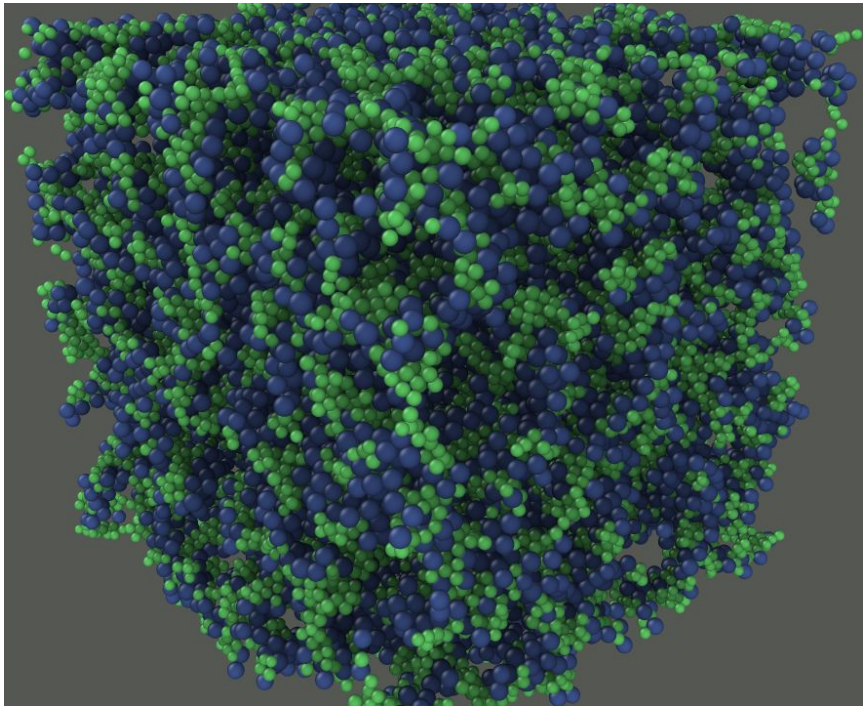
Simulation de particules sur GPU

B. Crespin - XLIM/ASALI/SIR

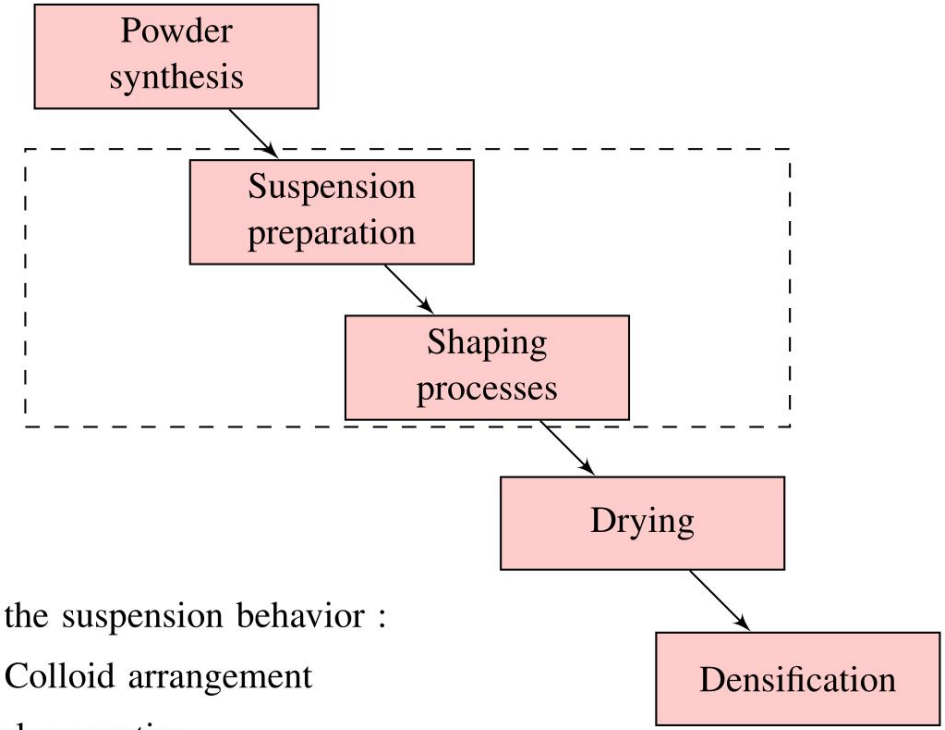
Journée Simulation Numérique et ses Applications
MIRES/MARGAUx
25 Septembre 2024

Simulation de particules sur GPU

- Simulation numérique de suspensions colloïdales
- Recherche de voisinage
- Parallélisation sur GPU et ajustement des tailles de noyaux CUDA
- Recherche de voisinage sur des triangulations de Delaunay



Suspensions colloïdales pour les matériaux céramiques



Control of the suspension behavior :

- Structure, Colloid arrangement
- Rheological properties

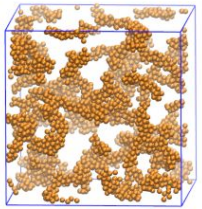
Simulation numérique de suspensions colloïdales

Brownian Dynamics (BD)

Fluid : continuous medium

$$m\frac{dv}{dt} = \sum F(r) - \zeta v + \Gamma(t)$$

Interactions between particles — Friction force — Random force



No hydrodynamic interactions

'Stochastic rotation dynamics-molecular dynamics' (SRD-MD)

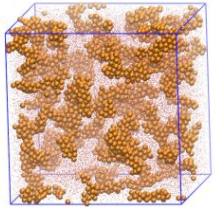
Fluid : particles

Streaming :

$$x(t+\delta t) = x(t) + v(t)\delta t$$

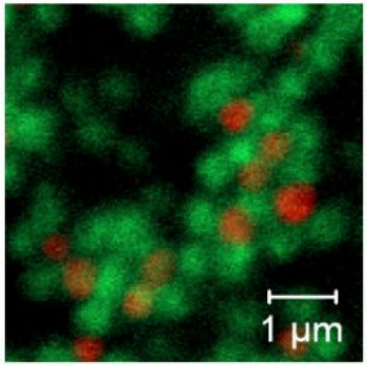
Collision :

