# Лабораторный практикум 2. Определители II и III порядков. Формулы Крамера. Обратная матрица.

## Определители II и III порядков

Загрузите необходимые библиотеки:

import numpy as np

from sympy import \*

Так как матрица – прямоугольная таблица чисел, то задание матрицы в Python будет предполагать построение соответствующего двумерного массива. Для создания такого массива из стандартных средств Python наиболее подходящими являются тип ***array*** (произвольные многомерные массивы) или ***matrix*** (двухмерные квадратные массивы) из библиотеки Numpy или ***Matrix*** из библиотеки Sympy.

### Пример 1. Введем квадратную матрицу второго порядка:

arr = np.array([[1, 2], [3, 4]])

print(arr)

или

arr = np.matrix([[1, 2], [3, 4]])

print(arr)

Так же в библиотеке Numpy задать матрицу можно следующим способом:

arr = np.matrix(‘1 2 3; 4 5 6; 7 8 9’)

print(arr)

В библиотеке Sympy:

a=Matrix([[1,2],[3,4]])

a

***Вычисление определителя***

1. Можно вычислить определитель матрицы *A*, обращаясь к индексам элементов массива *A.* Отсчёт индексов в Python начинается с нуля:

arr = np.matrix(‘1 2; 3 4’)

detA = arr[0,0] \* arr[1, 1] – arr[0, 1] \* arr[1, 0]

2. Можно вычислить определитель матрицы *A* с помощью метода ***det*** из модуля библиотеки Numpy ***linalg***:

detA = np.linalg.det(arr)

print(detA)

В библиотеке Sympy:

det(a)

### 

### Упражнение 2.1. Вычислить определители второго порядка, обращаясь через индексы к элементам массива, и сделать проверку с помощью функции *det*. Буквенные выражения необходимо задать в символьном виде:

a,b=symbols('a b')

v=Matrix([…])

det(v)

1. , 2. , 3..

**Упражнение 2.2.** Вычислить определители третьего порядка с помощью функции ***det***:

, 2. , 3. , 4. .

## Приложение определителя n-го порядка к решению систем линейных уравнений по формулам Крамера

Пусть задана система *n* линейных уравнений с *n* неизвестными вида:

или в матричной форме , где

Квадратная матрица называется *вырожденной*, если ее определитель равен нулю, и *невырожденной* в противном случае. Если , то система имеет единственное решение:

,

где - определитель, получаемый из определителя , заменой -го столбца на столбец свободных членов *B*.

**Пример 2.** Решить систему уравнений:

A = np.matrix(‘3 -1 2; 1 4 -1; 2 3 1’)

print(A)

B = np.matrix(‘-4; 10; 8’)

print(B)

A\_det = np.linalg.det(A)

print(A\_det)

X\_m = np.matrix(A)

X\_m[:, 0] = B

print(X\_m)

Y\_m = np.matrix(A)

Y\_m[:, 1] = B

print(Y\_m)

Z\_m = np.matrix(A)

Z\_m[:, 2] = B

print(Z\_m)

x = np.linalg.det(X\_m) / A\_det

y = np.linalg.det(Y\_m) / A\_det

z = np.linalg.det(Z\_m) / A\_det

print(x)

print(y)

print(z)

**Упражнение 2.3.** Решить системы уравнений по формулам Крамера**.** Сделать проверку, используя равенство: .

Для перемножения массивов типа matrix используйте метод ***dot*** или **\*** (внимание, \* используется для перемножения матрицы размера на матрицу размера ), типа array – ***dot*** или **@**.

2. 3. 4.

**Упражнение 2.4.** Решить системы уравнений упражнения 2.3(1) и 2.3(3) в библиотеке Sympy**.** Сделать проверку (метод \*).

## Обратная матрица

Если *A* – невырожденная матрица, то существует и притом единственная матрица такая, что

,

где – единичная матрица. Матрица называется ***обратной*** к матрице *A*. В Numpy для получения обратной матрицы используется метод ***inv***()модуля ***linalg***:

A\_inv = np.linalg.inv(A)

или операция возведения в степень (–1):

A\_inv =A\*\*-1

В Sympy соответственно: ***A.inv()***.

Обратную матрицу удобно использовать при решении систем линейных уравнений (при условии ). Решение матричного уравнения с неизвестной матрицей *Х* ищется в виде:

.

**Пример 3.** Решить систему уравнений

A = np.matrix(‘3 -1 2; 1 4 -1; 2 3 1’)

B = np.matrix(‘-4; 10; 8’)

print(A)

print(B)

A\_inv = np.linalg.inv(A)

print(A\_inv)

X = A\_inv.dot(B)

X

**Упражнение 2.5.** Решить системы уравнений из упражнений 2.3, используя обратную матрицу.