Введение в высшую математику

Урок 7. Видеоурок “Введение в линейную алгебру”

Практическое задание к уроку 7

1 - часть

Практическое задание №5

5.1. Вектор – это частный случай матрицы 1xN и Nx1. Повторите материал для векторов, уделяя особое внимание умножению A·B.

Вычислите, по возможности не используя программирование: (5E)-1 где E – единичная матрица размера 5x5.

5.2. Вычислите определитель:

|1 2 3|
|4 0 6|
|7 8 9|

5.3.

1. Вычислите матрицу, обратную данной:
   |1 2 3|
   |4 0 6|
   |7 8 9|
2. Приведите пример матрицы 4x4, ранг которой равен 1.

5.4.
Вычислите скалярное произведение двух векторов:
(1, 5) и (2, 8)

5.5
(1, 5, 0), (2, 8, 7) и (7, 1.5, 3) Вычислите смешанное произведение трех векторов:

2 - часть

Практическое задание №6

1. Решите линейную систему:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 12 \\ 2 \\ 1 \end{bmatrix}$$

2. Найдите псевдорешение:
   x + 2y – z = 1
   3x – 4y = 7
   8x – 5y + 2z = 12
   2x – 5z = 7
   11x +4y – 7z = 15

3. Сколько решений имеет линейная система:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 12 \\ 2 \\ 1 \end{bmatrix}$$

Если ноль – то измените вектор правой части так, чтобы система стала совместной, и решите ее.

4. Вычислите LU-разложение матрицы:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 16 & 21 \\ 4 & 28 & 73 \end{bmatrix}$$

5. Найдите нормальное псевдорешение недоопределенной системы:
   x + 2y – z = 1
   8x – 5y + 2z = 12
   Для этого определите функцию Q(x,y,z), равную норме решения, и найдите ее минимум.

6. Найдите одно из псевдорешений вырожденной системы:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 2 \\ 5 \\ 11 \end{bmatrix}$$

Попробуйте также отыскать и нормальное псевдорешение.

В [1]:
```
import numpy as np
```

1 - часть

Задание 5.1

Вектор – это частный случай матрицы 1xN и Nx1. Повторите материал для векторов, уделяя особое внимание умножению A·B.

Вычислите, по возможности не используя программирование: $(5E)^{-1}$ где E – единичная матрица размера 5x5.

$$(5E)(5E)^{-1} = E$$

Из определения обратной матрицы $A \cdot A^{-1} = E$ и того что $E^{-1} = E$ и $det(5E) = 5$, следует

$$(5E) \cdot (5E)^{-1} = 5E \cdot \frac{1}{det(5E)} A' = 5E \cdot \frac{1}{5} A' = 5 \cdot \frac{1}{5} E \cdot A' = 5 \cdot \frac{1}{5} E \cdot E^{-1} = E$$

.

Следовательно матрица $A' = E$. Поэтому обратная матрица в нашем случае будет иметь вид $(5E)^{-1} = \frac{1}{5} E$

Задание 5.2

Вычислите определитель:

|1 2 3|
|4 0 6|
|7 8 9|

$$def A = 1 \cdot (0 \cdot 9 - 8 \cdot 6) - 2 \cdot (4 \cdot 9 - 7 \cdot 6) + 3 \cdot (4 \cdot 8 - 7 \cdot 0) =$$
$$= -48 - 2(36 - 42) + 3(32 - 0) = -48 + 12 + 96 = 60$$

В [2]:
```
a = np.array([[1, 2, 3], [4, 0, 6], [7, 8 ,9]])
np.linalg.det(a)
```

Out[2]:
```
59.999999999999986
```

Задание 5.3

1. Вычислите матрицу, обратную данной:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Главный определитель Δ
$\Delta = 1 * (0 * 9 - 8 * 6) - 4 * (2 * 9 - 8 * 3) + 7 * (2 * 6 - 0 * 3) = 60$

Определитель отличен от нуля, следовательно, матрица является невырожденной и для нее можно найти обратную матрицу А.

Обратная матрица будет иметь следующий вид:

$$A^{-1} = \frac{1}{60} \begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

где Aij - алгебраичеякое дополнение.

