МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

# ОТЧЁТ

по лабораторной работе №2

" РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХУРАВНЕНИЙ"

по дисциплине

*Вычислительная математика*

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Панкратова А.З.\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Халеев А.А. \_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_\_21-ВМз-4\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2023

**Тема работы:**

Решение систем линейных уравнений с одной неизвестной.

**Цель работы**:

Изучить численные методы и алгоритмы решения систем линейных уравнений

**Постановка задачи:**

Реализовать изученные алгоритмы решения систем линейных уравнений и провести сравнение методов.

**Вариант №7:**

Решить систему линейных уравнений методом Гаусса, итерационным методом и методом Гаусса-Зейделя. При необходимости преобразовать систему к диагонально преобладающему виду. Сделать оценку количества итераций для итерационных методов, сравнить. Точность ε=0.001.

Дана система n алгебраических уравнений с n неизвестными:

Эту систему можно записать в матричном виде:

A - квадратная матрица коэффициентов, X - вектор-столбец неизвестных, B - вектор-столбец свободных членов.

Численные методы решения систем линейных уравнений делятся на прямые и итерационные.

Прямые методы используют конечные соотношения для вычисления неизвестных. Эти методы сравнительно просты и пригодны для широкого класса систем. Недостатки: требуют хранения в памяти ЭВМ сразу всей матрицы A. При больших порядках системы расходуется много места в памяти и накапливается вычислительная погрешность. Кроме того, существенно возрастает время вычисления вектора X. Поэтому прямые методы обычно применяют при небольших порядках системы (n<200). Примеры прямых методов - метод определителей Крамера, метод Гаусса. Первый из них применяется крайне редко, так как с ростом n алгоритм нахождения определителей резко возрастает. Метод Гаусса будет подробно рассмотрен в дальнейшем.

Итерационные методы основаны на последовательных приближениях. Задается некоторое приближенное значение вектора X – начальное приближение. Затем с помощью некоторого алгоритма проводится первый цикл вычислений – итерация, в результате которого получается новое приближение вектора X. Итерации проводятся до получения решения с заданной точностью. Алгоритм решения систем линейных уравнений здесь более сложен, чем у прямых методов. Не всегда выполняется условие сходимости. Однако, в ряде случаев итерационные методы предпочтительнее. Они требуют хранения в памяти ЭВМ не всей матрицы A, а лишь нескольких векторов. Вычислительная погрешность практически не накапливается. Поэтому итерационные методы применимы и для больших порядков системы. Примеры - метод простой итерации и метод Зейделя.

***Метод Гаусса***

Метод основан на приведении матрицы системы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается x1 из всех последующих уравнений. Затем с помощью второго уравнения исключается x2 из последующих и т.д. Этот процесс называется прямым ходом метода Гаусса и продолжается до тех пор, пока в левой части последнего n-го уравнения не останется лишь один член с неизвестным xn.

В результате прямого хода система принимает вид:

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных, начиная с xn и кончая x1.