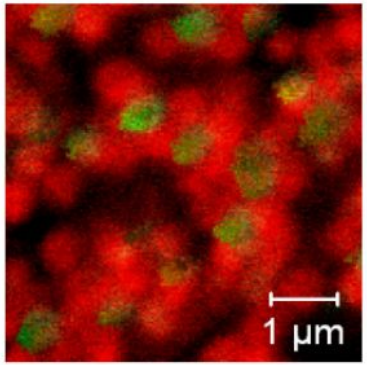
$$v(t+\delta t) = v_{cm}(t) + R(v(t) - v_{cm})$$

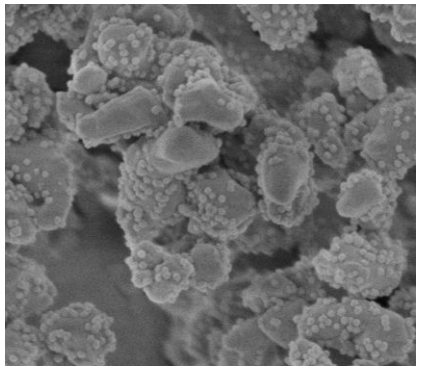


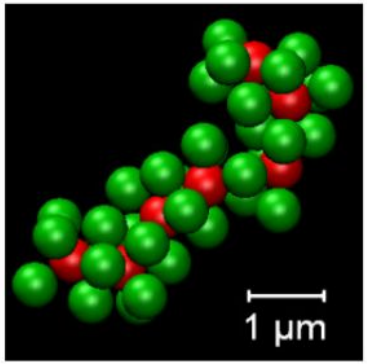
Good description of the hydrodynamic interactions

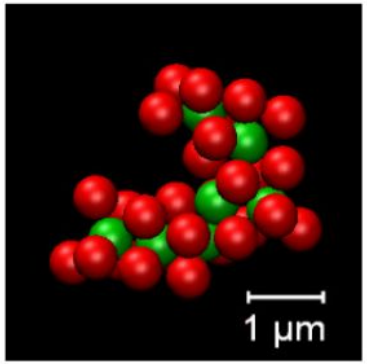
Simulation numérique de suspensions colloïdales

