Obliczenia Naukowe Laboratoria - Lista 3

Kajetan Plewa

23 listopada 2025

1 Metoda bisekcji

1.1 Krótki opis problemu

Celem zadania było opracowanie i implementacja numerycznej funkcji o nazwie `bisection`, której zadaniem jest znalezienie przyblizonej wartosci pierwiastka r podanej funkcji.

Funkcja bisection wymaga podania:

- funkcji $f(x)$ (jako funkcji lambda),
- krańców początkowego przedziału $[a, b]$,
- dwóch kryteriów dokładności: δ (tolerancja długości przedziału) oraz ϵ (tolerancja wartości funkcji).

Wynikiem działania funkcji miała być czwórka `(r, v, it, err)`, zawierająca przybliżenie pierwiastka, wartość funkcji w tym punkcie, liczbę wykonanych iteracji oraz kod błędu (1 w przypadku braku zmiany znaku funkcji w przedziale $[a, b]$, 0 w przeciwnym razie).

1.2 Rozwiązanie

Implementacja algorytmu została wykonana na podstawie zaprezentowanego na wykładzie pseudokodu. Kod źródłowy został napisany w języku Julia, co pozwoliło na spełnienie wymogu precyzji **Float64**.

Kroki Algorytmu

1. **Sprawdzenie Wstępne:** Na początku sprawdzono, czy funkcja $f(x)$ zmienia znak na krańcach zadanego przedziału $[a, b]$. Jeśli $f(a)\cdot f(b) > 0$, oznacza to, że w tym przedziale nie ma pierwiastka (lub jest parzysta ich liczba), a funkcja zwraca błąd `err` $= 1$.

2. **Pętla Iteracyjna:** Dopóki długość przedziału $|b - a|$ jest większa niż tolerancja δ (główne kryterium stopu) i nie jest spełnione kryterium ϵ, algorytm wykonuje następujące kroki:
   - Obliczenie punktu środkowego: $r = (a + b)/2$.
   - Sprawdzenie kryterium wartości: Jeśli $|f(r)| \leq \epsilon$, algorytm kończy działanie i zwraca r jako pierwiastek.
   - Wybór nowego przedziału: Sprawdzenie, w której z nowych połówek ($[a, r]$ czy $[r, b]$) funkcja zmienia znak. Nowy przedział jest wybierany tak, aby $f(x)$ wciąż zmieniała w nim znak, skutecznie zmniejszając jego długość o połowę.
   - Inkrementacja licznika iteracji `it`.

3. **Zakończenie:** Po zakończeniu pętli `for` (gdy $|b - a| \leq \delta$), ostateczne przybliżenie pierwiastka r jest przyjmowane jako środek ostatniego przedziału, a funkcja zwraca wyniki wraz z `err` $= 0$.

2 Metoda newtona

2.1 Krótki opis problemu

Podobnie jak ostatnio, celem implementacji tej metody jest znalezienie pierwiastka podanej funkcji f.

Funkcja newton wymaga podania:

- funkcji $f(x)$ (jako funkcji lambda),
- funkcji $f'(x)$ - pochodnej funkcji lambda,
- przybliżenie początkowe $[x0]$,
- dwóch kryteriów dokładności: δ (tolerancja długości przedziału) oraz ϵ (tolerancja wartości funkcji).
- *maxit* - maksymalnej liczby iteracji

Wynikiem działania funkcji miała być czwórka `(r, v, it, err)`, zawierająca przybliżenie pierwiastka, wartość funkcji w tym punkcie, liczbę wykonanych iteracji oraz kod błędu (1 w przypadku nie osiągnięcia wymaganej dokładności w *maxit* iteracji, 2 w przypadku pochodnej bliskiej zeru i 0 w przeciwnym razie).

2.2 Rozwiązanie

Implementacja algorytmu została wykonana na podstawie zaprezentowanego na wykładzie pseudokodu. Kod źródłowy został napisany w języku Julia, co pozwoliło na spełnienie wymogu precyzji **Float64**.

Kroki Algorytmu

1. **Wstępna weryfikacja**: Na początku sprawdzamy czy podane na wejściu przybliżenie nie jest wystarczająco dokładne. Jeśli $|v| \leq \epsilon$ kończymy działanie algorytmu.
2. W przeciwnym wypadku rozpoczynamy działanie algorytmu. **Pętla iteracyjna**
   - Obliczenie wartości pochodnej w punkcie $x0$.
   - Sprawdzenie czy $|pv| \leq 1.0 \cdot 10^{-15}$. If **true** -> zwracamy error 2 gdyż pochodna jest praktycznie równoległa do osi **OX**. Przecięcie znajduje się daleko - funkcja zwracałaby nieprawidłowe wyniki.
   - Wyliczenie nowej wartości $x_1 = x_0 - \frac{v}{pv}$
   - Obliczenie nowego $v = f(x_1)$ i sprawdzenie czy $|x_1 - x_0| \leq \delta |||v| \leq \epsilon$. If **true** -> zwracamy wyniki.
   - w przeciwnym przypadku $x_0 = x_1$
3. Jeśli nie uda osiągnąć się oczekiwanej precyzji w *maxit* zwracamy $error = 1$

3 Metoda siecznych

3.1 Krótki opis problemu

Przybliżenie pierwiastka funkcji f poprzez geomatryczne wykorzystanie równania siecznej: $x_(n+1) = x_n - v_0 * \frac{x_(n+1) - x_n}{v_(n+1) - v_n}$.

Funkcja secant wymaga podania:

- funkcji $f(x)$ (jako funkcji lambda),

- przybliżenie początkowe x_0, x_1,
- dwóch kryteriów dokładności: δ (tolerancja długości przedziału) oraz ϵ (tolerancja wartości funkcji).
- *maxit* - maksymalnej liczby iteracji

Wynikiem działania funkcji miała być czwórka `(r, v, it, err)`, zawierająca przybliżenie pierwiastka, wartość funkcji w tym punkcie, liczbę wykonanych iteracji oraz kod błędu (1 w przypadku nie osiągnięci wymaganej dokładności w *maxit* iteracji, 0 w przeciwnym razie).

3.2 Rozwiązanie

Implementacja algorytmu została wykonana na podstawie zaprezentowanego na wykładzie pseudokodu. Kod źródłowy został napisany w języku Julia, co pozwoliło na spełnienie wymogu precyzji **Float64**.

Kroki Algorytmu

1. Wyliczenie początkowych wartości $v_0 = f(x_0)$ i $v_1 = f(x_1)$. Co ważne x_1 jest wartością nowszą i bliższą pierwiastkowi.
2. **Pętla iteracyjna:**
   - Sprawdza czy $|v_0| \geq |v_1|$ -> If **true** wtedy ją zamienia - ta linijka kodu ma na zadaniu utrzymanie niezmiennika - x_1 jest zawsze bliższym przybliżeniem.
   - Sprawdza czy $|v_1 - v_0| \leq 1.0 \cdot 10^{-15}$ -> If **true** zwraca wyniki gdyż sieczna jest praktycznie równoległa do osi **OX**. Przecięcie znajduje się daleko - funkcja zwracałaby niedokładne wyniki.
   - Wyliczenie nowego przybliżenia ze wzoru na sieczną - $x_1 = x_0 - v_0 * \frac{x_1 - x_0}{v_1 - v_0}$.
   - Sprawdzenie $|x_1 - x_0| \leq \delta |||v_0| \leq \epsilon$ -> If **true** znaczy że spełnilismy wymogi zadania - zwracamy wynik.
3. **Zakończenie:** Jeśli przekroczyliśmy *maxit* zwracamy $error = 1$.

4 Zadanie 4

4.1 Krótki opis problemu

Zadanie polegało na przetestowaniu zaimplementowanych metod i porównanie otrzymanych wyników.

4.2 Wyniki i ich interpretacja

Poniższa tabela zestawienia wyniki uzyskane dla każdej metody. Wszystkie metody osiągnęły wymaganą zbieżność (err = 0).

Tabela 1: Porównanie Metod dla $f(x) = \sin(x) - (0.5x)^2$

Metoda	**Liczba Iteracji (it)**	**Pierwiastek r**	**Wartość $f(r)$**	**Rząd Zbieżności**
Bisekcji	16	1.933753967 ...	$-2.70 \cdot 10^{-7}$	Liniowy ($p = 1$)
Newtona	4	1.933753780 ...	$-2.24 \cdot 10^{-8}$	Kwadratowy ($p = 2$)
Siecznych	5	1.933753877 ...	$-1.51 \cdot 10^{-7}$	Nadliniowy ($p \approx 1.618$)

4.3 Interpretacja Wyników

Wyniki są w pełni zgodne z teorią rzędów zbieżności metod numerycznych:

1. **Metoda Newtona** okazała się najszybsza, osiągając wymaganą dokładność już po **4 iteracjach**. Jest to bezpośrednią konsekwencją jej **kwadratowego** rzędu zbieżności ($p = 2$), gdzie liczba poprawnych cyfr podwaja się w każdej iteracji. Uzyskana wartość funkcji ($|v| \approx 2.24 \cdot 10^{-8}$) jest najmniejsza, potwierdzając najwyższą precyzję.

2. **Metoda Siecznych** osiągnęła zbieżność w **5 iteracjach**. Jej wydajność jest bliska Metodzie Newtona dzięki **nadliniowemu** rzędowi zbieżności ($p \approx 1.618$).

3. **Metoda Bisekcji** wymagała **16 iteracji**. Jej **liniowy** rząd zbieżności ($p = 1$) powoduje, że jest najwolniejsza, ponieważ długość przedziału jest dzielona tylko przez dwa w każdym kroku.

4.4 Wnioski

Metoda Newtona okazała się najbardziej efektywną funkcją aproksymacji pierwiastka. Posiada ona jednak dość istotny minus - wymaga policzenia pochodnej. W pojedynczym przypadku nie jest to problem - łatwo jest ją wyprowadzić samemu, lecz po wprowadzeniu możliwości wpisywania funkcji przez użytkownika, należałoby zastosować metody aproksymacji pochodnej. Przykładem takiej aproksymacji jest metoda siecznych - jak widać wyniki przez nią zwrócone są tylko o rząd mniej dokładne, a liczba iteracji nieznaczącą większa.
W przypadku metody bisekcji widać, że precyzja nie jest o wiele gorsza od tej zwrócnej przez funkcje siecznej. Natomiast liczba iteracji rośnie kwadratowo co jest poważnym problemem.
Podsumowując - metoda Newtona jest bezsprzecznie najszybsza i najdokładniejsza. Jednakże, w przypadku projektowania algorytmu ogólnego, przyjmującego dowolną funkcje, metoda siecznych wydaje się być najlepszym kompromisem.

5 Zadanie 5

5.1 Opis Problemu

Zadanie polegało na znalezieniu wartości zmiennej x, dla której wykresy funkcji $y = 3x$ oraz $y = e^x$ się przecinają. Problem został przekształcony do znalezienia pierwiastków równania:

$$f(x) = e^x - 3x = 0$$

Z analizy wynika, że równanie to posiada dwa pierwiastki. Do rozwiązania użyto **Metody Bisekcji** z wymaganej dokładności obliczeń $\delta = 10^{-4}$ i $\epsilon = 10^{-4}$.

5.2 Dobór Przedziałów i Wyniki

Pierwiastki zostały znalezione poprzez dobór dwóch różnych przedziałów, w których funkcja $f(x)$ zmienia znak (co jest koniecznym warunkiem dla Metody Bisekcji).

Tabela 2: Wyniki Metody Bisekcji dla $f(x) = e^x - 3x$

Pierwiastek	**Przedział Startowy**	**Liczba Iteracji (it)**	**Pierwiastek r**	**Wartość $f(r)$**
r_1	$[0.5, 1.0]$	9	0.619140625	$9.07 \cdot 10^{-5}$
r_2	$[1.5, 2.0]$	13	1.5120849609375	$7.62 \cdot 10^{-5}$

5.3 Interpretacja Wyników

1. **Zbieżność Gwarantowana:** Metoda Bisekcji, będąca metodą przedziałową, z powodzeniem znalazła oba pierwiastki w zadanym przedziale, co jest typowe dla tej niezawodnej metody. Oba wyniki osiągnęły wymaganą dokładność (err = 0).

2. **Liczba Iteracji (it):** Dla pierwiastka r_1, startując z przedziału o długości $L_0 = 0.5$, wymagane 9 iteracji jest w pełni zgodne z liniowym rzędem zbieżności.
   Dla pierwiastka r_2, startując z przedziału o długości $L_0 = 0.5$, wymagane 13 iteracji jest wyższe, co wskazuje, że kryterium stopu mogło być kontrolowane przez dokładność wartości ϵ, lub przedział zbieżności był nieco wolniejszy.

5.4 Wnioski

Funkcja bisekcji prawidłowo znajduje miejsce zerowe funkcji $y = 3x - e^x$. Co więcej widać, że im bliżej pierwiastek jest któregoś z ograniczeń przedziałów, tym więcej iteracji musi wykonać bisekcja.

6 Zadanie 6

6.1 Opis Problemu

Celem zadania było wyznaczenie miejsc zerowych dla dwóch funkcji $f_1(x) = e^{1-x} - 1$ oraz $f_2(x) = x \cdot e^{-x}$, stosując metody Bisekcji, Newtona i Siecznych. Wymagana dokładność to $\delta = 10^{-5}$ i $\epsilon = 10^{-5}$. Drugą częścią zadania była analiza zachowania Metody Newtona w niekorzystnych warunkach początkowych.

6.2 Wyniki Standardowe (Porównanie Metod)

Wszystkie metody skutecznie znalazły pierwiastki ($r = 1$ dla f_1 i $r = 0$ dla f_2).

Tabela 3: Porównanie Metod dla Zadania 6

Funkcja	**Metoda**	**Liczba Iteracji (it)**	**Pierwiastek r**	**Wartość $f(r)$**
$f_1(x) = e^{1-x} - 1$	Bisekcji	16	$0.99999237\ldots$	$7.63 \cdot 10^{-6}$
	Newtona	**4**	$0.99999843\ldots$	$1.56 \cdot 10^{-6}$
	Siecznych	**4**	$0.99999941\ldots$	$5.90 \cdot 10^{-7}$
$f_2(x) = x \cdot e^{-x}$	Bisekcji	16	$7.63 \cdot 10^{-6}$	$7.63 \cdot 10^{-6}$
	Newtona	5	$-3.06 \cdot 10^{-7}$	$-3.06 \cdot 10^{-7}$
	Siecznych	5	$-1.22 \cdot 10^{-7}$	$-1.22 \cdot 10^{-7}$

Interpretacja Wyników Standardowych

1. Metody Newtona i Siecznych osiągnęły zbieżność w zaledwie 4-5 iteracjach, demonstrując swoje wyższe rzędy zbieżności ($p = 2$ i $p \approx 1.618$).

2. Metoda Bisekcji, o liniowym rzędzie ($p = 1$), wymagała 16 iteracji, co jest zgodne z teoretycznym minimum dla osiągnięcia dokładności $\delta = 10^{-5}$ w przedziałach startowych o długości około 1.0.

6.3 Analiza Przypadków Granicznych Metody Newtona

Zbadano zachowanie Metody Newtona w warunkach, które naruszają warunki zbieżności lub stabilności.

Tabela 4: Wyniki Analizy Przypadków Granicznych Metody Newtona

Funkcja	**Warunek Startowy**	**Wynik** r	**Iteracje (it)**	**Err**	**Wniosek**
$f_1(x) = e^{1-x} - 1$	$x_0 \in (1, \infty]$	14.39866...	10	0	Zbieżność do dużej wartości, poza pierwiastkiem $r = 1$.
$f_2(x) = x \cdot e^{-x}$	$x_0 > 1$	14.39866...	10	0	Zbieżność do fałszywego pierwiastka ($r \neq 0$). Zły wybór x_0.
$f_2(x) = x \cdot e^{-x}$	$x_0 = 1$	1.0	1	2	**Błąd Stabilności**: Wykryto $f'(x_0) \approx 0$ (punkt krytyczny).

Interpretacja Przypadków Granicznych

1. **Niestabilna Zbieżność (Warunki $x_0 > 1$):** Zły wybór przybliżenia początkowego spowodował zbieżność algorytmu do niepoprawnej, bardzo dużej wartości ($r \approx 14.4$). To podkreśla wrażliwość Metody Newtona na wybór x_0, gdy nie ma gwarancji zbieżności.
2. **Dzielenie przez Zero ($x_0 = 1$ dla f_2):** W punkcie $x_0 = 1$ pochodna $f_2'(x)$ wynosi zero, co powoduje próbę dzielenia przez zero we wzorze Newtona. Algorytm słusznie zwrócił kod błędu 2, sygnalizując niestabilność numeryczną w pobliżu lokalnego ekstremum.

6.4 Wnioski

Dla wyników standardowych wnioski byłyby podobne co w zadaniu 4.
Warto zająć się więc podanymi przykładami "granicznymi". Pokazują one cechę metody Newtona, która wcześniej nie musiała być oczywista - jest ona podatna na niestabilność spowodowaną m.in. równelogłości pochodnej do osi OX.
Podsumowując, podczas wybierania metody do aproksymacji pierwiastka funkcji należy brać pod uwagę nie tylko szybkość i dokładność, ale także niezawodność. Z wprowadzonych w tym sprawozdaniu funkcji tylko metoda bisekcji gwarantuje nam zwrócenie prawidłowego wyniku - dzieje się to jednak kosztem efektywności. Zadanie to jest świetnym przykładem że, programista zawsze powinien dobierać odpowiednie narzędzia z ostrożnością, analizując wymagania zadania. Osobiście traktowałbym metodę bisekcji jako funkcje "fallback", która byłaby wykonywana w przypadku zwrócenia błędu przez którąś z pozostałych metod.