**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Факультет №8**

**«Информационных технологий и прикладной математики»**

**Кафедра вычислительной математики и программирования**

**Курсовой проект**

**по курсу**

**«Численные методы»**

|  |  |
| --- | --- |
| **Студент:** | Сорокин Д.М. |
| **Группа:** | | 8О-304Б |
| **Преподаватель:** | | Сластушенский Ю.В. |
|  | |  |
| **Оценка:** | |  |
| **Дата:** | |  |

3 курс, 6 семестр

Москва 2018

# Постановка задачи

Нахождение собственных значений несимметричных разреженных матриц большой размерности. Метод Арнольди.

# Описание метода Арнольди

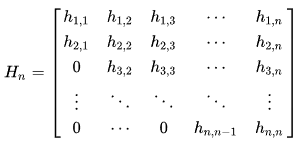
Метод Арнольди использует стабилизированный процесс Грамма-Шмидта для создания последовательности ортонормированных векторов  , называемых векторами Арнольди.

На псевдокоде метод выглядит так

* Начинаем с произвольного вектора с нормой

Алгоритм «ломается», когда - нулевой вектор.

В результате получается две матрицы –

и

посчитанная в ходе алгоритма. Легко заметить, что данная матрица находится в верхней форме Хессенберга. Данный факт является ключевым для поиска собственных значений, ведь теперь, можно без особых затрат применять QR-алгоритм к данной матрице и получить собственные значения.

# Описание работы программы

Программа начинается с предложения сгенерировать новую матрицу. Сгенерированная матрица записывается в файл. Далее из этого файла составляется матрица и начинается работа алгоритма. Выходными данными будут подряд напечатанные собственные значения.

# Код программы

using System;

using System.CodeDom.Compiler;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using NMlab1;

namespace CP

{

class Program

{

static void Main(string[] args)

{

while (true)

{

Console.WriteLine("Gen new mtx?");

var ansMtx = Console.ReadLine();

if (ansMtx == "y")

GenerateMatrix();

var A = new Matrix();

A.ReadFromFile("mtx.txt");

A.Print("A");

var H = Arnoldi(A);

for (int i = 0; i < 50; ++i)

{

if (H == null)

{

Console.WriteLine("Неподходящая матрица");

break;

}

H = QR(H);

}

if (H == null)

continue;

H.Print("H after QR");

FindEigvalues(H);

}

}

private static void FindEigvalues(Matrix H)

{

decimal eps = Convert.ToDecimal(0.000001);

var n = H.mtx.Count;

var eigvalues = new List<Complex>();

for (int i = 0; i < n - 1; ++i)

{

var lower = H.mtx[i + 1][i];

if (Math.Abs(lower) < eps)

{

eigvalues.Add(new Complex(H.mtx[i][i], 0));

continue;

}

var b = -H.mtx[i][i] - H.mtx[i + 1][i + 1];

var c = H.mtx[i][i] \* H.mtx[i + 1][i + 1] - H.mtx[i][i + 1] \* H.mtx[i + 1][i];

var D = Convert.ToDouble(b \* b - 4 \* c);

var sqrtD = Convert.ToDecimal(Math.Sqrt(Math.Abs(D)));

if (D < 0)

{

eigvalues.Add(new Complex(-b / 2, sqrtD / 2));

eigvalues.Add(new Complex(-b / 2, -sqrtD / 2));

}

else

{

eigvalues.Add(new Complex(-b / 2 + sqrtD / 2, 0));

eigvalues.Add(new Complex(-b / 2 - sqrtD / 2, 0));

}

++i;

}

if (eigvalues.Count < n)

eigvalues.Add(new Complex(H.mtx[n - 1][n - 1], 0));

LambdaPrint(eigvalues);

}

private static void LambdaPrint(List<Complex> lambda)

{

for (int i = 0; i < lambda.Count; ++i)

{

if (lambda[i].I >= 0)

Console.WriteLine($"l{i} = {lambda[i].R} + {lambda[i].I}i");

else

Console.WriteLine($"l{i} = {lambda[i].R} - {Math.Abs(lambda[i].I)}i");

}

}

private static Matrix Arnoldi(Matrix A)

{

var n = A.mtx.Count();

var q = new List<decimal>();

q.Add(Convert.ToDecimal(0.78289));

q.Add(Convert.ToDecimal(0.62216));

for (int i = 0; i < n - 2; ++i)

q.Add(0);

var Q = new Matrix(n, n + 1);

var H = new Matrix(n + 1, n);

Q.SetColumn(0, q);

Q.Print("Q");

for (int i = 0; i < n; ++i)

{

var v = A \* q;

for (int j = 0; j < i + 1; ++j)

{

H.mtx[j][i] = Vector.Scalar(Q.GetColumn(j), v);

v = Vector.SumVec(v, Vector.DigitOnVector(-1, Vector.DigitOnVector(H.mtx[j][i], Q.GetColumn(j))));

}

Vector.Print(v, "V");

H.mtx[i + 1][i] = Vector.Norm(v);

if (H.mtx[i + 1][i] == 0)

return null;

H.Print("H");

q = Vector.FindNormVec(v);

//q = Vector.DigitOnVector(1 / H.mtx[i + 1][i], v);

Q.SetColumn(i + 1, q);

Q.Print("Q");

}

Q.Print("Result Q");

H.Print("Result H");

H.mtx.RemoveAt(H.mtx.Count - 1);

--H.rows;

//var H1 = new Matrix();

//for (int i = 0; i < n; ++i)

//{

// H1.mtx.Add(H.mtx[i]);

//}

H.Print("H1");

return H;

}

private static Matrix QR(Matrix A)

{

var R = new Matrix(A);

var Q = new Matrix(A.columns);

Q.MakeUnitMatrix();

for (int i = 0; i < R.rows - 1; i++)

{

var b = R.GetColumn(i);

var v = CalcV(b, i);

var H = HouseholderCalc(v);

if (H == null)

return null;

Q \*= H;

R = H \* R;

}

//Q.Print("Q");

//R.Print("R");

return R \* Q;

}

public static Matrix HouseholderCalc(List<decimal> v)

{

Matrix E = new Matrix(v.Count);

E.MakeUnitMatrix();

if (Vector.Scalar(v, v) == 0)

return null;

var koef = -2 / Vector.Scalar(v, v);

Matrix H = E + koef \* Vector.MultiplyVectors(v, v);

return H;

}

public static decimal CalcNorm(List<decimal> b)

{

double norm = 0;

foreach (var el in b)

norm += Math.Pow(Convert.ToDouble(el), 2);

norm = Math.Pow(norm, 0.5);

return Convert.ToDecimal(norm);

}

public static List<decimal> CalcV(List<decimal> b, int i)

{

var e = new List<decimal>();

var v = b;

for (int j = 0; j < b.Count; j++)

{

if (i > j)

v[j] = 0;

if (i == j)

e.Add(1);

else

e.Add(0);

}

var res = Vector.SumVec(v, Vector.DigitOnVector(Math.Sign(b[i]) \* CalcNorm(b), e));

return res;

}

private static void GenerateMatrix()

{

if (File.Exists("mtx.txt"))

File.Delete("mtx.txt");

var kek = new Random();

Console.WriteLine("Size?");

var n = Convert.ToInt16(Console.ReadLine());

var mtx = new Matrix(n);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (kek.Next(10) % 3 == 0)

{

var next = kek.Next(-20, 20);

mtx.mtx[i][j] = next;

File.AppendAllText("mtx.txt", next.ToString());

}

else

{

mtx.mtx[i][j] = 0;

File.AppendAllText("mtx.txt", "0");

}

if (j != n - 1)

File.AppendAllText("mtx.txt", " ");

}

if (i != n - 1)

File.AppendAllText("mtx.txt", "\n");

}

mtx.Print("Generated new mtx");

}

}

}

# Пример работы программы

### 6х6

Gen new mtx?

1

Matrix A:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,0000 0,0000 0,0000 0,0000 14,0000 0,0000

-1,000 0,0000 0,0000 0,0000 0,0000 17,0000

0,0000 13,0000 0,0000 0,0000 0,0000 13,0000

-2,000 0,0000 0,0000 12,0000 7,0000 0,0000

0,0000 0,0000 0,0000 0,0000 1,0000 8,0000

0,0000 14,0000 0,0000 0,0000 3,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,6222 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,381332286486536

-0,479846538772416

8,08808

-1,56578

0,00000

8,71024

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 0,0000 0,0000 0,0000 0,0000 0,0000

12,0047 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 0,0000 0,0000 0,0000 0,0000 0,0000

0,6222 -0,040 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,6737 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 -0,130 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,7256 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-6,1615171616437557788279445567

7,7532975527525583333121901869

5,3287596577846127742499705718

-0,9348652902614203679901029508

5,8045566226367672157974884752

-4,4193098427693750246071749851

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 0,0000 0,0000 0,0000 0,0000

12,0047 5,3196 0,0000 0,0000 0,0000 0,0000

0,0000 13,4377 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,0000 0,0000 0,0000 0,0000

