Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра Систем Штучного Інтелекту

Звіт

до лабораторної роботи № 4

з дисципліни

Чисельні методи

на тему:

“ Методи розв’язування алгебричних проблем власних значень. ”

Варіант №24(Степеневий Метод)

Виконав: студент КН-217

Ратушняк Денис

Прийняла: доцент каф. СШІ

Мочурад Л. І.

Львів – 2022

Мета роботи: набути навиків вирішення часткової та повної проблеми на власні значення та дослідити обчислювальні аспекти цієї проблеми.

Завдання.

1. Складіть програму, яка знаходить найбільше за модулем власне число і відповідний йому власний вектор

а) степеневим методом (РМ - алгоритм);

Точність (в евклідовій нормі). Eps = 10^-6

Представте детально документовану програму з результатами знаходження вказаними методами наближених розв'язків задач на власні значення для таких матриць

$$\begin{pmatrix} 7 & -1 & -2 & 3 \\ -1 & 6 & 0 & 2 \\ -2 & 0 & 5 & 1 \\ 3 & 2 & 1 & 7 \end{pmatrix}, \qquad \begin{pmatrix} 5 & 2 & 0 & -1 \\ 2 & 7 & -3 & 1 \\ 0 & -3 & 9 & 4 \\ -1 & 1 & 4 & 8 \end{pmatrix}$$

Програмна реалізація на мові програмування С++:

```
#include <bits/stdc++.h>
#include <windows.h>
using namespace std;
typedef long double ld;
typedef long long ll;
typedef vector< vector<ld > > matrix;
const ld eps = 1e-6;

ifstream tests("input.txt");
ofstream answers("output.txt");

void print(vector<ld> &A, ll cnt = 3)
{
    cout << fixed << setprecision(cnt);
    for(int i = 0; i < A.size(); ++i) cout << A[i] << " ";
    cout << endl;
    cout << fixed << setprecision(0);
}

void print(vector<ld> &A, ofstream &answers, ll cnt = 3)
{
    answers << fixed << setprecision(cnt);
    for(int i = 0; i < A.size(); ++i) answers << A[i] << " ";
    answers << "\n\n";
    answers << fixed << setprecision(0);
}

void print(matrix &A, ofstream &answers, ll cnt=3)
{
    ll n = A.size();
    answers << fixed << setprecision(cnt);
    for(int i = 0; i < n; ++i)
    {
```

```
        for(int j = 0; j < A[i].size(); ++j) answers << A[i][j] << " ";
        answers << "\n";
    }
    answers << fixed << setprecision(0);
    answers << "\n";
}

void print(matrix &A, ll cnt=3)
{
    ll n = A.size();
    cout << fixed << setprecision(cnt);
    for(int i = 0; i < n; ++i)
    {
        for(int j = 0; j < A[i].size(); ++j) cout << A[i][j] << " ";
        cout << endl;
    }
    cout << fixed << setprecision(0);
    cout << endl;
}

matrix operator * (const matrix &A, const matrix &B)
{
    ll n = A.size();
    matrix res(n, vector<ld>(n, 0.0));
    for(int k = 0; k < n; ++k)
        for(int i = 0; i < n; ++i)
            for(int j = 0; j < n; ++j)
                res[i][j] += A[i][k] * B[k][j];
    return res;
}

void matrix_mult_vector(matrix &A, vector<ld> &B, vector<ld> &res)
{
    ll n = A.size();
    res.resize(n, 0);
    for(int i = 0; i < n; ++ i)
    {
        res[i] = 0;
        for(int j = 0; j < n; ++ j)
            res[i] += A[i][j] * B[j];
    }

}


matrix get_random_matrix(ll size_)
{
    ///randomize due to clock
    srand(clock());
    matrix res;
    res.resize(size_);
    for(int i = 0; i < size_; ++i)
    {
        res[i].resize(size_);
        for(int j = 0; j < size_; ++j) res[i][j] = rand() - rand()/2;
    }
    return res;
}


void check_ans(matrix &A, vector<ld> &X, vector<ld> &B, ll cnt=0)
{
```

