Rozwiązywanie równań nieliniowych

Sprawozdanie z laboratorium 8

Jakub Grześ

10.05.2024

1 Treść zadań

Zadanie 1

Dla poniższych funkcji i punktów początkowych metoda Newtona zawodzi. Wyjaśnij dlaczego. Następnie znajdź pierwiastki, modyfikując wywołanie funkcji `scipy.optimize.newton` lub używając innej metody.

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

Zadanie 2

Dane jest równanie:

$$f(x) = x^2 - 3x + 2 = 0$$

Każda z następujących funkcji definiuje równoważny schemat iteracyjny:

$$g_1(x) = \frac{x^2 + 2}{3},$$
$$g_2(x) = \sqrt{3x - 2},$$
$$g_3(x) = 3 - \frac{2}{x},$$
$$g_4(x) = \frac{x^2 - 2}{2x - 3}.$$

(a) Przeanalizuj zbieżność oraz rząd zbieżności schematów iteracyjnych odpowiadających funkcjom $g_i(x)$ dla pierwiastka $x = 2$ badając wartość $|g_i'(2)|$.

(b) Potwierdź analizę teoretyczną implementując powyższe schematy iteracyjne i weryfikując ich zbieżność (lub brak). Każdy schemat iteracyjny wykonaj przez 10 iteracji.

Wyznacz eksperymentalnie rząd zbieżności każdej metody iteracyjnej ze wzoru

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

gdzie błąd bezwzględny ε_k definiujemy jako $\varepsilon_k = |x_k - x^*|$, x_k jest przybliżeniem pierwiastka w k-tej iteracji, a x^* dokładnym położeniem pierwiastka równania.

Zadanie 3

Napisz schemat iteracji wg metody Newtona dla każdego z następujących równań nieliniowych:

(a) $x^3 - 2x - 5 = 0$

(b) $e^{-x} = x$

(c) $x \sin(x) = 1$.

Jeśli x_0 jest przybliżeniem pierwiastka z dokładnością 4 bitów, ile iteracji należy wykonać aby osiągnąć:

- 24-bitową dokładność
- 53-bitową dokładność?

Zadanie 4

Napisz schemat iteracji wg metody Newtona dla następującego układu równań nieliniowych:

$$x_1^2 + x_2^2 = 1, \quad x_1^2 - x_2 = 0.$$

Korzystając z faktu, że dokładne rozwiązanie powyższego układu równań to:

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

oblicz błąd względny rozwiązania znalezionego metodą Newtona.

2 Rozwiązania zadań

2.1 Zadanie 1

Metoda Newtona, znana również jako metoda stycznych, jest techniką iteracyjną służącą do znajdowania pierwiastków równań nieliniowych. Rozpoczyna się od początkowego przybliżenia x_0, a następnie używa się poniższej formuły iteracyjnej do generowania kolejnych przybliżeń pierwiastka:

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

2.1.1 Podpunkt a

Metoda Newtona dla tak dobranego punktu początkowego wchodzi w nieskończony cykl:

$$\begin{aligned}
x_0 = 1: &\quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -1 \\
x_1 = -1: &\quad x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1 \\
x_2 = 1: &\quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -1 \\
&\vdots
\end{aligned}$$

Nie może więc ona zbiec się do pierwiastka niezależnie od liczby iteracji. Metoda zawodzi dla tak dobranego punktu początkowego.

2.1.2 Podpunkt b

Dla tej funkcji jej pierwsza pochodna w punkcie 2 osiąga wartość 0. Zgodnie ze wzorem metody Newtona prowadzi to do zabronionego dzielenia przez 0. Trzeba dobrać punkt początkowy tak, aby $f'(x) \neq 0$.

2.1.3 Podpunkt c

Dla tej funkcji również metoda zawodzi przez wartość pochodnej w punkcie startowym. $f'(x_0) = -10^{-8}$ jest bardzo niewielką wartością, co zapewnia zbieżność, ale bardzo powolną. Potrzebne jest bardzo wiele iteracji lub wybranie innego punktu początkowego.

2.1.4 Podpunkt d

Ta funkcja, dla tego punktu startowego podobnie jak podpunkt *a* generuje nieskończony cykl. Metoda ponownie zawodzi dla tych warunków.

2.1.5 Znalezienie pierwiastków

Przyjęto nowe wartości punktów początkowych, które zagwarantowały odpowiednią wartość pochodnej i uniknęły cykli.

(a) $f(x) = x^3 - 5x$, $x_0 = -0.1$

(b) $f(x) = x^3 - 3x + 1$, $x_0 = 0$

(c) $f(x) = 2 - x^5$, $x_0 = 0.9$

(d) $f(x) = x^4 - 4.29x^2 - 5.29$, $x_0 = 1.7$

Do obliczeń wykorzystano funkcję *optimize.newton* z modułu *Scipy*. Znajduje ona pierwiastek metodą Newtona-Raphsona. Jako argumenty funkcji podano funkcje z treści oraz nowe punkty początkowe. Porównano otrzymane wyniki z wyznaczonymi prawdziwymi pierwiastkami równania otrzymanymi z użyciem oprogramowania *Wolphram Alpha*.

