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

Лабораторная работа №5
Вариант 4
«Метод Данилевского и итерационный степенной метод»

Выполнил: Снежко Лев Владимирович,
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 А:

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

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 минимальное 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 А приводится к канонической форме Фробениуса:

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

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 $S = M_{n-1} M_{n-2} .. M_{n-1}$

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

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

Из $\mathrm{P_n}(\lambda) = 0$ находим λ_i, далее решая $\Phi y = \lambda_i y,\ i = \overline{1,n}$ находим собственный вектор матрицы Ф: $y = (\lambda_i^{n-1}, \lambda_i^{n-2}, \ldots \lambda_i\ , 1)^T$, 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:

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

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

Для нахождения минимального собственного значения следует применить описанный алгоритм к матрице A^{-1}. Полученное максимальное собственное значение будет равно:

$$\frac{1}{\lambda_{min}}$$

Где λ_{min} – минимальное собственное значение исходной матрицы А.

Листинг

```
import numpy as np

from tabulate import tabulate

from sympy import Symbol, solve

A = np.loadtxt(fname="A.txt", dtype=float)
A = A.T @ A


def format_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)min 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 format_print_danilevsky(eigenvalue, eigenvector, r_eigenvalue, r_eigenvector):
    print(
        '\nDanilevsky eigenvector:\n',\
        f'5)min 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'
    )


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


#find min eigenvalue
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
    u_prev = u
    prev_eigenvalue = 0
    c = 0
    for k in range(1, num_iter + 1):
        c += 1
        v = np.linalg.solve(A, u)
        v_norm = np.linalg.norm(v, np.inf)
        u = v / v_norm
        eigenvalue = v_norm
        if abs(1 – abs(u_prev @ u)) < tol:
            break
        prev_eigenvalue = eigenvalue
        u_prev = u
    return eigenvalue, u, c

def danilevsky_power_method(A: np.ndarray, type_eigenvector='danilevsky'):
    X = np.linalg.inv(A)
    n = X.shape[0]
    X, r1, r2, s, p = danilevsky_method(A)
    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, 1/eigenvalue, eigenvector, r_eigenvalue, r_eigenvector


X, p, r1, r2, c, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector =
danilevsky_power_method(A)
format_print(X, p, r1, r2, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector, c)
_, _, _, _, _, eigenvalue, eigenvector, r_eigenvalue, r_eigenvector =
danilevsky_power_method(A, type_eigenvector='')
format_print_danilevsky(eigenvalue, eigenvector, r_eigenvalue, r_eigenvector)
```

Результаты и их анализ

Матрица входные данные:

A:

0.9546	-0.1256	0.0251	0.0502	0.1758
0.1306	1.4946	0.0000	-0.1005	0.1005
0.0754	0.0000	1.2007	-0.3517	0.2010
-0.1507	0.3165	0.0000	1.1806	-0.0502
0.6280	0.0000	0.2261	0.0251	1.4067

A.T @ A:

1.3511	0.0276	0.2565	-0.1539	1.0871
0.0276	2.3498	-0.0032	0.2171	0.1122
0.2565	-0.0032	1.4934	-0.4154	0.5638
-0.1539	0.2171	-0.4154	1.5308	-0.0959
1.0871	0.1122	0.5638	-0.0959	2.0627

Результаты:

1)Frobenius normal form:

-8.788	28.694	-42.960	29.029	-6.925
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 - 8.788x^4 + 28.694x^3 - 42.96x^2 + 29.029x - 6.925
3)r1 = p1 - SpA = 1.412e-12
4)r2 = p5 - detA = 9.329e-12
5)min eigenvalue: 0.31777861046620404
6)eigenvector:

0.70063	0.09638	0.51124	-0.24419	1.00000

7)r of eigenvalue: -1.9058958855566743e-06
8)r of eigenvector:

3.716e+01	3.973e-01	-1.512e+00	1.280e+00	-3.391e+00

9)count of iterations: 58

Danilevsky eigenvector:
5)min eigenvalue: 0.31777861046620404
6)eigenvector:

1.00000	0.31778	0.10098	0.03209	0.01020

7)r of eigenvalue: -1.9058958855566743e-06
8)r of eigenvector:

-7.342e-08	0.000e+00	9.648e-09	7.055e-09	3.892e-09

Метод Данилевского является точным методом, это подтверждает маленькая погрешность. $|\lambda_1| = 3.14684458$, $|\lambda_2| = 2.41830293$, отсюда $\frac{|\lambda_2|}{|\lambda_1|} = 0.7684850235363426 < 1$ => выполняется достаточное условие сходимости степенного метода.