БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №3

Итерационный степенной метод

**Преподаватель:** Горбачева Юлия Николаевна

**Студент:**  Жиркевич Александр

2 курс 10 группа

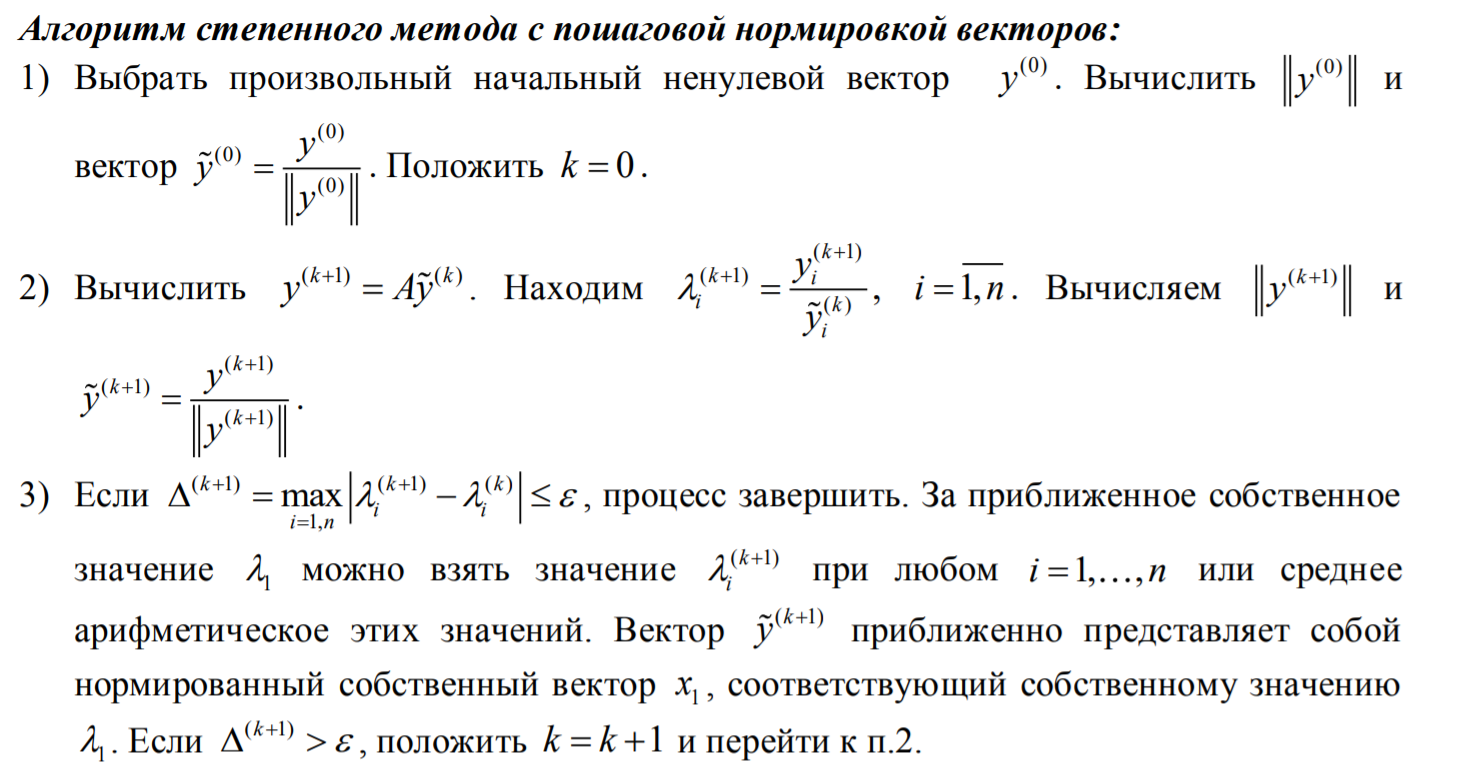
**Минск, 2020**

**ПОСТАНОВКА ЗАДАЧИ**

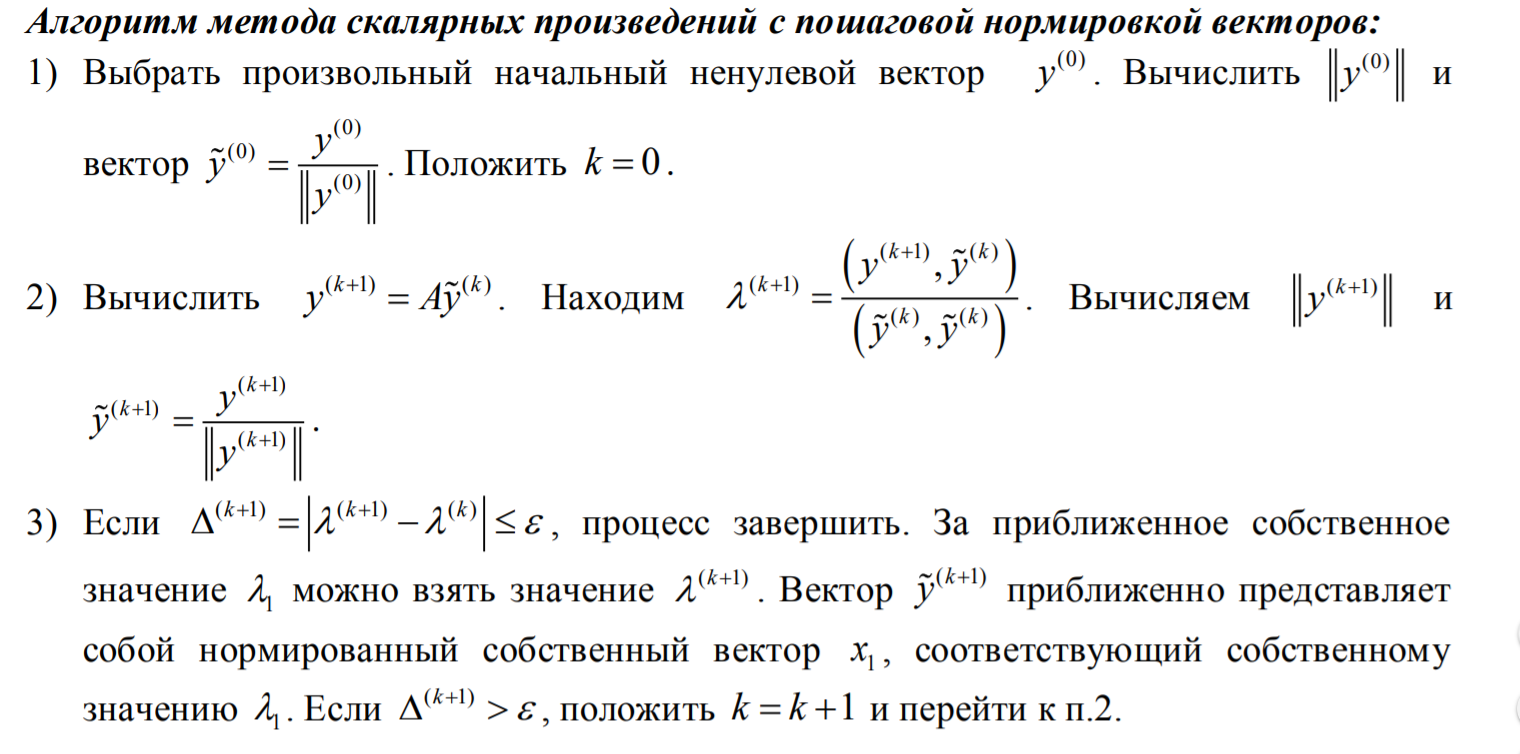
Итерационным степенным методом и методом скалярных произведений с точностью ε = 10-6 найти наибольшее по модулю собственное значение матрицы A и соответствующий собственный вектор. Вычислительный процесс проводить с нормировкой векторов итерационной последовательности.

**КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

**Итерационный степенной метод** предназначен для нахождения одного или нескольких собственных значений и соответствующих собственных векторов. Пусть A – вещественная матрица порядка n. Мы рассмотрим степенной метод для случая диагонализируемых матриц (матриц простой структуры). Матрица заведомо диагонализируема в двух важных частных случаях: если она симметри чная или если ее собственные значения различны. Диагонализируемая матрица имеет ровно n линейно независимых собственных векторов.



Если матрица A симметричная (A = AT), то для нахождения λ1 можно применять **метод скалярных произведений**.



**ЛИСТИНГ ПРОГРАММЫ (Python)**

class Eigen(ABC):

    def \_\_init\_\_(self, var:int) -> None:

   B =   [[1.342, 0.432, -0.599, 0.202, 0.603, -0.202],

                    [0.432, 1.342, 0.256, -0.599, 0.204, 0.304],

                    [-0.599, 0.256, 1.342, 0.532, 0.101, 0.506],

                    [0.202, -0.599, 0.532, 1.342, 0.106, -0.311],

                    [0.603, 0.204, 0.101, 0.106, 1.342, 0.102],

                    [-0.202, 0.304, 0.506, -0.311, 0.102, 1.342]]

