Лабораторная работа №7  
Гайворонская Екатерина Александровна   
010304 КМСб-о23

ЛАБОРАТОРНАЯ РАБОТА 7.

Применение инструментария Питон

Для решения вычислительных задач и обработки данных

Задачи:

1. Научиться разрабатывать подпрограммы на языке Python для решения некоторых задач вычислительной математики.

Вариант p = 0 q = 1

**Порядок выполнения работы**

**Задача1.** Разработайте программу для решения СЛАУ методом Крамера.

Порядок решения:

1. Теорема Крамера. Если определитель системы не равен нулю, то рассматриваемая система имеет одно и только одно решение, причем

Программа для решения задачи:

def det(matrix):

    return matrix[0][0] \* matrix[1][1] \* matrix[2][2] + matrix[0][1] \* matrix[1][2] \* matrix[2][0] + matrix[0][2] \* matrix[1][0] \* matrix[2][1] - matrix[0][2] \* matrix[1][1] \* matrix[2][0] - matrix[0][1] \* matrix[1][0] \* matrix[2][2] - matrix[0][0] \* matrix[1][2] \* matrix[2][1]

def solve\_system(coefficients, constants):

    denominator = det([[coefficients[0][0], coefficients[0][1], coefficients[0][2]],

                       [coefficients[1][0], coefficients[1][1], coefficients[1][2]],

                       [coefficients[2][0], coefficients[2][1], coefficients[2][2]]])

    x\_matrix = [[constants[0], coefficients[0][1], coefficients[0][2]],

                [constants[1], coefficients[1][1], coefficients[1][2]],

                [constants[2], coefficients[2][1], coefficients[2][2]]]

    y\_matrix = [[coefficients[0][0], constants[0], coefficients[0][2]],

                [coefficients[1][0], constants[1], coefficients[1][2]],

                [coefficients[2][0], constants[2], coefficients[2][2]]]

    z\_matrix = [[coefficients[0][0], coefficients[0][1], constants[0]],

                [coefficients[1][0], coefficients[1][1], constants[1]],

                [coefficients[2][0], coefficients[2][1], constants[2]]]

    x = det(x\_matrix) / denominator

    y = det(y\_matrix) / denominator

    z = det(z\_matrix) / denominator

    return x, y, z

# Коэффициенты перед неизвестными

coefficients = [[7, 2, 3], [10, -6, 4], [5, -7, 3]]

# Константы

constants = [15, 30, 21]

x, y, z = solve\_system(coefficients, constants)

print(f'x = {x}, y = {y}, z = {z}')

Результаты выполнения программы:



**Задача2.** 1)Дополнить полученную программу процедурой перестановки строк. Добавить возможность задания коэффициентов системы уравнений с клавиатуры для произвольного значения n. 2) Решить методом Гаусса систему уравнений

Порядок решения 1):

1. Дополняем программу процедурой перестановки
2. Добавляем возможность ввода коэффициентов с клавиатуры

Программа для решения задачи:

def main():

    n = int(input("Введите размерность системы уравнений: "))

    a = []

    for i in range(n):

        row = [float(x) for x in input(f"Введите коэффициенты {i+1}-го уравнения через пробел: ").split()]

        a.append(row)

    b = [float(input(f"Введите свободный член {i+1}-го уравнения: ")) for i in range(n)]

    # Прямой ход

    for k in range(n-1):

        if a[k][k] == 0:

            max\_index = k

            max\_value = abs(a[k][k])

            for i in range(k+1, n):

                if abs(a[i][k]) > max\_value:

                    max\_index = i

                    max\_value = abs(a[i][k])

            # Поменять строки местами

            a[k], a[max\_index] = a[max\_index], a[k]

            b[k], b[max\_index] = b[max\_index], b[k]

        for i in range(k+1, n):

            factor = a[i][k] / a[k][k]

            for j in range(k, n):

                a[i][j] -= factor \* a[k][j]

            b[i] -= factor \* b[k]

    # Обратный ход

    x = [0] \* n

    for i in range(n-1, -1, -1):

        s = b[i]

        for j in range(i+1, n):

            s -= a[i][j] \* x[j]

        x[i] = s / a[i][i]

    print("Решение:")

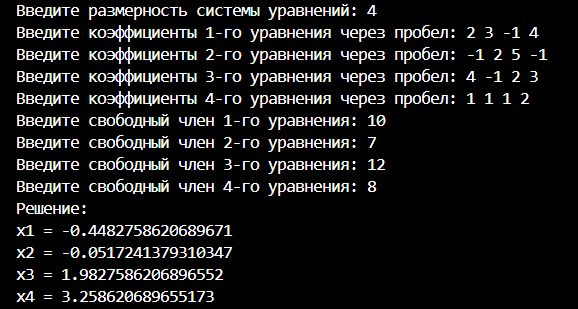
    for i in range(n):

        print(f"x{i+1} = {x[i]}")

if \_\_name\_\_ == "\_\_main\_\_":

   main()

Результаты выполнения программы:



Порядок решения 2):

1. Решаем систему уравнений методом Гауса
2. Метод Гаусса – последовательное исключение переменных, когда система приводится к равносильной системе ступенчатого вида, из которого последовательно, начиная с последних переменных, находятся все остальные

Программа для решения задачи:

def gauss\_elimination(coefficients, constants):

    n = len(coefficients)

    # Прямой ход

    for i in range(n):

        # Поиск ведущего элемента

        max\_row = i

        for j in range(i+1, n):

            if abs(coefficients[j][i]) > abs(coefficients[max\_row][i]):

                max\_row = j

        # Перестановка строк

        coefficients[i], coefficients[max\_row] = coefficients[max\_row], coefficients[i]

        constants[i], constants[max\_row] = constants[max\_row], constants[i]

        # Приведение к ступенчатому виду

        for j in range(i+1, n):

            factor = coefficients[j][i] / coefficients[i][i]

            for k in range(i, n):

                coefficients[j][k] -= factor \* coefficients[i][k]

            constants[j] -= factor \* constants[i]

    # Обратный ход

    solutions = [0] \* n

    for i in range(n-1, -1, -1):

        total = constants[i]

        for j in range(i+1, n):

            total -= coefficients[i][j] \* solutions[j]

        solutions[i] = total / coefficients[i][i]

    return solutions

# Пример использования

coefficients = [[7, 2, 3], [10, -6, 4], [5, -7, 3]]

constants = [15, 30, 21]

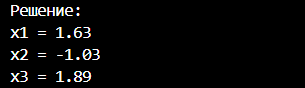
solutions = gauss\_elimination(coefficients, constants)

print("Решение:")

for i, solution in enumerate(solutions):

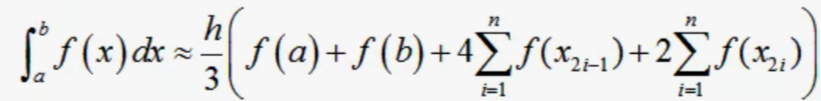
    print(f"x{i+1} = {solution:.2f}")

Результаты выполнения программы:



**Задача3.** 1) Напишите программу, которая вычисляет интеграл методом трапеций и методом Симпсона для n = 12. Сравните результаты с точечным значением интеграла. 2) Найти площадь фигуры, ограниченной линиями x = a, x = b, y = 0, y = f(x), используя методы прямоугольников, трапеций и Симпсона. Найти точечное значение интеграла.Сравнить полученные значения. Какой из методов окажется точнее?

Порядок решения 1):

1. Метод трапеции основан на том, что криволинейная трапеция приближается прямолинейно, т.е площадь элементарного прямоугольника вычисляют по формулам
2. Метод Симпсона - вычисления происходят по формуле 

Программа для решения задачи:

import numpy as np

# Функция для вычисления значения подынтегральной функции

def f(x):

    return 3 \* x / (x\*\*2 + 1)

# Метод трапеций

def trapezoidal\_rule(f, a, b, n):

    h = (b - a) / n

    result = 0.5 \* (f(a) + f(b))

    for i in range(1, n):

        result += f(a + i\*h)

    result \*= h

    return result

# Метод Симпсона

def simpsons\_rule(f, a, b, n):

    h = (b - a) / n

    x = np.linspace(a, b, n+1)

    result = f(a) + f(b)

    for i in range(1,n):

        if i % 2 == 0:

            result += 2\*f(x[i])

        else:

            result += 4\*f(x[i])

    return result \* h/3

a = 0 # Нижний предел интегрирования

b = 4 # Верхний предел интегрирования

n = 12 # Количество разбиений

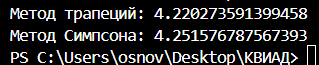
integral\_trapezoidal = trapezoidal\_rule(f,a,b,n)

integral\_simpsons= simpsons\_rule(f,a,b,n)

print("Метод трапеций:", integral\_trapezoidal)

print("Метод Симпсона:", integral\_simpsons)

Результаты выполнения программы:



Порядок решения 2):

1. Метод прямоугольников – Отрезок делится на равные части. После чего происходят вычисления.
2. Метод Симпсона окажется точнее. Если увеличить кол-во точек, то это приведет к уменьшению погрешности.

Программа для решения задачи:

import numpy as np

# Заданная функция

def f(x):

    return np.sqrt(6\*x - 5)

# Точное значение интеграла

def F(x):

    return (2/3)\*(6\*x - 5)\*\*(3/2)

a = 1 # Начало отрезка

b = 9 # Конец отрезка

# Точное значение интеграла на отрезке [a, b]

exact\_integral = F(b) - F(a)

print("Точное значение интеграла:", exact\_integral)

# Метод прямоугольников (левых)

n\_rectangles = 1000 # Количество прямоугольников для разбиения отрезка

dx = (b-a)/n\_rectangles

rect\_area\_left = sum(f(a + i\*dx)\*dx for i in range(n\_rectangles))

print("Площадь методом прямоугольников:", rect\_area\_left)

# Метод трапеций

trapezoid\_area = dx\*(f(a) + f(b))/2 + sum(f(a + i\*dx) for i in range(1, n\_rectangles))

print("Площадь методом трапеций:", trapezoid\_area)

# Метод Симпсона

n\_intervals\_simpson = n\_rectangles // 2 # Число интервалов должно быть четным для метода Симпсона

h\_simpson = (b-a)/(2\*n\_intervals\_simpson)

simpson\_sum\_odd = sum(f(a + (2\*i-1)\*h\_simpson) for i in range(1, n\_intervals\_simpson+1))

simpson\_sum\_even= sum(f(a + 2\*i\*h\_simpson) for i in range(1,n\_intervals\_simpson))

simpsons\_rule\_area=(h\_simpson/3)\*(f(a)+4\*simpson\_sum\_odd+2\*simpson\_sum\_even+f(b))

print("Площадь методом Симпсона:", simpsons\_rule\_area)

Результаты выполнения программы:

