实验报告

姓名	
评分	

实验报告

课程名称： 数值分析

课题名称： 数值分析第一次实验报告

专　　业： 地球物理

姓　　名： 王锴

班　　级： 201145

完成日期： 2016 年 10 月 23 日

实验报告

一、实验名称

数值分析第一次实验报告

二、 实验目的

（1） 掌握数值分析的基本方法；
（2） 掌握迭代算法，分析算法数值稳定性及算法复杂度；
（3） 培养编程与上机调试能力；
（4） 熟悉 Matlab 软件环境。

三、 实验要求

（1） 通过两种近似算法计算 ln2，并对它们进行收敛性分析、收敛速率比较，以及算法复杂度对比，寻找一种较优算法；

（2） 通过两种算法计算$I_n=e^{-1}\int_0^1 x^n e^x dx$，并分析它们的数值稳定性；

（3） 通过两种算法计算$P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0$,分析它们的算法复杂度；

（4） 利用 Matlab 软件作为辅助工具来实现该实验。

四、 实验原理

（1） $\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\ldots+(-1)^{n-1}\frac{x^n}{n}+\ldots$，令 x=1，进行 n 次迭代算法，可以近似求出 ln2 的值；

（2） $\ln\frac{1+x}{1-x}=2(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots+\frac{x^{2n-1}}{2n-1}+\ldots)$，令 x=1/3，进行 n 次迭代算法，可以近似求出 ln2 值；

（3） $I_n=e^{-1}\int_0^1 x^n e^x dx=1-nI_{n-1}$，(n=1,2,…)，求出$I_n$中某一项的值的近似值，可由迭代算法求出$I_n$；

（4） $P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0$可由$a_kx^k$逐项叠加得到；

（5） $P_n(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0$可由秦九韶算法

$$\begin{cases} S_n=a_n, \\ S_k=xS_{k+1}+a_k(k=n-1,\ldots,2,1,0), \\ P_n(x)=S_0. \end{cases}$$

迭代计算得到。

五、 实验题目

（1）对比求 ln2 近似值的两种算法：

ln2=1-$\frac{1}{2}$+$\frac{1}{3}$-…+$(-1)^{n-1}\frac{1}{n}$+…， （5.1）

ln2=2[$\frac{1}{3}$+$\frac{1}{3*3^3}$+ $\frac{1}{5*3^5}$+…+$\frac{1}{(2n-1)3^{(2n-1)}}$+…]。 （5.2）

（2）对比求I_n=$e^{-1}\int_0^1 x^n e^x dx$=1-n$I_{n-1}$，(n=1,2,…)的两种算法并估计误差：

I_0=1-e^{-1}≈0.6321，I_n=1-nI_{n-1}， （5.3）

I_9≈0.0684，$I_{n-1}=\frac{1}{n}(1-I_n)$ （n=9, 8, …, 1）； （5.4）

（3）对比求$P_n(x)=a_nx^n+a_{n-1}x^{n-1}$+…+$a_1x$+$a_0$的两种算法：

$P_n(x)=\sum_{k=0}^{n} a_kx^k$， （5.5）

$$\begin{cases} S_n=a_n, \\ S_k=xS_{k+1}+a_k(k=n-1,\ldots,2,1,0), \\ P_n(x)=S_0. \end{cases}$$ （5.6）

六、 实验步骤

（1）求 ln2 的近似值：

①指定迭代次数 N=300000，s1、s2 为两个 1*N 的一维数组，s1(1,i)表示（5.1）式中 n=i 时 ln2 的近似值，s2(1,i)表示（5.2）式中 n=i 时 ln2 的近似值；

②设定初始值 s1(1,1)=1，s2(1,1)=2/3；sgn=1 表示（5.1）式中第 n 个数的符号，以后每次迭代乘以-1，以减少幂函数的使用；a=1 表示（5.2）式中第 n 个数分母部分的 2n-1，以后每次迭代加 2，以减少乘法的使用；b=3 表示（5.2）式中第 n 个数分母部分的 3^(2n-1)，以后每次迭代乘以 9，以减少幂函数的使用；

③对（5.1）式的求解过程为：使用 for 循环，循环参数 i 从 2 到 N，每次迭代计算令 sgn=sgn*(-1)，s1(1,i)=s1(1,i-1)+sgn/i，最终的近似值为 s_1=s1(1,N)；对（5.2）式的求解过程为：使用 for 循环，循环参数 i 从 2 到 N，每次迭代计算令 a=a+2，b=b*9，s2(1,i)=s2(1,i-1)+2/(a*b)，最终的近似值为 s_2=s2(1,N)；

④用 Matlab 软件编程实现③所述算法；

⑤运行并分别计算两种算法各自所需时间，并在同一坐标系中做出两种算法迭代过程的函数图像；

⑥进行数值实验，改变迭代次数 N，重复上述算法，分析比较两种算法。

（2）求I_n=$e^{-1}\int_0^1 x^n e^x dx$=1-n$I_{n-1}$，(n=1,2,…)

①指定迭代次数 N=9，IA、IB 为两个 1*N 的一维数组，IA(1,i)表示（5.3）式中 n=i 时I_n的值，IB(1,i)表示（5.4）式中 n=i 时I_n的值；