Транспонированная матрица

$$A^T = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 0 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

Найдем алгебраические дополнения матрицы A^T
$A_{1,1} = (-1)^{1+1}(0 * 9 - 6 * 8) = -48$
$A_{1,2} = (-1)^{1+2}(2 * 9 - 3 * 8) = 6$
$A_{1,3} = (-1)^{1+3}(2 * 6 - 3 * 0) = 12$

$A_{2,1} = (-1)^{2+1}(4 * 9 - 6 * 7) = 6$
$A_{2,2} = (-1)^{2+2}(1 * 9 - 3 * 7) = -12$
$A_{2,3} = (-1)^{2+3}(2 * 6 - 3 * 4) = 6$

$A_{3,1} = (-1)^{3+1}(4 * 8 - 0 * 7) = 32$
$A_{3,2} = (-1)^{3+2}(1 * 8 - 2 * 7) = 6$
$A_{3,3} = (-1)^{3+3}(1 * 0 - 2 * 4) = -8$

Обратная матрица

$$A^{-1} = \frac{1}{60} \begin{pmatrix} -48 & 6 & 12 \\ 6 & -12 & 6 \\ 32 & 6 & -8 \end{pmatrix}$$

Проверка:

В [3]:
```
A=(np.matrix([[ 1,  2,  3],
 [ 4, 0, 6],
 [7, 8, 9]]))

print('A:\n', A,'\n')

A_1 = (np.matrix(
[[ -48,  6,  12],
 [ 6, -12, 6],
 [32, 6, -8]]))/60

print('A_1:\n',A_1,'\n')
print('A*A_1:\n',A*A_1)
```

```
A:
 [[1 2 3]
 [4 0 6]
 [7 8 9]]

A_1:
 [[-0.8         0.1         0.2       ]
 [ 0.1        -0.2         0.1       ]
 [ 0.53333333  0.1        -0.13333333]]

A*A_1:
 [[ 1.00000000e+00 -2.77555756e-17  2.77555756e-17]
 [-2.22044605e-16  1.00000000e+00  5.55111512e-17]
 [-7.77156117e-16  2.77555756e-17  1.00000000e+00]]
```

В [4]:
```
A_2 = np.linalg.inv(A)
print('A_2:\n', A_2)
```

```
A_2:
 [[-0.8         0.1         0.2       ]
 [ 0.1        -0.2         0.1       ]
 [ 0.53333333  0.1        -0.13333333]]
```

2. Приведите пример матрицы 4x4, ранг которой равен 1.

В [5]:
```
A=(np.matrix([[ -1/2,  -1,  3/2, 0],
 [ 1, 2, -3, 0],
 [ 4, 8, -12, 0],
 [ -1, -2, 3, 0]]))

A
```

Out[5]:
```
matrix([[ -0.5,  -1. ,   1.5,   0. ],
        [  1. ,   2. ,  -3. ,   0. ],
        [  4. ,   8. , -12. ,   0. ],
        [ -1. ,  -2. ,   3. ,   0. ]])
```

Задание 5.4

Вычислите скалярное произведение двух векторов:
(1, 5) и (2, 8)

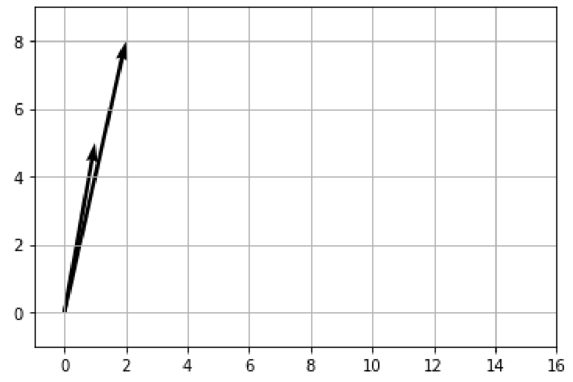
$$\begin{pmatrix} 1 & 5 \end{pmatrix} \begin{pmatrix} 2 \\ 8 \end{pmatrix} = 1 \cdot 2 + 5 \cdot 8 = 42$$

В [2]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

В [7]:
```
i = np.array([1, 5], float)
j = np.array([2, 8], float)
X, Y = np.array([0, 0]), np.array([0, 0])
U, V = np.array([i[0], j[0]]), np.array([i[1], j[1]])

plt.quiver(X, Y, U, V, angles='xy', scale_units='xy', scale=1)

plt.xlim(-1, 16)
plt.ylim(-1, 9)
plt.grid(True)
plt.show()
np.inner(i, j)
```



Out[7]: 42.0

Задание 5.5

(1, 5, 0), (2, 8, 7) и (7, 1.5, 3) Вычислите смешанное произведение трех векторов:

$a = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix}$

$b = \begin{pmatrix} 2 & 8 & 7 \end{pmatrix}$

$c = \begin{pmatrix} 7 & 1.5 & 3 \end{pmatrix}$

$$\vec{a} \times \vec{b} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

$\vec{a} \times \vec{b} = -\Sigma_{j,k=1}^{3} e_{ijk} \cdot a_j \cdot b_k$, где e_{ijk} - символ Леви-Чивиты

$\vec{a} \times \vec{b} = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x) = (5 \cdot 7 - 0 \cdot 2; 0 \cdot 2 - 1 \cdot 7; 1 \cdot 8 - 5 \cdot 2) = (35, -7, -2)$

$(\vec{a} \times \vec{b}) \cdot c = 35 \cdot 7 + (-7 \cdot 1.5) + (-2 \cdot 3) = 228.5$

В [8]:
```
35*7-(7*1.5)-(2*3)
```

Out[8]:
```
228.5
```

$$a * b = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 8 \\ 7 \end{pmatrix} = 1 \cdot 2 + 5 \cdot 8 + 0 \cdot 7 = 42$$

$$a * b * c = 42 * \begin{pmatrix} 7 & 1.5 & 3 \end{pmatrix} = \begin{pmatrix} 494 & 10.5 & 126 \end{pmatrix}$$

Смешанное произведение

$$(\vec{a} \times \vec{b}) \cdot c = a \cdot (\vec{b} \times \vec{c})$$

В [9]:
```
a = np.array([1, 5, 0], float)
b = np.array([2, 8, 7], float)
c = np.array([7, 1.5, 3], float)
v = np.cross(a, b)
print(v)
print (np.inner(v, c))
w = np.cross(b, c)
print (np.inner(w, a))
```

```
[35. -7. -2.]
228.5
228.5
```

2 - часть

Практическое задание №6

Задание 1

Решите линейную систему:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 12 \\ 2 \\ 1 \end{bmatrix}$$

В [3]:
```
A = np.array([[1, 2, 3], [4, 0, 6], [7, 8, 9]])
B = np.array([12, 2, 1])
np.linalg.solve(A, B)
```

Out[3]:
```
array([-9.2       ,  0.9       ,  6.46666667])
```

В [5]:
```
# Решение СЛАУ через обратную матрицу
A1 = np.linalg.inv(A)
print(A1)
print("del =", np.linalg.det(A))
print('Решение:')
np.dot(A1, B)
```

```
[[-0.8         0.1         0.2       ]
 [ 0.1        -0.2         0.1       ]
 [ 0.53333333  0.1        -0.13333333]]
del = 59.999999999999986
Решение:
```

Out[5]:
```
array([-9.2       ,  0.9       ,  6.46666667])
```

Задание 2

Найдите псевдорешение:
x + 2y – z = 1
3x – 4y = 7
8x – 5y + 2z = 12
2x – 5z = 7
11x +4y – 7z = 15

В [16]:
```
import numpy as np
import matplotlib.pyplot as plt

# переопределенное СЛАУ
A = np.array([[1, 2, -1], [3, -4, 0], [8, -5, 2], [2, 0, -5], [11, 4, -7]], float)
B = np.array([1, 7, 12, 7, 15])

# Решение минимизирующее длину вектора невязки методом наименьших квадратов
np.linalg.lstsq(A, B)


# (array([ 1.13919353, -0.90498444, -0.9009803 ])  - псевдорешение
# array([0.71523211]) - значение функции невязки (сумма квадратов невязок каждого уравнения)
# 3 - ранг матрицы A
# array([15.2817306 ,  9.59852942,  3.65197794])) - дополнительные параметры
```

```
<ipython-input-16-f61ac1e827eb>:9: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  np.linalg.lstsq(A, B)
```

Out[16]:
```
(array([ 1.13919353, -0.90498444, -0.9009803 ]),
 array([0.71523211]),
 3,
 array([15.2817306 ,  9.59852942,  3.65197794]))
```

В [12]:
```
# Проверка
np.dot(A, [1.13919353, -0.90498444, -0.9009803])
```

Out[12]:
```
array([ 0.23020495,  7.03751835, 11.83650984,  6.78328856, 15.21805317])
```

В [14]:
```
# Подсчитаем невязку каждого уравнения
np.dot(A, [ 1.13919353, -0.90498444, -0.9009803 ]) - B
```

Out[14]:
```
array([-0.76979505,  0.03751835, -0.16349016, -0.21671144,  0.21805317])
```

