Численное решение обыкновенных дифференциальных уравнений первого порядка

1 Теоретическая часть

1.1 Основные понятия

Обыкновенные дифференциальные уравнения - уравнения вида

$$F(x, y, y', y'' \dots, y^{(n)}) = 0 \tag{1}$$

, где $y = y(x)$ - неизвестная функция. **Порядком** такого уравнения называют порядок старшей производной n, входящей в уравнение. **Решением** такого уравнения будем называть функцию $y(x)$, n раз дифференцируемую и удовлетворяющую исходному уравнению во всех точках своей области опеделения.

Задачей Коши называют задачу о нахождении решения дифференциального уравнения, удовлетворяющего **начальным условиям** вида $y(x_0) = y_0, y'(x_0) = y_0^{(1)}, y''(x_0) = y_0^{(2)}, \dots, y_0^{(n-1)} = y_0^{(n-1)}$, где x_0 - фиксированная точка, а $y_0, y_0^{(1)}, \dots y_0^{(n-1)}$ - численные значения функции и её производных от первой до $(n-1)$-й включительно в данной точке.

Будем рассматривать уравнения вида

$$y' = f(x, y) \tag{2}$$

Задача Коши для данного уравнения состоит в отыскании решения, удовлетворяющего условию $y(x_0) = y_0$. Задача численного решения такого уравнения в таком случае сводится к отысканию значений функции $y(x)$ в некотором множестве точек на заданном отрезке $[a, b]$, причём, как правило, одна из границ этого отрезка совпадает с точкой x_0.

1.2 Метод Эйлера

Выберем на отрезке $[a, b]$ точки $\{x_0, \dots x_n\}$, расположенные на одинаковом расстоянии $h = \frac{b-a}{n}$ друг от друга (здесь и далее предполагается, что начальная точка x_0 совпадает с левой границей рассматриваемого отрезка). При этом $x_i = x_0 + i \cdot h$ для $i = 0, 1, \dots, n$. Будем находить приближенные значения $y_i = y(x_i)$ в этих точках.

По определению, производная функции записывается как

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{3}$$

Иначе это соотношение можно записать, исходя из свойств дифференциала:

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{4}$$

где $O(\Delta x)$ - величина погрешности нашего вычисления $f'(x)$, зависящая от выбора приращения Δx.

Используя введённые для нашего дифференциального уравнения обозначения, перепишем зависимость 4:

$$y'(x_0) = \frac{y(x_0 + h) - y(x_0)}{h} + O(h) \tag{5}$$

Тогда

$$y(x_1) = y(x_0 + h) = y(x_0) + h \cdot y'(x_0) + O(h^2) = y(x_0) + h \cdot f(x_0, y_0) + O(h^2) \tag{6}$$

Поскольку значение $O(h^2)$ нам неизвестно (известно только, что оно зависит от h квадратично, то есть уменьшается в 4 раза при уменьшении h в два раза), определим приближенное значение функции y в точке x_1:

$$y(x_1) \approx y(x_0) + h \cdot f(x_0, y_0) \tag{7}$$

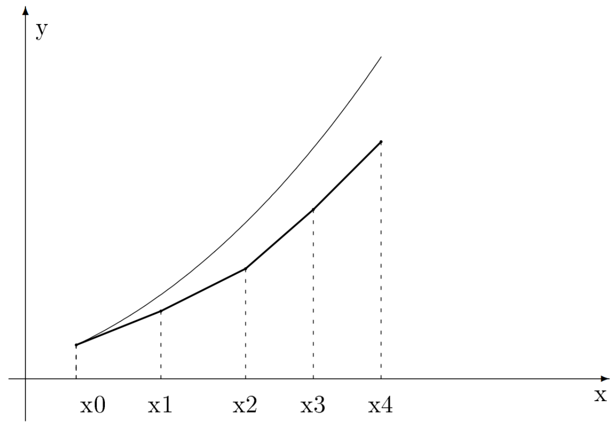

Рис. 1: Иллюстрация накопления погрешности при использовании метода Эйлера

Для дальнейших подсчётов придётся использовать приближенное значение функции. Обозначим приближенное (вычисленное нами) значение функции в точке x_i как $\bar{y}_i$. Тогда расчётная формула примет следующий вид:

$$\bar{y}_{i+1} = \bar{y}_i + h \cdot f(x_i, \bar{y}_i),\ i = 1, \ldots, n \tag{8}$$

Такой способ нахождения значений искомой функции называется *методом Эйлера*.

Заметим, что первое вычисленное значение имеет погрешность $O(h^2)$, однако вычисление последующих значений основывается на уже неточной величине y_1, что делает общую погрешность метода в точке b порядка $O(h)$.

Пример того, как происходит накопление погрешности, приведен на рисунке 1.

2 Практические рекомендации

2.1 Передача функции как параметра

При разработке программного обеспечения часто возникает необходимость использовать тот или иной алгоритм (например, вычисления интеграла или решения уравнения) к различным функциям. Было бы нерационально для каждой интегрируемой функции записывать свою функцию для вычисления интеграла, особенно если окажется, что функцию вычисления интеграла надо заменить или исправить.

В языке программирования PascalABC.Net существует возможность передавать функции как параметры для других функций. В этом случае при вызове вычисляющей функции (например, функции, вычисляющей интеграл) в качестве одного из параметров передаётся имя вычисляемой функции (той функции, от которой берётся интеграл), и далее вычисляющая функция может вызывать вычисляемую, используя то имя, которое указано в списке формальных параметров.