②设定初始值 IA(1,1)=0.6321，IB(1.N+1)=0.0684；

③对（5.3）式的求解过程为：使用 for 循环，循环参数 i 从 2 到 N+1，每次迭代

计算令 IA(1,i)=1-(i-1)*IA(1,i-1)；对（5.4）式的求解过程为：使用 for 循环，循环参数 i 从 N 到 1，每次迭代计算令 IB(1,i)=(1-IB(1,i+1))/i；

④用 Matlab 软件编程实现③所述算法；

⑤运行并分别计算两种算法各自所需时间，并在同一坐标系中做出两种算法迭代结果的函数图像；

⑥进行数值实验，改变初始值 IA(1,1)、IB(1,N+1)，重复上述算法，分析比较两种算法。

（3）求$P_n(x)=a_nx^n+a_{n-1}x^{n-1}$+…+$a_1x$+$a_0$：

①指定迭代次数 N=200000，an=1:N+1 为系数序列，x=0.08，P1 为（5.5）式中$P_n(x)$的值，P2 为（5.6）式中$P_n(x)$的值，S 为 1*N+1 的一维数组，表示（5.6）式中S_n序列；

②设定初始值 P1=0，b=1 为（5.5）式中的x^k项，以后每次迭代乘以 x，以减少幂函数的使用，S(1,N+1)=an(1,N+1)；

③对（5.5）式的求解过程为：使用 for 循环，循环参数 i 从 0 到 N，每次迭代计算令 P1= P1+an(1,i+1)*b;b=b*x；对（5.6）式的求解过程为：使用 for 循环，循环参数 i 从 N-1 到 0，每次迭代计算令 S(1,i+1)=x*S(1,i+2)+an(1,i+1)；

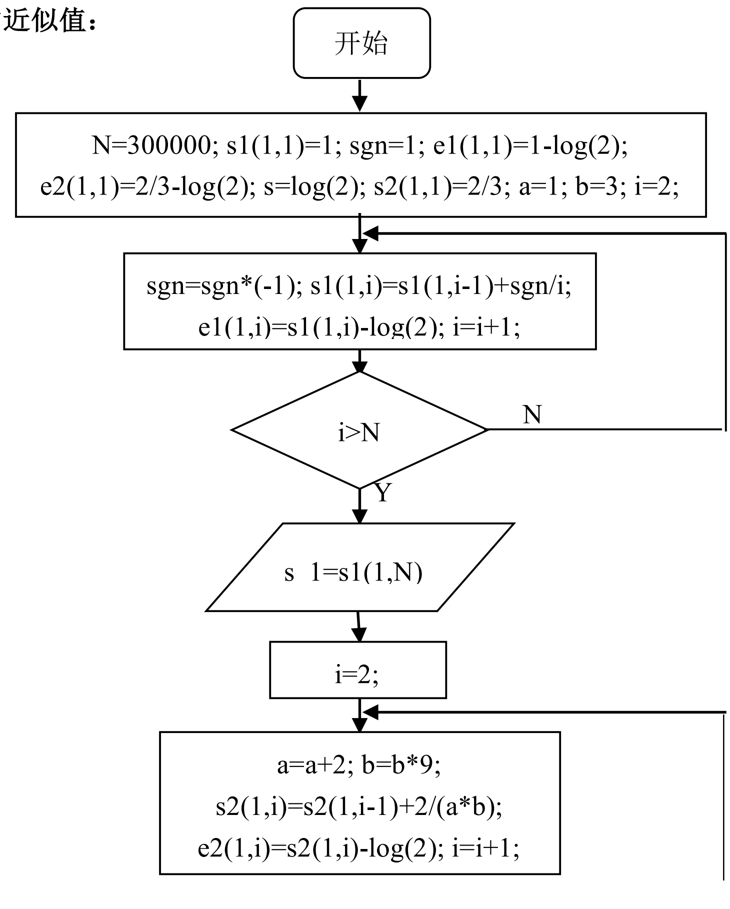
④用 Matlab 软件编程实现③所述算法；

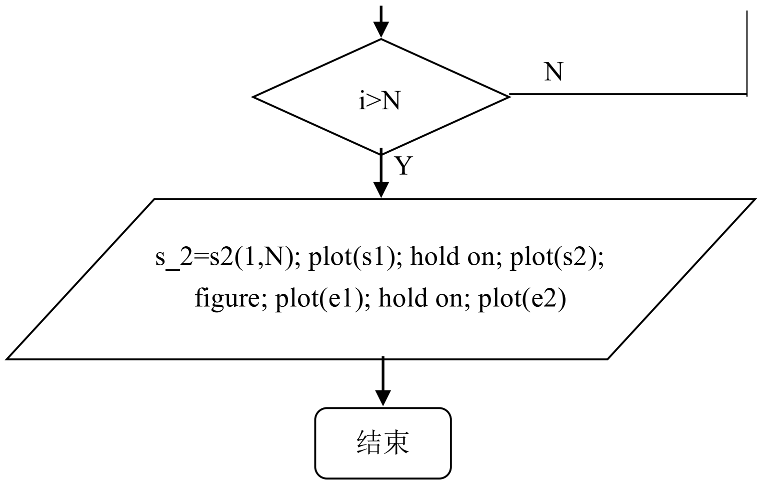
⑤运行并分别计算两种算法各自所需时间；

⑥进行数值实验，改变 N 和 x 的值，重复上述算法，分析比较两种算法。

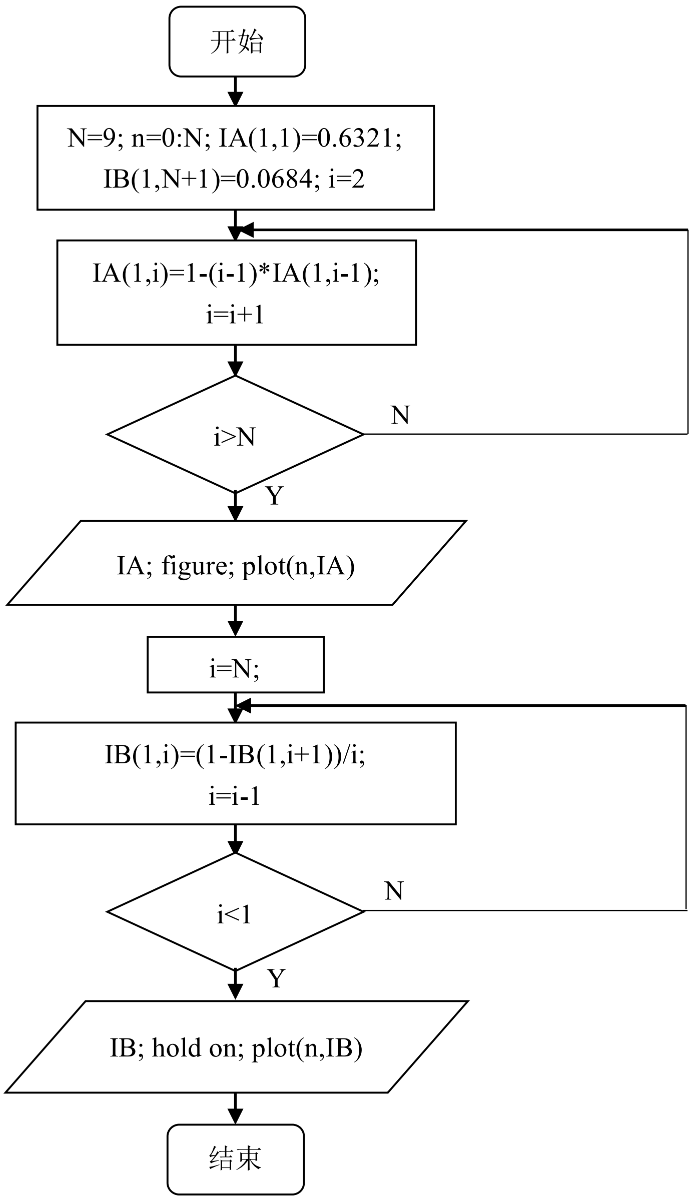
七、 实验整体流程图或算法

（1）求 ln2 的近似值：

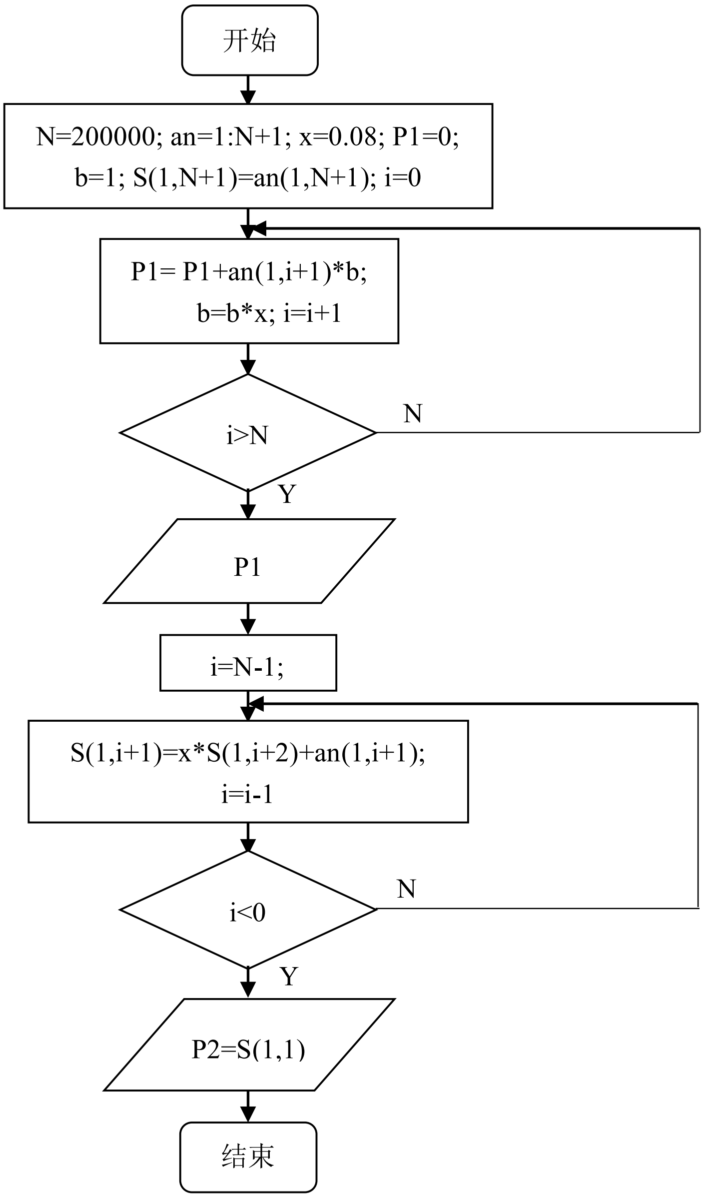




（2）求$I_n=e^{-1}\int_0^1 x^n e^x dx=1-nI_{n-1}$，(n=1,2,…)：



（3）求$P_n(x) = a_n x^n + a_{n-1} x^{n-1}$+…+$a_1 x$+$a_0$：



八、 程序及其运行结果

（1）程序：

①求 ln2 的近似值：

```
N=300000;              %迭代次数
s=log(2)
s1(1,1)=1;         %算法 5.1 的初始值
e1(1,1)=1-log(2);
sgn=1;                %算法 5.1 式中第 n 个数的符号
s2(1,1)=2/3;      %算法 5.2 的初始值
e2(1,1)=2/3-log(2);
```

```
a=1;                %5.2 式中第 n 个数分母部分的 2n-1
b=3;                %5.2 式中第 n 个数分母部分的 3^(2n-1)

for i=2:N                          %算法 5.1
      sgn=sgn*(-1);                    %每次迭代乘以-1，以减少幂函数的使用
      s1(1,i)=s1(1,i-1)+sgn/i;   %迭代计算 ln2 近似值
      e1(1,i)=s1(1,i)-log(2);   %绝对误差
end
s_1=s1(1,N)      %输出 5.1 式第 N 个迭代结果

for i=2:N                          %算法 5.2
    a=a+2;                           %每次迭代加 2，以减少乘法的使用
    b=b*9;                           %每次迭代乘以 9，以减少幂函数的使用
    s2(1,i)=s2(1,i-1)+2/(a*b); %迭代计算 ln2 近似值
    e2(1,i)=s2(1,i)-log(2); %绝对误差
end
s_2=s2(1,N)     %输出 5.2 式第 N 个迭代结果

plot(s1); hold on; plot(s2) %在同一张图中绘制算法 5.1 及 5.2 的收敛过程
figure; plot(e1); hold on; plot(e2) %绝对误差曲线
```

②求I_n=$e^{-1}\int_0^1 x^n\,e^x dx$=1-n$I_{n-1}$，(n=1,2,…):

```
N=9;                     %迭代次数
n=0:N;                   %用于绘制坐标

IA(1,1)=0.6321;        %算法 5.3 的初始值
for i=2:N+1
      IA(1,i)=1-(i-1)*IA(1,i-1);   %迭代计算 I
end
IA                       %输出 IA
figure;plot(n,IA)     %绘制 IA 迭代过程图

IB(1,N+1)=0.0684;     %算法 5.4 的初始值
for i=N:-1:1
    IB(1,i)=(1-IB(1,i+1))/i;        %迭代计算 I
end
IB                       %输出 IB
hold on;plot(n,IB)   %在同一张图上绘制 IB 迭代过程图
```

③求$P_n(x)=a_n x^n+a_{n-1}x^{n-1}$+…+$a_1 x$+$a_0$：

```
N=200000;           %迭代次数
an=1:N+1;           %系数序列
x=0.08;             %未知数 x
```

```
P1=0;                %算法 5.5 的初始值
b=1;                 %算法 5.5 式中的 x^k 项
for i=0:N
    P1= P1+an(1,i+1)*b;   %迭代计算 P
    b=b*x;                    %每次迭代乘以 x，以减少幂函数的使用
end
P1                    %输出 P

S(1,N+1)=an(1,N+1);        %算法 5.6 的初始值
for i=N-1:-1:0
    S(1,i+1)=x*S(1,i+2)+an(1,i+1);     %迭代计算 P
end
P2=S(1,1)           %输出 P
```

（2）运行结果

①求 ln2 的近似值：

算法 5.1 用时 0.051s，

```
s_1 =

   0.693145513896101
```

图 8.1 算法 5.1 的近似结果

算法 5.2 用时 0.039s

```
s_2 =

   0.693147180559945
```

图 8.2 算法 5.2 的近似结果

其中 ln2≈0.693147180559945。

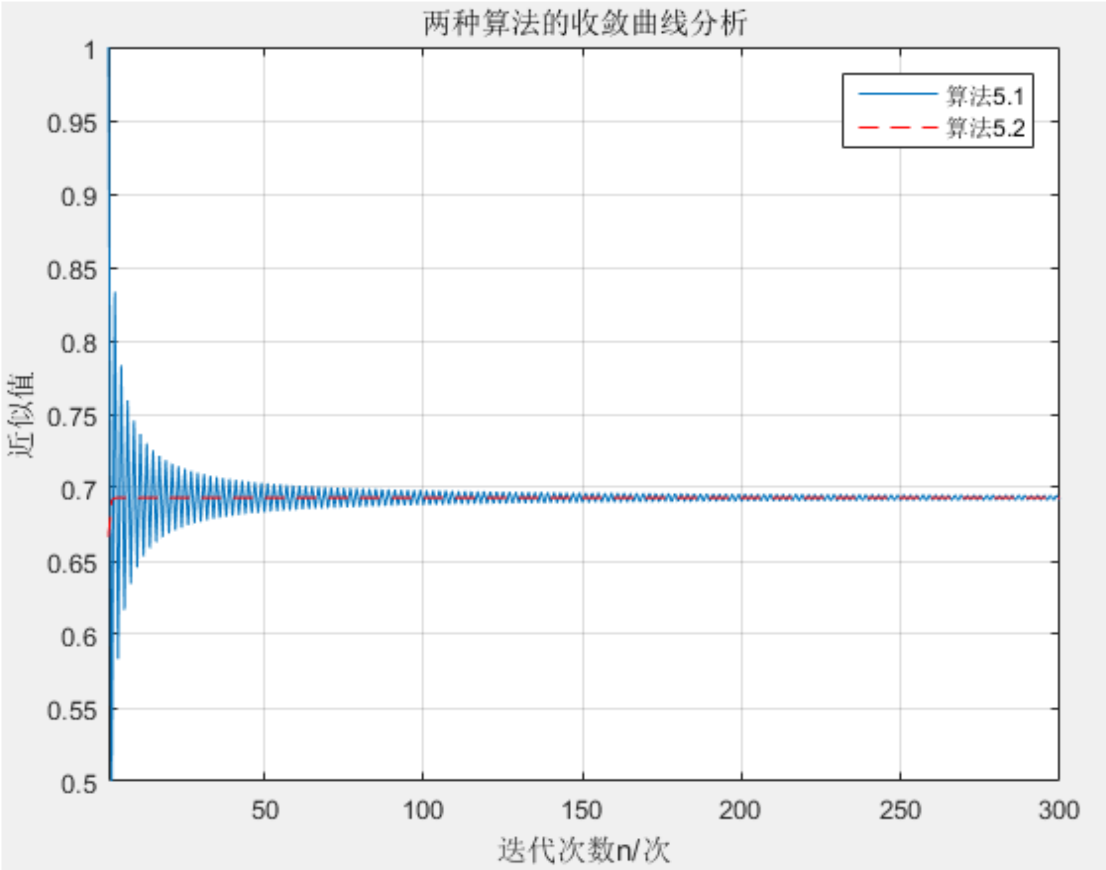


图 8.3 两种算法的收敛曲线分析

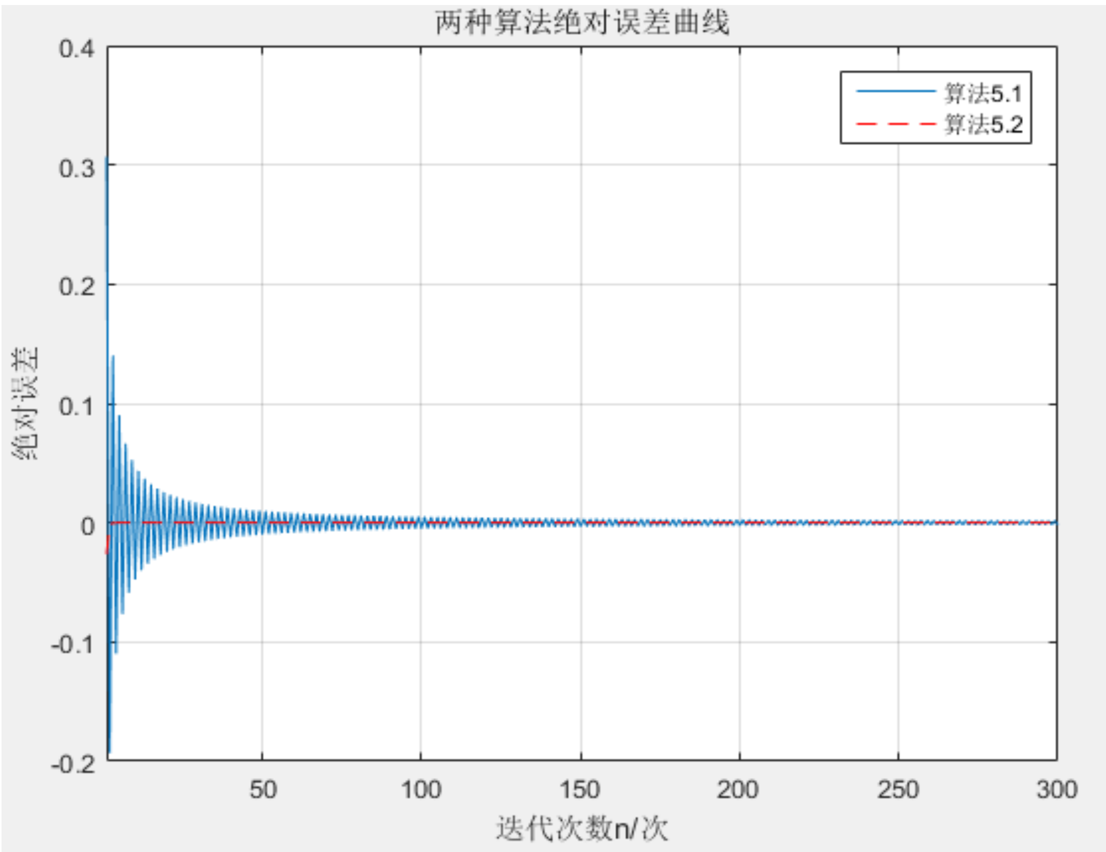


图 8.4 两种算法绝对误差曲线比较

改变迭代次数 N：

N=3000 时，

算法 5.1 用时 0.003s，算法 5.2 用时 0.002s，

```
s_1 =

   0.692980541671060


s_2 =

   0.693147180559945
```

图 8.5 N=3000 时两种算法的近似结果

N=10000000 时，

算法 5.1 用时 1.692s，算法 5.2 用时 1.333s，

```
s_1 =

   0.693147130560106


s_2 =

   0.693147180559945
```

图 8.6 N=10000000 时两种算法的近似结果

②求$I_n=e^{-1}\int_0^1 x^n e^x dx$=1-n$I_{n-1}$，(n=1,2,…):

```
IA =

   0.6321    0.3679    0.2642    0.2074    0.1704    0.1480    0.1120    0.2160   -0.7280    7.5520
```

图 8.7 算法 5.3 的求解结果

```
IB =

   0.6321    0.3679    0.2642    0.2073    0.1709    0.1455    0.1268    0.1121    0.1035    0.0684
```

图 8.8 算法 5.4 的求解结果

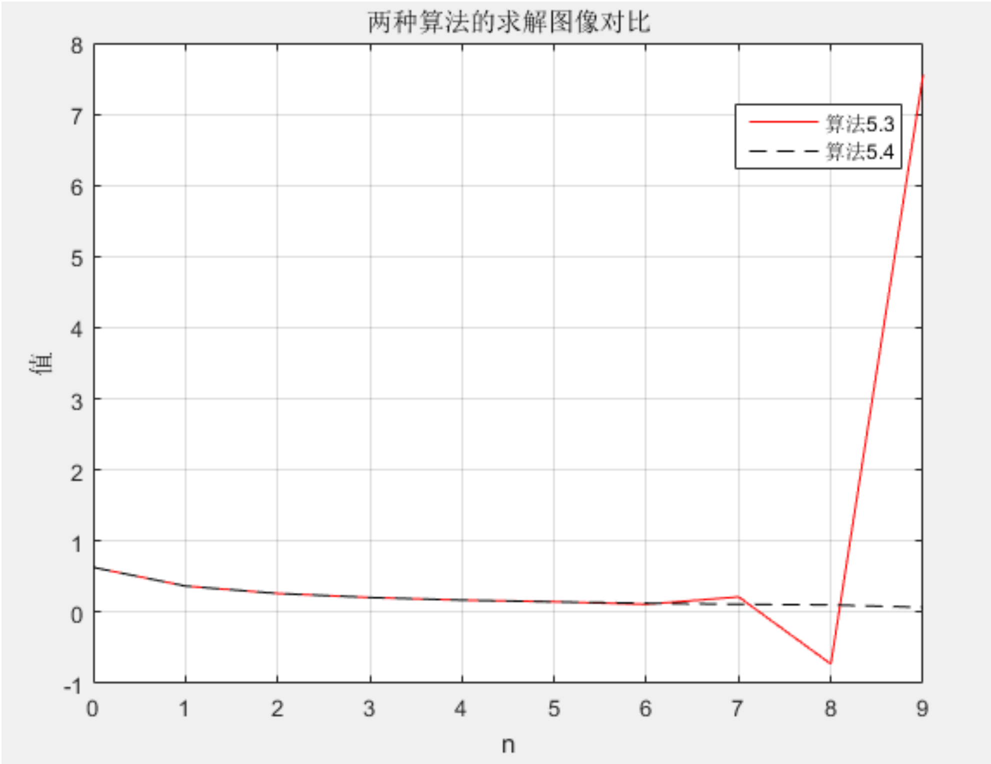


图 8.9 两种算法的求解图像对比

改变初始值 IA(1,1)、IB(1,N+1)：

IA(1,1)=0.63212，IB(1,N+1)=0.05 时，

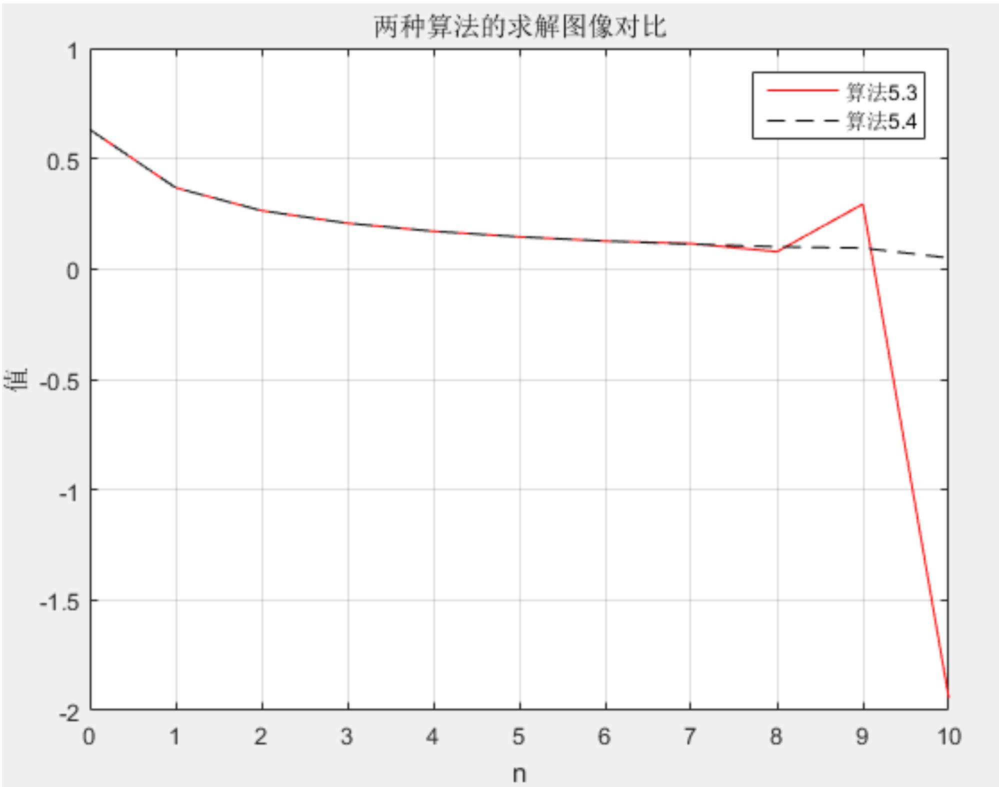


图 8.10 IA(1,1)=0.63212，IB(1,N+1)=0.05 时两种算法的求解图像对比

IA(1,1)=0.63212056，IB(1,N+1)=0.1 时，

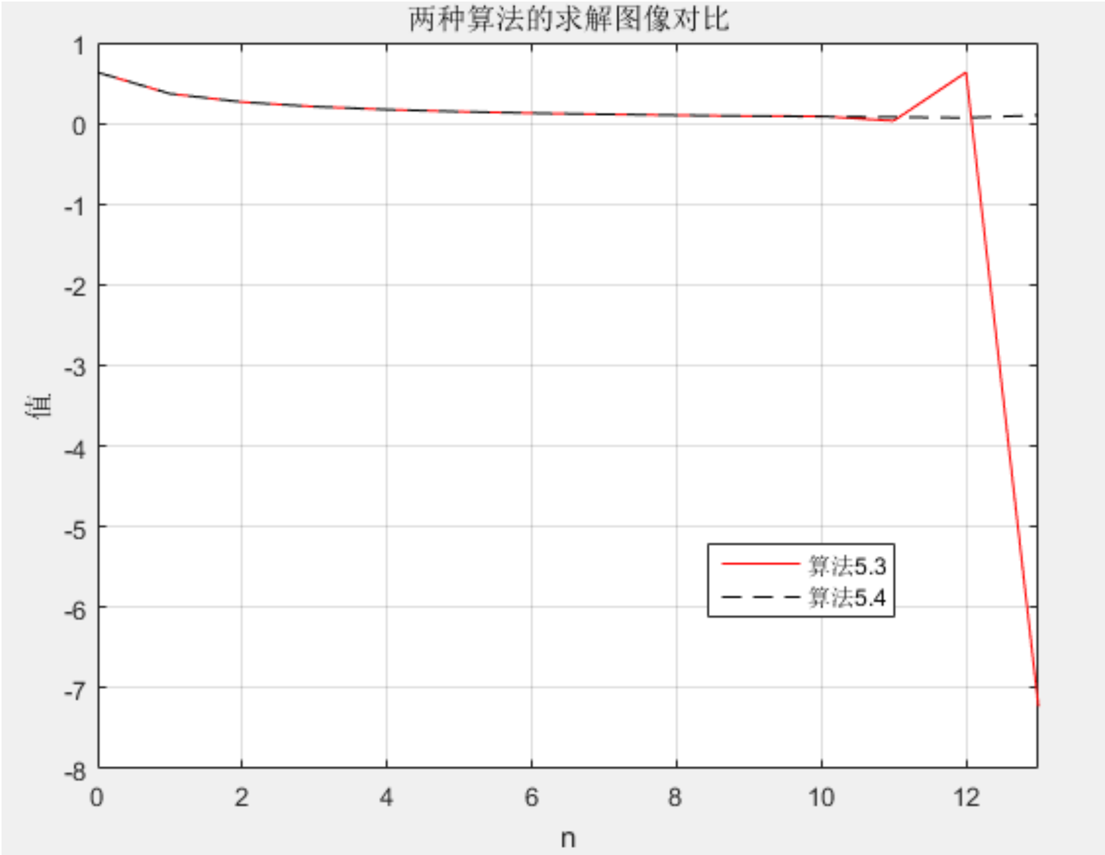


图 8.11　IA(1,1)=0.63212056，IB(1,N+1)=0.1 时两种算法的求解图像对比

③求$P_n(x) = a_n x^n + a_{n-1} x^{n-1}+\ldots+a_1 x+a_0$：

算法 5.5 用时 0.026s，

```
P1 =

   1.181474480151229
```

图 8.12 算法 5.5 的运算结果

算法 5.6 用时 0.025s，

```
P2 =

   1.181474480151229
```

图 8.13 算法 5.6 的运算结果

改变 N、x：

N=2000000，x=0.8 时，

算法 5.5 用时 0.513s，算法 5.6 用时 0.213s，

```
P1 =

  25.000000000000007


P2 =

    25
```

图 8.14 N=2000000，x=0.8 时两种算法的运算结果

N=20000000，x=0.1 时，
算法 5.5 用时 2.539s，算法 5.6 用时 2.111s，

```
P1 =

   1.234567901234568


P2 =

   1.234567901234568
```

图 8.15 N=20000000，x=0.1 时两种算法的运算结果

九、 实验结果分析

（1）求 ln2 的近似值：

①收敛性分析：

对于算法 5.1，通项为a_n=(-1)^(n-1)/n，因为|a_n|=1/n 为正项递减数列，故|a_n|为收敛的，即a_n为绝对收敛的，则a_n为收敛的。

对于算法 5.2，通项为b_n=2/[3^(2n-1)* (2n-1)]，因为$b_{n+1}/b_n=\frac{1}{9}*\frac{2n-1}{2n+1}$<1 恒成立，故$b_n$为收敛的。

可见两种算法都满足收敛性的要求。

②收敛速率比较：

由图 8.3 可以看出，算法 5.2 比算法 5.1 更快地收敛于收敛值，算法 5.2 第 4 次迭代值即为 0.6931，算法 5.1 第 500 次迭代值才到 0.6921。由图 8.4 可以看出，算法 5.2 在迭代刚开始绝对误差就迅速趋近于 0，而算法 5.1 在迭代 200 次以后绝对误差才逐渐趋于 0。可见在收敛速率上，算法 5.2 极大地优于算法 5.1。

③复杂度比较：

在时间复杂度上，N=3,000、3,000,000、10,000,000 时，算法 5.2 用时均低于算法 5.1 用时，可见算法 5.2 有更低的时间复杂度。

在算法复杂度上，对于 N 此迭代，算法 5.1 使用了 2(N-1)次乘法和 2(N-1)次加法，

为 O(n)，算法 5.2 使用了 3(N-1)次乘法和 3(N-1)次加法，也为 O(n)，可见在算法复杂度上，两种算法相差不大，算法 5.2 只稍复杂于算法 5.1。

④数值稳定性比较：

对于算法 5.1，若初始值 s1 的绝对误差为 e(s1*)，则根据迭代公式，最终近似值的误差为 N*e(s1*)，与迭代次数呈线性关系，误差的积累效应可以通过调整初始值的精确度抵消。

对于算法 5.2，其与算法 5.1 相类似，误差积累不明显。

⑤结论

根据上述分析在收敛性、数值稳定性方面，两种算法均表现出较好的特性；在复杂度方面，算法 5.1 的算法复杂度稍低于算法 5.2，但算法 5.2 具有更低的时间复杂度；在收敛速率方面，算法 5.2 明显优于算法 5.1，前者能够在迭代运算之初即收敛于收敛值。综合而言，算法 5.2 优于算法 5.1。

（2）求I_n=$e^{-1}\int_0^1 x^n e^x dx$=1-n$I_{n-1}$，(n=1,2,...)：

①数值稳定性分析：

由图 8.9、8.10、8.11 可以看出，算法 5.3 分别在 n=8、10、13 处为负值，这与I_n>0 相矛盾，实际上，由积分估值得$\frac{e^{-1}}{n+1}$<I_n<$\frac{1}{n+1}$，因此算法 5.3 所估计的值得误差明显偏大，假设初始值I_0的误差为e_0，由推导公式可以求得误差$e_n = -ne_{n-1}$，因此有e_n=$(-1)^n n!\, e_0$，这说明I_n的误差是I_0的 n!倍，若 n=8，则误差将增大到 40,320 倍，n=10 时，误差将增大到 3,628,800 倍，这表明算法 5.3 是数值不稳定的。

相反的，对于算法 5.4，误差则将会减小到原来的$\frac{1}{n!}$倍，因此，即使初始值I_9、I_{10}、I_{13}给出的值误差范围很大，但随着计算的进行，误差将会越来越小，这体现了算法 5.4 极好的数值稳定性。

②结论：

从误差传递及算法数值稳定性的角度来考虑，算法 5.4 优于算法 5.3。

（3）求$P_n(x) = a_n x^n + a_{n-1} x^{n-1}$+...+$a_1 x$+$a_0$：

①复杂度比较：

在时间复杂度上，当 N=200,000、2,000,000、20,000,000 时，算法 5.6 用时均低于算法 5.5，体现了算法 5.6 具有较低的时间复杂度。

在算法复杂度上，对于 N 次迭代，算法 5.5 使用了 N+1 个加法和 2(N+1)个乘法，复杂度为 O(n)，算法 5.6 使用了 N 个加法和 N 个乘法，复杂度也为 O(n)，但略小于算法 5.5，折体现了算法 5.6（秦九韶算法）具有较低的算法复杂度。

②结论：

从复杂度上来说，算法 5.6 由于算法 5.5，同时由于计算步骤的减少，算法 5.6 对误差的传递有较好的控制作用。

十、 实验体会

本次实验是第一次数值分析实验，其目的主要在于熟悉数值分析的思路、方法与过程。

在本次实验中一共解决了三个问题，第一个题目是对比两种求近似值的方法，从中我们学习到了收敛性分析的方法、收敛速率的比较方法和复杂度比较的方法；第二个题目是对同一个递推关系式用两种不同的算法求解，我们发现正向的递推导致了误差的扩大，而反向的递推则使得误差不断地缩小，这提示我们在设计算法的时候可以改变递推的方向从而大幅的减小误差；第三个题目是用两种方法对同一个多项式进行幅值运算，一种是单纯的多项式叠加，一种是秦九韶算法，虽然采用了一定的方法尽可能的减小了前者的计算复杂度，但运算结果显示秦九韶算法不仅拥有较低的算法复杂度，而且拥有更低的时间复杂度。

这次数值分析实验，大大加深了我对数值分析的理解，使我对数值分析有了更加深刻的印象，同时也增加了我对数值分析的兴趣。并且从这三道小小的题目当中，我认识到了要写出一个好的算法还要注意很多问题，比如误差传递及数值稳定性。这对于我对算法的理解有着极大的帮助。

最后，感谢老师的谆谆教诲和帮助。