Akademia Górniczo-Hutnicza
Wydział Informatyki
Kierunek Informatyka

Metody Obliczeniowe w Nauce i Technice

Laboratorium 8

Równania nieliniowe

Wojciech Michaluk, Kyrylo Iakymenko

Kraków, 26 kwietnia 2024

1 Wprowadzenie

W ramach zadań z dzisiejszego laboratorium przeanalizujemy sposób działania metod iteracyjnych rozwiązywania równań nieliniowych na przykładzie metody Newtona. Sprawdzimy, czy wyznaczony doświadczalnie rząd zbieżności metody zgadza się z wartością teoretyczną oraz jaki ma on wpływ na liczbę cyfr po przecinku, które są wyznaczone dokładnie. Zwrócimy także uwagę na przypadki, kiedy metoda Newtona nie działa zgodnie z zamierzeniem i stosując tylko tę metodę, nie uda się znaleźć pierwiastka równania - wtedy należy pomocniczo użyć innej metody, np. metody bisekcji. Wykorzystamy uzyskaną wiedzę również do rozwiązania układu równań nieliniowych, traktując go jako funkcję wielu zmiennych, rozszerzając metodę Newtona na przypadek większej liczby zmiennych.

Metody iteracyjne można w ogólnym przypadku opisać następująco:

- szukamy rozwiązania równania $f(x) = 0$ (w razie potrzeby dokonujemy odpowiednich przekształceń),
- wybieramy pewien punkt początkowy x_0 oraz funkcję $g(x)$, którą w każdej iteracji działamy na punkt z poprzedniej iteracji: $x_{k+1} = g(x_k)$ dla $k = 0, 1, ...,$
- powtarzamy obliczenia aż do osiągnięcia przyjętego warunku stopu, np. $|f(x_k) < \epsilon|$ dla pewnego ϵ lub ustalonej liczby iteracji.

W przypadku metody Newtona $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

2 Zadanie 1

2.1 Opis zadania

W pierwszym zadaniu zbadamy metodę Newtona na wybranych funkcjach, co pozwoli nam zaobserwować jej wady i porównać do innej metody iteracyjnej - metody *bisekcji*. W tym zadaniu użyliśmy funkcji `scipy.optimize.newton`, którą importujemy pod nazwą `newton` oraz `scipy.optimize.bisect` używanej pod nazwą `bisect`.

Poniżej przedstawiamy badane funkcje oraz punkt początkowy x_0, w którym zaczynamy iterację:

(a) $f(x) = x^3 - 5x, x_0 = 1$

(b) $f(x) = x^3 - 3x + 1, x_0 = 1$

(c) $f(x) = 2 - x^5, x_0 = 0.01$

(d) $f(x) = x^4 - 4.29x^2 - 5.29, x_0 = 0.8$

2.2 Opracowanie zadania

Najpierw używamy funkcji `newton` bez dodatkowych argumentów:

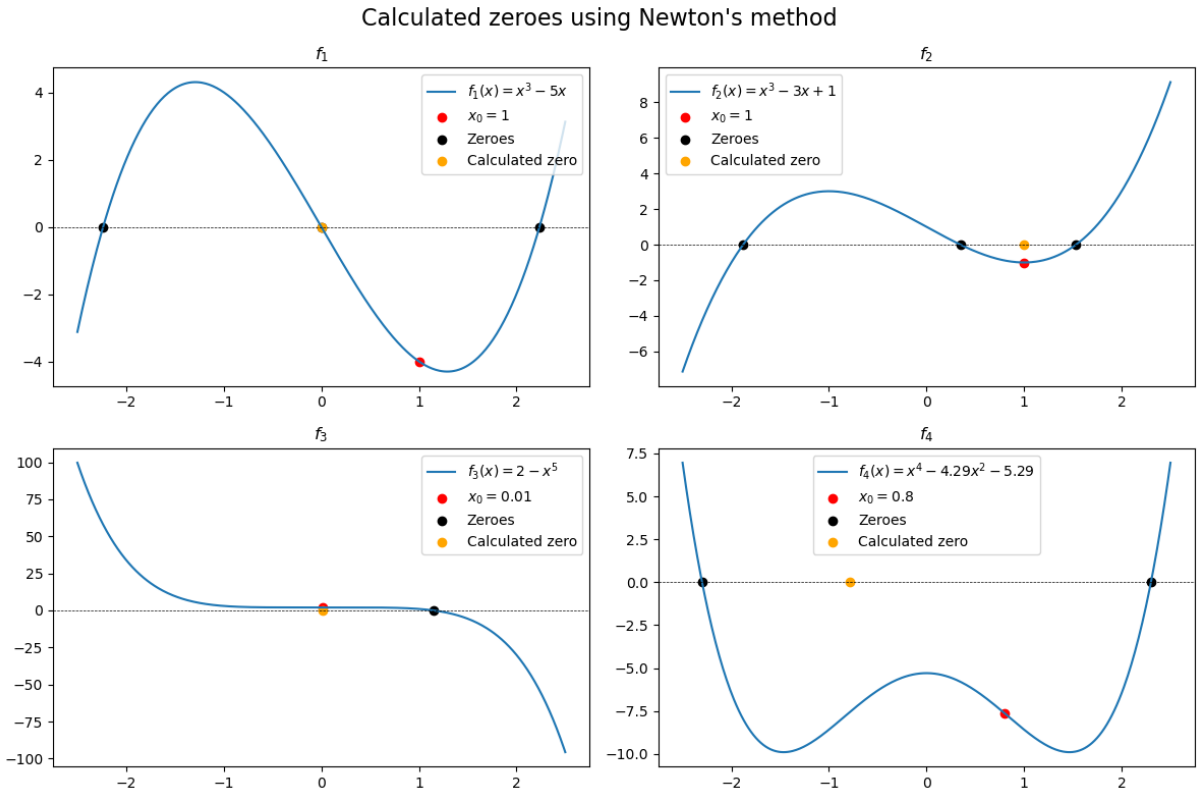
```
newton(f, x0) # powtarzamy dla każdej funkcji f
```

Listing 1: Wywołanie funkcji *newton* (nie używaliśmy dodatkowych argumentów)

Poniżej przedstawiamy tabelę i wykres z uzyskanymi wynikami:

Funkcja	Miejsce zerowe
f_1	4.745×10^{-24}
f_2	1
f_3	0.010
f_4	-0.787

Tabela 1: Zaokrąglone wartości obliczonych zer badanych funkcji (metoda Newtona)



Rysunek 1: Wspólny wykres badanych funkcji z obliczonymi miejscami zerowymi (metoda Newton'a).

