Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Теоретическая информатика и компьютерные технологии»

Отчёт о лабораторной работе № 1

по курсу «Численные методы»

ЧИСЛЕННОЕ РЕШЕНИЕ КРАЕВОЙ ЗАДАЧИ ДЛЯ ЛИНЕЙНОГО ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ 2-ОГО ПОРЯДКА

Студент: К. Лозовска

Группа: ИУ9И-64Б

Преподаватель: А.Б.Домрачева

Москва, 2023

СОДЕРЖАНИЕ

ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ

Цель: Найти численное решение краевой задачи для линейного дифференциального уравнения второго порядка методом прогонки и стрельбы. Сравнить точность методов.

Постановка задачи:

Написать и отладить процедуру для решения трехдиагональной линейной системы методом прогонки.

Решить аналитически задачу Коши:

$$y'' + py' + qy = f(x), \qquad y(0) = y_0, \qquad y'(0) = y_0'$$

По найденному решению задачи Коши $y(x)$ вычислить $b = y(1)$.

Найти численное решение $(x_i, y_i), i = 0, \dots, n$ краевой задачи для того же уравнения с краевыми условиями $y(0) = a, y(1) = b$ при $n = 10$.

Найти погрешность численного решения $\left||y - \tilde{y}|\right| = \max|y(x_i) - \widetilde{y_i}|, 0 \leq i \leq n$.

Тестовый пример:

$$f(x) = 3\sin(2x), \qquad p = 0, q = 4, \qquad y_0 = 2, y_0' = 0{,}75$$

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Решение краевой задачи методом прогонки.

Краевая задача для линейного дифференциального уравнения второго порядка имеет вид:

$$y'' + p(x)y' + q(x)y = f(x) \qquad (1)$$

$$y(0) = a, \qquad y(1) = b \qquad (2)$$

Требуется найти частное решение уравнения (1), отвечающее краевым условиям (2).

Приближенным численным решением этой задачи называется сеточная функция $(x_i, y_i), i = 0, \dots, n$, заданная в в $(n-1)$-й точке $x_i = ih, h = \frac{1}{n}$.

Обозначим через $p_i = p(x_i), q_i = q(x_i), f_i = f(x_i)$ значения коэффициентов уравнения в точках x_i (узлах сетки), $i = 0, \dots, n$. Применяя разностную аппроксимацию производных по формулам численного дифференцирования, получим приближенную систему уравнений относительно ординат сеточной функции y_i:

$$\frac{y_{i+1} - 2y_i + 2y_{i-1}}{h^2} + \frac{p_i(y_{i+1} - y_{i-1})}{2h} + q_i y_i = f_i,$$

или после преобразований:

$$y_{i-1}\left(1 - \frac{h}{2}p_i\right) + y_i(h^2 q_i - 2) + y_{i+1}\left(1 + \frac{h}{2}p_i\right) = h^2 f_i, \qquad i = 1, \dots, n-1 \qquad (3)$$

с краевыми условиями:

$$y_0 = a, y_n = b. \qquad (4)$$

Система (3) является разностной системой с краевыми условиями (4) и представляет собой трехдиагонльную систему линейных алгебраических уравнений $(n-1)$-ого порядка. Трехдиагональную линейную систему решаем методом прогонки.

Решение краевой задачи методом стрельбы.

Рассмотрим краевую задачу для линейного дифференциального уравнения второго порядка:

$$y'' + p(x)y' + q(x)y = f(x), \qquad y(a) = A, \qquad y(b) = B \qquad (5)$$

Общее решение уравнения (1) имеет вид:

$$y_{OH} = C_1 y_1 + C_2 y_2 + y_{\text{ЧН}},$$

где y_{OH} – общее решение неоднородного уравнения, $C_1 y_1 + C_2 y_2$ – общее решение соответствующего однородного уравнения, C_1 и C_2 – произвольные постоянные, $y_{\text{ЧН}}$ – частное решение неоднородного уравнения. Величины C_1 и C_2 определяют из системы уравнений:

$$C_1 y_1(a) + C_2 y_2(a) + y_{\text{ЧН}}(a) = A,$$
$$C_1 y_1(b) + C_2 y_2(b) + y_{\text{ЧН}}(b) = B.$$

Если частное решение неоднородного уравнения удовлетворяет условию $y_{\text{ЧН}}(a) = A$, а одно из частных решений однородного уравнения, например $y_1(x)$, условию $y_1(a) = 0$, то первое уравнение системы принимает вид:

$$C_1 \times 0 + C_2 y_2(a) + A = A,$$

и следовательно, $\mathrm{C}_2 = 0$. Постоянную C_1 определяют из второго уравнения:

$$C_1 y_1(b) + y_{\text{ЧН}}(b) = B.$$

Описанный метод носит название метода стрельбы.

Рассмотрим его сеточный аналог. Для этого разобьем отрезок $[a, b]$ на n частей точками $x_0, \dots, x_n$, где $x_i = a + ih, h = (b - a)/n$, а производные в уравнении (5) во всех внутренних точках заменим их разностными аналогами:

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h}, y_i'' = \frac{y_{i+1} - 2y_i + y_{i+1}}{h^2}, i = 1, \dots, n-1.$$

Будем искать решение, удовлетворяющие условиям:

$$y_0(x_0) = A, \qquad y_0(x_1) = D_0, \qquad y_1(x_0) = 0, \qquad y_1(x_1) = D_1 \neq 0 \qquad (6)$$

Для уменьшения вычислительной погрешности обычно берут $D_0 = A + O(h)$ и $D_1 = O(h)$.

Для определения y_0 и y_1 получим уравнения:

$$\frac{y_0(x_{i+1}) - 2y_0(x_i) - y_0(x_{i-1})}{h^2} + \frac{p_i\big(y_0(x_{i+1}) - y_0(x_{i-1})\big)}{2h} + q_i y_0(x_i) = f_i,$$

$$\frac{y_1(x_{i+1}) - 2y_1(x_i) - y_1(x_{i-1})}{h^2} + \frac{p_i\big(y_1(x_{i+1}) - y_1(x_{i-1})\big)}{2h} + q_i y_1(x_i) = 0.$$

Отсюда, используя (6), имеем:

$$y_0(x_{i+1}) = (f_i h^2 + (2 - q_i h^2) y_0(x_i) - \left(1 - \frac{p_i h}{2}\right) y_0(x_{i-1}))/(1 + p_i h/2),$$

$$y_i(x_{i+1}) = ((2 - q_i h^2) y_1(x_i) - \left(1 - \frac{p_i h}{2}\right) y_1(x_{i-1}))/(1 + p_i h/2),$$

$i = 1, \dots, n$ и $\int_a^b K(x, y)\varphi(y)dy \approx h\sum_{j=1}^n K\big(x, x_j\big)\varphi\big(x_j\big)$.

Система линейных уравнений относительно искомых значений $\varphi_i = \varphi(x_i)$ имеет вид:

$$(1 - hK_{11})\varphi_1 - hK_{12}\varphi_2 - hK_{13}\varphi_3 - \dots - hK_{1n}\varphi_n = f_1,$$
$$-hK_{21}\varphi_1 + (1 - hK_{22})\varphi_2 - hK_{23}\varphi_3 - \dots - hK_{2n}\varphi_n = f_2,$$

… …

$$-hK_{n1}\varphi_1 + hK_{n2}\varphi_2 - hK_{n3}\varphi_3 - \cdots + (1 - hK_{nn})\varphi_n = f_n.$$

Здесь $K_{ij} = K(x_i, x_j)$ и $f = f(x_i), i, j = 1, \ldots, n$. Система уравнений решается любым из методов численного решения СЛАУ, например методом Гаусса, или, в случае симметричной матрицы, - методом квадратного корня.

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Данный код реализует **метод прогонки** для решения дифференциального уравнения второго порядка.

Функция ***direct*** решает систему уравнений методом прогонки, а функция ***reverse*** находит решение системы уравнений, используя найденные коэффициенты.

```python
import math

def direct(b, a, c, d, size):
    alpha = [-c[0] / b[0]]
    beta = [d[0] / b[0]]
    y = 0.0
    for i in range(1, size - 1):
        y = a[i - 1] * alpha[i - 1] + b[i]
        alpha.append(-c[i] / y)
        beta.append((d[i] - a[i - 1] * beta[i - 1]) / y)
    y = a[size - 2] * alpha[size - 2] + b[size - 1]
    beta.append((d[size - 1] - a[size - 2] * beta[size - 2]) / y)
    return alpha, beta

def reverse(alpha, beta, size):
    x = [0.0] * size
    x[size - 1] = beta[size - 1]
    for i in range(size - 2, -1, -1):
        x[i] = alpha[i] * x[i + 1] + beta[i]
    return x
```

Функция ***f*** задает правую часть дифференциального уравнения, а функция ***analytical*** задает аналитическое решение уравнения.

```
def f(x):
    return 3 * math.sin(2 * x)

def analytical(x):
    return (2 - 0.75 * x) * math.cos(2 * x) + 1.5 * math.sin(x) * math.cos(x)
```

Переменные ***p*** и ***q*** задают коэффициенты уравнения, а ***a*** и ***b*** задают граничные условия. Переменная ***n*** задает количество разбиений отрезка, на котором решается уравнение.

```
p = 0.0
q = 4.0
a = analytical(0)
b = analytical(1)
```

```

n = 40
h = 1.0 / float(n)
xs = [i * h for i in range(n + 1)]

```

Далее создаются списки ***as_***, ***bs***, ***cs*** и ***ds***, которые задают коэффициенты системы уравнений.

```
as_ = [1 - h / 2 * p for i in range(n - 2)]
bs = [h * h * q - 2 for i in range(n - 1)]
cs = [1 + h / 2 * p for i in range(n - 2)]
ds = [h * h * f(0) - a * (1 - h / 2 * p)]
```

```

for i in range(2, n):
    ds.append(h * h * f(i * h))
ds[-1] = h * h * f((len(ds) - 1) * h) - b * (1 + h / 2 * p)

```

Затем вызываются функции ***direct*** и ***reverse*** для нахождения решения системы уравнений и решения дифференциального уравнения соответственно. В цикле находится максимальная погрешность. В конце выводится значение максимальной погрешности.

```
alpha, beta = direct(bs, as_, cs, ds, len(ds))
ys = [a] + reverse(alpha, beta, len(ds)) + [b]

maxR = 0.0
for i in range(0, len(ys), 4):
    inaccuracy = abs(ys[i] - analytical(xs[i]))
    print("x=%.1f, y=%.6f, y*=%.6f  |y-y*|=%.6f" % (i * h, analytical(xs[i]), ys[i], inaccuracy))
    if inaccuracy > maxR:
        maxR = inaccuracy

print("||y-y*||=%.6f" % maxR)
```

Данный код реализует **метод стрельбы** для решения дифференциального уравнения второго порядка.

Функция ***f(x)*** задает правую часть дифференциального уравнения. Функция ***analytical(x)*** задает аналитическое решение дифференциального уравнения.

```
import math

def f(x):
    return 3 * math.sin(2 * x)

def analytical(x):
    return (2 - 0.75 * x) * math.cos(2 * x) + 1.5 * math.sin(x) * math.cos(x)
```

Переменные ***n***, ***p***, ***q***, ***a***, ***b*** и ***ys*** используются для хранения параметров метода стрельбы и результатов его работы.

```
n = 10
p = 0.0
q = 4.0
a = analytical(0)
b = analytical(1)
ys = [[], []]
```

Функция ***getC1()*** вычисляет коэффициент C_1 для метода стрельбы. Функция ***getYi(i)*** вычисляет значение y для заданного индекса i.

```
def getC1():
    return (b - ys[0][n]) / ys[1][n]

def getYi(i):
    return ys[0][i] + getC1() * ys[1][i]
```

В ***цикле for*** вычисляются значения $ys(x_0)$ и $ys(x_1)$ для каждого шага разбиения. Затем вычисляются значения y для каждого шага разбиения.

```
delta = h * 100
xs = []
for i in range(n + 1):
    xs.append(float(i) * h)
for i in range(2):
    ys[i] = [a, a + delta]
ys[1] = [0, delta]

for i in range(1, n):
    ys[0].append((h * h * f(xs[i]) + (2.0 - q * h * h) * ys[0][i] - (1.0 - h / 2 * p) * ys[0][i - 1]) / (1 + h / 2 * p))
    ys[1].append(((2.0 - q * h * h) * ys[1][i] - (1.0 - h / 2 * p) * ys[1][i - 1]) / (1 + h / 2 * p))

y = []
for i in range(n + 1):
    y.append(getYi(i))
print(len(y))
```

В конце кода вычисляется максимальная погрешность ***maxR*** и выводятся результаты работы метода стрельбы.

```
47  maxR = 0.0
48
49  for i in range(len(y)):
50      print("x=%.1f, y=%.6f, y*=%.6f  |y-y*|=%.6f" % (float(i) * h, analytical(xs[i]), y[i], math.fabs(y[i] - analytical(xs[i]))))
51      if math.fabs(y[i] - analytical(xs[i])) > maxR:
52          maxR = math.fabs(y[i] - analytical(xs[i]))
53
54  print("||y-y*||=%.6f" % maxR)
```

РЕЗУЛЬТАТЫ

Для примера было использовано следующее дифференциальное уравнение:

$$y'' + 0.0y' + 4.0y = (2 - 0.75 * x) * cos(2x) + 1.5 * sin(x) * cos(x)$$

Результаты работы обоих методов представлены в таблице 1.

Таблица 1. Результаты работы программ.

x	**y**	**Прогонка**		**Стрельба**	
0.0	*2*	*2*	*0*	*2*	*0*
0.1	*2.035630*	*2.035760*	*0.000130*	*2.036225*	*0.000595*
0.2	*1.996027*	*1.996174*	*0.000147*	*1.996961*	*0.000934*
0.3	*1.888453*	*1.888599*	*0.000147*	*1.890501*	*0.002049*
0.4	*1.722418*	*1.722550*	*0.000132*	*1.723401*	*0.000982*
0.5	*1.509094*	*1.509200*	*0.000106*	*1.509885*	*0.000790*
0.6	*1.260684*	*1.260758*	*0.000074*	*1.261217*	*0.000534*
0.7	*0.989789*	*0.989829*	*0.000040*	*0.990063*	*0.000274*
0.8	*0.708801*	*0.708810*	*0.000009*	*0.708869*	*0.000068*
0.9	*0.429343*	*0.429326*	*0.000016*	*0.429308*	*0.000035*
1.0	*0.161790*	*0.161790*	*0*	*0.161790*	*0*
			Max = 0.000147		***Max = 0.002049***

ВЫВОДЫ

В лабораторной работе было проведено сравнение двух методов решения дифференциальных уравнений второго порядка: метода прогонки и метода стрельбы. В результате сравнения было выявлено, что метод прогонки более точен и эффективен, особенно при большом количестве узлов разбиения. Однако, метод стрельбы может быть использован в тех случаях, когда метод прогонки не может быть применен.

Таким образом, при решении дифференциальных уравнений второго порядка рекомендуется использовать метод прогонки, особенно при большом количестве узлов разбиения. Метод стрельбы может быть использован только в случае, если невозможно применить метод прогонки.