Obliczenia Naukowe

Sprawozdanie z laboratorium nr 3

Piotr Kawa

Wrocław 18.11.2017

0. Wprowadzenie

Lista trzecia dotyczyła rozwiązywania równań różnymi metodami – bisekcji, Newtona, siecznych. Zadania zostały napisane w języku Julia.

1. Zadanie 1 - metoda bisekcji

Pierwsze trzy zadania polegały na napisaniu funkcji rozwiązujących równanie $f(x) = 0$ za pomocą metod bisekcji, Newtona oraz siecznych. Musiały one mieć postać przedstawioną w treści zadania oraz zostać umieszczone w module.

1.1 Opis algorytmu

Metoda bisekcji (zwana również metodą połowienia) opiera się na twierdzeniu Bolzana-Cauchy'ego, które brzmi:

„Jeżeli funkcja ciągła $f(x)$ ma na końcach przedziału domkniętego wartości różnych znaków, to wewnątrz tego przedziału, istnieje co najmniej jeden pierwiastek równania $f(x) = 0$ ".

Algorytm rozpoczyna się obliczenia wartości funkcji na końcach podanego wcześniej zakresu - $f(a)$ oraz $f(b)$. Jeśli otrzymane wartości mają te same znaki następuje przerwanie obliczeń (przeciwne znaki na końcach przedziału oznaczają istnienie pierwiastka w tymże zakresie). W przypadku, gdy znaki $f(a)$ i $f(b)$ różnią się następuje przejście do drugiej części algorytmu.

Jest ona wykonywana M razy, gdzie M to zadana zmienna. Część ta polega na wyznaczeniu punktu c będącego środkiem aktualnego przedziału oraz obliczenia wartości funkcji w tym punkcie.

Jeśli spełniony jest warunek $|w| < \delta$ lub $|e| < \varepsilon$ to następuje przerwanie obliczeń oraz zwrócenie wyniku – algorytm osiągnął cel. W przeciwnym razie następuje zmiana przedziału – w zależności od znaku funkcji w jego połowie oraz zaczęcie kolejnej iteracji.

Specyfikacja zadania nie określa podania komunikatu błędu w przypadku wykonania wszystkich iteracji bez zwrócenia wyniku – w tym przypadku zdecydowano, ze zwracane są dane dla ostatniej iteracji.

Utworzona metoda bisekcji miała mieć postać:

$$function\ mbisekcji(f, a :: Float64, b :: Float64, delta :: Float64, epsilon :: Float64)$$

gdzie:

f – funkcja $f(x)$ zadana jako anonimowa funkcja

a, b – końce przedziału początkowego

$delta, epsilon$ – dokładności obliczeń

Oraz zwracać czwórkę (r, v, it, err), gdzie:

r – przybliżenie pierwiastka równania $f(x) = 0$

v – wartość $f(r)$

it – liczba wykonanych iteracji

err – sygnalizacja błędu (0 gdy jego brak, 1 gdy funkcja nie zmienia znaku w przedziale $[a, b]$).

2. Zadanie drugie - metoda stycznych (Newtona)

2.1 Opis algorytmu

Metoda stycznych wymaga wybrania pewnego punktu startowego x_0 – z niego prowadzona jest styczna w $f(x_0)$. Miejsce jej przecięcia z osią OX jest pewnym przybliżeniem, obliczenia powtarzane dla nowego punktu są tak długo jak nie zostanie osiągnięta wymagana dokładność (lub limit iteracji).

Algorytm wygląda następująco. Zostaje obliczona wartość funkcji w podanym w argumencie wywołania punkcie (x_0) – jeśli jej wartość bezwzględna jest mniejsza niż dokładność (ε) następuje zwrócenie aktualnych wartości wraz z komunikatem 0.

Następnie wykonywane są obliczenia, które jak w przypadku poprzedniego algorytmu trwają $maxit$ iteracji. Jeśli wartość bezwzględna pochodnej $f(x)$ jest mniejsza niż ε zwracany jest aktualny wynik z komunikatem 2 – pochodna bliska zeru. Następuje wyliczenie x_1 oraz $f(x_1)$ za pomocą następującego wzoru:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Jeśli spełnione są warunki $|x_1 - x_0| < \delta$ lub $|f(x_1)| < \varepsilon$ następuje zakończenie algorytmu oraz zwrócenie aktualnych danych – $(x_1, f(x_1), k, 0)$ w przeciwnym razie obliczania jest ponownie wartość x_1.

Jeśli żadna wartość nie zostanie zwrócona w czasie trwania pętli następuje zwrócenie aktualnych wartości wraz z komunikatem 1 – nie osiągnięto wymaganej dokładności w zadanej ilości iteracji.

Metoda Newtona miała wyglądać następująco:

$$function\ mstycznych(f, pf, x0 :: Float64, x1 :: Float64, delta :: Float64, epsilon :: Float64, maxit :: Int)$$

gdzie :

f, pf – funkcja f(x) oraz pochodna $f'(x)$ zadane jako anonimowe funkcje

x_0 – przybliżenie początkowe

$delta, epsilon$ – dokładności obliczeń

$maxit$ – maksymalna dopuszczalna liczba iteracji

Zwracanymi przez nią danymi miała być czwórka (r, v, it, err), gdzie:
r – przybliżenie pierwiastka równania $f(x) = 0$

v – wartość $f(r)$

it – liczba wykonanych iteracji

err – sygnalizacja błędu (0 gdy metoda jest zbieżna, 1 gdy nie osiągnięto w $maxit$ iteracji wymaganej dokładności, 2 gdy pochodna jest bliska zeru).

3. Zadanie trzecie - metoda siecznych (Eulera)

3.1 Opis algorytmu

Ostatnia z implementowanych metod polegała na tym, że na pewnym małym odcinku $[x_0, x_1]$ (podanym jako argument) funkcja zmienia się w sposób liniowy – dzięki temu jest możliwe zastąpienie na danym odcinku krzywej sieczną.

Algorytm zaczyna się od wyliczenia wartości funkcji w punktach x_0 oraz x_1. Następnie następują wyliczenia, które wykonywane są $maxit$ razy (zmienna podana w argumencie wywołania). W przypadku gdy wartość bezwzględna funkcji w lewym końcu przedziału ($f(x_0)$) jest większa niż ta w prawym ($f(x_1)$) następuje zamiana zarówno punktów jak i wartości funkcji w nich – operacja $swap(x, y)$.

Następnie następuje przypisanie do $f(x_1)$ i x_1 odpowiednio wartości $f(x_0)$ i x_0. Nowe wartości $f(x_0)$ i x_0 zostają wyliczone z wykorzystaniem następującego wzoru:

$$x_0 = x_0 - f(x_0)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

Jeśli została osiągnięta odpowiednia dokładność przy wyznaczaniu pierwiastka ($|b - a| < \delta$ lub $|f(x_0)| < \varepsilon$) następuje zakończenie wyliczeń oraz zwrócenie czwórki $(x_0, f(x_0), k, 0)$.

Jeśli obliczenia zostaną wykonane $maxit$ razy bez zwrócenia wyniku następuje zwrócenie $(x_0, f(x_0), maxit, 1)$, gdzie 1 to komunikat o błędzie (brak osiągnięcia odpowiedniej dokładności).

Opisana wyżej metoda miała zostać przedstawiona w następującej formie:

$$function\ msiecznych(f, x0 :: Float64, x1 :: Float64, delta :: Float64, epsilon :: Float64, maxit :: Int)$$

gdzie :

f – anonimowa funkcja $f(x)$

x_0, x_1 – końce przedziału początkowego

$delta, epsilon$ – dokładności obliczeń

$maxit$ – maksymalna dopuszczalna liczba iteracji

Zwracanymi przez nią danymi miała być czwórka (r, v, it, err), gdzie: r – przybliżenie pierwiastka równania $f(x) = 0$

v – wartość $f(r)$

it – liczba wykonanych iteracji

err – sygnalizacja błędu (0 gdy metoda jest zbieżna, 1 gdy nie osiągnięto w $maxit$ iteracji wymaganej dokładności).

Wszystkie opisane wcześniej algorytmy znajdowały się w specjalnym module nazwanym $ComputeLinearEquations$. Zostały do niego dopisane testy – składały się one z 2 funkcji mających na celu przetestowanie poprawności implementacji – odpowiednio:

1) $f(x) = x - 5$

2) $f(x) = \ln(x)$

Wynik otrzymany dzięki funkcji był porównywany z wynikiem uzyskanym dzięki ręcznym obliczeniom bądź też dzięki użyciu zewnętrznych narzędzi. Wyświetlany był odpowiedni komunikat w zależności od tego czy testy zostały zdane czy też nie.

