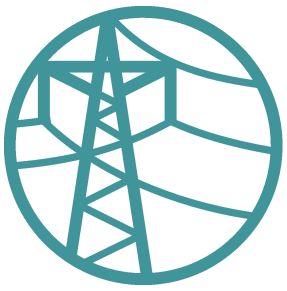


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Kamil Madej

Analiza i interpretacja wybranych metod całkowania numerycznego

Praca dyplomowa inżynierska

Opiekun pracy:
(dr. inż) Mariusz Borkowski (prof. PRz)

Rzeszów, 2022

Spis treści

1. Wstęp/wprowadzenie

Całkowanie numeryczne, zwane też kwadraturą to szeroka rodzina algorytmów służących do obliczania liczbowej wartości całki oznaczonej. Przy pomocy kwadratur problemy matematyczne są formułowane w taki sposób, że można je rozwiązać przy pomocy operacji arytmetycznych.

Podstawowym problemem całkowania numerycznego jest obliczenie przybliżonego rozwiązanai całki oznaczonej

$$\int_a^b f(x)dx \tag{1.1}$$

z pewnym stopniem dokładności

Do uzyskania tego celu opracowanych zostało wiele algorytmów. Najpopularniejsze z nich zostały bliżej omówionę w rozdziale 2.

Cel całkowania numerycznego

Istnieje wiele przypadków gdy całkowania numeryczne jest lepszym rozwiązaniem niż całkowanie analityczne lub jedynym rozwiązaniem:

- gdy całka z funkcji $f(x)$ znana jest jedynie dla pojedyńczych punktów, na przykład w przypadku próbkowania
- gdy wzór całki jest znany, ale znalezienie funkcji pierwotnej jest bardzo skomplikowane lub nie możliwe do zrobienia. Przykładem takiej funkcji jest $f(x) = exp(-x^2)$
- gdy znalezienie funkcji pierwotnej jest możliwe analitycznie, ale obliczenie numeryczne jest prostsze. Taki przypadek może wysąpić, gdy funkcja pierwotna jest dana jako nieskończony szereg lib iloczyn.

[**?**]

2. Użyte narzędzia

2.1. Język Python

Python jest wysokopoziomowym,interpertowanym,zorientowanym obiektowo językiem programowania, który w ostatnich latach zdobył wielką popularność i stał się głównym językiem wykorzystywanym w data science.[2] Pozwala on na przeprowadzanie złożonych obliczeń statystycznych, tworzenia wizualizacji danych, budowania algorytmów uczenia maszynowego, manipulowaniu i analizowaniu danych.
Przy pomocy bibliotek takich jak matplotlib python daje olbrzymie możliwości wizualizacji danych przy pomocy całego szeregu dostępnych wykresów. Biblioteki takie jak TensorFlow i Keras, pozwalają na szybkie i efektowne tworzenie programów do analizy danych i uczenia maszynowego.[3]

2.2. Użyte biblioteki

NumPy to jedna z najpopularniejszych bibliotek Python'a, skupiająca się na obliczeniach naukowych. Dostarcza wielowymiarowe obiekty tablicowe, które są lepsze pod względem wydajności i szybkości od standardowych list występujących w Pythonie.[4] Tablice te są homogeniczne, co oznacza, że mogą przechowywać tylko jeden typ danych na raz.Ta restrykcja umożliwia oddelegowanie wykonywania matematycznych operacji na tablicy do zoptymalizowanego kodu w języku C, który jest znacznie szybszy od interpretera Pythona[5]

Matplotlib jest biblioteką dostarczającą szeroki zakres funkcji do wizualizacji danych. Biblioteka ta jest rozszerzeniem biblioteki NumPy. Jako, że biblioteka jest darmowa, oraz zawiera proceduralny interfejs "Pylab", który został zaprojektowany tak, by w jak najbardziej przypominać MATLAB stanowi ona jego realną alternatywę.

3. Metody całkowania numerycznego

Istnieje wiele sposobów na obliczanie pola powierzchni pod krzywą przy pomocy metod numerycznych. Do dwóch najbardziej popularnych i poruszanych w tej pracy należą:

1) Kwadratury bazujące na interpolowaniu funkcji

- Metody Netwona-Cotesa
- Kwadratury Gaussa
- Metody adaptacyjne

2) Kwadratury bazujące na próbkowaniu funkcji

- Metody Monte Carlo

3.1. Metody Netwona-Cotesa

W analizie matematycznej, metodami Newtona-Cotesa, nazywamy grupę metod do całkowania numerycznego, które bazują na oszacowaniu wartości całki na skończonym przedziale, przy użyciu interpolacji wielomianem odpowiedniego stopnia, wyznaczając $n+1$ równo rozmieszczonych punktów, które dzielą przedział całkowania na n podprzedziałów. [6]

Metodę Newtona-Cotesa dla $n+1$ punktów można zdefiniować jako:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \tag{3.2}$$

gdzie x_i jest zbiorem równo rozmieszczonych punktów z przedziału [a,b]
a w_i jest zbiorem wag

W metodzie Newtona-Cotesa wagi poszczególnych węzłów otrzymywane są poprzez interpolację wielomianem Lagrange'a:

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \int_a^b \sum_{i=0}^{n} f_i L_{n,j}(x)dx = \sum_{i=0}^{n} (\int_a^b L_{n,j}(x)dx) f_i = \sum_{i=0}^{n} w_i f_i \tag{3.3}$$

[7]

3.1.1. Metoda prostokątów

Metoda prostokątów to najprostszy wariant metody Newtona-Cotesa gdzie całkę $\int_a^b f(x)$ przybliżamy przy użyciu interpolacji wielomianem Lagrange'a stopnia 1. Dla pojedyńczego podprzedziału metoda prostokątów wygląda następująco:

$$\int_a^b f(x) = I = f(a) * h \tag{3.4}$$

dla n podprzedziałów metoda prostokątów przybiera formę:

$$\int_a^b f(x) = h * \sum_{i=0}^{n-1} f(x_i) \tag{3.5}$$

gdzie h jest szerokością pojedyńczego podprzedziału i równa się $h = \frac{b-a}{n}$ [8]

Listing 1 pokazuje przykładowy kod w języku Python implementujący metodą trapezów.

```
import numpy as np

def f_rectI(f,a,b,n):
    h = (b - a) / n

    X = np.linspace(a + 0.5 * h, b - 0.5 * h, num=n)

    Y = []
    for i in range(0, n):
        Y.append(f(X[i]))

    Y = np.array(Y)

    I = h * Y
    I = np.sum(I)
    return I
```

Listing 1: Kod w języku python implementujący metodę prostokątów

3.1.2. Metoda trapezów

Metoda trapezów to kolejny wariant metody Newtona-Cotesa, w którym całkę $\int_a^b f(x)$ przybliżamy przy użyciu wielomianu Lagrange'a stopnia 2. Dla pojedyńczego

podprzedziału metoda trapezów wygląda następująco:

$$\int_a^b f(x) = \frac{1}{2} * h[f(a) + f(b)] \tag{3.6}$$

dla n podprzedziałów metoda prostokątów przybiera formę:

$$\int_a^b f(x) = \frac{1}{2} \sum_{i=0}^{n} [f(x_i + f(x_{i+1}))] \tag{3.7}$$

Listing 2 pokazuje przykładowy kod w języku Python implementujący metodą trapezów.

```
import numpy as np


def f_trapI(f, a, b, n):
    h = (b - a) / n

    X = np.linspace(a, b, num=n + 1)

    Y = []
    for i in range(0, n + 1):
        Y.append(f(X[i]))

    Y = np.array(Y)

    I = []
    for i in range(0, n + 1):
        if i == 0 or i == n:
            I.append(h * Y[i] / 2)
        else:
            I.append(h * Y[i])

    I = np.sum(I)
    return I
```

Listing 2: Kod w języku python implementujący metodę trapezów

3.1.3. Metoda Simpsona

Metoda Simpsona zwana też metodą parabol, to kolejny wariant metody Newtona-Cotesa. Tym razem całkę $\int_a^b f(x)$ przybliżamy przy użyciu wielomianu Lagrange'a stopnia 3, czyli paraboli.