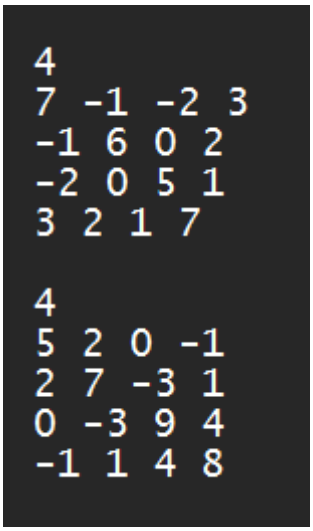
```
    answers << "HERE IS WHAT B SHOULD BE " << "\n";
    print(B, answers, cnt);
    answers << "HERE IS CALCULATED B" << "\n";
    ll n = A.size();
    vector<ld> ans(n,0);
    vector<ld> diff(n);
    ld euNorm = 0.0;
    for(int i = 0; i < n; ++i)
    {
        for(int j = 0; j < n; ++j)
        {
            ans[i] += A[i][j] * X[j];
        }
        diff[i] = B[i] - ans[i];
        euNorm += diff[i] * diff[i];
    }
    print(ans, answers, cnt);
    answers << "HERE is B - B(calculated)" << "\n";
    print(diff, answers, cnt);
    answers << "Norm of this vector = " << fixed << setprecision(cnt) << sqrt(euNorm) << "\n";
    answers << fixed << setprecision(0);
}


ld eu_norm_vector(vector<ld> &A)
{
    ld res = 0;
    for(auto to: A) res += to * to;
    return sqrt(res);
}


ld eu_norm_matrix(matrix &A)
{
    ld res = 0;
    for(int i = 0; i < A.size(); ++ i)
        for(int j = 0; j < A[i].size(); ++j)
            res += A[i][j] * A[i][j];
    return sqrt(res);
}


bool is_determinant_zero(matrix &A)
{
    ll n = A.size();
    matrix AE;
    AE.resize(n);

    for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++ j)
            AE[i].emplace_back(A[i][j]);

    for(int k = 0; k < n; ++k)
    {
        int row_max_el = k;

        /// finding maximum element
        for(int i = k + 1; i < n; ++i)
            if(fabs(AE[i][k]) > fabs(AE[row_max_el][k])) row_max_el = i;

        if(fabs(AE[row_max_el][k]) < eps) return 0;
```

```
        /// change of 2 rows for optimization of division by the maximum element
        if(k != row_max_el)
            for(int j = k; j < n; ++j) swap(AE[k][j], AE[row_max_el][j]);


        /// forming zeros under main diagonal
        for(int i = k + 1; i < n; ++i)
        {
            ld mik = -AE[i][k]/AE[k][k];

            AE[i][k] = 0;
            for(int j = k + 1; j < n; ++j) AE[i][j] += mik * AE[k][j];
        }
    }
    for(int i = 0; i < n; ++i)
    {
        if(fabs(AE[i][i]) < eps) return 0;
    }
    return 1;

}

void solve(matrix &A)
{
    ll n = A.size();
    answers << "Size = " << A.size() << "\n";
    answers << "Matrix A:\n";
    print(A, answers);

    vector<ld> x[2], y[2], lambda[2];
    ld y_norm;
    x[0].resize(n);
    x[1].resize(n);
    y[0].resize(n);
    y[1].resize(n);
    lambda[0].resize(n);
    lambda[1].resize(n);
    vector<bool> was[2];
    was[0].resize(n);
    was[1].resize(n);

    for(int i = 0; i < n; ++ i) y[0][i] = i + 1;
    y_norm = eu_norm_vector(y[0]);

    answers << "Vector y0:\n";
    print(y[0], answers, 7);

    answers << fixed << setprecision(10) << "Norm of y0 = " << y_norm << "\n";

    for(int i = 0; i < n; ++ i)
    {
        x[0][i] = y[0][i] / y_norm;
        if(fabs(x[0][i]) > eps) was[0][i] = 1;
        else was[0][i] = 0;
    }

    ld ANS = 0;

    answers << "\n";

    vector<ld> ansV(n);
```

```
ll step;
for(step = 1; ; ++step)
{
    matrix_mult_vector(A, x[!(step&1)], y[step&1]);

    answers << "STEP = " << step << " \n";


    y_norm = eu_norm_vector(y[step&1]);

    answers << "Vector y" << step << ":\n";
    print(y[step&1], answers, 7);
    answers << fixed << setprecision(10) << "Norm of y" << step << " = " << y_norm << "\n";

    for(int i = 0; i < n; ++ i)
    {
        x[step&1][i] = y[step&1][i] / y_norm;
        if(step == 1)
        {
            if(fabs(x[step&1][i]) > eps) was[1][i] = 1;
            else was[1][0] = 0;
        }
    }



    answers << "Vector x" << step << ":\n" ;
    print(x[step&1], answers, 7);



    for(int i = 0; i < n; ++ i)
    {
        if(fabs(x[!(step&1)][i]) > eps ) lambda[step&1][i] = y[step&1][i] / x[!(step&1)][i];
    }

    answers << "Lambda" << step << "Vector:\n" ;
    print(lambda[step&1], answers, 7);
    answers << "\n";

    if(step > 1)
    {
        ld sum = 0;
        ll total = 0;
        bool good = true;
        for(int i = 0; i < n; ++ i)
        {
            if(was[0][i] & was[1][i])
            {
                total ++;
                sum += lambda[step&1][i];
                if(fabs(lambda[0][i] - lambda[1][i]) >= eps) good = false;
            }
        }

        if(good)
        {
            ANS = sum / ld(total);
            for(int i = 0; i < n; ++ i) ansV[i] = x[step&1][i];
            break;
        }
```

```
                for(int i = 0; i < n; ++ i) was[0][i] = was[1][i];
                for(int i = 0; i < n; ++ i)
                {
                    if(fabs(x[step&1][i]) > eps) was[1][i] = 1;
                    else was[1][0] = 0;

                }
            }
        }

        answers << fixed << setprecision(10) << "Biggest Eigen Value = " << ANS << "\n";
        answers << "Eigen Vector:\n";
        print(ansV, answers, 7);

        for(int i = 0; i < n; ++ i) A[i][i] -= ANS;
        answers << "TOTAL STEPS = " << step << "\n";
        if(is_determinant_zero(A)) answers << "Determinant of (A - lambda * E) matrix -> 0 so found Eigen Value is
found correctly\n";
        //cout << step << "\n";

        answers << "------------------------------------------------------------\n";
    }
    int main()
    {
        ll n;
        while(tests >> n)
        {
            matrix A(n, vector<ld>(n, 0));
            for(int i = 0; i < n; ++i)
                for(int j = 0; j < n; ++j)
                    tests >> A[i][j];

            solve(A);
        }
        ll sz = 300;
        matrix A(sz, vector<ld>(sz, 0));
        A = get_random_matrix(sz);
        solve(A);
        return 0;
    }
```