4. Zadanie czwarte

4.1 Opis zadania

Zadanie czwarte polegało na wyznaczeniu pierwiastka równania $\sin(x) - \left(\frac{1}{2}x\right)^2 = 0$. W tym celu należało wykorzystać stworzone wcześniej metody

a) bisekcji, gdzie przedział początkowy to $[1.5, 2]$, $\delta = \frac{1}{2}10^{-5}$, $\varepsilon = \frac{1}{2}10^{-5}$.

b) Newtona, gdzie przybliżenie początkowe to $x_0 = 1.5$, $\delta = \frac{1}{2}10^{-5}$, $\varepsilon = \frac{1}{2}10^{-5}$.

c) siecznych, gdzie przybliżenie początkowe to $x_0 = 1$, $x_1 = 2$ $\delta = \frac{1}{2}10^{-5}$, $\varepsilon = \frac{1}{2}10^{-5}$.

4.2 Rozwiązanie

W celu wyliczenia metodą Newtona została obliczona pochodna równania $\sin(x) - \left(\frac{1}{2}x\right)^2$, która wynosi $\cos(x) - \frac{1}{2}x$.

4.3 Wyniki

Zwrócone przez funkcje wartości prezentowały się następująco:

Metoda	r	v	it
bisekcji	1.933753967285156	$-2.7027680138402843e-7$	16
Newtona	1.933753779789742	$-2.2423316314856834e-8$	4
siecznych	1.933753644474301	$1.564525129449379e-7$	4

4.4 Wnioski

Naturalnie podana funkcja ma dwa miejsca zerowe, jednakże podane dane służyły tylko do obliczenia jednej z nich – znajdującej się w punkcie 1.93375376282702125331.

Różnice między wynikami otrzymanymi dzięki powyższym metodom zaczynają się dopiero od szóstego miejsca po przecinku – najdokładniejsza ze wszystkich jest metoda Newtona.

W celu obliczenia wyniku najwięcej iteracji potrzebowała metoda bisekcji – było ich 16. Algorytm ten nie stosuje żadnego przybliżenia, dzieli podany przedział na dwie części – duża ilość iteracji wynikała z umiejscowienia miejsca zerowego względem zastosowanego przedziału. Pozostałe dwa algorytmy zrobiły to w ciągu czterech iteracji.

Wartości r nie są równe faktycznej wartości dla której funkcja przyjmuje $0,$ jest to jednak zależne od przybliżenia – jego zmiana z pewnością ma wpływ na uzyskany wynik.

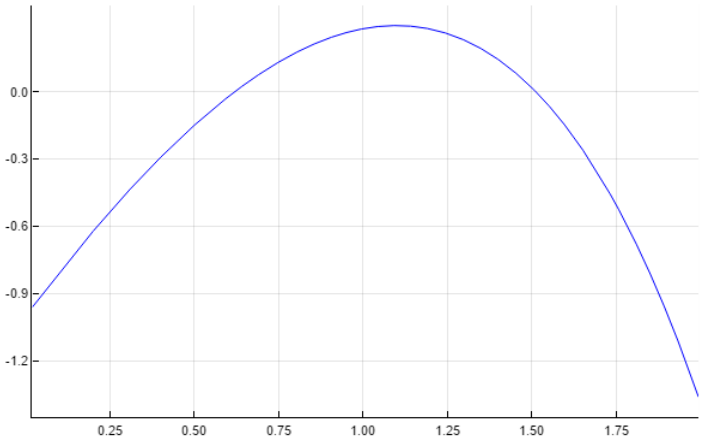
5. Zadanie piąte

5.1 Opis zadania

Zadanie piąte polegało na wykorzystaniu metody bisekcji w celu znalezienia wartości zmiennej x, dla której przecinają się wykresy funkcji $y = 3x$ oraz $y = e^x$. Należało wykorzystać dokładności $\delta = \varepsilon = 10^{-4}$.

5.2 Rozwiązanie

Przyrównując do siebie funkcje $y = 3x$ oraz $y = e^x$ otrzymujemy $3x - e^x = 0$. Wygenerowany za pomocą języka Julia wykres tej funkcji prezentuje się następująco:



W celu znalezienia wartości zmiennej x musimy najpierw wybrać przedziały, które będą argumentami naszej funkcji. Pierwszym odruchem mogłyby być ustalenie przedziału jako $[0, 2]$.

