

Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Чисельні методи в інформатиці

Лабораторна робота №3
“Наближені методи розв’язання систем нелінійних рівнянь
та задач власні значення”

Варіант №7

Виконала студентка групи ІПС-31
Сенечко Дана Володимирівна

Київ - 2025

Постановка задачі

Знайти найменше власне значення степеневим методом та наближення до всіх власних значень методом обертань Якобі (або виконати 3-4 ітерації):

$$\begin{matrix} 5 & 0 & 2 & 1 \\ 0 & 4 & 0 & 1 \\ 2 & 0 & 2 & 0 \\ 1 & 1 & 0 & 3 \end{matrix}$$

Розв'язати (або виконати 5 ітерацій) методом Ньютона:

$$\begin{cases} 5x - 6y + 20\lg x = -16, \\ 2x + y - 10\lg y = 4. \end{cases}$$

Теоретичні відомості

Степеневий метод.

Нехай $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$. Будемо також шукати максимальне власне значення λ_1 . Початкове наближення \bar{x}^0 обираємо довільним, але $\bar{x}^0 \neq \bar{0}$.

Ітераційний процес має вигляд:

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

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

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

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

де $B = \lambda_{max}(A)E - A$, а E – одинична матриця.

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

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

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 наближенням 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:

$$U_k = \begin{pmatrix} & i_k & j_k & & & \\ 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \cos \varphi_k & \dots & \sin \varphi_k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & -\sin \varphi_k & \dots & \cos \varphi_k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} i_k \\ j_k \end{matrix}$$

φ_k – кут обертань:

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

i_k та 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:

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

Після виконання цієї умови діагональні елементи матриці A_{k+1} є наближеними власними значеннями з точністю ε :

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

при чому швидкість збіжності:

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

Власним векторам λ_i , $i = \overline{1, n}$, відповідають власні вектори $(u_{1i}, \dots, u_{ii}, \dots, u_{ni})^T$, які є стовпцями матриці U :

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

Зauważення. Можна використовувати ітераційний процес вигляду: $A_{k+1} = U_k^T A_k U_k$, але тоді матриця обертань буде:

$$U_k = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \cos \varphi_k & \dots & -\sin \varphi_k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \sin \varphi_k & \dots & \cos \varphi_k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix}.$$

Метод Ньютона.

Розв'язання системи нелінійних рівнянь методом Ньютона зводиться до розв'язання систем лінійних алгебраїчних рівнянь.

Алгоритм методу Ньютона. Обираємо початкове наближення \bar{x}^0 . На кожній ітерації обчислюється матриця Якобі:

$$A_k = \bar{F}'(\bar{x}_k); \quad \bar{F}'(\bar{x}) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \dots & \frac{\partial f_2(x)}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \dots & \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix}.$$

Ітераційний процес має вигляд:

$$\bar{x}^{k+1} = \bar{x}^k - \bar{z}^k,$$

де z_k знаходять із системи лінійних алгебраїчних рівнянь:

$$A_k \bar{z}^k = \bar{F}(\bar{x}^k).$$

Умова припинення: $\|\bar{z}^k\| \leq \varepsilon$.

Для того, щоб метод Ньютона збігався, потрібно щоб виконувались умови таких теорем.

Теорема 1. (про збіжність методу Ньютона)

Нехай $\delta_a = \{\bar{x} : \|\bar{x} - \bar{x}^*\| < a\}$ та при деяких a, a_1, a_2 :
 $0 < a, 0 \leq a_1, a_2 < \infty$ виконуються умови:

- 1) $\|(\bar{F}'(\bar{x}))^{-1}\| \leq a_1, x \in \delta_a,$
- 2) $\|\bar{F}(\bar{x}_1) - \bar{F}(\bar{x}_2) - \bar{F}'(\bar{x}_2)(\bar{x}_1 - \bar{x}_2)\| \leq a_2 \|\bar{x}_1 - \bar{x}_2\|^2,$ при чому $\bar{x}_1, \bar{x}_2 \in \delta_a.$
- 3) $\bar{x}^0 \in \delta_b, b = \min(a, c^{-1}), c = a_1 a_2.$

Тоді ітераційний процес методу Ньютона для системи нелінійних рівнянь збігається та має місце оцінка точності:

$$\|\bar{x}^n - \bar{x}^*\| \leq c^{-1}(c \|\bar{x}^0 - \bar{x}^*\|)^{2^n}.$$

Теорема 2. (про збіжність методу Ньютона)

