**Код программы:**

import copy

from prettytable import PrettyTable as PT

def printMatrix(array):

for line in array:

print(line)

print()

def find(array, value):

for line in range(len(array)):

for index in range(len(array[line])):

if array[line][index] == value:

return line, index

return -1, -1

def to\_num(matrix):

res = 0

new\_vec = []

i = -1

for stolbec in matrix:

new\_vec.append(0)

i += 1

for num in stolbec:

new\_vec[i] += num\*\*2

res += new\_vec[i]\*\*2

return res

def maxValue(value1, value2):

if abs(value1) > abs(value2): return value1

return value2

class Gauss:

matrix = []

tmpMatrix = []

roundMatrix = []

answer = []

tmpAnswer = [] # столбец который меняется при методе гауса

errAnswer = [] # столбец Б полученный при внесении ошибки в матрицу и исходный столбец Б

tmpResult = [] # столбец решения, который мы и вычисляем

dim = 0

eps = 6

correctResult = []

def \_\_init\_\_(self, matrix, answer):

self.matrix = matrix

self.answer = answer

self.dim = len(matrix)

self.diagonal()

self.getResult(redefine=True)

def diagonal(self, eps=10, mustRoundMatrix=False, mustRoundAnswer=False):

self.eps = eps

self.tmpMatrix = []

self.tmpAnswer = []

self.roundMatrix = []

for j in range(self.dim):

self.tmpMatrix.append([])

if (mustRoundAnswer):

self.tmpAnswer.append(round(self.answer[j], eps))

else:

self.tmpAnswer.append(self.answer[j])

for i in range(self.dim):

# обрезание всех чисел после eps цифры после запятой (без окргуления в большую/меньшую сторону)

if (mustRoundMatrix):

self.tmpMatrix[j].append(round(self.matrix[j][i] - self.matrix[j][i] % (10 \*\* (-eps)), eps))

else:

self.tmpMatrix[j].append(self.matrix[j][i])

self.roundMatrix = copy.deepcopy(self.tmpMatrix)

for j in range(self.dim):

# блок кода до следующего фора можно пропустить, но значения могут получится менее точные

# прям совсем немного точнее

maxA = 0

for i in range(j, self.dim):

maxA = maxValue(maxA, self.tmpMatrix[i][j]) # ищем максимальный элемент столбца j

x, y = find(self.tmpMatrix, maxA) # x - строчка y - столбец

if (x != j): # не надо менять строчку с собой же

self.tmpMatrix[x], self.tmpMatrix[j] = self.tmpMatrix[j], self.tmpMatrix[

x] # swap строчек, чтобы строчка с макс эл-ом была выше

self.tmpAnswer[x], self.tmpAnswer[j] = self.tmpAnswer[j], self.tmpAnswer[x]

# тот самый блок улучшения значений закончен

for line in range(j + 1, self.dim): # + 1 т.к. из себя вычитать не надо

if (self.tmpMatrix[j][j] != 0):

k = self.tmpMatrix[line][j] / self.tmpMatrix[j][j]

else:

k = 0

self.tmpAnswer[line] -= k \* self.tmpAnswer[j]

for index in range(j, self.dim):

self.tmpMatrix[line][index] -= k \* self.tmpMatrix[j][index]

# можно вообще присваивать 0, нам k для В только нужно (следующая строка необязательна)

if (abs(self.tmpMatrix[line][index]) < 1 \* 10 \*\* (-15)): self.tmpMatrix[line][index] = 0.0

# собственно я это и сделал, чтобы было красивее))

# обратный ход

for j in range(self.dim - 1, 0, -1):

for i in range(0, j):

k = self.tmpMatrix[i][j] / self.tmpMatrix[j][j]

self.tmpAnswer[i] -= k \* self.tmpAnswer[j]

self.tmpMatrix[i][j] -= k \* self.tmpMatrix[j][j]

