«Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Проскурин А.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

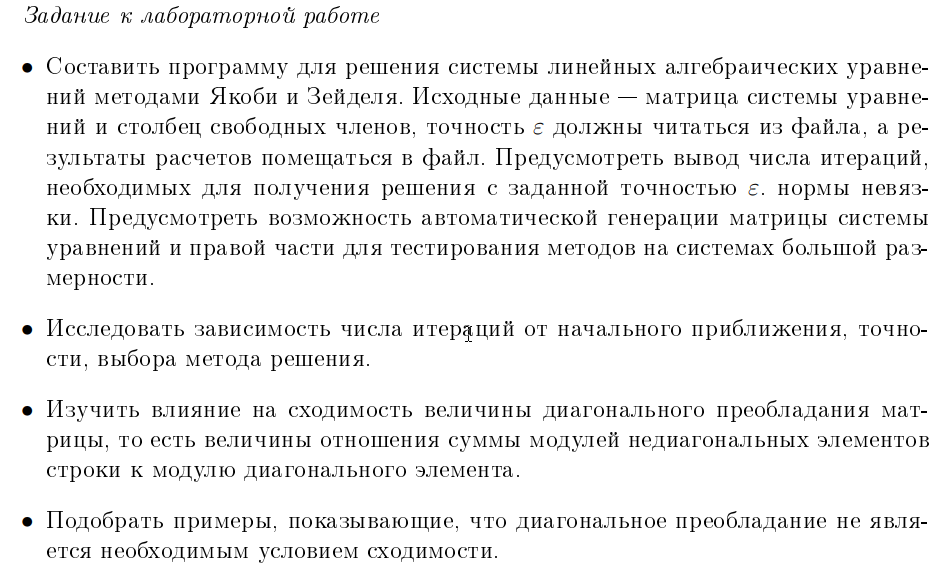
Лабораторной работе №2

«Итерационные методы Якоби и Зейделя»

Студент группы ПИ 92 В.М. Шульпов

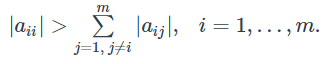
Преподаватель доцент, к. м. н. Проскурин А.В.

Барнаул 2022



**Краткое описание алгоритма и расчетные формулы**

Изначально производится проверка достаточного условия сходимости для методов Якоби и Зейделя. Условие сходимости, заключается в том, сумма модулей не диагональных элементов строки должна быть меньше модуля диагонального элемента строки в каждой строке (диагональное преобладание).

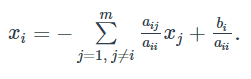


**Метод Якоби**

Поиск решения происходит до тех пор, пока выполняется условие:

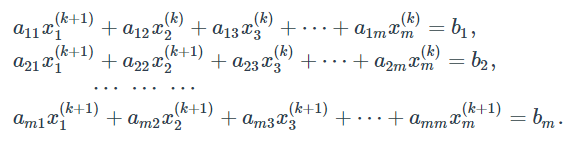
где x^(k+1) и x^(k) – решение системы на k+1 и k итерации соответственно, а e – точность.

На k итерации в текущую строку системы подставляется значение X с k-1 итерации, причем в первой итерации подставляем первоначальное приближение, кроме диагональных элементов. После чего из получившихся уравнений находим x на главной диагонали, разделив полученное значение разности на коэффициент при текущем X.



**Метод Зейделя**

Аналогичен методу Якоби, разница в том, что в каждой итерации, до диагонального элемента, X подставляется с текущей итерации, после диагонального элемента – с прошлой итерации.



**Описание функций программы:**

1. **def** read\_input\_data(filename, expended\_matrix):  
    *"""чтение расширенной матрицы из файла"""*
2. **def** read\_input\_epsilon(filename):  
    *"""чтение точности из файла"""*
3. **def** write\_output\_matrix(filename, matrix, title=**""**):  
    *"""запись результатов в файл"""*
4. **def** write\_output\_list(filename, \_list, title=**""**):  
    *"""запись результатов в файл"""*
5. **def** format\_print(n, a, b=**None**):  
    *"""вывод системы на экран"""*
6. **def** check\_diagonal\_dominance(factors\_at\_unknowns):  
    *""" проверка матрицы на диагональное преобладание """*
7. **def** count\_null\_x(b, diagonal\_a):  
    *""" посчет x для нулевого приближения  
    b - свободный член строки,  
    diagonal\_a - свободный член диагонального элемента строки  
    """*
8. **def** get\_null\_x\_list(factors\_at\_unknowns, free\_factors):  
    *""" подсчет всех x для нулевого приближения и их возврат  
    factors\_at\_unknowns - двумерных массив членов при неизвестных  
    free\_factors - столбец свободных членов """*
9. **def** func\_x(b, a\_array, diagonal\_x\_index):  
    *""" динамическое создание функции для подсчёта x (для строки)  
    b - свободный член строки,  
    a\_array - массив свободных членов строки,  
    diagonal\_x\_index - индекс подсчитываемого x (этот элем. на главной диагонали)"""* **def** created\_func\_x(x\_array, k):  
    *""" формула подсчета x с предустановленными свободными членами """* a\_array\_len = len(a\_array)  
    x = b  
    **for** i **in** range(0, a\_array\_len):  
    **if** i != diagonal\_x\_index:  
    x -= a\_array[i] \* x\_array[i] *# вычитаем все a[i]\*x[i], кроме диагонального* x /= a\_array[diagonal\_x\_index] *# делим на коэф. a текущего x* **return** x  
    **return** created\_func\_x
10. **def** create\_x\_functions(factors\_at\_unknowns, free\_factors):  
     *""" динамическое создание формул (функций) для подсчёта x (для строки)  
     factors\_at\_unknowns - двумерных массив членов при неизвестных  
     free\_factors - столбец свободных членов """*
11. **def** count\_x\_error(x\_prev, x):  
     *""" подсчитывает погрешность для x  
     |x (k) - x (k-1)|/ |x (k)| """*
12. **def** check\_accuracy\_is\_done(epsilon, errors):  
     *""" проверка выполнения всеми x заданной погрешности,  
     если все погрешности меньше, чем заданная - True, иначе - False """*
13. **def** format\_print\_x\_and\_errors(x\_lists, x\_errors):  
     *""" вывод таблицы вида | k | x1 x2 ... xn | e1 e2 ... en |"""*
14. **def** solve\_by\_yacobi(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
     *""" решение системы методом Якоби  
     factors\_at\_unknowns - двумерных массив членов при неизвестных  
     free\_factors - столбец свободных членов  
     x\_lists - таблица x-ов по всем приближениям  
     x\_errors - таблица погрешностей (по x-ам) для всех приближений  
     epsilon - величина допустимой погрешности """*
15. **def** solve\_by\_yacobi(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
     *""" решение системы методом Якоби  
     factors\_at\_unknowns - двумерных массив членов при неизвестных  
     free\_factors - столбец свободных членов  
     x\_lists - таблица x-ов по всем приближениям  
     x\_errors - таблица погрешностей (по x-ам) для всех приближений  
     epsilon - величина допустимой погрешности """*
16. **def** check\_discrepancy(a\_array, b\_array, x\_array):  
     *"""проверка на соотвествие (невязка)  
     a\_array - квадртная матрица  
     b\_array - столбец свободных членов  
     x\_array - итоговый ответ с заданной точностью """*
17. **def** generate\_rand\_matrix(i, j):  
     *""" генерация матрицы определенной размерности со случайными элементами """*
18. **def** main():

