Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Лабораторная работа №1
по дисциплине
“Линейная алгебра и анализ данных”

Семестр I

Выполнили:
студенты

Тиганов Вадим Игоревич
гр. J3112
ИСУ 467701

Вагин Арсений Антонович
гр. J3112
ИСУ 465339

Отчет сдан:
XX.12.2024

Санкт-Петербург
2024

Цель лабораторной работы

Освоить основные концепции линейной алгебры и анализа данных по работе с матрицами. Научиться реализовывать и тестировать алгоритмы работы с матрицами в разреженно-строчном формате. Изучить и понять принципы работы алгоритмов, а также .tex верстания для создания отчета.

Задачи лабораторной работы

1. Реализация хранения матриц в разреженно-строчном виде.
2. Реализация операций над матрицами.
3. Вычисление определителя и проверка существования обратной матрицы.
4. Тестирование и проверка правильности работы алгоритмов.
5. Верстка отчета в формате LaTeX

Ход выполнения лабораторной работы

Задача 1

Задача заключалась в реализации следующих функий в классе: (был выбран ЯП Python, полный листинг кода см. в приложении А)

- Ввод матрицы заданного размера пользователем.
- Подсчет следа матрицы.
- Поиск и вывод элемента матрицы по заданным индексам.
- Тестирование работы программы и создание консольного пользовательского интерфейса.

Использованные библиотеки и инструменты языка

В ходе написания программы были использованы только стандартные средства языка. Также для удобства и лучшей читаемости кода была импортирована библиотека typing.

Реализация функций и основные идеи

Все функции были реализованы в классе MatrixKeeper

Написаны функции:
1. inputMatrix - ввод матрицы пользователем,
2. trace - поиск следа матрицы,
3. findByIndex - поиск элемента по введенному индексу.

Суть работы алгоритмов:

- inputMatrix: Приглашение пользователя ко вводу. Вначале через пробел вводятся два числа типа int - размер матрицы. Вторым приглашением вводится матрица по строке, элементы в строке разделяются пробелом.
- trace: След матрицы - сумма элементов главной диагонали этой матрицы. Циклом, оставаясь в пределах матрицы, проходимся по элементам с индексами вида [i][i], считаем сумму таких элементов. Можем так делать по той причине, что матрица имеет следующую структуру в классе:

```
self.matrix: Optional[List[List[float]]]
```

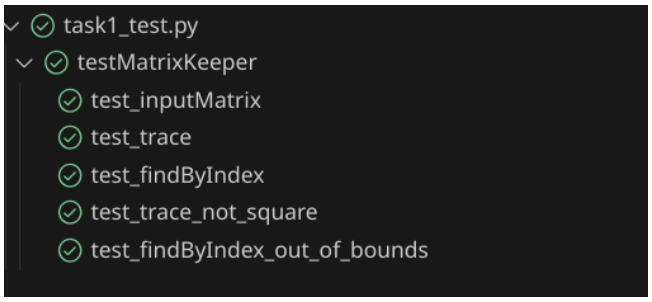
- то есть храним матрицу как список, каждый элемент которого является тоже списком.

- findByIndex: возвращаем элемент из матрицы, отнимая от индексов по единице, т.к. в ЯП отсчет начинается с нуля.

```
return self.matrix[n-1][m-1]
```

Тестирование программы

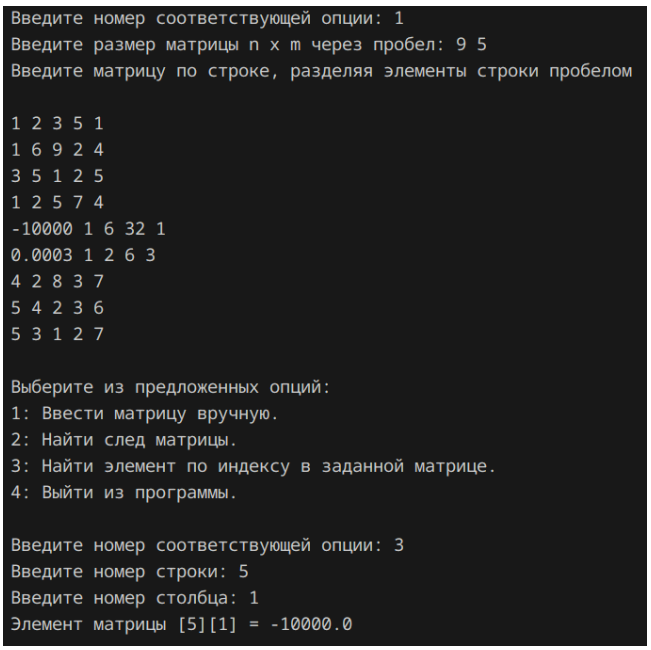
Написаны юниттесты для каждой функции класса с помощью стандартной библиотеки unittest. Также протестировано вручную.
Листинг кода теста см. в приложении A-test.