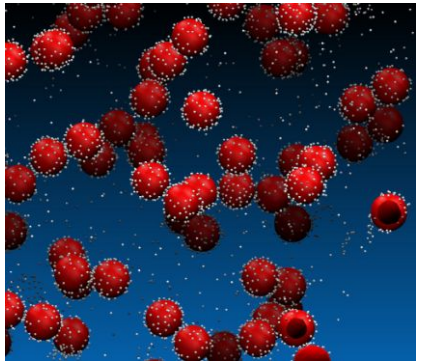




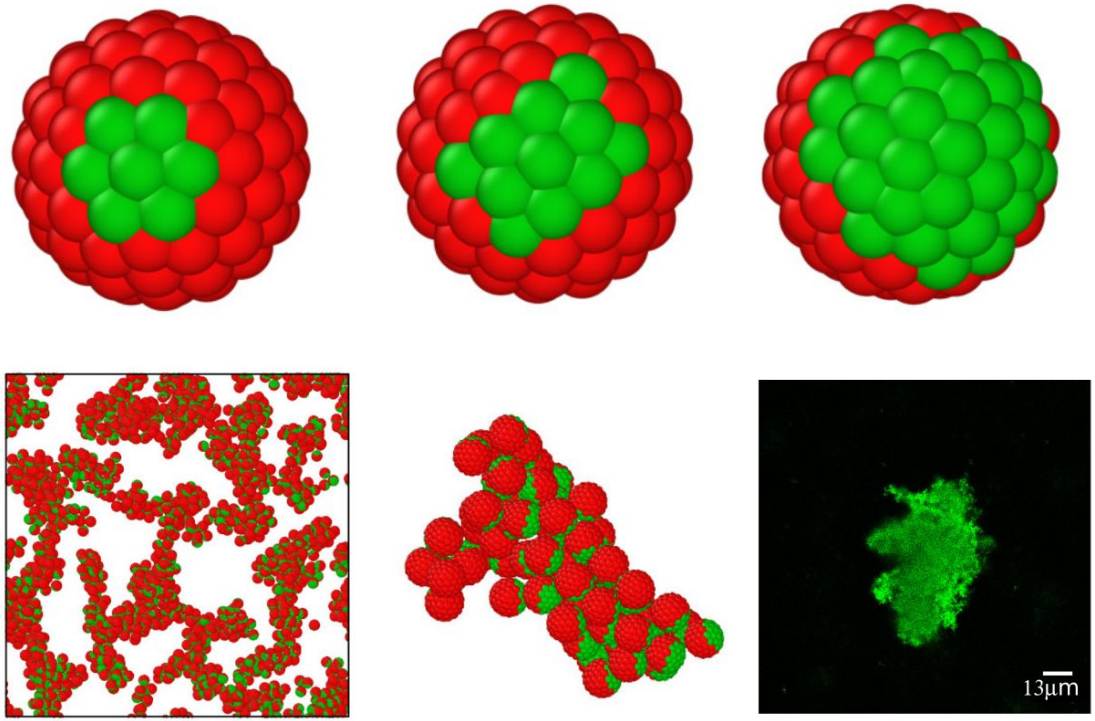




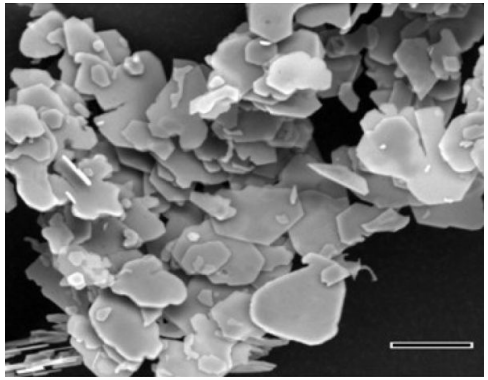


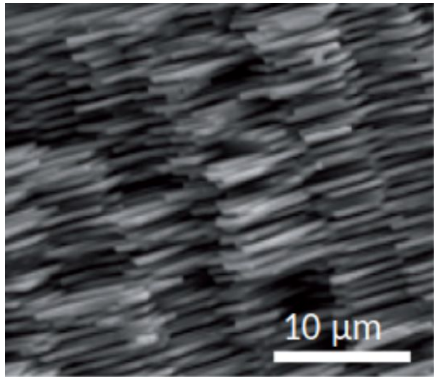


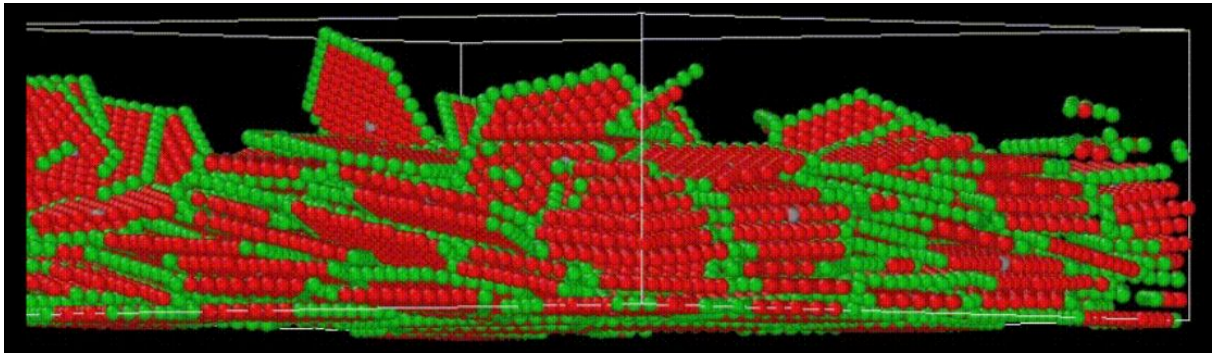
Simulation numérique de suspensions colloïdales

