Sprawozdanie z projektu

Wyznaczanie miejsc zerowych wielomianu przy użyciu Metody Newtona

Miłosz Zieliński

17.12.2024

Krótki opis metody:

Metoda Newtona służy do iteracyjnego znajdowania przybliżonych wartości miejsc zerowych funkcji ciągłej na zadanym przedziale. Kolejne przybliżenia wartości szukanego miejsca zerowego wyrażają się wzorem:

$x_{k+1} = x_k + \frac{f(x_k)}{f'(x_k)}, k = 1,2, \dots$

W założeniach metody mamy:

1. Na zadanym przedziale znajduje się jedno miejsce zerowe.
2. Funkcja ma różne znaki na końcach zadanego przedziału.
3. Pierwsza i druga pochodna mają stały znak na zadanym przedziale.

W moim programie sprawdzany jest warunek 2. Warunek pierwszy niestety nie zawsze jest spełniony. 3 również nie jest sprawdzany

Warunkiem stopu funkcji jest w moim przypadku $|x_{k+1} - x_k| < d$, d – dopuszczalny błąd

Opis rozwiązania problemu:

Tematem projektu jest napisanie programu, który pozwoli na znalezienie miejsc zerowych wielomianu opisanego w następujący sposób:

$$w_n(x) = a_0 + a_1\big(T_1(x) - U_1(x)\big) + \cdots + a_n\big(T_n(x) - U_n(x)\big)$$

W tym celu stworzyłem zestaw następujących funkcji:

1. chebyshevDiffAndDerivative(n,x)
   Funkcja ta rekurencyjnie wylicza wartości kolejnych składowych wielomianu $w_n(x)$, oraz pochodnej tego wielomianu. Wyniki umieszczone są w wektorze. Ważne, aby wektor współczynników był o 1 dłuższy od podawanych współczynników, ostatnim współczynnikiem powinno być 0.

   Funkcja ta przyjmuje dwa argumenty:

   n – maksymalny stopień wielomianu Chebysheva (powiązane z długością podawanego przez użytkownika wektora współczynników)

   x - wartość skalarna, dla której obliczamy różnice

   Funkcja ta zwraca dwa wektory kolejno długości n + 2 i n+1:
   valueDiff - wektor różnic T_0(x)-U_0(x), ..., T_n(x)-U_n(x)
   derivDiff - wektor różnic T'_0(x)-U'_0(x), ..., T'_n(x)-U'_n(x)

2. chebyshevSum(a,x)
   Liczone są tutaj wartości wielomianu i jego pochodnej.

   Funkcja przyjmuje argumenty:
   a – wektor współczynników podany przez użytkownika
   x – wartość skalara dla której liczymy wartość wielomianu i jego pochodnej w zadanym punkcie

   Zwracane są dwie wartości:

   sumValueDiff – wartość całego wielomianu w zadanym punkcie

   sumDerivDiff - wartość pochodnej całego wielomianu w zadanym punkcie

3. newtonChebyshev(a, x0, tol, maxIter)
   Jest tutaj zaimplementowana właściwa metoda Newtona.

   Funkcja przyjmuje argumenty:
   a – wektor współczynników podany przez użytkownika
   x0 – początkowe przybliżenie
   tol – tolerancja błędu wyniku (liczone jako różnica między $x_k\ i\ x_{k+1}$)
   maxIter – maksymalna liczba iteracji

   Funkcja zwraca znalezione miejsce zerowe.

4. findRootsInSubintervals(a, intervalStart, intervalEnd, numSubintervals, tol, maxIter)

