

# 第一次上机作业

Anonymity 3220100000\*

\* College of Control Science and Engineering, Robotics Engineering

July 8, 2024

## 1 问题描述

分别采用以下两种方法估计  $\ln 2$  和  $\ln 1.5$  的值, 计算 100 项的结果, 进行对比, 并分析原因。

已知:  $\ln 2 = 0.693147180559945 \dots$ ;  $\ln 1.5 = 0.40546510810816438 \dots$ 。

方法一:  $\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \frac{1}{5}x^5 + \dots + \frac{(-1)^{n+1}}{n}x^n + \dots$

方法二:  $\ln\left(\frac{1+x}{1-x}\right) = 2\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \dots + \frac{1}{2n-1}x^{2n-1} + \dots\right)$

如果要求计算结果具有 6 位有效数字, 请采用上述两种方法再进行计算, 并进行对比。

## 2 思路分析

首先, 我们可以看到两种方法都采用了级数展开的形式来估计自然对数的值。方法一是通过展开  $\ln(1+x)$  的级数, 而方法二是通过展开  $\ln\left(\frac{1+x}{1-x}\right)$  的级数。

对于方法一, 使用的级数展开式是  $\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \frac{1}{5}x^5 + \dots$ 。这个级数展开式在  $|x| \leq 1$  的范围内是收敛的。对于估计  $\ln(2)$  和  $\ln(1.5)$ , 取  $x = 1$  和  $x = 0.5$ 。对于  $x = 1$ , 在范围内, 级数应该是收敛的。但是对于  $x = 0.5$ , 由于 0.5 的绝对值小于 1, 所以级数也是收敛的。

对于方法二, 使用的级数展开式是  $\ln\left(\frac{1+x}{1-x}\right) = 2\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \dots\right)$ 。这个级数不收敛。对于估计  $\ln(2)$  和  $\ln(1.5)$ , 分别取  $x = 1/3$  和  $x = 1/5$ , 可能存在一些问题。

而对于要求“计算结果具有 6 位有效数字”有两种理解方式。一是后面级数的项小于  $0.5 \times 10^{-6}$ , 此时两次计算出来的结果相差/当前近似值小于  $0.5 \times 10^{-6}$ , 即变化的相对幅度小于  $0.5 \times 10^{-6}$ , 从而认为计算结果具有 6 位有效数字。而另一种理解是与题目中所给的标准值进行对比, 直至与标准值之差的绝对值小于  $0.5 \times 10^{-6}$ 。

## 3 运行结果与分析

### 3.1 前 100 项结果的计算

matlab 中运行结果:

与真实值对比, 计算相对误差分别为:

```
命令行窗口
方法一估计ln(2)的值:
0.688172179310195

方法一估计ln(1.5)的值:
0.405465108108164

方法二估计ln(2)的值:
0.693147180559945

方法二估计ln(1.5)的值:
0.405465108108164
```

对于方法一估计  $\ln(2)$  的值:

$$\text{Relative Error}_{\ln(2)} = \left| \frac{0.688172179310195 - 0.693147180559945}{0.693147180559945} \right| \times 100 \approx 0.7177\%$$

对于方法一估计  $\ln(1.5)$  的值:

$$\text{Relative Error}_{\ln(1.5)} = \left| \frac{0.405465108108164 - 0.40546510810816438}{0.40546510810816438} \right| \times 100 \approx 9.58 \times 10^{-16}$$

对于方法二估计  $\ln(2)$  的值:

$$\text{Relative Error}_{\ln(2)} = \left| \frac{0.693147180559945 - 0.693147180559945}{0.693147180559945} \right| \times 100 = 0\%$$

对于方法二估计  $\ln(1.5)$  的值:

$$\text{Relative Error}_{\ln(1.5)} = \left| \frac{0.405465108108164 - 0.40546510810816438}{0.40546510810816438} \right| \times 100 \approx 9.58 \times 10^{-16}$$

不难发现, 在计算到 100 项后, 方法二比方法一的结果要更精确。

由于两个级数展开都是关于  $x$  的幂级数, 因此在  $x = 0$  时, 无论展开到多少项, 最终得到的值永远与标准值相同。因此, 可以进一步得出结论, 在近似计算过程中, 所取得的  $x$  的值越接近 0, 最终计算得到的误差就越小。在方法一中, 计算  $\ln(2)$  和  $\ln(1.5)$  分别取  $x = 1$  和  $x = 0.5$ , 因此在计算  $\ln(1.5)$  时的相对误差更小。

### 3.2 计算 6 位有效数字

对于“6 位有效数字”的两种理解, 在思路分析中已经阐述。

下面是两种思路的运行结果:

对于与前一项对比计算, 得到的结果存在误差, 误差计算如下:

对于方法一估计  $\ln(2)$  的值:

$$\text{Relative Error}_{\ln(2)} = \left| \frac{0.693149 - 0.693147}{0.693147} \right| \times 100 \approx 0.002885\%$$

```

命令窗口
方法一计算ln(2)满足精度要求, n = 288539
方法一计算ln(1.5)满足精度要求, n = 16
方法二计算ln(2)满足精度要求, n = 6
方法二计算ln(1.5)满足精度要求, n = 5
方法一估计ln(2)的值:
0.693149
方法一估计ln(1.5)的值:
0.405465
方法二估计ln(2)的值:
0.693147
方法二估计ln(1.5)的值:
0.405465
fx >>

```

Figure 1: 与前一项对比结果

```

命令窗口
方法一估计ln(2)的值(前六位相同): 0.693147
对应的n值: 6
方法一估计ln(1.5)的值(前六位相同): 0.405465
对应的n值: 4
方法二估计ln(2)的值(前六位相同): 0.693147
对应的n值: 6
方法二估计ln(1.5)的值(前六位相同): 0.405465
对应的n值: 4
fx >>

```

Figure 2: 与标准值对比结果

对于方法一估计  $\ln(1.5)$  的值:

$$\text{Relative Error}_{\ln(1.5)} = \left| \frac{0.405465 - 0.405465}{0.405465} \right| \times 100 \approx 0\%$$

对于方法二估计  $\ln(2)$  的值:

$$\text{Relative Error}_{\ln(2)} = \left| \frac{0.693147 - 0.693147}{0.693147} \right| \times 100 = 0\%$$

对于方法二估计  $\ln(1.5)$  的值:

$$\text{Relative Error}_{\ln(1.5)} = \left| \frac{0.405465 - 0.405465}{0.405465} \right| \times 100 = 0\%$$

因此从误差角度分析, 计算到 6 位有效数字时方法二的相对误差要更小更精确, 同时我们对比迭代次数  $n$  可以发现方法二的迭代次数也更少。

而对于另一种理解方式, 即将得到的值与标准值进行对比从而得到六位有效数字, 显而易见的误差为 0, 这时候我们就要对比迭代次数。与上一种理解相比, 方法二的迭代次数没有变, 当时方法一, 尤其是在计算  $\ln(2)$  的时候计算次数  $n$  有显著的减少。由此我们可以判断, 方法一并不是要迭代数万次后才能接近真实值, 而是一直在真实值附近振荡。

### 3.3 小结

同时, 用 matlab 绘制出结果值随迭代次数  $n$  前 100 次的变化图像如下: (对于每个  $n$  对应的估计值应为离散的点云图, 但为了观测方便绘制成折线图)。

对于方法一的展开式, 由于相邻的级数项为正负交替, 因此得到的结果是振荡的。这种算法能够更加快速的接近最终的准确值, 但是在  $n$  足够大后, 由于振荡的扰动, 反而会导致相对误差较大。

而对于方法二的展开式, 每一项都为正, 因此最终得到估计值为一个单调递增函数。这样计算的结果在前期靠近最终准确值的速度较慢, 但是当迭代足够多次后, 相较于方法一得到的值更加精确。

综上所述, 如果要追求速度, 更快的得到一个较为粗略的估计值, 应该采取“振荡”的泰勒展开方式, 即类似于方法一的展开形式; 而若要准确的计算出真实值, 则应该采用“单调”的泰勒展开方式, 即类似于方法二的展开形式。

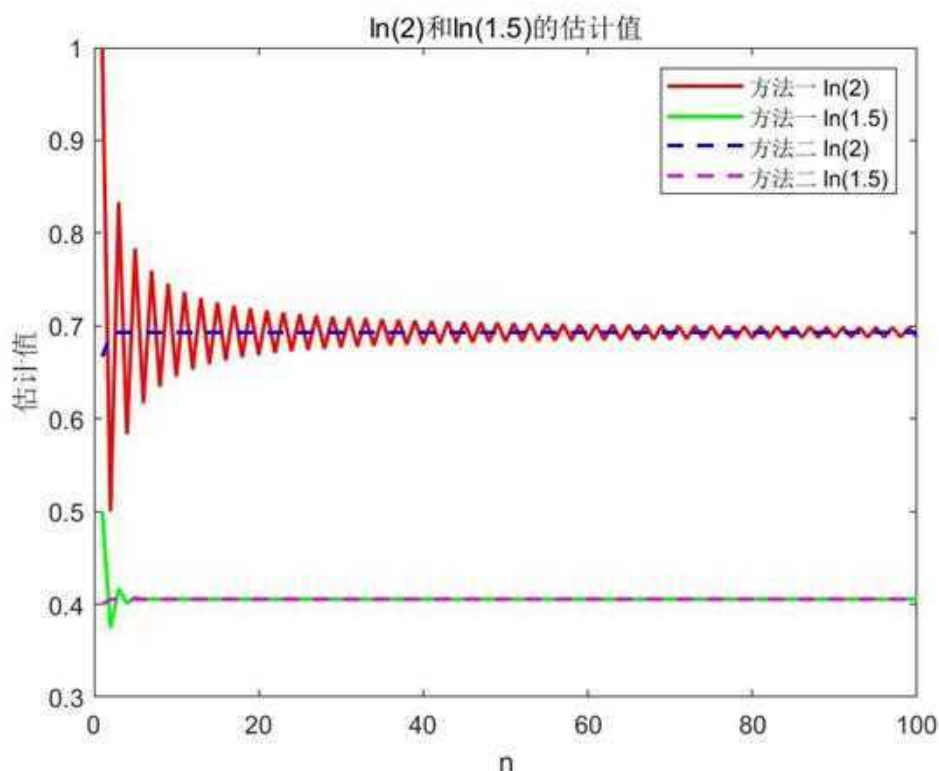


Figure 3: the result of sample in PTA

## 4 实验心得

本次实验是数值计算方法的第一份实验报告。整体的编程思路很简单也很容易上手，由于多个问题之间的相似性，代码没有进行大幅度的改动，因此可能对于某些问题的算法不是最优的。

然而在用第一种理解计算六位有效数字的时候，迭代次数  $n$  始终无法输出，最后才发现竟然要循环上万次才能够达到最终的要求。当然，这与展开级数的振荡有着相当大的关系。

同时本次实验采用 matlab 进行，充分地利用了其强大的绘图和可视化工具，更直观地展示和分析数据。

## 5 源代码

### 5.1 问题 1 运行代码

```

1 % 方法一
2 x = 1;
3 result_method1_ln2 = 0; % 初始化方法一计算ln(2)的结果变量
4 result_method1_ln1_5 = 0; % 初始化方法一计算ln(1.5)的结果变量
5
6 for n = 1:100
7     x_2=1;
8     term = ((-1)^(n+1) / n) * x_2^n; % 计算每一项的值

```

```

9      result_method1_ln2 = result_method1_ln2 + term; % 累加到ln(2)的结果
10  end
11
12  for n = 1:100
13      x_1_5=0.5;
14      term_ln1_5 = ((-1)^(n+1) / n) * x_1_5^n; % 计算每一项的值
15      result_method1_ln1_5 = result_method1_ln1_5 + term_ln1_5; % 累加
        到ln(1.5)的结果
16  end
17
18  % 方法二
19  result_method2_ln2 = 0; % 初始化方法二计算ln(2)的结果变量
20  result_method2_ln1_5 = 0; % 初始化方法二计算ln(1.5)的结果变量
21
22  for n = 1:100
23      x_2_method=1/3;
24      term = (1 / (2*n - 1)) * x_2_method^(2*n - 1); % 计算每一项的值
25      result_method2_ln2 = result_method2_ln2 + term; % 累加到ln(2)的结果
        果
26  end
27
28  for n = 1:100
29
30      x_1_5_method=1/5;
31      term_ln1_5 = (1 / (2*n - 1)) * x_1_5_method^(2*n - 1); % 计算每一
        项的值
32      result_method2_ln1_5 = result_method2_ln1_5 + term_ln1_5; % 累加
        到ln(1.5)的结果
33  end
34
35  % 输出结果
36  disp('方法一估计ln(2)的值: ');
37  disp(result_method1_ln2);
38  disp('方法一估计ln(1.5)的值: ');
39  disp(result_method1_ln1_5);
40
41  disp('方法二估计ln(2)的值: ');
42  disp(result_method2_ln2 * 2);
43  disp('方法二估计ln(1.5)的值: ');
44  disp(result_method2_ln1_5 * 2);

```

## 5.2 问题 2 思路 1 运行代码

```
1 % 方法一
2 x = 1;
3 result_method1_ln2 = 0; % 初始化方法一计算ln(2)的结果变量
4 result_method1_ln1_5 = 0; % 初始化方法一计算ln(1.5)的结果变量
5
6 tolerance = 0.5 * 10e-6; % 允许的误差范围
7
8 for n = 1:10e5
9     x_2 = 1;
10    term = ((-1)^(n+1) / n) * x_2^n; % 计算每一项的值
11    result_method1_ln2 = result_method1_ln2 + term; % 累加到ln(2)的结果
12
13    if abs(term/result_method1_ln2) < tolerance
14        disp(['方法一计算ln(2)满足精度要求, n = ', num2str(n)]);
15        break; % 如果满足精度要求, 提前结束循环
16    end
17 end
18
19 for n = 1:100
20    x_1_5 = 0.5;
21    term_ln1_5 = ((-1)^(n+1) / n) * x_1_5^n; % 计算每一项的值
22    result_method1_ln1_5 = result_method1_ln1_5 + term_ln1_5; % 累加
23    到ln(1.5)的结果
24
25    if abs(term_ln1_5/result_method1_ln1_5) < tolerance
26        disp(['方法一计算ln(1.5)满足精度要求, n = ', num2str(n)]);
27        break; % 如果满足精度要求, 提前结束循环
28    end
29 end
30 % 方法二
31 result_method2_ln2 = 0; % 初始化方法二计算ln(2)的结果变量
32 result_method2_ln1_5 = 0; % 初始化方法二计算ln(1.5)的结果变量
33
34 for n = 1:10
35    x_2_method = 1/3;
36    term = (1 / (2*n - 1)) * x_2_method^(2*n - 1); % 计算每一项的值
37    result_method2_ln2 = result_method2_ln2 + term; % 累加到ln(2)的结果
```

```

38
39     if abs(term/result_method2_ln2) < tolerance
40         disp(['方法二计算ln(2)满足精度要求, n = ', num2str(n)]);
41         break; % 如果满足精度要求, 提前结束循环
42     end
43 end
44
45 for n = 1:10
46     x_1_5_method = 1/5;
47     term_ln1_5 = (1 / (2*n - 1)) * x_1_5_method^(2*n - 1); % 计算每一
        项的值
48     result_method2_ln1_5 = result_method2_ln1_5 + term_ln1_5; % 累加
        到ln(1.5)的结果
49
50     if abs(term_ln1_5/result_method2_ln1_5) < tolerance
51         disp(['方法二计算ln(1.5)满足精度要求, n = ', num2str(n)]);
52         break; % 如果满足精度要求, 提前结束循环
53     end
54 end
55
56 % 四舍五入, 保留六位有效数字
57 result_method1_ln2_rounded = round(result_method1_ln2, 6);
58 result_method1_ln1_5_rounded = round(result_method1_ln1_5, 6);
59 result_method2_ln2_rounded = round(result_method2_ln2 * 2, 6);
60 result_method2_ln1_5_rounded = round(result_method2_ln1_5 * 2, 6);
61
62 % 输出结果, 保留六位有效数字
63 disp('方法一估计ln(2)的值: ');
64 fprintf('%0.6f\n', result_method1_ln2_rounded);
65 disp('方法一估计ln(1.5)的值: ');
66 fprintf('%0.6f\n', result_method1_ln1_5_rounded);
67
68 disp('方法二估计ln(2)的值: ');
69 fprintf('%0.6f\n', result_method2_ln2_rounded);
70 disp('方法二估计ln(1.5)的值: ');
71 fprintf('%0.6f\n', result_method2_ln1_5_rounded);

```

### 5.3 问题 2 思路 2 运行代码

```

1 % 方法一
2 x = 1;

```

```

3 result_method1_ln2 = 0; % 初始化方法一计算ln(2)的结果变量
4 result_method1_ln1_5 = 0; % 初始化方法一计算ln(1.5)的结果变量
5
6 target_ln2 = 0.6931471; % 目标ln(2)的前六位
7 target_ln1_5 = 0.4054651; % 目标ln(1.5)的前六位
8
9 n_ln2 = 0; % 初始化n值
10 n_ln1_5 = 0; % 初始化n值
11
12 for n = 1:10e6
13     x_2=1;
14     term = ((-1)^(n+1) / n) * x_2^n; % 计算每一项的值
15     result_method1_ln2 = result_method1_ln2 + term; % 累加到ln(2)的结果
16
17     % 检查是否满足六位有效数字的要求
18     if abs(result_method1_ln2 - target_ln2) < 0.5*1e-6
19         n_ln2 = n;
20         break;
21     end
22 end
23
24 for n = 1:10e6
25     x_1_5=0.5;
26     term_ln1_5 = ((-1)^(n+1) / n) * x_1_5^n; % 计算每一项的值
27     result_method1_ln1_5 = result_method1_ln1_5 + term_ln1_5; % 累加
28     % 检查是否满足六位有效数字的要求
29     if abs(result_method1_ln1_5 - target_ln1_5) < 0.5*1e-6
30         n_ln1_5 = n;
31         break;
32     end
33 end
34
35
36 % 方法二
37 result_method2_ln2 = 0; % 初始化方法二计算ln(2)的结果变量
38 result_method2_ln1_5 = 0; % 初始化方法二计算ln(1.5)的结果变量
39
40 for n = 1:10e6
41     x_2_method=1/3;

```



```

42     term = (1 / (2*n - 1)) * x_2_method^(2*n - 1); % 计算每一项的值
43     result_method2_ln2 = result_method2_ln2 + term; % 累加到ln(2)的结果
44
45     % 检查是否满足六位有效数字的要求
46     if abs(result_method2_ln2 * 2 - target_ln2) < 0.5*1e-6
47         n_ln2 = n;
48         break;
49     end
50 end
51
52 for n = 1:10e6
53     x_1_5_method=1/5;
54     term_ln1_5 = (1 / (2*n - 1)) * x_1_5_method^(2*n - 1); % 计算每一项的值
55     result_method2_ln1_5 = result_method2_ln1_5 + term_ln1_5; % 累加到ln(1.5)的结果
56
57     % 检查是否满足六位有效数字的要求
58     if abs(result_method2_ln1_5 * 2 - target_ln1_5) < 0.5*1e-6
59         n_ln1_5 = n;
60         break;
61     end
62 end
63
64 % 输出结果（保留六位有效数字）
65 disp(['方法一估计ln(2)的值（前六位相同）：', num2str(
66     result_method1_ln2, '%.6f')]);
67 disp(['对应的n值：', num2str(n_ln2)]);
68 disp(['方法一估计ln(1.5)的值（前六位相同）：', num2str(
69     result_method1_ln1_5, '%.6f')]);
70 disp(['对应的n值：', num2str(n_ln1_5)]);
71
72 disp(['方法二估计ln(2)的值（前六位相同）：', num2str(
73     result_method2_ln2 * 2, '%.6f')]);
74 disp(['对应的n值：', num2str(n_ln2)]);
75 disp(['方法二估计ln(1.5)的值（前六位相同）：', num2str(
76     result_method2_ln1_5 * 2, '%.6f')]);
77 disp(['对应的n值：', num2str(n_ln1_5)]);

```

## 5.4 绘制图像运行代码

```
1 % 方法一计算ln(2)和ln(1.5)的结果
2 x = 1;
3 result_method1_ln2 = zeros(1, 100);
4 result_method1_ln1_5 = zeros(1, 100);
5
6 for n = 1:100
7     x_2=1;
8     term = ((-1)^(n+1) / n) * x_2^n;
9     if n == 1
10         result_method1_ln2(n) = term;
11     else
12         result_method1_ln2(n) = result_method1_ln2(n-1) + term;
13     end
14
15     x_1_5=0.5;
16     term_ln1_5 = ((-1)^(n+1) / n) * x_1_5^n;
17     if n == 1
18         result_method1_ln1_5(n) = term_ln1_5;
19     else
20         result_method1_ln1_5(n) = result_method1_ln1_5(n-1) +
            term_ln1_5;
21     end
22 end
23
24 % 方法二计算ln(2)和ln(1.5)的结果
25 result_method2_ln2 = zeros(1, 100);
26 result_method2_ln1_5 = zeros(1, 100);
27
28 for n = 1:100
29     x_2_method=1/3;
30     term = (1 / (2*n - 1)) * x_2_method^(2*n - 1);
31     if n == 1
32         result_method2_ln2(n) = term;
33     else
34         result_method2_ln2(n) = result_method2_ln2(n-1) + term;
35     end
36
37     x_1_5_method=1/5;
38     term_ln1_5 = (1 / (2*n - 1)) * x_1_5_method^(2*n - 1);
39     if n == 1
```

```

40         result_method2_ln1_5(n) = term_ln1_5;
41     else
42         result_method2_ln1_5(n) = result_method2_ln1_5(n-1) +
            term_ln1_5;
43     end
44 end
45
46 % 绘制折线图
47 figure;
48
49 plot(1:100, result_method1_ln2, 'r', 'LineWidth', 1.5, 'DisplayName',
    '方法一 ln(2)');
50 hold on;
51 plot(1:100, result_method1_ln1_5, 'g', 'LineWidth', 1.5, 'DisplayName',
    '方法一 ln(1.5)');
52 plot(1:100, result_method2_ln2 * 2, 'b--', 'LineWidth', 1.5, '
    DisplayName', '方法二 ln(2)');
53 plot(1:100, result_method2_ln1_5 * 2, 'm--', 'LineWidth', 1.5, '
    DisplayName', '方法二 ln(1.5)');
54
55 title('ln(2)和ln(1.5)的估计值');
56 xlabel('n');
57 ylabel('估计值');
58 legend('show');
59
60 hold off;

```