0,6222 -0,040 0,5770 0,0000 0,0000 0,0000 0,0000

0,0000 0,6737 0,3966 0,0000 0,0000 0,0000 0,0000

0,0000 -0,130 -0,070 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,4320 0,0000 0,0000 0,0000 0,0000

0,0000 0,7256 -0,329 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,5649700857458783282530817920

-0,7109234823873345941254799744

0,6358721246582094064396596358

3,6699053242101821064670316136

1,5607097060572789079823223356

0,0053619920446073531704104389

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 0,0000 0,0000 0,0000

12,0047 5,3196 8,9665 0,0000 0,0000 0,0000

0,0000 13,4377 -8,704 0,0000 0,0000 0,0000

0,0000 0,0000 4,1392 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,1365 0,0000 0,0000 0,0000

0,6222 -0,040 0,5770 -0,172 0,0000 0,0000 0,0000

0,0000 0,6737 0,3966 0,1536 0,0000 0,0000 0,0000

0,0000 -0,130 -0,070 0,8866 0,0000 0,0000 0,0000

0,0000 0,0000 0,4320 0,3771 0,0000 0,0000 0,0000

0,0000 0,7256 -0,329 0,0013 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-1,1131230386934546594558413430

1,4006851549398347608198005097

0,0451052457016647674376795566

1,529172436275996119330011092

-2,5743816787766984660852171737

0,3589012999195549727449174952

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 4,0615 0,0000 0,0000

12,0047 5,3196 8,9665 -3,941 0,0000 0,0000

0,0000 13,4377 -8,704 -3,684 0,0000 0,0000

0,0000 0,0000 4,1392 12,0755 0,0000 0,0000

0,0000 0,0000 0,0000 3,5068 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,1365 -0,317 0,0000 0,0000

0,6222 -0,040 0,5770 -0,172 0,3994 0,0000 0,0000

0,0000 0,6737 0,3966 0,1536 0,0129 0,0000 0,0000

0,0000 -0,130 -0,070 0,8866 0,4361 0,0000 0,0000

0,0000 0,0000 0,4320 0,3771 -0,734 0,0000 0,0000

0,0000 0,7256 -0,329 0,0013 0,1023 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,2845037959505007813949964488

0,3580056426509653993798564957

-0,7235347057727547625003285010

0,0522221728099821847040116720

0,4356034447353199020043903082

0,7134192936497267365565501116

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 4,0615 -6,766 0,0000

12,0047 5,3196 8,9665 -3,941 6,3504 0,0000

0,0000 13,4377 -8,704 -3,684 7,3574 0,0000

0,0000 0,0000 4,1392 12,0755 -0,072 0,0000

0,0000 0,0000 0,0000 3,5068 4,7705 0,0000

0,0000 0,0000 0,0000 0,0000 1,1975 0,0000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,1365 -0,317 -0,238 0,0000

0,6222 -0,040 0,5770 -0,172 0,3994 0,2990 0,0000

0,0000 0,6737 0,3966 0,1536 0,0129 -0,604 0,0000

0,0000 -0,130 -0,070 0,8866 0,4361 0,0436 0,0000

0,0000 0,0000 0,4320 0,3771 -0,734 0,3638 0,0000

0,0000 0,7256 -0,329 0,0013 0,1023 0,5957 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,0000000000000484802491266061

-0,0000000000000610047478170448

-0,0000000000000418306905410029

0,0000000000000074254781596214

-0,0000000000000458789369250823

0,0000000000000346944012248047

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 4,0615 -6,766 10,4358

12,0047 5,3196 8,9665 -3,941 6,3504 10,9501

0,0000 13,4377 -8,704 -3,684 7,3574 8,4918

0,0000 0,0000 4,1392 12,0755 -0,072 5,7855

0,0000 0,0000 0,0000 3,5068 4,7705 0,9932

0,0000 0,0000 0,0000 0,0000 1,1975 0,0255

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,1365 -0,317 -0,238 0,4581

0,6222 -0,040 0,5770 -0,172 0,3994 0,2990 -0,576

0,0000 0,6737 0,3966 0,1536 0,0129 -0,604 -0,395

0,0000 -0,130 -0,070 0,8866 0,4361 0,0436 0,0702

0,0000 0,0000 0,4320 0,3771 -0,734 0,3638 -0,434

0,0000 0,7256 -0,329 0,0013 0,1023 0,5957 0,3278

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Result Q:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,7829 0,0318 -0,459 0,1365 -0,317 -0,238 0,4581