Dla pojedyńczego podprzedziału metoda trapezów wygląda następująco:

$$\int_a^b f(x) = \frac{1}{3} * h[f(a) + 4f(a+h) + f(a+2h)] \tag{3.8}$$

dla n podprzedziałów metoda prostokątów przybiera formę:

$$\int_a^b f(x) = \frac{1}{2} h[f(a) + f(b)] + \sum_{i=1,3,5}^{n-1} 4f(x_i) + \sum_{i=2,4,6}^{n-2} f(x_i) \tag{3.9}$$

Listing 3 pokazuje przykładowy kod w języku Python implementujący metodą simpsona.

```
1
2  def f_simp2I(f,a,b,n):
3
4      h = (b-a)/n
5
6      I = f(a) + f(b)
7
8      for i in range(1,n):
9          if i%2==1:
10             I += 4 * f(a + i*h)
11         else:
12             I += 2 * f(a +i*h)
13
14     I = h/3 * I
15     return I
```

Listing 3: Kod w języku python implementujący metodę simpsona

3.2. Kwadratury Gaussa

Kwadratury Gaussa to metody numeryczne służące do przybliżania skończonych całek, najczęściej określane jako ważona suma wartości funkcji w określonych punktach dziedziny całkowania. Całkowanie metodą Gauss'a opiera się na użyciu wielomianów to

przybliżenia funkcji podcałkowej $f(x)$ na przedziale [-1,1], poprzez użycie odpowiednich węzłów x_i oraz wag w_i.

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{3.10}$$

Dokładność i optymalność wyniku całkowania zależy od odpowiedniego wyboru wielomianu interpolacyjnego. Użycie wielomianów Legendre'a całkowanie metodą Gaussa daje dokładny wynik dla wilomianów stopnia $2n-1$ lub niższego.

3.2.1. Gauss Legrande

Kwadratura Gaussa-Legendre'a jest specjalny przypadek kwadratury Gaussa, która pozwala na efektywne przyblienie funkcji ze znanym zachowaniem asymptotycznym na brzegach przedziału całkowania. Kwadratura Gaussa jest szczególnie zalecana, jeśli całka jest holomorficzna w sąsiedztwie przediału całkowania.

Węzły x_i są pierwiastkami wielomianu Legendre'a $P(x)$ stopnia n. Nie istnieje prosty sposób wyznaczenia pierwiastków x_i, mogę one jednak zostać aproksymowane z dużo dokładnością przy użyciu wzoru:

$$x_i \approx cos(\pi \frac{\frac{1}{2}+i}{N}) \tag{3.11}$$

Wagi w_i można wyrazić wzorem:

$$w_i = \frac{2(1-x_i^2)}{[nP_{n-1}(x_i)]^2} = \frac{2}{[P_n^1(x_i)]^2} \tag{3.12}$$

[9]

Listing 4 pokazuje przykładowy kod w języku Python implementujący kwadrature Gauss Legendre'a.

```
1
2  import numpy as np
3
4  def f_gauss_legrande(f,a,b,n):
5      half = float(b-a)/2
6      mid = (a+b)/2
7      [t,w] = np.polynomial.legendre.leggauss(n)
8
9      I = 0
```

```
    for i in range(n):
        I += w[i] * f(mid+half*t[i])

    I *= half

    return I
```

Listing 4: Kod w języku python implementujący metodę simpsona

3.3. Metody adaptacyjne

Tradycyjne metody całkowania Newtona-Cotesa, ignorują fakt, że całkowana funkcja posiada regiony o dużej, jak i małej zmienności. Metody adaptacyjne rozwiązują ten problem poprzez dostosowanie wielkości podprzedziałów, na mniejsze, w miejscach gdzie funkcja zmienia się gwałtownie i większe, w miejscach o mniejszej zmienności.[10]

Metody adaptacyjne polegają na wykorzystaniu tradycyjnych metod całkowania takich jak: Metoda Spimsona, trapezów do obliczenia całki na przedziale [a,b] a zadaną dokładnością ξ. Jeśli po pierwszej iteracji wartość całki nie jest dostatecznie dokładna, to przedział dzielimy na połowy, i ponownie całkujemy każdą z otrzymanych połówek. Proces ten powtarzamy do uzyskania zadanej dokładności.[11]

Całkowita wartość całki jest obliczana jako suma przybliżeń całki na wszystkich podprzedziałach.

Na przykładzie metody trapezów adaptacyjna metoda wygląda następująco Przedziale $[a,b]$ jest dzielona na n podprzedziałów $[a_j,b_j]$, dla j $= 0,1,...,n-1$, a następnie dla każdego podprzedziału obliczana jest całka według wzoru:

$$I_j(f) = \int_{a_j}^{b_j} f(x)dx \tag{3.13}$$

Powyższe podejście nie różni się niczym od klasycznej metody trapezów, jednak w metodach adaptacyjnych podprzedział $[a_j,b_j]$ jest dzielony na pół, gdy wartość $I_j(f)$ nie została obliczona z zadaną dokładnością. Do ustalenia dokładności, używamy kwadratur na przedziale $[a_j,b_j]$, by uzyskać przybliżenie $I1j$, a następnie całkujemy funkcję dzieląc przedział $[a_j,b_j]$ na dwa podprzedziały, by obliczyć drugie przybliżenie I_2. Jeśli I_1 oraz I_2 są dostatecznie zbliżone, wtedy możemy stwierdzić, że przybliżenie jest wystarczające i nie ma potrzeby dalszego dzielenia $[a_j,b_j]$. W przeciwnym przypadku

dzielimy $[a_j, b_j]$ na dwa podprzedziały i powtarzamy proces ponownie. Używamy tej techniki na wszystkich podprzedziałach tak długo, aż funkcja podcałkowa f zostanie przybliżona z zadaną dokładnością.[12]

Listing 5 pokazuje przykładowy kod w języku Python implementujący adaptacyjną metodę trapezów.

```
1
2  def f_trapI(f,xa,xb):
3      h = xb-xa
4      I = h*(f(xa) + f(xb))/2
5      return I
6
7  def f_adapt(f,ax,bx,tol):
8      m = (ax+bx)/2.0
9      P1 = f_trapI(f,ax,m)
10     P2 = f_trapI(f,m,bx)
11
12     if abs(P1 - P2 ) < 3 * tol:
13         return  P2
14     else:
15         return f_adapt(f,ax,m,tol/2) + f_adapt(f,m,bx,tol)
```

Listing 5: Kod w języku python implementujący metodę simpsona

3.4. Metody Monte Carlo

Metoda Monte Carlo to zupełnie odmienne od metod Newtona Cotesa oraz metod Gaussa podejście do obliczania wartości całki. Polega ona na obliczaniu pola powierzchni pod krzywą używając losowo rozmieszczonych punktów w obrębie granic całkowania.
Istnieją dwa rodzaje metod Monte Carlo.

3.4.1. Crude Monte Carlo

Podstawowa metoda Mone Carlo, zwana też Crude Monte Carlo polega ona na wylosowaniu n punktów w obrębie przedziału całkowania i na podstawie tych danych obliczenie średniej wartości funkcji. [14]

$$f_{sr} = \frac{f(x_1) + f(x_2) + ... + f(x_n)}{n} \tag{3.14}$$

Przybliżoną wartość całki otrzymujemy dzieląc uzyskaną średnią wartość funkcji przez długość przedziału całkowania.

$$I = f_{sr} * |b - a| \tag{3.15}$$

3.4.2. Monte Carlo

Bardziej dokładną wersją jest metoda Monte Carlo, która polega na wylosowaniu n punktów znajdujących się w polu kwadratu, który wyznaczany jest przez przedział całkowania $< a, b >$ oraz zakres wartości funkcji w tym przedziale $< f(a), f(b) >$. Po wylosowaniu n punktów wartość całki wyrażana jest jako:

$$I = P * \frac{c}{n} \tag{3.16}$$

gdzie c to ilość punktów leżących się pod krzywą.

Wraz ze zwiększaniem ilości losowanych punktów, rozkładają się one bardziej równomiernie w obrębie wyznaczonego prostokąta, dając coraz dokładniejszy wynik.[15]

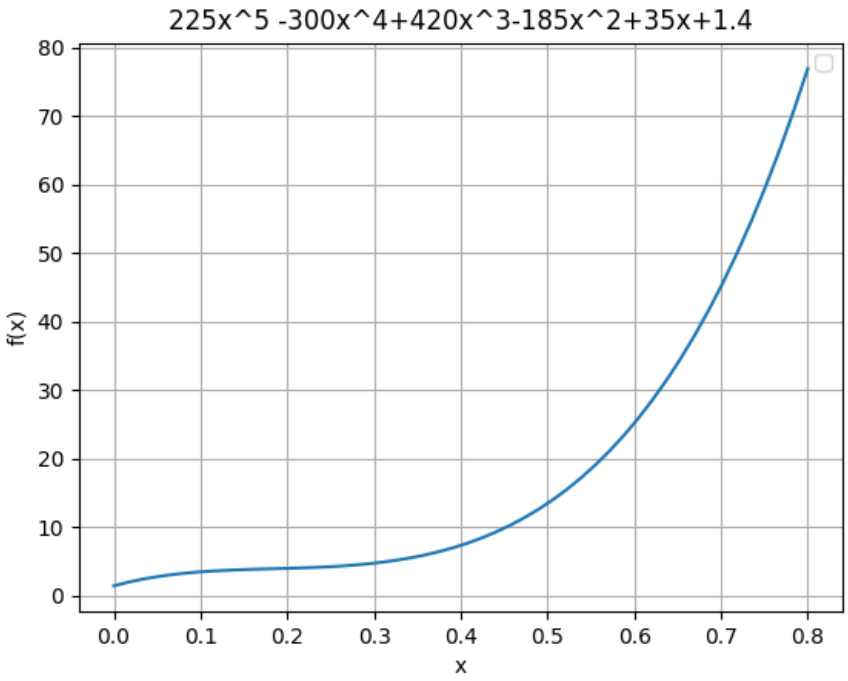
4. Bibliotek do całkowania numerycznego

Następne podrozdziały przedstawiają wyniki i omówienie działania zaimplementowanych przeze mnie metod całkowania dla równan (5.23) (5.24) (**??**). Dla metod całkowania, których logiką działania jest liczenie całki przez podział przedziału całkowania na podprzedziały, skuteczność metody wyrażana będzie w ilości podprzedziałów, na które musiał zostać podzielony przedział całkowania, by otrzymać dokładnoć przybliżenia rzędu 0.01 lub jeśli to możliwe, dokładną wartość całki. Dla metod Monte Carlo skuteczność liczona będzie ilością punktów potrzebnych do uzyskania zadowalającej dokładności. W adaptacyjnej metodzie trapezów miarą skuteczności będzie ilość wywołań funkcji, potrzebna do uzyskania zadanej dokładności przybliżenia funkcji.

Dokładność wszystkich metod całkowania zostanie zbadana na podstawie funkcji wielomianowej (5.23) oraz trygonometrycznej (5.24). Dodatkowo funkcja (5.25), która cechuje się dużą zmiennością przy prawym krańcu przedziału całkowania zostanie użyta do zbadania wyższości metody adaptacyjnej trapezów nad zwykłą metodą trapezów.

Funkcja wielomianoa

$$f(x) = 225x^5 - 300x^4 + 420x^3 - 185x^2 + 35x + 1.4 \tag{4.17}$$



Rysunek 4.1: Wykres funkcji zdefiniowanej wzorem(5.23)

Funkcja trygonometryczna

$$f(x) = sin(8x) \tag{4.18}$$

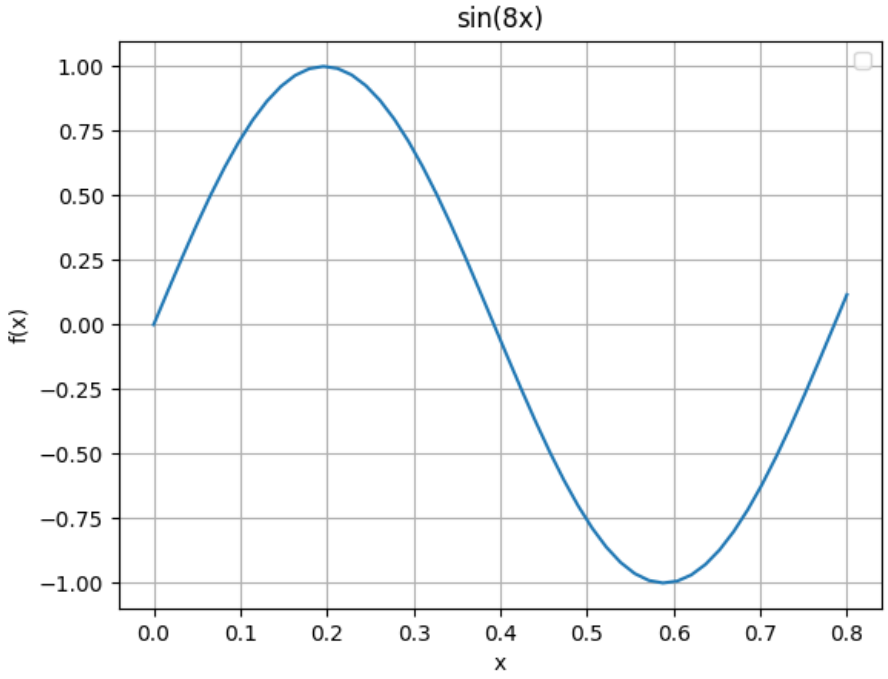
Funkcja oscylacyjna

$$f(x) = (x+1)^2 * cos((2*x+1)/(x-4.3)) \tag{4.19}$$

4.1. Przykład użycia biblioteki

4.1.1. Metoda prostokątów

Funkcja implementująca metodę prostokątów, przyjmuje 4 parametry. Pierwszym parametrem jest badana funkcja, drugim początek dolny przedział całkowania, trzecim górny przedział całkowania, a ostatnim, ilość podprzedziałów na które dzielimy odcinek $[a, b]$. Metoda zwraca przybliżoną wartość całki.



Rysunek 4.2: Wykres funkcji zdefiniowanej wzorem(5.24)

Tabela (5.1) przedstawia dane otrzymane w wyniku działania implementacji metody Prostokątów dla funkcji (5.23) na przedziale $[0, 0.8]$. Pierwszą kolumną tabeli jest obliczona wartość całki, drugą ilość podprzedziałów dla danej iteracji, a trzecią błąd metody wyrażony jako różnica wartości całki obliczonej analitycznie i wartości obliczonej numerycznie.

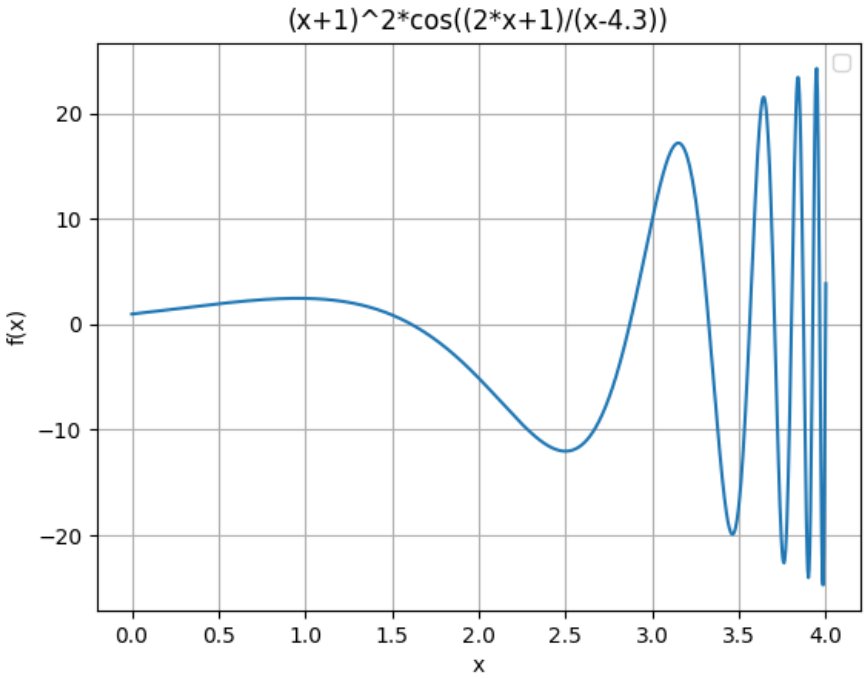
Jak widać w tabeli początkowo dla pojedyńczego podprzedziału błąd metody był bardzo duży i wynosił -8.08110. Wynika to z kształtu funkcji (5.23).

Do osiągnięcia założonej dokładności metoda prostokątów potrzebowała 31 podprzedziałów.

```
1 def f_rectI(f,a,b,n):
2     """
3     """
```

Listing 6: Kod w języku python implementujący metodę prostokątów

Funkcja wielomianowa



Rysunek 4.3: Wykres funkcji zdefiniowanej wzorem(5.25)

Tabela 4.1: Dane z iteracji w metodzie prostokątów dla funkcji wielomianowej

	Wartość całki	l.podp	Błąd
0	5.843200	1	-8.081100
1	11.635200	2	-2.289100
2	12.884780	3	-1.039520
3	13.335200	4	-0.589100
4	13.545974	5	-0.378326
5	13.661077	6	-0.263223
6	13.730687	7	-0.193613
7	13.775950	8	-0.148350
8	13.807020	9	-0.117280
9	13.829263	10	-0.095037
10	13.845731	11	-0.078569
11	13.858262	12	-0.066038
12	13.868017	13	-0.056283
13	13.875760	14	-0.048540
14	13.882008	15	-0.042292
15	13.887122	16	-0.037178
16	13.891361	17	-0.032939
17	13.894914	18	-0.029386

Tabela (5.2) przedstawia dane zwrócone z działania metody prostokątów dla funkcji (5.24) na przedziale $[0, 0.8]$. Początkowo wartość błędu obliczonej całki jest duża, z racji przyjętego kształtu funkcji oraz przyjętego przedziału całkowania. Obie wartości funkcji na krańcach przedziałów są dodatnie oraz leżą bardzo blisko osi odciętych, przez co prostokąt przybliżający funkcje nijak ma się do rzeczywistego kształtu funkcji. Wraz ze zwiększającą się ilością podprzedziałów dokładność przybliżenia znacząco rośnie, aż dla 16 podprzedziałów przyjmuję przyjmuję zakładaną dokładność.

Funkcja trygonometryczna

Tabela 4.2: Dane z iteracji w metodzie prostokątów dla funkcji trygonometrycznej

	Wartość całki	l.podp	Błąd
0	1.978716	1	1.734009
1	-1.293375	2	-1.538083
2	1.427055	3	1.182348
3	0.538234	4	0.293527
4	0.391699	5	0.146991
5	0.335697	6	0.090990
6	0.307385	7	0.062677
7	0.290809	8	0.046102
8	0.280172	9	0.035465
9	0.272899	10	0.028192
10	0.267689	11	0.022982
11	0.263820	12	0.019113
12	0.260863	13	0.016156
13	0.258550	14	0.013843
14	0.256705	15	0.011998
15	0.255209	16	0.010502
16	0.253978	17	0.009271

4.1.2. Metoda trapezów

Funkcja implementująca metodę trapezów, przyjmuje dokładnie te same parametry co metoda prostokątów. Pierwszym parametrem jest funkcja podcałkowa, kolejnymi dwoma kolejno górny i dolny przedział całkowania, a ostatnim ilość podprzedziałów, na które dzielimy przedział całkowania. Funkcja zwraca przybliżoną wartość całki.

```
1 def f_trapI(f, a, b, n):
2     """
3     """
```

Listing 7: Kod w języku python implementujący metodę trapezów

Funkcja wielomianowa

Tabela (5.3) przedstawia wyniki uzyskane z działania metody trapezów na funkcji (5.23).
Dla pojedyńczego podprzedziału błąd funkci jest bardzo duży i wynika, z tego, że wartość funkcji dla końcowych argumentów z przedziału gwałtownie rośnie, przez co prosta łącząca krańce przedziału całkowanai robi to z dużym nadmiarem. Po podzieleniu podprzedziału na 2, błąd maleje o 1 rząd wielkości, a po kolejnych 3 o kolejny rząd. Do otrzymania zadanej dokładności metoda trapezów potrzebuje 44 podprzedziałów.

Funkcja trygonometryczna

Tabela (5.4) przedstawia dane uzyskane z metody trapezów na funkcji podcałkowej (5.24). Największy błąd dla tej funkcji pojawia się, gdy ilość podprzedziałów wynosi 3, dzieje się tak dlatego, że wartości funkcji dla granic środkowego podprzedziału przymuja bardzo niskie wartości, przez co trapez przybliżający funkcję na tym przedziale pomija całkowice całą dodatnią cześć sinusiody.
Metoda trapezów przyjmuję wartość z zadaną dokładnością przy ilości podprzedziałów równej 23.

że przy takim podziale każdy prawy kraniec danego podprzedziału znajduje się pod osią odciętych oraz dla każdego punku granicznego wartość funkcji przyjmują niską wartość, co skutkuje tym, że trapezy przybliżające funkcję podcałkową

Tabela 4.3: Dane z iteracji w metodzie trapezów dla funkcji wielomianowej

	Wartość całki	l.podp	Błąd
0	31.315200	1	17.390900
1	18.579200	2	4.654900
2	16.018410	3	2.094110
3	15.107200	4	1.182900
4	14.682819	5	0.758519
*	*	*	*
39	13.936159	40	0.011859
40	13.935586	41	0.011286
41	13.935054	42	0.010754
42	13.934558	43	0.010258
43	13.934095	44	0.009795

Tabela 4.4: Dane z iteracji w metodzie trapezów dla funkcji trygonometrycznej

	Wartość całki	l.podp	Błąd
0	-0.287903	1	-0.532611
1	0.845407	2	0.600699
2	-1.269118	3	-1.513825
3	-0.223984	4	-0.468692
4	-0.011437	5	-0.256145
5	0.078969	6	-0.165739
6	0.127564	7	-0.117144
7	0.157125	8	-0.087583
8	0.176585	9	-0.068122
9	0.190131	10	-0.054577
10	0.199961	11	-0.044746
11	0.207333	12	-0.037375
12	0.213008	13	-0.031699
13	0.217474	14	-0.027233
14	0.221053	15	-0.023654
15	0.223967	16	-0.020740
16	0.226371	17	-0.018336
17	0.228379	18	-0.016329
18	0.230073	19	-0.014635

4.1.3. Metoda Simpsona

```
1 def f_simp2I(f,a,b,n):
2     """
3     """
```

Listing 8: Kod w języku python implementujący metodę prostokątów

Funkcja wielomianowa

Tabela (5.5) przedstawia wyniki uzyskanie z działania metody Simpsona dla funkcii podcałkowej (5.23). Metoda Simpsona potrzebowała 6 podprzedziałów by osiągnąć zadaną dokładność.

Tabela 4.5: Dane z iteracji w metodzie Simpsona dla funkci wielomianowej

	Wartość całki	l.podp	Błąd
0	14.333867	2	0.409567
1	13.949867	4	0.025567
2	13.929323	6	0.005023

Funkcja trygonometryczna

Tabela (5.6) przedstawia wyniki uzyskanie z działania metody Simpsona dla funkcii podcałkowej (5.24). Metoda Simpsona potrzebowała 12 podprzedziałów by osiągnąć zadaną dokładność.

Tabela 4.6: Dane z iteracji w metodzie Simpsona dla funkci trygonometrycznej

	Wartość całki	l.podp	Błąd
0	1.223177	2	0.978469
1	-0.580448	4	-0.825156
2	0.528331	6	0.283623
3	0.284161	8	0.039454
4	0.257320	10	0.012613
5	0.250121	12	0.005413

4.1.4. Metoda Gaussa Legendre'a

```
1 def f_gauss_legrande(f,a,b,n):
2     """
3     """
```

Listing 9: Kod w języku python implementujący metodę trapezów

Funkcja wielomianowa

Tabela (5.7) przedstawia wyniki uzyskanie z działania metody Gaussa Legendre'a dla funkcii podcałkowej (5.23). Funkcja , dla której liczona jest całka jest wielomianem stopnia 5, a metoda Gaussa Legendre'a daje wynik dokładny dla wielomianów stopnia $2n-1$, dlatego do uzyskania zadanej dokładności metoda potrzebowała 3 węzłów interpolacyjnych. Uzyskany błąd wynika z zaokrągleń.

Tabela 4.7: Dane z iteracji w metodzie Gauss'a Legendre'a dla funkcji wielomianowej

	Wartość całki	l.podp	Błąd
0	5.843200	1	-8.081100
1	13.651200	2	-0.273100
2	13.924267	3	-0.000033

funkcja trygonometryczna

Tabela (5.8) przedstawia wyniki uzyskanie z działania metody Gaussa Legendre'a dla funkcii podcałkowej (5.24). Całkowana funkcja nie jest wielomianem, więc uzyskanie dokładnej wartości przy użyciu metody Gaussa jest niemożliwe, lecz jej kształ sprawia, że daje się ona łatwo przybliżyć, przy użyciu wielomianów, co skutkuje, że do uzyskania zakładanej dokładności kwadratura Gaussa potrzebowałą 7 węzłów interpolacyjnych.

Tabela 4.8: Dane z iteracji w metodzie Gauss'a Legendre'a dla funkcji trygonometrycznej

	Wartość całki	l.podp	Błąd
0	1.978716	1	1.734009
1	-0.184912	2	-0.429619
2	1.974615	3	1.729907
3	-0.611561	4	-0.856269
4	0.456455	5	0.211748
5	0.212634	6	-0.032073
6	0.248022	7	0.003315

4.1.5. Metoda adaptacyjna trapezów

```
1 def f_adapt(f,ax,bx,tol):
2     """
3     """
```

Listing 10: Kod w języku python implementujący metodę trapezów

Funkcja wielomianowa

Tabela (5.9) przedstawia wyniki uzyskanie z działania metody Adaptacyjnej Trapezów dla funkcii podcałkowej (5.23). Pomimo tego, że bład nie jest mniejszy niż zakładana tolerancja, to obie te wartości zmniejszają się razem, oraz mają taki sam rząd wielkości.

Tabela 4.9: Dane z iteracji w metodzie Adaptacyjnej Trapezów dla funkcji wielomianowej

	Wartość całki	l.wywołań funkcji	Tol	Błąd
0	14.087879687500003	7	0.1	-0.16357968750000218
1	13.95071159667969	21	0.01	-0.026411596679690064
2	13.92913151855469	47	0.001	-0.004831518554690106
3	13.925455606651312	97	0.0001	-0.0011556066513112029
4	13.925455606651312	207	0.00001	-0.00026474552704236487
5	13.924337389287741	441	0.000001	-0,0000373892877405523
6	13.924280646049143	999	0.0000001	0,0000193539508579476
7	13.92426944676001	2207	0.00000001	0,000030553239991348846
8	13.924267226688606	4811	0.000000001	0,0000327733113945072
9	13.92426678139501	10345	0.0000000001	0,00003321860499028162
10	13.924266690629675	22017	0.00000000001	0,0000333093703250853
11	13.924266671787425	46421	0.000000000001	0,000033328212575511884
12	13.924266667844629	98387	0.0000000000001	0,0000333321553718235 46e

Funkcja trygonometryczna

Tabela (5.10) przedstawia wyniki uzyskanie z działania metody Adaptacyjnej Trapezów dla funkcii podcałkowej (5.24).

Tabela 4.10: Dane z iteracji w metodzie Adaptacyjnej Trapezów dla funkcji trygonometrycznej

	Wartość całki	l.podprzedziałów	Tol	Błąd
0	0.275097393643396	12	0.1	-0.030389958602972944
1	0.22651620332825226	26	0.01	0.018191231712170824
2	0.24356462655643027	54	0.001	0.0011428084839928132
3	0.24428699646689261	116	0.0001	0.0004204385735304683
4	0.24464360546255373	234	0.00001	0.00006382957786935095
5	0.24468621101711638	476	0.000001	0,000021224023306704
6	0.24470234198590066	964	0.0000001	0,00000509305452242592
7	0.2447062019886146	1952	0.00000001	0,000002330518084913 14
8	0.24470726279412686	5571	0.000000001	0,000001722462962261 41
9	0.24468621101711638	12406	0.0000000001	0,000000032682360706637 3
10	0.24470742830603642	26536	0.00000000001	0,000000006734386659434 93
11	0.24470743356253416	55622	0.000000000001	0,000000001477888922352 12
12	0.24470743470247994	115119	0.0000000000001	0,000000000337943145689578

4.1.6. Metoda Crude Monte Carlo

```
1 def f_crudeMonteC(f,a,b,n):
2     """
3     """
```

Listing 11: Kod w języku python implementujący metodę Crude Monte Carlo

Funkcja wielomianowa

W tabeli (5.11) przedstawiono wyniki uzyskanie z działania metody Crude Monte Carlo dla funkcii podcałkowej (5.23). Otrzymany błąd dla jednego podprzedziału może wydawać, się zaskakująco mały, lecz trzeba wziąć pod uwagę, że jest on średnim błędem dla 100 iteracji. Dla jednego podprzedziału wartości całki może się wahać od 1.2 do 61.5103. Wraz ze zwiększaniem ilości punktów, zakres otrzymywanych wyników ulega zmniejszeniu. w tabeli można również zauważyć, że wartości otrzymane dla 100 jak i 1000 punktów, są bardzo podobne i wynika to ze "szczęścia"z jakim punkty

zostaną wylosowane. Przy 1000 losowych punktów średni otrzymany błąd przybliżenia wynosi 0.01

Tabela 4.11: Dane z iteracji w metodzie Crude Monte Carlo dla funkcji wielomianowej

Row	Points	minValue	maxValue	AVG	Błąd
1	1	1.416807038	60.99560428	11.987235	13.91%
1	10	2.70581996	26.80458791	13.662372	1.88%
2	100	10.89037611	19.49690441	13.98955	0.47%
3	1000	12.37476196	14.94271099	13.941598	0.12%
4	10000	13.61746918	14.30824118	13.906282	0.13%
5	100000	13.76707602	14.03293431	13.922587	0.01%

Funkcja trygonometryczna

Tabela (5.12) przedstawia wyniki uzyskanie z działania metody Crude Monte Carlo dla funkcii podcałkowej (5.24). Na podstawie pierwszych 3 wierszy tabeli widać, że otrzymywany błąd wraz z dziecięsiokrotnym wzrostem ilości punktów maleje około 4-krotnie. Dalsze 1-krotne zwiększanie ilości użytych punktów daje około 7-krotne zmniejszenie błędu. Przy 10000 punktów błąd maleje o połowę, a użycie 100000 daje błąd przybliżenia całki na poziomie 0.12

Tabela 4.12: Dane z iteracji w metodzie Crude Monte Carlo dla funkcji trygonometrycznej

Row	Points	minValue	maxValue	AVG	Błąd
1	1	-1.999712843	1.999075162	0.438109	79.03%
1	10	-0.6383128719	1.133784472	0.190505	22.15%
2	100	-0.1079735697	0.5645576157	0.261066	6.68%
3	1000	0.1529912591	0.3583314798	0.246922	0.9%
4	10000	0.2148820935	0.2831131619	0.243623	0.44%
5	100000	0.2349722501	0.2594303218	0.244408	0.12%

4.1.7. Metoda Monte Carlo

```python
def f_MonteC(f,a,b,n):
    """
    """
```

Listing 12: Kod w języku python implementujący metodę Monte Carlo

Funkcja wielomianowa

Tabela (5.13) przedstawia wyniki uzyskanie z działania metody Monte Carlo dla funkcii podcałkowej (5.23).

Tabela 4.13: Dane z iteracji w metodzie Monte Carlo dla funkcji wielomianowej

Row	Points	minValue	maxValue	AVG	Błąd
1	10	1,853055156	26,60291651	11,033428	20,76%
2	100	6,629125753	18,74562867	12,657178	9,1%
3	1000	10,92932294	14,48626121	12,808812	8,01%
4	10000	12,17390193	13,47542511	12,777447	8,24%
5	100000	12,649035	12,98188584	12,806892	8,02%

Funkcja trygonometryczna

Tabela (5.14) przedstawia wyniki uzyskanie z działania metody Monte Carlo dla funkcii podcałkowej (5.24).

Tabela 4.14: Dane z iteracji w metodzie Monte Carlo dla funkcji trygonometrycznej

Row	Points	minValue	maxValue	AVG	Błąd
1	10	-1,928549934	1,816778457	0,211325	13,64%
2	100	-2,098836469	2,203746727	0,215333	12%
3	1000	-2,393655983	2,760053799	0,243658	0,43%
4	10000	-3,115304313	3,196864709	0,243453	0,51%
5	100000	-2,799281898	3,260480744	0,243275	0,59%

4.2. Porównanie zbieżności

Tabela (5.15) przedstawia porównanie metod całkowania opartych o interpolację wielomianową dla 2 podprzedziałów. Jak widzimy metodą dającą najgorszy wynik jest Metoda Trapezów, zwraca ona wynik 2-krotnie gorszy niż Metoda Prostokątów. Metoda Simpsona daje o 1 rząd wielkości błąd mniejszy niż metody trapezów i prostokątów. Metoda Gaussa Legendrea uzyskała błąd prawie o połowę niższy niż metoda Simpsona.

Tabela 4.15: Tabela porównująca metody całkowania oparte na interpolacji wielomianem dla 2 podprzedziałów

l.podp	Metoda	l.podp	Błąd
2	Prostokątów	11.6352	2.2891
2	Trapezów	18.579200	4.654900
2	Simpsona	14.333867	0.409567
2	Gaussa Legendre	13.651200	0.273100

W tabeli (5.16) umieszczone zostały wyniki z działania metod dla 4 podprzedziałów. Jak widać w tabeli stosunek różnicy pomiędzy metodą Prostokątów i Trapezów pozostał podobny. Błąd w metodzie Prostokątów i Trapezów wynosi $\frac{1}{2^2}$ w stosunku do błędu dla 2 podprzedziałów. W metodzie Simpsona wraz z podwojeniem podprzedziałów błąd zmniejsza się o $\frac{1}{2^4}$. Metoda Gaussa Legendrea daje wynik dokładny dla wielomianów stopnia $2n-1$, w związku z czym już dla 3 węzłów zwróciła ona wynik dokładny.

Porównanie metody Simpsona oraz Gaussa Legendra dla funkcji trygonometrycznej.

Jak widać w tabeli (5.17) początkowoa metoda Simpsona wykazuje dokładniejsze przybliżenie całki niż metoda Gaussa Legendre'a, przy 6 podprzedziałach zwraza błąd 10-krotnie mniejszy. Dopiero gdy liczba podprzedziałów wynosi 8 metoda Gaussa okazuje się być lepsza i zwracać błąd mniejszy o 2 rzędy wielkości. Przy 10 podprzedziałach kwadratura Gaussa zwraca wynik dokładny jeśli zaokrąglimy analitycznie obliczoną całkę do 6 miejsca po przecinkuo.

Porównanie metody Trapezów i Adaptacyjnej metody trapezów dla funkcji

Tabela 4.16: Tabela porównująca metody całkowania oparte na interpolacji wielomianem dla 4 podprzedziałów

l.podp	Metoda	l.podp	Błąd
4	Prostokątów	13.33520	0.58910
4	Trapezów	15.107200	1.182900
4	Simpsona	13.949867	0.025567
4	Gaussa Legendre	13.924267	0.000033

Tabela 4.17: Tabela porównująca Gaussa i Simpsona

l.podp	Metoda	I	Błąd
2	Simpsona	1.223177	9.784691e-01
4	Simpsona	-0.580448	8.251555e-01
6	Simpsona	0.528331	2.836232e-01
8	Simpsona	0.284161	3.945382e-02
10	Simpsona	0.257320	1.261270e-02
2	Gaussa Legendre	-0.184912	4.296190e-01
4	Gaussa Legendre	-0.611561	8.562686e-01
6	Gaussa Legendre	0.212634	3.207296e-02
8	Gaussa Legendre	0.244457	2.499462e-04
10	Gaussa Legendre	0.244707	6.560406e-07

Tabela (5.18) przedstawia porównanie Metody Trapezów i Adaptacyjnej Metody Trapezów. Porównanie zostało przeprowadzone na funkcji (5.23). Funkcja ta wraz ze wzrostem wartości argumentów przyjmuje coraz większą wartość, przez co żeby metoda mogła osiągnąć zadeklarowaną dokładność, dzieli prawą stronę równania na mniejsze podprzedziały, gdzie zwykła metoda trapezów dzieli funkcję na podprzedziały równo rozmieszczone wzdłuż całego przedziału całkowania. Stąd wynik uzyskany przez metodę trapezów jest 2-krotnie dokładniejszy dla ostatnich dwóch iteracji pokazanych w tabeli.

Metoda adaptacyjna radzi sobię lepiej od metody trapezów gdy całkowana funk-

Tabela 4.18: Tabela porównująca metodę trapezów i adaptacyjną metodę Trapezów

l.podp	Metoda	I	Błąd
4	Trapezów	15.1072	1.1829
6	Trapezów	14.451595	0.527295
8	Trapezów	14.2212	0.2969
10	Trapezów	14.158947	0.234647
12	Trapezów	14.056336	0.132036
16	Trapezów	13.998575	0.074275
4	Adaptacyjna	15.1072	1.1829
6	Adaptacyjna	14.236700	0,3124
8	Adaptacyjna	14,08788	0,16358
10	Adaptacyjna	14.017669	0,093369
12	Adaptacyjna	13.995822	0,071522
16	Adaptacyjna	13.95722	0,03292

cja przyjmuje różne tempo zmian wartości na zadanym przedziale. Przykładem takiej funkcji jest (**??**). Funkcja ta coraz bardziej oscyluje w pobliżu prawego punktu końcowego.

Wyniki działania obu metod całkowania tej funkcji prezentuje tebela (5.19).

Liczba podprzedziałów nie jest parametrem metody adaptacyjnej dlatego została dobrana na podstawie tego na jaką ilość podprzedziałów została podzielona funkcja przy zadanej tolerancji.

Dla 4 podprzedziałów widzimy, że zadana tolerancja była na tyle duża, że metoda adaptacyjna podzieliła przedział całkowania w ten sam sposób co metoda trapezów. Wraz ze wzrostem ilości podprzedziałów adaptacyjna metoda wykazuje znacznie lepszą dokładność w przybliżaniu całki. Dla 98 podprzedziałów błąd metody adaptacyjnej jest o 2 rzędy wielkości mniejszy niż przy zwykłej metodzie trapezów.

Porównanie metod Monte Carlo

Tabela (5.20) przedstawia wyniki z całkowania funkcji (5.24) metodami Crude Monte Carlo oraz Monte Carlo.

Tabela (5.20) przedstawia wyniki uzyskane z całkowania funkcji (5.23) meto-

Tabela 4.19: Tabela porównująca metodę trapezów i adaptacyjną metodę Trapezów

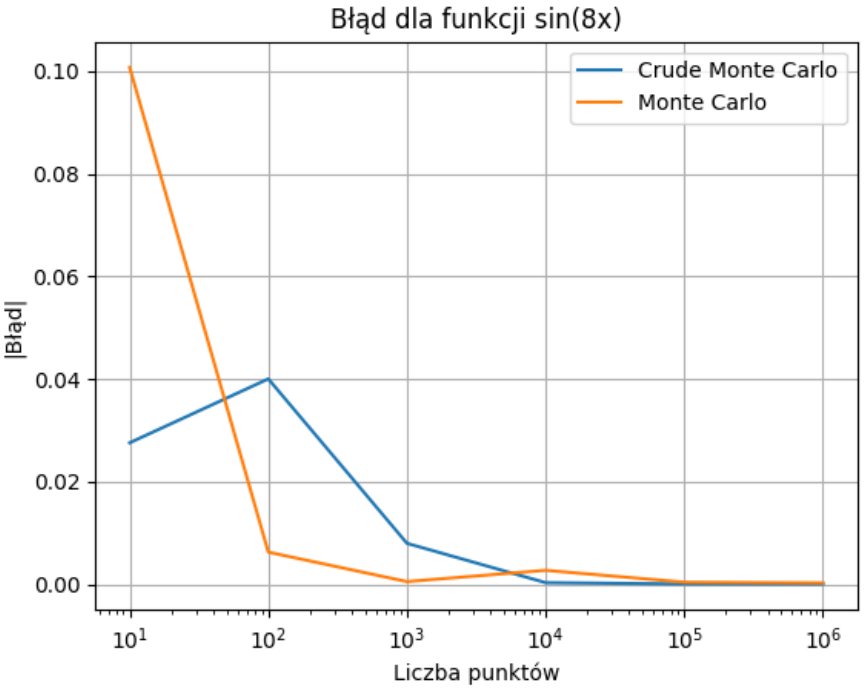
l.podp	Metoda	I	Błąd
4	Trapezów	9.731300	12.556630
8	Trapezów	-8.227000	-5.401670
10	Trapezów	7.068914	9.894244
18	Trapezów	5.822512	-2.997182
46	Trapezów	-2.827854	-0.002524
56	Trapezów	-1.467744	1.357586
98	Trapezów	2.295961	0.529369
4	Adaptacyjna	9.731300	-12.556630
8	Adaptacyjna	-9.439613	6.614283
10	Adaptacyjna	-11.1515886	8.326258
18	Adaptacyjna	-3.881971	1.056641
46	Adaptacyjna	-2.868944	0.043614
56	Adaptacyjna	-2.846306	0.020976
98	Adaptacyjna	-2.825574	0.000244

dami Crude Monte Carlo oraz Monte Carlo.

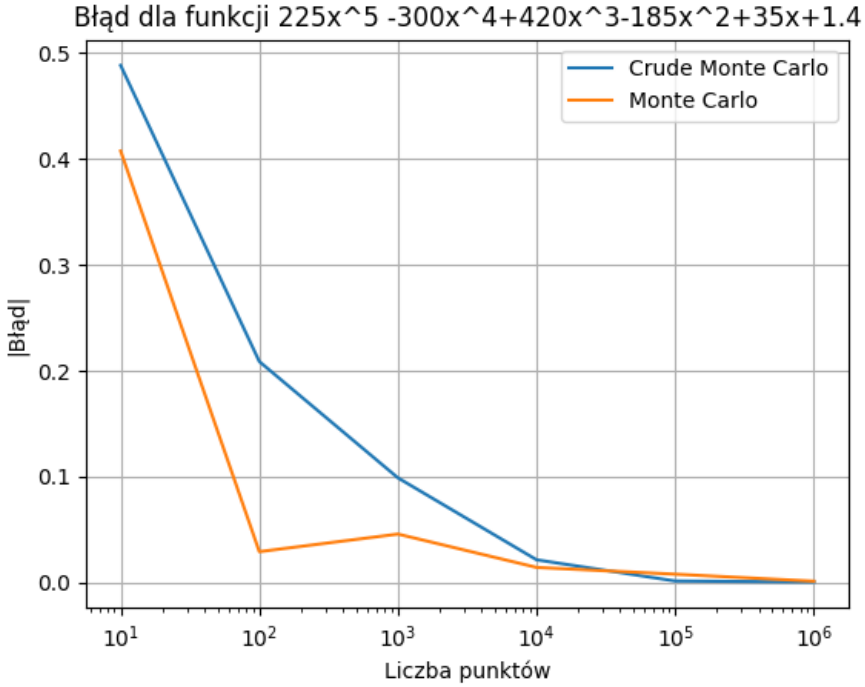
5. Podsumowanie i wnioski końcowe

$1 \div 3$ stron merytorycznie podsumowanie najważniejszych elementów pracy oraz wnioski wynikające z osiągniętego celu pracy. Proponowane zalecenia i modyfikacje oraz rozwiązania będące wynikiem realizowanej pracy.

Ostatni akapit podsumowania musi zawierać wykaz własnej pracy dyplomanta i zaczynać się od sformułowania: „Autor za własny wkład pracy uważa: . . . ”.



Rysunek 4.4: Wykres funkcji zdefiniowanej wzorem(5.23)



Rysunek 4.5: Wykres funkcji zdefiniowanej wzorem(5.23)

Tabela 4.20: Tabela porównująca metodę Crude Monte Carlo oraz Monte Carlo dla funkcji trgonometrycznej

l.podp	Metoda	I	Błąd
10	Crude	0.217109	0.027599
100	Crude	0.284756	0.040048
1000	Crude	0.25273	0.008022
10000	Crude	0.245071	0.000364
100000	Crude	0.244593	0.000115
1000000	Crude	0.244602	0.000106
10	Monte Carlo	0.144	0.100707
100	Monte Carlo	0.2384	0.006307
1000	Monte Carlo	0.24528	0.000573
10000	Monte Carlo	0.247472	0.002765
100000	Monte Carlo	0.245164	0.000457
1000000	Monte Carlo	0.245026	0.000318

Tabela 4.21: Tabela porównująca metodę Crude Monte Carlo oraz Monte Carlo dla funkcji wielomianowej

l.podp	Metoda	I	Błąd
10	Crude	13,43583	0,48847
100	Crude	14,133026	0,208726
1000	Crude	13,825492	0,098808
10000	Crude	13,945694	0,021394
100000	Crude	13,925673	0,001373
1000000	Crude	13,92384	0,00046
10	Monte Carlo	14,331923	0,407623
100	Monte Carlo	13,895199	0,029101
1000	Monte Carlo	13,878592	0,045708
10000	Monte Carlo	13,910085	0,014215
100000	Monte Carlo	13,93213	0,00783
1000000	Monte Carlo	13,92549	0,00119

Załączniki

Według potrzeb zawarte i uporządkowane uzupełnienie pracy o dowolny materiał źródłowy (wydruk programu komputerowego, dokumentacja konstrukcyjno-technologiczna, konstrukcja modelu – makiety – urządzenia, instrukcja obsługi urządzenia lub stanowiska laboratoryjnego, zestawienie wyników pomiarów i obliczeń, informacyjne materiały katalogowe itp.).

Literatura

[1] https://en.wikipedia.org/wiki/Numerical_integration

[2] https://www.python.org/doc/essays/blurb/

[3] https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[4] https://www.w3schools.com/python/numpy/numpy_intro.asp

[5] https://www.pythonlikeyoumeanit.com/Module3_IntroducingNumpy/VectorizedOperations.html

[6] https://en.wikipedia.org/wiki/Newton

[7] Willie Aboumrad, CME 108/MATH 114 Introduction to Scientific Computing

[8] https://www.cs.mcgill.ca/

[9] Abramovitz, Milton; I. Stegun (1964). Handbook of mathematical functions. National Bureau of Standards. ISBN 0-486-61272-4.

[10] Numerical Methods for Engineers, 6th Ed

[11] https://www.mimuw.edu.pl/ leszekp/dydaktyka/MO19L-g/adaptn.pdf

[12] https://www.math.usm.edu/lambers/mat460/fall09/lecture30.pdf

[13] https://home.agh.edu.pl/ zak/downloads/wyk5.pdf

[14] http://www.algorytm.org/procedury-numeryczne/calkowanie-numeryczne-metoda-monte-carlo-ii.html

[15] http://www.algorytm.org/procedury-numeryczne/calkowanie-numeryczne-metoda-monte-carlo-i.html

[16] http://weii.portal.prz.edu.pl/pl/materialy-do-pobrania. Dostęp 5.01.2015.

[17] Jakubczyk T., Klette A.: Pomiary w akustyce. WNT, Warszawa 1997.

[18] Barski S.: Modele transmitancji. Elektronika praktyczna, nr 7/2011, str. 15-18.

[19] Czujnik S200. Dokumentacja techniczno-ruchowa. Lumel, Zielona Góra, 2001.

[20] Pawluk K.: Jak pisać teksty techniczne poprawnie, Wiadomości Elektrotechniczne, Nr 12, 2001, str. 513-515.

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2022
Wydział Elektrotechniki i Informatyki

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

ANALIZA I INTERPRETACJA WYBRANYCH METOD CAŁKOWANIA NUMERYCZNEGO

Autor: Kamil Madej, nr albumu: 161876
Opiekun: (dr. inż) Mariusz Borkowski (prof. PRz)
Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po polsku

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2022
Faculty of Electrical and Computer Engineering

BSC THESIS ABSTRACT

TEMAT PRACY PO ANGIELSKU

Author: Kamil Madej, nr albumu: 161876
Supervisor: (academic degree) Imię i nazwisko opiekuna
Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po angielsku