**Алгоритм функции решения метдом Якоби:**

**def** solve\_by\_yacobi(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
 *""" решение системы методом Якоби  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов  
 x\_lists - таблица x-ов по всем приближениям  
 x\_errors - таблица погрешностей (по x-ам) для всех приближений  
 epsilon - величина допустимой погрешности """* print(**'\nРЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ'**)  
 cols = len(factors\_at\_unknowns[0])  
 rows = len(factors\_at\_unknowns)  
  
 null\_x\_list = get\_null\_x\_list(factors\_at\_unknowns, free\_factors) *# подсчет всех x для нулевого приближения* x\_lists.append(null\_x\_list) *# добавление x-ов (0-го приближения) в таблицу x-ов всех приближений* empty\_error\_list = [**"-" for** \_ **in** range(cols)] *# пустая строка для первой строки таблицы погрешностей* x\_errors.append(empty\_error\_list) *# добавление пустой строки в таблицу погрешностей* x\_counting\_functions = create\_x\_functions(factors\_at\_unknowns, free\_factors) *# создание формул для подсчёта x-ов  
 # цикл по приближениям (k)* accuracy\_is\_done = **False** k = 0  
 **while** accuracy\_is\_done **is not True**:  
 *# счетчик номера приближения* k += 1  
 *# подсчёт x-ов их погрешностей* cur\_x\_list = []  
 cur\_x\_error\_list = []  
 **for** i **in** range(rows):  
 x = x\_counting\_functions[i](x\_lists[k - 1], k)  
 cur\_x\_list.append(x)  
 x\_prev = x\_lists[k - 1][i] *# соотвествующий предыдущий x (k-1)* x\_error = count\_x\_error(x\_prev, x)  
 cur\_x\_error\_list.append(x\_error)  
 x\_lists.append(cur\_x\_list)  
 x\_errors.append(cur\_x\_error\_list)  
 *# print(f'k: {k};\nx: {cur\_x\_list};\nпогрешности: {cur\_x\_error\_list}')  
 # проверка достижения требуемой погрешности* accuracy\_is\_done = check\_accuracy\_is\_done(epsilon, cur\_x\_error\_list)  
 **if** k > 200:  
 print(**'k > 200 до сих пор не сходится!'**)  
 **return False** print(**'\nрешение системы сходится'**)  
 **return True**

**Алгоритм функции решения метдом Зейделя:**

**def** solve\_by\_zeydel(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
 *""" решение системы методом Зейделя  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов  
 x\_lists - таблица x-ов по всем приближениям  
 x\_errors - таблица погрешностей (по x-ам) для всех приближений  
 epsilon - величина допустимой погрешности """* print(**'\nРЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ'**)  
 cols = len(factors\_at\_unknowns[0])  
 rows = len(factors\_at\_unknowns)  
  
 null\_x\_list = get\_null\_x\_list(factors\_at\_unknowns, free\_factors) *# подсчет всех x для нулевого приближения* x\_lists.append(null\_x\_list) *# добавление x-ов (0-го приближения) в таблицу x-ов всех приближений* empty\_error\_list = [**"-" for** \_ **in** range(cols)] *# пустая строка для первой строки таблицы погрешностей* x\_errors.append(empty\_error\_list) *# добавление пустой строки в таблицу погрешностей* x\_counting\_functions = create\_x\_functions(factors\_at\_unknowns, free\_factors) *# создание формул для подсчёта x-ов  
  
 # цикл по приближениям (k)* accuracy\_is\_done = **False** k = 0  
 **while** accuracy\_is\_done **is not True**:  
 *# счетчик номера приближения* k += 1  
 *# print(f'\nПОДСЧЕТ X И ПОГРЕШНОСТЕЙ ДЛЯ {k}-ОГО ПРИБЛИЖЕНИЯ')  
 # подсчёт x-ов их погрешностей* cur\_x\_list = []  
 cur\_x\_error\_list = []  
 x\_list\_tmp = x\_lists[k - 1].copy() *# для хранения списка иксов* **for** i **in** range(rows):  
 x = x\_counting\_functions[i](x\_list\_tmp, k)  
 cur\_x\_list.append(x)  
 x\_prev = x\_lists[k - 1][i] *# соотвествующий предыдущий x (k-1)* x\_error = count\_x\_error(x\_prev, x)  
 cur\_x\_error\_list.append(x\_error)  
 x\_list\_tmp[i] = x  
 x\_lists.append(cur\_x\_list)  
 x\_errors.append(cur\_x\_error\_list)  
 *# print(f'k: {k};\nx: {cur\_x\_list};\nпогрешности: {cur\_x\_error\_list}')  
 # проверка достижения требуемой погрешности* accuracy\_is\_done = check\_accuracy\_is\_done(epsilon, cur\_x\_error\_list)  
 **if** k > 200:  
 print(**'k > 200 до сих пор не сходится!'**)  
 **return False** print(**'\nрешение системы сходится'**)  
 **return True**

**Вывод**: с увеличением значения диагональных элементов кол-во итераций значительно уменьшается. Метод Зейделя является более рациональным, так как при одинаковых входных данных требует значительно меньшего кол-ва итераций для достижения заданной точности вычислений. Стоит заметить, что в методе Зейделя, изменения значения точности, влияет на кол-во итераций в меньшей степени, нежели в методе Якоби. Диагональное преобладание не является необходимым условием сходимости решения и чем оно выше, тем меньше итераций требуется.

