

Laboratorium 5 – Poszukiwanie pierwiastków

Tomasz Belczyk

05.06.2021

Metody Obliczeniowe w Nauce i Technice

Informatyka niestacjonarna 2020/2021

Wydział Informatyki, Elektroniki i Telekomunikacji

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

1. Treść zadań

1. Uruchomić program root\_finding.tgz.

* Umieć odpowiedzieć na pytanie, co on robi.
* Narysować (np. za pomocą gnuplota ) wykres funkcji, której miejsc zerowych szukamy.

2. Zmienić program tak, aby znajdował pierwiastek metodą siecznych oraz Brent-Dekker'a.

* Porownać metody.
* Zamienić program tak, aby spróbował znaleźć pierwiastek równania x^2-2\*x+1=0.
* Narysować wykres tej funkcji za pomocą np. gnuplota.
* Wyjasnić działanie programu - dlaczego nie może znaleźć miejsc zerowych dla tego równania?

3. Napisać program szukający miejsc zerowych za pomocą metod korzystających z pochodnej funkcji. Czym różnia się od poprzednich metod i dlaczego potrafią znaleźć pierwiastek równania x^2-2\*x+1=0?

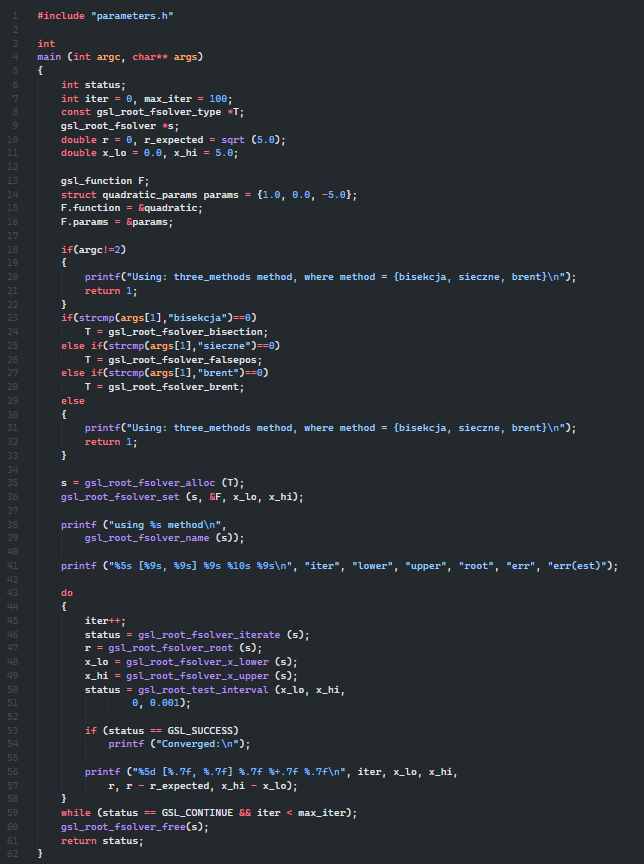
* Porównać metodę Newtona, uproszczoną Newtona i Steffensona.

Podejście do rozwiązania zadań

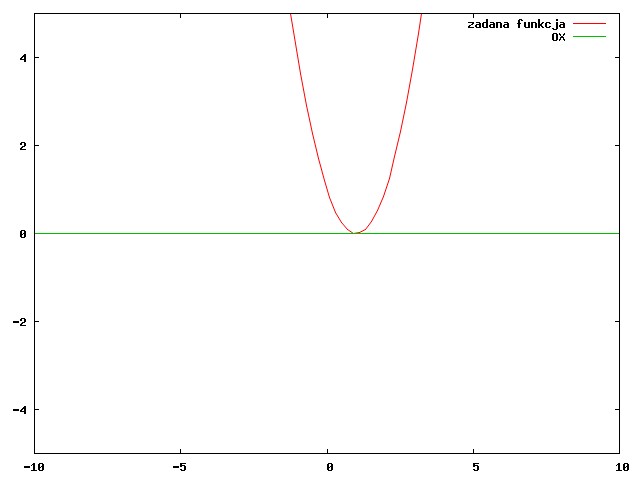


Program root\_finding szuka pierwiastka funkcji kwadratowej o postaci: x^2-5w przedziale [0,5] używając metody bisekcji. Wiadomo, że w tym przedziale musi znajdować się pierwiastek, ponieważ funkcja zmienia tam znak i jest ciągła. Dokonujemy kolejnych połowień przedziału i odrzucamy podprzedział, dla którego nie występuje zmiana znaku między końcami. Długość pozostałego podprzedziału jest oszacowaniem błędu na danym etapie, a jego środek aktualnym przybliżeniem pierwiastka. Analitycznie odnaleziona wartość pierwiastka: 2.236068 (oraz - 2.236068) Numerycznie odnaleziona wartość pierwiastka (dokładność = 0.001): 2.2357178(oraz – 2.2357178)

1. Zmieniamy program według opisu



Z programu wynika, że metoda Brenta poradziła sobie najszybciej w 6 iteracjach. Metoda bisekcji (będąca najprostszą) potrzebowała dwa razy więcej iteracji, by uzyskać mniej dokładny wynik, natomiast metoda siecznych zajęła miejsce pomiędzy poprzednio wymienionymi metodami. Jedyna różnica w kodzie polega na zmianie struct quadratic\_params params = {1.0, -2.0, 1.0};



Żadna z trzech podanych metod nie odnajduje pierwiastka, ponieważ badana funkcja nie spełnia ich założeń. Prawdą jest, że pierwiastek znajduje się w zadanym przedziale (x(0) = 1), jednak ma parzysty stopień, więc funkcja nie zmienia tu znaku.

3.

