МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики

Кафедра математического моделирования и анализа данных

Лабораторная работа 1

Численные методы решения краевых задач для ОДУ второго порядка

Вариант 11

Выполнил
Цуранов Никита Васильевич
3 курс, 7 группа

Преподаватель
Радкевич Елена Владимировна

Минск, 2020

Содержание

Постановка задачи

$$\begin{cases}(k(x)u'(x))' - q(x)u(x) = -f(x), 0 \le x \le 1\\ k(0)u'(0) = a_0u(0) - b_0,\\ -k(1)u'(1) = a_1u(1) - b_1,\end{cases}$$

$f(x)$	$k(x)$	$q(x)$	a_0	b_0	a_1	b_1
$x\sin(2x)$	$\cos^2(x)$	$\sin(2x)$	0	-1	1	$\cos^2(1)$

1. Аппроксимировать поставленную задачу разностной схемой 2-го порядка на минимальном шаблоне. Для повышения порядка аппроксимации граничных условий выделить главный член погрешности и заменить его, используя следующий вид исходного дифференциального уравнения:

$$k(x)u''(x) + k'(x)u'(x) - q(x)u(x) = -f(x)$$

2. При помощи интегро-интерполяционного метода построить консервативную разностную схему, вычисляя интегралы по формуле средних прямоугольников

3. Аппроксимировать исходную задачу вариационно-разностным методом, для вычисления коэффициентов используя квадратурную формулу трапеций

4. Методом разностной прогонки реализовать полученные разностные схемы с шагом 0.1, провести сравнительный анализ решений при разных значениях шага

1 Метод повышения порядка точности

Используя шаблон $\{x-h, x, x+h\}$ для аппроксимации схемой второго порядка построим разностный оператор:

$$k(x)y_{lr} + k'(x)y_m - q(x)y = -f(x)$$

где y_l, y_m, y_r — левая, центральная, правая производные

Условия на границах можно аппроксимировать только первым порядком, т.к. $x-h$ не входит в сетку в точке 0, а $x+h$ — в 1:

$$y_r(0) = 1 \qquad y_l(1) = 1 - \frac{u(1)}{\cos^2(1)}$$

Для повышения порядка разложим погрешность аппроксимации в ряд Тейлора:

$$u(h) = u(0) + u'(0)h + \frac{u''(0)h^2}{2} + o(h^2) \qquad u(1-h) = u(1) - u'(1)h + \frac{u''(1)h^2}{2} + o(h^2)$$

$$u'(0) - \frac{u(h)-u(0)}{h} = u'(0) - \frac{u(0) + u'(0)h + \frac{u''(0)h^2}{2} + o(h^2) - u(0)}{h} = -\frac{u''(0)h}{2} + O(h^2)$$

$$u'(1) - \frac{u(1)-u(1-h)}{h} = u'(1) - \frac{u(1) - u(1) + u'(1)h - \frac{u''(1)h^2}{2} + o(h^2)}{h} = \frac{u''(1)h}{2} + O(h^2)$$

$$k(x)u''(x) + k'(x)u'(x) - q(x)u(x) = -f(x) \Rightarrow u''(x) = \frac{k'(x)u'(x) + q(x)u(x) - f(x)}{k(x)}$$

$$u'(0) \approx u_r(0) - \frac{u''(0)h}{2} = u_r(0) - \frac{k'(0)(a_0u(0) - b_0) + q(0)u(0)k(0) - f(0)k(0)}{k(0)^2}$$

$$u'(1) \approx u_l(1) + \frac{u''(1)h}{2} = u_l(1) - \frac{k'(1)(a_1u(1) - b_1) - q(1)u(1)k(1) + f(1)k(1)}{k(1)^2}$$

Получим следующую схему:

$$\begin{cases} k(x)y_{lr} + k'(x)y_m - q(x)y = -f(x), x \in w_h \\ y_r(0)k(0)^2 - k'(0)(a_0y(0) - b_0) - q(0)y(0)k(0) + f(0)k(0) = (a_0y(0) - b_0)k(0), \\ y_r(1)k(1)^2 - k'(1)(a_1y(1) - b_1) + q(1)y(1)k(1) - f(1)k(1) = -(a_1y(1) - b_1)k(1). \end{cases}$$

Или в индексном виде:

$$\begin{cases} k_i\frac{y_{i+1}-2y_i+y_{i-1}}{h^2} + k'_i\frac{y_{i+1}-y_{i-1}}{2h} - q_iy_i = -f_i, i = \overline{1, N-1} \\ \frac{y_1-y_0}{h}k_0^2 - k'_0(a_0y_0 - b_0) - q_0y_0k_0 + f_0k_0 = (a_0y_0 - b_0)k_0, \\ \frac{y_N-y_{N-1}}{h}k_N^2 - k'_N(a_1y_N - b_1) + q_Ny_Nk_N - f_Nk_N = -(a_1y_N - b_1)k_N. \end{cases}$$

$$\begin{cases} \frac{2k_i-k'_ih}{2h^2}y_{i-1} - \frac{2k_i+q_ih^2}{h^2}y_i + \frac{2k_i+k'_ih}{2h^2}y_{i+1} = -f_i, i = \overline{1, N-1} \\ (-\frac{k_0^2}{h} - k'_0a_0 - q_0k_0 - a_0k_0)y_0 + \frac{k_0^2}{h}y_1 = -f_0k_0 + b_0(k'_0 - k_0), \\ -\frac{k_N^2}{h}y_{N-1} + (\frac{k_N^2}{h} - k'_Na_1 + q_Nk_N + a_1k_N)y_N = f_Nk_N + b_1(k'_N + k_N). \end{cases}$$

Полученную систему можно свести к системе линейных уравнений. Матрица системы трехдиагональная, а значит применим метод прогонки.

2 Метод Баланса

Проинтегрировав исходное уравнение на отрезке $[x_{i-0.5}, x_{i+0.5}]$ получим:

$$k(x)u'(x)\Big|_{x_{i-0.5}}^{x_{i+0.5}} - \int_{x_{i-0.5}}^{x_{i+0.5}} q(x)u(x)dx = -\int_{x_{i-0.5}}^{x_{i+0.5}} f(x)dx, i = \overline{1, N-1}$$

Используя квадратурную формулу средних прямоугольников найдем:

$$\int_{x_{i-0.5}}^{x_{i+0.5}} f(x)dx \approx f(x_i)h$$

$$\int_{x_{i-0.5}}^{x_{i+0.5}} q(x)u(x)dx \approx q(x_i)u(x_i)h$$

Рассмотрим первое слагаемое:

$$k(x)u'(x)\Big|_{x_{i-0.5}}^{x_{i+0.5}} = k(x_{i+0.5})u'(x_{i+0.5}) - k(x_{i-0.5})u'(x_{i-0.5})$$

$u'(x_{i-0.5}) \approx u_m(x_{i-0.5}) = \dfrac{u(x_i) - u(x_{i-1})}{h}$ (рассматривая с половинным шагом)

Тогда все полученное подставим в исходное уравнение и пусть $c_i = k(x_{i-0.5})$:

$$c_{i+1}\frac{y_{i+1} - y_i}{h} - c_i\frac{y_i - y_{i-1}}{h} - q_i y_i h = -f_i h, i = \overline{1, N-1}$$

$$c_i y_{i-1} - (q_i h^2 + c_i + c_{i+1})y_i + c_{i+1}y_{i+1} = -f_i h^2, i = \overline{1, N-1}$$

Рассмотрим аппроксимацию краевых условий:

$k(0)u'(0) = a_0 u(0) - b_0,$

$$\begin{gathered} k(x)u'(x)|_0^{\frac{h}{2}} - \int_0^{\frac{h}{2}} q(x)u(x)dx = -\int_0^{\frac{h}{2}} f(x)dx \\ c_1\tfrac{y_1 - y_0}{h} - a_0 y_0 + b_0 - \tfrac{h}{2}q(\tfrac{h}{4})y_0 = -\tfrac{h}{2}f(\tfrac{h}{4}) \\ \Rightarrow -(\tfrac{c_1}{h} + a_0 + q(\tfrac{h}{4})\tfrac{h}{2})y_0 + \tfrac{c_1}{h}y_1 = -b_0 - \tfrac{h}{2}f(\tfrac{h}{4}) \end{gathered}$$

$-k(1)u'(1) = a_1 u(1) - b_1,$

$$\begin{gathered} k(x)u'(x)|_{1-\frac{h}{2}}^1 - \int_{1-\frac{h}{2}}^1 q(x)u(x)dx = -\int_{1-\frac{h}{2}}^1 f(x)dx \\ -a_1 y_N + b_1 - c_N\tfrac{y_N - y_{N-1}}{h} - \tfrac{h}{2}q(1-\tfrac{h}{4})y_N = -\tfrac{h}{2}f(1-\tfrac{h}{4}) \\ \Rightarrow \tfrac{c_N}{h}y_{N-1} - (\tfrac{c_N}{h} + a_1 + q(1-\tfrac{h}{4})\tfrac{h}{2})y_N = -b_1 - \tfrac{h}{2}f(1-\tfrac{h}{4}) \end{gathered}$$

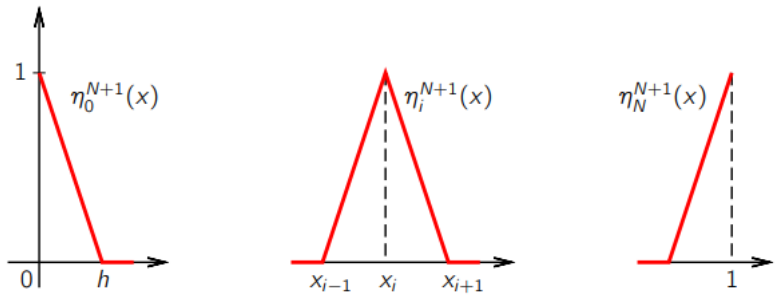
Исходя из всех полученных уравнений мы опять можем построить систему линейных уравнений с трехдиагональной матрицей системы и решить методом прогонки.

3 Метод Ритца

Метод Ритца позволяет искать минимум функционала, тогда исходная задача будет эквивалентна отысканию минимума функционала:

$$J(u) = \frac{1}{2}\Big(a_0u^2(0)+a_1u^2(1)+\int_0^1 (k(x)(u'(x))^2+q(x)u^2(x))dx\Big)-b_0u(0)-b_1u(1)-\int_0^1 f(x)u(x)dx$$

Суть поиска заключается в построении подпространства и поиск минимума в нем. Чтобы построить разностную схему нужно подобрать координатные функции специальным образом.



Тогда система примет вид:

$$\begin{cases} \frac{c_i}{h^2}y_{i-1} - (\frac{c_i+c_{i+1}}{h^2} + d_i)y_i + \frac{c_{i+1}}{h^2}y_{i+1} = -\phi_i, i = \overline{1, N-1} \\ -(a_0 + \frac{c_1}{h} + \frac{h}{2}d_0)y_0 + \frac{c_1}{h}y_1 = -b_0 - \frac{h}{2}\phi_0 \\ \frac{c_N}{h}y_{N-1} - (a_1 + \frac{c_N}{h} + \frac{h}{2}d_N)y_N = -b_1 - \frac{h}{2}\phi_N \end{cases}$$, где:

$$c_i = \frac{1}{h}\Big(\int_{x_{i-1}}^{x_i} k(x)dx - \int_{x_{i-1}}^{x_i} q(x)(x_i - x)(x - x_{i-1})dx\Big), i = \overline{1, N}$$

$$d_i = \frac{1}{h^2}\Big(\int_{x_{i-1}}^{x_i} q(x)(x - x_i)dx - \int_{x_i}^{x_{i+1}} q(x)(x_{i+1} - x)dx\Big), i = \overline{1, N-1}$$

$$d_0 = \frac{2}{h^2}\int_0^h q(x)(h - x)dx; d_N = \frac{2}{h^2}\int_{1-h}^1 q(x)(x - 1 + h)dx$$

$$\phi_i = \frac{1}{h^2}\Big(\int_{x_{i-1}}^{x_i} f(x)(x - x_i)dx - \int_{x_i}^{x_{i+1}} f(x)(x_{i+1} - x)dx\Big), i = \overline{1, N-1}$$

$$\phi_0 = \frac{2}{h^2}\int_0^h f(x)(h - x)dx; \phi_N = \frac{2}{h^2}\int_{1-h}^1 f(x)(x - 1 + h)dx$$

Интегралы найдем численно по методу трапеций:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\big(f(a) + f(b)\big)$$

$$c_i = \frac{k(x_{i-1})+k(x_i)}{2} = \frac{k_{i-1}+k_i}{2}, i = \overline{1, N}$$

$$d_i = q(x) = q_i, i = \overline{1, N-1}$$
$$d_0 = -q_0; d_N = q_N$$

$$\phi_i = f(x) = f_i, i = \overline{1, N-1}$$
$$\phi_0 = -f_0; \phi_N = f_N$$

Листинг программы:

```
import numpy as np
from numpy import power, sin, cos, tan
from scipy.linalg import norm, solve_banded

true_ans = np.linspace(-1, 0, 11)
f = lambda x: x * sin(2 * x)
k = lambda x: power(cos(x), 2)
q = lambda x: sin(2 * x)
dk = lambda x: -sin(2 * x)
a_0 = 0
b_0 = -1
a_1 = 1
b_1 = cos(1)**2

x = np.linspace(0, 1, 11)[1: -1]
i_range = np.arange(1, x.size - 1)
h = 0.1
f_v = f(x)
k_v = k(x)
q_v = q(x)
dk_v = dk(x)

def Solve(A_0_0, A_0_1, A_n_nm1, A_n_n,
          A_below, A_in, A_above,
          b_beg, b_mid, b_end):
    A = np.array([
        [0, A_0_1, *A_above],
        [A_0_0, *A_in, A_n_n],
        [*A_below, A_n_nm1, 0],
    ])

    b = [b_beg, *b_mid, b_end]

    solution = solve_banded((1, 1), A, b)
    print('Solution:')
    for i in solution:
    print('{:.5f}'.format(i), end=' ')
    disc = solution - true_ans
    print('\nDiscrepancy:')
    for i in disc:
    print('{:+.1e}'.format(i), end=' ')
    print('\nRelative error: {:.2e}'.format(norm(disc) / norm(solution)))
    print()

```

```
def IncreasingOrderMethod():
    print('Increasing order method: ')
    Solve(-k(0) ** 2 / h - dk(0) * a_0 - q(0) * k(0) - a_0 * k(0), k(0) ** 2 / h,
          -k(1) ** 2 / h, k(1) ** 2 / h - dk(1) * a_1 + q(1) * k(1) + a_1 * k(1),
          (2 * k_v - dk_v * h) / h ** 2 / 2,
          -(2 * k_v + q_v * h ** 2) / h ** 2,
          (2 * k_v + dk_v * h) / h ** 2 / 2,
          -f(0) * k(0) + b_0 * (dk(0) - k(0)),
          -f_v,
          f(1) * k(1) + b_1 * (dk(1) + k(1)))

def BalanceMethod():
    print('Balance method: ')
    c = np.append(k(x - h / 2), k(1 - h / 2))
    Solve(-(c[0] / h + a_0 + q(h / 4) * h / 2), c[0] / h,
          c[-1] / h, -(c[-1] / h + a_1 + q(1 - h / 4) * h / 2),
          c[:-1], -(q_v * h ** 2 + c[1:] + c[:-1]), c[1:],
          -b_0 - h / 2 * f(h / 4),
          -f_v * h**2,
          -b_1 - h / 2 * f(1 - h / 4))

def RitzMethod():
    print('Ritz method: ')
    x_ext = np.linspace(0, 1, 11)
    c = (k(x_ext[:-1]) + k(x_ext[1:])) / 2
    Solve(-(a_0 + c[0] / h - h * q(0) / 2), c[0] / h,
          c[-1] / h, -(a_1 + c[-1] / h + h * q(1) / 2),
          c[:-1] / h**2, -(c[:-1] + c[1:]) / h**2 - q_v, c[1:] / h**2,
          -b_0 + h * f(0) / 2, -f_v, -b_1 - h * f(1) / 2)

print('True solution:')
for i in true_ans:
print('{:+.5f}'.format(i), end=' ')
print('\n')

IncreasingOrderMethod()
BalanceMethod()
RitzMethod()
```

Использовался метод прогонки из библиотеки numpy. На вход принимает матрицу из диагоналей и их количество (в нашем случае одна над и одна под)

Вывод программы

Вывод для h = 0.1

```
True solution:
-1.00000 -0.90000 -0.80000 -0.70000 -0.60000 -0.50000 -0.40000 -0.30000 -0.20000 -0.10000 +0.00000

Increasing order method:
Solution:
-1.00000 -0.90000 -0.80000 -0.70000 -0.60000 -0.50000 -0.40000 -0.30000 -0.20000 -0.10000 -0.00000
Discrepancy:
-2.2e-15 -2.0e-15 -1.8e-15 -1.8e-15 -1.8e-15 -1.7e-15 -1.5e-15 -1.2e-15 -8.9e-16 -5.6e-16 -1.8e-16
Relative error: 2.60e-15

Balance method:
Solution:
-0.99944 -0.89944 -0.79945 -0.69947 -0.59949 -0.49952 -0.39957 -0.29965 -0.19977 -0.09994 -0.00021
Discrepancy:
+5.6e-04 +5.6e-04 +5.5e-04 +5.3e-04 +5.1e-04 +4.8e-04 +4.3e-04 +3.5e-04 +2.3e-04 +5.9e-05 -2.1e-04
Relative error: 7.41e-04

Ritz method:
Solution:
-1.00416 -0.90366 -0.80317 -0.70270 -0.60226 -0.50186 -0.40152 -0.30124 -0.20106 -0.10101 -0.00117
Discrepancy:
-4.2e-03 -3.7e-03 -3.2e-03 -2.7e-03 -2.3e-03 -1.9e-03 -1.5e-03 -1.2e-03 -1.1e-03 -1.0e-03 -1.2e-03
Relative error: 4.06e-03
```

Вывод для h = 0.01

```
Increasing order method:
Solution:
-1.00000 -0.99000 -0.98000 -0.97000 -0.96000 -0.95000 -0.94000 -0.93000 -0.92000 -0.91000 -0.90000
Discrepancy:
+4.5e-14 +4.6e-14 +4.6e-14 +4.7e-14 +4.8e-14 +4.8e-14 +4.9e-14 +4.9e-14 +4.9e-14 +4.9e-14 +4.9e-14
Relative error: 6.16e-14

Balance method:
Solution:
-0.99999 -0.98999 -0.97999 -0.96999 -0.95999 -0.94999 -0.93999 -0.92999 -0.91999 -0.90999 -0.89999
Discrepancy:
+5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06 +5.2e-06
Relative error: 7.20e-06

Ritz method:
Solution:
-1.00004 -0.99004 -0.98004 -0.97004 -0.96004 -0.95004 -0.94004 -0.93004 -0.92004 -0.91004 -0.90004
Discrepancy:
-4.1e-05 -4.1e-05 -4.0e-05 -4.0e-05 -3.9e-05 -3.9e-05 -3.8e-05 -3.8e-05 -3.7e-05 -3.7e-05 -3.6e-05
Relative error: 4.01e-05
```

Выводы

Каждый из методов обладает одинаковыми вычислительной сложностью и порядком точности, а именно вторым, что можно увидеть по результатам. Но на практике для этой задачи оказался лучшим метод повышения порядка аппроксимации, т.к. его невязка связанна с машинной погрешностью, потому что ошибка в 15 знаке.

В случае моего варианта посчитать производную от k(x) не составило труда, поэтому мы с могли избежать дополнительной погрешности. Хотя для более сложных функций можно было использовать численную производную. Бóльшую погрешность метода Ритца можно попробовать объяснить большим количеством интегралов, которые считались приближенно.

При уменьшении шага, как и ожидалась возросла точность, но невязка 1 метода увеличилась из-за машинной погрешности.