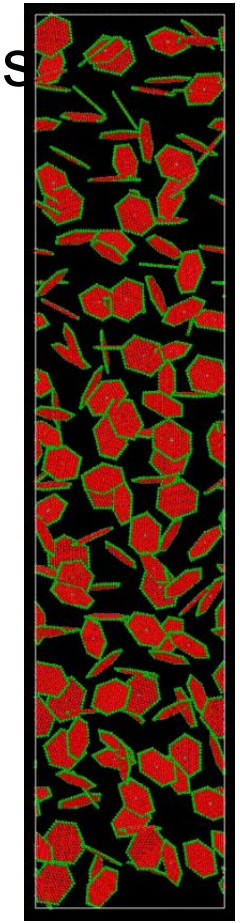


Simulation numérique de suspensions colloïdales

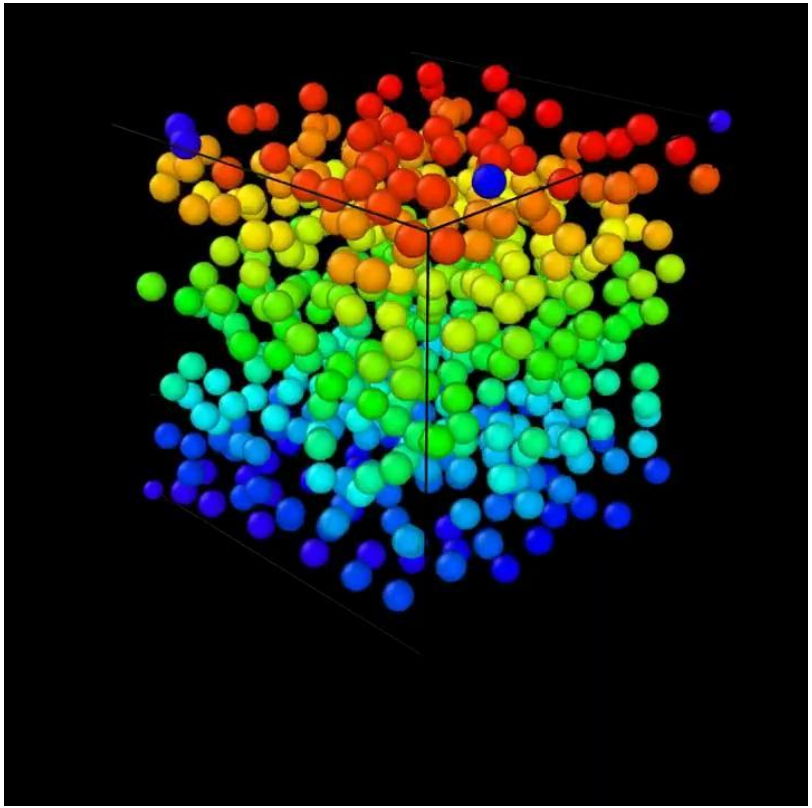




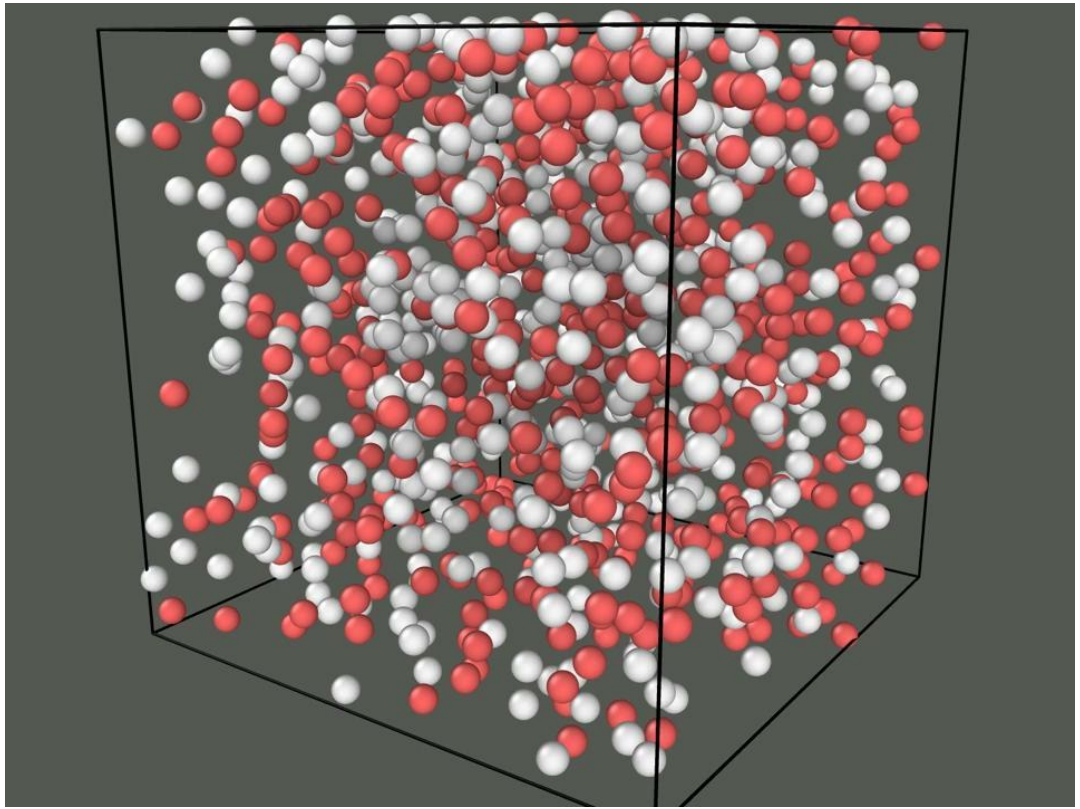




Simulation numérique de suspensions colloïdales



Simulation numérique de suspensions colloïdales



Simulation numérique de suspensions colloïdales

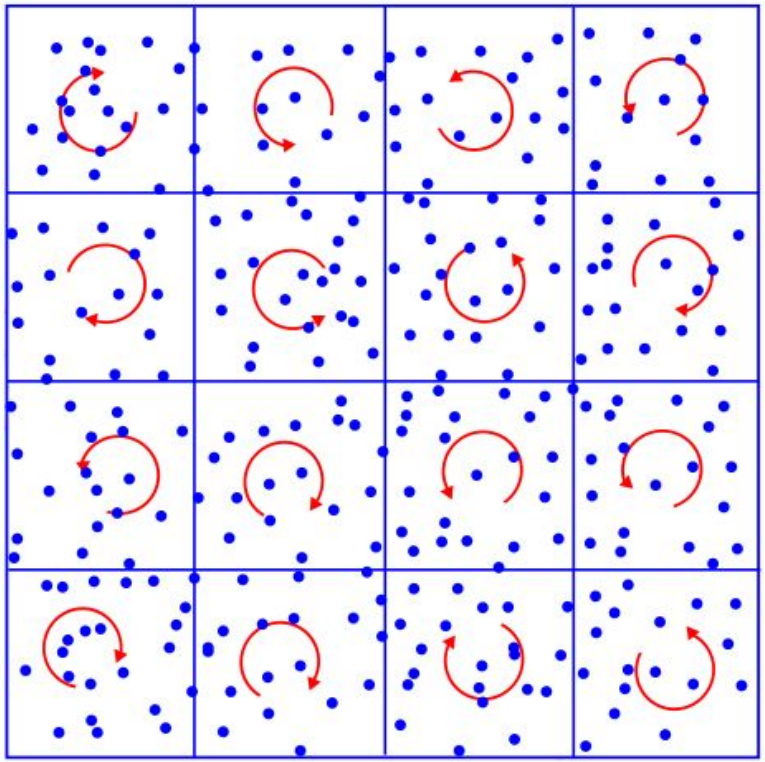
Fluid dynamics :

Streaming :

$$x(t+\delta t) = x(t) + v(t)\delta t$$

Collision :

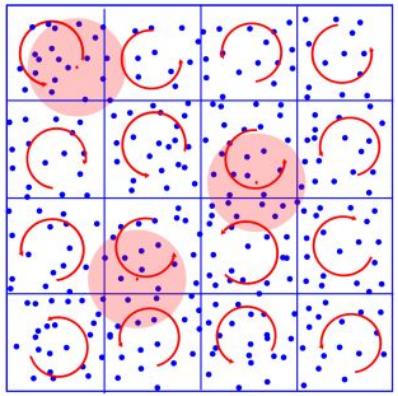
$$v(t+\delta t) = v_{cm}(t) + R(v(t) - v_{cm})$$



Simulation numérique de suspensions colloïdales

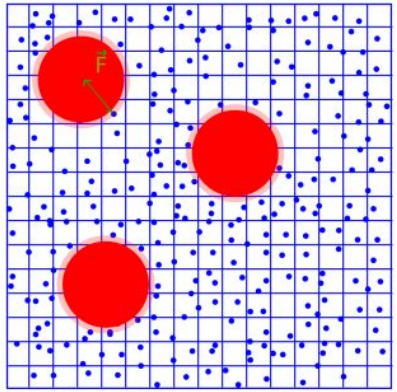
Fluid dynamics : SRD — Colloid dynamics : MD

Coupling scheme I : colloids in SRD cells



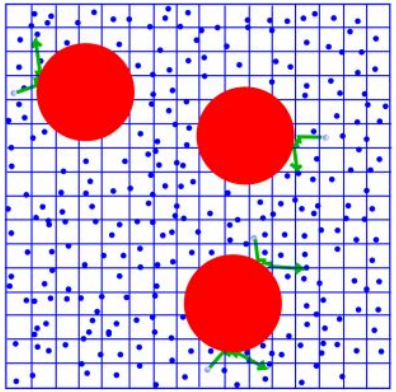
- Fast
- Ponctual colloids
- No lubrication

Coupling scheme II : fluid-colloid interactions



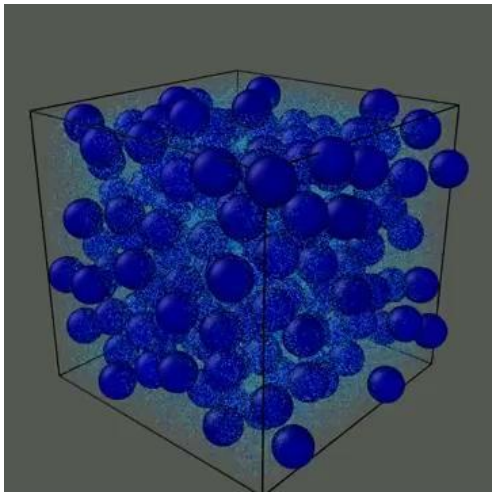
- Lubrication effects
- Colloid size not well defined

