Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе

на тему

Вычисление собственных значений и векторов

Выполнил: студент группы 153501
Тимофеев Кирилл Андреевич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Вариант 8

Цель выполнения задания:

- Освоить методы вычисления собственных значений и векторов

Краткие теоретические сведения:

Метод Якоби (вращений) использует итерационный процесс, который приводит исходную симметрическую матрицу A к диагональному виду с помощью последовательности *элементарных ортогональных преобразований* (в дальнейшем называемых *вращениями Якоби* или *плоскими вращениями).* Процедура построена таким образом, что на *(k+1)*-ом шаге осуществляется преобразование вида

$$A^{(k)} \to A^{(k+1)} = V^{(k)*} A^{(k)} V^{(k)} = V^{(k)*} \dots V^{(0)*} A^{(0)} V^{(0)} \dots V^{(k)}, \quad k=0,1,2\dots, \qquad (5.1)$$

где $A^{(0)} = A$, $V^{(k)} = V^{(k)}{}_{ij}\,(\varphi)$ — ортогональная матрица, отличающаяся от единичной матрицы только элементами

$$v_{ii} = v_{jj} = \cos\varphi \quad v_{ij} = -v_{ji} = -\sin\varphi \ , \qquad (5.2)$$

значение φ выбирается при этом таким образом, чтобы обратить в 0 наибольший по модулю недиагональный элемент матрицы $A^{(k)}$. Итерационный процесс постепенно приводит к матрице со значениями недиагональных элементов, которыми можно пренебречь, т.е. матрица $A^{(k)}$ все более похожа на диагональную, а диагональная матрица Λ является пределом последовательности $A^{(k)}$ при $k \to \infty$.

Алгоритм метода вращений.

1) В матрице $A^{(k)}$ $(k=0,1,2,\dots.)$ среди всех недиагональных элементов выбираем максимальный по абсолютной величине элемент, стоящий выше главной диагонали; определяем его номера i и j строки и столбца, в которых он находится.

2) По формулам

$$\frac{a_{jj}^{(i)} - a_{kk}^{(i)}}{2a_{jk}^{(i)}} = \frac{c^2 - s^2}{2sc} = \frac{\cos(2\theta)}{\sin(2\theta)} = \operatorname{ctg}(2\theta) \equiv \tau.$$

Если $a_{jj}^{(i)} = a_{kk}^{(i)}$, то выбирается $\theta = \frac{\pi}{4}$, в противном случае

