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

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.

4. $$A = \begin{pmatrix} 2.5 & 1 & -0.5 & 2 \\ 1 & 2 & 1.2 & 0.4 \\ -0.5 & 1.2 & -1 & 1.5 \\ 2 & 0.4 & 1.5 & 1 \end{pmatrix}$$

Теоретические сведения

1. Отыскание собственных значений матрицы. Академик А. Н. Крылов одним из первых предложил довольно удобный метод раскрытия определителя (2) § 1.

Суть метода А. Н. Крылова состоит в преобразовании определителя $D(\lambda)$ к виду

$$D_1(\lambda) = \begin{vmatrix} b_{11}-\lambda & b_{12} & \dots & b_{1n} \\ b_{21}-\lambda^2 & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1}-\lambda^n & b_{n2} & \dots & b_{nn} \end{vmatrix}, \tag{1}$$

причем при некоторых условиях уравнения $D(\lambda)=0$ и $D_1(\lambda)=0$ имеют одни и те же корни. Раскрыть определитель $D_1(\lambda)$ значительно проще, чем $D(\lambda)$, так как λ содержится только в первом столбце. Как мы увидим позже, нам даже не придется вычислять миноры $D_1(\lambda)$.

Преобразование $D(\lambda)$ к виду (1) будем осуществлять следующим образом. Возьмем первое из уравнений (1) § 1:

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = \lambda x_1 \tag{2}$$

и умножим его на λ. Появившиеся в левой части выражения λx_i заменим на равные им в силу системы (1) § 1 выражения:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \qquad (i=1,\ 2,\ \dots,\ n). \tag{3}$$

Получим новое уравнение:

$$b_{21}x_1 + b_{22}x_2 + \dots + b_{2n}x_n = \lambda^2 x_1, \tag{4}$$

где

$$b_{2i} = \sum_{k=1}^{n} a_{1k}a_{ki} \qquad (i=1,\ 2,\ \dots,\ n). \tag{5}$$

С уравнением (4) поступаем так же, как и с уравнением (2). При этом получим новое уравнение:

$$b_{31}x_1 + b_{32}x_2 + \dots + b_{3n}x_n = \lambda^3 x_1, \tag{6}$$

где

$$b_{3i} = \sum_{k=1}^{n} b_{2k}a_{ki} \qquad (i=1,\ 2,\ \dots,\ n). \tag{7}$$

Этот процесс продолжаем до тех пор, пока не придем к уравнению

$$b_{n1}x_1 + b_{n2}x_2 + \dots + b_{nn}x_n = \lambda^n x_1. \tag{8}$$

Для единообразия обозначений условимся считать $b_{1i} = a_{1i}$.

умножений и делений. Продолжая эти подсчеты и дальше, мы обнаружим, что всего потребуется

$$n^2+n(n-1)+n(n-2)+\ldots+n\cdot 1+n+[n+(n-1)]+$$
$$+[n+(n-1)+(n-2)]+\ldots$$
$$\ldots+[n+(n-1)+\ldots+1]=\frac{5n^3+6n^2+n}{6} \qquad (60)$$

операций умножения и деления.

При этом подсчете мы не учитывали действий с последним столбцом матрицы (48). Для этих действий потребуется дополнительно

$$1+(1+2)+(1+2+3)+\ldots$$
$$\ldots+(1+2+3+\ldots+(n-2))=\frac{n^3-3n^2+2n}{6} \qquad (61)$$

операций умножения.

Таким образом, если все n шагов осуществимы, то метод Крылова раскрытия векового определителя потребует

$$\frac{2n^3+n^2+n}{2} \qquad (62)$$

операций умножения и деления.