0,6222 -0,040 0,5770 -0,172 0,3994 0,2990 -0,576

0,0000 0,6737 0,3966 0,1536 0,0129 -0,604 -0,395

0,0000 -0,130 -0,070 0,8866 0,4361 0,0436 0,0702

0,0000 0,0000 0,4320 0,3771 -0,734 0,3638 -0,434

0,0000 0,7256 -0,329 0,0013 0,1023 0,5957 0,3278

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix Result H:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 4,0615 -6,766 10,4358

12,0047 5,3196 8,9665 -3,941 6,3504 10,9501

0,0000 13,4377 -8,704 -3,684 7,3574 8,4918

0,0000 0,0000 4,1392 12,0755 -0,072 5,7855

0,0000 0,0000 0,0000 3,5068 4,7705 0,9932

0,0000 0,0000 0,0000 0,0000 1,1975 0,0255

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H1:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-0,487 7,6544 1,5414 4,0615 -6,766 10,4358

12,0047 5,3196 8,9665 -3,941 6,3504 10,9501

0,0000 13,4377 -8,704 -3,684 7,3574 8,4918

0,0000 0,0000 4,1392 12,0755 -0,072 5,7855

0,0000 0,0000 0,0000 3,5068 4,7705 0,9932

0,0000 0,0000 0,0000 0,0000 1,1975 0,0255

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Matrix H after QR:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7,1924 14,2056 -2,442 -6,631 -11,712 -11,487

14,4116 -7,121 1,6461 2,2263 -4,774 -5,470

0,0000 0,0000 12,0000 0,5991 7,4144 -2,177

0,0000 0,0000 0,0000 2,9824 9,7117 -9,136

0,0000 0,0000 0,0000 0,0000 -2,054 2,1573

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

l0 = 16,034079782363463657827628369 + 0i

l1 = -15,962264909954736342172371631 + 0i

l2 = 11,999999522568942355159280821 + 0i

l3 = 2,9823885523308306300855308812 + 0i

l4 = -2,0542029473085096576909913722 + 0i

l5 = 0,0000000000000095342371078464 + 0i

## 10х10

Gen new mtx?

y

Size?

10

Matrix Generated new mtx:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0,0000 0,0000 -9,000 -7,000 0,0000 -11,000 -16,000 0,0000 -18,000 0,0000

0,0000 11,0000 -8,000 0,0000 0,0000 0,0000 0,0000 1,0000 0,0000 11,0000

-8,000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

0,0000 -18,000 0,0000 0,0000 -1,000 0,0000 13,0000 0,0000 0,0000 -12,000

-1,000 6,0000 0,0000 0,0000 0,0000 0,0000 8,0000 0,0000 18,0000 0,0000

-20,000 0,0000 0,0000 0,0000 0,0000 0,0000 8,0000 9,0000 0,0000 -4,000

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 -20,000 -13,000 0,0000 4,0000

0,0000 -7,000 15,0000 0,0000 0,0000 0,0000 19,0000 0,0000 0,0000 0,0000

0,0000 0,0000 0,0000 16,0000 15,0000 -8,000 0,0000 0,0000 0,0000 0,0000

0,0000 -18,000 6,0000 0,0000 0,0000 0,0000 0,0000 -5,000 13,0000 -12,000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

…

Matrix H after QR:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-24,127 6,0062 3,5082 -3,847 6,7980 6,3504 14,4623 -1,820 8,7420 -3,112

0,0000 5,5012 -4,256 -3,254 -14,137 -29,543 -7,871 -1,307 -11,841 -1,402

0,0000 -10,034 4,9683 -6,054 -10,645 -3,834 -2,538 2,2847 -7,028 0,2658

0,0000 0,0000 18,5888 3,5220 -8,369 -1,357 -0,390 -15,438 -0,491 -5,705

0,0000 0,0000 0,0000 -14,803 -7,888 -0,154 5,3648 5,8793 2,7929 -2,141

0,0000 0,0000 0,0000 0,0000 7,7833 -11,640 -1,676 10,7688 -3,995 -6,838

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 6,9805 -12,127 24,4085 -9,917

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 6,5662 -3,451 11,0469 4,6891

