Дисциплина: Численные методы
Лабораторное задание №2

Отчет
Тема: Решение задачи Коши с заданной точностью с автоматическим выбором максимальной длины шага.

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

Проверила:
старший преподаватель Фролова О.А.

1. Постановка задачи

Назначение

Интегрирование обыкновенного дифференциального уравнения

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

с начальным условием

$$y(c)=y_c,$$

где точка **c** совпадает либо с началом, либо с концом отрезка интегрирования.

Входные параметры:

data – имя файлаисходных данных;
f – имя п роцедуры – функции с двумя параметрами, которая должна быть описана в программе (функция f – вычисляет значение правой части уравнения (1));
rez имя файла выходных данных

Замечание о структуре файла исходных данных.
Первая строка значения A, B, C, Yc
Вторая строка h минимальный допустимый шаг интегрирования;
E – наибольшее допустимое значение абсолютной погрешности.

Выходные параметры:

Замечание о структуре выходного файла.
Первая и последующие строки -x координата точки интегрирования, полученное приближенное значение в этой точке, погрешность в этой точке.
Последняя строка файла число точек интегрирования; число точек, в которых не достигается заданная точность; общее количество минимальных шагов интегрирования;

1) Конкретный вид метода Рунге-Кутта и способ оценки локальной погрешности приближенного решения на шаге определяется номером Вашего варианта.
2) Длина самого первого шага интегрирования берется равной (B-A)/10.
3) Для достижения заданной точности шаг h_n в каждой точке интегрирования выбирается методом удвоения и деления шага пополам. Если при делении шага он становится меньше $h_{\min}$, то деление недопустимо и шагу присваивается значение $h_{\min}$.
4) Для каждого вычисленного шага h_n делается проверка на конец интервала. Пусть интегрирование происходит слева направо, тогда проверяется выполнение неравенства $B-(x_n+h_n)<h_{\min}$. Если оно не удовлетворяется, то следующей точкой назначается x_n+h_n. Если неравенство справедливо, то для достижения конца отрезка интегрирования B необходимо сделать один или два шага, что регламентируется следующим правилом:
   a) Если $B-x_n \geq 2h_{\min}$, то делается два шага; $x_{n+1}=B-h_{\min}$, $x_{n+2}=B$;
   b) Если $B-x_x \leq 1{,}5h_{\min}$, то выполняется один шаг; $x_{n+1}=B$;
   c) Если $1{,}5h_{\min}<B-x_x<2h_{\min}$, то делается два шага; $x_{n+1}=x_n+(B-x_n)/2$, $x_{n+2}=B$.

Замечания по программированию.

1. После вычисления очередного приближенного значения решения оно сразу выводится в файл, занимать машинную память для хранения приближенных значений решения недопустимо.
2. Целесообразно написать подпрограмму, являющуюся интегратором уравнения на одном шаге.

2. Теоретическая часть

Вычисление значения функции y1 в точке x1 через x0, y0

$$y_1 = y_0 + \frac{1}{2}(K_1 + K_2), \tag{20}$$

$$K_1 = hf(x_0, y_0),\ K_2 = hf(x_0 + h, y_0 + K_1).$$

Формула уточнения решения для расчёта погрешности

$$y_1 = y_0 + \frac{1}{6}(K_1 + 4K_2 + K_3), \tag{30}$$

$$K_1 = hf(x_0, y_0),\ K_2 = hf\left(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}K_1\right),\ K_3 = hf(x_0 + h, y_0 - K_1 + 2K_2).$$

E = (30)-(20)

<u>Замечание</u>
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^h_{n+1}. Пусть наибольшая допустимая локальная погрешность шага интегрирования равна $\varepsilon > 0$. Тогда если

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

то приближенное значение y^h_{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 $y^{h^{(1)}}_{n+1}$ 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. Алгоритм

1. Начинаем функцию integrate()
- Она очищает файл для записи данных.
- Запускает цикл while, который продолжается, пока переменная flagEnd не станет истинной
- После функция записывает число точек, число точек, в которых не достигается заданная точность, общее количество минимальных шагов интегрирования .
    - В каждой итерации цикла вызывается функция calculateNext().