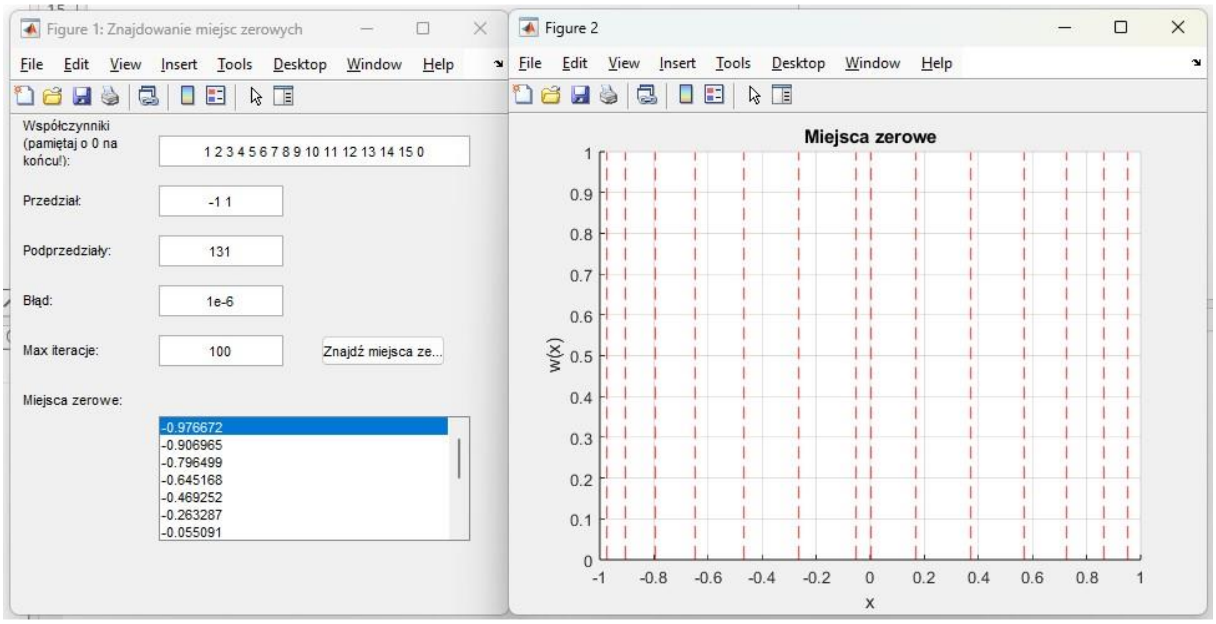
   Funkcja szuka na zadanym przedziale miejsc zerowych wielomianu, jeżeli wartości na końcach przedziału mają różne znaki. Robi to wywołując funkcję newtonChebyshev na środkach podprzedziałów na jakie podzielony jest nasz przedział. Liczbę podprzedziałów ustala użytkownik.

   Funkcja przyjmuje argumenty:

   a – wektor współczynników podany przez użytkownika
   intervalStart, intervalEnd – początek i koniec przedziału na którym szukamy miejsc zerowych
   numSubintervals – liczba na ile podprzedziałów dzielimy nasz przedział
   tol – tolerancja błędu wyniku (liczone jako różnica między $x_k\ i\ x_{k+1}$)
   maxIter – maksymalna liczba operacji

5. Stworzone jest także proste GUI (chebyshevRootsGUI) i plik „przypadki”, gdzie zapisane są poniższe przypadki.

Wygląd GUI:



Ciekawe przypadki

1. Zwykły (wielomian stopnia 6)

```
%Wektor współczynników
a = [1,4,1,0,-2,0,4,0]

% Parametry
intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 121;  % Liczba podprzedziałów
tol = 1e-6;             % Tolerancja błędu
maxIter = 1000;          % Maksymalna liczba iteracji

% Znajdowanie miejsc zerowych
roots = findRootsInSubintervals(a, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
roots =

   -0.9317    0.1253    0.6459    0.8485
```

Można tutaj zauważyć, że wyszły 4 miejsca zerowe. Można jednak spodziewać się było 6, bo wielomian jest 6 stopnia. Nie mamy 6, ponieważ mamy przesunięcie w górę o 1. Wszystkie miejsca zerowe są w przedziale między -1, a 1. Tak będzie bardzo często, jednak nie zawsze. Wyniki zgadzają się z tymi wynikającymi z programu WolframAlpha. Na końcu wektora a jest współczynnik 0, co może wydawać się niepotrzebne, ale jest koniecznie, aby policzyć prawidłowo U'_0(x).

2. Miejsce zerowe poza wybranym przedziałem

```
b=[1,-2,0.5,0]

% Parametry
intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 121;  % Liczba podprzedziałów
tol = 1e-6;            % Tolerancja błędu
maxIter = 100;         % Maksymalna liczba iteracji

roots = findRootsInSubintervals(b, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
roots =

   -0.4142
```

Funkcja wskazuje, że wielomian ten ma jedno miejsce zerowe, nie jest to jednak prawda, co widać, gdy sprawdzimy to za pomocą WolframAlpha.

