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 4](#_Toc127292736)

[1 Выполнение работы 5](#_Toc127292737)

[2 Тестирование программного продукта 6](#_Toc127292738)

[Заключение 7](#_Toc127292739)

[ПРИЛОЖЕНИЕ А Листинг кода 8](#_Toc127292740)

# ВВЕДЕНИЕ

Как известно, алгоритм обращения квадратной матрицы размерности 𝑛 × 𝑛 имеет время выполнения Θ(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 𝐴̅ размерности 𝑛 × 𝑛, использующий дополнительные данные: – матрицу 𝐴, отличающуюся от матрицы 𝐴̅единственным столбцом с номером 𝑖; – матрицу 𝐴-1, обратную 𝐴. Как будет показано, используя эти данные, можно снизить временную сложность поиска обратной матрицы до Θ(𝑛2 ).

# 1 Выполнение работы

Используя данные, предоставленные в условии лабораторной работы, составим алгоритм нахождения матрицы 𝐴̅−1 .

1 Найти вектор 𝑙 = 𝐴 −1𝑥. Если 𝑙𝑖 = 0, матрица 𝐴̅необратима, алгоритм завершает работу. Иначе матрица 𝐴̅ обратима.

2 Создать вектор 𝑙 ̃, который получается из вектора 𝑙 путём замены в нём 𝑖-го элемента на −1.

3 Найти 𝑙 ̂ = (− 1/ 𝑙𝑖) 𝑙 ̃.

4 Сформировать матрицу 𝑄, путём замены в единичной матрице 𝐸 𝑖-го столбца столбцом 𝑙 ̂ .

5 Найти 𝐴̅−1 = 𝑄𝐴 −1 . Временная сложность представленного алгоритма равна Θ(𝑛2 ). Действительно, шаги 1 − 4 могут быть выполнены за Θ(𝑛2 ) операций. Умножение матриц, однако, требует 𝜔(𝑛2 ) операций, однако матрица 𝑄 разрежена: в каждой её строке не более двух ненулевых элементов, номера которых известны, поэтому умножение можно выполнить за Θ(𝑛2 ) операций.

# 2 тестирование программного продукта

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

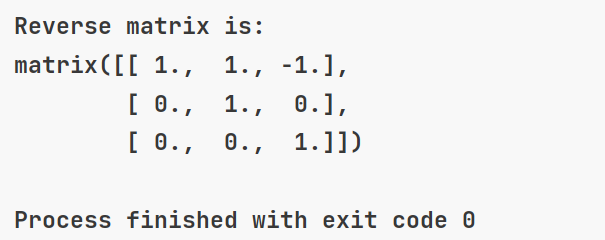


Рисунок 1. Результат вывода программного продукта

При 𝑙𝑖 = 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**