МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

**Определение собственных векторов матрицы методом**

**Крылова**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №7

по дисциплине

**Вычислительная Математика**

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

СТУДЕНТ:

Цветков Николай Максимович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

[Цель 3](#_Toc71824175)

[Постановка задачи 4](#_Toc71824176)

[Теоретические сведения 5](#_Toc71824177)

[Листинг разработанной программы 10](#_Toc71824178)

[Результаты работы программы 14](#_Toc71824179)

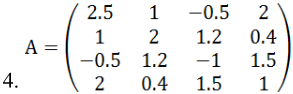
[Вывод 15](#_Toc71824180)

# Цель

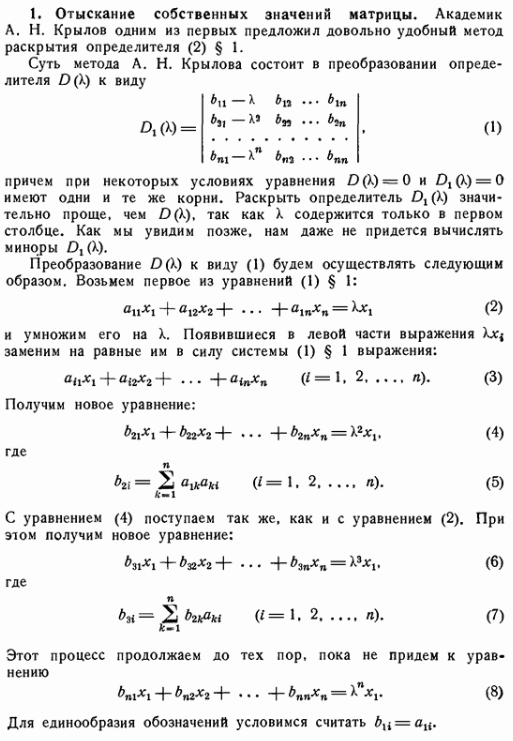
Закрепление знаний и умений определения собственных числе и векторов матрицы методом Крылова.

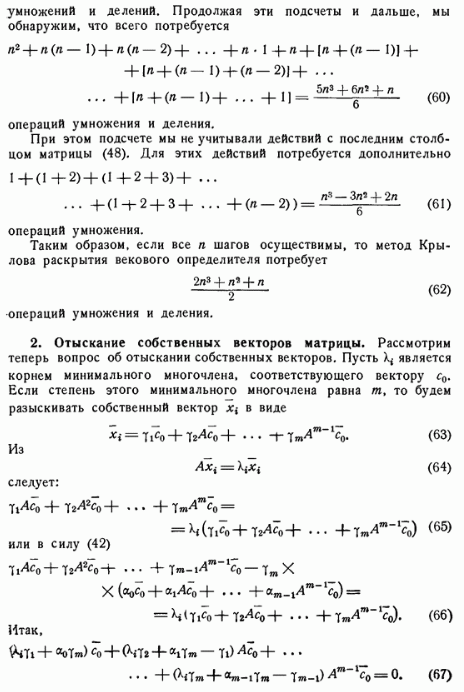
# Постановка задачи

Используя метод Крылова, найти собственные числа и собственные векторы матрицы. Собственные числа определить с четырьмя верными цифрами, а собственные векторы – с тремя десятичными знаками.