**Тесты:**

**Тест 1 (данные из файлы 3x3 e=0.01 Якоби сошлось на 16 итерации, Зейделя на 5)**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab2/main.py

Использовать данные из файлов на диске - 1

Ввести свои данные - 2 :1

Чтение матрицы из файла проведено успешно

Чтение точности из файла проведено успешно

Исходная система:

( 5.00 -1.00 3.00 ) = ( 5.00)

( 1.00 -4.00 2.00 ) = ( 20.00)

( 2.00 -1.00 5.00 ) = ( 10.00)

Запись в файл проведена успешно (Расширенная матрица)

Решаем:

ПРОВЕРКА МАТРИЦЫ НА ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ

Диаг. элемент 0-й строки = 5.0; сумма недиаг. элементов строки = 4.0

В строке выполняется диагональное преобладание

Диаг. элемент 1-й строки = -4.0; сумма недиаг. элементов строки = 3.0

В строке выполняется диагональное преобладание

Диаг. элемент 2-й строки = 5.0; сумма недиаг. элементов строки = 3.0

В строке выполняется диагональное преобладание

В матрице выполняется диагональное преобладание

ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ ВЫПОЛНЯЕТСЯ

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ

решение системы сходится

ВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ

| k | x1 | x2 | x3 | e1 | e2 | e3 |

| 0 | 1.0000 |-5.0000 | 2.0000 | - | - | - |

| 1 |-1.2000 |-3.7500 | 0.6000 | 2.2000 | 1.2500 | 1.4000 |

| 2 |-0.1100 |-5.0000 | 1.7300 | 1.0900 | 1.2500 | 1.1300 |

| 3 |-1.0380 |-4.1625 | 1.0440 | 0.9280 | 0.8375 | 0.6860 |

| 4 |-0.4589 |-4.7375 | 1.5827 | 0.5791 | 0.5750 | 0.5387 |

| 5 |-0.8971 |-4.3234 | 1.2361 | 0.4382 | 0.4141 | 0.3466 |

| 6 |-0.6063 |-4.6063 | 1.4942 | 0.2908 | 0.2829 | 0.2581 |

| 7 |-0.8178 |-4.4045 | 1.3213 | 0.2114 | 0.2018 | 0.1729 |

| 8 |-0.6737 |-4.5438 | 1.4462 | 0.1441 | 0.1393 | 0.1249 |

| 9 |-0.7765 |-4.4453 | 1.3607 | 0.1028 | 0.0985 | 0.0855 |

| 10 |-0.7055 |-4.5138 | 1.4215 | 0.0710 | 0.0685 | 0.0608 |

| 11 |-0.7557 |-4.4656 | 1.3794 | 0.0502 | 0.0482 | 0.0421 |

| 12 |-0.7208 |-4.4992 | 1.4091 | 0.0349 | 0.0336 | 0.0297 |

| 13 |-0.7453 |-4.4756 | 1.3885 | 0.0245 | 0.0236 | 0.0207 |

| 14 |-0.7282 |-4.4921 | 1.4030 | 0.0171 | 0.0165 | 0.0145 |

| 15 |-0.7402 |-4.4805 | 1.3929 | 0.0120 | 0.0115 | 0.0101 |

| 16 |-0.7318 |-4.4886 | 1.4000 | 0.0084 | 0.0081 | 0.0071 |

Запись в файл проведена успешно (Значения x для метода Якоби)

Запись в файл проведена успешно (Значения точности для метода Якоби)

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ

решение системы сходится

ВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ

| k | x1 | x2 | x3 | e1 | e2 | e3 |

| 0 | 1.0000 |-5.0000 | 2.0000 | - | - | - |

| 1 |-1.2000 |-4.3000 | 1.6200 | 2.2000 | 0.7000 | 0.3800 |

| 2 |-0.8320 |-4.3980 | 1.4532 | 0.3680 | 0.0980 | 0.1668 |

| 3 |-0.7515 |-4.4613 | 1.4084 | 0.0805 | 0.0633 | 0.0448 |

| 4 |-0.7373 |-4.4801 | 1.3989 | 0.0143 | 0.0189 | 0.0095 |

| 5 |-0.7354 |-4.4844 | 1.3973 | 0.0019 | 0.0043 | 0.0016 |

Запись в файл проведена успешно (Значения x для метода Зейделя)

Запись в файл проведена успешно (Значения точности для метода Зейделя)

Подсчёт невязки:

Невязка в 1-й строке = -0.0005906943999995917

Невязка в 2-й строке = -0.0032329216000022143

Невязка в 3-й строке = 0.0

Норма невязки: 0.0032329216000022143

Запись в файл проведена успешно (Итоговые значения X, полученные с заданной точностью)

Process finished with exit code 0

**Тест 2 (на случайных данных не сходится 3x3 e=0.01 до 200 не сошлось в обоих методах)**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab2/main.py

Использовать данные из файлов на диске - 1

Ввести свои данные - 2 :2

Введите размерность матрицы: 3

Введите точность, например (0.01): 0.01

Исходная система:

( 20.00 5.00 1.00 ) = ( 1.00)

( 6.00 11.00 9.00 ) = ( 15.00)

( 5.00 19.00 6.00 ) = ( 4.00)

Запись в файл проведена успешно (Расширенная матрица)

Решаем:

ПРОВЕРКА МАТРИЦЫ НА ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ

Диаг. элемент 0-й строки = 20; сумма недиаг. элементов строки = 6

В строке выполняется диагональное преобладание

Диаг. элемент 1-й строки = 11; сумма недиаг. элементов строки = 15

В строке не выполняется диагональное преобладание

Диаг. элемент 2-й строки = 6; сумма недиаг. элементов строки = 24

В строке не выполняется диагональное преобладание

В матрице не выполняется диагональное преобладание

ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ НЕ ВЫПОЛНЯЕТСЯ

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ

k > 200 до сих пор не сходится!

Запись в файл проведена успешно (Значения x для метода Якоби)

Запись в файл проведена успешно (Значения точности для метода Якоби)

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ

k > 200 до сих пор не сходится!

Запись в файл проведена успешно (Значения x для метода Зейделя)

Запись в файл проведена успешно (Значения точности для метода Зейделя)

Подсчёт невязки:

Невязка в 1-й строке = 6.189665977677475e+83

Невязка в 2-й строке = -9.380266003732556e+84

Невязка в 3-й строке = -4.0

Норма невязки: 9.380266003732556e+84

Запись в файл проведена успешно (Итоговые значения X, полученные с заданной точностью)

Process finished with exit code 0

**Тест 3 (на случайных данных 3x3 e=0.1, Якоби до 200 не сошлось, Зейделя до 200 сошлось.**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab2/main.py

Использовать данные из файлов на диске - 1

Ввести свои данные - 2 :2

Введите размерность матрицы: 3

Введите точность, например (0.01): 0.1

Исходная система:

( 6.00 20.00 10.00 ) = ( 13.00)

( 8.00 20.00 7.00 ) = ( 4.00)

( 2.00 15.00 7.00 ) = ( 1.00)

Запись в файл проведена успешно (Расширенная матрица)

Решаем:

ПРОВЕРКА МАТРИЦЫ НА ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ

Диаг. элемент 0-й строки = 6; сумма недиаг. элементов строки = 30

В строке не выполняется диагональное преобладание

Диаг. элемент 1-й строки = 20; сумма недиаг. элементов строки = 15

В строке выполняется диагональное преобладание

Диаг. элемент 2-й строки = 7; сумма недиаг. элементов строки = 17

В строке не выполняется диагональное преобладание

В матрице не выполняется диагональное преобладание

ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ НЕ ВЫПОЛНЯЕТСЯ

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ

k > 200 до сих пор не сходится!

Запись в файл проведена успешно (Значения x для метода Якоби)

Запись в файл проведена успешно (Значения точности для метода Якоби)

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ

решение системы сходится

ВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ

| k | x1 | x2 | x3 | e1 | e2 | e3 |

| 0 | 2.1667 | 0.2000 | 0.1429 | - | - | - |

| 1 | 1.2619 |-0.3548 | 0.5425 | 0.9048 | 0.5548 | 0.3997 |

| 2 | 2.4450 |-0.9679 | 1.5183 | 1.1831 | 0.6131 | 0.9758 |

| 3 | 2.8624 |-1.4764 | 2.4887 | 0.4174 | 0.5085 | 0.9704 |

| 4 | 2.9401 |-1.8471 | 3.2609 | 0.0777 | 0.3707 | 0.7722 |

Запись в файл проведена успешно (Значения x для метода Зейделя)

Запись в файл проведена успешно (Значения точности для метода Зейделя)

Подсчёт невязки:

Невязка в 1-й строке = 0.3076004153821117

Невязка в 2-й строке = 5.405215134286898

Невязка в 3-й строке = 0.0

Норма невязки: 5.405215134286898

Запись в файл проведена успешно (Итоговые значения X, полученные с заданной точностью)

Process finished with exit code 0

**Тест 4 (из файла 3x3 e=0.0001 достаточное условие не выполняется, но решение все равно сходится 3)**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab2/main.py

Использовать данные из файлов на диске - 1

Ввести свои данные - 2 :1

Чтение матрицы из файла проведено успешно

Чтение точности из файла проведено успешно

Исходная система:

( 3.10 0.60 2.40 ) = ( 0.34)

( 1.00 -3.40 0.70 ) = ( 1.76)

( -2.00 2.74 3.20 ) = ( 3.42)

Запись в файл проведена успешно (Расширенная матрица)

Решаем:

ПРОВЕРКА МАТРИЦЫ НА ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ

Диаг. элемент 0-й строки = 3.1; сумма недиаг. элементов строки = 3.0

В строке выполняется диагональное преобладание

Диаг. элемент 1-й строки = -3.4; сумма недиаг. элементов строки = 1.7

В строке выполняется диагональное преобладание

Диаг. элемент 2-й строки = 3.2; сумма недиаг. элементов строки = 4.74

В строке не выполняется диагональное преобладание

В матрице не выполняется диагональное преобладание

ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ НЕ ВЫПОЛНЯЕТСЯ

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ

решение системы сходится

ВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ

| k | x1 | x2 | x3 | e1 | e2 | e3 |

| 0 | 0.1097 |-0.5176 | 1.0687 | - | - | - |

| 1 |-0.6176 |-0.2654 | 1.5805 | 0.7272 | 0.2523 | 0.5118 |

| 2 |-1.0626 |-0.3739 | 0.9100 | 0.4451 | 0.1085 | 0.6705 |

| 3 |-0.5225 |-0.6428 | 0.7248 | 0.5401 | 0.2690 | 0.1852 |

| 4 |-0.3270 |-0.5221 | 1.2926 | 0.1955 | 0.1207 | 0.5679 |

| 5 |-0.7900 |-0.3477 | 1.3114 | 0.4630 | 0.1744 | 0.0188 |

| 6 |-0.8383 |-0.4800 | 0.8727 | 0.0483 | 0.1323 | 0.4387 |

| 7 |-0.4731 |-0.5845 | 0.9558 | 0.3653 | 0.1045 | 0.0831 |

| 8 |-0.5172 |-0.4600 | 1.2736 | 0.0441 | 0.1245 | 0.3178 |

| 9 |-0.7873 |-0.4075 | 1.1394 | 0.2701 | 0.0525 | 0.1342 |

| 10 |-0.6936 |-0.5146 | 0.9256 | 0.0937 | 0.1071 | 0.2138 |

| 11 |-0.5073 |-0.5311 | 1.0759 | 0.1862 | 0.0164 | 0.1503 |

| 12 |-0.6205 |-0.4454 | 1.2064 | 0.1132 | 0.0857 | 0.1305 |

| 13 |-0.7381 |-0.4518 | 1.0623 | 0.1176 | 0.0064 | 0.1441 |

| 14 |-0.6253 |-0.5160 | 0.9943 | 0.1128 | 0.0643 | 0.0680 |

| 15 |-0.5602 |-0.4968 | 1.1198 | 0.0651 | 0.0192 | 0.1255 |

| 16 |-0.6611 |-0.4519 | 1.1440 | 0.1009 | 0.0450 | 0.0242 |

| 17 |-0.6886 |-0.4765 | 1.0425 | 0.0275 | 0.0247 | 0.1016 |

| 18 |-0.6052 |-0.5055 | 1.0464 | 0.0834 | 0.0290 | 0.0040 |

| 19 |-0.6026 |-0.4802 | 1.1234 | 0.0025 | 0.0254 | 0.0770 |

| 20 |-0.6671 |-0.4636 | 1.1033 | 0.0645 | 0.0166 | 0.0201 |

| 21 |-0.6547 |-0.4867 | 1.0488 | 0.0124 | 0.0231 | 0.0545 |

| 22 |-0.6081 |-0.4943 | 1.0763 | 0.0467 | 0.0076 | 0.0275 |

| 23 |-0.6279 |-0.4749 | 1.1119 | 0.0198 | 0.0194 | 0.0357 |

| 24 |-0.6593 |-0.4734 | 1.0829 | 0.0314 | 0.0015 | 0.0290 |

| 25 |-0.6371 |-0.4886 | 1.0620 | 0.0222 | 0.0152 | 0.0209 |

| 26 |-0.6180 |-0.4864 | 1.0889 | 0.0191 | 0.0022 | 0.0269 |

| 27 |-0.6392 |-0.4752 | 1.0990 | 0.0212 | 0.0112 | 0.0101 |

| 28 |-0.6492 |-0.4794 | 1.0761 | 0.0099 | 0.0042 | 0.0228 |

| 29 |-0.6307 |-0.4870 | 1.0735 | 0.0185 | 0.0076 | 0.0026 |

| 30 |-0.6272 |-0.4821 | 1.0916 | 0.0035 | 0.0049 | 0.0181 |

| 31 |-0.6421 |-0.4774 | 1.0896 | 0.0149 | 0.0048 | 0.0020 |

| 32 |-0.6415 |-0.4822 | 1.0762 | 0.0006 | 0.0048 | 0.0134 |

| 33 |-0.6302 |-0.4848 | 1.0807 | 0.0113 | 0.0026 | 0.0045 |

| 34 |-0.6332 |-0.4805 | 1.0900 | 0.0030 | 0.0043 | 0.0093 |

| 35 |-0.6412 |-0.4795 | 1.0845 | 0.0080 | 0.0010 | 0.0055 |

| 36 |-0.6371 |-0.4830 | 1.0786 | 0.0041 | 0.0035 | 0.0059 |

| 37 |-0.6319 |-0.4830 | 1.0841 | 0.0052 | 0.0000 | 0.0055 |

| 38 |-0.6361 |-0.4803 | 1.0874 | 0.0043 | 0.0027 | 0.0033 |

| 39 |-0.6392 |-0.4809 | 1.0824 | 0.0031 | 0.0006 | 0.0050 |

| 40 |-0.6352 |-0.4828 | 1.0810 | 0.0040 | 0.0019 | 0.0014 |

| 41 |-0.6338 |-0.4819 | 1.0851 | 0.0015 | 0.0009 | 0.0041 |

| 42 |-0.6371 |-0.4806 | 1.0853 | 0.0034 | 0.0013 | 0.0002 |

| 43 |-0.6375 |-0.4816 | 1.0821 | 0.0004 | 0.0010 | 0.0032 |

| 44 |-0.6349 |-0.4824 | 1.0827 | 0.0027 | 0.0008 | 0.0006 |

| 45 |-0.6352 |-0.4815 | 1.0850 | 0.0003 | 0.0009 | 0.0023 |

| 46 |-0.6371 |-0.4811 | 1.0840 | 0.0020 | 0.0004 | 0.0010 |

| 47 |-0.6365 |-0.4819 | 1.0825 | 0.0007 | 0.0008 | 0.0016 |

| 48 |-0.6351 |-0.4820 | 1.0835 | 0.0014 | 0.0001 | 0.0011 |

| 49 |-0.6359 |-0.4814 | 1.0845 | 0.0008 | 0.0006 | 0.0010 |

| 50 |-0.6368 |-0.4814 | 1.0835 | 0.0009 | 0.0000 | 0.0010 |

| 51 |-0.6360 |-0.4819 | 1.0830 | 0.0008 | 0.0005 | 0.0005 |

| 52 |-0.6355 |-0.4817 | 1.0839 | 0.0005 | 0.0001 | 0.0009 |

| 53 |-0.6362 |-0.4814 | 1.0841 | 0.0007 | 0.0003 | 0.0002 |

| 54 |-0.6364 |-0.4816 | 1.0833 | 0.0002 | 0.0002 | 0.0007 |

| 55 |-0.6358 |-0.4818 | 1.0833 | 0.0006 | 0.0002 | 0.0000 |

| 56 |-0.6358 |-0.4816 | 1.0839 | 0.0000 | 0.0002 | 0.0006 |

Запись в файл проведена успешно (Значения x для метода Якоби)

Запись в файл проведена успешно (Значения точности для метода Якоби)

РЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ

решение системы сходится

ВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ

| k | x1 | x2 | x3 | e1 | e2 | e3 |

| 0 | 0.1097 |-0.5176 | 1.0687 | - | - | - |

| 1 |-0.6176 |-0.4792 | 1.0931 | 0.7272 | 0.0384 | 0.0244 |

| 2 |-0.6439 |-0.4820 | 1.0790 | 0.0263 | 0.0027 | 0.0141 |

| 3 |-0.6324 |-0.4815 | 1.0858 | 0.0115 | 0.0005 | 0.0068 |

| 4 |-0.6377 |-0.4817 | 1.0826 | 0.0053 | 0.0002 | 0.0032 |

| 5 |-0.6352 |-0.4816 | 1.0841 | 0.0025 | 0.0001 | 0.0015 |

| 6 |-0.6364 |-0.4816 | 1.0834 | 0.0012 | 0.0000 | 0.0007 |

| 7 |-0.6359 |-0.4816 | 1.0837 | 0.0005 | 0.0000 | 0.0003 |

| 8 |-0.6361 |-0.4816 | 1.0836 | 0.0003 | 0.0000 | 0.0002 |

| 9 |-0.6360 |-0.4816 | 1.0836 | 0.0001 | 0.0000 | 0.0001 |

| 10 |-0.6361 |-0.4816 | 1.0836 | 0.0001 | 0.0000 | 0.0000 |

Запись в файл проведена успешно (Значения x для метода Зейделя)

Запись в файл проведена успешно (Значения точности для метода Зейделя)

Подсчёт невязки:

Невязка в 1-й строке = -8.231831009392065e-05

Невязка в 2-й строке = -2.3695808790025197e-05

Невязка в 3-й строке = 4.440892098500626e-16

Норма невязки: 8.231831009392065e-05

Запись в файл проведена успешно (Итоговые значения X, полученные с заданной точностью)