Coupling scheme III : stochastic reflections



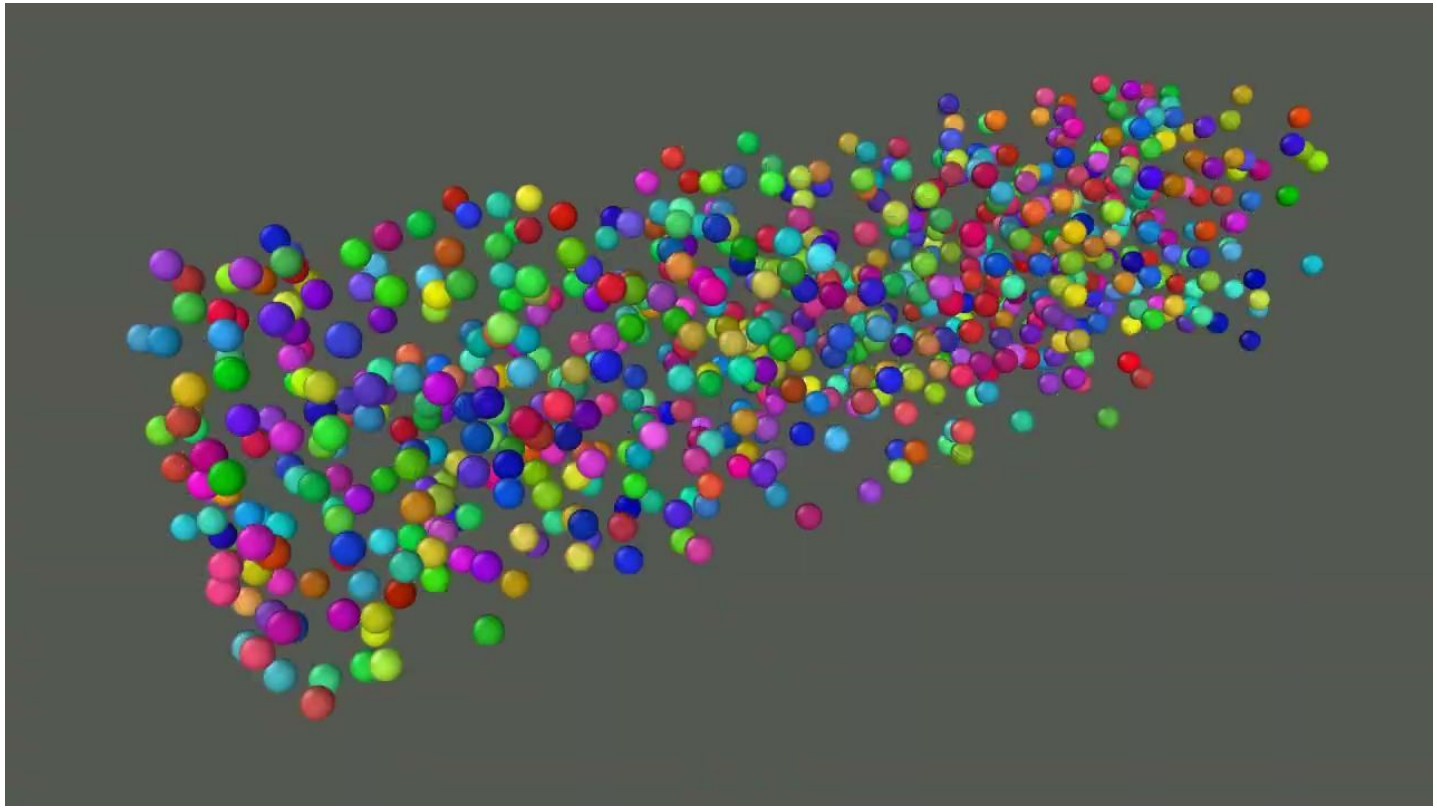
- Good description of colloid size
- Heavy computations

Simulation numérique de suspensions colloïdales

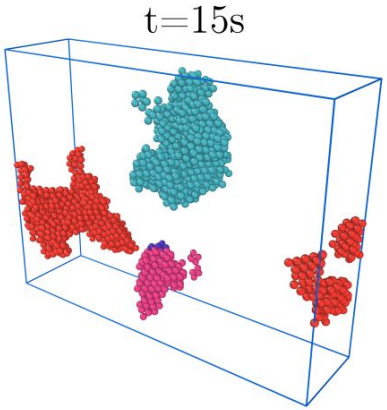


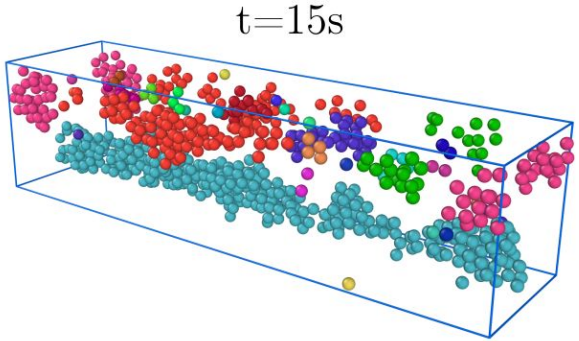
Number of Colloids	Number of fluid particles
500	788 981
2000	3 120 942
4000	6 311 854

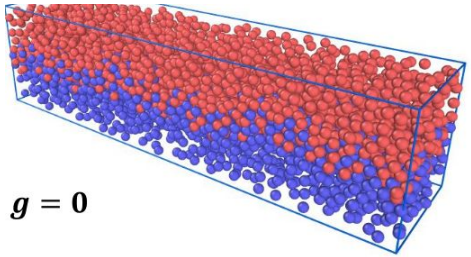
Simulation numérique de suspensions colloïdales

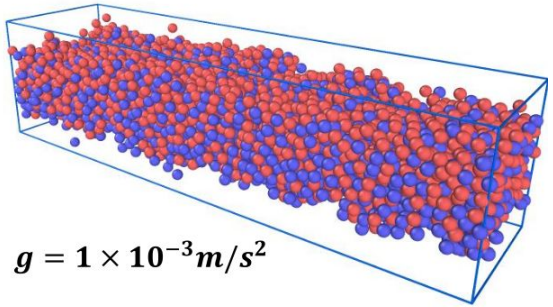


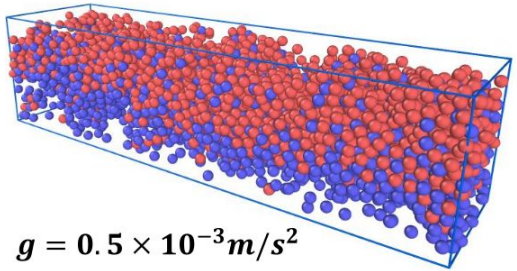
Simulation numérique de suspensions colloïdales

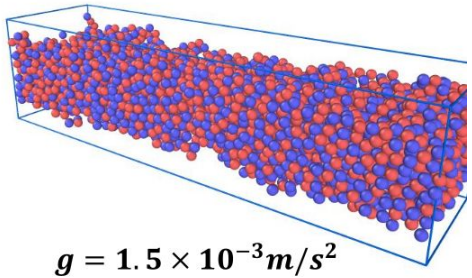




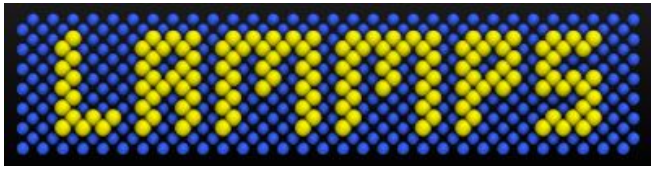








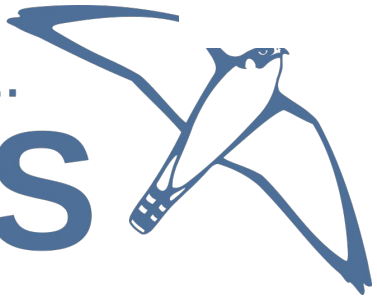
Simulation numérique de suspensions colloïdales



tinkerHP

FAST. FLEXIBLE. FREE.

