PH415 Assignment 1
Monte Carlo Integration Test

S Venkat Bharadwaj

August 26, 2023

Problem

Packing of hard spheres of radius R (between 1 & 2) randomly inside a cube of side L taking L = 20 $\times$ R. The spheres should not overlap. Make sure that any part of the sphere is not outside the cube. Since N is not exactly known, take a stopping criteria for a particular realization as the number of consecutive failure is equal to $10 * N_{max}$ (N_{max} is the number of spheres for packing fraction = 0.74).

Take 10000 realizations and evaluate the following:

i Number of Spheres placed (N) and Average Spacing between spheres (r_mean) in each iteration.

ii Store the values of N and r_{mean} in a file, and plot their distribution. Calculate the Mean and Standard Deviation of the distributions.

iii Arrange r_{mean} (along with N) in ascending order and plot r_{mean} vs N

iv Plot 3D configuration of one of the realizations

General Algorithm

- Step 1: Get value of R, L using RNG.
- Step 2: Get the value of N_{max} from the R and packing fraction given.
- Step 3: Generate x,y,z coordinates inside cube of size L for center of sphere using RNG.
- Step 4: Check if the sphere at the random coordinate can be placed without intersection.
- Step 5: If sphere can be placed, store the coordinates, reset reject to 0 and go back to Step 3. Else go to Step 6.
- Step 6: Increase reject by 1. If reject = 10*N_{max} stop the program. Else go back to Step 3.

Code

Code for fitting the Spheres (MATLAB)

```
R = 1+rand;
L = 20*R;
R
Nmax = floor(L^3*0.74*3/(4*pi*R^3));
M = 10000;
N = zeros(M,1);
rmean = zeros(M,1);

for i = 1:M
    accept = 1;
    reject = 0;
```

```
    spheres = [[R+(L-2*R)*rand R+(L-2*R)*rand R+(L-2*R)*rand]];
    while(reject<10*Nmax)
        tmp=0;
        rm=0;
        x1 = R+(L-2*R)*rand;
        y1 = R+(L-2*R)*rand;
        z1 = R+(L-2*R)*rand;
        for j = 1:accept
            r = sqrt((x1-spheres(j,1))^2+(y1-spheres(j,2))^2+(z1-spheres(j,3))^2);
            if(r>=2*R)
                rm=rm+r;
            else
                tmp=1;
                break
            end
        end
        if tmp==0
            reject=0;
            accept=accept+1;
            rmean(i) = rmean(i)+rm;
            spheres(accept,1) = x1;
            spheres(accept,2) = y1;
            spheres(accept,3) = z1;
        else
            reject=reject+1;
        end
    end
    N(i) = accept;
    rmean(i) = 2*rmean(i)/(N(i)*(N(i)-1));
end

figure(4)
hold on;
[x, y, z] = sphere;
x = x*R;
y = y*R;
z = z*R;
light
lighting gouraud

plot3([0, L],[0, 0],[0, 0], 'k');
plot3([0, L],[L, L],[0, 0], 'k');
plot3([0, L],[0, 0],[L, L], 'k');
plot3([0, L],[L, L],[L, L], 'k');
plot3([0, 0],[0, L],[0, 0], 'k');
plot3([L, L],[0, L],[0, 0], 'k');
plot3([0, 0],[0, L],[L, L], 'k');
plot3([L, L],[0, L],[L, L], 'k');
plot3([0, 0],[0, 0],[0, L], 'k');
plot3([L, L],[L, L],[0, L], 'k');
plot3([0, 0],[L, L],[0, L], 'k');
plot3([L, L],[0, 0],[0, L], 'k');
xlabel('x');
ylabel('y');
zlabel('z');


for count = 1:accept
    surf(x+spheres(count,1), y+spheres(count,2), z+spheres(count,3),'EdgeColor','none'
    ,'Facecolor','r')
end
annotation('textbox',[0.6, 0.85, 0.1, 0.1] ,'String', "L = " + L + "; r = " + R)

view([30,30])

writematrix([N rmean], 'Data1.csv')
```

