Program-M12

Devansh Shukla
I18PH021

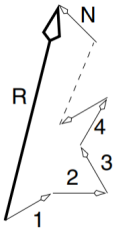
a) Estimate the radial distance (R) covered by a particle in a Random Walk and provide MC algorithm to estimate Rrms distance.

b) Write and execute a FORTRAN program for a Random walk and to estimate average radial distance (R) employing Monte Carlo method. Plot 2D visualization of Random walk with your generated coordinates

1 Theory

There are many physical processes, such as Brownian motion and electron transport through metals, in which a particle appears to move randomly. For example, consider a perfume molecule released in the middle of a classroom. It collides randomly with other molecules in the air and eventually reaches the instructor's nose. The problem is to determine how many collisions, on average, the molecule must make to travel a radial distance of R, if it travels a step length r_{rms} on the average (root-mean-square average) between collisions

2 Numerical Solution



For our model, we start at the origin and take N steps in the x–y plane of lengths

$$(\Delta x1,\ \Delta y1),\ (\Delta x2,\ \Delta y2),\ (\Delta x3,\ \Delta y3),\ ...,\ (\Delta xn,\ \Delta yN) \tag{1}$$

The radial distance R from the starting point traveled after N steps is

$$R^2 = (\Delta x_1 + \Delta x_2 + ... + \Delta x_N)^2 + (\Delta y_1 + \Delta y_2 + ... + \Delta y_N)^2 \tag{2}$$

$$R^2 = \Delta x_1^2 + \Delta x_2^2 + ... + \Delta x_N^2 + 2\Delta x_1\Delta x_2 + 2\Delta x_1\Delta x_3 + 2\Delta x_2\Delta x_1 + ... \tag{3}$$

$$R^2 \approx N|r^2| \tag{4}$$

$$R \approx \sqrt{N} r_{rms} \tag{5}$$

Here r_{rms} is the average (root-mean-squared) step size

In summary, the equation that for a walk of N steps covering a total distance of Nr_{rms}, the walk ends up, on the average, a radial distance $\sqrt{N}r_{rms}$ from the starting point. For large N this is significantly less than Nr_{rms}, but is not zero (which is the average of the displacement vector).

Practical simulations agree with this theory, but rarely perfectly, with the level of agreement depending upon how the averages are taken, and just how the randomness is built into each step.

3 Program Algorithm

NOTE: Blue-colored text represents variables in the algorithm, eg. variable.

1. Program open
2. Define X, Y, pi, x_value, y_value, a, b, step, R, i, j, n.
3. Open a file "walk.dat" with write access.
4. Get the input from the user for the total no of steps (n)
5. Open a do-loop for index (i) from 1 to n
6. Compute two random numbers, a and b
7. Compute x_value and y_value, using

$$x_{value} = step * \cos(2\pi a)$$
$$y_{value} = step * \sin(2\pi b)$$

8. Compute new coordinates in X array and Y array.
9. Compute the new radial distance using,
$$R = R + (x^2 + y^2) \tag{6}$$
10. Write the paramters to the file, x_value, y_value, X(i) and Y(i)
11. Terminate the do-loop.
12. Compute and print the square root of R.
13. Close file.
14. End program.

4 Program

4.1 Fortran program:

For computing the parameters

```
!=========================================
! random_walk.f90
! Author: Devansh Shukla
!-----------------------------------------
program walk
    implicit none
    ! Define the parameters
    real*8, dimension(1000000) :: X, Y
    real*8 :: pi=3.14159
    real*8 :: x_value=0.0, y_value=0.0, a=0.0, b=0.0, step=1, R=0.0
    integer :: i=0, j=0, n=0
    character(len=*), parameter :: fmt="(F12.3,xF12.3,xF12.3,F12.3)"

    ! open the data file
    open(unit=8, file="walk.dat")
    ! get input from user
    print *, "Enter the no. of steps (n)"
    read *, n

    ! initialize the array
    do j=1, n
       X(j) = 0.0
       Y(j) = 0.0
    enddo

    ! compute
    do i=1, n, 1
        call random_number(a)
        call random_number(b)
        x_value = step * cos(2*pi*a)
        y_value = step * sin(2*pi*b)
        X(i) = X(i-1) + x_value
        Y(i) = Y(i-1) + y_value
        write (8, fmt) x_value, y_value, X(i), Y(i)
        R = R + x_value**2 + y_value**2
    enddo
    ! print for the user
    print "(A, F10.3)", "The value of Radius, R =", sqrt(R)
    ! close the data file
    close(8)
end program walk
```

