《数值分析与算法》

大作业２　求取 ln(x)

项目报告

班级：自 45

姓名：林子坤

学号：2014011541

目录

大作业 2　　求取 ln(x)

项目报告

林子坤

（自动化系　自 45 班　2014011541）

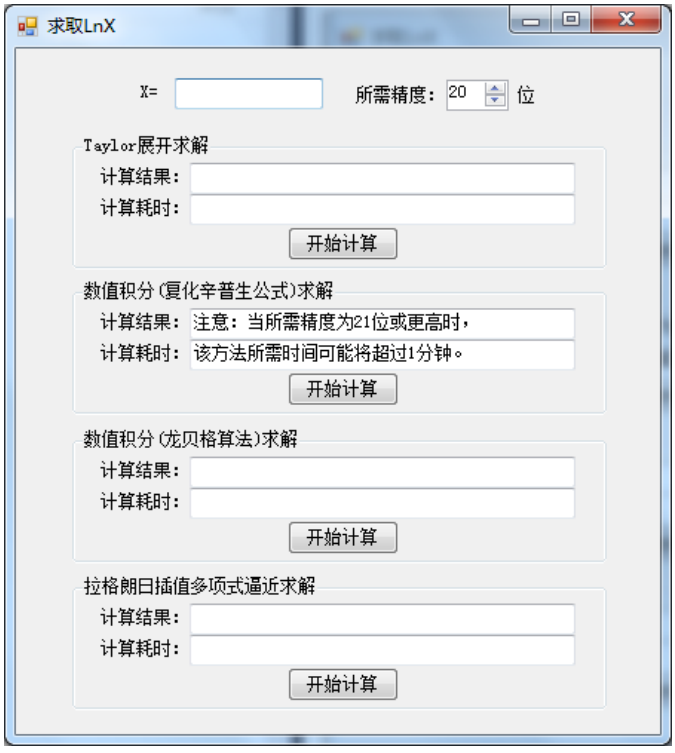
一、项目简介

本次大作业通过使用不同的函数逼近与数值积分算法，对[1,100]范围内的 ln(*x*)的高精度值进行求取，并可根据使用者需求，在前三种方法中给出小数点后 0~32 位精度的结果并确保这 32 位的正确性，在第四种方法中给出小数点后 0~32 位精度的结果并至少能够确保前 24 位的正确性。

二、程序属性

本程序使用 C#语言编写，使用 Visual Studio 2013 Community 编译器。支持的运行环境是 Windows 7 至 Windows 10 各版本。

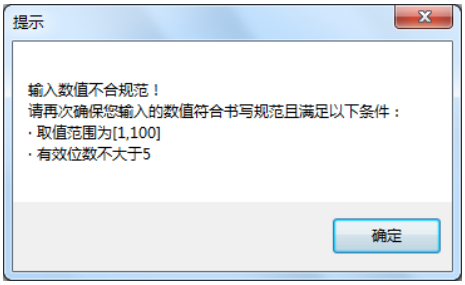
程序界面如下图所示：



三、程序功能与使用方法

3.1 输入自变量

在“X=”后的文本框中输入待求取对数的自变量 X，且确保这个数值符合规范、输入最多 5 位有效数字，且在[1,100]范围内。若输入数值不符合以上要求，在点击“开始计算”后将弹出如下提示窗。



3.2 选择所需精度

在“所需精度”后的数值框中调整所需要的精度，即小数点后保留位数（默认为 20 位精度）。该精度可在 0~32 范围内变化。由于该精度数值不仅将影响到算法迭代次数，也将影响到运行过程中除法有效位数的保留问题，因此随着精度增加，运算时间的增加速度也是较为可观的。经过本人测试，使用“Taylor 展开求解”、“数值积分（龙贝格算法）求解”与“拉格朗日插值多项式求解”这三种方式均基本能够在任何精度要求下保证在 5 秒钟内显示所需结果。

3.3 开始计算

点击各方法框内的“开始计算”按钮，程序将使用该方法对 ln(x)的值进行计算，并给出符合精确度要求的计算结果与运行时间。

3.4 特别注意

值得注意的是，与另外三种方法的高性能不同，随着精度增加，**复化辛普生公式**求解的速度将成倍增加。当所需精度要求为 20 位时，所需要的时间为 1 秒至 50 秒不等；而当精度要求为 21 位或更高时，除了距离 1.5 的 n 次幂较近的数，该方法所需时间可能将达到难以忍受的地步（超过 1 分钟）；当精度为 32 位左右时，除了距离 1.5 的 n 次幂较近的数，该方法所需时间可能将超过 1 小时（出于时间因素考虑并未做完整个实验过程）。因此请助教老师**在测试复化辛普生公式时选取 20 位或 20 位以内的合适的精度，以免花费老师过多时间**。

四、方法原理与性能分析

本部分将从**原理**的角度，对所使用的四种方法进行阐述，并分析各种方法的**计算代价**、

收敛速度与误差。该部分中牵涉到的加减乘除计算量分析基于本人编写的大数类算法，其实现过程将在第五部分说明。

4.1 Taylor 展开

4.1.1 方法原理

在数学中，泰勒公式是将一个在$x = x_0$处具有 n 阶导数的函数$f(x)$利用关于$(x - x_0)$的 n 次多项式来逼近函数的方法。如果函数足够平滑的话，在已知函数在某一点的各阶导数值的情况之下，泰勒公式可以用这些导数值做系数构建一个多项式来近似函数在这一点的邻域中的值。泰勒公式还给出了这个多项式和实际的函数值之间的偏差。

若函数$f(x)$在包含x_0的某个闭区间[a,b]上具有 n 阶导数，且在开区间(a,b)上具有(n+1)阶导数，则对闭区间[a,b]上任意一点x，成立下式：

$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n(x)$$

将$f(x) = \ln(x)$带入上式，在$x_0 = 1$处展开得：

$$\ln(x) = \frac{x - 1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \cdots + (-1)^{n+1}\frac{(x-1)^n}{n}$$

$$= \sum_{i=1}^{n}(-1)^{i+1}\frac{(x-1)^i}{i}$$

注意到在$x \le 2$时$(x-1)^i$将逐渐收敛到 0；而在$x > 2$时，$(x-1)^i$将快速增大，使得上述方法的收敛效率大大降低。因此在使用 Taylor 展开时，需要对原始数据进行一定处理，使得 Taylor 展开过程中使用的数值在 1 和 2 之间。本人采用的方法是将x按照以下方式分解：

$$x = t \times 1.5^{exp}\ (1 < t < 1.5)$$

从而$\ln(x)$可以分解为以下形式：

$$\ln(x) = \ln(t) + exp\ln(1.5)$$

程序预先置入$\ln(1.5)$的准确数值，从而仅需求取$\ln(t)$的数值。由于$1 < t < 1.5$，则$(t-1)^i$将逐渐收敛到 0，从而减小了截止阶数。

在进行截止阶数的判断时，假设要求的精度(precision)为 p，为了保证 p 位的精度，要求前(p+1)位的数值是完全正确的（考虑到四舍五入），因此误差项应小于$10^{-(p+1)}$，即需要计算到满足如下条件，其中 n 为截止阶数：

$$\frac{(t-1)^n}{n} < 10^{-(p+1)}$$

根据对以上不等式的分析可以看出，当$t \to 1.5$时，n值将达到最大。因此令$t = 0.5$，计算各精度下的迭代次数最大值——取$p = 20$时，解如上不等式可得$n \ge 73$时能保证 20 位的精度；取$p = 32$时，解如上不等式可得$n \ge 104$时能保证 32 位的精度……因此可以看出，

Taylor 展开能够以较少的计算次数获得所要求的精度，是一种较为迅速的方法。

4.1.2 计算代价

4.1.2.1 第 1 步：$\ln(x)$分解

该步计算主要由以下部分组成：

• 乘法 1 次：该步乘法需要计算 1.5 的第exp次幂的精确数值。对于次数exp，所需要的计算次数为2^{exp}次（大数类乘法算法将在报告第六部分说明）。

• 减法exp次：该步减法需要比较输入数值与 1.5 的第exp次幂的精确数值的大小。对于次数exp，所需要的计算次数为$2exp$次。

综上：该步骤的计算量为$(2^{exp}+2exp^2)$，将$exp=\lfloor\log_{1.5}x\rfloor$代入，得到计算量为：

$$2^{\lfloor\log_{1.5}x\rfloor}+2(\lfloor\log_{1.5}x\rfloor)^2$$

4.1.2.2 第 2 步：求取各阶 Taylor 展开式

以下针对 Taylor 展开式的第 i 项$(1\le i\le \mathrm{n})$分析其计算代价：

• 乘法：由于题目要求的输入有效位数不多于 5，则$(t-1)$的有效位数不多于 4，则$(t-1)^i$的有效位数不多于4^i，则第 i 次运算牵涉的乘法将是一个有效位数不多于4^{i-1}的数与一个有效位数不多于 4 的数的乘法，所需要的运算次数为4^i次。