Code for Plotting (Python)

```
import numpy as np
import matplotlib.pyplot as plt

arr = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment 1\Data.csv",delimiter=",", dtype=float)
print('Mean of Average Spacing = ', np.mean(arr[:,1]))
print('Standard Deviation of Average Spacing = ', np.std(arr[:,1]))
print('Mean of Number of Spheres = ', np.mean(arr[:,0]))
print('Standard Deviation of Number of Spheres = ', np.std(arr[:,0]))

hist1, bins1 = np.histogram(arr[:,1], bins=100)
center = (bins1[:-1] + bins1[1:]) / 2
plt.bar(center, hist1, align='center')
plt.title('Frequency of Average Spacing')
plt.ylabel('Frequency')
plt.xlabel('Average Spacing')
plt.show()

hist2, bins2 = np.histogram(arr[:,0], bins=100)
center = (bins2[:-1] + bins2[1:]) / 2
plt.bar(center, hist2, align='center')
plt.title('Frequency of Number of Spheres')
plt.ylabel('Frequency')
plt.xlabel('Number of Spheres')
plt.show()

arr = arr[arr[:,1].argsort()]
plt.scatter(arr[:,1], arr[:,0], s=1)
plt.title('Average Spacing vs. Number of Spheres')
plt.xlabel('Average Spacing')
plt.ylabel('Number of Spheres')
plt.show()
```

