浮点数误差实验报告

冯卓尔 计86 201701998

实验要求

实验题目

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，将它和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?

解题思路

由于Matlab的实现是使用numpy的库，而python的底层依赖于C++实现，故此我直接使用具有相同IEEE精度标准的C++求解此题。

（1）单精度浮点数条件下，结果不再变化的时候应当发生了"大数吃小数"的情况，因而记S为当前和，不再变化的时候有

$$\frac{\frac{1}{n}}{S}\leq\frac{1}{2}\epsilon\star,\quad \epsilon\star\, is\ machine\ exactitude$$

因而， 估算出n大约在$200,000$左右，直接暴力枚举可以找到$n=2097152$。具体输出结果在附件out2.txt中。

（2）使用double重复上述操作，在 $n = 2097152$ 时，float（单精度）的和为 15.403682708740234375，而double（双精度）的和为 15.133306695078193371，其差为（保留6位有效数字）-0.270376。输出结果在附件out3.txt中。

（3）先粗略估计n的大小

由定理1.6，大数吃小数发生的情况下

$$\frac{\frac{1}{n}}{S} \leq \frac{1}{2}\epsilon\star, \quad \epsilon\star \, is \; machine \; exactitude$$

$$lim_{n\to\infty}\sum_{n=1}^{\infty}\frac{1}{n} = ln(n)$$

IEEE双精度浮点数的误差为 1.1×10^{-16}，那么估算n在 $10^{14} \sim 10^{15}$ 数量级之间。

进行时间最短的保守估计：

记 $n = 10^{14}$，按照本台计算机一秒计算次数 10^9 来算，则需要 10^5 秒的时间，即6.94年的时间才能够完成计算。在这里，我使用的假设仅仅是计算机完成一次操作而不是一次运算的时间，即一秒中计算机完成的操作数。那么实际上的时间要比这个时间更加长一些。

实验结论

我们发现，单精度浮点数虽然能够满足我们一些运算需求，但是在数值运算的精确度上远远不如双精度浮点数。进行大型数值运算的时候首先应该使用精度更高的双精度浮点数。

单精度对于无穷级数求和只能支持到 10^6 数量级，双精度浮点数能够支持到 10^{14} 数量级，是前者的平方以上。双精度浮点数的精度直观的更高。

代码

```
#include <bits/stdc++.h>
using namespace std;

float series1(int n) {
    float sum = 0;
    for (int i = 1; i <= n; ++i) {
        sum += (float) (1.0 / i);
    }
    return sum;
```

```
}

double series2(long long n) {
    double sum = 0;
    for (long long i = 1; i <= n; ++i) {
        sum += (double) (1.0 / i);
    }
    return sum;
}

int main() {
    for (int i = 2097100; i < 2097300; i++) {
        printf("single: input %d, output... %.20f\n", i, series1(i));
    }

    for (long long i = 2097100; i < 2097300; i++) {
        printf("double: input %lld, output... %.20lf\n", i,
series2(i));
    }

    cout << series2(2097152) - series1(2097152) << endl;

    return 0;
}
```

输出结果

见附件out2.txt（单精度结果）out3.txt（双精度结果）