Parametry: 4 2 5 1

Raport z Pracowni nr 3

Zadanie 1

1. Cel zadania

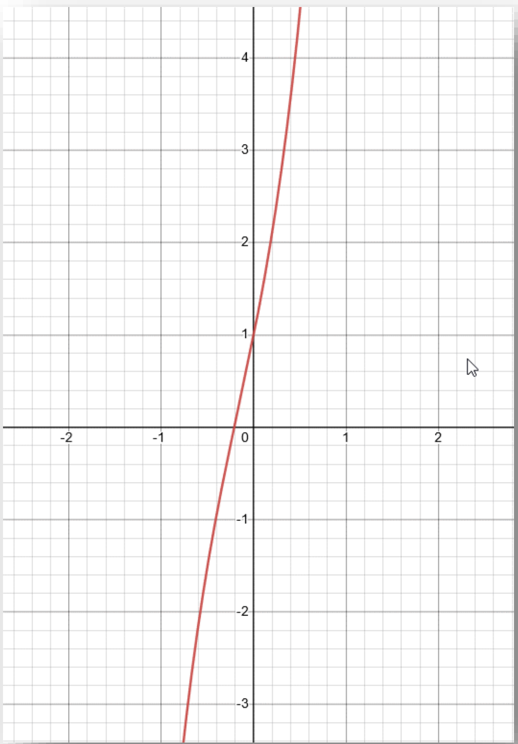
Celem zadania było porównanie efektywności metod stycznych i metody siecznych, za pomocą funkcji *iteruj_roznica.*

2. Metody

Zadanie rozwiązywano z wykorzystaniem metody stycznych oraz metody siecznych, z pomocą pliku *zadanie1.py*. Eksperymenty przeprowadzono za pomocą programu Visual Studio Code na laptopie z procesorem intel i5 w języku programowania python.

