数值分析 大作业 2
编写求$ln(x)$的函数
项目报告

晏筱雯

自 42

2014011459

2016.12.21

目录

1 作业要求

编写求 ln(x)的函数要求：

（1）采用方法：

a. Taylor 展开（最佳或近似最佳逼近）；

b. 数值积分；

c. 非 Taylor 展开的函数逼近方法（选作）；

（2）给出小数点后 20 位精度的结果（需要自行编写能够达到指定任意精度的四则运算）；

（3）分析选用方法的计算代价、收敛速度等；

（4）分析选用方法的方法误差和存储误差对最终结果的影响（除法带来误差忽略不计）。

说明：

（1）x 的取值范围为[1,100]，输入最多 5 位有效数字；

（2）具体要求参见第一次大作业要求；

（3）自行编写全部算法；

（4）报告内容应完整，包括：上一页“要求”中所列内容、程序框图、实验结果及分析等。

（5）对于大作业抄袭与被抄袭者，以 0 分处理。

2 编译环境和语言

Windows10,Visual Studio 2012,C#.

3 需求分析

3.1 lnx的数值解法

本次大作业要求实现lnx的求取，根据课上所学的内容和作业要求，实现方法可以分为 Taylor 展开法、数值积分的方法和非 Taylor 展开的函数逼近方法。

3.1.1 Taylor 展开

由 Taylor 展开的公式我们可以得到：

$$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\cdots+(-1)^{n-1}\cdot\frac{x^n}{n}+\cdots$$

由 Taylor 级数的性质我们可以知道，此展开的收敛域为$x\in(-1,1)$，故通过这一展开式我们只能得到$(ln(0),ln(2))$的值。因此，在利用 Taylor 展开的方法求取lnx的值时，我们可以采取换元 Taylor 展开和减半 Taylor 展开两种方法。

3.1.1.1 换元 Taylor 展开

令$x=\frac{1+y}{1-y}$，则当$y\in(-1,1)$时$x\in(0,\infty)$，即y的定义域符合 Taylor 展开的收敛域，而此时

的x的范围刚好符合lnx的定义域，因此我们有：

$$ln\,x = ln\left(\frac{1+y}{1-y}\right) = ln(1+y) - ln(1-y)$$

对$ln(1+y)$和 $ln(1-y)$分别 Taylor 展开即可得到lnx的值。

3.1.1.2 减半 Taylor 展开

令$x = a \cdot 2^r$，其中$a \in (0,2)$，r是非负整数，于是有

$$lnx = ln(\,a \cdot 2^r) = ln\,a + ln2^r = lna + rln2$$

对于$ln\,a$，有

$$lna = ln\big(1 + (a-1)\big) = (a-1) - \frac{(a-1)^2}{2} + \frac{(a-1)^3}{3} - \cdots + (-1)^{n-1} \cdot \frac{(a-1)^n}{n} + \cdots \qquad ①$$

如果我们事先存储了$ln2$的值，再通过迭代算出a和r的值，我们就可以用上述两个公式算出lnx。

在①式中，我们取其前n项来近似lna，于是

$$R_n(a-1) = (-1)^n \cdot \frac{(a-1)^{n+1}}{n+1} + \cdots$$

$$\therefore \quad |R_n(a-1)| \le \frac{(a-1)^{n+1}}{n+1} \qquad ②$$

由②式我们可以确定n，以使求取的lnx符合精度要求。

3.1.2 数值积分

由于$(lnx)' = \frac{1}{x}$,故$lnx = \int_1^x \frac{1}{a}da$，利用数值积分的方法，我们就可以求出$lnx$的值。课上讲到了很多求数值积分的方法，本次大作业中我采用了比较简单的复化梯形公式的方法。

