Дисциплина: Численные методы
Лабораторное задание №3

Отчет

Тема: Решение линейной краевой задачи для дифференциального уравнения второго порядка методом универсальной дифференциальной прогонки.

Выполнила:
студентка 3 курса 62 группы
Пахомова П.В.

Проверила:
старший преподаватель Фролова О.А.

1. Постановка задачи

Назначение

Интегрирование на заданной сетке узлов уравнения:

$$y''(x)+p(x)y'(x)=q(x)y(x)+f(x)$$

с линейными краевыми условиями (3.2), (3.3) без оценки точности.

$$\alpha_1 y'(a)+\beta_1 y(a)=\gamma_1,\quad \alpha_1^2+\beta_1^2>0 \qquad (3.2)$$

$$\alpha_2 y'(a)+\beta_2 y(a)=\gamma_2,\quad \alpha_2^2+\beta_2^2>0 \qquad (3.3)$$

Описание параметров.

data - имя файла исходных данных;

p, q, f - имена процедур-функций с одним параметром, которые должны быть описаны в основной программе (функции p(x), q(x), f(x) вычисляют значения коэффициентов уравнения (4.2));

$$y''(x)+p(x)y'(x)=q(x)y(x)+f(x) \qquad (4.2)$$

rez - имя файла выходных данных;

Icod - код завершения программы, выходная переменная, принимающая следующие значения:

- Icod=0 – нет ошибок, решение получено;
- Icod=1 – решение не получено, задача имеет бесконечно много решений;
- Icod=2 – решение не получено, задача не имеет решения.

Замечание о структуре файла исходных данных.

1. Первая строка – значения коэффициентов граничных условий.
2. Вторая строка – значения левого и правого концов отрезка интегрирования, количество точек в заданной сетке узлов.
3. Строки с третьей и далее – номер узла, значение аргумента x в узле.

Замечание о структуре выходного файла.

Первая и последующие строки содержат номер точки, ее x - координату, значение вычисленного решения и значение производной решения в этой точке.

Никакого анализа точности полученных результатов не предполагается.

Для решения использовать метод Рунге-Кутта, решающий:

$$y' = f(x,y),\quad x\in[A,B] \quad (1)$$

$$y(c)=y_c,$$

Использовать формулы: $y_1 = y_0 + \frac{1}{4}(K_1 + 3K_2)$,

$$K_1 = hf(x_0, y_0),\ K_2 = hf\left(x_0 + \frac{2}{3}h, y_0 + \frac{2}{3}K_1\right).$$

2. Теоретическая часть

Вычисление значения функции y1 в точке x1 через x0, y0

$$y_1 = y_0 + \frac{1}{4}(K_1 + 3K_2), \qquad K_1 = hf(x_0, y_0),\ K_2 = hf\left(x_0 + \frac{2}{3}h, y_0 + \frac{2}{3}K_1\right).$$

Замечание

Для обратного хода h становится отрицательным, формула не меняется.

5.1. Метод удвоения и деления шага пополам

Пусть для получения приближенного решения задачи Коши (1),(2) выбран некоторый метод типа Рунге-Кутта и имеется в распоряжении способ оценки локальной погрешности выбранного метода. Как и ранее, обозначим через ε_{n+1} локальную погрешность метода в точке $x_n + h$, а приближенное значение, вычисленное с шагом h – y^k_{n+1}. Пусть наибольшая допустимая локальная погрешность шага интегрирования равна $\varepsilon > 0$. Тогда если

$$|\varepsilon_{n+1}| > \varepsilon, \tag{139}$$

то приближенное значение y^k_{n+1} считается неудовлетворительным по точности и выбирается новое значение шага

$$h^{(1)} = h/2. \tag{140}$$

С этим новым шагом по той же формуле Рунге-Кутта вычисляется новое значение $y^{k^{(1)}}_{n+1}$ в новой точке $x_n + h^{(1)}$. Если оценка локальной погрешности $\varepsilon^{(1)}_{n+1}$ на новом шаге $h^{(1)}$ опять превосходит заданную наибольшую допустимую локальную погрешность ε

$$|\varepsilon^{(1)}_{n+1}| > \varepsilon, \tag{141}$$

то шаг снова делится пополам

$$h^{(2)} = h^{(1)}/2. \qquad (142)$$

и вычисления повторяются. Так происходит до тех пор, пока локальная погрешность не станет меньше или равна ε при какой-то величине шага, которую обозначим h_n:

$$|\varepsilon_{n+1}| \le \varepsilon. \qquad (143)$$

Дальнейшее интегрирование уравнения будет производиться из точки $x_{n+1} = x_n + h_n$ с шагом h_{n+1}, который выбирается по следующему правилу. Если локальная погрешность ε_{n+1} на шаге $h_n = x_{n+1} - x_n$ удовлетворяет неравенству

$$|\varepsilon_{n+1}| < \frac{\varepsilon}{k}, \qquad (144)$$

где k – константа, то шаг интегрирования удваивается $h_{n+1} = 2h_n$. Если выполняется неравенство

$$\frac{\varepsilon}{k} \le |\varepsilon_{n+1}| \le \varepsilon, \qquad (145)$$

то шаг интегрирования не меняется,

$$h_{n+1} = h_n. \qquad (146)$$

Константа k полагается равной 2^s, где s – порядок используемой оценки локальной погрешности метода.