Якщо в області $G \subset R^n$ функції $f_i(x_1, x_2, \dots, x_n) \in C^2(G),$
 $|\frac{\partial^2 f_i}{\partial x_j^2}| \leq L$, в точці $\bar{x}^0 \in G$ матриця $\left(\frac{\partial f_i}{\partial x_i}\right)_{i,j=1}^n$ – невиродже-
на, $\left\| \left(\frac{\partial f_i}{\partial x_i}\right)_{i,j=1}^n \right\|^{-1} \leq M, |f_i(x_1^0, x_2^0, \dots, x_n^0)| \leq \delta$ та вико-
нується умова

$$h = M^2 L \delta n^2 \leq 1/2,$$

тоді система заданих нелінійних рівнянь має розв'язок в області $\|\bar{x} - \bar{x}^0\|_1 \leq \frac{1 - \sqrt{1 - 2h}}{h} \delta$ та має місце оцінка:

$$\|\bar{x}^n - \bar{x}^*\|_1 \leq \frac{1}{2^{n-1}} (2h)^{2^n-1} \delta.$$

Хід роботи

Мова програмування – Python. Використана бібліотека: numpy.

Завдання №1.

Степеневий метод.

1) Перевіряємо умови застосування методу (симетричність, позитивна визначеність матриці):

```
def min_eigenvalue(A):  
    n = A.shape[0]  
    # 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 the power method are met.")

Matrix minors:
|A1| = 5.0
|A2| = 20.0
|A3| = 24.0
|A4| = 58.0

Conditions for the power method are met.

```

2) Обчислюємо матричну норму для матриці A, за формулою
обираємо матрицю B:

```

Max ||A|| = 8

B = ||A||E - A :
[[ 3.  0. -2. -1.]
 [ 0.  4.  0. -1.]
 [-2.  0.  6.  0.]
 [-1. -1.  0.  5.]]

```

3) Знайшовши матрицю B переходимо до знаходження найбільшого
власного числа степеневим методом – починаємо ітераційний процес:

```

Power Method
Initial vector x0: [1. 1. 1. 1.]
Eps: 0.0001
Maximum iterations: 50

Iteration 1:
A * x = [0. 3. 4. 3.]
λ_new = x_new[3] / x[3] = 3.000
||x_new|| = 5.831
Normalized vector e1 = [0.      0.514  0.686  0.514]

Iteration 2:
A * x = [-1.886  1.543  4.116  2.058]
λ_new = x_new[3] / x[3] = 4.000
||x_new|| = 5.207
Normalized vector e2 = [-0.362  0.296  0.79   0.395]

...
Iteration 33:
A * x = [-3.35 -0.596  5.974  1.859]
λ_new = x_new[3] / x[3] = 7.122
||x_new|| = 7.122
Normalized vector e33 = [-0.47 -0.084  0.839  0.261]

Converged after 33 iterations.

```

Dominant eigenvalue $\lambda \approx 7.1219$

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

```
max_lam(B) = 7.121922263993244
min_lam(A) = ||A|| - max_lam_B
min_lam(A) = 8 - 7.121922263993244
min_lam(A) = 0.8780777360067562
```

Коректність результату перевіримо далі, за допомогою методу Якобі.

Метод обертань (Якобі).

1) Перевіряємо 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
```

2) 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
```

3) Знаходимо φ_q , $\sin(\varphi_k)$, $\cos(\varphi_k)$, формуємо матрицю U_k та оновлюємо матрицю A за формулою. Далі також оновлюємо матрицю власних векторів і перевіряємо умову зупинки на кожній ітерації:

```
Jacobi Rotation Method
Initial matrix A:
[[5 0 2 1]
 [0 4 0 1]
 [2 0 2 0]
 [1 1 0 3]]
Eps: 0.0001
Maximum iterations: 50
```

```

Iteration 1:
Pivot indices (p, q) = (0, 2)
a[p,q] = 2.000000
Off-diagonal norm = 3.464102
φ_k = 0.5 * arctan(2 * a[0,2] / (a[0,0] - a[2,2]))
φ_k = 0.463648
cos(φ_k) = 0.894427
sin(φ_k) = 0.447214

Rotation matrix U_k:
[[ 0.894427  0.        -0.447214  0.        ]
 [ 0.         1.        0.        0.        ]
 [ 0.447214  0.        0.894427  0.        ]
 [ 0.         0.        0.        1.        ]]

Matrix A after rotation:
[[ 6.        0.        0.        0.8944]
 [ 0.        4.        0.        1.        ]
 [-0.        0.        1.        -0.4472]
 [ 0.8944  1.        -0.4472  3.        ]]

```

...

```

Iteration 12:
Pivot indices (p, q) = (1, 2)
a[p,q] = 0.000236
Off-diagonal norm = 0.000336
φ_k = 0.5 * arctan(2 * a[1,2] / (a[1,1] - a[2,2]))
φ_k = 0.000065
cos(φ_k) = 1.000000
sin(φ_k) = 0.000065

```

```

Rotation matrix U_k:
[[ 1.        0.        0.        0.        ]
 [ 0.        1.        -0.000065  0.        ]
 [ 0.        0.000065  1.        0.        ]
 [ 0.        0.        0.        1.        ]]

Matrix A after rotation:
[[ 6.2848 -0.        -0.        -0.        ]
 [-0.        4.5045  0.        -0.        ]
 [-0.        0.        0.8784  0.        ]
 [-0.        -0.        0.        2.3323]]

```

```

Iteration 13:
Pivot indices (p, q) = (1, 3)
a[p,q] = -0.000027
Off-diagonal norm = 3.8e-05

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

```

Finished after 13 iterations.

