Белорусский государственный университет

Факультет прикладной математики и информатики

Кафедра технологий программирования

Стефанович Константин Андреевич

**Лабораторная работа №4**

**Метод Данилевского решения Проблемы Собственных Значений**

студента 2 курса 6 группы

**Преподаватель**

***Радкевич Елена Владимировна*** Ассистент кафедры вычислительной математики ФПМИ

Минск, 2016

Оглавление

[1. Техническое задание 3](#_Toc463559007)

[2.Алгоритм решения и формулы 3](#_Toc463559008)

[3. Листинг программы 4](#_Toc463559009)

[4.Результаты и вывод 5](#_Toc463559010)

# 1. Техническое задание

1. Построить характеристический многочлен матрицы А.
2. Найти
3. Найти собственные векторы матрицы А

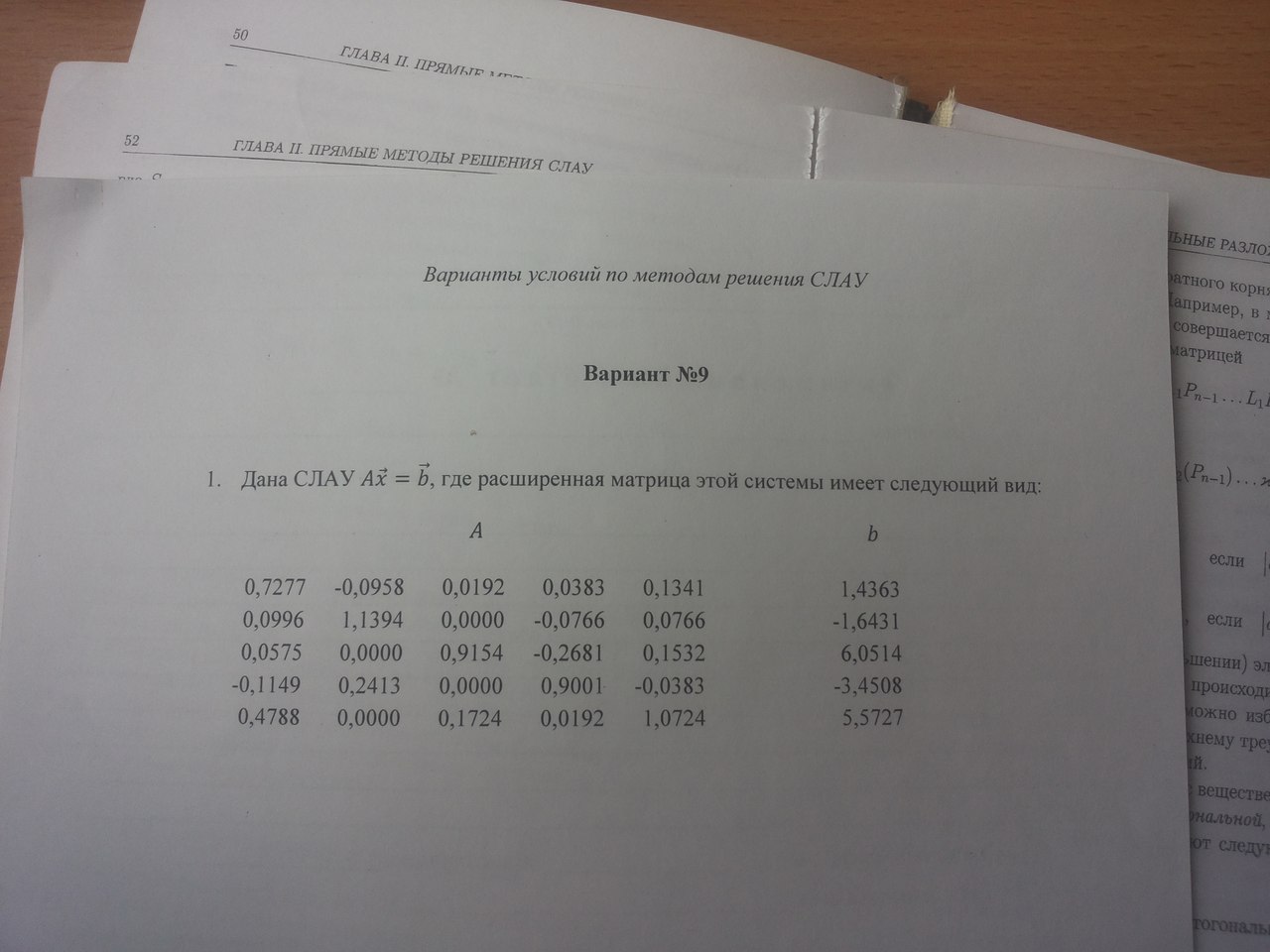


Рис.1. Данные для ввода.

# 2.Алгоритм решения и формулы

Метод Данилевского является прямым методом решения полной задачи собственных значений (т.е. позволяет весь спектр). Метод построен на том факте, что преобразование подобия S-1AS не изменяет характеристического многочлена. С помощью такого преобразования исходная матрица А приводится к канонической форме Фробениуса:

,

После разложения определителя получим:

() =

Матрица *A* приводится к *Ф*, в результате последовательного домножения справа на *Mn-1*и слева на , а *S* можно получить как *S = Mn-1 Mn-2.. Mn-1*

Из Pn( находим находим собственный вектор матрицы Ф: *y =* (, далее находим собственные векторы матрицы *А* из *x = Sy = Mn-1 Mn-2.. Mn-1y.*

# 3. Листинг программы

//1.py

**from** numpy **import** \*  
**from** numpy.linalg **import** \*  
**from** sympy.solvers **import** solve  
**from** sympy **import** Symbol  
inp = open(**'in.txt'**, **'r'**)  
A = []  
n = 5  
**for** line **in** inp:  
 temp = [float(x) **for** x **in** line.split()]  
 A.append(temp)  
a = array(A)  
f = a  
print(**'A:\n {}'**.format(a))  
s = identity(n)  
**for** i **in** range(n - 1):  
 m = identity(n)  
 m[n - 2 - i][:] = f[n - 1 - i][:]  
 f = dot(m, f)  
 f = dot(f, inv(m))  
 s = dot(s, inv(m))  
print(**'F:\n {}'**.format(f))  
p = f[0][:]  
print(**'p:\n {}'**.format(p))  
x = Symbol(**'x'**)  
Lambda = solve(x \*\* 5 - p[0] \* x \*\* 4 - p[1] \* x \*\* 3- p[2] \* x \*\* 2- p[3] \* x - p[4],x)  
print(**'lambda:\n {}'**.format(Lambda))  
l = Lambda[0]  
print(**' l{0}: {1}'**.format(0,l))  
y = [l \*\* i **for** i **in** range(n - 1, -1, -1)]  
print(**' x{0}: {1}'**.format(0,dot(s, y)))  
print(**"Остальные собственные вектора будут комплексными"**)

# 4.Результаты и вывод

Все приведённые методы имеют много общего, но на практике лучше всего использовать метод Гаусса-Зейделя, так как он требует намного меньшего числа итераций.

В результате работы программы получается следующий вывод:



Рис.2. Окно вывода