METODY NUMERYCZNE

RAPORT

Mateusz Wójcicki ISSP rok 2

01.2024

1.Cel raportu

Dana jest funkcja g(x) postaci:

$$g(x) = \int_0^x cos(t^2)\, dt$$

Moim zadaniem było wyznaczenie przebiegu wartości tej całki dla przedziału x=[0.0, 2.0], określenie, dla jakiej wartości x całka ta ma ekstremum oraz jaką wartość dla tego ekstremum przyjmuje.

Aby podołać tym zadaniom, będę korzystał z bibliotek języka Python: SciPy oraz NumPy, które oferują nam szeroki wachlarz możliwości.

2.Wstępna analiza oraz miejsce zerowe

Jak wiadomo całkowanie jest działaniem odwrotnym do różniczkowania. Stąd - funkcja podcałkowa $\cos(t^2)$ jest pochodną naszej funkcji $g(x)$. Wyznaczając miejsca zerowe tej funkcji można znaleźć ekstrema funkcji $g(x)$.

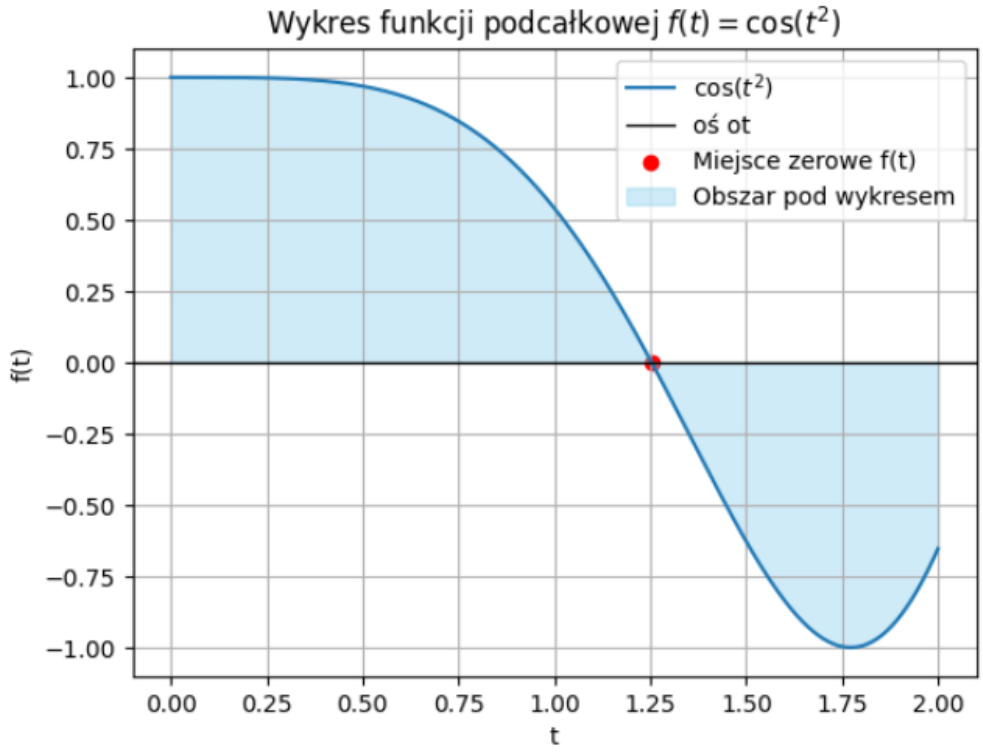
Aby zilustrować wykres funkcji podcałkowej w zadanym przedziale x=[0.0, 2.0] użyję biblioteki matplotlib, która umożliwia generowanie wykresów i analizę graficznej reprezentacji funkcji.

