Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Введение в численные методы

Отчёт по практическому заданию

Студент Кибизов Кирилл, группа 207

2024

Оглавление

Постановка задачи

Дано:

1. Уравнение в частных производных с граничными условиями:

$$\begin{cases} k_x \frac{\partial^2 u}{\partial x^2} + k_y \frac{\partial^2 u}{\partial y^2} = 0, \quad (x, y) \in [0,1] \times [0,1], \\ u(x,0) = 0, \quad x \in [0,1] \\ u(0,y) = 0, \quad y \in [0,1] \\ u(x,1) = \sin(\pi x), \quad x \in [0,1] \\ u(1,y) = 0, \quad y \in [0,1] \end{cases}$$

2. Разностная схема:

$$\begin{cases} k_x \dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + k_y \dfrac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0, \quad i = \overline{1, N-1}, \ j = \overline{1, N-1}, \\ u_{i,0} = 0, \quad i = \overline{0, N}, \\ u_{0,j} = 0, \quad j = \overline{0, N}, \\ u_{i,N} = 0, \quad i = \overline{0, N}, \\ u_{N,j} = 0, \quad j = \overline{0, N}. \end{cases}$$

   где

$$u_{i,j} \approx u(x_i, y_j), \quad x_i = \frac{i}{N}, \quad y_j = \frac{j}{N}, \quad h = \frac{1}{N}.$$

3. Аналитическое решение данной задачи:

$$u(x,y) = \frac{\sinh(\pi y/\sqrt{k_y})}{\sinh(\pi/\sqrt{k_y}))} \sin(\pi x)$$

Задача:

Требуется решить данную СЛАУ с помощью итерационного метода Якоби (где он применим) для $N = 100$, рассматривая следующие случаи:

1. $k_x = k_y = 1$,
2. $k_x = 1$, $k_y = 10^6$.

В случае неприменимости итерационного метода Якоби предложить рабочий альтернативный метод.

Описание используемых числовых методов

Итерационные алгоритмы

При применении итерационных методов решения СЛАУ $Ax = f$ ответ получается в процессе построения последовательных приближений (итераций) $x_k = \{x_1^k, x_2^k, \ldots, x_n^k\}$, сходящихся к решению исходной системы в пространстве E_n с евклидовой нормой $\|x\|$: $\lim_{k\to\infty} x_k = x$, где i - номер компоненты, а k - номер итерации.

Сходимость обеспечивает принципиальную возможность получить в процессе итераций ответ с любой наперед заданной степенью точности.

Если очередной член последовательности x_{k+1} может выражаться только через предыдущий $x_{k+1} = F(x_k)$. Такие итерационные алгоритмы называют одношаговыми. Обычно линейно одношаговые алгоритмы записывают в стандартной канонической форме: $B_{k+1}\frac{x_{k+1}-x_k}{\tau_{k+1}} + Ax_k = f$ и $\det B_{k+1} \neq$ и $\tau_{k+1} > 0$. В такой записи процесс характеризуется последовательностью матриц B_{k+1} и числовых параметров τ_{k+1}, которые называют итерационными параметрами.

Идея метода Якоби

Метод Якоби строится на основе разностной схемы, в которой значения функции в каждом узле сетки обновляются независимо, используя значения с предыдущей итерации.

Идея метода верхней релаксации (SOR)

Итерационная схема метода SOR Метод SOR (Successive Over-Relaxation) представляет собой модификацию метода Зейделя, где значения в узлах обновляются последовательно и с учётом нового вычисленного значения, а также дополнительно корректируются с помощью параметра ω.

Анализ применимости используемых числовых методов

Перед тем как применять итерационные методы для решения системы линейных алгебраических уравнений (СЛАУ), необходимо убедиться, что они сходятся в рассматриваемом случае. Это включает в себя проверку структуры и свойств матрицы системы, а также оценку выполнения достаточных условий сходимости итерационных методов.

Достаточные условия сходимости итерационного процесса

Самосопряжённость матрицы

В одномерном случае в направлении x вторая производная $\frac{\partial^2 u}{\partial x^2}$ аппроксимируется по формуле:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2}.$$

Важно, что коэффициенты при $u_{i+1,j}$ и $u_{i-1,j}$ одинаковы (оба равны $\frac{1}{h_x^2}$). Аналогично производится аппроксимация второй производной по оси y. Благодаря симметрии разностной схемы все связи между узлами в матрице A получаются парными и "зеркальными". Это говорит о симметричности матрицы. Известные значения на границах области не делают матрицу несимметричной, так как они просто выносятся в вектор правой части f. Таким образом, можно сделать вывод, что матрица A — самосопряжённая. В случае вещественной матрицы (все элементы матрицы A — вещественные) понятия самосопряжённости и симметричности совпадают.

Положительно определённая матрица

Чтобы доказать, что матрица A положительно определённая, нужно показать, что для любого ненулевого вектора v выполняется неравенство: $v^T Av > 0$. Рассмотрим выражение $v^T Av$. Это скаляр, который можно записать как:

$$v^T Av = \sum_{i,j} v_{i,j}(Av)_{i,j}.$$

Для матрицы A, полученной из разностной аппроксимации второго порядка, можно записать, что A действует на вектор v следующим образом:

$$(Av)_{i,j} = \frac{k_x}{h^2}(v_{i+1,j} - 2v_{i,j} + v_{i-1,j}) + \frac{k_y}{h^2}(v_{i,j+1} - 2v_{i,j} + v_{i,j-1}).$$

Подставим это в $v^T Av$ и раскроем сумму:

$$v^T Av = \sum_{i,j} v_{i,j} \left(\frac{k_x}{h^2}(v_{i+1,j} - 2v_{i,j} + v_{i-1,j}) + \frac{k_y}{h^2}(v_{i,j+1} - 2v_{i,j} + v_{i,j-1}) \right).$$

При раскрытии суммы оказывается, что многие члены сокращаются. Получаем:

$$v^T Av = \sum_{i,j} \frac{k_x}{h^2}(v_{i+1,j} - v_{i,j})^2 + \frac{k_y}{h^2}(v_{i,j+1} - v_{i,j})^2.$$

В выражении $v^T Av$ остались только суммы квадратов разностей значений v в соседних узлах. Так как $k_x > 0$, $k_y > 0$ и $h > 0$, каждый член суммы неотрицателен. Так как $v^T Av$ является суммой строго неотрицательных слагаемых, и каждое из них положительно, если $v \neq 0$, то $v^T Av > 0$. Это доказывает, что матрица A положительно определённая.

Теорема Самарского

Пусть A — самосопряжённая положительно определённая матрица: $A = A^T, \quad A > 0$, и $B = A - \frac{\tau}{2}A$ — положительно определённая матрица, τ — положительное число: $B = A - \frac{\tau}{2}A > 0$.

Можно утверждать, что для матрицы A, которая:

- симметрична ($A^T = A$),
- положительно определённая ($v^T A v > 0$ для любого $v \neq 0$),

выполняются достаточные условия сходимости итерационных методов, таких как методы Якоби и верхней релаксации (SOR).

Сходимость методов

В итоге методы Якоби и верхней релаксации (SOR) применимы, к данной задаче, однако важно также учитывать:

- **Точность решения:** Точность определяется выбранным критерием остановки (например, достижением малого значения невязки или изменения решения между итерациями).
- **Быстрота сходимости:** Для улучшения быстроты сходимости можно:
  - уменьшить шаг h,
  - использовать "ускоряющие"параметры, такие как ω в методе верхней релаксации,

Реализация используемых числовых методов

```c
int solve_slae_via_jacobi(double u[N + 1][N + 1], double kx, double ky) {
    double u_new[N + 1][N + 1] = {{0.0}};
    for (int i = 0; i <= N; ++i) {
        double x = i * h;
        u_new[i][N] = u[i][N] = sin(M_PI * x);
    }

    int iter = 0;
    double max_dif;
    do {
        max_dif = 0.0;
        for (int i = 1; i < N; ++i) {
            for (int j = 1; j < N; ++j) {
                u_new[i][j] = (kx*(u[i+1][j] + u[i-1][j]) + ky*(u[i][j+1] + u[i][j-1])) /
                     (2*(kx+ky));
                double dif = fabs(u[i][j] - u_new[i][j]);
                if (max_dif < dif) {
                    max_dif = dif;
                }
            }
        }

        for (int i = 1; i < N; ++i) {
            for (int j = 1; j < N; ++j) {
                u[i][j] = u_new[i][j];
            }
        }

    } while ((++iter < MAX_ITERS) && (max_dif > EPS));

    if (iter == MAX_ITERS) {
        return -1;
    }

    return iter;
}
```

Реализация метода Якоби

```
int solve_slae_via_w(double u[N + 1][N + 1], double kx, double ky, double w) {
    for (int i = 0; i <= N; ++i) {
        double x = i * h;
        u[i][N] = sin(M_PI * x);
    }

    int iter = 0;
    double max_dif;
    do {
        max_dif = 0.0;
        for (int i = 1; i < N; ++i) {
            for (int j = 1; j < N; ++j) {
                double old_val = u[i][j];
                double tmp = (kx*(u[i+1][j] + u[i-1][j]) + ky*(u[i][j+1] + u[i][j-1])) /
                    (2*(kx+ky));
                u[i][j] = (1 - w)*old_val + w*tmp;

                double dif = fabs(u[i][j] - old_val);
                if (max_dif < dif) {
                    max_dif = dif;
                }
            }
        }

    } while ((++iter < MAX_ITERS) && (max_dif > EPS));

    if (iter == MAX_ITERS) {
        return -1;
    }

    return iter;
}
```

Реализация метода SOR

Результаты

1ый столбец - координаты точки
2ой столбец - численное решение
3ий столбец - аналитическое решение

```
./a.out
Please, input amount of tests (max 10): 1
Leave 3rd argument as 0 (for Jacobi) or as w (w = 1 for Gauss-Seidel; 1 < w < 2 for SOR)
Input kx and ky and w; for test #1: 1 1 0
...
u(0.250000, 0.500000) | 0.139489 | 0.140904
u(0.250000, 0.750000) | 0.319105 | 0.320099
u(0.250000, 1.000000) | 0.707107 | 0.707107
u(0.500000, 0.000000) | 0.000000 | 0.000000
...
u(0.500000, 0.750000) | 0.451283 | 0.452688
u(0.500000, 1.000000) | 1.000000 | 1.000000
u(0.750000, 0.000000) | 0.000000 | 0.000000
u(0.750000, 0.250000) | 0.052184 | 0.053187
...
--------
Test #1:
Iterations = 10247
```

Вывод метода Якоби дл 1го теста (для некоторых точек, покоординатно кратных 0.25)

```
guest@host:/media/sf_Shared/jacobi$ ./a.out
Please, input amount of tests (max 10): 1
Leave 3rd argument as 0 (for Jacobi) or as w (w = 1 for Gauss-Seidel; 1 < w < 2 for SOR)
Input kx and ky and w; for test #1: 1 1000000 1.0
...
u(0.250000, 0.500000) | 0.140196 | 0.140904
u(0.250000, 0.750000) | 0.319612 | 0.320099
u(0.250000, 1.000000) | 0.707107 | 0.707107
u(0.500000, 0.000000) | 0.000000 | 0.000000
...
u(0.500000, 1.000000) | 1.000000 | 1.000000
u(0.750000, 0.000000) | 0.000000 | 0.000000
u(0.750000, 0.250000) | 0.052690 | 0.053187
u(0.750000, 0.500000) | 0.140214 | 0.140904
...
--------
Test #1:
Iterations = 5851
```

Вывод метода SOR для 2го теста с w = 1.0 (для некоторых точек, покоординатно кратных 0.25)

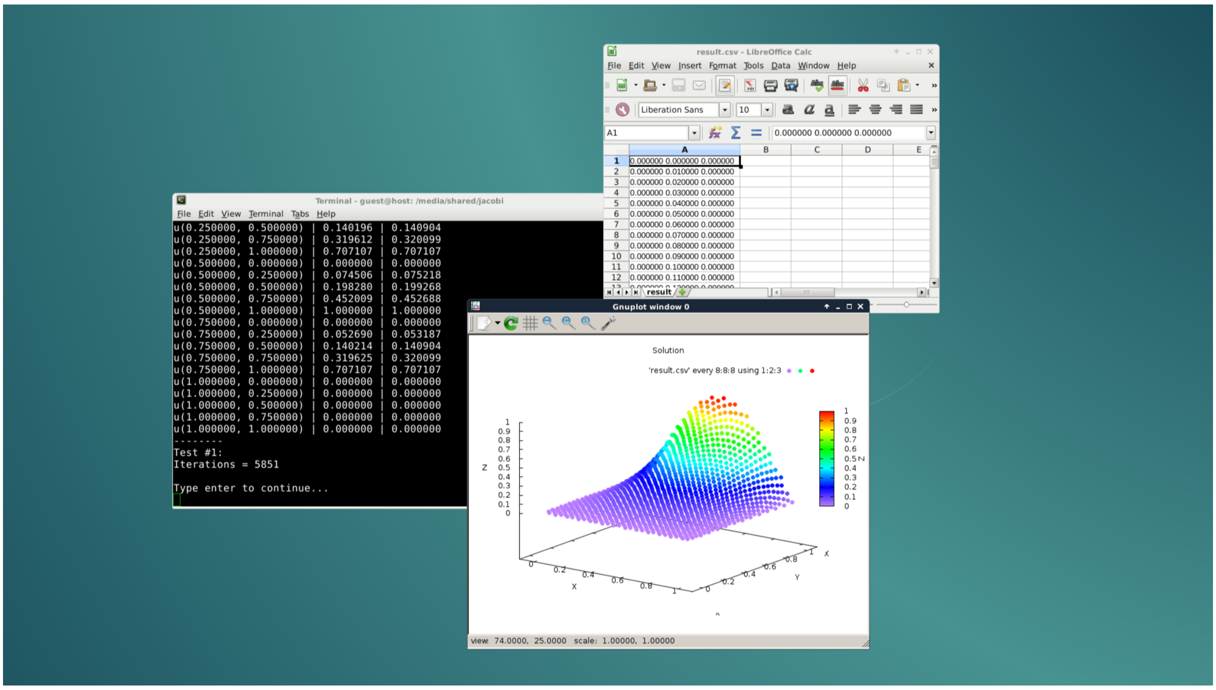
Заключение

Критерий	Метод Якоби	Метод SOR
Реализация	- Два массива, поэтапное обновление. - В каждой итерации эл-ты не зависимы. - Без вспомогательных параметров.	- Один массив, обновление "на ходу". - В каждой итерации эл-ты зависимы. - Требуется параметр ω $(0 < \omega \leq 2)$, влияющий на сходимость.
Сходимость	Медленная	Более быстрая

В данном отчёте была рассмотрена задача решения уравнения в частных производных с помощью итерационных методов, таких как метод Якоби и метод верхней релаксации. Реализация методов была выполнена на языке программирования С. Были получены результаты со сравнимо высокой точностью относительно предложенного аналитического решения.

Приложения

https://github.com/kibizoffs/jacobi



Литература

[1] Костомаров Д. П., Фаворский А. П. *Вводные лекции по численным методам.* — М.: Логос, 2004. — 184 с.

[2] Самарский А. А. *Введение в численные методы.* — М.: Наука, 1989. — 416 с.