OBLICZENIA NAUKOWE

Lista nr 3.

Patrycja Paradowska
nr indeksu: 244952
Prowadzący laboratoria: Mgr inż. Marta Słowik

23.11.2019r.

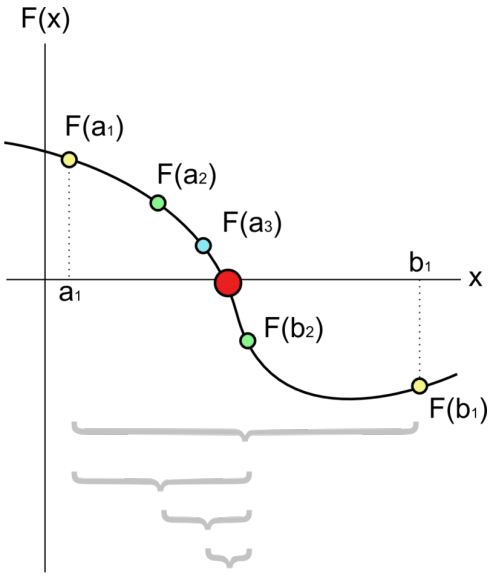
Lista 3. dotyczy problemu rozwiązywania równań nieliniowych, używając różnych metod - bisekcji, Newtona oraz siecznych. Do wymienionych metod napisano również programy testujące.

1 Zadanie 1. - Metoda bisekcji

1.1 Przedstawienie problemu

Pierwsze zadanie obejmowało implementację w języku `Julia` funkcji rozwiązującej równanie $f(x) = 0$ metodą bisekcji, znanej również pod nazwą metody połowienia przedziału. Korzysta ona z konsekwencji własności Darboux dla funkcji ciągłej f: jeżeli funkcja f jest ciągła w przedziale $[a, b]$ i jeśli $f(a) \cdot f(b) < 0$, a więc f zmienia swój znak w przedziale $[a, b]$, to musi mieć zero w (a, b). Z danej własności korzystamy w następujący sposób: Jeśli $f(a) \cdot f(b) < 0$, to obliczamy $c = \frac{1}{2}(a + b)$ oraz sprawdzamy, czy $f(a) \cdot f(c) < 0$.
1. Jeżeli tak, to f ma zero w przedziale $[a, c]$, wtedy pod b podstawiamy c.
2. W przeciwnym razie zachodzi $f(c) \cdot f(b) < 0$, wtedy pod a zostaje podstawione c.

W obu przypadkach otrzymujemy nowy przedział $[a, b]$ dwa razy krótszy od poprzedniego i zawierający zero funkcji f. Możemy więc ponownie przeprowadzić na nim powyższe rozumowanie. Zastosowanie takiego schematu prowadzi do znalezienia jednego zera funkcji f (nie wszystkich). Jeśli $f(a) \cdot f(c) = 0$, to $f(c) = 0$ i tym samym znaleźliśmy miejsce zerowe f.



Wykonując obliczenia w arytmetyce `Float32` czy `Float64` musimy zdefiniować inny warunek zakończenia obliczeń, ponieważ w praktyce pojawiają się błędy zaokrągleń i otrzymanie $f(c) = 0$ jest mało prawdopodobne. Dlatego za kryterium zakończenia obliczeń należy przyjąć wartość odpowiednio bliską zeru w danej arytmetyce. Definiujemy wartości δ i ϵ , które kolejno definiują dostatecznie małą wartość błędu przybliżeń wykonywanych obliczeń oraz pożądaną bliskość otrzymanej wartości iloczynu $f(a) \cdot f(c)$ do zera.

1.2 Opis algorytmu

Zaprezentowany poniżej algorytm przedstawia schemat działania metody bisekcji.

Algorithm 1: Metoda bisekcji

Dane : (f, a, b, delta, epsilon), gdzie:
- f — funkcja $f(x)$ zadana jako anonimowa funkcja,
- a, b — końce przedziału początkowego,
- delta, epsilon — dokładność obliczeń

Wyniki: (r, v, it, err), gdzie:
- r — przybliżenie pierwiastka równania $f(x) = 0$,
- v — wartość f(r),
- it — liczba wykonanych iteracji,
- err — sygnalizacja błędu:
  - 0 — brak błędu,
  - 1 — funkcja nie zmienia znaku w przedziale [a, b].

```
1  Function MBISEKCJI(f, a, b, delta, epsilon)
2      r ← 0
3      v ← 0
4      it ← 0
5      err ← 0
6      e ← b − a
7      f_a ← f(a)
8      f_b ← f(b)
9      if SGN(f_a) = SGN(f_b) then
10         err ← 1
11         return (r, v, it, err)
12     end
13     while e > epsilon do
14         it ← it + 1
15         e ← e/2
16         r ← a + e
17         v ← f(r)
18         if |e| < delta ∨ |v| < epsilon then
19             return (r, v, it, err)
20         end
21         if SGN(v) ≠ SGN(f_a) then
22             b ← r
23             f_b ← v
24         else
25             a ← r
26             f_a ← v
27         end
28     end
```

Na początku sprawdzane jest, czy funkcja zmienia znak na zadanym przedziale. Jeżeli funkcja f nie zmienia znaku w zadanym przedziale $[a, b]$, metody nie można zastosować i zwracany jest kod błędu `1`. W przeciwnym przypadku, rozpoczyna się wykonywanie pętli `while`. Dopóki długość przedziału $|b - a|$ jest większa od zadanej dokładności obliczeń ϵ, wyznaczany jest środek przedziału $[a, b]$. Liczona jest również wartość funkcji f dla wyznaczonego środka. Warunkiem końca wykonywania pętli jest osiągnięcie przedziału o szerokości mniejszej od *epsilon*. Jeżeli osiągnięto zadaną precyzję dla szerokości przedziału lub wartość funkcji w punkcie r jest mniejsza od zadanego ϵ, czyli jest wystarczająco bliska zeru, to zwracany jest jako rozwiązanie rezultat obliczeń. W przeciwnym wypadku wybierany jest nowy przedział $[a, r]$ lub $[r, b]$,

w którym funkcja zmienia znak, i zaczynamy działanie od nowa, czyli szukamy w tym nowym podprzedziale przybliżenia miejsca zerowego.

Warto zwrócić uwagę na fakt, że jeśli wezmiemy dany punkt środkowy c, to obliczany jest on w sposób: $c \leftarrow a + (b-a)/2$, co jest lepsze z numerycznego punktu widzenia. Korzystniejszy rozwiązaniem jest bowiem obliczanie nowej wielkości poprzez niewielką poprawkę poprzedniej. Istnieje ryzyko, że zastosowannie instrukcji $c \leftarrow (a+b)/2$ mogłoby spowodować w wyjątkowych przypadkach znalezienie się punktu c nawet poza przedziałem $[a, b]$, dlatego ten sposób nie został użyty. Drugą sprawą bedzie sposób badania zmiany znaku wartości funkcji. Zmiana znaku została zbadana za pomocą nierówności $SGN(v) \neq SGN(f_a)$. Miało to na celu pozbycie się zbędnej operacji mnożenia przy sprawdzaniu $f(a) \cdot f(c) < 0$, mogącej spowodować w trakcie obliczeń numerycznych nadmiar lub niedomiar - w przypadku bardzo małych lub bardzo dużych liczb iloczyn mógłby przyjąć wartość 0 lub ∞. Unikamy tego w prezentowanej implementacji algorytmu bisekcji. Podsumowując, program przewiduje trzy sposoby zakończenia obliczeń: $e > \epsilon$ mówi o tym, kiedy jeszcze jest możliwa iteracja w danym przedziale, ponadto obliczenia są przerywane kiedy błąd jest dostatecznie mały ($|e| < \delta$) lub gdy $f(c)$ jest dostatecznie bliskie zeru ($|v| < \epsilon$).

2 Zadanie 2. - Metoda stycznych (metoda Newtona)

2.1 Przedstawienie problemu

W kolejnym zadaniu należało zaimplementować w języku `Julia` algorytm, który wyznacza miejsce zerowe danej funkcji przy użyciu metody stycznych (metody Newtona). W metodzie Newtona przyjmuje się następujące założenia dla funkcji f:
1. W przedziale $[a, b]$ znajduje się dokładnie jeden pierwiastek.
2. Funkcja ma różne znaki na krańcach przedziału, tj. $f(a) \cdot f(b) < 0$.
3. Pierwsza i druga pochodna f mają stały znak w przedziale $[a, b]$.

Metoda Newtona wykorzystuje do obliczenia miejsca zerowego funkcji f tak zwaną metodę linearyzacji funkcji, czyli zastąpienie f funkcją liniową. Jest nią suma dwóch początkowych składników we wzorze Taylora dla funkcji f. Jeżeli:

$$f(x) = f(c) + f^{'}(c)(x-c) + \tfrac{1}{2!}f^{''}(c)(x-c)^2 + \dots$$

to w wyniku zastosowania metody linearyzacji otrzymujemy funkcję liniową l:

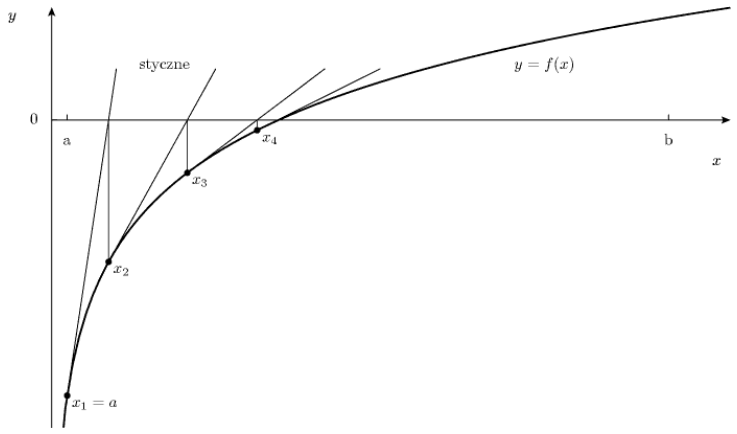
$$l(x) = f(c) + f^{'}(c)(x-c).$$

Jest to dobre przybliżenie funkcji f w pobliżu argumentu c. Do obliczania kolejnych przybliżeń pierwiastka r zadanej funkcji f wykorzystywany jest następujący wzór:

$$x_{n+1} = x_n - \frac{f(x_n)}{f^{'}(x_n)}$$

gdzie $n \leqslant 0$.

Wizualizacja działania metody dla pierwszych czterech kroków prezentuje się następująco:



Metoda Newtona jest zazwyczaj szybsza niż metody bisekcji i siecznych, ponieważ posiada zbieżność kwadratową. W chwili, kiedy przybliżenia tworzone metodą Newtona są wystarczająco bliskie pierwiastka, staje się ona tak szybko zbieżna, że kilka przybliżeń umożliwia otrzymanie maksymalnej dokładności.Dzieje się tak, gdyż przy spełnionych założeniach błąd maleje kwadratowo wraz ze wzrostem ilości iteracji. Metoda ta posiada jednak wadę - nie zawsze jest zbieżna, dlatego można stosować ją w kombinacji z jakąś wolniejszą metodą już zbieżną globalnie. W wielu przypadkach, szczególnie gdy punkt startowy jest daleko od szukanego pierwiastka, metoda bywa rozbieżna. Za kolejną wadę można uznać także konieczność liczenia pochodnej funkcji. Dodatkowo wprowadza się zabezpieczenie przed nadmierną ilością wykonywanych iteracji, których przekroczenie wiąże się z niepowodzeniem obliczeń.

2.2 Opis algorytmu

Utworzona na potrzeby zadania implementacja algorytmu w języku `Julia` przed przystąpieniem do właściwych obliczeń dokonuje sprawdzenia, czy możliwe jest zastosowanie metody. Dokonuje się to za pomocą sprawdzenia, czy pochodna funkcji f nie jest odpowiednio bliska zeru, czyli na potrzeby tej metody numerycznej - mniejsza od wartości *epsilon*. Jeśli dzieje się tak, że jest, to algorytm kończy działanie z kodem błędu 2. Następnie dla zadanej maksymalnej dopuszczalnej liczby iteracji `maxit` wyznaczane są kolejne przybliżenia zera funkcji za pomocą wcześniej wymienionego wzoru, wynikającego z definicji. Obliczana zostaje x_{n+1} oraz wartość f dla tego argumentu. Kiedy znalezione zero jest już wystarczająco dokładne i mieści się w zadanej dokładności, bądź też odległość kolejnych przybliżeń jest dostatecznie mała, nasz algorytm kończy działanie i zwraca otrzymane wyniki. W przeciwnym wypadku wykonywana jest następna iteracja. W razie nieosiągnięcia wyniku w zadanej liczbie iteracji `maxit`, zwracany jest kod błędu 1.

Na kolejnej stronie przedstawiony jest pseudokod opisywanego algorytmu:

Algorithm 2: Metoda stycznych

Dane : (f, pf, x_0, delta, epsilon, maxit), gdzie:
- f, pf — funkcja $f(x)$ oraz $f'(x)$ zadane jako anonimowe funkcje,
- x_0 — przybliżenie początkowe,
- delta, epsilon — dokładność obliczeń,
- maxit — maksymalna dopuszczalna liczba iteracji

Wyniki: (r, v, it, err), gdzie:
- r — przybliżenie pierwiastka równania $f(x) = 0$,
- v — wartość $f(\mathsf{r})$,
- it — liczba wykonanych iteracji,
- err — sygnalizacja błędu:
  - 0 — metoda zbieżna,
  - 1 — nie osiągnięto wymaganej dokładności w maxit iteracji,
  - 2 — pochodna bliska zeru.

```
1  Function MSTYCZNYCH(f, pf, x_0, delta, epsilon, maxit)
2      r ← 0
3      v ← 0
4      r ← x_0
5      v ← f(r)
6      it ← 0
7      err ← 0
8      if |pf(r)| < epsilon then
9          err ← 2
10         return (r, v, it, err)
11     end
12     for it ← 1 to maxit do
13         x_1 ← r − v/pf(r)
14         v ← f(x_1)
15         if |x_1 − r| < delta ∨ |v| < epsilon then
16             r ← x_1
17             return (r, v, it, err)
18         end
19         r ← x_1
20     end
21     err ← 1
22     return (r, v, it, err)
```

3 Zadanie 3. - Metoda siecznych

3.1 Przedstawienie problemu

Zadanie polegało na napisaniu funkcji rozwiązującej metodą siecznych równanie $f(x) = 0$ dla danej funkcji f. Metoda siecznych jest również metodą iteracyjną, ale w przeciwieństwie do metody Newtona, nie wymaga obliczania pochodnej funkcji. Zamiast pochodnej jest tu obliczany iloraz różnicowy:

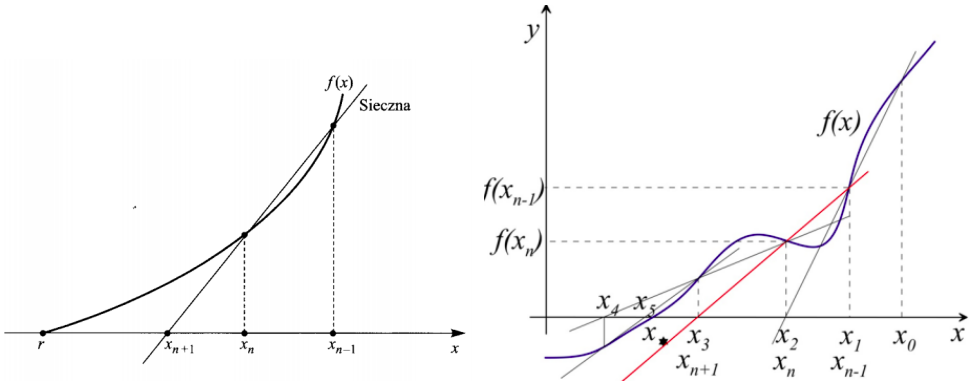
$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Kolejne iteracje metody siecznych obliczane są więc przy pomocy wzoru:

$$x_{n+1} := x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}, \qquad n \geqslant 1. \tag{1}$$

Dostrzegamy, że x_{n+1} wyrażone jest za pomocą x_n oraz x_{n-1}. Potrzebujemy zatem dwóch punktów początkowych, jednak każde nowe x_{n+1} wymaga już tylko obliczenia jednej nowej wartości funkcji f, w przeciwieństwie do metody Newtona, dla której w każdym kroku oprócz $f(x)$ należało także obliczyć $f'(x)$.

W graficznym porównaniu do metody Newtona metoda siecznych różni się jedynie zastąpieniem stycznej do krzywej sieczną, co ilustruje poniższy rysunek:



Metoda siecznych zbiega nadliniowo, w tempie $\frac{1+\sqrt{5}}{2} \approx 1.62$, a więc wolniej od metody Newtona (ale szybciej od metody bisekcji). Jednak każdy krok metody siecznych wymaga obliczenia tylko jednej wartości funkcji, a każdy krok metody Newtona aż dwóch (wartość funkcji i wartość pochodnej). W obu metodach najbardziej kosztowne jest obliczanie właśnie tych wartości i w tym sensie para kroków metody siecznych jest porównywalna z jednym krokiem metody Newtona.