• 除法：大数类除法的运算次数仅与数字本身和精度要求相关，其最高运算次数为$9(p+5)$次。

综上：该步骤的最高计算量为：

$$\sum_{\mathrm{i}=1}^{n}[4^i+9(p+5)]=\frac{4}{3}(4^n-1)+9n(p+5)$$

4.1.2.3 第 3 步：结果整合

该步计算主要由以下部分组成：

• 各项加法：每一次计算完毕后需要进行一次加法，将最新计算出的项与之前的结果相加。由于题目要求的输入有效位数不多于 5，则$(t-1)$的有效位数不多于 4，则$(t-1)^i$的有效位数不多于4^i，则第 i 次运算牵涉的加法需要的运算次数为4^i次。

• 结果加法：在上述计算完成后需要将结果与$exp\ln(1.5)$相加，需要的加法次数为exp次，每次加法需要的运算次数为4^n次。

综上，该步骤的计算量为：

$$\sum_{\mathrm{i}=1}^{n}4^i+exp\times4^n=\left(\frac{4}{3}+exp\right)\times4^n-\frac{4}{3}=\left(\frac{4}{3}+\lfloor\log_{1.5}x\rfloor\right)\times4^n-\frac{4}{3}$$

综上所述：使用 Taylor 展开所需要的最高计算代价约为：

$$2^{\lfloor \log_{1.5} x \rfloor} + 2(\lfloor \log_{1.5} x \rfloor)^2 + 9n(p+5) + \left(\frac{8}{3} + \lfloor \log_{1.5} x \rfloor\right) \times 4^n - \frac{8}{3} = O(4^n)$$

其中，x为输入数值；p为所要求的精度；n为所需展开次数，与x, p有关。

4.1.3 收敛速度

在 4.1.1 节已进行阐述，展开次数与t, p应满足以下关系：

$$\frac{(t-1)^n}{n} < 10^{-(p+1)}$$

由于$t = \frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}$，代入得：

$$\frac{\left(\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}} - 1\right)^n}{n} < 10^{-(p+1)}$$

在 32 位精度的范围内，Taylor 展开均能在 100 次展开之内实现所需精度。

4.1.4 误差分析

4.1.4.1 方法误差

由上文分析可得，求取$\ln(t)$的方法误差为：

$$\delta_{\mathrm{t}} = \frac{(t-1)^n}{n}$$

根据$\ln(x) = \ln(t) + exp\ln(1.5)$，由于$\ln(1.5)$为预先存储的精确值，故

$$\delta_x = \delta_{\mathrm{t}} = \frac{(t-1)^n}{n}$$

4.1.4.2 舍入误差

由于本人编写的大数类能够保证加减乘不舍弃任何小数位，且除法保证了一定的误差裕量，故大数类运算误差忽略不计。

设精度为 p，各阶舍入误差为：

$$\sigma = \frac{1}{2} \times 10^{-(p+1)}$$

因此，累积到第 n 阶的舍入误差为：

$$\sigma = \frac{n}{2} \times 10^{-(p+1)}$$

由于$\ln(1.5)$为预先存储的精确到小数点后 32 位的精确值，因此舍入误差为$\frac{1}{2} \times 10^{-32}$

综上：综合方法误差与舍入误差，总误差为：

$$\frac{(t-1)^n}{n} + \frac{n}{2} \times 10^{-(p+1)} + \frac{1}{2} \times 10^{-32}$$

由于$t=\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}$，代入得总误差为：

$$\frac{\left(\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}-1\right)^n}{n}+\frac{n}{2}\times 10^{-(p+1)}+\frac{1}{2}\times 10^{-32}$$

其中，x为输入数值；p为所要求的精度；n为所需展开次数，与x,p有关。

4.2 复化辛普生公式

4.2.1 方法原理

辛普生公式为：

$$I \doteq S=\frac{(b-a)}{6}\left[f(a)+4f(\frac{a+b}{2})+f(b)\right]$$

在区间[*a*,*b*]上进行 *n* 等分，得到 *n* 个子区间，$[x_k,x_{k+1}]$，*k*=0,1,2,⋯,*n*-1，在$[x_k,x_{k+1}]$上使用辛甫生公式，得到复化的辛普生公式为：

$$I=\int_0^x f(x)dx=\sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)dx=\sum_{k=0}^{n-1}\frac{h}{6}\left[f(x_k)+4f(x_k+\frac{h}{2})+f(x_{k+1})\right]$$

$$I=\frac{h}{6}\left[f(0)+2\sum_{k=1}^{n-1}f(x_k)+4\sum_{k=0}^{n-1}f(x_k+\frac{h}{2})+f(x)\right]$$

复化辛普生公式的 *n* 阶误差$R_{\mathrm{n}}(f)$为：

$$R_n(f)=-\frac{b-a}{180}\left(\frac{h}{2}\right)^4 f^{(4)}(\eta)\ (a<\eta<b)$$

为了使每次运算的误差尽可能小，从而减少计算次数，将$\ln(x)$分解为以下形式：

$$\ln(x)=\ln(t)+exp\ln(1.5)$$

程序预先置入$\ln(1.5)$的准确数值，从而仅需求取$\ln(t)$的数值。

在进行截止阶数的判断时，假设要求的精度(precision)为 p，为了保证 p 位的精度，要求前(p+1)位的数值是完全正确的（考虑到四舍五入），因此误差项应小于$10^{-(p+1)}$，即需要计算到满足如下条件，其中 *n* 为截止阶数：

$$\frac{b-a}{180}\left(\frac{h}{2}\right)^4 f^{(4)}(\eta)<10^{-(p+1)}$$

将$f^{(4)}(\eta)\le 6$，$h=\frac{b-a}{n}$代入上式可得：

$$n^4\ge\frac{(b-a)^5}{480}\times 10^{(p+1)}$$

根据对以上不等式的分析可以看出，当$t \to 1.5$时，n值将达到最大。因此令$t = 0.5$，计算各精度下的迭代次数最大值——取$p = 10$时，解如上不等式可得$n \geq 51$时能保证 10 位的精度；取$p = 20$时，解如上不等式可得$n \geq 15974$时才能保证 20 位的精度；取$p = 32$时，解如上不等式可得$n \geq 15973577$时才能保证 32 位的精度……因此可以看出，复化辛普生公式在精度要求较高时，所需要的等分份数与运算次数高得令人发指。

4.2.2 计算代价

4.2.2.1 第 1 步：$\ln(x)$分解

该步算法与 Taylor 展开中的相同，所需计算量为：

$$2^{\lfloor \log_{1.5} x \rfloor} + 2(\lfloor \log_{1.5} x \rfloor)^2$$

4.2.2.2 第 2 步：数值积分求$\ln(t)$

以下针对第 i 个等分节点$(1 \leq i \leq \mathrm{n})$分析其计算代价：

- 计算x_k与$x_{k+\frac{1}{2}}$的数值：对于每个数值，需要分别进行一次计算量最高为$9(p+5)$次的除法运算。由于除法运算的结果位数为$(p+5)$，因此还需要进行计算量最高为$2(p+5)$的乘法运算。此后，将结果与之前的x_k相加，所需运算量为$(p+5)$。因此，该步总计算量为$24(p+5)$。
- 计算$\frac{1}{x_k}$与$\frac{1}{x_{k+\frac{1}{2}}}$的数值：对于每个数值，需要分别进行一次计算量最高为$9(p+5)$次的除法运算，该步总计算量为$18(p+5)$。
- 计算结果整合：将以上数值乘以系数并进行加和，需要的计算量为$6(p+5)$。

综上：该步骤的最高计算量为$48n(p+5)$。

4.2.2.3 第 3 步：结果整合

该步算法与 Taylor 展开中的相同，所需计算量为：

$$exp \times (p+5) = \lfloor \log_{1.5} x \rfloor (p+5)$$

综上所述：使用复化辛普生公式所需要的最高计算代价为：

$$2^{\lfloor \log_{1.5} x \rfloor} + 2(\lfloor \log_{1.5} x \rfloor)^2 + 48n(p+5) + exp \times 4^{(p+5)}$$