Ten przedział ma jednak obydwa końce mają o wartości ujemnej, a więc nie jest poprawnym argumentem dla metody bisekcji. $[0, 1]$ oraz $[1, 2]$ spełniają jednak jej założenia.

5.3 Wyniki

W poniższej tabeli przedstawione zostały wartości zwrócone przez funkcję $mbisekcji()$ użytą do rozwiązania zadania dla ww. przedziałów.

$Przedział$	r	v	it	err
$[0,1]$	0.619140625	$9.066320343276146e-5$	9	0
$[1,2]$	1.5120849609375	$7.618578602741621e-5$	13	0

5.4 Wnioski

Celem zadania było znalezienie wartości x. Analiza wykresu funkcji okazała się wielce pomocna – dzięki niej wiadomo, że jakie dane początkowe warto zastosować. Wynikami otrzymanymi przez wywołanie funkcji były odpowiednio 0.619140625 oraz 1.5120849609375. Analizując podany wcześniej wykres widać, że wartości te odpowiadają w przybliżeniu wartościom miejsc zerowych tejże funkcji. Naturalnie wyniki te mogłyby być dokładniejsze przy zastosowaniu innych przybliżeń bądź dokładności.

6. Zadanie szóste

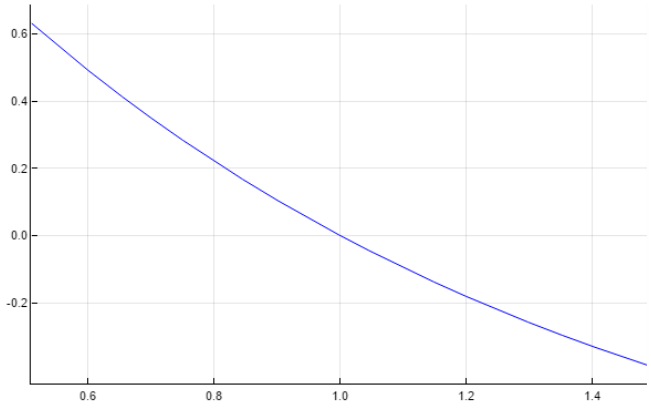
6.1 Opis zadania

Celem zadania szóstego było znalezienie miejsc zerowych dwóch funkcji: $f_1(x) = \mathrm{e}^{1-x} - 1$ oraz $f_2(x) = \mathrm{xe}^{-x}$. W tym celu należało skorzystać z metod bisekcji, Newtona oraz siecznych z dokładnościami $\delta = 10^{-5}$, $\varepsilon = 10^{-5}$. Przedział i przybliżenia początkowe miały zostać odpowiednio dobrane.

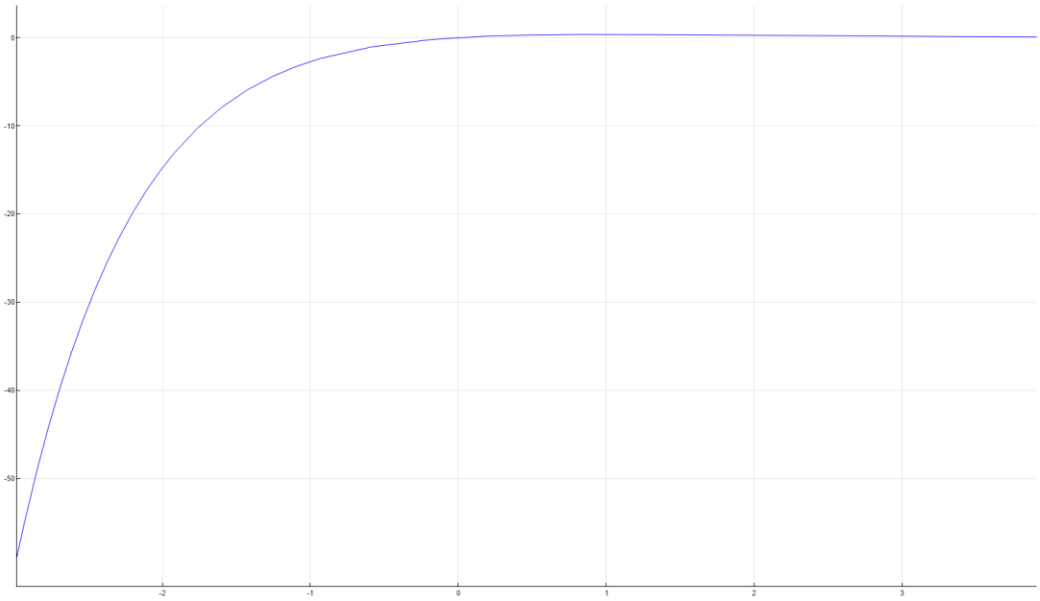
Drugą częścią zadania było sprawdzenie wpływu zmiany przybliżenia początkowego na wynik obydwu funkcji – odpowiednio dla f_1 $x_0 \in (1, \infty]$, a dla f_2 $x_0 = 1$ oraz $x_0 > 1$.