3.2 Opis algorytmu

W zaimplementowanym algorytmie zastosowano metodę siecznych, w której kolejne wartości funkcji mają nierosnące moduły. Algorytm rozpoczyna działanie wyliczając wartości funkcji f dla zadanych x_0 oraz x_1. Następnie przechodzi do próby znalezienia przybliżenia pierwiastka f w zadanej liczbie iteracji `maxit`. Na początku każdego przebiegu pętli sprawdzany jest warunek, czy $|f_{x_0}| > |f_{x_1}|$. Jeżeli nierówność zachodzi, to wartości x_0 oraz x_1 są zamieniane miejscami. Ma to zapewnienić, że moduły funkcji f w kolejnych punktach x_n nie rosną. Potem obliczamy iloraz różnicowy, ale także wartość dla kolejnego x_{n+1}. Obliczenia kończą się, gdy przybliżenie pierwiastka jest dokładne - odległość punktów przecięcia siecznej z wykresem funkcji jest mniejsza od *delta* - lub wartość funkcji dla bieżącego argumentu jest bliska zeru - mniejsza od *epsilon*-, a algorytm zwraca wynik. W przypadku nieosiągnięcia wymaganej dokładności w zadanej liczbie iteracji, zwracany jest kod błędu 1. Szczegóły działania przedstawione są poniżej.

Algorithm 3: Metoda siecznych

Input : (f, x_0, x_1, *delta*, *epsilon*, maxit), gdzie:

- f — funkcja $f(x)$ zadana jako anonimowa funkcja,
- x_0, x_1 — przybliżenia początkowe,
- *delta*, *epsilon* — dokładność obliczeń,
- maxit — maksymalna dopuszczalna liczba iteracji

Output: (r, v, it, err), gdzie:

- r — przybliżenie pierwiastka równania $f(x) = 0$,
- v — wartość $f(\mathsf{r})$,
- it — liczba wykonanych iteracji,
- err — sygnalizacja błędu:
  - 0 — metoda zbieżna,
  - 1 — nie osiągnięto wymaganej dokładności w maxit iteracji.

```
1  Function MSIECZNYCH(f, x_0, x_1, delta, epsilon, maxit)
2      r ← 0
3      v ← 0
4      it ← 0
5      err ← 0
6      f_{x_0} ← f(x_0)
7      f_{x_1} ← f(x_1)
8      for it ← 1 to maxit do
9          if |f_{x_0}| > |f_{x_1}| then
10             swap(x_0, x_1)
11             swap(f_{x_0}, f_{x_1})
12         end
13         s ← (x_1 − x_0)/(f_{x_1} − f_{x_0})
14         x_1 ← x_0
15         f_{x_1} ← f_{x_0}
16         x_0 ← x_0 − (f_{x_0} · s)
17         f_{x_0} ← f(x_0)
18         if |f_{x_0}| < epsilon ∨ |x_1 − x_0| < delta then
19             r ← x_0
20             v ← f_{x_0}
21             return (r, v, it, err)
22         end
23     end
24     r ← x_0
25     v ← f_{x_0}
26     err ← 1
27     return (r, v, it, err)
```

Wszystkie opisane wcześniej algorytmy dla zadań 1-3 zostały umieszczone w specjalnym module nazwanym *ComputeLinearEquations*. Zostały również do niego utworzone testy, zawierające 2 funkcje mające na celu przetestowanie poprawności implementacji: $f(x) = x - 5$ oraz $f(x) = ln(x)$. Wynik zwrócony przez funkcję był porównywany z uzyskanym za pomocą analitycznych obliczen lub przy użyciu zewnętrznych narzędzi.

4 Zadanie 4. - Wyznaczanie pierwiastka równania $\sin x - (\frac{1}{2}x)^2 = 0$

4.1 Opis problemu

Kolejne zadanie polegało na wyznaczeniu pierwiastka równania

$$\sin x - (\frac{1}{2}x)^2 = 0 \tag{2}$$

używając funkcji napisanych w zadaniach 1–3 z dokładnością obliczeń $\delta = \frac{1}{2} \cdot 10^{-5}$, $\epsilon = \frac{1}{2} \cdot 10^{-5}$ oraz następującymi warunkami początkowymi:

(1) z przedziałem początkowym $[1.5; 2]$ dla *metody bisekcji*,

(2) z przybliżeniem początkowym $x_0 = 1.5$ dla *metody Newtona*,

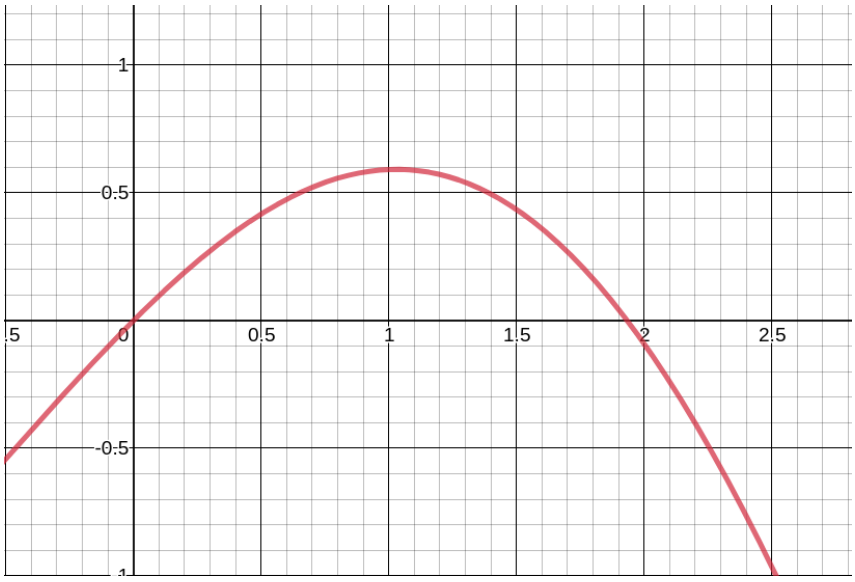
(3) z przybliżeniami początkowymi $x_0 = 1$, $x_1 = 2$ dla *metody siecznych*.

4.2 Rozwiązanie

Używając funkcji zaimplementowanych w zadaniach 1–3 wyznaczono pierwiastek funkcji $f(x) = \sin x - (\frac{1}{2}x)^2$ z zadanymi warunkami początkowymi. Dla metody stycznych przyjęto obliczoną analitycznie pochodną $f'(x) = \cos x - \frac{1}{2}x$, niezbędną do poprawnego działania metody Newtona. Dla metod stycznych i siecznych przyjęto maksymalną liczbę iteracji jako 50.

4.3 Otrzymane wyniki

Fragment przebiegu funkcji $f(x) = \sin x - (\frac{1}{2}x)^2$ przedstawia poniższy rysunek, wygenerowany przy użyciu programu Desmos.



Wyniki obliczeń uzyskanych przez kolejne algorytmy możemy obejrzeć w poniższej tabeli:

metoda	x	$f(x)$	iteracje	błąd
bisekcji	1.933 753 967 285 156	$-2.702\,768\,013\,840\,284 \times 10^{-7}$	16	0
stycznych	1.933 753 779 789 742	$-2.242\,331\,631\,485\,683 \times 10^{-8}$	4	0
siecznych	1.933 753 644 474 301	$1.564\,525\,129\,449\,379 \times 10^{-7}$	4	0

4.4 Obserwacje i wnioski

Podana funkcja posiada oczywiście dwa miejsca zerowe, ale wprowadzone dane służyły tylko do obliczenia jednego z nich - znajdującego się w punkcie 1.93375376282702125331. Różnice pomiędzy wynikami otrzymanymi dzięki powyższym metodom zaczynają się dopiero około szóstego miejsca po przecinku - najdokładniejsza wydaje się być metoda Newtona. Powyższy przykład dobrze ukazuje różnicę w liczbie wykonanych iteracji dla poszczególnych metod. Zauważmy, że metoda bisekcji potrzebowała najwięcej iteracji do znalezienia przybliżenia pierwiastka funkcji f - mianowicie 16 (algorytm ten dzieli podany przedział na dwie części, duża ilość iteracji mogła wynikać z umiejscowienia miejsca zerowego względem zastosowanego przedziału), podczas gdy pozostałe metody (stycznych i siecznych) odnalazły poszukiwaną wartość już po 4 iteracjach.

Uzyskane wyniki odzwierciedlają zbieżność każdej z metod. Metoda bisekcji posiada bowiem zbieżność liniową, metoda Newtona ma kwadratowy współczynnik zbieżności, a metoda siecznych zbiega nadliniowo (≈ 1.62). Metoda bisekcji, mimo że zbiega najwolniej spośród wszystkich trzech metod, to jest to metoda najstabilniejsza. Znalazła ona wartość nie tyle najbliższą zeru, co zadanej na wejściu dokładności. Ważną zaletą jest również to, że jest ona zbieżna globalnie i o ile końce przedziału początkowego faktycznie będą miały różne znaki, to zawsze zbiegnie do zera funkcji. Metody Newtona i siecznych są najszybciej zbieżne, w tym wypadku okazały się być bardziej skuteczne dla aproksymacji rozwiązania, osiągnęły większą dokładność i ostatecznie znalazły się bliżej rzeczywistego zera. Jednak warto pamiętać o tym, że na przykład dla innych funkcji albo inaczej dobranych przedziałów, może się okazać, iż takie zachowanie nie zostanie osiągnięte.

5 Zadanie 5. - Punkt przecięcia wykresów funkcji $y = 3x$ i $y = e^x$

5.1 Opis problemu

Zadanie piąte polegało na znalezieniu punktu przecięcia wykresów funkcji:

$$f_1(x) = 3x$$
$$f_2(x) = e^x$$

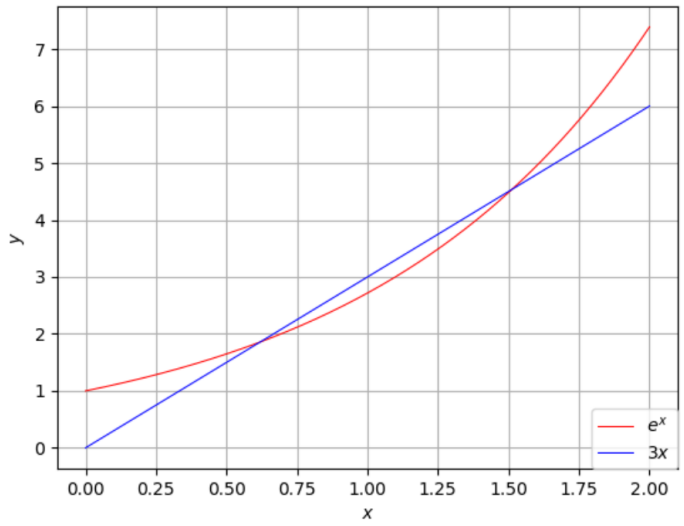
używając metody bisekcji z dokładnością $\delta = 10^{-4}$, $\epsilon = 10^{-4}$

5.2 Rozwiązanie

Punkt przecięcia wykresów funkcji f_1, f_2 to taki punkt, w którym funkcja:

$$f(x) = 3x - e^x \tag{3}$$

ma miejsce zerowe. W celu jego znalezienia została zaimplementowana funkcja f oraz skorzystano z metody bisekcji stworzonej w zadaniu pierwszym. Dokładności obliczeń przedstawiono już wcześniej. Przedziały początkowe wyznaczono, oszacowując wartości funkcji. W tym celu narysowano wykres funkcji f_1 oraz f_2 i zamieszczono go poniżej:



Z jego analizy wynika, iż wykresy zadanych funkcji przecinają się w dwóch miejscach, odpowiadających argumentom x około 0.6 oraz 1.5. Aby zobaczyć zmiany wyników w zależności od przyjętego przedziału początkowego, wybrano następujące przedziały: $[0; 2]$, $[0; 1]$, $[0.5; 0.65]$, $[1; 2]$, $[1.5, 1.8]$.

przedział	x	$f(x)$	iteracje	błąd
$[0.0, 1.0]$	0.619 140 625 000 000	$9.066\,320\,343\,276\,146 \times 10^{-5}$	9	0
$[0.5, 0.65]$	0.619 091 796 874 999	$3.486\,661\,493\,345\,977 \times 10^{-5}$	10	0
$[1.0, 2.0]$	1.512 084 960 937 500	$7.618\,578\,602\,741\,621 \times 10^{-5}$	13	0
$[1.5, 1.8]$	1.512 109 375 000 000	$3.868\,007\,140\,983\,565 \times 10^{-5}$	8	0
$[0.0, 2.0]$	0	0	0	1

5.3 Obserwacje i wnioski

Celem było znalezienie wartości x, a analiza wykresu funkcji okazała się bardzo pomocna, gdyż korzystając z metody bisekcji konieczna jest znajomość przebiegu zmienności funkcji i głównym problemem był wybór przedziałów początkowych. Dzięki analizie wykresu mogliśmy wywnioskować, jakie dane początkowe warto zastosować. Analizując podany wcześniej wykres widać, że uzyskane wartości odpowiadają w przybliżeniu wartościom miejsc zerowych zadanej funkcji. W przypadku braku możliwości skorzystania z graficznej reprezentacji funkcji lub dla szczególnie skomplikowanych funkcji określenie przedziałów, w których mają znajdować się miejsca zerowe, może okazać się bardzo trudne. Jak widzimy, wywołanie metody bisekcji dla przedziału $[0, 2]$ będzie skutkować zakończeniem wykonywania algorytmu z kodem błędu 1, ponieważ funkcja nie zmienia znaku na końcach zadanego przedziału. Wyniki obliczeń w pozostałych przedziałach zawierają się w granicach założonych błędów. Jednak dla konkretnego miejsca zerowego różnią się nieco w zależności od wyboru przedziału początkowego. Odmienna jest również liczba iteracji dla poszczególnych warunków początkowych. W przypadku pierwszego miejsca zerowego (przedziały $[0; 1]$ i $[0.5; 0.65]$) różnica między wynikami jest jednak pomijalnie mała, a dla przedziału $[0.5; 0.65]$ została wykonana zaledwie jedna iteracja więcej. Istotniejsze różnice można zaobserwować w przypadku przedziałów $[1; 2]$ i $[1.5, 1.8]$, dla których obliczenia zbiegają do drugiego miejsca zerowego. Względna różnica wyników jest tu rzędu 10^{-5}, natomiast dla przedziału $[1; 2]$ funkcja wykonała aż 5 iteracji więcej. Otrzymane wyniki prowadzą do wniosku, że skracanie przedziału początkowego nie musi koniecznie prowadzić do szybszej zbieżności metody bisekcji, która zależy raczej od położenia pierwiastka funkcji w stosunku do granicy przyjętego przedziału. Można zauważyć, że przy rozsądnie dobranych przybliżeniach początkowych znajomość

w miarę ogólnego przebiegu funkcji f pozwala dużo łatwiej zastosować metodę siecznych czy metodę Newtona, które w tym wypadku okazują się zbieżne i nie wymagają znajomości aż tak dokładnego przebiegu funkcji.

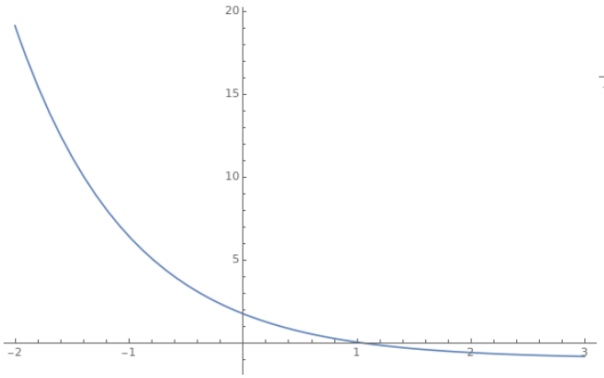
6 Miejsce zerowe funkcji $f_1(x) = e^{1-x} - 1$ i $f_2(x) = xe^{-x}$

6.1 Opis problemu

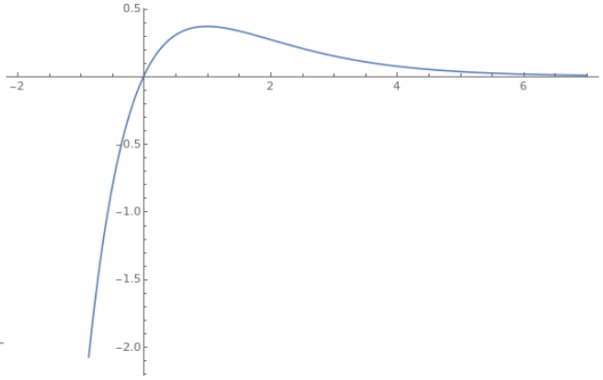
Ostatnie zadanie polegało na znalezieniu miejsc zerowych funkcji $f_1(x) = e^{1-x} - 1$ oraz $f_2(x) = xe^{-x}$ przy pomocy metod bisekcji, stycznych oraz siecznych z dokładnością obliczeń $\delta = 10^{-5}$ oraz $\epsilon = 10^{-5}$. Należało dobrać odpowiednio przedział i przybliżenia początkowe. Drugą częścią zadania było sprawdzenie wpływu zmiany przybliżenia początkowego na wynik obydwu funkcji - odpowiednio dla f_1 x_0 należy do przedziału $(1, \infty]$, a dla f_2 $x_0 = 1$ oraz $x_0 > 1$.

