Laboratorium 3 - Metoda bisekcji, siecznych i Newtona

Miłosz Sawicki

31 października 2019

1 Instrukcja oraz działanie programu

Projekt jest w całości napisany w języku Python3. Wszystkie pliki należy umieścić w jednym katalogu. Uruchomienie skryptu main.py wykona obliczenia dla trzech metod w zadanych przedziałach dla wybranej funkcji, jej pochodnej, oraz epsilona. Następnie program wyświetli wyniki i przedstawi wykres ilustrujący zestawienie kolejnych iteracji dla każdej z metod.

Możliwe jest również uruchomienie samego skryptu z wybraną metodą. Jeżeli użytkownik zdecyduje się na uruchomienie jednego ze skryptów bisekcja.py, newtona.py lub sieczne.py to w konsoli zostanie wyświetlony wynik końcowy, wraz z kolejnymi iteracjami dążącymi do wyniku. W każdym ze skryptów parametry są zostawione do zmiany według preferencji.

W pliku funkcje.py znajdują się definicje przykładowych funkcji wraz z ich pochodnymi. Na ich podstawie można wykonać zaprezentowane metody. Plik wykres.py jest w całości modułem rysującym wykres z zebranych danych. Wykorzystana została biblioteka matplotlib oraz numpy.

2 Opis wykorzystanych metod

2.1 Metoda bisekcji

Metoda bisekcji (metoda równego podziału) polega na szukaniu miejsc zerowych funkcji, która spełnia założenia:

1. funkcja $f(x)$ jest ciągła na przedziale $[a, b]$
2. funkcja przyjmuje różne znaki na końcach przedziału, tzn. $f(a)f(b) < 0$

Algorytm rozpoczyna się od sprawdzenia różności znaków oraz wyznaczenia punktu x_0, czyli punktu środkowego przedziału. Następnie obliczamy wartość funkcji w tym punkcie. Jeśli jest ona dostatecznie bliska zeru, to jest to miejsce zerowe funkcji. W przeciwnym razie zmieniamy przedział $[a, b]$ na $[a, x_0]$, jeśli iloczyn $f_{x_0}\ f(a) < 0$. Gdy iloczyn ten jest dodatni, nowym przedziałem jest $[x_0, b]$. Program powtarza tę czynność do momentu osięgnięcia żądanej dokładności.

2.2 Metoda Newtona

Metoda Newtona pozwala wyliczyć punkt przecięcia stycznej do wykresu funkcji w punkcie x_{i1} z osią OX. Do wyznaczenia kolejnego przybliżenia pierwiastka xi potrzebujemy tylko jednego punktu, który został wyznaczony w poprzednim obiegu (w metodzie siecznych potrzebne były dwa punkty). Zaletą metody Newtona jest bardzo szybka zbieżność. Wadą – występowanie we wzorze pochodnej, której obliczenie może być trudne dla niektórych funkcji. Jednakże metodę Newtona najczęściej stosuje się do wielomianów, których pochodne są bardzo proste i liczy się je algorytmicznie. Zasada metody Newtona jest następująca: Obliczenia rozpoczynamy od punktu x_0 leżącego dostatecznie blisko poszukiwanego pierwiastka funkcji. W przedziale pomiędzy punktem x_0 a docelowym pierwiastkiem funkcja musi posiadać niezerową pierwszą pochodną. Pożądane jest również, aby w punkcie xo druga pochodna miała ten sam znak, co funkcja $f(x)$. W przeciwnym razie metoda Newtona, zamiast zbliżać się do punktu pierwiastka ucieknie od niego. Obliczamy nowy punkt xo zgodnie ze wzorem i sprawdzamy, czy wartość funkcji w tym punkcie jest dostatecznie bliska 0. Jeśli tak, kończymy. W przeciwnym razie wyznaczony kolejny punkt xo wykorzystując ostatnio wyliczony. Działania te prowadzimy dotąd, aż zbliżymy się dostatecznie do pierwiastka funkcji - różnica pomiędzy dwoma kolejno wyznaczonymi pierwiastkami będzie dostatecznie mała.