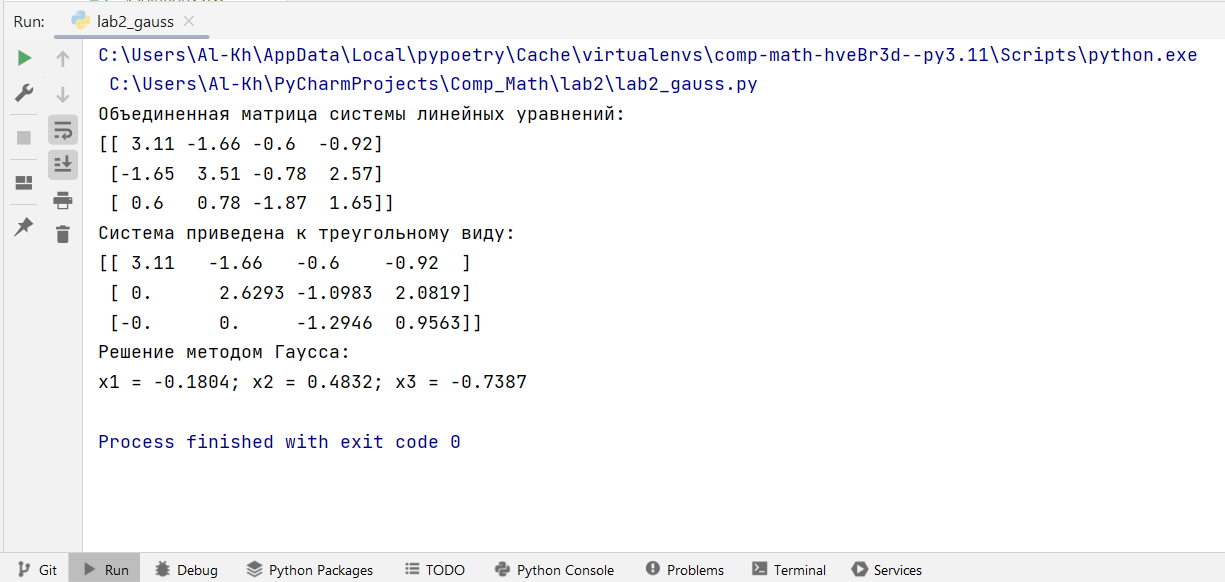
***Решение в Excel:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Решение системы методом Гаусса | | | | | |
| 3,1100 | -1,6600 | -0,6000 | -0,9200 |  |  |
| -1,6500 | 3,5100 | -0,7800 | 2,5700 |  |  |
| 0,6000 | 0,7800 | -1,8700 | 1,6500 |  |  |
|  |  |  |  |  |  |
| 1,0000 | -0,5338 | -0,1929 | -0,2958 |  |  |
| 0,0000 | 2,6293 | -1,0983 | 2,0819 |  |  |
| 0,0000 | 1,1003 | -1,7542 | 1,8275 |  |  |
|  |  |  |  |  |  |
| 1,0000 | -0,5338 | -0,1929 | -0,2958 | x1 | -0,1804 |
| 0,0000 | 1,0000 | -0,4177 | 0,7918 | x2 | 0,4832 |
| 0,0000 | 0,0000 | -1,2946 | 0,9563 | x3 | -0,7387 |
|  |  |  |  |  |  |

***Программа на Python:***

import numpy as np  
  
  
def gauss(a: np.array, b: np.array) -> np.ndarray:  
 *"""  
 Рассчитывает решение системы линейных уравнений с использованием метода Гаусса.  
  
 Параметры:  
 a (numpy.ndarray): Матрица коэффициентов системы линейных уравнений.  
 b (numpy.ndarray): Вектор правых частей системы линейных уравнений.  
  
 Возвращает:  
 x (numpy.ndarray): Вектор решения системы линейных уравнений.  
 """* n = len(a)  
 system = np.hstack([a, b.reshape(-1, 1)])  
 print(f"Объединенная матрица системы линейных уравнений:\n{system}")  
  
 for i in range(n):  
 *# нахождение индекса строки с максимальным абсолютным значением* max\_row\_index = np.abs(system[i:, i]).argmax() + i  
  
 *# перестановка строк* if i != max\_row\_index:  
 system[[i, max\_row\_index]] = system[[max\_row\_index, i]]  
  
 *# зануление i-тых элементов строк* for j in range(i + 1, n):  
 system[j] = system[j] - system[i] \* system[j, i] / system[i, i]  
 print(f"Система приведена к треугольному виду:\n{system.round(4)}")  
 x = np.zeros(n)  
 for i in range(n - 1, -1, -1):  
 x[i] = (system[i, -1] - np.dot(system[i, :-1], x)) / system[i, i]  
  
 return x  
  
  
def main() -> None:  
 *# Использование методов* A = np.array([[3.11, -1.66, -0.6],  
 [-1.65, 3.51, -0.78],  
 [0.6, 0.78, -1.87]])  
  
 B = np.array([-0.92, 2.57, 1.65])  
  
 solution = gauss(A, B)  
 print(f"Решение методом Гаусса:")  
 print("; ".join(f'x{i} = {x:.4f}' for i, x in enumerate(solution, 1)))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Вывод программы:



***Метод простых итераций (Якоби)***

Решение систем линейных уравнений с помощью метода простой итерации сводится

к следующему алгоритму.

1. Проверка условия сходимости. Для сходимости метода необходимо и достаточно,

чтобы в матрице A абсолютные значения всех диагональных элементов были больше

суммы модулей всех остальных элементов в соответствующей строке:

Недостатком итерационных методов является это достаточно жесткое условие

сходимости, которое выполняется далеко не для всех систем.

