БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа
по вычислительным методам алгебры на тему:

Нахождение собственных значений и собственных векторов методом А.Н. Крылова

Выполнил:
Архангельский И.А.

Проверил:
Кондратюк А.П.

Минск, 2012

Входные и выходные данные.

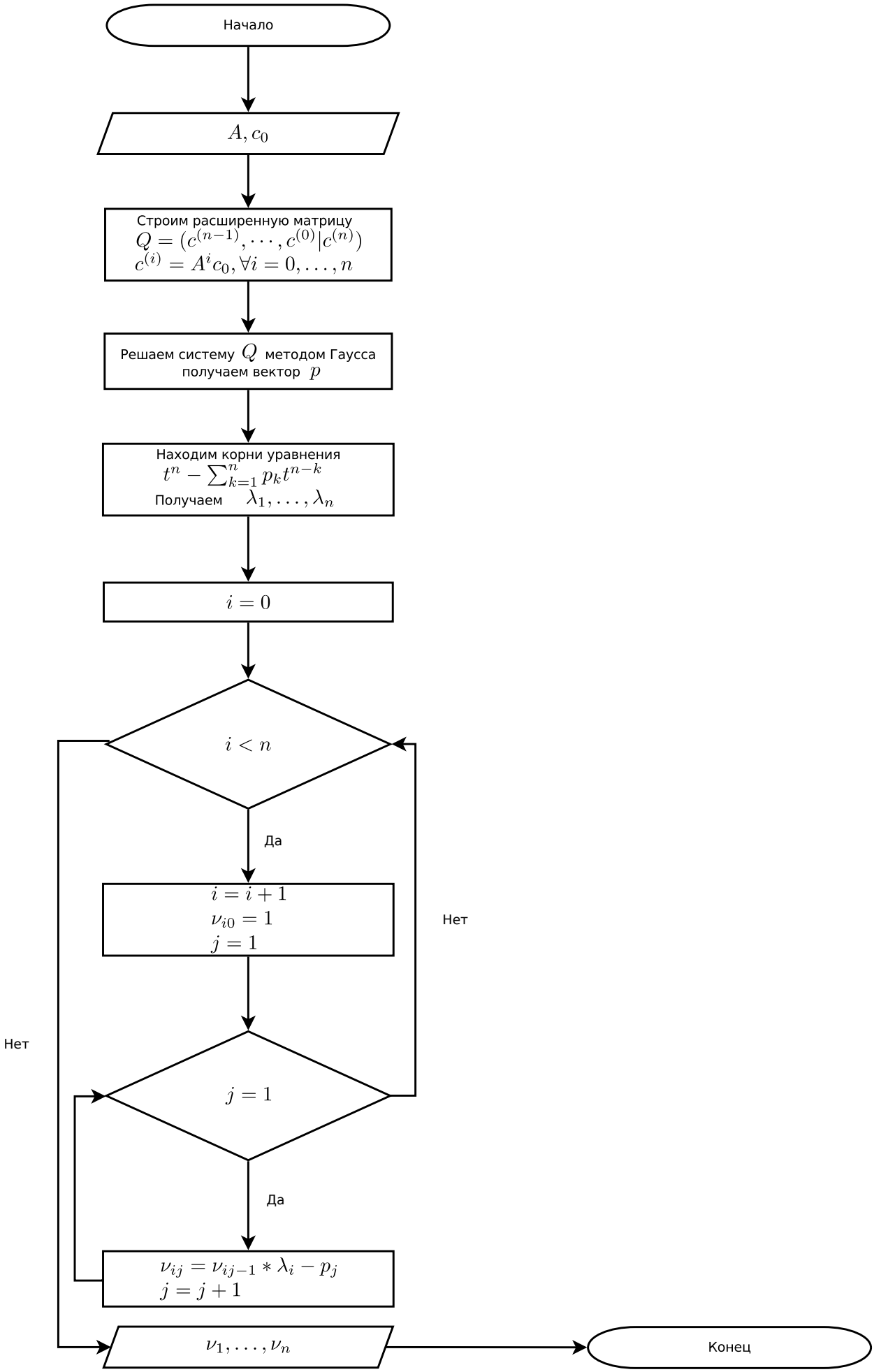
Входные данные

Входной файл содержит размерность матрицы, матрицу A и начальный вектор c_0

Выходные данные

В случае если возможно найти многочлен степени n, программа выводит коэффициенты многочлена и ожидает ввода n его корней (собственных значений матрицы A). Далее выполняется расчет и вывод собственных векторов матрицы A.

Блок-схема


Начало
A, c_0
Строим расширенную матрицу
$Q = (c^{(n-1)}, \cdots, c^{(0)} | c^{(n)})$
$c^{(i)} = A^i c_0, \forall i = 0, \ldots, n$
Решаем систему Q методом Гаусса
получаем вектор p
Находим корни уравнения
$t^n - \sum_{k=1}^{n} p_k t^{n-k}$
Получаем $\lambda_1, \ldots, \lambda_n$
$i = 0$
$i < n$
Да
Нет
$i = i + 1$
$\nu_{i0} = 1$
$j = 1$
$j = 1$
Нет
Да
$\nu_{ij} = \nu_{ij-1} * \lambda_i - p_j$
$j = j + 1$
$\nu_1, \ldots, \nu_n$
Конец

Реализация

krylov.h

```
#ifndef KRYLOV_H
#define KRYLOV_H

#include <cstdio>

class Krylov
{
    double** matrix;
    double* c_0;
    int size;
    double* mulMatrix (double* y);
public:
    Krylov(char* filename);
    double* getPoly ();
    int getSize (){return size;}
    ~Krylov();
    double* getEigenVector(double*, double);
};

#endif // KRYLOV_H
```

krylov.cpp

```
#include "krylov.h"
#include "gauss/gauss.h"

Krylov::Krylov(char *filename)
{
    FILE* fin = fopen(filename,"r");
    fscanf (fin,"%d\n",&size);
    matrix = new double* [size];
    for (int i=0;i<size;i++)
    {
        matrix[i]= new double [size];
        for (int j=0;j<size;j++)
        {
            fscanf(fin,"%lf",&matrix[i][j]);
        }
    }
    c_0 = new double [size];
    for (int i=0;i<size;i++)
    {
        fscanf (fin,"%lf",&c_0[i]);
    }
    fclose (fin);
}

double* Krylov::mulMatrix(double* x)
    {
        double* res = new double[size];
        for (int i = 0; i < size; i++)
        {
            res[i] = 0;
            for (int j = 0; j <size; j++) {res[i] += x[j] * matrix[i][j];}
        }
        return res;
    }

Krylov::~Krylov ()
{
    for (int i=0;i<size;i++) delete [] matrix[i];
    delete [] matrix;
    delete [] c_0;
}

double* Krylov::getPoly ()
{
    double** Q = new double* [size];
    for (int i=0;i<size;i++)
    {
        Q[i] = new double[size+1];
    }
    double* c = new double [size];
    for (int i=0;i<size;i++)
```

```
52      {
53          c[i]= c_0[i];
54      }
55      for (int i=0;i<size+1;i++)
56      {
57          for (int j=0;j<size;j++)
58          {
59              Q[j][((size+1)+size-1-i)%(size+1)] = c[j];
60          }
61          c = mulMatrix (c);
62
63      }
64      Gauss gauss = Gauss(Q,size);
65      int m = 0;
66      double* p = gauss.solve (m);
67      if (m==size)
68      {
69          return p;
70      }
71      else
72      {
73          printf ("Only %d steps on get Poly\n",m);
74      }
75      for (int i=0;i<size;i++) delete [] Q[i];
76      delete [] Q;
77      delete [] c;
78      return NULL;
79  }
80
81  double* Krylov::getEigenVector (double* poly, double eigenVal)
82  {
83      double* vector = new double [size];
84      vector[0]=1;
85      for (int i=1;i<size;i++)
86      {
87          vector[i] = vector[i-1]*eigenVal-poly[i-1];
88      }
89      return vector;
90  }
```

main.cpp

```
1   #include <cstdlib>
2   #include <iostream>
3   #include "krylov.h"
4   #include <math.h>
5   #include <gsl/gsl_poly.h>
6
7   using namespace std;
8
9   double const E = 0.00000000001;
10
11  int main(int argc, char *argv[])
12  {
13      for (int i=1;i<argc;i++)
14      {
15          printf ("RUNNING ON TEST: %s\n",argv[i]);
16          Krylov krylov =  Krylov(argv[i]);
17          double * poly = NULL;
18
19          poly = krylov.getPoly ();
20          if (poly==NULL)
21          {
22              printf ("Some errors on getPoly\n");
23              return 200;
24          }
25          printf ("t^%d ",krylov.getSize ());
26          for (int i=0;i<krylov.getSize();i++)
27          {
28              if (fabs(poly[i])<E) continue;
29              if (poly[i]<0) printf ("+");
30              if (i==krylov.getSize ()-1)
31              {
32                  printf ("%12.6lf ",-1*poly[i]);
33              }
34              else
35                  if (i==krylov.getSize ()-2)
36                  {
37                      printf ("%12.6lf*t ",-1*poly[i]);
```

```
38                  }
39                  else
40                  {
41                       printf ("%12.6lf*t^%d ",-1*poly[i],krylov.getSize ()-i-1);
42                  }
43
44          }
45          printf ("\n");
46          printf ("roots([1, ");
47          for (int i=0;i<krylov.getSize ();i++)
48          {
49              printf ("%12.6lf",-1*poly[i]);
50              if (i!=krylov.getSize ()-1) printf (", ");
51              else printf (" ");
52          }
53          printf ("])\n");
54          double* eigenVal = new double [krylov.getSize ()];
55          for (int i=0;i<krylov.getSize ();i++)
56          {
57              printf("l%d = ",i+1);
58              scanf("%lf",&eigenVal[i]);
59          }
60
61          for (int i=0;i<krylov.getSize ();i++)
62          {
63              double* eigenVector = krylov.getEigenVector(poly,eigenVal[i]);
64              printf ("EigenValue: %12.6lf  EigenVector: (\t",eigenVal[i]);
65              for (int j=0;j<krylov.getSize ();j++) printf ("%5.3lf  ",eigenVector[j]);
66              printf (")\n");
67              delete [] eigenVector;
68          }
69
70          delete[] poly;
71      }
72      return 0;
73
74  }
```

Тестовые данные

Матрица:

$$A = \begin{pmatrix} -5.509882 & 1.870086 & 0.422908 & 0.008814 \\ 0.287865 & -11.811654 & 5.711900 & 0.058717 \\ 0.049099 & 4.308033 & -12.970687 & 0.229326 \\ 0.006235 & 0.269851 & 1.397369 & -17.596207 \end{pmatrix}$$

Начальный вектор:

$$c_0 = (1, 0, 0, 0)$$

Полученный многочлен:

$$\varphi(t) = t^4 + 47.888430 * t^3 + 797.278765 * t^2 + 5349.455515 * t + 12296.550566$$

Корни многочлена(собственные значения):

$$\lambda_1 = -17.8633, \lambda_2 = -17.1524, \lambda_3 = -7.5740, \lambda_4 = -5.2987$$

Собственные вектора, соответсвующие собственным значениям:

$$\nu_1 = (1.000, 30.025, 260.931, 688.369)$$

$$\nu_2 = (1.000, 30.736, 270.082, 716.900)$$

$$\nu_3 = (1.000, 40.314, 491.937, 1623.523)$$

$$\nu_4 = (1.000, 42.590, 571.609, 2320.673)$$