Jak widzimy, tylko w przypadku f_1 udało nam się w miarę dokładnie wyznaczyć miejsce zerowe. W przypadku funkcji f_2 i f_3 pochodna w punkcie x_0 wynosi 0, co spowodowało zwrócenie wartości x_0 jako miejsca zerowego. W przeciwnym wypadku mogłoby dojść do dzielenia przez 0. W przypadku funkcji f_4 prawdopodobnie metoda `newton` przyjmuje przedział, w którym nasza funkcja nie zmienia znaku, co jest jednym z warunków koniecznych na poprawne działanie metody Newtona (wtedy mamy gwarancję, że w badanym przedziale jest pierwiastek).

Jak się okazuje, oprócz wspomnianego warunku jest kilka innych, które stanowią warunki wystarczające (a w powyższym przypadku nie były spełnione), żeby metoda Newtona była zbieżna do szukanego pierwiastka, mianowicie:

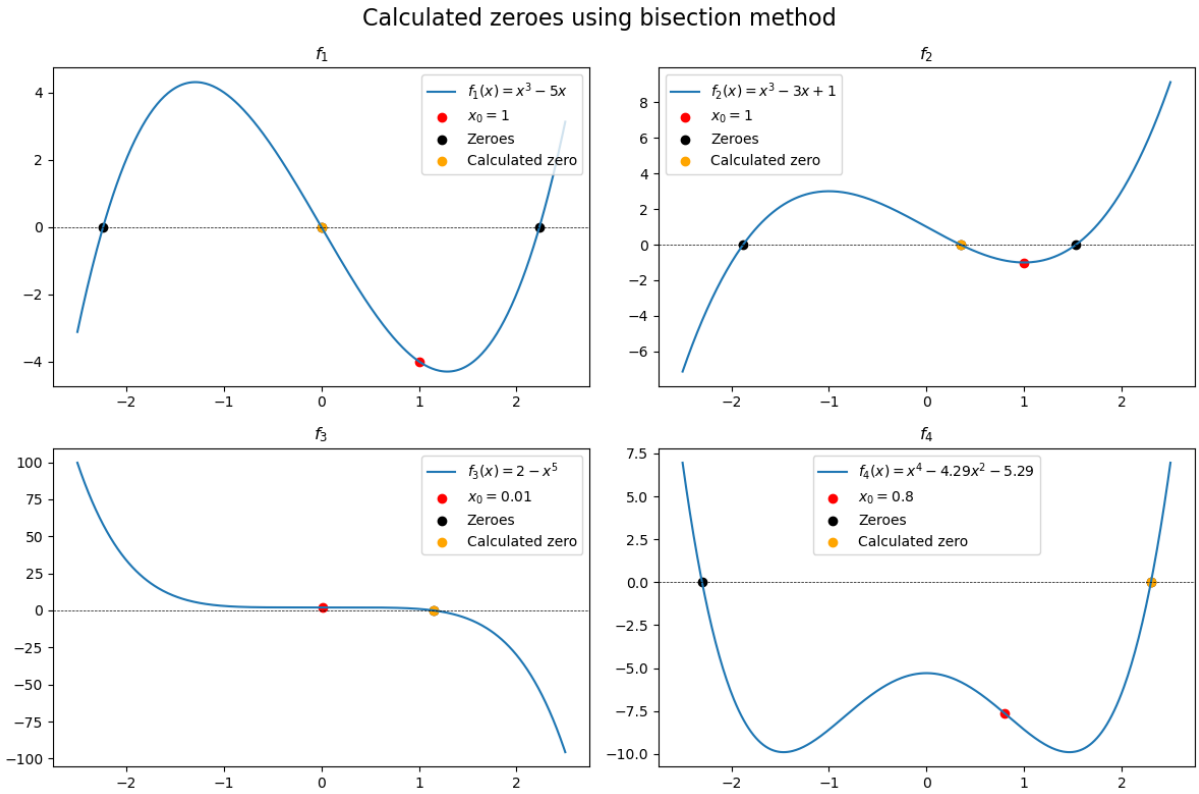
1. $f \in C^2[a;b]$, co oznacza, że f, f', f'' są ciągłe na przedziale $[a;b]$,
2. f' i f'' nie zmieniają znaku na przedziale $[a;b]$,
3. x_0 spełnia warunek $f(x_0) \cdot f''(x_0) > 0$, gdzie $x_0 = f(a)$ lub $x_0 = f(b)$.

Jeżeli x_0 należy do przedziału $[a;b]$ spełniającego te warunki, to mamy gwarancję zbieżności. Stając przed wyborem użycia metody Newtona, ale z innymi parametrami lub użycia innej metody, postanowiliśmy, że porównamy teraz działanie metody Newtona do metody bisekcji na tych samych funkcjach. W uproszczeniu, metoda bisekcji w każdej iteracji zawęża rozważany przedział dwukrotnie, przesuwając jeden z końców przedziału na środek przedziału w zależności od znaku przyjmowanego w tych punktach (warunkiem jest, aby iloczyn wartości funkcji w końcach nowego przedziału był ujemny).

Poniżej przedstawiamy wyniki obliczeń (użyliśmy metody `bisect`):

Funkcja	Miejsce zerowe
f_1	0
f_2	0.347
f_3	1.149
f_4	2.300

Tabela 2: Zaokrąglone wartości obliczonych zer badanych funkcji (metoda bisekcji)



Rysunek 2: Wspólny wykres badanych funkcji z obliczonymi miejscami zerowymi (metoda bisekcji)

We wszystkich przypadkach zostały wyznaczone miejsca zerowe dużo bardziej dokładne w porównaniu do metody Newtona. Widzimy to zarówno na wykresie, jak i w tabeli. Metoda bisekcji nie ma problemów z miejscami, w których pochodna jest równa 0 i najprawdopodobniej dopasowanie przedziału przez funkcję `bisect` zapewnia, że badana funkcja jest rozpatrywana na przedziale, w którym następuje zmiana znaku (a więc istnieje tam pierwiastek - z *tw. Darboux*) co powoduje, że metoda bisekcji nie zawiodła dla żadnego z rozpatrywanych wielomianów. Z drugiej strony warto wspomnieć, że stosowanie metody bisekcji na bardziej skomplikowanych problemach (np. wyznaczanie od razu dużej ilości miejsc zerowych) może się wiązać z większym kosztem obliczeniowym ze względu na większą złożoność tej metody.

3 Zadanie 2

3.1 Opis zadania

W zadaniu drugim będziemy badać różne (ale równoważne) postacie schematu iteracyjnego dla wielomianu $f(x) = x^2 - 3x + 2$:

$$\begin{aligned} g_1(x) &= (x^2 + 2)/3 \\ g_2(x) &= \sqrt{3x - 2} \\ g_3(x) &= 3 - 2/x \\ g_4(x) &= (x^2 - 2)/(2x - 3) \end{aligned} \tag{1}$$

Najpierw przeanalizujemy zbieżność oraz rzędy zbieżności tych schematów iteracyjnych oraz zbadamy, czy ta analiza teoretyczna pokrywa się z wartością uzyskaną poprzez implementację tych schematów iteracyjnych.

Wyznaczymy empiryczne rzędy zbieżności i przedstawimy na wspólnym wykresie najpierw dla wszystkich metod, później tylko dla metod, które okazały się zbieżne, wartość błędu względnego w zależności od numeru iteracji, ograniczając się do 10 iteracji.

3.2 Opracowanie zadania

Najpierw zajmiemy się analizą teoretyczną problemu: sprawdzimy, które z danych schematów są zbieżne oraz wyznaczymy ich rząd zbieżności. W tym celu policzymy wartość $|g_i\prime(2)|$ dla $i = 1, 2, 3, 4$ i sprawdzimy, czy jest mniejsza od 1.

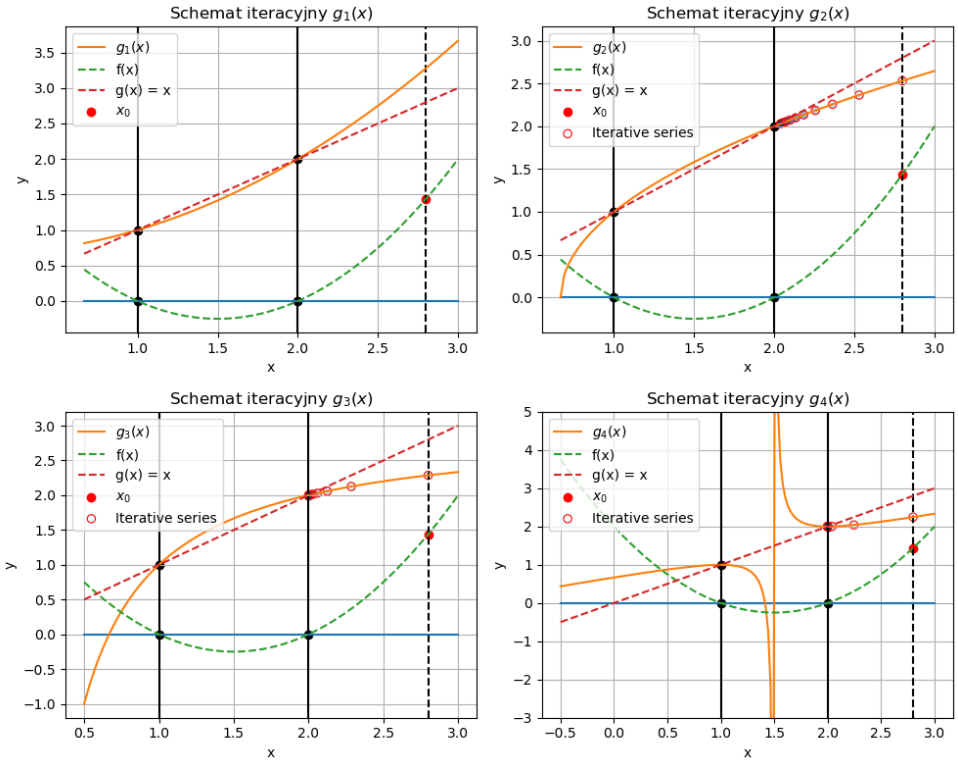
$$\begin{aligned} |g_1'(2)| &= 4/3 \\ |g_2'(2)| &= 3/5 \\ |g_3'(2)| &= 1/2 \\ |g_4'(2)| &= 0 \end{aligned} \tag{2}$$

W teorii wszystkie schematy oprócz pierwszego powinny być zbieżne. Zobaczmy teraz, jak to wygląda w praktyce, mając do czynienia z arytmetyką komputerową. Wybraliśmy punkt startowy $x_0 = 2.8$, ale zmiana punktu startowego na inny znajdujący się w przedziale zbieżności schematu iteracyjnego nie miała większego wpływu na wyniki (testowaliśmy również punkty 3, 3.5, 4). Poniżej przedstawiona jest tabela z pierwszymi 10 wyrazami każdego ciągu.

i	$g_1(x)$	$g_2(x)$	$g_3(x)$	$g_4(x)$
1	2.800	2.800	2.800	2.800
2	3.280	2.530	2.286	2.246
3	4.253	2.364	2.125	2.041
4	6.695	2.257	2.059	2.002
5	15.610	2.184	2.029	2.000
6	81.887	2.134	2.014	2.000
7	2235.814	2.098	2.007	2.000
8	$1.667e+6$	2.072	2.003	2.000
9	$9.256e+11$	2.053	2.002	2.000
10	$2.850e+23$	2.040	2.001	2.000

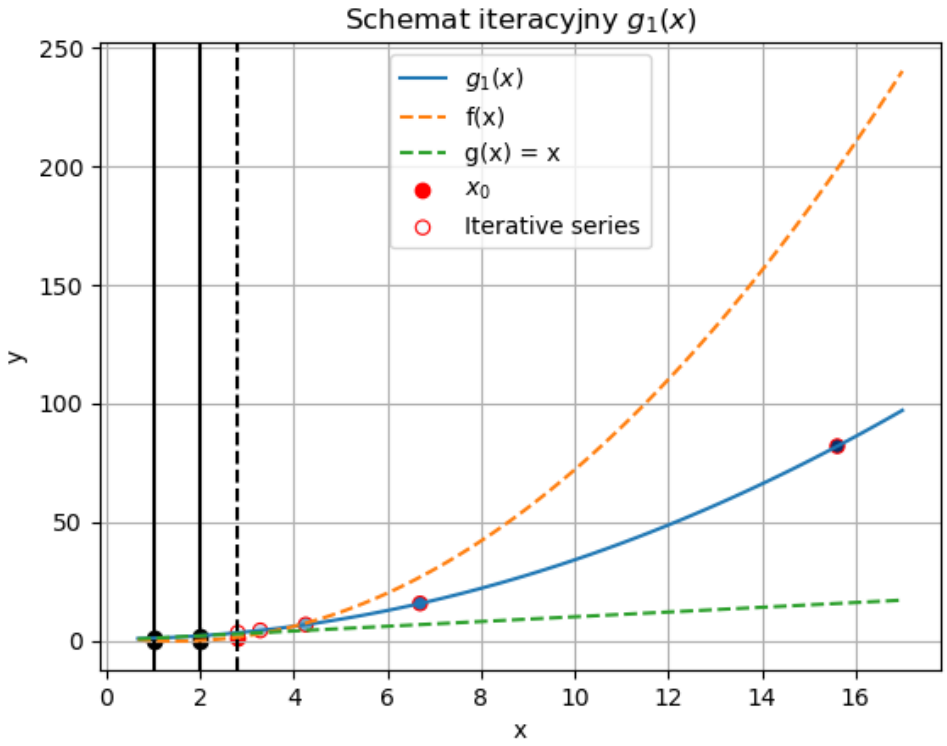
Tabela 3: Wartości wyrazów ciągów badanych schematów iteracyjnych (pierwsze 10 wyrazów)

Widać, że schemat numer 1 jest jedynym rozbieżnym ze wszystkich badanych. Przedstawiamy wyrazy ciągu naszych schematów na oddzielnych wykresach.



Rysunek 3: Wykresy wartości schematów iteracyjnych

Następnie prezentujemy wykres dla wyrazów ciągu dla pierwszego schematu, żeby lepiej było widać rozbieżność.



Rysunek 4: Wykres pierwszego schematu iteracyjnego

Możemy zaobserwować ciekawą geometrię wykresów naszych metod. Cztery punkty zaznaczone na wykresach na czarno jednoznacznie wyznaczają trapez prostokątny (leżący na boku), który nie jest zależny od wybranej metody. Tak naprawdę na zbieżność będzie miała wpływ wartość pochodnej w punkcie $x = 2$, co można również na oko oszacować patrząc na dany wykres. Możemy również próbować przewidzieć rząd zbieżności metody oraz ogólnie to, jak szybko będzie zbiegała do rzeczywistego miejsca zerowego, patrząc na zachowanie wykresu funkcji $g_i(x)$ na odcinku $[2, x_0]$, gdzie x_0 jest naszym punktem startowym. Na wykresie ten odcinek jest pomiędzy drugą czarną prostą pionową a prostą przerywaną. Z tego możemy na przykład przewidzieć, że metoda pierwsza nie jest zbieżna (widzimy jak szybko rośnie na tym kluczowym dla nas przedziale). Również możemy zabserwować to, że metoda 4 byłaby jednym z kandydatów na mającą największy rząd zbieżności, jej pochodna na przedziale od 2 do x_0 jest bardzo mała. Stosując podobną analizę możemy wytłumaczyć szybszą zbieżność metody 3 w porównaniu do 2 (chociaż rzędy zbieżności okazują się jednakowe).

Teraz zbadamy rzędy zbieżności r naszych metod. W tym celu skorzystamy ze wzoru

$$r = \frac{ln\frac{\varepsilon_k}{\varepsilon_{k+1}}}{ln\frac{\varepsilon_{k-1}}{\varepsilon_k}},$$

gdzie ε_k to błąd bezwzględny w k-tej iteracji, tzn. $|x_k - x_*|$ dla x_* - dokładne położenie pierwiastka równania. Zobaczmy, jak wygląda to w kodzie:

```
def convergence_order(data):
        res = []
        for i in range(1, len(data) - 1):
                res.append(np.log(data[i] / data[i + 1]) /
                  np.log(data[i - 1] / data[i]))
        return res
```

Listing 2: Obliczanie rzędu zbieżności dla otrzymanych ciągów

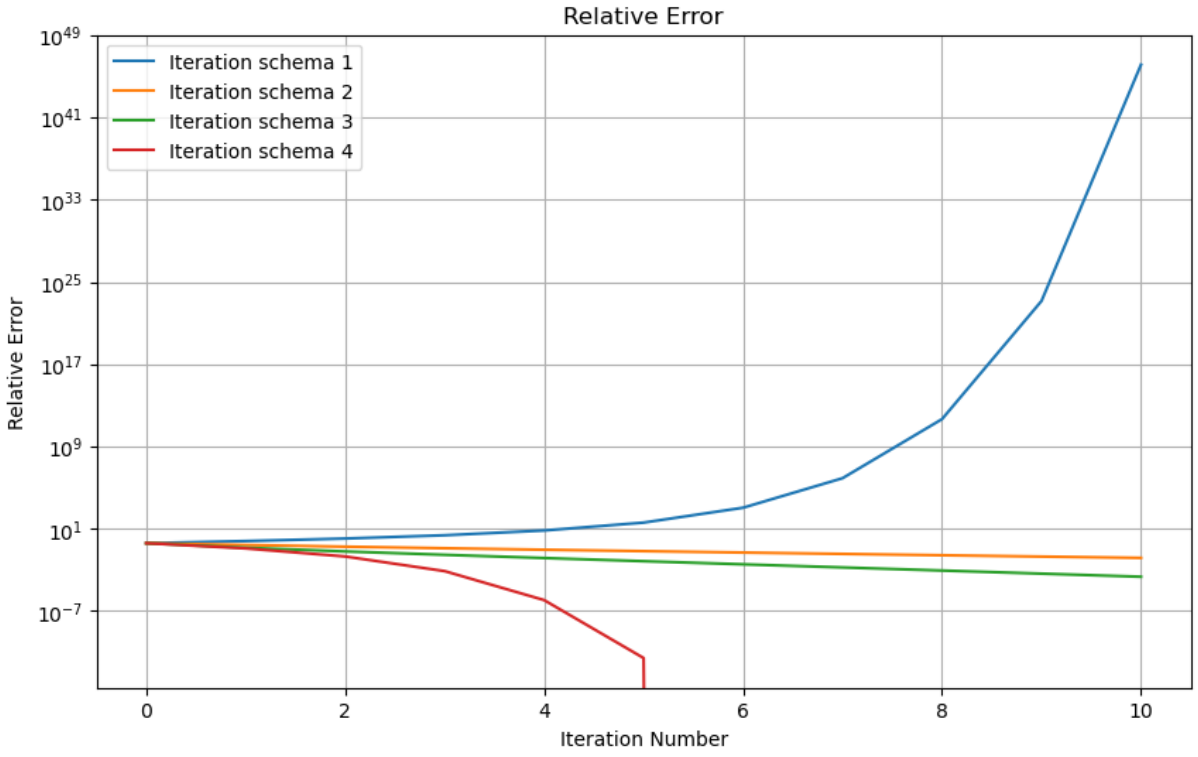
Poniżej przedstawiamy tabele z wynikami dla $k = 2, 3..., 10$.

k	$g_1(x)$	$g_2(x)$	$g_3(x)$	$g_4(x)$
2	1.20	0.90	0.80	1.52
3	1.29	0.93	0.91	1.82
4	1.44	0.95	0.95	1.97
5	1.66	0.96	0.97	1.99
6	1.88	0.97	0.98	inf
7	1.98	0.97	0.99	nan
8	1.99	0.98	0.99	nan
9	2.00	0.98	0.99	nan
10	2.00	0.99	0.99	nan

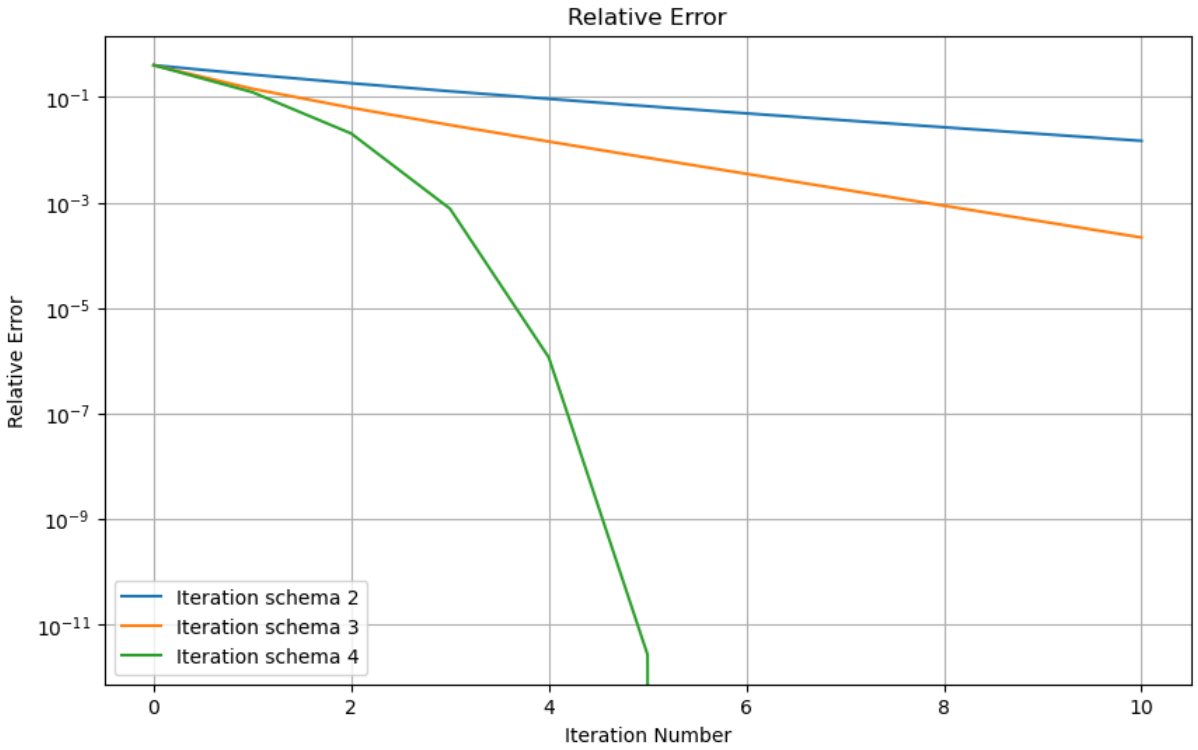
Tabela 4: Wartości empirycznych rzędów zbieżności badanych schematów iteracyjnych

Jak widzimy, empiryczne rzędy zbieżności 1. i 4. metody są równe 2, zatem mamy do czynienia z metodami kwadratowymi, w przypadku metody 4. w pewnym momencie osiągana jest dokładna wartość 2, będąca pierwiastkiem naszego wielomianu, a więc nie jesteśmy w stanie określić dalszych rzędów zbieżności (stąd *nan* i *inf* w tabeli - próba dzielenia przez 0 - niezalecane!). Pozostałe metody (2. i 3.) mają empiryczny rząd zbieżności równy 1. Można było przewidzieć, że rząd zbieżności tych metod będzie gorszy od np. metody 4. już z wartości w tabeli 3, wtedy zaobserwowaliśmy, na ile szybko metoda 4. zbiegała do rzeczywistego miejsca zerowego naszego wielomianu.

Ostatnią rzeczą, którą porównamy, będą błędy względne użytych schematów. Poniżej przedstawiamy wykresy błędu względnego schematów iteracyjnych w zależności od numeru iteracji. Najpierw przedstawimy wykres wszystkich metod iteracyjnych, następnie skupimy się na metodach zbieżnych, tzn. wszystkich oprócz 1.



Rysunek 5: Wspólny wykres błędów względnych badanych schematów iteracyjnych w zależności od numeru iteracji



Rysunek 6: Wspólny wykres błędów względnych zbieżnych schematów iteracyjnych w zależności od numeru iteracji

Wyraźnie widać, że metoda 4 ma rząd zbieżności większy od liniowego. W przypadku metody 2. i 3., możemy zaobserwować różnice, których nie byliśmy w stanie znaleźć, badając same rzędy zbieżności. Metoda 3., choć posiada podobny rząd zbieżności do metody 2., w praktyce jest szybciej zbieżna. Na wykresie logarytmicznym wykresy obydwu metod wygladają w miarę liniowo, w porównaniu do parabolicznego wykresu metody 4., ale jednak widzimy, że im większa liczba iteracji, tym bardziej zwiększa się różnica pomiędzy metodą 2. i 3. (na korzyść 3.). Można było tego się spodziewać, porównując wykresy metody 2. i 3. na przedziale, na którym obserwujemy wyrazy naszych ciągów. Pochodna metody 3. na tym przedziale jest mniejsza od pochodnej metody 2., co powoduje szybszą zbieżność wyrazów.

4 Zadanie 3

4.1 Opis zadania

W zadaniu trzecim należy napisać schematy iteracji według metody Newtona dla trzech równań nieliniowych:

1. $x^3 - 2x - 5 = 0$,
2. $e^{-x} = x$,
3. $xsin(x) = 1$,

oraz określić liczbę iteracji potrzebną do osiągnięcia odpowiednio 24-bitowej i 53-bitowej dokładności rozwiązania, przy założeniu, że punkt x_0, z którego startujemy jest dokładny na poziomie 4 bitów.

4.2 Opracowanie zadania

Teoretyczna zbieżność metody Newtona jest kwadratowa, co objawia się tym, że liczba cyfr wyniku, które są dokładne powinna podwajać się w każdej iteracji. Jednakże, nauczeni doświadczeniami poprzednich zadań wiemy, że metoda Newtona jest zbieżna tylko przy spełnieniu odpowiednich warunków. Tak jak wspomniano w opracowaniu **Zadania 1**,

problem może wystąpić np. jeżeli nie jest spełniony któryś z warunków lub wartość pochodnej jest bardzo mała / zmienia znak itp. Zatem wyniki empiryczne mogą być zgoła odmienne od teoretycznych. Najpierw przekształcamy każde równanie do postaci funkcji f takiej, że szukamy x dla którego $f(x) = 0$ i wyliczamy pochodną uzyskanej funkcji, otrzymując:

Przekształcone równanie	Pochodna funkcji
$f_1(x) = x^3 - 2x - 5$	$f_1'(x) = 3x^2 - 2$
$f_2(x) = e^{-x} - x$	$f_2'(x) = -e^{-x} - 1$
$f_3(x) = x\sin x - 1$	$f_3'(x) = \sin x + x\cos x$

Tabela 5: Funkcje i pochodne potrzebne do metody iteracyjnej Newtona

W kodzie wygląda to bardzo podobnie (dosłowny zapis funkcji z powyższej tabeli), zatem skupimy się na wyborze punktu początkowego iteracji.
Uwaga! Przyjmujemy interpretację, że *przybliżenie pierwiastka z dokładnością 4 bitów* dotyczy części ułamkowej. Mamy nadzieję, że jest ona zgodna z intencją autora zadania.

Aby wyznaczyć wartości dokładne pierwiastków równań, korzystamy z funkcji `fsolve` z biblioteki `scipy`. Wartości początkowe, które jej podajemy (oszacowania pierwiastka) zostały odczytane z wykresów funkcji, które narysowaliśmy, korzystając ze strony *Geogebra Wykresy*: `https://www.geogebra.org/graphing`.
W kodzie wygląda to następująco:

```
x01_sol = fsolve(r1, 2.1)[0]  # 2.1  is estimated, initial guess
x02_sol = fsolve(r2, 0.55)[0] # 0.55 is estimated, initial guess
x03_sol = fsolve(r3, 1.1)[0]  # 1.1  is estimated, initial guess
```

Listing 3: Znajdowanie dokładnych pierwiastków równań

Tutaj $r1, r2, r3$ oznaczają kolejne równania, ri odpowiada f_i z tabeli powyżej.

Następnie, aby ustalić wartości początkowe iteracji korzystamy z funkcji pomocniczych, które najpierw konwertują liczbę zmiennoprzecinkową na zapis binarny (do zadanej precyzji, w naszym przypadku 4 pierwsze bity części ułamkowej). Następnie z powrotem dokonujemy konwersji na liczbę zmiennoprzecinkową [1]. Kod jest dość długi, więc tutaj go szerzej nie omawiamy - więcej szczegółów jest w pliku *.ipynb* z implementacją.
Poniżej przedstawiamy wartości dokładne (biorąc pod uwagę reprezentację komputerową) pierwiastków równań i wartości początkowe dla metody Newtona, przedstawione z dokładnością 4 bitów.

Nr równania	Dokładna wartość pierwiastka	Przybliżona wartość pierwiastka
1	2.0945514815423265	2.0625
2	0.5671432904097840	0.5625
3	1.1141571408719293	1.0625

Tabela 6: Porównanie rzeczywistych pierwiastków i wartości początkowych iteracji

Teraz pozostaje sprawdzić, po jakiej liczbie iteracji osiągniemy dokładność 24 bitów oraz 53 bitów. W tym celu po każdej iteracji wedle schematu Newtona sprawdzamy, na ilu bitach po przecinku zgadzają się rzeczywisty wynik oraz wynik uzyskany w obecnej iteracji. Ogólny schemat iteracji jest prosty:

```
prev_iter = curr_iter
curr_iter = prev_iter - fun(prev_iter) / der(prev_iter)
```

Listing 4: Pojedyncza iteracja w metodzie Newtona

[1] Bazujemy na kodzie ze strony: `https://www.geeksforgeeks.org/python-program-to-convert-floating-to-binary`.

Tutaj *curr_iter* oznacza obecną wartość, *prev_iter* - poprzednią, natomiast *fun* i *der* odpowiednio badaną funkcję i jej pochodną.

W tabeli poniżej przedstawiamy obliczoną liczbę iteracji, natomiast gdyby coś poszło niezgodnie z planem - zabezpieczamy się, ustawiając górny limit na 1000 iteracji, bowiem normalnie powinno ich wystarczyć dużo mniej.

Nr równania	Liczba iteracji dla 24 bitów	Liczba iteracji dla 53 bitów
1	3	4
2	2	1001
3	2	1001

Tabela 7: Liczba iteracji potrzebnych do uzyskania określonej dokładności

Widzimy, że dla 2. i 3. równania nie udało się osiągnąć dokładności na 53 bitach, zapewne któryś z warunków co do zbieżności metody Newtona nie był spełniony. Z kolei każda metoda bardzo szybko osiąga dokładność na 24 bitach. Aby próbować dowiedzieć się czegoś więcej, poniżej rozpisujemy początkowe 10 iteracji, skupiając się na liczbie dokładnych bitów.

Nr iteracji	Dokładność dla równania 1.	Dokładność dla równania 2.	Dokładność dla równania 3.
1	9	17	15
2	20	37	37
3	41	49	48
4	53	49	48
5	53	49	48
6	53	49	48
7	53	49	48
8	53	49	48
9	53	49	48
10	53	49	48

Tabela 8: Liczba bitów dokładności w 10 początkowych iteracjach

Można zauważyć, że w przypadku równania 1. uzyskujemy wynik zgodny z przewidywaniami teoretycznymi - w 1. iteracji mamy 9 dokładnych bitów (5 więcej niż na początku), w 2. iteracji 20 bitów (czyli 11 więcej niż w 1. iteracji - zyskujemy mniej więcej 2 razy więcej nowych dokładnych bitów), z kolei w 3. iteracji 41 dokładnych bitów, co oznacza zyskanie 21 nowych dokładnych bitów - około 2 razy więcej niż w 2 iteracji. W następnych iteracjach mamy dokładność 53 bitów, jest to spowodowane ograniczeniami reprezentacji liczby zmiennoprzecinkowej na komputerze - osiągnęliśmy maksymalną dokładność.

W przypadku pozostałych równań początkowe zachowanie jest podobne (a nawet można powiedzieć, że dynamika przybywania dokładnych bitów jest większa), ale już od 3. iteracji, kiedy uzyskano dokładność odpowiednio 49 bitów i 48 bitów, nie zyskujemy nowych bitów. Mimo że to jest oczywiście i tak bardzo dobra dokładność (rzędu 10^{-14}), to jednak w treści zadania dążymy do dokładności na 53 bitach, a takowej dla tych równań, korzystając z metody Newtona, nie uzyskujemy.

5 Zadanie 4

5.1 Opis zadania

W tym zadaniu celem jest napisanie schematu iteracji według metody Newtona dla układu równań nieliniowych:

$$x_1^2 + x_2^2 = 1$$
$$x_1^2 - x_2 = 0$$

Wiemy, że rozwiązaniem równania jest

$$x_1 = \pm\sqrt{\frac{\sqrt{5}}{2} - \frac{1}{2}}$$
$$x_2 = \frac{\sqrt{5}}{2} - \frac{1}{2}$$

Po wyznaczeniu rozwiązania tego układu, korzystając z dokładnych rozwiązań, obliczymy błąd względny wyników.

5.2 Opracowanie zadania

W tym przypadku korzystamy z metody Newtona dla funkcji wielu (dwóch) zmiennych. Sprowadzamy układ równań do funkcji F dwóch zmiennych, przekształcając go do takiej postaci, żeby rozwiązanie układu było równoważne ze znalezieniem miejsca zerowego tej funkcji: $F(\mathbf{x}) = \mathbf{0}$, przy czym $\mathbf{x}$ to kolumna dwóch zmiennych x_1, x_2, podobnie jak $\mathbf{0}$ w tym przypadku to transponowany wektor zerowy długości 2.