Приведём пример использования данной возможности:

```
{ Процедура табулирования функции.                  }
{ Производит вывод на экран значений функции на     }
{ отрезке от a до b, выводя на экран n значений.    }
{ В качестве параметров принимает a, b - границы    }
{ отрезка, n - количество значений, f - функция,    }
{ для которой надо построить таблицу значений.      }
procedure tabulate(a, b: Real; n: Integer;
                   f: function(t: Real):Real);
begin
    var h = (b - a) / (n - 1);
    for var i := 0 to n - 1 do begin
        var x := a + i * h;
{ Используется форматированный вывод: для каждого   }
{ числа резервируется 7 десятичных позиций, и для   }
{ каждого числа выводится 3 знака после запятой     }
        writeln(x:7:3, '|', f(x):7:3);
```

```
    end;
end;

function f1(x: Real): Real;
begin
    result := x * x;
end;

function f2(x: Real): Real;
begin
    result := 2 - x;
end;

{ Вызов подпограммы табулирования с двумя различными }
{ функциями.                                         }
begin
    tabulate(0, 10, 10, f1);
    tabulate(-10, 10, 21, f2);
end.
```

Обратим внимание на то, что, хотя в программе нет функции f(x), она всё равно работает и работает верно - сначала выводится результат табулирования для f1, затем для f2. Это происходит потому, что в процедуре tabulate мы создали формальный параметр f, который имеет тип функции; конкретно - function(t: Real):Real - то есть функции, которая принимает один параметр типа Real и возвращает значение типа Real. Далее, при вызове процедуры tabulate мы передаём ей как границы отрезка, так и имя той функции, которую надо табулировать. При работе программы в первом случае при вызове функции f реально будет вызвана функция f1, во втором - функция f2.

Следует обратить внимание, что имя переменной, передаваемой в функцию-параметр, может отличаться от имени, указанного в «настоящей», передаваемой функции (в примере происходит именно это). Важно лишь совпадение типов передаваемых переменных, их количество и порядок, в котором они указываются, а также тип возвращаемого значения функции.

2.2 Подсчёт количества вызовов функции

Методы вычисления приближенных значений решения дифференциального уравнения следует сравнивать по зависимости погрешности от шага разбиения и по вычислительной сложности. Вычислительную сложность можно оценить с помощью замеров количества вызовов функции $f(x, y)$ в зависимости от шага разбиения. Эти замеры можно производить с помощью глобальной переменной, величина которой будет увеличиваться на 1 при каждом вызове функции. Пример реализации этой идеи показан ниже:

```
{ Здесь будет храниться количество вызовов }
var CallCount: Integer;

function f(x,y: Real): Real;
begin
{ Первым делом увеличиваем счётчик вызовов }
        CallCount := CallCount + 1;
        result := x * x + y * y;
end;

{ Данная процедура будет вычислять значения}
{ частного решения ОДУ на отрезке от a до b}
{ для n точек на этом отрезке             }
procedure odeEuler(a, b: Real; n: Integer;
                   f: function(x, y: Real): Real
                   y0: Real);
begin
{ ——————————————————————————————————————— }
{ Здесь должна быть реализация метода Эйлера }
{ ——————————————————————————————————————— }
end;
```

```
begin
{ Первым делом обнуляем счётчик                }
        CallCount := 0;
{ Вычисляем значения частного решения для      }
{ 1000 точек                                   }
        odeEuler(0, 10, 1000, f, 1);
        writeln('Функция f(x, y) была вычислена ',
             CallCount, 'раз');
end.
```

2.3 Проверка корректности

Заметим, что при $f(x,y) = g(x)$ дифференциальное уравнение принимает вид $y' = g(x)$. Очевидно, решением такого уравнения будет $y = G(x) = \int g(x)dx$. В случае постановки задачи Коши получаем:

$$y = \int_{x_0}^{x} g(t)dt + y_0$$

Очевидно, метод Эйлера, примененный к такому уравнению, будет выдавать те же значения, что и метод левых прямоугольников при интегрировании функции с заданным шагом. Следовательно, один из способов проверки своей программы - для уравнения вида $y' = g(x)$ сравнить результаты вычисления по методу Эйлера и прямым интегрированием.

Другой, ещё более правильный, но более трудоёмкий способ - найти аналитическое решение какого-либо уравнения и сравнить значения полученной функции со значениями, выдаваемыми программой. При этом следует учесть, что некоторые расхождения всё равно будут появляться - важно, чтобы общая картина решения соответствовала истине.

Для примера можно взять уравнение $y' = y/x$; это уравнение с разделяющимися переменными, его решение тривиально:

$$\frac{dy}{y} = \frac{dx}{x}, \ln y = \ln x + C, y = C_1 x$$

Возьмём для этого уравнения условие $y(1) = 1$: тогда частным решением (решением задачи Коши) этого уравнения будет функция $y = x$. При использовании данного уравнения и такого начального условия программа должна выдавать, в целом, схожие значения для x и y.

3 Задание

Требуется написать программу, которая будет получать на вход номер уравнения, начало и конец отрезка, количество точек на отрезке и начальное значение функции. Программа должна выводить на печать координаты точек, в которых происходит вычисление решения, и значения решения в этих точках.

Уравнения заданы в таблице:

№	Уравнение
1	$y' = x^2$
2	$y' = y^2$
3	$y' = xy$
4	$y' = y - x$
5	$y' = sin(x) + y$
6	$y' = \frac{x^2-3x+9}{y}$

В программе должна присутствовать функция odeSolve, принимающая в качестве параметров координаты начала и конца отрезка, на котором производится вычисление решения, количество точек для вычисления, функцию, находящуюся в правой части, а также начальное значение решения. Допускается написание функций для уравнений, не входящих в список.