МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №5
Вариант 5
«Метод Данилевского и итерационный степенной метод»

Выполнил: Благодарный Артём Андреевич,
студент 3 курса, 3 группы
Дисциплина: «Численные методы»
Преподаватель: Будник А.М.

Минск, 2024

1) Постановка задачи

Необходимо найти собственные значения и собственные векторы матрицы A:

$$\mathrm{A}-\lambda E_n=\begin{bmatrix} a_{11}-\lambda & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & a_{23} & \cdots & \\ \vdots & & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn}-\lambda \end{bmatrix}, \qquad \det(A-\lambda E_n)=(-1)^n\,\mathrm{Pn}(\lambda)$$

Для этого требуется:

1. Найти с помощью метода Данилевского форму Фробениуса, характеристический многочлен, r1 = p1 – SpA и r2=p5 – detA.
2. С помощью степенного метода найти максимальное собственное значение, определить количество итераций для eps=1e-5.
3. С помощью метода Данилевского найти собственный вектор, соответствующий максимальному собственному значению, найти невязку собственного значения и собственного вектора.

2)Алгоритм решения

Метод Данилевского является прямым методом решения полной задачи собственных значений (т.е. позволяет весь спектр). Метод построен на том факте, что преобразование подобия $S^{-1}AS$ не изменяет характеристического многочлена. С помощью такого преобразования исходная матрица A приводится к канонической форме Фробениуса:

$$\Phi=\begin{bmatrix} p_1 & p_2 & p_3 & \cdots & p_{n-1} & p_n \\ 1 & 0 & 0 & \cdots & & \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \quad |\Phi-\lambda E_n|=\begin{vmatrix} p_1-\lambda & p_2 & p_3 & \cdots & p_{n-1} & p_n \\ 1 & -\lambda & 0 & \cdots & & \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -\lambda \end{vmatrix}$$

После разложения определителя получим:

$$det|\Phi-\lambda E_n|=\ (-1)^n(\,\lambda^n-p_n\lambda^{n-1}-\ldots-p_1\lambda\ \ -p_0)=(-1)^n\,\mathrm{Pn}(\lambda)$$

Матрица A приводится к Φ, в результате последовательного домножения справа на M_{n-1} и слева на M_{n-1}^{-1}, а S можно получить как $S = M_{n-1}\,M_{n-2}..\,M_{n-1}$

$$M_{n-1}=\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ -\dfrac{a_{n1}}{a_{nn-1}} & -\dfrac{a_{n2}}{a_{nn-1}} & -\dfrac{a_{n3}}{a_{nn-1}} & \cdots & \dfrac{1}{a_{nn-1}} & -\dfrac{a_{nn}}{a_{nn-1}} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$M_{n-1}^{-1}=\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n1} & a_{n1} & \cdots & a_{n1} & a_{n1} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Из $\mathrm{P_n}(\lambda)=0$ находим λ_i, далее решая $\Phi y=\ \lambda_i y,\ i=\overline{1,n}$ находим собственный вектор матрицы Φ: $y = (\lambda_i^{n-1}, \lambda_i^{n-2}, \ldots \lambda_i\ , 1)^T$, далее находим собственные векторы матрицы A из $x = Sy = M_{n-1}\,M_{n-2}..\,M_{n-1}y$.

Алгоритм степенного метода

Пусть y^0 – произвольный ненулевой вектор (например, $y^0=[1, 0, \ldots, 0]$). Основные вычисления метода – это реализация итерационного процесса

$$y^{k+1} = A\,y^k, \quad k = 0, 1, 2, \ldots \tag{1}$$

Получим представление вектора y^k, которое понадобится для исследования поведении y^k при больших значениях k. Для этого разложим y^0 по базису из собственных векторов $\{x_1, x_2,\ldots, x_n\}$ (x_i – собственный вектор матрицы A, отвечающий собственному значению λ_i):

$$y^0 = \alpha_1 x_1+\alpha_2 x_2+\ldots+\alpha_n x_n. \tag{2}$$

Здесь α_i – некоторые числа, среди которых могут быть, вообще говоря, равные нулю. Так как (следует из (2))

$$A^k y^0 = \alpha_1 A^k x_1+\alpha_2 A^k x_2+\ldots+\alpha_n A^k x_n,$$

то, с учетом $y^k=A^k y^0$ (следует из (1)) и $A^k x_i=\lambda_i^k x_i$ (следует из $Ax_i=\lambda_i x_i$), получим

$$y^k=\alpha_1 \lambda_1^k x_1+\alpha_2 \lambda_2^k x_2+\ldots+\alpha_n \lambda_n^k x_n. \tag{3}$$

Если $|\lambda_1|>1$, $\alpha_1\neq 0$, то при вычислении последовательности (1) на компьютере координаты вектора y^k могут сильно расти (напомним, $|\lambda_1|\geq|\lambda_i|$), что может привести к переполнению. Если $|\lambda_1|<1$, $\alpha_1\neq 0$, то координаты вектора y^k будут сильно убывать, что может привести к машинному нулю. Поэтому на практике требуется производить нормировку: $u^0=y^0$, $u^k=\dfrac{Au^{k-1}}{\left\|Au^{k-1}\right\|}$.

