

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

**«Исследование методов работы с матрицами и векторами с помощью
библиотеки NumPy»**

ОТЧЕТ
по лабораторной работе №3
дисциплины
«Технологии распознавания образов»

Выполнила:
Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

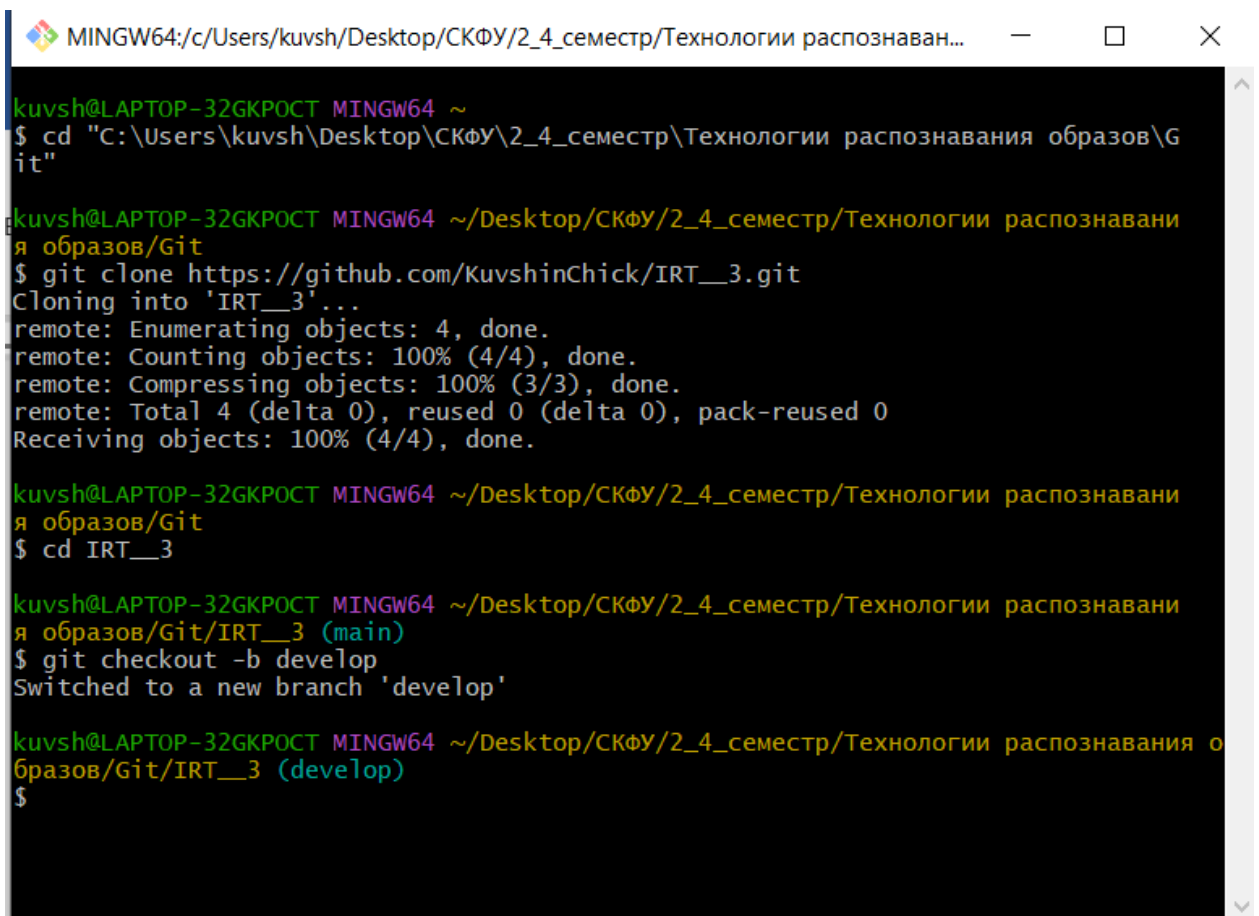
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Цель работы: исследовать методы работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).
3. Выполните клонирование созданного репозитория на рабочий компьютер.
4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Технологии распознаван...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Технологии распознавания образов\Git"

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания образов/Git
$ git clone https://github.com/KuvshinChick/IRT__3.git
Cloning into 'IRT__3'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

