HW3

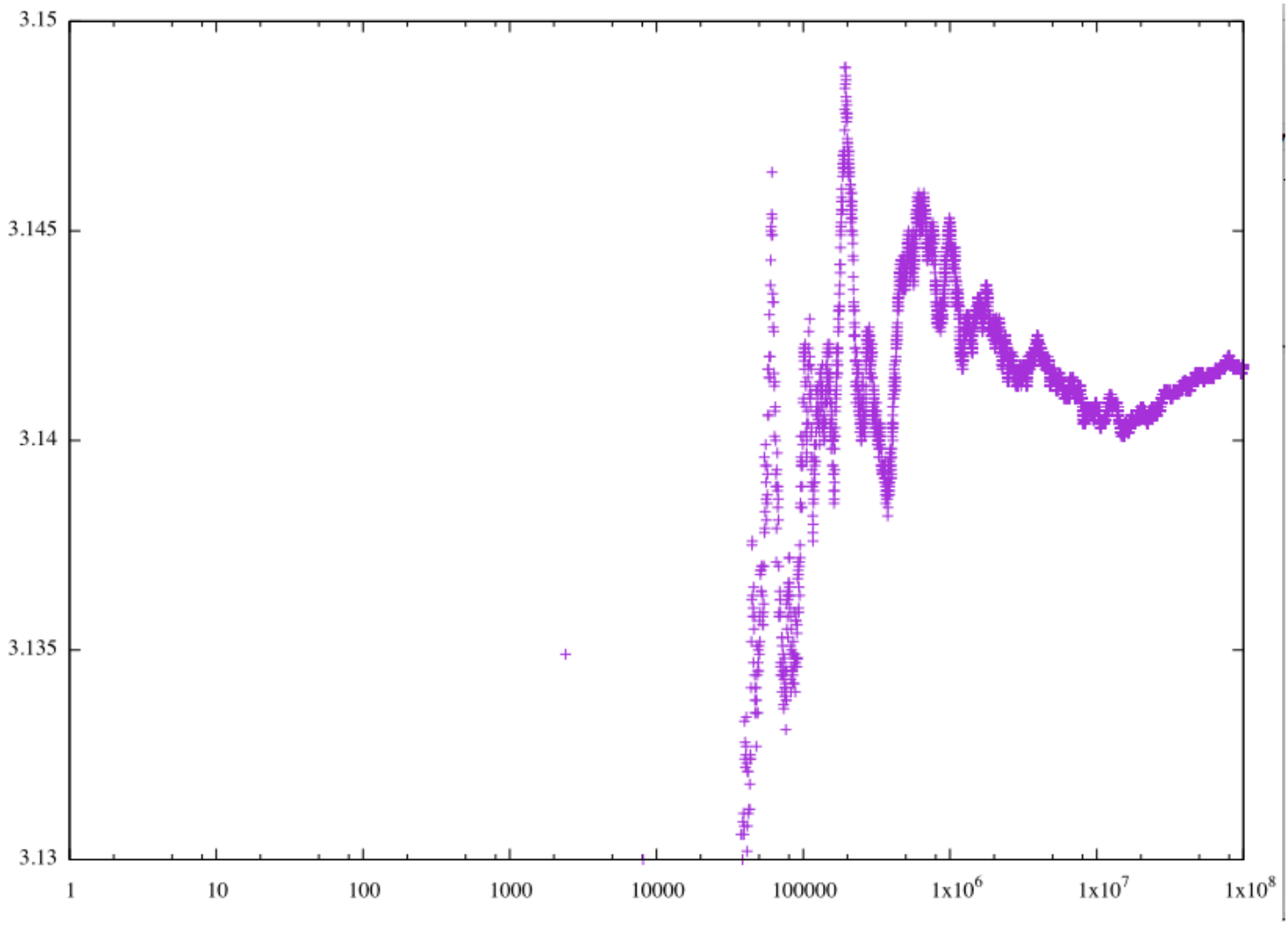
Path to the source code: `/home/d/dx/dxj4360/HW3`

P1: π from sphere

Monte Carlo Method

- Counting condition: `((x*x + y*y + z*z).LE.1)`
- Final equation: $\frac{\text{points in sphere}}{\text{total points in space}} \sim \frac{4/3\pi r^3}{2^3}$ --> $\pi = 6/r^3 \times \frac{\text{points in sphere}}{\text{total points in space}}$ where $r = 1$