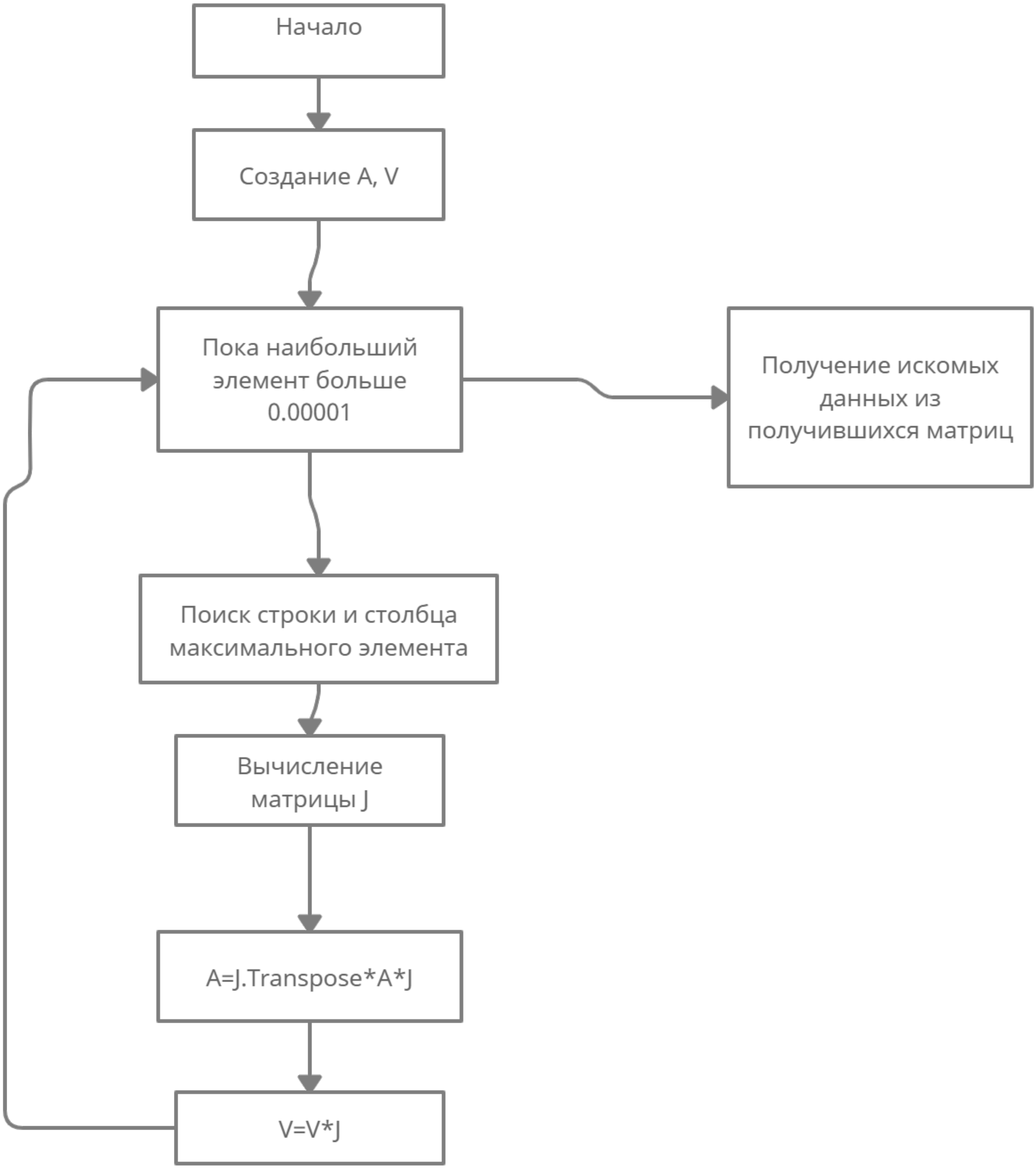
вводится $t = \frac{s}{c} = \operatorname{tg}(\theta)$ и тогда $t^2 - 2t\tau + 1 = 0$. Решение квадратного уравнения даёт $t = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}$, $c = \frac{1}{\sqrt{1+t^2}}$, $s = tc$.

где c - косинус, s – синус нужного угла

3) Находится матрица V(k)
4) A(k+1)=1/V(k)*A(k)*V(k)
5) Итерационный процесс останавливается, когда недиагональными элементами можно пренебречь.

   В качестве собственных чисел берем диагональные элементы матрицы A(к+1), в качестве собственных векторов – соответствующие столбцы матрицы V=V(0)*V(1)*V(2)...*V(k)

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.


Начало
Создание A, V
Пока наибольший элемент больше 0.00001
Получение искомых данных из получившихся матриц
Поиск строки и столбца максимального элемента
Вычисление матрицы J
A=J.Transpose*A*J
V=V*J

Метод Данилевского относится к прямым методам и является достаточно простым и экономичным. Известно, что матрицы S-1 AS, полученные преобразованием подобия из с A, имеют тот же характеристический многочлен, что и A. Известно так же, что любая матрица приводима преобразованием подобия к так называемой канонической форме Фробениуса

$$F=\begin{pmatrix} -p_1 & -p_2 & \cdots & \cdots & -p_n \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

в первой строке которой стоят коэффициенты характеристического многочлена, взятые с обратным знаком. Таким образом, основная задача сводится к нахождению матрицы S такой, что F= S-1 AS. Предположим, что элемент ann-1 матрицы A отличен от нуля. Разделим (n-1)-й столбец этой матрицы на ann-1 и вычтем его из i–го столбца, умноженного на ani (для всех i=1,2,…,n). Тогда последняя строка примет такой же вид как в матрице F. Непосредственно проверяется, что проделанная операция равносильна умножению A справа на матрицу

$$M_{n-1}=\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -\frac{a_{n1}}{a_{nn-1}} & \cdots & -\frac{a_{nn-2}}{a_{nn-1}} & \frac{1}{a_{nn-1}} & -\frac{a_{nn}}{a_{nn-1}} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}.$$