Natomiast aby wyznaczyć miejsce zerowe funkcji $f(t)$ posłużę się funkcją fsolve z biblioteki SciPy, która wyznacza miejsce zerowe funkcji dla podanego początkowego oszacowania (u mnie przyjąłem, że miejsce zerowe będzie w pobliżu wartości t = 1.25). Dokumentacja funkcji fsolve wskazuje nam, że ma ona zdefiniowany błąd nie większy niż $1.49012 * 10^{-8}$. Można podać własną wartość błędu do którego musi się dostosować, jednak ja stwierdziłem, że będzie to wynik dość dokładny i nie ma takiej potrzeby.

(Na następnej stronie znajduje się Kod 1)

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt

# Definicja funkcji
def f(t):
    return np.cos(t**2)

# Generuję 1000 wartości t w przedziale [0.0, 2.0]
t_values = np.linspace(0.0, 2.0, 1000)

# Obliczam wartości tej funkcji w tym przedziale
f_values = f(t_values)

# Plotuję funkcję
plt.plot(t_values, f_values, label=r'$\cos(t^2)$')

# Dodaję czarną linię dla y=0 aby pokazać oś ot
plt.axhline(y=0, color='black', linestyle='-', linewidth=1, label='oś ot')

# Znajduję punkt gdzie f(t) jest bliskie zeru za pomocą
# fsolve podając jako oszacowanie początkowe miejsca zerowego = 1.25
root = opt.fsolve(f, 1.25)

# Dodaję miejsce zerowe do wykresu
plt.scatter(root, np.zeros_like(root), color='red', marker='o', label='Miejsce zerowe f(t)')

# Zamalowanie obszaru pod wykresem
plt.fill_between(t_values, f_values, color='skyblue', alpha=0.4, label='Obszar pod wykresem')

# Opis wykresu i jego wyświetlenie
plt.title('Wykres funkcji podcałkowej $f(t) = \cos(t^2)$')
plt.xlabel('t')
plt.ylabel('f(t)')
plt.legend()
plt.grid(True)
plt.show()

# Wyświetlam wartość t miejsca zerowego
print("Miejsce zerowe funkcji f(t): t =", root[0])
```

Kod 1 : Kod generujący wykres funkcji $f(t)$ oraz podający wartość t dla miejsca zerowego tej funkcji.




```
Miejsce zerowe funkcji f(t): t = 1.2533141373154897
```

Na wykresie można zauważyć czerwony punkt, którym oznaczyłem miejsce zerowe funkcji podcałkowej $f(t)$. To właśnie w tym punkcie funkcja podcałkowa $f(t)$ zmienia wartości z dodatnich na ujemne, co wskazuje nam, że funkcja $g(x)$ posiada maksimum w tym miejscu.

3.Wyznaczenie wartości badanej całki

Wiedząc, że pole pod wykresem funkcji podcałkowej $f(t)$ będzie równe wartościom naszej całki w zadanym przedziale x=[0.0, 2.0] mogę wybrać metodę do ich wyznaczenia.

Zwracam teraz uwagę na charakterystykę funkcji. Wykres pochodnej, czyli $f(t)$ stosunkowo szybko się zmienia, co ma wpływ na nasze obliczenia. Najczęściej proste metody całkowania numerycznego przybliżają wartość całki sumując wartości całki w kilku punktach. W naszym przypadku, na tym przedziale, przy takiej oscylacji funkcji, wiele z tych metod mogły by być zbyt niedokładne.

W związku z tym zastosuję bardziej zaawansowaną metodę z biblioteki SciPy o nazwie „quad”.

Metoda quad wykorzystuje powszechnie znaną bibliotekę Fortran o nazwie „QUADPACK”. Używane są tutaj reguły Gaussa-Kronroda. Dużą zaletą tego rozwiązania jest automatyczne dostosowywanie się do kształtu funkcji, biorąc pod uwagę liczbę oscylacji. Ponadto metoda ta dostosowuje liczbę punktów kwadratury, aby osiągnąć zadaną dokładność, co sprawia, że nasze obliczenia będą bardzo dokładne.

```python
import numpy as np
from scipy.integrate import quad

# Definiuję funkcję f(t) = cos(t^2)
def f(t):
    return np.cos(t**2)

# Definiuję przedział x = [0.0, 2.0]
x_lower = 0.0
x_upper = 2.0

# Obliczam całkę z funkcji f(x) w przedziale x
result, error = quad(f, x_lower, x_upper)

# Obliczam całkę dla znalezionego maksimum
root = opt.fsolve(f, 1.25)
result_2, error_2 = quad(f, x_lower, root)

# Wyświetlam wyniki
print(f'Wynik całkowania: {result}')
print(f'Błąd oszacowania: {error}\n')
print(f'Wynik całkowania dla maksimum: {result_2}')
print(f'Błąd oszacowania dla maksimum: {error_2}')
```

```
Wynik całkowania: 0.46146146243321634
Błąd oszacowania: 3.035883305565741e-13

Wynik całkowania dla maksimum: 0.9774514242913296
Błąd oszacowania dla maksimum: 1.0851890767008871e-14
```

Kod 2 : Obliczenie wartości całki dla przedziału x=[0.0, 0.2], oraz dla maksimum za pomocą quad

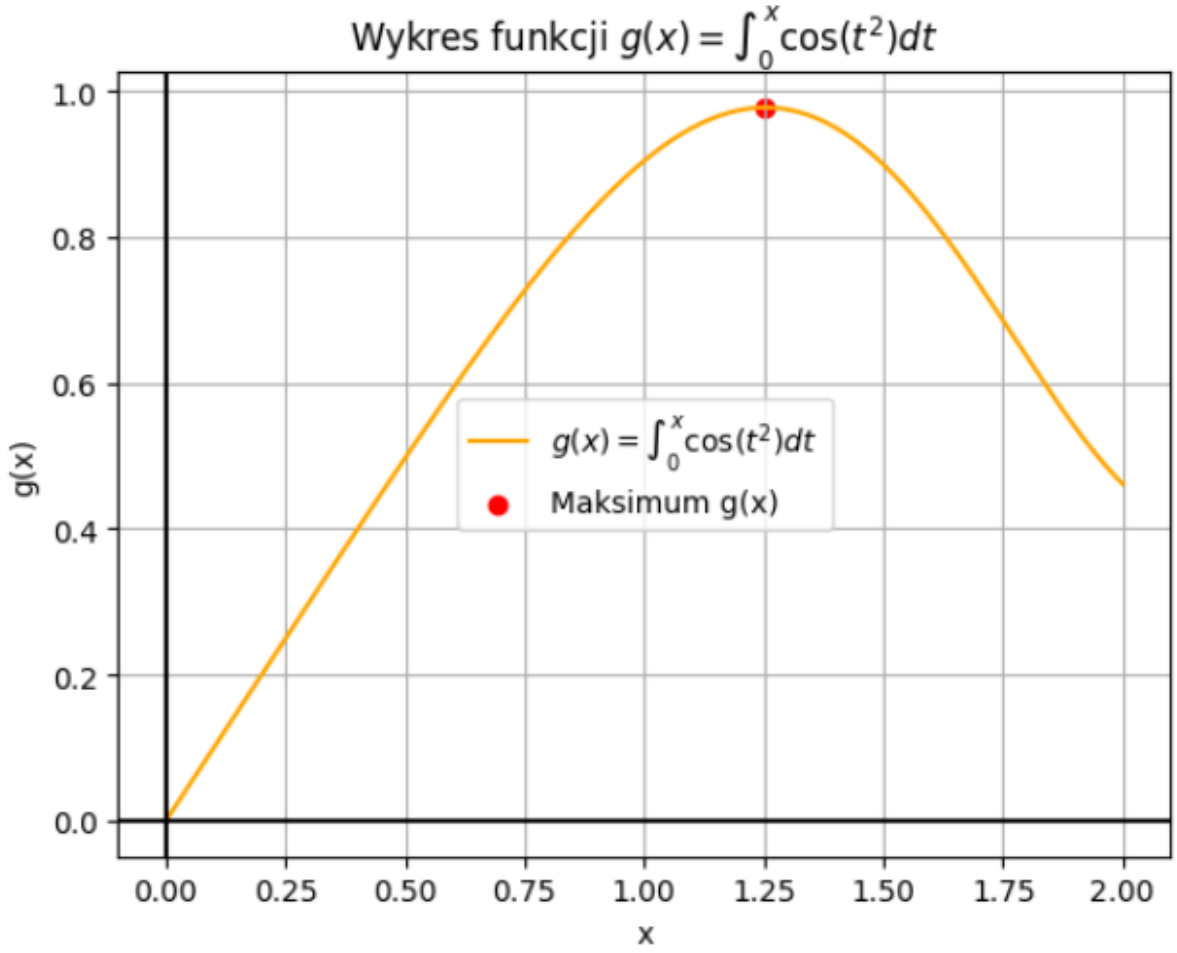
Wyniki są obarczone małymi błędami. Metoda quad oblicza poprawnie całkę, czyli pole pod wykresem funkcji, uwzględniając również przypadek, gdy część tej funkcji przechodzi pod oś poziomą.

W celu dodatkowego sprawdzenia możemy nasz wynik porównać np. z wynikiem podanym przez serwis WolframAlpha.
(link: https://www.wolframalpha.com/input?i2d=true&i=Integrate%5Bcos%5C%2840%29Power%5Bt%2C2%5D%5C%2841%29%2C%7Bt%2C0%2C2%7D%5D)

Wyniki zgadzają się.
Możemy teraz graficznie zaprezentować przebieg funkcji g(x) dla różnych wartości $x \in [0.0, 2.0]$ oraz nanieść maksimum.

```python
from scipy.integrate import quad
import numpy as np

# Definicja funkcji podcałkowej f(t)
def f(t):
    return np.cos(t**2)

# Tworzenie punktów x w przedziale x=[0.0, 2.0]
x = np.linspace(0, 2, 201)

# Dolna granica x
lower_limit = 0

# Tablica wyników
g = []

# Obliczanie całki z f(x)
for u in x:
    result, error = quad(f, lower_limit, u)
    g.append(result)


# Rysowanie wykresu
plt.plot(x, g, label='$g(x)= \int_{0}^{x} \cos(t^2) dt$', color='orange')

# Dodaję punkt maksimum
root = opt.fsolve(f, 1.25)
result_2, error_2 = quad(f, lower_limit, root)
plt.scatter(root, result_2, color='red', marker='o', label='Maksimum g(x)')

# Osie
plt.axvline(0, color='black', linestyle='-')
plt.axhline(0, color='black', linestyle='-')

# Opis wykresu
plt.title('Wykres funkcji $g(x)= \int_{0}^{x} \cos(t^2) dt$ ')
plt.xlabel('x')
plt.ylabel('g(x)')
plt.legend()
plt.grid(True)
plt.show()
```

Kod 3 : Kod generujący wykres funkcji $g(x) = \int_0^x \cos(t^2)\, dt$



Wykres pokrywa się z oczekiwaniami.

4. Podsumowanie

Podsumowując udało się pomyślnie wyznaczyć przebieg wartości funkcji podcałkowej $f(x) = \cos(t^2)$ oraz jej całki $g(x) = \int_0^x cos(t^2)\, dt$. Wyznaczono z dużą dokładnością ekstremum funkcji $g(x)$ z i określono wartość tej funkcji dla tego ekstremum. Obliczono również wartość tej całki dla całego przedziału x=[0.0, 2.0].

Metody quad oraz fsolve z biblioteki SciPy okazały się skuteczne w realizacji wyżej wymienionych zadań. Błędy wyznaczeń okazały się bardzo małe. Pomocna okazała się również biblioteka NumPy oraz Matplotlib służąca do między innymi graficznego przedstawienia przebiegu funkcji.

Na koniec chciałbym w celu wizualizacji rozwiązań, zestawić wykres funkcji $f(t) = \cos(t^2)$ oraz jej całkę $g(x) = \int_0^x cos(t^2)\, dt$ w przedziale x = [0.0, 2.0]: