Obliczenia Naukowe,
Laboratorium, Lista 3

Jerzy Wroczyński

2020-11-22

1 Zadanie 1.

1.1 Opis problemu

Napisać funkcję rozwiązującą równanie $f(x) = 0$ metodą bisekcji.

1.1.1 Dane wejściowe

- `f` — funkcja f w postaci anonimowej funkcji
- `a,b` — liczby typu `Float64` określające końce przedziału początkowego
- `delta, epsilon` — liczby typu `Float64` określające dokładności obliczeń

1.1.2 Dane wyjściowe

Czwórka wartości `(r,v,it,err)`.

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

1.2 Rozwiązanie

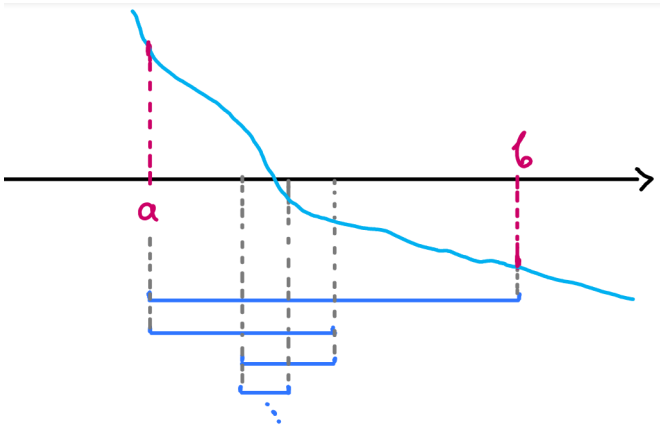
Kod źródłowy znajduje się w pliku `code/iterative-methods.jl` jako funkcja `mbisekcji`.

1.3 Opis metody

Mając pewien przedział $[a; b]$ gdzie $f(a) \cdot f(b) = -1$ (mamy inne znaki wartości funkcji w granicach przedziału) mamy pewność, że funkcja ma w tym przedziale miejsce zerowe. Idea jest taka: dzielimy przedział na dwie części (wyznaczając środek przedziału) i sprawdzamy gdzie mamy spełniony warunek $f(a') \cdot f(b') = -1$ w nowym przedziale $[a'; b']$ będący połówką poprzedniego przedziału $[a; b]$.

Powtarzamy czynność aż nie osiągniemy oczekiwanej precyzji (określonej przez odpowiednie argumenty ze specyfikacji), czyli $|b-a| < \delta$ lub $|f(c)| < \epsilon$ gdzie c to środek przedziału.

Oczywiście, żeby metoda działała poprawnie zadana funkcja wejściowa musi być ciągła w tym przedziale.



Rysunek 1: Przykład zastosowania metody bisekcji, kilka iteracji zawężania przedziału

1.4 Opis algorytmu

Algorytm bisekcji opisuje poniższa lista kroków (przyjmujemy wcześniej opisaną specyfikację danych wejściowych i wyjściowych):

1: $u \leftarrow f(a)$
2: $v \leftarrow f(b)$
3: $e \leftarrow (b-a)$
4: **if** $\text{sgn}(u) = \text{sgn}(v)$ **then**
5: **return** $(a, u, 0, 1)$
6: **end if**
7: **for** $k \leftarrow 0 \ldots$ `MAX_ITERATIONS` **do**
8: $e \leftarrow \frac{e}{2}$
9: $c \leftarrow a + e$
10: $w \leftarrow f(c)$
11: **if** $|e| < \delta$ or $|w| < \epsilon$ **then**
12: **return** $(c, w, k, 0)$
13: **end if**
14: **if** $\text{sgn}(w) \neq \text{sgn}(u)$ **then**
15: $b \leftarrow c$
16: $v \leftarrow w$
17: **else**
18: $a \leftarrow c$
19: $u \leftarrow w$
20: **end if**
21: **end for**

W linijkach 1–3 przygotowujemy program do działania (obliczamy wartości w punktach granicznych przedziału początkowego). W linijce 4 mamy pierwsze sprawdzenie poprawności danych. Jeśli w podanym przedziale $[a; b]$ podana funkcja

nie zmienia znaku — odrzucamy, zwracamy wynik z zaznaczeniem błędu.

W linijce 7 zaczynamy iterować. Najpierw dzielimy przedział na dwie części (8), ale jeszcze przed wyborem, którą część bierzemy, sprawdzamy jeszcze czy nie spełniliśmy warunku wyjściowego (11).

Na koniec wybieramy oczywiście tę połowę przedziału (w celu zdefiniowania nowego przedziału do kolejnych iteracji), która zawiera miejsce zerowe (mamy różne znaki wartości funkcji f w granicach nowego przedziału).

2 Zadanie 2.

2.1 Opis problemu

Napisać funkcję rozwiązującą równanie $f(x) = 0$ metodą Newtona.

2.1.1 Dane wejściowe

- `f` — funkcja f w postaci anonimowej funkcji
- `pf` — pochodna funkcji f w postaci anonimowej funkcji
- `x0` — przybliżenie początkowe
- `delta, epsilon` — liczby typu `Float64` określające dokładności obliczeń
- `maxit` — liczba całkowita określająca dopuszczalną liczbę iteracji

2.1.2 Dane wyjściowe

Czwórka wartości `(r,v,it,err)`.

- `r` — przybliżenie pierwiastka równania $f(x) = 0$
- `v` — wartość funkcji w r
- `it` — liczba wykonanych iteracji
- `err` — sygnalizacja błędu, możliwe wartości:
  - `0` — metoda zbieżna
  - `1` — nie osiągnięto wymaganej dokładności w `maxit` iteracji
  - `2` — pochodna bliska zeru

2.2 Rozwiązanie

Kod źródłowy znajduje się w pliku `code/iterative-methods.jl` jako funkcja `mstycznych`.

2.3 Opis metody

Idea metody Newtona (metody stycznych) opiera się na linearyzacji funkcji czyli zastąpieniu funkcji wejściowej f funkcją liniową. Chcemy uzyskać funkcję liniową, która w jakiś sposób przybliża prawdziwą funkcję. Wykorzystujemy tutaj rozwinięcie w szereg Taylora, przy czym bierzemy tylko dwa pierwsze wyrazy. Ta metoda jest nazywana metodą stycznej, bo właśnie tym jest ta nowa funkcja (i właśnie dlatego też potrzebujemy pochodnej funkcji wejściowej, bo pochodna właśnie określa styczną do funkcji początkowej w danym punkcie).

Możemy sobie pozwolić na wzięcie tylko dwóch pierwszych wyrazów szeregu Taylora pod warunkiem, że to odchylenie między punktem początkowym a docelowym nie jest duże.

Metoda Newtona zaczyna od jakiegoś początkowego punktu x_n, który jest w pewnej odległości od szukanego punktu. Czyli mamy od tego x_n pewne odchylenie. Wówczas mamy

$$f(\tilde{r}) \approx f(x_n) + f'(x_n) \cdot (\tilde{r} - x_n)$$

korzystając z dwóch pierwszych wyrazów szeregu Taylora. Oczywiście $f(\tilde{r})$ jest równe zero (szukamy miejsc zerowych), wówczas

$$0 \approx f(x_n) + f'(x_n) \cdot (\tilde{r} - x_n)$$

co po pewnych przekształceniach daje nam

$$\tilde{r} \approx x_n - \frac{f(x_n)}{f'(x_n)}.$$

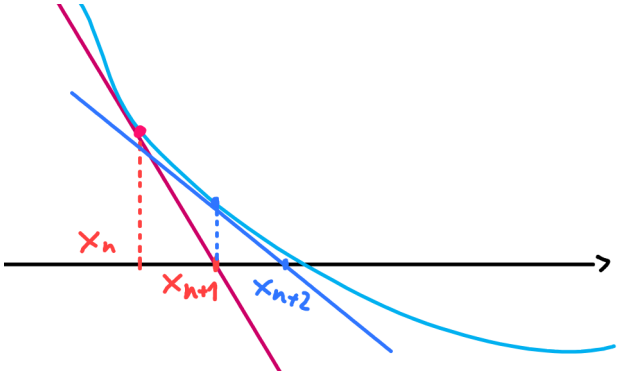
Oczywiście, mając obliczony przybliżony kolejny punkt możemy zrobić znowu to samo, żeby się zbliżyć do rozwiązania. Dlatego definiujemy wzór rekurencyjny

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

który tworzy nasz ciąg coraz to lepszych wyników. Oczywiście musimy też zadbać o to, żeby „wystartować" w miejscu niedalekim od szukanego zera funkcji, żeby metoda ta miała szansę znaleźć to miejsce zerowe.