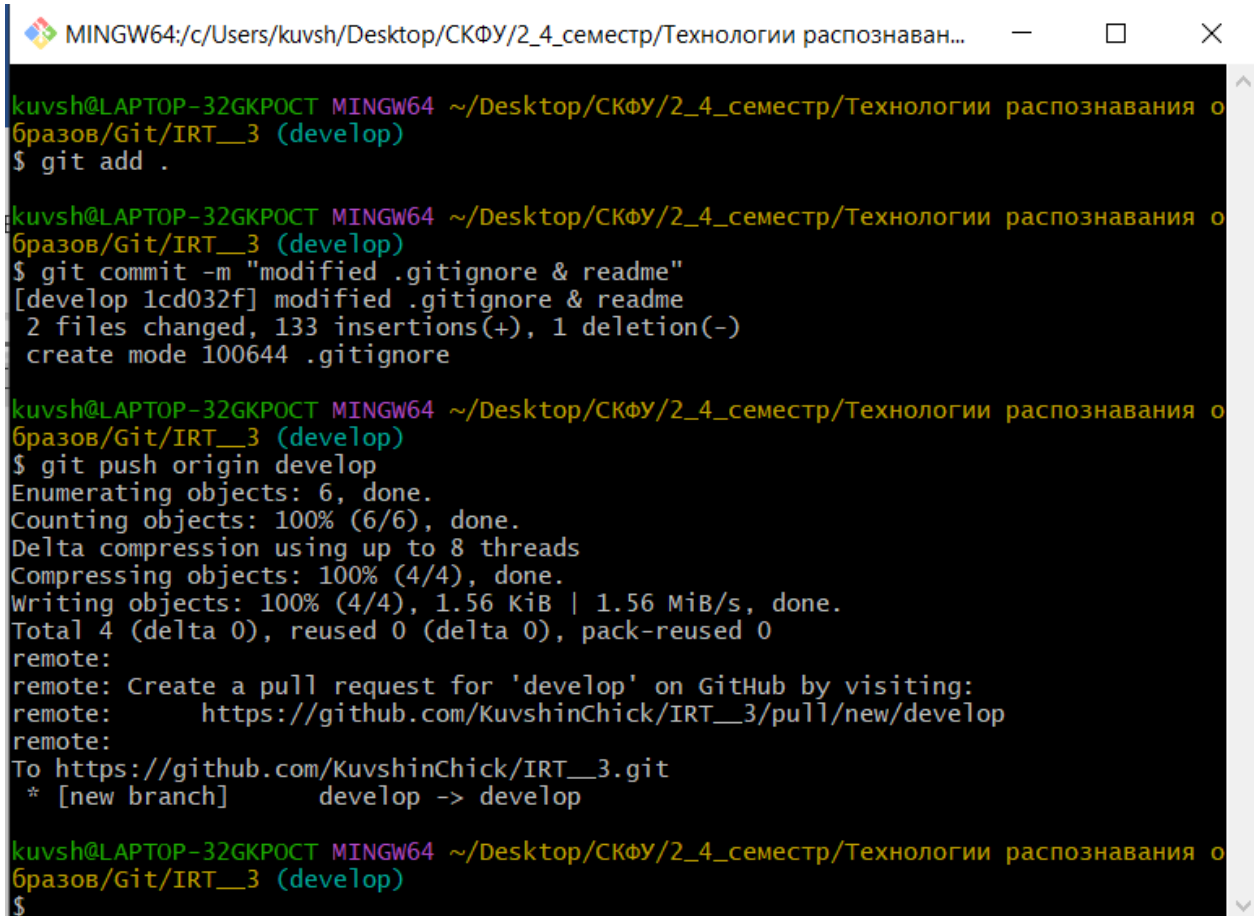
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания образов/Git
$ cd IRT__3

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания образов/Git/IRT__3 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания образов/Git/IRT__3 (develop)
$
```

Рисунок 3.1 – Клонирование репозитория и создание ветки develop

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.



```
MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Технологии распознаван...  
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания о  
бразов/Git/IRT__3 (develop)  
$ git add .  
  
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания о  
бразов/Git/IRT__3 (develop)  
$ git commit -m "modified .gitignore & readme"  
[develop 1cd032f] modified .gitignore & readme  
2 files changed, 133 insertions(+), 1 deletion(-)  
create mode 100644 .gitignore  
  
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания о  
бразов/Git/IRT__3 (develop)  
$ git push origin develop  
Enumerating objects: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (4/4), done.  
Writing objects: 100% (4/4), 1.56 KiB | 1.56 MiB/s, done.  
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
remote:  
remote: Create a pull request for 'develop' on GitHub by visiting:  
remote:      https://github.com/KuvshinChick/IRT__3/pull/new/develop  
remote:  
To https://github.com/KuvshinChick/IRT__3.git  
* [new branch]      develop -> develop  
  
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Технологии распознавания о  
бразов/Git/IRT__3 (develop)  
$
```

Рисунок 3.2 – Обновление .gitignore и readme

6. Проработать примеры лабораторной работы.

IRT_3 IRT3_ex.ipynb

Project IRT_3 C:\Users\kuvsh\Desktop\CKB\V2_4

gitignore IRT3_ex.ipynb LICENSE README.md External Libraries Scratches and Consoles

Full Requests

Bookmarks

Terminal

Python Packages

Python Console

Services

Current File

Managed Jupyter server: auto-start

Python 3 (ipykernel)

Trusted

Вектор

Вектор-строка

```
In 4: 1 v_hor_np = np.array([1, 2])
      2 print(v_hor_np)

[1 2]
```

```
In 6: 1 v_hor_zeros_v1 = np.zeros((5,))
      2 print(v_hor_zeros_v1)

[0. 0. 0. 0. 0.]
```

```
In 7: 1 v_hor_zeros_v2 = np.zeros((1, 5))
      2 print(v_hor_zeros_v2)

[[0. 0. 0. 0. 0.]]
```

IRT_3 IRT3_ex.ipynb

Project IRT_3 C:\Users\kuvsh\Desktop\CKB\V2_4

gitignore IRT3_ex.ipynb LICENSE README.md External Libraries Scratches and Consoles

Full Requests

Bookmarks

Terminal

Python Packages

Python Console

Services

Current File

Managed Jupyter server: auto-start

Python 3 (ipykernel)

Trusted

```
In 8: 1 v_hor_one_v1 = np.ones((5,))
      2 print(v_hor_one_v1)
      3 v_hor_one_v2 = np.ones((1, 5))
      4 print(v_hor_one_v2)

[1. 1. 1. 1. 1.]
[[1. 1. 1. 1. 1.]]
```

Вектор-столбец

```
In 9: 1 v_vert_np = np.array([[1], [2]])
      2 print(v_vert_np)

