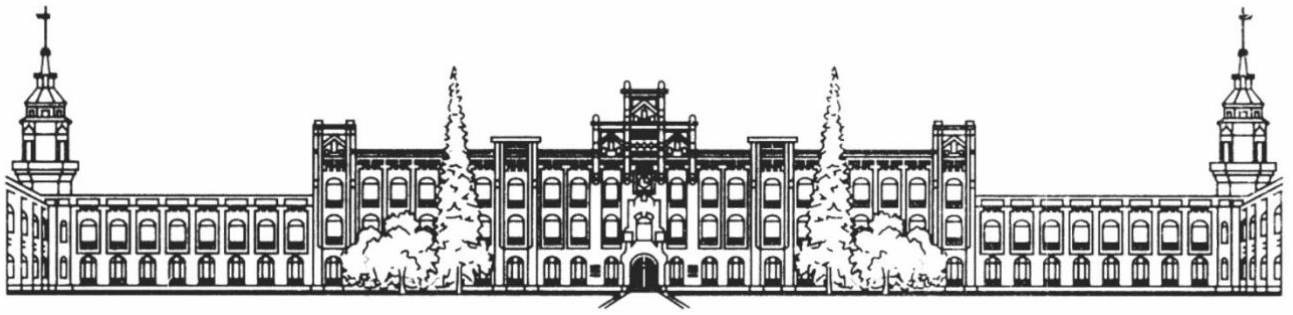

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» Інститут
прикладного системного аналізу

Лабораторна робота № 2

з курсу «Чисельні методи»

з теми «Ітераційні методи розв'язвння СЛАР»

Варіант № 9

Виконав студент 2 курсу групи КА-02
Романович Володимир Володимирович
перевірила старший викладач
Хоменко Ольга Володимирівна

Київ – 2022

Завдання

Методом Якобі або Зейделя розв'язати систему рівнянь:

$$\begin{cases} 2.389x_1 + 0.273x_2 + 0.126x_3 + 0.418x_4 = 0.144 \\ 0.329x_1 + 2.796x_2 + 0.179x_3 + 0.278x_4 = 0.297 \\ 0.186x_1 + 0.275x_2 + 2.987x_3 + 0.316x_4 = 0.529 \\ 0.197x_1 + 0.219x_2 + 0.274x_3 + 3.127x_4 = 0.869 \end{cases}$$

Маємо матрицю A $= \begin{pmatrix} 2.389 & 0.273 & 0.126 & 0.418 \\ 0.329 & 2.796 & 0.179 & 0.278 \\ 0.186 & 0.275 & 2.987 & 0.316 \\ 0.197 & 0.219 & 0.274 & 3.127 \end{pmatrix}$

Перевіримо її на діагональну перевагу:

$2.389 > 0.817 = 0.273 + 0.126 + 0.418$

$2.796 > 0.786 = 0.329 + 0.179 + 0.278$

$2.987 > 0.777 = 0.275 + 0.186 + 0.316$

$3.127 > 0.69 = 0.219 + 0.274 + 0.197$

Отже умова діагональної переваги виконується, тому за теоремою метод Зейделя для цієї системи є збіжним, причому швидше ніж метод Якобі, тому використовуватимемо саме його.

Запишемо систему у вигляді $\vec{x} = (-D^{-1}(L+R))\vec{x} + D^{-1}\vec{c}$:

$$\vec{x} = \begin{pmatrix} 0 & -\frac{0.273}{2.389} & -\frac{0.126}{2.389} & -\frac{0.418}{2.389} \\ -\frac{0.329}{2.796} & 0 & -\frac{0.179}{2.796} & -\frac{0.278}{2.796} \\ -\frac{0.186}{2.987} & -\frac{0.275}{2.987} & 0 & -\frac{0.316}{2.987} \\ -\frac{0.197}{3.127} & -\frac{0.219}{3.127} & -\frac{0.274}{3.127} & 0 \end{pmatrix} \cdot \vec{x} + \begin{pmatrix} \frac{1}{2.389} & 0 & 0 & 0 \\ 0 & \frac{1}{2.796} & 0 & 0 \\ 0 & 0 & \frac{1}{2.987} & 0 \\ 0 & 0 & 0 & \frac{1}{3.127} \end{pmatrix} \cdot \begin{pmatrix} 0.144 \\ 0.297 \\ 0.529 \\ 0.869 \end{pmatrix}$$

