Белорусский государственный университет
Факультет прикладной математики и информатики

Методы решения граничной задачи для ОДУ-2

Отчет по лабораторной работе
студента 3 курса 3 группы
Аквуха Джеймса

Преподаватель:
Будник А.М.

1. Постановка задачи

Поставлена задача Коши для ОДУ-2:

$$\begin{cases} (k(x)u'(x))' - q(x)u(x) = -f(x), \quad a \le x \le b \\ k(0)u'(0) = \alpha_0 u(0) - \mu_0, \\ -k(1)u'(1) = \alpha_1 u(1) - \mu_1. \end{cases}$$

Требуется найти значения функции на отрезке [a, b] на равномерной сетке из N + 1 точки.

2. Методы решения

2.1 Метод Ритца

Искомая функция приближается линейной комбинацией линейно независимых функций:

$$u_n(x) = \varphi_0(x) + \sum_{i=1}^{n} a_i \varphi_i(x)$$

$$\begin{cases} \varphi_i = x^{i+1}(x-1)^2, \quad i = \overline{1,n} \\ \varphi_0 = c_1 + c_2 x \end{cases}$$

Коэффициенты a_i находятся из условия:

$$\begin{cases} \sum_{j=1}^{n} (A\varphi_j, \varphi_i) a_j = (\varphi_i, f) - (A\varphi_0, \varphi_i), \quad i = \overline{1,n} \\ (\varphi_i, f) = \int_0^1 \varphi_i f dx \\ (A\varphi_j, \varphi_i) = \int_0^1 (k(x)\varphi_i' \varphi_j' + q(x)\varphi_i \varphi_j) dx \end{cases}$$

Интегралы вычисляются с помощью приближенной формулы. В результате получается СЛАУ порядка n относительно a_i, решая которую, находим приближенную формулу искомой функции.

2.2 Метод сеток

Искомое ОДУ-2 и граничные условия записываются на имеющейся сетке на трехточечном шаблоне:

$$\frac{k_{i+1} - k_{i-1}}{2h} \frac{y_{i+1} - y_{i-1}}{2h} + k_i \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - q_i y_i = -f_i, i = \overline{1, N-1}.$$

$$\left(\frac{hq_0}{2} + \alpha_0 + \frac{k_0 + k_1}{2h}\right) y_0 - \frac{k_0 + k_1}{2h} y_1 = \mu_0 + \frac{hf_0}{2},$$

$$-\frac{k_N + k_{N-1}}{2h} y_{N-1} + \left(\frac{k_N + k_{N-1}}{2h} + \alpha_1 + \frac{hq_N}{2}\right) y_N = \mu_1 + \frac{hf_N}{2}.$$

В результате получается СЛАУ порядка n относительно y, решая которую, находим приближенные значения искомой функции на сетке.

3. Результаты

Задача Коши для ОДУ-2:

$$\begin{cases} (\cos^2(x)u'(x))' - \sin(2x)u(x) = -x\sin 2x, \quad 0 \le x \le 1 \\ u'(0) = 1 \\ -\cos^2(1)u'(1) = u(1) - \cos^2 1 \end{cases}$$

$$N = 10$$

$$n = 5$$

Точное решения находилось с помощью Wolfram Mathematica.

```
→ cma git:(master) ✗ node ./ritz.js
0     1           2          3           4           5           6           7           8           9           10
----  ----------- ---------- ----------- ----------- ----------- ----------- ----------- ----------- ----------- ----
0.00  0.10        0.20       0.30        0.40        0.50        0.60        0.70        0.80        0.90        1.00
0e+0  -4.66647e-7 4.92384e-7 -1.89768e-7 -8.84908e-7 -6.09853e-7 -8.12535e-8 -4.15436e-7 -1.40353e-6 -1.29217e-6 0e+0
```

```
→ cma git:(master) ✗ node ./nets.js
0          1          2          3          4          5         6          7          8          9          10
---------- ---------- ---------- ---------- ---------- --------- ---------- ---------- ---------- ---------- ----------
0.00       0.10       0.20       0.30       0.40       0.50      0.60       0.70       0.80       0.90       1.00
4.16736e-3 3.66653e-3 3.17653e-3 2.70707e-3 2.26758e-3 1.8683e-3 1.52166e-3 1.24447e-3 1.06186e-3 1.01461e-3 1.17476e-3
```

4. Вывод

Погрешность метода сеток не превосходит ожидаемую погрешность для метода второго порядка точности с шагом 0.1. Метод Ритца обладает намного меньшей погрешностью, так как в данной работе с помощью метода Ритца находилось наилучшее приближение искомой функции многочленом 8 степени, у которого коэффициенты при x^2, x^3 равны 0, а точное решение - это многочлен первой степени. Следовательно, ожидалось, что метод Ритца найдет точное решение, и поэтому погрешность метода Ритца - это погрешность вычислений.
Тем не менее, метод сеток имеет большее практическое применение, так как для его реализации достаточно иметь условия, заданные на сетке.

4. Листинги

```
// ritz.js

require('console.table')
const {sin, cos, abs, pow} = Math
const {lusolve, subtract, transpose, sum} = require('mathjs')

const array = n => [...Array(n).keys()].map(_ => 0)
const range = (a, b, n = 1) => [...Array(n + 1).keys()].map(i => a + (b - a) / n * i)

const n = 5
const N = 10
const x = range(0, 1, N)
const y = x.map(x => x - 1)
const f = x => x * sin(2 * x)
const k = x => pow(cos(x), 2)
const q = x => sin(2 * x)

const phi = [
    x => x - 1,
    ...[...Array(n).keys()].map(i => i + 1).map(i => x => pow(x, i + 1) * pow(x - 1, 2))
]

const phid = [
    x => 1,
    ...[...Array(n).keys()].map(i => i + 1).map(i => x => (i + 1) * pow(x, i) * pow(x - 1, 2) + 2 * pow(x, i +
1) * (x - 1))
]

const rectangles = (f, a = 0, b = 1, n = 1000) => {
    const h = (b - a) / n
    const nodes = [...Array(n).keys()].map(i => a + h * i + h / 2)
    return h * sum(nodes.map(f))
}

const A = array(n).map(_ => array(n))
const b = array(n)

for (let i = 1; i <= n; ++i) {
    for (let j = 1; j <= n; ++j) {
        const createF = (i, j) => x => k(x) * phid[i](x) * phid[j](x) + q(x) * phi[i](x) * phi[j](x)
        A[i - 1][j - 1] = rectangles(createF(i, j))
        b[i - 1] = rectangles(x => f(x) * phi[i](x)) - rectangles(createF(0, i))
    }
}

const ai = transpose(lusolve(A, b))[0]

const u = x => phi[0](x) + sum(phi.slice(1).map((f, i) => f(x) * ai[i]))
const answer = x.map(u)

const prettify = x => Number(x.toPrecision(6)).toExponential()
console.table([x.map(x => x.toFixed(2)), subtract(y, answer).map(prettify)])

// nets.js

require('console.table')
const {sin, cos, abs, pow} = Math
const {lusolve, subtract, transpose} = require('mathjs')

const array = n => [...Array(n).keys()].map(_ => 0)
const range = (a, b, n = 1) => [...Array(n + 1).keys()].map(i => a + (b - a) / n * i)

const N = 10
const x = range(0, 1, N)
const f = x.map(x => x * sin(2 * x))
const k = x.map(x => pow(cos(x), 2))
const q = x.map(x => sin(2 * x))
const h = abs(x[1] - x[0])
const y = x.map(x => x - 1)

const alpha0 = 0
```

```
const mu0 = -1
const alpha1 = 1
const mu1 = pow(cos(1), 2)

let A = array(N + 1).map(_ => array(N + 1))
let b = array(N + 1)

for (let i = 1; i < N; ++i) {
    A[i][i - 1] = -(k[i + 1] - k[i - 1]) / (4 * pow(h, 2)) + k[i] / pow(h, 2)
    A[i][i] = -2 * k[i] / pow(h, 2) - q[i]
    A[i][i + 1] = (k[i + 1] - k[i - 1]) / (4 * pow(h, 2)) + k[i] / pow(h, 2)
    b[i] = -f[i]
}

A[0][0] = h * q[0] / 2 + alpha0 + (k[0] + k[1]) / (2 * h)
A[0][1] = -(k[0] + k[1]) / (2 * h)
b[0] = mu0 + h * f[0] / 2

A[N][N - 1] = -(k[N] + k[N - 1]) / (2 * h)
A[N][N] = - A[N][N - 1] + alpha1 + h * q[N] / 2
b[N] = mu1 + h * f[N] / 2

const prettify = x => Number(x.toPrecision(6)).toExponential()
console.table([x.map(x => x.toFixed(2)), subtract(y, transpose(lusolve(A, b))[0]).map(prettify)])

// wolfram.nb

sol = NDSolveValue[{Cos[x]^2*u''[x] - Sin[2 x]*u'[x] - Sin[2 x]*u[x] == -x  Sin[2 x],
   u[0] == -1 , -Cos[1]^2*u'[1] == u[1] - Cos[1]^2}, u, {x, 0, 1}, PrecisionGoal -> 10]
Plot[sol[x], {x, 0, 1}]
Table[Round[sol[x], .00001], {x, 0, 1, .1}]
MatrixForm[%]
```