3.1.3 方程求解

3.1.3.1 四阶龙格—库塔公式

设$y = lnx$,则

$$y' = \frac{1}{x}$$

通过解此常微分方程，我们可以求出lnx的值。对于解常微分方程的方法，我们采用经典的四阶龙格—库塔法（R—K 法）。

3.1.3.2 牛顿迭代公式

对于方程

$$e^y = x \qquad ③$$

易知其解为$y = lnx$，故求取lnx的问题就转化为求方程③的数值解的问题。本次大作业中，我采用了经典的牛顿迭代公式来求解。

3.2 数据结构

这次大作业要求给出小数点后 20 位精度的结果，而 C#的 double 型数据只有 16 位有效数字，不能满足精度要求，所以需要自己编写高精度数算法。本次大作业中，我把大数封装成一个 60 位的整数数组，前 20 位是高精度数的整数部分，后 40 位是高精度数的小数部分，并创建了一个高精度数类（BigNumber），其包含的函数如下：

（1）大数类的构造函数和复制函数：

- BigNumber()
- copy(Ln.BigNumber)

（2）大数类、int 型、double 型、string 型数据互相转换的函数：

- BnToString()
- StringToBn(string)
- Trans(int)
- Trans(Ln.BigNumber)
- Trans(double)

（3）对大数是否为 0、比较两个大数的大小和设置大数的符号的函数：

- Comparer(Ln.BigNumber, Ln.BigNumber)
- zero(Ln.BigNumber)
- set(Ln.BigNumber, bool)

（4）对大数求绝对值、e 的大数次幂、取反、开根号和平方的函数：

- abs(Ln.BigNumber)
- exp(Ln.BigNumber)
- invert(Ln.BigNumber)
- sqrt(Ln.BigNumber)
- square(Ln.BigNumber)

（5）输出函数

- output()

（6）对于+ −×÷的重载：

- operator -(Ln.BigNumber, Ln.BigNumber)
- operator -(Ln.BigNumber, int)
- operator -(Ln.BigNumber, double)
- operator -(double, Ln.BigNumber)
- operator -(int, Ln.BigNumber)
- operator *(Ln.BigNumber, int)
- operator *(Ln.BigNumber, Ln.BigNumber)
- operator *(int, Ln.BigNumber)
- operator *(Ln.BigNumber, double)
- operator *(double, Ln.BigNumber)
- operator /(Ln.BigNumber, int)
- operator /(Ln.BigNumber, Ln.BigNumber)
- operator /(int, Ln.BigNumber)
- operator /(Ln.BigNumber, double)
- operator /(double, Ln.BigNumber)
- operator +(Ln.BigNumber, Ln.BigNumber)
- operator +(Ln.BigNumber, int)
- operator +(int, Ln.BigNumber)
- operator +(Ln.BigNumber, double)
- operator +(double, Ln.BigNumber)
- operator ++(Ln.BigNumber)
- operator <(Ln.BigNumber, Ln.BigNumber)
- operator >(Ln.BigNumber, Ln.BigNumber)

如上图，为了方便后续计算，我对于大数与大数、int 型、double 型之间的四则运算都做了重载。

4 算法详述

4.1 lnx的数值解法

4.1.1 Taylor 展开法

4.1.1.1 换元 Taylor 展开法

根据 3.1.1 中的推导，令$x=\frac{1+y}{1-y}$，则

$$ln\,x=ln(1+y)-ln(1-y)$$

$$=\left(y-\frac{y^2}{2}+\frac{y^3}{3}-\cdots+(-1)^{n-1}\cdot\frac{y^n}{n}+\cdots\right)-\left(-y-\frac{y^2}{2}-\frac{y^3}{3}-\cdots(-1)^{n-1}\cdot\frac{y^n}{n}-\cdots\right)$$

$$=2\left(y+\frac{y^3}{3}+\frac{y^5}{5}+\cdots+\frac{y^{2n-1}}{2n-1}+\cdots\right)$$

当我们取级数的前 n 项近似lnx的时候，有

$$R_n(y)=2\left(\frac{y^{2n+1}}{2n+1}+\cdots\right)$$

$$|R_n(y)|\le\left|2\left(\frac{y^{2n+1}+y^{2n+3}+y^{2n+5}+\cdots}{2n+1}\right)\right|=\left|\frac{2y^{2n+1}}{(1-y^2)(2n+1)}\right|$$

根据精度的要求，我们可以计算出 n，从而求取符合精度要求的lnx。

在实际计算中，可以设计迭代算法计算其前 n 项的和，每次迭代完，都要判断余项$|R_n(y)|\le\left|\frac{2y^{2n+1}}{(1-y^2)(2n+1)}\right|$和要求的精度（$10^{-20}$）的大小关系，如果$|R_n(y)|\le 10^{-20}$，则结束迭代，此时的前 n 项和即为近似的$lnx$的值，否则继续迭代。

4.1.1.2 减半 Taylor 展开法

根据 3.1.1 中的推导，令$x=a\cdot 2^r$，则

$$lnx=ln(\,a\cdot 2^r)=ln\,a+ln2^r=lna+rln2$$

$$lna=(a-1)-\frac{(a-1)^2}{2}+\frac{(a-1)^3}{3}-\cdots+(-1)^{n-1}\cdot\frac{(a-1)^n}{n}+\cdots$$

我们取前n项来近似lna，于是

$$R_n(a-1)=(-1)^n\cdot\frac{(a-1)^{n+1}}{n+1}+\cdots$$

$$\therefore\quad |R_n(a-1)|\le\frac{(a-1)^{n+1}}{n+1}$$

在实际计算中，可以设计迭代算法计算前 n 项的和，每次迭代完判断余项$|R_n(a-1)|\le\frac{(a-1)^{n+1}}{n+1}$是否符合精度要求，如果达到了要求则结束迭代，求得的前项和加上$rln2$即为近似的lnx值，否则继续迭代。

4.1.2 数值积分

根据复化梯形公式，令$f(t)=\frac{1}{t}$,则

$$lnx=I=\int_1^x f(t)\,dt=\frac{h}{2}\cdot[f(1)+2\sum_{k=1}^{n-1}f(x_k)+f(x)=\frac{h}{2}\cdot[1+\frac{1}{x}+2\sum_{k=1}^{n-1}\frac{1}{x_k}]$$

$$|R(f)|=\left|-\frac{x-1}{12}\cdot \mathrm{h}^2\cdot f^{(2)}(\eta)\right|\leq\left|\frac{x-1}{12}\cdot \mathrm{h}^2\cdot\max_{1\leq t\leq x}\frac{1}{t^3}\right|=\left|\frac{x-1}{12}\cdot \mathrm{h}^2\right|$$

其中，$x-1=n\cdot h$，即把$x-1$均分为n段，$x_k=1+k\cdot h$，$\eta\in(1,x)$

因此在实际计算中我们可以先根据$|R(f)|\leq\left|\frac{x-1}{12}\cdot \mathrm{h}^2\right|$计算出符合要求的$h$的值和$n$的值，从而合适地划分区间，并由上式计算出近似的$lnx$的值。

4.1.3 方程求解

4.1.3.1 四阶龙格—库塔公式

$$y'=\frac{1}{x},f(x,y)=\frac{1}{x}$$

由四阶龙格—库塔公式有:

$$K_1=f(x_n,y_n)=\frac{1}{x_n}$$

$$K_2=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1\right)=\frac{1}{x_n+\frac{h}{2}}$$

$$K_3=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_2\right)=\frac{1}{x_n+\frac{h}{2}}$$

$$K_4=f(x_{n+1},y_n+hK_3)=\frac{1}{x_n+h}$$

$$y_{n+1}=y_n+\frac{h}{6}(K_1+2K_2+2K_3+K_4)=y_n+\frac{h}{6}\cdot\left(\frac{1}{x_n}+\frac{2}{x_n+\frac{h}{2}}+\frac{2}{x_n+\frac{h}{2}}+\frac{1}{x_n+h}\right)$$

$$=y_n+\frac{h}{6}\cdot\left(\frac{1}{x_n}+\frac{4}{x_n+\frac{h}{2}}+\frac{1}{x_n+h}\right)$$

$y(x_{n+1})-y_{n+1}=o(h^5)$，累积误差为$o(h^4)$.由$h^4\leq 10^{-20}$可以计算出$h\leq 1\times 10^{-5}$。

在实际计算中，由于$n=\frac{x-1}{h}$，所以当x很大时，n就会非常大，故令$x=a\cdot 2^r$，则

$$lnx=ln(a\cdot 2^r)=ln\,a+ln2^r=lna+rln2$$

故仅需对$ln\,a$利用上述四阶龙格—库塔算法计算数值积分即可。

为了留有裕量，取$h=5\times 10^{-6}$，$x_0=1$，$x_{n+1}=x_n+h$，迭代计算直至x_{n+1}和x_n足够接近。

4.1.3.2 牛顿迭代公式

令$f(t)=e^t-x$，则求取lnx等价于求$f(t)=0$的根，其中$x\in(1,\infty).t\in(0,\infty)$.

