Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Студент: Карандашева Анастасия Денисовна

P3268

1. Цель работы

Реализовать алгоритм решения системы линейных уравнений методом простых итераций с заданной размерностью (до 20), точностью и возможностью ввода коэффициентов как из файла, так и с клавиатуры, оценить погрешность расчётов.

2. Описание метода, расчетные формулы

А. МЕТОД ПРОСТЫХ ИТЕРАЦИЙ

Методика решения задачи

Шаг 1. Преобразовать систему $Ax = b$ к виду $x = \alpha x + \beta$ одним из описанных способов.

Шаг 2. Задать начальное приближение решения $x^{(0)}$ произвольно или положить $x^{(0)} = \beta$, а также малое положительное число ε (точность). Положить $k = 0$.

Шаг 3. Вычислить следующее приближение $x^{(k+1)}$ по формуле

$$x^{(k+1)} = \alpha x^{(k)} + \beta .$$

Шаг 4. Если выполнено условие окончания $\left\| x^{(k+1)} - x^{(k)} \right\| < \varepsilon$, процесс завершить и положить $x_* \cong x^{(k+1)}$. Иначе положить $k = k + 1$ и перейти к п.3.

Пример 1. Методом простых итераций с точностью $\varepsilon = 0{,}01$ решить систему линейных алгебраических уравнений:

$$\begin{aligned} 2x_1 + 2x_2 + 10x_3 &= 14, \\ 10x_1 + x_2 + x_3 &= 12, \\ 2x_1 + 10x_2 + x_3 &= 13. \end{aligned}$$

□ 1. Так как $|2| < |2| + |10|$, $|1| < |10| + |1|$, $|1| < |2| + |10|$, условие преобладания диагональных элементов не выполняется. Переставим уравнения местами так, чтобы выполнялось условие преобладания диагональных элементов:

$$\begin{aligned} 10x_1 + x_2 + x_3 &= 12, \\ 2x_1 + 10x_2 + x_3 &= 13, \\ 2x_1 + 2x_2 + 10x_3 &= 14. \end{aligned}$$

Получаем $|10| > |1| + |1|$, $|10| > |2| + |1|$, $|10| > |2| + |2|$. Выразим из первого уравнения x_1, из второго x_2, из третьего x_3:

$$
\begin{aligned}
x_1 &= -0{,}1 \cdot x_2 - 0{,}1 \cdot x_3 + 1{,}2, \\
x_2 &= -0{,}2 \cdot x_1 - 0{,}1 \cdot x_3 + 1{,}3, \\
x_3 &= -0{,}2 \cdot x_1 - 0{,}2 \cdot x_2 + 1{,}4;
\end{aligned}
\qquad
\alpha = \begin{pmatrix} 0 & -0{,}1 & -0{,}1 \\ -0{,}2 & 0 & -0{,}1 \\ -0{,}2 & -0{,}2 & 0 \end{pmatrix}; \quad \beta = \begin{pmatrix} 1{,}2 \\ 1{,}3 \\ 1{,}4 \end{pmatrix}.
$$

Заметим, что $\|\alpha\|_1 = \max\{0{,}2; 0{,}3; 0{,}4\} = 0{,}4 < 1$, следовательно, условие сходимости (теорема) выполнено.

2. Зададим $x^{(0)} = \beta = \begin{pmatrix} 1{,}2 \\ 1{,}3 \\ 1{,}4 \end{pmatrix}$. В поставленной задаче $\varepsilon = 0{,}01$.

3. Выполним расчеты по формуле $x^{(k+1)} = \alpha x^{(k)} + \beta$:

$$
x^{(k+1)} = \begin{pmatrix} 0 & -0{,}1 & -0{,}1 \\ -0{,}2 & 0 & -0{,}1 \\ -0{,}2 & -0{,}2 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix} + \begin{pmatrix} 1{,}2 \\ 1{,}3 \\ 1{,}4 \end{pmatrix}, \quad k = 0, 1, \ldots,
$$

или

$$
\begin{aligned}
x_1^{(k+1)} &= -0{,}1 x_2^{(k)} - 0{,}1 x_3^{(k)} + 1{,}2; \\
x_2^{(k+1)} &= -0{,}2 x_1^{(k)} - 0{,}1 x_3^{(k)} + 1{,}3; \qquad k = 0, 1, \ldots, \\
x_3^{(k+1)} &= -0{,}2 x_1^{(k)} - 0{,}2 x_2^{(k)} + 1{,}4;
\end{aligned}
$$

до выполнения условия окончания и результаты занесем в табл. 1.

Таблица 1

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\left\| x^{(k)} - x^{(k-1)} \right\|_1$
0	1,2000	1,3000	1,4000	-
1	0,9300	0,9200	0,900	0,5
2	1,0180	1,0240	1,0300	0,13
3	0,9946	0,9934	0,9916	0,0384
4	1,0015	1,0020	1,0024	0,0108
5	0,9996	0,9995	0,9993	0,0027<ε

4. Расчет закончен, поскольку условие окончания $\left\| x^{(k+1)} - x^{(k)} \right\| = 0{,}0027 < \varepsilon$ выполнено.

Приближенное решение задачи: $x_* \cong (0{,}9996; 0{,}9995; 0{,}9993)^T$. Очевидно, точное решение: $x_* = (1; 1; 1)^T$.

3. Листинг программы

```
def simple_iteration_method(accuracy, matrix):
```

```python
    max_iteration = 1000
    matrix = make_diagonal(matrix)
    solution_x = []
    d = []

    for i in range(len(matrix)):
        d.append(matrix[i][-1] / matrix[i][i])

    last_x = d
    iteration_amount = 0

    print("Результаты итераций:")
    while True:
        iteration_amount += 1
        solution_x = []
        accuracy_vec = []

        for i in range(len(matrix)):
            result = last_x[i] + d[i]

            for j in range(len(matrix[i]) - 1):
                result += - matrix[i][j] / matrix[i][i] * last_x[j]

            solution_x.append(result)
            accuracy_vec.append(abs(last_x[i] - result))
```

```
        cur_accuracy = max(accuracy_vec)

        last_x = solution_x



        if cur_accuracy <= accuracy:

            break

        if iteration_amount >= max_iteration:

            raise ValueError("Программа достигла максимума итераций: "
+ str(max_iteration) + ". Точность на последней итерации: "  +
str(cur_accuracy))

        print(iteration_amount, solution_x)



    return solution_x, iteration_amount, accuracy_vec



def make_diagonal(matrix):

    max_el_index_list = []

    for i in range(len(matrix)):

        if matrix[i] == max(matrix[i][:-1], key=abs):

            max_el_index_list.append(i)

        else:


max_el_index_list.append(matrix[i].index(max(matrix[i][:-1], key=abs)))

    if len(list(set(max_el_index_list))) != len(max_el_index_list):

        print("Достижение диагонального доминирования невозможно. Метод
простых итераций нельзя применить.")

        error_lines = set()

        for i in range(len(max_el_index_list)):

            if max_el_index_list[i] != i:
```

```
            error_lines.add(i)

        print("Эти строки не соответствуют условию диагонального
доминирования", error_lines)

    diagonal_matrix = []



    try:

        for i in range(len(matrix)):

            diagonal_matrix.append(matrix[max_el_index_list.index(i)])

        print("Матрица преобразована, условие диагонального
преобладания выполнено:")

    except ValueError as e:

        print("Матрицу не удалось преобразовать:")

    for line in diagonal_matrix:

        for x in line:

            print(x, end=" ")

        print()



    if len(diagonal_matrix) != len(matrix):

        return matrix

    return diagonal_matrix





def main():

    filename = "file.txt"



    matrix = []
```

```
    n = 0

    accuracy = 0


    ask_input = input(

        "Введите f, чтобы вставить матрицу из файла \"" + filename +
"\" или k, чтобы ввести матрицу с клавиатуры\n")

    if ask_input == "k":

        print("Введите размерность матрицы:")

        n = int(input())


        if n > 20:

            raise ValueError("n не должно быть больше 20")


        print("Введите точность:")

        accuracy = float(input())

        for i in range(n):

            print(str(i+1) + " строка матрицы:")

            line = input()

            matrix.append([float(x) for x in line.strip().split(" ")])


    elif ask_input == "f":

        file = open(filename, "r")


        n = int(file.readline())

        accuracy = float(file.readline())
```

```
        for line in file:
            matrix.append([float(x) for x in line.strip().split(" ")])

        file.close()

    else:
        raise ValueError("Введено неверное значение")


    if len(matrix) != n:
        raise ValueError("Неверное количество строк")
    elif len(matrix) != 0:
        solution = simple_iteration_method(accuracy, matrix)
        print("Количество итераций:", solution[1])
        print("Вектор неизвестных:", solution[0])
        print("Вектор погрешностей:", solution[2])

try:
    main()
except ValueError as e:
    print("Ошибка: ", e)
except KeyboardInterrupt as e:
    print(e)
except ZeroDivisionError as e:
```

```
    print("Невозможно применить метод простых итераций для этой матрицы")
```

4. Примеры и результаты работы программы

Методом простых итераций с точностью $\varepsilon = 0{,}01$ решить систему линейных алгебраических уравнений:

$$\begin{cases} 2x_1 + 2x_2 + 10x_3 = 14 \\ 10x_1 + x_2 + x_3 = 12 \\ 2x_1 + 10x_2 + x_3 = 13 \end{cases}$$

Введите f, чтобы вставить матрицу из файла "file.txt" или k, чтобы ввести матрицу с клавиатуры

k

Введите размерность матрицы:

3

Введите точность:

0.01

1 строка матрицы:

2 2 10 14

2 строка матрицы:

10 1 1 12

3 строка матрицы:

2 10 1 13

Матрица преобразована, условие диагонального преобладания выполнено:

10.0 1.0 1.0 12.0

2.0 10.0 1.0 13.0

2.0 2.0 10.0 14.0

Результаты итераций:

```
1 [0.9299999999999998, 0.9200000000000003, 0.8999999999999999]

2 [1.018, 1.0239999999999998, 1.0299999999999998]

3 [0.9945999999999999, 0.9934000000000003, 0.9916]

4 [1.0015, 1.0019199999999997, 1.0024000000000002]

Количество итераций: 5

Вектор неизвестных: [0.9995680000000001, 0.9994600000000003,
0.9993159999999999]

Вектор погрешностей: [0.0019319999999999338, 0.0024599999999993516,
0.0030840000000003087]
```

5. Выводы

Был реализован метод простых итераций на python, позволяющий получить решение системы линейных уравнений, а также вывести количество итераций и погрешности. Погрешность оказалась небольшой, что характерно для данного метода.