GROMACS



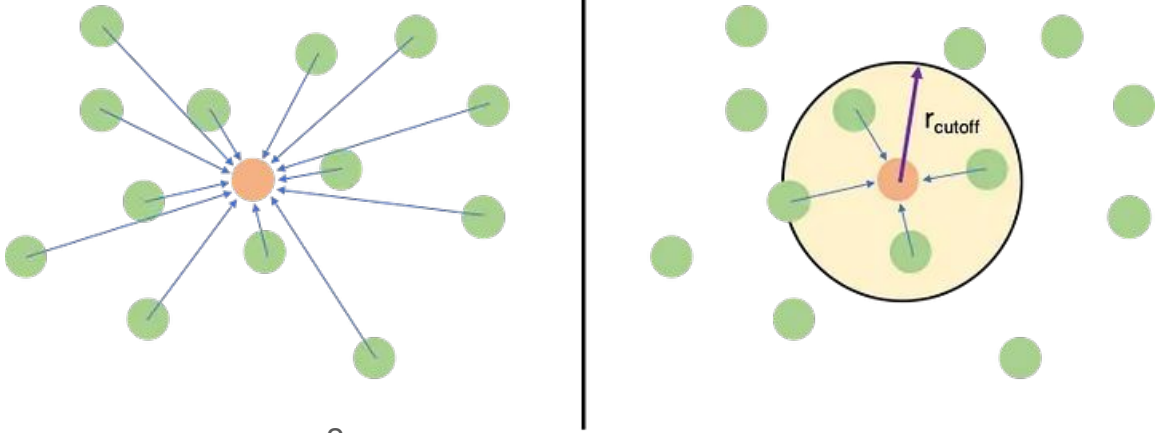
HALMD ● HAL's MD package

Highly Accelerated Large-scale Molecular Dynamics

UAMD

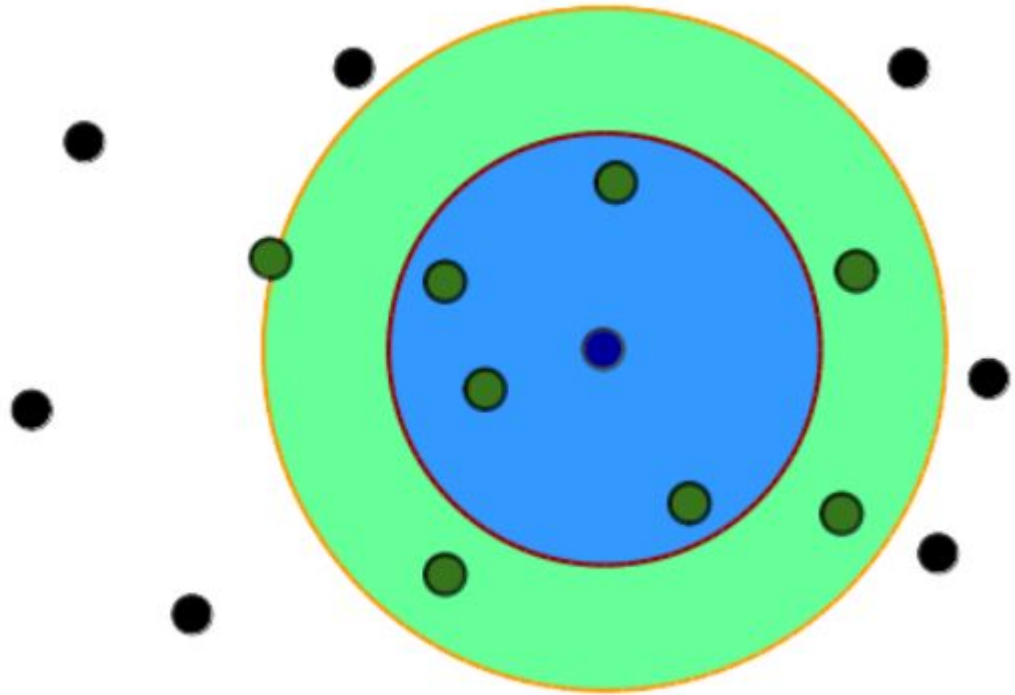
Recherche de voisinage

- 80% du temps de calcul : déterminer si deux particules sont assez proches pour entrer en interaction