# можно вообще присваивать 0, нам k для В только нужно (следующая строка необязательна)

if (abs(self.tmpMatrix[i][j]) < 1 \* 10 \*\* (-15)): self.tmpMatrix[i][j] = 0.0

# собственно я это и сделал, чтобы было красивее))

def getResult(self, redefine=False):

result = [0 if self.tmpMatrix[x][x] == 0 else self.tmpAnswer[x] / self.tmpMatrix[x][x] for x in range(self.dim)]

if (redefine): self.correctResult = result

answer = []

for j in range(self.dim):

sum = 0

for i in range(self.dim):

sum += self.roundMatrix[j][i] \* result[i]

answer.append(sum)

# self.errAnswer = [round(i, self.eps) for i in answer]

self.errAnswer = answer

self.tmpResult = result

return result, answer

def absErrorSolution(self):

sum = 0

errResult = self.getResult()[0]

for i in range(self.dim):

sum += (self.correctResult[i] - errResult[i]) \*\* 2

return sum \*\* (1 / 2)

def otnErrorSolution(self):

errResult = self.getResult()[0]

summ = 0

for i in range(self.dim):

summ += (errResult[i]) \*\* 2

return self.absErrorSolution() / (summ \*\* (1 / 2))

def absErrorB(self):

sum = 0

errAnswer = self.getResult()[1]

for i in range(self.dim):

sum += (self.answer[i] - errAnswer[i]) \*\* 2

return sum \*\* (1 / 2)

def otnErrorB(self):

errAnswer = self.getResult()[1]

summ = 0

for i in range(self.dim):

summ += (errAnswer[i]) \*\* 2

return self.absErrorB() / (summ \*\* (1 / 2))

def absErrorA(self):

matrix\_to\_vector = []

for i in range(len(self.matrix)):

matrix\_to\_vector.append([])

matrix\_to\_vector[i] = list(map(lambda x, y: x - y, self.matrix[i], self.roundMatrix[i]))

res = to\_num(matrix\_to\_vector)

print("абсолютная погрешность матрицы: ", res)

print("Верхняя граница: ", 7.370494\*res)

# округленная матрица (или нет, если поставлен флаг), округленное решение и корни с таким решением

def getViewResult(self):

# TODO добавить исходный стоблец Б для наглядности, заменить Б на невязку

tabel = PT()

column = ["{}".format(i + 1) for i in range(self.dim)]

column.extend(["Решение", "Б", "Исходное значение Б"])

tabel.field\_names = column

for j in range(self.dim):

row = self.roundMatrix[j]

row.extend([self.tmpResult[j], self.errAnswer[j], self.answer[j]])

tabel.add\_row(row)

print(tabel)

with open('Matrix{}.csv'.format(self.eps), 'w', newline='') as f\_output:

f\_output.write(tabel.get\_csv\_string())

def main():

matrix = [[-4.571747, 3.039542, -2.503332, 1.382615, 4.341474],

[0.6473, -1.66098, 3.06268, -5.538428, -3.673399],

[-0.332372, 3.236347, -2.652089, -3.43368, -3.945653],

[2.838451, -0.236414, 4.255314, 0.146155, -0.557685],

[-4.598564, -4.12698, -0.174609, 3.04406, -3.700862]]

answer = [0.49259, -2.904252, 4.70038, -1.576158, 3.835539]

gauss = Gauss(matrix, answer)

for error in range(7):

# округляются и матрица и столбец б

gauss.diagonal(eps=error, mustRoundMatrix=True, mustRoundAnswer=True)

otnErrorSolution = gauss.otnErrorSolution()

absErrorSolution = gauss.absErrorSolution()

otnErrorB = gauss.otnErrorB()

absErrorB = gauss.absErrorB()

obsl = 7.370494

granitsa = absErrorB \* obsl

print("Верхняя граница: {}\n"

"Количество цифр после запятой: {}\n"

"Относительная погрешность: {}\n"

"Абсолютная погрешность: {}\n"

"Относительная погрешность Б: {}\n"

"Абсолютная погрешность Б: {}".format(granitsa, error, otnErrorSolution, absErrorSolution, otnErrorB, absErrorB))

gauss.getViewResult()

gauss.absErrorA()

print()

return 0

if \_\_name\_\_ == '\_\_main\_\_':

main()