龙格-库塔方法与亚当姆斯方法相结合求解常微分方程

第七小组：任浩辰　刘阳

科目：数值分析

日期：2020 年 5 月 2 日

1　实验目的

通过编程实践，熟练掌握龙格-库塔和亚当姆斯方法如何求解常微分方程，验证亚当姆斯方法求解的正确性，以及亚当姆斯预报-矫正系统的作用。

2　实验步骤

1. 验证亚当姆斯方法求解常微分方程的正确性
2. 验证亚当姆斯预报-校正系统的作用

3　实验内容

3.1　四阶龙格-库塔方法

四阶龙格-库塔经典格式如下：

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_{n+\frac{1}{2}}, y_n + \frac{h}{2}K_1) \\ K_3 = f(x_{n+\frac{1}{2}}, y_n + \frac{h}{2}K_2) \\ K_3 = f(x_{n+1}, y_n + hK_3) \end{cases} \tag{1}$$

在 $[x_n, x_{n+1}]$ 区间上多预报几个点的斜率，然后将他们加权平均作为平均斜率，则可以构造出更高精度的公式，这就是四阶龙格库塔方法的设计思想。

3.2　亚当姆斯方法

亚当姆斯方法的设计思想是充分利用计算 y_{n+1} 之前已得到一系列结点 $x_n, x_{n_1}, \ldots$ 上的斜率值来减少计算量。譬如，可以用 x_n, x_{n-1} 两点的斜率的加权平均作为区间 $[x_n, x_{n+1}]$ 上的平均斜率，于是可以设计出如下二阶亚当姆斯格式：

$$y_{n+1} = y_n + \frac{h[3y_n^{'} - y_{n-1}^{'}]}{2}$$

类似的，可以得到三阶、四阶亚当姆斯格式：

$$y_{n+1} = y_n + \frac{h[23y'_n - 16y'_{n-1} + 5y'_{n-2}]}{12}$$

$$y_{n+1} = y_n + \frac{h[55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3}]}{24}$$

同样，也可导出如下隐式的二阶、三阶和四阶亚当姆斯格式：

$$y_{n+1} = y_n + \frac{h[y'_{n+1} + y'_n]}{2}$$

$$y_{n+1} = y_n + \frac{h[5y'_{n+1} + 8y'_n - y'_{n-1}]}{12}$$

$$y_{n+1} = y_n + \frac{h[9y'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2}]}{24}$$

将显式和隐式两种亚当姆斯格式相匹配，可构成下列亚当姆斯预报-校正系统

$$\begin{cases} \text{预报} & \bar{y}_{n+1} = y_n + \frac{h[55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3}]}{24} \\ & \bar{y}'_{n+1} = f(x_{n+1}, \bar{y}_{n+1}) \\ \text{校正} & y_{n+1} = y_n + \frac{h[9y'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2}]}{24} \\ & y'_{n+1} = f(x_{n+1}, y_{n+1}) \end{cases} \tag{2}$$

4 实验过程及主要代码

4.1 验证亚当姆斯方法求解常微分方程的正确性

首先选择偏微分方程：$y' = y - \frac{2x}{y}$，此方程具有解析解：$y = \sqrt{1 + 2x}$。首先完成龙格-库塔方法的函数编写，作为亚当姆斯方法的启动值，然后按照亚当姆斯格式的方程计算出结果，与精确值相比较。

4.2 验证亚当姆斯预报-校正系统的作用

按照亚当姆斯校正公式，计算亚当姆斯方法的校正值，并比较预测值、校正值与精确值之间的差距大小，得出结论。

4.3 主要代码

偏微分方程的计算函数如下：

```
/**
 * 原偏微分方程计算函数
 * @param x x的值
 * @param y y的值
 * @return 函数计算结果
 */
double f(double x, double y){
  double ans;
```

```
9
10    ans = y - 2 * x / y;
11
12    return ans;
13  }
```

龙格库塔方法的函数编写如下：

```
1   /**
2   * 龙格-库塔方法
3   * @param x0 初始条件
4   * @param x1 表示xn+1
5   * @param y0 初始条件
6   * @return 表示yn+1
7   */
8   double Runge_Kutta(double x0, double x1, double y0){
9     double K1, K2, K3, K4;
10    double h = x1-x0;
11    double ans;
12
13    K1 = f(x0, y0);
14    K2 = f(x0 + h/2.0, y0 + h/2.0 * K1);
15    K3 = f(x0 + h/2.0, y0 + h/2.0 * K2);
16    K4 = f(x1, y0 + h * K3);
17
18    ans = y0 + h/6.0 * (K1 + 2*K2 + 2*K3 + K4);
19
20    return ans;
21  }
```

