Spatial Boundary Conditions

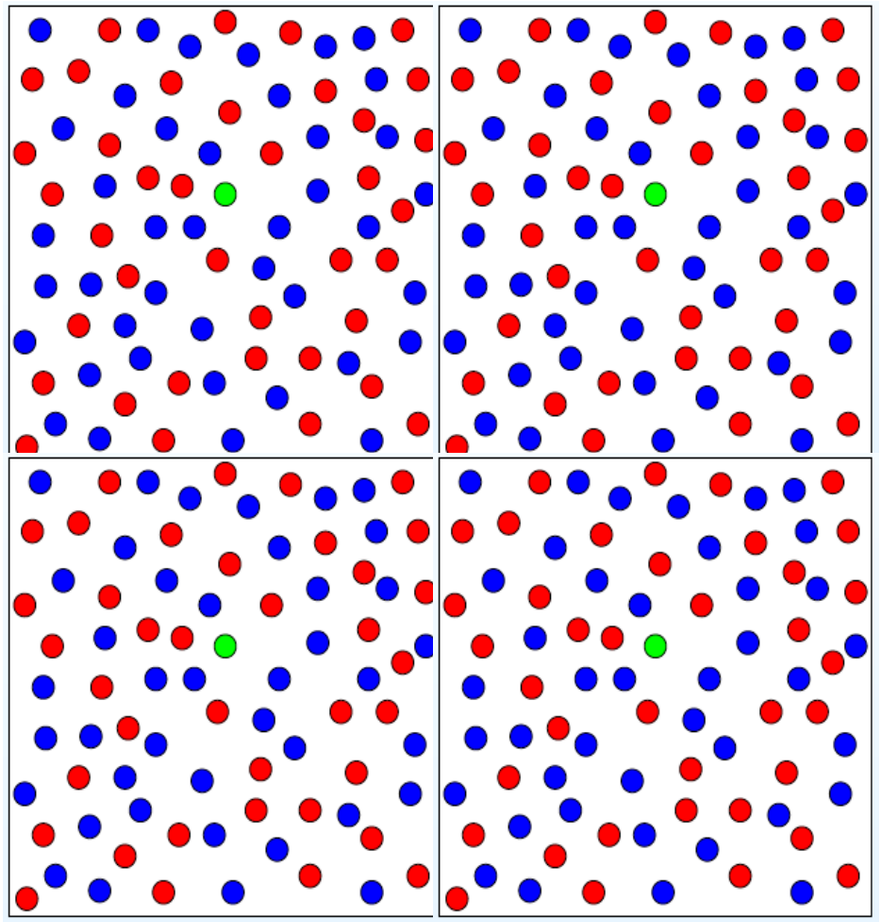
Important because spatial scales are limited.

What BC can we choose? There are different possibilities:

- **Free boundaries**: e.g. droplet, protein in vacuum.
  - If droplet has 1 million = (100^3) atoms and surface layer is 5 atoms thick, then *25% of atoms are on the surface*.
- **Periodic boundaries**: most popular choice because there are no surfaces (see next slide) but there can still be problems.
- Simulations on a sphere
- External potentials
- Mixed boundaries (e.g. free in z, periodic in x and y)
- **Coupled boundaries**: "embed" simulation in another through additional forces, constraints, displacements...

Why use Periodic Boundary Conditions?