$1 - 2\,(T_1(x) - U_1(x)) + 0.5\,(T_2(x) - U_2(x))$

$x \approx -0.414214$

$x \approx 2.41421$

Gdy jednak zmienimy przedział:

```
intervalEnd = 4;
roots = findRootsInSubintervals(b, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots

roots =

   -0.4142    2.4142
```

Wydaje mi się, że sytuacja, gdy jakieś miejsce zerowe jest poza przedziałem <-1,1> może się zdarzyć, gdy jakieś współczynniki będą mniejsze od 1.

3. Nie jest znajdowane miejsce zerowe pomimo zawierania się w przedziale

   Taka sytuacja może się zdarzyć, gdy miejsce zerowe będzie w miejscu gdzie następuje podział i wtedy nie zostanie ono znalezione.

```
c = [0,0,0,0,1,0,0,0,0,1,2,3,0]

% Parametry
intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 120;  % Liczba podprzedziałów
tol = 1e-6;            % Tolerancja błędu
maxIter = 100;         % Maksymalna liczba iteracji

roots = findRootsInSubintervals(c, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots

roots =

   -0.9702   -0.8266   -0.6667   -0.1839    0.0812    0.3333    0.6219    0.8283    0.9493
```

   Dla takiego wielomianu program wskazał 9 miejsc zerowych. Nie zgadza się to jednak z wynikiem WolframAlpha.

$$(T_4(x) - U_4(x)) + (T_9(x) - U_9(x)) + 2\,(T_{10}(x) - U_{10}(x)) + 3\,(T_{11}(x) - U_{11}(x))$$

$x \approx -0.5$

$x = 0$

$x \approx -0.97019$

$x \approx -0.826555$

$x \approx -0.666707$

$x \approx -0.183919$

$x \approx 0.0812423$

$x \approx 0.333322$

$x \approx 0.621918$

$x \approx 0.828274$

$x \approx 0.949281$

   Brakuje miejsc zerowych -0.5 i 0. Gdy zmienimy liczbę przedziałów ze 120 do 121:

```
numSubintervals = 121;
roots = findRootsInSubintervals(c, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
```

```
roots =

   -0.9702   -0.8266   -0.6667   -0.5000   -0.1839         0    0.0812    0.3333    0.6219    0.8283    0.9493
```

To już mamy wszystkie miejsca zerowe. Można zauważyć, że dzieląc odcinek od -1 do 1 na 120 fragmentów granice podziałów będą właśnie w -0.5 i 1. Jest to z mojej perspektywy duża wada, którą nie do końca wiem jak obejść. Być może warto dawać liczby nieparzyste jako liczbę podziałów.

4. Za mała liczba podziałów

```
d = ones(1, 27);
d(27) = 0;

% Parametry
intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 175;  % Liczba podprzedziałów
tol = 1e-6;             % Tolerancja błędu
maxIter = 100;          % Maksymalna liczba iteracji

roots = findRootsInSubintervals(d, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
```

Jest to funkcja, która będzie stopnia 25, tym samym nie będzie więcej niż 25 miejsc zerowych. Możemy zatem przypuszczać, że podzielenie odcinka <-1,1> na aż 175 podprzedziałów pozwoli znaleźć je wszystkie. Tak jednak nie jest:

```
roots =

   -0.9906   -0.9752    0.6104    0.6998    0.7695    0.8649    0.8942
```

Znaleziono 7 miejsc zerowych, program WolframAlpha wskazuje 9. (Niewygodne byłoby ich podawanie tutaj). Jednak, gdy zwiększyć liczbę przedziałów tylko o 2, to już znajdujemy wszystkie miejsca zerowe.

```
numSubintervals = 177;
roots = findRootsInSubintervals(d, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
```

```
roots =

   -0.9906   -0.9752    0.6104    0.6998    0.7695    0.8649    0.8942    0.9659    0.9731
```

Trudno określić zatem ile dokładnie podprzedziałów jest potrzebnych.

