# Лабораторный практикум 11. Методы решения систем линейных алгебраических уравнений (СЛАУ)

Загрузите необходимые библиотеки:

>> import numpy as np

>> from sympy import \*

## Совместность системы линейных алгебраических уравнений

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

или в матричной форме , где

Исследовать СЛАУ на совместность означает выяснить, есть у этой системы решения, или же их нет. Если решения есть, то указать сколько их. По теореме Кронекера-Капелли СЛАУ совместна тогда и только тогда, когда ранг матрицы системы равен рангу ее расширенной матрицы, т.е. . Ответ на вопрос о количестве этих решений даёт следствие из теоремы Кронекера-Капелли:

1. Если , то СЛАУ несовместна (не имеет решений).
2. Если, то СЛАУ является неопределённой (имеет бесконечное количество решений).
3. Если , то СЛАУ является определённой (имеет одно решение).

## ****Метод обратной матрицы. Метод Крамера. Метод solve****

Напомним, что **метод** обратной матрицы и метод Крамера применяют к решению систем уравнений, у которых (см.практикум 2).

Библиотека Numpy содержит метод ***linalg.solve*()**, который используется для решения СЛАУ, у которых . В библиотеке Sympy используется метод **.*linsolve*()**.

**Пример 1. Исследовать на совместность и** решить систему линейных уравнений методом обратной матрицы, методом Крамера и методом solve. Сделать проверку.

#Метод обратной матрицы

A = np.array([[4, 3, 2], [-2, 2, 3], [3, -5, 2]])

B = np.array([25, -10, -4])

#Исследуем систему на совместность

D = np.column\_stack((A,B)) # добавляем к матрице А столбец В

n=3

rkA=np.linalg.matrix\_rank(A)

rkD=np.linalg.matrix\_rank(D)

print('Проверка совместности:',[rkA,rkD,n])

X1=np.linalg.inv(A).dot(B)

print('Ответ:',X1)

Проверка совместности: [3, 3, 3]

Ответ: [ 5. 3. -2.]

# Метод Крамера

DA=np.linalg.det(A)

D1=A.copy()

D1[:,0]=B

D11=np.linalg.det(D1)

D2=A.copy()

D2[:,1]=B

D22=np.linalg.det(D2)

D3=A.copy()

D3[:,2]=B

D33=np.linalg.det(D3)

X2=[D11/DA,D22/DA,D33/DA]

print('Ответ:',X2)

# Метод solve

X3 = np.linalg.solve(A,B)

print(X3)

# Проверка

np.dot(A,X3)==B

[ 5. 3. -2.]

array([ True, True, True])

**Упражнение 11.1. Исследовать на совместность и** решить системы линейных уравнений любым методом в случае существования единственного решения, сделать проверку. Почему данные методы не подходят в случае бесконечного количества решений?

1. 2.

3. 4.

## Метод Гаусса (****последовательного исключения неизвестных****)

С методом Гаусса Вы уже познакомились в курсе Линейной алгебры. Данным методом можно решать не только СЛАУ, в которых число уравнений совпадает с количеством неизвестных переменных и основная матрица системы невырожденная, но и системы уравнений, в которых число уравнений не совпадает с количеством неизвестных переменных или определитель основной матрицы равен нулю.

Общее решение для системы линейных уравнений находят методом Гаусса. При этом расширенная матрица системы приводится к специальному ступенчатому виду. В библиотеке Sympy данный метод реализует метод **.*rref*()** (см.практикум 9).

**Пример 2.** Найти общее решение системы из упражнения 11.1(4): определить базисные переменные и выразить их через свободные переменные. Сделать проверку.

При исследовании на совместность было получено, что данная система совместна , а .

Применяем метод *rref* к расширенной матрице системы:

A = Matrix([[2,1,-1,-3], [4,0,1,-7], [0,2,-3,1],[2,3,-4,-2]])

RA = A.rref()

print(RA)

(Matrix([

[1, 0, 1/4, -7/4],

[0, 1, -3/2, 1/2],

[0, 0, 0, 0],

[0, 0, 0, 0]]), (0, 1))

По виду преобразованной матрицы очевидно, что базисными переменными являются и , свободными и . Свободные переменные могут принимать произвольные действительные значения: пусть и . Далее воспользуемся методом **linsolve**:

x1, x2, x3, x4 = symbols('x1 x2 C1 C2')

B = Matrix([2,3,1,3])

X=linsolve((A,B),x1,x2,x3,x4)

Print(X)

{(-C1/4 + 7\*C2/4 + 3/4, 3\*C1/2 - C2/2 + 1/2, C1, C2)}

# Проверка

X=Matrix([[-C1/4+7\*C2/4+3/4],[3\*C1/2-C2/2+1/2],[C1],[C2]])

A\*X==B

True

**Упражнение 11.2. Р**ешить систему линейных уравнений методом Гаусса. Сделать проверку.

1. 2.3.

4. 5.

## Однородные системы

Для существования нетривиального решения однородной системы необходимо и достаточно, чтобы (при это условие означает, что ).

Общее решение однородной системы имеет вид , где – произвольные постоянные, – *фундаментальная система решений*.

Метод ***nullspace*()** библиотеки Sympy находит фундаментальную систему решений однородной системы уравнений, образованную матрицей *А*.

**Пример 3.** Решить однородную систему уравнений

Найдем ранг матрицы системы:

АA = Matrix([[1,3,4,-2],[0,5,7,-4],[1,8,11,-6],[-1,2,3,-2]])

#Ранг матрицы системы

print('Ранг системы=',AA.rank())

Ранг системы= 2

Ранг меньше числа неизвестных – система имеет бесконечное множество решений.

E=AA.nullspace()

print('Фундаментальная система решений:',E)

[Matrix([

[ 1/5],

[-7/5],

[ 1],

[ 0]]),

Matrix([

[-2/5],

[ 4/5],

[ 0],

[ 1]])]

C1, C2 = symbols('C1 C2')

print('Общее решение системы:')

C1\*E[0]+C2\*E[1]

**Упражнение 11.3.** Найти общее решение и фундаментальную систему решений однородной системы уравнений. Сделать проверку.

2. 3.

## Дополнительное задание

Создать функцию, которая исследует совместность неоднородной системы линейных уравнений произвольного порядка. Если система совместна, то данная функция должна определять имеет система единственное решение или количество базисных и свободных переменных в решении. Проверить работу этой функции на примере упражнения 1.