

1 Exercise 4

In this exercise we have a couple of assignment that have to be done. The web app used for plotting is found at <https://webspace.science.uu.nl/~herme107/viscol/>.

1.1 4.a)

Here we had to make code that tiled the space with spheres in a cubic lattice formation. The code for the generation of this lattice is found in Appendix A.1.

On the web app for plotting this lattice, Figure 1 was made using the file that was generated.

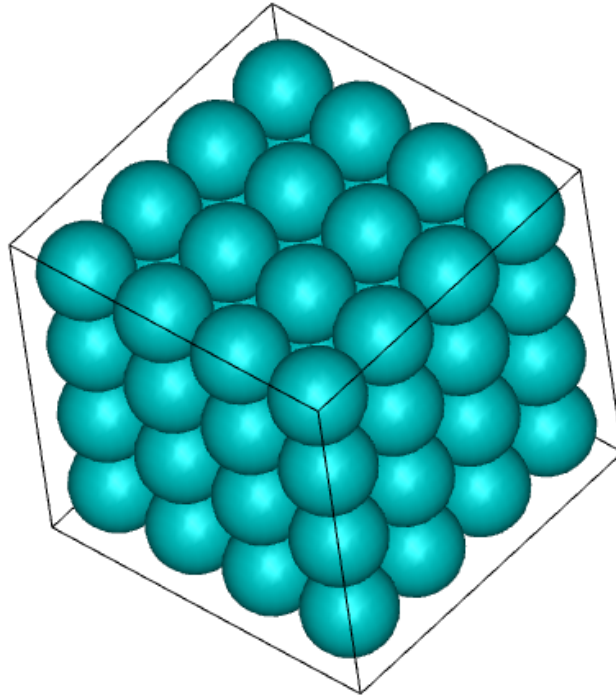


Figure 1: Here the cubic lattice generated by the code is graphed

1.2 4.b)

We want to know the maximum packing density for spheres in a cubic lattice.

1.3 4.c)

Here we had to make code that tiled a space with spheres in a face-centered cubic (FCC) lattice. The code for the generation of this lattice is found in Appendix A.2.

On the web app for plotting this lattice, Figure 2 was made using the file that was generated.

