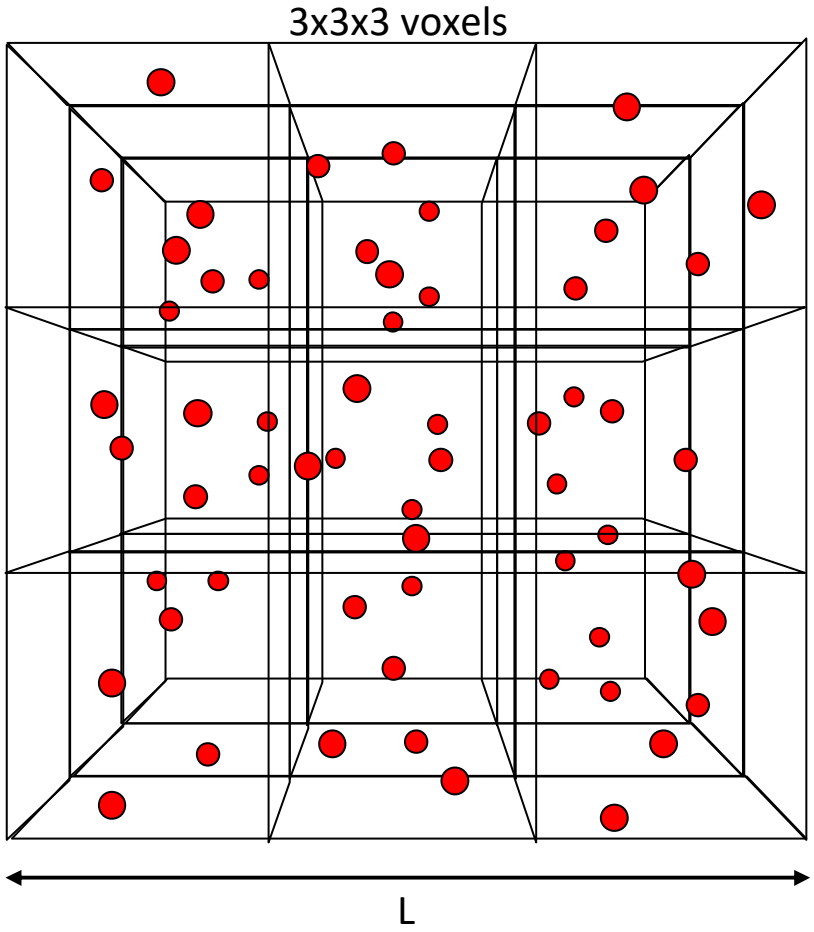
Exercise: training with Matlab – Part 1



Develop an algorithm that counts the number of particles within each voxel of the MxMxM voxels forming a cubic box of side L. Each voxel will have volume (L/M)x(L/M)x(L/M).

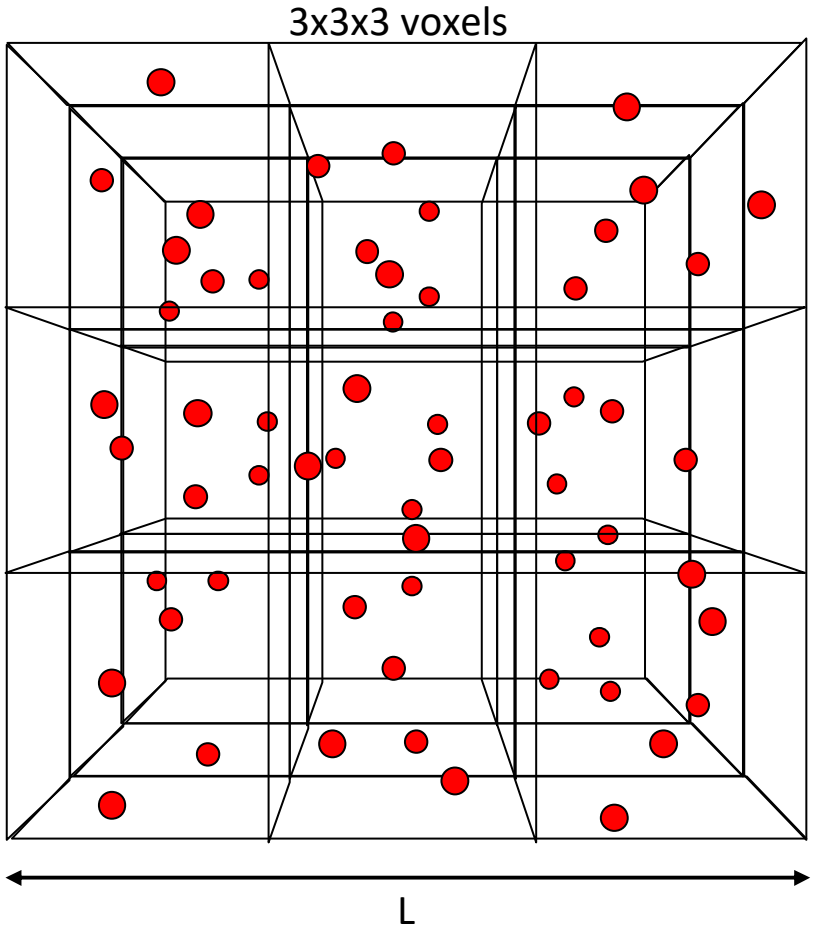
The function will be defined as follows:

```
function C=CountParticles(pos,L,M)

%Syntax C=CountParticles(pos,L,M);
%
%Input:
%pos is a Nx3 matrix containing the positions of N
%particles randomly distributed within the box.
%
%L is the box side.
%
%M is the number of voxels along one dimension.
%
%Output:
%C is a MxMxM matrix having the (i,j,k)-th element
%equal to the number of particles within the (i,j,k)-
%th voxel of the box.
```

If the box contains 10^6 particles distributed in 10^6 voxels, a fast algorithm should perform the computation in less than 0.1 second by a i7 Intel CPU. Note: it's not only a matter of speeding up your code, verify if your algorithm performs a correct particle counting (e.g. by simulating a few particles).

Exercise: training with Matlab – Part 2



Rename CountParticles function by including a second output:

```
function [C, Npos]=CountParticles2(pos,L,M)

%Syntax [C, Npos]=CountParticles2(pos,L,M);
%
%Input:
%pos is a Nx3 matrix containing the positions of N
%particles randomly distributed within the box.
%
%L is the box side.
%
%M is the number of voxels along one dimension.
%
%Output:
%C is a MxMxM matrix having the (i,j,k)-th element
%equal to the number of particles within the (i,j,k)-
%th voxel of the box.
%
%Npos is a Nx1 array having the i-th element equal
%to the number of particles within the voxel of the
%i-th particle in pos matrix.
```

If the box contains 10^6 particles distributed in 10^6 voxels, a fast algorithm should perform the computation in less than 1 second by a i7 Intel CPU. Suggestion: try to use built-in Matlab functions optimized for matrix computation.