2. Функция getNextX() определяет следующее значение curX (текущего x) в соответствии с некоторыми условиями
- Если шаг h положительный, проверяется, чтобы B - (curX + h) было больше чем hMin. Если это условие выполняется, curX увеличивается на h.
- Если условие не выполняется, значение tmpX устанавливается равным текущему curX.
  Затем проверяются различные условия:
    - Если B - curX больше или равно 2.0 * h, это означает, что мы находимся на предпоследнем шаге, и устанавливаем curX в B h.
    - Если B - curX находится в диапазоне от 1.5 * h до 2.0 * h (или flagLastStep истинен), это означает, что мы находимся на последнем шаге, и устанавливаем curX в B.
    - Если B - curX находится между 1.5 * h и 2.0 * h (или flagLastStep ложен), мы устанавливаем curX в значение, равное текущему curX плюс половину разницы между B и curX.
    - h затем устанавливается равным разнице между текущим curX и tmpX.
- Если шаг h отрицательный, происходит процедура, но с условиями, проверяющими, чтобы A - (curX + h) было больше чем hMin, остальные действия аналогичны.

3. Функция calculateNext() выполняет следующие действия в цикле do-while
   - Увеличивает счетчик hMinCount, если h равно hMin.
   - Запоминает текущее значение curX в переменной tmpX.
   - Вызывает getNextX() для определения следующего значения curX.
   - Вычисляет значения k1_y, k2_y, k2_E и k3_E на основе текущих значений C и yC.

     k1_y, k2_y (20), k2_E, k3_E (30)
   - Вычисляет текущее значение acurY, используя формулу:

$$y' = h \cdot f(C, Yc)$$

   - Вычисляет текущее значение curY как

$$yC + 0.5 * (k1_y + k2_y).$$

   - Вычисляет текущее значение acurY как

$$yC + (1/6) * (k1_y + 4.0 * k2_E + k3_E).$$

   - Вычисляет текущее значение curE как разницу между acurY и curY.
   - Проверяет условия для изменения шага h:
     - Если абсолютное значение curE равно нулю или меньше E / k и isDivision ложно, удваивает h.
     - Проверяет, если значение curE больше E, и если да, проверяет условия для деления шага:
       - Если h не равно hMin, устанавливает isDivision в истину, возвращает curX к предыдущему значению tmpX, и устанавливает h равным половине его предыдущего значения или hMin, если абсолютное значение hMin больше или равно абсолютному значению половины предыдущего h.
       - Если h равно hMin, увеличивает счетчик incorrectXCount и добавляет узел с текущими значениями curX, curY и curE.
     - Если ни одно из условий не выполняется, добавляет узел с текущими значениями curX, curY и curE.

4. Функция addNode()

- открывает файл для записи
- добавляет строку с текущими значениями x, y и E
- закрывает файл
- Обновляет значения yC и C для следующей итерации

Тестирование

1. y` = x+y+1, [A,B] = [2..10] , x0=2, y(x0)=-4.0
   * y = -x - 2
   data:
   2.0 10.0 2.0 -4.0
   0.0001 0.00000000001
   rez:
   x: 2.008, y: -4.008, E: 0.0e+00
   x: 2.024, y: -4.024, E: 0.0e+00
   x: 2.056, y: -4.056, E: 0.0e+00
   x: 2.12, y: -4.12, E: 0.0e+00
   x: 2.248, y: -4.248, E: 0.0e+00
   x: 2.504, y: -4.5040000000000004, E: 0.0e+00
   x: 3.016, y: -5.016, E: 0.0e+00
   x: 4.040, y: -6.039999999999999, E: 0.0e+00
   x: 6.088, y: -8.087999999999992, E: 1.8e-15
   x: 10.0, y: -11.99999999999989, E: 9.2e-14
   Ex: 0, hMin count: 0, X count: 10