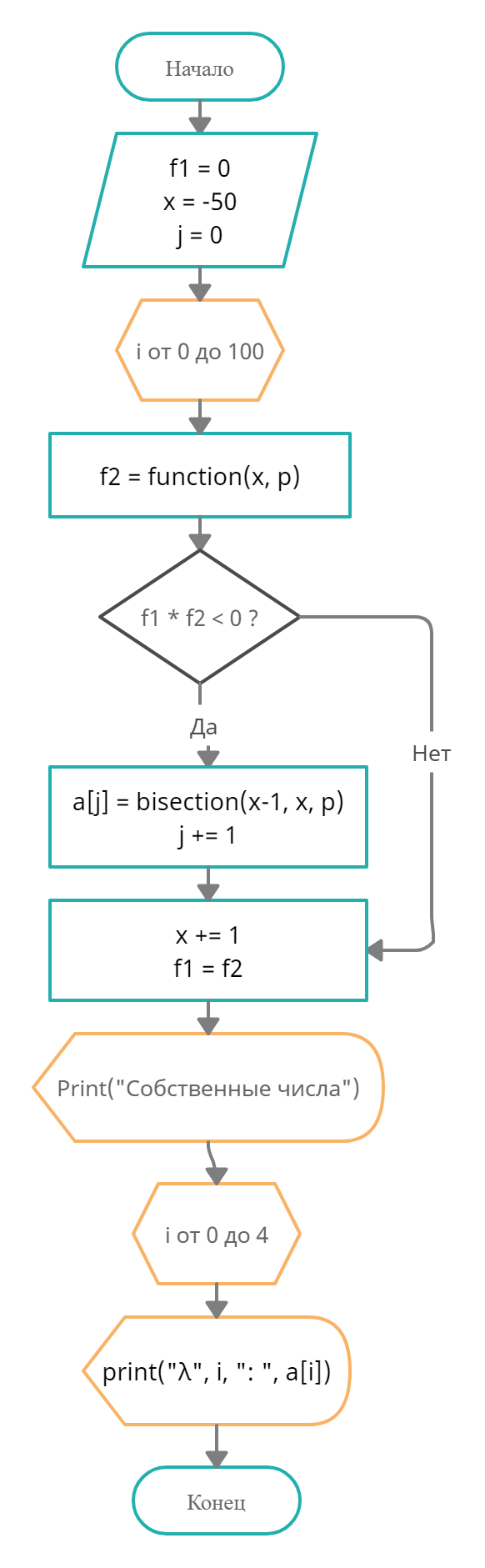


# Теоретические сведения

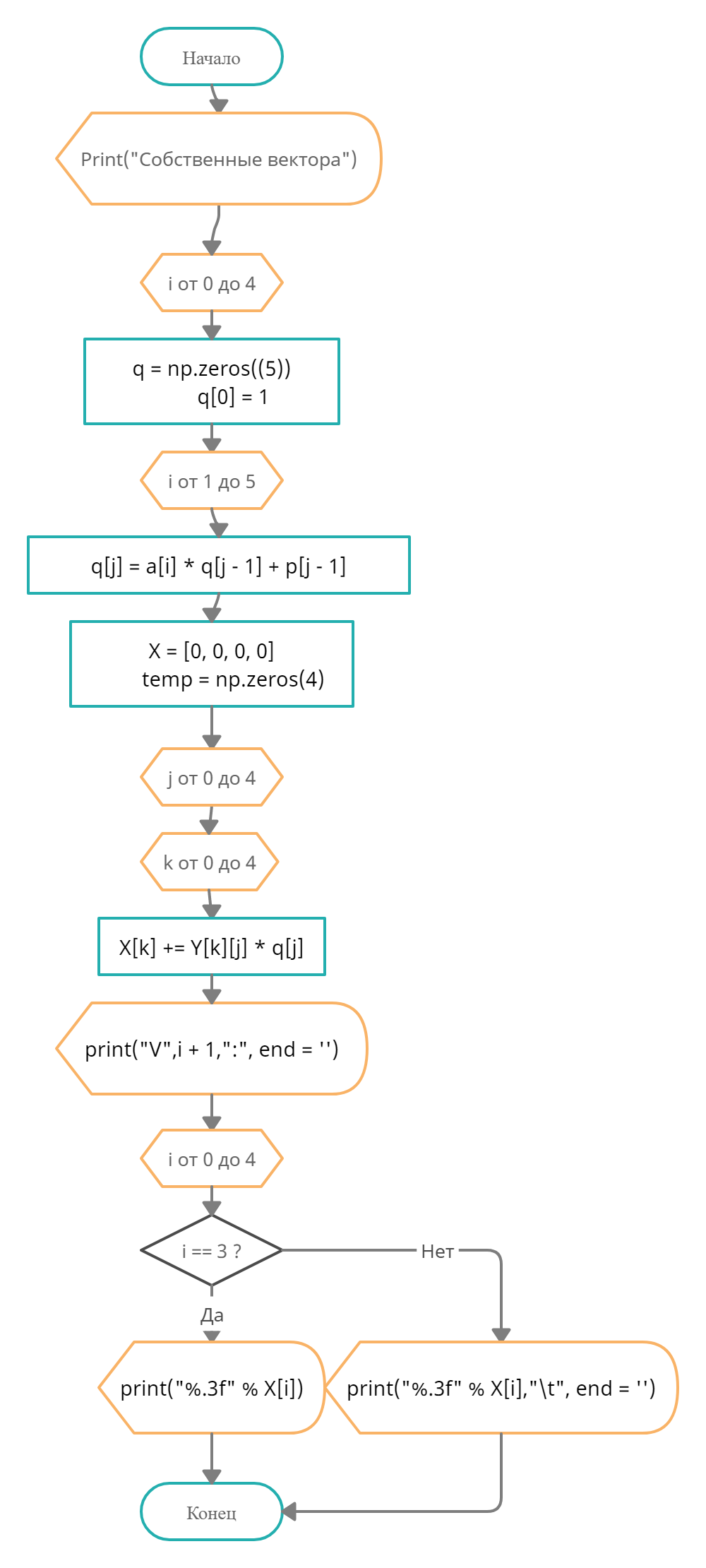




**Собственные числа матрицы**

****

**Собственные вектора**

****

**Расчетные данные**

Собственные числа

|  |  |
| --- | --- |
| λ1 | -2.5 |
| λ2 | 0.5 |
| λ3 | 1.875 |
| λ4 | 4.5 |

Собственные вектора

|  |  |
| --- | --- |
| V1 | (-9.660, 7.200, -23.935, 14.955) |
| V2 | (3.540, 1.800, -3.835, -4.095) |
| V3 | (-3.193, 5.341, 2.370, -0.795) |
| V4 | (35.140, 22.600, 8.965, 26.505) |

# Листинг разработанной программы

**Main.py**

from solutionMethods import \*

M = np.array([[2.5, 1, -0.5, 2],

            [1, 2, 1.2, 0.4],

            [-0.5, 1.2, -1, 1.5],

            [2, 0.4, 1.5, 1]])

Y = np.zeros((4,5)) # Считаем вектора Y

vectorComputation(M, Y)

p = np.zeros(4) # Находим корни системы линейных уравнений

Gauss(Y, p)

a = np.zeros(4) # Считаем собственные числа

calculationEigenvalues(p, a)

calculationEigenvectors(Y, p, a) # Считаем собственные вектора

**solutionMethods.py**

import numpy as np

def Gauss(matrix, p): # matrix - наша система уравнений, p - результат

   a = 0

   m = np.zeros((4,5))

   for i in range(4):

       for j in range(5):

           m[i][j] = matrix[i][j]