2. Отыскание собственных векторов матрицы. Рассмотрим теперь вопрос об отыскании собственных векторов. Пусть λ_i является корнем минимального многочлена, соответствующего вектору c_0. Если степень этого минимального многочлена равна m, то будем разыскивать собственный вектор $\bar{x}_i$ в виде

$$\bar{x}_i=\gamma_1\bar{c}_0+\gamma_2A\bar{c}_0+\ldots+\gamma_mA^{m-1}\bar{c}_0. \qquad (63)$$

Из

$$A\bar{x}_i=\lambda_i\bar{x}_i \qquad (64)$$

следует:

$$\gamma_1A\bar{c}_0+\gamma_2A^2\bar{c}_0+\ldots+\gamma_mA^m\bar{c}_0=$$
$$=\lambda_i(\gamma_1\bar{c}_0+\gamma_2A\bar{c}_0+\ldots+\gamma_mA^{m-1}\bar{c}_0) \qquad (65)$$

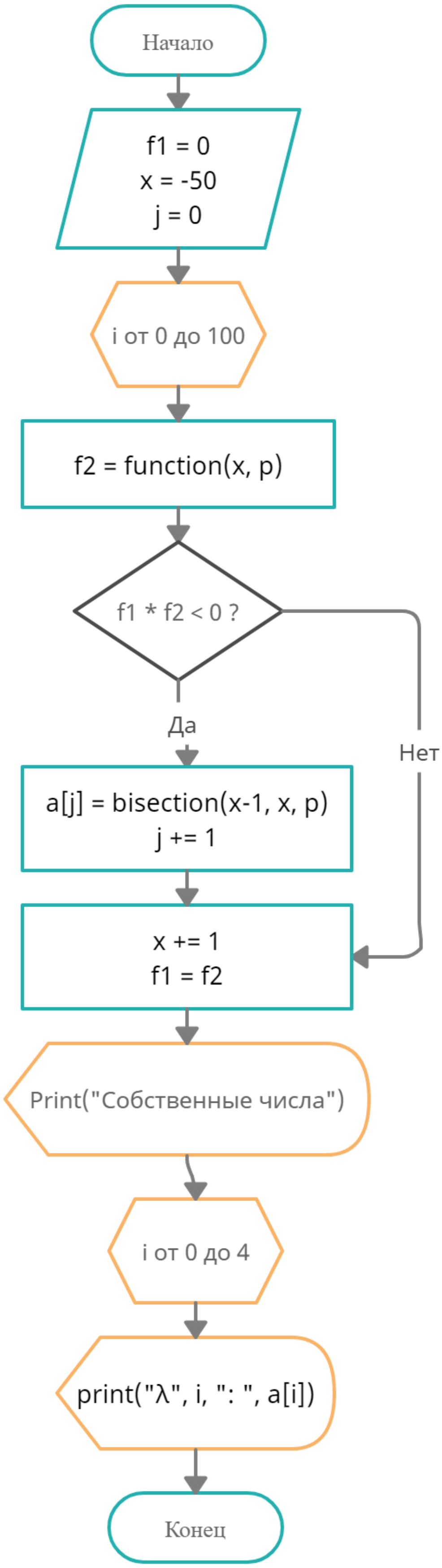
или в силу (42)

$$\gamma_1A\bar{c}_0+\gamma_2A^2\bar{c}_0+\ldots+\gamma_{m-1}A^{m-1}\bar{c}_0-\gamma_m\times$$
$$\times(\alpha_0\bar{c}_0+\alpha_1A\bar{c}_0+\ldots+\alpha_{m-1}A^{m-1}\bar{c}_0)=$$
$$=\lambda_i(\gamma_1\bar{c}_0+\gamma_2A\bar{c}_0+\ldots+\gamma_mA^{m-1}\bar{c}_0). \qquad (66)$$