Таким образом осуществляется изменение шага интегрирования в зависимости от характера решения: там, где высока точность приближенного решения, шаг возрастает, а там, где заданная точность не достигается, шаг уменьшается. Изложенный выше алгоритм отражает лишь основную идею метода удвоения и деления шага пополам. В хороших программных реализациях этот метод содержит много особенностей, которые делают его более надежным и экономичным. В нижеследующих замечаниях приведены некоторые из этих особенностей.

Замечание 1. Для сокращения числа бесполезных проверок применимости шага интегрирования алгоритм модифицируется следующим образом. Если при интегрировании из точки x_n в точку $x_{n+1} = x_n + h_n$ шаг интегрирования уменьшался хотя бы один раз, то при выборе следующего значения шага интегрирования h_{n+1} удвоение предыдущего шага не происходит, даже если удовлетворяются соотношения (144). Иными словами, если на данном шаге была неудачная попытка применения шага интегрирования, то для следующего шага не допускается увеличение его длины.

Замечание 2. За два шага вперед проверяется точка конца интервала интегрирования с тем, чтобы исправить при необходимости величину шага, чтобы достигнуть конца отрезка интегрирования без слишком резких изменений в величине шага.

Замечание 3. Пользователь должен иметь возможность заказывать время счета по программе. Это можно сделать тремя способами: или программист ставит счетчик числа вычислений правой части уравнения, ограничив число вычислений правой части некоторым параметром, или ставит ограничение на длину шага интегрирования, или ставит ограничения на число шагов интегрирования. Реакция программы на превышение этих параметров оговаривается с заказчиком программы, в роли которого в учебном процессе выступает преподаватель, ведущий занятия.

Замечание 4. При выборе следующего шага число удвоений длины шага может быть ограничено некоторым параметром; обычно не допускают 5-кратного удвоения длины шага.

Замечание 5. Чтобы избежать зацикливаний программы, необходимо проверять, что в условиях машинной арифметики выполнятся неравенство

$$x_n + h_n \neq x_n, \tag{147}$$

т.е. что величина h_n не меньше расстояния от x_n до соседнего справа (при $h_n > 0$) или слева (при $h_n < 0$) вещественного числа, которое представлено на используемой ЭВМ. Для этого шаг интегрирования должен удовлетворять условию

$$|h| \geq \max\{\sigma, r\}, \tag{148}$$

где σ – наименьшее положительное число, представимое на данной ЭВМ, $\tau \cong macheps \cdot |x|$, $macheps$ – машинное эпсилон [7].

Значение, близкое к машинному эпсилон, вычисляется последующему простому алгоритму:

```
R := 1
пока  (1 + R) > 1
  нц
    R := R/2
  кц
macheps := R * 2
```
(149)

Замечание 6. Для того, чтобы избежать неприятностей, о которых говорилось в Замечании 5, можно ставить ограничения на число делений первоначального шага интегрирования. Например, если ограничить количество делений двадцатью, то допускается максимальное уменьшение шага в 10^6 раз.

Замечание 7. Выбор оптимального самого первого шага интегрирования – трудная задача, если решать ее в полном объеме. Самый простой выход – положить первоначальный шаг равным некоторой части отрезка интегрирования, надеясь на то, что метод удвоения и деления шага пополам довольно быстро выйдет на удовлетворительное значение длины шага.

3.2. Построение решения линейной самосопряженной краевой задачи для уравнения второго порядка

Рассмотрим самосопряженное дифференциальное уравнение второго порядка

$$(p(x)y'(x))'-q(x)y(x)=f(x),\quad x\in[a,b] \tag{3.11}$$

с граничными условиями (3.2), (3.3). Будем полагать, что функции $p(x),q(x),f(x)$ и граничные условия (3.2), (3.3) удовлетворяют известным условиям самосопряженности краевой задачи (см. п.2).

Потребуем, чтобы уравнение

$$u(x)p(x)y'=v(x)y+w(x) \tag{3.12}$$

Было эквивалентно (3.11). Для этого, продифференцировав уравнение (3.12) и исключив слагаемое $(py')'$ с помощью (3.11), получим

$$(pu'-v)y'=(v'-qu)y+(w'-fu)\ . \tag{3.13}$$

Далее, как и в п.3.1 нетрудно показать, что решение $y(x)$ краевой задачи (3.11), (3.2), (3.3) и его производную $y'(x)$ можно получить, решая линейную алгебраическую систему уравнений

$$\begin{cases} u(x)p(x)y'(x)-v(x)y(x)=w(x) \\ \alpha(x)p(x)y'(x)-\beta(x)y(x)=\gamma(x), \end{cases} \tag{3.14}$$

где прогоночные коэффициенты $u(x), v(x), w(x)$ определяются из задачи Коши для линейной однородной системы

$$\begin{cases} pu' = v \\ v' = qu \\ w' = fu \\ u(a) = \alpha_1 / p(a),\ v(a) = -\beta_1,\ w(a) = \gamma_1, \end{cases} \tag{3.15}$$

а коэффициенты $\alpha(x), \beta(x), \gamma(x)$ - из задачи Коши

$$\begin{cases} p\alpha' = \beta \\ \beta' = q\alpha \\ \gamma' = f\alpha \\ \alpha(b) = \alpha_2 / p(b),\ \beta(b) = -\beta_2,\ w(b) = \gamma_2 \end{cases} \tag{3.16}$$

В /3/ показано, что в случае существования единственного решения самосопряженной краевой задачи (3.11), (3.2), (3.3) линейная система уравнений (3.14) относительно $y(x), y'(x)$ имеет единственное решение.

Таким образом, метод универсальной дифференциальной прогонки для линейной самосопряженной задачи (3.11), (3.2), (3.3) сводится к решению двух линейных однородных задач Коши (3.15), (3.16) относительно прогоночных коэффициентов с последующим решением линейной алгебраической системы уравнений (3.14).

3. Алгоритм

Разберем шаги и функции по частям.

1. Функция solve()
  - Создается файл для записи результатов.
  - Вызываются функции solveFirstSystem(), solveSecondSystem() и solveUVWABYSystem() для решения систем уравнений.

2. Функция solveFirstSystem()
  - Инициализируются начальные значения u0, v0 и w0.
  - Значения добавляются в списки uValues, vValues и wValues.
  - В цикле вычисляются новые значения переменных с помощью формул, содержащих параметры p, q и f (функции зависящие от x).
  - Формулы для вычисления новых значений:

$$u_i = p \cdot (u_{i-1}) + v_{i-1}$$
$$v_i = q \cdot (u_{i-1})$$
$$w_i = f \cdot (u_{i-1})$$

- Вычисление значений u_i, v_i, w_i происходит в getFValues()

3. Функция getFValues()
  - Принимает начальное значение _f0, формулу fi, номер итерации i и флаг isBack.
  - В зависимости от значения флага isBack определяются значения A, B и C.
  - Создается объект IntegrationOrdinaryDifferentialEquation для решения дифференциального уравнения.
  - Результаты интеграции парсятся и возвращается полученное значение функции.

4. Формула для интеграции
- Непосредственная формула интегрирования определена в классе IntegrationOrdinaryDifferentialEquation. В ней используется метод Рунге-Кутты для вычисления значений.

$$y_1 = y_0 + \frac{1}{4}(K_1 + 3K_2), \qquad K_1 = hf(x_0, y_0),\ K_2 = hf\left(x_0 + \frac{2}{3}h, y_0 + \frac{2}{3}K_1\right).$$

4. Функция solveSecondSystem()
- Шаги и вычисления аналогичны solveFirstSystem(), но вычисления идут с конца отрезка до его начала.

5. Функция solveEquationSystemUVWABY:

1. Инициализация переменных и функции определителя:

- Определяется функция det, которая вычисляет определитель матрицы 2x2.

- Вычисляются значения D, D1 и D2 как определители матрицы, используя входные параметры функции.

2. Проверка на невырожденность системы:

- Проверяется, что определитель D не равен 0. Если равен, функция возвращает null и система решения не имеет, иначе продолжает вычисления.

3. Вычисление решения системы:

- Вычисляются значения x1 (y') и x2 (y) как результат деления соответствующих определителей на D.

- Возвращается пара значений (x1 (y'), x2 (y)).

6. функция solveUVWABYSystem:

1. Итерация по данным:

- Устанавливается начальное значение счетчика i=0.

- В цикле выполняются следующие действия для каждого элемента данных до достижения data.N из входных параметров:

2. Решение системы и запись результатов:

- Для текущего индекса i вызывается функция solveEquationSystemUVWABY, передавая соответствующие значения из списков uValues, vValues, wValues, aValues, bValues и yValues.

- Если решение не равно null, то результат записывается с помощью функции writeToOut, передавая узел данных data.nodeValues[i] (значения x_i, i=0,N) и значения решения (y, y').

4. Тестирование.

Входные данные:
$y''(x)+y'(x)+y(x)=x^3+3*x^2+6*x$

$y'(0)+y(0)=0$
$y'(1)+y(1)=4$

$0 \le x \le 1$

x: (0,0.0) (1,0.1) (2,0.2) (3,0.3) (4,0.4) (5,0.5) (6,0.6) (7,0.7) (8,0.8) (9,0.9) (10,1.0)

Аналитическое решение:
$y'(x) = 3*x^2$, $y(x) = x^3$

Результат:

x	y	y'
0.0	-0.5862665343095347	0.5862665343095347
0.1	-0.5362382201898374	0.5136393981713276
0.2	-0.49138681036757104	0.5163459884111278
0.3	-0.44357272831539807	0.5924234598396324
0.4	-0.38485317606054464	0.7416208256134796
0.5	-0.3072833434235233	0.9652082940545351
0.6	-0.20273030180726323	1.2658327217451393
0.7	-0.0626916259793328	1.6474142909612353
0.8	0.1218895497482088	2.1150817230714347
0.9	0.36082230205487986	2.6751454677396325
1.0	0.6648894642789198	3.33511053572108

Входные данные:
$y''(x)+y'(x)+y(x)=x^3+3*x^2+6*x$

$y'(0)+y(0)=0$
$y'(1)+y(1)=4$

$0 \le x \le 1$

x: (0,0.0) (1,0.05) (2,0.1) (3,0.15) (4,0.2) (5,0.25) (6,0.3) (7,0.35) (8,0.4) (9,0.45) (10,0.5) (11,0.55) (12,0.6) (13,0.65) (14,0.7) (15,0.75) (16,0.8) (17,0.85) (18,0.9) (19,0.95) (20,1.0)

Аналитическое решение:
$y'(x) = 3*x^2$, $y(x) = x^3$

Результат:

x	y	y'
0.00	-0.2033738511748	0.1220243107049
0.05	-0.2064697163286	0.1253143008076
0.10	-0.2075855787854	0.1427643017901
0.15	-0.2063456153198	0.1749880082361
0.20	-0.2023083041853	0.2225321099986
0.25	-0.1949615692327	0.2858739183369
0.30	-0.1837182609604	0.3654225246880
0.35	-0.1679118115230	0.4615235818983
0.40	-0.1467917871013	0.5744675921055
0.45	-0.1195189412799	0.7045014004849
0.50	-0.0851592500710	0.8518424518832
0.55	-0.0426762827631	1.0166952852176
0.60	0.0090788717771	1.1992697289329
0.65	0.0713810563668	1.3998003231457
0.70	0.1456489945350	1.6185666278659
0.75	0.2334691033608	1.8559142757734
0.80	0.3366268745088	2.1122768864680
0.85	0.4571480058663	2.3881992748377
0.90	0.5973521506603	2.6843627655651
0.95	0.7599231387487	3.0016138887992
1.00	0.8600009607480	3.1409983186909

Входные данные:
y''(x)=6*x

y'(0)+y(0)=0
y'(1)+y(1)=4

$0 \leq x \leq 1$

x: (0,0.0) (1,0.1) (2,0.2) (3,0.3) (4,0.4) (5,0.5) (6,0.6) (7,0.7) (8,0.8) (9,0.9) (10,1.0)

Аналитическое решение:
$y'(x) = 3*x^2$, $y(x) = x^3$

Результат:

x	y	y'
0.0	-0.14499999999999924	0.14499999999999924
0.1	-0.12479999999999937	0.1719999999999993
0.2	-0.09139999999999952	0.25299999999999934
0.3	-0.040599999999999685	0.3879999999999994
0.4	0.03180000000000027	0.5769999999999996
0.5	0.50.13000000000000012	0.8199999999999996
0.6	0.2582000000000003	1.1169999999999993
0.7	0.42060000000000036	1.4679999999999993
0.8	0.6214000000000004	1.8729999999999996
0.9	0.8648000000000005	2.3319999999999994
1.0	1.1550000000000002	2.8449999999999993

Вывод:
Размер сетки влияет на точность значений функции и ее производных.
В общем случае, чем меньше размер сетки, тем точнее аппроксимация значений функции и ее производных. Это связано с тем, что небольшой размер сетки лучше улавливает локальные изменения функции, повышая тем самым точность аппроксимации. Однако слишком малый размер сетки также увеличивает вычислительные усилия и требования к хранению данных, что может привести к снижению эффективности вычислений.

Однако, в некоторых задачах уменьшение размера сетки может не привести к значительному повышению точности