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:
Кариков Е. Б.

Белгород 2023

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:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Приведем матрицу A к ступенчатому виду:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 7 & 8 & 9 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{pmatrix}$$

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

Реализация алгоритма на языке программирования Python

```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, ElementaryEransformationsMixin):
    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

```
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9

1 | 2 | 3
0 | -3 | -6
0 | -6 | -12

1 | 2 | 3
0 | -3 | -6
0 | 0 | 0

[*] Время выполнения: 1000 микросекунд:
res= 2  answer= 2
```

```
2 | 1 | -1
-3 | -1 | 2
-2 | 1 | 2

2 | 1 | -1
0 | 1/2 | 1/2
0 | 2 | 1

2 | 1 | -1
0 | 1/2 | 1/2
0 | 0 | -1

2 | 1 | -1
0 | 1/2 | 1/2
0 | 0 | -1

[*] Время выполнения: 1000 микросекунд:
res= 3  answer= 3
```

```
1 | 2 | 3 | 14
2 | 3 | 4 | 15
3 | 4 | 51 | 6
4 | 5 | 16 | 7

1 | 2 | 3 | 14
0 | -1 | -2 | -13
0 | -2 | 42 | -36
0 | -3 | 4 | -49

1 | 2 | 3 | 14
0 | -1 | -2 | -13
0 | 0 | 46 | -10
0 | 0 | 10 | -10

1 | 2 | 3 | 14
0 | -1 | -2 | -13
0 | 0 | 46 | -10
0 | 0 | 0 | -180/23

1 | 2 | 3 | 14
0 | -1 | -2 | -13
0 | 0 | 46 | -10
0 | 0 | 0 | -180/23

[*] Время выполнения: 1159 микросекунд:
res= 4  answer= 4
```

```
1 | 20 | 3 | 4 | 51
34 | 3 | 56 | -2 | 3
43 | 0 | 1 | 67 | 78
1 | 34 | 56 | 87 | 12
11 | 22 | 33 | 44 | 55

1 | 20 | 3 | 4 | 51
0 | -677 | -46 | -138 | -1731
0 | -860 | -128 | -105 | -2115
0 | 14 | 53 | 83 | -39
0 | -198 | 0 | 0 | -506

1 | 20 | 3 | 4 | 51
0 | -677 | -46 | -138 | -1731
0 | 0 | -47096/677 | 47595/677 | 56805/677
0 | 0 | 35237/677 | 54259/677 | -50637/677
0 | 0 | 9108/677 | 27324/677 | 176/677

1 | 20 | 3 | 4 | 51
0 | -677 | -46 | -138 | -1731
0 | 0 | -47096/677 | 47595/677 | 56805/677
0 | 0 | 0 | 6251827/47096 | -80853/6728
0 | 0 | 0 | 635283/11774 | 27731/1682

1 | 20 | 3 | 4 | 51
0 | 0 | -47096/677 | 47595/677 | 56805/677
0 | 0 | 0 | 6251827/47096 | -80853/6728
0 | 0 | 0 | 0 | 133611148/6251827

1 | 20 | 3 | 4 | 51
0 | -677 | -46 | -138 | -1731
0 | 0 | -47096/677 | 47595/677 | 56805/677
0 | 0 | 0 | 6251827/47096 | -80853/6728
0 | 0 | 0 | 0 | 133611148/6251827

[*] Время выполнения: 2999 микросекунд:
res= 5  answer= 5
```

Сравнение с существующей библиотекой

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

```
Numpy:
[*] Время выполнения: 2000 микросекунд:
rank =  100
Gauss method:
[*] Время выполнения: 1509041 микросекунд:
rank =  100
```

В качестве тестовой матрицы была сгенерирована матрица порядка 100x100. 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)