



---

# High-Performance Computing

---

Jiarui XIE

Semester: 2018/2019

Supervisor: David Hill

## I. INTRODUCTION

Beware of floating point arithmetics by doing this lab.

## II. FLOAT AND DOUBLE

The standard way to show the double is %.18f. If we use for example %.15 instead of %.18 we will get the result with lost precision.

If we are using C++ to display double, we would better use setprecision(18) to set precision ahead.

## III. KAHAN SUM

We used the following code to realize the KANHAN SUM and test with the code following. The result like Figure1, obviously it is better than the normal method. Why we use float is because it is easy to test the lost precision.

```
double KahanSum(float *input, int size)
{
    double sum = 0.0;
    double c = 0.0;
    for (int i = 0; i < size; i++)
    {
        double y = input[i] - c;
        double t = sum + y;
        c = (t - sum) - y;
        sum = t;
    }
    return sum;
}

int main()
{
    float a = 123456;
    float b = 2.123456;
    float aaaa[] = {123456, 2.123456};
    printf("the sum is %.18f\n", KahanSum(aaaa, 2));
    printf("%.18f\n", a+b);
    return 0;
}
```

```
Result of KahanSum      : 123458.123456001280000000
Result without KahanSum:123458.125000000000000000
```

Figure1. Contraction with Kahan method

## IV. PTHREAD WITH PI

During the simulation only one time I get the difference between Kahan method and normal Method the result like Figure2.

```
QueFir: 1000000, 3.127564
线程ID=2 finish calculating
QueFir: 1000000, 3.129796
线程ID=3 finish calculating
QueFir: 1000000, 3.129264
线程ID=6 finish calculating
QueFir: 1000000, 3.130824
线程ID=1 finish calculating
QueFir: 1000000, 3.128208
线程ID=7 finish calculating
QueFir: 1000000, 3.128616
线程ID=4 finish calculating
QueFir: 1000000, 3.129492
线程ID=5 finish calculating
QueFir: 1000000, 3.122504
线程ID=9 finish calculating
QueFir: 1000000, 3.126776
线程ID=8 finish calculating
QueFir: 1000000, 3.125024
线程ID=10 finish calculating
AveNormal = 3.127806799999999700
AveKahan = 3.127806800000000100
```

Figure2. Calculate pi with thread