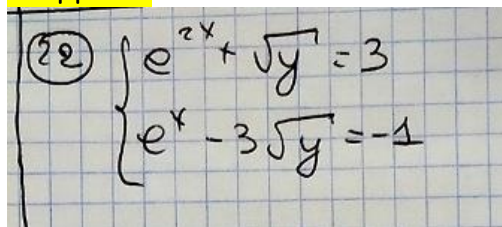


Метод Ньютона

Підготувала:
Студентка ПМІ-23
Шувар Софія

Мета: реалізувати алгоритм методу Ньютона для розв'язання систем нелінійних рівнянь. Продемонструвати роботу програми на конкретному прикладі.

Завдання:

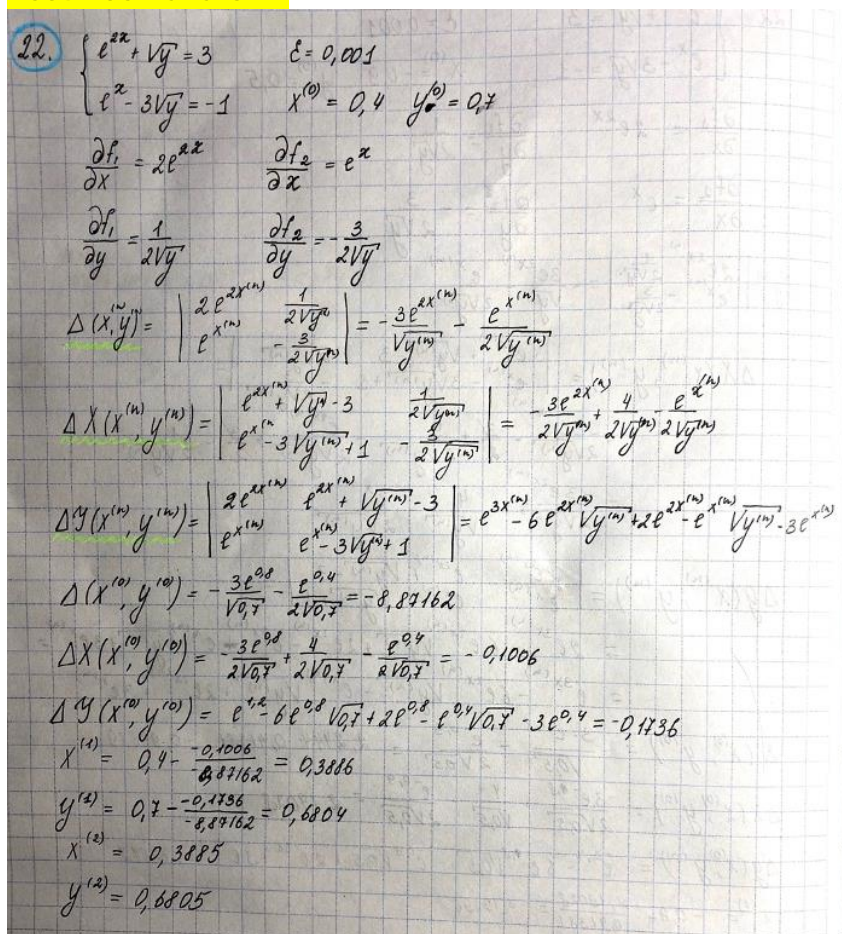


$$(22) \begin{cases} e^{2x} + \sqrt{y} = 3 \\ e^x - 3\sqrt{y} = -1 \end{cases}$$

Хід роботи:

1. Розв'язала систему нелінійних рівнянь методом Ньютона.
2. Реалізувати алгоритм Ньютона на довільній мові програмування.
3. Продемонструвати результати роботи алгоритму на конкретному прикладі.

Розв'язок системи:



$$22. \begin{cases} e^{2x} + \sqrt{y} = 3 \\ e^x - 3\sqrt{y} = -1 \end{cases} \quad \varepsilon = 0,001$$

$$x^{(0)} = 0,4 \quad y^{(0)} = 0,7$$

$$\frac{\partial f_1}{\partial x} = 2e^{2x} \quad \frac{\partial f_1}{\partial y} = \frac{1}{2\sqrt{y}}$$

$$\frac{\partial f_2}{\partial x} = e^x \quad \frac{\partial f_2}{\partial y} = -\frac{3}{2\sqrt{y}}$$

