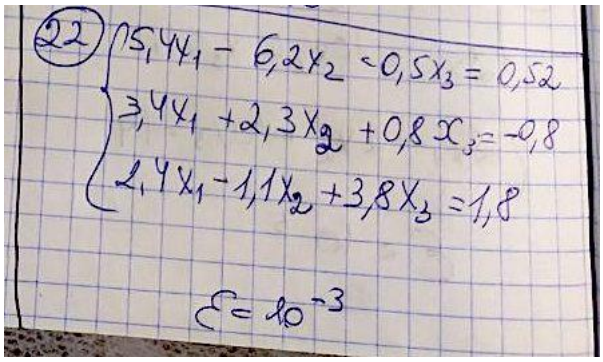
Метод Якобі

Підготувала:
Студентка ПМІ-23
Шувар Софія

Мета: реалізувати алгоритм методу Якобі для розв'язання симетричних СЛАР. Продемонструвати роботу програми на конкретному прикладі.

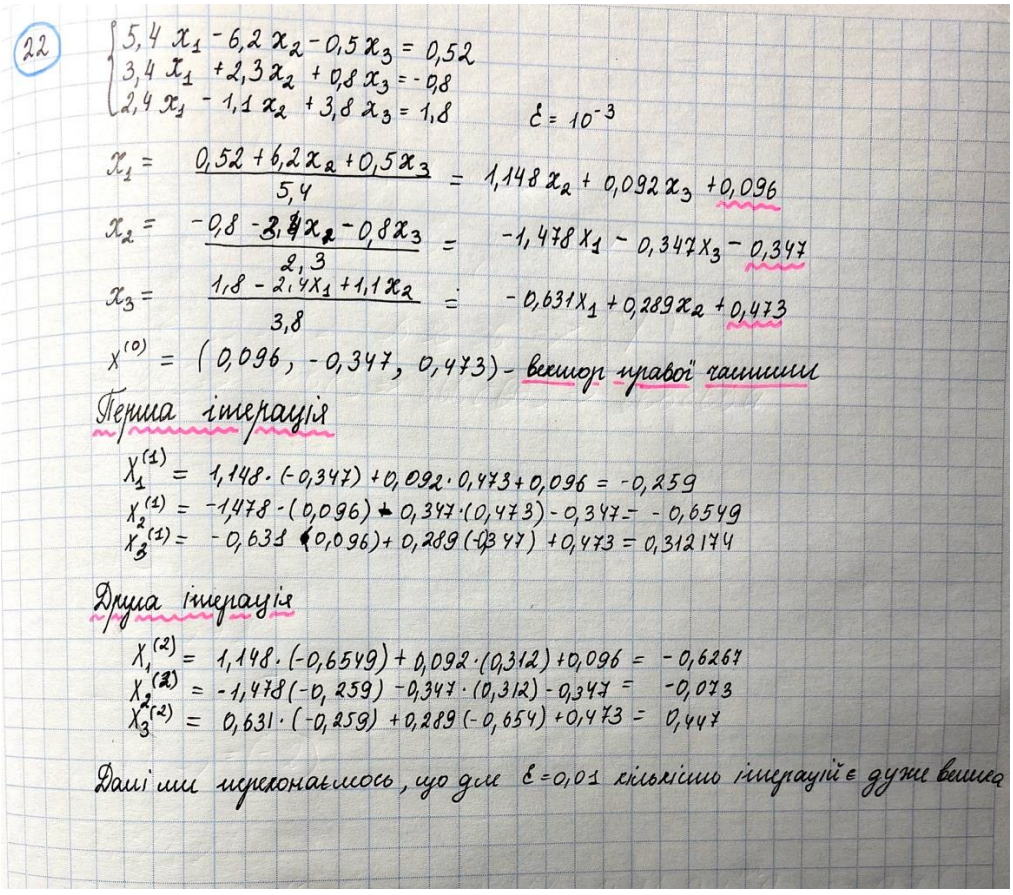
Завдання:



Хід роботи:

1. Розв'язала систему лінійних рівнянь методом Якобі.
2. Реалізувати алгоритм Якобі на довільний мові програмування.
3. Продемонструвати результати роботи алгоритму на конкретному прикладі.

Розв'язок системи:



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.

```python
import numpy as np

MAX_ITERATIONS = 1000
message = ""

```

Функція user_input(вхідні дані):

Функція наповнює матрицю коефіцієнтів A введену користувачем з клавіатури, масив b вільних членів та значення E.

```python
def user_input():
    n = int(input("Enter matrix size: "))
    print("Enter A matrix:")
    a = np.array([input().strip().split() for _ in range(n)], float)
    print("Enter B vector:")
    b = np.array(input().strip().split(), dtype=float)
    exp = float(input("Enter exp:"))
    return n, a, b, exp

```

Функція print_equation_system():

Аргументи: n – розмір матриці; A – початкова матриця; b – вектор вільних членів.