Tak samo jak w przypadku metody bisekcji — zatrzymujemy się wtedy kiedy osiągnęliśmy zadaną precyzję: $|x_{n+1} - x_n| < \delta \vee |f(x_{n+1})| < \epsilon$.

Minusem tej metody, że potrzebujemy mieć zdefiniowaną pochodną zadanej funkcji wejściowej.



Rysunek 2: Przykład zastosowania metody Newtona

2.4 Opis Algorytmu

Algorytm wykorzystujący metodę Newtona opisuje poniższa lista kroków (przyjmujemy wcześniej opisaną specyfikację danych wejściowych i wyjściowych):

1: $v \leftarrow f(x_0)$
2: **if** $|v| < \epsilon$ **then**
3: **return** $(x_0, v, 0, 0)$
4: **end if**
5: $x_n \leftarrow x_0$
6: **for** $k \leftarrow 0 \ldots$ `maxit` **do**
7: $x_{n+1} \leftarrow x_n - \frac{v}{f'(x_n)}$
8: $v \leftarrow f(x_{n+1})$
9: **if** $f'(x_n) <$ `NEAR_ZERO` **then**
10: **return** $(x_n, f(x_n), k, 2)$
11: **end if**
12: **if** $|x_{n+1} - x_n| < \delta \vee |v| < \epsilon$ **then**
13: **return** $(x_{n+1}, v, k, 0)$
14: **end if**
15: $x_n \leftarrow x_{n+1}$
16: **end for**
17: **return** $(x_{n+1}, v, k, 1)$

Algorytm najpierw sprawdza czy wynik początkowy nie jest już wystarczający, biorąc pod uwagę podaną precyzję (2). Następnie przechodzimy do iterowania kolejnych wyrazów naszego ciągu, który zbiega do faktycznego rozwiązania. W środku pętli 6 oprócz sprawdzania czy nie dotarliśmy do wyniku o żądanej precyzji 12, ale też patrzymy czy wartość pochodnej nie jest bliska zeru — jeżeli jest to przerywamy program i zwracamy informację z błędem o określonym kodzie ze specyfikacji.

W linijce 12 sprawdzamy, czy żądana precyzja została osiągnięta — jeśli tak to zwracamy aktualnie wyliczoną wartość. W przeciwnym przypadku w linijce 15 „przesuwamy" wartość wyrazu ciągu i kontynuujemy iterowanie.

Jeśli przekroczymy liczbę iteracji, zwracamy ostanią wyliczoną wartość wraz z błędem `1` zgodnie ze specyfikacją.

3 Zadanie 3.

3.1 Opis problemu

Napisać funkcję rozwiązującą równanie $f(x) = 0$ metodą siecznych.

3.1.1 Dane wejściowe

- `f` — funkcja f w postaci anonimowej funkcji
- `x0,x1` — przybliżenia początkowe
- `delta, epsilon` — liczby typu `Float64` określające dokładności obliczeń
- `maxit` — liczba całkowita określająca dopuszczalną liczbę iteracji

3.1.2 Dane wyjściowe

Czwórka wartości `(r,v,it,err)`.

- `r` — przybliżenie pierwiastka równania $f(x) = 0$

- `v` — wartość funkcji w r
- `it` — liczba wykonanych iteracji
- `err` — sygnalizacja błędu, możliwe wartości:
  - `0` — metoda zbieżna
  - `1` — nie osiągnięto wymaganej dokładności w `maxit` iteracji

3.2 Rozwiązanie

Kod źródłowy znajduje się w pliku `code/iterative-methods.jl` jako funkcja `msiecznych`.

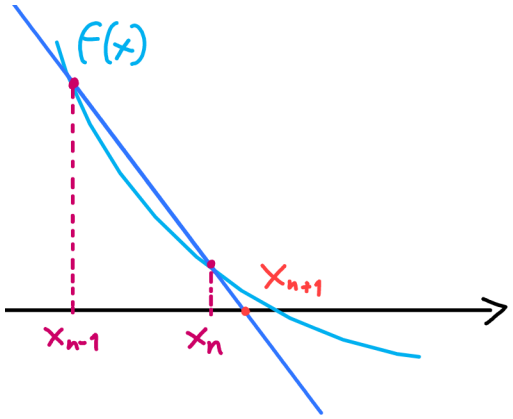
3.3 Opis metody

Metoda siecznych jest nieco zmienioną wersją metody Newtona. Po pierwsze mamy zmianę w danych wejściowych do algorytmu, a po drugie mamy inną interpretację geometryczną tego podejścia.

Zamiast podawania dodatkowo pochodnej funkcji, której chcemy znaleźć pierwiastek, podajemy dwa przybliżone punkty i tak naprawdę to sami obliczamy taką pseudo-pochodną ze wzoru z definicji na pochodną:

$$f'(x) = \frac{f(x) - f(x+h)}{h}.$$

Oczywiście, oryginalnie mówimy o pochodnej kiedy h dąży do zera. Tutaj, punkty x, $(x+h)$ są jednak od siebie relatywnie daleko — dlatego też nie mówimy już o stycznej, a raczej właśnie o siecznej.



Rysunek 3: Przykład zastosowania metody siecznych

Zdefiniujmy naszą pochodną w tych nowych warunkach:

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

i skorzystajmy ze wzoru 1, który w zadaniu 2. definiował nam kolejne wyrazy — jednakże tym razem wzór na kolejny wyraz ciągu jest uzależniony od dwóch poprzednich:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{2}$$

Tak samo jak w poprzednich metodach — zatrzymujemy się wtedy kiedy osiągnęliśmy zadaną precyzję: $|x_{n+1} - x_n| < \delta \vee |f(x_{n+1})| < \epsilon$.

3.4 Opis algorytmu

Algorytm wykorzystujący metodę siecznych opisuje poniższa lista kroków (przyjmujemy wcześniej opisaną specyfikację danych wejściowych i wyjściowych):

1: $v \leftarrow f(x_0)$
2: $w \leftarrow f(x_1)$
3: $a \leftarrow x_0$
4: $b \leftarrow x_1$
5: **for** $k \leftarrow 0 \ldots$ `maxit` **do**
6: **if** $|v| > |w|$ **then**
7: $a, b \leftarrow b, a$
8: $v, w \leftarrow w, v$
9: **end if**
10: $d \leftarrow \frac{b-a}{w-v}$
11: $b \leftarrow a$
12: $w \leftarrow v$
13: $a \leftarrow a - d \cdot v$
14: $v \leftarrow f(a)$
15: **if** $|v| < \epsilon \vee |b-a| < \delta$ **then**
16: **return** $(a, v, k, 0)$
17: **end if**
18: **end for**
19: **return** $(a, v, k, 1)$

W linijkach 1–4 przygotowujemy dane do działania algorytmu — obliczamy wartości w danych punktach początkowych oraz wkładamy pierwsze dwie aproksymacje do odpowiednich zmiennych.

W linijce 5 zaczynamy iterować. W środku pętli upewniamy się, że mamy zachowany porządek obliczonych wartości (6). Dzięki temu wartości kolejnych wyrazów ciągu nie będą rosnąć — a tego chcemy, bo szukamy miejsca zerowego.

W linijkach 10–14 obliczamy już następny wyraz ciągu.

Standardowo, w linijce 15 zwracamy miejsce zerowe jeśli obliczony wynik jest w ryzach określonych przez podaną precyzję.

4 Zadanie 4.

4.1 Opis problemu

Należy obliczyć pierwiastek równania $\sin x - \left(\frac{1}{2}x\right)^2 = 0$ przy pomocy wcześniej zaprogramowanych metod:

- metoda bisekcji z przedziałem początkowym $[1.5; 2]$
- metoda Newtona z przybliżeniem początkowym $x_0 = 1.5$
- metoda siecznych z przybliżeniami początkowymi $x_0 = 1$, $x_1 = 2$

i precyzją $\delta = \frac{1}{2}10^{-5}$, $\epsilon = \frac{1}{2}10^{-5}$.