Tabela 1: Porównanie znalezionych pierwiastków z prawdziwymi dla zadania 1

Funkcja	**Znaleziony pierwiastek**	**Prawdziwy pierwiastek**
f1	-3.4×10^{-27}	$x_0 = 0$
f2	1.5321	$x_0 = 1.5321$
f3	1.1487	$x_0 = \sqrt[5]{2} \approx 1.1487$
f4	2.3	$x_0 = 2.3$

Dzięki odpowiedniemu dobraniu punktów startowych udało się tanim kosztem obliczeniowym otrzymać precyzyjne wyniki. Warto zwrócić uwagę na to, że np. funkcja f_1 ma więcej pierwiastków, które nie zostaną wykryte przy pojedynczym zastosowaniu metody Newtona-Raphsona.

2.2 Zadanie 2

2.2.1 Podpunkt a

Zbieżność ustalimy przez badanie wartości pierwszej pochodnej.

- Jeśli $x^* = g(x^*)$ i $|g'(x^*)| \leq 1$, wówczas istnieje przedział zawierający x^* taki, że iteracja
$$x_{k+1} = g(x_k)$$
zbiega do x^*, jeśli zacznie się w tym przedziale.
- Jeśli $|g'(x^*)| > 1$, wówczas schemat iteracyjny rozbiega.
- Jeśli $g'(x^*) = 0$, wówczas wskaźnik zbieżności jest co najmniej kwadratowy.

Pierwszą pochodną obliczono użyciem funkcji *diff* z biblioteki *Sympy*. Następnie obliczono jej wartość dla argumentu 2 i porównano z powyższymi twierdzeniami.

Tabela 2: Zbieżność funkcji z zadania 2

Funkcja	**Zbieżność**
g1	Nie
g2	Tak
g3	Tak
g4	Tak

Dla g4 pochodna wynosi 0, co oznacza co najmniej kwadratową zbieżność.

2.2.2 Podpunkt b

Przygotowano dwie funkcje, *iterate* implementująca schemat iteracyjny oraz *calculate_order_of_convergence* obliczająca rząd zbieżności ze wzoru (6).

```
def iterate(x0, function, n = 10):
    for i in range(n):
        x0 = function(x0)
    return x0

def calculate_order_of_convergence(x0, function, exact_solution=2, n=10):
    errors = [abs(x0 - exact_solution)]
    orders = []
    for i in range(n):
        x0 = function(x0)
        error = abs(x0 - exact_solution)
        error = error
        errors.append(float(error))

    for i in range(1, len(errors) - 1):
        if errors[i] != 0 and errors[i + 1] != 0 and errors[i - 1] != 0:
            order = np.log(errors[i] / errors[i + 1])
            / np.log(errors[i - 1] / errors[i])
            orders.append(order)

    return np.mean(orders), errors
```

Za punkt startowy przyjęto $x_0 = 1.75$.

Schemat iteracyjny wykonano dla każdej funkcji 10 razy. Wyniki przedstawiono w tabeli.

Tabela 3: Obliczony pierwiastek po 10 iteracjach dla g_i

Funkcja	**Wynik**
g1	1.08
g2	1.98
g3	2.00
g4	2.00

Prawdziwym pierwiastkiem równania jest $x_0 = 2$. Zgodnie z oczekiwaniami, funkcje, które dawały zbieżny schemat iteracyjny, pozwoliły na osiągnięcie wyniku bliskiego temu wyznaczonemu analitycznie. Wynik dla g_1 jest daleki od prawidłowego.

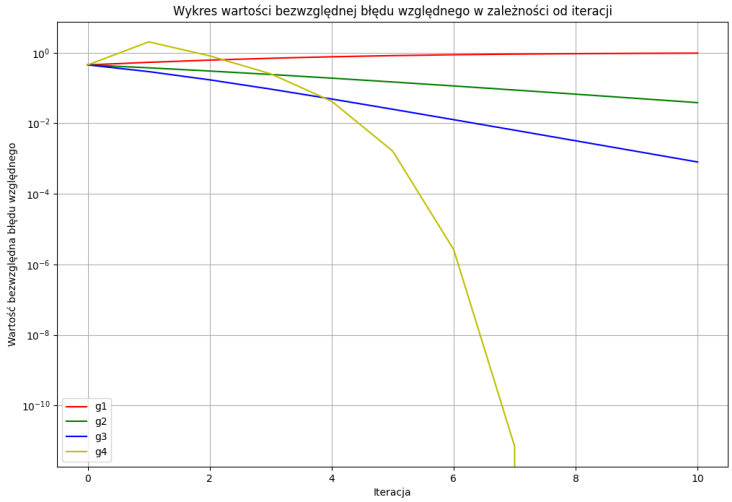
Następnie wyznaczono empirycznie rzędy zbieżności dla funkcji g_i. Wyniki przedstawiono w tabeli 4.