$$\Delta(x^{(n)}, y^{(n)}) = \begin{vmatrix} 2e^{2x^{(n)}} & \frac{1}{2\sqrt{y^{(n)}}} \\ e^{x^{(n)}} & -\frac{3}{2\sqrt{y^{(n)}}} \end{vmatrix} = -\frac{3e^{2x^{(n)}}}{\sqrt{y^{(n)}}} - \frac{e^{x^{(n)}}}{2\sqrt{y^{(n)}}}$$

$$\Delta x(x^{(n)}, y^{(n)}) = \frac{e^{2x^{(n)}} + \sqrt{y^{(n)}} - 3}{e^{x^{(n)}} - 3\sqrt{y^{(n)}} + 1} - \frac{\frac{1}{2\sqrt{y^{(n)}}}}{-\frac{3}{2\sqrt{y^{(n)}}}} = \frac{-\frac{3e^{2x^{(n)}}}{\sqrt{y^{(n)}}} + \frac{4}{2\sqrt{y^{(n)}}} - \frac{e^{x^{(n)}}}{2\sqrt{y^{(n)}}}}{e^{3x^{(n)}} - 6e^{2x^{(n)}}\sqrt{y^{(n)}} + 2e^{x^{(n)}} - e^{x^{(n)}}\sqrt{y^{(n)}} - 3e^{x^{(n)}}}$$

$$\Delta y(x^{(n)}, y^{(n)}) = \frac{2e^{2x^{(n)}}}{e^{x^{(n)}}} \frac{e^{2x^{(n)}} + \sqrt{y^{(n)}} - 3}{e^{x^{(n)}} - 3\sqrt{y^{(n)}} + 1} = \frac{2e^{3x^{(n)}} - 6e^{2x^{(n)}}\sqrt{y^{(n)}} + 2e^{x^{(n)}} - e^{x^{(n)}}\sqrt{y^{(n)}} - 3e^{x^{(n)}}}{e^{3x^{(n)}} - 6e^{2x^{(n)}}\sqrt{y^{(n)}} + 2e^{x^{(n)}} - e^{x^{(n)}}\sqrt{y^{(n)}} - 3e^{x^{(n)}}}$$

$$\Delta x(x^{(0)}, y^{(0)}) = -\frac{3e^{0,8}}{\sqrt{0,7}} - \frac{e^{0,4}}{2\sqrt{0,7}} = -8,87162$$

$$\Delta x(x^{(0)}, y^{(0)}) = \frac{-\frac{3e^{0,8}}{2\sqrt{0,7}} + \frac{4}{2\sqrt{0,7}} - \frac{e^{0,4}}{2\sqrt{0,7}}}{e^{3x^{(0)}} - 6e^{2x^{(0)}}\sqrt{y^{(0)}} + 2e^{x^{(0)}} - e^{x^{(0)}}\sqrt{y^{(0)}} - 3e^{x^{(0)}}} = -0,1006$$

$$\Delta y(x^{(0)}, y^{(0)}) = \frac{2e^{1,2}}{e^{0,4}} \frac{e^{2x^{(0)}} + \sqrt{y^{(0)}} - 3}{e^{x^{(0)}} - 3\sqrt{y^{(0)}} + 1} = \frac{2e^{1,2} - 6e^{0,8}\sqrt{0,7} + 2e^{0,4} - e^{0,4}\sqrt{0,7} - 3e^{0,4}}{e^{3x^{(0)}} - 6e^{2x^{(0)}}\sqrt{y^{(0)}} + 2e^{x^{(0)}} - e^{x^{(0)}}\sqrt{y^{(0)}} - 3e^{x^{(0)}}} = -0,1736$$

$$x^{(1)} = 0,4 - \frac{-0,1006}{-8,87162} = 0,3886$$

$$y^{(1)} = 0,7 - \frac{-0,1736}{-8,87162} = 0,6804$$

$$x^{(2)} = 0,3885$$

$$y^{(2)} = 0,6805$$

Реалізація алгоритму:

Для реалізації алгоритму я обрала мову програмування Python. Алгоритм написаний з використанням бібліотеки numpy.

Глобальні змінні

MAX_ITERATIONS – максимальна кількість ітерацій, у випадку перевищення програма завершає роботу.

message – змінна для збереження інформації про ітерації.

```
1 import numpy as np
2
3 MAX_ITERATIONS = 100
4 message = ''
5
```

Функція Newton_system (реалізація алгоритму):

Аргументи: F – функція системи нелінійних рівнянь, що вертає масив значень в точках x та y, J - функція $F'(x)$, що вертає квадратну матрицю значень x та y, x – початкове наближення (x_0, y_0), eps – значення E.

Метод Ньютона - ітераційний метод розв'язку системи нелінійних рівнянь.

