

# HW3

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

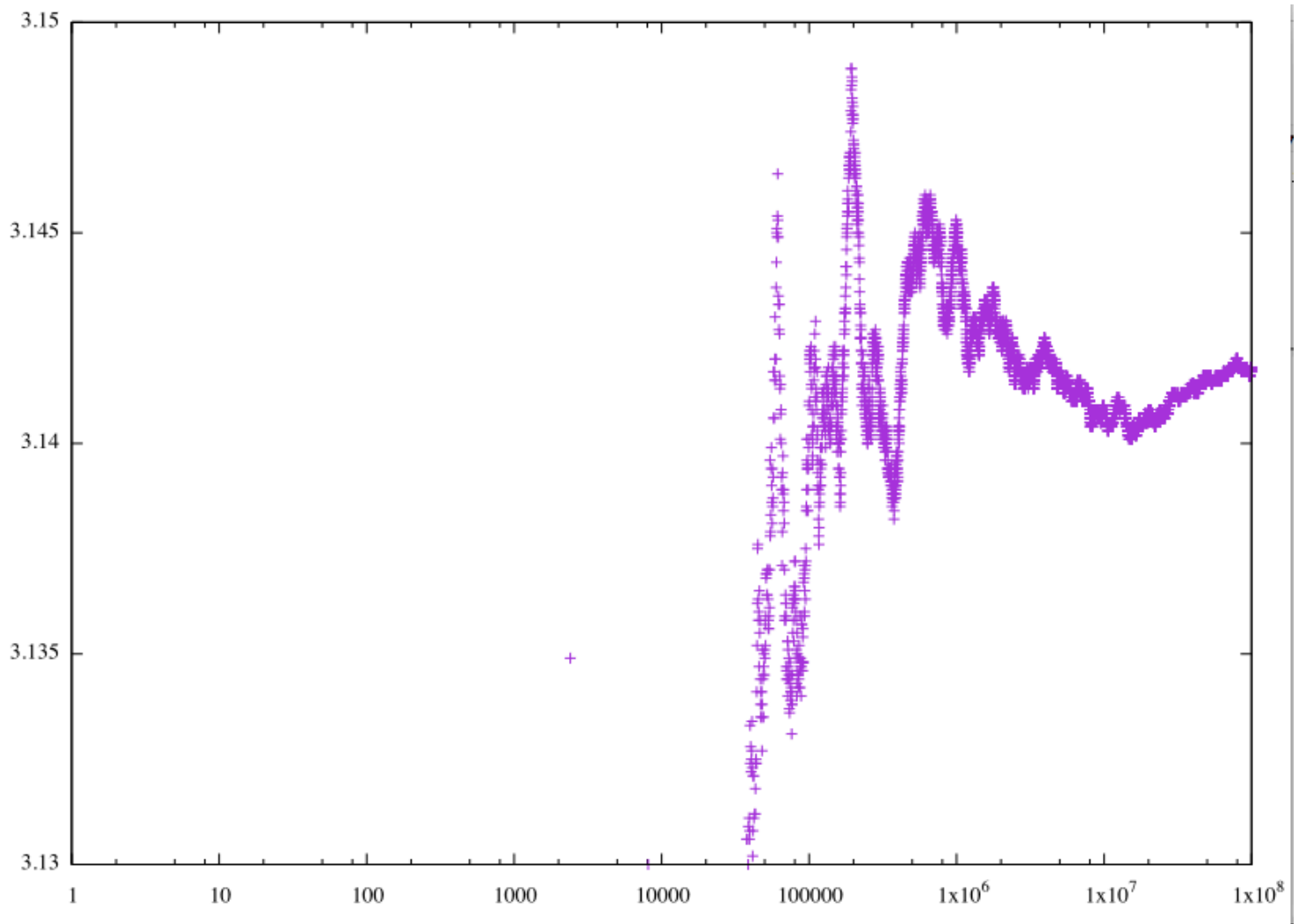
## P1: $\pi$ from sphere

---

### Monte Carlo Method

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