$f'(t)=e^t>0$恒成立，故如果方程有解则一定是唯一的解。

$f(t)\in C^{(2)}[0,\infty]$满足：

i. $f(0)\,f(\infty)<0$;

ii. $f''(t)=e^t$在$[0,\infty]$不变号；

iii. $\forall t\in[0,\infty],f'(t)=e^t\neq 0$;

iv. $\frac{|f(0)|}{\infty-1}\to 0<f'(0)=1,\frac{|f(\infty)|}{\infty-1}=\frac{e^{\infty}-x}{\infty-1}<f'(\infty)=e^{\infty}$.

故$\forall t\in[0,\infty]$，牛顿迭代法收敛。

令$\varphi(t)=t-\frac{f(t)}{f'(t)}$，则

$$t_{n+1}=t_n-\frac{f(t_n)}{f'(t_n)}=t_n-\frac{e^{t_n}-x}{e^{t_n}}=t_n-1+\frac{x}{e^{t_n}}$$

选取初始值$t_n=0$.

每次迭代之后，比较$|t_{n+1}-t_n|\le 10^{-20}$是否成立，如果成立就停止迭代。

4.2 数据结构

4.2.1 高精度数的构造

对于本次大作业中所使用的高精度数，如 3.2 中所述，使用 60 位 int 型数组存储，其中前 20 位为整数部分，后 40 位为小数部分，输出时小数部分只输出 20 位。另外，我还用了一个 int 型的数字 positive 来表示高精度数的符号，positive 为 1 时表示正数。

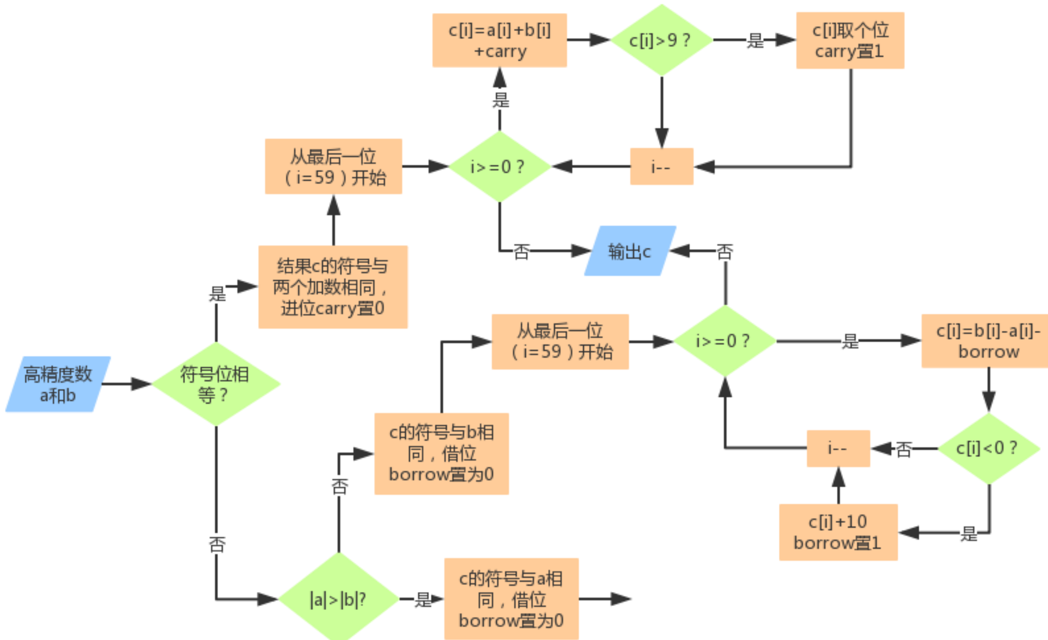
除了构造函数以外，为了防止高精度数在赋值的时候出错，我还写了一个复制函数；为了后续使用方便，高精度数与 string 类型的相互转化以及 double 型和 int 型转化为高精度数也分别有对应的函数。