4.2 Rozwiązanie

4.2.1 Metoda bisekcji

Wykorzystujemy odpowiednią funkcję z wcześniej zaprogramowanego modułu do naszego testu (`code/ex-4.1.jl`). Wyniki wyświetlone po uruchomieniu programu `./code/ex-4.1.jl`:

- pierwiastek, $r = 1.9337539672851562$
- wartość funkcji, $f(r) = -2.7027680138402843 \cdot 10^{-7} < \epsilon$
- liczba iteracji. $k = 15$
- brak błędu, $\text{err} = 0$

4.2.2 Metoda Newtona

Wykorzystujemy odpowiednią funkcję z wcześniej zaprogramowanego modułu do naszego testu (`code/ex-4.2.jl`).

Przy tej metodzie musimy jeszcze obliczyć pochodną naszej funkcji:

$$\frac{d}{dx} \sin x - \left(\frac{1}{2}x\right)^2 = \cos(x) - \frac{x}{2}.$$

Wyniki wyświetlone po uruchomieniu programu `./code/ex-4.2.jl`:

- pierwiastek, $r = 1.933753779789742$
- wartość funkcji, $f(r) = -2.2423316314856834 \cdot 10^{-8} < \epsilon$
- liczba iteracji, $k = 4$
- brak błędu, $\text{err} = 0$

4.2.3 Metoda siecznych

Wykorzystujemy odpowiednią funkcję z wcześniej zaprogramowanego modułu do naszego testu (`code/ex-4.3.jl`). Wyniki wyświetlone po uruchomieniu programu `./code/ex-4.3.jl`:

- pierwiastek, $r = 1.933753644474301$
- wartość funkcji, $f(r) = 1.564525129449379 \cdot 10^{-7} > \epsilon$
- liczba iteracji, $k = 3$
- brak błędu, $\text{err} = 0$

4.3 Interpretacja wyników i wnioski

W tym zadaniu wszystkie metody dały rezultaty zadowalające, bo wartość funkcji w obliczonym przybliżeniu pierwiastka była mniejsza niż zadana precyzja. Co warto zauważyć, metody dały bardzo bliskie sobie wyniki — mamy zgodność pierwszych 6 miejsc po przecinku.

Różnica jest zauważalna w liczbie iteracji, w której ten wynik został otrzymany w zależności od metody. Metoda bisekcji jest metodą globalną, która liniowo dąży do wyniku, dlatego też zajęło jej to najwięcej iteracji. W przypadku metod Newtona i siecznych są to metody lokalne, które znacznie szybciej doszły do wyniku. Obie te metody mają parametr zbieżności $\alpha > 1$ i dlatego są szybsze. Przy czym oczywiście wymagają więcej danych początkowych — metoda Newtona żąda dodatkowo pochodnej funkcji, metoda siecznych potrzebuje dwóch początkowych aproksymacji, kiedy metoda bisekcji potrzebuje tylko zakresu, w którym należy szukać zera funkcji.

5 Zadanie 5.

5.1 Opis problemu

Należy wyznaczyć punkt, w którym przecinają się funkcje $y = 3x$ oraz $y = e^x$. Dokładność obliczeń wynosi: $\delta = 10^{-4}$, $\epsilon = 10^{-4}$. Należy użyć metody bisekcji.

5.2 Rozwiązanie

Zadanie sprowadza się do wyznaczenia miejsca zerowego funkcji $f(x) = 3x - e^x$ przy pomocy metody bisekcji. Wykorzystujemy tutaj odpowiednią funkcję z wcześniej zaprogramowanego modułu.

Najpierw jednak musimy się zastanowić w jakim przedziale początkowym należy szukać tego punktu. W przedziale $(-\infty; 0)$ na pewno nie znajdziemy przecięcia funkcji, jako, że funkcja $3x$ jest tam zawsze ujemna, kiedy funkcja wykładnicza nigdy nie przyjmuje wartości ujemnych. W przedziale $[0; \infty)$ będziemy mieli na pewno przynajmniej jedno przecięcie. Przyjrzyjmy się kilku wartościom obu funkcji:

- $x = 0$:
  - $3x = 0$
  - $e^x = 1$
  - $3x < e^x$
- $x = 1$:
  - $3x = 3$
  - $e^x = e \approx 2.71828$
  - $3x > e^x$
- $x = 2$:
  - $3x = 6$
  - $e^x = e^2 \approx 7.38906$
  - $3x < e^x$

Z powyższych danych można wywnioskować, że mamy tak naprawdę dwa miejsca, w których te dwie funkcje się przecinają. Mamy dwa momenty przejścia: raz z sytuacji $3x < e^x$ do sytuacji $3x > e^x$ i raz z sytuacji $3x > e^x$ do sytuacji $3x < e^x$. Później nie mam więcej takich przejść, ponieważ funkcja liniowa $3x$ nie jest w stanie „dogonić" funkcji wykładniczej. Daje name to dwa miejsca zerowe funkcji f. Czyli żeby obliczyć oba *zera* tej funkcji ustalamy przedział najpierw na $[0; 1]$ a potem na $[1; 2]$.