Непосредственно проверяется также, что M_{n-1} не вырождена и, следовательно, существует

$$M^{-1}_{n-1}=\begin{pmatrix} 1 & 0 & \dots & & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & \dots & \dots & \dots & a_{nn} \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}.$$

Заметим, что матрицы M_{n-1} *и* M^{-1}_{n-1}, умножением на которые мы переходим от матрицы A к матрице $A^{(1)}$, выписываются непосредственно по виду матрицы A. Предположим далее, что элемент $a^{(1)}_{n-1n-2}$ тоже отличен от нуля. Делаем второй шаг, полностью аналогичный предыдущему, и приводим вторую снизу строку матрицы к виду необходимому для формы Фробениуса (сохраняя последнюю строку без изменений). Получаем

$$M^{-1}_{n-2}\, M^{-1}_{n-1} A M_{n-1} M_{n-2} = M^{-1}_{n-2}\, A^{(1)}\, M_{n-2} =$$

$$= A^{(2)}=\begin{pmatrix} a^{(2)}_{11} & \dots & \dots & \dots & a^{(2)}_{1n} \\ a^{(2)}_{21} & \dots & \dots & \dots & a^{(2)}_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a^{(2)}_{n-21} & \dots & \dots & a^{(2)}_{n-2n-1} & a^{(2)}_{n-2n} \\ 0 & \dots & 1 & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix},$$

Правило построения матриц M_{n-2} и M^{-1}_{n-2} по виду матрицы $A^{(1)}$, как видим, полностью сохраняется. Оно сохраняется и на следующих шагах метода. Таким образом, если имеет место так называемый регулярный случай, когда

$$a_{nn-1} \neq 0,\ a^{(1)}_{n-1n-2} \neq 0,\ a^{(2)}_{n-2n-3} \neq 0, \dots,\ a^{(n-2)}_{21} \neq 0,$$

то после $(n-1)$ шагов метода Данилевского получим следующий результат

$$A^{(n-1)} = M^{-1}_{1} \dots M^{-1}_{n-1} A M_{n-1} \dots M_{1} =$$

$$= \begin{pmatrix} a^{(n-1)}_{11} & a^{(n-1)}_{12} & \dots & \dots & a^{(n-1)}_{1n} \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} -p_1 & -p_2 & \dots & \dots & -p_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix} = F = S^{-1} AS.$$

В таком случае мы можем непосредственно выписать характеристическое уравнение

$$\lambda^n + p_1 \lambda^{n-1} + \dots + p_n = 0$$

Для нахождения собственных векторов в регулярном случае нет необходимости решать систему (4.2). Как уже говорилось выше, матрицы F и A имеют одни и те же собственные значения. Собственные векторы, отвечающие одному и тому же собственному значению, вообще говоря, будут разными. Однако они связаны между собой преобразованием подобия. Так, если $\overline{y}$ собственный вектор матрицы F, отвечающий собственному значению λ, то вектор $S\overline{y}$ будет собственным вектором матрицы A, отвечающим тому же собственному значению.

Действительно, поскольку $F\overline{y} = \lambda\, \overline{y}$ и $F = S^{-1}AS$, то $S^{-1}AS\overline{y} = \lambda\, \overline{y}$. Умножая это равенство слева на матрицу S, получим $AS\overline{y} = \lambda\, S\overline{y}$. Последнее означает, что $S\overline{y}$ будет собственным вектором матрицы A. Таким образом, собственные векторы матрицы A находятся пересчетом собственных векторов матрицы Фробениуса. Собственные же векторы $\overline{y}$ матрицы Фробениуса определяются из системы

$$\begin{pmatrix} -p_1 & -p_2 \ldots\ldots\ldots\ldots\ldots\ldots & & -p_n \\ 1 & 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots & & 0 \\ 0 & 1 & 0 \ldots\ldots\ldots\ldots\ldots & 0 \\ 0 & 0 & 1 \ldots\ldots\ldots\ldots\ldots & 0 \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \\ 0 \ldots\ldots\ldots\ldots & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \\ y_n \end{pmatrix} = \lambda \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \\ y_n \end{pmatrix}.$$