[[1]
 [2]]
```

```
In 10: 1 v_vert_zeros = np.zeros((5, 1))
       2 print(v_vert_zeros)

[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

```
In 13: m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_arr)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

In 14: m_sqr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
m_sqr_arr = np.array(m_sqr)
print(m_sqr_arr)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

In 15: m_sqr_mx = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_mx)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

In 16: m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
```

Рисунок 3.3 – Результат проработки примеров

7. Создать ноутбук, в котором будут приведены собственные примеры на языке Python для каждого из представленных свойств матричных вычислений.

```
In 63: import numpy as np

In 64: A = np.matrix('3 6 5; 9 8 1')
t_matrix = (A.T).T
print(t_matrix)
[[3 6 5]
 [9 8 1]]

In 65: A = np.matrix('2 3 6; 5 9 1')
B = np.matrix('4 3 5; 9 8 0')
first = (A + B).T
second = A.T + B.T
print(first)
print(second)
```

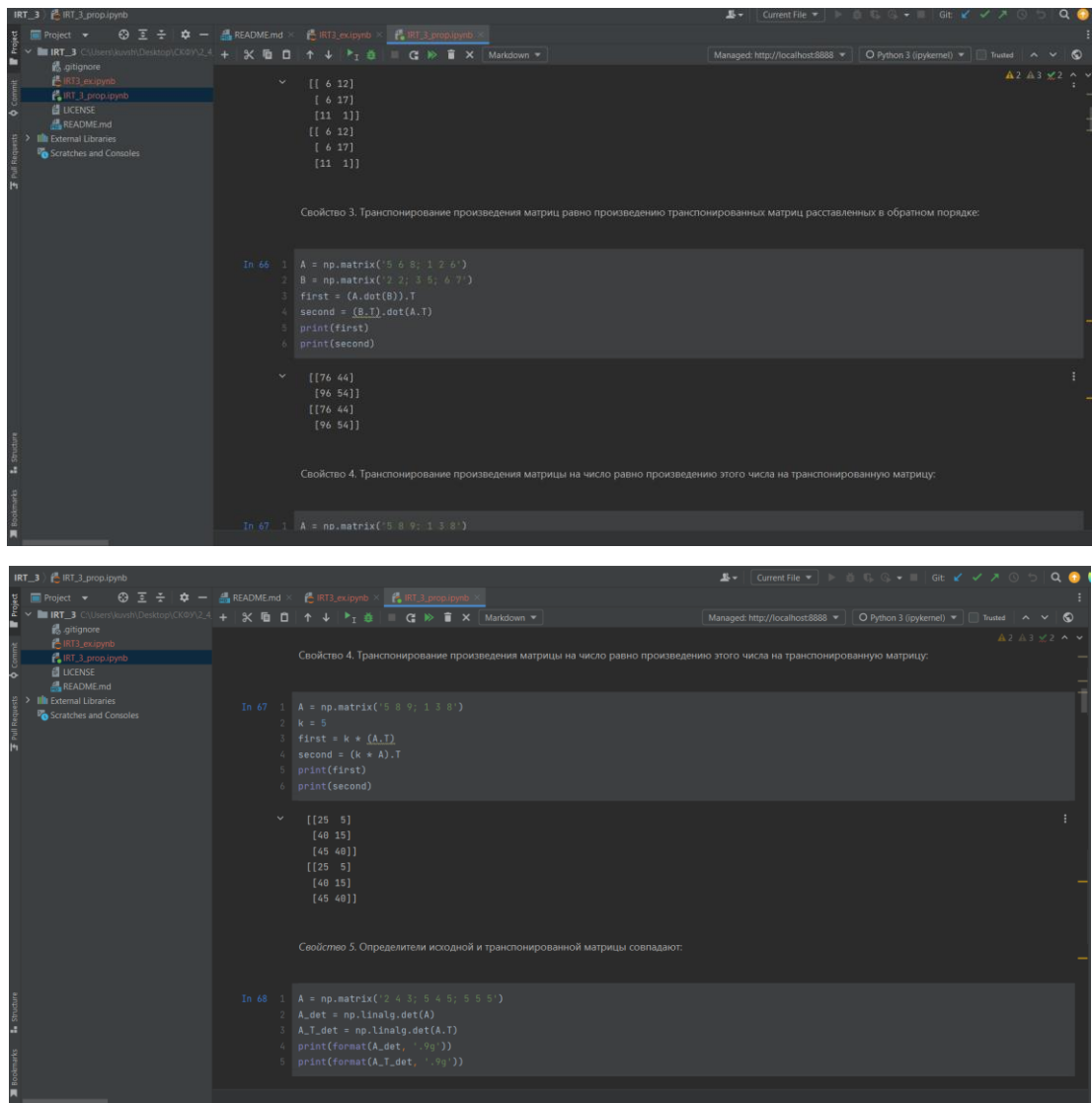


Рисунок 3.4 – Результат проработки свойств матричных вычислений

8. Создать ноутбук, в котором будут приведены собственные примеры решения систем линейных уравнений матричным методом и методом Крамера.

