

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 ЛИНЕЙНЫХ УРАВНЕНИЙ"

по дисциплине

Вычислительная математика

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

(подпись)

Панкратова А.З.
(фамилия, и.,о.)

СТУДЕНТ:

(подпись)

Халеев А.А.
(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 $\varepsilon=0.001$.

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

Дана система n алгебраических уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots a_{2n}x_n = b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots a_{nn}x_n = b_n \end{cases}$$

Эту систему можно записать в матричном виде: $A * X = B$, где

$$A = \begin{pmatrix} a_{11}, a_{12}, \dots a_{1n} \\ a_{21}, a_{22}, \dots a_{2n} \\ \dots \\ a_{n1}, a_{n2}, \dots a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}.$$

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 x_1 из всех последующих уравнений. Затем с помощью второго уравнения исключается x_2 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 x_n .

В результате прямого хода система принимает вид:

$$\begin{cases} a'_{11}x_1 + a'_{12}x_2 + \dots + a'_{1n}x_n = b'_1 \\ 0 + a'_{22}x_2 + \dots + a'_{2n}x_n = b'_2 \\ \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \\ 0 + 0 + \dots + a'_{n-1n}x_n = b'_{n-1} \\ 0 + 0 + \dots + a'_{nn}x_n = b'_n \end{cases}$$

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных, начиная с x_n и кончая x_1 .

Исходная система:

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

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:

```
Run: lab2_gauss x
C:\Users\A1-Kh\AppData\Local\pypoetry\Cache\virtualenvs\comp-math-hveBr3d--py3.11\Scripts\python.exe
C:\Users\A1-Kh\PyCharmProjects\Comp_Math\lab2\lab2_gauss.py
Объединенная матрица системы линейных уравнений:
[[ 3.11 -1.66 -0.6 -0.92]
 [-1.65  3.51 -0.78  2.57]
 [ 0.6   0.78 -1.87  1.65]]
Система приведена к треугольному виду:
[[ 3.11  -1.66  -0.6  -0.92 ]
 [ 0.      2.6293 -1.0983  2.0819]
 [-0.      0.     -1.2946  0.9563]]
Решение методом Гаусса:
x1 = -0.1804; x2 = 0.4832; x3 = -0.7387

Process finished with exit code 0
```

Git Run Debug Python Packages TODO Python Console Problems Terminal Services

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:

$$|a_{ii}| > \sum_{i=1, i \neq j}^n |a_{ii}|$$

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:

$$x_1^{(0)} = x_2^{(0)} = \dots = x_n^{(0)} = 0$$

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:

$$\begin{cases} x_1^{(k)} = [b_1 - (a_{12}x_2^{(k-1)} + \dots + a_{1n}x_n^{(k-1)})]/a_{11} \\ x_2^{(k)} = [b_2 - (a_{21}x_1^{(k-1)} + \dots + a_{2n}x_n^{(k-1)})]/a_{22} \\ \vdots \\ x_n^{(k)} = [b_n - (a_{n1}x_1^{(k-1)} + \dots + a_{nn}x_n^{(k-1)})]/a_{nn} \end{cases}$$

4. Итерационный процесс заканчивается, если для каждой i -й компоненты вектора неизвестных будет выполнено условие достижения точности:

$$\left| x_i^{(k)} - x_i^{(k-1)} \right| < \varepsilon$$

где k - номер итерации, ε - заданная точность.

Решение:

Исходная система:

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

Проверка сходимости:

$$\begin{aligned} |3.11| &< |-1.66| + |-0.60| + |-0.92| = |3.18| \text{ - нет} \\ |3.51| &< |-1.65| + |-0.78| + |2.57| = |5.00| \text{ - нет} \\ |-1.87| &< |0.60| + |0.78| + |1.65| = |3.03| \text{ - нет} \end{aligned}$$

Сходимости нет.

