**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра технической кибернетики

Дисциплина: Теория матриц

Практическая работа № 1

Тема: «Вычисление ранга матрицы методом Гаусса»

Выполнил:

Студент группы МТК-233

Орлов-Куреши М. Н.

Проверил:

Кириков И. Б.

Белгород 2021

**Цель работы:** изучить метод Гаусса для нахождения ранга матрицы. Реализовать метод Гаусса на языке программирования Python.

**Метод Гаусса**

Нахождение ранга методом Гаусса или методом элементарных преобразований сводится к преобразованию матрицы к ступенчатому виду. Ранг ступенчатой матрицы равен количеству ненулевых строк в данной матрице.

Пример:

Приведем матрицу А к ступенчатому виду:

Количество ненулевых строк равно 2, следовательно Rank(A) = 2

**Реализация алгоритма на языке программирования Python**

# Метод Гаусса

class GaussMethodMixin:

    def to\_fractions(self) -> None:

        for i in range(self.row):

            for j in range(self.column):

                self.\_elements[i][j] = Fraction(self.\_elements[i][j])

    def \_set\_max\_el\_in\_row(self, i:int, count\_swap:int) -> int:

        max\_el = abs(self.\_elements[i][i])

        max\_row = i

        for j in range(i+1, self.row):

            if max\_el < abs(self.\_elements[j][i]):

                max\_el = abs(self.\_elements[j][i])

                max\_row = j

        if i != max\_row:

            self.swap\_rows(i, max\_row)

            count\_swap += 1

        return count\_swap

    def \_set\_max\_el\_in\_column(self, i:int, count\_swap:int) -> int:

        max\_el = abs(self.\_elements[i][i])

        max\_column = i

        for j in range(i+1, self.column):

            if max\_el < abs(self.\_elements[i][j]):

                max\_el = abs(self.\_elements[i][j])

                max\_column = j

        if i != max\_column:

            self.swap\_columns(i, max\_column)

            count\_swap += 1

        return count\_swap

    def get\_triangle(self):

        matr = self.copy()

        matr.to\_fractions()

        count\_swap = 0

        for i in range(matr.row):

if matr.\_elements[i][i] == 0:

            count\_swap += matr.\_set\_max\_el\_in\_row(i, count\_swap)

            if matr.\_elements[i][i] == 0:

                count\_swap += matr.\_set\_max\_el\_in\_column(i, count\_swap)

            if matr.\_elements[i][i] == 0:

                return matr, None

            for j in range(i+1, matr.row):

                c = -(matr.\_elements[j][i] / matr.\_elements[i][i])

                for k in range(i, matr.column):

                    matr.\_elements[j][k] += c \* matr.\_elements[i][k]

        return matr, count\_swap

class Matrix(GaussMethodMixin, ElementaryЕransformationsMixin):

    def \_\_init\_\_(self, row:int, column:int, elements:list|None = None) -> None:

        self.\_row = row

        self.\_column = column

        self.\_elements = []

        if elements:

            index = 0

            for i in range(row):

                self.\_elements.append([])

                for \_ in range(column):

                    self.\_elements[i].append(elements[index])

                    index += 1

    @property

    def row(matrix) -> int:

        return matrix.\_row

    @property

    def column(matrix) -> int:

        return matrix.\_column

    def \_is\_zeros(self, row:list) -> bool:

        for el in row:

            if el != 0:

                return False

        return True

    @benchmark

    def rank(self) -> int:

        triangle\_matr, \_ = self.get\_triangle()

        rank = 0

        for i in range(self.row):

            if triangle\_matr.\_elements[i][i] != 0 or not triangle\_matr.\_is\_zeros(triangle\_matr.\_elements[i]):

                rank += 1

        return rank

    def copy(self):

        matr = Matrix(self.row, self.column,)

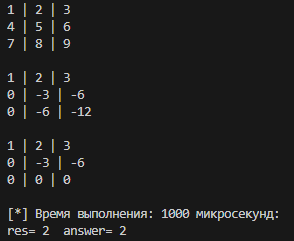
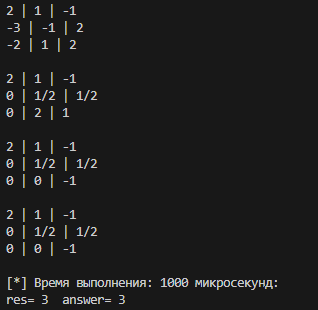
        matr.\_elements = [row.copy() for row in self.\_elements]

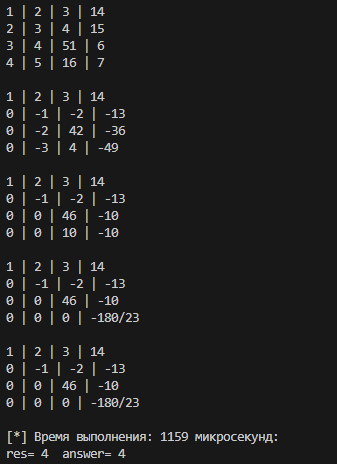
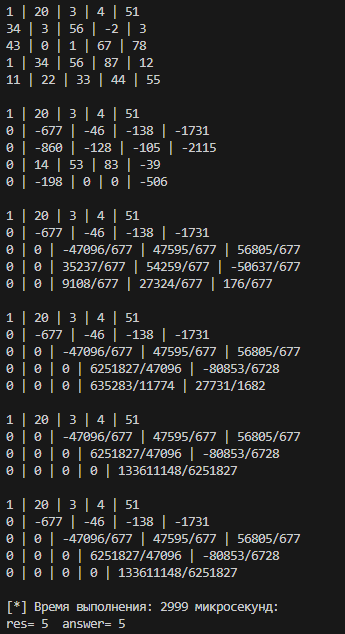
        return matr

    def \_\_str\_\_(self) -> str:

        return '\n'.join(' | '.join(map(str, row)) for row in self.\_elements)

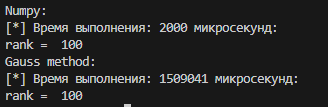
**Скриншоты работы программы**

** **

** **

**Сравнение с существующей библиотекой**

Для сравнения использовалась библиотека numpy. Нахождение ранга в данной библиотеки происходит при помощи сингулярного разложения матрицы (SVD, Singular Value Decomposition).



В качестве тестовой матрицы была сгенерирована матрица порядка 100х100. Numpy нашел решение очень быстро, поскольку написан на Си и Фортран. В тоже время, у данной реализации метода Гаусса нахождение ранга заняло полторы секунды, что очень медленно по сравнению с готовой библиотекой.

**Вывод:** в ходе работы был изучен и реализован метод Гаусса для нахождения ранга матрицы. Алгоритм был протестирован на матрицах ранга: 2, 3, 4 и 5. Также, алгоритм был протестирован в сравнении с готовой библиотекой numpy.

**Ссылки на использованные источники**

1. Ранг матрицы: определение, методы нахождения, примеры, решения. [Электронный ресурс]: URL: http://www.cleverstudents.ru/matrix/rank.html (дата обращения: 09.09.2023)
2. NumPy documentation [Электронный ресурс]: URL: https://numpy.org/doc/stable/ (дата обращения: 09.09.2023)