将$n = \left\lceil \sqrt[4]{\frac{(t-1)^5}{480} \times 10^{(p+1)}} \right\rceil, t = \frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}$代入得所需计算代价为：

$$2^{\lfloor \log_{1.5} x \rfloor} + 2(\lfloor \log_{1.5} x \rfloor)^2 + 48 \left\lceil \sqrt[4]{\frac{(\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}} - 1)^5}{480} \times 10^{(p+1)}} \right\rceil (p+5) + \lfloor \log_{1.5} x \rfloor (p+5)$$

其中，x为输入数值；p为所要求的精度，其中在运算时间中起主要决定作用的一项是：

$$48\left\lceil\sqrt[4]{\frac{\left(\frac{x}{1.5^{\lfloor\log_{1.5}x\rfloor}}\text{-}1\right)^5}{480}\times10^{(p+1)}}\right\rceil(p+5)$$

4.2.3 收敛速度

在 4.2.1 节已进行阐述，需要的等分份数为：

$$n=\left\lceil\sqrt[4]{\frac{(\mathrm{t}-1)^5}{480}\times10^{(p+1)}}\right\rceil=\left\lceil\sqrt[4]{\frac{(\frac{x}{1.5^{\lfloor\log_{1.5}x\rfloor}}-1)^5}{480}\times10^{(p+1)}}\right\rceil$$

在 10 位精度的范围内，复化辛普生公式能在 50 次等分计算之内实现所需精度；在 20 位精度的范围内，复化辛普生公式能在 15000 次等分计算之内实现所需精度；在 30 位精度的范围内，复化辛普生公式能在 5000000（五百万）次等分计算之内实现所需精度。

4.2.4 误差分析

4.2.4.1 方法误差

由上文分析可得，求取$\ln(t)$的方法误差为：

$$\delta_t=\frac{\mathrm{t}-1}{180}\left(\frac{t-1}{2n}\right)^4f^{(4)}(\eta)\le\frac{6(\mathrm{t}-1)^5}{2880n^4}$$

根据$\ln(x)=\ln(t)+exp\ln(1.5)$，由于$\ln(1.5)$为预先存储的精确值，故

$$\delta_x=\delta_{\mathrm{t}}\le\frac{6(\mathrm{t}-1)^5}{2880n^4}$$

4.2.4.2 舍入误差

由于本人编写的大数类能够保证加减乘不舍弃任何小数位，且除法保证了一定的误差裕量，故大数类运算误差忽略不计。

设精度为 p，各阶舍入误差为：

$$\sigma=\frac{1}{2}\times10^{-(p+1)}$$

因此，累积到第 n 阶的舍入误差为：

$$\sigma=\frac{n}{2}\times10^{-(p+1)}$$

由于$\ln(1.5)$为预先存储的精确到小数点后 32 位的精确值，因此舍入误差为$\frac{1}{2}\times10^{-32}$

综上：综合方法误差与舍入误差，总误差为：

$$\frac{6(\mathrm{t}-1)^5}{2880n^4}+\frac{n}{2}\times10^{-(p+1)}+\frac{1}{2}\times10^{-32}$$

其中，x为输入数值；p为所要求的精度；$t=\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}$；$n=\left\lceil \sqrt[4]{\frac{(\frac{x}{1.5^{\lfloor \log_{1.5} x \rfloor}}-1)^5}{480}\times 10^{(p+1)}} \right\rceil$。

4.3 龙贝格算法

4.3.1 方法原理

以$T_0^{(k)}$表示二分 k 次后求得的梯形值，以$T_m^{(k)}$表示序列$\{T_m^{(k)}\}$的 m 次加速值

可得：

$$T_m^{(k)}=\frac{4^m}{4^m-1}T_{m-1}^{(k+1)}-\frac{1}{4^m-1}T_{m-1}^{(k)} \text{ (记为\#式)}$$

其中二分 k 次后求得的梯形值为：

$$T_{2n}=\frac{1}{2}T_n+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}) \text{ (记为*式)}$$

计算过程如下：

（1）取$h=0, h=b-a$，求$T_0^{(0)}$：

$$T_0^{(0)}=\frac{h}{2}[f(a)+f(b)]$$

令$1\to k$（k 为区间$[a,b]$的二分次数）

（2）求梯形值$T_0(\frac{b-a}{2^k})$，即按照上文所述(*式)计算$T_0^{(\mathrm{k})}$

（3）求加速值，按照上文所述(#式)逐个求出 T 表的第 k 行其余各元素$T_{\mathrm{j}}^{(k-j)}(\mathrm{j}=1,2,\ldots,\mathrm{k})$

（4）若$\left|T_k^{(0)}-T_{k-1}^{(0)}\right|<\varepsilon$（精度要求），则终止计算，并令最终结果$I=T_k^{(0)}$，否则$k+1\to k$，转（2）继续计算。

k	h	$T_0^{(k)}$	$T_1^{(k)}$	$T_2^{(k)}$	$T_3^{(k)}$	$T_4^{(k)}$ …
0	$b-a$	$T_0^{(0)}$				
1	$\frac{b-a}{2}$	$T_0^{(1)}$ ↓①	$T_1^{(0)}$			
2	$\frac{b-a}{4}$	$T_0^{(2)}$ ↓②	$T_1^{(1)}$ ↓③	$T_2^{(0)}$		
3	$\frac{b-a}{8}$	$T_0^{(3)}$ ↓④	$T_1^{(2)}$ ↓⑤	$T_2^{(1)}$ ↓⑥	$T_3^{(0)}$	

为了使每次运算的误差尽可能小，从而减少计算次数，将$\ln(x)$分解为以下形式：

$$\ln(x) = \ln(t) + exp\ln(1.5)$$

程序预先置入$\ln(1.5)$的准确数值，从而仅需求取$\ln(t)$的数值。

在进行截止阶数的判断时，由于龙贝格算法的方法误差估计采用事后估计法，而在龙贝格算法的初期就需要建立对象数组并计算出一系列的$T_0^{(k)}$。为了合理确定建立的对象数组的大小与需要计算的$T_0^{(k)}$数量，在设定计算次数前本人进行了一系列实验，并根据实验经验获得如下经验值，实际计算数量根据下表确定：

精度要求	k_{max}
0	2
0~3	3
3~5	4
5~7	5
7~10	6
10~13	7
13~17	8
17~21	9
21~26	10
26~31	11
32	12

可以看出，即使在 32 位误差要求下也只需要计算 13 行共计 78 个数值，计算量并不算很大。

4.3.2 计算代价

4.3.2.1 第 1 步：$\ln(x)$分解

该步算法与 Taylor 展开中的相同，所需计算量为：

$$2^{\lfloor \log_{1.5} x \rfloor} + 2(\lfloor \log_{1.5} x \rfloor)^2$$

4.3.2.2 第 2 步：数值积分求$\ln(t)$

以下针对第 k 行$(1 \le k \le k_{max})$分析其计算代价，以下步骤需要重复2^{k-1}次：

- 求取x_j：乘法步骤为[2 位有效数字×最多 4 位有效数字]，所需运算次数为 8 次；除法步骤最高运算次数为$9(p+5)$次；加法步骤所需运算次数为$(p+5)$次。

- 求取$\frac{1}{x_j}$：除法步骤最高运算次数为$9(p+5)$次。

- 累加求取$T_i^{(j)}$：乘法步骤为[$(p+5)$位有效数字×最多 4 位有效数字]，所需运算次数为$4(p+5)$次；除法步骤最高运算次数为$9(p+5)$次；加法步骤所需运算次数为$(p+5)$次。

综上：该步骤的最高计算量为：

$$\sum_{k=1}^{k_{max}}\{2^{k-1}[8+33(p+5)]\}=(2^{k_{max}}-1)[8+33(p+5)]$$

4.3.2.3 第 3 步：结果整合

该步算法与 Taylor 展开中的相同，所需计算量为：

$$exp(p+5)=\lfloor\log_{1.5}x\rfloor(p+5)$$

综上所述：使用龙贝格算法所需要的最高计算代价为：

$$2^{\lfloor\log_{1.5}x\rfloor}+2(\lfloor\log_{1.5}x\rfloor)^2+(2^{k_{max}}-1)[8+33(p+5)]+\lfloor\log_{1.5}x\rfloor(p+5)=O(2^{k_{max}})$$