2. y` = 2*x, [A,B] = [2..10] , x0=10, y(x0)=100.0
   * $y = x^2$
   data:
   2.0 10.0 10.0 100.0
   0.0001 0.00000000001
   rez:
   x: 9.992, y: 99.840064, E: 0.0e+00
   x: 9.976, y: 99.52057599999999, E: 0.0e+00
   x: 9.944, y: 98.883136, E: 0.0e+00
   x: 9.88, y: 97.61439999999999, E: 0.0e+00
   x: 9.752, y: 95.10150399999999, E: 0.0e+00
   x: 9.496, y: 90.174016, E: 0.0e+00
   x: 8.984, y: 80.712256, E: 0.0e+00
   x: 7.96, y: 63.361599999999996, E: 0.0e+00
   x: 5.912, y: 34.951744, E: 0.0e+00
   x: 2.0, y: 4.0, E: 3.6e-15
   Ex: 0, hMin count: 0, X count: 10

3. y` = 2*x, [A,B] = [2..10] , x0=2, y(x0)=7.0
   * $y = x^2 + 3$
   data:
   2.0 10.0 2.0 7
   1.1 0.00000000001
   rez:
   x: 3.1, y: 12.610000000000001, E: -1.8e-15
   x: 5.3, y: 31.090000000000003, E: 0.0e+00
   x: 10.0, y: 103.0, E: 0.0e+00
   Ex: 0, hMin count: 1, X count: 3

4. y` = 2*x, [A,B] = [2..100] , x0=2, y(x0)=7.0
   * $y = x^2 + 3$
   data:
   2.0 100.0 2.0 7
   1.1 0.000000000000001
   rez:
   x: 3.1, y: 12.610000000000001, E: -1.8e-15
   x: 4.2, y: 20.64, E: 0.0e+00
   x: 5.300000000000001, y: 31.090000000000003, E: 0.0e+00
   x: 6.4, y: 43.96000000000001, E: 0.0e+00
   x: 8.600000000000001, y: 76.96000000000001, E: 0.0e+00
   x: 13.000000000000002, y: 172.00000000000003, E: 0.0e+00
   x: 17.400000000000002, y: 305.76000000000005, E: 0.0e+00
   x: 19.6, y: 387.1600000000001, E: 0.0e+00
   x: 21.8, y: 478.2400000000001, E: 0.0e+00
   x: 24.0, y: 579.0000000000001, E: 0.0e+00
   x: 26.2, y: 689.44, E: 0.0e+00
   x: 28.4, y: 809.5600000000001, E: 0.0e+00
   x: 30.599999999999998, y: 939.3600000000001, E: 0.0e+00
   x: 35.0, y: 1228.0, E: 0.0e+00
   x: 43.8, y: 1921.44, E: 0.0e+00
   x: 61.4, y: 3772.96, E: 0.0e+00
   x: 96.6, y: 9334.560000000001, E: 0.0e+00
   x: 100.0, y: 10003.000000000002, E: 0.0e+00
   Ex: 1, hMin count: 4, X count: 23

5. y` = 2*x, [A,B] = [-10..-2] , x0=-10, y(x0)=103.0
   * $y = x^2 + 3$
   data:
   -10.0 -2.0 -10.0 103
   0.5 0.000000000000001
   rez:
   x: -9.5, y: 93.25, E: 0.0e+00
   x: -8.5, y: 75.25, E: 0.0e+00
   x: -6.5, y: 45.25, E: 0.0e+00
   x: -2.5, y: 9.25, E: 0.0e+00
   x: -2.0, y: 7.0, E: 0.0e+00
   Ex: 0, hMin count: 1, X count: 5