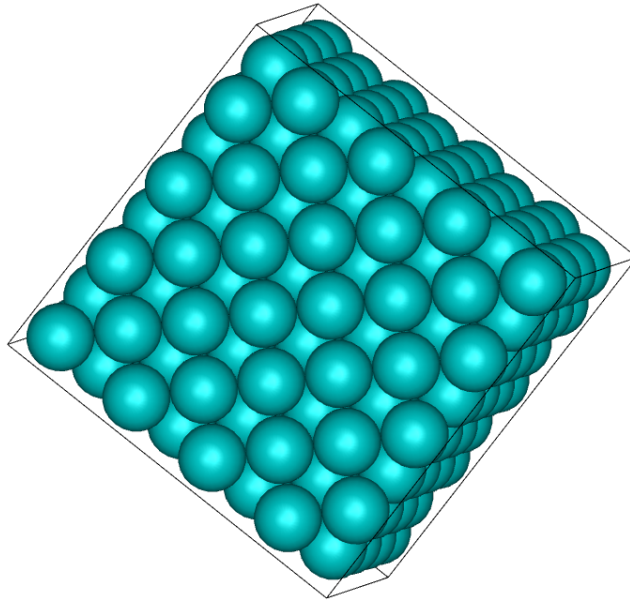


Figure 2: Here the cubic lattice generated by the code is graphed

A Code Exercise 4

A.1 Code 4.a)

```
1 #include <stdio.h>
2 #include <math.h>
3 // in this file we will make the cubic latice
4
5 int main(){
6     int N = 4; // The number of particles in each dirrection
7     float d = 1.0; // the distance between two spheres
8     float a = 1.0; // the radius of an sphere
9
10    // Make a file where we can save the position data
11    FILE *print_coords; // inititilises a file variable
12    print_coords = fopen("cubic_xyz.dat","w"); // defining the file variable to be the opening
        of some file cubic.xyz
13
14    // Let us print some initial coordinates
15    fprintf(print_coords, "%i\n", N*N*N); // the total number of particles
16    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The ocupied space in the x direction
17    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The ocupied space in the y direction
18    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The ocupied space in the z direction
19
20    // we first initialise the particle possision saving arrays
21    float x[N*N*N], y[N*N*N], z[N*N*N], r[N*N*N];
22
23
24    // now we start generating particle possitions and radiuses
25    int n = 0; // this is our counting variable, it wil index which particle we will consider
26
27    /*
28    The latice points are described by
29    R= a_x n_x + a_y n_x + a_z n_z
30    a_x = i a
31    a_y = j a
32    a_z = k a
33    */
34
35
36    // sweeping over the N_x particles
37    for(int i=0; i<N; i++){
38        // sweeping over the N_y particles
39        for(int j=0; j<N; j++){
40            // sweeping over the N_z particles
41            for(int k=0; k<N; k++){
42                // generating the possition for i,j,k latice cite, also the radius of the particle
43                x[n]= (i+0.5)*d;
44                y[n]= (j+0.5)*d;
45                z[n]= (k+0.5)*d;
46                r[n]= a;
47
48                // saving the x,y,z possition and radius of the particle
49                fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
50
51                n++;
52            }
53        }
54    }
55
56    fclose(print_coords);
57    return 0;
58 }
59 }
```

A.2 Code 4.b)

```
1 #include <stdio.h>
2 #include <math.h>
3 // in this file we will make the cubic latice
4
5 int main(){
6     int N = 4; // The number of particles in each dirrection
7     float d = 1.0; // the distance between two spheres
8     float a = 1.0; // the radius of an sphere
9 }
```

```

10 // creating an distance variable that makes les typing
11 float l = sqrt(2.0)*d;
12
13 // defining the size of the box that will be spanned
14 float x_max = N*l;
15
16 // definging a variable such that the outline of the box aligns with the border of the
   particles
17 float s = 0.5*d;
18
19 // Make a file where we can save the position data
20 FILE *print_coords; // initialises a file variable
21 print_coords = fopen("FCC_xyz.dat","w"); // defining the file variable to be the opening of
   some file cubic.xyz
22
23 // Let us print some initial coordinates
24 fprintf(print_coords, "%i\n", 4*N*N*N); // the total number of particles
25 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The ocupied space in the
   x direction
26 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The ocupied space in the
   y direction
27 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The ocupied space in the
   z direction
28
29 // we first initialise the particle possision saving arrays
30 float x[4*N*N*N], y[4*N*N*N], z[4*N*N*N], r[4*N*N*N];
31
32 // now we start generating particle possitions and radiuses
33 int n = 0; // this is our counting variable, it wil index which particle we will consider
34
35 /*
36 The lattice points are described by
37  $R = a_1 n_x + a_2 n_y + a_3 n_z$ 
38  $a_1 = a/2 (j + k)$ 
39  $a_2 = a/2 (i + k)$ 
40  $a_3 = a/2 (i + j)$ 
41  $i, j, k$  are the unit vectors in x, y and z directions respectively (not the counts)
42 */
43
44
45
46 // sweeping over the N_x particles
47 for(int i=0; i<N; i++){
48     // sweeping over the N_y particles
49     for(int j=0; j<N; j++){
50         // sweeping over the N_z particles
51         for(int k=0; k<N; k++){
52             // generating the possition for i,j,k lattice cite, also the radius of the particle
53
54             // first we start on the base vector because we know this pattern reapeats every 2*unit
   vector in each direction
55             x[n]= (i)*l;
56             y[n]= (j)*l;
57             z[n]= (k)*l;
58             r[n]= a;
59
60             // saving the x,y,z possition and radius of the particle
61             fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
62
63             n++;
64
65             // here we will add the a_1 vector and make the same spacing
66             x[n]= (i)*l;
67             y[n]= (j+0.5)*l;
68             z[n]= (k+0.5)*l;
69             r[n]= a;
70
71             // saving the x,y,z possition and radius of the particle
72             fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
73
74             n++;
75
76             // here we will add the a_2 vector and make the same spacing
77             x[n]= (i+0.5)*l;
78             y[n]= (j)*l;
79             z[n]= (k+0.5)*l;

```

```

80     r[n]= a;
81
82     // saving the x,y,z position and radius of the particle
83     fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
84
85     n++;
86
87     // here we will add the a3 vector and continue the same spacing
88     x[n]= (i+0.5)*1;
89     y[n]= (j+0.5)*1;
90     z[n]= (k)*1;
91     r[n]= a;
92
93     // saving the x,y,z position and radius of the particle
94     fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
95
96     n++;
97
98
99
100
101 }
102 }
103 }
104 }
105
106
107 fclose(print_coords);
108 return 0;
109 }

```