MATEMATYKA

Metody Numeryczne

Projekt Indywidualny

Wykonał:

Oleg Łyżwiński 305158

Warszawa 2024

Poniżej przedstawiono rozwiązanie zadań F oraz G projektu indywidualnego z przedmiotu Metody Numeryczne. Do rozwiązań wykorzystano język Python. Zastosowano następujące biblioteki: numpy, matplotlib, math, scipy oraz sklearn.

Zad F 14

```python
# Metoda prostokątów
def metoda_prostokatow(fun, a, b, npanel):
    h = (b - a) / (npanel)
    x = np.linspace(a, b, npanel+1)
    integral = 0
    # Sumowanie wartośći na środku przediału
    for i in range(0, npanel):
        integral += fun((x[i] + x[i+1]) / 2)
    # Przemożenie przez długość przediału
    integral = integral * h
    return integral, h
```

```python
# Metoda trapezów
def metoda_trapezow(fun, a, b, npanel):
    h = (b - a) / (npanel)
    x = np.linspace(a, b, npanel+1)
    # Połowa wartości początu i końca przedziału
    integral = (fun(a) + fun(b)) / 2
    # Suma w węzłach
    for i in range(1, npanel):
        integral += fun(x[i])
    integral *= h
    return integral, h

# Metoda parabol
def metoda_parabol(fun, a, b, npanel):
    h = (b - a) / (npanel)
    x = np.linspace(a, b, npanel+1)
    integral = fun(a) + fun(b)
    for i in range(1, npanel):
        # Pażystą wartość razy 2 a niepażystą 4
        if i % 2 == 0:
            integral += 2 * fun(x[i])
        else:
            integral += 4 * fun(x[i])
    integral *= h / 3
    return integral, h
```

Celem zadania było wyznaczenie wartości całki opisanej funkcją :

```python
# Definicja funkcji
def funkcja(x):
    return x * np.exp(-x)
```

w przedziale (0, 5).

Liczbę przedziałów wyznaczono jako:

$$npanel = (n-1) \cdot 2$$

Gdzie: npanel to liczba przedziałów, a n to liczba węzłów.

Dla przyjętej liczby węzłów wyznaczono liczbę przedziałów oraz ich długość:

liczba wezlow	2	4	8	16	32
liczba przedziałów	2	6	14	30	62
dlugosc przedialu	2.5	0.8333333333333334	0.35714285714285715	0.16666666666666666	0.08064516129032258

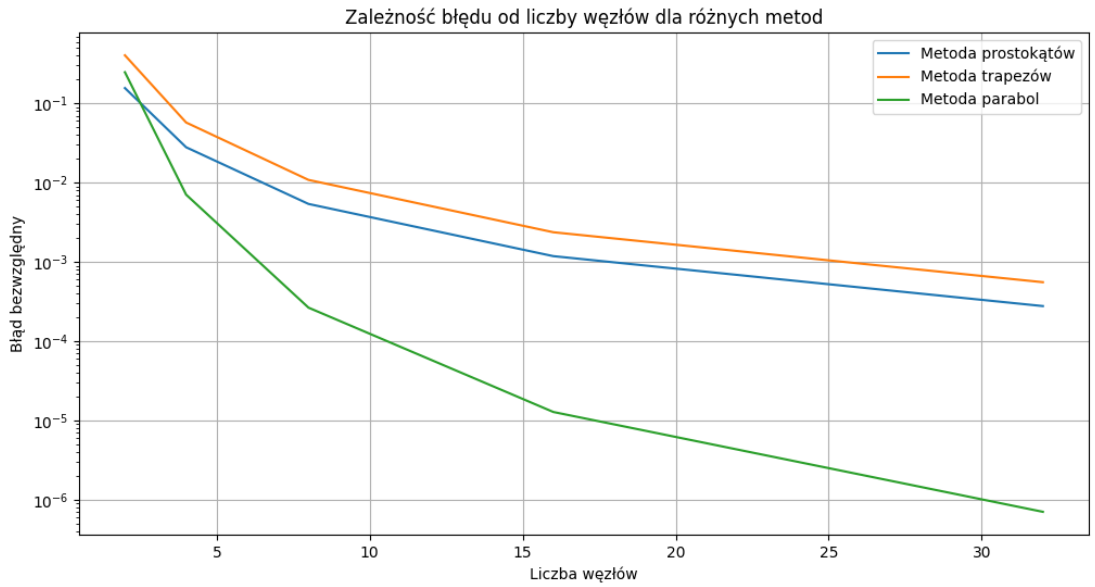
Wyznaczono wartości całki metodą prostokątów, trapezów oraz parabol. Wyznaczone wartości całki dla poszczególnej liczby węzłów przedstawiono poniżej:

liczba wezlow	2	4	8	16	32
wartosc dokladna	0.9595723180054871	0.9595723180054871	0.9595723180054871	0.9595723180054871	0.9595723180054871
metoda prostokatow	1.1158063575881794	0.9875735856437747	0.964970931945718	0.9607580972621597	0.959850451765255
metoda trapezow	0.5551434101436516	0.9021065919688964	0.9487243404726734	0.9571983407144093	0.9590159177555888
metoda parabol	0.7121164343620128	0.952502761485663	0.9593066816108966	0.959559470746215	0.9595716107916641