   a = m[0][0] # Делим первую строку на коэффициент a11

   for i in range(5):

       m[0][i] = m[0][i] / a

   a = m[1][0] # Вычитаем из второй строки первую, умноженную на коэффициент a21

   for i in range(5):

       m[1][i] = m[1][i] - a \* m[0][i]

   a = m[2][0] # Вычитаем из третьей строки первую, умноженную на коэффициент a31

   for i in range(5):

       m[2][i] = m[2][i] - a \* m[0][i]

   a = m[3][0] # Вычитаем из четвертой строки первую, умноженную на коэффициент a41

   for i in range(5):

       m[3][i] = m[3][i] - a \* m[0][i]

   a = m[1][1] # Делим вторую строку на коэффициент a22

   for i in range(1, 5):

       m[1][i] = m[1][i] / a

   a = m[2][1] # Вычитаем из третьей строки вторую, умноженную на коэффициент а32

   for i in range(1, 5):

       m[2][i] = m[2][i] - a \* m[1][i]

   a = m[3][1] # Вычитаем из четвертой строки вторую, умноженную на коэффициент а42

   for i in range(1, 5):

       m[3][i] = m[3][i] - a \* m[1][i]

   a = m[2][2] # Делим третью строку на коффициент а33

   for i in range(2, 5):

       m[2][i] = m[2][i] / a

   a = m[3][2] # Вычитаем из четвертой строки третью, умноженню на коэффициент a43

   for i in range(1, 5):

       m[3][i] = m[3][i] - a \* m[2][i]

