Obliczenia naukowe
Sprawozdanie 3

Józef Piechaczek

23 listopada 2019

Notatka do zadań 1, 2 i 3

Do zadań 1, 2 i 3 zostały napisane funkcje testujące znajdujące się w pliku `tests.jl`. Testy sprawdzają wartości miejsc zerowych policzone za pomocą danych metod dla funkcji $f_1(x) = x$, $f_2(x) = x^2$, $f_3(x) = x + 1$, $f_4(x) = sinx$.

1 Zadanie 1

Zadanie 1 polega na utworzeniu funkcji rozwiązującej równianie $f(x) = 0$ za pomocą metody bisekcji.
Definicja funkcji:

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

Dane:

f	funkcja $f(x)$ zadana jako anonimowa funkcja
a, b	końce przedziału początkowego
$delta, epsilon$	dokładności obliczeń

Wyniki:

(r, v, it, err)	czwórka, gdzie
r	przybliżenie pierwiastka równiania
v	wartość $f(r)$
it	liczba iteracji
err	sygnalizacja błędu 0 - brak błędu 1 - funkcja nie zmienia znaku w przedziale $[a, b]$

Opis metody:
Metoda bisekcji korzysta z twierdzenia Darboux do szacowania wartości pierwiastka funkcji. Twierdzenie to mówi, że funkcja ciągła na danym przedziale $[a, b]$ i zmieniająca znak na tym przedziale (co opisuje nierówność $f(a)f(b) < 0$) posiada miejsce zerowe w $[a, b]$. Metoda w pojedynczej iteracji dzieli podany przedział na połowę, obliczając środek przedziału $c = \frac{1}{2}(b + a)$. Następnie obliczamy wartość funkcji w punkcie c, tj. $f(c)$. Sprawdzamy znak wartości $f(c)$ i rozpatrujemy nowy przedział (a, c) albo (c, b) taki, że funkcja zmienia znak w danym przedziale. Metoda kończy działanie w momencie gdy osiągamy zadaną dokładność, tj. przedział jest dostatecznie niewielki lub wartość w punkcie c jest dostatecznie bliska zeru. Metoda ma zbieżność liniową.

Opis algorytmu:
Aby algorytm był jak najlepszy z numerycznego punktu widzenia wprowadziłem następujące zmiany:

1. Środek przedziału liczymy za pomocą wzoru $e = (b - a)/2, c = a + e$. Jest to dokładniejszy sposób niż obliczanie środka klasyczną metodą $c = (a + b)/2$, gdyż klasyczna metoda mogła powodować, iż środek przedziału znajdował się poza przedziałem dla niektórych przypadków.

2. Wartości funkcji na krańcach przedziału porównujemy za pomocą `sign(f(c)) != sign(f(a))`, gdyż sprawdzanie tego za pomocą warunku $f(a)f(b) < 0$ mogło powodować overflow lub underflow.

Algorytm kończy pracę na jeden z dwóch sposobów

1. Gdy zadana dokładność została osiągnięta

2. Gdy funkcja nie zmienia znaku w zadanym przedziale - error code 1

2 Zadanie 2

Zadanie 2 polega na utworzeniu funkcji rozwiązującej równianie $f(x) = 0$ za pomocą metody Newtona.
Definicja funkcji:

```
function mstyczych(f, pf, x0::Float64, delta::Float64,
epsilon::Float64, maxit::Int)
```

Dane:

f, pf	funkcja $f(x)$ oraz pochodna $f'(x)$ zadane jako anonimowe funkcje
$x0$	przybliżenie początkowe
$delta, epsilon$	dokładności obliczeń
$maxit$	maksymalna liczba iteracji

Wyniki:

(r, v, it, err)	czwórka, gdzie
r	przybliżenie pierwiastka równiania
v	wartość $f(r)$
it	liczba iteracji
err	sygnalizacja błędu 0 - brak błędu 1 - nie osiągnieto wymaganej dokładności w $maxit$ operacji 2 - pochodna bliska zeru

Opis metody:
Metoda Newtona polega na lokalnej aproksymacji liniowej za pomocą stycznych. W tej metodzie iteracyjnej x_{n+1} jest określone jako odcięta punktu przecięcia z osią x stycznej do krzywej $y = f(x)$

w punkcie $(x_n, f(x_n))$. Zatem kolejne przybliżenia pierwiastka równiania można określić za pomocą wzoru:

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

Metoda Newtona ma zbieżność kwadratową, zatem zbiega szybciej niż metoda bisekcji i siecznych. Wadą tej metody natomiast jest konieczność liczenia pochodnej funkcji. Inna wadą jest fakt, iż zbieżność nie zachodzi zawsze. W wielu przypadkach metoda jest rozbieżna, kiedy punkt startowy znajduje się zbyt daleko od poszukiwanego miejsca zerowego.

Opis algorytmu:
Program może kończyć działanie na jeden z czterech sposobów:

1. Gdy spełniona jest wymagana dokładność na początku działania programu, tj. $f(x_0) < epsilon$
2. Gdy pochodna jest bliska zeru ($|f'(x_0)| < epsilon$) - funkcja zwraca error code 2
3. Gdy w danej iteracji spełniona jest zadana dokładność ($|x_{n+1} - x_n| < delta$ lub $|f(x_{n+1})| < epsilon$)
4. Gdy nie osiągnięto zadanej dokładności w $maxit$ iteracji - error code 1

3 Zadanie 3

Zadanie 3 polega na utworzeniu funkcji rozwiązującej równianie $f(x) = 0$ za pomocą metody siecznych.
Definicja funkcji:

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

Dane:

f	funkcja $f(x)$ zadana jako anonimowa funkcja
$x0, x1$	przybliżenia początkowe
$delta, epsilon$	dokładności obliczeń
$maxit$	maksymalna liczba iteracji

Wyniki:

(r, v, it, err)	czwórka, gdzie
r	przybliżenie pierwiastka równiania
v	wartość $f(r)$
it	liczba iteracji
err	sygnalizacja błędu 0 - brak błędu 1 - nie osiągnieto wymaganej dokładności w $maxit$ operacji

Opis metody:
Metoda siecznej jest podobna do metody Newtona, jednak w tej metodzie pochodną zastępujemy ilorazem różnicowym postaci:

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

Otrzymujemy więc następujący wzór rekurencyjny

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

Metoda zatem potrzebuje dwóch punktów startowych. Metoda ta ma zbieżność wynoszącą $\frac{1+\sqrt{5}}{2}$, zatem jest wolniejsza od metody Newtona, ale za to nie wymaga liczenia pochodnej funkcji.

Opis algorytmu:
Algorytm kończy działanie na jeden z dwóch sposobów:

1. Gdy spełniona jest zadana dokładność
2. Gdy zadana dokładność nie została osiągnięta w *maxit* iteracji - error code 1

4 Zadanie 4

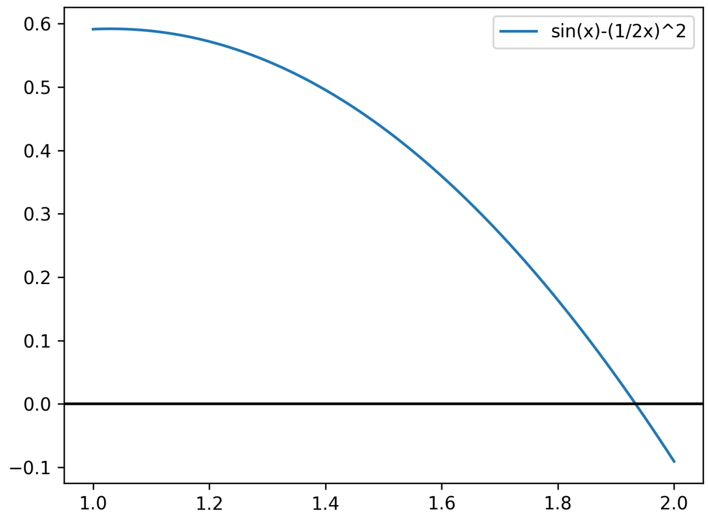
Celem zadania 4 jest wyznaczenie pierwiastków równania $sinx - (\frac{1}{2}x)^2 = 0$ za pomocą zaprogramowanych wcześniej metod:

1. bisekcji z przedziałem początkowym $[1.5, 2]$ i $\delta = \frac{1}{2}10^{-5}$, $\epsilon = \frac{1}{2}10^{-5}$
2. Newtona z przybliżeniem początkowym $x_0 = 1.5$ i $\delta = \frac{1}{2}10^{-5}$, $\epsilon = \frac{1}{2}10^{-5}$
3. siecznych z przybliżeniami początkowymi $x_0 = 1$, $x_1 = 2$ i $\delta = \frac{1}{2}10^{-5}$, $\epsilon = \frac{1}{2}10^{-5}$

Wyniki:

Metoda	Miejsce zerowe x_0	$f(x_0)$	Liczba iteracji	Kod błędu
bisekcja	1.9337539672851562	-2.7027680138402843e-7	16	0
stycznych	1.933753779789742	-2.2423316314856834e-8	4	0
siecznych	1.933753644474301	1.564525129449379e-7	4	0

Wnioski:
Patrząc na wyniki możemy dostrzec iż wszystkie metody wykonały obliczenia poprawnie i nie zwróciły błędu. Najwydajniejsze okazały się metody siecznych i stycznych, co jest zgodne z oczekiwaniami. Metody te mają większą zbieżność od metody bisekcji i wykonały obliczenia w 4 razy mniejszej liczbie iteracji.



Rysunek 1: Wykres w pyplot na przedziale $[1, 2]$

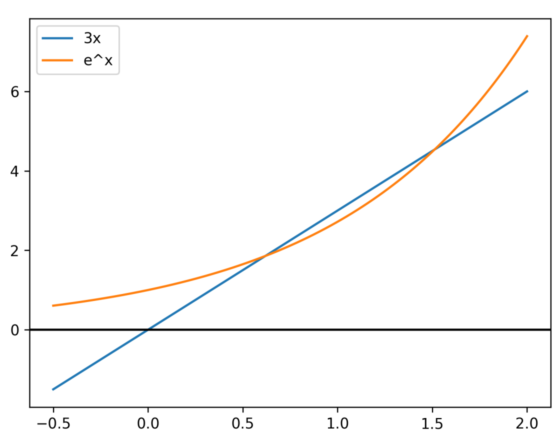
5 Zadanie 5

Celem zadania 5 jest znalezienie za pomocą metody bisekcji wartości zmiennej x, dla której przecinają się wykresy funkcji $y_1 = 3x$ i $y_2 = e^x$. Wymagane dokładności obliczeń wynoszą: $\delta = 10^{-4}$, $\epsilon = 10^{-4}$. Aby znaleźć wartość zmiennej x porównujemy podane funkcje, dzięki czemu otrzymujemy równanie:

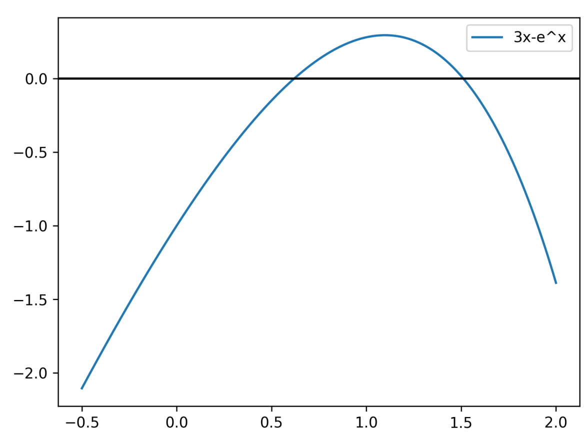
$$3x = e^x$$

Zatem będziemy szukać miejsc zerowych funkcji:

$$f(x) = 3x - e^x$$



Rysunek 2: Wykres w pyplot dla y_1 i y_2



Rysunek 3: Wykres w pyplot dla $f(x)$

Po wstępnej analizie funkcji można zobaczyć, iż funkcja ma dwa miejsca zerowe. Jako przedziały początkowe dla metody bisekcji wybrałem: $[0.5, 1.0]$ oraz $[1.0, 2.0]$

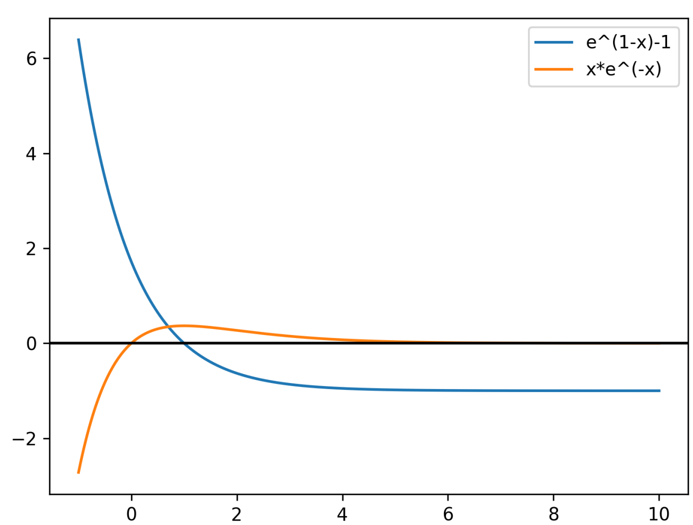
Wyniki:
Wyniki przedstawia następująca tabela:

Miejsce zerowe x_0	$f(x_0)$	Liczba iteracji	Kod błędu
0.619140625	9.066320343276146e-5	8	0
1.5120849609375	7.618578602741621e-5	13	0

Wnioski:
Algorytm obliczył poszukiwane miejsca zerowe oraz nie zwrócił błędu. Oznacza to że wybraliśmy odpowiednie przedziały. W zadaniu tym główną trudność sprawiało właśnie dobranie odpowiednich przedziałów. W celu znalezienia ich posłużyłem się wykresem funkcji, jednak w przypadku braku dostępu do tego narzędzia musielibyśmy wykonać większą liczbę prób lub użyć innego algorytmu.

6 Zadanie 6

Zadanie 6 polega na znalezieniu miejsc zerowych funkcji $f_1(x) = e^{1-x} - 1$ oraz $f_2(x) = xe^{-x}$ za pomocą metod bisekcji, Newtona i siecznych. Wymagane dokładności obliczeń wynoszą $\delta = 10^{-4}$, $\epsilon = 10^{-4}$. Należy dobrać odpowiednio przedział i przybliżenia początkowe.



Rysunek 4: Wykres w pyplot na przedziale $[-1, 10]$

Po wstępnej analizie funkcji można w łatwy sposób dostrzec, że funkcja f_1 ma miejsce zerowe w $x_0 = 1$, a funkcja f_2 ma miejsce zerowe w $x_0 = 0$
Wyniki:
Wyniki przedstawia następujące tabele:

Przedział	Miejsce zerowe r	$f(r)$	Liczba iteracji	Kod błędu
		f_1		
$[0.5, 1.5]$	1.0	0.0	1	0
$[-2.0, 3.0]$	0.9999923706054688	7.629423635080457e-6	17	0
$[-20.0, 10.0]$	0.9999942779541016	5.722062269342132e-6	20	0
$[-2000.0, 3000.0]$	1.0000094771385193	-9.477093611320875e-6	27	0
		f_2		
$[-0.5, 0.5]$	0.0	0.0	1	0
$[-2.0, 3.0]$	7.62939453125e-6	7.62933632381113e-6	17	0
$[-20.0, 15.0]$	-9.5367431640625e-6	-9.53683411396636e-6	19	0
$[-2000.0, 3000.0]$	500.0	3.562288203370643e-215	1	0

Tabela 1: Metoda bisekcji

x_0	Miejsce zerowe r	$f(r)$	Liczba iteracji	Kod błędu
		f_1		
1.0	1.0	0.0	0	0
5.0	0.9999996427095682	3.572904956339329e-7	54	0
6.0	0.9999999573590406	4.264096031825204e-8	147	0
7.5	0.9999994332744109	5.667257496622113e-7	662	0
10.0	-	-	-	1
100.0	-	-	-	2
1000.0	-	-	-	2
		f_2		
0.0	0.0	0.0	0	0
0.5	-3.0642493416461764e-7	-3.0642502806087233e-7	5	0
1.0	-	-	-	2
1.01	102.00999999999992	5.0846685549318855e-43	1	0
10.0	14.380524159896261	8.173205649825554e-6	4	0
100.0	100.0	3.7200759760208363e-42	0	0
1000.0	1000.0	0.0	0	0

