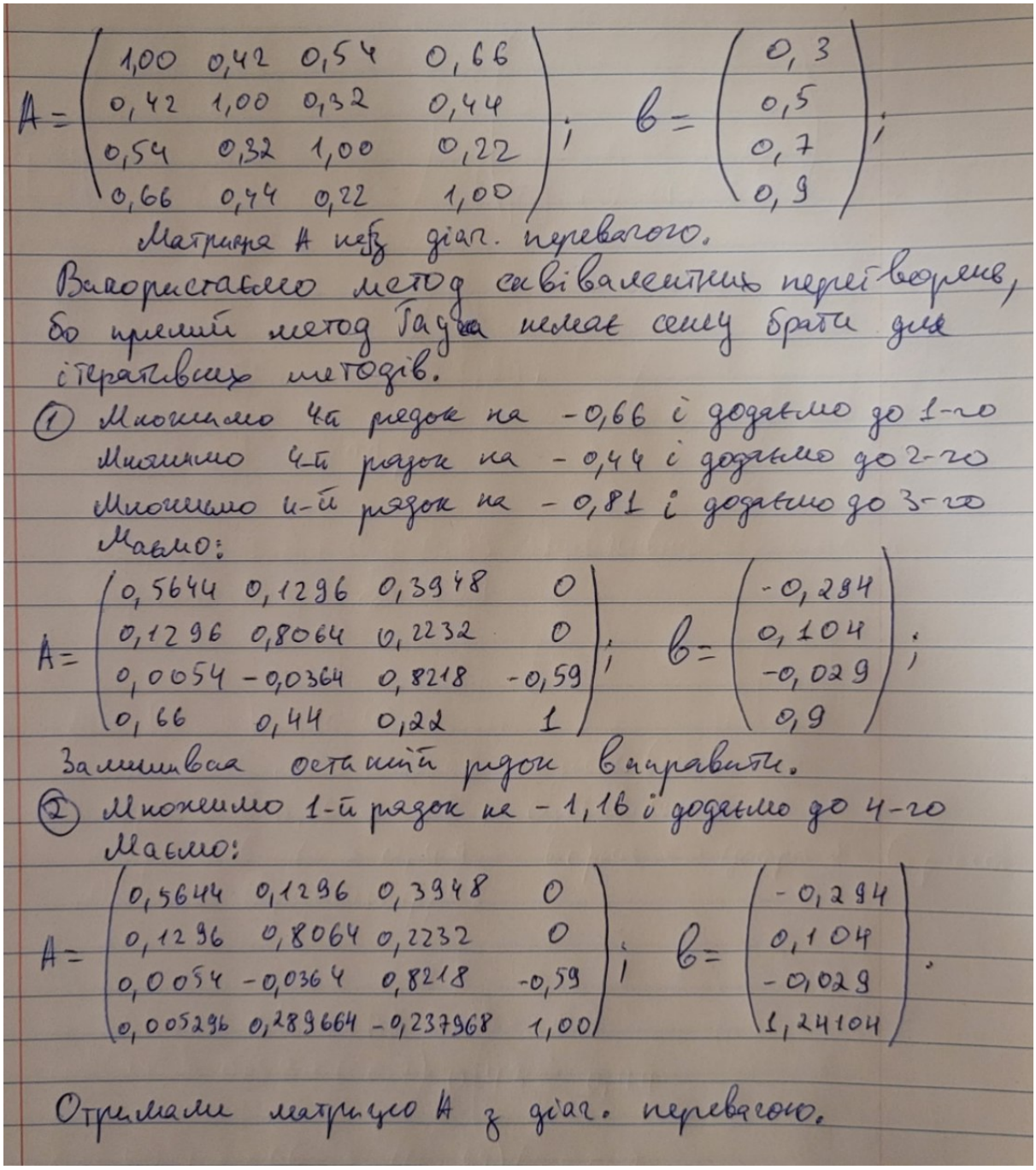
Комп’ютерний практикум №3
Розв’язання СЛАР ітераційними методами

Виконав:
Студент 3 курсу ФТІ
групи ФІ-92
Поночевний Назар Юрійович
Варіант 12

Завдання:
Якщо матриця не є матрицею із діагональною перевагою, привести систему до еквівалентної, у якій є діагональна перевага (письмово). Реалізувати програму, що реалізує розв’язання за ітераційним методом, який відповідає заданому варіантові (метод простої ітерації). Обчислення проводити з 10^−4. Для кожної ітерації розраховувати вектор нев’язки r = |b – Ax|, де x - отриманий розв’язок.