Results

Frequency Distribution

Number of Spheres

Mean of Distribution = 574.1618
Standard Deviation of Distribution = 9.8368

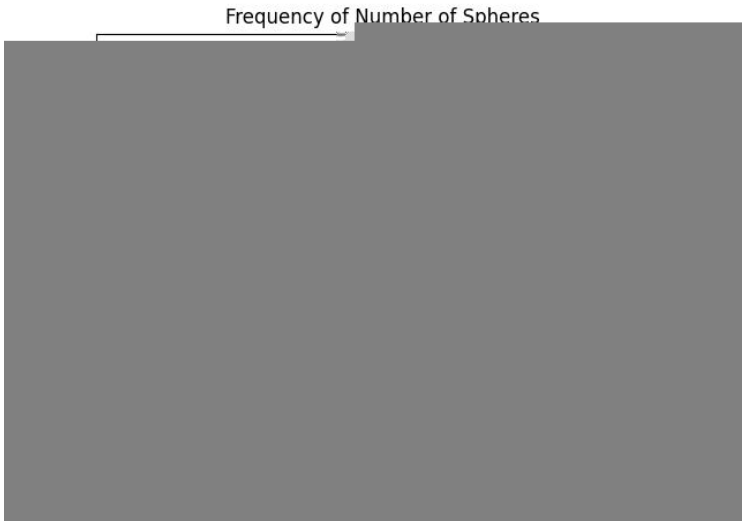

Figure 1: Frequency histogram for Number of Spheres with 100 bins

Average Spacing between Spheres

Mean of Distribution = 13.5449
Standard Deviation of Distribution = 0.0447

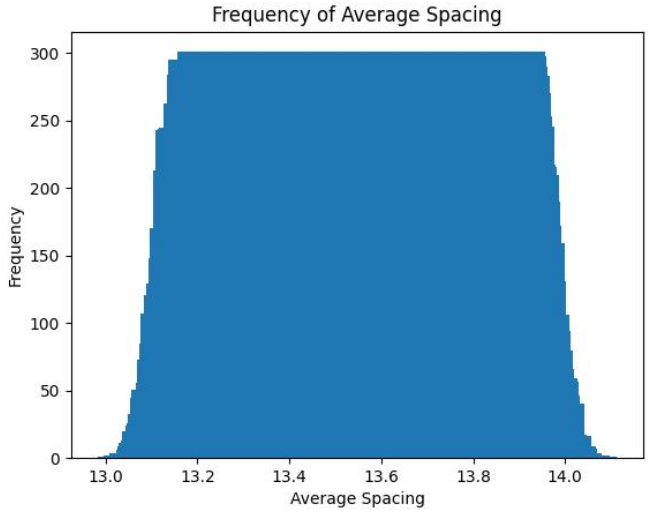

Figure 2: Frequency histogram for Average Spacing with 100 bins

Average Spacing vs. Number of Spheres

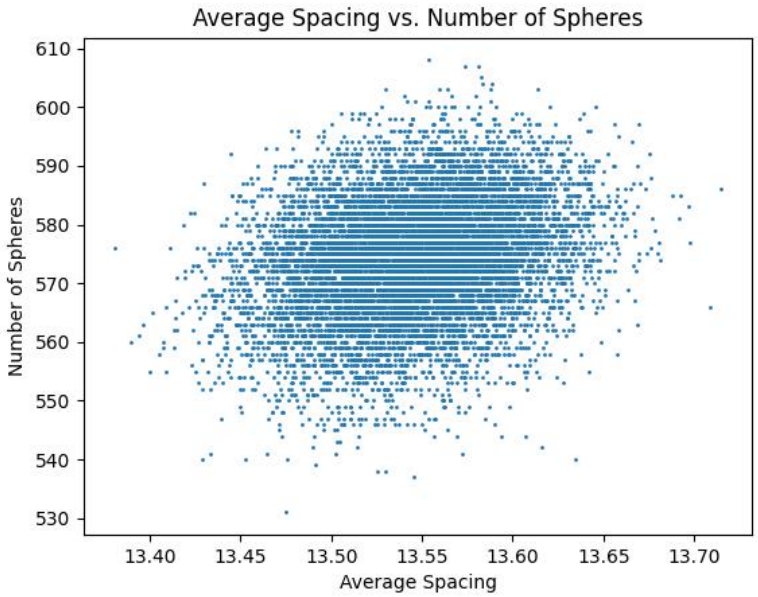

Figure 3: Plot for Average Spacing (r_{mean}) vs Number of Spheres (N)

