Metody numeryczne

Nikodem Kocjan

09.04.2024

Wyznaczanie pierwiastków równania nieliniowego metodą siecznych

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia było wyznaczenie pierwiastków równania nieliniowego metodą siecznych:

$$f(x) = x^4 - 7.899x^3 + 23.281114x^2 + 14.73866033 - 30.33468152x$$

2 Wstęp teorytyczny

2.1 Metoda siecznych

Metoda siecznych jest iteracyjną metodą numeryczną wyznaczania przybliżonych wartości pierwiastków rzeczywistych równań nieliniowych. Stanowi alternatywę dla metod takich jak metoda bisekcji czy metoda Newtona-Raphsona. Metoda siecznych polega na przybliżaniu funkcji lokalnie poprzez linie prostą, która przecina oś OX w punkcie, który jest następnym przybliżeniem pierwiastka równania.

Załóżmy, że mamy funkcję $f\colon \mathbb{R} \to \mathbb{R}$ oraz chcemy znaleźć taki punkt $x \in \mathbb{R}$, dla którego $f(x) = 0$. Wybieramy dwa początkowe punkty x_0 i x_1, które są naszymi pierwszymi przybliżeniami rozwiązania i obliczamy kolejne przybliżenia według wzoru:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}. \tag{1}$$

Iteracyjne stosowanie powyższego wzoru pozwala, przy spełnieniu pewnych warunków, na szybkie zbliżanie się do szukanego pierwiastka funkcji. Kryterium zakończenia iteracji zazwyczaj stanowi osiągnięcie zadanej dokładności, co można wyrazić jako:

$$\epsilon_{i+1} = |x_{i+1} - x_i| < 10^{-6}. \tag{2}$$

W pracy zostanie przedstawiona metoda siecznych w ujęciu klasycznym oraz jej modyfikacja, która pozwala na wyznaczanie pierwiastków o krotności parzystej.

2.2 Modyfikowana metoda siecznych

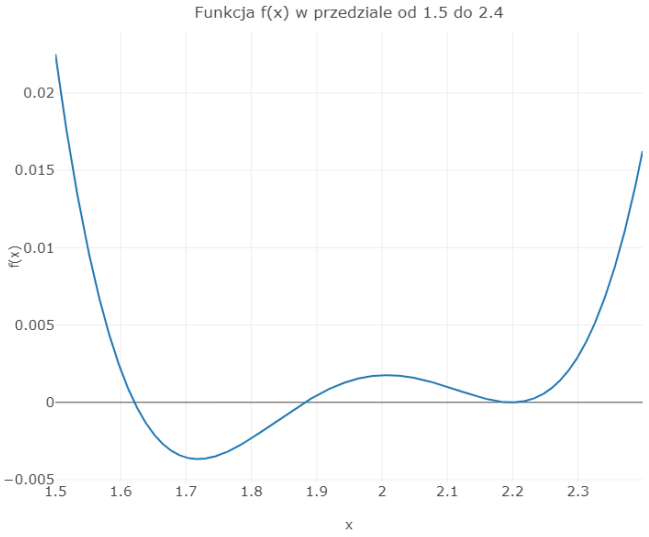
Modyfikacja metody siecznych stosowana jest w przypadkach, gdy szukany pierwiastek jest pierwiastkiem wielokrotnym, czyli ma krotność parzystą. Metoda klasyczna może w takich przypadkach wykazywać wolną zbieżność, dlatego stosuje się modyfikację polegającą na podzieleniu wartości funkcji przez jej pierwszą pochodną. Nowy wzór iteracyjny przyjmuje postać:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \cdot \frac{x_n - x_{n-1}}{f(x_n)/f'(x_n) - f(x_{n-1})/f'(x_{n-1})}. \tag{3}$$

Taka zmiana pozwala na przyspieszenie zbieżności metody w przypadku pierwiastków wielokrotnych.

2.3 Wykres funkcji

Wykres funkcji podanej w poleceniu zadania, w granicach od $x = 1.5$ do $x = 2.4$, prezentuje się następująco:



Rysunek 1: Wartości własne

Można odczytać z niego, że funkcja posiada 3 miejsca zerowe w podanym zakresie (w tym jedno podwójne).

3 Deklaracja zmiennych oraz implementacja bibliotek

Opisane zagadnienie rozwiązano za pomocą programu Visual Studio Code w języku c++
Wykorzystane zostały biblioteki
<iostream> - standardowa biblioteka c++
<fstream> - operacje na plikach
<cmath> - funkcje matematyczne

Zadeklarowane stałe, macierze oraz wektory użyte w programie to:

```
double x0; //x_k-1
double x1; //x_k
double x2; //x_k+1

double epsilon = 1e-6;  //przyblizenie
double blad; // blad obliczany dla kazdej iter
double delta = 0.1; //delta x

//Stworzenie 4 plikow z tabelkami
std::ofstream outFile1("miejsce_zerowe_1.csv");
std::ofstream outFile2("miejsce_zerowe_2.csv");
std::ofstream outFile3("miejsce_zerowe_3.csv");
std::ofstream outFile_mod("miejsce_zerowe_mod.csv");

//Nadanie nazw kolumn dla plikow
outFile1 << "Iteracja,x,blad,f(x)\n";
outFile2 << "Iteracja,x,blad,f(x)\n";
outFile3 << "Iteracja,x,blad,f(x)\n";
outFile_mod << "Iteracja,x,blad,u(x)\n";
```

Fragment kodu 1: Deklaracja stalych

4 Opis działania

4.1 Krótki opis programu

Program składa się z funkcji main oraz 3 funkcji.