1) Приведемо матрицю до діагональної переваги



2) Реалізуємо метод простої ітерації

```
"""
Iteratively solving a system of equations

"""

# ------------ Input ------------


import numpy as np
```

```python
A = np.array([[0.5644, 0.1296, 0.3948, 0.],
              [0.1296, 0.8064, 0.2232, 0.],
              [0.0054, -0.0364, 0.8218, -0.59],
              [0.005296, 0.289664, -0.237968, 1.]])

b = np.array([-0.294, 0.104, -0.029, 1.24104]).reshape(-1, 1)

e = 10**-4


# ------------ Code ------------


def simple_iteration_method(C, x, d):
    return C.dot(x) + d


def main():
    C = np.zeros_like(A)
    for i in range(len(C)):
        for j in range(len(C)):
            if i != j:
                C[i, j] = -A[i, j] / A[i, i]
    d = np.zeros_like(b)
    for i in range(len(d)):
        d[i, 0] = b[i, 0] / A[i, i]
    x = np.ones_like(b)
    q = max(np.sum(np.abs(C), axis=1)) + 0.01
    print(f"C:\n{C}")
    print(f"d:\n{d}")
    print(f"x0:\n{x}")
    print(f"q:\n{q}")

    i = 1
    x_new = simple_iteration_method(C, x, d)
    r = np.sum(np.abs(b - A.dot(x_new)))
    criteria = (1 / (1 - q)) * max(np.abs(x_new - x))
    print(f"\nIteration #{i}\nx:\n{x_new}")
    print(f"r: {r}")
    print(f"criteria: {criteria}")
    while criteria >= e:
        x = x_new
        i += 1
        x_new = simple_iteration_method(C, x, d)
        r = np.sum(np.abs(b - A.dot(x_new)))
```

```
        criteria = (1 / (1 - q)) * max(np.abs(x_new - x))
        print(f"\nIteration #{i}\nx:\n{x_new}")
        print(f"r: {r}")
        print(f"criteria: {criteria}")


if __name__ == "__main__":
    main()
```

3) Результат

```
C:
[[ 0.         -0.22962438 -0.6995039  -0.        ]
 [-0.16071429  0.         -0.27678571 -0.        ]
 [-0.00657094  0.04429302  0.          0.71793624]
 [-0.005296   -0.289664    0.237968    0.        ]]
d:
[[-0.52090716]
 [ 0.12896825]
 [-0.03528839]
 [ 1.24104   ]]
x0:
[[1.]
 [1.]
 [1.]
 [1.]]
q:
0.9391282778171509
```

```
Iteration #1
x:
[[-1.45003544]
 [-0.30853175]
 [ 0.72036992]
 [ 1.184048  ]]
r: 1.0595765686032106
criteria: [40.2491559]

Iteration #2
x:
[[-0.95396231]
 [ 0.16262156]
 [ 0.81064487]
 [ 1.50951492]]
r: 0.5052599366210451
criteria: [8.14948393]

Iteration #3
x:
[[-1.12529828]
 [ 0.05790871]
 [ 1.0619185 ]
 [ 1.39189411]]
r: 0.28282780429387067
criteria: [4.12792044]
```

```
Iteration #15
x:
[[-1.25777144]
 [ 0.04349842]
 [ 1.03916641]
 [ 1.48239265]]
r: 2.9073593836383832e-05
criteria: [0.00078303]

Iteration #16
x:
[[-1.25779641]
 [ 0.04348367]
 [ 1.03916643]
 [ 1.48238958]]
r: 1.1758490339148359e-05
criteria: [0.00041031]

Iteration #17
x:
[[-1.25779304]
 [ 0.04348768]
 [ 1.03916374]
 [ 1.48239399]]
r: 5.25595786150454e-06
criteria: [7.24238432e-05]
[Finished in 0.2s]
```

4) Контрольні запитання

В чому полягає основна відмінність прямих та ітераційних методів розв'язання СЛАР?

Основна відмінність полягає в тому, що через природу ітераційних методів ми можемо регулювати кількість ітерацій і відповідно точність обчислення. На відміну від прямих методів, де точність і сходимість алгоритму є завчасно відомою.

Який метод буде збігатись швидше при однакових вихідних даних – метод Зейделя чи метод простої ітерації?

Метод Зейделя буде швидше збігатись, бо через умову симетричності й додатної визначеності та алгоритму обрахунків, критерій завершення ітерацій буде не таким суворим, як у методі простої ітерації.