3D Realization

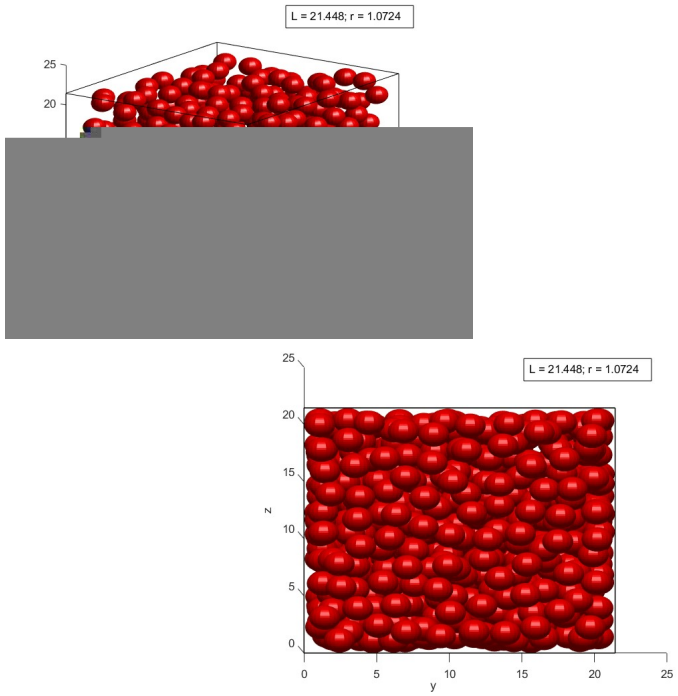

Figure 4: The 3D realization for r = 1.0724 and L = 21.448

Other Trends

Average Spacing vs. Radius

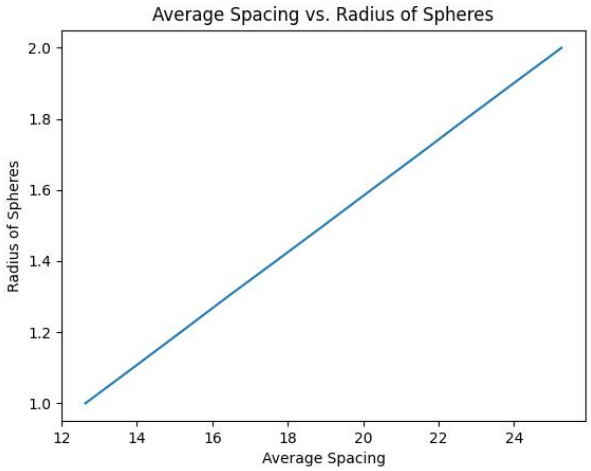

Figure 5: The trend for average spacing(r_{mean}) vs radius of sphere(R) over 100 realizations

Consecutive Failures vs. Number of Spheres

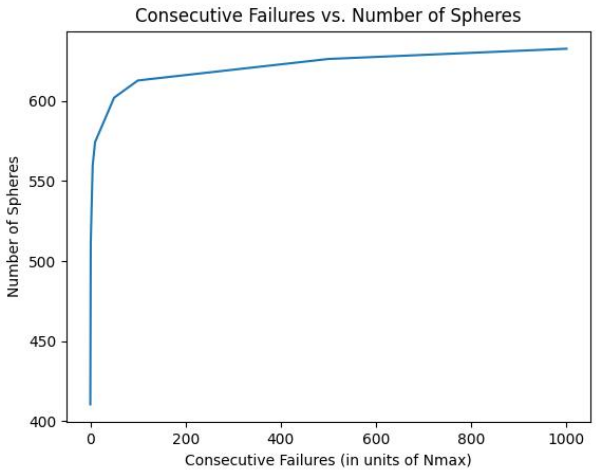

Figure 6: The trend for consecutive failures vs average number of spheres over 100 iterations

Discussion

- The maximum number of spheres that can be placed inside the cube depends only on the packing fraction and is independent of the radius of the spheres. This is because N_{max} depends on the ratio (L^3/R^3) and it is constant on our case. For pf = 0.74, N_{max} comes out to be 1413.
- The average number of spheres that is placed inside the cube is much lesser the value of N_{max}. This is expected as the coordinates for the center of the sphere are generated randomly. This also shows that for crystalline structures are able to have such high packing fractions only because they are ordered.
- The average number of spheres placed can also be increased by increasing the number of consecutive failures. In our case (10*N_{max}), the average value is around 574. By increasing it to a higher value, we will be able to place more spheres. From the trend, we can see that even for 1000*N_{max} iterations, the number of spheres is less than 650. Thus, using random coordinates, the number of spheres that can be place saturates at a value much lesser than N_{max}.
- The average spacing between the spheres, unlike number of spheres, is dependent on the value of R. For larger R values, the r_{mean} is larger (i.e Linear dependence). This is also totally intuitive as the spacing between spheres increases with increase in R and as size of the cube increases. In our case (R=1.0724), the average distance between spheres comes out to be 13.545. Due to the linear dependence, the value of r_{mean}/R is almost constant and comes out to be around 12.63.
- It can be inferred from the frequency distribution plot of N and r_{mean} that they follow normal distribution. This is expected as suggested by the Central Limit Theorem that states that the distribution of sample means approximates a normal distribution as the sample size gets larger, regardless of the population's distribution.
- The scatter plot for N vs r_{mean} shows that the center of the distribution is more densely populated, and the density of points decreases as we go outwards. Since both N and r_{mean} follow the normal distribution, it is not very surprising that the plot shows such a pattern.