Для организации нормированных вычислений удобно использовать две одновременно вычисляемые последовательности:

$$u^0=y^0,$$

$$v^k = Au^{k-1}, \quad u^k = \frac{v^k}{\left\|v^k\right\|}, \tag{4}$$

Листинг

```
import numpy as np
from tabulate import tabulate
from sympy import Symbol, solve

A = np.array([[0.8894, 0.0000, -0.2323, 0.1634, 0.2723],
              [-0.0545, 0.5808, 0.0000, -0.1107, 0.0363],
              [0.0182, -0.1634, 1.0527, 0.0200, 0.0635],
              [0.0545, 0.0000, -0.1325, 1.0527, 0.0000],
              [0.0363, -0.0545, 0.2632, -0.0218, 0.7623]])

A = A.T @ A

def pretty_print(X, p, r1, r2, eigenvalue, eigenvector, r_eigenvalue,
r_eigenvector, c=None):
    p *= -1
    x = [f"x^{i}" if i != 1 else "x" for i in range(5, 0, -1)]
    polynom = "p(x)="
    for i, j in zip(x, p):
        polynom += i
        polynom += f" - {-round(j, 3)}" if np.sign(j) == -1 else f" + {round(j,
3)}"
    print(f'1)Frobenius normal form:\n{tabulate(X, tablefmt="grid",
floatfmt=".3f")}\n',\
        f'2)Characteristic polynomial:\n{polynom}\n',\
        f'3)r1 = p1 - SpA = {r1:.3e}\n',\
        f'4)r2 = p5 - detA = {r2:.3e}\n',\
        f'5)max eigenvalue: {eigenvalue}\n',\
        f'6)eigenvector:\n{tabulate([eigenvector], tablefmt="grid",
floatfmt=".5f")}\n',\
        f'7)r of eigenvalue: {r_eigenvalue}\n',\
        f'8)r of eigenvector:\n{tabulate([r_eigenvector], tablefmt="grid",
floatfmt=".3e")}\n',\
          "" if c is None else f'9)count of iterations: {c}')

def danilevsky_method(A: np.ndarray):
    X = A.copy()
    n = X.shape[0]
    s = np.eye(n)
    n -= 1
    for i in np.arange(n):
        ones_left = np.eye(n+1)
        ones_left[n-1-i] = X[n-i]
        ones_right = ones_left.copy()
        ones_right[n-1-i] /= -X[n-i, n-1-i]
        ones_right[n-1-i, n-1-i] = 1 / X[n-i, n-1-i]
        X = ones_left @ X @ ones_right
        s = s @ ones_right
    p = X[0]
    r1 = p[0] - np.trace(A)
    detA = np.linalg.det(A)
    r2 = p[n] - detA
```

```
    return X, r1, r2, s, p

def power_iteration(A: np.ndarray, num_iter: int=1000, tol: float=1e-5):
    n = A.shape[0]
    u = np.zeros(n)
    u[0] = 1
    prev_eigenvalue = 0
    c = 0
    for k in range(1, num_iter + 1):
        c += 1
        v = A @ u
        v_norm = np.linalg.norm(v, np.inf)
        u = v / v_norm
        eigenvalue = v_norm
        if abs(eigenvalue - prev_eigenvalue) < tol:
            break
        prev_eigenvalue = eigenvalue
    return eigenvalue, u, c

def danilevsky_power_method(A: np.ndarray, type_eigenvector='danilevsky'):
    X = A.copy()
    n = X.shape[0]
    X, r1, r2, s, p = danilevsky_method(X)
    eigenvalue, u, c = power_iteration(X)
    r_eigenvalue = np.sum([p[n-1-i] * (eigenvalue ** i) for i in range(n)]) -
eigenvalue ** (n)
    if type_eigenvector == 'danilevsky':
        y = np.array([eigenvalue ** i for i in np.arange(n-1, -1, -1)])
        eigenvector = s @ y
        r_eigenvector = X @ eigenvector - eigenvalue * eigenvector
    else:
        r_eigenvector = X @ u - eigenvalue * u
        eigenvector = u
    return X, p, r1, r2, c, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector

X, p, r1, r2, c, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector =
danilevsky_power_method(A)
pretty_print(X, p, r1, r2, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector, c)
```

Результаты и их анализ

Матрица A:

0.88940	0.00000	-0.23230	0.16340	0.27230
-0.05450	0.58080	0.00000	-0.11070	0.03630
0.01820	-0.16340	1.05270	0.02000	0.06350
0.05450	0.00000	-0.13250	1.05270	0.00000
0.03630	-0.05450	0.26320	-0.02180	0.76230

Матрица A.T @ A:

0.79862	-0.03661	-0.18512	0.20831	0.26903
-0.03661	0.36700	-0.18636	-0.06637	-0.03084
-0.18512	-0.18636	1.24897	-0.16212	0.20423
0.20831	-0.06637	-0.16212	1.14801	0.02513
0.26903	-0.03084	0.20423	0.02513	0.66060

Результаты:

1)Frobenius normal form:

-4.223	6.614	-4.722	1.511	-0.174
1.000	0.000	0.000	0.000	0.000
0.000	1.000	0.000	0.000	0.000
0.000	-0.000	1.000	-0.000	0.000
0.000	-0.000	0.000	1.000	-0.000

2)Characteristic polynomial:
p(x)=x^5 - 4.223x^4 + 6.614x^3 - 4.722x^2 + 1.511x - 0.174
3)r1 = p1 - SpA = 0.000e+00
4)r2 = p5 - detA = -2.776e-17
5)max eigenvalue: 1.4875417189615943
6)eigenvector:

-5.44753	-1.34242	12.10364	-8.78724	1.00000

7)r of eigenvalue: 4.440892098500626e-15
8)r of eigenvector:

6.458e+01	-3.451e+00	-1.935e+01	2.518e+01	-1.027e+01

9)count of iterations: 42
6)eigenvector:

1.00000	0.67225	0.45192	0.30380	0.20422

7)r of eigenvalue: -8.969077144982407e-06
8)r of eigenvector:

-7.459e-06	0.000e+00	4.242e-06	6.441e-06	7.366e-06

9)count of iterations: 42

Метод Данилевского является точным методом, это подтверждает маленькая погрешность, близкая к машинной ошибке.