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

Выбор начального приближения:

$$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$$

Запись приведенной системы уравнений:

$$\begin{cases} x_1^{(k)} = -\frac{0.92}{3.11} - \frac{1}{3.11} \cdot [(-1.66x_2^{(k-1)} - 0.60x_3^{(k-1)})] \\ x_2^{(k)} = \frac{2.57}{3.51} - \frac{1}{3.51} \cdot [(-1.65x_1^{(k-1)} - 0.78x_3^{(k-1)})] \\ x_3^{(k)} = -\frac{1.65}{1.87} + \frac{1}{1.87} \cdot [(0.60x_1^{(k-1)} + 0.78x_2^{(k-1)})] \end{cases}$$

Решение в Excel:

Решение методом простой итерации							
k	x1	x2	x3	Погрешность			Выполнено условие точности (ε=0.001)
	x1	x2	x3	x1	x2	x3	
0	0	0	0				
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) # начальное приближение (нулевой вектор)
    it_counter = 0 # счетчик итераций
    for _ in range(max_iterations):
        x_new = np.copy(x)
        it_counter += 1

        for i in range(n):
            # произведение векторов (срезов, чтобы исключить диагональные элементы)
            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] # вычисление значений нового вектора

        if np.allclose(x, x_new, rtol=epsilon): # проверка на соответствие заданной точности
            break

        x = x_new
    print(f"Количество итераций: {it_counter}")
    return x

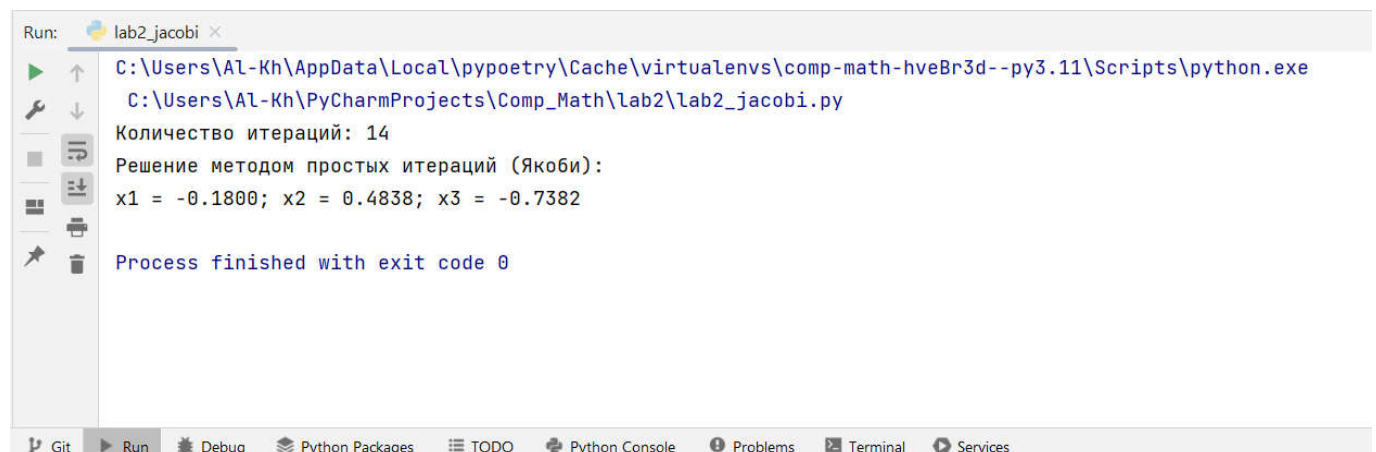
def main() -> None:
    # Использование методов
    A = np.array([[3.11, -1.66, -0.6],
                  [-1.65, 3.51, -0.78],
                  [0.6, 0.78, -1.87]])

    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:



```
Run: lab2_jacobi x
C:\Users\Al-Kh\AppData\Local\pypoetry\Cache\virtualenvs\comp-math-hveBr3d-py3.11\Scripts\python.exe
C:\Users\Al-Kh\PyCharmProjects\Comp_Math\lab2\lab2_jacobi.py
Количество итераций: 14
Решение методом простых итераций (Якоби):
x1 = -0.1800; x2 = 0.4838; x3 = -0.7382

Process finished with exit code 0
```

Метод Гаусса-Зейделя

Отличие метода Зейделя от метода простой итерации заключается в том, что при вычислении очередного приближения вектора неизвестных используются уже уточненные значения на этом же шаге итерации. Это обеспечивает более быструю сходимость метода Зейделя. Алгоритм метода Зейделя весьма похож на алгоритм предыдущего метода. Первые два пункта (проверка условия сходимости и выбор начального приближения), а также четвертый пункт (проверка достижения заданной точности) остаются без изменения.

