DOI:10.13274/j.cnki.hdzj.2007.07.037

中图分类号:TP301.6　　文献标识码:A　　文章编号:1009-2552(2007)07-0092-03

基于C++的常微分方程欧拉解法的误差分析

崔均亮，邓易冬，徐宏坤

（成都理工大学应用核技术与自动化工程学院，成都 610059）

摘　要：工程计算中常微分方程的欧拉解法存在一定的误差，而实际往往要求其满足一定的精度。以工程中的一个常微分方程为例，运用C++程序对其欧拉解法的误差进行分析。
关键词：常微分方程；欧拉解法；误差分析

Error analysis of Euler method for ordinary differential equation based on C++

CUI Jun-liang，DENG Yi-dong，XU Hong-kun

（School of Nuclear Technology and Automation Engineering, Chengdu University of Technology, Chengdu 610059, China）

Abstract: In Euler method in engineering calculation for ordinary differential equations exists certain error, and the practice often required to meet a certain accuracy. This paper takes an ordinary differential equation of project for example to analyze its error by method of C++.

Key words: ordinary differential equation; Euler method; error analysis

0　引言

常微分方程在科学技术和工程上有着广泛的应用。然而只有很少简单的常微分方程能够用初等方法求解，其大多数的解通常是利用数值方法求出的。常用的方法有欧拉法、单步法、Runge－Kutta 法等。其中欧拉法有着简单易算的特点，由 y_0 可直接计算出 y_1，由 y_1 可直接算出 y_2，…，无需用迭代法求解任何方程。因此，这种方法在实际中得到了广泛的运用。由欧拉法的几何意义可知，欧拉法是以折线代替积分曲线，因此这种算法会存在一定的误差。在工程计算上往往要求满足一定的精度，其计算误差的大小通常以标准差来衡量。本文以工程中的一个常微分方程为例，运用C++程序来分析欧拉法的误差大小。

1　基本原理

对于常微分方程：

$$\begin{cases}\dfrac{dy}{dx}=f(x,y)\\ y\big|_{x=x_0}=y(x_0)\end{cases}\qquad X_0<x\le X$$

其欧拉解法为：

由　$\dfrac{dy}{dx}=f(x,y)$

得　$y(x_0+h)-y(x_0)=\int_{x_0}^{x_0+h}f(x,y)dx$

当 h 的值较小时，可用矩形面积近似代替曲边梯形 $\int_{x_0}^{x_0+h}f(x,y)dx$ 的面积，即用 $f(x_0,y_0)$ 代替 $[x_0,x_0+h]$ 上的 $f(x,y)$ 的值，由此可得：

$$y(x_0+h)\approx y(x_0)+hf((x_0,y_0)$$

令：$y(x_0)=y_0$，$y(x_0+h)=y_1$，$x_1=x_0+h$ 有：

$$\begin{cases}y_{n+1}=y_n+hf(x_n,y_n)\quad n=0,1,2,\cdots\\ y\big|_{x=x_0}=y(x_0)\end{cases}$$

此式即为欧拉方法的计算公式。

从该解法中可以看出，欧拉解法中无需用迭代法求解任何方程，因此由于误差的积累而造成的偏差比较小。下面来分析一下减少运用C++程序运算时带来的系统误差的方法。由于计算机不论运算器能进行多少位的运算，总是有限位二进制运算，因此

收稿日期：2006－11－03
作者简介：崔均亮(1974－)，男，1996 年毕业于上海理工大学，现为成都理工大学硕士研究生，主要研究方向为智能仪器。

对十进制的数字运算总会出现舍入误差以及在计算中误差的传递。而在实际编程时是不考虑计算机的舍入误差以及它在计算中误差的传递造成的影响的，这势必带来计算的误差。为了进一步提高计算精度，在程序设计中采用双精度的数据类型，这样有效地减小了舍入误差。

2 误差理论

由于欧拉法采用 $f(x_0, y_0)$ 来代替 $f(x, y)$ 的值，因些对运算结果可用误差理论来进行分析，把 $f(x, y)$ 的值当成真实值，而把 $f(x_0, y_0)$ 的值当成测量值。由误差理论可知，标准差有下式：

$$\sigma = \sqrt{(\delta\frac{2}{1} + \delta\frac{2}{2} + \delta\frac{2}{3} + \cdots + \delta\frac{2}{n})\backslash n} = \sqrt{(\sum_{i=1}^{n} \delta\frac{2}{i})\backslash n}$$