5. Za mały limit iteracji

Może zdarzyć się, gdyby błąd miałby być bardzo mały lub funkcja by w rozbiegała, że funkcja nie zakończy się w określonej liczbie iteracji. Wtedy w miejsce oczekiwanego miejsca zerowego pojawia się wartość NaN.

```
e = [0,0,0,0,1,0,0,0,0,1,2,3,0]

intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 121;  % Liczba podprzedziałów
tol = 1e-15;            % Tolerancja błędu
maxIter = 5;         % Maksymalna liczba iteracji
Warning: Nie znaleziono miejsca zerowego w 5 iteracjach.
> In newtonChebyshev (line 82)
In findRootsInSubintervals (line 41)
In test (line 87)

roots =

  -0.9702   -0.8266   -0.6667   -0.5000   -0.1839         0    0.0812    0.3333    0.6219    0.8283       NaN
```

Jednak jak widać limit ten jest tu bardzo mały, w pozostałych przypadkach liczba iteracji nie była większa od 5. W funkcji newtonChebyshev można odkomentować linijkę, która wyświetla liczbę iteracji dla każdego miejsca zerowego.

```
Metoda Newtona zakończyła się po 5 iteracjach.
Metoda Newtona zakończyła się po 4 iteracjach.
Metoda Newtona zakończyła się po 5 iteracjach.
Metoda Newtona zakończyła się po 4 iteracjach.
Metoda Newtona zakończyła się po 4 iteracjach.
Metoda Newtona zakończyła się po 1 iteracjach.
Metoda Newtona zakończyła się po 5 iteracjach.
Metoda Newtona zakończyła się po 4 iteracjach.
Metoda Newtona zakończyła się po 5 iteracjach.
Metoda Newtona zakończyła się po 4 iteracjach.
Warning: Nie znaleziono miejsca zerowego w 5 iteracjach.
> In newtonChebyshev (line 41)
In findRootsInSubintervals (line 41)
In przypadki (line 87)
```

6. Pochodna bliska 0

```
f = [1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,1e-15,0]

intervalStart = -1;  % Początek przedziału
intervalEnd = 1;     % Koniec przedziału
numSubintervals = 121;  % Liczba podprzedziałów
tol = 1e-8;            % Tolerancja błędu
maxIter = 100;         % Maksymalna liczba iteracji

roots = findRootsInSubintervals(f, intervalStart, intervalEnd, numSubintervals, tol, maxIter);
roots
```

```
Warning: Pochodna jest zbyt bliska zeru.
> In newtonChebyshev (line 21)
In findRootsInSubintervals (line 41)
In przypadki (line 103)
Warning: Pochodna jest zbyt bliska zeru.
> In newtonChebyshev (line 21)
In findRootsInSubintervals (line 41)
In przypadki (line 103)
Warning: Pochodna jest zbyt bliska zeru.
> In newtonChebyshev (line 21)
In findRootsInSubintervals (line 41)
In przypadki (line 103)
Warning: Pochodna jest zbyt bliska zeru.
> In newtonChebyshev (line 21)
In findRootsInSubintervals (line 41)
In przypadki (line 103)

roots =

   NaN   NaN   NaN   NaN
```

Wyniki z WolframAlpha:

Input interpretation

$$\frac{1}{10^{15}} + \sum_{k=1}^{10} \frac{T_k(x) - U_k(x)}{10^{15}} = 0$$

Real solutions

$x \approx -0.971837$

$x \approx 0.548846$

$x \approx 0.754198$

$x \approx 0.86825$

Gdy pochodna funkcji dla danej wartości jest bliska zero to wynik może być niestabilny, ponieważ dzielimy przez bardzo małą liczbę, co sprawi, że $\frac{f(x_k)}{f'(x_k)}$ będzie bardzo duże. W podanym przypadku pochodna funkcji jest właśnie bardzo bliska 0, więc mimo iż algorytm „widzi” (wartości NaN) te miejsca zerowe to dla bezpieczeństwa nie są one liczone, pomimo iż wartość samej funkcji również jest bardzo mała. Wartość funkcji nie byłaby mała gdyby a_0 było „dalekie” od 0. W funkcji, gdy pochodna ma wartość mniejszą niż 1e-10 to uznawana jest za bliską 0.

Wniosek:

Z powyższych przypadków widać, że zastosowanie metody Newtona do wyszukiwania wielu miejsc zerowych wielomianu $w(x)$ może w niektórych przypadkach wiązać się z problemami. Jednak jeżeli jesteśmy świadomi jak działa algorytm i jak powinniśmy szukać tych miejsc zerowych to jest to bardzo szybki sposób na znalezienie odpowiednich wartości.