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 5,5318 3,8094

0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 -0,398

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

l0 = 5,50115962073909783555657707 + 0i

l1 = -24,12690378924860216444342293 + 0i

l2 = 4,245138932552991468970600711 + 10,5840100510229i

l3 = 4,245138932552991468970600711 - 10,5840100510229i

l4 = -8,240369794729807356581869073 + 0i

l5 = -11,287105563735667356581869073 + 0i

l6 = 1,7645466507068532857957954791 + 7,2405301374205i

l7 = 1,7645466507068532857957954791 - 7,2405301374205i

l8 = 5,5318058993099945248093806287 + 0i

l9 = -0,3979575388475083335659096812 + 0i

## 50x50

0 1 -19 0 5 0 -19 9 -10 0 17 -9 -14 0 0 12 -19 -16 0 0 0 -14 0 -8 0 0 0 -9 -12 0 0 0 2 -3 0 -9 -1 0 -11 0 3 0 10 -5 -1 0 -3 0 0 0

0 0 3 -16 0 0 0 -18 0 13 0 -6 0 0 0 0 0 -2 0 0 -15 8 18 0 0 0 -13 -11 0 0 0 3 0 0 -17 0 -10 0 -18 8 0 10 0 0 0 -6 8 -3 -2 -19

0 0 12 0 0 -18 0 -19 0 0 0 -19 -1 0 5 0 0 -7 0 -4 5 0 0 0 0 0 0 16 13 0 -5 0 0 0 0 4 0 0 0 -5 0 10 0 0 0 0 -13 2 0 9

0 11 0 0 0 0 0 0 0 0 12 -17 -19 14 0 -13 0 0 8 0 0 0 16 15 13 0 -8 0 0 0 -3 8 0 16 0 0 18 15 -20 0 13 9 9 0 0 11 0 0 0 8

0 0 9 9 0 0 0 0 0 0 -19 19 0 0 -1 -15 0 0 0 0 11 0 10 -8 0 -6 -9 0 8 13 0 0 0 0 11 0 10 -11 0 0 0 0 0 0 0 14 0 1 -1 5

0 0 0 10 0 -15 0 -13 -20 0 0 -19 -18 0 0 -4 -13 11 -18 -9 0 -15 0 0 0 0 0 1 0 -11 -3 -17 0 -1 0 0 0 0 -15 0 0 6 -18 -8 0 14 0 0 0 0

-16 -20 -13 10 7 0 13 -6 0 1 6 0 -9 0 17 3 0 0 0 0 0 0 0 0 0 0 0 1 -7 -5 -10 -18 0 0 3 0 2 0 18 -10 0 2 0 7 0 0 -12 0 0 -16

8 -13 7 1 0 0 0 -15 8 0 0 0 0 -16 0 0 0 -5 0 0 6 -10 0 18 0 0 0 -9 0 0 -9 0 0 0 0 0 -7 -18 -20 17 0 0 0 0 0 0 0 0 0 -17

0 0 6 0 0 -12 -1 -8 0 0 0 -7 0 0 0 -18 -20 -1 0 0 11 5 -2 1 6 0 0 12 0 0 0 0 0 0 0 0 0 0 14 -6 16 -20 0 0 0 0 -7 -6 15 0

-8 -11 0 0 0 0 0 0 0 -12 0 6 0 18 0 0 -15 0 17 0 -11 0 -18 0 0 0 0 4 0 0 0 0 -11 0 6 0 6 0 -16 17 -13 0 0 0 -4 0 -4 0 0 0

15 -12 0 0 -9 15 0 0 0 0 2 0 17 0 0 -9 -17 1 -9 13 -1 0 0 0 3 0 -10 17 0 10 9 5 -19 0 0 0 4 -16 0 0 -20 -19 -10 -18 0 0 0 0 -8 0

19 0 4 0 0 9 0 5 14 0 -2 0 -8 4 0 -1 6 0 0 0 0 1 0 0 -17 0 0 0 0 3 0 5 0 0 0 0 -8 0 -10 0 0 13 -15 0 0 6 -17 0 17 0

-9 0 -10 -20 0 0 0 0 0 0 13 0 0 0 0 0 -9 -16 0 0 0 1 0 0 18 0 -20 19 0 0 0 -13 15 4 0 0 0 0 -15 0 -11 -15 0 0 -8 0 0 0 6 0

1 0 -15 18 0 -12 0 9 0 1 0 0 0 -20 0 14 -8 0 17 0 -2 -20 0 15 0 -3 0 0 9 -1 8 7 0 -17 12 0 0 0 12 0 0 0 0 0 0 9 0 5 0 -5

0 0 0 -2 0 -17 -11 -4 -9 0 0 0 -4 0 0 0 0 0 0 -2 15 0 -13 0 0 0 0 -13 0 0 0 0 16 11 8 0 -6 13 0 0 0 7 0 -20 0 0 0 0 0 0