Поскольку собственный вектор определяется с точностью до постоянного множителя, можно принять $y_n = 1$ и вычислить остальные координаты собственного вектора:

$$y_n = 1, \quad y_{n-1} = \lambda \ , \ldots\ldots\ldots\ldots, y_1 = \lambda^{\,n-1}.$$

Равенство же

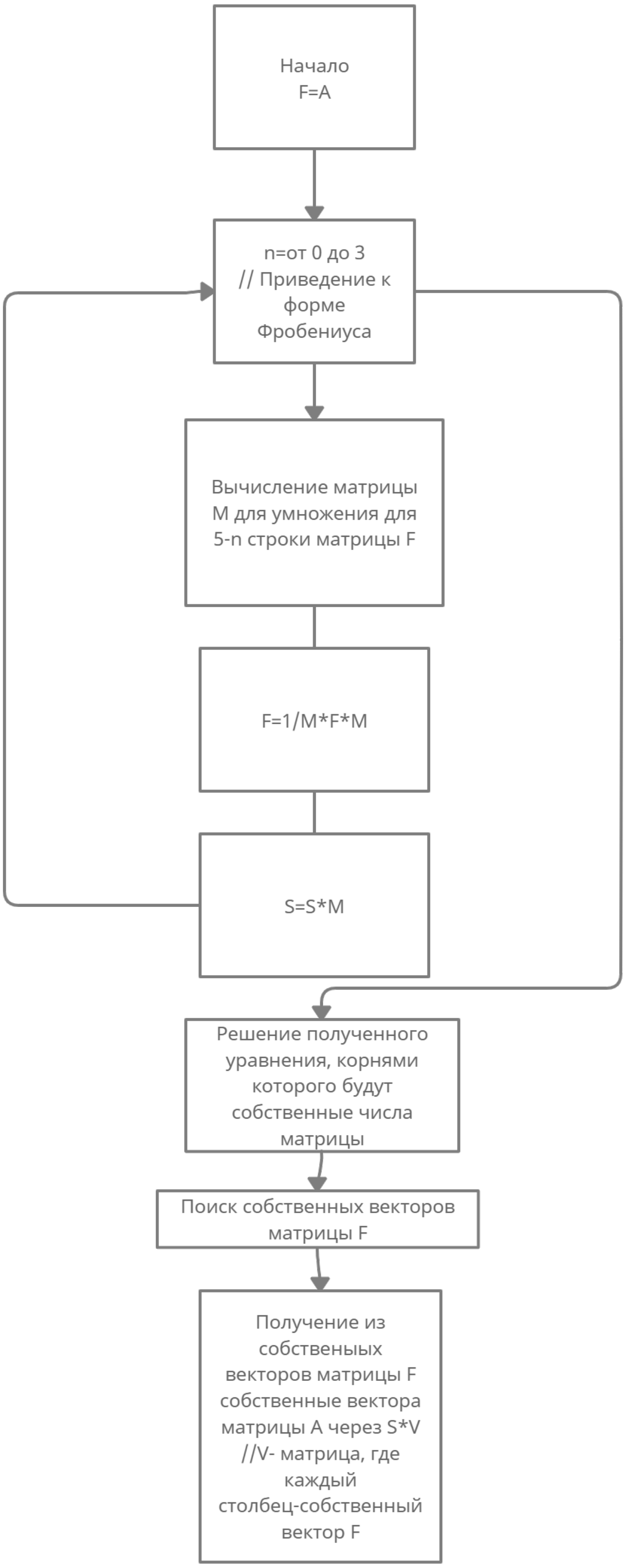
$$-p_1 y_1 - p_2 y_2 - \ldots\ldots\ldots - p_n y_n = \lambda\ y_1$$

принимает при этом тривиальный вид

$$\lambda^{\,n} + p_1 \lambda^{\,n-1} + \ldots\ldots + p_n = 0$$

и используется для контроля вычислений.