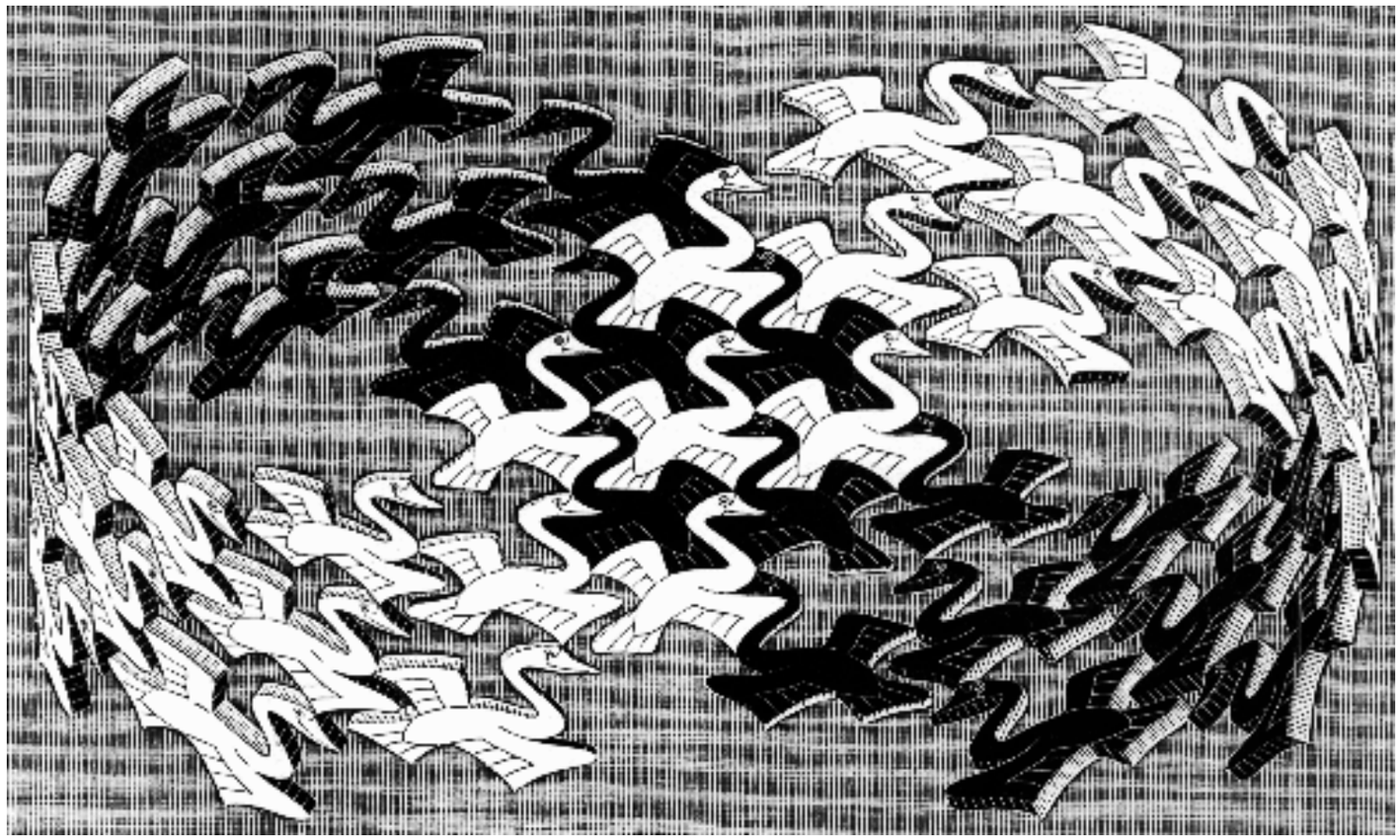
- Macroscopic systems $O(10^{23})$ particles.
- We eliminate surfaces from simulation.
- Allows us to get at bulk properties with few particles.
- Applied to solids, liquids, gases, and plasmas.
- Some finite-size effects remain, but can often be removed with scaling.