2.3 Metoda siecznych

Metoda siecznych to algorytm interpolacji liniowej. W literaturze polskojęzycznej nazywana czasem bywa metodą cięciw. Polega na przyjęciu, że funkcja ciągła na dostatecznie małym odcinku w przybliżeniu zmienia się w sposób liniowy. Możemy wtedy na odcinku [a, b] krzywą $y = f(x)$ zastąpić sieczną. Za przybliżoną wartość pierwiastka przyjmujemy punkt przecięcia siecznej z osią OX. Metodę siecznych dla funkcji $f(x)$, mającej pierwiastek w przedziale $[a, b]$ można zapisać następującym wzorem iteracyjnym:

$$x_0 = a$$

$$x_1 = b$$

$$x_n + 1 = \frac{f(x_n)x_n - 1 - f(x_n - 1)x_n}{f(x_n) - f(xn - 1)}$$

Metoda siecznych ma tę zaletę, że do wykonania interpolacji za jej pomocą niepotrzebna jest znajomość pochodnej danej funkcji, gdyż przybliżamy ją za pomocą powyższego wzoru. Aby metoda się powiodła dla każdego n musi zachodzić $f(xn)f(xn - 1) < 0$, gdyż tylko wtedy sieczna przechodząca przez punkty $(xn, f(xn))$ i $(xn - 1, f(xn - 1))$ przecina oś OX. Metoda ta nie zawsze jest zbieżna.

3 Analiza wyników

3.1 Przykład 1

W przykładzie wykorzystano funkcje:

$$f(x) = x^3 + 1$$

razem z jej pochodną:

$$\frac{d}{dx}f(x) = 3x^2$$

dla punktu początkowego $x_0 = 1$ dla metody Newtona i w przedziale $[-2, 4]$ dla metody bisekcji i siecznych. Maksymalna ilość iteracji została ustawiona na 10, zmienna precyzji $epsilon = 0.000000000001$.

3.1.1 Metoda bisekcji

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 bisekcja.py
wynik: -0.998046875
x1 = 1.0
x2 = -0.5
x3 = -1.25
x4 = -0.875
x5 = -1.0625
x6 = -0.96875
x7 = -1.015625
x8 = -0.9921875
x9 = -1.00390625
x10 = -0.998046875
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