其中，x为输入数值；p为所要求的精度，k_{max}参照上文所列的精度表格进行对应。

4.3.3 收敛速度

在 4.2.1 节已进行阐述，T 表需要的行数上文所列的精度表格进行对应。

在 32 位精度的范围内，需要计算的数值不超过 13 行共计 78 个数。

4.3.4 误差分析

4.3.4.1 方法误差

由上文分析可得，求取$\ln(t)$的方法误差计算如下：

经过 k 次外推后，对[1,t]进行 l 等分，

$$\delta_t=O(h^{2l+2})$$

其中$l\mathrm{h}=t-1$

根据$\ln(x)=\ln(t)+exp\ln(1.5)$，由于$\ln(1.5)$为预先存储的精确值，故

$$\delta_x=\delta_{\mathrm{t}}=O(\frac{(t-1)^{2l+2}}{l})$$

4.3.4.2 舍入误差

由于本人编写的大数类能够保证加减乘不舍弃任何小数位，且除法保证了一定的误差裕量，故大数类运算误差忽略不计。

设精度为 p，每计算一次的舍入误差为：

$$\sigma_k=\frac{\mathrm{h}}{2}\left(\frac{1}{x_k^2}+\frac{1}{x_{k+1}^2}\right)\times\frac{1}{2}\times10^{-(p+1)}+\frac{1}{2}\times10^{-(p+1)}$$

因此，累积到第 n 阶的舍入误差为：

$$\sigma\le\sum_{k=0}^{n-1}\sigma_k$$

由递推公式可得：

$$\delta_m^{(k)} = \frac{4^m}{4^m-1}\delta_m^{(k+1)} - \frac{1}{4^m-1}\delta_{m-1}^{(k)} + \frac{1}{2}\times 10^{-m}$$

由于$\ln(1.5)$为预先存储的精确到小数点后 32 位的精确值，因此舍入误差为$\frac{1}{2}\times 10^{-32}$

因此，总的舍入误差为$\delta_{\mathrm{m}}^{(k)} + \frac{1}{2}\times 10^{-32}$

4.4 拉格朗日插值多项式逼近

4.4.1 方法原理

在平面上有$(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$共 n 个点，现作一条函数$f(x)$使其图像经过这 n 个点。

作法：作 n 个多项式$p_j(x), j=1,2,\cdots,n$。对于第 j 个多项式$p_j(x)$构造 n-1 次多项式

$$p_j(x) = \prod_{i\in I_j}\frac{x-x_i}{x_j-x_i}$$

满足$\forall i\in I_j, p_j(x_i)=0$ 并且$p_j(x_j)=1$ 。

最后可得

$$L_n(x) = \sum_{j=1}^{n} y_j p_j(x)$$

为了使每次运算的误差尽可能小，从而减少计算次数，将$\ln(x)$分解为以下形式：

$$\ln(x) = \ln(t) + exp\ln(1.5)$$

程序预先置入$\ln(1.5)$的准确数值，从而仅需求取$\ln(t)$的数值。在程序中取 n=40，可以保证至少 24 位，至多>32 位的数值精确性。

4.4.2 计算代价

4.4.2.1 第 1 步：$\ln(x)$分解

该步算法与 Taylor 展开中的相同，所需计算量为：

$$2^{\lfloor \log_{1.5} x\rfloor} + 2(\lfloor \log_{1.5} x\rfloor)^2$$

4.4.2.2 第 2 步：逼近求$\ln(t)$

该步算法需要进行 40 次$\frac{x-x_i}{x_j-x_i}$的求取，每次求取需要进行 40 次以下运算：

- 减法运算 2 次，每次所需计算量最多为 5
- 除法运算 1 次，最高运算次数为$9(p+5)$次。
- 乘法运算 1 次，第 j 次运算牵涉的乘法将是一个有效位数不多于$(p+5)^{j-1}$的数与一个有效位数不多于$(p+5)$的数的乘法，所需运算次数为$(p+5)^j$。

在 40 次求取最后，分别需要进行一次运算量为$(p+5)^{40}$的加法运算。

综上所述，该步骤最多所需总运算量为：

$$40\left\{40\left[10+9(p+5)+\sum_{j=1}^{40}(p+5)^j\right]+(p+5)^{40}\right\}$$

$$=16000+14400(p+5)+1600\frac{(p+5)[(p+5)^{40}-1]}{p+4}+40(p+5)^{40}$$

4.4.2.3 第 3 步：结果整合

该步算法与 Taylor 展开中的相同，所需计算量为：

$$exp\times 4^{(p+5)}=\lfloor\log_{1.5}x\rfloor\times 4^{(p+5)}$$

综上所述：使用拉格朗日插值多项式逼近所需要的最高计算代价为：

$$2^{\lfloor\log_{1.5}x\rfloor}+2(\lfloor\log_{1.5}x\rfloor)^2+16000+14400(p+5)+1600\frac{(p+5)[(p+5)^{40}-1]}{p+4}$$

$$+40(p+5)^{40}+\lfloor\log_{1.5}x\rfloor\times 4^{(p+5)}=O(p^{41})$$

其中，x为输入数值；p为所要求的精度。

4.4.3 收敛速度

由于本方法限定其使用的点数为 40，因此收敛速度是恒定的。

4.4.4 误差分析

4.4.4.1 方法误差

$\ln(t)$的插值计算余项为：

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)\le\frac{0.5^{n+1}}{(n+1)}$$

因为$max\{f^{(n+1)}(\xi)\}<n!$, $max\{\omega n+1(x)\}\le 0.5^{n+1}$

因此

$$\delta_{\mathrm{t}}\le\frac{0.5^{n+1}}{n+1}$$

根据$\ln(x)=\ln(t)+exp\ln(1.5)$，由于$\ln(1.5)$为预先存储的精确值，故

$$\delta_x=\delta_{\mathrm{t}}\le\frac{0.5^{n+1}}{n+1}$$

4.4.4.2 舍入误差

由于本人编写的大数类能够保证加减乘不舍弃任何小数位，且除法保证了一定的误差裕量，故大数类运算误差忽略不计。

设精度为 p，各阶舍入误差为：

$$\sigma=\frac{1}{2}\times 10^{-(p+1)}$$

由于$\ln(1.5)$为预先存储的精确到小数点后 32 位的精确值，因此舍入误差为$\frac{1}{2}\times10^{-32}$

综上：综合方法误差与舍入误差，总误差为：

$$\frac{0.5^{n+1}}{n+1}+\frac{1}{2}\times10^{-(p+1)}+\frac{1}{2}\times10^{-32}$$

其中，p为所要求的精度；n为等分分数，本程序中取 40。

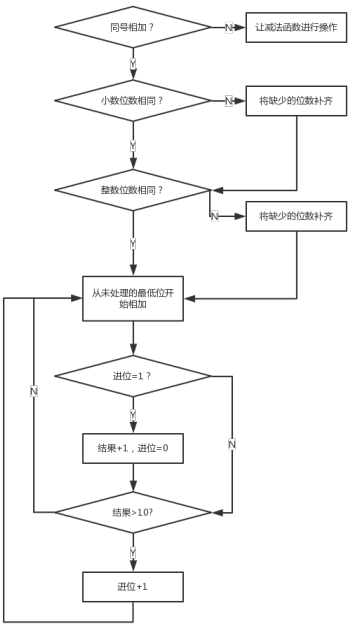
五、程序框图

5.1 大数类

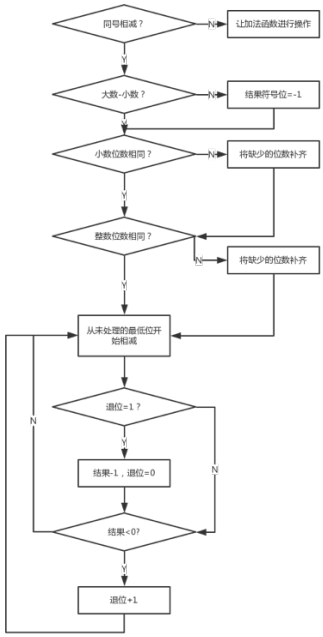
虽说大数类仅仅是程序中的一部分，但这却是不可或缺的一部分，且占到了代码总量的一半以上！本次编写程序的大数类纯本人手工打造，未直接使用任何网上的大数类代码。因此在编写逼近方法（尤其是拉格朗日方法）的过程中，也经常发现之前编写代码的 bug 并加以修正。也许现在的大数类（最终版）也还存在一些考虑不周的情况，无法直接移植到其它的程序中，但在本次项目的使用范围内，经过多次实验尝试，没有出现过问题。