4.2.2 运算符的重载

由于高精度数是自己写的一个类，故 C#的$+-\times\div$不适用于高精度数，因此在本次大作业中完成了对$+-\times\div$的重载，这也是高精度数类算法的核心。

4.2.2.1 “+”的重载

对于高精度数的加法，我的思路用流程图表示如下：



其中，|a| > |b|在图中没有完整画出来，事实上它与|a|＜|b|不同之处仅仅在于减数和被减数的位置互换了。

4.2.2.2 “−”的重载

由于a − b = a + (−b)，故对于高精度数的减法，仅需要把减数取反，再与被减数相加即可。为了使用方便，这里我增加了一个取反的函数：

```
/// <summary>
/// 取反函数
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static BigNumber invert(BigNumber a)...
```

取反之后存储数值的数组不变，仅符号位取反。

4.2.2.3 “×”的重载

对于高精度数乘法的实现，模拟列竖式的过程，用一个乘数的各位与另一个乘数逐位相乘。于是两个高精度数相乘的过程可以拆成一个高精度数与若干个 int 型整数相乘的过程，最后再求和。结果的符号位 positive 即为两个乘数的 positive 之积。

需要注意的有两点：

（1） 两个乘数的末位需要“对齐”；

（2） 乘数 a 小数点后第 i 位与乘数 b 小数点后第 j 位相乘后，其结果对应乘积小数点后第（i+j） 位。

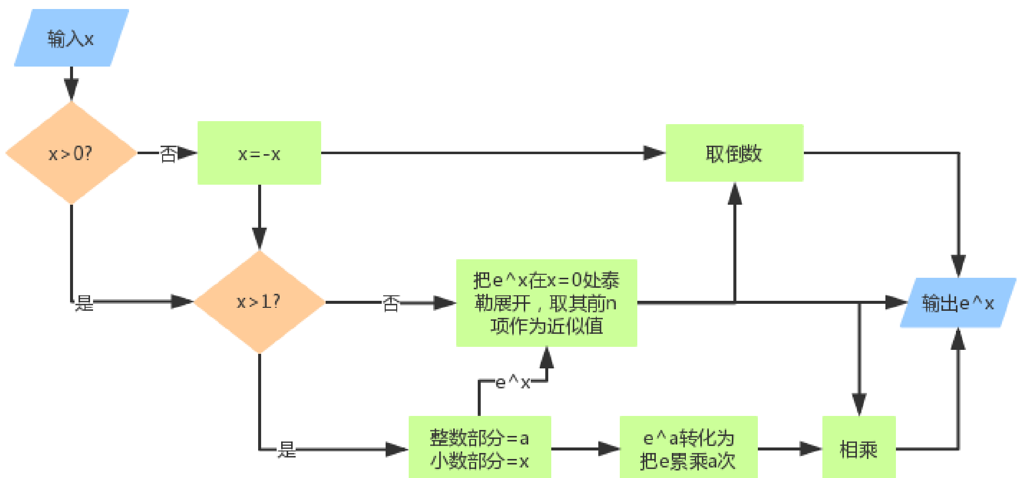
基于上述原则，把高精度数 a 分为小数部分和整数部分，逐位与高精度数 b 相乘即可，在处理乘数和乘积的数字位置时需要格外注意。

4.2.2.4 “÷”的重载

高精度数的除法的实现过程可以类比乘法，即把两个高精度数的除法转换为一个高精度数除以若干 int 型的数字。模拟长除法的过程，从高位逐位向下计算，将除法转化为减法，即将除数一次一次从被除数中减去，每减去一次，商加 1，被除数如果小于除数，则将被除数扩大10^n此后相应的每减去除数一次，商变为加10^{-n}。

