

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №5

на тему:

«ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ЗНАЧЕНИЙ И ВЕКТОРОВ»

БГУИР 1-40 04 01

Выполнил студент группы 253505
Форинов Егор Вячеславович

(дата, подпись студента)

Проверил доцент кафедры
информатики
АНИСИМОВ Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2023

Содержание

1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

Цель работы

- изучить метод Якоби (вращений) для поиска собственных значений и векторов матрицы;
- составить программу поиска собственных значений и векторов матрицы указанными методами, применимую для организации вычислений на ЭВМ;
- выполнить тестовые примеры и проверить правильность работы программы

ЗАДАНИЕ 5. С точностью 0,0001 вычислить собственные значения и собственные векторы матрицы A ,

где $A = kC + D$, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы C, D заданы ниже:

$$C = \begin{bmatrix} 0,2 & 0 & 0,2 & 0 & 0 \\ 0 & 0,2 & 0 & 0,2 & 0 \\ 0,2 & 0 & 0,2 & 0 & 0,2 \\ 0 & 0,2 & 0 & 0,2 & 0 \\ 0 & 0 & 0,2 & 0 & 0,2 \end{bmatrix}, \quad D = \begin{bmatrix} 2,33 & 0,81 & 0,67 & 0,92 & -0,53 \\ 0,81 & 2,33 & 0,81 & 0,67 & 0,92 \\ 0,67 & 0,81 & 2,33 & 0,81 & 0,92 \\ 0,92 & 0,67 & 0,81 & 2,33 & -0,53 \\ -0,53 & 0,92 & 0,92 & -0,53 & 2,33 \end{bmatrix}.$$

Вариант 12

Программная реализация

Исходные данные

Матрица, полученная в результате подстановки $A = 12 * C + D$:

```
DenseMatrix 5x5-Double  
 4,73  0,81  3,07  0,92 -0,53  
 0,81  4,73  0,81  3,07  0,92  
 3,07  0,81  4,73  0,81  3,32  
 0,92  3,07  0,81  4,73 -0,53  
-0,53  0,92  3,32 -0,53  4,73
```

Код метода Якоби(вращений):

```
Tuple<Vector<double>, Matrix<double>> Jacobi(Matrix<double> matrix)  
{  
    if (!matrix.IsSymmetric())  
        throw new ArgumentException("Source matrix is not symmetric");
```

```

var n = matrix.RowCount;
const double maxIterations = 1000;
var eigenvectors = Matrix<double>.Build.DenseDiagonal(n, n, 1);

for (var iteration = 0; iteration < maxIterations; iteration++)
{
    double maxOfDiagonal = 0.0;
    int p = 0, q = 0;

    for (var i = 0; i < n - 1; i++)
    {
        for (var j = i + 1; j < n; j++)
        {
            if (Math.Abs(matrix[i, j]) > maxOfDiagonal)
            {
                maxOfDiagonal = Math.Abs(matrix[i, j]);
                p = i;
                q = j;
            }
        }
    }

    if (maxOfDiagonal < 1e-8)
        break;

    double angle = 0.5 * Math.Atan(2 * matrix[p, q] / (matrix[p, p] - matrix[q, q]));
    double cos = Math.Cos(angle);
    double sin = Math.Sin(angle);

    var rotationMatrix = Matrix<double>.Build.Dense(n, n);
    for (var i = 0; i < n; i++)
    {
        for (var j = 0; j < n; j++)
        {
            if (i == j && i != p && i != q)
                rotationMatrix[i, j] = 1.0;
            else if (i == p && j == p)
                rotationMatrix[i, j] = cos;
            else if (i == p && j == q)
                rotationMatrix[i, j] = -sin;
            else if (i == q && j == p)
                rotationMatrix[i, j] = sin;
            else if (i == q && j == q)
                rotationMatrix[i, j] = cos;
            else
                rotationMatrix[i, j] = 0.0;
        }
    }
}

```

```

    }

    matrix = rotationMatrix.Transpose().Multiply(matrix).Multiply(rotationMatrix);
    eigenvectors = eigenvectors.Multiply(rotationMatrix);
}

var eigenValues = matrix.Diagonal();
return new (eigenValues, eigenvectors);
}

```

Полученные результаты

DenseVector 5-Double

```

5,13672
7,00555
10,0952
1,62309
-0,21052

```

DenseMatrix 5x5-Double

```

0,733979 -0,0003375 0,43388 -0,255572 0,455748
-0,36435 0,457528 0,432191 -0,658517 -0,19361
0,133152 -0,391698 0,594103 0,234807 -0,648653
-0,11349 0,600264 0,37566 0,667606 0,199961
-0,545816 -0,526236 0,361775 0,0143726 0,542286

```

По значениям определителей матрицы для каждого собственного значения можно сделать вывод, что точность 0,0001 была достигнута.