2. Если условие сходимости выполнено, то на следующем этапе необходимо задать

начальное приближение вектора неизвестных, в качестве которого обычно выбирается

нулевой вектор:

3. Затем организуется циклический вычислительный процесс, каждый цикл которого

представляет собой одну итерацию. В результате каждой итерации получается новое

значение вектора неизвестных. Для организации итерационного процесса запишем

систему в приведенном виде. При этом слагаемые, стоящие на главной диагонали

нормируются и остаются слева от знака равенства, а остальные переносятся в правую

часть. Приведенная система уравнений имеет вид:

4. Итерационный процесс заканчивается, если для каждой i-й компоненты вектора

неизвестных будет выполнено условие достижения точности:

где k - номер итерации, ε - заданная точность.

***Решение:***

Исходная система:

***Проверка сходимости:***

|3.11| < |-1.66| + |-0.60|+|-0.92| = |3.18| **- нет**

|3.51| < |-1.65| + |-0.78|+|2.57| = |5.00| - **нет**

|-1.87| < |0.60| + |0.78| +|1.65| = |3.03| - **нет**

**Сходимости нет.**

**Хотя данная система не имеет сходимости при любом начальном приближении, можно попробовать достигнуть заданной точности за некоторое количество итераций.**

Выбор начального приближения:

Запись приведенной системы уравнений:

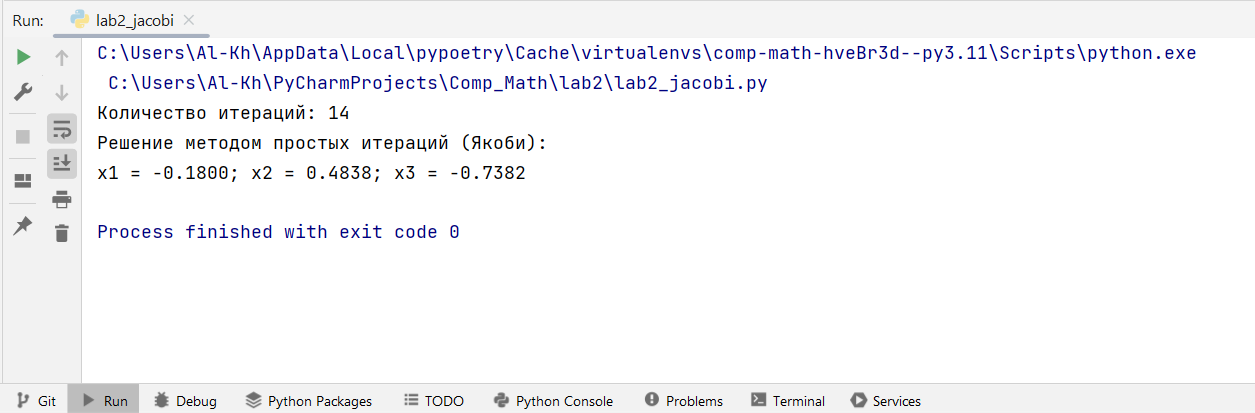
***Решение в Excel:***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Решение методом простой итерации | | | | | | | |
| k | x1 | x2 | x3 | Погрешность | | | Выполнено условие точности (ε=0.001) |
| 0 | 0 | 0 | 0 | x1 | x2 | x3 |  |
| 1 | -0,2958 | 0,7322 | -0,8824 | 0,2958 | 0,7322 | 0,8824 | НЕТ |
| 2 | -0,0752 | 0,3971 | -0,6719 | 0,2206 | 0,3351 | 0,2105 | НЕТ |
| 3 | -0,2135 | 0,5475 | -0,7409 | 0,1383 | 0,1505 | 0,0690 | НЕТ |
| 4 | -0,1465 | 0,4672 | -0,7225 | 0,0670 | 0,0803 | 0,0184 | НЕТ |
| 5 | -0,1858 | 0,5028 | -0,7345 | 0,0393 | 0,0356 | 0,0120 | НЕТ |
| 6 | -0,1692 | 0,4816 | -0,7323 | 0,0167 | 0,0212 | 0,0022 | НЕТ |
| 7 | -0,1800 | 0,4899 | -0,7357 | 0,0109 | 0,0083 | 0,0035 | НЕТ |
| 8 | -0,1762 | 0,4841 | -0,7358 | 0,0038 | 0,0059 | 0,0000 | НЕТ |
| 9 | -0,1794 | 0,4858 | -0,7370 | 0,0031 | 0,0018 | 0,0012 | НЕТ |
| 10 | -0,1787 | 0,4841 | -0,7373 | 0,0007 | 0,0018 | 0,0003 | НЕТ |
| 11 | -0,1797 | 0,4844 | -0,7378 | 0,0010 | 0,0003 | 0,0005 | ДА |
| 12 | -0,1796 | 0,4838 | -0,7380 | 0,0000 | 0,0006 | 0,0002 | ДА |
| 13 | -0,1800 | 0,4838 | -0,7382 | 0,0003 | 0,0000 | 0,0002 | ДА |
| 14 | -0,1800 | 0,4836 | -0,7383 | 0,0001 | 0,0002 | 0,0001 | ДА |
| 15 | -0,1802 | 0,4835 | -0,7384 | 0,0001 | 0,0001 | 0,0001 | ДА |
| 16 | -0,1802 | 0,4834 | -0,7385 | 0,0000 | 0,0001 | 0,0001 | ДА |
| 17 | -0,1803 | 0,4834 | -0,7385 | 0,0001 | 0,0000 | 0,0001 | ДА |
| 18 | -0,1803 | 0,4833 | -0,7386 | 0,0000 | 0,0000 | 0,0000 | ДА |
| 19 | -0,1803 | 0,4833 | -0,7386 | 0,0000 | 0,0000 | 0,0000 | ДА |
| 20 | -0,1803 | 0,4833 | -0,7386 | 0,0000 | 0,0000 | 0,0000 | ДА |