Process finished with exit code 0

**Код программы:**

**import** csv  
**import** random  
  
*# ----- ----- ----- ----- ----- ----- ----- ----- ----- -----*INPUT\_FILENAME = **'input.csv'**OUTPUT\_FILENAME = **'output.csv'**EPSILON\_INPUT\_FILENAME = **'epsilon\_input.dat'**MIN\_RAND = 1 *# минимальное число для элемента матрицы (при генерации)*MAX\_RAND = 20 *# максимальное число для элемента матрицы (при генерации)***def** read\_input\_data(filename, expended\_matrix):  
 *"""чтение расширенной матрицы из файла"""* **with** open(filename) **as** File:  
 reader = csv.reader(File, delimiter=**';'**)  
 **for** row\_index, row **in** enumerate(reader):  
 expended\_matrix.append([])  
 **for** item **in** row:  
 expended\_matrix[row\_index].append(float(item))  
 print(**'Чтение матрицы из файла проведено успешно'**)  
  
  
**def** read\_input\_epsilon(filename):  
 *"""чтение точности из файла"""* epsilon = 0  
 **with** open(filename) **as** f:  
 **for** line **in** f:  
 epsilon = float(line)  
 print(**'Чтение точности из файла проведено успешно'**)  
 **return** epsilon  
  
  
**def** write\_output\_matrix(filename, matrix, title=**""**):  
 *"""запись результатов в файл"""* **with** open(filename, mode=**"a"**, encoding=**'utf-8'**) **as** FILE:  
 file\_writer = csv.writer(FILE, delimiter=**","**, lineterminator=**"\r"**)  
 file\_writer.writerow([**'\n'** + title])  
 **for** row **in** matrix:  
 file\_writer.writerow(row)  
 print(**f'Запись в файл проведена успешно ({**title**})'**)  
  
  
**def** write\_output\_list(filename, \_list, title=**""**):  
 *"""запись результатов в файл"""* **with** open(filename, mode=**"a"**, encoding=**'utf-8'**) **as** FILE:  
 file\_writer = csv.writer(FILE, delimiter=**","**, lineterminator=**"\r"**)  
 file\_writer.writerow([**'\n'** + title])  
 file\_writer.writerow([\_list])  
 print(**f'Запись в файл проведена успешно ({**title**})'**)  
  
  
**def** format\_print(n, a, b=**None**):  
 *"""вывод системы на экран"""* **for** row\_ind **in** range(n):  
 print(**"("**, end=**''**)  
 **for** col\_ind **in** range(n):  
 print(**"\t{0:10.2f}"**.format(a[row\_ind][col\_ind]), end=**''**)  
 **if** b **is not None**:  
 print(**"\t) = (\t{1:10.2f})"**.format(row\_ind + 1, b[row\_ind]))  
 **else**:  
 print(**"\t)"**)  
  
  
**def** check\_diagonal\_dominance(factors\_at\_unknowns):  
 *""" проверка матрицы на диагональное преобладание """* print(**'\nПРОВЕРКА МАТРИЦЫ НА ДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ'**)  
 cols = len(factors\_at\_unknowns[0])  
 rows = len(factors\_at\_unknowns)  
 is\_diagonal\_dominant = **True  
 for** i **in** range(rows):  
 row\_off\_diagonal\_sum = 0  
 **for** j **in** range(cols):  
 **if** i != j:  
 row\_off\_diagonal\_sum += abs(factors\_at\_unknowns[i][j])  
 diagonal\_elem = abs(factors\_at\_unknowns[i][i])  
 print(  
 **f'Диаг. элемент {**i**}-й строки = {**factors\_at\_unknowns[i][i]**}; сумма недиаг. элементов строки = {**row\_off\_diagonal\_sum**}'**)  
 **if** diagonal\_elem > row\_off\_diagonal\_sum:  
 print(**'В строке выполняется диагональное преобладание'**)  
 **else**:  
 is\_diagonal\_dominant = **False** print(**'В строке не выполняется диагональное преобладание'**)  
 **if** is\_diagonal\_dominant:  
 print(**'В матрице выполняется диагональное преобладание'**)  
 **else**:  
 print(**'В матрице не выполняется диагональное преобладание'**)  
 **return** is\_diagonal\_dominant  
  
  
**def** count\_null\_x(b, diagonal\_a):  
 *""" посчет x для нулевого приближения  
 b - свободный член строки,  
 diagonal\_a - свободный член диагонального элемента строки  
 """* **return** b / diagonal\_a  
  
  
**def** get\_null\_x\_list(factors\_at\_unknowns, free\_factors):  
 *""" подсчет всех x для нулевого приближения и их возврат  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов """* null\_x\_list = []  
 *# print('\nПОДСЧЕТ ВСЕХ X ДЛЯ НУЛЕВОГО ПРИБЛИЖЕНИЯ (k=0)')* **for** i **in** range(len(free\_factors)):  
 null\_x = count\_null\_x(free\_factors[i], factors\_at\_unknowns[i][i])  
 null\_x\_list.append(null\_x)  
  
 *# for i in range(len(null\_x\_list)):  
 # print(f'x{i + 1} (k=0) = {null\_x\_list[i]}')* **return** null\_x\_list  
  
  
**def** func\_x(b, a\_array, diagonal\_x\_index):  
 *""" динамическое создание функции для подсчёта x (для строки)  
 b - свободный член строки,  
 a\_array - массив свободных членов строки,  
 diagonal\_x\_index - индекс подсчитываемого x (этот элем. на главной диагонали)  
 """* **def** created\_func\_x(x\_array, k):  
 *""" формула подсчета x с предустановленными свободными членами """  
 # print(f'подсчёт x{diagonal\_x\_index + 1} (k={k})')* a\_array\_len = len(a\_array)  
 x = b  
 **for** i **in** range(0, a\_array\_len):  
 **if** i != diagonal\_x\_index:  
 x -= a\_array[i] \* x\_array[i] *# вычитаем все a[i]\*x[i], кроме диагонального* x /= a\_array[diagonal\_x\_index] *# делим на коэф. a текущего x* **return** x  
 **return** created\_func\_x  
  
