Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы оптимизации и управления»

ОТЧЁТ
к лабораторной работе
на тему:
«ОБРАЩЕНИЕ МАТРИЦЫ С ИЗМЕНЕННЫМ СТОЛБЦОМ»

Выполнил студент группы 053505
Слуцкий Никита Сергеевич

Проверил ассистент
каф.информатики
Туровец Николай Олегович

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Как известно, алгоритм обращения квадратной матрицы размерности $n \times n$ имеет время выполнения $\Theta(n\ 3\ )$. Однако, обладая некоторыми дополнительными данными о матрице или используя особенности её структуры (если таковые имеются), для решения задачи её обращения можно построить алгоритм с меньшей временной сложностью. В рамках данной работы рассматривается и разрабатывается алгоритм обращения квадратной матрицы $\overline{A}$ размерности $n \times n$, использующий дополнительные данные: – матрицу A, отличающуюся от матрицы $\overline{A}$единственным столбцом с номером i; – матрицу A^{-1}, обратную A. Как будет показано, используя эти данные, можно снизить временную сложность поиска обратной матрицы до $\Theta(n^2\ )$.

1 ВЫПОЛНЕНИЕ РАБОТЫ

Используя данные, предоставленные в условии лабораторной работы, составим алгоритм нахождения матрицы $\bar{A}^{-1}$.

1 Найти вектор $l = A^{-1}x$. Если $l_i = 0$, матрица $\bar{A}$необратима, алгоритм завершает работу. Иначе матрица $\bar{A}$ обратима.

2 Создать вектор $\tilde{l}$, который получается из вектора l путём замены в нём i-го элемента на -1.

3 Найти $\hat{l} = (-1/l_i)\,\tilde{l}$.

4 Сформировать матрицу Q, путём замены в единичной матрице E i-го столбца столбцом $\hat{l}$.

5 Найти $\bar{A}^{-1} = QA^{-1}$. Временная сложность представленного алгоритма равна $\Theta(n^2)$. Действительно, шаги $1-4$ могут быть выполнены за $\Theta(n^2)$ операций. Умножение матриц, однако, требует $\omega(n^2)$ операций, однако матрица Q разрежена: в каждой её строке не более двух ненулевых элементов, номера которых известны, поэтому умножение можно выполнить за $\Theta(n^2)$ операций.

2 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

Для заданного в условии лабораторной работы примера программный продукт после отработки выдаёт корректный ответ.

```
Reverse matrix is:
matrix([[ 1.,  1., -1.],
        [ 0.,  1.,  0.],
        [ 0.,  0.,  1.]])

Process finished with exit code 0
```

Рисунок 1. Результат вывода программного продукта

При $l_i = 0$ реализованная функция возвращает результат None, который значит, что обратная матрица не существует.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы был изучен альтернативный способ нахождения обратной матрицы для матрицы с изменённой колонкой. Конечно же, всегда можно найти обратную матрицу по определению, высчитывая алгебраические дополнения или, например, методом Гаусса, однако изученный алгоритм тем и интересен, что позволяет, основываясь на факте, что для исходной матрицы обратная уже известна, найти якобы более быстрым способом обратную матрицы для чуть-чуть преобразованной исходной.

Цели лабораторной работы можно считать достигнутыми. Работа выполнена.

ПРИЛОЖЕНИЕ А

Листинг кода

```
def read_square_matrix_from_file(file_name: str) -> (int, np.matrix):
    file: typing.IO = open(file_name, 'r')
    lines: list[str | bytes] = file.readlines()
    file.close()

    raw_matrix: list[list[int]] = list()
    matrix_size: int = int(lines[0])

    for rows_counter in range(1, matrix_size + 1):
        raw_matrix.append([int(x) for x in lines[rows_counter].split(' ')])

    return matrix_size, np.matrix(raw_matrix)


def read_column_from_file(file_name: str) -> (int, np.array):
    file: typing.IO = open(file_name, 'r')
    lines: list[str | bytes] = file.readlines()
    file.close()

    return int(lines[0]), np.array([int(item) for item in lines[1].split(' ')])

def get_inverse_for_matrix_with_modified_column(matrix_file_name: str, column_file_name: str) ->
np.matrix | None:
    matrix_dimension, matrix = read_square_matrix_from_file(matrix_file_name)

    print('Source matrix:')
    pprint.pprint(matrix)

    print('\nReverse matrix:')
    reversal_matrix: np.matrix = np.linalg.inv(matrix)
    pprint.pprint(reversal_matrix)

    column_number, replacing_column = read_column_from_file(column_file_name)
    replacing_column = replacing_column.reshape(-1,1)

    print(f'\nThis column will replace #{column_number}:')
    pprint.pprint(replacing_column)

    l = reversal_matrix * replacing_column

    print('\nl vector equals: ')
    pprint.pprint(l)

    if l[column_number - 1][0] == 0:
        return None

    l_with_roof = copy.deepcopy(l)
    l_with_roof[column_number - 1] = -1
    l_with_triangle = l_with_roof * (-1 / l[column_number - 1])
    Q = np.identity(len(matrix))

    for row in range(0, len(Q)):
        Q[row][column_number - 1] = l_with_triangle[row]

    response = Q * reversal_matrix

    return response
```