Tabela 4: Empiryczny rząd zbieżności g_i

Funkcja	**Rząd zbieżności**
g1	0.83
g2	1.02
g3	1.02
g4	2.29

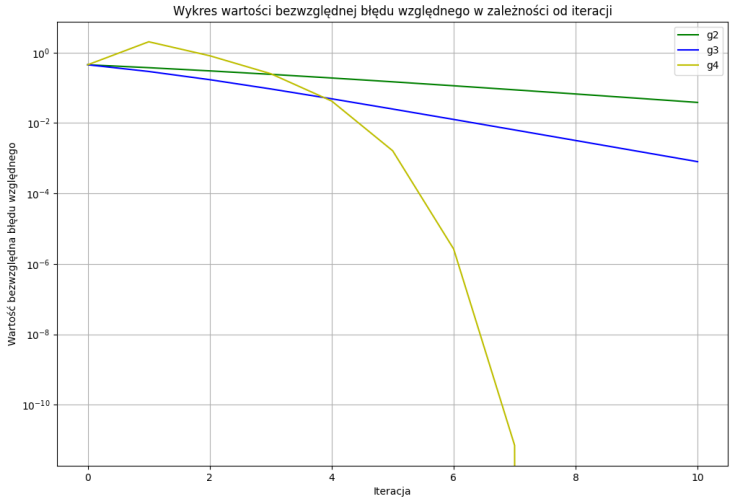
Funkcja g_1 ma rząd zbieżności < 1 zgodny z oczekiwanym brakiem zbieżności. Pozostałe funkcje są zbieżne, przy czym g_4 daje najwyższy rząd zbieżności, spełniający warunek ≥ 2.

Na poniższym wykresie przedstawiono wartość bezwzględną błędu względnego w zależności od iteracji dla każdej funkcji.



Wykres 1: Porównanie błędów względnych w zależności od iteracji

Funkcja g_1 nie zmniejsza systematycznie błędu przybliżenia, co wynika z jej rozbieżności. Pozostałe funkcje zmniejszają błąd w kolejnych iteracjach. Funkcja g_4 najszybciej zyskuje na dokładności. Poniżej ten sam wykres, bez prowadzącej do rozbieżności funkcji g_1.



Wykres 2: Porównanie błędów względnych w zależności od iteracji (tylko zbieżne)

2.3 Zadanie 3

2.3.1 Rozwiązanie równań

Zdefiniowano każdą z funkcji oraz jej pochodną. Następnie stworzono funkcję wykonującą metodę Newtona.

```
def newtons_method(f, f_derivative, x0, n = 10):
    for i in range(n):
        x0 = x0 - f(x0) / f_derivative(x0)
    return x0
```

Wykonano ją dla każdego z równań. Wyniki przedstawiono w tabeli. Prawdziwy wynik otrzymano, korzystając z oprogramowania *Wolphram Alpha*

Tabela 5: Rozwiązania równań nieliniowych

Równanie	**Otrzymany wynik**	**Prawdziwy wynik**
1	2.0946	≈ 2.0946
2	0.5671	≈ 0.5671
3	-9.3172	≈ -9.3172

Dla każdego z równań, zaledwie 10 iteracji wystarczyło do uzyskania wysokiej precyzji wyników.

2.3.2 Precyzja bitowa w metodzie Newtona-Raphsona - obliczenie teoretyczne

Metoda Newtona-Raphsona cechuje się drugim rzędem zbieżności, co oznacza, że błąd iteracyjny maleje kwadratowo z każdą kolejną iteracją. To znaczy

$$e_i \approx C(e_{i-1})^2 \Rightarrow e_n \approx C(e_0)^{2^n}$$

gdzie e_i jest błędem w i-tej iteracji, a C jest pewną stałą zależną od szczegółów funkcji i jej pochodnych.

Dokładność p-bitowa oznacza, że żądany błąd e_i jest mniejszy niż 2^{-p}. Można określić więc ilość iteracji n potrzebnych do osiągnięcia p-bitowej dokładności jako najmniejszą liczbę naturalną spełniającą równanie

$$2^{-p} > C(e_0)^{2^n}$$

Jeśli znamy przybliżenie pierwiastka z dokładnością do 4-bitów t.j. jego błąd $e_0 < 2^{-4}$ to

$$\begin{gathered} 2^{-p} \approx (2^{-4})^{2^n} = 2^{-4 \cdot 2^n} \\ -p \approx -4 \cdot 2^n \\ p \approx 4 \cdot 2^n \\ 2^n \approx \frac{p}{4} \\ n \approx \log_2\left(\frac{p}{4}\right) \end{gathered}$$

- **Dla 24-bitowej dokładności:**

$$n \approx \log_2\left(\frac{24}{4}\right) = \log_2(6) \approx 2.58$$

Czyli potrzebne będą $\lceil 2.58 \rceil = 3$ iteracje.