式中 n 为测量次数，δ 为测得值与被测值之差。当被测量的真值为未知时，上式不能求得标准差。实际上，在有限次测量情况下，可用残余误差 υ 来代替真实误差，从而得到标准差的估计值。由于计算机的舍入误差，可以把由实际函数计算出的值等同于残余误差。即标准差可用下式由残余误差求得：

$$\sigma = \sqrt{(\sum_{i=1}^{n} \upsilon_i^2)\backslash n - 1}$$

3 运算实例

下面以一个某工程中用到的常微分方程实例来说明欧拉解法的误差大小，其常微分方程为：

$$\begin{cases} \frac{dy}{dx} = \frac{1}{1+x^2} - 2y^2 & x \in (0, 1] \\ y(0) = 0 \end{cases}$$

其解析解为 $y = x/(1 + x^2)$，设步长 $h = 0.1$，通过 C++ 程序来求出其标准差的大小，程序代码如下：

```
#include<iostream.h>
#include<iomanip.h>
#include<math.h>
const double h=0.1; //定义步长常量
void main( )
{
    double *y_numerical=new double[11]; //开辟存贮数值解的数组
    double *y_precision=new double[11]; //开辟存贮精确解的数组
    double *subtract=new double[11]; //开辟存贮数值解与精确解的差的数组
    double *subtract_square=new double[11]; //开辟存贮数值解
                                            //与精确解的差的平方数组
    y_numerical[0]=0;
    for(int i=0;i<11;i++) //数值解求值
    {
    y_numerical[i+1]=y_numerical[i]+h*(1/(1+h*i*h*i)-2*y_numerical[i]*y_numerical[i]);
    }
    for(int k=0;k<11;k++) //精确解求值
    {
      y_precision[k]=h*k/(1+h*k*h*k);
    }
    for(int j=0;j<11;j++) //数值解与精确解的差的绝对值
    {
      double a=y_numerical[j]-y_precision[j];
          if(a>0)
          {
            !subtract[j]=a;
          }
          else
          {
          subtract[j]=-a;
          }
    }
    for(int n=0;n<11;n++) //数值解与精确解的差的平方
    {
      subtract_square[n]=subtract[n]*subtract[n];
    }
    double sum=0;
    for(int m=0;m<11;m++) //求平方和
    {
      sum=sum+subtract_square[m];
    }
    double standard_subtract=sqrt(sum/(11-1)); //求标准差
    cout.setf(ios::left);
    cout<<setw(5)<<"X"<<setw(15)<<"精确解 yn"<<setw(15)<<"数值解 y(xn)";
    cout<<setw(15)<<"|yn-y(xn)|"<<setw(15)<<"|yn-y(xn)| 的平方"<<endl;
    for(int nn=0;nn<11;nn++)
```

```
    {
        cout<<setw(5)<<h*nn<<setw(15)<<
y_numerical[nn];
        cout<<setw(15)<<y_precision[nn];
        cout<<setw(15)<<subtract[nn]<<setw
(15)<<subtract_square[nn];
        cout<<endl;
    }
    cout<<endl;
    cout<<"标准差为:  "<<standard_subtract
<<endl;
    cout<<endl;
}
```

4 结束语

运行程序，得到如下图 1 所示的数据。可以看出，该常微分方程欧拉解法的精确解 y_n 和其数值解 $y(x_n)$ 很接近，它们的偏差最大值为 0.2594。在上例中我们取了 10 个精确解和数值解，得到它们的标准差为 0.0197。由此可知，常微分方程的欧拉解法对大量计算数据来说精度是相当高的，可以满足绝大多数工程上的要求。由于欧拉法的稳定性和收敛性，可以通过减小 h 的值来进一步提高其精度。