### Result

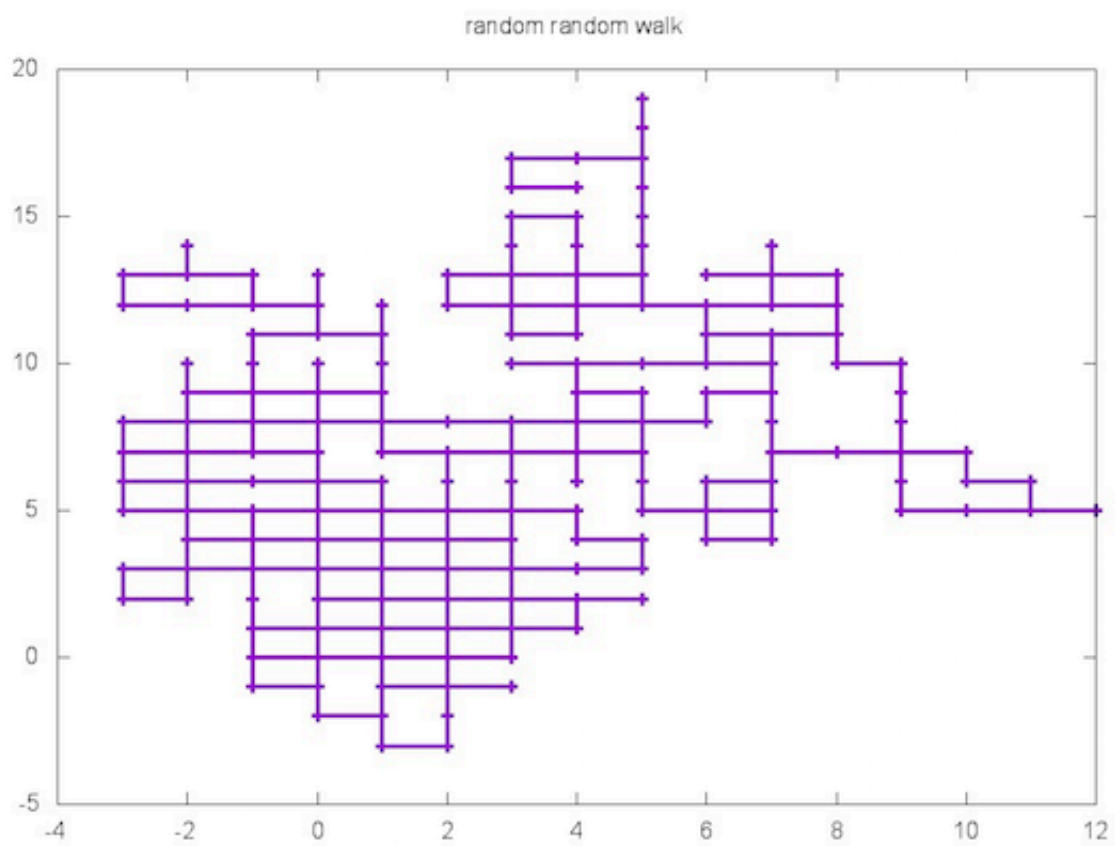


From the plot, we can estimate that the accuracy will reach 4 significant figures around  $5 * 10^6$  trials.

