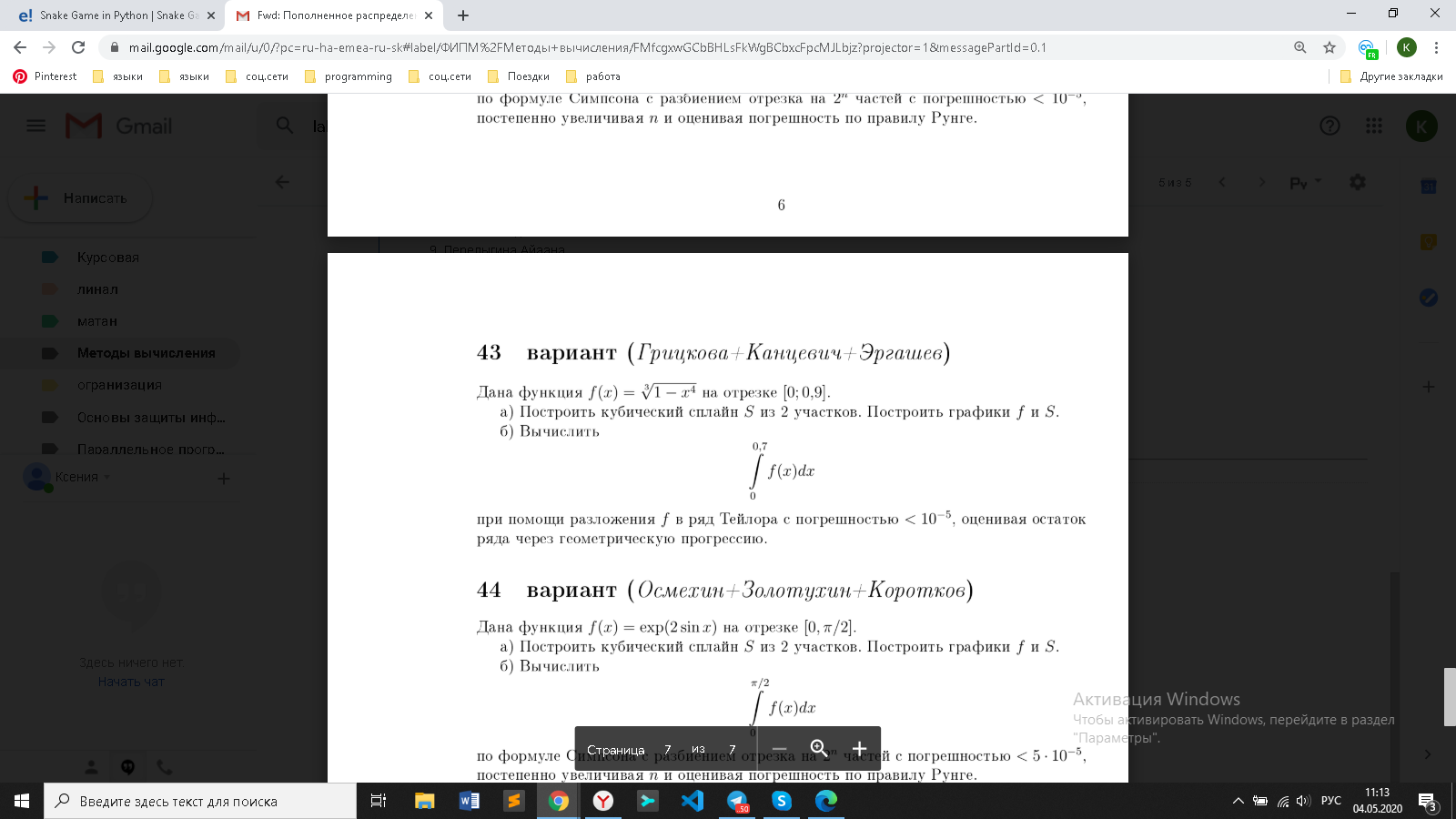
# Задание.



# Файл dz4.py

from findRoots import findRoots

from utils import readMatrixFromFile

import matplotlib.pyplot as plt

import numpy as np

from math import exp, sin, pi

matrix = readMatrixFromFile('input.txt')

# Выделяем столбец свободных членов

col = [row[len(matrix)] for row in matrix]

# Удаляем столбец свободных членов из матрицы

for i in range(len(matrix)):

    matrix[i] = matrix[i][:-1]

coff = findRoots(matrix, col)

coff\_1 = coff[:4]

coff\_2 = coff[4:]

def f(x): # Задаем функцию

    return pow(1 - x\*\*4, 0.333333)

def S\_1(x): # Первый сплайн

    return coff\_1[0] \* x\*\*3 \

        + coff\_1[1] \* x\*\*2 \

        + coff\_1[2] \* x \

        + coff\_1[3]

def S\_2(x): # Второй сплайн

    return coff\_2[0] \* x\*\*3 \

        + coff\_2[1] \* x\*\*2 \

        + coff\_2[2] \* x \

        + coff\_2[3]

x = np.linspace(-0.00001, 0.9)

