

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 №13

Группа: Р3216

Выполнил:

Сиразетдинов А.Н.

Проверил:

Малышева Т.А.

Г. Санкт-Петербург

2023

Оглавление

Цель работы	3
Описание метода, расчётные формулы	4
Листинг программы	6
Примеры и результаты работы программы.....	7
Работа из файла	7
Работа из консоли.....	8
Вывод.....	9

Цель работы

Используя известные методы вычислительной математики, написать программный код, осуществляющий решение СЛАУ. Проанализировать полученные результаты, оценить погрешность.

Описание метода, расчётные формулы

Итерационные методы. Метод простой итерации

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

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

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):

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

Обозначим:

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

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

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: $\mathbf{x} = \mathbf{Cx} + \mathbf{D}$, где \mathbf{x} – вектор неизвестных, \mathbf{C} – матрица коэффициентов преобразованной системы размерности $n \times n$, \mathbf{D} – вектор правых частей преобразованной системы.

Систему (6) представим в сокращенном виде:

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

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

Рабочая формула метода простой итерации:

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

где k – номер итерации.

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

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

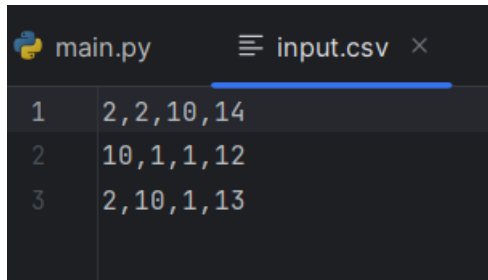
Листинг программы

```
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:  
        iter_count += 1  
        if iter_count > self.MAX_ITERATION_COUNT:  
            print("Превышено максимальное количество итераций!", file=sys.stderr)  
            break  
        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:  
            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



1	2, 2, 10, 14
2	10, 1, 1, 12
3	2, 10, 1, 13

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

Результат работы программы:

```
Получена матрица:
    2.0    2.0   10.0   14.0
  10.0    1.0    1.0   12.0
    2.0   10.0    1.0   13.0
Приведение к матрице преобладания диагональных коэффициентов:
  10.0    1.0    1.0   12.0
    2.0   10.0    1.0   13.0
    2.0    2.0   10.0   14.0
[[0, -0.1, -0.1], [-0.2, 0, -0.1], [-0.2, -0.2, 0]]
[1.2, 1.3, 1.4]
Вектор неизвестных:
[0.999568, 0.99946, 0.9993159999999999]
Количество итераций: 5
Вектор погрешностей
[0.0019320000000000448, 0.00246000000000001288, 0.0030840000000000867]
```

Работа из консоли

```
Введите погрешность: 0.01
Введите количество строк: 3
Вводите матрицу:
2 2 10 14
10 1 1 12
2 10 1 13
Получена матрица:
      2.0      2.0      10.0      14.0
10.0      1.0      1.0      12.0
      2.0     10.0      1.0      13.0
Приведение к матрице преобладания диагональных коэффициентов:
10.0      1.0      1.0      12.0
      2.0     10.0      1.0      13.0
      2.0      2.0     10.0      14.0
[[0, -0.1, -0.1], [-0.2, 0, -0.1], [-0.2, -0.2, 0]]
[1.2, 1.3, 1.4]
Вектор неизвестных:
[0.999568, 0.99946, 0.9993159999999999]
Количество итераций: 5
Вектор погрешностей
[0.00193200000000000448, 0.00246000000000001288, 0.00308400000000000867]
```


Вывод

В ходе работы реализован метод простых итераций, позволяющий решать СЛАУ с высокой точностью. Метод несложен в программном изложении и довольно быстро справляется с небольшими по размеру матрицами.