4.2.3 e 的高精度数幂的求取

e^x（其中x是高精度数）的求取用流程图表示如下：



4.2.3 输入输出

输入时 textbox 里得到的是 string 类型的数据，这里用之前写的 StringToBn 函数转化为高精度数；输出时主要是对小数点和无效的 0 的消除的处理，这里其实我也处理了负数的输出，但是由于这一次输入的数字在（1,100），不会出现输出为负数的情况，所以并没有用上。

具体步骤如下：

（1）判断 positive，如果为-1，输出一个“—”；

（2）对高精度数的整数部分（第 0~19 位）逐位遍历，从第一个不是零的数字开始输出，如果每一位都是 0 则整数部分就是 0；

（3）输出一个“.”（小数点）；

（4）输出高精度数的第 20~39 位。

5 程序框图

需要用流程图描述的算法 前面已经给出，下面给出整个程序的框图：

6 结果及误差分析

6.1 结果展示

考虑到四阶龙格—库塔公式和复化梯形公式在输入的数字与 1 的差比较大时由于迭代的步数过多，所以测试的时候在输入比较大的数字的时候仅用其他三种方法测试。

(1)$x = 25.253$

(2) $x = 99.999$

(3) $x = 12$

(4) $x = 1.0001$

6.2 误差分析

6.1 换元 Taylor 法

6.1.1 方法误差

$$x=\frac{1+y}{1-y},\quad \therefore y=\frac{x-1}{x+1}$$

Taylor 展开公式为:

$$lnx=2\left(y+\frac{y^3}{3}+\frac{y^5}{5}+\cdots+\frac{y^{2n-1}}{2n-1}+\cdots\right)$$

方法误差是由 Taylor 余项造成的，故

$$\Delta(lnx)=|R_n(y)|\le\left|2\left(\frac{y^{2n+1}+y^{2n+3}+y^{2n+5}+\cdots}{2n+1}\right)\right|=\left|\frac{2y^{2n+1}}{(1-y^2)(2n+1)}\right|$$

6.2.2 舍入误差

每次计算时，如果结果中小数点位数超过存储位数（40 位），程序将直接把小数点后 40 位以后的数据舍去，故截断存储产生的误差为10^{-40}。加法和减法运算时，结果的小数位数不会超过加数或者减数的小数位数，所以不会产生舍入误差；但是进行乘法和除法运算时，可能发生结果的小数位数超过乘数或者除数的情况，因此会产生10^{-40}的舍入误差。

在换元 Taylor 法中，假设x没有误差，$y=\frac{x-1}{x+1}$，故$\delta(y)=10^{-40}$。

$y_{i+1}=\frac{2y^{2i+1}}{2i+1}$，$y_i=\frac{2y^{2i-1}}{2i+1}$，由$y_{i+1}$和$y_i$之间的关系:

$$y_{i+1}=\frac{2y^{2i-1}\times y\times y}{2i+1}=\frac{2y_i\times y\times y}{2i+1}$$

我们可以推出$\delta(y_{i+1})$和$\delta(y_i)$之间的关系如下:

$$\delta(y_{i+1})\le\frac{2\delta(y^{2i-1}\times y\times y)}{2i+1}+10^{-40}$$

$$\le\frac{2(\delta(y^{2i-1}\times y)\times|y|+|y^{2i}|\times\delta(y)+10^{-40})}{2i+1}+10^{-40}$$

$$\le\frac{2((\delta(y^{2i-1})\times|y|+|y^{2i-1}|\times\delta(y)+10^{-40})\times|y|+|y^{2i}|\times\delta(y)+10^{-40})}{2i+1}+10^{-40}$$

$$=\frac{2\left((\delta(y_i)\times|y|+|y^{2i-1}|\times\delta(y)+10^{-40})\times|y|+|y^{2i}|\times\delta(y)+10^{-40}\right)}{2i+1}+10^{-40}$$

$$\le\delta(y_i)+4\times10^{-40}$$

$\delta(y_i)\le(4i-3)\times10^{-40}$，故$\delta u_i\le\frac{8i-6}{2i-1}10^{-40}+10^{-40}\le5\times10^{-40}$。

