

Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

**Лабораторна робота №3**

Чисельні методи в інформатиці

“Власні значення та системи нелінійних рівнянь”

Варіант №8

Виконав студент групи ІПС-31

Тесленко Назар Олександрович

Київ - 2025

## **Постановка задачі:**

### Варіант №8

1. Знайти найменше власне значення степеневим методом, та наближення до всіх власних значень методом обертань Якобі

$$\begin{matrix} 4 & 1 & 0 & 1 \\ 1 & 3 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 1 & 0 & 0 & 2 \end{matrix}$$

2. Розв'язати модифікованим методом Ньютона

$$\sin(x - 0.6) - y = 1.6$$

$$3x - \cos(y) = 0.9$$

## **Теоретичний опис та обґрунтування:**

### Степеневий метод

Нехай  $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$ . Будемо також шукати максимальне власне значення  $|\lambda_1|$ . Початкове наближення  $\bar{x}^0$  обираємо довільним, але  $\bar{x}^0 \neq \bar{0}$ .

Ітераційний процес:

$$\bar{x}^{k+1} = A\bar{x}^k; \quad \lambda_1^{k+1} = \frac{\bar{x}_m^{k+1}}{\bar{x}_m^k}, \quad \forall m: 1 \leq m \leq n$$

Умова припинення:  $|\lambda_1^{k+1} - \lambda_1^k| \leq \varepsilon$

Якщо  $A = A^T > 0$ , то можна знайти мінімальне власне значення:

$$\lambda_{min}(A) = \lambda_{max}(A) - \lambda_{max}(B)$$

Якщо скористатися властивістю норм:  $\lambda_{\max}(A) \leq \|A\|_{\infty}$ , то можна уникнути знаходження  $\lambda_{\max}(A)$ :

$$\lambda_{\min}(A) = \|A\|_{\infty} - \lambda_{\max}(B), \quad \text{де } B = \|A\|_{\infty}E - A$$

### Метод обертань (Якобі)

Метод Якобі використовують, якщо матриця  $A$  є симетричною, тобто  $A = A^T$ . Тоді за допомогою ортогональних перетворень матриця  $A$  зводиться до діагонального вигляду, елементи діагоналі якої будуть відповідати наближеним власним значенням вихідної матриці. Покладемо  $A_0 = A$ . Ітераційний процес має вигляд:

$$A_{k+1} = U_k A_k U_k^T,$$

де  $U_k$  - матриця обертань:

$$U_k = \begin{array}{ccccccccc} & & i_k & & & j_k & & & \\ & 1 & \dots & 0 & \dots & 0 & \dots & 0 & \\ & \dots & \\ 0 & \dots & \cos(\varphi_k) & \dots & \sin(\varphi_k) & \dots & 0 & i_k \\ \dots & \dots \\ 0 & \dots & -\sin(\varphi_k) & \dots & \cos(\varphi_k) & \dots & 0 & j_k \\ \dots & \dots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 & \end{array}$$

$\varphi_k$  - кут обертань:

$$\varphi_k = \frac{1}{2} \arctan \frac{2a_{i_k j_k}^k}{a_{i_k i_k}^k - a_{j_k j_k}^k}$$

$i_k, j_k$  - номери рядочка та стовпчика в матриці  $A_k$ :

$$a_{i_k j_k}^k = \max_{i \neq j} |a_{ij}^k|, \quad i = \overline{1, n}, \quad j = \overline{2 + 1, n}$$

Умова припинення:

$$t(A_{k+1}) = \sum_{i,j=1 (i \neq j)}^n a_{ij}^2 \leq \varepsilon$$

Після виконання цієї умови діагональні елементи матриці  $A_{k+1}$  є

наближеним валсним значенням з точністю  $\varepsilon$ :

$$\lambda_i \approx a_{ii}^{k+1}, \quad i = \overline{1, n},$$

при чому швидкість збіжності:

$$t(A_{k+1}) \leq qt(A_k); \quad q = 1 - \frac{2}{n(n-1)}$$

Власним векторам  $\lambda_i, i = \overline{1, n}$  відповідають власні вектори

$(u_{1i}, \dots, u_{ii}, \dots, u_{ni})^T$ , які є стовпцями матриці U:

$$U = \prod_{k=1}^n U_k = \begin{matrix} & u_{11} & \cdots & u_{1i} & \cdots & u_{1n} \\ & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_{i1} & \cdots & u_{ii} & \cdots & u_{in} \\ & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_{n1} & \cdots & u_{ni} & \cdots & u_{nn} \end{matrix}$$

