# uData School

**Лабораторна робота №2**

## на тему:

# “Розв’язання систем лінійних алгебраїчних рівнянь. Метод простих ітерацій. Метод Зейделя ”

## з курсу “MathPro”

## Варіант №7

Виконав:

Опанасенко Я.П.

Перевірила:

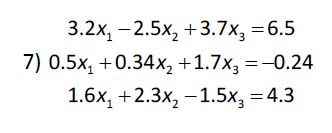
Прощенко Т.М.

# Мета

Вивчення алгоритмів для розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) ітераційними методами.

# Завдання

З’ясувати факт збіжності чи розбіжності ітераційних процесів простих ітерацій на Зейделя. У випадку збіжності знайти розв’язок СЛАР з точністю 0.001 та перевірити його, підставляючи в СЛАР отримані розв’язки і обраховуючи нев’язки. Визначити порядок збіжності ітераційного процесу.



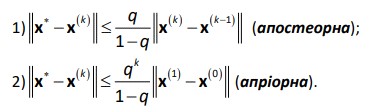
# Теоретичні відомості

1. **Метод простих ітерацій**

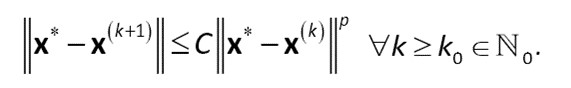
Метод простих ітерацій – ітераційний метод, в якому вектор х обчислюється як:

*xk+1*=*Bxk*+*c*

Оцінки похибок:

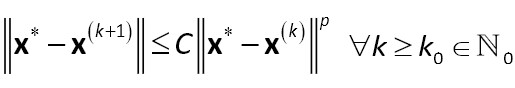


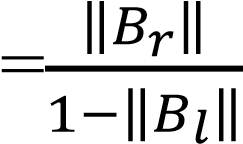
Для оцінки порядку збіжності використовується нерівність:



1. **Метод Зейделя**

Метод Зейделя аналогічний методові простих ітерацій, але при знаходженні *xk+1* використовуються вже обчислені значення *xk+1*. Для оцінки порядку збіжності використовується нерівність:



*C*

1. **Програмна реалізація методів Якобі і Зейделя**

**def** modificate\_matrix\_to\_Jacobi(matrix\_a, b): matrix\_start = copy.deepcopy(matrix\_a) matrix\_start\_b = copy.deepcopy(b)

*# modificate matrix*

*# check diagonal domination* **for** k **in** range(len(matrix\_start)):

summ = 0 **for** i **in** range(len(matrix\_start)): summ += np.abs(matrix\_start[i][k]) **if** abs(matrix\_start[k][k]) < summ/2:

**raise** ValueError(**"Error! Matrix without diagonal domination!"**)

*# in diagonal 1*

**for** i **in** range(len(matrix\_start)):

coef = matrix\_start[i][i] **for** j **in** range(len(matrix\_start)): matrix\_start[i][j] /= coef matrix\_start\_b[i][0] /= coef

*# Ax = b to x = Bx + c*

matrix\_b = np.eye(len(matrix\_start)) - matrix\_start c = matrix\_start\_b **return**(matrix\_b, c)

**def** simple\_iterations(matrix\_b, c, matrix\_a, b, zeid):

*# max number of iterations* n = len(c)

norm = np.linalg.norm(c, np.inf)

*# formulae (11)*

maximum\_i = int(np.floor(np.log(eps / norm \* (1 - np.linalg.norm(matrix\_b, np.inf)))

/ np.log(np.linalg.norm(matrix\_b, np.inf)))) print(**"max number of iteration: "**, maximum\_i) x\_current = copy.deepcopy(c) x\_previous = copy.deepcopy(c) x\_errors = copy.deepcopy(x\_current) maximum\_sum = eps it = 1 **while** maximum\_sum >= eps: **for** i **in** range(len(matrix\_b)):

x\_previous[i][0] = x\_current[i][0] x\_current[i][0] = copy.deepcopy(c[i][0]) **for** i **in** range(n): **for** j **in** range(n): **if** j < i:

x\_current[i][0] += matrix\_b[i][j] \* x\_current[j][0] **if** zeid **else** matrix\_b[i][j] \* x\_previous[j][0] **else**:

x\_current[i][0] += matrix\_b[i][j] \* x\_previous[j][0] maximum\_sum\_prev = maximum\_sum

= 0 maximum = 0 **for** i **in** range(len(matrix\_b)):

x\_errors[i][0] = x\_current[i][0] - x\_previous[i][0] maximum\_sum += np.abs(x\_errors[i][0]) **if** np.abs(x\_errors[i][0]) > maximum: maximum = np.abs(x\_errors[i][0]) print(**"\n\nIteration "**, it, **":"**, sep=**""**) print\_matrix\_solution(x\_current, **False**)

print(**"Apriori: "**, norm\*\*it/(1-norm)\*np.linalg.norm(c, np.inf)) print(**"Aposteriori: "**, maximum\*norm/(1-norm)) errors(matrix\_a, x\_current, b) it += 1 **if** it >= maximum\_i:

print(it)

**raise** Exception(**"Error! More than max number of iterations!"**) **if** zeid:

print(**"\n\nOrder of convergence "**,

np.floor(np.log(maximum\_sum \* norm / (1 - norm)) / np.log(maximum\_sum\_prev \* norm / (1 - norm)))) **else**:

print(**"\n\nOrder of convergence**

**"**,np.floor(np.log(maximum\_sum\*norm/(1-norm))/np.log(maximum\_sum\_prev\*norm/(1norm)))) **return** x\_current

**Результати роботи програми**

Мовою програмування Python реалізовано програму для розв’язання систем лінійних рівнянь та пошуку обернених матриць.

Для матриці, заданої за умовою були проведені лінійні перетворення таким чином, що матриця отримала характеристику діагонального домінування.

**Висновки**

За результатами виконання лабораторної роботи можна зробити наступні висновки:

1. Запрограмовано мовою Python алгоритм розв’язання довільної системи лінійних рівнянь методом простих ітерацій та методом Зейделя.
2. Запрограмоване приведення матриці А до вигляду В, де В- матриця, діагональні елементи якої- нулі.
3. Оброблені ситуації, коли розв’язати систему неможливо.
4. Підраховані апріорні та апостеріорні.
5. Підраховані порядки збіжності для методів Якобі та Зейделя.