Periodic Boundary Conditions

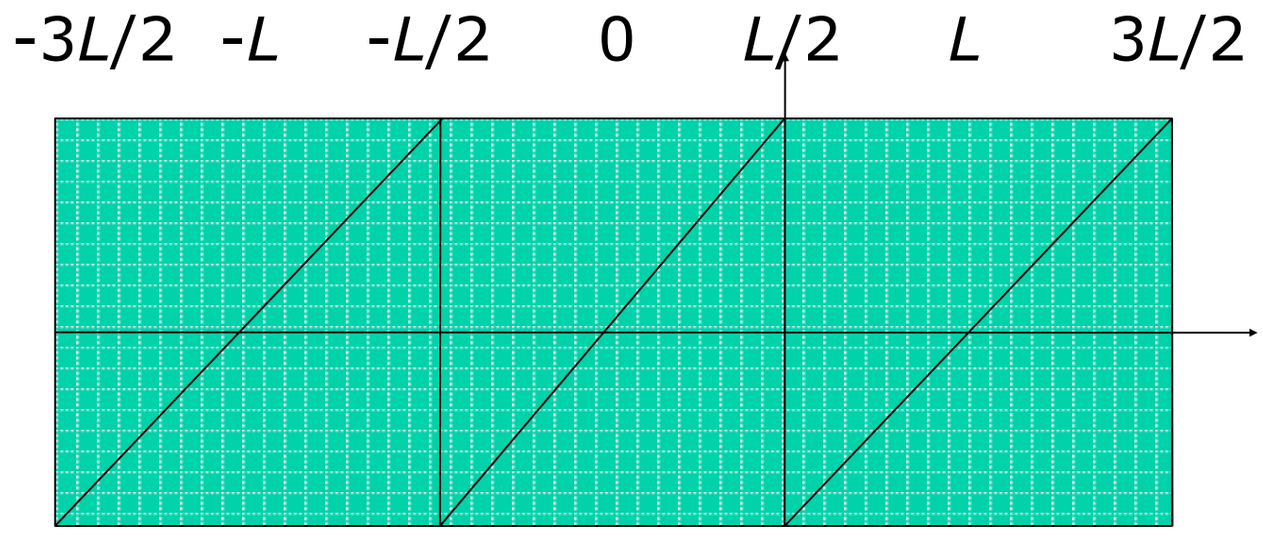
Is the system periodic or just an infinite array of images?

How do you calculate distances in periodic system?



Periodic distances

Minimum Image Convention: Take the closest distance $|r|_M = \min(r+nL)$



How to handle the "tail" of the potential:

- Potential cutoff :

$$V_c(r) = 0 \text{ for } r > r_c$$

  - continuous potential: $V_c(r) = V(r) - V(r_c) \text{ for } r \le r_c$
  - continuous pot. and F: $V_c(r) = V(r) - V(r_c) - \nabla V(r_c)\cdot(r - r_c) \text{ for } r \le r_c$

- Image potential:

$$V_{\text{image}}(r_i - r_j) = \sum_n V(r_i - r_j + nL)$$

For long-range potential this leads to the Ewald image potential.

You need a charge background and convergence factor (more later)

Lennard-Jones Force calculation

```
for i in range(Natoms):
  for j in range(i+1,Natoms):
    dx=x[i]-x[j]          # this will be a vector if x&y are array
    for d in range(3):    # more clever ways to do this?
      if dx[d]>L[d]/2:  dx[d] -= L[d]
      if dx[d]<-L[d]/2: dx[d] += L[d]
    r2 = sum(dx*dx)       # dx[0]*dx[0]+dx[1]*dx[1]+dx[2]*dx[2]
    if r2>rcutoff2: continue  # outside of cutoff distance^2
    r2i = sigma/r2
    r6i = r2i**3
    pot += eps4*r6i*(r6i-1) - potcut
    rforce = eps24*r6i*r2i*(2*r6i-1)
    F[i] = F[i] + rforce*dx # F[i] and dx are vectors!
    F[j] = F[j] - rforce*dx
```

We can do better with the PBC! rewrite as a remainder of division

Need to initialize forces and potential: `F.fill(0), pot=0`

Complexity of Force Calculations

- Complexity is defined as how a computer algorithm scales with the number of degrees of freedom (atoms).

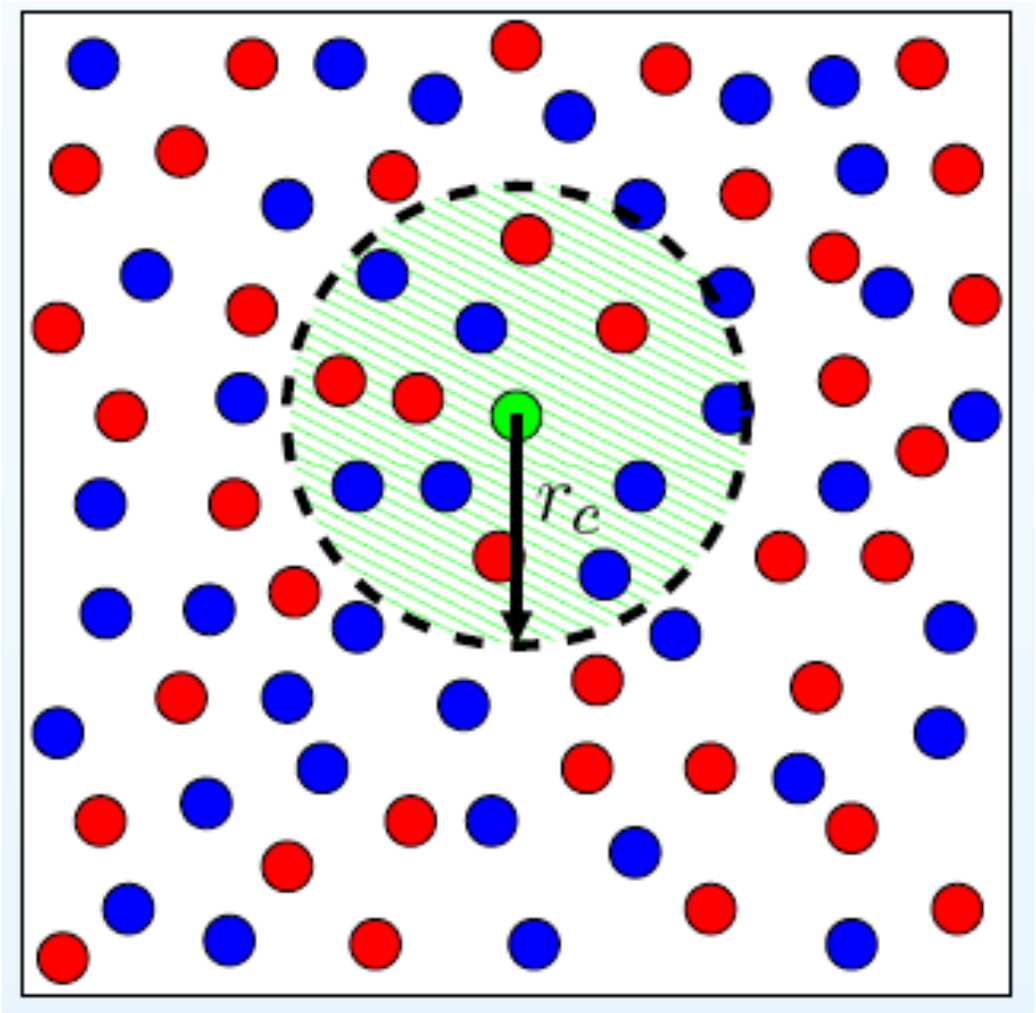
- Number of terms in pair potential is $N(N\text{-}1)/2 \approx O(N^2)$

- For short-range V(r), use neighbor tables to reduce it to $O(N)$.
  - Explicit Neighbor list (Verlet) for systems that move slowly (keep a list and update occassionally.)
  - Bin Sort list (sort particles into bins and sum over neighboring bins)

- Long-range potentials require more sophisticated treatment:
  - Ewald sums are $O(N^{3/2})$
  - Fast Multipole Algorithms are $O(N)$, for very large N.
  - Tree Methods
- More later…

Neighbor Lists: $O(N^2)$ to $O(N)$

- Pair Potentials: If $V_c(r) = 0$ for $r > r_c$ (i.e., short-ranged), then
  - Number of computations of the potential is mN, where m is average number of neighbors inside r_c and is independent of system size!
  - So, once we have the neighbor table, the calculation is $O(N)$!
- Constructing Neighbor Tables:
- $O(N^2)$, must check if particles fall inside each other's radius.
- only have to reconstruct the table occasionally.

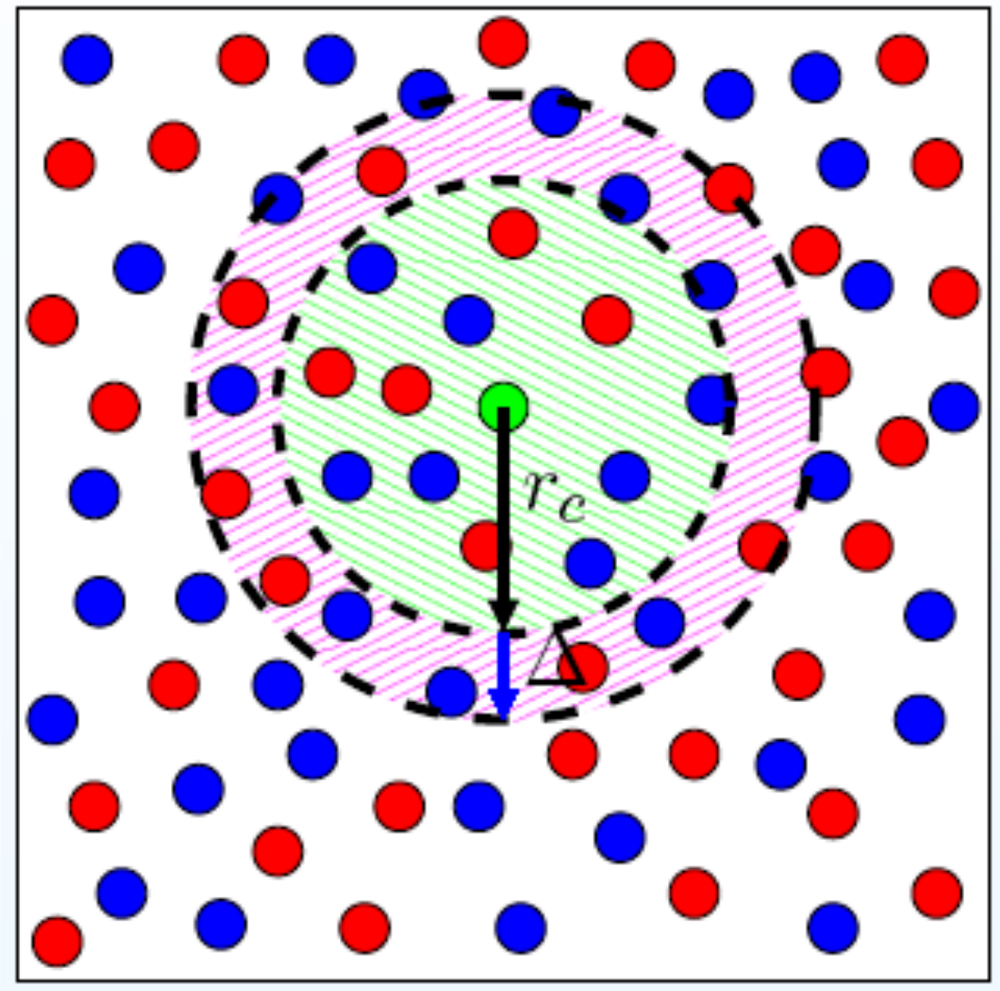


An example neighbor table

Neighbors \ Particles	1	2	3	4	5	6	. . .
	6	45	9	15	19	1	. . .
	34	57	16	43	31	3	. . .
	17	16	25	36	14	19	. . .
	29	10	61	13	59	40	. . .
	12	8	58	7	63	21	. . .
	18		12	27	26	64	. . .
	47		30	38		39	. . .
	19		21			23	. . .
			51			14	. . .

Maintaining neighbor Lists: use skin depth

- Cut off table at $r_c + \Delta$ (skin depth).
  - allows a particle to move a while before new table needed.
- Keep track of the two largest distances traveled, d_1 and d_2,
- when $d_1+d_2 \geq \Delta$ get new table.
- Choose Δ to optimize efficiency
  - As Δ decreases, fewer neighbors and force evaluations.
  - As Δ increases, need to calculate neighbor table less often. $O(N^2)$ operation.
  - Dynamically optimize Δ by fitting $t(\Delta)$ to a polynomial and minimizing.



Improvements: the Cell Method

- divide box into cells of size $L > r_c + \Delta$.
  - Need only particle's own and neighboring cells.
- table construction is $O(N)$!
- To find neighbors use a linked list
- Memory is also N (not mN)