        C =   [[0.05, 0, 0, 0, 0, 0],

                    [0, 0.03, 0, 0, 0, 0],

                    [0, 0, 0.02, 0, 0, 0],

                    [0, 0, 0, 0.04, 0, 0],

                    [0, 0, 0, 0, 0.06, 0],

                    [0, 0, 0, 0, 0, 0.07]]

        self.\_A = Matrix.sum(B, Matrix.mulFloat(C, var))

    def getMatrix(self) -> List:

        return [list(line) for line in self.\_A]

    @abstractmethod

    def find(self, eps:float) -> (float, List, int, List):

        pass

#Итерационно-степенной метод

class ItPowMethod(Eigen):

    def find(self,eps:float) -> (float, List, int, List):

        delta = float("inf")

        startVector = [1,1,1,1,1,1]

        y0 = Vector.normalize(startVector)

        Lambda0 = [random.randrange(0,10)/10 for \_ in self.\_A]

        iterations = 0

        while (delta > eps):

            y = Matrix.mulVector(self.\_A,y0)

            Lambda = [el1/el2 for el1,el2 in zip(y,y0)]

            delta = abs(max(Vector.sub(Lambda0,Lambda)))

            y0 = Vector.normalize(y)

            Lambda0 = Lambda

            iterations+=1

        return sum(Lambda)/len(Lambda), y0, iterations, startVector

#Метод скалярных произведений

class ScalMulMethod(Eigen):

    def find(self,eps:float) -> (float, List, int, List):

        delta = float("inf")

        startVector = [1,1,1,1,1,1]

        y0 = Vector.normalize(startVector)

        Lambda0 = random.randrange(1,1000)/1000

        iterations = 0

        while (delta > eps):

            y = Matrix.mulVector(self.\_A,y0)

            Lambda = (sum([el1\*el2 for el1,el2 in zip(y,y0)]) /

                sum([el1\*el2 for el1,el2 in zip(y0,y0)]))

            delta = abs(Lambda - Lambda0)

            y0 = Vector.normalize(y)

            Lambda0 = Lambda

            iterations+=1

        return Lambda, y0, iterations, startVector

if \_\_name\_\_ == "\_\_main\_\_":

    M = ItPowMethod(7)

    first = M.find(1e-6)

    print("\*\*\* Итерационно-степенной метод \*\*\*")

    print(f"Начальный вектор: {first[3]}")

    print(f"Максимальное по модулю собственное значение: {first[0]}")

    print(f"Собственный вектор: {first[1]}")

    r = Vector.sub(Matrix.mulVector(M.getMatrix(),first[1]),

     Vector.mulFloat(first[1], first[0]))

    print(f"Вектор r: {r}")

    print(f"Количество итераций: {first[2]}")

    print()

    M = ScalMulMethod(7)

    second = M.find(1e-6)

    print("\*\*\* Метод скалярных произведений \*\*\*")

    print(f"Начальный вектор: {second[3]}")

    print(f"Максимальное по модулю собственное значение: {second[0]}")

    print(f"Собственный вектор: {second[1]}")

    r = Vector.sub(Matrix.mulVector(M.getMatrix(),second[1]),

     Vector.mulFloat(second[1], second[0]))

    print(f"Вектор r: {r}")

    print(f"Количество итераций: {second[2]}")

**РЕЗУЛЬТАТЫ**

\*\*\* Итерационно-степенной метод \*\*\*

Начальный вектор: [1, 1, 1, 1, 1, 1]

Максимальное по модулю собственное значение: 2.6393439045812004

Собственный вектор: [1.0, 0.3639100345621695, -0.6081115679959351, -0.1592853383200852, 0.641760144318817, -0.35185616246514]

Вектор r: [4.733080769980802e-06, -5.205598701096825e-06, -6.058244118811729e-06, 4.344976968995162e-06, 4.534284541168887e-07, -9.599905414492405e-06]

Количество итераций: 248

\*\*\* Метод скалярных произведений \*\*\*

Начальный вектор: [1, 1, 1, 1, 1, 1]

Максимальное по модулю собственное значение: 2.6393369419687116

Собственный вектор: [1.0, 0.3719171384914634, -0.6044362559476335, -0.16517841517177964, 0.644746707215417, -0.3426857568991469]

Вектор r: [2.732665205806839e-05, -0.0008411966931555215, -0.0004072969063320109, 0.0006220710100379168, -0.00029992926314448276, -0.0009844131225112962]

Количество итераций: 130

**ВЫВОД**

В ходе данной лабораторной работы была написана программа, находящая максимальное по модулю собственное значение матрицы и соответствующий ему собственный вектор. Для выполнения этой задачи были реализованы алгоритмы итерационно-степенного метода и метода скалярных произведений. Метод скалярных произведений сходится примерно в два раза быстрее.