Sprawozdanie MNUM Projekt 03

Autor: **TOMASZ SACHANOWSKI**

Grupa: **czwartek 8-10**

Nr. Indexu: **276467**

Nr. Zadania**: 3.55**

Spis treści

[Treść zadań 2](#_Toc39150478)

[Zadanie 1 3](#_Toc39150479)

[Cel: 3](#_Toc39150480)

[Teoria: 3](#_Toc39150481)

[Wynik: 8](#_Toc39150482)

[Podsumowanie: 9](#_Toc39150483)

[Zadanie 2 10](#_Toc39150484)

[Cel: 10](#_Toc39150485)

[Teoria: 10](#_Toc39150486)

[Wynik 14](#_Toc39150487)

[Podsumowanie: 17](#_Toc39150488)

[Dodatek do zadania 1 18](#_Toc39150489)

[Func 18](#_Toc39150490)

[bisection 18](#_Toc39150491)

[secant 19](#_Toc39150492)

[compose\_task\_1 19](#_Toc39150493)

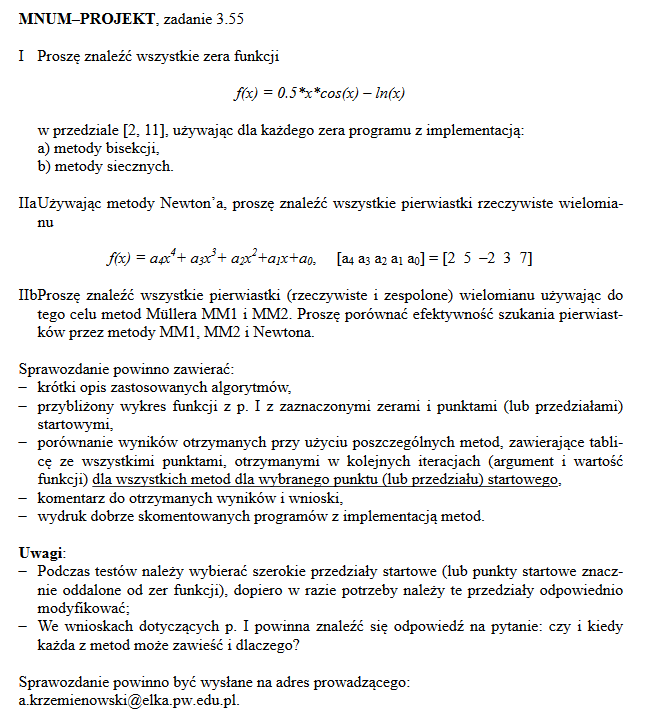
[Dodatek do zadania 2 19](#_Toc39150494)

[newton 20](#_Toc39150495)

[mm1 20](#_Toc39150496)

[mm2 21](#_Toc39150497)

# Treść zadań



# Zadanie 1

## Cel:

Celem zadania jest znalezienie wszystkich pierwiastków funkcji w zadanym przedziale przy pomocy wskazanych metod.

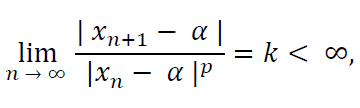
## Teoria:

Pierwiastek jest argument x dla którego funkcja przyjmuje wartość zero, czyli f() = 0. Aby wyznaczyć takie miejsca zerowe, trzeba najpierw oszacować przedziały w którym znajdują się nasze rozwiązania (pierwiastki zerowe). Jest to tak zwane ***przedziały izolacji pierwiastka***. Przedział taki możemy odczytać w najprostszy sposób z uproszczonego wykresu funkcji (np. narysowanego w programie graficznym). Podstawową metodą wyznaczenia tego przedziału jest badanie iloczynu wartości funkcji na końcach przedziału – jeśli ten iloczyn jest ujemny (a funkcja ta jest ciągła) wówczas w przedziale tym znajduje się co najmniej jeden pierwiastek. Warto zaznaczyć, że taki przedział nie powinien być zbytnio szeroki i pochodna powinny być w nim monotoniczna (nie zmieniać się).

Po wyznaczeniu przedziału izolacji pierwiastka, kolejnym krokiem jest znalezienie naszego miejsca zerowego. Mamy do dyspozycji wiele metod iteracyjnych:

* Bisekcji
* Siecznych

Szybkość zbieżności metody określamy za pomocą rzędu (wykładnika zbieżności). Jest to największa liczba 𝑝≥1 taka, że:



k– współczynnik lub iloraz zbieżności.

𝑝=1 metoda jest zbieżna liniowo

𝑝=2 metoda jest zbieżna kwadratowo

Im większy jest rząd metody, tym metoda jest szybsza.

Metody iteracyjne dla problemów nieliniowych są ,na ogół, zbieżne tylko lokalnie.

Kulą zbieżności metody iteracyjnej nazywamy otoczenie rozwiązanie α o takim promieniu δ, że dla każdego punktu początkowego x0 spełniającego:

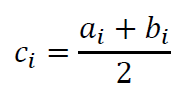


Metoda bisekcji

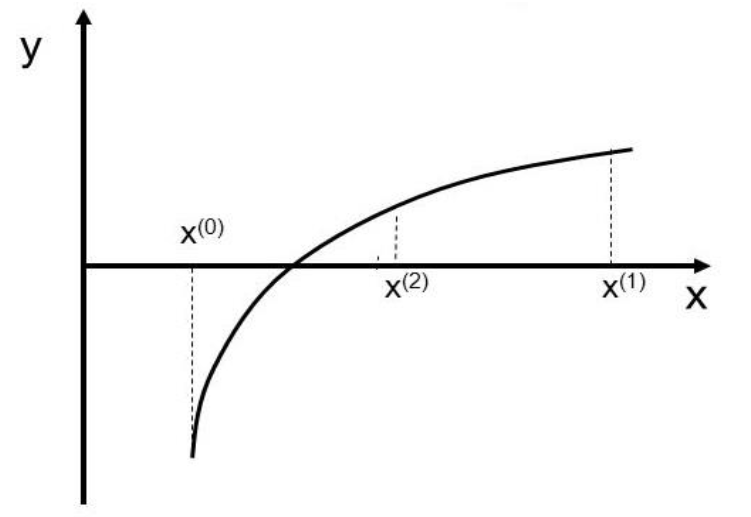
Dość naturalna metoda obliczeniowa zer skalarnych funkcji ciągłych określonych na danym przedziale [a, b] i zmieniających znak (tzn. funkcja przyjmuje na końcu przedziałów wartości przeciwnego znaku). Na mocy twierdzenia *Darboux* wiemy, że jest przynajmniej jedno zero funkcji.

Algorytm:

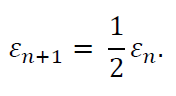
1. Aktualny przedział zawierający zero funkcji [, ] jest dzielony na dwie połowy:



1. Liczymy wartość funkcji w punkcie .
2. Liczymy iloczyny f() \* f() i f() \* f()
3. Nowym przedziałem będzie ten podprzedział, gdzie odpowiada ujemna wartość funkcji na jego końcach.
4. Procedura jest postarzana tak długo, aż zostanie osiągnięta zakładana dokładność.



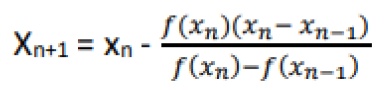
Jeśli przez oznaczymy długość przedziału w 𝑛-tym kroku, to:



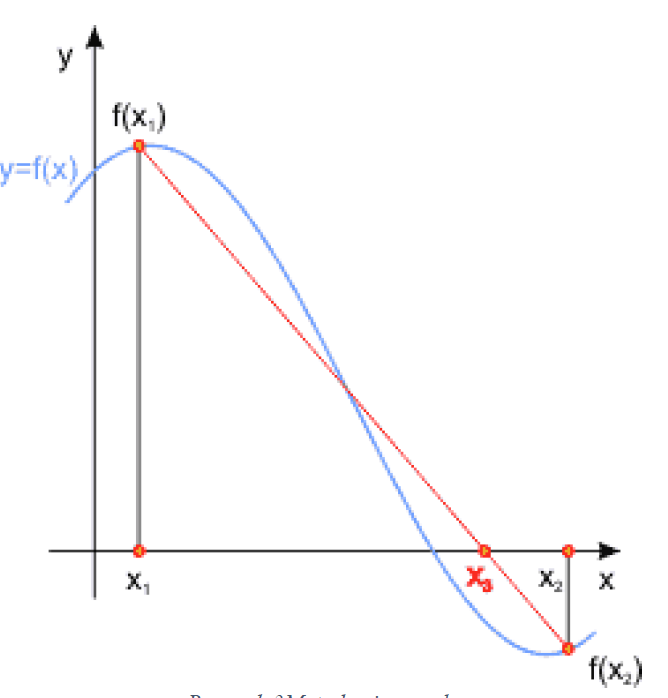
Dokładność rozwiązania zależy jedynie od ilości wykonanych iteracji, jest ona zbieżna liniowo). Jest to metoda zbieżna globalnie, co oznacza, że zawsze znajdziemy pierwiastek w danym przedziale, jeżeli ten tylko istnieje. Metoda bisekcji jest zbieżna globalnie( znajdzie się miejsce zerowe funkcji, choćby początkowa długość przedziału była bardzo duża).

Metoda siecznych

Metoda siecznych różni się tym od metody bisekcji, że aktualny przedział izolacji pierwiastka dzielony jest nie na dwa równe, ale na dwa najczęściej nierówne podprzedziały, prostą (sieczną) łączącą na płaszczyźnie dwa punkty (f() , ) i (f(), ), przecinającą oś rzędnych w punkcie oznaczonym jako , gdzie i to dwa ostatnio wyznaczone punkty. Nowy punkt określony jest wzorem:



Metoda ta jest szybsza od metody bisekcji, gdyż Rząd zbieżności metody siecznych p =(1 + √5)/2 ≈ 1.618. Jednakże, jest ona zbieżna tylko lokalnie, stąd w praktyce może być niezbieżna (przedział izolacji nie dostatecznie mały). Dlatego też algorytm wymaga użycia określonej ilości iteracji, gdyż rozwiązanie może nie zostać znalezione albo gdy sieczna jest równoległa do osi OX.



## Wynik:

Wykres funkcji ***0.5 \* x \* cos(x) - log(x)*** na przedziale ***<2, 11>***

Obraz zawierający mężczyzna, stojące

Opis wygenerowany automatycznie

Na postawie wykresu określone są przedziały izolacji:

Duże przedziały:

<4, 6>

<6, 8>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EPS | Bisekcja | | | | Sieczne | | | |
| Przedzial\_1 | | Przedzial\_2 | | Przedzial\_1 | | Przedzial\_2 | |
| iter | wynik | iter | wynik | iter | wynik | iter | wynik |
| 0.0010 | 11 | 5.3876953125 | 9 | 7.27734375 | 5 | 5.387894529147541 | 7 | 7.276905962745723 |
| 0.00010 | 11 | 5.3876953125 | 15 | 7.27703857421875 | 6 | 5.387698378950402 | 8 | 7.276999699969140 |
| 0.000010 | 17 | 5.387741088867188 | 17 | 7.277023315429688 | 8 | 5.387735529347010 | 10 | 7.277023303042951 |
| 0.0000010 | 21 | 5.387738227844238 | 21 | 7.277024269104004 | 10 | 5.387737921754509 | 11 | 7.277024151067826 |
| 0.00000010 | 24 | 5.387738108634949 | 24 | 7.277024388313293 | 11 | 5.387738128220748 | 13 | 7.277024364577876 |
| 0.000000010 | 28 | 5.387738086283207 | 27 | 7.277024373412132 | 13 | 5.387738089121902 | 14 | 7.277024372248815 |
| 0.0000000010 | 28 | 5.387738086283207 | 31 | 7.277024374343455 | 15 | 5.387738086603932 | 16 | 7.277024374180153 |
| 0.00000000010 | 34 | 5.387738086399622 | 35 | 7.277024374285247 | 16 | 5.387738086386628 | 17 | 7.277024374249542 |
| 0.000000000010 | 36 | 5.387738086428726 | 37 | 7.277024374270695 | 18 | 5.387738086427780 | 19 | 7.277024374267013 |
| 0.0000000000010 | 40 | 5.387738086430545 | 41 | 7.277024374267967 | 20 | 5.387738086430430 | 20 | 7.277024374267638 |

Małe przedziały

<5, 5.5>

<7, 7.5>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EPS | Bisekcja | | | | Sieczne | | | |
| Przedzial\_1 | | Przedzial\_2 | | Przedzial\_1 | | Przedzial\_2 | |
| iter | wynik | iter | wynik | iter | wynik | iter | wynik |
| 0.0010 | 9 | 5.3876953125 | 7 | 7.27734375 | 2 | 5.387594617550044 | 3 | 7.276760254516209 |
| 0.00010 | 9 | 5.3876953125 | 13 | 7.27703857421875 | 3 | 5.387741727294921 | 4 | 7.277000277488797 |
| 0.000010 | 15 | 5.387741088867188 | 15 | 7.277023315429688 | 3 | 5.387741727294921 | 5 | 7.277022176193655 |
| 0.0000010 | 19 | 5.387738227844238 | 19 | 7.277024269104004 | 4 | 5.387737994017899 | 6 | 7.277024173765735 |
| 0.00000010 | 22 | 5.387738108634949 | 22 | 7.277024388313293 | 5 | 5.387738088776229 | 7 | 7.277024355978603 |
| 0.000000010 | 26 | 5.387738086283207 | 25 | 7.277024373412132 | 5 | 5.387738088776229 | 8 | 7.277024372599518 |
| 0.0000000010 | 26 | 5.387738086283207 | 29 | 7.277024374343455 | 6 | 5.387738086371076 | 9 | 7.277024374115628 |
| 0.00000000010 | 32 | 5.387738086399622 | 33 | 7.277024374285247 | 7 | 5.387738086432124 | 10 | 7.277024374253924 |
| 0.000000000010 | 34 | 5.387738086428726 | 35 | 7.277024374270695 | 7 | 5.387738086432124 | 11 | 7.277024374266539 |
| 0.0000000000010 | 38 | 5.387738086430545 | 39 | 7.277024374267967 | 8 | 5.387738086430575 | 12 | 7.277024374267690 |

## Podsumowanie:

Obje metody wykazały możliwość znalezienia miejsca zerowego. Z powyższych tabel wynika, że metoda bisekcji jest wolniejsza i potrzebuje znacznie więcej iteracji niż metoda siecznych. Jest to ponad dwa razy więcej iteracji a przy większej dokładności nawet 4 razy więcej. Szerokość przedziału izolacji wpływa w obu metodach na szybkość algorytmu. Mniejszy przedział pozwala wykonać algorytmy w mniejszej ilości iteracji.

Ponadto dla przedziału <3,7> metoda bisekcji znalazła poprawny pierwiastek 5.38 natomiast metoda siecznych już nie. Wynik dla niej był drugim pierwiastkiem 7.27

**Metoda bisekcji**:

Wadą metody jest to, iż zbieżność ta nie jest imponująca.

Ma ona parę zalet jest: ona w pewien sposób uniwersalna, ma ona zbieżność globalną, wystarczy dla niej jedynie ciągłość funkcji.

**Metoda siecznych**:

Metoda siecznych może zawieść. Jeśli jest ona jedynie lokalnie, stąd w praktyce może być niezbieżna – jeśli początkowy przedział izolacji pierwiastka nie jest dostatecznie mały.

Ponadto, gdy żądanie przez użytkownik dokładności są bardzo wielkie, a sama funkcja „złośliwa”, metoda siecznych może cierpieć z powodu redukcji cyfr przy odejmowaniu.

# Zadanie 2

## Cel:

Celem jest znalezienie wszystkich pierwiastków rzeczywistych wielomianu przy pomocy metody Newtona oraz znalezienie pierwiastków rzeczywistych i zespolonych przez metody MM1 MM2.

## Teoria:

Wielomian stopnia n podsiada dokładnie n pierwiastków:

* pierwiastki mogą być zarówno rzeczywiste oraz zespolone
* pierwiastki mogą być pojedyncze lub wielokrotne

Do poszukiwania pierwiastków rzeczywistych możemy korzystać z metod wyznaczania przeszukiwania zer funkcji nieliniowej (np. Newton).

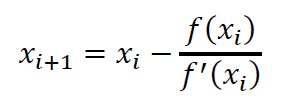
Jednak istnieją metody bardziej złożone, które są opracowane specjalnie dla wielomianów (wykorzystują właściwość – wielokrotna różniczkowalność).

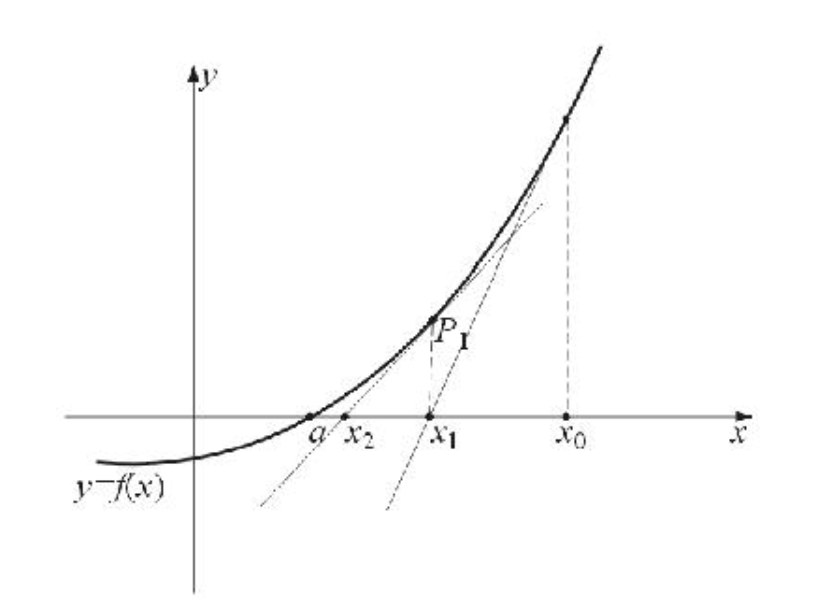
Do metod tych należą:

* metoda Müllera (aproksymacja wielomianu funkcją kwadratową w otoczeniu rozwiązania – uogólniona metoda siecznych – MM1, wykorzystanie informacji o wielomianie jedynie w jednym punkcie, tzn. wykorzystująca do wyznaczenia funkcji kwadratowej wartości wielomianu i jego pierwszej i drugiej pochodnej w danym punkcie – MM2)
* metoda Laguerre’a.

**Metoda Newtona**

Metoda Newtona, zwana też metodą stycznych zakłada aproksymację funkcji jej liniowym przybliżeniem wynikającym z uciętego rozwinięcia w szereg Taylora w aktualnym punkcie (aktualnym przybliżeniu pierwiastka), a następnie przyrównania do zera sformułowanej lokalnej aproksymacji funkcji f(x), co prowadzi do zależności iteracyjnej:





Metoda Newtona jest zbieżna lokalnie (jeśli zaczniemy ją stosować w punkcie zbytnio oddalonym od rozwiązania, to może być ona rozbieżna). Jej zbieżność jest kwadratowa.

Metoda stycznych jest szczególnie efektywna w przypadku, gdy krzywa jest bardzo stroma w otoczeniu danego pierwiastka (nie zaleca się stosowania, gdy krzywa jest w otoczeniu pierwiastka pozioma – innymi słowy pochodna w tym punkcie ma bardzo małą wartość).

Metoda Newtona znajduje tylko pierwiastki rzeczywiste.

**Metoda Müllera**

Metoda polega na aproksymacji wielomianu w otoczeniu rozwiązania funkcją kwadratową. Może być traktowana jako uogólnienie metody siecznych - zamiast interpolacji w dwóch punktach funkcją liniową (tzn. sieczną) wykonujemy interpolację w trzech punktach funkcją kwadratową. Istnieje również efektywna realizacja oparta na wykorzystaniu informacji o wielomianie jedynie w jednym punkcie, tzn. wykorzystująca do wyznaczenia funkcji kwadratowej wartości wielomianu i jego pierwszej i drugiej pochodnej w aktualnym punkcie.

**MM1**

Rozważmy trzy punkty , , wraz z wartościami wielomianu w tych punktach 𝑓(), 𝑓(), 𝑓(). Skonstruujemy funkcję kwadratową przechodzącą przez te punkty, a następnie wyznaczymy pierwiastki tej funkcji i potraktujemy jeden z nich jako kolejne, poprawione przybliżenie rozwiązania (pierwiastka wielomianu).

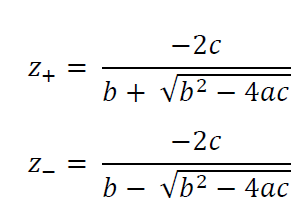
Wprowadzamy zmienną przyrostową 𝑧 = 𝑥− i różnice:

oznaczając poszukiwaną parabolę przez:

Biorąc pod uwagę trzy dane punkty, mamy:

Stąd, do wyznaczenia 𝑎 i 𝑏 należy rozwiązać układ równań liniowych:

Ponieważ interesuje nas pierwiastek paraboli o najmniejszym module (tzn. położony jak najbliżej ), więc do numerycznego wyznaczenia tego pierwiastka najlepiej wykorzystać wzory:



Do kolejnego przybliżenia rozwiązania bierzemy pierwiastek położony jak najbliżej , tj. o mniejszym module:

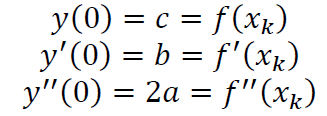


Przed przejściem do następnej iteracji odrzucamy spośród, , punkt położony najdalej od ostatnio wyznaczonego przybliżenia rozwiązania, tj. punktu . Algorytm działa prawidłowo również w przypadku, gdy , prowadzi to do wyznaczenia zera zespolonego.

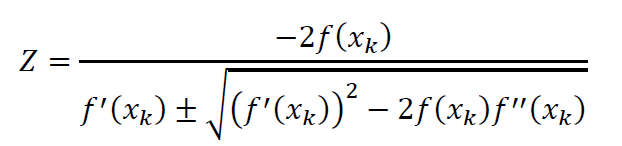
**MM2**

Ta wersja metody wykorzystuje informację o wartości wielomianu i jego pochodnych, pierwszego i drugiego w aktualnym punkcie (przybliżeniu zera). Wersja nieco efektywniejsza obliczeniowo z powodu, iż obliczenie wartości wielomianu w k+1 punktach jest kosztowniejsze niż obliczanie wartości wielomianów i jego k kolejnych pochodnych w jednym punkcie.

Wiemy, iż:



co prowadzi do wzoru na pierwiastki:



Do przybliżenia zera α bierzemy pierwiastek paraboli o mniejszym module:



Gdzie jest wybierany spośród {𝑧+, 𝑧−} w taki sam sposób jak w wersji MM1.

Podobnie jak MM1, MM2 znajduje zespolone pierwiastki wielomianu.

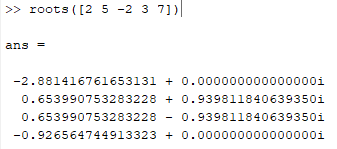
Metoda Müllera jest zbieżna lokalnie, z rzędem zbieżności 1.84. Jest więc (lokalnie) bardziej efektywna niż metoda siecznych, jest niewiele wolniejsza od metody Newtona. Z konstrukcji metody wynika, że może ona być stosowana do poszukiwania zer rzeczywistych i zespolonych nie tylko wielomianów, ale i innych funkcji nieliniowych (analitycznych).

## Wynik

Obraz zawierający mapa

Opis wygenerowany automatycznie

Pierwiastki wielomianu wyliczone za pomocą funkcji roots():



**Newton**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteracja | Punkt startowy | | | |
| -4 | | 1 | |
| x | y | x | y |
| 1 | -4.0581395348837 | 170.1540580308711 | 0.318181818181818 | 7.933628167474899 |
| 2 | -3.4241628096059 | 47.4841874695916 | -1.946260167752689 | -14.579304169415323 |
| 3 | - 3.0549620725214 | 10.8149402214608 | -0.256026234945652 | 6.025504012886236 |
| 4 | - 2.9065660079964 | 1.3505357750028 | -1.492512156700246 | -8.631929231036642 |
| 5 | - 2.8820518766141 | 0. 0332518257881 | -0.945712191777129 | -0.255169129479358 |
| 6 | - 2.8814171807482 | 0.0000219275005 | -0.926714912232297 | -0.001985380334572 |
| 7 | - 2.8814167616533 | 0.0000000000096 | -0.926564754456485 | -0.000000126163282 |
| 8 | -2.8814167616531 | 0.0000000000000 | -0.926564744913323 | -0.000000000000000 |
| 9 | -2.8814167616531 | 0.0000000000000 | -0.926564744913323 | -0.000000000000000 |

Przy dalszych liczbach np. zamiast 1 wybrać 10 potrzeba 15 iteracji.

|  |  |  |
| --- | --- | --- |
| Iteracja | Punkt startowy | |
| 10 | |
| x | y |
| 1 | 7.375356652224 | 7844.104359982050 |
| 2 | 5.413815325157 | 2476.082381276474 |
| 3 | 0.003949016682629 | 0781.968263365834 |
| 4 | 2.853509732667 | 0248.050454548380 |
| 5 | 2.025575824497 | 80.093588627872 |
| 6 | 1.374031648209 | 27.445581717420 |
| 7 | 0.784770788198 | 11.297726902040 |
| 8 | -0.086604407580 | 6.722050834808 |
| 9 | 2.032922919376 | -15.212722844532 |
| 10 | 0.540962524213 | 9.000421126069 |
| 11 | -0.845377666505 | 1.035225367226 |
| 12 | -0.929760643411 | -0.042307755407 |
| 13 | -0.926569044876 | -0.000056846818 |
| 14 | -0.926564744921 | -0.000000000103 |
| 15 | -0.926564744913 | -0.000000000000 |

**MM1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iteracja | Punkty startowe x0, x1, x2 | | | | | | | |
|  | {-10, 0, 10} | | {-2,2,3} | | {2, -2, -3} | | {-50, -49, -48} | |
|  | x | y | x | y | x | y | x | y |
| 1 | -0.0140 + 0.0000i | 6.9576 + 0.0000i | 1.5195 + 0.0000i | 35.1465 + 0.0000i | -2.7571 | -5.6993 | -32.877105816 + 11.382766213i | 0.573901646632334 - 2.669456182012985i |
| 2 | -0.0172 + 0.2166i | 7.0580 + 0.6164i | 1.0109 + 0.0000i | 15.2417 + 0.0000i | -2.8774 | -0.2106 | - 26.376510579 + 13.363875118i | -481365.906018125 – 1329287.732593518i |
| 3 | -1.1766 - 0.1715i | -3.5194 - 2.6712i | 0.8619 + 0.5404i | 6.7809 + 6.6696i | -2.8814 | -0.0011 | -20.090785714 + 14.453855359i | -576802.628071662 – 378751.076674116i |
| 4 | -0.9096 + 0.0030i | 0.2221 + 0.0389i | 0.6555 + 0.7599i | 3.0516 + 2.4015i | -2.8814 | -0.0000 | -13.953173900 + 14.725133625i | -304929.457805091 + 64293.605359580i |
| 5 | -0.9267 - 0.0004i | -0.0024 - 0.0048i | 0.6271 + 0.9171i | 0.8542 - 0.0556i | -2.8814 | 0.0000 | -9.412341374 + 13.899543734i | -91746.421939821 + 115090.725117909i |
| 6 | -0.9266 + 0.0000i | -0.0000 + 0.0000i | 0.6528 + 0.9406i | 0.0071 - 0.0344i | -2.8814 | 0.0000 | -5.756425277 + 12.581348731i | 2319.682676483 + 69136.187982402i |
| 7 | -0.9266 + 0.0000i | -0.0000 + 0.0000i | 0.6540 + 0.9398i | -0.0001 + 0.0001i |  |  | -2.922617144 + 10.972147036i | 22167.486243304 + 236553.73517508i |
| 8 | -0.9266 + 0.0000i | -0.0000 + 0.0000i | 0.6540 + 0.9398i | -0.0000 - 0.0000i |  |  | -0.875252590 + 9.248045403i | 15137.857253547 + 1700.285620649i |
| 9 | -0.9266 + 0.0000i | -0.0000 + 0.0000i | 0.6540 + 0.9398i | -0.0000 - 0.0000i |  |  | 5.52998290 + 7.546738994i | 5928.909825376 – 3999.762855613i |
| 10 | -0.9266 + 0.0000i | -0.0000 + 0.0000i | 0.6540 + 0.9398i | -0.0000 - 0.0000i |  |  | 1.476775210 + 5.956743366i | 907.106999747 – 3222.858823751i |
| 11 |  |  |  |  |  |  | 2.001784147 + 4.535455841i | -642.168305974 – 1419.596620488i |
| 12 |  |  |  |  |  |  | 2.226829567 + 3.312172665i | 648.444990719 – 309.604600075i |
| 13 |  |  |  |  |  |  | 2.232151177 + 2.296249462i | 12.736439055 + 68.107562466i |
| 14 |  |  |  |  |  |  | 2.082227405 + 1.481118292i | 28.307112780 + 20.004763623i |
| 15 |  |  |  |  |  |  | 1.826912307 + 0.846125108i | 19.249731391 – 1.096745719i |
| 16 |  |  |  |  |  |  | 1.505904453 + 0.356281304i | 9.645464288 – 5.101285504i |
| 17 |  |  |  |  |  |  | 1.162390716 – 0.035192098i | 4.583989924 – 2.798619853i |
| 18 |  |  |  |  |  |  | 0.852495346 – 0.368527709i | 1.664727370 + 0.063581096i |
| 19 |  |  |  |  |  |  | 0. 649874972 - 0. 943757030i | 0. 005556893 + 0. 144133549i |
| 20 |  |  |  |  |  |  | 0. 654042975 - 0. 939796609i | -0. 000674451 - 0. 001198952i |
| 21 |  |  |  |  |  |  | 0. 653990758 - 0. 939811850i | -0. 000000265 + 0. 000000085i |
| 22 |  |  |  |  |  |  | 0. 653990753 - 0. 939811841i | 0 |
| 23 |  |  |  |  |  |  | 0. 653990753 - 0. 939811841i | 0 |