- **Dla 53-bitowej dokładności:**

$$n \approx \log_2\left(\frac{53}{4}\right) = \log_2(13.25) \approx 3.73$$

Czyli potrzebne będą $\lceil 3.73 \rceil = 4$ iteracje.

Widać tutaj, że niewielka różnica w liczbie iteracji, może istotnie wpłynąć na precyzję wyniku.

2.4 Zadanie 4

- W n wymiarach, metoda Newtona ma postać:

$$x_{k+1} = x_k - J(x_k)^{-1} F(x_k)$$

gdzie $J(x)$ to macierz Jacobiego funkcji f, z elementami:

$$\{J(x)\}_{ij} = \frac{\partial f_i(x)}{\partial x_j}$$

- W praktyce, zamiast wyliczać odwrotność macierzy $J(x_k)$, rozwiązujemy układ liniowy:

$$J(x_k)s_k = -F(x_k)$$

gdzie krok Newtona s_k, a następnie przyjmujemy jako kolejną iterację:

$$x_{k+1} = x_k + s_k$$

Rozpatrujemy układ równań:

$$\begin{cases} x_1^2 + x_2^2 = 1 \\ x_1^2 - x_2 = 0 \end{cases}$$

Definiujemy wektor funkcji $\mathbf{F}(\mathbf{x})$ jako:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} x_1^2 + x_2^2 - 1 \\ x_1^2 - x_2 \end{bmatrix}$$

Macierz Jacobiego $J(\mathbf{x})$ wynosi:

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

Zaimplementowano funkcję *newton_ multidimension*, która przyjmuje macierz Jacobiego oraz wektor funkcji jako argumenty. Jako punkt startowy użyto wektora $\mathbf{x}_0 = [1, 1]$. Funkcja wykonuje 10 iteracji metody Newtona dla układu równań nieliniowych. Odpowiednie operacje na macierzach są wykonwyane za pomocą funkcji biblioteki *Numpy*.

```
def newton_multidimension(x0, F, J, n=10):
    for i in range(n):
        J_curr = J(x0)
        F_curr = F(x0)
        x0 = x0 - np.linalg.inv(J_curr).dot(F_curr)
    return x0
```

Wyniki przedstawiono poniżej.

Tabela 6: Porównanie wyników metody Newtona z wartościami dokładnymi

Zmienna	**Wynik metody Newtona**	**Błąd względny**
x_1	0.78615138	0.0
x_2	0.61803399	1.796×10^{-16}

Jak widać, metoda skutecznie rozwiązała układ równań znajdując precyzyjne rozwiązanie już po 10 iteracjach.

3 Wnioski

- **Znaczenie wyboru punktu startowego:** Metoda Newtona jest bardzo efektywna, ale jej skuteczność jest silnie zależna od właściwego wyboru punktu startowego. Powinien on być jak najbliżej szukanego pierwiastka. Istotne jest także unikanie punktów, w których pochodna funkcji jest bliska zero, co może prowadzić do braku zbieżności lub jej znacznego spowolnienia. Istnieje również ryzyko wpadnięcia w nieskończone cykle, co uniemożliwi zbieżność do oczekiwanego rozwiązania.
- **Analiza zbieżności schematów iteracyjnych:** Zbieżność schematów iteracyjnych nie jest gwarantowana dla wszystkich funkcji. Należy dokładnie analizować właściwości funkcji oraz pochodnych, aby ocenić, czy dany schemat iteracyjny zapewni skuteczną zbieżność i jaki będzie jej rząd.
- **Zbieżność metody Newtona:** Metoda Newtona charakteryzuje się kwadratową zbieżnością, co oznacza, że błędy maleją kwadratowo z każdą kolejną iteracją. Nawet niewielkie różnice w liczbie wykonanych iteracji mogą znacząco wpływać na precyzję uzyskanych wyników.
- **Uogólnienie do postaci wielowymiarowej:** Metoda Newtona może być z powodzeniem uogólniona na przypadki wielowymiarowe, co pozwala na stosowanie jej do znajdowania pierwiastków układów równań z wieloma zmiennymi.

Bibliografia

[1] dr inż. Marcin Kuta, *Solving nonlinear equations*

[2] dr inż. Marian Bubak, dr inż. Katarzyna Rycerz, *Wykład 7 - Równania nieliniowe*

[3] prof. Michael T. Heath *Chapter 5: Nonlinear Equations*