亚当姆斯方法即输出结果函数如下：

```
1   /**
2   * 亚当姆斯方法
3   * @param a x属于(a, b)
4   * @param b x属于(a, b)
5   * @param y0 初始值
6   * @param h 步长
7   */
8   void Adams(double a, double b, double y0, double h){
9     double x[100],y[100];
10    x[0] = a;//储存x值
11    y[0] = y0;//储存预测值
12    double ans[100];//储存校正值
13    ans[0] = y0;
14    int n = (int)((b - a) / h);
15
16    for (int i = 1; i < 4; ++i) {
17      x[i] = a + i * h;
```

```
    ans[i] = Runge_Kutta(x[i - 1], x[i], ans[i - 1]);//龙格库塔方法求前四个值
    y[i] = ans[i];
  }

  for(int i = 4; i <= n; i++){

    x[i] = a + i * h;

    y[i] = y[i-1] + h/24.0 * (55 * f(x[i-1], y[i-1]) - 59 * f(x[i-2],
    y[i-2]) + 37 * f(x[i-3], y[i-3]) - 9 * f(x[i-4], y[i-4]));//计算预测值
  }

  for(int i = 4; i <= n; i++){
    ans[i] = ans[i-1] + h/24.0 * (9 * f(x[i], y[i]) + 19 * f(x[i-1],
    y[i-1]) - 5 * f(x[i-2], y[i-2]) + f(x[i-3], y[i-3]));//计算校正值
  }

  //输出结果
  cout << endl;
  cout << " Xn   |      预测      |       校正       |       真实" << endl;
  cout << "-------------------------------------------------------" << endl;
  for(int i = 0; i <= n; i++) {
    if (i == 4) cout << "-------------------------------------------------------"
        << endl;
    cout << fixed << setprecision(1) << x[i]  << "  | ";
    if (i >= 4)
    cout << fixed << setprecision(8) << y[i] << "  | \t";
    else
    cout << "\t\t\t";
    cout << fixed << setprecision(8) << ans[i] << "  |  "  << sqrt(1 + 2 * (a
         + i * h)) << endl;

  }
  cout << "-------------------------------------------------------" << endl;
}
```

主函数如下：

```
int main() {
  double a = 0, b = 1.0;           //x属于(a, b)
  double h = 0.1, y0 = 1.0;        //h步长, y0 = f(x0)

  cout << "偏微分方程: y'= y - (2x/y)" << endl;
  cout << "解析解: y=sqrt(1+2x)" << endl;

  Adams(a, b, y0, h);
  return 0;
}
```

5 实验结果及分析

程序运行结果如图

```
偏微分方程：y'= y-(2x/y)
解析解：y=sqrt(1+2x)

 Xn  |      预测      |      校正      |      真实
-------------------------------------------------
0.0  |                 1.00000000  |  1.00000000
0.1  |                 1.09544553  |  1.09544512
0.2  |                 1.18321675  |  1.18321596
0.3  |                 1.26491223  |  1.26491106
-------------------------------------------------
0.4  | 1.34155176  |    1.34164136  |  1.34164079
0.5  | 1.41404642  |    1.41419688  |  1.41421356
0.6  | 1.48301891  |    1.48319449  |  1.48323970
0.7  | 1.54891887  |    1.54911043  |  1.54919334
0.8  | 1.61211643  |    1.61232061  |  1.61245155
0.9  | 1.67291703  |    1.67312947  |  1.67332005
1.0  | 1.73156975  |    1.73178751  |  1.73205081
-------------------------------------------------
```

在计算中，偏微分方程为 $y' = y - \frac{2x}{y}$，此方程的解析解为 $y = \sqrt{1+2x}$。计算了区间 [0,1] 上，以 x=0 时的函数值作为 y_0 初始值，步长 h=0.1 的 11 个点的值。表格从左到右每一列分别为：x_n 的值，预测值，校正值，真实值。

从运行结果可以看出，亚当姆斯方法精确度较高，比较接近真实值。而当使用亚当姆斯预报-校正系统时，可以得到比预测值更加精确的计算值，在结果中列为校正值一栏。

6 实验体会

在这次的实验过程中，利用 c++ 编写了龙格-库塔算法和亚当姆斯算法的有关代码，增加了编程能力，也进一步理解了这两种常微分方程的计算方法。

龙格-库塔方法是显式的自开始方法，而且精度比较高，具有易于改变步长等优点。但使用龙格-库塔方法时，每一步需要多次计算函数 $f(x,y)$ 的值，计算量大。而且由于龙格-库塔方法的导出基于泰勒展开，所以要求函数具有较高的光滑性。对于光滑性不太好的解，最好采用低阶算法而将步长 h 取小。

亚当姆斯方法的计算量比龙格-库塔方法少，却具有同样的精度，但必须用龙格库塔方法提供开头几个函数值。