   a = m[3][3] # Делим четвертую строку на коэффициент а44

   for i in range(2, 5):

       m[3][i] = m[3][i] / a

   p[3] = m[3][4]

   p[2] = m[2][4] - m[2][3] \* p[3]

   p[1] = m[1][4] - m[1][2] \* p[2] - m[1][3] \* p[3]

   p[0] = m[0][4] - m[0][1] \* p[1] - m[0][2] \* p[2] - m[0][3] \* p[3]

def function(x, p): # Считаем значение функции в заданной точке, где x - значение, p - коэффициенты уравнения

    f = x \* x \* x \* x + p[0] \* x \* x \* x + p[1] \* x \* x + p[2] \* x + p[3]

    return f

def bisection(a, b, p): # a, b - границы, p - коэффициенты уравнения, res - результат

    e = 0.001 # находим корень уравнения с заданной точностью

    while True:

        c = (a + b) / 2 # находим середину отрезка

        if (function(a,p) \* function(c, p) < 0): # определяем границы, где находится корень

            b = c

        else:

            a = c

        if(abs((function(c, p)) < e)):

            break

    return c

def multiplicationMatrix(a, b, Y): # Перемножаем матрицы, где a,b - матрицы, Y - результат

    for i in range(4):

        Y[i] = 0

        for j in range(4):

            Y[i] += a[i][j] \* b[j]

def vectorComputation(M, Y): # Считаем вектора Y, где M - заданная матрица, Y - результат

    Y0 = np.array([1, 0, 0, 0])

    Y1 = np.zeros(4)

    Y2 = np.zeros(4)

    Y3 = np.zeros(4)

    Y4 = np.zeros(4)

    multiplicationMatrix(M, Y0, Y1)

    multiplicationMatrix(M, Y1, Y2)

    multiplicationMatrix(M, Y2, Y3)

    multiplicationMatrix(M, Y3, Y4)

    for i in range(4): # Заносим вектора Y в общую матрицу

        Y[i][0] = Y3[i]

    for i in range(4):

        Y[i][1] = Y2[i]

    for i in range(4):

        Y[i][2] = Y1[i]

    for i in range(4):

        Y[i][3] = Y0[i]

    for i in range(4):

        Y[i][4] = -Y4[i]

def calculationEigenvalues(p, a): # Считаем собственные числа матрицы, где p - корни системы линейных уравнений, а - результат

    f1 = 0

    x = -50

    j = 0

    for i in range(100): # нахождим промежуток с корнями

        f2 = function(x, p)

        if (f1 \* f2 < 0): # Проверка на наличие корней

            a[j] = bisection(x - 1, x, p)

            j += 1

        x += 1

        f1 = f2

    print("Собственные числа:")

    for i in range(4):

        print("λ", i, ": ", a[i])

def calculationEigenvectors(Y, p, a): # считаем собственные вектора матрицы, где Y - вектора y, p - корни системы линейных уравнений,

    print("Собственные вектора:")  # а - собственные числа

    for i in range(4):

        q = np.zeros((5))

        q[0] = 1

        for j in range(1, 5): # Считаем вектор q по формуле q[j] = a(i) \* q(j - 1) + p(j - 1)

            q[j] = a[i] \* q[j - 1] + p[j - 1]

        X = [0, 0, 0, 0]

        temp = np.zeros(4)

        for j in range(4): # Считаем собственный вектор по формуле

            for k in range(4): # X = q(0) \* Y(3) + q(2) \* Y(1) + q(3) \* Y(0)

                X[k] += Y[k][j] \* q[j]

        print("V",i + 1,":", end = '')

        for i in range(4):

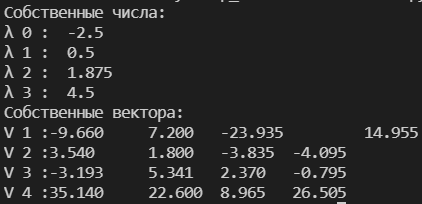
            if (i == 3):

                print("%.3f" % X[i])

            else:

                print("%.3f" % X[i],"\t", end = '')

# Результаты работы программы



# Вывод

В ходе данной работы были закреплены знания и умения по нахождение собственных чисел и собственных векторов методом Крылова.