Uzyskana funkcja F to $F(x_1, x_2) = (x_1^2 + x_2^2 - 1, x_1^2 - x_2)$. W metodzie Newtona potrzebujemy także pochodnej funkcji. W przypadku funkcji wielu zmiennych pochodna jest reprezentowana przez tzw. *macierz Jacobiego*. Zatem wzór iteracyjny w metodzie Newtona przyjmuje nieco inną postać:

$$x_{k+1} = x_k - (F'(x_k))^{-1} F(x_k), \tag{3}$$

gdzie $(F'(x_k))^{-1}$ oznacza macierz odwrotną do macierzy Jacobiego w danym punkcie $\mathbf{x}$. Macierz Jacobiego dla punktu $\mathbf{x} = [x_1, x_2]^T$ prezentuje się następująco:

$$J(\mathbf{x}) = \begin{bmatrix} 2x_1 & 2x_2 \\ 2x_1 & -1 \end{bmatrix}.$$

Teraz możemy przejść do głównej części rozwiązania. Wybieramy arbitralnie początkowy punkt iteracji, np. $\mathbf{x}_0 = [1, 1]^T$ (zwykle wybiera się $[0, 0]^T$, ale w naszym przypadku dla tego punktu macierz Jacobiego byłaby osobliwa i nie dałoby się policzyć macierzy odwrotnej). Ustalamy limit 10 iteracji i przechodzimy do obliczeń.
W kodzie wygląda to następująco:

```
x0 = np.array([1., 1.])
prev_iter = x0
curr_iter = prev_iter - np.linalg.inv(F_prim(prev_iter)) @ F(prev_iter)
no_iter = 1

while no_iter < 10:
        prev_iter = curr_iter
        curr_iter = prev_iter - np.linalg.inv(F_prim(prev_iter)) @ F(prev_iter)
        no_iter += 1
```

Listing 5: Schemat iteracji w metodzie Newtona dla układu równań

Tutaj `F` i `F_prim` oznaczają odpowiednio funkcję F oraz jej pochodną.

Zobaczmy teraz, jak dokładne są uzyskane wyniki (oczywiście w ramach reprezentacji liczby zmiennoprzecinkowej na komputerze).
Popatrzmy na poniższą tabelę.

Pierwiastek	Rzeczywista wartość pierwiastka	Obliczona wartość pierwiastka
x_1	$\pm$0.7861513777574233	0.7861513777574233
x_2	0.6180339887498949	0.6180339887498948

Tabela 9: Porównanie rzeczywistych pierwiastków i obliczonych

Jak widać, obliczone wartości są prawie identyczne (dla pierwiastka x_2 różnica jest jedynie na 16. cyfrze po przecinku). W przypadku pierwiastka x_1 algorytm znalazł pierwiastek dodatni, bowiem obie podane wartości są prawidłowym rozwiązaniem układu. Przedstawiamy jeszcze błędy: bezwzględny i względny rozwiązań.

Błąd	Pierwiastek x_1	Pierwiastek x_2
bezwzględny	0	$1.110223 \cdot 10^{-16}$
względny	0	$1.796379 \cdot 10^{-16}$

Tabela 10: Przedstawienie błędów bezwzględnych i względnych rozwiązań

Jak widać, jeżeli warunki zbieżności są spełnione, metoda Newtona pozwala dość szybko obliczyć wartości rozwiązań z dużą dokładnością - przyjęliśmy zaledwie 10 iteracji. Tymczasem obliczone pierwiastki są prawie równe rzeczywistym, a punkt początkowy nawet nie był szczególnie blisko rozwiązania.

6 Podsumowanie i wnioski

W ramach tego laboratorium przedstawiliśmy wyniki eksperymentów dotyczących stosowania metod iteracyjnych dla różnych równań nieliniowych. Skupiliśmy się głównie na opisanych i znanych metodzie Newtona (zadania 1., 3. i 4.), metodzie bisekcji (zadanie 1.), natomiast w ramach zadania 2. przetestowaliśmy różne, ale równoważne schematy iteracyjne dla podanego równania.

Zrealizowanie tych zadań pozwoliło przedstawić i doświadczyć tego, że mimo opisywania tego samego problemu, niektóre próby prowadzą do bardzo dokładnego rozwiązania dość szybko, inne są wolniejsze, a jeszcze inne okazują się podejściem błędnym, nieprowadzącym do prawidłowego rozwiązania. Tak było właśnie w zadaniu 2., gdzie jeden ze schematów iteracyjnych, mimo opisywania tego samego równania, okazał się rozbieżny.

Z kolei jeżeli chodzi o metodę iteracyjną Newtona, mieliśmy okazję się przekonać, że przy braku spełnienia odpowiednich warunków, metoda ta może się okazać rozbieżna. W zadaniu 1., gdzie rozważaliśmy cztery równania, wybór punktu początkowego oraz chrakterystyka tych równań wpływała na to, że metoda Newtona nie znajdowała poprawnie pierwiastków równania. Wtedy użyliśmy metody bisekcji, która przy spełnieniu o wiele bardziej liberalnych warunków zawsze powinna znaleźć przybliżony pierwiastek, natomiast jest to metoda o większej złożoności - stąd często te metody współpracują ze sobą i stosujemy metodę Newtona, a gdy sobie ona nie radzi - np. metodę bisekcji.

W zadaniu 3 skupialiśmy się głównie na dokładności metody Newtona i na tym, ile iteracji jest potrzebne, żeby osiągnąć dokładność na zadanej liczbie bitów. Jak się okazało, metoda Newtona pozwala nam uzyskać bardzo dużą dokładność, ale nie zawsze osiąga ona maksymalną wartość (dla *float64* na 53 bitach), bowiem możemy dojść do pewnej granicy tej dokładności, czego doświadczyliśmy w dwóch rozważanych przypadkach.

Finalnie, metoda Newtona może być także użyta do rozwiązania układów równań nieliniowych, wtedy taki układ należy odpowiednio przekształcić na funkcję wielu zmiennych i problem sprowadza się do znalezienia miejsca zerowego takiej funkcji.

Oczywiście istnieje wiele innych metod iteracyjnych, ale metoda Newtona i metoda bisekcji to jedne z popularniejszych i łatwiejszych w implementacji, dlatego warto się z nimi zaznajomić.

Literatura

[1] Materiały pomocnicze do laboratorium zamieszczone na platformie Teams w katalogu *lab08/lab8-intro.pdf*.

[2] Treść przedstawiona na wykładzie dotycząca równań nieliniowych.

[3] Metoda Newtona opisana na Wikipedii - głównie przypadek wielowymiarowy: `https://pl.wikipedia.org/wiki/Metoda_Newtona`