

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 whit 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.

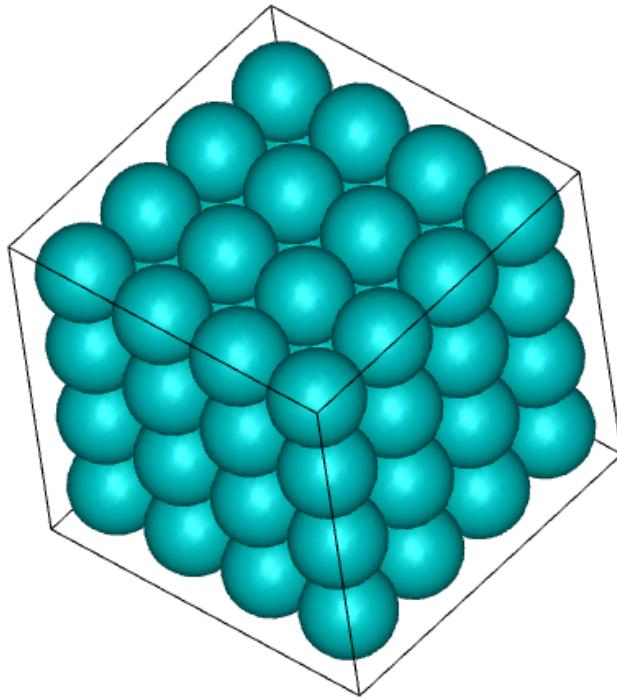


Figure 1: Here the cubic lattice generated by the code is graphed

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.

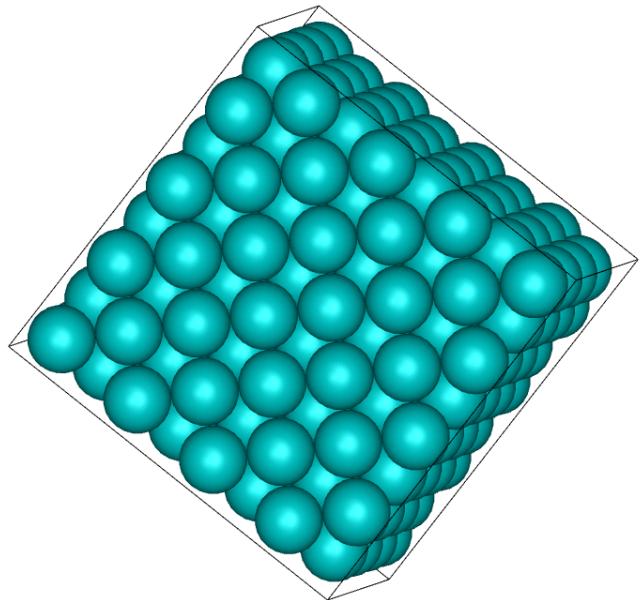


Figure 2: Here the cubic lattice generated by the code is graphed

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 lattice
4
5 int main(){
6     int N = 4; // The number of particles in each direction
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; // initialises a file variable
12    print_coords = fopen("cubic_xyz.dat","w"); // defining the file variable to be the opening
13        of some file cubic.xyz
14
15    // Let us print some initial coordinates
16    fprintf(print_coords, "%i\n", N*N*N); // the total number of particles
17    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The occupied space in the x direction
18    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The occupied space in the y direction
19    fprintf(print_coords, "%lf\t%lf\n", -0.0, 1.0*d*N); // The occupied space in the z direction
20
21    // we first initialise the particle possision saving arrays
22    float x[N*N*N], y[N*N*N], z[N*N*N], r[N*N*N];
23
24
25    // now we start generating particle possisions and radiuses
26    int n = 0; // this is our counting variable, it wil index which particle we will consider
27
28    /*
29    The lattice points are described by
30    R= a_x n_x + a_y n_y + a_z n_z
31    a_x = i a
32    a_y = j a
33    a_z = k a
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++){
30            // sweeping over the N_z particles
41            for(int k=0; k<N; k++){
42                // generating the possition for i,j,k lattice 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
57    fclose(print_coords);
58    return 0;
59 }
```

A.2 Code 4.b)

```
1 #include <stdio.h>
2 #include <math.h>
3 // in this file we will make the cubic lattice
4
5 int main(){
6     int N = 4; // The number of particles in each direction
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 less 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 // defining a variable such that the outline of the box aligns with the border of the
17 // particles
18 float s = 0.5*d;
19
20 // Make a file where we can save the position data
21 FILE *print_coords; // initialises a file variable
22 print_coords = fopen("FCC_xyz.dat","w"); // defining the file variable to be the opening of
23 // some file cubic.xyz
24
25 // Let us print some initial coordinates
26 fprintf(print_coords, "%i\n", 4*N*N*N); // the total number of particles
27 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The occupied space in the
28 // x direction
29 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The occupied space in the
30 // y direction
31 fprintf(print_coords, "%lf\t%lf\n", -s, x_max-sqrt(2)*s+0.5*d); // The occupied space in the
32 // z direction
33
34 // we first initialise the particle possision saving arrays
35 float x[4*N*N*N], y[4*N*N*N], z[4*N*N*N], r[4*N*N*N];
36
37 // now we start generating particle possisions and radiuses
38 int n = 0; // this is our counting variable, it wil index which particle we will consider
39
40 /*
41 The lattice points are described by
42 R= a_1 n_x + a_2 n_y + a_3 n_z
43 a_1 = a/2 (j + k)
44 a_2 = a/2 (i + k)
45 a_3 = a/2 (i + j)
46 i, j, k are the unit vectors in x, y and z directions respectively (not the counts)
47 */
48
49
50 // sweeping over the N_x particles
51 for(int i=0; i<N; i++){
52     // sweeping over the N_y particles
53     for(int j=0; j<N; j++){
54         // sweeping over the N_z particles
55         for(int k=0; k<N; k++){
56             // generating the possition for i,j,k lattice cite, also the radius of the particle
57
58             // first we start on the base vector because we know this patern reapeats every 2*unit
59             // vector in each direction
60             x[n]=(i)*l;
61             y[n]=(j)*l;
62             z[n]=(k)*l;
63             r[n]= a;
64
65             // saving the x,y,z possition and radius of the particle
66             fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
67
68             n++;
69
70             // here we will add the a_1 vector and make the same spacing
71             x[n]=(i)*l;
72             y[n]=(j+0.5)*l;
73             z[n]=(k+0.5)*l;
74             r[n]= a;
75
76             // saving the x,y,z possition and radius of the particle
77             fprintf(print_coords, "%lf\t%lf\t%lf\t%lf\n", x[n], y[n],z[n],r[n]);
78
79             n++;
80
81             // here we will add the a_2 vector and make the same spacing
82             x[n]=(i+0.5)*l;
83             y[n]=(j)*l;
84             z[n]=(k+0.5)*l;

```

```

80     r[n]= a;
81
82     // saving the x,y,z possition 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)*l;
89     y[n]= (j+0.5)*l;
90     z[n]= (k)*l;
91     r[n]= a;
92
93     // saving the x,y,z possition 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 }
```