# создаём область, в которой будет отображаться график

f\_y = [f(x) for x in x]

S\_y = [S\_1(x) if x < 0.45 else S\_2(x) for x in x]

plt.plot(x, f\_y, label='f')

plt.plot(x, S\_y, label = 'S')

plt.legend()

plt.show()

# Файл findRoots.py

Данный файл реализует метод Гауса (из 1 лр), с помощью которого находим коэффициеты при помощи решения матрицы.

def findMaxInColumn(column, matrix):

    max = abs(matrix[column][column])

    max\_row = column

    for row in range(column, len(matrix)):

        if (abs(matrix[row][column]) > max):

            max = abs(matrix[row][column])

            max\_row = row

    return max\_row

def swapLines(x, y, matrix):

    temp = matrix[x]; matrix[x] = matrix[y]; matrix[y] = temp

    return matrix

def mult(num, row):

    r = list()

    for item in row:

        r.append(item \* num)

    return r

def sum(row1, row2):

    result = list()

    for idx, item in enumerate(row1):

        result.append(row1[idx] + row2[idx])

    return result

def forwardPropagation(matrix, col):

    for column in range(0, len(matrix)):

        max\_row = findMaxInColumn(column, matrix)

        matrix = swapLines(column, max\_row, matrix)

        col = swapLines(column, max\_row, col)

        col[column] = col[column] / matrix[column][column]

        matrix[column] = mult(1/matrix[column][column], matrix[column])

        for num in range(column + 1, len(matrix)):

            col[num] = col[num] + col[column] \* -matrix[num][column]

            matrix[num] = sum(matrix[num], mult(-matrix[num][column], matrix[column]))

def backwardPropagation(matrix, col):

    for i in range(len(matrix)-1, 0, -1):

        for j in range(i-1, -1, -1):

            col[j] = col[j] + col[i]\*-matrix[j][i]

            matrix[j] = sum(matrix[j], mult(-matrix[j][i], matrix[i]))

def findRoots(matrix, col):

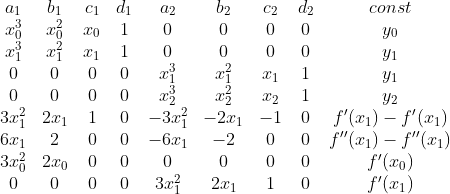
    forwardPropagation(matrix, col)

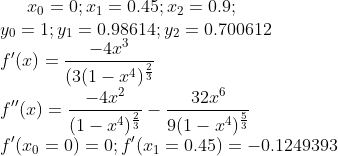
    backwardPropagation(matrix, col)

    return col

# Файл input.txt

Матрица, которую строим для нахождения коэффициентов:

**



0 0 0 1 0 0 0 0 1

0.091125 0.2025 0.45 1 0 0 0 0 0.98614

0 0 0 0 0.091125 0.2025 0.45 1 0.98614

0 0 0 0 0.729 0.81 0.9 1 0.700612

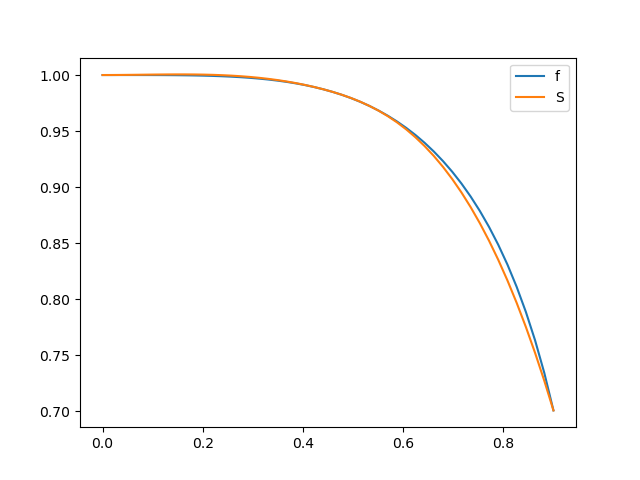
0.6075 0.9 1 0 -0.6075 -0.9 -1 0 0

2.7 2 0 0 -2.7 -2 0 0 0

0 0 1 0 0 0 0 0 0

0 0 0 0 0.6075 0.9 1 0 -0.1249393

# Результат работы программы



# Вычисление интеграла и оценка погрешности