## P2: Random Walk

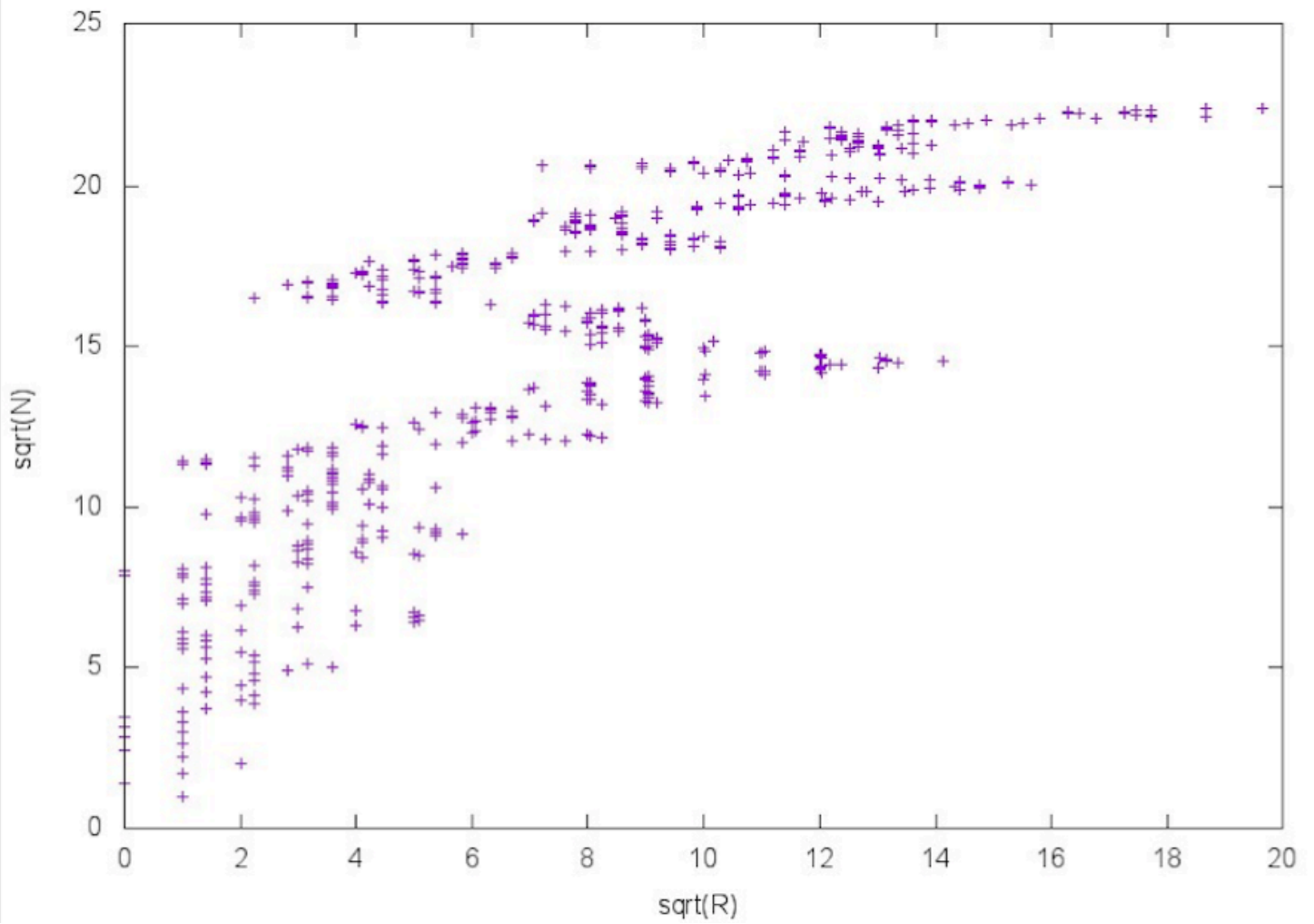
*The plot function is integrated in the code. Just run the executable file will produce the data and generate the plots.*