Zmieniamy program w następujący sposób



Wyniki pokazują, że najlepiej radzi sobie metoda Steffensona, działając bardzo szybko I dając bardzo dokładny wynik. Z kolei metoda Newtona działa w tym przypadku szybciej I dokładniej niż jej uproszczona odmiana, która liczy przybliżone pochodne na podstawie wzoru funkcji.

Wykresy, tabele, wyniki liczbowe

Ad 2.

bisection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| iter | lower | upper | root | err | err(est) |
| 1 | 0 | 2.5 | 1.25 | -0.98607 | 2.5 |
| 2 | 1.25 | 2.5 | 1.875 | -0.36107 | 1.25 |
| 3 | 1.875 | 2.5 | 2.1875 | -0.04857 | 0.625 |
| 4 | 2.1875 | 2.5 | 2.34375 | 0.107682 | 0.3125 |
| 5 | 2.1875 | 2.34375 | 2.265625 | 0.029557 | 0.15625 |
| 6 | 2.1875 | 2.265625 | 2.226563 | -0.00951 | 0.078125 |
| 7 | 2.226563 | 2.265625 | 2.246094 | 0.010026 | 0.039063 |
| 8 | 2.226563 | 2.246094 | 2.236328 | 0.00026 | 0.019531 |
| 9 | 2.226563 | 2.236328 | 2.231445 | -0.00462 | 0.009766 |
| 10 | 2.231445 | 2.236328 | 2.233887 | -0.00218 | 0.004883 |
| 11 | 2.233887 | 2.236328 | 2.235107 | -0.00096 | 0.002441 |
| 12 | 2.235107 | 2.236328 | 2.235718 | -0.00035 | 0.001221 |

falsepos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| iter | lower | upper | root | err | err(est) |
| 1 | 1 | 2.5 | 1 | -1.23607 | 1.5 |
| 2 | 2.142857 | 2.5 | 2.142857 | -0.09321 | 0.357143 |
| 3 | 2.230769 | 2.321429 | 2.230769 | -0.0053 | 0.090659 |
| 4 | 2.235969 | 2.276099 | 2.235969 | -9.9E-05 | 0.04013 |
| 5 | 2.236067 | 2.256034 | 2.236067 | -9E-07 | 0.019967 |
| 6 | 2.236068 | 2.24605 | 2.236068 | 0 | 0.009983 |
| 7 | 2.236068 | 2.241059 | 2.236068 | 0 | 0.004991 |
| 8 | 2.236068 | 2.238564 | 2.236068 | 0 | 0.002496 |
| 9 | 2.236068 | 2.236068 | 2.236068 | 0 | 0 |

brent

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| iter | lower | upper | root | err | err(est) |
| 1 | 1 | 5 | 1 | -1.23607 | 4 |
| 2 | 1 | 3 | 3 | 0.763932 | 2 |
| 3 | 2 | 3 | 2 | -0.23607 | 1 |
| 4 | 2.2 | 3 | 2.2 | -0.03607 | 0.8 |
| 5 | 2.2 | 2.23663 | 2.23663 | 0.000562 | 0.03663 |
| 6 | 2.236063 | 2.23663 | 2.236063 | -4.6E-06 | 0.000567 |

Ad 3.

newton

|  |  |  |  |
| --- | --- | --- | --- |
| Iter | root | err | err(est) |
| 1 | 3 | 2 | -2 |
| 2 | 2 | 1 | -1 |
| 3 | 1.5 | 0.5 | -0.5 |
| 4 | 1.25 | 0.25 | -0.25 |
| 5 | 1.125 | 0.125 | -0.125 |
| 6 | 1.0625 | 0.0625 | -0.0625 |
| 7 | 1.03125 | 0.03125 | -0.03125 |
| 8 | 1.015625 | 0.015625 | -0.01563 |
| 9 | 1.007813 | 0.007813 | -0.00781 |
| 10 | 1.003906 | 0.003906 | -0.00391 |
| 11 | 1.001953 | 0.001953 | -0.00195 |
| 12 | 1.000977 | 0.000977 | -0.00098 |

secant

|  |  |  |  |
| --- | --- | --- | --- |
| iter | root | err | err(est) |
| 1 | 3 | 2 | -2 |
| 2 | 2.333333 | 1.333333 | -0.66667 |
| 3 | 1.8 | 0.8 | -0.53333 |
| 4 | 1.5 | 0.5 | -0.3 |
| 5 | 1.307692 | 0.307692 | -0.19231 |
| 6 | 1.190476 | 0.190476 | -0.11722 |
| 7 | 1.117647 | 0.117647 | -0.07283 |
| 8 | 1.072727 | 0.072727 | -0.04492 |
| 9 | 1.044944 | 0.044944 | -0.02778 |
| 10 | 1.027778 | 0.027778 | -0.01717 |
| 11 | 1.017167 | 0.017167 | -0.01061 |
| 12 | 1.01061 | 0.01061 | -0.00656 |
| 13 | 1.006557 | 0.006557 | -0.00405 |
| 14 | 1.004053 | 0.004053 | -0.0025 |
| 15 | 1.002505 | 0.002505 | -0.00155 |
| 16 | 1.001548 | 0.001548 | -0.00096 |

Steffenson

|  |  |  |  |
| --- | --- | --- | --- |
| iter | root | err | err(est) |
| 1 | 3 | 2 | -2 |
| 2 | 2 | 1 | -1 |
| 3 | 1 | 0 | -1 |
| 4 | 1 | 0 | 0 |

Z tego wynika, że najlepiej radzi sobie metoda Steffensona, działając bardzo szybko I dając bardzo dokłedny wynik. Z kolei metoda Newtona działa w tym przypadku szybciej I dokładniej niż jej uproszczona odmiana, która liczy przybliżone pochodne na podstawie wzoru funkcji.