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 № 2

Тема: «Метод Гревилля последовательного нахождения псевдообратной матрицы»

Выполнил:

Студент группы МТК-233

Орлов-Куреши М. Н.

Проверил:

Кариков Е. Б.

Белгород 2023

Цель работы: изучить метод Гревилля для последовательного нахождения псевдообратной матрицы. Реализовать метод Гревилля на языке программирования Python.

Метод Гревилля последовательного нахождения псевдообратнойматрицы

Пусть a_k – k-й столбец в $m \times n$ – матрице A, $A_k = (a_1, ..., a_k)$ – матрица, образованная первыми k столбцами матрицы A, b_k – последняя строка в матрице A_k^+, $k = 1, \ldots, n$, $A_1 = a_1$, $A_n = A$.

Тогда

$$A_1^+ = a_1^+ = \frac{a_1^*}{a_1^* a_1},$$

и для $k > 1$ имеют место рекуррентные формулы

$$A_k^+ = \begin{pmatrix} B_k \\ b_k \end{pmatrix}, \; B_k = A_{k-1}^+ - d_k b_k, \; d_k = A_{k-1}^+ a_k.$$

При этом, если $c_k = a_k - A_{k-1} d_k \neq 0$, то

$$b_k = c_k^+ = (a_k - A_{k-1} d_k)^+;$$

если же $c_k = 0$, т.е. $a_k = A_{k-1} d_k$, то

$$b_k = (1 + d_k^* d_k)^{-1} d_k^* A_{k-1}^+.$$

Этот метод не требует вычисления детерминантов и может быть использован для вычисления обратной матрицы. [1]

Реализация алгоритма на языке программирования Python

```
class GrevilleMethod:
    def is_zeros_matr(self):
        for i in range(self.row):
            for j in range(self.column):
                if self._elements[i][j] != 0:
                    return False
        return True

    def get_pseudoinverse_matrix(self):
        list_A = []
        k = 0
        while k < self.column:
            a_k = self.get_column(k, 1)
            k += 1
            if k == 1:
                a_kT = a_k.T
                list_A.append(((a_kT * a_k) ** -1) * a_kT)
            else:
                d_k = list_A[-1] * a_k
```

```
            c_k = a_k - (self.get_column(range(k-1), k-1) * d_k)
            if c_k.is_zeros_matr():
                temp_d = d_k.T * d_k
                temp_d.sum_num(1)
                b_k = temp_d ** -1 * d_k.T * list_A[-1]
            else:
                b_k = (c_k.T * c_k) ** -1 * c_k.T
            B = list_A[-1] - d_k * b_k
            B.add_row(b_k)
            list_A.append(B)
    return list_A[-1]
```

Скриншоты работы программы

```
1 | -1 | 0
-1 | 2 | 1
2 | -3 | -1
0 | 1 | 1

[*] Время выполнения: 0 микросекунд:
0.333 | 0.111 | 0.222 | 0.444
0.0 | 0.111 | -0.111 | 0.111
0.333 | 0.222 | 0.111 | 0.556

Numpy:

[[ 3.33333333e-01  1.11111111e-01  2.22222222e-01  4.44444444e-01]
 [-1.96332622e-16  1.11111111e-01 -1.11111111e-01  1.11111111e-01]
 [ 3.33333333e-01  2.22222222e-01  1.11111111e-01  5.55555556e-01]]
[*] Время выполнения: 1001 микросекунд:
```

```
1 | -1 | 2 | 1
-1 | 2 | -3 | 1
0 | 1 | -1 | 1

[*] Время выполнения: 1000 микросекунд:
-1.0 | -1.667 | 2.667
-1.0 | -1.333 | 2.333
0.0 | -0.333 | 0.333
1.0 | 1.0 | -1.0

Numpy:

[[-1.00000000e+00 -1.66666667e+00  2.66666667e+00]
 [-1.00000000e+00 -1.33333333e+00  2.33333333e+00]
 [-6.15297315e-16 -3.33333333e-01  3.33333333e-01]
 [ 1.00000000e+00  1.00000000e+00 -1.00000000e+00]]
[*] Время выполнения: 1001 микросекунд:
```

```
1 | -1 | 0
-1 | 2 | 1

[*] Время выполнения: 1001 микросекунд:
1.0 | 0.333
0.0 | 0.333
1.0 | 0.667

Numpy:

[[1.00000000e+00 3.33333333e-01]
 [1.10343860e-16 3.33333333e-01]
 [1.00000000e+00 6.66666667e-01]]
[*] Время выполнения: 11582 микросекунд:
```

Решение системы уравнений

```
AX = B
A:
1 | -1 | 1
-1 | 2 | 1
2 | -3 | -1
B:
3
6
0
X=
[*] Время выполнения: 1000 микросекунд:
21.0
15.0
-3.0
```

Вывод: в ходе работы был изучен и реализован метод Гревилля для нахождения псевдообратной матрицы.

Список литературы

1. Юдин Д.А. Прикладные аспекты теории матриц: учебное пособие / Д.А. Юдин. - Белгород: Изд-во БГТУ, 2016. – С. 20-24.

2. Переопределённая система [Электронный ресурс] // URL: https://ru.wikipedia.org/wikiПереопределённая_система (дата обращения: 05.11.2020).

3. Функция поиска наименьших квадратов для линейного матричного уравнения. [Электронный ресурс] // URL: https://pyprog.pro/linear_algebra_functions/linalg_lstsq.html (дата обращения: 05.10.2020).