes = (b-a)/(2) * nodes + (b+a)/2
    mapped_weights = (b-a)/(2) * weights

    return np.double(sum(mapped_weights * f_x_2(mapped_nodes)))
```

Obliczanie wartości całki oraz wartości bezwzględnej błedu względnego.

In [ ]:
```
f_2_values = np.array([gauss_l(f_x_2, np.double(0), np.double(1), i) for i in range(1,m+2)
gauss_l_errors = [np.abs((np.pi - f_2_values[i])/np.pi) for i in range(1, m+1)]
```

In [ ]:
```
x_no_points = [i for i in range(1, m)]

plt.semilogy([i for i in range(1, m+1)], gauss_l_errors, linewidth=1,
             color="black", label="Błąd względny metody Gaussa-Legendre'a")

plt.semilogy(x_no_points, error_trapz, linewidth=1,
             color="red", label="Błąd względny kwadratury trapezów")

plt.semilogy(x_no_points, error_simps, linewidth=1,
             color="green", label="Błąd względny kwadratury Simpsona")

plt.semilogy(x_no_points, error_rectangle, linewidth=1,
             color="blue", label="Błąd względny kwadratury prostokątów")

plt.gca().yaxis.set_major_formatter(matplotlib.ticker.LogFormatterSciNotation())
plt.grid(axis='x', color='0.95')
```

```python
plt.title("Wykres wartości bezwzględnej błędu względnego w zależności od m")
plt.xlabel("m")
plt.ylabel("Błąd względny")
plt.legend()

plt.gca().yaxis.set_major_formatter(matplotlib.ticker.LogFormatterSciNotation())
plt.grid(axis='x', color='0.95')

plt.show()
```



Porównanie metod przybliżających wartość całki.

Bibliografia

- Katarzyna Rycerz: Wykłady z przedmiotu Metody Obliczeniowe w Nauce i Technice
- Marcin Kuta: Materiały z zajęć - Quadratures
- https://en.wikipedia.org/wiki/Lagrange_polynomial