Result

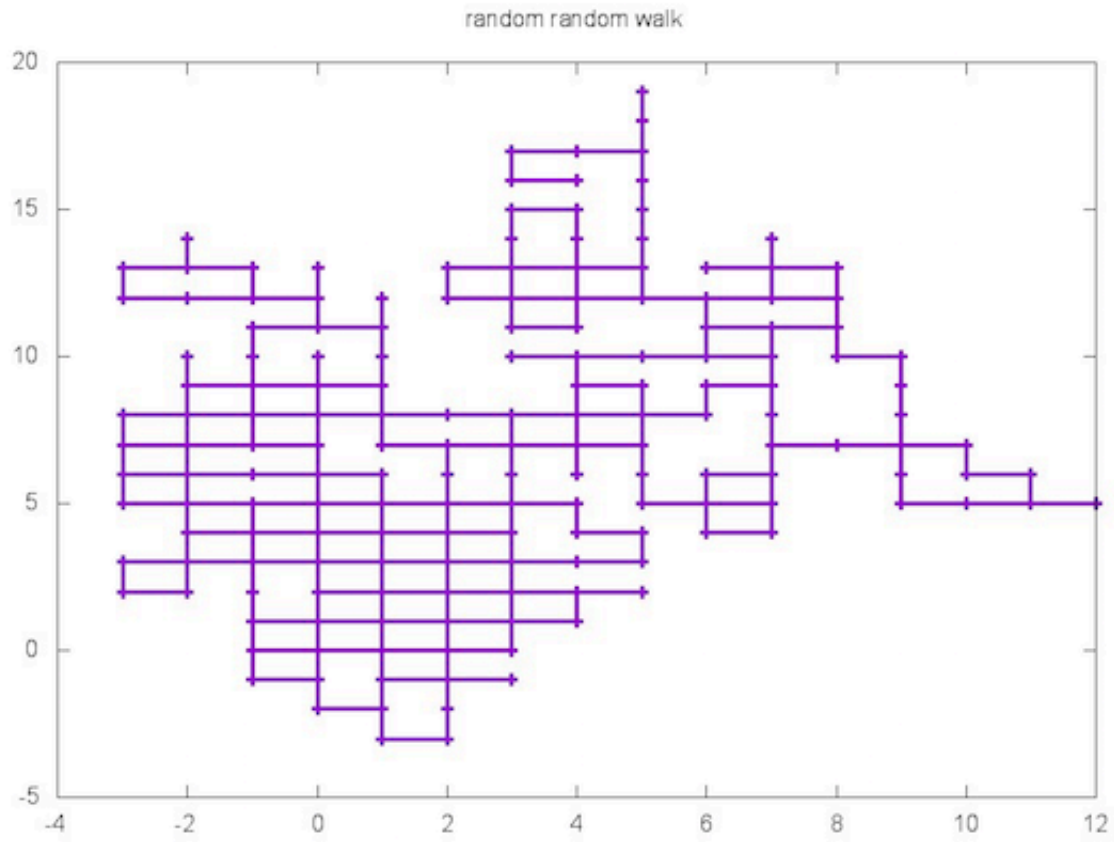


From the plot, we can estimate that the accuracy will reach 4 significant figures around $5 * 10^6$ trials.

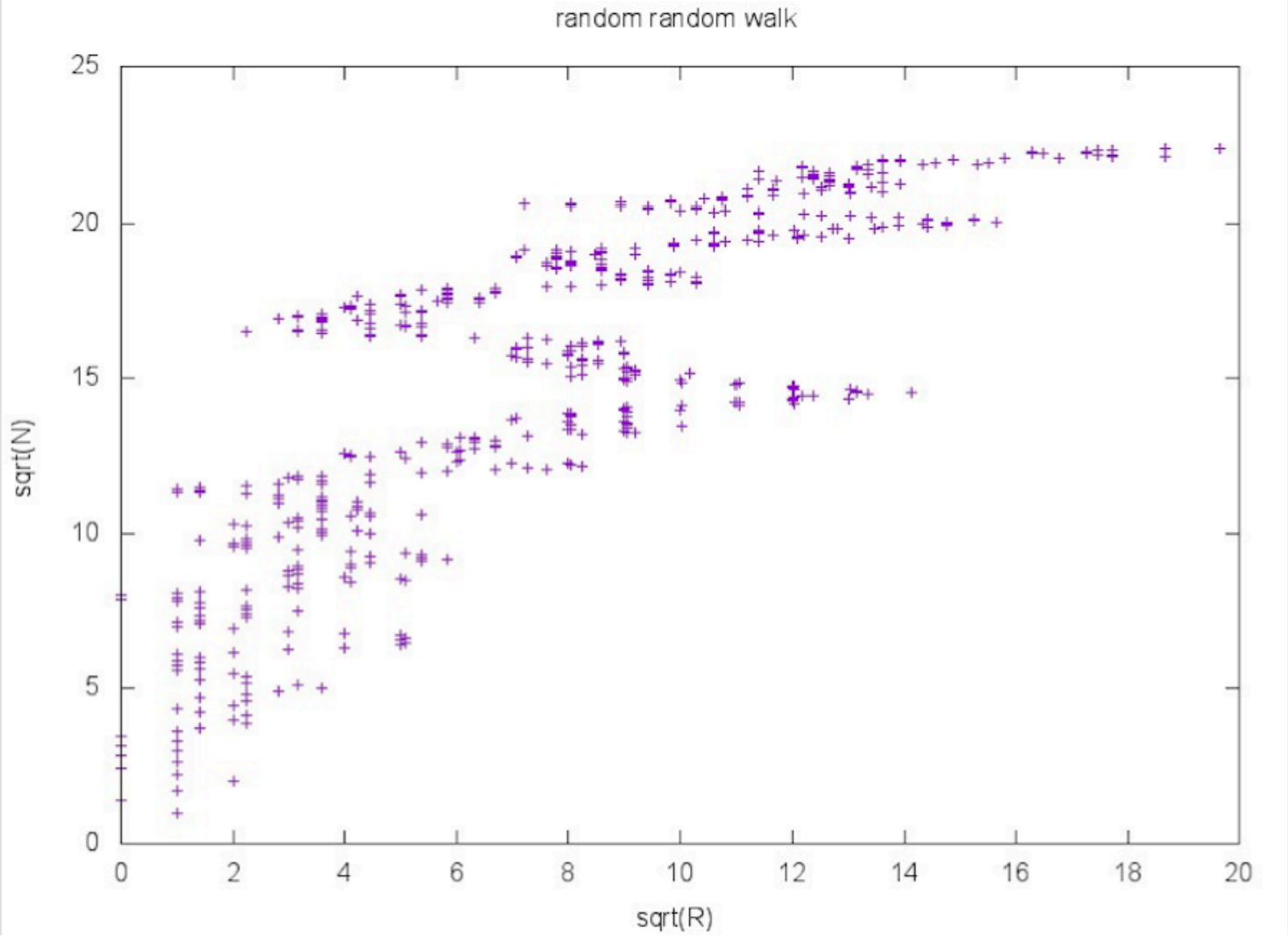
P2: Random Walk

The plot function is integrated in the code. Just run the executable file will produce the data and generate the plots.