6.2 Rozwiązanie

Podane funkcje zostały zaimplementowane oraz wyliczono ich pochodne (potrzebne w metodzie Newtona). Aby znaleźć pierwiastki, wykorzystano metody stworzone w zadaniach 1 – 3. Przeprowadzona została analiza funkcji f_1 i f_2, które zostały przedstawione na wykresach poniżej. Na tej podstawie zostały wybrane odpowiednie parametry mające na celu ukazanie właściwości poszczególnych metod oraz przykładów ich złego stosowania. Analiza poniższych wykresów funkcji ogromnie ułatwia odpowiedni dobór przedziałów i przybliżeń początkowych.



(a) $f_1(x) = \exp(1 - x) - 1$



(b) $f_2(x) = x \exp(-x)$

Analiza podanych wzorów lub wykresów funkcji f_1 i f_2 natychmiast prowadzi do wniosku, że miejscem zerowym funkcji f_1 jest 1, a dla f_2 jest to 0. Będziemy jednak chcieli przeprowadzić eksperymenty dla różnych wartości początkowych oraz dla różnych przedziałów, aby sprawdzić, jak z tym zadaniem poradzą sobie poszczególne metody. Kolejne tabele zawierają wyniki działania metody bisekcji dla różnych przedziałów początkowych, wyniki uzyskane za pomocą metody Newtona dla różnych przybliżeń początkowych x_0 i wreszcie wyniki metody siecznych dla różnych przybliżeń x_0 i x_1. Nasze badania rozpoczniemy od sprawdzenia, jakie wyniki otrzymamy dobierając różne przedziały początkowe dla metody bisekcji dla obu funkcji. Obrane krańce przedziałów oraz rezultaty przeprowadzonych obliczeń prezentują się następująco dla dla funkcji f_1:

Zadany przedział	**Miejsce zerowe** r	**Wartość** $f_1(r)$	**Liczba iteracji**
$[0.0, 2.0]$	1.0	0.0	1
$[-2.0, 2.0]$	1.0	0.0	2
$[0.1, 1.8]$	0.9999992370605468	$7.6293974426988 \cdot 10^{-7}$	17
$[0.1, 1.2]$	0.9999938964843748	$6.10353425178 \cdot 10^{-6}$	14
$[-0.2, 1.8]$	0.9999969482421875	$3.051762469175045 \cdot 10^{-6}$	17
$[-5.0, 500.0]$	0.9999921917915344	$7.80823894963589 \cdot 10^{-6}$	24

Natomiast dla funkcji f_2:

Zadany przedział	**Miejsce zerowe** r	**Wartość** $f_1(r)$	**Liczba iteracji**
$[-0.5, 0.5]$	0.0	0.0	1
$[-0.7, 0.4]$	$4.5776367188509 \cdot 10^{-6}$	$4.577615764140961 \cdot 10^{-6}$	16
$[-10.0, 100.0]$	45.0	$1.288133361247227 \cdot 10^{-18}$	1

Zauważmy, że nieumiejętne

Wyniki dla metody Newtona dla funkcji f_1:

Przybliż.pocz. x_0	**Msc. zerowe** r	**Wartość** $f(r)$	**Iteracje**	**Błąd**
-1.0	0.9999922654776594	$7.73455225200336 \cdot 10^{-6}$	5	0
0.0	0.9999984358892101	$1.564112013019425 \cdot 10^{-6}$	4	0
1.1	0.99999999991094	$8.90598705893808 \cdot 10^{-11}$	3	0
2.0	0.9999999810061002	$1.899390000836831 \cdot 10^{-8}$	5	0
6.0	0.9999999573590406	$4.26409603182520 \cdot 10^{-8}$	147	0
8.0	NaN	NaN	0	1
15.0	15.0	-0.9999991684712809	0	2

Wyniki dla tej samej metody, ale dla funkcji f_2:

Przybliż.pocz. x_0	**Msc. zerowe** r	**Wartość** $f(r)$	**Iteracje**	**Błąd**
-1.0	$-3.064249341646176 \cdot 10^{-7}$	$-3.064250280608723 \cdot 10^{-7}$	5	0
-0.4	$-1.844031330942592 \cdot 10^{-8}$	$-1.84403136494710 \cdot 10^{-8}$	4	0
0.0	0.0	0.0	1	0
1.0	1.0	0.36787944117144233	0	2
6.0	14.97432014974184	$4.69983382720811 \cdot 10^{-6}$	8	0
8.0	14.636807965014	$6.43815521984328 \cdot 10^{-6}$	6	0
15.0	15.0	$4.58853480752738 \cdot 10^{-6}$	0	2

A teraz jako ostatnia, prezentacja wyników dla funkcji f_1, używając zaimplementowanej metody siecznych:

x_0	x_1	**Miejsce zerowe** r	Wartość $f(r)$	**Liczba iteracji**	**Błąd**
-2.0	2.0	1.0000000080618678	$-8.06186783997020 \cdot 10^{-9}$	8	0
-0.3	1.8	1.0000003464708873	$-3.464708272504779 \cdot 10^{-7}$	6	0
0.1	1.3	0.9999999935820667	$6.4179332959213 \cdot 10^{-9}$	5	0