### a) x-y path of the walk

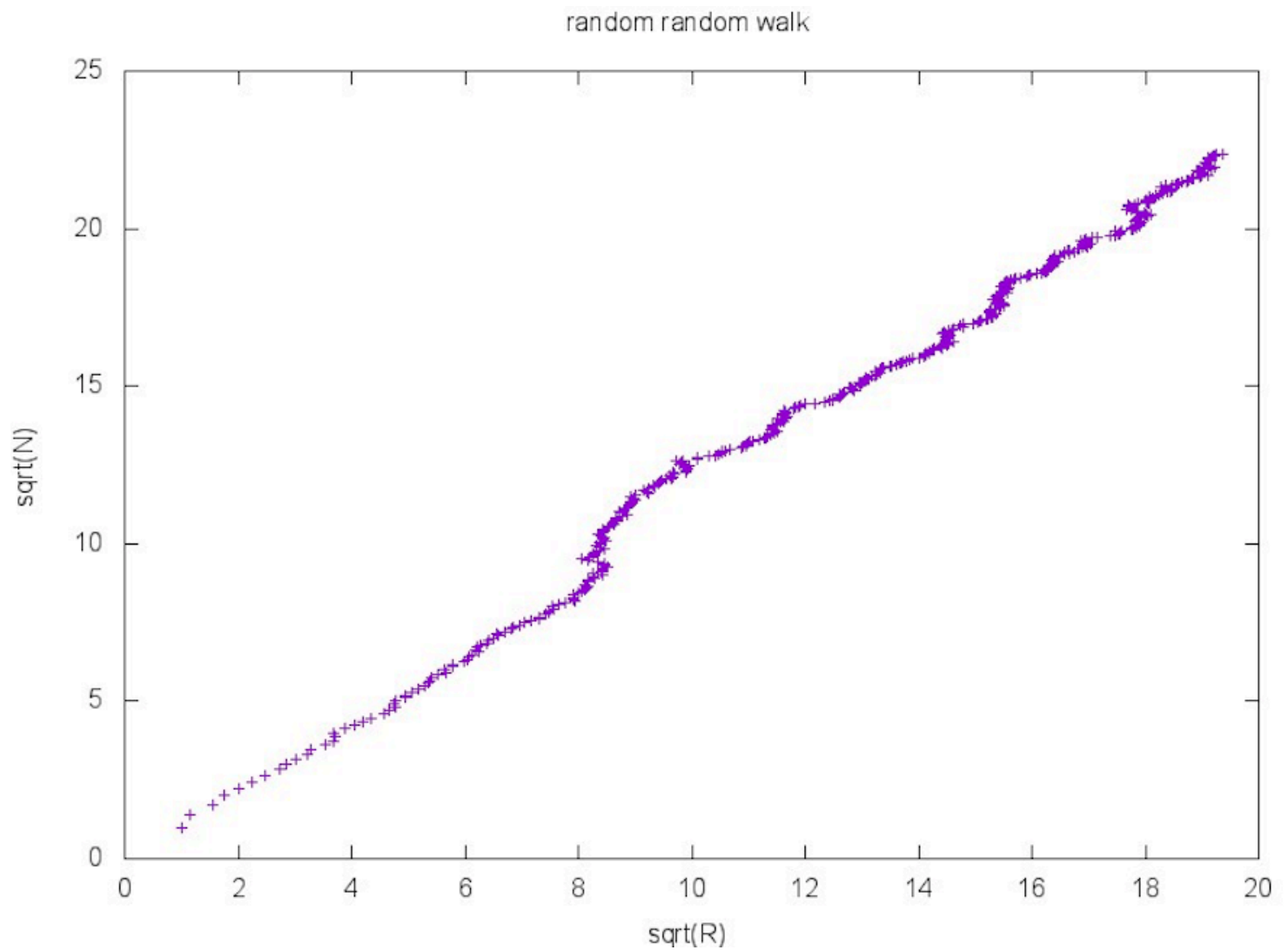


**b)  $\sqrt{R}$  vs.  $\sqrt{N}$**

random random walk



c)  $\sqrt{R}$  vs.  $\sqrt{N}$  over 100 trials

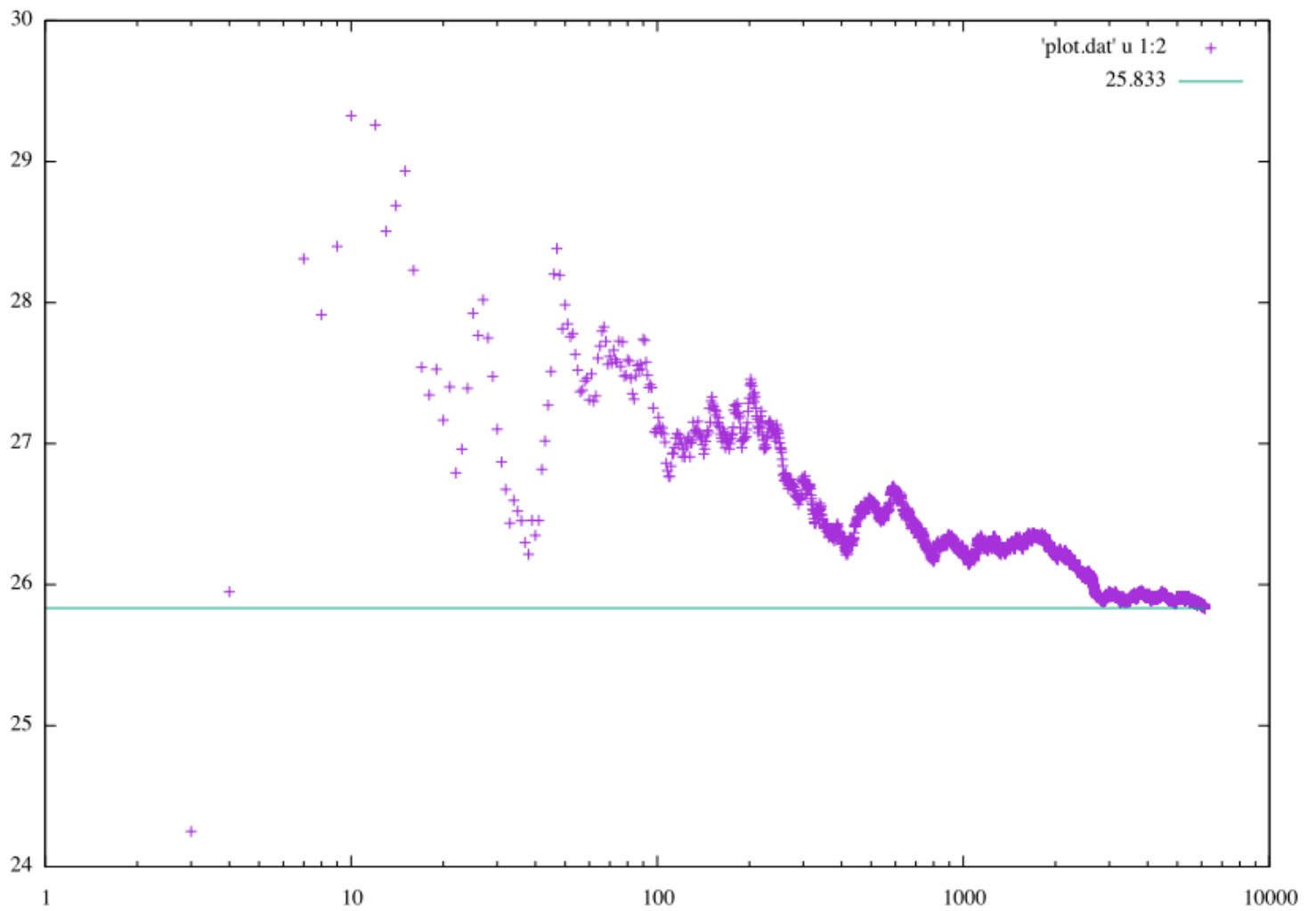


The distance random walk can reach should be proportional to the  $\sqrt{\text{steps}}$ . Single round will have fluctuation but 100 rounds will eventually show this trend. Plot c) is the trend of plot b)

## P3: Monte Carlo Integration

---

### Accuracy increase as number of trails



## Appendix

---

*Additional file drand48.f is needed for P2 and P3*

**P1**

```

1      Program pai
2      Implicit none
3      integer max
4  c declarations
5      Real volume, x, y, z, area
6      Integer i, pi3, pi2
7      print*, 'input the number for try'
8      read(5,*) max
9      pi3=0
10     c use max as random seed
11         call srand(max)
12     c execute
13         Do 10 i=1, max
14     c generate (x,y) within [-1,1]:
15         x = rand()*2-1
16         y = rand()*2-1
17         z = rand()*2-1
18         If ((x*x + y*y + z*z) .LE. 1) pi3 = pi3 + 1
19         volume = 6.0 * pi3/Real(i)
20     c volume inside sphere ~ pts/i = 4/3*pi*r**3 --> pi= pts*8
21         if(mod(i,int(sqrt(i*1.0))).eq.1) Write(6,100) i, volume
22     10     Continue
23
24         Write(6,100) max, volume
25 100 format(1x, 'try= ', i10, 2x, 'pai=', f8.4)
26     c do-while loop to get enough significant numbers
27         pi3=0
28         i=0
29         do while(abs(volume-3.1415927).GT.1E-4)
30             x = rand()*2-1
31             y = rand()*2-1
32             z = rand()*2-1
33             i = i+1
34             if ((x*x + y*y + z*z).LE.1) pi3 = pi3 +1
35             volume = 6.0 * pi3/Real(i)
36         enddo
37         write (6,200), i,volume
38 200 format(i8," trials are needed, the final value of Pi is ",f5.3)
39         stop
40     end