14 0 0 11 0 11 0 0 0 0 0 0 16 -19 19 0 -8 0 0 0 0 0 0 9 0 -10 0 1 -11 -8 -8 -10 -12 0 -10 0 0 0 0 0 0 0 0 -6 0 0 0 0 -12 0

16 0 -17 0 7 0 8 0 -12 -16 -11 0 0 0 0 0 12 15 0 0 0 0 0 17 -7 15 0 -14 0 0 -3 0 0 0 0 15 0 0 0 -4 -7 0 16 0 0 -4 0 -2 0 -18

0 0 0 0 4 -1 0 0 0 0 0 8 2 0 -20 0 0 0 0 0 13 0 0 0 -10 0 0 0 -11 7 0 13 1 0 -6 0 0 12 0 19 -15 0 0 -9 0 0 0 -5 0 -13

0 12 3 -5 0 0 0 0 -9 0 0 0 17 0 19 14 0 -5 -1 -5 0 0 -13 6 0 0 0 0 -3 0 0 -20 -6 3 0 0 -12 0 0 3 0 15 0 0 1 0 0 -14 14 -6

1 0 0 19 18 0 0 0 -1 19 0 0 4 13 0 0 4 12 0 -12 0 0 0 0 0 10 -4 0 0 0 0 -17 -19 0 0 7 -11 1 -5 0 18 0 0 16 0 2 13 6 0 0

3 0 -1 -10 -11 0 1 0 0 0 -14 0 -9 0 0 -14 0 0 11 9 0 -15 0 0 12 15 0 -19 8 0 -9 0 0 7 -8 0 3 0 0 -11 0 0 0 0 -3 0 -8 0 0 0

14 0 0 0 -1 0 -18 9 14 0 -16 0 5 16 0 0 0 0 4 0 13 0 -2 6 7 3 0 -16 0 0 0 0 0 0 6 0 0 0 0 15 14 0 0 -9 0 0 0 -13 0 0

0 0 0 -6 0 0 -20 0 -3 1 3 0 -11 0 0 13 0 18 -5 0 0 0 -18 0 0 0 15 16 0 0 -5 0 18 0 0 4 0 -18 0 0 0 -5 0 0 0 0 7 13 9 -4

0 -10 18 0 0 19 5 17 6 2 5 0 19 9 2 -2 -15 19 -12 4 0 1 0 0 -5 11 0 0 0 -4 0 0 0 0 0 0 0 -6 0 14 5 9 0 -14 0 0 0 0 4 0

15 12 0 0 0 15 0 0 18 -15 0 0 0 1 -15 0 0 0 -11 0 4 -1 17 0 -6 0 -18 -8 0 0 -7 14 0 8 0 -18 0 0 -12 -2 -14 -9 0 0 0 8 0 0 0 2

0 -5 12 0 0 -8 15 -16 0 0 0 -14 16 12 -2 10 0 -15 0 0 0 0 0 0 -4 -5 3 13 -5 0 0 0 0 1 3 0 0 0 16 -18 14 0 -3 0 0 -5 0 2 0 0

-6 19 -13 18 0 0 0 14 0 0 0 0 0 -7 9 0 0 18 11 0 2 13 -16 0 0 14 0 0 6 -6 0 0 -18 -3 0 0 0 0 0 -13 -16 0 0 0 8 0 0 -17 -7 17

4 -6 0 -4 9 0 0 12 0 0 0 0 0 0 0 0 -15 0 -17 -7 3 -11 0 0 0 10 7 0 0 0 0 15 -1 0 0 0 15 -7 17 12 -5 0 0 -14 0 0 0 0 3 0

0 6 0 0 0 0 0 0 0 0 -15 -6 0 0 2 17 -8 0 -14 0 0 -1 -1 0 13 0 15 0 -2 -11 0 0 0 0 0 6 0 0 14 0 0 -16 0 0 -3 0 0 11 0 0

0 0 -2 0 0 0 0 0 0 6 -6 0 0 -18 10 -17 10 0 0 0 19 -17 -20 3 0 6 10 -4 0 0 0 12 -14 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 -20

0 0 0 0 0 0 0 0 0 -9 -2 0 0 0 0 0 0 0 -14 0 0 0 -19 0 0 0 0 0 12 0 -20 -14 0 -6 13 0 10 10 0 18 0 4 0 0 -1 0 0 4 -7 0

6 -15 0 0 6 -20 0 0 0 0 -16 -8 0 0 0 -13 0 -6 0 0 0 10 -1 -18 0 4 0 -7 0 4 2 0 1 0 -8 0 0 -8 0 -19 0 0 0 -4 0 -12 0 0 0 0