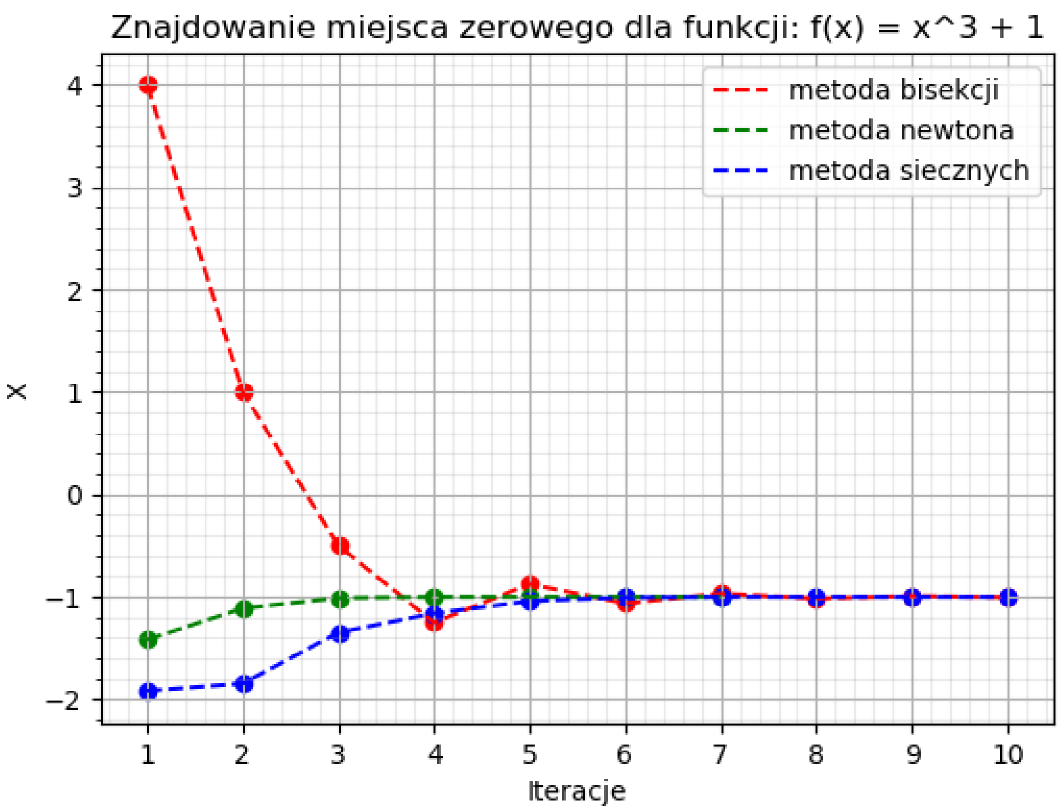
3.1.2 Metoda Newtona

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 newtona.py
wynik: -1.00000000000000
x1 = 1
x2 = 0.333333333333333
x3 = -2.77777777777778
x4 = -1.89505185185185
x5 = -1.35618682965904
x6 = -1.08535856331203
x7 = -1.00653708245686
x8 = -1.00004236399751
x9 = -1.00000000179461
x10 = -1.00000000000000
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

3.1.3 Metoda siecznych

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 sieczne.py
wynik: -0.9999988579621061
x1 = -0.25
x2 = -0.5263157894736843
x3 = -2.339581036383681
x4 = -0.6486597305897963
x5 = -0.7467535225489226
x6 = -1.1457035199072068
x7 = -0.9608455860177404
x8 = -0.9946885733629205
x9 = -1.0002142853428495
x10 = -0.9999988579621061
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

3.1.4 Wykres



3.2 Przykład 2

W przykładzie wykorzystano funkcje:

$$f(x) = x^5 sin(x)$$

razem z pochodną:

$$\frac{d}{dx} f(x) = x^4(5sin(x) + cos(x))$$

dla punktu początkowego $x_0 = 3$ dla metody Newtona i w przedziale $[3, 4]$ dla metody bisekcji i siecznych. Maksymalna ilość iteracji została ustawiona na 20, zmienna precyzji $epsilon = 0.000000000001$.

3.2.1 Metoda bisekcji

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 bisekcja.py
wynik: 3.1415929794311523
x1 = 3.5
x2 = 3.25
x3 = 3.125
x4 = 3.1875
x5 = 3.15625
x6 = 3.140625
x7 = 3.1484375
x8 = 3.14453125
x9 = 3.142578125
x10 = 3.1416015625
x11 = 3.14111328125
x12 = 3.141357421875
x13 = 3.1414794921875
x14 = 3.14154052734375
x15 = 3.141571044921875
x16 = 3.1415863037109375
x17 = 3.1415939331054688
x18 = 3.141590118408203
x19 = 3.141592025756836
x20 = 3.1415929794311523
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