同时，本人清楚的认识到，自己编写的大数类还存在一定的缺点：首先，这个数值必须有小数部分，以至于在进行一些简单整数的运算时也要在其后面加上一个“.0”，造成了一定不方便；其次，乘法部分没有去除小数位最后的“0”的功能，因此会造成尾“0”的累计，增加空间复杂度和时间复杂度。考虑到期末时间紧迫，加上实验所用时间也在可接受范围内，对这两个缺点暂时没有进行改进。

5.1.1 加减法

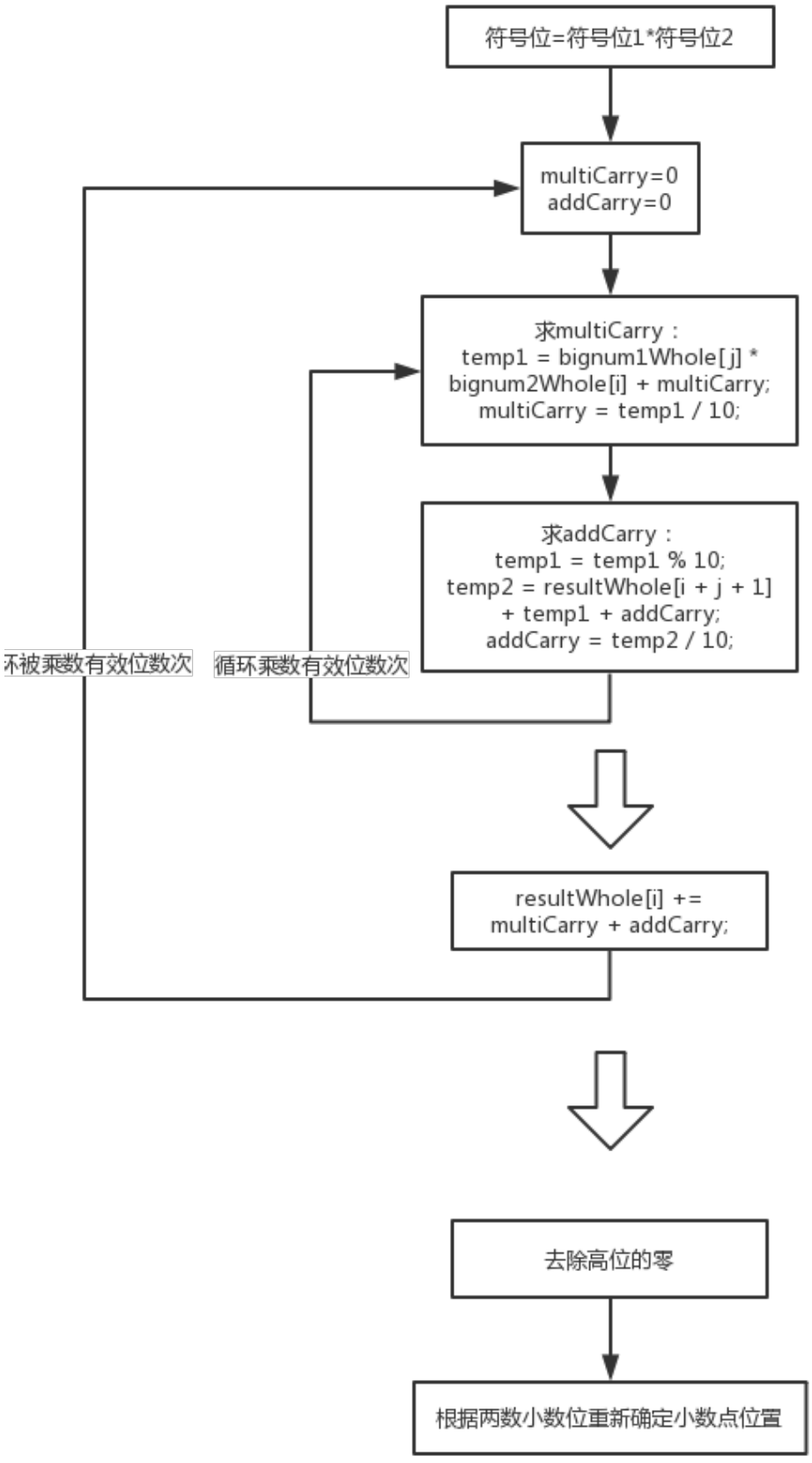


加法

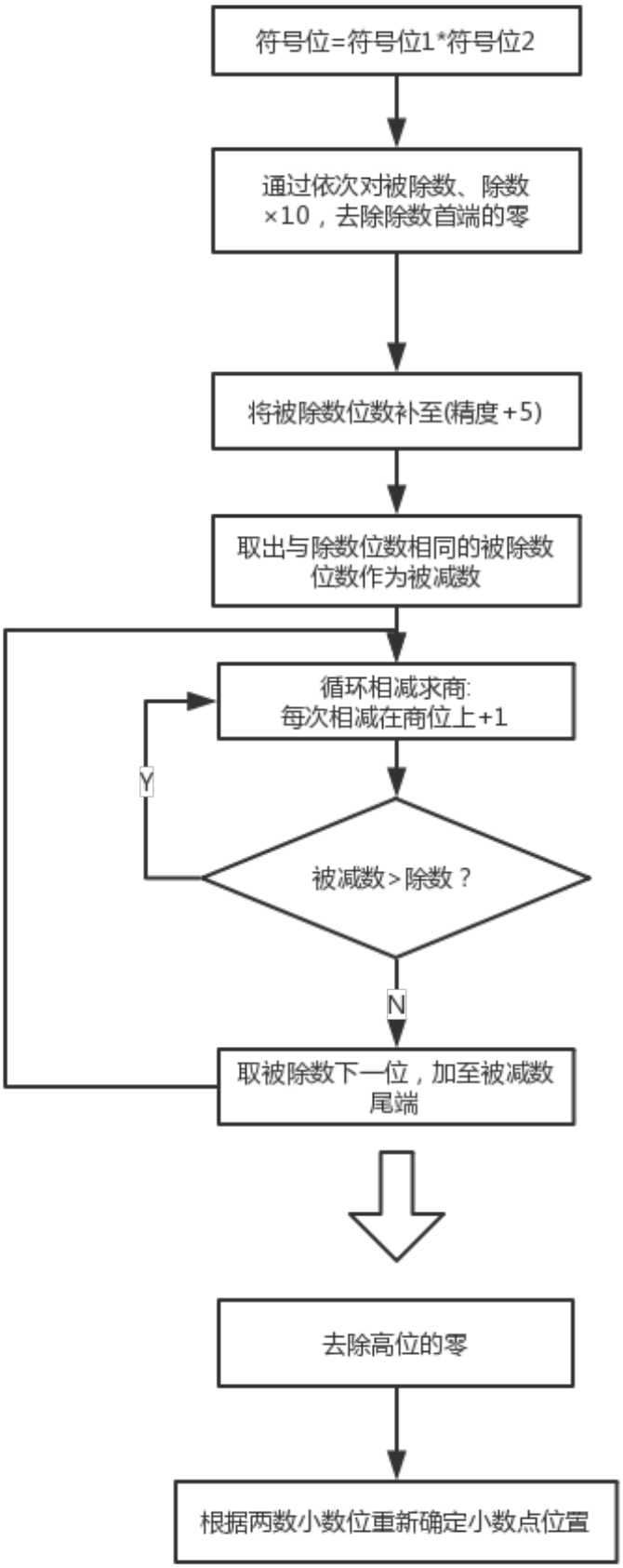


减法

5.1.2 乘法


符号位=符号位1*符号位2
multiCarry=0
addCarry=0
求multiCarry：
temp1 = bignum1Whole[j] *
bignum2Whole[i] + multiCarry;
multiCarry = temp1 / 10;
求addCarry：
temp1 = temp1 % 10;
temp2 = resultWhole[i + j + 1]
+ temp1 + addCarry;
addCarry = temp2 / 10;
环被乘数有效位数次
循环乘数有效位数次
resultWhole[i] +=
multiCarry + addCarry;
去除高位的零
根据两数小数位重新确定小数点位置

5.1.3 除法

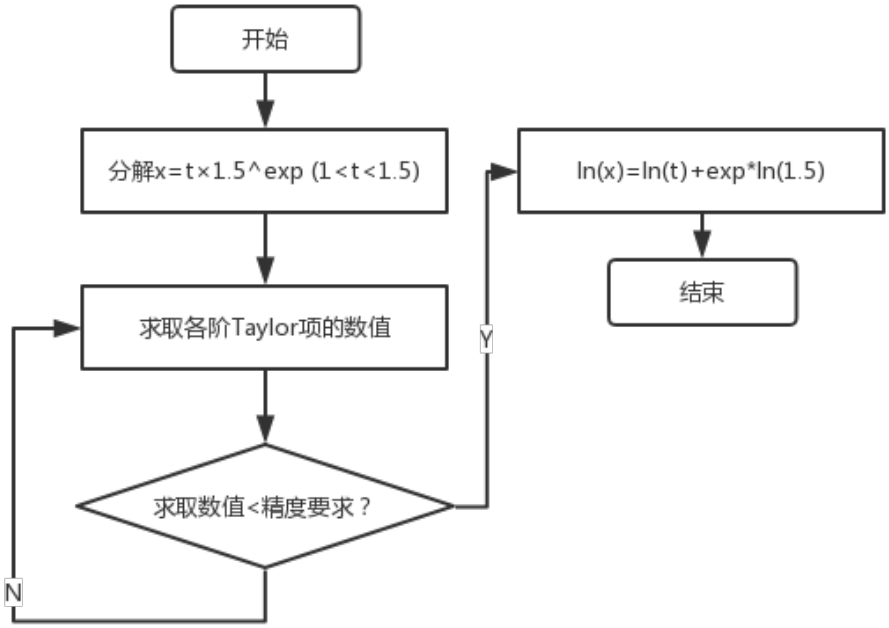


由于除法是一个结果小数位数不取决于输入的运算（而且可能无限），因此将小数位数截断为(精度+5)，如20+5=25

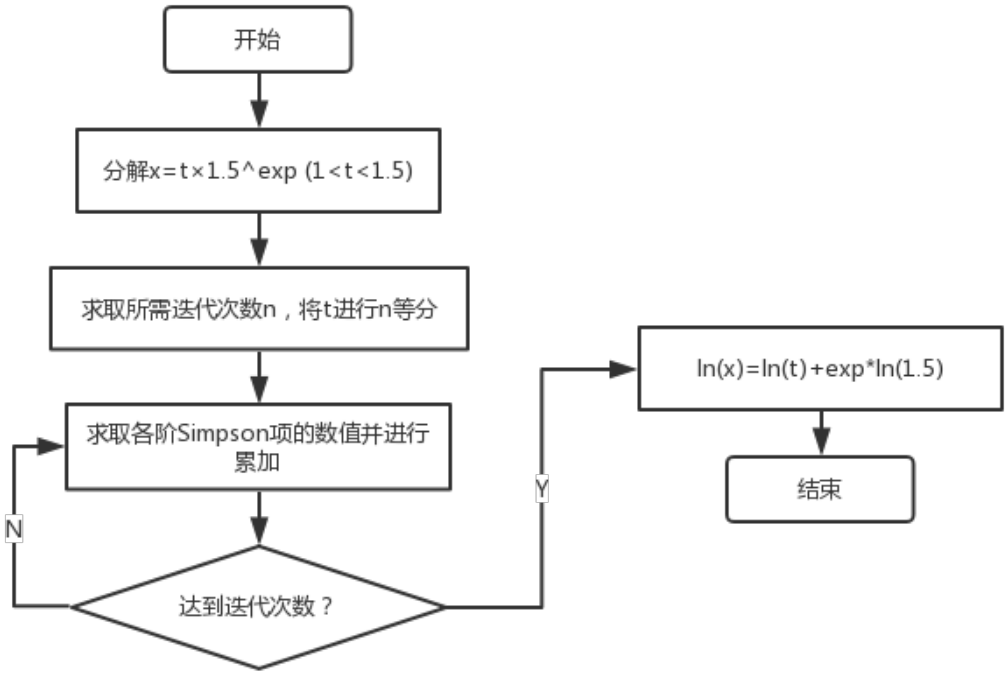
5.2 求取方法

四种求取方法的步骤严格按照以上各方法中“方法原理”进行处理。

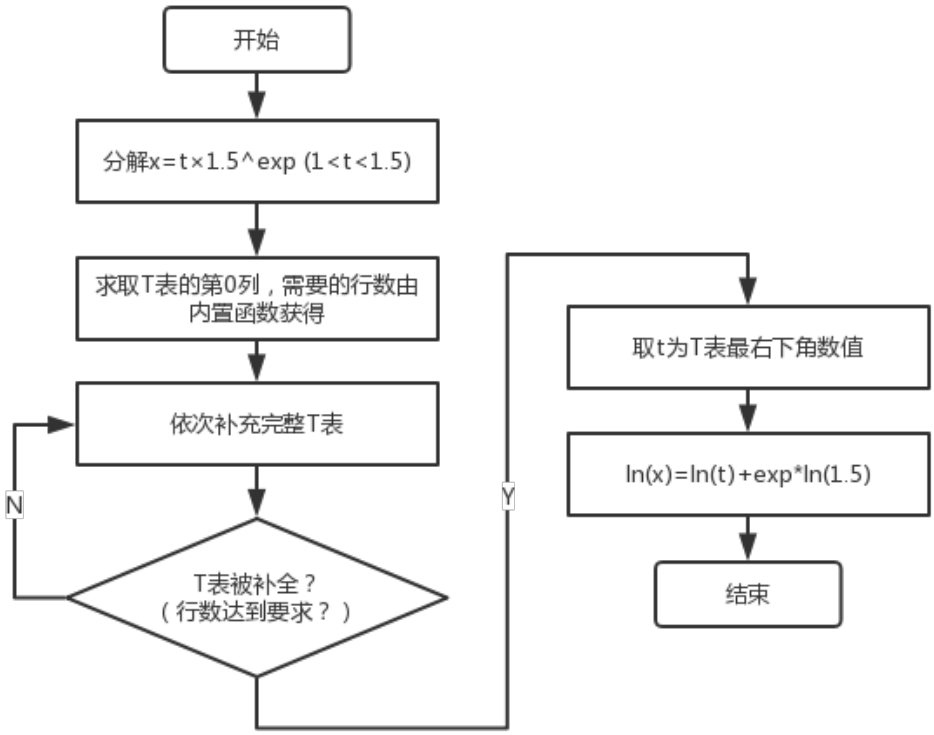
5.2.1 Taylor 展开



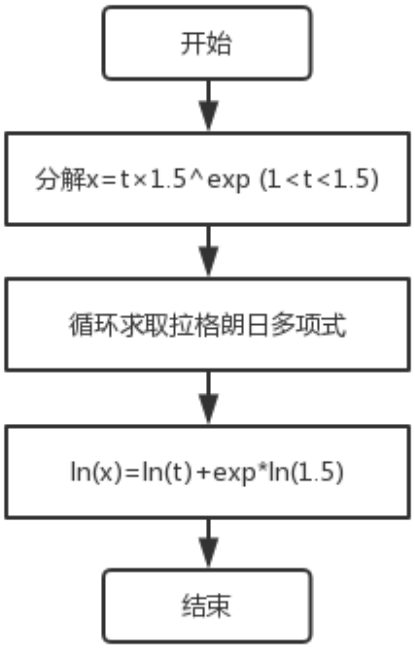
5.2.2 复化辛普生公式



5.2.3 龙贝格算法



5.2.4 拉格朗日插值多项式逼近



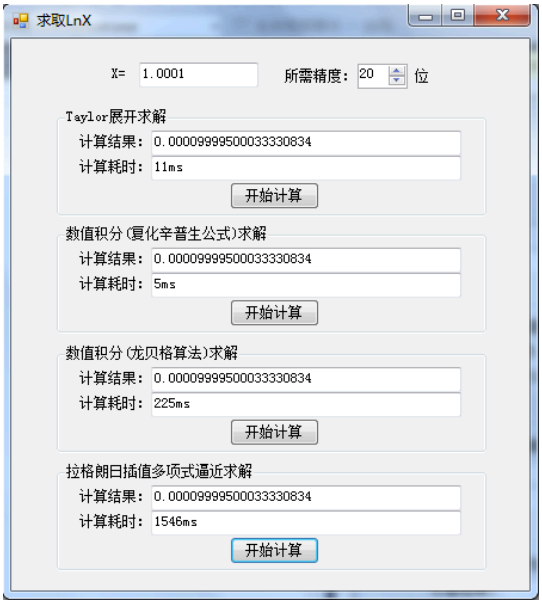
六、实验结果与分析

将x按照以下方式分解：

$$x = t \times 1.5^{exp} \ (1 < t < 1.5)$$

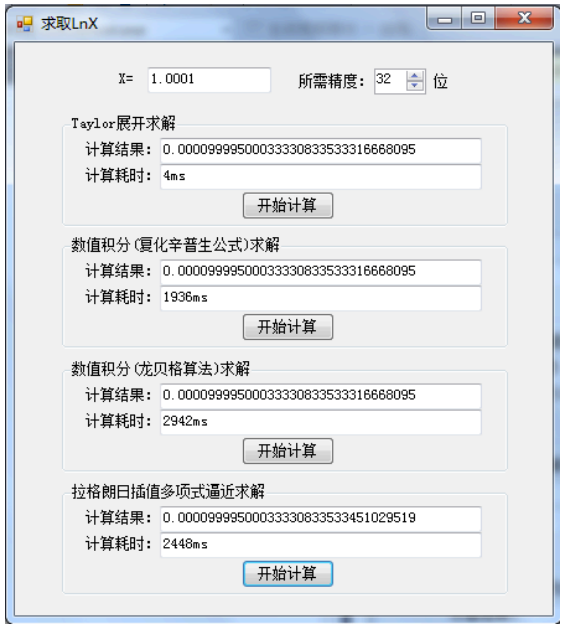
对于不同的t值，所需要的计算量是不同的（这在第四部分已进行阐述），因此以下将对不同的t值和不同的精度要求进行实验分析。

6.1 $t \to 1$，低精度要求



可以看出，在 t 值较小时，前三种方法均能够以很快的速度输出结果。

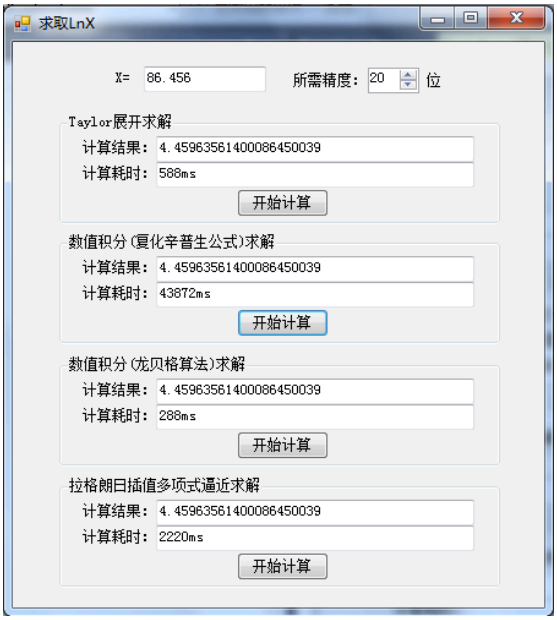
6.2 $t \to 1$，高精度要求



经过对比可以发现，Taylor 展开求解在 t 值较小时均能够保证较高的运算速度。而复化

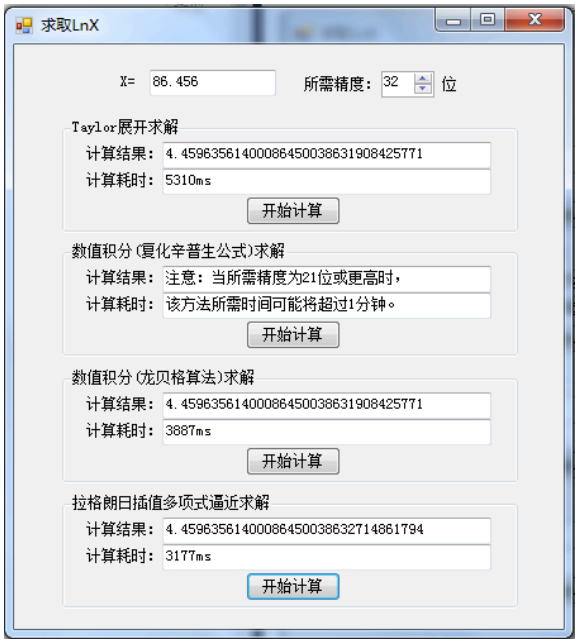
辛普生公式和龙贝格算法的运算耗时随着精度要求提高而增加。拉格朗日插值多项式逼近求解的方法随精度要求提高，耗时有略微增加，且可能无法达到 24 位以上的精度要求。

6.3 $t \to 1.5$，低精度要求



经过对比可以发现，当$t \to 1.5$时，Taylor 展开求解和复化辛普生公式耗时有了成倍的增长。龙贝格算法与拉格朗日插值多项式逼近求解的方法耗时有略微增加，但并不明显。

6.4 $t \to 1.5$，高精度要求



经过对比可以发现，当$t \to 1.5$时且精度要求较高时，Taylor 展开求解的耗时增加特别大。龙贝格算法与拉格朗日插值多项式逼近求解的方法耗时有略微增加，但并不明显。复化辛普生公式在 32 位精度下的运算时间超过 1 小时，故没有进行$t \to 1.5$且精度要求较高时的完整

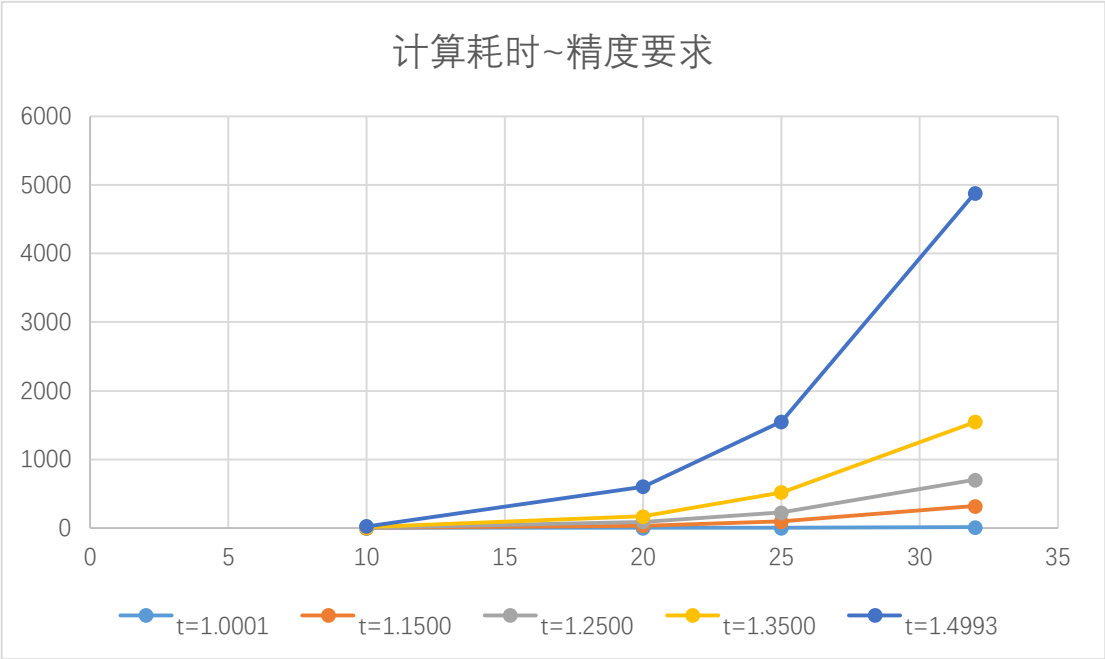
测试。

6.5 实验结果分析

对更多数据进行测试，以下数据从左至右t值逐渐增大，列举出所需时间（单位：ms）如以下图表所示。

Taylor 展开：

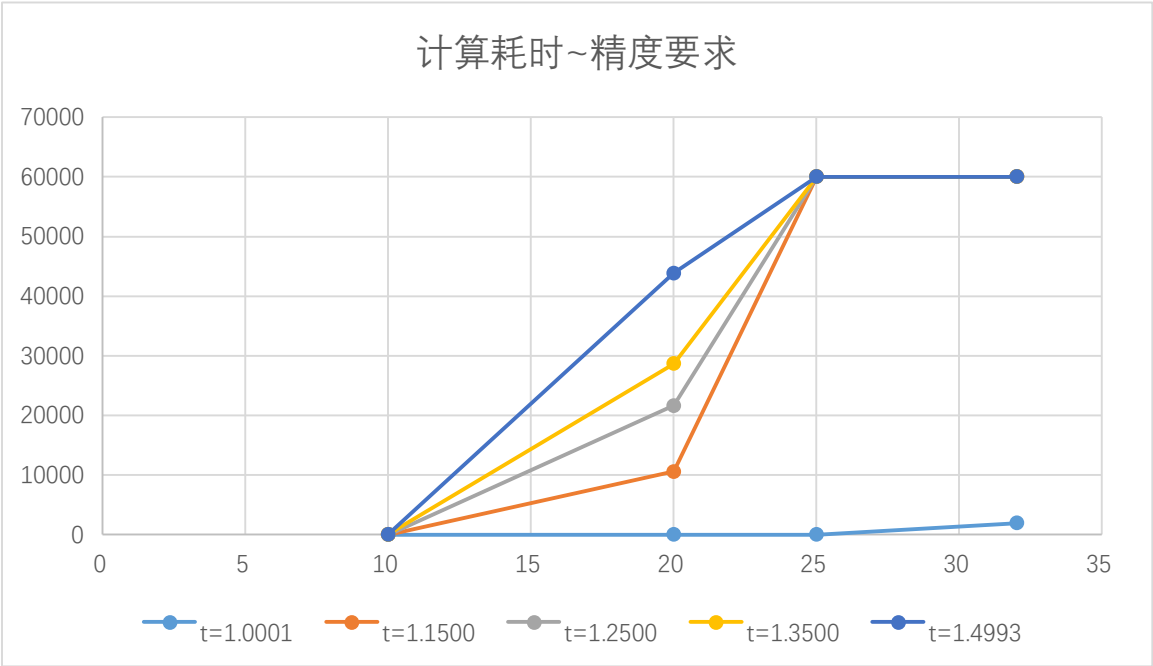
精度要求|输入值	1.0001	3.8812	9.4922	23.067	86.456
10	0	3	5	11	27
20	1	33	89	171	603
25	2	97	226	520	1549
32	12	318	700	1547	4883



从以上图表可以看出：Taylor 展开方法计算耗时与 t 值和精度要求均成正相关。且 t 值越大，随精度提升导致的的计算耗时增加效应越明显。

复化辛普生公式：

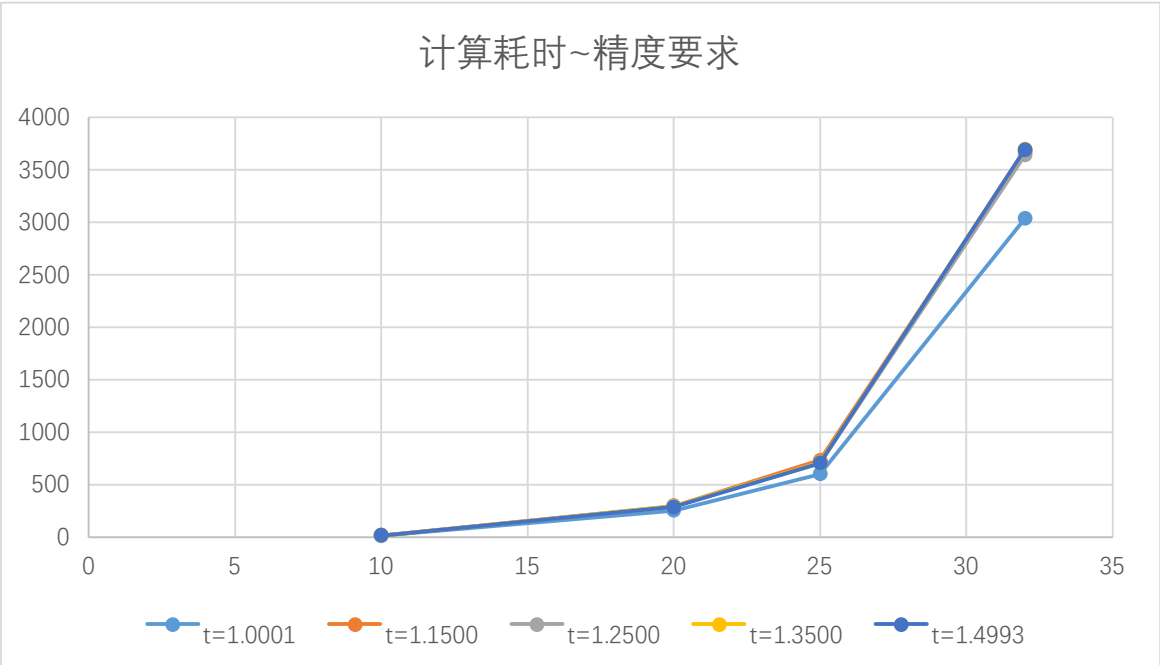
精度要求|输入值	1.0001	3.8812	9.4922	23.067	86.456
10	2	15	32	46	60
20	2	10599	21651	28665	43872
25	23	>1min	>1min	>1min	>1min
32	1922	>1min	>1min	>1min	>1min



从以上图表可以看出：复化辛普生公式方法计算耗时与 t 值和精度要求均成正相关。且 t 值越大，随精度提升导致的的计算耗时增加效应越明显。

龙贝格算法：

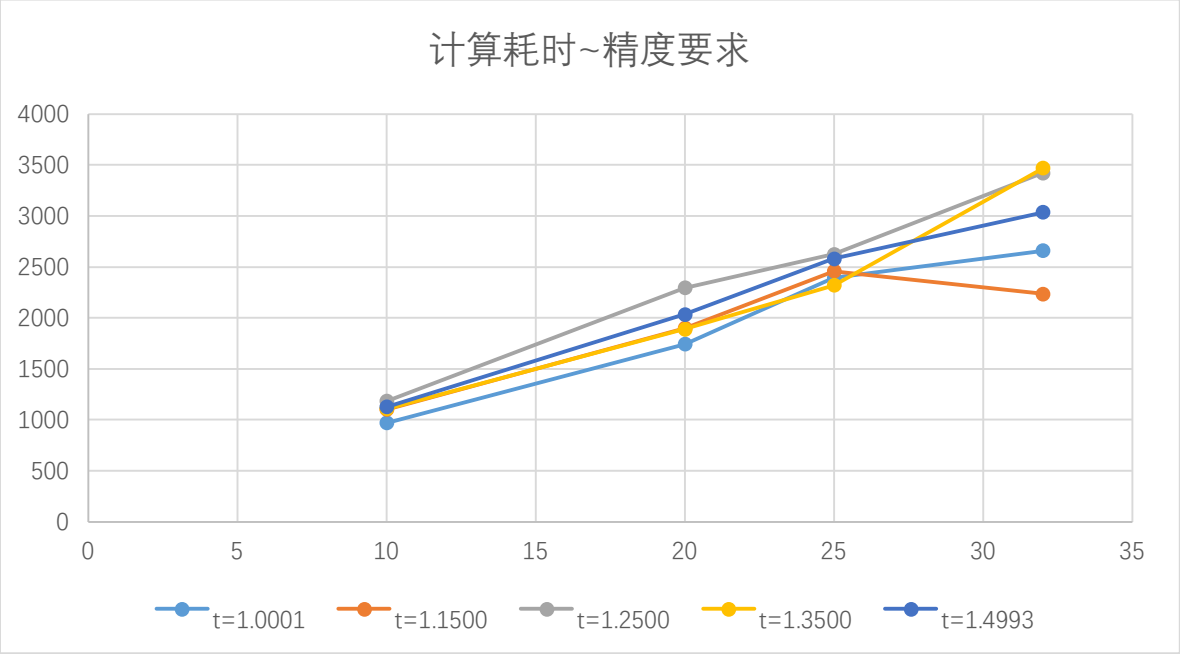
精度要求|输入值	1.0001	3.8812	9.4922	23.067	86.456
10	12	15	16	14	16
20	253	296	296	294	287
25	600	733	706	711	704
32	3036	3674	3641	3696	3692



从以上图表可以看出：龙贝格算法计算耗时与精度要求成正相关，与 t 值关系并不明显。

拉格朗日插值多项式逼近：

精度要求|输入值	1.0001	3.8812	9.4922	23.067	86.456
10	968	1102	1184	1107	1127
20	1743	1898	2294	1890	2034
25	2391	2457	2623	2319	2581
32	2658	2236	3421	3469	3036



从以上图表可以看出：拉格朗日插值多项式逼近方法计算耗时与精度要求成正相关，与 t 值关系并不明显。

同时从以上数据可以看出，从平均角度来看，Taylor 展开拥有最高的运算效率。而对于 t 值较大且精度要求较高的数值，选取龙贝格算法可以获得比 Taylor 展开更快的速度。而无论何时，复化辛普生公式都不是最佳选择。