Kacper Korta,

25.04.2023

Laboratorium 07

Poszukiwanie pierwiastków równań nieliniowych

1. Treść 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?

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

3. Zapisz iteracje Newtona do rozwiązywania następującego układu równań nieliniowych.

$X1^2$ + x1$x2^3$= 9

3$x1^2$ x2 - $x2^3$= 4

2. Rozwiązania

Zadanie 1.

a) korzystając z równania xcos(x)-1=0:

$$x_{k+1} = x_k - \frac{xcosx - 1}{cosx - xsinx}$$

dla x0 = 4 otrzymujemy:

```
i:  1 x:  5.522845345218782
i:  2 x:  4.860294809581786
i:  3 x:  4.917565305315841
i:  4 x:  4.9171859390089105
i:  5 x:  4.917185925287132
i:  6 x:  4.917185925287132
i:  7 x:  4.917185925287132
i:  8 x:  4.917185925287132
i:  9 x:  4.917185925287132
i:  10 x:  4.917185925287132
```

b) Korzystając z równania

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

Dla x0 = 4

Otrzymujemy tablę z wynikami

```
C:/Users/Kacper/ProgramowaniePython/
i:  1 x:  3.116279069767442
i:  2 x:  2.7565563488220954
i:  3 x:  2.6911929055245243
i:  4 x:  2.689097446679465
i:  5 x:  2.6890953236398376
i:  6 x:  2.6890953236376594
i:  7 x:  2.6890953236376594
i:  8 x:  2.6890953236376594
i:  9 x:  2.6890953236376594
i:  10 x:  2.6890953236376594
```

c) Korzystając z równania

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

Dla x0 = 2

Otrzymujemy tablę z wynikami

```
C:/Users/Kacper/Programowanie Python/Mownit/_zad
i:  1 x:  1.3072714736394255
i:  2 x:  1.1553178500127186
i:  3 x:  1.1477759644499919
i:  4 x:  1.1477576322529561
i:  5 x:  1.1477576321447436
i:  6 x:  1.1477576321447436
i:  7 x:  1.1477576321447436
i:  8 x:  1.1477576321447436
i:  9 x:  1.1477576321447436
i:  10 x:  1.1477576321447436
```

Wnioski: Metoda Newtona jest efektywnym narzędziem do rozwiązywania równań nieliniowych. Dobór początkowych przybliżeń x0 ma duże znaczenie dla skuteczności metody Newtona.

Zadanie 2.

a) Rozważając wzór podany w zadaniu:

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

Do licznika dodajemy sztuczne zero:

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

W kolejnym kroku grupujemy wyrazy:

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

Po uproszczeniu otrzymujemy:

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

Zauważmy że otrzymany wzór jest taki sam jak ten podany w podpunkcie b).
Zatem oznacza to, że metody są sobie równoważne przy rozwiązywaniu skalarnego nieliniowego równania f (x) = 0.

b) Metoda Newtona i metoda siecznych to dwie techniki numeryczne używane do znajdowania zer funkcji. Porównując obie metody, można stwierdzić, że wadą metody Newtona jest to, że wymaga ona znajomości pochodnej funkcji, co w przypadku bardziej skomplikowanych funkcji może być trudne lub czasochłonne do obliczenia. Metoda siecznych natomiast nie wymaga znajomości pochodnej, co oznacza, że jest bardziej uniwersalna i może być stosowana do bardziej skomplikowanych funkcji. Jednak metoda siecznych jest bardziej podatna na utratę cyfr znaczących, szczególnie gdy różnica między kolejnymi punktami jest bliska zeru. To może prowadzić do utraty dokładności i utraty zbieżności. W porównaniu, metoda Newtona może być bardziej stabilna, szczególnie jeśli funkcja jest dobrze zachowana wokół punktu zerowego i wymagana jest wysoka dokładność. W przypadku arytmetyki zmiennoprzecinkowej o skończonej precyzji, obie metody mogą być podatne na błędy numeryczne, zwłaszcza w przypadku funkcji o dużych wartościach lub pochodnych bliskich zeru.

Zadanie 3.

Zapiszmy układ równań jako:

$$\begin{cases} x_1^2 + x_1 x_3^3 - 9 = 0 \\ 3x_1^2 x_2 - x_2^3 - 4 = 0 \end{cases}$$

Korzystając z programu napisanego w języku python otrzymujemy:

```
i:  1 x1:  2.074074074074074 x2:  2.3772290809327847
i:  2 x1:  1.5706579429913439 x2:  1.9701512017893925
i:  3 x1:  1.3710934256680567 x2:  1.7883526546543393
i:  4 x1:  1.3372924358809901 x2:  1.7552189430689191
i:  5 x1:  1.3363560996966315 x2:  1.7542360364584966
i:  6 x1:  1.3363553772176113 x2:  1.7542351976523052
i:  7 x1:  1.336355377217167 x2:  1.7542351976516988
i:  8 x1:  1.336355377217167 x2:  1.7542351976516988
i:  9 x1:  1.336355377217167 x2:  1.7542351976516988
i:  10 x1:  1.336355377217167 x2:  1.7542351976516988
```

Kod:

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


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


def der_f1_x(x, y):
    return 2*x + y**3


def der_f1_y(x, y):
    return 3*(y**2)*x


def der_f2_x(x, y):
    return 6*x*y


def der_f2_y(x, y):
    return 3*(x**2) - 3*(y**2)


def jacobian(x, y):
    return der_f1_x(x, y)*der_f2_y(x, y) - der_f1_y(x, y) * der_f2_x(x, y)


def Newton(x0, y0, n):
    x = x0
    y = y0
    for i in range(1, n + 1):
        delta_x = (-f1(x, y) * der_f2_y(x, y) + f2(x, y) * der_f1_y(x, y))
/ jacobian(x, y)
        delta_y = (-f2(x, y) * der_f1_x(x, y) + f1(x, y) * der_f2_x(x, y))
```

```
/ jacobian(x, y)
        x = x + delta_x
        y = y + delta_y
        print("i: ", i,  "x1: ", x, "x2: ", y)
```

Wnioski: wraz ze wzrostem iteracji wynik staje się coraz bardziej podobny. Oznacza to, że większa ilość iteracji powoduje zwiększenie dokładności wyniku

3. Bibliografia

Wykład

https://www.desmos.com/calculator?lang=pl

https://www.wolframalpha.com/

https://home.agh.edu.pl/~funika/mownit/lab7/RF.pdf

https://pl.wikipedia.org/wiki/Metoda_Newtona