Також можемо використовувати ітераційни процес вигляду

$A_{k+1} = U_k^T A_k U_k$ , але в такому випадку матриця обертань буде:

$$\begin{array}{ccccccccc}
& & i_k & & & j_k & & & \\
& 1 & \dots & 0 & \dots & 0 & \dots & 0 & \\
& \dots & \\
0 & \dots & \cos(\varphi_k) & \dots & \sin(\varphi_k) & \dots & 0 & & i_k \\
\\
U_k = & \dots \\
& 0 & \dots & -\sin(\varphi_k) & \dots & \cos(\varphi_k) & \dots & 0 & j_k \\
& \dots \\
0 & \dots & 0 & \dots & 0 & \dots & 0 & \dots & 1
\end{array}$$

### Модифікований метод Ньютона

$$A_k = \overline{F}'(\bar{x}_k)$$

$$\begin{array}{cccc}
\frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\
\overline{F}'(\bar{x}) = & \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \dots & \frac{\partial f_2(x)}{\partial x_n} \\
& \dots & \dots & \dots & \dots \\
\frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \dots & \frac{\partial f_n(x)}{\partial x_n}
\end{array}$$

Ітераційний процес:  $\bar{x}^{k+1} = \bar{x}^k - A_0^{-1} \overline{F}'(\bar{x}^k)$

Обирається початкове наближення  $x^0$ , для якого обчислюється матриця

$$\text{Якобі: } A_0 = \overline{F'}(\overline{x}_0)$$

$$\text{Умова припинення: } ||\overline{x}^{k+1} - \overline{x}^k|| \leq \varepsilon$$

# Хід роботи

Мова реалізації: Python

## Завдання №1

### Степеневий метод

Спочатку перевіряємо умови застосування методу (симетричність і позитивна визначеність матриці) :

```
#Check condition to use the power method
#A=A^T >0
if not np.allclose(A, A.T):
    print("Matrix A isn't symmetrical. Method can't be
          applied.")
    return

matrix_minor = np.zeros(n)
print("Matrix minors:")
for i in range(1, n + 1):
    minor = np.linalg.det(A[:i, :i])
    matrix_minor[i - 1] = round(minor)
    print(f" |A{i}|={matrix_minor[i-1]}")
    if minor <= 0:
        print(f"Matrix is not positive definite (|A{i}| =
              {round(minor)}))")
        return None

print("\nConditions for power method is satisfied!")
```

```
Matrix minors:
|A1|=4.0
|A2|=11.0
|A3|=28.0
|A4|=48.0
```

```
Conditions for power method is satisfied!
```

Обчислюємо матричну норму для А, та обраховуємо за формулою матрицю В:

```

Max ||A|| = 6

B = ||A||E - A :
[[ 2. -1.  0. -1.]
 [-1.  3. -2.  0.]
 [ 0. -2.  2.  0.]
 [-1.  0.  0.  4.]]

```

Знайшовши матрицю B, переходимо до занходження її найбільшого власного числа степеневим методом:

- обчислюємо новий вектор, результат множення матриці A на поточний вектор x
- lam\_new = x\_new[m] / x[m] – приблизне значення власного числа визначається як відношення відповідного елемента нового та старого вектора.
- x\_new\_norm = np.linalg.norm(x\_new) – обчислюємо норму нового вектора.
- x = x\_new / x\_new\_norm – нормалізуємо вектор.
- if np.abs(lam\_new - lam\_old) <= eps - перевіряємо умову збіжності

```

-----
Iteration 1:
A * x = [0. 0. 0. 3.]
λ_new = x_new[3] / x[3] = 3.000
||x_new|| = 3.000
Normalized vector e1 = [0. 0. 0. 1.]

Iteration 2:
A * x = [-1. 0. 0. 4.]
λ_new = x_new[3] / x[3] = 4.000
||x_new|| = 4.123
Normalized vector e2 = [-0.243 0. 0. 0.97]

Iteration 3:
A * x = [-1.455 0.243 0. 4.123]
λ_new = x_new[3] / x[3] = 4.250
||x_new|| = 4.379
Normalized vector e3 = [-0.332 0.055 0. 0.942]