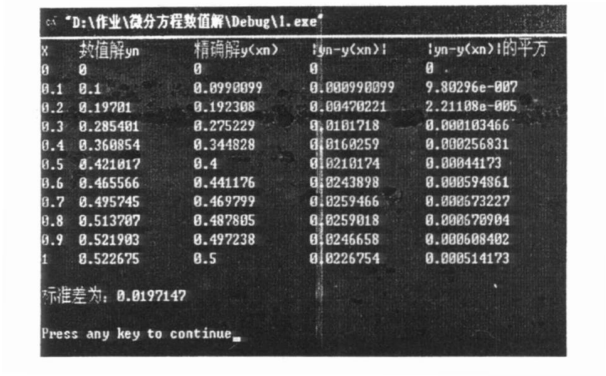

图 1　程序运行结果

参 考 文 献:

[1] 哈克布思. W. 多重网格方法[M]. 北京: 科学出版社, 1988.

[2] 南京大学. 偏微分方程数值解法[M]. 北京: 科学出版社, 1990.

[3] 李荣华, 冯果忱. 微分方程数值解法[M]. 北京: 人民教育出版社, 1997.

[4] 李桂成, 等. 测量误差与数据处理原理[M]. 吉林: 吉林大学出版社, 1991.

[5] 袁慰平, 孙志忠, 吴宏伟. 计算方法与实习[M]. 南京: 东南大学出版社, 2000.

[6] 徐孝凯. C++ 语言基础教程[M]. 北京: 清华大学出版社, 2001.

[7] 何旭初, 苏煜诚. 计算数学简明教程[M]. 北京: 人民教育出版社, 1980.

责任编辑: 肖 滨

(上接第 91 页)织序。这里以 m 序列的线性移位寄存器状态序列作交织序，即当本原多项式确定后，其初始状态可以有 L－1 种(初态不能为全“0”)。

设 $L = 2^4 = 16$，4 级 m 序列的本原多项式为 $f(x) = x^4 + x^3 + 1$，当初态为1111时，将各状态按二进制计数，所得结果即为置换序列，置换关系为:

输入序列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
置换后序列	0	15	7	3	1	8	4	2	9	12	6	11	5	10	13	14

卷积码结合伪随机交织编码需注意卷积码参数和交织参数要满足以下条件:

(1) 交织长度 $L \geqslant n \times (N+1)$。

(2) L 能被 n 整除。

(3) 卷积码序列中分组起点与交织分组起点一致。

4 卷积与交织联合编码性能比较

在 Matlab 环境下，用以上简单马尔可夫链模型模拟具有突发错误的信道，比较错误率，原始数据则采用一段符合 Hdlc 规程的数据(模拟 VSAT 网络控制信号)，采用上述卷积编译码方法，共对以上 2 种不同的交织器进行了仿真，仿真结果如图 4 所示。仿真结果显示伪随机交织器的性能要优于矩阵交织器。

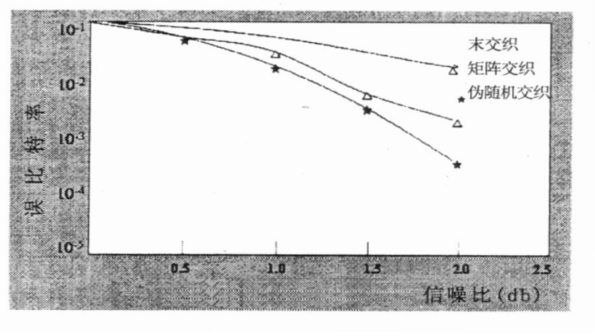

图 4　仿真结果

5 结束语

为提高卫星 VSAT 网络通信的可靠性，本文提出了一种差错控制方案。仿真结果表明: 卷积码与交织联合编码可以较好地保障传输的可靠性，并且算法简单易实现，效率高，满足 VSAT 系统的通信需要。

参 考 文 献:

[1] 杨运年. VSAT 卫星通信网[M]. 北京: 人民邮电出版社, 1997.

[2] Costello D J, Hagenauer J, Imai H, et al. Applications of Error－Control Coding[J]. IEEE Trans. Inform. Theory, 1998, 44(6): 2531－2560.

[3] 王新梅, 肖国镇. 纠错码——原理与方法[M]. 西安: 西安电子科技大学出版社, 1996.

责任编辑: 肖 滨