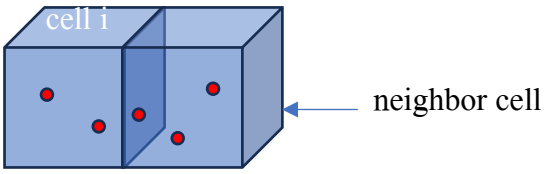
Possible approaches:

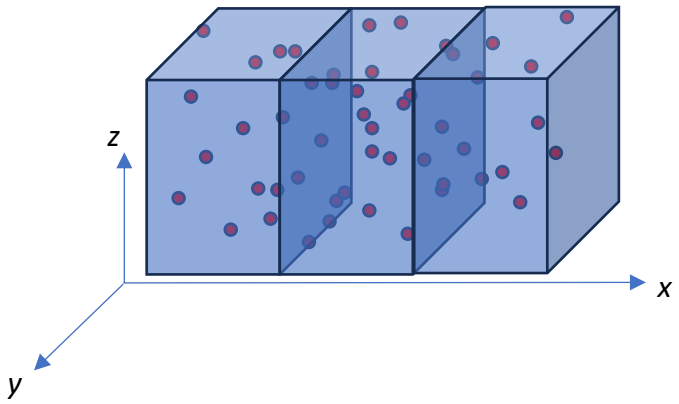
- The “brute force” approach, with a double nested loop on the point set, is not considered feasible.

- One can think of subdividing the space in cubes and creating a hash table that assigns points to cells. Then, one could search in parallel in each cell the pair of points located at a distance smaller than a given threshold. The neighbors’ cells must be explored as well, since some points might be located at a distance that is very close to the cell considered. This is shown in the scheme below. “Cell i” is the current cell.

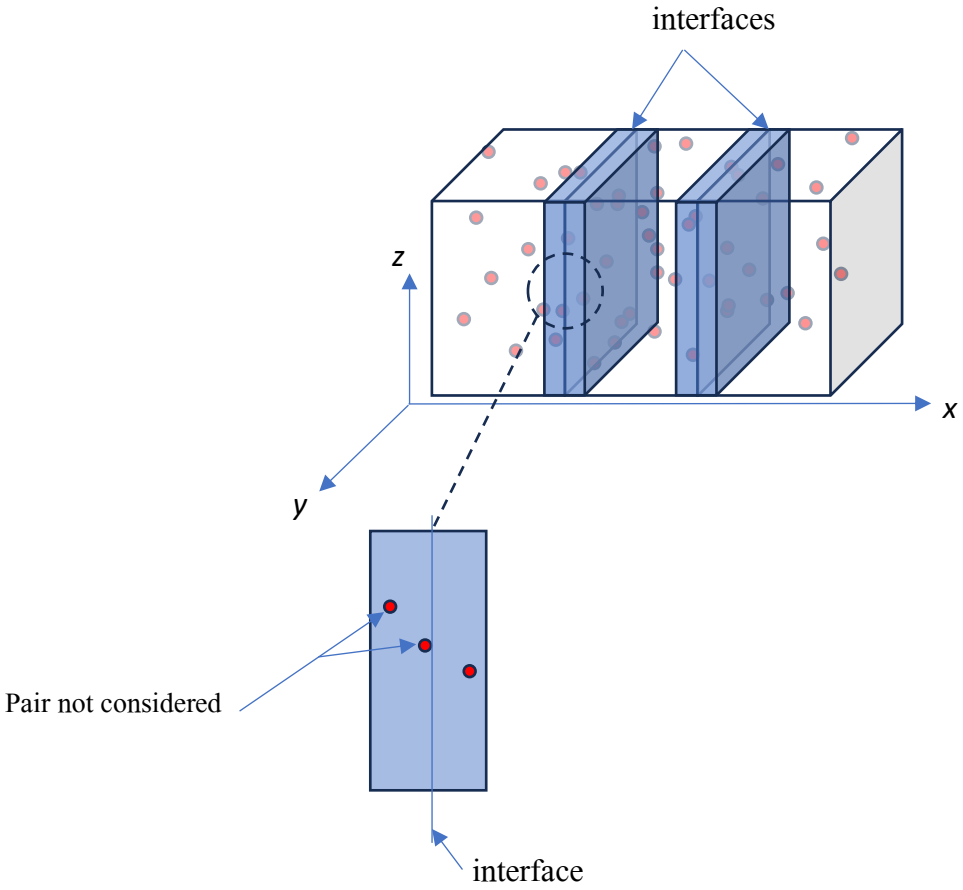


- Another option is to use kd-tree data structure that allow for efficient search of the nearest neighbor. This is what I am going to implement.

In any case the number of points (10^7) is too large. Therefore, I am going to split the set of points in chunks along the x-direction. Then I analyze “chunk by chunk”. The image below represents 3 chunks of points enclosed by blue boxes.



Finally, I consider each interface between the chunks (see figure below for the interface definition). I define a small box across each of these interfaces. Within each box, I will find all the pairs of points at a distance smaller than the assigned distance. Then I will filter the result to remove the repeated pairs of points. The filter allows to keep only the pairs involving points located on both sides of the interface. The figure below shows the interfaces between the original chunks and the “volume of points” (blue color) considered for the new pair search.


interfaces
z
x
y
Pair not considered
interface