Успешное прохождение unittests

```
Выберите из предложенных опций:
1: Ввести матрицу вручную.
2: Найти след матрицы.
3: Найти элемент по индексу в заданной матрице.
4: Выйти из программы.

Введите номер соответствующей опции: 1
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

2 2
2 2

Выберите из предложенных опций:
1: Ввести матрицу вручную.
2: Найти след матрицы.
3: Найти элемент по индексу в заданной матрице.
4: Выйти из программы.

Введите номер соответствующей опции: 2
След матрицы: 4.0
```

Ручной тест поиска следа матрицы 2x2 со всеми элементами, равными 2.



Ручной тест поиска страшного элемента в страшной матрице 9x5.

Итак, справились с первым заданием.

Задача 2

Во второй задаче требуется реализовать три функции для операций с матрицами: (полный листинг кода см. в приложении Б)

- Сложение двух матриц.
- Умножение двух матриц.
- Умножение матрицы на скаляр.

Использованные библиотеки и инструменты языка

В ходе написания программы были использованы только стандартные средства языка. Также для удобства и лучшей читаемости кода была импортирована библиотека typing.

Были реализованы 3 функции:
1. matrixAddition - сложение двух матриц.
2. matrixByMatrixMultiplication - перемножение двух матриц.
3. matrixScalarMultiplication - умножение одной из двух матриц на заданное число.

Согласно техническому заданию, функция ввода матрицы пользователем была импортирована из файла предыдущего задания. (inputMatrix)

Суть работы алгоритмов:

- matrixAddition: Классическое сложение матрицы. Возвращаем матрицу, где каждый элемент с определенными индексами равен сумме элементов с соответствующими индексами из складываемых матриц.

$$(A + B)_{i,k} = A_{i,k} + B_{i,k}$$

- matrixByMatrixMultiplication: Умножение матрицы на матрицу. Возвращаем матрицу, где каждый элемент с определенными индексами равен сумме произведений элементов соответствующей строки первой матрицы и столбца второй матрицы.

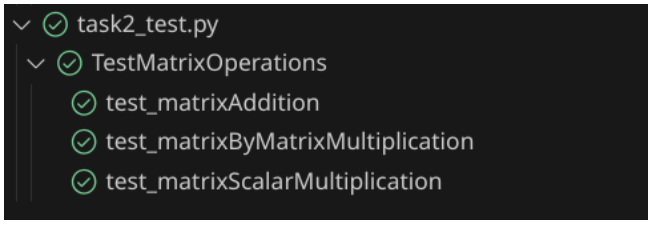
$$(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k} \cdot B_{k,j}$$

- matrixScalarMultiplication: Умножение матрицы на скаляр. Возвращаем матрицу, где каждый элемент равен произведению соответствующего элемента исходной матрицы и скаляра.

$$(cA)_{i,j} = c \cdot A_{i,j}$$

Тестирование программы

Написаны юниттесты для каждой функции класса с помощью стандартной библиотеки unittest. Также протестировано вручную.
Листинг кода теста см. в приложении Б-test.



Успешное прохождение unittests

```
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

1 2
3 4

Выберите из предложенных опций:
1: Ввести первую матрицу вручную.
2: Ввести вторую матрицу вручную.
3: Сложить две матрицы.
4: Умножить матрицу на матрицу.
5: Умножить матрицу на скаляр.
6: Выйти из программы.

Введите номер соответствующей опции: 2
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

6 5
7 9

Выберите из предложенных опций:
1: Ввести первую матрицу вручную.
2: Ввести вторую матрицу вручную.
3: Сложить две матрицы.
4: Умножить матрицу на матрицу.
5: Умножить матрицу на скаляр.
6: Выйти из программы.

Введите номер соответствующей опции: 3
Результат сложения матриц:
[7.0, 7.0]
[10.0, 13.0]
```

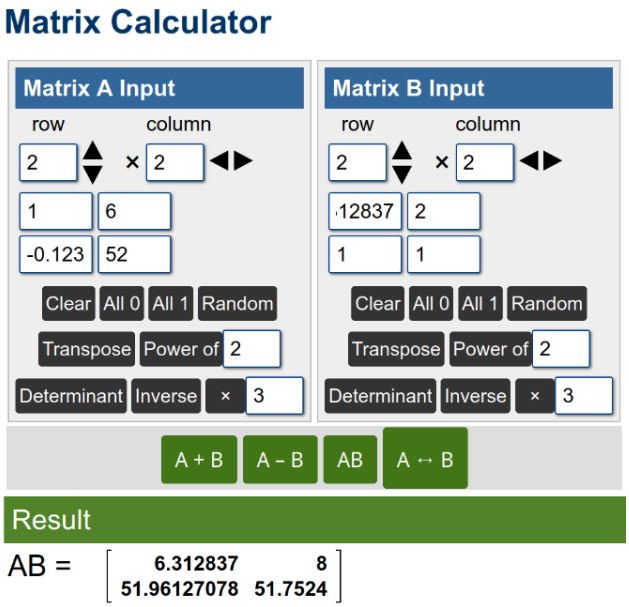
Ручной тест сложения двух матриц 2 на 2.

```
Введите номер соответствующей опции: 1
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

1 2
6 7

Выберите из предложенных опций:
1: Ввести первую матрицу вручную.
2: Ввести вторую матрицу вручную.
3: Сложить две матрицы.
4: Умножить матрицу на матрицу.
5: Умножить матрицу на скаляр.
6: Выйти из программы.

Введите номер соответствующей опции: 5
Выберите матрицу для умножения на скаляр (1 - первая матрица, 2 - вторая матрица): 1
Введите скаляр: 2.718281828459045
Результат умножения матрицы на скаляр:
[2.718281828459045, 5.43656365691809]
[16.30969097075427, 19.027972799213316]
```

Ручной тест умножения матрицы 2 на 2 на число Эйлера с небольшим количеством знаков после запятой.

```
Введите номер соответствующей опции: 1
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

1 6
-0.1238 52

Выберите из предложенных опций:
1: Ввести первую матрицу вручную.
2: Ввести вторую матрицу вручную.
3: Сложить две матрицы.
4: Умножить матрицу на матрицу.
5: Умножить матрицу на скаляр.
6: Выйти из программы.

Введите номер соответствующей опции: 2
Введите размер матрицы n x m через пробел: 2 2
Введите матрицу по строке, разделяя элементы строки пробелом

0.312837 2
1 1

Выберите из предложенных опций:
1: Ввести первую матрицу вручную.
2: Ввести вторую матрицу вручную.
3: Сложить две матрицы.
4: Умножить матрицу на матрицу.
5: Умножить матрицу на скаляр.
6: Выйти из программы.

Введите номер соответствующей опции: 4
Результат умножения матриц:
[6.312837, 8.0]
[51.9612707794, 51.7524]
```

Ручной тест умножения двух страшных матриц 2 на 2.



Удостоверимся в правильности полученного результата, умножив матрицы на сайте-калькуляторе. Результат правильный.

Справились со вторым заданием.

Задача 3

В третьей задача требуется реализовать следующие функции: (полный листинг кода см. в приложении В)

- Вычисление определителя матрицы. По тз - матрицы размером до 100x100
- Ответ на вопрос: существует ли матрица, обратная данной.

Использованные библиотеки и инструменты языка

В ходе написания программы были использованы только стандартные средства языка. Также для удобства и лучшей читаемости кода была импортирована библиотека typing.