Матричный метод решения СЛАУ

Пусть дана система линейных уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + \dots + a_{1n}x_n = b_1; \\ \dots \\ a_{n1}x_1 + \dots + a_{nn}x_n = b_n; \end{cases}$$

Её можно переписать в матричной форме: $AX = B$, где A — основная матрица системы. B и X — столбцы свободных членов и решений системы соответственно:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$
$$B = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$
$$X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

Умножим это матричное уравнение слева на A^{-1} — матрицу, обратную к матрице A : $A^{-1}(AX) = A^{-1}B$

Так как $A^{-1}A = E$, получаем $X = A^{-1}B$. Правая часть этого уравнения даст столбец решений исходной системы. Условием применимости данного метода (как и вообще существования решения

Пример решения неоднородной системы линейных алгебраических уравнений:

$$\begin{cases} 3x + 2y - z = 4; \\ 2x - y + 5z = 23; \\ x + 7y - z = 5; \end{cases}$$
$$A = \begin{pmatrix} 3 & 2 & -1 \\ 2 & -1 & 5 \\ 1 & 7 & -1 \end{pmatrix}$$
$$B = \begin{pmatrix} 4 \\ 23 \\ 5 \end{pmatrix}$$
$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Убедимся в том, что определитель матрицы, составленный из коэффициентов при неизвестных системы линейных алгебраических уравнений не равен нулю.

```
In 130: 1 import numpy as np
2 A = np.matrix('3 2 -1; 2 -1 5; 1 7 -1')
3 B = np.matrix('4; 23; 5')
4 print('Определитель матрицы: ', np.linalg.det(A))
```

Определитель матрицы: -103.00000000000001

Найдем обратную матрицу:

```
In 131: 1 A_inv = np.linalg.inv(A)
2 print("Обратная матрица: ")
3 print(A_inv)
```

Обратная матрица:

```
[[ 0.33009709  0.04854369 -0.08737864]
 [-0.06796117  0.01941748  0.16594854]
 [-0.14563107  0.18446602  0.06796117]]
```

Найдем неизвестные. Для этого перемножим обратную матрицу и столбец свободных членов.

```
In 132: 1 X = A_inv.dot(B)
2 print("Решение системы: ")
3 print(X)
```