В [17]:
```
# Определим функцию нормы невязки
def Q(x,y,z):
    return ((np.linalg.norm(np.dot(A, [x, y, z]) - B))**2)

Q(1.13919353, -0.90498444, -0.9009803)

# Значение совпадает с результатом применения численного метода
# 0.7152321111819737
```

Out[17]:
```
0.7152321111819737
```

В [7]:
```
import numpy as np
from numpy import linalg as LA
print(A)

# Функция linalg.pinv() вычисляет псевдообратную матрицу (Мура-Пенроуза).
a_pinv = LA.pinv(A)
print(a_pinv)
```

```
[[ 1.  2. -1.]
 [ 3. -4.  0.]
 [ 8. -5.  2.]
 [ 2.  0. -5.]
 [11.  4. -7.]]
[[ 0.0184525  -0.01309971  0.06358533 -0.05827435  0.0571557 ]
 [ 0.05588372 -0.11181622 -0.03160448 -0.12284939  0.07073632]
 [ 0.03245479 -0.06855478  0.06726537 -0.17860416 -0.00070046]]
```

Псевдорешение $X' = A' b$

В [8]:
```
X_1=np.dot(a_pinv, B)
print(f'Псевдорешение X_1:\n{X_1}')
```

```
Псевдорешение X_1:
[ 1.13919353 -0.90498444 -0.9009803 ]
```

Задание 3

Сколько решений имеет линейная система:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 12 \\ 2 \\ 1 \end{bmatrix}$$

Если ноль – то измените вектор правой части так, чтобы система стала совместной, и решите ее.

В [23]:
```
import numpy as np
import matplotlib.pyplot as plt

# print(np.__version__)

A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
B = np.array([[12, 2, 1]])
C = np.concatenate((A,B.T), axis=1)

print (A)
print (C)
np.linalg.matrix_rank(A, 0.0001), np.linalg.matrix_rank(C, 0.0001)
np.linalg.matrix_rank(A), np.linalg.matrix_rank(C)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[ 1  2  3 12]
 [ 4  5  6  2]
 [ 7  8  9  1]]
```

Out[23]:
```
(2, 3)
```

Теорема Кронекера-Каппели утверждает, что в этом случае **решений нет** т. к. $rank A \neq rank C$

В [35]:
```
# Подбираем вектор B
B = np.array([[1, 1, 1]])

C = np.concatenate((A,B.T), axis=1)
print (C)
print(A)
np.linalg.matrix_rank(A, 0.0001), np.linalg.matrix_rank(C, 0.0001)
# np.linalg.matrix_rank(A), np.linalg.matrix_rank(C)
```

```
[[1 2 3 1]
 [4 5 6 1]
 [7 8 9 1]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Out[35]:
```
(2, 2)
```

Ранг основной матрицы равен рангу расширенной матрицы (2, 2) и меньше числа неизвестных.
Следовательно система имеет бесконечно много решений.

В [34]:
```
# np.linalg.solve(A, B)
```

В [26]:
```
print(A)
print("del =", np.linalg.det(A))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
del = -9.51619735392994e-16
```

Задание 4

Вычислите LU-разложение матрицы:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 16 & 21 \\ 4 & 28 & 73 \end{bmatrix}$$

LU-разложение определяет алгоритм Гауса решения линейных систем

В [15]:
```
import scipy
import scipy.linalg

A = np.array([ [1, 2, 3], [2, 16, 21], [4, 28, 73]])
P, L, U = scipy.linalg.lu(A)

print(P)
print(L)
print(U)

print(np.dot(P.transpose(), A) - np.dot(L, U))
```

```
[[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
[[ 1.    0.    0.  ]
 [ 0.25  1.    0.  ]
 [ 0.5  -0.4   1.  ]]
[[  4.    28.    73.  ]
 [  0.    -5.   -15.25]
 [  0.     0.   -21.6 ]]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

В [ ]:

Задание 5

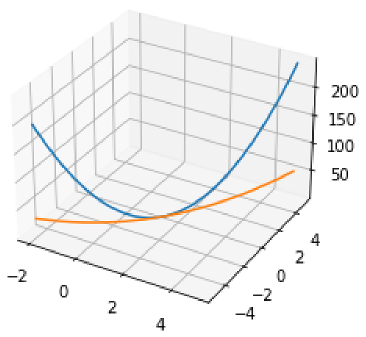
Найдите нормальное псевдорешение недоопределенной системы:
$x + 2y – z = 1$
$8x – 5y + 2z = 12$
Для этого определите функцию Q(x,y,z), равную норме решения, и найдите ее минимум.