***Программа на Python:***

import numpy as np  
def jacobi(a: np.ndarray, b:np.ndarray, epsilon:float, max\_iterations:int=1000) -> np.ndarray:  
 *"""  
 Рассчитывает решение системы линейных уравнений с использованием метода Якоби.  
  
 Параметры:  
 a (numpy.ndarray): Матрица коэффициентов системы линейных уравнений.  
 b (numpy.ndarray): Вектор правых частей системы линейных уравнений.  
 epsilon (float, опционально): Точность сходимости. По умолчанию 0.0001.  
 max\_iterations (int, опционально): Максимальное количество итераций. По умолчанию 1000.  
  
 Возвращает:  
 x (numpy.ndarray): Вектор решения системы линейных уравнений.  
 """* n = len(a)  
 x = np.zeros(n)  
 tmp = 0  
 for \_ in range(max\_iterations):  
 x\_new = np.copy(x)  
 tmp += 1  
  
 for i in range(n):  
 s1 = np.dot(a[i, :i], x[:i])  
 s2 = np.dot(a[i, i + 1:], x[i + 1:])  
 x\_new[i] = (b[i] - s1 - s2) / a[i, i]  
  
 if np.allclose(x, x\_new, rtol=epsilon):  
 break  
  
 x = x\_new  
 print(f"Количеcтво итераций: {tmp}")  
 return x  
  
  
def main() -> None:  
 *# Использование методов* A = np.array([[3.11, -1.66, -0.6],  
 [-1.65, 3.51, -0.78],  
 [0.6, 0.78, -1.87]])  
  
 B = np.array([-0.92, 2.57, 1.65])  
  
 solution = jacobi(A, B, epsilon=0.001)  
 print(f"Решение методом простых итераций (Якоби):")  
 print("; ".join(f'x{i} = {x:.4f}' for i, x in enumerate(solution, 1)))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Вывод программы:



***Метод Гаусса-Зейделя***

***Решение в Excel:***

***Программа на Python:***

Вывод программы:

Достоинство метода: очень быстрая сходимость к заданной точности.

Недостаток: громоздкий алгоритм, на каждой итерации необходимо вычислять значение функции и ее первой производной.

**Вывод:**

В данной лабораторной работе были изучены следующие методы численного решения нелинейных уравнений:

- шаговый метод

- метод половинного деления

- метод Ньютона

- метод простых итераций (Якоби)

Также были проведены программные вычисления с помощью MS Excel и языка программирования Python для уравнения, полученного согласно варианту.

Все решения сходятся на заданной точности. В процессе решения были проанализированы особенности, плюсы и минусы каждого из использованных методов.