Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

Высшего образования

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 1 по Вычислительной Математике

Метод простых итераций

Вариант №4

Группа: P3216

Выполнил:

Исмоилов Ш. Б

Проверил:

Рыбаков С. Д.

Г. Санкт-Петербург

2025

Оглавление

Цель работы

Используя известные методы вычислительной математики, написать программу, осуществляющий решение СЛАУ итерационным методом. Проанализировать полученные результаты, оценить погрешность.

Описание метода, расчётные формулы

Итерационные методы. Метод простой итерации

Рассмотрим систему линейных уравнений с невырожденной матрицей ($\det A \neq 0$):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}\,x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}\,x_n = b_2 \\ \dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}\,x_n = b_n \end{cases} \qquad (5)$$

Приведем систему уравнений к виду (6), выразив неизвестные $x_1, x_2, \dots, x_n$ соответственно из первого, второго и т.д. уравнений системы (5):

$$\begin{cases} x_1 = -\dfrac{a_{12}}{a_{11}}x_2 - \dfrac{a_{13}}{a_{11}}x_3 - \dots - \dfrac{a_{1n}}{a_{11}}x_n + \dfrac{b_1}{a_{11}} \\ x_2 = -\dfrac{a_{21}}{a_{22}}x_1 - \dfrac{a_{23}}{a_{22}}x_3 - \dots - \dfrac{a_{2n}}{a_{22}}x_n + \dfrac{b_2}{a_{22}} \\ \dots\dots \\ x_n = -\dfrac{a_{n1}}{a_{nn}}x_1 - \dfrac{a_{n2}}{a_{nn}}x_2 - \dots - \dfrac{a_{n-1n-1}}{a_{nn}}x_{n-1} + \dfrac{b_n}{a_{nn}} \end{cases} \qquad (6)$$

Обозначим:

$$c_{ij} = \begin{cases} 0, & \text{при } i = j \\ -\dfrac{a_{ij}}{a_{ii}}, & \text{при } i \neq j \end{cases}$$

$$d_i = \frac{b_i}{a_{ii}} \quad i = 1, 2, \dots, n$$

Тогда получим:

$$\begin{cases} x_1 = c_{11}x_1 + c_{12}x_2 + \dots + c_{1n}\,x_n + d_1 \\ x_2 = c_{21}x_1 + c_{22}x_2 + \dots + c_{2n}\,x_n + d_2 \\ \dots\dots \\ x_n = c_{n1}x_1 + c_{n2}x_2 + \dots + c_{nn}\,x_n + d_n \end{cases}$$

Или в векторно-матричном виде: $\boldsymbol{x = Cx + D}$, где x – вектор неизвестных, C – матрица коэффициентов преобразованной системы размерности $n*n$, D – вектор правых частей преобразованной системы.

Систему (6) представим в сокращенном виде:

$$\boldsymbol{x_i = \sum_{j=1}^{n} c_{ij}x_j + d_i}\ , \qquad i = 1, 2, \dots, n$$

$$c_{ij} = \begin{cases} 0, \text{ при } i = j \\ -\frac{a_{ij}}{a_{ii}}, \text{ при } i \neq j \end{cases} \qquad d_i = \frac{b_i}{a_{ii}} \quad i = 1, 2, \dots, n$$

Рабочая формула метода простой итерации:

$$\boldsymbol{x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^k}\ , \qquad \boldsymbol{i = 1, 2, \dots, n}$$

где k – номер итерации.

За начальное (нулевое) приближение выбирают вектор свободных членов: $x^{(0)} = D$ или нулевой вектор: $x^{(0)} = 0$

Следующее приближение: $\vec{x}^{(1)} = c\vec{x}^{(0)} + \vec{d}$, $\vec{x}^{(2)} = c\vec{x}^{(1)} + \vec{d}$ …

Листинг программы

```python
def solve(self) -> (list[float], int, list[float]):
    C = [[-1 * num / line[i] if i != j else 0
          for j, num in enumerate(line)
          ]
         for i, line in enumerate(self.matrix)
         ]
    D = [num / self.matrix[i][i] for i, num in enumerate(self.right_parts)]
    X = D.copy()
    # print(C)
    # print(D)
    iter_count = 0
    while True:
        print("Итерация ", iter_count)
        print("Значения X: ", X)
        iter_count += 1
        if iter_count > self.MAX_ITERATION_COUNT:
            print("Превышено максимальное количество итераций!", file=sys.stderr)
            sys.exit(1)
        X_next = [
            sum(self.mul_vectors(C[i], X)) + D[i]
            for i in range(self.size)
        ]
        if max(map(abs, self.sub_vectors(X_next, X))) < self.accuracy:
            r = []
            for i in range(len(self.matrix)):
                r.append(sum(self.mul_vectors(self.matrix[i], X)) - self.right_parts[i])
            print("Критерий по невязке:")
            print(r)
            return X_next, iter_count, list(map(abs, self.sub_vectors(X_next, X)))
        X = X_next
```

[Полный код на github]

Примеры и результаты работы программы

Работа из файла

main.py   input.csv

```
2,4,10,1
1,7,4,8
11,3,5,2
```

Программу запускаем командой, передаем в аргументы путь к файлу и требуемую точность (0.01)

Результат работы программы:

```
Получена матрица:
     2.0     4.0    10.0     1.0
     1.0     7.0     4.0     8.0
    11.0     3.0     5.0     2.0
Приведение к матрице преобладания диагональных коэффициентов:
    11.0     3.0     5.0     2.0
     1.0     7.0     4.0     8.0
     2.0     4.0    10.0     1.0
Итерация  0
Значения X:  [0.18181818181818182, 1.1428571428571428, 0.1]
Итерация  1
Значения X:  [-0.17532467532467527, 1.0597402597402596, -0.39350649350649347]
Итерация  2
Значения X:  [0.07166469893742625, 1.392764378478664, -0.28883116883116877]
Итерация  3
Значения X:  [-0.06673975375274074, 1.297665710912464, -0.471438691178951]
```

```
Итерация  9
Значения X:  [0.0037379662452339812, 1.4011698374184625, -0.4687237212594636]
Критерий по невязке:
[-0.0989914653443571, -0.06296805686338303, -0.07508193043031763]
Вектор неизвестных:
[0.012737190367448248, 1.4101652741132313, -0.46121552821643186]
Количество итераций:  10
Вектор погрешностей
[0.008999224122214267, 0.008995436694768877, 0.007508193043031741]
```

Работа из консоли

```
Введите погрешность: 0.01
Введите количество строк: 3
Вводите матрицу:
2 4 10 1
1 7 4 8
11 3 5 2
Получена матрица:
     2.0     4.0    10.0     1.0
     1.0     7.0     4.0     8.0
    11.0     3.0     5.0     2.0
Приведение к матрице преобладания диагональных коэффициентов:
    11.0     3.0     5.0     2.0
     1.0     7.0     4.0     8.0
     2.0     4.0    10.0     1.0
Итерация  0
Значения X:  [0.18181818181818182, 1.1428571428571428, 0.1]
Итерация  1
```

```
Итерация  9
Значения X:  [0.0037379662452339812, 1.4011698374184625, -0.4687237212594636]
Критерий по невязке:
[-0.0989914653443571, -0.06296805686338303, -0.07508193043031763]
Вектор неизвестных:
[0.012737190367448248, 1.4101652741132313, -0.46121552821643186]
Количество итераций:  10
Вектор погрешностей
[0.008999224122214267, 0.008995436694768877, 0.007508193043031741]
```

Вывод

В ходе работы реализован метод простых итераций, позволяющий решать СЛАУ итерационным методом. Метод показал быструю сходимость. Самой сложной частью работы оказалось приведение к диагональному преобладанию.