- double f(double x)
- double df(double x, double delta)
- double u(double x, double delta)

Główna funkcja main odpowiada za deklarowanie stałych oraz zmiennych, zapis rezultatów do pliku w formacie csv, wypisanie rezultatów w konsoli. Składa się z jednej głównej pętli *do while*. Na początku ustawia się x0, oraz x1 na wartości początkowe, a następnie na ich podstawie obliczane jest przybliżone miejsce zerowe funkcji.

4.2 Funkcja f

Funkcja f odpowiada za obliczenie wartości f(x) zależnie od danego x. Zwraca wartośc double.

```
double f(double x){
    return pow(x, 4) - 7.899*pow(x, 3) + 23.281114*pow(x, 2) +
    14.73866033 - 30.33468152*x;
}
```

Fragment kodu 2: f

4.3 Funkcja df

Funkcja df odpowiada za obliczenie wartości pochodnej funkcji f(x) zależnie od danego x oraz Δx zgodnie z wzorem:

$$\frac{df(x)}{dx} = \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}$$

```
double df(double x, double delta) {
    return (f(x + delta) - f(x - delta)) / (2 * delta);
}
```

Fragment kodu 3: df

4.4 Funkcja u

Funkcja u odpowiada za obliczenie wartości u(x) zgodnie z wzorem:

$$u(x) = \frac{f(x)}{f'(x)}$$

```
double u(double x, double delta) {
    double derivative = df(x, delta);
    double result = f(x) / derivative;
    return result;
}
```

Fragment kodu 4: u

4.5 Opis głównej pętli dla niezmodyfikowanej metody

Poniżej przedstawiono kod, za pomocą którego wyznaczono kolejne wartości x, *blad* oraz $f(x)$, dla każdej iteracji. Rezultat zapisywany jest w pliku *outFile*.

```
    do {
        // Obliczenie nowego przyblizenia
        x2 = x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0));

        // Obliczenie bledu
        blad = std::abs(x2 - x1);

        // Aktualizacja x0 i x1 dla nastepnej iteracji
        x0 = x1;
        x1 = x2;
        iteracja++;

        //Zapis do pliku
        outFile1 << iteracja << "," << x2 << "," << blad << "," <<
    f(x2) << "\n";


        // Wypisanie informacji o iteracji
        std::cout << "Iteracja " << iteracja << ": x = " << x2 << "
    , blad = " << blad << ", f(x) = " << f(x2) << std::endl;

```

```
20      } while (blad > epsilon); // Warunek zakonczenia - osiagniecie
        zadanej dokladnosci
21
22      //Wypisanie rezultatu.
23      std::cout << "Znaleziony pierwiastek: " << x2 << std::endl;
```

Fragment kodu 5: main

Zależnie od tego, które miejsce zerowe chcemy wyznaczyć, wybieramy plik outFile1, lub outFile2, lub outFile3

4.6 Opis głównej pętli dla zmodyfikowanej metody

```
1       do {
2           double u1 = u(x1, delta);
3           double u0 = u(x0, delta);
4
5           x2 = x1 - u1 * (x1 - x0) / (u1 - u0);
6           blad = std::abs(x2 - x1);
7           x0 = x1;
8           x1 = x2;
9           iteracja++;
10
11          //zapis do pliku outFile_mod
12          outFile_mod << iteracja << "," << x2 << "," << blad << ","
        << f(x2) << "\n";
13
14          // Informacja o iteracjach
15          std::cout << "Iteracja " << iteracja << ": x = " << x2 << "
        , blad = " << blad << ", u(x) = " << u(x2, delta) << std::endl;
16
17      } while (blad > epsilon);
```

Fragment kodu 6: main

5 Analiza wyników

5.1 Rezultaty metody podstawowej

Wynikiem zastosowania metody podstawowej, są 3 tablki z wypisanymi kolejnymi iteracjami koniecznymi do osiągnięcia odpowiedniego miejsca zerowego.

5.1.1 Miejsce zerowe 1

Do wyznaczenia pierwszego miejsca zerowego ustawiono wartości początkowe $x_0 = 1.5$ oraz $x_1 = 1.7$. W rezultacie otrzymano tabelkę:

Iteracja	x	blad	$f(x)$
1	1.67266	0.0273394	-0.00298462
2	1.53226	0.140402	0.0137557
3	1.64763	0.11537	-0.00189043
4	1.63369	0.0139394	-0.00100336
5	1.61792	0.0157669	0.0000273503
6	1.6213	0.00337725	-2.59523e-05
7	1.62101	0.000292689	-5.76045e-07
8	1.621	6.64408e-06	1.26027e-09
9	1.621	1.45042e-08	-7.10543e-14

Tabela 1: Iteracje kolejnych przybliżeń pierwszego miejsca zerowego

Można zauważyć że przy takich wartościach początkowych oraz tej metodzie miejsce zerowe zostało znaleznione już po 9 iteracjach.

