实验报告1

姓名：刘家骥

学号：PB20071417

1. 第一题:
   1. 计算结果:

|  |  |  |
| --- | --- | --- |
| *x* |  |  |
| 2.500000000000e-01 | 1.039864495397e-02 | 1.039864495397e-02 |
| 6.250000000000e-02 | 6.509710219689e-04 | 6.509710219689e-04 |
| 1.562500000000e-02 | 4.068982889294e-05 | 4.068982889294e-05 |
| 3.906250000000e-03 | 2.543130449340e-06 | 2.543130449340e-06 |
| 9.765625000000e-04 | 1.589457099271e-07 | 1.589457099271e-07 |
| 2.441406250000e-04 | 0.000000000000e+00 | 9.934107758625e-09 |
| 6.103515625000e-05 | 0.000000000000e+00 | 6.208817349140e-10 |
| 1.525878906250e-05 | 0.000000000000e+00 | 3.880510843213e-11 |
| 3.814697265625e-06 | 0.000000000000e+00 | 2.425319277008e-12 |
| 9.536743164062e-07 | 0.000000000000e+00 | 1.515824548130e-13 |
| 2.384185791016e-07 | 0.000000000000e+00 | 9.473903425812e-15 |

* 1. 结果分析：

显而易见，两式都是对进行计算，但是当x的值达到，结果出现了偏差。这主要是因为算法的不同而引起的。

第一种算法，当x很小的时候，就是两个相近的数相减，而这会增大相对误差。而第二种算法，避免了两个相近的数相减，并且避免了小数作除数，从而减小了绝对误差和相对误差。

1. 第二题：

2.1 计算结果:

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 方法（a） | 方法（b） | 方法（c） |
| 结果 | 1.02518814e-10 | -1.56433089e-10 | 0.00000000e+00 |

2.2 结果分析：

我认为方法（a）结果相对更加精确。

这里要考虑到双精度浮点数在寄存器中的保存机制：它保存的尾数部分使用52位存储，因此双精度数的有效位数是16位。而这个有效位数是同时包含了整数部分和小数部分的。因此当数（绝对值）很大时，能保存的小数位数就会减少，从而造成误差。而方法（c）正数和负数分别相加，两个数绝对值都很大。因此方法（c）并不精确。

同样，对于方法（b）（逆序求和），一开始就是一个很大的数和很小的数相加，因此也会造成前面所说的误差。

因此相对而言方法（a）结果更加精确。

1. 实验结论：

计算同一个算式，不同的算法会导致不同大小的误差。此外，选用不同精度的浮点数进行运算和输出，也会影响到最终的精确度。