Были реализованы 3 функции:
1. determinantOfMatrix - поиск определителя матрицы.
2. isMatrixInvertable - функция, которая отвечает на вопрос, существует ли обратная матрица к данной.
3. gauss - вспомогательная функция - метод Гаусса, с помощью которого считал определитель любой квадратной матрицы.

Суть работы алгоритмов:

- determinantOfMatrix: оболочка функции, проверяем размер матрицы и обращаемся к функции Гаусса для поиска определителя.

- isMatrixInvertable: сравниваем полученный из первой функции определитель с нулем, возвращаем логическое значение "да" или "нет".

- gauss: самая интересная функция. Для матрицы 1x1 и 2x2 посчитано вручную.

```
    n = len(matrix)
    if n == 1:
      return matrix[0][0]
    elif n == 2:
      return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
```

1x1: Определитель равен единственному элементу.
2x2: Определитель считаем как разность произведения элементов главной и побочной диагоналей.

Проверка на вырожденную матрицу:

```
    for i in range(n):
      max_row = max(range(i, n), key=lambda r: abs(matrix[r][i]))
      if matrix[max_row][i] == 0:
        raise ValueError("Matrix is singular,       determinant is zero.")
```

Как работает:

Цикл проходит по каждому столбцу i от 0 до $n-1$. На каждом шаге ищется ведущий элемент в текущем столбце. Для этого используется команда:

```
    max_row = max(range(i, n), key=lambda r: abs(matrix[r][i]))
```

Здесь range(i, n) генерирует индексы строк от текущей строки i до последней n-1, lambda r: abs(matrix[r][i]) вычисляет модуль элемента в текущем столбце для каждой строки r, а max(..., key=...) выбирает индекс строки max_row, где модуль элемента в столбце максимален.

Далее выполняется проверка:

```
    if matrix[max_row][i] == 0:
      raise ValueError("Matrix is singular, determinant is zero.")
```

Если максимальный элемент в столбце равен 0, то все элементы ниже i-й строки в этом столбце тоже равны 0. Это означает, что матрица вырожденная, и её определитель равен нулю, поэтому дальнейшие вычисления прекращаются.

Приведение к нижнетреугольному виду и вычисление определителя:

Для приведения матрицы к нижнетреугольному виду используется следующий алгоритм. Если строка с максимальным элементом в текущем столбце max_row не совпадает с текущей строкой i, строки меняются местами:

```
if max_row != i:
  matrix[i], matrix[max_row] = matrix[max_row], matrix[i]
  swap_count += 1
```

После этого элементы ниже ведущего в текущем столбце обнуляются. Для каждой строки j, начиная с i + 1, вычисляется коэффициент factor, с помощью которого корректируются элементы строки:

```
for j in range(i + 1, n):
  factor = matrix[j][i] / matrix[i][i]
  for k in range(i, n):
    matrix[j][k] -= factor * matrix[i][k]
```

После завершения приведения к нижнетреугольному виду вычисляется определитель матрицы. Определитель является произведением всех диагональных элементов приведённой матрицы:

```
determinant = 1.0
for i in range(n):
  determinant *= matrix[i][i]
```

Учитывается знак определителя, зависящий от числа перестановок строк:

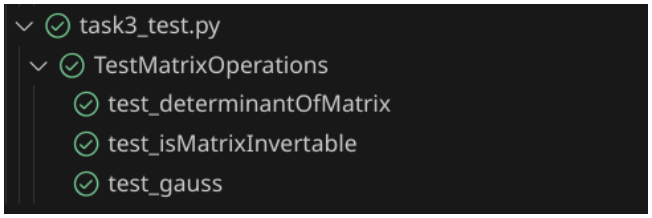
```
return (-1) ** swap_count * determinant
```

Здесь каждая перестановка строк меняет знак определителя, что учитывается выражением (-1) ** swap_count.

Таким образом реализован метод Гаусса для поиска определителя любой квадратной матрицы.

Тестирование программы

Написаны юниттесты для каждой функции класса с помощью стандартной библиотеки unittest. Также протестировано вручную.
Листинг кода теста см. в приложении B-test.



Успешное прохождение unittests

Вывод

Приложения

Список использованной литературы