5.1.2 Miejsce zerowe 2

Do wyznaczenia drugiego miejsca zerowego ustalono wartości początkowe $x_0 = 1.8$ oraz $x_1 = 2.0$. W rezultacie otrzymano tabelkę:

Iteracja	x	blad	$f(x)$
1	1.91402	0.0859807	0.000756643
2	1.84874	0.0652755	-0.000923854
3	1.88463	0.0358853	6.80715e-05
4	1.88217	0.00246265	4.34379e-06
5	1.882	0.000167858	-2.91977e-08
6	1.882	1.12076e-06	1.21858e-11
7	1.882	4.6756e-10	0

Tabela 2: Iteracje kolejnych przybliżeń drugiego miejsca zerowego

5.1.3 Miejsce zerowe 3

Do wyznaczenia trzeciego miejsca zerowego ustalono wartości początkowe $x_0 = 2.0$ oraz $x_1 = 2.2$. W tym wypadku, przy takich wartościach początkowych, miejsce zerowe zostało wyznaczone dopiero po 2660 iteracjach.

Iteracja	x	blad	$f(x)$
1	2.20009	8.51334e-05	8.10367e-07
2	2.19901	0.000107182	1.97658e-07
3	2.19867	0.000345766	9.10238e-08
...	...	...	...
2658	2.41184	0.213812	0.0191609
2659	2.19803	0.213812	9.65439e-09
2660	2.19803	1.07731e-07	9.65325e-09

Tabela 3: Iteracje kolejnych przybliżeń trzeciego miejsca zerowego

5.2 Rezultaty metody rozszerzonej

Metoda rozszerzona została wykorzystana tylko do obliczenia pierwiastka o krotności parzystej. Ustawiono warunki początkowe tak samo jak w punkcie 5.1.3 tj. $x_0 = 2.0$, $x_1 = 2.2$.

5.2.1 Dla $delta_x = 0.1$

Iteracja	x	blad	$f(x)$
1	2.20002	2.43055e-05	7.64099e-07
2	2.19895	0.00107189	1.75673e-07
3	2.19861	0.00034002	7.80916e-08
...	...	...	...
656	2.1573	0.040567	0.000244547
657	2.19787	0.0405662	1.2618e-08
658	2.19787	7.94857e-07	1.2656e-08

Tabela 4: Iteracje kolejnych przybliżeń metody rozszerzonej dla miejsca zerowego krotności parzystej

5.2.2 Dla $delta_x = 0.001$

Iteracja	x	blad	$f(x)$
1	2.2002	0.000204008	9.04802e-07
2	2.19794	0.00226267	1.0131e-08
3	2.19864	0.000698876	8.44738e-08
...	...	...	...
280	2.198	0.000125487	9.50526e-09
281	2.19787	0.000125749	1.25079e-08
282	2.19787	2.61806e-07	1.25202e-08

Tabela 5: Iteracje kolejnych przybliżeń metody rozszerzonej dla miejsca zerowego krotności parzystej

5.3 Porównanie metod dla pierwiastka krotności parzystej

6 Wnioski

Skuteczność pojedyńczych pierwiastków Metoda siecznych okazała się skuteczną techniką do znajdowania pojedyńczych pierwiastków równań nieliniowych. Przy odpowiednio dobranych wartościach startowych, metoda szybko dorpowadza do dokładnego rozwiązania, co potwierdza teorytyczne właściwości metody, która jest znana z szybkiej zbieżności przy odpowiednich założeniach.

Pierwiastki podwójne metoda podstawowa Znalezienie pierwiastka o krotności parzystej metodą podstawową wymagało znacznie większej liczby iteracji, co jest zgodne z teorią sugerującą, że standardowa metoda siecznych może mieć problemy z zbieżnością w przypadkach pierwiastków wielokrotnych.

Metoda rozszerzona Modyfikacja metody siecznych pozwoliła na znaczące przyspieszenie zbieżności w przypadku pierwiastka o krotności parzystej. Potwierdza to założenie, że dzielenie wartości funkcji przez jej pochodną w metodzie siecznych efektywnie przyspiesza zbieganie do pierwiastka wielokrotnego.

Różne wartości Δx Porównanie wyników dla różnych wartości delty (0.1 oraz 0.001) w modyfikowanej metodzie siecznych ukazuje, że wybór parametru Δx ma duży wpływ na szybkość zbiegania metody. Przy takich samych wartościach początkowych okazało się osiągnąć rezultat około 2.3 raza szybciej, a w porównaniu z metodą podstawową około 9.4 raza szybciej.