Знайдемо $\|B\|_\infty = \| - D^{-1}(L+R)\|_\infty = \max\left\{\frac{817}{2389}, \frac{786}{2796}, \frac{777}{2987}, \frac{690}{3127}\right\} = \frac{817}{2389} \approx 0.34 < 0.5$

Тому критерієм зупинки можемо взяти умову $\|x_* - x^{(k)}\|_\infty \leq \varepsilon = 0.00001$

Реалізувавши метод Зейделя в Python отримуємо таку відповідь: $\vec{x} \approx \begin{pmatrix} -0.000983 \\ 0.071286 \\ 0.143047 \\ 0.260437 \end{pmatrix}$

Iteration	x_1	x_2	x_3	x_4	$\|\|x^{(k+1)} - x^{(k)}\|\|$
0	0.060276	0.106223	0.177101	0.277902	
1	-0.009827	0.068410	0.142015	0.261286	0.070103
2	-0.000748	0.071240	0.142947	0.260434	0.009079
3	-0.000972	0.071292	0.143046	0.260436	0.000224
4	-0.000983	0.071286	0.143047	0.260437	0.000011
5	-0.000983	0.071286	0.143047	0.260437	0.000000

Табл. 1: Результат роботи алгоритму з початковим наближенням $D^{-1}\vec{c}$

Також в програмі знайдемо розв'язок за допомогою функції linalg.solve() з бібліотеки Numpy, обчислимо вектор нев'язки $\vec{b} - A \cdot \vec{x}_*$, та застосуємо цей метод з іншим початковим наближенням.

Вихідні дані програми:

Our solution: [-0.0009827818268132518, 0.07128626360839627, 0.14304684446284438, 0.26043718609108546]

Numpy function's solution: [-0.00098275 0.07128627 0.14304684 0.26043718]

b-A*x: ['0.000000', '0.000000', '-0.000000', '-0.000000']

Our solution with (10, 20, 30, 40) starting vector: [-0.0009826903538848741, 0.07128627985793794, 0.1430468342637694, 0.2604371800839636]

Бачимо що наш розв'язок $\vec{x}_* \approx \begin{pmatrix} -0.000983 \\ 0.071286 \\ 0.143047 \\ 0.260437 \end{pmatrix}$ дорівнює розв'язку Numpy $\tilde{\vec{x}} = \begin{pmatrix} -0.00098275 \\ 0.07128627 \\ 0.14304684 \\ 0.26043718 \end{pmatrix}$ з точністю ε

Також бачимо що вектор нев'язки $\vec{b} - A \cdot \vec{x}_* \approx \vec{0}$

Взявши за початкове наближення вектор $\vec{a} = (10, 20, 30, 40)^T$ отримаємо таку ж відповідь з точністю до ε, але кількість ітерацій збільшилася

Iteration	x_1	x_2	x_3	x_4	$\|\|x^{(k+1)} - x^{(k)}\|\|$
0	0.060276	0.106223	0.177101	0.277902	
1	-10.806195	-4.519943	-2.965538	1.535096	38.464904
2	0.464601	0.088777	-0.022404	0.244378	11.270796
3	0.008555	0.082353	0.143133	0.259054	0.456047
4	-0.002010	0.071539	0.143234	0.260468	0.010814
5	-0.001027	0.071276	0.143047	0.260441	0.000983
6	-0.000982	0.071286	0.143046	0.260437	0.000045
7	-0.000983	0.071286	0.143047	0.260437	0.000000

Табл. 2: Результат роботи алгоритму з початковим наближенням $\vec{a}$

Висновок

Переконавшись у виконанні достатньої умови збіжності для нашої системи, ми використали метод Зейделя для розв'язання системи із заданою точністю. Отримали відповідь $\vec{x} = (-0.00098, 0.07128, 0.14304, 0.26437)^T$, знайшли вектор нев'язки і він дорівнює $\vec{0}$ із заданою точністю, тому можемо сказати що отримана відповідь є правильною. Також ми застосували алгоритм вже з іншим початковим наближенням, і, як і очікувалось отримали таку ж відповідь(з точністю до ε), але знадобилось на 2 ітерації більше. Такий результат зрозумілий, оскільки якщо метод Зейделя для системи збіжний, то він збіжний при будь якому скінченному початковому наближенню.