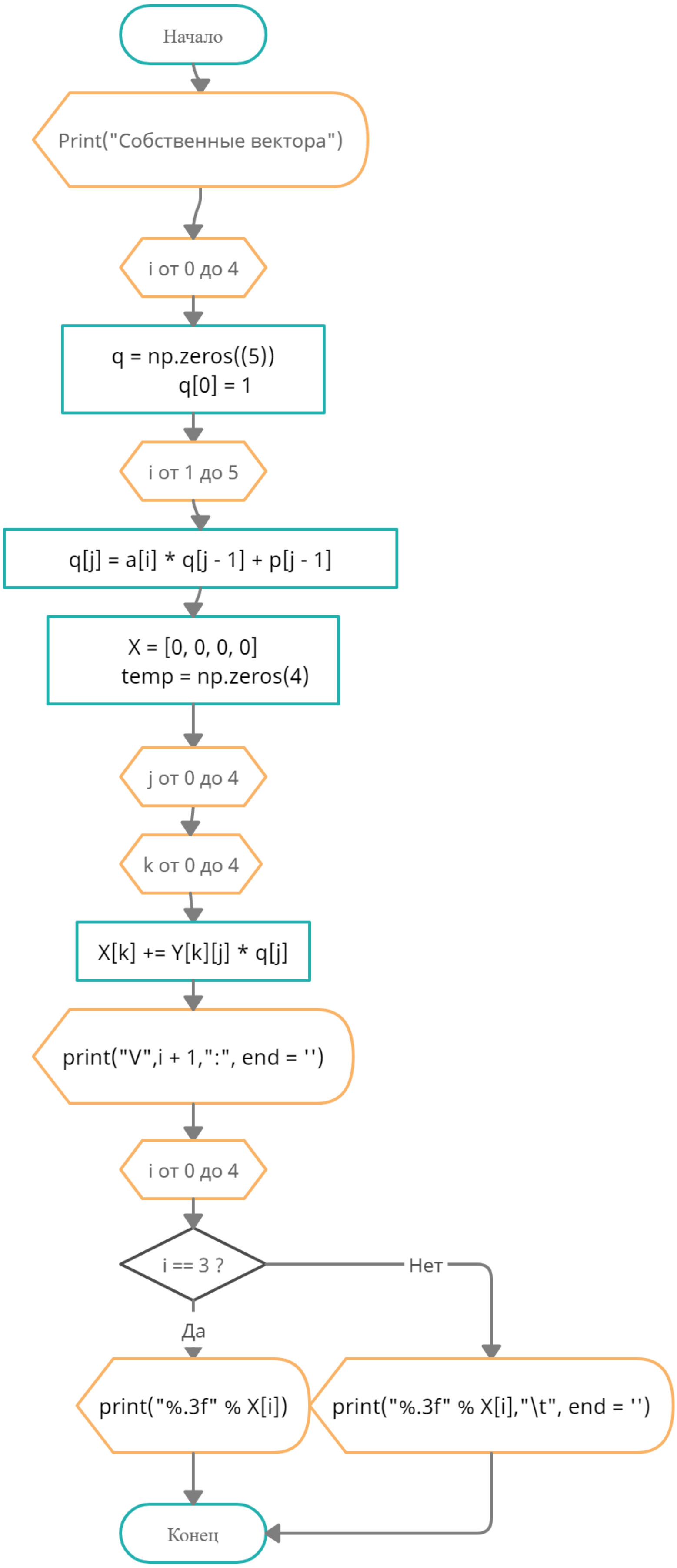
Итак,

$$(\lambda_i\gamma_1+\alpha_0\gamma_m)\bar{c}_0+(\lambda_i\gamma_2+\alpha_1\gamma_m-\gamma_1)A\bar{c}_0+\ldots$$
$$\ldots+(\lambda_i\gamma_m+\alpha_{m-1}\gamma_m-\gamma_{m-1})A^{m-1}\bar{c}_0=0. \qquad (67)$$

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 a
41
   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 a32
    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 a
42
    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 a33
    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 a4
3
    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 a44
    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 - резул
ьтат
    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 находи
тся 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 - р
езультат
    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 - ко
рни системы линейных уравнений, a - 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:")  # a - 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

```
Собственные числа:
λ 0 :  -2.5
λ 1 :  0.5
λ 2 :  1.875
λ 3 :  4.5
Собственные вектора:
V 1 :-9.660     7.200   -23.935         14.955
V 2 :3.540      1.800   -3.835  -4.095
V 3 :-3.193     5.341   2.370   -0.795
V 4 :35.140     22.600  8.965   26.505
```

Вывод

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