Tabela 2: Metoda Newtona dla 10^5 iteracji

x_0, x_1	Miejsce zerowe r	$f(r)$	Liczba iteracji	Kod błędu
		f_1		
$0.5, 1.5$	0.9999999624498374	3.755016342310569e-8	5	0
$0.0, 3.0$	0.999999739048799	2.6095123506486573e-7	9	0
$-5.0, 5.0$	4.975665372740593	-0.9812331895845449	3	0
$-10.0, 10.0$	9.99966600720658	-0.999876548971044	3	0
$0.0, 100.0$	-	-	-	1
		f_2		
$-0, 5, 0.5$	5.38073548562323e-6	5.380706533386756e-6	6	0
$0.5, 1.5$	5.372227830220525e-6	5.372198969466189e-6	9	0
$-8.0, 7.0$	6.999995985033118	0.006383195725973712	1	0
$-10.0, 10.0$	9.999999958776927	0.0004539993144685704	1	0
$-1.0, 100.0$	100.0	3.7200759760208363e-42	1	0

Tabela 3: Metoda siecznych dla 10^5 iteracji

Wnioski:

W metodzie bisekcji w większości przypadków osiągnęliśmy poszukiwany wynik z zadaną dokładnością, nawet mimo wybierania dużych przedziałów. Jedynym wyjątkiem było obliczanie wartości miejsca zerowego dla przedziału $[-2000, 3000]$ dla f_2. Dla danego przedziału otrzymaliśmy wartość 500. Wynika to z faktu, iż wartość funkcji dla środka przedziału w pierwszej iteracji (czyli właśnie dla $x = 500$) jest bardzo bliska zeru i spełnia warunek zakończenia obliczeń $|f(r)| < epsilon$. Zatem stosując metodą bisekcji musimy wybierać przedział ostrożnie w przypadku gdy wartości funkcji są bliskie zeru.

W metodzie Newtona dla funkcji f_1 dla niewielkich wartości osiągaliśmy poprawne miejsce zerowe w stosunkowo niewielkiej liczbie operacji. Jednak wraz z wzrostem wartości x_0 liczba iteracji gwałtownie wzrasta, chociaż wciąż zwraca poprawny wynik. Wynika to z faktu, iż funkcja dla coraz większych wartości wzrasta w coraz wolniejszym tempie, zbliżając się do funkcji stałej. Zatem dla niektórych wartości nie jesteśmy osiągnąć wyniku w zadanej liczbie iteracji - przykładowo dla $x_0 = 10.0$ otrzymujemy error code 1. Dla jeszcze większych wartości x_0 pochodna staję się tak bliska zeru, że otrzymujemy error code 2.

Dla funkcji f_2 metoda zwraca poprawne wartości dla wartości mniejszych od 1. Dla $x_0 = 1$ pochodna wynosi zero, więc otrzymujemy error code 2. Dla wartości większych od 1, nawet minimalnie większych, funkcja okazuje się rozbieżna. Algorytm mimo to jednak zwraca pewne wyniki. Wynika to z faktu, iż podobnie jak w metodzie bisekcji spełniony zostaje warunek zakończenia obliczeń $|f(r)| < epsilon$.

Korzystając z metody Newtona musimy zatem odpowiednio dobierać wartość początkową, tak aby metoda była zbieżna. Musimy również zwrócić szczególną uwagę w przypadku gdy funkcja dąży do zera.

Metoda siecznych zachowała się podobnie do metody Newtona. Dla odpowiednio dobranych wartości początkowych zwracała poprawne miejsce zerowe. W funkcji f_2 po wybraniu niektórych wartości kończyła pracę po jednej iteracji ze względu na spełnienie warunku $|f(r)| < epsilon$, zwracając wartości bliskie x_1. Jedyną różnicą są rozwiązania dla przedziałów $[-5.0, 5.0]$ i $[-10.0, 10.0]$ w f_1. Dla podanych przykładów funkcja zwracała wartości zbliżone do x_1 ponieważ krańce przedziałów przyjmowały wartości bardzo odległe od siebie, więc następne wartości x_n znajdowały się blisko wartości x_1. Algorytm kończył pracę przez spełnienie warunku $|x_{n+1} - x_n| < delta$. Korzystając z metody siecznych musimy zatem uważać, gdy wybieramy wartości np. dla funkcji, która dąży do zera lub gdy wartości na krańcach przedziału znacznie się różnią.