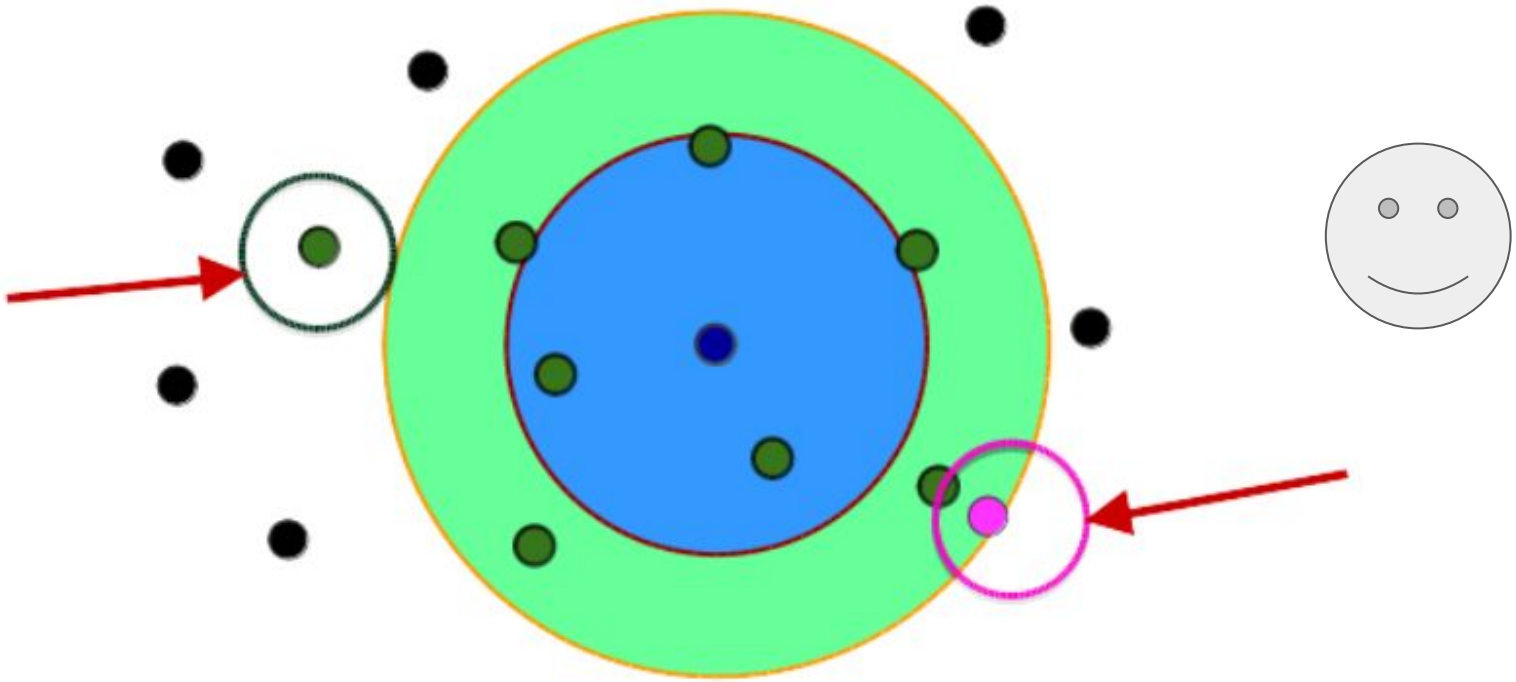


- Solution naïve : force brute $O(n^2)$
- Simulations type “SPH” : structure accélératrice
- Simulations type “Molecular Dynamics” : structure accélératrice + **liste de voisins**

