# 数值分析实验报告

张钰晖
2015011372, yuhui-zh15@mails.tsinghua.edu.cn, 185-3888-2881

2017 年 6 月 15 日

## 1 题目描述

1-3 编程观察无穷级数

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

的求和计算.

**(1)** 采用 IEEE 单精度浮点数，观察当 n 为何值时求和结果不再变化，将它与理论分析的结论进行比较.

**(2)** 用 IEEE 双精度浮点数计算 (1) 中前 n 项的和，评估 IEEE 单精度浮点数计算结果的误差.

**(3)** 如果采用 IEEE 双精度浮点数，估计当 n 为何值时求和结果不再变化，这在当前做实验的计算机上大概需要多长的计算时间？

## 2 解题思路

无穷级数 $\sum_{n=1}^{\infty} \frac{1}{n}$ 的精确值是发散的，但是由于浮点运算系统中会发生 (1) 上溢 ( 部分和非常大，达到 OFL ) 或 (2) 下溢 ( 当两个加数的比值小于机器精度的一半 )，计算结果是不准确的。

在本题中，会发生下溢，因此计算结果将在一定的 n 时收敛。

为此，我们可以逐步计算，当两步的计算结果不再变化时，退出循环，输出结果。

本题求和所需时间复杂度为 O(n)。

# 3 实验结果

算法运行的结果如下：

**(1)** SINGLE: n = 2097152, result = 15.403683

采用 IEEE 单精度浮点数计算，当 $n = 2097152$ 时求和结果不再变化。理论计算表明由于 $\sum_{k=1}^{n} \frac{1}{k} \approx \ln n$，可用 $\frac{1}{n} \leq \frac{1}{2}\epsilon_{mach} \ln(n-1)$，对于单精度，取 $\epsilon_{mach} = 2^{-24}$，求解可得 $n \approx 2.3 * 10^6$ 时求和结果不再变化，与实验结果近似相等。误差的来源可能包括舍入误差，以及无穷级数求和估算值的误差。

**(2)** DOUBLE: n = 2097152, result = 15.133307, delta = 0.270376

采用 IEEE 双精度浮点数计算，当 $n = 2097152$ 时求和结果为 15.133307，单精度浮点数计算结果与之相比，绝对误差为 0.270376，相对误差为 1.7866%。

**(3)** 理论计算表明，对于双精度，取 $\epsilon_{mach} = 2^{-53}$，求解可得 $n \approx 5.3 * 10^{14}$ 时求和结果不再变化。

用双精度浮点数计算级数前 $10^9$ 项的和，所用总时间为 $8.2s$。可估计 $n = 5.3 * 10^{14}$ 时，求和时间约为 $\frac{5.3*10^{14}}{10^9} * 8.2s = 4.3 * 10^6(s) \approx 50d$。

# 4 实验心得

通过这次试验，我重新温习了计算机浮点数系统，并通过实验加强了对于计算机浮点数系统的理解。

首先，通过实验中求和级数无法一直变化的事实，我在此体会到了浮点数系统“有限而不连续”的特点，即会出现“大数吃掉小数”的现象，在平日的数值试验中，我们要避免或者评估这类误差。

其次，笔者体会到了双精度浮点数比单精度浮点数在数值计算上巨大的精度优势，当我们需要高精度计算的时候，应该首先选择使用双精度浮点数。

最后，实验中，double 的求和在可观测的时间内实际上并无法结束，这说明 double 类的计算机浮点数在应对大多数数值计算工作的时候，是十分精确的。

另外做实验之前，应该估算实验所需时间，可以看出双精度累加实验估算时间非常长，很难通过实验方法进行验证。

# 5 源代码

```
% nowSum和preSum分别记录当前结果和上一步结果，变量相等时，停止迭代
nowSum1 = single(0);
preSum1 = single(-1);
nowSum2 = double(0);
preSum2 = double(-1);
% realSum记录用双精度浮点数结果
realSum = double(0);
% cnt记录迭代次数n
cnt1 = 0;
cnt2 = 0;

% 当上一步结果与当前结果不相等时，累加1/cnt
while (preSum1 ~= nowSum1)
   preSum1 = nowSum1;
   cnt1 = cnt1 + 1;
   nowSum1 = nowSum1 + single(1.0 / cnt1);
   realSum = realSum + 1.0 / cnt1;
end

fprintf('SINGLE: n = %d, result = %f\n', cnt1, nowSum1);
fprintf('DOUBLE: n = %d, result = %f\n', cnt1, realSum);
fprintf('DELTA = %f\n', nowSum1 - realSum);

% 当上一步结果与当前结果不相等时，累加1/cnt，实际很长时间迭代才会停止
% while (preSum2 ~= nowSum2)
% preSum2 = nowSum2;
% cnt2 = cnt2 + 1;
% nowSum2 = nowSum2 + double(1.0 / cnt2);
% end

%fprintf('DOUBLE: n = %d, result = %f\n', cnt2, nowSum2);
```

# 参考文献

[1] 数值分析与算法（第二版）, 喻文健, 2015.