Approximated eigenvalues:
[6.284774 4.504461 0.878421 2.332344]

Corresponding eigenvectors (columns):
[[0.856 -0.2075 -0.4704 -0.055]
 [0.1316 0.8608 -0.0837 -0.4845]
 [0.3995 -0.1657 0.8388 -0.3307]
 [0.3006 0.4342 0.2611 0.808]]

Отримавши наближені значення власних чисел матриці, можемо переконатись в правильності виконання обох методів. (В методі Якобі в Approximated eigenvalues мінімальне значення: 0.8784, в степеневому отримали $\min_lam(A) = 0.8780$, значення достатньо близькі).

Завдання №2.

Метод Ньютона.

1) Задаємо початкове наближення $\overline{x}_0 = [0.5, 0.5]$ і обчислюємо матрицю Якобі:

```
Newton's Method
Initial approximation x0: [0.5 0.5]
Eps: 0.0001
Maximum iterations: 50

# func vector F(x) = (f1, f2)^T , f1 = 0 and f2 = 0
def F(x_vec):
    x, y = x_vec
    # check for domain validity before calculation
    if x <= 0 or y <= 0:
        return None

    # f1(x, y) = 5x - 6y + 20*log10(x) + 16
    f1 = 5 * x - 6 * y + 20 * np.log10(x) + 16
    # f2(x, y) = 2x + y - 10*log10(y) - 4
    f2 = 2 * x + y - 10 * np.log10(y) - 4

    return np.array([f1, f2])

# Jacobian matrix J(x) = F'(x)
def J(x_vec):
    x, y = x_vec
    # check for domain validity before calculation
    if x <= 0 or y <= 0:
        pass

    # we use ln(10) to convert the derivative of log10(x)
    ln10 = np.log(10)

    # df1/dx = 5 + 20 / (x * ln10)
    df1_dx = 5 + 20 / (x * ln10)
    # df1/dy = -6
    df1_dy = -6
    # df2/dx = 2
    df2_dx = 2
    # df2/dy = 1 - 10 / (y * ln10)
    df2_dy = 1 - 10 / (y * ln10)
```

```

    return np.array([
        [df1_dx, df1_dy],
        [df2_dx, df2_dy]
    ])

```

2) Ітераційний процес:

```

Iteration 1:
 Jacobian Matrix J_k:
 [[22.371779 -6.      ]
 [ 2.          -7.68589 ]]
 Function Vector F_k = F(x_k): [9.4794  0.5103]
 Linear system J_k * z_k = -F_k solved for z_k: [-0.436368 -0.047156]
 Max norm of correction ||z_k||_inf = 0.436368
 New approximation x_k+1 = x_k + z_k: [0.063632  0.452844]

Iteration 2:
 Jacobian Matrix J_k:
 [[141.502704 -6.      ]
 [ 2.          -8.59038 ]]
 Function Vector F_k = F(x_k): [-10.325444   0.020622]
 Linear system J_k * z_k = -F_k solved for z_k: [0.0738   0.019583]
 Max norm of correction ||z_k||_inf = 0.073800
 New approximation x_k+1 = x_k + z_k: [0.137432  0.472427]

...
Iteration 5:
 Jacobian Matrix J_k:
 [[48.898815 -6.      ]
 [ 2.          -7.90072 ]]
 Function Vector F_k = F(x_k): [-0.011053  0.000063]
 Linear system J_k * z_k = -F_k solved for z_k: [0.000234 0.000067]
 Max norm of correction ||z_k||_inf = 0.000234
 New approximation x_k+1 = x_k + z_k: [0.198096  0.487999]

Iteration 6:
 Jacobian Matrix J_k:
 [[48.846894 -6.      ]
 [ 2.          -7.899492]]
 Function Vector F_k = F(x_k): [-0.000006  0.      ]
 Linear system J_k * z_k = -F_k solved for z_k: [0. 0.]
 Max norm of correction ||z_k||_inf = 0.000000

Converged after 6 iterations.
Final solution x* ≈ [0.198096  0.487999]

```

Перевіримо, підставивши отримані значення:

$$5 * 0.198096 - 6 * 0.487999 + 20 \lg(0.198096) \approx -16$$

$$2 * 0.198096 + 0.487999 - 10 \lg(0.487999) \approx 4$$

Отже, отримані значення правильні.