**MM2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iteracja | Punkty startowe x0 | | | | | | | |
|  | -2 | | -8 | | 2 | | 8 | |
|  | x | y | X | Y | x | y | x | y |
| 1 | -1.226101239108213 | -4.381123241495223 | -5.599845201238 + 1.653198412628i | 237.684849604767 – 322.997684776859i | 1.217105263157895 + 0.632640281488440i | 7.791489110302638 +18.268880300926476i | 5.163092550790 1.994471231320i | 537.629510166468 + 2591.010183082371i |
| 2 | -0.934466340373082 | -0.104809514192177 | -4.470758055054 + 0.015075876042i | 305.787710856789 –5.942539107302i | 0.742841310880678 + 0.909783890095000i | -0.894643640791758 + 2.299695789648731i | 3.759075115397 - 0. 000525716811i | 654.956962625370 - 0. 328503951709i |
| 3 | -0.926564836706602 | -0.000001213532979 | -3.322010510607 - 0.668071940422i | -0.340767034154 + 68.031724398398i | 0.654380286791291 + 0.939970377880037i | -0.009847824810922 + 0.004027726270736i | 2.363870878929 - 0. 989414778669i | 34.930570190877 - 157.937983815370i |
| 4 | -0.926564744913323 | -0.00000000000000 | -3.002006805545 - 0.039022708619i | 7.039636281266 + 2.583931684862i | 0.653990753251856 + 0.939811840659630i | 0.000000000208382 - 0.000000000921391i | 1.253583319196 - 1.301337210448i | -32.249381390757 - 154.42668869560i |
| 5 |  |  | -2.881416761517 - 0.000000000035i | -0.000000007115 + 0.000000001850i | 0.653990753283228 + 0.939811840639351i | 0.000000000000003 + 0.000000000000000i | 0.577010487981 - 1.128839411265i | -1.076666950734 + 5.678750271861i |
| 6 |  |  | -2.881416761653 - 0.000000000000i | 0.000000000000 + 0.000000000000i |  |  | 0. 655765188385 - 0.943608978952i | -0.099644120266 + 0.037543463304i |
| 7 |  |  |  |  |  |  | 0.653990756897 - 0.939811803782i | 0.000000586538 - 0.000000730094i |
| 8 |  |  |  |  |  |  | 0.653990753283 - 0.939811840639i | 0.000000000000 - 0.000000000000i |

## Podsumowanie:

Metoda Newtona poradziła sobie dobrze, znalazła rozwiązanie w dość szybkim czasie bo poniżej 10 iteracji. Niestety potrafi ona znaleźć tylko rzeczywiste pierwiastki to co jest dużym minusem. Ważny jest tu też punkt startowy, zbyt odległy przyczynia się do znacznego zwiększenia czasu znalezienia rozwiązania.

Podobnie mają się metody MM1 i MM2 dla dobrze dobranych punktów ilość iteracji jest niewielka będąca mniejsza niż przy metodzie Newtona. Dużą korzyścią w tych metodach jest możliwość znalezienia pierwiastków zespolonych. Niestety problemem na jaki się natknąłem przy metodzie MM1 było znalezienie 3 punktów startowych dla jednego pierwiastka. Bez problemu znalazłem 3 punkty lasowe dla 3 różnych pierwiastków. Dlatego też jeden pierwiastek ma tak odległe punkty {-50, -49, -48}. Zalazłem to w sposób „brutal force”. Przez 3 pętle iteracyjne od 50 do -50 dla każdej iteracji wstawiłem punkty do zaimplementowanej metody mm1 a następnie porównywałem czy otrzymany pierwiastek był tym którego szukam. Z przeprowadzonych testów metoda MM2 okazała się mniej zawodniejsza przy szukaniu punktów startowych i szybka.

# Dodatek do zadania 1

## Func

function [y] = func(x)

y = 0.5 \* x \* cos(x) - log(x);

end

## bisection

function [c, iter] = bisection(a, b, eps, iter\_max)

iter = 0; % ilosc iteracji

c = (a + b)/2; % sordek przedzialu

value\_c = func(c); % wartosc dla srodka (rozwiazanie)

tmp\_a = a;

tmp\_b= b;

while((abs(value\_c) > eps)&& iter <= iter\_max)

% koniec gdy osiagniemy dobra dokladnosc albo ilosc iteracji

value\_a = func(tmp\_a); % wartosci na koncach przedzialu

value\_b = func(tmp\_b);

%sprawdzamy obszar izolacji

if(value\_a\*value\_b>0)

error('Brak obszaru izolacji');

end

c = (tmp\_a + tmp\_b)/2; % nowy srodek przedzialu

value\_c = func(c);

if (value\_a\*value\_c < 0) % sprawdzenie zmiany znaku

tmp\_b = c;

end

if (value\_b\*value\_c < 0)

tmp\_a = c;

end

iter = iter + 1;

end

end

## secant

function [c, iter] = secant(a, b, eps, iter\_max)

iter = 0; % ilosc iteracji

c = (a + b)/2; % sordek przedzialu

value\_c = func(c); % wartosc dla srodka (rozwiazanie)

tmp\_a = a;

tmp\_b= b;

value\_a = func(tmp\_a); % wartosci na koncach przedzialu

value\_b = func(tmp\_b);

%sprawdzamy obszar izolacji

if(value\_a\*value\_b>0)

error('Brak obszaru izolacji');

end

while((abs(value\_c) > eps)&& iter <= iter\_max)

% koniec gdy osiagniemy dobra dokladnosc albo limit iteracji

value\_a = func(tmp\_a); % wartosci na koncach przedzialu

value\_b = func(tmp\_b);

c = (tmp\_a\*value\_b - tmp\_b\*value\_a)/(value\_b - value\_a); % nowa wartosc

% dla metody siecznych

value\_c = func(c); % wartosc dla siecnych (rozwiazanie)

tmp\_a = b;

tmp\_b = c;

iter = iter + 1;

end

if iter > iter\_max

disp("Przekorczono limit iteracji");

end

end

## compose\_task\_1

function [results] = compose\_task\_1()

format long;

results = zeros(9,10);

eps = 0.001;

iter\_max = 100;

X = 2:0.1:11; % Nasza dziedzina

Y = arrayfun(@(x) func(x),X); %Wartosc funkcji w dziedzinie

figure;

% rysuje wykres funkcji

plot(X,Y);

% os OX

line([2,11],[0,0],'Color','b')

% dla 10 coraz to mniejszych eps

for i=1:10

[x0, iter] = bisection(3, 7, eps, iter\_max);

results(1, i) = eps;

results(2, i) = x0;

results(3, i) = iter;

[x0, iter] = bisection(7, 7.5, eps, iter\_max);

results(4, i) = x0;

results(5, i) = iter;

[x0, iter] = secant(3, 7, eps, iter\_max);

results(6, i) = x0;

results(7, i) = iter;

[x0, iter] = secant(7, 7.5, eps, iter\_max);

results(8, i) = x0;

results(9, i) = iter;

eps = eps/10; % zminejszam eps

end

end

# Dodatek do zadania 2

## newton

function [iter\_results] = newton(x0,iter\_max)%(x0, eps, iter\_max)

iter = 0; % ilosc iteracji

% mamy tu tylko dwa pierwiastki

% y to wartosc wielomianu w punkcie x0

% y\_2 to wartosc pochodnej wielomianu w punkcjie x0

[y, y\_2] = wielomian(x0);

% zapiansie naszych wynikow w danej iteracji

iter\_results = zeros(iter\_max, 2);

for i=0:iter\_max

%while abs(y) > eps && iter <= iter\_max

x0 = x0 - y/y\_2;

[y, y\_2] = wielomian(x0);

iter = iter + 1;

iter\_results(iter, 1) = x0;

iter\_results(iter, 2) = y;

end

end

## mm1

function [iter\_results] = mm1( x0, x1, x2, iter\_max)

iter = 0; % ilosc iteracji

% zapiansie naszych wynikow w danej iteracji

iter\_results = zeros(iter\_max, 2);

%while abs(y) > eps && iter <= iter\_max

for i = 1:iter\_max

% zmienne przyrostowe

z0 = x0 - x2;

z1 = x1 - x2;

[c, ~]= wielomian(x2);

%tworzymy układ równań do obliczenia a,b i rozwiązujemy go

A = [z0^2 , z0; z1^2, z1];

[f\_x0, ~] = wielomian(x0);

[f\_x1, ~] = wielomian(x1);

B = [f\_x0 - c; f\_x1 - c];

[w] = linsolve(A, B);

a = w(1);

b = w(2);

%wybieramy zmin jako ten o najmniejszym module

if(abs(b + sqrt(b^2 - 4\*a\*c)) >= abs(b - sqrt(b^2 - 4\*a\*c)))

zmin = (-2\*c)/(b + sqrt(b^2 - 4\*a\*c));

else

zmin = (-2\*c)/(b - sqrt(b^2 - 4\*a\*c));

end

%obliczamy kolejne przybliżenie miejsca zerowego

x3 = x2 + zmin;

% zapisujemy wyniki

iter\_results(i, 1) = x3;

[iter\_results(i, 2), ~] = wielomian(x3);

% polozenie punktow

d0 = abs(x3-x0);

d1 = abs(x3-x1);

d2 = abs(x3-x2);

%Odrzucamy spośród x0,x1,x3 to przybliżenie które jest najbardziej

%oddalone of x3

if(d1 < d0)

u = x1;

x1 = x0;

x0 = u;

end

if(d2<d1)

u = x2;

x2 = x1;

x1 = u;

end

%Przygotowujemy się do kolejnej iteracji

x2 = x3;

iter =iter + 1;

end

end

## mm2

function [iter\_results] = mm2(x,iter\_max)

iter = 0;% ilosc iteracji

% zapiansie naszych wynikow w danej iteracji

iter\_results = zeros(iter\_max, 2);

for i = 1:iter\_max

[y, der1] = wielomian(x);

% wspolcynik dla 2 pochodnej wielomianu

der2 = polyval([24 30 -4], x);

%Wybiermay pierwiastek o mniejszym module

z1 = -2\*y/(der1+sqrt(der1^2-2\*y\*der2));

z2 = -2\*y/(der1-sqrt(der1^2-2\*y\*der2));

if abs(z1) > abs(z2)

z\_min = z2;

else

z\_min = z1;

end

%obliczmy kolejne przybliżenie miejsca zerowego

x = x + z\_min;

iter =iter + 1;

%zapisujemy wyniki

iter\_results(iter, 1) = x;

[iter\_results(iter, 2), ~] = wielomian(x);

end

end