Лабораторная работа 5

«метод Данилевского»

Выполнил

Студент 1 группы, 2 курса

Быченок Егор

1. **Постановка задачи**

Разработать программу приведения матрицы к канонической форме Фробениуса методом Данилевского (регулярный случай); сохранить матрицы, используемые для получения собственных векторов.

Для вычислений:

* *n=*4 – порядок матрицы;
* при заполнении матрицы *A* использовать случайные числа из диапазона от −50 до 50. Для получения случайных чисел использовать библиотечную функцию rand(), подключив хедер stdlib.h;
* использовать тип float;
* «ведущий элемент» ** на (*n*–*k*+1)-м шаге, *k=n*, *n*–1,... , 2, (*=*) должен быть отличным от нуля (рассматривается только такой регулярный случай);
* считать ** равным нулю, если ;
* если , то выдать соответствующее сообщение и заново заполнить матрицу *A*.

Программно реализовать для рассматриваемого примера алгоритм приведения матрицы к канонической форме Фробениуса. Порядок матрицы *n* должен быть в коде параметром (это требование не является обязательным). Сохранить матрицы *Mn–*1, *Mn–*2,... , *M*1, используемые для получения собственных векторов (понадобятся для другой работы). Вывести на печать матрицу *A* (входные данные), полученную каноническую форму Фробениуса Φ, матрицы *Mn–*1, *Mn–*2, ... , *M*1, коэффициент *p*1 (полученный из формы Фробениуса), след матрицы Sp *A* (для контроля вычислений: должно приближенно выполняться равенство *p*1*=*Sp *A=a*1,1+*a*2,2+... +*an,n*).

Содержание работы должно включать следующие пункты.

1. Постановка задачи.
2. Входные данные.
3. Листинг программы.
4. В листинге программы обязательны подробные комментарии.
5. Выходные данные.
6. **Входные данные**

n = 4

A =

[

27 24 15 -22

-40 28 9 -7

-17 -2 14 -6

-15 -43 47 -26

]

1. **Листинг программы**

bool getFNormalFrom() { *// получение нормальной формы матрицы, возвращает true, если преобразование прошло удачно,*

*// иначе - false*

int i;

**for** (i = fSize - 1; i > 0; --i) {

**if** (fabs(a[i][i - 1]) > 1e-8) {

generateMatriciesToMultiply(i - 1); *// вычисление матриц M и M^-1*

*// умножаем слева на m^-1, справа на m*

a = invertM[i - 1] \* a;

a = a \* m[i - 1];

} **else** { *// если получили ведущий 0*

**return** false;

}

}

**return** true;

}

float getTrace() { *// вычисление следа матриц*

float sum = 0;

**for** (int i = 0; i < a.size(); ++i) {

sum += a[i][i];

}

**return** sum;

}

void generateMatriciesToMultiply(int i) { *// вычисление дополнительных матриц*

**for** (int j = 0; j < fSize; ++j) {

m[i][j][j] = 1; *// диагональ*

m[i][i][j] = -a[i + 1][j] / a[i + 1][i]; *// i-ая*

*строка, получается из i+1 мастрицы a*

invertM[i][j][j] = 1; *// диагональ*

invertM[i][i][j] = a[i + 1][j]; *// i-ая строка,*

*получается из i+1 мастрицы a*

**if** (i == j) {

m[i][i][j] /= -a[i + 1][i];

}

}

}

1. **Выходные данные**

Sp A = 43

F = [

43.00 -616.00 -66658.99 252849.03

1.00 0.00 0.00 0.00

0.00 1.00 0.00 0.00

0.00 0.00 1.00 0.00

]

M3 = [

1.000000 0.000000 0.000000 0.000000

0.000000 1.000000 0.000000 0.000000

0.319149 0.914894 0.021277 0.553191

0.000000 0.000000 0.000000 1.000000

]

M2 = [

1.000000 0.000000 0.000000 0.000000

0.328409 -0.000619 -0.015484 0.231724

0.000000 0.000000 1.000000 0.000000

0.000000 0.000000 0.000000 1.000000

]

M1 = [

0.000016 0.000018 -0.012942 0.201434

0.000000 1.000000 0.000000 0.000000

0.000000 0.000000 1.000000 0.000000

0.000000 0.000000 0.000000 1.000000

]