```

...

```

Iteration 59:
A * x = [-1.846  3.374 -2.312  2.012]
λ_new = x_new[3] / x[3] = 4.917
||x_new|| = 4.918
Normalized vector e59 = [-0.375  0.686 -0.47   0.409]

Iteration 60:
A * x = [-1.846  3.374 -2.312  2.012]
λ_new = x_new[3] / x[3] = 4.917
||x_new|| = 4.918
Normalized vector e60 = [-0.375  0.686 -0.47   0.409]

-----
Converged after 60 iterations.
Dominant eigenvalue λ ≈ 4.9173
=====
```

Весь ітераційни процес додаю в окремий файл(power\_method\_iter.txt)

Знайшовши максимальне власне число матриці B, переходимо до знаходження мінімального власного числа матриці A:

```

max_lam(B)= 4.917286195679672
min_lam(A)=| |A| |-max_lam_B
min_lam(A)=6-4.917286195679672
min_lam(A)=1.0827138043203277
```

Коректність результату можемо перевірити за допомгою наступного реалізованого методу, а саме методу обертань (Якобі)

### Метод обертань (Якобі)

Для застосування даного методу, перевіримо матрицю на симетричність, і задаємо початкову матрицю власних векторів V=I:

```

if not np.allclose(A, A.T):
    print("Matrix A isn't symmetrical. Jacobi method can't
    be applied.")
    return None, None

n = A.shape[0]
V = np.eye(n) # for eigenvectors
```

На кожній ітерації знаходимо найбільший за модулем позадіагональний елемент  $a_{pq}$ :

```
for k in range(1, max_iter + 1):
    p, q = 0, 1
    max_val = 0.0
    for i in range(n):
        for j in range(i + 1, n):
            if abs(A[i, j]) > abs(max_val):
                max_val = A[i, j]
                p, q = i, j
```

- Знаходимо  $\varphi_k$ ,  $\sin(\varphi_k)$ ,  $\cos(\varphi_k)$
- Формуємо матрицю  $U_k$  та оновлюємо матрицю A за формулою
- Онволюємо матрицю власних векторів
- Перевіряємо умову зупинки

```
===== JACOBI ROTATION METHOD =====
Initial matrix A:
[[4 1 0 1]
 [1 3 2 0]
 [0 2 4 0]
 [1 0 0 2]]
Eps: 1e-08
Maximum iterations: 100
-----
```

```
Iteration 1:
Pivot indices (p, q) = (1, 2)
a[p,q] = 2.000000
off-diagonal norm = 3.464102
phi_k = 0.5 * arctan(2 * a[1,2] / (a[1,1] - a[2,2]))
phi_k = -0.662909
cos(phi_k) = 0.788205
sin(phi_k) = -0.615412

Rotation matrix U_k:
[[ 1.          0.          0.          0.          ]
 [ 0.          0.788205  0.615412  0.          ]
 [ 0.          -0.615412  0.788205  0.          ]
 [ 0.          0.          0.          1.          ]]
-----
```

```
Matrix A after rotation:
[[ 4.          0.7882  0.6154  1.          ]
 [ 0.7882  1.4384  -0.        0.          ]
 [ 0.6154  -0.        5.5616  0.          ]
 [ 1.          0.        0.          2.          ]]
-----
```

```
Iteration 14:  
Pivot indices (p, q) = (2, 3)  
a[p,q] = -0.000009  
Off-diagonal norm = 1.2e-05  
 $\phi_k = 0.5 * \arctan(2 * a[2,3] / (a[2,2] - a[3,3]))$   
 $\phi_k = -0.000002$   
 $\cos(\phi_k) = 1.000000$   
 $\sin(\phi_k) = -0.000002$ 
```

```
Rotation matrix U_k:  
[[ 1.        0.        0.        0.        ]  
 [ 0.        1.        0.        0.        ]  
 [ 0.        0.        1.        0.000002]  
 [ 0.        0.        -0.000002  1.        ]]
```

```
Matrix A after rotation:  
[[ 4.3343  0.        -0.        -0.        ]  
 [-0.        1.0822  0.        -0.        ]  
 [-0.        0.        5.8274  0.        ]  
 [-0.        -0.        -0.        1.7561]]
```

```
-----  
Iteration 15:  
Pivot indices (p, q) = (1, 3)  
a[p,q] = -0.000000  
Off-diagonal norm = 0.0
```

```
Convergence criterion satisfied (off-diagonal norm < eps).
```

```
-----  
Finished after 15 iterations.
```

```
Approximated eigenvalues:  
[4.33432  1.082167 5.827408 1.756106]
```

```
Corresponding eigenvectors (columns):  
[[ 0.814 -0.3753  0.3935 -0.2041]  
 [-0.0766  0.6861  0.6162 -0.379 ]  
 [-0.4581 -0.4703  0.6744  0.3378]  
 [ 0.3487  0.4089  0.1028  0.837 ]]
```