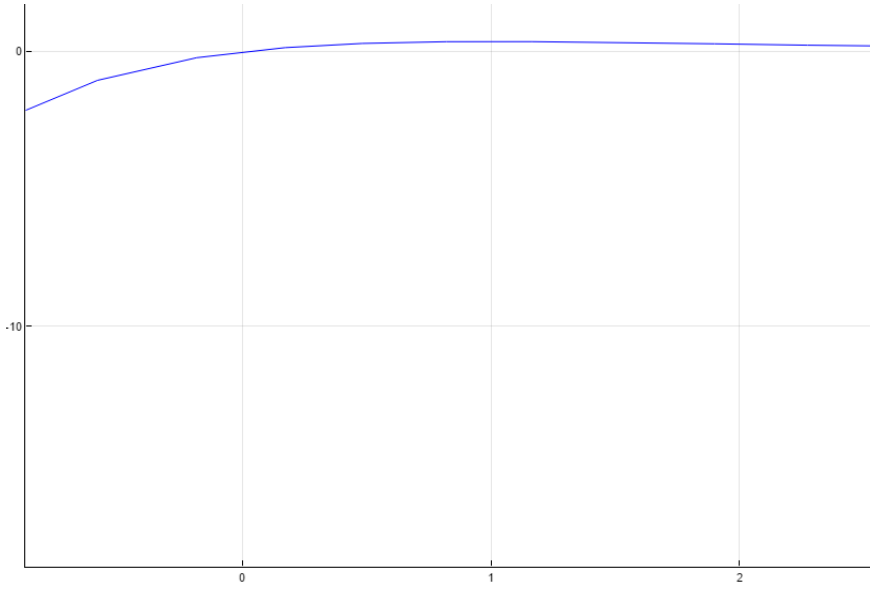
6.2 Rozwiązanie i wyniki

W celu rozwiązania zadania wykorzystane zostały ponownie funkcje napisane w ramach poleceń zadań 1-3. Analiza wykresu funkcji wielce ułatwia odpowiedni dobór przedziałów i przybliżeń początkowych.



$f_1(x) = e^{1-x} - 1$





$$f_2(x) = xe^{-x}$$

6.2.1. $f_1(x) = \mathrm{e}^{1-x} - 1$

Ręczne obliczenie miejsca zerowego tejże funkcji (za pomocą podstawienia pod zmienną wartości 1.0) dało punkt odniesienia wykorzystany w celu określenia wiarygodności rozwiązań.

6.2.1.1 Metoda bisekcji

W celu zastosowania metody bisekcji należy ustalić pewne przedziały początkowe. Pierwszym sprawdzonym przedziałem był przedział $[0.0, 2.0]$, który okazał się zwracać dokładnie szukane wartości – $(1.0, 0.0, 1, 0)$ – spowodowane było to tym, ze miejsce zerowe znajdowało się dokładnie w środku sprawdzanego przedziału.

Jak wiadomo wymogiem poprawnego przedziału początkowego jest to, że znaki na końcach przedziału są różne. Dlatego właśnie zostały sprawdzone eksperymentalnie inna wartość końców przedziałów: $[-1.0, 2.0]$ która zwróciła wynik

$$(0.99999237060, 7.6294236350e-6, 17, 0).$$

Nie jest on dokładnym wynikiem, który był szukany, jednakże bardzo bliskim mu. Powodem tego stanu rzeczy jest wybrany przedział – widoczny jest również kontrast między ilością iteracji – w pierwszym przypadku wymagana była tylko jedna, w drugim aż siedemnaście.

6.2.1.2 Metoda Newtona

Wyniki poprzedniej metody oraz wykres funkcji był duża podpowiedzią – zastosowany został punkt $x_0 = 0.5$ – wynik wynosił $(0.99999999988, 1.1216494399e-10, 4, 0)$.

Wartym przetestowania był również przypadek, który ukazywał co by się stało, gdyby dane dobierane byłyby bez uprzedniej wiedzy o wyglądzie funkcji – zastosowano $x_0 = 7.3$.

Otrzymanym wynikiem była czwórka $(-37.2719101, 4.1809924e16, 500, 1)$ – wynik dalece odległy od oczekiwanego, zwracający komunikat o błędzie – brak wymaganej dokładności w podanej ilości iteracji.

Jednym z celów tego zadania było sprawdzenie wyników dla $x_0 \in (1, \infty]$. Testowane było kilkadziesiąt różnych danych – najpierw testowana była każda liczba od 2.0 do 100.0 z krokiem co 1.0. Dokładności wynosiły 10^{-5}, natomiast $maxit = 10000$. Dane dla $x_0 \in [2.0, 7.0]$ prezentowały się następująco:

x_0	r	v	it	err
2.0	0.9999999810061002	$1.8993900008368314e-8$	5	0
3.0	0.9999999710783241	$2.892167638712806e-8$	9	0
4.0	0.9999999995278234	$4.721765201054495e-10$	21	0
5.0	0.9999996427095682	$3.572904956339329e-7$	54	0
6.0	0.9999999573590406	$4.264096031825204e-8$	147	0
7.0	0.9999999484165362	$5.15834650549607e-8$	401	0

W przypadku gdy $12 \geq x_0 \geq 8$ otrzymane dane wyglądały następująco – $(NaN, NaN, 10000, 1)$. Powodem tego stanu rzeczy było dzielenie wartości funkcji i jej pochodnej, które wynosiły $Infinity$. $f(x)$ miała taką wartość ze względu na dużą wartość e^x. Pochodna tejże funkcji naturalnie również wynosiła $Infinity$. Wynikiem działania było więc NaN. Remedium na to byłaby na przykład zmiana arytmetyki, jednakże specyfikacja zadania zakładała $Float64$.

Sprawa wygląda inaczej dla kolejnych argumentów – tutaj już następuje stwierdzenie, że wartość pochodnej jest mniejsza niż ε, dlatego właśnie dla każdego kolejnego argumentu (niezależnie czy należały one do pierwszego zestawu danych – liczb mniejszych od 100 czy też były to liczby 1000 bądź 10000) zwracana była wartość x_0 wraz z komunikatem błędu 2 – pochodna była bliska zeru.

6.2.1.3 Metoda siecznych

Metoda siecznych była wykonywana dla podanych wartości – $x_0 = 0.0, x_0 = 2.0$. Otrzymane wyniki wynosiły $(1.00000175, -1.759711721e-6, 6, 0)$.

6.2.2. $f_2(x) = \mathrm{x}\mathrm{e}^{-x}$

W przypadku drugiej funkcji również możliwe wyliczenie miejsca zerowego funkcji poprzez podstawienie zmiennej do równania – tutaj $x = 0.0$.

6.2.2.1 Metoda bisekcji

Wykorzystanym w obliczeniach przedziałem był $[-1.0, 1.0]$ – zgodnie z oczekiwaniami wynik wynosił:

$$(0.0, 0.0, 1, 0).$$

Drugim ze sprawdzonych przedziałów był $[-1.0, 1.01]$, wynik był równy:

$$(7.6293945301e-7, 7.6293887094e-7, 16, 0).$$

Widoczna jest spora różnica w ilości iteracji, przy jedynej różnicy danych wejściowych wynoszącej 0.01 w przypadku prawej strony przedziału. Jest to spowodowane tym, że miejsce zerowe w pierwszym przypadku leżało dokładnie po środku przedziału, drugi eksperyment potrzebował więcej iteracji do zwrócenia mniej dokładnego wyniku.

6.2.2.2 Metoda Newtona

Wiedza o tym jak wygląda wykres funkcji była ponownie pomocna – wykorzystane x_0 wynosiło 0.5 i zwróciło wynik $(-3.06424934e-7, -3.06425028e-7, 5, 0)$.

Należało również sprawdzić czy argumentem x_0 może być liczba 1.0. W tym celu przeprowadzono kilka testów – funkcja była wykonywana dla następujących wartości $maxit$ - $1, 10, 100, 1000$. Zwracane wyniki wynosiły $(1.0, 0.36787944117144233, 0, 2)$. Liczba 1.0 nie może być argumentem – wartość pochodnej w tym punkcie jest równa 0.

Również dla $x_0 > 1$ otrzymywane wartości są niepoprawne – od początku wszystkie zwrócone miejsca zerowe są dalekie od właściwych (wynoszą około 14) natomiast od $x_0 = 15$ są one równe przybliżeniu początkowemu. Nie zostaje również podany komunikat o błędzie – wszystkie wyniki zwracają 0. Powód tego stanu rzeczy jest następujący - z powodu umiejscowienia warunku początkowego ($abs(v) < \varepsilon$) od pewnego momentu zwracane są złe wyniki – wartość funkcji jest bardzo bliska zera, ale jednocześnie jest mniejsza od ε. Kolejne rezultaty będą wyglądały podobnie ponieważ ta funkcja jest zbieżna do zera.

6.2.2.3 Metoda siecznych

Nie zawsze dane które wyglądają poprawnie są właśnie takie. Pierwszym eksperymentem było sprawdzenie wyniku dla danych $x_0 = -10.0$ oraz $x_1 = 10.0$. Zwrócone dane wynosiły $(9.9999999587, 0.000453999314, 1, 0)$ – wyglądały one bardzo przekonywująco – wartość funkcji w podanym punkcie była bardzo bliska zeru. Uwagę jednak zwróciła ilość iteracji – tylko jedna. Okazało się, że były to jednak błędne wyniki – wykres ten miał miejsce zerowe w punkcie 0.0. Wartość funkcji w punkcie wynikała ze specyfiki tejże funkcji – od momentu przekroczenia osi OX wykres znajdował się bardzo blisko jej.

W celu upewnienia się została zastosowana inna para danych – tym razem $[-0.75, 1.0]$. Dane zwrócone były już poprawne – $(-2.20931426030e-6, -2.20931914137e-6, 13, 0)$.

6.3 Wnioski

Poprzednie zadania pokazały, że metoda bisekcji jest wolna – wymaga wielu iteracji by osiągnąć dane przybliżenie miejsca zerowego. Jednakże niewątpliwą jej zasługą jest to, że niezależnie jak dalekie od miejsca szukanego końce przedziału wybierzemy, tak długo jak

spełniony jest warunek o różnych znakach na ich końcach, to zwróci ona poprawne przybliżenie szukanego wyniku – jest ona zbieżna globalnie.

Eksperymenty wykonane przy metodzie siecznych i stycznych pokazują, że jest to bardzo ważne – najbardziej widoczne było to w przypadku funkcji $f_2(x)$. Metody te dla pewnych danych zwracały wyniki, które bez wiedzy o wyglądzie funkcji można by było zinterpretować jako poprawne – wynikało to z tego, że są one zbieżne lokalnie. Bez wiedzy o tym jak wygląda wykres funkcji, bądź też bez zastosowania metody bisekcji wyniki te mogłyby być (błędnie) przedstawione jako poprawne.

Parametrem, który z pewnością jest ważny w przypadku porównywania powyższych algorytmów jest również szybkość ich wykonywania. Metody stycznych i siecznych miały podobne ilości wymaganych iteracji – na przykład w zadaniu trzecim, w innych przypadkach zazwyczaj lepiej wypadała metoda Newtona. Jednakże interpretowanie tejże ilości jako szybkości jest niepoprawne. Wszystkie przedstawione metody różnią się od siebie w kwestii wykonywanych przez nie operacji, a metoda Newtona zawiera operacja obliczenia pochodnej. Może być ona wymagająca obliczeniowo, co oznacza, że mimo, iż ilość iteracji będzie mniejsza bądź równa, to nakłady czasowe potrzebne na wykonanie algorytmu będą zgoła różne. Czasem również może się zdarzyć tak, że w danym punkcie nie ma pochodnej – co zaburzy pracę algorytmu.

Dzięki powyższym wnioskom można dojść do tego, że nie ma jednej metody, która będzie idealna w każdych warunkach, warto jest mądrze wybrać narzędzie do wykonywanego zadania z uwzględnieniem tego na czym nam zależy – czy to będzie szybkość wyniku czy też jego dokładność.