Uruchamiając program `./code/ex-5.jl` otrzymujemy wyniki:

- Pierwszy pierwiastek:
  - pierwiastek, $r = 6.19140625 \cdot 10^{-1}$
  - wartość funkcji, $f(r) = 9.066320343276146 \cdot 10^{-5}$
  - liczba iteracji, $k = 8$
  - brak błędu, $\text{err} = 0$
- Drugi pierwiastek:

– pierwiastek, 1.5120849609375

– wartość funkcji, $f(r) = 7.618578602741621 \cdot 10^{-5}$

– liczba iteracji, $k = 12$

– brak błędu, err $= 0$

5.3 Interpretacja wyników i wnioski

Otrzymane wyniki są poprawne — wartości funkcji w wyliczonych miejscach zerowych są bardzo bliskie zeru (mieszczą się w podanej precyzji). Jednakże, co warto zauważyć, najpierw przed użyciem funkcji odpowiadającej metodzie bisekcji musieliśmy przeanalizować zachowanie tej funkcji. Bez takiej analizy nie wiedzielibyśmy o liczbie zer tej funkcji i nie wiedzielibyśmy w jakim przedziale (przedziałach) zacząć.

6 Zadanie 6.

6.1 Opis problemu

Należy znaleźć miejsca zerowe funkcji $f_1(x) = e^{1-x} - 1$ oraz $f_2(x) = xe^{-x}$ za pomocą metod bisekcji, Newtona i siecznych. Wymagane dokładności obliczeń: $\delta = 10^{-5}$, $\epsilon = 10^{-5}$. Należy dobrać odpowiednio przedział i przybliżenia początkowe.

Dodatkowo, sprawdzić co się stanie, gdy w metodzie Newtona dla f_1 wybierzemy $x_0 \in (1; \infty]$, a dla f_2 wybierzemy $x_0 > 1$, czy możemy wybrać $x_0 = 1$ dla f_2?

6.2 Rozwiązanie dla f_1

Funkcja $f_1(x) = e^{1-x} - 1$ jest zwykłą funkcją eksponenty odwrócona w osi x, przesunięta o 1 w prawo i o 1 w dół. Co warto zauważyć, że miejsce zerowe będzie wynosić po prostu $r = 1$, z bardzo prostego faktu. Po przesunięciu w dół o 1 funkcji e^x miejsce zerowe naturalnie jest w punkcie $x = 0$, bo wartość e^x w tym momencie wynosi 1, więc $e^0 - 1 = 0$ — mamy miejsce zerowe. Tutaj tylko przesuwamy jeszcze funkcję w prawo o 1 dodatkowo. Czyli nasze miejsce zerowe również zostanie przesunięte o 1 w prawo.

6.2.1 Metoda bisekcji

Mając wiedzę z sekcji 6.2 możemy określić przedział początkowy na np. $[0.5; 1.5]$.

Uruchamiając program `./code/ex-6.1.1.jl` bez zdziwienia zauważamy, że zero funkcji f_1 zostało poprawnie wyliczone od razu, bo było one zlokalizowane na samym środku przedziału.
Jednakże, jeżeli uruchomimy program `./code/ex-6.1.1.jl a` wówczas dostajemy inny wynik:

- pierwiastek, $r = 1.0000021362304687$
- wartość funkcji, $f(r) = -2.136228187010758 \cdot 10^{-6} < \epsilon$
- liczba iteracji, $k = 14$
- brak błędu, err $= 0$

Program uruchomiony z dodatkowym parametrem zmienia przedział na nieco inny ($[0.23; 1.56]$). Mamy nieco inny wynik, ponieważ tym razem miejsce zerowe funkcji f_1 nie jest idealnie na środku zadanego przedziału. Dostajemy zadowalający wynik z perspektywy zadanej precyzji, jednak nie jest to *idealny* wynik.