  
**def** create\_x\_functions(factors\_at\_unknowns, free\_factors):  
 *""" динамическое создание формул (функций) для подсчёта x (для строки)  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов  
 """* x\_counting\_functions = []  
 rows = len(free\_factors)  
 **for** i **in** range(rows):  
 x\_counting\_func = func\_x(free\_factors[i], factors\_at\_unknowns[i], i)  
 x\_counting\_functions.append(x\_counting\_func)  
 **return** x\_counting\_functions  
  
  
**def** count\_x\_error(x\_prev, x):  
 *""" подсчитывает погрешность для x  
 |x (k) - x (k-1)|/ |x (k)|  
 """* **return** abs(x - x\_prev)  
  
  
**def** check\_accuracy\_is\_done(epsilon, errors):  
 *""" проверка выполнения всеми x заданной погрешности,  
 если все погрешности меньше, чем заданная - True, иначе - False """* **for** i **in** range(len(errors)):  
 **if** epsilon < errors[i]:  
 *# print('проверка достижений погрешности: требуемая погрешность пока не достигнута')* **return False** *# print('проверка достижений погрешности: требуемая погрешность достигнута')* **return True  
  
  
def** format\_print\_x\_and\_errors(x\_lists, x\_errors):  
 *""" вывод таблицы вида | k | x1 x2 ... xn | e1 e2 ... en |"""* print(**'\nВЫВОД X-ОВ И ИХ ПОГРЕШНОСТЕЙ'**)  
 print(**"|{0:^4}"**.format(**'k'**), end=**''**)  
 **for** i **in** range(len(x\_lists[0])):  
 print(**'|{0:^8}'**.format(**f'x{**i + 1**}'**), end=**''**)  
 **for** i **in** range(len(x\_lists[0])):  
 print(**'|{0:^8}'**.format(**f'e{**i + 1**}'**), end=**''**)  
 print(**'|'**)  
 **for** k **in** range(len(x\_lists)):  
 print(**"|{0:^4}"**.format(k), end=**''**)  
 **for** i **in** range(len(x\_lists[0])):  
 print(**'|{0:^8.4f}'**.format(x\_lists[k][i]), end=**''**)  
 **for** i **in** range(len(x\_lists[0])):  
 **if** k == 0:  
 print(**'|{0:^8}'**.format(x\_errors[k][i]), end=**''**)  
 **else**:  
 print(**'|{0:^8.4f}'**.format(x\_errors[k][i]), end=**''**)  
 print(**'|'**)  
  
  
**def** solve\_by\_yacobi(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
 *""" решение системы методом Якоби  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов  
 x\_lists - таблица x-ов по всем приближениям  
 x\_errors - таблица погрешностей (по x-ам) для всех приближений  
 epsilon - величина допустимой погрешности """* print(**'\nРЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЯКОБИ'**)  
 cols = len(factors\_at\_unknowns[0])  
 rows = len(factors\_at\_unknowns)  
  
 null\_x\_list = get\_null\_x\_list(factors\_at\_unknowns, free\_factors) *# подсчет всех x для нулевого приближения* x\_lists.append(null\_x\_list) *# добавление x-ов (0-го приближения) в таблицу x-ов всех приближений* empty\_error\_list = [**"-" for** \_ **in** range(cols)] *# пустая строка для первой строки таблицы погрешностей* x\_errors.append(empty\_error\_list) *# добавление пустой строки в таблицу погрешностей* x\_counting\_functions = create\_x\_functions(factors\_at\_unknowns, free\_factors) *# создание формул для подсчёта x-ов  
  
 # цикл по приближениям (k)* accuracy\_is\_done = **False** k = 0  
 **while** accuracy\_is\_done **is not True**:  
 *# счетчик номера приближения* k += 1  
 *# print(f'\nПОДСЧЕТ X И ПОГРЕШНОСТЕЙ ДЛЯ {k}-ОГО ПРИБЛИЖЕНИЯ')  
 # подсчёт x-ов их погрешностей* cur\_x\_list = []  
 cur\_x\_error\_list = []  
 **for** i **in** range(rows):  
 x = x\_counting\_functions[i](x\_lists[k - 1], k)  
 cur\_x\_list.append(x)  
 x\_prev = x\_lists[k - 1][i] *# соотвествующий предыдущий x (k-1)* x\_error = count\_x\_error(x\_prev, x)  
 cur\_x\_error\_list.append(x\_error)  
 x\_lists.append(cur\_x\_list)  
 x\_errors.append(cur\_x\_error\_list)  
 *# print(f'k: {k};\nx: {cur\_x\_list};\nпогрешности: {cur\_x\_error\_list}')  
 # проверка достижения требуемой погрешности* accuracy\_is\_done = check\_accuracy\_is\_done(epsilon, cur\_x\_error\_list)  
 **if** k > 200:  
 print(**'k > 200 до сих пор не сходится!'**)  
 **return False** print(**'\nрешение системы сходится'**)  
 **return True  
  
  
def** solve\_by\_zeydel(factors\_at\_unknowns, free\_factors, x\_lists, x\_errors, epsilon):  
 *""" решение системы методом Зейделя  
 factors\_at\_unknowns - двумерных массив членов при неизвестных  
 free\_factors - столбец свободных членов  
 x\_lists - таблица x-ов по всем приближениям  
 x\_errors - таблица погрешностей (по x-ам) для всех приближений  
 epsilon - величина допустимой погрешности """* print(**'\nРЕШЕНИЕ СИСТЕМЫ МЕТОДОМ ЗЕЙНДЕЛЯ'**)  
 cols = len(factors\_at\_unknowns[0])  
 rows = len(factors\_at\_unknowns)  
  
 null\_x\_list = get\_null\_x\_list(factors\_at\_unknowns, free\_factors) *# подсчет всех x для нулевого приближения* x\_lists.append(null\_x\_list) *# добавление x-ов (0-го приближения) в таблицу x-ов всех приближений* empty\_error\_list = [**"-" for** \_ **in** range(cols)] *# пустая строка для первой строки таблицы погрешностей* x\_errors.append(empty\_error\_list) *# добавление пустой строки в таблицу погрешностей* x\_counting\_functions = create\_x\_functions(factors\_at\_unknowns, free\_factors) *# создание формул для подсчёта x-ов  
  