Зная матрицу *S,* не трудно теперь найти собственные векторы матрицы A.



Програмная реализация:

```
using System;
using System.IO;

namespace Lab5
{
    class Program
    {
        static int rows = 5;
        static int columns = 5;
        static double k = 8;
        static string projectLocation = "D:\\Mcha\\Lab5\\";
        static void Main(string[] args)
        {
            Matrix A = read(0);
            Jacobe(A);
        }

        static Matrix read(int mode)
        {
            Matrix C = new Matrix(rows, columns);
            Matrix D = new Matrix(rows, columns);
            ReadC(C, mode);
            ReadD(D, mode);



            return k * C + D;
        }

        static void ReadC(Matrix C, int mode)
        {
            string[] CPath = new string[2] { "CDefault.txt", "C.txt" };
            StreamReader sr = new StreamReader(projectLocation + CPath[mode]);
            for (int i = 0; i < rows; ++i)
            {
                string line = sr.ReadLine();
```

```
        for (int j = 0; j < columns; ++j)
        {
            int f = line.IndexOf(' ');
            //Console.WriteLine("f = " + f);
            if (f != -1) C[i, j] = Convert.ToDouble(line.Substring(0, f));
            else C[i, j] = Convert.ToDouble(line);
            line = line.Substring(f + 1);
        }
    }
    Console.WriteLine(C);
}

static void ReadD(Matrix D, int mode)
{
    string[] DPath = new string[2] { "DDefault.txt", "D.txt" };
    StreamReader sr = new StreamReader(projectLocation + DPath[mode]);
    for (int i = 0; i < D.Rows; ++i)
    {
        string line = sr.ReadLine();

        for (int j = 0; j < D.Columns; ++j)
        {
            int f = line.IndexOf(' ');
            //Console.WriteLine("f = " + f);
            if (f != -1) D[i, j] = Convert.ToDouble(line.Substring(0, f));
            else D[i, j] = Convert.ToDouble(line);
            line = line.Substring(f + 1);
        }
    }
    Console.WriteLine(D);
}

static void Jacobe(Matrix a)
{
    Matrix A = new Matrix(a);
    int i = 0, j = 1;
```

```
Matrix J = Matrix.E(A.Rows);
Matrix V = Matrix.E(A.Rows);
Console.WriteLine("E\n" + J);
do
{

    for (int i1 = 0; i1 < A.Rows; ++i1)
    {
        for (int j1 = i1 + 1; j1 < A.Columns; ++j1)
        {
            if (Math.Abs(A[i, j]) < Math.Abs(A[i1, j1]))
            {
                i = i1;
                j = j1;
            }
        }
    }
    Console.WriteLine("Matrix\n" + A);
    //Console.WriteLine(A.Transpon());
    Console.WriteLine($"{A[i, j]} i = {i} j = {j}");
    if (Math.Abs(A[i, j]) < 0.0001) break;
    J = Matrix.E(A.Rows);

    double sin, cos;
    if (A[i, i] == A[j, j])
    {
        sin = Math.Sin(3.1415926 / 4);
        cos = Math.Cos(3.1415926 / 4);
    }
    else
    {
        double Tau = (A[i, i] - A[j, j]) / (2*A[i, j]);
        double t = Math.Sign(Tau) / (Math.Abs(Tau) + Math.Sqrt(1 + Tau * Tau));
        cos = 1 / Math.Sqrt(1 + t * t);
        sin = t * cos;
    }
```

```
    Console.WriteLine($"cos = {cos} sin = {sin}");

    J[i, i] = cos;
    J[j, j] = cos;

    J[i, j] = -sin;

    J[j, i] = sin;
    V *= J;
    Console.WriteLine("Jac\n" + J);
    Console.WriteLine("Jac trans\n" + J.Transpon());
    A = (J.Transpon() * A) * J;
}
while (true);

Console.WriteLine(A + "\n\n");
Console.WriteLine(J + "\n\n");

//Checking

double[] numbs = new double[A.Rows];
for (i = 0; i < A.Rows; ++i) numbs[i] = A[i, i];

for(i = 0; i < A.Columns; ++i)
{
    Matrix V1 = new Matrix(A.Rows, 1);
    for (j = 0; j < A.Rows; ++j) V1[j,0] = V[j,i];
    Matrix res = a * V1;
    for(j = 0; j < res.Rows; ++j)
    {
        res[j, 0] /= numbs[i];
    }

    Console.WriteLine("Vector\n" + V1);
    Console.WriteLine("res\n" + res);
}
```

```
        }

     }
}
```

```
S := DiagonalMatrix([1, 1, 1, 1, 1]) :
for n from 0 to 3 do
M := DiagonalMatrix([1, 1, 1, 1, 1]) :
for i from 1 to 5 do M[4 - n, i] := - A[5 - n, i] / A[5 - n, 4 - n]; end do;
M[4 - n, 4 - n] := 1 / A[5 - n, 4 - n];
M;
S := S · M;
M1 := M^(-1);
A := M1 · A · M;
end do:
equal := l^5 - A[1, 1]·l^4 - A[1, 2]·l^3 - A[1, 3]·l^2 - A[1, 4]·l - A[1, 5] = 0 :
vec := Vector([solve(equal, l)]);
```

```
V := Matrix([[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]) :
for j from 1to 5 do
v2 := Vector([vec[j]^4, vec[j]^3, vec[j]^2, vec[j], 1]);
for i from 0 to 4 do V[5 - i, j] := vec[j]^i end do;
S · v2;
end do:
#V
#vec, S.V;
res := S · V :
```