Функція виводить матрицю та вектор введений користувачем у вигляді системи лінійних рівнянь.

```python
def print_equation_system(n, A, b):
    for i in range(n):
        print(" + ".join([str(A[i, j]) + "*x" + str(j + 1) for j in range(n)]), "=", b[i])

```

Функція Jacobi (реалізація алгоритму):

Аргументи: Аргументи: n – розмір матриці; A – початкова матриця; b – вектор вільних членів, exp – значення E.

Метод Якобі: ітераційний метод розв’язку системи лінійних рівнянь.

Метод виражається формулою:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

Алгоритм складається з наступних частин:

Рядок 24 – оголошуємо x – масив розв'язків системи;

Рядок 25 – розпочинаємо цикл довжини максимальної кількості ітерацій

Рядок [26-31] – обчислюємо проміжні значення x використовуючи формулу вище.

Рядок 32 – зупинити ітерацію, якщо при кожній наступній ітерації значення x змінюються менше ніж на E.

Рядок [35-36] – заповнюємо глобальну змінну message інформацією про ітерації та різницю.

```
16
17  def print_equation_system(n, A, b):
18      for i in range(n):
19          print(" + ".join([str(A[i, j]) + "*x" + str(j + 1) for j in range(n)]), "=", b[i])
20
21
22  def Jacobi(n, A, b, exp):
23      global message
24      x = np.zeros(n)
25      for iteration in range(MAX_ITERATIONS):
26          xx = np.zeros(n)
27          for i in range(n):
28              x1 = A[i, :i]@ x[:i]
29              x2 = A[i, i + 1:]@ x[i + 1:]
30              xx[i] = (b[i] - x1 - x2) / A[i, i]
31
32          if np.allclose(x, xx, atol=exp):
33              break
34
35          message += "Iteration " + str(iteration + 1) + ": " + str(x) + "\n"
36          message += "Difference: " + str(x - xx) + "\n\n"
37
38          x = xx
39      return x
40
```

Функція demonstration(вивід усіх отриманих результатів):

```
39
40  def demonstration():
41      n, A, b, exp = user_input()
42      print("\nEquation system:")
43      print_equation_system(n, A, b)
44      x = Jacobi(n, A, b, exp)
45      print("\nIterations:")
46      print(message)
47      print("Solution:")
48      print(x)
49
50
51  if __name__ == "__main__":
52      demonstration()
53
```

Результати роботи програми на конкретному прикладі:

```
Enter matrix size: 3
Enter A matrix:
5.4 -6.2 -0.5
3.4 2.3 0.8
2.4 -1.1 3.8
Enter B vector:
0.52 -0.8 1.8
Enter exp:0.001

Equation system:
5.4*x1 + -6.2*x2 + -0.5*x3 = 0.52
3.4*x1 + 2.3*x2 + 0.8*x3 = -0.8
2.4*x1 + -1.1*x2 + 3.8*x3 = 1.8

Iterations:
Iteration 1: [0. 0. 0.]
Difference: [-0.0962963   0.34782609 -0.47368421]

Iteration 2: [ 0.0962963  -0.34782609  0.47368421]
Difference: [0.35549623 0.30711077 0.16150521]

Iteration 3: [-0.25919993 -0.65493686  0.312179  ]
Difference: [ 0.36756285 -0.58169189 -0.13562345]

Iteration 4: [-0.62676278 -0.07324497  0.44780245]
Difference: [-0.68042619 -0.49618041 -0.40052945]

Iteration 5: [0.05366341 0.42293544 0.8483319 ]
Difference: [-0.60677467  1.14516201  0.28611169]

Iteration 6: [ 0.66043808 -0.72222657  0.56222021]
Difference: [1.34130746 0.79745415 0.71472038]

Iteration 7: [-0.68086938 -1.51968072 -0.15250017]
Difference: [ 0.98177332 -2.23140073 -0.61629956]

Iteration 8: [-1.66264269  0.71172001  0.4637994 ]
Difference: [-2.61904339 -1.23695201 -1.26599915]

Iteration 9: [0.95640069 1.94867202 1.72979855]
Difference: [-1.53742631  4.31197688  1.29606761]

Iteration 10: [ 2.493827   -2.36330486  0.43373094]
Difference: [5.07079453 1.82191102 2.21920992]

Iteration 11: [-2.57696753 -4.18521589 -1.78547899]
Difference: [ 2.29730617 -8.26785624 -2.67521178]

Iteration 12: [-4.8742737   4.08264035  0.88973279]
Difference: [-9.74042862 -2.46550937 -3.84425702]

Iteration 13: [4.86615492 6.54814972 4.73398981]
```