Wyniki otrzymane również dla metody siecznych, ale dla funkcji f_2:

x_0	x_1	**Miejsce zerowe** r	**Wartość** $f(r)$	**Liczba iteracji**	**Błąd**
-2.0	2.0	14.294924723787231	$8.8506454983386 \cdot 10^{-6}$	15	0
-0.3	1.8	14.661140570698615	$6.29383436678240 \cdot 10^{-6}$	14	0
0.1	1.3	$2.482998842799128 \cdot 10^{-7}$	$2.4829982262708 \cdot 10^{-7}$	5	0
-2.0	6.0	14.812321857602736	$5.46655212223931 \cdot 10^{-6}$	12	0
-10.0	10.0	9.999999958776927	0.0004539993144685704	1	0
10.0	100.0	100.0	$3.720075976020836 \cdot 10^{-42}$	1	0

6.3 Obserwacje i wnioski

Analiza tabel pokazuje, że metoda bisekcji jest zbieżna do pierwiastka funkcji f_1 nawet dla bardzo dużych przedziałów początkowych, potrzebuje wtedy jednak odpowiednio większej liczby iteracji. Dla metody bisekcji nie ma znaczenia, jak duży przedział wybierzemy, metoda ta jest globalnie zbieżna, zatem zawsze dostaniemy mniej lub bardziej dokładny, poprawny wynik. Metoda Newtona potrzebuje natomiast bardzo dużej liczby iteracji już dla wartości początkowych niewiele większych od 1. Bardzo szybko też jej pochodna staje się bliska zeru, co uniemożliwia zastosowanie tej metody. Nieco inaczej prezentują się wyniki dla funkcji f_2. Należy uważać na dobór przedziałów w metodzie bisekcji, gdyż funkcja ta w nieskończoności dąży do zera, więc gdy weźmiemy wystarczająco duży przedział, obliczenia zakończą się po jednej iteracji, gdyż wartość funkcji będzie dostatecznie bliska zera, jednak bardzo odległa od faktycznego pierwiastka funkcji. Wyniki umieszczone w tabelach prezentujących zastosowanie metody Newtona pozwalają dostrzec, iż dla f_1 dla wartości nieco bardziej oddalonych od zera liczba iteracji pozwalających znaleźć ten pierwiastek rośnie, aż w końcu metoda staje się rozbieżna. Dla przybliżenia początkowego $x_0 > 8$ zaczynamy otrzymywać błędy metody. Najpierw dana wartość funkcji okazuje się niewystarczająca (kod błędu 1 - metoda rozbieżna), a potem funkcja f_1 zaczyna maleć bardzo powoli, a co za tym idzie jej pochodna jest bliska zeru, co uniemożliwia korzystanie z metody Newtona (kod błędu 2). W przypadku funkcji f_2 pochodna zeruje się dla $x_0 = 1$ i otrzymujemy kod błędu 2. Natomiast dla każdej wartości $x_0 > 1$, metoda Newtona nie zbiega do faktycznego zera, lecz znajduje wystarczająco małą wartość daleką od miejsca zerowego z powodu granicy f_2 równej zero w nieskończoności.
Ostatnia para tabel dla funkcji f_1 i f_2 zawiera wyniki obliczeń wykonanych metodą siecznych. Dla funkcji f_1 dla wartości przybliżeń początkowych nie tak bliskich zeru możemy obserwować rozbieżność podobną jak w metodzie Newtona. Dla funkcji f_2 analogicznie do metody Newtona, metoda siecznych dla odpowiednio odległych od pierwiastka przybliżeń początkowych nie znajduje faktycznego zera, ale punkty od niego oddalone ze względu na granicę funkcji.
W przypadku metody Newtona zawuważalny jest znaczny wzrost szybkości wyznaczania przybliżenia pierwiastka r dla danej funkcji w porównaniu z metodą bisekcji. Dla funkcji f_1 przekazanie algorytmowi wartości początkowej $x_0 \in (1, \infty]$ powoduje drastyczny wzrost liczby potrzebnych iteracji do wyznaczenia przybliżenia miejsca zerowego. W przypadku funkcji f_2 przekazanie do algorytmu $x_0 = 1$ powoduje natychmiastowe zakończenie działania programu, gdyż $f_2'(x_0)$ jest bliska wartości 0. Wzrost wartości x_0 powoduje zmniejszanie się dokładności wyznaczonego przybliżenia. Dla wartości przybliżeń zbyt odległych od faktycznego pierwiastka, metoda stycznych nie jest w stanie osiągnąć poprawnego przybliżenia. W przypadku metody siecznych wybranie początkowych wartości x_0, x_1 zbyt odległych od faktycznej wartości pierwiastka funkcji powoduje, że metoda nie jest w stanie osiągnąć przybliżenia o wymaganej dokładności i zwraca wartość bardzo odległą od rzeczywistej. Im bardziej ścisły przedział wokół rzeczywisstego miejsca zerowego, tym mniej potrzeba iteracji do obliczenia przybliżenia z zadaną dokładnością.

Fakt, iż metoda Newtona jest szybsza od metody bisekcji wynika wynika z jej lepszej zbieżności - metoda bisekcji ma zbieżność kwadratową. Podobnie, jak metoda Newtona, metoda siecznych jest o wiele szybsza od metody bisekcji, ponieważ posiada ona złożoność nadliniową. Wszystkie

przedstawione metody różnią się od siebie w kwestii wykonywanych przez nie operacji, a metoda Newtona zawiera operację obliczania pochodnej. Może ona być wymagająca obliczeniowo, co oznacza, że mimo, iż ilość iteracji będzie mniejsza bądź równa, to nakłady czasowe potrzebne na wykonanie algorytmu będą różne. Dzięki powyższym wnioskom można dojść do tego, iż nie ma jednej metody która będzie idealna w każdych warunkach i warto mądrze wybrać narzędzie do wykonywanego zadania z uwzględnieniem tego, na czym nam zależy - czy będzie to szybkość wyniku, czy jego dokładność.