Решение системы:

```
[[2.]
 [1.]
 [4.]]
```

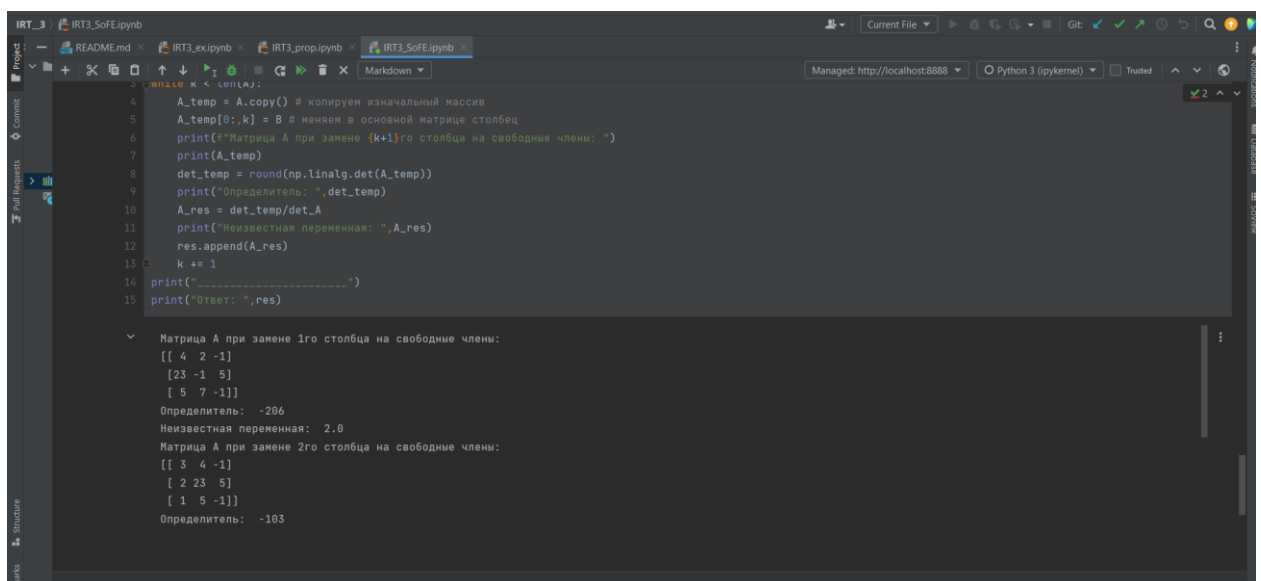
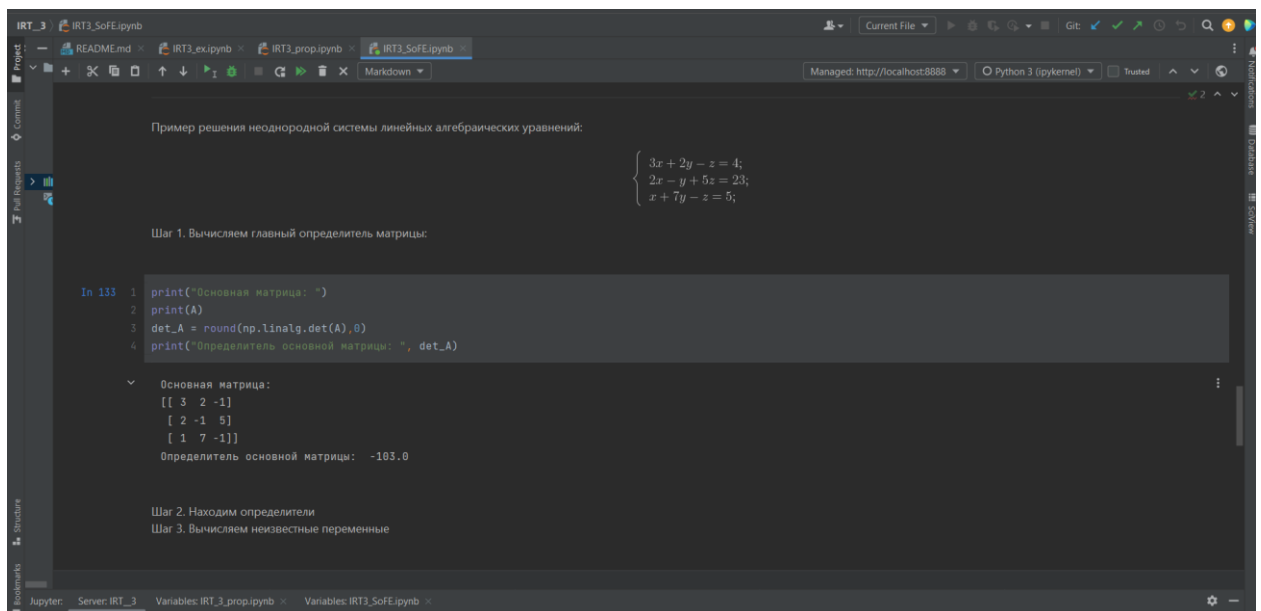
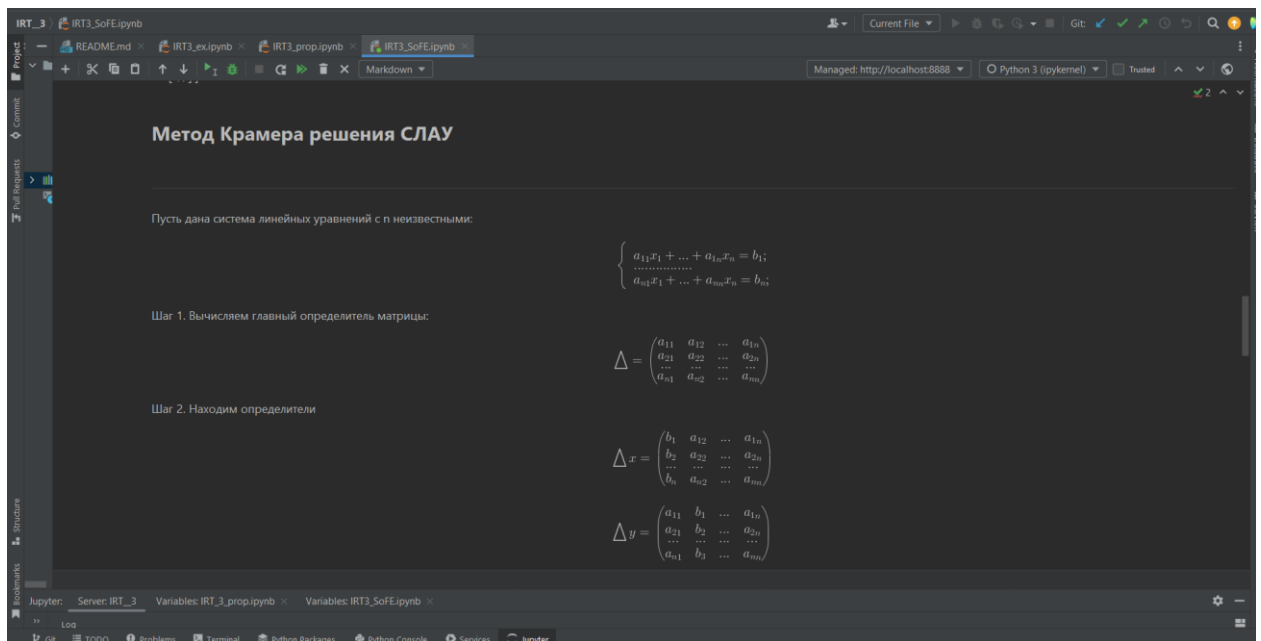


Рисунок 3.5 – Результат выполнения задания

9. Зафиксируйте сделанные изменения в репозитории.
10. Выполните слияние ветки для разработки с веткой main (master).
11. Отправьте сделанные изменения на сервер GitHub.

Контрольные вопросы

1. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python.

Вектор-строка

Вектор-строка имеет следующую математическую запись.

$$v = (1 \ 2) \quad (3)$$

Такой вектор в *Python* можно задать следующим образом.

```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np )
[1 2]
```

Если необходимо создать **нулевой** или **единичный вектор**, то есть вектор, у которого все элементы нули либо единицы, то можно использовать специальные функции из библиотеки *Numpy*.

Создадим нулевую вектор-строку размера 5.

```
>>> v_hor_zeros_v1 = np.zeros((5,))
>>> print(v_hor_zeros_v1 )
[0. 0. 0. 0. 0.]
```

Вектор-столбец

Вектор-столбец имеет следующую математическую запись.

$$v = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad (4)$$

В общем виде вектор столбец можно задать следующим образом.

```
>>> v_vert_np = np.array([[1], [2]])
>>> print(v_vert_np)
[[1]
 [2]]
```

Квадратная матрица

Довольно часто, на практике, приходится работать с **квадратными матрицами**. Квадратной называется матрица, у которой количество столбцов и строк совпадает. В общем виде они выглядят так.

$$M_{sqr} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}. \quad (5)$$

Создадим следующую матрицу.

$$M_{sqr} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}. \quad (6)$$

В *Numpy* можно создать квадратную матрицу с помощью метода `array()`.

```
>>> m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> print(m_sqr_arr)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Диагональная матрица

Особым видом квадратной матрицы является **диагональная** – это такая матрица, у которой все элементы, кроме тех, что расположены на главной диагонали, равны нулю.

$$M_{diag} = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix} \quad (7)$$

Диагональную матрицу можно построить вручную, задав только значения элементам на главной диагонали.