Отличается здесь только третий пункт алгоритма. При вычислении x_1 используется информация об остальных неизвестных, найденных на предыдущей итерации. При вычислении x_2 используется значение x_1 , найденное на текущей итерации и значения остальных переменных, найденные на предыдущей итерации и т.д. Наконец, при вычислении последней компоненты вектора неизвестных x_n используется информация об остальных компонентах, найденных на текущей итерации. Приведенная система уравнений имеет вид:

$$\begin{cases} x_1^{(k)} = [b_1 - (a_{12}x_2^{(k-1)} + \dots + a_{1n}x_n^{(k-1)})]/a_{11} \\ x_2^{(k)} = [b_2 - (a_{21}x_1^{(k)} + \dots + a_{2n}x_n^{(k-1)})]/a_{22} \\ \vdots \\ x_n^{(k)} = [b_n - (a_{n1}x_1^{(k)} + \dots + a_{nn}x_n^{(k)})]/a_{nn} \end{cases}$$

Решение:

Исходная система:

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

Проверка сходимости:

$$|3.11| < |-1.66| + |-0.60| + |-0.92| = |3.18| \text{ - нет}$$

$$|3.51| < |-1.65| + |-0.78| + |2.57| = |5.00| \text{ - нет}$$

$$|-1.87| < |0.60| + |0.78| + |1.65| = |3.03| - \text{нет}$$

Сходимости нет.

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

Выбор начального приближения:

$$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$$

Запись приведенной системы уравнений:

$$\begin{cases} x_1^{(k)} = -\frac{0.92}{3.11} - \frac{1}{3.11} \cdot [(-1.66x_2^{(k-1)} - 0.60x_3^{(k-1)})] \\ x_2^{(k)} = \frac{2.57}{3.51} - \frac{1}{3.51} \cdot [(-1.65x_1^{(k)} - 0.78x_3^{(k-1)})] \\ x_3^{(k)} = -\frac{1.65}{1.87} + \frac{1}{1.87} \cdot [(0.60x_1^{(k)} + 0.78x_2^{(k)})] \end{cases}$$

Решение в Excel:

Решение методом Гаусса - Зейделя							
k	x1	x2	x3	Погрешность			Выполнено условие точности (ε=0.001)
0	0	0	0	x1	x2	x3	
1	-0,2958	0,5931	-0,7299	0,2958	0,5931	0,7299	НЕТ
2	-0,1200	0,5136	-0,7067	0,1758	0,0796	0,0232	НЕТ
3	-0,1580	0,5009	-0,7241	0,0380	0,0127	0,0175	НЕТ
4	-0,1682	0,4922	-0,7310	0,0102	0,0087	0,0069	НЕТ
5	-0,1741	0,4879	-0,7347	0,0059	0,0043	0,0037	НЕТ
6	-0,1771	0,4857	-0,7366	0,0030	0,0022	0,0019	НЕТ
7	-0,1787	0,4845	-0,7376	0,0016	0,0012	0,0010	НЕТ
8	-0,1795	0,4839	-0,7381	0,0008	0,0006	0,0005	ДА
9	-0,1799	0,4836	-0,7384	0,0004	0,0003	0,0003	ДА
10	-0,1802	0,4834	-0,7385	0,0002	0,0002	0,0001	ДА
11	-0,1803	0,4833	-0,7386	0,0001	0,0001	0,0001	ДА
12	-0,1803	0,4833	-0,7386	0,0001	0,0000	0,0000	ДА
13	-0,1804	0,4833	-0,7386	0,0000	0,0000	0,0000	ДА
14	-0,1804	0,4833	-0,7387	0,0000	0,0000	0,0000	ДА
15	-0,1804	0,4833	-0,7387	0,0000	0,0000	0,0000	ДА
16	-0,1804	0,4833	-0,7387	0,0000	0,0000	0,0000	ДА
17	-0,1804	0,4833	-0,7387	0,0000	0,0000	0,0000	ДА
18	-0,1804	0,4833	-0,7387	0,0000	0,0000	0,0000	ДА
19	-0,1804	0,4832	-0,7387	0,0000	0,0000	0,0000	ДА
20	-0,1804	0,4832	-0,7387	0,0000	0,0000	0,0000	ДА