Лістинг програми

```
import numpy
import pandas
```

```
import openpyxl


def chebyshev_norm(vector):
    """Returns Chebyshev's norm of vector"""
    abs_vector = [abs(val) for val in vector]
    return max(abs_vector)


def vector_subtract(vec1, vec2):
    """Returns vector that is result of subtracting 2 vectors"""
    vec3 = []
    for i in range(len(vec1)):
        vec3.append(vec1[i] - vec2[i])
    return vec3


A_matrix = numpy.array([        # from A*x = b
    [2.389, 0.273, 0.126, 0.418],
    [0.329, 2.796, 0.179, 0.278],
    [0.186, 0.275, 2.987, 0.316],
    [0.197, 0.219, 0.274, 3.127]
])
b_vector = numpy.array([0.144, 0.297, 0.529, 0.869])    # from A*x = b
B_matrix = [        # from x = Bx + c
     [-1/2.389 * val for val in [0, 0.273, 0.126, 0.418]],
     [-1/2.796 * val for val in [0.329, 0,  0.179, 0.278]],
     [-1/2.987 * val for val in [0.186, 0.275, 0, 0.316]],
     [-1/3.127 * val for val in [0.197, 0.219, 0.274, 0]]]
c_vector = [0.144/2.389, 0.297/2.796, 0.529/2.987, 0.869/3.127]     # from x = Bx + c


def zeydel_method(B, c, norm, epsilon, starting_val=None):
    """Solve system of linear equations x = B*x + c using Zeydel method"""
    if not starting_val:
        # Setting vector c as starting value
        prev = list(c)
    else:
        prev = list(starting_val)
    x = list(prev)
    n = len(x)
    # Variables for displaying information in table
    iteration = 1
    # Array that holds information for displaying in table
    info_arr = [[0] + list(c)]
    while True:

        for i in range(n):
            x_i = 0
            # Finding b_i0 * x_0 + b_i1 * x_1 + ... + b_in * x_n sum and storing in x_i
            for j in range(n):
                x_i += B[i][j] * x[j]

            # Finally, value of the coordinate of vector is x[i] = b_i0 * x_0 + b_i1 * x_1 + ... + b_in * x_n + c_i
            x[i] = x_i + c[i]

        vector_subtract_norm = norm(vector_subtract(x, prev))    # ||x^(k+1) - x^(k)||
        # Getting iteration, x_1, x_2, ..., x_n, ||x^(k+1) - x^(k)|| in a list to use in table
        info_arr.append([iteration] + list(x) + [vector_subtract_norm])

        # Break criteria ||x^(k+1) - x^(k)|| <= eps
        if vector_subtract_norm <= epsilon:
```

```
            # Formatting numbers
            for k in range(len(info_arr)):
                info_arr[k] = ["{:.6f}".format(value) if isinstance(value, float) else value for value in info_arr[k]]

            # Making .xlsx table
            df = pandas.DataFrame(info_arr)
            df.to_excel("output.xlsx",
                        header=["Iteration", "$x_1$", "$x_2$", "$x_3$", "$x_4$", "$||x^{(k+1)} - x^{(k)}||$"],
                        index=None)
            return x

        else:
            prev = list(x)
            iteration += 1


x = zeydel_method(B_matrix, c_vector, chebyshev_norm, 0.00001)      # Our solution
numpy_x = numpy.linalg.solve(A_matrix, b_vector)    # Numpy function solution
print("Our solution:", x)
print("Numpy function's solution:", numpy_x)

Ax_vector = numpy.matmul(A_matrix, x)   # A*x
b_minus_Ax_vector = vector_subtract(b_vector, Ax_vector)    # b-A*x
b_minus_Ax_vector = ["{:.6f}".format(coord) for coord in b_minus_Ax_vector]     # Format nicely b-A*x vector
print("b-A*x: ", b_minus_Ax_vector)
# Our solution with starting vector (10,20,30,40)
y = zeydel_method(B_matrix, c_vector, chebyshev_norm, 0.00001, [10, 20, 30, 40])
print("Our solution with (10, 20, 30, 40) starting vector:", y)
```