В [ ]:
```
A = np.array([[1, 2, -1], [8, -5, 2]])
B = np.array([1, 12])
```

В [61]:
```
def Q(x, y, z):
    return (x**2 + y**2 + z**2)
```

В [68]:
```
# x = np.linspace(-np.pi, np.pi, 50)
x = np.linspace(-2, 5, 201)
y = np.linspace(-5, 5, 201)
z = x + 2*y -1
z1 = 6 + (5/2)*y - 4*x
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, Q(x, y, z), label='1')
ax.plot(x, y, Q(x, y, z1), label='2')
```

Out[68]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x75e050f220>]
```



Решение около точки (1.3, 0, 0)

В [69]:
```
# Найдем нормальное псевдорешене методом библиотеки nunpy
A = np.array([[1, 2, -1], [8, -5, 2]])
B = np.array([1, 12])

np.linalg.lstsq(A, B)
```

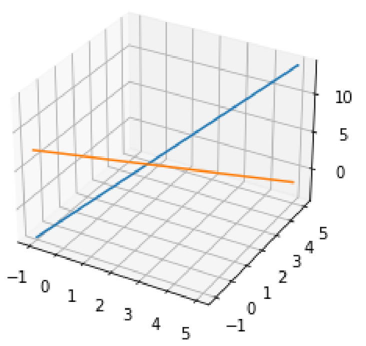
```
<ipython-input-69-279d39a06aaa>:5: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  np.linalg.lstsq(A, B)
```

Out[69]:
```
(array([ 1.38191882, -0.18081181,  0.0202952 ]),
 array([], dtype=float64),
 2,
 array([9.65316119, 2.41173777]))
```

В [ ]:
```
Решение [ 1.38191882, -0.18081181,  0.0202952 ] близко к найденному нами решению (1.3, 0, 0)
```

В [70]:
```
# x = np.linspace(-np.pi, np.pi, 50)
x = np.linspace(-1, 5, 201)
y = np.linspace(-1, 5, 201)
z = x + 2*y -1
z1 = 6 + (5/2)*y - 4*x
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='1')
ax.plot(x, y, z1, label='2')
```

Out[70]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x75e057c340>]
```



В [ ]:

Задание 6

Найдите одно из псевдорешений вырожденной системы:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * X = \begin{bmatrix} 2 \\ 5 \\ 11 \end{bmatrix}$$

Попробуйте также отыскать и нормальное псевдорешение.

В [72]:
```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]], float)
B = np.array([2, 5, 11])
# Найдём QR разложение матрицы A
# Q-ортогональная матрица, R - верхняя треугольная матрица
Q, R = np.linalg.qr(A)

print(A)
print(Q)
print(R)
```

```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
[[-0.12309149  0.90453403  0.40824829]
 [-0.49236596  0.30151134 -0.81649658]
 [-0.86164044 -0.30151134  0.40824829]]
[[-8.12403840e+00 -9.60113630e+00 -1.10782342e+01]
 [ 0.00000000e+00  9.04534034e-01  1.80906807e+00]
 [ 0.00000000e+00  0.00000000e+00 -1.11164740e-15]]
```

В [73]:
```
print(np.dot(Q, R))
print(np.dot(np.transpose(Q), Q))
```

```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
[[ 1.00000000e+00 -5.26517217e-16 -2.55176183e-16]
 [-5.26517217e-16  1.00000000e+00  3.37757775e-16]
 [-2.55176183e-16  3.37757775e-16  1.00000000e+00]]
```

В [74]:
```
R1 = R[:2, :2]
R1
```

Out[74]:
```
array([[-8.1240384 , -9.6011363 ],
       [ 0.        ,  0.90453403]])
```

В [76]:
```
B1 = np.dot(np.transpose(Q), B)[:2]
B1
```

Out[76]:
```
array([-1.21860576e+01,  8.54871729e-15])
```

В [77]:
```
X1 = np.linalg.solve(R1, B1)
X1
```

Out[77]:
```
array([1.50000000e+00, 9.45096256e-15])
```

В [80]:
```
# Получим псевдорешение для исходной системы
X = np.append(X1, 0)
print(X)
np.linalg.norm(X)
```

```
[1.50000000e+00 9.45096256e-15 0.00000000e+00]
```

Out[80]:
```
1.4999999999999893
```

В [79]:
```
np.linalg.norm(np.dot(A, X) - B)
```

Out[79]:
```
1.2247448713915885
```

В [ ]: