**Исламов Радмир Лабораторная 6**

import numpy as np

import random as rd

import sympy as sp

def main():

n = 5

cond = 10\*\*3

A = np.random.random((n, n))

while np.linalg.cond(A) < cond:

A = np.random.random((n, n))

B = np.random.random((n, 1))

print("task 1")

print(solve\_inv(A, B))

print(solve\_kramer(A, B))

print("task 2")

print(solve\_gauss(\*diag(A, B)))

print("task 3")

print(solve\_rref(A, B))

main()

**Задание 1**

def solve\_inv(A, B):

"""

Решает систему линейных уравнений методом обратной матрицы.

"""

if np.linalg.det(A):

return np.dot(np.linalg.inv(A), B)

def solve\_kramer(A, B):

"""

Решает систему линейных уравнений методом Крамера.

"""

if not np.linalg.det(A):

return None

X = []

dA = np.linalg.det(A)

for i in range(0, A.shape[1]):

buf = np.copy(A)

buf[:, i][:, np.newaxis] = B

X += [np.linalg.det(buf)/dA]

return X

**Вывод:**

task 1

[[-19.83395251]

[ 14.1546572 ]

[ 16.11404312]

[-10.76301409]

[ 5.3491825 ]]

[-19.833952510624904, 14.154657201708892, 16.114043122190655, -10.763014092863102, 5.34918249505438]

**Задание 2**

def diag(A, B):

"""

Приводит матрицу A к диагональному виду.

"""

n = A.shape[0]

A = np.array(A)

B = np.array(B)

for i in range(n):

col = list(map(abs, A[:, i]))[i:n]

row\_A = np.array(A[i, :])

row\_B = np.array(B[i, 0])

main\_index = col.index(max(col)) + i

main\_el = A[main\_index, i]

A[i, :], A[main\_index, :] = A[main\_index, :], row\_A

B[i, 0], B[main\_index, 0] = B[main\_index, 0], row\_B

for j in range(i, n):

A[i, j] /= main\_el

B[i, 0] /= main\_el

row = np.array(A[i, :])

for j in range(i + 1, n):

k = A[j, i]

A[j, :] -= row\*k

B[j, 0] -= B[i, 0]\*k

return A, B

def solve\_gauss(A, B):

"""

Решает систему линейных уравнений методом Гаусса.

"""

n = A.shape[1]

X = np.zeros((n, 1))

X[n - 1, 0] = B[n - 1, 0]

for i in range(n - 2, -1, -1):

c = 0

for j in range(n - 1, i, -1):

c += A[i, j] \* X[j, 0]

X[i, 0] = B[i, 0] - c

return X

**Вывод:**

task 2

[[-19.83395251]

[ 14.1546572 ]

[ 16.11404312]

[-10.76301409]

[ 5.3491825 ]]

**Задание 3**

def solve\_rref(A, B):

"""

Решает систему линейных уравнений методом приведения матрицы к ступенчатому виду.

"""

n = A.shape[0]

A = np.array(A).T

B = np.array(B).T

Q = np.vstack((A, B)).T

Q = sp.Matrix(Q).rref()[0]

return np.array(sp.Matrix(Q[:, n]))

**Вывод:**

task 3

[[-19.8339525106255]

[14.1546572017093]

[16.1140431221911]

[-10.7630140928634]

[5.34918249505453]]

**Контрольные вопросы**

Что такое точные методы решения СЛАУ? Что такое итерационные? Куда отнести методы Гаусса, прогонки, Зейделя, Якоби?

**Ответы:**

Точные методы решения систем линейных алгебраических уравнений (СЛАУ) позволяют найти точное решение за конечное число арифметических операций при отсутствии округлений. К ним относятся методы Гаусса, прогонки, метод квадратного корня и метод отражений.

Итерационные методы, напротив, находят приближенное решение, которое уточняется на каждой итерации. К ним относятся методы Якоби, Зейделя, релаксации и простой итерации.

Методы Гаусса, прогонки, Зейделя и Якоби относятся к точным методам, так как они позволяют найти точное решение за конечное число операций без округлений.