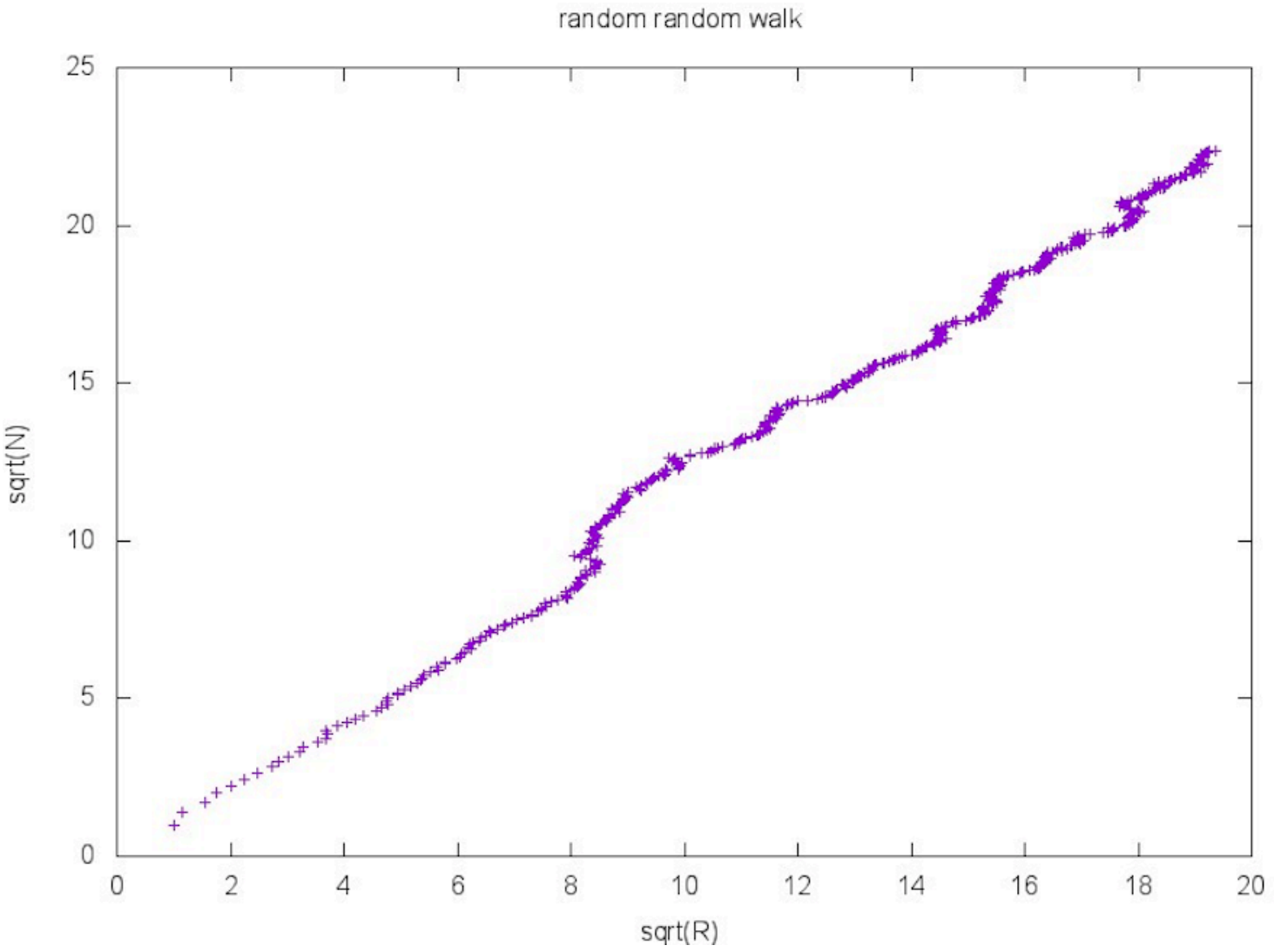
a) x-y path of the walk



b) sqrt(R) vs. sqrt(N)



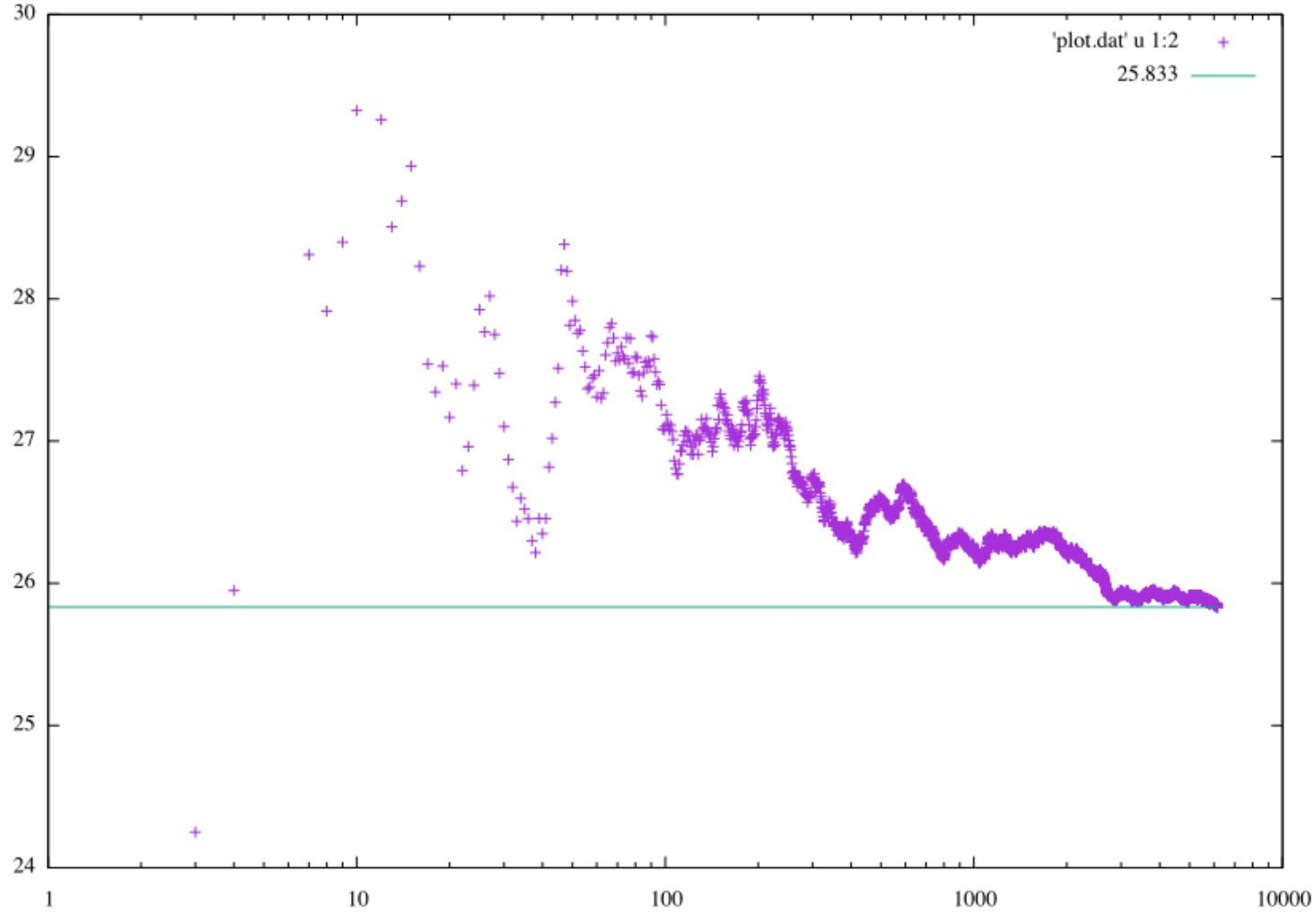
c) sqrt(R) vs. sqrt(N) over 100 trials



The distance random walk can reach should be propotional to the sqrt(steps). Single round will have fluctuation but 100 rounds will eventually show this trend. Plot c) is the trend of plot b)

P3: Monte Carlo Integration

Accuracy increase as number of trails



Appendix

Additional file drand48.f is needed for P2 and P3

P1

```
      Program pai
      Implicit none
      integer max
c declarations
      Real volume, x, y, z, area
      Integer i, pi3, pi2
      print*,'input the number for try'
      read(5,*) max
      pi3=0
c use max as random seed
      call srand(max)
c execute
      Do 10 i=1, max
c generate (x,y) within [-1,1]:
         x = rand()*2-1
         y = rand()*2-1
         z = rand()*2-1
         If ((x*x + y*y + z*z) .LE. 1) pi3 = pi3 + 1
         volume = 6.0 * pi3/Real(i)
c volume inside sphere ~ pts/i = 4/3*pi*r**3 --> pi= pts*8
         if(mod(i,int(sqrt(i*1.0))).eq.1) Write(6,100) i, volume
 10       Continue

          Write(6,100) max, volume
100 format(1x,'try= ',i10,2x,'pai=',f8.4)
c do-while loop to get enough significant numbers
      pi3=0
      i=0
      do while(abs(volume-3.1415927).GT.1E-4)
         x = rand()*2-1
         y = rand()*2-1
         z = rand()*2-1
         i = i+1
         if ((x*x + y*y + z*z).LE.1) pi3 = pi3 +1
             volume = 6.0 * pi3/Real(i)
      enddo
      write (6,200), i,volume
200 format(i8," trials are needed, the final value of Pi is ",f5.3)
      stop
      end
```

P2

```
 program P2
  implicit none
  integer :: i
  open(unit=20,file='walk1.dat')
  open(unit=21,file='walk2.dat')
  do i=1,100
    call walk1
    call walk2
  enddo
  close(20)
  close(21)
  call system("gnuplot plot.gnu")
  print *,"The distance random walk can reach should be propotional to the sqrt(steps). P
  ! plot.gnu will generate 2-[a,b,c].jpeg, 2-[a,b].jpeg will only use first 500 lines in 
end program P2

subroutine walk1
  ! this version will only walk to x+ and y+ directions
  implicit none
  real*8, external :: drand48
  integer :: n, x, y
  x=0
  y=0
  do n=1,500
    ! map rand to [-1, 1], positive value for heading to y direction; negative value for 
     if (2*drand48()-1.le.0) then
        x=x+1
     else
        y=y+1
     endif
     write (20,*),x,y,sqrt(x*x*1.0+y*y*1.0),sqrt(n*1.0),n
     ! output x,y,sqrt(R),sqrt(N),n to plot.dat for plotting
  enddo
end subroutine walk1

subroutine walk2
  ! this version will walk to x+,x-,y+,y- directions
  implicit none
  real*8, external :: drand48
  integer :: n, x, y
  x=0
  y=0
  do n=1,500
    ! map rand to [-1, 1], positive value for heading to y direction; negative value for 
```

```
      if (2*drand48()-1.le.0) then
         ! map rand to [-1, 1], positive value for heading to x+ direction; negative value
         if (2*drand48()-1.le.0) then
           x=x-1
         else
           x=x+1
         endif
      else
         ! map rand to [-1, 1], positive value for heading to y+ direction; negative value
         if (2*drand48()-1.le.0) then
           y=y-1
         else
           y=y+1
         endif
      endif
      write (21,*),x,y,sqrt(x*x*1.0+y*y*1.0),sqrt(n*1.0),n
      ! output x,y,sqrt(R),sqrt(N),n to plot.dat for plotting
   enddo
end subroutine walk2
```

gnuplot file

```
#P2-a: x-y path for the walk
set term jpeg size 800,1200;
set out "2-a.jpeg";
set multiplot
set autoscale

set origin 0,0.5
set size 1,0.5
set title "positive random walk"
plot "walk1.dat" every ::::499 u 1:2 notitle w lp lw 3

set origin 0,0
set size 1,0.5
set title "random random walk"
plot "walk2.dat" every ::::499 u 1:2 notitle w lp lw 3

unset multiplot;
unset out;

#P2-b: sqrt(R) vs. sqrt(N)
set term jpeg size 800,1200;
set out "2-b.jpeg";
```

```
set multiplot
set autoscale
set xlabel "sqrt(R)"
set ylabel "sqrt(N)"

set origin 0,0.5
set size 1,0.5
set title "positive random walk"
plot "walk1.dat" every ::::499 u 3:4 notitle

set origin 0,0
set size 1,0.5
set title "random random walk"
plot "walk2.dat" every ::::499 u 3:4 notitle

unset multiplot;
unset out;

#P2-c: sqrt(R) vs. sqrt(N) over 100 trials
set term jpeg size 800,1200;
set out "2-c.jpeg";
set multiplot
set autoscale
set xlabel "sqrt(R)"
set ylabel "sqrt(N)"

set origin 0,0.5
set size 1,0.5
set title "positive random walk"
plot "< awk '{R[$5]=R[$5]+$3;N[$5]=N[$5]+$4; nr[$5]++} END {for (i in R) {print i, R[i]/n

set origin 0,0
set size 1,0.5
set title "random random walk"

plot "< awk '{R[$5]=R[$5]+$3;N[$5]=N[$5]+$4; nr[$5]++} END {for (i in R) {print i, R[i]/n

unset multiplot;
unset out;
```

P3

```
program P3
  implicit none
  real*8, external :: drand48
  real*8 :: I=0,x
  integer :: n,j
  n = 0
  ! do-while loop until reach desired accuracy
  do while (abs(I-155.0/6)>1E-4)
     x = 0
     n = n + 1
     ! do loop to scatter one point in 10-D space
     do j = 1,10
       x = x + drand48()
     enddo
     I = I * ( n - 1 ) + x**2 ! reverse the total value in 10-D space for n trails
     I = I / n ! average value of all the n possible points is the integration result
     print "(i6,2x,f6.3)", n, I
  enddo
end program P3
```