0 0 0 2 0 0 0 0 0 0 0 0 0 0 4 0 -6 0 -6 0 0 -7 0 -10 8 0 11 -16 0 0 0 0 -5 1 0 0 0 0 0 -14 0 0 0 0 8 -11 0 0 -11 0

0 0 -15 0 0 10 -18 0 10 -17 0 0 0 17 0 0 0 0 0 0 0 0 0 17 0 0 -7 0 -5 0 -4 0 0 0 0 0 12 0 -17 -9 -19 10 0 0 -15 0 0 0 0 0

0 10 1 0 0 -14 0 0 0 0 0 -20 2 0 14 17 -4 6 0 0 -7 -5 0 0 -18 -4 5 10 -12 12 0 -15 0 0 0 0 -12 0 0 15 3 13 -15 -13 0 18 0 0 0 0

0 10 13 0 9 0 0 0 9 0 -10 0 0 -19 16 -17 0 7 -14 10 -5 0 -16 3 -10 0 0 0 5 0 0 -6 -14 0 0 -13 8 6 0 0 3 0 16 0 0 0 0 0 0 0

0 -13 -2 -13 -9 0 0 0 -19 0 -5 16 0 0 0 0 0 0 0 0 0 0 0 0 -16 0 0 0 0 0 0 19 0 -16 1 0 0 11 0 0 5 -6 0 7 0 0 -17 0 -16 2

0 0 0 -15 0 0 -18 0 9 0 0 0 0 2 0 0 0 0 0 0 -1 0 -20 -16 -1 -20 0 0 0 0 10 0 0 0 17 0 0 0 10 0 2 8 0 0 0 -17 -20 0 0 0

0 0 0 0 0 0 0 0 0 -13 0 0 0 18 -20 0 -10 -7 0 -18 -16 0 0 0 0 0 0 -8 3 0 5 14 0 0 6 -19 0 0 2 0 0 -12 6 0 0 0 0 0 0 18

0 0 0 0 -10 -7 -18 0 0 0 -18 0 14 -5 -7 0 17 0 0 8 -2 0 0 13 0 0 -17 11 0 0 0 0 -4 0 10 0 19 0 0 0 0 3 10 4 3 0 0 0 0 -7

0 0 3 -19 8 0 0 -9 -13 0 0 0 0 2 8 0 0 17 0 0 5 3 -4 0 0 18 0 0 0 10 0 0 0 14 -7 0 -14 0 -20 -1 0 0 0 -19 0 -2 2 -7 0 0

0 0 5 8 0 0 -5 0 10 -3 0 -17 0 0 -20 15 2 0 0 0 11 0 0 4 -3 -19 5 -16 14 -3 -19 -1 0 0 0 -14 0 0 15 0 -11 9 0 0 11 0 0 -12 0 16

-20 -15 0 0 0 0 0 0 12 -2 -5 10 17 0 0 0 0 -9 0 0 17 15 0 0 0 0 -10 -4 5 0 0 0 -17 -15 -4 0 0 0 -13 0 0 0 0 -12 0 0 -1 -14 0 0

0 -5 -8 0 4 -6 0 -17 0 0 -9 0 0 0 0 0 0 0 0 3 -11 -18 0 0 -18 0 0 0 0 0 0 0 -19 0 0 -18 0 0 0 0 -8 0 0 0 0 -17 -10 0 0 0

0 0 0 0 -3 11 0 0 -16 0 0 0 0 0 -16 0 0 0 10 0 18 0 0 0 0 9 0 0 15 0 0 -2 0 0 0 -14 0 -5 0 0 -20 0 -18 0 0 17 0 0 19 9

-11 -17 -9 0 0 -9 1 0 -12 -7 0 0 4 -16 0 0 0 0 0 0 0 0 0 0 0 0 0 -9 0 0 3 11 0 14 0 0 0 0 -12 -15 -15 0 -1 15 0 0 0 0 0 12

0 0 8 0 -20 -7 0 0 12 0 -6 0 -14 -2 10 -5 0 -8 0 5 0 0 -20 13 5 -16 11 16 0 0 -13 0 -14 0 0 -7 -16 -2 8 0 -2 0 0 0 0 1 0 0 -5 0

0 -12 0 -15 0 -1 4 0 -19 0 -14 -5 12 0 -18 -11 0 -12 0 5 9 -18 4 2 -9 12 0 -15 12 0 0 0 0 0 0 0 -14 0 0 0 0 18 -8 -17 16 0 13 0 -10 0

