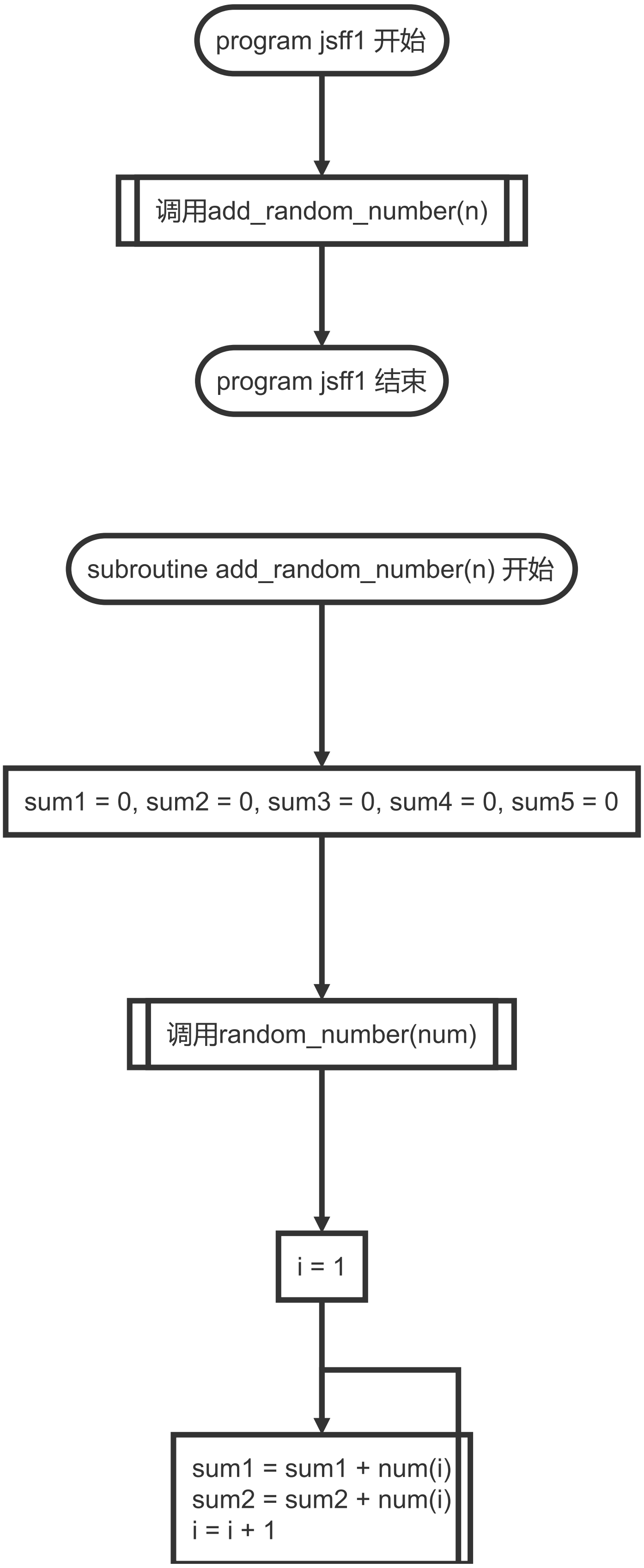
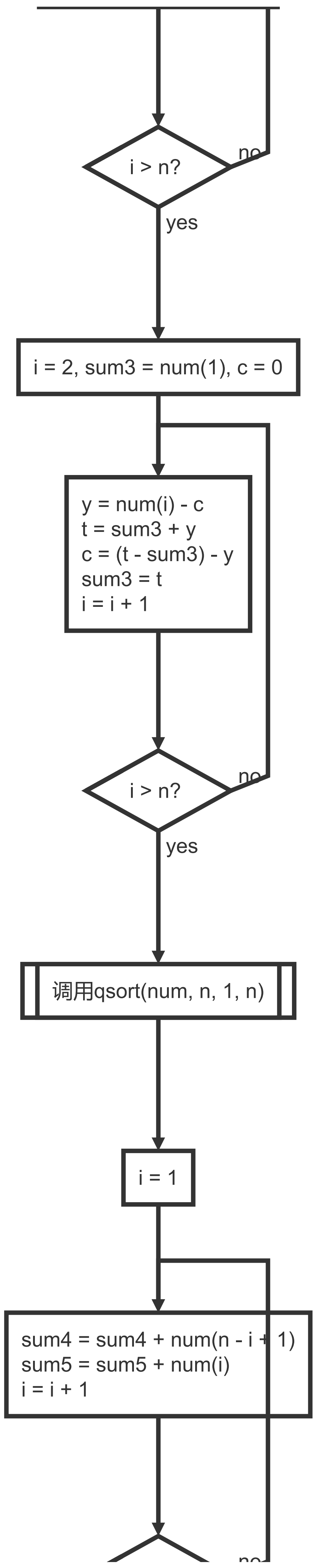
计算方法上机实习一 实习报告

2019级 大气科学学院 赵志宇

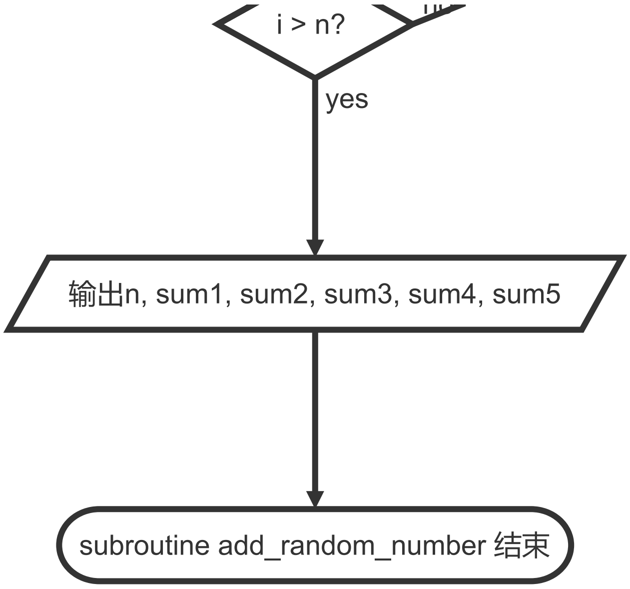
学号：191830227

一、编程流程图





i > n?
no
yes
i = 2, sum3 = num(1), c = 0
y = num(i) - c
t = sum3 + y
c = (t - sum3) - y
sum3 = t
i = i + 1
i > n?
no
yes
调用qsort(num, n, 1, n)
i = 1
sum4 = sum4 + num(n - i + 1)
sum5 = sum5 + num(i)
i = i + 1
no



二、源代码

```fortran
Program jsff1
    ! homework1 of Numerical Methods
    ! arthor : zzy
    implicit none

    call add_random_numbers(10)
    call add_random_numbers(int(1e4))
    call add_random_numbers(int(1e7))

end program jsff1

subroutine add_random_numbers(n)
    ! generate random numbers and calculate summation in 5 different methods
    ! parameters: n, the length of array
    ! author: zzy
    implicit none
    ! n is the length of array num
    integer, intent(in) :: n
    real(8), dimension(1:n) :: num
    ! sum1, sum2, sum3, sum4, sum5 correspond with method a), b), c), d), e)
    real(8) :: sum1 = 0.0
    real(8) :: sum3
    real(4) :: sum2 = 0.0
    real(4) :: sum4 = 0.0
    real(4) :: sum5 = 0.0
    ! c, y, t is used in method c)
    real(8) :: c = 0.0, y, t
    ! i is a loop varible
    integer :: i

    ! generate random numbers
    call random_number(num)

    ! method a) and b)
    do i = 1, n
```

```
        ! method a) : ditrectly add
        sum1 = sum1 + num(i)
        ! method b) : use single precision
        sum2 = sum2 + sngl(num(i))
    end do

    ! method c) : use auxiliary varibles c, y, t
    sum3 = num(1)
    do i = 2, n
        y = num(i) - c
        t = sum3 + y
        c = (t - sum3) - y
        sum3 = t
    end do

    ! quick sort algorithm, let the numbers be sorted in ascending order
    call qsort(num, n, 1, n)

    ! method d) and e)
    do i = 1, n
        ! method d) : use single precision, add in descending order
        sum4 = sum4 + sngl(num(n - i + 1))
        ! method e) : use single precision, add in ascending order
        sum5 = sum5 + sngl(num(i))
    end do

    ! print the results
    print *, "n = ", n
    print *, "sum1 = ", sum1
    print *, "sum2 = ", sum2
    print *, "sum3 = ", sum3
    print *, "sum4 = ", sum4
    print *, "sum5 = ", sum5

end subroutine add_random_numbers

recursive subroutine qsort(a,n,l,r)
    ! quick sort algorithm, let the array be sorted in ascending order
    ! parameters :
    ! a, the array to be sorted
    ! n, the length of the array
    ! l, left boundary of the interval to be sorted
    ! r, right boundary of the interval to be sorted
    ! author : zzy
    implicit none
    integer, intent(in) :: n
    real(8), intent(in out), dimension(l:r) :: a
    integer, intent(in) :: l
    integer, intent(in) :: r
    integer :: i, j
    real(8) :: val, temp

    i = l
    j = r
    val = a((l + r) / 2)

    do while(i <= j)
        do while(a(i) < val)
```

```
            i = i + 1
        end do

        do while(a(j) > val)
            j = j - 1
        end do

        if(i <= j) then
            temp = a(i)
            a(i) = a(j)
            a(j) = temp
            i = i + 1
            j = j - 1
        end if

    end do

    if (l < j) call qsort(a(l:j), n, l, j)
    if (i < r) call qsort(a(i:r), n, i, r)

end subroutine qsort
```

三、运行结果

不失一般性，每次运行程序分别取$n=10, n=10^4, n=10^7$并将将程序重复运行10次，结果如下:

第1、2次:

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    5.7653302764076262
 sum2 =    5.76533079
 sum3 =    5.7653302764076262
 sum4 =    5.76533079
 sum5 =    5.76533031
 n =        10000
 sum1 =    5000.7198167264914
 sum2 =    5000.72607
 sum3 =    4994.9544864500940
 sum4 =    5000.72510
 sum5 =    5000.71924
 n =     10000000
 sum1 =    5004938.1204222292
 sum2 =    5004601.50
 sum3 =    4999937.4006057447
 sum4 =    5004236.00
 sum5 =    5064274.50
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    4.4623620779453930
 sum2 =    4.46236229
 sum3 =    4.4623620779453930
 sum4 =    4.46236229
 sum5 =    4.46236229
 n =        10000
 sum1 =    5002.2685412186829
 sum2 =    5002.26855
 sum3 =    4997.8061791407245
 sum4 =    5002.29150
 sum5 =    5002.27197
 n =     10000000
 sum1 =    5005006.9525715429
 sum2 =    5005029.00
 sum3 =    5000004.6840296527
 sum4 =    5005331.50
 sum5 =    5063766.50
```

第3、4次：

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    5.0149753996103303
 sum2 =    5.01497555
 sum3 =    5.0149753996103303
 sum4 =    5.01497555
 sum5 =    5.01497555
 n =        10000
 sum1 =    4974.1175503542345
 sum2 =    4974.10791
 sum3 =    4969.1025749546052
 sum4 =    4974.10791
 sum5 =    4974.11475
 n =     10000000
 sum1 =    5005102.2663945379
 sum2 =    5005274.00
 sum3 =    5000128.1488446388
 sum4 =    5004925.00
 sum5 =    5064092.00
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    3.9526176470068801
 sum2 =    3.95261788
 sum3 =    3.9526176470068806
 sum4 =    3.95261765
 sum5 =    3.95261765
 n =        10000
 sum1 =    4987.0372569723713
 sum2 =    4987.02734
 sum3 =    4983.0846393253341
 sum4 =    4987.05225
 sum5 =    4987.04053
 n =     10000000
 sum1 =    5004124.4861021861
 sum2 =    5003666.00
 sum3 =    4999137.4488454824
 sum4 =    5004002.50
 sum5 =    5062864.00
```

第5、6次：

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    4.5036596804512552
 sum2 =    4.50366020
 sum3 =    4.5036596804512552
 sum4 =    4.50365973
 sum5 =    4.50365925
 n =        10000
 sum1 =    5013.6576691366672
 sum2 =    5013.66455
 sum3 =    5009.1540094562024
 sum4 =    5013.66113
 sum5 =    5013.66553
 n =     10000000
 sum1 =    5006138.1776785497
 sum2 =    5006230.50
 sum3 =    5001124.5200089104
 sum4 =    5005744.00
 sum5 =    5064939.50
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    5.9983760922813119
 sum2 =    5.99837589
 sum3 =    5.9983760922813119
 sum4 =    5.99837637
 sum5 =    5.99837589
 n =        10000
 sum1 =    4974.4553803208191
 sum2 =    4974.46094
 sum3 =    4968.4570042285268
 sum4 =    4974.45654
 sum5 =    4974.46387
 n =     10000000
 sum1 =    5004642.2469762312
 sum2 =    5004576.00
 sum3 =    4999667.7915956778
 sum4 =    5004535.00
 sum5 =    5064063.50
```

第7、8次：

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    6.2682015790744066
 sum2 =    6.26820183
 sum3 =    6.2682015790744066
 sum4 =    6.26820135
 sum5 =    6.26820135
 n =        10000
 sum1 =    5053.7577726207910
 sum2 =    5053.76807
 sum3 =    5047.4895710417359
 sum4 =    5053.76221
 sum5 =    5053.76758
 n =     10000000
 sum1 =    5005482.6300859479
 sum2 =    5005560.00
 sum3 =    5000428.8723134231
 sum4 =    5005818.50
 sum5 =    5064807.50
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    5.4664456169482074
 sum2 =    5.46644545
 sum3 =    5.4664456169482074
 sum4 =    5.46644545
 sum5 =    5.46644545
 n =        10000
 sum1 =    5052.9617677015076
 sum2 =    5052.94727
 sum3 =    5047.4953220845464
 sum4 =    5052.96631
 sum5 =    5052.97217
 n =     10000000
 sum1 =    5004074.3609734662
 sum2 =    5003782.50
 sum3 =    4999021.3992050830
 sum4 =    5004000.00
 sum5 =    5062657.00
```

第9、10次：

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    4.6632008452061431
 sum2 =    4.66320086
 sum3 =    4.6632008452061422
 sum4 =    4.66320086
 sum5 =    4.66320086
 n =        10000
 sum1 =    4985.7896585624549
 sum2 =    4985.78662
 sum3 =    4981.1264577172442
 sum4 =    4985.79053
 sum5 =    4985.79395
 n =     10000000
 sum1 =    5003788.0999626014
 sum2 =    5003923.00
 sum3 =    4998802.3103027130
 sum4 =    5003630.50
 sum5 =    5063155.50
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =           10
 sum1 =    6.5516075632755317
 sum2 =    6.55160761
 sum3 =    6.5516075632755308
 sum4 =    6.55160809
 sum5 =    6.55160713
 n =        10000
 sum1 =    5038.8021873419375
 sum2 =    5038.80127
 sum3 =    5032.2505797786844
 sum4 =    5038.80322
 sum5 =    5038.80664
 n =     10000000
 sum1 =    5004620.7278727600
 sum2 =    5004533.50
 sum3 =    4999581.9256848190
 sum4 =    5004618.00
 sum5 =    5063584.50
```

值得注意的是当$n = 10^8$时，结果超出了单精度实数的存储范围.

```
shenye@shenye-virtual-machine:~/FortranPrograms$ ./a.out
 n =    100000000
 sum1 =    50000797.014189027
 sum2 =    16777216.0
 sum3 =    50000797.014215067
 sum4 =    16777216.0
 sum5 =    16777216.0
```

四、分析报告

1.问题分析

本次的实习内容要求用五种不同的方法对n个分布在$[0,1]$随机数进行求和：

a). 按随机数数产生的顺序求和，其中随机值定义为双精度实数变量；
b). 同 a), 按随机数产生的顺序求和, 但使用单精度求和；
c). 同 b), 但使用下列方法按随机数产生的顺序求和：

```
s = x1
c = 0
for i = 2 to n
    y = xi - c
    t = s + y
    c = (t - s) – y
    s = t
end
```

d). 使用单精度，先对随机数排序，按从大到小的顺序求和（排序的函数可以使用编程语言自带的）；
e). 同 d), 但按从小到大的顺序求和；

Fortran语言自带的随机数生成函数random_number生成的数x服从$[0,1]$上的均匀分布，记作$X \sim U[0,1]$.

期望$E(X)=0.5$，因此n个数求和的结果约等于$\frac{n}{2}$.

2.编程思路

声明一个双精度实数数组num，调用random_number(num)生成随机数.
变量sum1，sum2，sum3，sum4，sum5分别存储用a), b), c), d), e)方法求和的结果.
变量sum1，sum3为双精度实数，变量sum2，sum4，sum5为单精度实数.
在使用b), d), e)方法求和的时候将num(i)转为单精度实数再进行累加.
由于需要排序的数据较多（最多的一组为10^7个数），使用时间复杂度为$O(nlog_2n)$的快速排序算法进行排序.

3.问题讨论

（1）造成几种方法计算结果的偏差的主要原因是什么？哪种方法的计算精度最高？哪种精度最低？为什么？

方法a)计算结果的偏差主要来源于“大数吃小数”的舍入误差.

方法b)计算结果的偏差主要来源于舍入误差，且由于单精度实数的有效位数比双精度实数要低，导致方法b的偏差大于方法a).

方法c)为Kahan's Summation Algorithm，用c记录舍入误差，极大地减小了“大数吃小数”误差.

方法d)计算结果的偏差主要来源于大数加小数. 由于方法d)使用从大到小的顺序进行累加，累加过程的后期累加和较大，但是参与累加的数却越来越小，导致越靠后累加的数对最终结果的有效数字位数的贡献越小.

方法e)计算结果的偏差主要来源于舍入误差.

由以上讨论可知，在“大数吃小数”误差显著时，方法c）精度最高，方法d）精度最低.

（2）不同方法的计算量有何差别?

方法a), b), c)的时间复杂度为$O(n)$，方法d), e)的时间复杂度为$O(nlog_2n)$.

方法c)的计算量约为方法b)的4倍（每次循环多做了3次加减法运算）.

总体上来说计算量从小到大为a)≈b)＜c)<d)≈e).

（3）以上两个问题的答案是否会随着 n 的取值不同发生变化？若存在变化，请解释原因。

对于问题（1），从实验结果可以看出，n越大，不同方法算出的结果差距越大. 当n较小时，数字总和也较小，“大数吃小数”产生的舍入误差很小，各个方法的差距不大，因此使用双精度求和的方法a)精度最高；当n逐渐增大时，“大数吃小数”误差也逐渐变大，此时方法c)精度逐渐提高，直到超过方法a).

综上，n较小时，方法a)精度最高；n较大时，方法c)精度最高. 无论n的大小，方法d)的精度最低.

对于问题（2），不同方法的计算量排序随n的取值不同不发生变化.