4.2 Python program: Plots

```
#!/usr/bin/env python
"""
Author: Devansh Shukla
"""
# In[0]
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

plt.style.use("rcStyleSheet.mplstyle")
mpl.use("pgf")
plt.ioff()
```

```
# In[2]
for n in [10000]:
    df = pd.read_csv(f"walk_{n}.dat", engine="python", delimiter=" ", header=None, skipinitialspace=True, comment="#")
    print(df)
    fig = plt.figure(figsize=(4, 4))
    gs = gridspec.GridSpec(1, 1)
    ax = fig.add_subplot(gs[0, 0])
    ax.plot(df[2], df[3], "-", markersize=1.5, color="C0")
    ax.set_xlabel("X-coordinate")
    ax.set_ylabel("Y-coordinate")
    plt.title(f"Random walk (n={n})")
    plt.savefig(f"outputs/walk_{n}.pdf")
```

5 Results

5.1 Terminal Output

5.1.1 n=100

```
 Enter the no. of steps (n)
100
The value of Radius, R =    10.057
```

5.1.2 n=1000

```
 Enter the no. of steps (n)
1000
The value of Radius, R =    31.760
```

5.1.3 n=2000

```
 Enter the no. of steps (n)
2000
The value of Radius, R =    44.967
```

5.1.4 n=5000

```
Enter the no. of steps (n)
5000
The value of Radius, R =    70.878
```

5.1.5 n=10000

```
 Enter the no. of steps (n)
10000
The value of Radius, R =   100.053
```

5.2 Data file

First 10 lines from the data files:

5.2.1 n=100

```
    -0.966      -0.076      -0.966      -0.076
     0.686       0.946      -0.279       0.870
    -0.793       0.226      -1.072       1.096
    -0.879       0.664      -1.951       1.760
    -0.767       0.781      -2.718       2.541
     0.958       0.772      -1.760       3.312
     0.481      -0.628      -1.279       2.684
    -0.998      -0.332      -2.277       2.352
     0.961       0.024      -1.316       2.376
     0.927       1.000      -0.390       3.376
```

5.2.2 n=1000

```
     0.999      -0.999       0.999      -0.999
    -1.000      -0.021      -0.001      -1.020
     0.450       0.820       0.449      -0.200
    -0.862       0.291      -0.413       0.091
    -1.000       0.752      -1.413       0.843
     0.513       0.467      -0.900       1.310
     0.603       0.979      -0.297       2.290
     0.537       0.120       0.240       2.410
    -0.753      -0.458      -0.513       1.953
    -0.960       0.983      -1.473       2.935
```

5.2.3 n=2000

0.523	0.103	0.523	0.103
-0.454	-0.552	0.069	-0.449
-0.568	-0.939	-0.499	-1.388
-0.064	0.920	-0.563	-0.469
-0.675	0.362	-1.238	-0.107
1.000	0.360	-0.238	0.253
0.615	0.995	0.377	1.249
-0.999	0.920	-0.623	2.169
0.644	-0.064	0.021	2.104
0.841	-0.945	0.862	1.159