6.2.2 Metoda Newtona

Do zadania liczymy pochodną

$$\frac{d}{dx}e^{1-x} - 1 = -e^{1-x}.$$

Po uruchomieniu programu `./code/ex-6.2.1.jl` mamy, podobnie jak to miało miejsce w metodzie bisekcji, odpowiedź od razu (tym razem sami ją podaliśmy), bo użyliśmy $x_0 = 1$.

Jeśli uruchomimy program z argumentem `./code/ex-6.2.1.jl a` (zmieniający $x_0 = 0.23$) mamy, znowu podobnie jak w bisekcji, nieco inny wynik od tego idealnego, ale nadal akceptowalny biorąc pod uwagę zadaną precyzję:

- pierwiastek, $r = 0.9999999507753422$
- wartość funkcji, $f(r) = 4.922465901557871 \cdot 10^{-8} < \epsilon$
- liczba iteracji, $k = 4$
- brak błędu, $\text{err} = 0$

6.2.3 Metoda siecznych

Tak jak w poprzednich metodach — jeśli zaczniemy od wartości, które od razu prowadzą do rozwiązania (tutaj $x_0 = 0$, $x_1 = 1$) otrzymujemy też rozwiązanie idealne $r = 1$.

W przypadku zmienionych aproksymacji początkowych (tutaj np. $x_0 = 0$, $x_1 = 0.8$) można znowu zauważyć, że mamy przynajmniej podobne wyniki w porównaniu z rozwiązaniem idealnym. Po uruchomieniu programu `./code/ex-6.3.1.jl a` otrzymujemy wyniki:

- pierwiastek, $r = 0.9999987795286631$
- wartość funkcji, $f(r) = 1.2204720816200165 \cdot 10^{-6} < \epsilon$
- liczba iteracji, $k = 3$
- brak błędu, $\text{err} = 0$

6.3 Rozwiązanie dla f_2

Tak samo jak dla funkcji f_1 (sekcja 6.2) dokonujemy analizy budowy funkcji f_2.
Funkcja $f_2 = xe^{-x}$ jest iloczynem dwóch elementarnych funkcji — funkcji liniowej $y = x$ oraz odwróconej względem osi OY funkcji eksponenty e^{-x}. Zbiór miejsc zerowych funkcji f_2 będzie równy sumie zbiorów miejsc zerowych obu wspomnianych funkcji elementarnych. Funkcja wykładnicza $y = e^{-x}$ nie ma żadnych miejsc zerowych, za to funkcja liniowa $y = x$ ma jedno miejsce zerowe $r = 0$.

Sytuacja jest bardzo podobna jak w przypadku funkcji f_1, jedynie mamy inne miejsce zerowe. Z tego powodu, poniższe pod-rozwiązania brzmią w taki sam sposób jak dla funkcji f_1.

6.3.1 Metoda bisekcji

Mając wiedzę z sekcji 6.3 możemy określić przedział początkowy na np. $[-0.5; 0.5]$.

Uruchamiając program `./code/ex-6.1.2.jl` bez zdziwienia zauważamy, że zero funkcji f_2 zostało poprawnie wyliczone od razu, bo było one zlokalizowane na samym środku przedziału.
Jednakże, jeżeli uruchomimy program `./code/ex-6.1.2.jl a` wówczas dostajemy inny wynik:

- pierwiastek, $r = 1.983642578152971 \cdot 10^{-6}$
- wartość funkcji, $f(r) = 1.9836386433189956 \cdot 10^{-6} < \epsilon$

- liczba iteracji, $k = 15$
- brak błędu, $\text{err} = 0$

Program uruchomiony z dodatkowym parametrem zmienia przedział na nieco inny ($[-0.29; 1]$). Mamy nieco inny wynik, ponieważ tym razem miejsce zerowe funkcji f_1 nie jest idealnie na środku zadanego przedziału. Dostajemy zadowalający wynik z perspektywy zadanej precyzji, jednak nie jest to *idealny* wynik.