Алгоритм складається з наступних частин:

Рядок 10 – розпочинаємо цикл довжини максимальної кількості ітерацій;

Рядок [11, 14] – отримуємо:

$$D_x = \Delta x(x^{(n)}, y^{(n)}), D_y = \Delta y(x^{(n)}, y^{(n)})$$

Рядок [16, 18] – обчислюємо визначники матриць J(x), D_x, D_y для поточних значень x та y.

Рядок 20 – обчислюємо нові значення x використовуючи формулу:

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

Рядок 26 – зупиняємо ітерацію, якщо при кожній наступній ітерації значення x змінюються менше ніж на E.

Рядок [24-25] – заповнюємо глобальну змінну message інформацією про ітерації та різницю.

```

7 def Newton_system(F, J, x, eps):
8
9     global message
10    for iteration in range(MAX_ITERATIONS):
11        D_x = J(x)
12        D_y = J(x)
13        D_x[:, 0] = F(x)
14        D_y[:, 1] = F(x)
15
16        det = np.linalg.det(J(x))
17        det_x = np.linalg.det(D_x)
18        det_y = np.linalg.det(D_y)
19
20        x_new = np.array([x[0] - det_x / det, x[1] - det_y / det])
21        delta = np.absolute(x_new-x)
22        x = x_new
23
24        message += "Iteration " + str(iteration + 1) + ": " + str(x) + "\n"
25        message += "Difference: " + str(delta) + "\n\n"
26        if np.all(delta < eps):
27            break
28
29    return x
30

```

Функція demonstration(вивід усіх отриманих результатів):

$F(x)$ – функція системи нелінійних рівнянь, що вертає масив значень в точках x та y ;

$J(x)$ - функція $F'(x)$, що вертає квадратну матрицю значень x та y ;

Функції введені для конкретного прикладу.

Початкове наближення: $x_0 = -0.9$ та $y_0 = 0.5$

$E = 0.001$

```

31
32 def demonstration():
33     def F(x):
34         return np.array(
35             [np.exp(2 * x[0]) + x[1] ** 0.5 - 3.0,
36              np.exp(x[0]) - 3 * (x[1] ** 0.5) + 1.0])
37
38     def J(x):
39         return np.array(
40             [[2 * np.exp(2 * x[0]), 1.0 / (2 * (x[1] ** 0.5))],
41              [np.exp(x[0]), -3.0 / (2 * (x[1] ** 0.5))]])
42
43     x = Newton_system(F, J, x=np.array([-0.9, 0.5]), eps=0.001)
44     print("\nIterations:")
45     print(message)
46     print("Solution:")
47     print(x)
48

```

Результати роботи програми на конкретному прикладі:

Результат для $x_0 = 0.4$ та $y_0 = 0.7$

```
Iterations:
Iteration 1: [0.38865297 0.68043155]
Difference: [0.01134703 0.01956845]

Iteration 2: [0.38852893 0.68052031]
Difference: [1.24034163e-04 8.87592500e-05]

Solution:
[0.38852893 0.68052031]

Process finished with exit code 0
```

Результат для $x_0 = -0.9$ та $y_0 = 0.5$

```
Iterations:
Iteration 1: [4.17560178 1.13584692]
Difference: [5.07560178 0.63584692]

Iteration 2: [ 3.67463857 22.64967685]
Difference: [ 0.50096321 21.51382993]

Iteration 3: [ 3.17338805 42.92495014]
Difference: [ 0.50125052 20.2752733 ]

Iteration 4: [ 2.67224409 13.49322532]
Difference: [ 0.50114396 29.43172482]

Iteration 5: [2.17284494 6.69750681]
Difference: [0.49939914 6.79571851]

Iteration 6: [1.68049542 2.72060032]
Difference: [0.49234953 3.97690649]

Iteration 7: [1.21031398 1.50652198]
Difference: [0.47018144 1.21407834]

Iteration 8: [0.79952727 0.92908909]
Difference: [0.41078671 0.57743289]

Iteration 9: [0.51534559 0.73672955]
Difference: [0.28418167 0.19235954]

Iteration 10: [0.40269155 0.68558522]
Difference: [0.11265404 0.05114433]

Iteration 11: [0.38871771 0.68058438]
Difference: [0.01397385 0.00500084]

Iteration 12: [0.38852895 0.68052031]
Difference: [1.88753030e-04 6.40612408e-05]

Solution:
[0.38852895 0.68052031]
```

Висновок

Виконуючи дану практичну роботу, я навчилась розв'язувати систему нелінійних рівнянь за допомогою ітераційного методу Ньютона та реалізувала алгоритм розв'язання на мові Python з використанням можливостей бібліотеки numpy.