Вхідні дані



Вихідні дані

```
Size = 4
Matrix A:
7.000 -1.000 -2.000 3.000
-1.000 6.000 0.000 2.000
-2.000 0.000 5.000 1.000
3.000 2.000 1.000 7.000

Vector y0:
1.0000000 2.0000000 3.0000000 4.0000000

Norm of y0 = 5.4772255751

STEP = 1
Vector y1:
2.0083160 3.4689095 3.1037612 6.9378191

Norm of y1 = 8.5926325031
Vector x1:
0.2337254 0.4037074 0.3612119 0.8074148

Lambda1Vector:
11.0000000 9.5000000 5.6666667 9.5000000


STEP = 2
Vector y2:
2.9321908 3.8033489 2.1460237 7.5217067

Norm of y2 = 9.1784909032
Vector x2:
```

```
Norm of y58 = 10.2155612773
Vector x58:
0.6797891 0.1716926 -0.1261213 0.7017847

Lambda58Vector:
10.2155620 10.2155573 10.2155651 10.2155607


STEP = 59
Vector y59:
6.9444281 1.7539361 -1.2883999 7.1691244

Norm of y59 = 10.2155612773
Vector x59:
0.6797892 0.1716926 -0.1261213 0.7017847

Lambda59Vector:
10.2155618 10.2155582 10.2155642 10.2155608


Biggest Eigen Value = 10.2155612756
Eigen Vector:
0.6797892 0.1716926 -0.1261213 0.7017847

TOTAL STEPS = 59
------------------------------------------------------------
```

```
Size = 4
Matrix A:
5.000 2.000 0.000 -1.000
2.000 7.000 -3.000 1.000
0.000 -3.000 9.000 4.000
-1.000 1.000 4.000 8.000

Vector y0:
1.0000000 2.0000000 3.0000000 4.0000000

Norm of y0 = 5.4772255751

STEP = 1
Vector y1:
0.9128709 2.0083160 6.7552449 8.2158384

Norm of y1 = 10.8627804912
Vector x1:
0.0840366 0.1848805 0.6218707 0.7563292

Lambda1Vector:
5.0000000 5.5000000 12.3333333 11.2500000


STEP = 2
Vector y2:
0.0336146 0.3529536 8.0675117 8.6389605

Norm of y2 = 11.8254847848
Vector x2:
0.0028426 0.0298469 0.6822140 0.7305375
```

```
STEP = 38
Vector y38:
-1.9307641 -4.2899952 9.9763574 7.2337629

Norm of y38 = 13.1904109177
Vector x38:
-0.1463763 -0.3252359 0.7563341 0.5484107

Lambda38Vector:
13.1904121 13.1904125 13.1904110 13.1904102


Biggest Eigen Value = 13.1904114479
Eigen Vector:
-0.1463763 -0.3252359 0.7563341 0.5484107

TOTAL STEPS = 38
Determinant of (A - lambda * E) matrix -> 0 so found Eigen Value is found correctly
------------------------------------------------------------
```

Аналіз чисельних експериментів

Size	10	50	100	200	500	1000	2000	5000
Time(s)	0.291	0.3	0.31	0.389	0.994	4.33	27.664	388.972
Steps	23	19	16	14	17	12	11	11

Знайдені власні числа степеневим методом є точними, оскільки детермінант матриці $A-\lambda I$ прямує до 0, що є основною властивістю власних чисел.

Степеневий метод генерує максимальне по модулю власне число, що може бути корисним в інших чисельних методах (метод Річардсона) і не тільки.

Степеневий метод спочатку генерує неточні результати, а потім уточнює свої результати на кожній ітерації. В багатьох випадках проміжний результат може бути корисним.

Для малих та середніх матриць степеневий метод працює швидко, проте зі збільшенням розміру матриць та з збільшенням точності час його роботи стрімко зростає, і стає досить великим.

У Степеневого методу, виникають проблеми, якщо найбільше власне значення відповідає власному простору з розмірністю більше 1, оскільки він знаходить лише один наближений власний вектор.

Висновок

Я здобув та закріпив вміння на навички вирішення часткової проблема на власні значення та дослідив обчислювальні аспекти цієї проблеми. Створив програму, яка за степеневим методом знаходить найбільше по модулю власне число і відповідний йому власний вектор, та виводить усі проміжні результати покроково.