 # цикл по приближениям (k)* accuracy\_is\_done = **False** k = 0  
 **while** accuracy\_is\_done **is not True**:  
 *# счетчик номера приближения* k += 1  
 *# print(f'\nПОДСЧЕТ X И ПОГРЕШНОСТЕЙ ДЛЯ {k}-ОГО ПРИБЛИЖЕНИЯ')  
 # подсчёт x-ов их погрешностей* cur\_x\_list = []  
 cur\_x\_error\_list = []  
 x\_list\_tmp = x\_lists[k - 1].copy() *# для хранения списка иксов* **for** i **in** range(rows):  
 x = x\_counting\_functions[i](x\_list\_tmp, k)  
 cur\_x\_list.append(x)  
 x\_prev = x\_lists[k - 1][i] *# соотвествующий предыдущий x (k-1)* x\_error = count\_x\_error(x\_prev, x)  
 cur\_x\_error\_list.append(x\_error)  
 x\_list\_tmp[i] = x  
 x\_lists.append(cur\_x\_list)  
 x\_errors.append(cur\_x\_error\_list)  
 *# print(f'k: {k};\nx: {cur\_x\_list};\nпогрешности: {cur\_x\_error\_list}')  
 # проверка достижения требуемой погрешности* accuracy\_is\_done = check\_accuracy\_is\_done(epsilon, cur\_x\_error\_list)  
 **if** k > 200:  
 print(**'k > 200 до сих пор не сходится!'**)  
 **return False** print(**'\nрешение системы сходится'**)  
 **return True  
  
  
def** check\_discrepancy(a\_array, b\_array, x\_array):  
 *"""проверка на соотвествие (невязка)  
 a\_array - квадртная матрица  
 b\_array - столбец свободных членов  
 x\_array - итоговый ответ с заданной точностью """* print(**"\nПодсчёт невязки:"**)  
 text\_discrepancy = **''** max\_sum = 0  
 **for** row\_ind **in** range(len(b\_array)):  
 \_sum = 0  
 **for** col\_ind **in** range(len(a\_array[row\_ind])):  
 \_sum += a\_array[row\_ind][col\_ind] \* x\_array[col\_ind]  
 \_sum = \_sum - b\_array[row\_ind]  
 text\_discrepancy += **f'Невязка в {**row\_ind + 1**}-й строке = {**\_sum**}\n'  
 if** max\_sum < abs(\_sum):  
 max\_sum = abs(\_sum)  
 print(**f'Невязка в {**row\_ind + 1**}-й строке = {**\_sum**}'**)  
 print(**f'Норма невязки: {**max\_sum**}'**)  
 text\_discrepancy += **f'\nНорма невязки: {**max\_sum**}'  
 return** text\_discrepancy  
  
  
**def** generate\_rand\_matrix(i, j):  
 *""" генерация матрицы определенной размерности со случайными элементами """* matrix = [[**None for** j **in** range(j)] **for** i **in** range(i)]  
 **for** \_i **in** range(i):  
 **for** \_j **in** range(j):  
 matrix[\_i][\_j] = random.randint(MIN\_RAND, MAX\_RAND)  
 **return** matrix  
  
  
**def** main():  
 epsilon = 0.01  
 expanded\_matrix = list()  
 choice = input(**'Использовать данные из файлов на диске - 1\nВвести свои данные - 2\t:'**)  
 **if** choice == **'2'**:  
 n = int(input(**'Введите размерность матрицы:\t'**))  
 expanded\_matrix = generate\_rand\_matrix(n, n + 1)  
 epsilon = float(input(**'Введите точность, например (0.01):\t'**))  
 **else**:  
 **with** open(OUTPUT\_FILENAME, mode=**"w"**, encoding=**'utf-8'**):  
 **pass  
 try**:  
 read\_input\_data(INPUT\_FILENAME, expanded\_matrix)  
 epsilon = read\_input\_epsilon(EPSILON\_INPUT\_FILENAME)  
 **except** Exception:  
 print(**"Ошибка при чтении файла"**)  
 **return** free\_factors = []  
 **for** \_row **in** expanded\_matrix:  
 free\_factors.append(\_row.pop())  
 factors\_at\_unknowns = expanded\_matrix  
 n = len(factors\_at\_unknowns)  
 print(**"\nИсходная система:"**)  
 format\_print(n, factors\_at\_unknowns, free\_factors)  
  
 write\_output\_matrix(OUTPUT\_FILENAME, expanded\_matrix, **'Расширенная матрица'**)  
  
 print(**"\nРешаем:"**)  
 is\_diagonal\_dominance = check\_diagonal\_dominance(factors\_at\_unknowns)  
 **if** is\_diagonal\_dominance:  
 print(**'\nДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ ВЫПОЛНЯЕТСЯ'**)  
 **else**:  
 print(**'\nДИАГОНАЛЬНОЕ ПРЕОБЛАДАНИЕ НЕ ВЫПОЛНЯЕТСЯ'**)  
 x\_lists\_yacobi = []  
 x\_errors\_yacobi = []  
 yacobi\_is\_true = solve\_by\_yacobi(factors\_at\_unknowns, free\_factors, x\_lists\_yacobi, x\_errors\_yacobi, epsilon)  
 **if** yacobi\_is\_true:  
 format\_print\_x\_and\_errors(x\_lists\_yacobi, x\_errors\_yacobi)  
  
 write\_output\_matrix(OUTPUT\_FILENAME, x\_lists\_yacobi, title=**'Значения x для метода Якоби'**)  
 write\_output\_matrix(OUTPUT\_FILENAME, x\_errors\_yacobi, title=**'Значения точности для метода Якоби'**)  
  
 x\_lists\_zeydel = []  
 x\_errors\_zeydel = []  
 zeydel\_is\_true = solve\_by\_zeydel(factors\_at\_unknowns, free\_factors, x\_lists\_zeydel, x\_errors\_zeydel, epsilon)  
 **if** zeydel\_is\_true:  
 format\_print\_x\_and\_errors(x\_lists\_zeydel, x\_errors\_zeydel)  
  
 write\_output\_matrix(OUTPUT\_FILENAME, x\_lists\_zeydel, title=**'Значения x для метода Зейделя'**)  
 write\_output\_matrix(OUTPUT\_FILENAME, x\_errors\_zeydel, title=**'Значения точности для метода Зейделя'**)  
  
 x\_answers = x\_lists\_zeydel[len(x\_lists\_zeydel) - 1]  
  
 check\_discrepancy(factors\_at\_unknowns, free\_factors, x\_answers)  
  
 write\_output\_list(OUTPUT\_FILENAME, x\_answers, title=**'Итоговые значения X, полученные с заданной точностью'**)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()