数值分析第一周上机

By 211870125 陈睿硕

2023.2.27

§ 1 问题

求计算机的规范化浮点数的上溢值(OFL)、下溢值(UFL)和计算机的机器精度(ε_{mach})。

§ 2 算法思路

问题即求解$-L$（即单精度数最小指数）、U（即单精度数最大指数）、t（即单精度数精度）。这是因为利用公式：

$$OFL = 2^U(2 - 2^{-(t-1)}), \tag{1}$$

$$UFL = 2^{-L}, \tag{2}$$

$$\varepsilon_{mach} = 2^{-t}, \tag{3}$$

可以求得问题之解。

下面我们将分三步求解上溢值，下溢值和机器精度。

I 求解$t + L$

我们将$a = 1$不断进行"$\div 2$"操作，直到其变为0，记录下操作的次数$count$。

在a变为0的前一刻，a的数值应该是单精度浮点数的最小正值，即$2^{-L-(t-1)}$。这是因为非规格数的指数始终默认为$-L$。从$a = 1$到$a = 2^{-L-(t-1)}$共进行了$L + t - 1$步，故最后当a恰变为0时，共进行了$t + L$步。则有:

$$t + L = count = 150, \tag{4}$$

II 求解t

我们令$c = 3, d = 1$,然后不断对两者进行“$\times 2 + 1$”操作，直到式 $c - 1 = 2b$ 不成立，记录下操作的次数，其即为$t - 1$。

这是因为规格数的有效数字不能超过t位，一旦超过t位便会产生舍入误差。易见每次操作都将c、d的有效位数增多一位，当式 $c - 1 = 2b$ 第一次不成立时，c的有效数字恰超过t位，d的有效数字恰为t位，故实际上有：

$$d = (\underbrace{11 \cdots\cdots 1}_{t\text{个}})_2 = 2^t - 1, \tag{5}$$

此时也正好进行了$t - 1$次操作，如此可求得$t = 24$。

III 求解上溢值

结合(5)改写(1)为：

$$OFL = 2^{U-t+1}(2^t - 1) = 2^{U-t+1}d$$

故类似于II中的方法，我们不断将d进行“$\times 2$”操作，直至发生舍入。记录下舍入前一刻的d的值，即为上溢值OFL。求得上溢值$OFL \approx 3.40282 \times 10^{38}$。

IV 求解下溢值及机器精度

由$t=24$及(4)知$L=150-t=126$,利用(2)计算得$UFL\approx 1.17549\times 10^{-38}$。

由(3)计算得$\varepsilon_{mach}\approx 5.96046\times 10^{-8}$

§ 3 结果分析

这个方法是通过找出t、U、L的值利用既知公式计算上溢值、下溢值及机器精度的，故和理论分析的上述三值完全相等。利用C++的char指针打印出所算出的三值的单精度表示，也和我们理论分析的形式无异，故正确性是肯定的。

§ 4 附录：程序代码

```cpp
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    float a=1,b=1;
    int t_plus_L=0;
    while(a!=0)
    {
        b=a;
        a/=2;
        t_plus_L++;
    }
    float c=3,d=1,t=1;
    while(c-1==2*d)
    {
        d=2*d+1;
        c=2*c+1;
        t++;
    }
    float u=2*d;
    while(u/2==d)
    {
        u*=2;d*=2;
    }
    cout<<"Upflow value:"<<d<<endl<<"Underflow value:"
    <<pow(2,t-t_plus_L)<<endl<<"Machine precision:"<<pow(2,-t)<<endl;
    return 0;
}
```