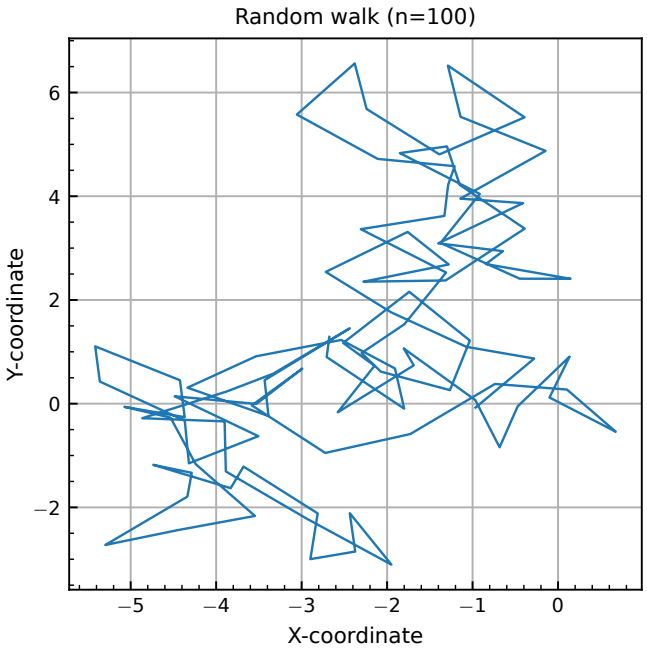
5.2.4 n=5000

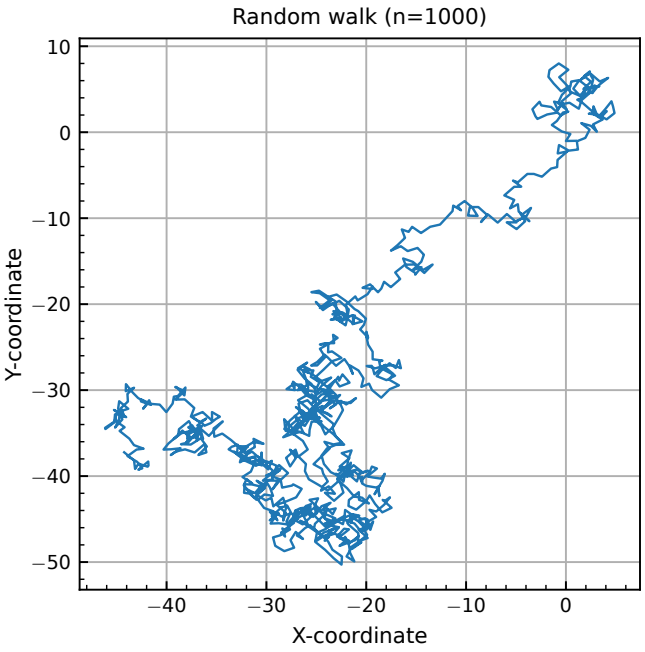
0.596	0.358	0.596	0.358
0.900	0.584	1.495	0.942
-0.461	0.680	1.034	1.622
-0.995	-0.689	0.039	0.933
0.821	0.948	0.860	1.881
-0.026	0.999	0.834	2.880
0.961	-0.760	1.795	2.120
0.337	-0.934	2.132	1.187
-0.009	-1.000	2.123	0.187
-0.321	0.942	1.802	1.129

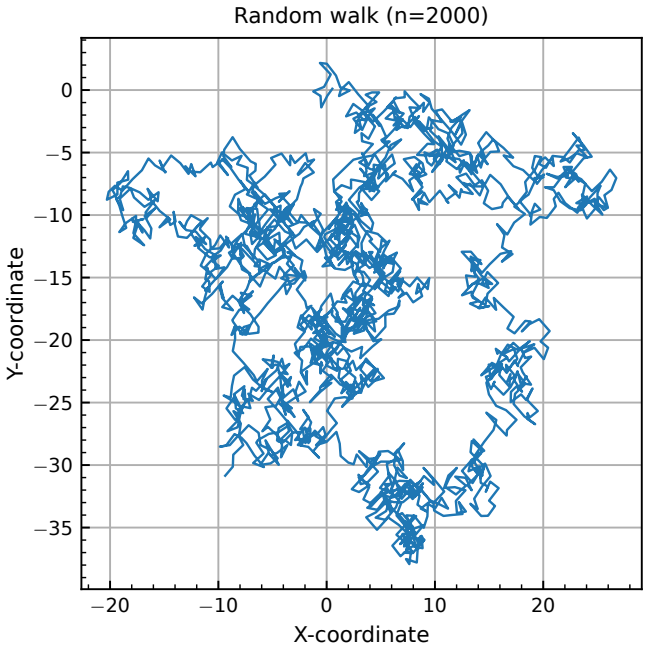
5.2.5 n=10000

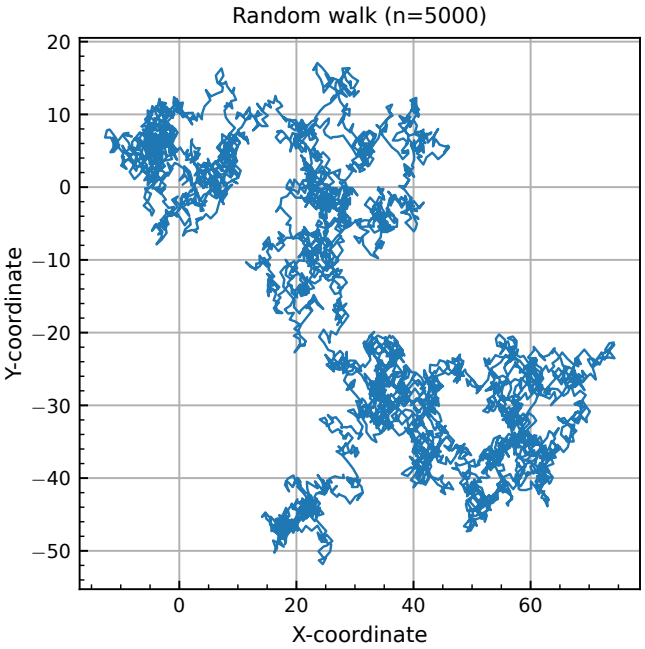
-0.017	0.548	-0.017	0.548
0.630	-0.978	0.612	-0.430
-0.454	-0.758	0.158	-1.188
-0.537	0.372	-0.378	-0.816
-0.414	0.881	-0.792	0.064
-0.936	-0.470	-1.728	-0.405
0.499	-0.602	-1.230	-1.007
0.939	-0.209	-0.291	-1.217
0.692	-0.720	0.401	-1.936
0.762	0.486	1.164	-1.451

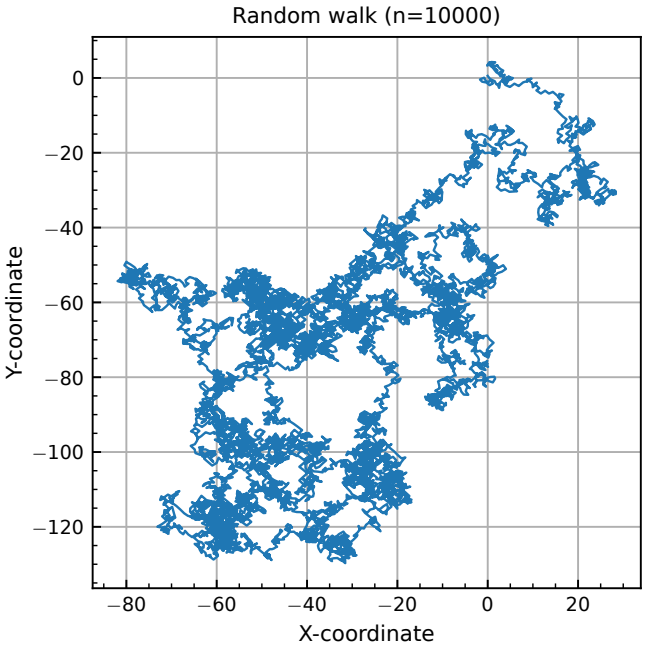
5.3 Plots











6 Remarks

The programs can be used to numerically trace and simulate the problem of random walk. The parameters computed numerically and via the programs are in agreement.