3. Przyjęte parametry:
- eps = $1.0e^{-15}$
- a = -0.35
- b = -0.19
4. Przebieg eksperymentu
    1. Badana funkcja

$$4x^3 + 2x^2 + 5x + 1$$



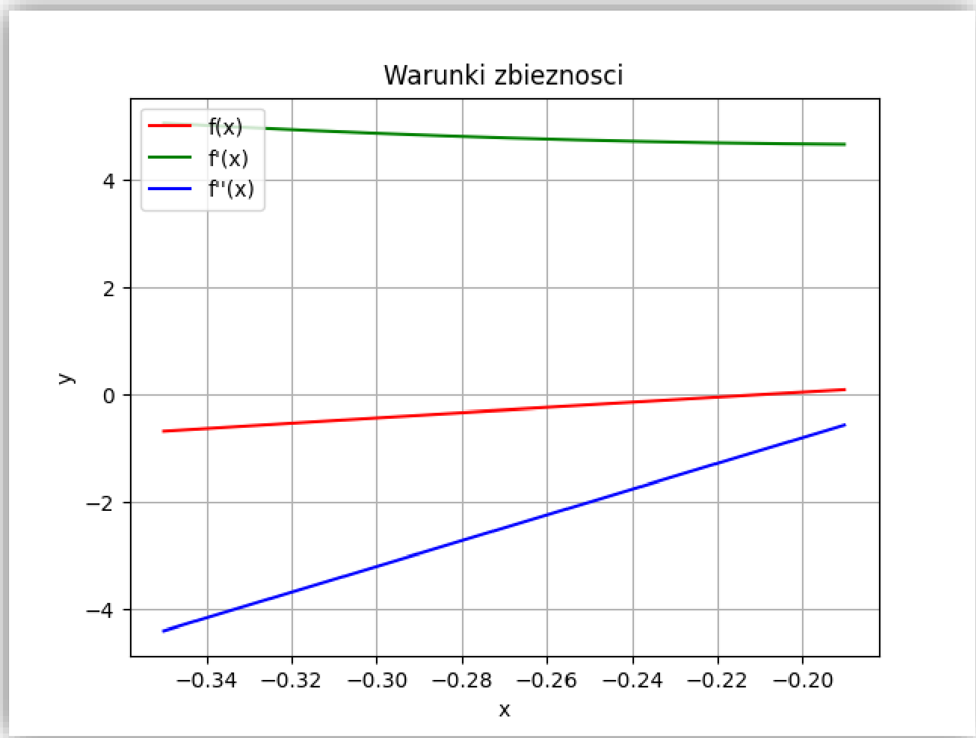
2. Lokalizacja pierwiastka

Na początku wybrałem przedział od *-2* do *1* lecz druga pochodna funkcji przecinała oś x. Następnie zmniejszyłem przedział do -0.5 do 0, lecz dalej druga pochodna przecinała oś x. Ostatecznie dobrałem parametry a = -0.35 oraz b = -0.19. Wartość funkcji w punkcie -0.35 przemnożona przez wartość funkcji w -0.19 daje ujemny iloczyn, co oznacza, że w danym przedziale istnieje dokładnie miejsce zerowe.

```
"""Porownanie metod przyblizonych"""

import wielomian, wykresy
import bisekcja, sieczne, styczne

class Zadanie1:

    def __init__(self):
        """Konstruktor klasy"""
        # dobor przedzialu
        # tworzymy wielomian wspisujac wspolczynniki a0, a1, a2, a3
        self.wsp = [1.0, 5.0, 2.0, 4.0]
        self.f = wielomian.Wielomian(self.wsp)
        # okreslamy krance przedzialu
        self.a = -0.35
        self.b = -0.19
        # okreslamy parametry eksperymentu
        self.n = 40
        self.epsilon = 1.0e-15

    def porownaj(self, war_stopu):
        """Metoda porownujaca metody przyblizone,
            parametr war_stopu okresla kryterium stopu:
            0 - zadana liczba iteracji
            1 - iteracje zatrzymywane, gdy dwa kolejne przyblizenia
                dostatecznie blisko siebie"""
        # metoda bisekcji
        # print("Metoda bisekcji")
        # test1 = bisekcja.Bisekcja(
        #     wiel = self.f,
        #     lewy_kp = self.a,
        #     prawy_kp = self.b)
        # if (war_stopu == 0):
        #     ans1 = test1.iteruj(iteracje = self.n, wyswietlaj = 1)
        # else:
        #     ans1 = test1.iteruj_roznica(eps = self.epsilon,
wyswietlaj = 1)
        # if ans1 > 0:
        #     test1.sprawdz_rozwiazanie()
        # metoda siecznych
        print("-"*30)
        print("Metoda siecznych")
        test2 = sieczne.Sieczne(
```

```
            wiel = self.f,
            lewy_kp = self.a,
            prawy_kp = self.b)
        if (war_stopu == 0):
            ans2 = test2.iteruj(iteracje = self.n, wyswietlaj = 1)
        else:
            ans2 = test2.iteruj_roznica(eps = self.epsilon, wyswietlaj
= 1)
        if ans2 > 0:
            test2.sprawdz_rozwiazanie()
        # metoda stycznych
        print("-"*30)
        print("Metoda stycznych")
        test3 = styczne.Styczne(
            wiel = self.f,
            lewy_kp = self.a,
            prawy_kp = self.b)
        if (war_stopu == 0):
            ans3 = test3.iteruj(iteracje = self.n, wyswietlaj = 1)
        else:
            ans3 = test3.iteruj_roznica(eps = self.epsilon, wyswietlaj
= 1)
        if ans3 > 0:
            test3.sprawdz_rozwiazanie()
        # obrazujemy zbieznosc tych metod na wykresie
        wykres = wykresy.Wykresy()
        # sprawdzamy, czy metody siecznych i stycznych zostaly
zastosowane
        if ans2 == 0:
            test2.x = []
        if ans3 == 0:
            test3.x = []
        wykres.badaj_zbieznosc(
            tytul = "Metody przyblizone",
            opis_OY = "Przyblizenia rozwiazania",
            # dane1 = test1.x,
            # opis1 = "Metoda bisekcji",
            dane1 = test2.x,
            opis1 = "Metoda siecznych",
            dane2 = test3.x,
            opis2 = "Metoda stycznych"
        )

    def warunki_zbieznosci(self):
        """Metoda obrazujaca warunki zbieznosci"""
        wykres1 = wykresy.Wykresy()
        wykres1.wykres_wielomianow(
            f = self.f,
            a = self.a,
            b = self.b,
            opis = "Warunki zbieznosci")
```