```
>>> m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
>>> m_diag_np = np.matrix(m_diag)
>>> print(m_diag_np)
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

Единичная матрица

Единичной матрицей называют такую квадратную матрицу, у которой элементы главной диагонали равны единицы, а все остальные нулю.

$$E = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}. \quad (8)$$

Создадим единичную матрицу на базе списка, который передадим в качестве аргумента функции *matrix()*.

```
>>> m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
>>> m_e_np = np.matrix(m_e)
>>> print(m_e_np)
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

2. Как выполняется транспонирование матриц?

Транспонируем матрицу с помощью метода ***transpose()***:

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
 [2 5]
 [3 6]]
```

3. Приведите свойства операции транспонирования матриц.

Свойство 1. Дважды транспонированная матрица равна исходной матрице:

$$(A^T)^T = A.$$

➤ Численный пример

$$\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T \right)^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
[[1 2 3]
 [4 5 6]]
>>> R = (A.T).T
>>> print(R)
[[1 2 3]
 [4 5 6]]
```

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц:

$$(A + B)^T = A^T + B^T.$$

➤ Численный пример

$$\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 8 & 9 \\ 0 & 7 & 5 \end{pmatrix} \right)^T = \begin{pmatrix} 8 & 10 & 12 \\ 4 & 12 & 11 \end{pmatrix}^T = \begin{pmatrix} 8 & 4 \\ 10 & 12 \\ 12 & 11 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T + \begin{pmatrix} 7 & 8 & 9 \\ 0 & 7 & 5 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 0 \\ 8 & 7 \\ 9 & 5 \end{pmatrix} = \begin{pmatrix} 8 & 4 \\ 10 & 12 \\ 12 & 11 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8 9; 0 7 5')
>>> L = (A + B).T
>>> R = A.T + B.T
>>> print(L)
[[ 8  4]
 [10 12]
 [12 11]]
>>> print(R)
[[ 8  4]
 [10 12]
 [12 11]]
```

Свойство 3. Транспонирование произведения матриц равно произведению транспонированных матриц расставленных в обратном порядке:

$$(AB)^T = B^T A^T.$$

➤ Численный пример

$$\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \right)^T = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}^T = \begin{pmatrix} 19 & 43 \\ 22 & 50 \end{pmatrix}.$$
$$\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}^T \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^T = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \times \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 19 & 43 \\ 22 & 50 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = (A.dot(B)).T
>>> R = (B.T).dot(A.T)
>>> print(L)
[[19 43]
 [22 50]]
>>> print(R)
[[19 43]
 [22 50]]
```

В данном примере, для умножения матриц, использовалась функция ***dot()*** из библиотеки *Numpy*.

Свойство 4. Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу:

$$(\lambda A)^T = \lambda A^T.$$

➤ Численный пример

$$\left(3 \cdot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}\right)^T = \begin{pmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \end{pmatrix}^T = \begin{pmatrix} 3 & 12 \\ 6 & 15 \\ 9 & 18 \end{pmatrix}.$$

$$3 \cdot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T = 3 \cdot \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 3 & 12 \\ 6 & 15 \\ 9 & 18 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> k = 3
>>> L = (k * A).T
>>> R = k * (A.T)
>>> print(L)
[[ 3 12]
 [ 6 15]
 [ 9 18]]
\>>> print(R)
[[ 3 12]
 [ 6 15]
 [ 9 18]]
```

***Свойство 5*.** Определители исходной и транспонированной матрицы совпадают:

$$|A| = |A^T|.$$

➤ Численный пример

$$\det \left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \right) = 4 - 6 = -2.$$

$$\det \left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^T \right) = \det \left(\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \right) = 4 - 6 = -2.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> A_det = np.linalg.det(A)
>>> A_T_det = np.linalg.det(A.T)
>>> print(format(A_det, '.9g'))
-2
>>> print(format(A_T_det, '.9g'))
-2
```

Ввиду особенностей *Python* при работе с числами с плавающей точкой, в данном примере вычисления определителя рассматриваются только первые девять значащих цифр после запятой (за это отвечает параметр `'.9g'`).

4. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц?

Решим задачу транспонирования матрицы на *Python*. Создадим матрицу *A*:

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
[[1 2 3]
 [4 5 6]]
```

Транспонируем матрицу с помощью метода ***transpose()***:

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
 [2 5]
 [3 6]]
```

5. Какие существуют основные действия над матрицами?

Умножение матрицы на число

Сложение матриц

Умножение матриц

Определитель матрицы

Транспонирование матрицы

6. Как осуществляется умножение матрицы на число?

► Пример на Python

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
[[ 3  6  9]
 [12 15 18]]
```

7. Какие свойства операции умножения матрицы на число?

