Laboratorium 7

Pierwiastki równań nieliniowych

Krzysztof Solecki

22 Kwietnia 2023

1 Treści zadań

1. Napisz iteracje wg metody Newtona do rozwiązywania każdego z następujących równań nieliniowych:
   (a) x cos(x) =1
   (b) $x^3 - 5x - 6 = 0$
   (c) $e^{-x} = x^2 - 1$
2. (a) Pokaż, że iteracyjna metoda: $x_{k+1} = \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$ matematycznie jest równoważna z metodą siecznych przy rozwiązywaniu skalarnego nieliniowego równania f (x) = 0.
   (b) Jeśli zrealizujemy obliczenia w arytmetyce zmiennoprzecinkowej o skończonej precyzji, jakie zalety i wady ma wzór podany w podpunkcie (a), w porównaniu ze wzorem dla metody siecznych podanym poniżej? $x_{k+1} = \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$
3. Zapisz iteracje Newtona do rozwiązywania następującego układu równań nieliniowych.
   (a) $x_1^2 + x_1 x_2^3 = 9$
   (b) $3x_1^2 x_2 - x_2^3 = 4$

2 Rozwiązania zadań

1. Wyniki zostały obliczone za pomocą programu napisanego w języku Python:

```
from math import e, cos, sin


# funkcja z pp. a
def f_a(x):
    return (1 + x ** 2 * sin(x)) / (cos(x) - x * sin(x))


# funkcja z pp. b
def f_b(x):
    return (2 * x ** 3 + 6) / (3 * x ** 2 - 5)


# funkcja z pp. c
def f_c(x):
    return (e ** (-x) + 1) * (-x - 1) / (-e ** (-x) - 2 * x)


def newton_iteration(x0: float, n: int, f):
    x_k = x0
    for k in range(1, n + 1):
        x_k = f(x_k)
        print("k: ", k, "x_k: ", x_k)


print("-------- xcos(x)=1 ---------")
newton_iteration(1, 20, f_b)
print("-------- x^3-5x-6=0 ---------")
newton_iteration(1, 20, f_a)
print("-------- e^(-x)=x^2-1 ---------")
newton_iteration(1, 20, f_c)
```

(a) Mamy:

$$x_{k+1} = x_k - \frac{x_k cos(x_k)-1}{cos(x_k)-x_k sin(x_k)} = \frac{1+x_k^2 sin(x_k)}{cos(x_k)-x_k sin(x_k)}$$

Niech $x_0 = 1$ (funkcja oraz jej dwie pochodne są ciągłe, x_0 jest blisko rozwiązania oraz $f'(x_0) \neq 0$. Wyniki przedstawia tabelka:

k	x_k
1	-6.114417313548893
2	3.6166531913233118
3	-6.513734068609697
4	16.886893233297762
5	-17.244815735599488
6	17.336975975684517
7	-17.221304526132844
8	17.337374152314872
9	-17.221307141408733
10	17.33737409942598

(b) Mamy:

$$x_{k+1} = x_k - \frac{x_k^3-5x_k-6}{3x_k^2-5} = \frac{2x_k^3+6}{3x_k^2-5}$$

Niech $x_0 = 1$ (funkcja oraz jej dwie pochodne są ciągłe, x_0 jest blisko rozwiązania oraz $f'(x_0) \neq 0$. Wyniki przedstawia tabelka:

k	x_k
1	-4.0
2	-2.8372093023255816
3	-2.0720209707579835
4	-1.4964181413556084
5	-0.40852227514846784
6	-1.303226219125557
7	16.52600182653632
8	11.09234954141944
9	7.512922382297259
10	5.1975195357565065
11	3.771749728508793
12	3.0074229336011444
13	2.728943551609753
14	2.689841309646591
15	2.689095592421677

(c) Mamy:

$$x_{k+1} = x_k - \frac{e^{-x_k} - x_k^2 + 1}{-e^{-x_k} - 2x_k} = \frac{(e^{-x_k} + 1)(-x_k - 1)}{-e^{-x_k} - 2x_k}$$

Niech $x_0 = 1$ (funkcja oraz jej dwie pochodne są ciągłe, x_0 jest blisko rozwiązania oraz $f'(x_0) \neq 0$. Wyniki przedstawia tabelka:

k	x_k
1	1.1553624034969636
2	1.079412675839093
3	1.1150100253233204
4	1.0979756805310898
5	1.1060484690522463
6	1.1022048537258669
7	1.1040308532102534
8	1.103162459719219
9	1.1035752374605103
10	1.1033789833288639

Wniosek: Wartość wybranego przez nas x_0 (o ile nie jest w pobliżu ekstremum) nie ma zbytniego znaczenia, ponieważ już po kilku iteracjach kolejne wartości są bardzo zbliżone.

2. a) Pokażemy, że poniższy wzór jest równoważny z metodą siecznych dla rozwiązywania równań nieliniowych f(x)=0:

$$x_{k+1} = \frac{x_{k-1} \cdot f(x_k) - x_k \cdot f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Przekształćmy najpierw wzór na metodę siecznych stosując proste algebraiczne przekształcenia:

$$x_{k+1} = x_k - \frac{x_k(f(x_{k-1}) - f(x_k))}{f(x_{k-1}) - f(x_k)} = \frac{x_k \cdot f(x_{k-1}) - x_k \cdot f(x_k) - x_k \cdot f(x_{k-1}) + x_k \cdot f(x_k)}{f(x_{k-1}) - f(x_k)} = \frac{x_k \cdot f(x_{k-1}) - x_{k-1} \cdot f(x_k)}{f(x_{k-1}) - f(x_k)} = \frac{x_{k-1} f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Otrzymaliśmy właśnie wzór podany wyżej, zatem udowodniliśmy równoważność obu wzorów za pomocą kilku prostych przekształceń algebraicznych.

b)

1. **Metoda Siecznych:**
   (a) Wady:
      i. Możliwość utraty precyzji dla małych różnic między x_{k-1} oraz x_k
   (b) Zalety:
      i. Nie wymaga obliczenia pochodnej funkcji.

2. **Metoda Newtona:**

   (a) Wady:

      i. Konieczność obliczenia pochodnej funkcji, co może być problematyczne dla skomplikowanych funkcji oraz czasochłonne.

   (b) Zalety:

      i. Może być stabilniejsza, gdy funkcja jest dobrze uwarunkowana wokół pierwiastka i wymagana jest duża precyzja.

W przypadku obu metod w artymetyce zmiennoprzecinkowej o ograniczonej precyzji istnieje podatność na błędy numeryczne szczególnie dla funkcji z oszarami o dużych wartościach lub pochodnej bliskiej 0.

3. Zapiszmy zadany układ jako:

$$\begin{cases} x_1^2 + x_1 x_2^3 - 9 = 0 \\ 3x_1^2 x_2 - x_2^3 - 4 = 0 \end{cases} . \tag{1}$$

Poniższy program w języku Python został napisany bazując na prezentacji. Na jego podstawie udało mi się wyliczyć x_1 oraz x_2 dla 10 iteracji.

```
def f1(x, y):
    return x**2+x*y**3-9


def f2(x, y):
    return 3*x**2*y-y**3-4


def f1x(x, y):
    return 2*x


def f1y(x, y):
    return 3*x*y**2


def f2x(x ,y):
    return 6*x*y


def f2y(x, y):
    return 3*x**2-3*y**2


def jacobian(x, y):
    return f1x(x, y)*f2y(x, y)-f1y(x, y)*f2x(x, y)


def newton_iteration(x0 ,y0, n):
    x = x0
```

```
    y = y0
    for k in range(1, n + 1):
        delta_x = (-f1(x, y)*f2y(x, y)+f2(x, y)*f1y(x, y))/jacobian(x, y)
        delta_y = (-f2(x, y)*f1x(x, y)+f1(x, y)*f2x(x, y))/jacobian(x, y)
        x = x + delta_x
        y = y + delta_y
        print("k: ", k, "x: ", x, "y: ",y)

newton_iteration(1,1,10)
```

Wyniki przedstawiłem w tabelce:

k	x_k	y_k
1	1.3333333333333333	3.111111111111111
2	1.2337931296549627	2.267473526878452
3	1.3087126323966336	1.8948717506437391
4	1.3351703679582776	1.775274872483528
5	1.336392549829994	1.7549825650430313
6	1.3363510477450014	1.7542202019069097
7	1.3363558696612006	1.754236985570715
8	1.3363553212049262	1.7542349943110938
9	1.336355383588176	1.7542352207806415
10	1.3363553764925082	1.7542351950209438

Wnioski: Widzimy, że już po kilku pierwszych iteracjach wyniki są do siebie bardzo zbliżone. Metoda ta, choć nie najprostsza w zrozumieniu, jest bardzo prosta w implementacji i daje naprawdę dobre rezultaty.

3 Bibliografia

1. Katarzyna Rycerz: Materiały wykładowe z przedmiotu Metody Obliczeniowe w Nauce i Technice
2. https://www.wolframalpha.com/
3. https://www.desmos.com/calculator?lang=pl
4. https://home.agh.edu.pl/ funika/mownit/lab7/RF.pdf