Программа на Python:

```
import numpy as np

def gauss_seidel(a, b, epsilon=0.001, max_iterations=1000):
    """
    Рассчитывает решение системы линейных уравнений с использованием метода Гаусса-Зейделя.

    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) # начальное приближение (нулевой вектор)
    it_counter = 0 # счетчик итераций
    for _ in range(max_iterations):
        it_counter += 1
        for i in range(n):
            # произведение векторов (срезов, чтобы исключить диагональные элементы)
            s1 = np.dot(a[i, :i], x[:i])
            s2 = np.dot(a[i, i + 1:], x[i + 1:])
            x[i] = (b[i] - s1 - s2) / a[i, i] # обновление значений вектора
        if np.allclose(np.dot(a, x), b, rtol=epsilon): # проверка заданной точности
            break
    print(f"Количество итераций: {it_counter}")
    return x

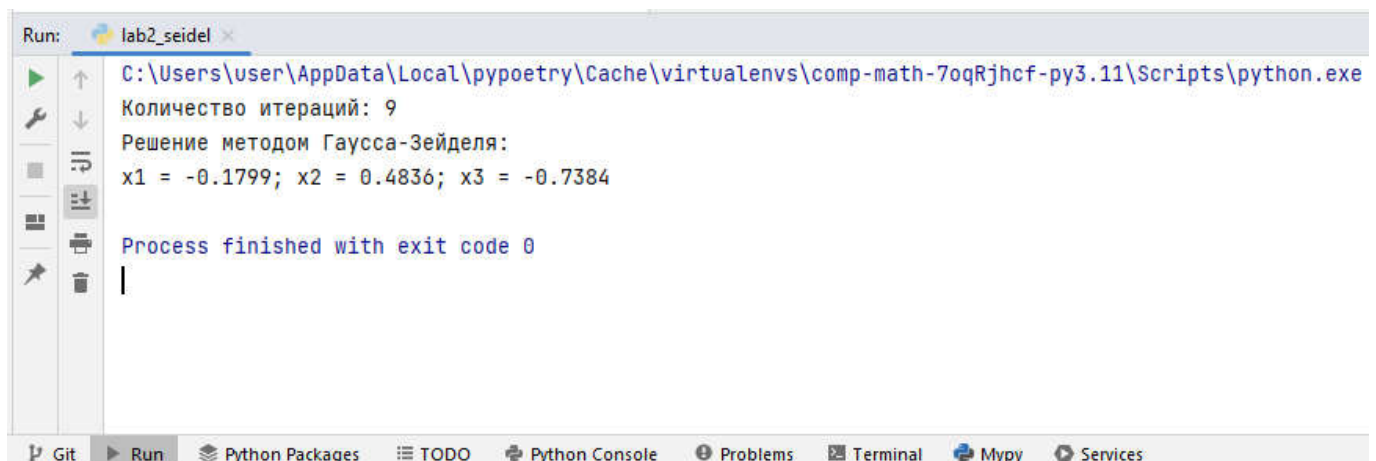
def main() -> None:
    # Использование методов
    A = np.array([[3.11, -1.66, -0.6],
                  [-1.65, 3.51, -0.78],
                  [0.6, 0.78, -1.87]])

    B = np.array([-0.92, 2.57, 1.65])

    solution = gauss_seidel(A, B, epsilon=0.001)
    print(f"Решение методом Гаусса-Зейделя:")
    print("; ".join(f'x{i} = {x:.4f}' for i, x in enumerate(solution, 1)))

if __name__ == '__main__':
    main()
```

Вывод программы:



The screenshot shows a Python IDE window titled 'lab2_seidel'. The output console displays the following text:

```
Run: C:\Users\user\AppData\Local\pypoetry\Cache\virtualenvs\comp-math-7oqRjhcf-py3.11\Scripts\python.exe
Количество итераций: 9
Решение методом Гаусса-Зейделя:
x1 = -0.1799; x2 = 0.4836; x3 = -0.7384
Process finished with exit code 0
```

The IDE interface includes a sidebar with icons for Run, Python Packages, TODO, Python Console, Problems, Terminal, Mypy, and Services. The bottom status bar shows 'Run', 'Python Packages', 'TODO', 'Python Console', 'Problems', 'Terminal', 'Mypy', and 'Services'.

Вывод:

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

- метод Гаусса
- метод простых итераций (Якоби)
- метод Гаусса-Зейделя

Проблемы итерационных методов (Якоби и Гаусса-Зейделя) в необходимости выполнения достаточно жесткого условия сходимости, а также достаточно большого количества итераций для достижения необходимой точности. Точный результат не может быть достигнут с использованием итерационных методов.

Полученные количества итераций для достижения заданной точности ($\varepsilon=0.001$):

метод Якоби: 14

метод Гаусса – Зейделя: 9

На основе полученных данных можно сделать заключение о том, что метод Гаусса-Зейделя выполняется быстрее и использует меньшее количество итераций, за счет использования сразу вычисленных на текущей итерации значений вектора.