Wyznaczono wartość błędu bezwzględny porównując z wartością wyznaczoną za pomocą funkcji quad z biblioteki scipy.integrate:

liczba wezlow	2	4	8	16	32
metoda prostokatow	0.15623403958269222	0.028001267638287586	0.00539861394023089	0.0011857792566725145	0.00027813375976781796
metoda trapezow	0.4044289078618355	0.05746572603659072 5	0.010847977532813746	0.0023739772910778	0.000556400249898336 7
metoda parabol	0.2474558836434743 7	0.007069556519824105	0.0002656363945905 4045	1.284725927219732e-05	7.072138230590497e-07

Poniżej przedstawiono wykres zależności błędu bezwzględnego od liczby węzłów:



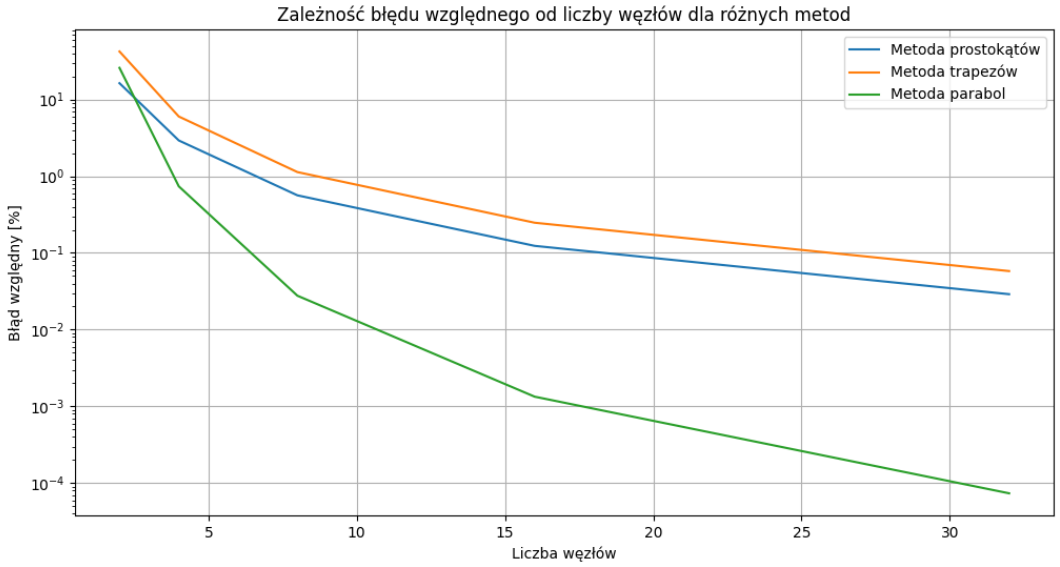
Na podstawie wykresu błędu bezwzględnego od liczby węzłów, wyraźnie możemy zaobserwować zależność dokładności uzyskanych wyników od zastosowanej metody. Najdokładniejszą metodą jest najbardziej złożona metoda parabol, a najmniej dokładną

metoda trapezów. Najmniej złożona metoda prostokątów pozwala na uzyskanie dokładniejszych wyników niż bardziej złożona metoda trapezów. Możemy również zaobserwować, że wraz ze wzrostem liczby węzłów, a co za tym idzie ilości przedziałów błąd wyznaczenia wartości całki maleje. Dla liczby węzłów równej 32 w przypadku metody prostokątów oraz trapezów różnica w stosunku do wyniku dokładnego jest rzędu 10^{-4}, a metody parabol 10^{-7}.

Wyznaczono również wartość błędu względny w [%]:

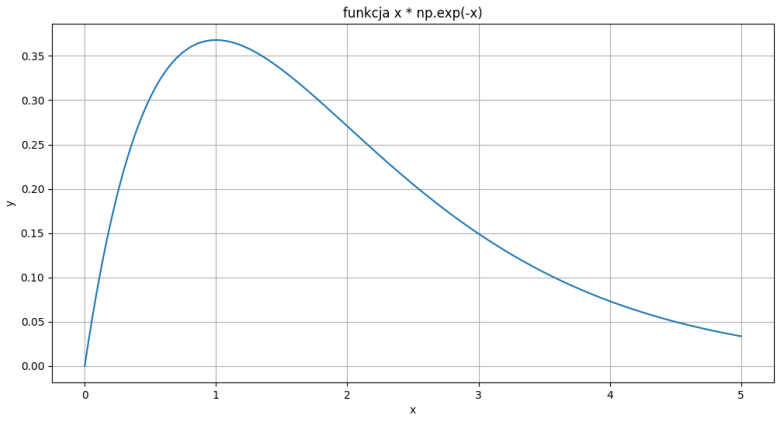
liczba wezlow	2	4	8	16	32
metoda prostokatow	16.28163262435826	2.9180987313691413	0.5626062610322214	0.12357372492124495	0.028985179600213053
metoda trapezow	42.146787717100736	5.988681098683187	1.1305013003461521	0.24739951815327638	0.05798419144216653
metoda parabol	25.788143217576575	0.7367403568413151	0.02768279051053466	0.0013388526358181017	7.370094049076201e-05

Poniżej przedstawiono wykres zależności błędu względnego od liczby węzłów:



Na podstawie uzyskanych wyników możemy zaobserwować, że metoda Możemy zaobserwować, że błąd względny dla metody parabol przy 32 węzłach wynosi jedynie $7{,}4 \cdot 10^{-5}\%$.

Dla liczby węzłów równej 2, a co za tym idzie liczby przedziałów równej 2 uzyskano najniższą wartość błędu (bezwzględnego oraz względnego) przy użyciu metody prostokątów, jest ciekawe ponieważ metodę tą cechuje najmniejsza złożoność. Wynik ten jest związany z przebiegiem funkcji (xexp(-x)):



Przy dwóch przedziałach wyznaczamy wartość funkcji w punkcie 1,25 (wartość duża)oraz 3,75 (wartość mała) złożenie tych dwóch wartości daje wartość odbiegającą od wartości oczekiwanej o 16,3%. W tym przypadku uzyskany wynik był związany z przebiegiem samej funkcji i nie jest to prawdą, że zawsze dl małej liczby węzłów uzyskamy najlepszy wynik metodą prostokątów.

Wykonano również sprawdzenie wartości całki dla 1 000 000 węzłów, a więc 19 999 998 przedziałów uzyskując następujące wartości błędu:

```
liczba wezlow                    1000000
metoda parabol     6.106226635438361e-15
```

Niestety takie obliczenia dla metody parabol zajęły, „aż”:

```
Czas obliczeń metyodą parabol dla 1000000 węzłów: 1.8877062797546387 sekundy
```

Porównując to z niemierzalną (nie możliwą do wyznaczania za pomocą funkcji time z biblioteki time)) wartością czasu dla 32 węzłów, możemy nazwać ją dużą.

Zad F 15

```python
# Metoda 3/8 Newtona
def metoda_trzy_osme(fun, a, b, npanel):
    h = (b - a) / npanel
    # x = np.linspace(a, b, npanel)
    integral = (fun(a) + fun(b))
    for i in range(1, npanel):
        x = 0.0
        x = a + i * h
        print(x)
        if i % 3 == 0:
            integral += 2 * fun(x)
        else:
            integral += 3 * fun(x)
    integral *= 3 * h / 8
    return integral, h
```

Celem zadania było wyznaczenie wartości całki :

$$Q = \frac{2}{\sqrt{\pi}} \int_0^1 e^{-x^2} dx$$

Liczbę przedziałów wyznaczono w następujący sposób:

$$npanel = (n-1) \cdot 3$$

Gdzie: npanel to liczba podprzedziałów, a n to liczba węzłów.

Dla przyjętej liczby węzłów wyznaczono liczbę podprzedziałów oraz ich długość:

liczba wezlow	3	6	9	12
liczba przedzaiłów	6	15	24	33
długość przedziału	0.16666666666666666	0.06666666666666667	0.041666666666666664	0.030303030303030304

Wyznaczone wartości całki metodą 3/8 Newtona oraz funkcji quad z biblioteki scipy.integrate:

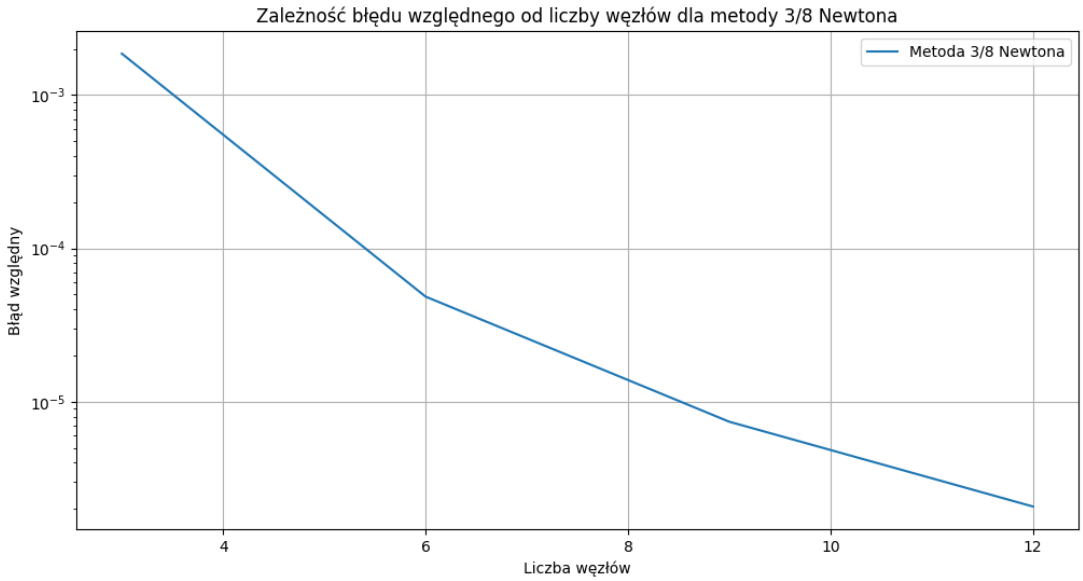
liczba wezlow	3	6	9	12
wartość całki	0.8427007929497149	0.8427007929497149	0.8427007929497149	0.8427007929497149
metoda 3/8	0.842716505290769	0.8427012020126396	0.8427008554551982	0.8427008104434505

Wyznaczone wartości błędu względnego oraz bezwzględnego przedstawiono w poniższej tabeli:

liczba wezlow	3	6	9	12
blad bezwzgledny	1.571234105413044e-05	4.0906292475551e-07	6.25054833447436e-08	1.7493735571250113e-08
blad wzgledny [%]	0.0018645219258821817	4.854189389375825e-05	7.417280708370401e-06	2.075913030770577e-06

Poniżej przedstawiono wykres zależności błędu bezwzględnego od liczby węzłów:

Poniżej przedstawiono wykres zależności błędu względnego od liczby węzłów:



Na podstawie powyższych wyników możemy zaobserwować, że wartość błędu dla metody 3/8 Newtona maleje wraz ze wzrostem ilości węzłów. Błąd bezwzględny dla 12 węzłów jest rzędu 10^{-8}. Metoda ta daje wysokiej jakości wyniki nawet dla małej liczby węzłów (3), dla której błąd względny wynosi : $1{,}8 \cdot 10^{-3}$ %.

„Alternatywny sposób wyznaczania”

Wykonano również podział przedziałów wynikających z ilości węzłów na podprzedziały złożone z trzech węzłów i w tych podprzedziałach stosowano metodę Newtona. Poniżej przedstawiono jak wyglądał ten podział dla 3 węzłów:

```
[0.  0.5 1. ]
0.0 0.5
0.16666666666666666
0.3333333333333333
0.5 1.0
0.6666666666666666
0.8333333333333333
```

Mamy trzy węzły od 0 do 1 więc nasze przedziały to od 0 do 0,5 oraz od 0,5 do 1. Teraz kolejno najpierw dla pierwszego przedziału od 0 do 0,5 przybliżamy wartość całki na tym przedziale metodą 3/8 Newtona tworząc w nim podprzedziały (od 0 do 1,1(6)), (od 1,1(6) do 0,(3)) oraz (od 0,(3) do 0,5).Powtórzono to dla następnego przedziału od 0,5 do 1. Podział na podprzedziały zastosowano dla każdej liczby węzłów uzyskując wyniki opisane poniżej.

Uzyskane wyniki są bardzo bliskie wynikom uzyskanej w pierwszej części zadania, różnicę widać dopiero na 16 miejscu po przecinku i są związane z dokładnością maszyny, a co za tym idzie błędami zaokrągleń:

liczba wezlow	3	6	9	12
wartość całki	0.8427007929497149	0.8427007929497149	0.8427007929497149	0.8427007929497149
metoda 3/8	0.842716505290769	0.8427012020126396	0.8427008554551985	0.8427008104434507

Poniżej przedstawiono wyznaczone wartości błędów:

liczba wezlow	3	6	9	12
blad bezwzgledny	1.571234105413044e-05	4.0906292475551e-07	6.25054835667882e-08	1.7493735793294718e-08
blad wzgledny [%]	0.0018645219258821817	4.854189389375825e-05	7.417280734719563e-06	2.07591305711974e-06

Metoda „Alternatywna” pozwala na uzyskanie identycznych wyników, ponieważ w przypadku kodu z pierwszej części zadania węzły pierwotne znajdujące się wewnątrz przedziału np. 0,5 są mnożone przez 2 ponieważ są węzły o numerach będących wielokrotnością 3

$$S_m(f) = \frac{3h}{8}\Big(f(x_0) + 3f(x_1) + 3f(x_2) + 2f(x_3) + 3f(x_4) + \ldots + 3f(x_{m-1}) + f(x_m)\Big).$$

Natomiast w przypadku metody podprzedziałów są węzłami krańcowymi, a więc mnożymy je przez 1, jednak robimy to dwa razy ponieważ początkowo są końcem przedziału, po czym stają się początkiem nowego podprzedziału.

$$S_m(f) = \frac{3h}{8}\Big(f(x_0) + 3f(x_1) + 3f(x_2) + 2f(x_3) + 3f(x_4) + \ldots + 3f(x_{m-1}) + f(x_m)\Big).$$

Podsumowując są to metody równoznaczne.

Zad F 16

```python
# Definicja funkcji podcałkowej
def f(x):
    return x**2

def funkcja(x):
    return x**2 / (np.sqrt((1 + x) * (1 - x)))

liczba_wezlow_tab = [11, 61, 101, 1001, 10001, 100001]
wynik_scipy, _ = quad(funkcja, -1, 1)

tab_wartosci = []
tab_blad_wzgledny = []

for liczba_wezlow in liczba_wezlow_tab:
    wezly = np.arange(1, liczba_wezlow+1, 1)
    wynik_gauss_czebyszew = 0.0
    pom = 0.0

    A_i = np.pi/(liczba_wezlow + 1)

    # Kwadratura Gaussa-Czebyszewa
    for i in wezly:
        x_i = np.cos(np.pi * ((2 * i )+ 1) / (2 * liczba_wezlow + 2))
        pom = (f(x_i ))
        wynik_gauss_czebyszew += pom

    wynik_gauss_czebyszew *= A_i
```

Celem zadania jest wyznaczenie wartości całki:

$$U = \int_{-1}^{1} \frac{x^2}{\sqrt{(1+x)(1-x)}} dx$$

Poniżej przedstawiono wzory na współczynniki kwadratury Gaussa-Czebyszewa w przedziale [-1;1]:

$$A_i = \frac{\pi}{n+1}, \qquad x_i = \cos\frac{(2i+1)\pi}{2n+2}.$$

Poniżej przedstawiono wzór przybliżonego całkowania dla kwadratury Gaussa-Czebyszewa:

$$\int_{-1}^{1} \frac{1}{(1-x^2)^{1/2}} f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) = S_n(f),$$

gdzie x_i są to zera wielomianu ortogonalnego $T_{n+1}(x)$.

Uzyskana wartość całki dla 61 węzłów:

```
Wynik z kwadratury Gaussa-Czebyszewa: 1.5201579953792734
Wynik dokładny:                       1.5707963267944247
```

Dla 61 węzłów błąd bezwzględny wynosił 0,0506, a błąd względny 3,2237%

Przeprowadzono testy metody dla liczby węzłów równej: 11, 61, 101, 1001, 10001, 100001.

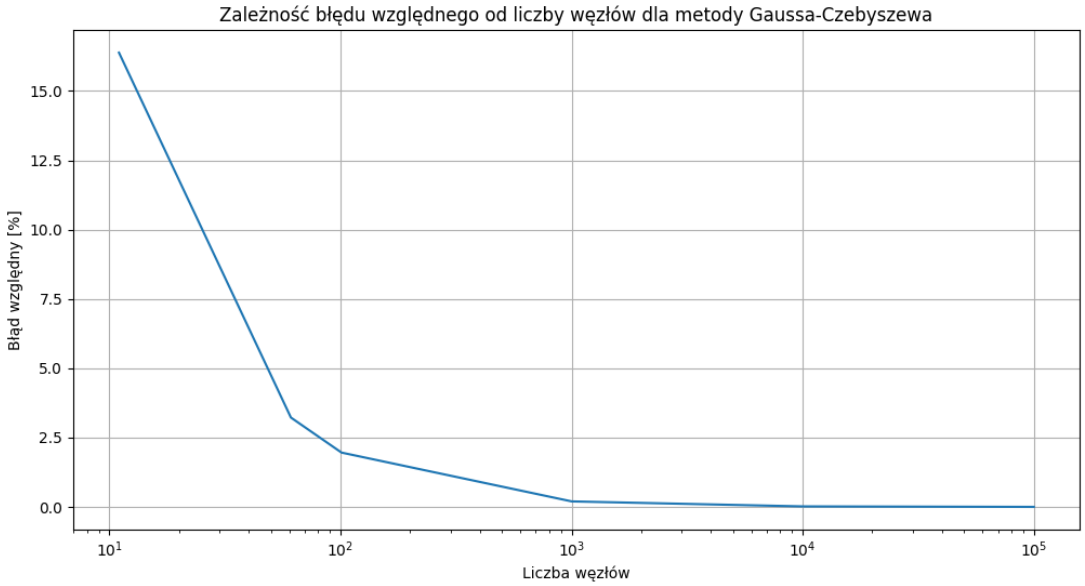
Uzyskano następujące wyniki:

liczba wezlow	11	61	101	1001	10001	100001
Wynik dokładny	1.5707963267944247	1.5707963267944247	1.5707963267944247	1.5707963267944247	1.5707963267944247	1.5707963267944247
Gauss-Czebyszew	1.3134572379043887	1.5201579953792734	1.5400037027178597	1.5676610124905646	1.5704822303565715	1.5707649114967028

Wartości błędu bezwzględnego oraz względnego dla przyjętych liczb węzłów przedstawiono w poniższej tabeli:

liczba wezlow	11	61	101	1001	10001	100001
Błąd bezwzględny	0.257339088890036	0.05063833141515128	0.030792624076565023	0.0031353143038601594	0.0003140964378531841	3.1415297721881785e-05
Błąd względny [%]	16.38271521905047	3.223736301859744	1.960319333022922	0.1996003078424876	0.01999600027676228	0.0019999599684570185

Poniżej przedstawiono charakterystykę błędu względnego od liczby węzłów dla kwadratury Gaussa-Czebyszewa:



Na podstawie wartości błędu względnego możemy zaobserwować, że błąd maleje wraz ze wzrostem ilości węzłów. Dla liczby węzłów z przedziału [101, 1001, 10001, 100001], wartość błędu bezwzględnego oraz względnego maleje liniowo (w skali logarytmicznej) z każdą iteracją zmniejszając wartość błędu o 10^{-1}.

Zad F 17

```python
# Definicja funkcji podcałkowej
def funkcja(x):
    return x**2 * np.exp(-x**2)

def f(x):
    return x**2

# Wartości wag i węzłów Gaussa-Hermite'a
wezly, wagi = hermgauss(5)

'''print(wezly,'\n')···

# Obliczenie wartości całki za pomocą kwadratury Gaussa-Hermite'a
wynik_gauss_hermite = np.sum(wagi * f(wezly))

# Obliczenie wartości całki za pomocą scipy.integrate.quad
wynik_scipy, _ = quad(funkcja, -np.inf, np.inf)
```

Celem zadania było wyznaczenie wartości niżej przedstawionej całki przy pomocy kwadratury Gaussa-Hermite'a:

$$V = \int_{-\infty}^{-\infty} x^2 e^{-x^2} dx.$$

Wzór przybliżonego całkowania metodą Gaussa-Hermite'a:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) = S_n(f),$$

gdzie x_i są zerami wielomianu Hermite'a $H_{n+1}(x)$.

Wartości współczynnika Ai oraz wartości miejsc zerowych wielomianu Hermite'a wczytano za pomocą funkcji hermgauss.

Uzyskano następujące wyniki:

```
Wynik z kwadratury Gaussa-Hermite'a:  0.8862269254527578
Wynik dokładny:                       0.8862269254527599
```

Poniżej przedstawiono wartości błędu względnego oraz bezwzględnego dla wyznaczonej całki:

```
Błąd bezwzględny: 2.1094237467877974e-15
Błąd względny:    2.380229810451905e-13
```

Wynik uzyskany za pomocą tej kwadratury obarczony był błędem bezwzględnym wysokości $2{,}11 \cdot 10^{-15}$. Możemy wnioskować, że metodę tą cechuje bardzo wysoka dokładność zbliżona do precyzji komputera.

Podjęto próbę wyznaczenia wartości kwadratury Gaussa-Hermite'a na podstawię poniższych wzorów:

$$A_i = \frac{2^{n+2}(n+1)!}{H'_{n+1}(x_i)H_{n+2}(x_i)}, \quad E(f) = \frac{[(n+1)!]\sqrt{\pi}}{2^{n+1}(2n+2)!} f^{(2n+2)}(\eta),$$

gdzie x_i są zerami wielomianu Hermite'a $H_{n+1}(x)$, $\eta \in (-\infty, \infty)$.