Результаты расчетов программы:

```
Собственное число 4,148017198596844 Вектор
0,7079933487695773
-0,48653898260823736
0,24471463994147713
-0,2301523291459436
-0,38622517640308107


Собственное число 8,288624627841667 Вектор
0,4301378612284879
0,46450630943072374
0,575221462850993
0,39951821676760624
0,3297283305741684


Собственное число 5,518782497311841 Вектор
-0,16061518922188456
-0,304157809679493
0,37487882944062645
-0,5557952056038773
0,6574559608575782
```

```
Собственное число 1,6162486561946996 Вектор
-0,3146937607525614
-0,6315855276984865
0,28708582402176885
0,6476359445463068
0,014729939666845151


Собственное число 0,07832702005495996 Вектор
0,4346233269975849
-0,23685745695330754
-0,6215163767406658
0,2430524071448103
0,5564566735509883
```

$$vec := \begin{bmatrix} 0.07832702012 \\ 1.616248657 \\ 4.148017199 \\ 5.518782497 \\ 8.288624628 \end{bmatrix}$$

$$\begin{bmatrix} 0.434619382787470 & -0.314692404098659 & -0.707995856783834 & -0.160623190490013 & 0.430135763543531 \\ -0.236854408436619 & -0.631583315850855 & 0.486540318528251 & -0.304168519909275 & 0.464502355047873 \\ -0.621517890506022 & 0.287082732049905 & -0.244711054766060 & 0.374873289210943 & 0.575226396933117 \\ 0.243053445295391 & 0.647640249860463 & 0.230150953533280 & -0.555795818753023 & 0.399510775113185 \\ 0.556458907548901 & 0.0147252315274863 & 0.386221977885188 & 0.657451682078144 & 0.329737048717638 \end{bmatrix}$$

```
Собственное число 4,14801719859684 Вектор
0,7079933487695773
-0,48653898260823736
0,24471463994147713
-0,2301523291459436
-0,38622517640308107

Результат умножения
0,7079957911196011
-0,4865403356883952
0,24471114294407048
-0,23015094095389824
-0,38622203773736047


Собственное число 8,288624627841667 Вектор
0,4301378612284879
0,46450630943072374
0,575221462850993
0,39951821676760624
0,3297283305741684

Результат умножения
0,43013714943547465
0,46450496163443483
0,5752231241411816
0,3995157538219858
0,32973124391458786
```

```
Собственное число 5,518782497311841 Вектор
-0,16061518922188456
-0,304157809679493
0,37487882944062645
-0,5557952056038773
0,6574559608575782

Результат умножения
-0,16061401685861204
-0,30415807780297227
0,37488420682592566
-0,5557891018501778
0,6574582170035752


Собственное число 1,6162486561946996 Вектор
-0,3146937607525614
-0,6315855276984865
0,28708582402176885
0,6476359445463068
0,014729939666845151

Результат умножения
-0,3146964631681522 5
-0,6315911911702837
0,2870924194281431
0,6476259081307951
0,014742098278593092


Собственное число 0,07832702005495996 Вектор
0,4346233269975849
-0,23685745695330754