Свойство 1. Произведение единицы и любой заданной матрицы равно заданной матрице:

$$1 \cdot A = A.$$

➤ Численный пример

$$1 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> L = 1 * A
>>> R = A
>>> print(L)
[[1 2]
 [3 4]]
>>> print(R)
[[1 2]
 [3 4]]
```

Свойство 2. Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы:

$$0 \cdot A = Z.$$

➤ Численный пример

$$0 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = 0 * A
>>> R = Z
>>> print(L)
[[0 0]
 [0 0]]
>>> print(R)
[[0 0]
 [0 0]]
```

Свойство 3. Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел:

$$(\alpha + \beta) \cdot A = \alpha \cdot A + \beta \cdot A.$$

➤ Численный пример

$$(2 + 3) \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + 3 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix} + \begin{pmatrix} 3 & 6 \\ 9 & 12 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix},$$

$$5 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p + q) * A
>>> R = p * A + q * A
>>> print(L
[[ 5 10]
 [15 20]]
>>> print(R)
[[ 5 10]
 [15 20]]
```

Свойство 4. Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число:

$$(\alpha \cdot \beta) \cdot A = \alpha \cdot (\beta \cdot A).$$

➤ Численный пример

$$(2 \cdot 3) \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 2 \cdot \left(3 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \right) = 2 \cdot \begin{pmatrix} 3 & 6 \\ 9 & 12 \end{pmatrix} = \begin{pmatrix} 6 & 12 \\ 18 & 24 \end{pmatrix},$$

$$(2 \cdot 3) \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 6 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 6 & 12 \\ 18 & 24 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p * q) * A
>>> R = p * (q * A)
>>> print(L)
[[ 6 12]
 [18 24]]
>>> print(R)
[[ 6 12]
 [18 24]]
```

Свойство 5. Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число:

$$\lambda \cdot (A + B) = \lambda \cdot A + \lambda \cdot B.$$

➤ Численный пример

$$3 \cdot \left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \right) = 3 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + 3 \cdot \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 18 & 24 \\ 30 & 36 \end{pmatrix},$$
$$3 \cdot \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix} = \begin{pmatrix} 18 & 24 \\ 30 & 36 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> k = 3
>>> L = k * (A + B)
>>> R = k * A + k * B
>>> print(L)
[[18 24]
 [30 36]]
>>> print(R)
[[18 24]
 [30 36]]
```

8. Как осуществляется операции сложения и вычитания матриц?

➤ Пример на Python

```
>>> A = np.matrix('1 6 3; 8 2 7')
>>> B = np.matrix('8 1 5; 6 9 12')
>>> C = A + B
>>> print(C)
[[ 9  7  8]
 [14 11 19]]
```

9. Каковы свойства операций сложения и вычитания матриц?

Свойство 1. Коммутативность сложения. От перестановки матриц их сумма не изменяется:

$$A + B = B + A.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A + B
>>> R = B + A
>>> print(L)
[[ 6  8]
 [10 12]]
>>> print(R)
[[ 6  8]
 [10 12]]
```

Свойство 2. Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться:

$$A + (B + C) = (A + B) + C.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \left(\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} + \begin{pmatrix} 1 & 7 \\ 9 & 3 \end{pmatrix} \right) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 6 & 13 \\ 16 & 11 \end{pmatrix} = \begin{pmatrix} 7 & 15 \\ 19 & 15 \end{pmatrix},$$
$$\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \right) + \begin{pmatrix} 1 & 7 \\ 9 & 3 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix} + \begin{pmatrix} 1 & 7 \\ 9 & 3 \end{pmatrix} = \begin{pmatrix} 7 & 15 \\ 19 & 15 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('1 7; 9 3')
>>> L = A + (B + C)
>>> R = (A + B) + C
>>> print(L)
[[ 7 15]
 [19 15]]
>>> print(R)
[[ 7 15]
 [19 15]]
```


Свойство 3. Для любой матрицы существует противоположная ей, такая, что их сумма является нулевой матрицей :

$$A + (-A) = Z.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + (-1) \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} -1 & -2 \\ -3 & -4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = A + (-1)*A
>>> print(L)
[[0 0]
 [0 0]]
>>> print(Z)
[[0 0]
 [0 0]]
```

10. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц?

```
>>> A = np.matrix('1 6 3; 8 2 7')
>>> B = np.matrix('8 1 5; 6 9 12')
>>> C = A + B
>>> print(C)
[[ 9  7  8]
 [14 11 19]]
```

11. Как осуществляется операция умножения матриц?

Решим задачу умножения матриц на языке *Python*. Для этого будем использовать функцию **dot()** из библиотеки *Numpy*:

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8; 9 1; 2 3')
>>> C = A.dot(B)
>>> print(C)
[[31 19]
 [85 55]]
```

12. Каковы свойства операции умножения матриц?

Свойство 1. Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет выполняться эта операция:

$$A \times (B \times C) = (A \times B) \times C.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \left(\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \times \begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix} \right) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 52 & 68 \\ 70 & 92 \end{pmatrix} = \begin{pmatrix} 192 & 252 \\ 436 & 572 \end{pmatrix},$$
$$\left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \right) \times \begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix} \times \begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 192 & 252 \\ 436 & 572 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('2 4; 7 8')
>>> L = A.dot(B.dot(C))
>>> R = (A.dot(B)).dot(C)
>>> print(L)
[[192 252]
 [436 572]]
>>> print(R)
[[192 252]
 [436 572]]
```