Тестовый пример 1. Нулевые строка и столбец

Исходная матрица:

```
[[0. 0. 0.]  
 [0. 7. 3.]  
 [0. 3. 4.]]
```

Кол-во итераций: 1

Матрица собственных значений(после метода):

```
[ 0.0000e+00 0.0000e+00 0.0000e+00 ]  
[ 0.0000e+00 8.8541e+00 4.4409e-16 ]  
[ 0.0000e+00 2.2204e-16 2.1459e+00 ]
```

Собственные значения(диагональ):

```
[0.          8.85410197 2.14589803]
```

Матрица свободных векторов:

```
[[ 1.          0.          0.          ]  
 [ 0.          0.85065081 -0.52573111]  
 [ 0.          0.52573111  0.85065081]]
```

Матрица после операции $V * A * V.T$:

```
[[0. 0. 0.]  
 [0. 7. 3.]  
 [0. 3. 4.]]
```

Определители матрицы для каждого соб. знач.

```
[ 0.0000e+00 -1.1796e-14 0.0000e+00 ]
```

Тестовый пример 2. Нулевая диагональ

Исходная матрица:

```
[[ 0.  3.  7.]  
 [ 3.  0. 11.]  
 [ 7. 11.  0.]]
```

Кол-во итераций: 7

Матрица собственных значений(после метода):

```
[ 1.4520e+01 -6.1972e-07 -3.4557e-04 ]  
[ -6.1972e-07 -2.6897e+00 -4.3368e-18 ]  
[ -3.4557e-04 -1.1796e-16 -1.1830e+01 ]
```

Собственные значения(диагональ):

```
[ 14.51960878 -2.6897146 -11.82989417]
```

Матрица свободных векторов:

```
[[ 0.44505376 -0.85044437 -0.28048444]  
 [ 0.59736648  0.51528705 -0.61451814]  
 [ 0.66714349  0.10594161  0.7373574 ]]
```

Матрица после операции $V * A * V.T$:

```
[[-1.99840144e-15  3.00000000e+00  7.00000000e+00]  
 [ 3.00000000e+00 -2.66453526e-15  1.10000000e+01]  
 [ 7.00000000e+00  1.10000000e+01 -1.77635684e-15]]
```

Определители матрицы для каждого соб. знач.

```
[ 2.0551e-06 3.5748e-12 -1.0915e-06 ]
```

Тестовый пример 3. Нулевая матрица

Исходная матрица:

```
[[3. 3. 3.]  
 [3. 3. 3.]  
 [3. 3. 3.]]
```

Кол-во итераций: 2

Матрица собственных значений(после метода):

```
[ 9.00000e+00 0.00000e+00 8.8818e-16 ]  
[ 0.00000e+00 0.00000e+00 0.00000e+00 ]  
[ 3.6260e-16 0.00000e+00 -2.5640e-16 ]
```

Собственные значения(диагональ):

```
[ 9.000000000e+00 0.000000000e+00 -2.56395025e-16]
```

Матрица свободных векторов:

```
[[ 0.57735027 -0.70710678 -0.40824829]  
 [ 0.57735027  0.70710678 -0.40824829]  
 [ 0.57735027  0.          0.81649658]]
```

Матрица после операции $V * A * V.T$:

```
[[3. 3. 3.]  
 [3. 3. 3.]  
 [3. 3. 3.]]
```

Определители матрицы для каждого соб. знач.

```
[ 0.00000e+00 0.00000e+00 3.5499e-30 ]
```

Тестовый пример 3. Несимметричная матрица

Исходная матрица:

```
[[9. 8. 7.]  
 [4. 5. 6.]  
 [3. 2. 1.]]
```

Кол-во итераций: 9

Матрица собственных значений(после метода):

```
[ 1.5390e+01 4.3713e-04 5.5231e-03 ]  
[ -6.4548e+00 1.5313e-02 4.0828e-04 ]  
[ -1.0052e+00 -2.3035e+00 -4.0538e-01 ]
```

Собственные значения(диагональ):

```
[ 1.53900638e+01 1.53130902e-02 -4.05376860e-01]
```

Матрица свободных векторов:

```
[[ 0.61388435 -0.73603235 0.28531102]  
 [ 0.57682284 0.17150817 -0.7986616 ]  
 [ 0.53890761 0.65485977 0.52984646]]
```

Матрица после операции $V * A * V.T$:

```
[[9. 8. 7.]  
 [4. 5. 6.]  
 [3. 2. 1.]]
```

Определители матрицы для каждого соб. знач.

```
[ -4.7806e-02 9.5392e-02 9.9311e-02 ]
```

Вывод

В ходе выполнения лабораторной работы я изучил метод Якоби(вращений), написал программу его реализации на языке C#, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

- Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^{-4});
- Основное достоинство метода Якоби заключается в том, что при выполнении каждого плоского вращения уменьшается сумма квадратов недиагональных элементов; сходимость этой суммы к нулю по мере увеличения числа шагов гарантирует сходимость процесса диагонализации.