-0,6215163767406658
0,243052407144810 3
0,5564566735509883

Результат умножения
0,4348325435376122
-0,2370072577731101
-0,621441761330093
0,24299011346986954
0,5563400490126851
```

```
A
4,148017198596844 2,710505431213761E-20 1,2655411388319298E-09 1,1336768460105955E-07 2,3392185001912975E-05
1,6951500966810862E-16 8,288624627841667 3,672968443582623E-05 -3,90614574927864E-12 -1,014925266157526E-09
1,2655411232185616E-09 3,6729684435801356E-05 5,518782497311841 2,941743841365948E-05 -1,661707091840956E-07
1,1336768483110196E-07 -3,906263501560878E-12 2,9417438413732695E-05 1,6162486561946996 6,369883213686486E-07
2,3392185001539196E-05 -1,0149253830874093E-09 -1,6617070879878819E-07 6,369883215174911E-07 0,07832702005495996


V
0,7079933487695773 0,4301378612284879 -0,16061518922188456 -0,3146937607525614 0,4346233269975849
-0,48653898260823736 0,46450630943072374 -0,304157809679493 -0,6315855276984865 -0,23685745695330754
0,24471463994147713 0,575221462850993 0,37487882944062645 0,28708582402176885 -0,6215163767406658
-0,2301523291459436 0,39951821676760624 -0,5557952056038773 0,6476359445463068 0,2430524071448103
-0,38622517640308107 0,3297283305741684 0,6574559608575782 0,014729939666845151 0,5564566735509883
```

Оценка:

Относительная погрешность приближенного числа:

Учитывая формулу

$$\left|\frac{a-a^*}{a}\right| = \frac{\beta_{1+1} r^{-(1+1)} + \beta_{1+1} r^{-(1+2)} + \ldots}{\beta_1 r^{-1} + \beta_2 r^{-2} + \ldots} \le \frac{r^{-1}}{\beta_1 r^{-1}} \le r^{1-1}$$

где r-основание системы исчисления

И параметры используемого типа double

Ниже в таблице даны параметры стандартных форматов чисел с плавающей запятой. Здесь: **w** — ширина битового поля для представления порядка, **t** — ширина битового поля для представления мантиссы, **k** — полная ширина битовой строки.

Параметр	*Binary32*	*Binary64*	*Binary128*	*Decimal64*	*Decimal128*
			Параметры формата		
b	2	2	2	10	10
p, цифры	24	53	113	16	34
emax	127	1023	16383	384	6144
			Параметры кодирования		
BIAS	127	1023	16383	398	6176
w, биты	8	11	15	13	17
t, биты	23	52	112	50	110
k, биты	32	64	128	64	128

То относительная погрешность приближенного числа не превышает 2^(1 – 52) = 4,4*10^(-16)

Вывод:

В ходе лабораторной работы были освоены методы получения собственных чисел и векторов имени Якоби и Данилевского.

СПИСОК ЛИТЕРАТУРЫ

1. ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ: БАЗОВЫЕ ПОНЯТИЯ И АЛГОРИТМЫ / Б.В, Фалейчик, 2010
2. КОМПЕНСАЦИЯ ПОГРЕШНОСТЕЙ ПРИ ОПЕРАЦИЯХ С ЧИСЛАМИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ[Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/266023/