МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Численные методы

# Лабораторная №1.

# Решение СЛАУ (Вариант 13).

Выполнил студент

Копосов Лев Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

**Постановка задачи.**

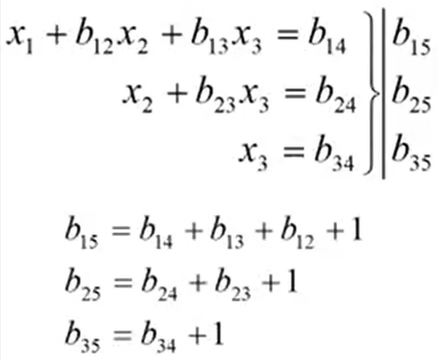
Написать программу на языке Python, которая решает систему линейных уравнений: 1) методом Гаусса и 2) методом простой итерации, не используя внешних модулей.

Вариант 13.

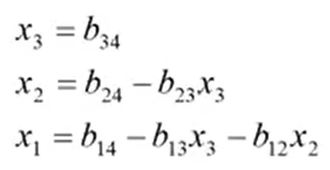
В трёх сосудах 54 л воды. Если из первого перелить во второй 4 л, то в обоих сосудах будет воды поровну, а если из третьего сосуда перелить во второй 17 л, то во втором окажется в четыре раза больше воды, чем в третьем. Сколько воды в каждом сосуде?

**Краткая теория используемых методов.**

Метод Гаусса - это метод, который используется для решения систем линейных уравнений. Основная идея метода заключается в последовательном исключении переменных, когда система уравнений будет приведена к треугольному виду (прямой ход метода Гаусса).

 (1.1)

Потом выполняется обратный ход метода Гаусса

 (1.2)

Чтобы выполнить элементарные преобразования, необходимо выполнить операцию деления каждого элемента выбранной строки на диагональный элемент системы.

aij = aij / aii (1.3), где aij - элемент строки.

Затем нужно для каждой нижестоящей строки выполнить операцию:

aij = aij - akjaik (1.4), где aij - элемент строки, r - номер опорной строки. Когда предыдущие шаги 1.3 и 1.4 выполнены, находим значения по формуле:

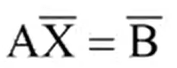
bin+2 = n+1Σi=j aij (1.5), где n - количество переменных, и находим новые переменные по формуле:

xi = bin+2 - nΣi=j aij \* xj (1.6)，где aij - коэффициент переменной.

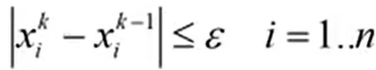
Результат получается точным, если матрица коэффициентов невырожденная.

Метод простой итерации - это метод, который используется для решения систем линейных уравнений приближением к точному решению.

Основная идея метода заключается в построении последовательности приближений к решению системы, которая сходится к решению системы.

Метод простой итерации заключается в приведении системы к виду, удобному для итераций. То есть переводим систему из вида  к виду . Для этого нужно каждую строку привести к виду:

kxi = Ci1k-1x1 + Ci2k-1x2 + … + Cii-1k-1x1-1 + Cii+1k-1xi+1 + … + Cink-1xn + fi (2.1) где k-1xi, kxi - элементы предпоследнего и последнего приближения, предварительно выполнив преобразование для каждой строки aij = aij / aii (2.2), где aij - элемент строки, и перенести элементы в правую часть уравнения, кроме xi.

Потом выбирают начальное приближение и продолжают процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие  (2.3), где xk и xk-1 - элементы последнего и предпоследнего приближения, ε - заданная точность.

**Алгоритм решения задачи, включая алгоритм решения СЛАУ.**

**Алгоритм решения СЛАУ методом Гаусса:**

Шаг 1. Выполняем матричное преобразование, чтобы на главной диагонали не стояли нули;

Шаг 2. Если дошли до конца системы уравнений, то Переходим к Шаг 5;

Иначе Переходим к Шаг 3;

Шаг 3. Выполняем преобразование опорной строки по формуле (1.3);

Шаг 4. Выполняем для каждой нижестоящей строки следующую операцию по формуле (1.4), Переходим к Шаг 2;

Шаг 5. Вычисляем переменную xi, выполняя обратный ход метода Гаусса (1.2).

**Алгоритм решения СЛАУ методом простой итерации:**

Шаг 1. Выполняем проверку матрицы на сходимость, сравнивая модуль элемента главной диагонали с суммой модулей остальных элементов строки;

Шаг 2. Если дошли до конца системы уравнений, то Переходим к Шаг 5;

Иначе Переходим к Шаг 3;

Шаг 3. Приводим строку к виду (2.1), выполнив преобразование по формуле (2.2);

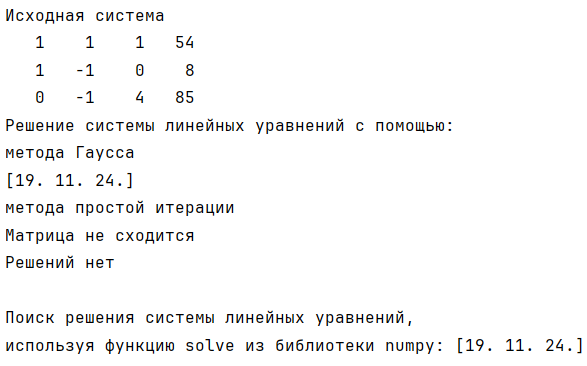
Шаг 4. Переносим элементы в правую часть уравнения, кроме xi;

Шаг 5. Находим значение следующего приближения по формуле (2.1);

Шаг 6. Если выполняется условие (2.3), то Выводим решение kxi, Конец.

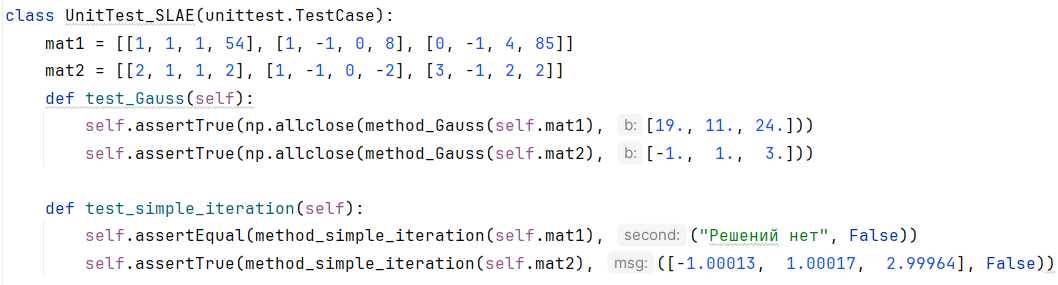
Иначе Переходим к Шаг 5;

**Вывод результата решения задачи.**



**Проверка правильности решения.**

Правильность решений была проверена с помощью unit тестов.



**Выводы.**

Решили задачу аналитически. Написали программу на языке Python, которая решает систему линейных уравнений методом Гаусса и методом простой итерации, не используя внешних модулей. Сравнили результаты решения между аналитическим и программным способом. Написали unit тесты для проверки разных типов ошибок.

**Приложение: код программы.**

| **from copy import deepcopy**  **import numpy as np**  **def display\_matrix(matrix):**  ***'''вывод матрицы в консоль'''***  **print("Исходная система")**  **for row in matrix:**  **print(" ".join("{:4}".format(el) for el in row))**  **def method\_Gauss(matrix):**  ***'''Решение систем линейных алгебраических уравнений***  ***методом Гаусса'''***  **def conversion\_elementary(mat):**  ***''' элементарное преобразование '''***  **row\_count = len(mat)**  **for k in range(row\_count):**  **d = mat[k][k]**  **if int(d) == 0:**  **for i in range(k + 1, row\_count):**  **if int(d) == 0:**  **print("Деление на ноль невозможно")**  **return None**  **for j in range(k, row\_count):**  **mat[i][j], mat[k][j] = mat[k][j], mat[i][j]**  **mat[i][-1], mat[k][-1] = mat[k][-1], mat[i][-1]**  **d = mat[k][k]**  **for i in range(row\_count):**  **mat[k][i] = mat[k][i] / d**  **mat[k][-1] = mat[k][-1] / d**  **for i in range(k + 1, row\_count):**  **mat[i][-1] = mat[i][-1] - mat[k][-1] \* mat[i][k]**  **t = mat[i][k]**  **for j in range(row\_count):**  **mat[i][j] = mat[i][j] - mat[k][j] \* t**  **return mat**  **def conversion\_reverse(mat):**  ***''' обратный ход метода Гаусса'''***  **row\_count = len(mat)**  **res = [0] \* row\_count**  **res[-1] = mat[-1][-1] / mat[-1][-2]**  **for i in range(row\_count - 2, -1, -1):**  **c = mat[i][-1]**  **for j in range(i + 1, row\_count):**  **c = c - mat[i][j] \* res[j]**  **res[i] = c / mat[i][i]**  **return res**  **mat\_copy = deepcopy(matrix)**  **conversion = conversion\_elementary(mat\_copy)**  **if conversion == None:**  **return "Решений нет"**  **res = conversion\_reverse(conversion)**  **return np.round(res, 5)**  **def method\_simple\_iteration(matrix,**  **max\_iter = 100,**  **accuracy = 1e-3):**  ***'''Решение систем линейных алгебраических уравнений***  ***методом простой итерации'''***  **def matrix\_norm(mat):**  ***''' проверка на сходимость матрицы '''***  **row\_count = len(mat)**  **for i in range(row\_count):**  **sum = 0**  **for j in range(row\_count - 1):**  **if i != j:**  **sum += abs(mat[i][j])**  **if sum >= abs(mat[i][i]):**  **return False**  **return True**  **def conversion(mat):**  ***''' преобразования от вида AX=B к X=CX+F '''***  **row\_count = len(mat)**  **for k in range(row\_count):**  **d = mat[k][k]**  **for i in range(row\_count):**  **if i != k and mat[k][i] != 0:**  **mat[k][i] = -mat[k][i] / d**  **if mat[k][-1] != 0:**  **mat[k][-1] = mat[k][-1] / d**  **mat[k][k] = mat[k][-1]**  **return mat**  **def iteration\_method(mat):**  **row\_count = len(mat)**  **t = [0] \* row\_count**  **for i in range(row\_count):**  **for j in range(row\_count):**  **if i != j:**  **t[i] = t[i] + mat[i][j] \* mat[j][-1]**  **else:**  **t[i] = t[i] + mat[i][j]**  **return t**  **mat\_copy = deepcopy(matrix)**  **convergence = True**  **if not matrix\_norm(mat\_copy):**  **convergence = False**  **res = conversion(mat\_copy)**  **cur\_iter = iteration\_method(res)**  **for n\_iter in range(max\_iter):**  **prev\_iter = cur\_iter**  **for u in range(len(res)):**  **res[u][-1] = prev\_iter[u]**  **cur\_iter = iteration\_method(res)**  **l\_iter = len(cur\_iter)**  **delta\_vec = [0] \* l\_iter**  **for i in range(l\_iter):**  **delta\_vec[i] = abs(cur\_iter[i] - prev\_iter[i])**  **if max(delta\_vec) < accuracy:**  **return np.round(cur\_iter, 5), convergence**  **return "Решений нет", convergence**  **if \_\_name\_\_ == '\_\_main\_\_':**  **matrix = [[1, 1, 1, 54],**  **[1, -1, 0, 8],**  **[0, -1, 4, 85]]**  **display\_matrix(matrix)**  **print("Решение системы линейных уравнений с помощью:")**  **print("метода Гаусса")**  **res\_list\_Gauss = method\_Gauss(matrix)**  **print(res\_list\_Gauss)**  **print("метода простой итерации")**  **res\_list\_iter, convergence = method\_simple\_iteration(matrix)**  **if not convergence:**  **print("Матрица не сходится")**  **print(res\_list\_iter)**  **a = [row[:-1] for row in matrix]**  **b = [row[-1] for row in matrix]**  **correct\_result = np.round(np.linalg.solve(a, b), 5)**  **print(f"\nПоиск решения системы линейных уравнений, \n"**  **f"используя функцию solve "**  **f"из библиотеки numpy: {correct\_result}")** |
| --- |