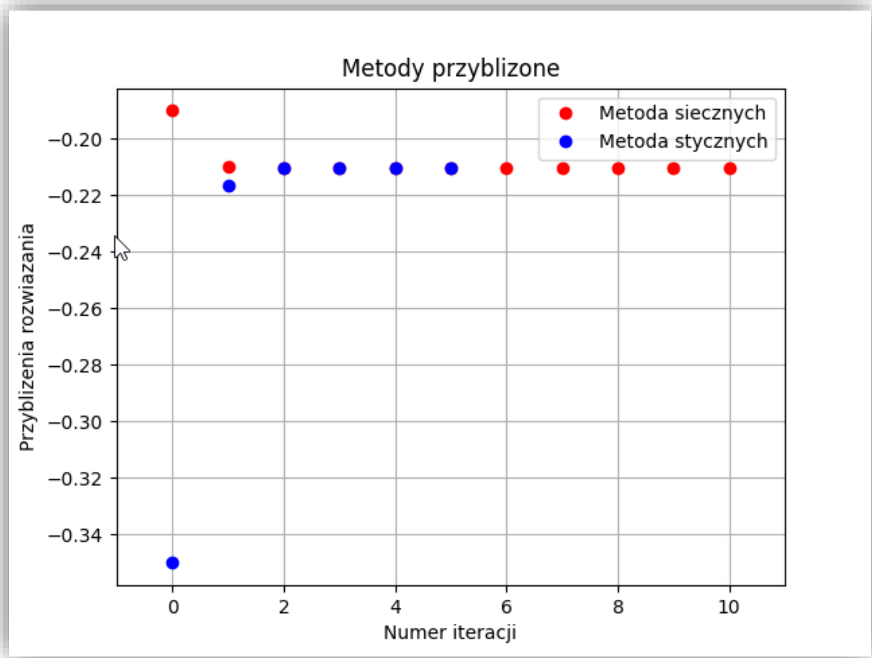
3. Warunki zbieżności



Wykres przedstawia warunki zbieżności

Założenia są spełnione ponieważ funkcja f(x) przecina oś x, f′(x) znajduje się nad osią x w określonym przedziale oraz f″(x) znajduje się pod osią x w określonym przedziale. F(x) jest dwukrotnie różniczkowalna oraz f′(x) stałego znaku i f″(x) też jest stałego znaku.

4. Przebieg iteracji



Wykres przedstawiający porównanie zbieżności metody siecznych oraz stycznych.

```
Metoda siecznych
0.   -0.19
1.   -0.2096589494647747
2.   -0.21022820061229217
3.   -0.21024594505182265
4.   -0.21024649930392308
5.   -0.2102465166172294
6.   -0.21024651715805037
7.   -0.2102465171749442
8.   -0.2102465171754719
9.   -0.2102465171754884
10.      -0.2102465171754889
Niedokładność rozwiązania: 7.632783294297951e-17
------------------------------
Metoda stycznych
0.   -0.35
1.   -0.21656804733727808
2.   -0.2102513972011013
3.   -0.21024651717814488
4.   -0.21024651717548892
5.   -0.21024651717548895
Niedokładność rozwiązania: 1.1796119636642288e-16
```

Metoda siecznych (czerwona) kończy się po 10 iteracjach, a stycznych (niebieska) po 5 iteracjach.

5. Wnioski

Miejsce zerowe wynosi około *-0.2102465171754889*. Szybszą metodą jest metoda stycznych, ponieważ zachodzi najszybsza zbieżność, co oznacza, że gorszą metodą jest metoda siecznych.

Zadanie 2

1. Cel zadania

Celem zadania było porównanie zbieżności metody parabol oraz metody 3/8 Newtona z wykorzystaniem funkcja *iteruj_roznica* z pomocą pliku z pomocą pliku *zadanie2.py*.

2. Metody

Zadanie rozwiązywano z wykorzystaniem dwóch metod 3/8 Newtona oraz metod parabol. Eksperymenty przeprowadzono za pomocą programu Visual Studio Code na laptopie z procesorem intel i5 w języku programowania python.

3. Przyjęte parametry:

- eps = $1.0e^{-6}$
- a = -2
- b = 4

4. Przebieg eksperymentu

```
"""Przyklad zastosowania metod przyblizonych"""

import funkcja, wykresy
import metodatrapezow, metodaparabol, metodanewtona

class Zadanie2:

    def __init__(self):
        """Konstruktor klasy"""
        # okreslamy funkcje
        self.f = funkcja.Funkcja()
        # okreslamy krance przedzialu
        self.a = -2.0
        self.b = 4.0
        # okreslamy parametry eksperymentu
        self.n = 100
        self.epsilon = 1.0e-6

    def porownaj(self, war_stopu):
        """Metoda porownujaca metody calkowania,
            parametr war_stopu okresla kryterium stopu:
            0 - zadana maksymalna liczba podprzedzialow
            1 - iteracje zatrzymywane, gdy dwa kolejne przyblizenia
                dostatecznie blisko siebie"""
        # metoda trapezow
        # print("Metoda trapezow")
        # test1 = metodatrapezow.MetodaTrapezow(
        #     funkcja = self.f,
        #     lewy_kp = self.a,
        #     prawy_kp = self.b)
        # if (war_stopu == 0):
        #     test1.iteruj(iteracje = self.n, wyswietlaj = 1)
        # else:
        #     test1.iteruj_roznica(eps = self.epsilon, wyswietlaj = 1)
        # # metoda parabol
        # print("-"*30)
        print("Metoda parabol")
        test2 = metodaparabol.MetodaParabol(
            funkcja = self.f,
            lewy_kp = self.a,
            prawy_kp = self.b)
        test2.iteruj_roznica(eps = self.epsilon, wyswietlaj = 1)
        # metoda 3/8 Newtona
        print("-"*30)
        print("Metoda 3/8 Newtona")
        test3 = metodanewtona.MetodaNewtona(
            funkcja = self.f,
            lewy_kp = self.a,
```

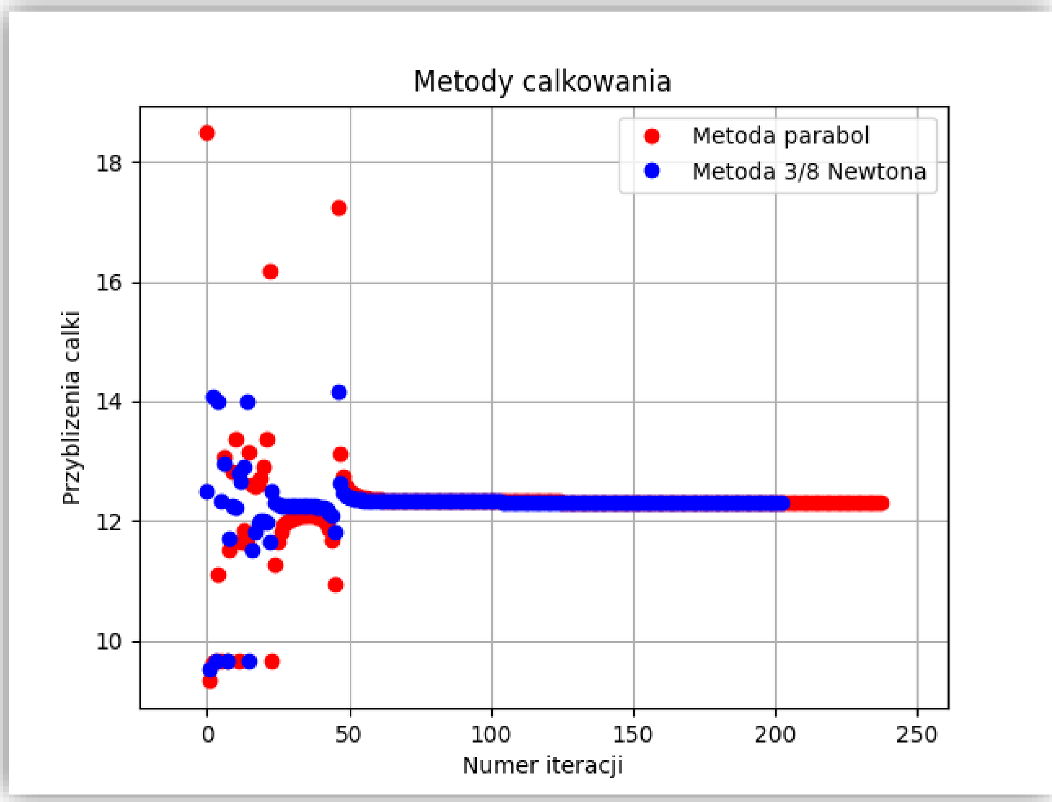
```
            prawy_kp = self.b)
        test3.iteruj_roznica(eps = self.epsilon, wyswietlaj = 1)
        # obrazujemy zbieznosc tych metod na wykresie
        wykres1 = wykresy.Wykresy()
        wykres1.badaj_zbieznosc(
            tytul = "Metody calkowania",
            opis_OY = "Przyblizenia calki",
            # dane1 = test1.calki,
            # opis1 = "Metoda trapezow",
            dane1 = test2.calki,
            opis1 = "Metoda parabol",
            dane2 = test3.calki,
            opis2 = "Metoda 3/8 Newtona"
        )
```

1. Całkowana funkcja oraz granice całkowania

$$4\sin(49 * x + 2) + \sqrt{1 + x^2}$$

$$x \in [\mathbf{-2}, \mathbf{4}]$$

2. Przebieg iteracji



Wykres przedstawia metody całkowania 2 metod (metody paraboli oraz metody 3/8 newtona)

```
Metoda parabol
1.   18.48702931770931
2.   9.320798728194436
3.   9.639012038388891
4.   9.65617516466805
5.   11.11529462184493
6.   9.668700061376539
7.   13.061179243595541
8.   9.670173604101185
9.   11.51432183817937
10.      12.817658885105145
11.      13.371248389460016
12.      9.670793588827332
13.      11.642493879600138
14.      11.847231338959192
15.      11.61607414172051
16.      13.144076216281968
17.      12.603929541302648
18.      12.574134887045696
19.      12.61779712477491
20.      12.71442539658273
…
…
225.     12.318169032897684
226.     12.318167751188573
```

```
227.     12.318166498400325
228.     12.318165273749448
229.     12.318164076477158
230.     12.318162905848602
231.     12.318161761151943
232.     12.318160641697512
233.     12.3181595468171
234.     12.318158475863079
235.     12.318157428207778
236.     12.318156403242776
237.     12.318155400378087
238.     12.318154419041733
------------------------------
Metoda 3/8 Newtona
1.   12.487420211860034
2.   9.521612461820387
3.   14.081338917699002
4.   9.66454391147332
5.   13.999850500552293
6.   12.331886401789639
7.   12.968209280899366
8.   9.670601646753767
9.   11.696188062441154
10.      12.25896631141678
11.      12.230811468480818
12.      12.785957280638168
13.      12.671369066365251
14.      12.904712243389783
15.      13.99215801173019
16.      9.670920633723632
17.      11.509749350014427
18.      11.815638034486
19.      11.937263630536922
20.      11.995843099945839
21.      12.017415601801257
…
…
192.     12.318158257365843
193.     12.3181569778098
194.     12.318155732248638
195.     12.318154519597304
196.     12.318153338811282
197.     12.31815218888476
198.     12.318151068849025
199.     12.318149977770801
200.     12.318148914750857
201.     12.318147878922678
202.     12.318146869450812
203.     12.318145885529896
```

W metodzie parabol, metoda stabilizuje się całkowicie po około 50 iteracjach (około 20 iteracji wyniki są bardzo zbliżone lecz czasami bardzo odbiegają od prawdziwego rozwiązania)

W metodzie 3/8 Newtona metoda stabilizuje się całkowicie po około 50 iteracjach (około 20 iteracji wyniki są bardzo zbliżone lecz czasami bardzo odbiegają od prawdziwego rozwiązania).

3. Wnioski

Metoda 3/8 Newtona lepiej poradziła sobie z policzeniem tej całki, ponieważ szybciej osiągnęła wcześniej podany błąd *eps* (wykonała mniej iteracji). Wartość całki wyniosła *12.318145885529896*