Свойство 2. Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц:

$$A \times (B + C) = A \times B + A \times C.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \left(\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} + \begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix} \right) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 7 & 10 \\ 14 & 16 \end{pmatrix} = \begin{pmatrix} 35 & 42 \\ 77 & 94 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix} + \begin{pmatrix} 16 & 20 \\ 34 & 44 \end{pmatrix} = \begin{pmatrix} 35 & 42 \\ 77 & 94 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('2 4; 7 8')
>>> L = A.dot(B + C)
>>> R = A.dot(B) + A.dot(C)
>>> print(L)
[[35 42]
 [77 94]]
>>> print(R)
[[35 42]
 [77 94]]
```

Свойство 3. Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется правило независимости произведения от перестановки множителей:

$$A \times B \neq B \times A.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix},$$

$$\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 23 & 34 \\ 31 & 46 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A.dot(B)
>>> R = B.dot(A)
>>> print(L)
[[19 22]
 [43 50]]
>>> print(R)
[[23 34]
 [31 46]]
```

Свойство 4. Произведение заданной матрицы на единичную равно исходной матрице:

$$E \times A = A \times E = A.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> E = np.matrix('1 0; 0 1')
>>> L = E.dot(A)
>>> R = A.dot(E)
>>> print(L)
[[1 2]
 [3 4]]
>>> print(R)
[[1 2]
 [3 4]]
>>> print(A)
[[1 2]
 [3 4]]
```

Свойство 5. Произведение заданной матрицы на нулевую матрицу равно нулевой матрице:

$$Z \times A = A \times Z = Z.$$

➤ Численный пример

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

➤ Пример на Python

```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = Z.dot(A)
>>> R = A.dot(Z)
>>> print(L)
[[0 0]
 [0 0]]
>>> print(R)
[[0 0]
 [0 0]]
>>> print(Z)
[[0 0]
 [0 0]]
```

13. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц?

Есть три основных способа выполнить умножение матрицы NumPy:

- `np.dot(array a, array b)`: возвращает скалярное произведение или скалярное произведение двух массивов
- `np.matmul(array a, array b)`: возвращает матричное произведение двух массивов
- `np.multiply(array a, array b)`: возвращает поэлементное матричное умножение двух массивов

14. Что такое определитель матрицы? Каковы свойства определителя матрицы?

Определитель матрицы

Определитель матрицы размера (n -го порядка) является одной из ее численных характеристик. Определитель матрицы A обозначается как $|A|$ или $\det(A)$, его также называют детерминантом. Рассмотрим квадратную матрицу 2×2 в общем виде:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

Определитель такой матрицы вычисляется следующим образом:

$$|A| = \det(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}.$$

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании:

$$\det(A) = \det(A^T).$$

Свойство 2. Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

Свойство 3. При перестановке строк матрицы знак ее определителя меняется на противоположный:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}; \quad A' = \begin{pmatrix} a_{21} & a_{22} & \dots & a_{2n} \\ a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

$$\det(A) = -\det(A').$$

Свойство 4. Если у матрицы есть две одинаковые строки, то ее определитель равен нулю:

$$\begin{vmatrix} a_1 & a_2 & \dots & a_n \\ a_1 & a_2 & \dots & a_n \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

Свойство 5. Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \lambda \cdot a_{21} & \lambda \cdot a_{22} & \dots & \lambda \cdot a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \lambda \cdot \det(A).$$

Свойство 6. Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}.$$

Свойство 7. Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} + \beta \cdot a_{11} & a_{22} + \beta \cdot a_{12} & \dots & a_{2n} + \beta \cdot a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}.$$

Свойство 8. Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю:

$$a_{2i} = \alpha \cdot a_{1i} + \beta \cdot a_{3i}; \quad \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

15. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?

```
>>> A = np.matrix(' -4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1 2]
 [10 4 -1]
 [ 8 3 1]]
```

Для вычисления определителя этой матрицы воспользуемся функцией `det()` из пакета `linalg`.

```
>>> np.linalg.det(A)
-14.000000000000009
```

16. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?

Обратной матрицей A^{-1} матрицы A называют матрицу, удовлетворяющую следующему равенству:

$$A \times A^{-1} = A^{-1} \times A = E,$$

где E это единичная матрица.

Для того, чтобы у квадратной матрицы A была обратная матрица необходимо и достаточно чтобы определитель $|A|$ был не равен нулю. Введем понятие **союзной матрицы**. Союзная матрица A строится на базе исходной A путем замены всех элементов матрицы A на их алгебраические дополнения.

Исходная матрица:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Союзная ей матрица A :

$$A^* = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

Транспонируя матрицу A , мы получим так называемую присоединенную матрицу A^T :

$$A^{*T} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

Теперь, зная как вычислять определитель и присоединенную матрицу, мы можем определить матрицу A^{-1} , обратную матрице A :

$$A^{-1} = \frac{1}{\det(A)} \times A^{*T}.$$

17. Каковы свойства обратной матрицы?

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица:

$$(A^{-1})^{-1} = A.$$

Свойство 2. Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы:

$$(A^T)^{-1} = (A^{-1})^T.$$

Свойство 3. Обратная матрица произведения матриц равна произведению обратных матриц:

$$(A_1 \times A_2)^{-1} = A_2^{-1} \times A_1^{-1}.$$

18. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы?

Решим задачу определения обратной матрицы на *Python*. Для получения обратной матрицы будем использовать функцию `*inv()*`:

```
>>> A = np.matrix('1 -3; 2 5')
>>> A_inv = np.linalg.inv(A)
>>> print(A_inv)
[[ 0.45454545  0.27272727]
 [-0.18181818  0.09090909]]
```

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Решение систем линейных уравнений методом крамера python (al-shell.ru)

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy

Решение систем линейных уравнений в Python (xn--80ahcjeib4ac4d.xn--p1ai)