0 -2 0 0 -13 -13 12 17 0 0 0 0 0 13 0 -12 0 0 0 0 0 0 0 0 0 0 0 6 0 7 11 0 0 -10 0 7 0 -14 0 -3 1 0 0 -8 19 -17 0 -5 -11 0

-17 0 -18 0 0 2 0 10 0 1 0 0 0 11 -4 17 0 0 -11 -13 -3 0 0 2 -6 -20 -11 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 -19 15 0 0 14

…

l0 = 29,790276561907396568371207422 + 0i

l1 = -58,267783913506203431628792578 + 0i

l2 = 39,697239805320883728732532768 + 0i

l3 = -28,410149506815916271267467232 + 0i

l4 = 11,779793159404628865846113280 + 0i

l5 = -22,610514430423571134153886720 + 0i

l6 = 10,398142941586322988869703910 + 0i

l7 = -11,653399870258277011130296090 + 0i

l8 = 10,806362171170478656382199571 + 0i

l9 = -14,284577020825121343617800429 + 0i

l10 = 10,064049929266262574733346848 + 0i

l11 = -7,1803830598676374252666531516 + 0i

l12 = -16,279127912844376385866513 + 17,7911525660139i

l13 = -16,279127912844376385866513 - 17,7911525660139i

l14 = -25,406772605261159427136078158 + 25,28145542760735i

l15 = -25,406772605261159427136078158 - 25,28145542760735i

l16 = 32,699135209162776093281296669 + 0i

l17 = -27,390523175277523906718703331 + 0i

l18 = 8,224566731868819661134518398 + 39,24021358731115i

l19 = 8,224566731868819661134518398 - 39,24021358731115i

l20 = 24,397598624154459452664340332 + 0i

l21 = -18,683587645744840547335659668 + 0i

l22 = 10,604164869593520188504682486 + 0i

l23 = -41,435314363078479811495317514 + 0i

l24 = 31,768289180856923985236387860 + 8,58442387141215i

l25 = 31,768289180856923985236387860 - 8,58442387141215i

l26 = -4,3046688349297320802836721915 + 32,7262088793088i

l27 = -4,3046688349297320802836721915 - 32,7262088793088i

l28 = -24,358768826710426047787483186 + 15,90844580984685i

l29 = -24,358768826710426047787483186 - 15,90844580984685i

l30 = -13,087764581790442115868985453 + 28,70318514980495i

l31 = -13,087764581790442115868985453 - 28,70318514980495i

l32 = 27,673801473469551061316230293 + 3,57962552045425i

l33 = 27,673801473469551061316230293 - 3,57962552045425i

l34 = 25,516143854990786820386753663 + 0i

l35 = -10,321435057371113179613246337 + 0i

l36 = 26,929082143016063676736330580 + 0i

l37 = -21,080025413757836323263669420 + 0i

l38 = 10,782867049390254612081424594 + 22,584378522032i

l39 = 10,782867049390254612081424594 - 22,584378522032i

l40 = 17,899113637717459671250249292 + 0i

l41 = -22,998787174617240328749750708 + 0i

l42 = 19,054018335345711147661488079 + 0i

l43 = -19,271029078954488852338511921 + 0i

l44 = -17,346610944492839999440822932 + 4,313300368155515i

l45 = -17,346610944492839999440822932 - 4,313300368155515i

l46 = -7,2143270769583587320821122261 + 0i

l47 = 6,2877976571590672349117548036 + 0i

l48 = -6,6482467212447327650882451964 + 0i

l49 = -4,8044568502076770588095757882 + 0i

# Выводы

Выполнив данную курсовую работу, я научился применять метод Арнольди для такой задачи, как поиск собственных значений. Этот метод помогает ускорить процесс их нахождения, т.к. преобразует исходную матрицу к матрице в виде верхней форме Хессенберга, что позволяет ускорить работу QR-алгоритма для поиска нужных нам неизвестных.

Работа была интересной, в ней применялись знания, полученные в течение всего изученного курса. В данном случае это QR-алгоритм. Этот факт мне понравился, потому что пройденные методы интересно было куда-нибудь применить.

По окончанию данной работы я глубже разобрался в численных методах и их применении на практике. Они помогут мне в дальнейшем обучении.

# Литература

* <https://en.wikipedia.org/wiki/Arnoldi_iteration>

### «[Численные методы», Формалев В.Ф., Ревизников Д.Л., 2006](http://nashol.com/2016011788006/chislennie-metodi-formalev-v-f-reviznikov-d-l-2006.html)

* «Вычислительная линейная алгебра с примерами на MATLAB», В.И.Горбаченко, Санкт -Петербург БХВ-Петербург, 2011