...

```
Iteration 986: [-1.14845601e+133  5.18048362e+132 -1.25910587e+132]
Difference: [-1.73159389e+133 -1.22346421e+133 -1.00121260e+133]

Iteration 987: [5.83137880e+132 1.74151257e+133 8.75302008e+132]
Difference: [-1.49742304e+133  2.90799535e+133  7.39477556e+132]

Iteration 988: [ 2.08056092e+133 -1.16648278e+133  1.35824452e+132]
Difference: [3.40727962e+133 1.95637229e+133 1.78752899e+133]

Iteration 989: [-1.32671871e+133 -3.12285508e+133 -1.65170454e+133]
Difference: [ 2.41171717e+133 -5.65859736e+133 -1.58564778e+133]

Iteration 990: [-3.73843588e+133  2.53574228e+133 -6.60567590e+131]
Difference: [-6.64372731e+133 -3.01361746e+133 -3.16120482e+133]

Iteration 991: [2.90529144e+133 5.54935974e+133 3.09514806e+133]
Difference: [-3.75278345e+133  1.09207116e+134  3.32367536e+133]

Iteration 992: [ 6.65807489e+133 -5.37135188e+133 -2.28527298e+132]
Difference: [1.28463425e+134 4.39153194e+133 5.53143765e+133]

Iteration 993: [-6.18826765e+133 -9.76288382e+133 -5.75996495e+133]
Difference: [ 5.55429941e+133 -2.09142238e+134 -6.84224657e+133]

Iteration 994: [-1.17425671e+134  1.11513400e+134  1.08228162e+133]
Difference: [-2.46461687e+134 -5.83079163e+133 -9.56209599e+133]

Iteration 995: [1.29036016e+134 1.69821316e+134 1.06443776e+134]
Difference: [-7.57999187e+133  3.97594132e+134  1.38781405e+134]

Iteration 996: [ 2.04835935e+134 -2.27772816e+134 -3.23376292e+133]
Difference: [4.69347096e+134 6.37802606e+133 1.62966671e+134]

Iteration 997: [-2.64511161e+134 -2.91553076e+134 -1.95304300e+134]
Difference: [ 8.83186947e+133 -7.50501506e+134 -2.77967038e+134]

Iteration 998: [-3.52829856e+134  4.58948430e+134  8.26627377e+133]
Difference: [-8.87424603e+134 -3.38738833e+133 -2.73030664e+134]

Iteration 999: [5.34594747e+134 4.92822313e+134 3.55693402e+134]
Difference: [-6.41728534e+133  1.40681225e+135  5.50673099e+134]

Iteration 1000: [ 5.98767601e+134 -9.13989940e+134 -1.94979697e+134]
Difference: [ 1.66621713e+135 -9.66742511e+133  4.47765349e+134]


Solution:
[-1.06744953e+135 -8.17315689e+134 -6.42745046e+134]

Process finished with exit code 0
```

Оскільки приклад заданий у завданні не збігається до заданого Е. Тому продемонструю роботу програми, на прикладі запропонованому на практичному занятті.

```
Enter matrix size: 3
Enter A matrix:
10 1 -1
1 10 -1
-1 1 10
Enter B vector:
11 10 10
Enter exp:0.001

Equation system:
10.0*x1 + 1.0*x2 + -1.0*x3 = 11.0
1.0*x1 + 10.0*x2 + -1.0*x3 = 10.0
-1.0*x1 + 1.0*x2 + 10.0*x3 = 10.0

Iterations:
Iteration 1: [0. 0. 0.]
Difference: [-1.1 -1.  -1. ]

Iteration 2: [1.1 1.  1. ]
Difference: [ 0.    0.01 -0.01]

Iteration 3: [1.1  0.99 1.01]
Difference: [-0.002 -0.001 -0.001]


Solution:
[1.102 0.991 1.011]

Process finished with exit code 0
```

Висновок

Виконуючи дану практичну роботу, я навчилась навчилася розв’язувати систему лінійних рівнянь за допомогою ітераційного методу Якобі та реалізувала алгоритм розв`язання на мові Python з використанням можливостей бібліотеки numpy.