总的舍入误差：$\delta(\ln x)\le\sum_{i=1}^{n}|\delta(y_i)|$

6.2.3 总误差

总误差是方法误差和舍入误差的和:

$$\Delta(lnx)=|R_n(y)|+\sum_{i=1}^{n}|\delta(y_i)|\le(5n+1)\times10^{-40}$$

其中，n是 Taylor 展开的级数。

6.2 减半 Taylor 法

6.2.1 截断误差

根据 4.1.1.2 中的推导， $lnx=lna+rln2$

$$lna=(a-1)-\frac{(a-1)^2}{2}+\frac{(a-1)^3}{3}-\cdots+(-1)^{n-1}\cdot\frac{(a-1)^n}{n}+\cdots$$

我们取前n项来近似lna，于是

$$R_n(a-1)=(-1)^n\cdot\frac{(a-1)^{n+1}}{n+1}+\cdots$$

$$\therefore\quad|R_n(a-1)|\le\frac{(a-1)^{n+1}}{n+1}$$

方法误差$\Delta(lnx)=\Delta(lna)=|R_n(a-1)|\le\frac{(a-1)^{n+1}}{n+1}$

6.2.2 舍入误差

$a=\frac{x}{2^r}$，假设$\delta(x)=0$，在进行到第 i 步时，累计舍入误差为$\delta(\frac{x}{2^i})$，则有

$$\delta(\frac{x}{2^{i+1}})=\frac{1}{2}\delta(\frac{x}{2^i})+10^{-40}$$

故$\delta(a)\le\frac{(2^r-1)10^{-40}}{2^{r-1}}$，由于减法不会产生舍入误差，故

$$\delta(a-1)=\delta(a)\le\frac{(2^r-1)10^{-40}}{2^{r-1}}<2\times10^{-40}$$

而在求lna的近似解时，令$t=a-1$，$u_{i+1}=(-1)^i\frac{t^{i+1}}{i+1}=(-1)^{i-1}\frac{t^i\times(-t)}{i+1}$，$\delta_i=(-1)^{i-1}t^i$，

则$\delta_1 = \delta(a-1)$。

$$\delta_{i+1} = \delta\left((-1)^{i-1} t^i (-t)\right) \le |t| \cdot \delta_i + \left|t^i\right| \cdot \delta_1 + 10^{-40} < \delta_i + 3 \cdot 10^{-40}$$

$$\delta(u_{i+1}) = \delta\left((-1)^i \frac{t^{i+1}}{i+1}\right) \le \frac{\delta_{i+1}}{i+1} + 10^{-40} = \frac{3i+2}{i+1} 10^{-40} + 10^{-40} < 4 \times 10^{-40}$$

总的舍入误差为$\delta(lnx) \le \delta\left(ln(1+t)\right) + r \times 10^{-40} < (4n-2+r) \times 10^{-40}$。

6.2.3 总误差

总误差是方法误差和舍入误差的和:

$$\Delta(lnx) = |R_n(a-1)| + (4n-2+r) \times 10^{-40} < (4n-1+r) \times 10^{-40}$$

其中，n是 Taylor 展开的级数。

6.3 数值积分—复化梯形公式

6.3.1 方法误差

根据 4.1.2 中的分析，有

$$|R(f)| = \left|-\frac{x-1}{12} \cdot h^2 \cdot f^{(2)}(\eta)\right| \le \left|\frac{x-1}{12} \cdot (\frac{x-1}{n})^2 \cdot \max_{1 \le t \le x} \frac{1}{t^3}\right| = \left|\frac{(x-1)^3}{12n^2}\right|$$

程序中我们令$|R(f)| < 10^{-40}$.

6.3.2 舍入误差

$$lnx = I = \int_1^x f(t)\,dt = \frac{h}{2} \cdot [f(1) + 2\sum_{k=1}^{n-1} f(x_k) + f(x) = \frac{h}{2} \cdot [1 + \frac{1}{x} + \sum_{k=1}^{n-1} \frac{2}{x_k}]$$

$$\begin{aligned}\delta(lnx) &= \delta\left(\frac{h}{2} \cdot [1 + \frac{1}{x} + \sum_{k=1}^{n-1} \frac{2}{x_k}]\right) \le \delta\left(\frac{h}{2}\right)\left|1 + \frac{1}{x} + \sum_{k=1}^{n-1} \frac{2}{x_k}\right| + \frac{h}{2}\delta\left(1 + \frac{1}{x} + \sum_{k=1}^{n-1} \frac{2}{x_k}\right) \\ &< \delta\left(\frac{x-1}{2n}\right) \cdot \left(2(n-1)\right) + \frac{x-1}{2n} \cdot n \cdot 10^{-40} \\ &= (2n-2) \cdot 10^{-40} + \frac{x-1}{2} \cdot 10^{-40} = \left(2n + \frac{x-5}{2}\right) \cdot 10^{-40}\end{aligned}$$