Повний ітераційний процес додаю у окремий файл (jacobi\_iter.txt)

Отримавши наближені значення власних чисел матриці, можемо переконатися у коректності виконання, як степеневого методу, так і методу обертань

## Завдання №2

### Модифікований метод Ньютона

Задамо початкове наближення  $\bar{x}_0$  і обчислюємо матрицю Якобі  $A_0 = F'(\bar{x}_0)$

```
Initial Jacobian A0:
[[ 0.8253 -1.
  [ 3.       0.     ]]

=====
Modified Newton Method =====
Initial guess x0 = [0. 0.]
Epsilon = 1e-06, Max iterations = 50
-----
```

```
def F(x):
    x1, x2 = x
    return np.array([
        np.sin(x1 - 0.6) - x2 - 1.6,
        3*x1 - np.cos(x2) - 0.9
    ])

def jacobian(x):
    x1, x2 = x
    return np.array([
        [np.cos(x1 - 0.6), -1],
        [3, np.sin(x2)]
    ])
```

На кожній ітерації обчислюємо  $F(x_k)$  і обраховуємо  $A^{-1} * F(x_k)$

Оновлюємо наближення:  $x_{k+1} = x_k - A^{-1} * F(x_k)$

Перевіряємо умову зупинки:  $\|x_{k+1} - x_k\| < \varepsilon$

```

Iteration 1:
x1 = [ 0.  0. ]
F(x1) = [-2.1646 -1.9    ]
A_inv*F(x1) = [-0.6333  1.6419]
x2 = [ 0.6333 -1.6419]
||x2 - x1|| = 1.759842e+00

```

```

Iteration 2:
x2 = [ 0.6333 -1.6419]
F(x2) = [ 0.0753  1.0711]
A_inv*F(x2) = [ 0.357   0.2194]
x3 = [ 0.2763 -1.8613]
||x3 - x2|| = 4.190542e-01

```

...

```

Iteration 13:
x13 = [ 0.1511 -2.034 ]
F(x13) = [-0.  0.]
A_inv*F(x13) = [0.  0.]
x14 = [ 0.1511 -2.034 ]
||x14 - x13|| = 1.826207e-06

```

```

Iteration 14:
x14 = [ 0.1511 -2.034 ]
F(x14) = [-0.  0.]
A_inv*F(x14) = [0.  0.]
x15 = [ 0.1511 -2.034 ]
||x15 - x14|| = 5.792239e-07

```

```

Convergence criterion satisfied!
Solution found: x ≈ [ 0.1511 -2.034 ]

```

Повний ітераційний процес додаю у окремий файл (modif\_newton\_iter.txt)

Розв'язок :  $x = (0.1511; -2.034)$