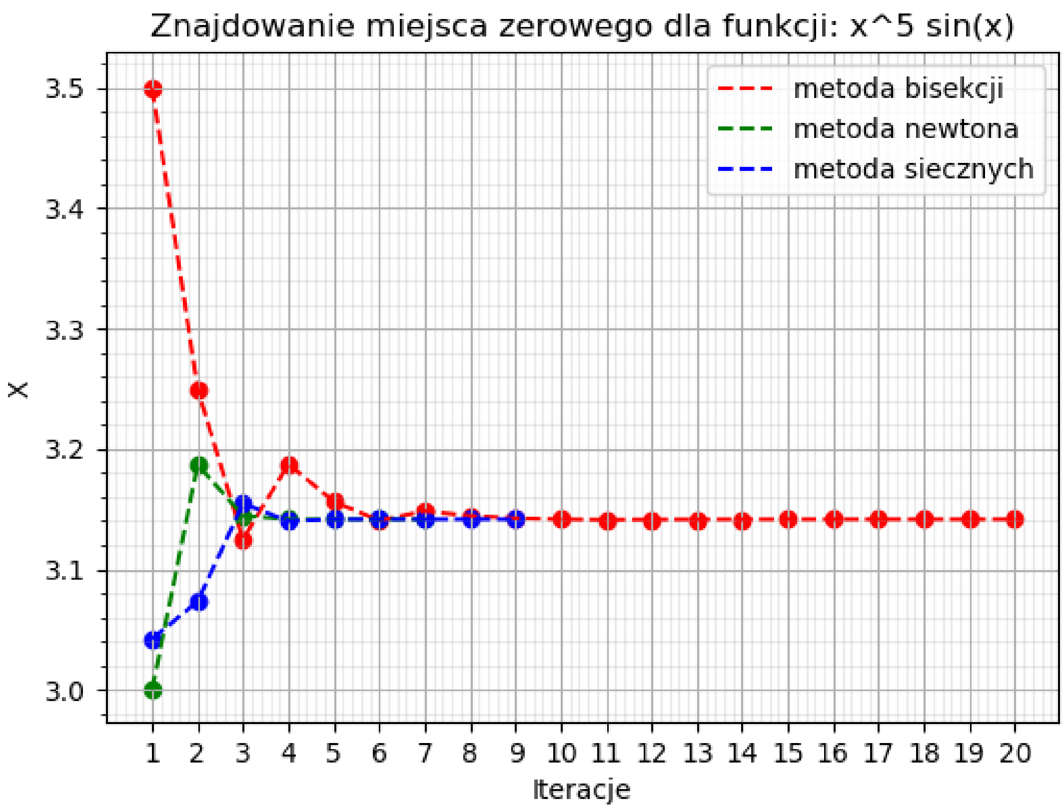
3.2.2 Metoda Newtona

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 newtona.py
wynik: 3.141592653589793
x1 = 3
x2 = 3.1869653066332693
x3 = 3.1445807009033837
x4 = 3.141606774180371
x5 = 3.1415926539071246
x6 = 3.141592653589793
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

3.2.3 Metoda siecznych

```
(base) milosz@deb:~/metody_numeryczne/zadanie3$ python3 sieczne.py
wynik: 3.141592653589793
x1 = 3.0423748241835584
x2 = 3.0732507691206785
x3 = 3.15469209027578
x4 = 3.1400816460177365
x5 = 3.141561535149803
x6 = 3.141592728547211
x7 = 3.1415926535860805
x8 = 3.141592653589793
(base) milosz@deb:~/metody_numeryczne/zadanie3$
```

3.2.4 Wykres



4 Wnioski

- Z przykładu drugiego wynika, że metoda Newtona jest najszybszą metodą znajdowania miejsca zerowego. Dla tej metody wynik osiągnął wyznaczoną precyzje już przy szóstej iteracji.
- Przy bardziej skomplikowanych funkcjach dla metody Newtona mogą wystąpić problemy wydajnościowe związane z wyliczaniem pochodnych. Metoda siecznych ma tą zaletę, że pochodna jest przybliżana za pomocą wzoru.
- Metoda bisekcji jest wyraźnie najmniej efektywna i zbiega dużo wolniej w stosunku do metody Newtona i siecznych.
- Przy pracy z funkcjami posiadającymi więcej niż jedno miejsce zerowe, należy mieć na uwadze wybór odpowiedniego przedziału w otoczeniu szukanego miejsca zerowego.