6.3.3 总误差

总误差是方法误差和舍入误差的和:

$$\Delta(lnx) = |R(f)| + \delta(lnx) < 10^{-40} + \left(2n + \frac{x-5}{2}\right) \cdot 10^{-40} = \left(2n + \frac{x-3}{2}\right) \cdot 10^{-40}$$

其中，n是根据误差限选择的点数。

6.4 四阶龙格—库塔公式

6.4.1 方法误差

$$y_{n+1} = y_n + \frac{h}{6}\cdot\left(\frac{1}{x_n} + \frac{4}{x_n + \frac{h}{2}} + \frac{1}{x_n + h}\right)$$

$$x_0 = 1，y_0 = 0，y_0 = 0.000005$$

4.3.1 中已经证明此四阶 R-K 方法具有四阶精度，即方法误差为$o(h^4)$.

6.4.2 舍入误差

在求lna中，由迭代公式可知，$\delta(y_{n+1}) \le \delta(y_n) + \frac{h}{6}\cdot 3\cdot 10^{-40} + 10^{-40}$

因为h很小，$\delta(y_0) = 0$，于是有

$$\delta(y_n) \le n10^{-40}$$

另外，$ln2$在存储时也有误差，$\delta(ln2) \le 10^{-40}$。

6.4.3 总误差

总误差为方法误差和舍入误差的和：

$$\delta(lnx) \le o(h^4) + \left(\frac{|a-1|}{h} + r\right)10^{-40} = o(10^{-22}) + 2|a-1|10^{-35} + r10^{-40}.$$

6.5 牛顿迭代法

$$\varphi(t) = t - \frac{f(t)}{f'(t)} = t - \frac{e^t - x}{e^t} = t - 1 + xe^{-t}, \mathrm{t} \in (0, \infty)$$

$$\varphi''^{(t)} = xe^{-t}, \therefore |\varphi''(t)|_{max} = x$$

则迭代到 n 步的方法误差为

$$e_n = \frac{2}{M}\left(\frac{M}{2}|e_0|\right)^{2^n} = \frac{2}{x}\left(\frac{x}{2}|e_0|\right)^{2^n}$$

在本次大作业中，应用了事后估计法，即每进行一次迭代，判断$|t_{n+1} - t_n|$是否达到精度。采用牛顿迭代法，需要迭代的步数很小。

7 心得体会

在这次的大作业中我总共尝试了 5 种不同的方法去解决 lnx 近似值的求解问题。其中换元 Taylor 法、减半 Taylor 法和牛顿迭代法是比较实用的方法，收敛速度快并且精度很高，能够比较切合地达到要求；而四阶龙格—库塔法和梯形法如果想要达到想要的精度，可能需要几十万甚至上百万次的迭代，速度非常慢，很不实用。

在这次完成大作业的过程中，为了达到要求的精度，我做了很多的误差分析，应用了课上学到的误差分析的方法，在实际编写程序的时候用得比较多实际上是事后估计法，总体来说精度控制得比较好。

这次作业的难点除了 lnx 的不同求解方法以外，还有高精度数的设计。一开始高精度数的符号位我是用 bool 型变量，后来在重载＋－×÷的过程中我发现这样不方便运算，所以改用 int 型变量，可以直接通过×÷来计算。另外，在重载×和÷的过程中，如何把“列竖

式”的过程转化为代码，如何让计算出来的数字在正确的位置，进位和借位怎么处理，这些都是经过了漫长的调试的。

经过这次大作业，我巩固、深化了课堂上学到的知识，也强化了实际编程的能力，收获非常大。