Implementacja numeryczna:

```
def wielomian_hermita(n):
    # Funkcja tworząca wielomian Hermite'a danego stopnia
    if n == 0:
        return [1]
    elif n == 1:
        return [0, 1]
    else:
        h = np.zeros((8, 9))
        h[0, 0] = 1
        h[1, 1] = 2

        for i in range(2, n+1 ):
            macierz = np.copy(h[i - 1, :])
            for k in range(9 - 1, 0, -1):
                macierz[k] = macierz[k-1]
            macierz[0] = 0
            for j in range(9):
                h[i,j] = 2 * macierz[j] - 2 * (i-1) * h[i - 2, j]

        return h
```

Wyznaczenie pochodnej:

```
def pochodna(wielomian_hermita_pochodna):
    for i in range(9):
        wielomian_hermita_pochodna[i] = wielomian_hermita_pochodna[i] * i

    for k in range(1, 8, 1):
        wielomian_hermita_pochodna[k-1] = wielomian_hermita_pochodna[k]
        wielomian_hermita_pochodna[8] = 0
    return wielomian_hermita_pochodna
```

Wyznaczenie współczynników Ai:

```
h = wielomian_hermita(7)
n = 5

const = (2**(n+2)*math.factorial(n+1))

h_n_1 = np.copy(h[6, :])
h_n_1_pochodna = pochodna(h_n_1)

h_n_2 = np.copy(h[7, :])

h_1_pochodna = 0.0
h_2_ = 0.0

A_i = []

for x_i in wezly:
    for i in range(0, 7, 1):
        h_1_pochodna += h_n_1_pochodna[i]*(x_i**i)
        h_2_ += h_n_2[i] * x_i**i
    pom = const / (h_1_pochodna * h_2_)
    A_i.append(pom)
```

Uzyskano następujące współczynniki wielomianów Hermite'a:

x^ 0	x^ 1	x^ 2	x^ 3	x^ 4	x^ 5	x^ 6	x^ 7	x^ 8
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-2.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	-12.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0
12.0	0.0	-48.0	0.0	16.0	0.0	0.0	0.0	0.0
0.0	120.0	0.0	-160.0	0.0	32.0	0.0	0.0	0.0
-120.0	0.0	720.0	0.0	-480.0	0.0	64.0	0.0	0.0
0.0	-1680.0	0.0	3360.0	0.0	-1344.0	0.0	128.0	0.0

Niestety nie udało się wyznaczyć wartości współczynników Ai ponieważ mianownik jest równy 0:

$$A_i = \frac{2^{n+2}(n+1)!}{H'_{n+1}(x_i)H_{n+2}(x_i)}, \qquad E(f) = \frac{[(n+1)!]\sqrt{\pi}}{2^{n+1}(2n+2)!} f^{(2n+2)}(\eta),$$

gdzie x_i są zerami wielomianu Hermite'a $H_{n+1}(x)$, $\eta \in (-\infty, \infty)$.

Uzyskano następujące wyniki wynoszące nieskończoność w wyniku dzielenia przez zero lub wartości bardzo duże wynikające z podzielnia przez liczbę bliską 0:

```
[inf, 39251110040702.04, 39251110040702.04, -38760649940206.71, inf]
```

Zad G 18

```python
# Definicja funkcji
def fun(x):
    return 2 + 2 * np.sin(x)**2 + np.cos(x)**2

# Symboliczna definicja funkcji dla obliczenia pochodnej analitycznej
x = sp.symbols('x')
y = 2 + 2 * sp.sin(x)**2 + sp.cos(x)**2

# Obliczenie pochodnej
dy_analitycznie = sp.diff(y, x)
print(dy_analitycznie)

# Konwersja pochodnej analitycznej na funkcję
dy_analitycznie_func = sp.lambdify(x, dy_analitycznie, 'numpy')

# Numeryczne obliczanie pochodnej
def numeryczna_pochodna(f, x, h):
    return (f(x + h) - f(x)) / h

# Zakres i krok różniczkowania
x_values = np.arange(0.0, 5.01, 0.01)
h = 0.01

# Obliczenie numerycznej pochodnej
dy_numerycznie = numeryczna_pochodna(fun, x_values, h)
```

Celem zadania było numeryczne wyznaczenia wartości pochodnej poniższej funkcji:

$$y = 2 + 2\sin^2(x) + \cos^2(x).$$

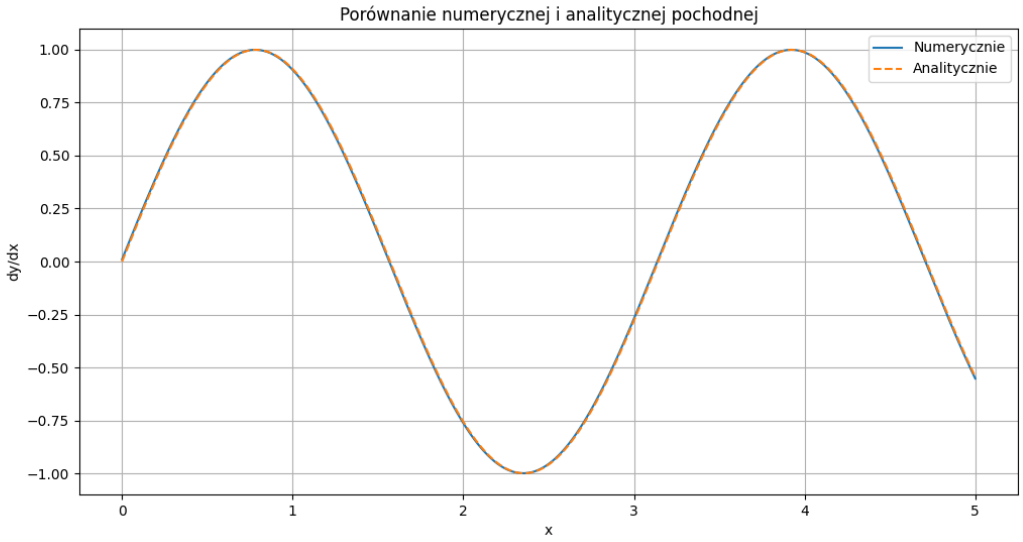
Wartość pochodnej wyznaczona przez funkcję sp.diff:

```
2*sin(x)*cos(x)
```

Wartość pochodnej przybliżono za pomocą poniższej funkcji:

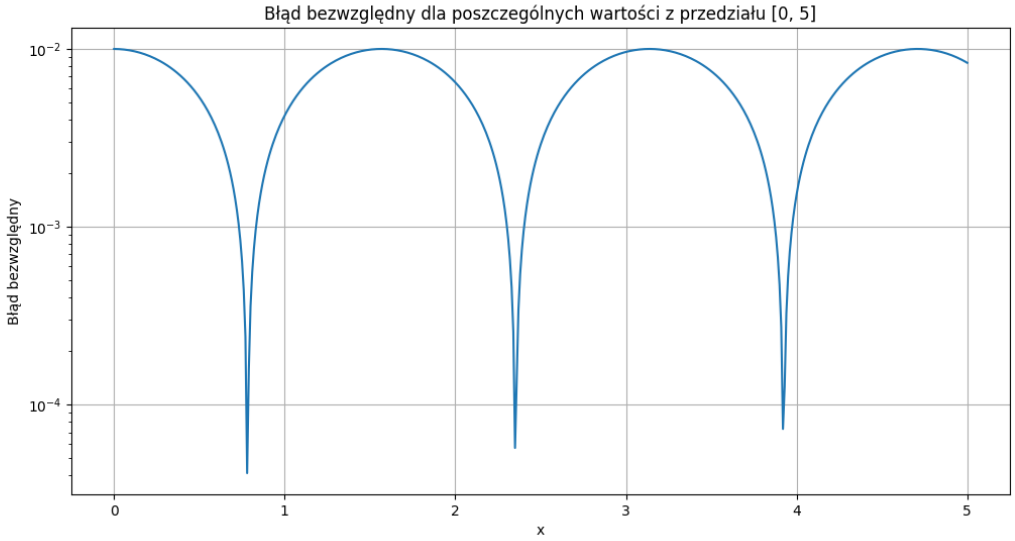
$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

Poniżej przedstawiono przebieg wartości pochodnej wyznaczonej numerycznie oraz analitycznie:



Na podstawie powyższego wykresu możemy zaobserwować, że przebieg pochodnej wyznaczonej analitycznie pokrywa się z przebiegiem pochodnej wyznaczonej numerycznie.

W celu porównania obu wartości wykreślono charakterystykę wartości błędu bezwzględnego dla zadanego przedziału:



Błąd bezwzględny zawierał się w następującym przedziale:

```
Maksymalna wartość błędu bezwzględnego: 0.009999871053723106
Minimalna wartość błędu bezwzględnego: 4.129612378966918e-05
```

Na podstawie powyższych wyników możemy ocenić, że metoda numeryczna pozwala na wyznaczanie pochodnej z wysoką dokładnością zawierającą się w przedziale od $9{,}9999 \cdot 10^{-3}$ do $4{,}1296 \cdot 10^{-5}$. Jest to bardzo duża dokładność biorąc pod uwagę prostotę metody.

W celu oceny podobieństwa przebiegów wyznaczono również wartość błędu średniokwadratowego:

```
Błąd średniokwadratowy: 0.007255433685715404
```

Oznacza to, że średnio wartość pochodnej wyznaczonej numerycznie różni się o ± 0,007255 od wartości wyznaczonej analitycznie. Jest to więc bardzo niska wartość sugerująca bliskość wartości całki wyznaczonej numerycznie do wyznaczonej analitycznie.

Wartość współczynnika R^2 potwierdza wysoką wartość dopasowania przebiegu pochodnych:

```
Wartość współczynnika R^2: 0.9998820240948112
```