6.3.2 Metoda Newtona

Do zadania liczymy pochodną

$$\frac{d}{dx} xe^{-x} - 1 = -e^{-x} \cdot (x - 1).$$

Po uruchomieniu programu `./code/ex-6.2.2.jl` mamy, podobnie jak to miało miejsce w metodzie bisekcji, odpowiedź od razu (tym razem sami ją podaliśmy), bo użyliśmy $x_0 = 0$.

Jeśli uruchomimy program z argumentem `./code/ex-6.2.2.jl a` (zmieniający $x_0 = -0.29$) mamy, znowu podobnie jak w bisekcji, nieco inny wynik od tego idealnego, ale nadal akceptowalny biorąc pod uwagę zadaną precyzję:

- pierwiastek, $r = -2.5146064072824716 \cdot 10^{-10}$
- wartość funkcji, $f(r) = -2.514606407914796 \cdot 10^{-10} < \epsilon$
- liczba iteracji, $k = 4$
- brak błędu, $\text{err} = 0$

6.3.3 Metoda siecznych

Tak jak w poprzednich metodach — jeśli zaczniemy od wartości, które od razu prowadzą do rozwiązania (tutaj $x_0 = -1$, $x_1 = 0$) otrzymujemy też rozwiązanie idealne $r = 0$.

W przypadku zmienionych aproksymacji początkowych (tutaj np. $x_0 = 0.2$, $x_1 = -0.6$) można znowu zauważyć, że mamy przynajmniej podobne wyniki w porównaniu z rozwiązaniem idealnym.
Po uruchomieniu programu `./code/ex-6.3.2.jl a` otrzymujemy wyniki:

- pierwiastek, $r = -1.1073076595221682 \cdot 10^{-7}$
- wartość funkcji, $f(r) = -1.1073077821352003 \cdot 10^{-7} > \epsilon$
- liczba iteracji, $k = 4$
- brak błędu, $\text{err} = 0$

6.4 Wnioski z pierwszej części rozwiązania

Czasami (prosta) analiza funkcji może osiągnąć znacznie więcej niż używanie podejścia *brute-force* i wykorzystywanie danej metody iteracyjnej dla funkcji, dla której chcemy znaleźć miejsce zerowe.

Drugim wnioskiem na pewno znowu może być to samo co w zadaniu 4. — metody Newtona i siecznych są szybsze niż metoda bisekcji.

6.5 Rozwiązanie dla zadanych *edge cases*

6.5.1 $x_0 \in (1; \infty)$ dla funkcji f_1

Po uruchomieniu programu `./code/ex-6.4.jl` (dla przykładu $x_0 = 10$) otrzymujemy wynik z błędem — pochodna bliska zeru (kod `2`). Nie jest to zaskakujące biorąc pod uwagę, że pochodna właśnie dla większych x będzie bardzo bliska zeru. Wynika to z tego, że pochodna to odwrócona względem osi OY funkcja eksponenty, czyli dla dużych x wartości tej funkcji to będą ułamki postaci $\frac{1}{e^x}$.

6.5.2 $x_0 > 1$ dla funkcji f_2

Po uruchomieniu programu `./code/ex-6.5.jl` (dla przykładu $x_0 = 1000$) otrzymujemy wynik, według programu, poprawny jednakże tak nie jest. Funkcja podała jako wynik $r = 1000$, a wartość funkcji obliczyła $f(r) = 0$. Jednakże ten wynik jest błędny — wynika on z ograniczonej precyzji arytmetyki, którą przyjęliśmy. Należy zauważyć, że dla dużych dodatnich x funkcja e^{-x}, będąca jednym ze składników funkcji f_2, przyjmuje bardzo małe wartości. Dodatkowo e^{-x} rośnie wykładniczo, przez to drugi składnik w postaci funkcji liniowej $y = x$ jest tutaj nieistotny. Funkcja f_2 dąży do zera i, matematycznie, nigdy go nie osiągnie, ale w przypadku naszej ograniczonej arytmetyki tak się jednak dzieje.

6.5.3 $x_0 = 1$ dla funkcji f_2

Nawet nie trzeba uruchamiać żadnego programu testującego — zwróciłby błąd o pochodnej bliskiej zera, oczywiście tutaj chodzi o *dokładnie* zero. Pochodna funkcji f_2 ma postać $-e^{-x} \cdot \boldsymbol{(x - 1)}$, przez co dla $x_0 = 1$ pochodna ta zeruje się (a przecież przez pochodną dzielimy).