```

```

1  program P2
2      implicit none
3      integer :: i
4      open(unit=20,file='walk1.dat')
5      open(unit=21,file='walk2.dat')
6      do i=1,100
7          call walk1
8          call walk2
9      enddo
10     close(20)
11     close(21)
12     call system("gnuplot plot.gnu")
13     print *, "The distance random walk can reach should be propotional to the sqrt(steps). P
14     ! plot.gnu will generate 2-[a,b,c].jpeg, 2-[a,b].jpeg will only use first 500 lines in
15 end program P2
16
17 subroutine walk1
18     ! this version will only walk to x+ and y+ directions
19     implicit none
20     real*8, external :: drand48
21     integer :: n, x, y
22     x=0
23     y=0
24     do n=1,500
25         ! map rand to [-1, 1], positive value for heading to y direction; negative value for
26         if (2*drand48()-1.le.0) then
27             x=x+1
28         else
29             y=y+1
30         endif
31         write (20,*)x,y,sqrt(x*x*1.0+y*y*1.0),sqrt(n*1.0),n
32         ! output x,y,sqrt(R),sqrt(N),n to plot.dat for plotting
33     enddo
34 end subroutine walk1
35
36 subroutine walk2
37     ! this version will walk to x+,x-,y+,y- directions
38     implicit none
39     real*8, external :: drand48
40     integer :: n, x, y
41     x=0
42     y=0
43     do n=1,500
44         ! map rand to [-1, 1], positive value for heading to y direction; negative value for

```



```

45     if (2*drand48()-1.le.0) then
46         ! map rand to [-1, 1], positive value for heading to x+ direction; negative value
47         if (2*drand48()-1.le.0) then
48             x=x-1
49         else
50             x=x+1
51         endif
52     else
53         ! map rand to [-1, 1], positive value for heading to y+ direction; negative value
54         if (2*drand48()-1.le.0) then
55             y=y-1
56         else
57             y=y+1
58         endif
59     endif
60     write (21,*),x,y,sqrt(x*x*1.0+y*y*1.0),sqrt(n*1.0),n
61     ! output x,y,sqrt(R),sqrt(N),n to plot.dat for plotting
62 enddo
63 end subroutine walk2

```

## gnuplot file

```

1  #P2-a: x-y path for the walk
2  set term jpeg size 800,1200;
3  set out "2-a.jpeg";
4  set multiplot
5  set autoscale
6
7  set origin 0,0.5
8  set size 1,0.5
9  set title "positive random walk"
10 plot "walk1.dat" every :::499 u 1:2 notitle w lp lw 3
11
12 set origin 0,0
13 set size 1,0.5
14 set title "random random walk"
15 plot "walk2.dat" every :::499 u 1:2 notitle w lp lw 3
16
17 unset multiplot;
18 unset out;
19
20 #P2-b: sqrt(R) vs. sqrt(N)
21 set term jpeg size 800,1200;
22 set out "2-b.jpeg";

```

```

23 set multiplot
24 set autoscale
25 set xlabel "sqrt(R)"
26 set ylabel "sqrt(N)"
27
28 set origin 0,0.5
29 set size 1,0.5
30 set title "positive random walk"
31 plot "walk1.dat" every :::499 u 3:4 notitle
32
33 set origin 0,0
34 set size 1,0.5
35 set title "random random walk"
36 plot "walk2.dat" every :::499 u 3:4 notitle
37
38 unset multiplot;
39 unset out;
40
41 #P2-c: sqrt(R) vs. sqrt(N) over 100 trials
42 set term jpeg size 800,1200;
43 set out "2-c.jpeg";
44 set multiplot
45 set autoscale
46 set xlabel "sqrt(R)"
47 set ylabel "sqrt(N)"
48
49 set origin 0,0.5
50 set size 1,0.5
51 set title "positive random walk"
52 plot "< awk '{R[$5]=R[$5]+$3;N[$5]=N[$5]+$4; nr[$5]++} END {for (i in R) {print i, R[i]/n
53
54 set origin 0,0
55 set size 1,0.5
56 set title "random random walk"
57
58 plot "< awk '{R[$5]=R[$5]+$3;N[$5]=N[$5]+$4; nr[$5]++} END {for (i in R) {print i, R[i]/n
59
60 unset multiplot;
61 unset out;

```

```

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

```