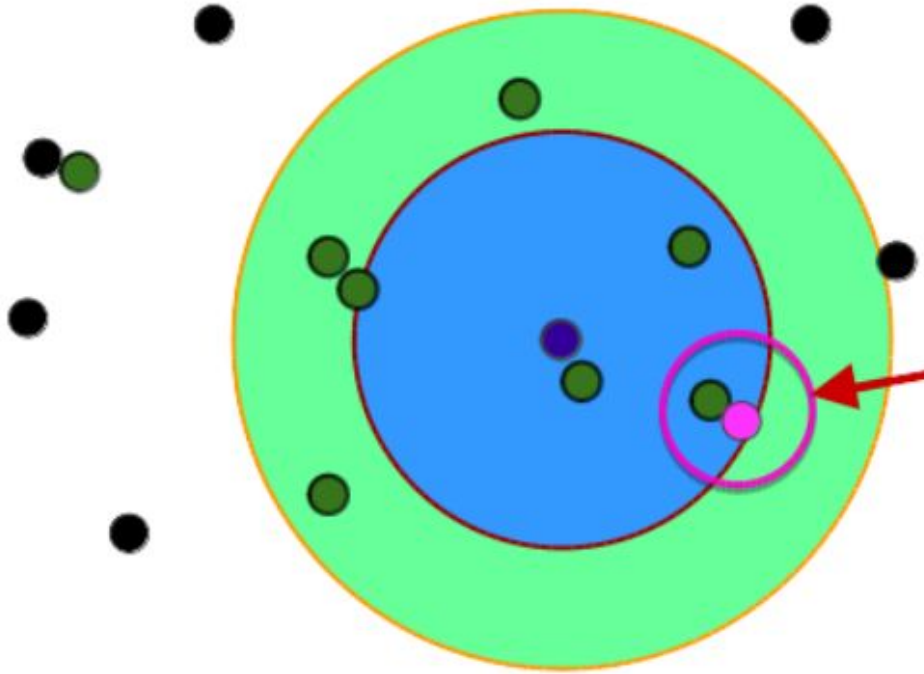
Recherche de voisinage



Recherche de voisinage

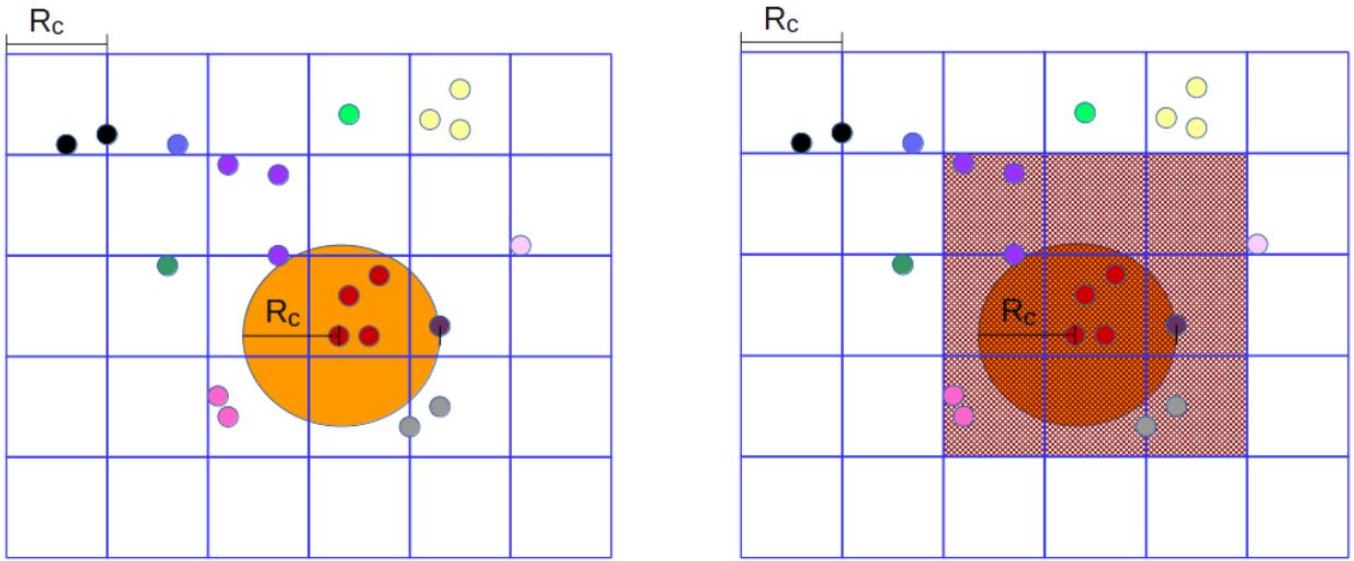


Recherche de voisinage

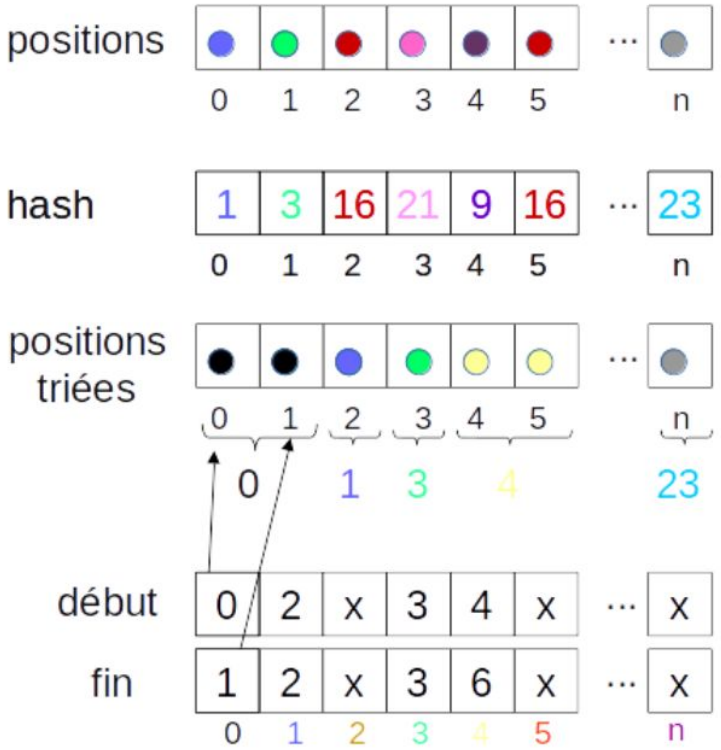


Un re-calcul de la liste est nécessaire

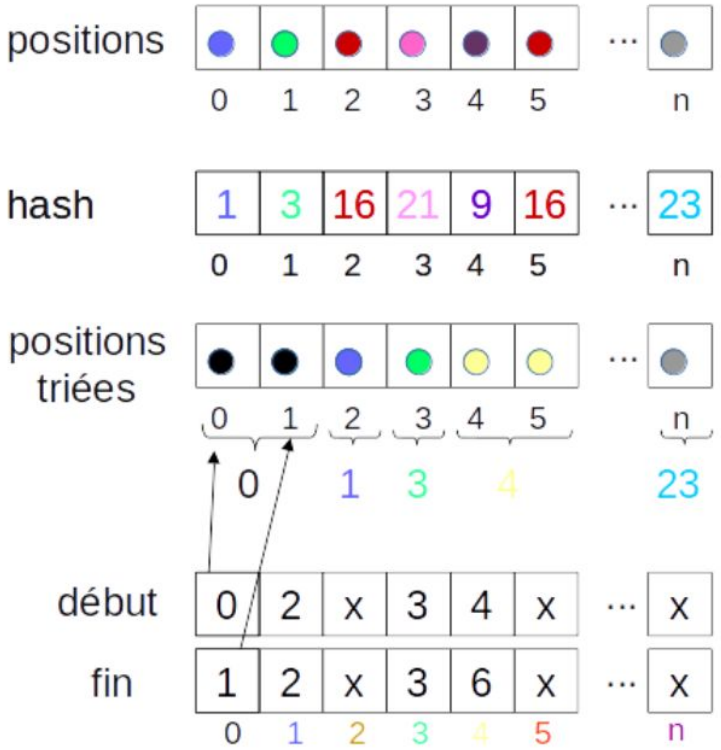
Parallélisation sur GPU - Cell lists (ou “grilles”)



Parallélisation sur GPU - Cell lists



Parallélisation sur GPU - Cell lists



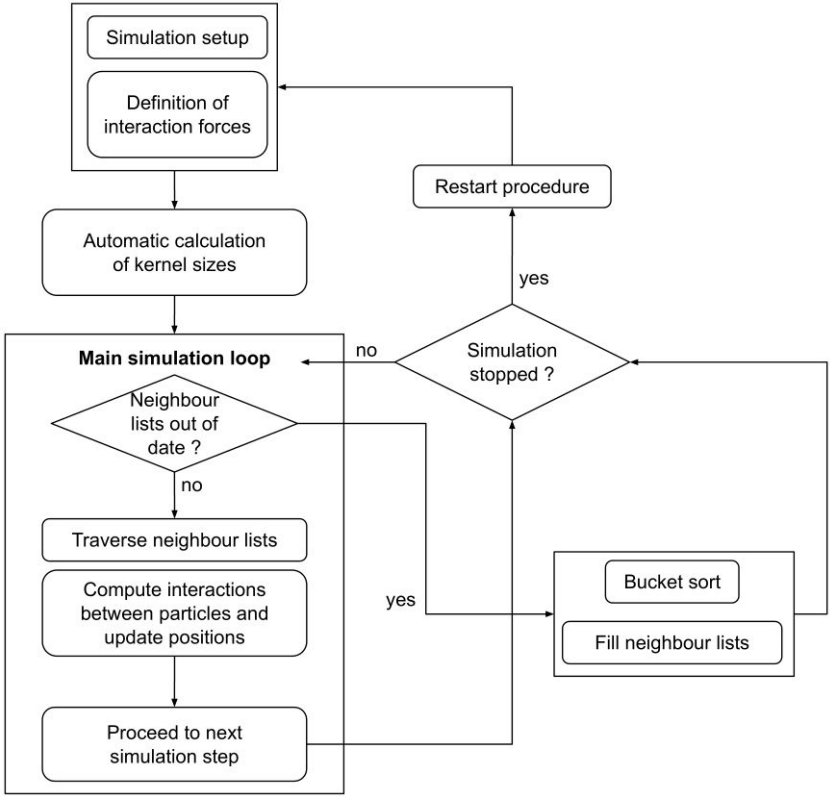
Parallélisation sur GPU - Cell lists

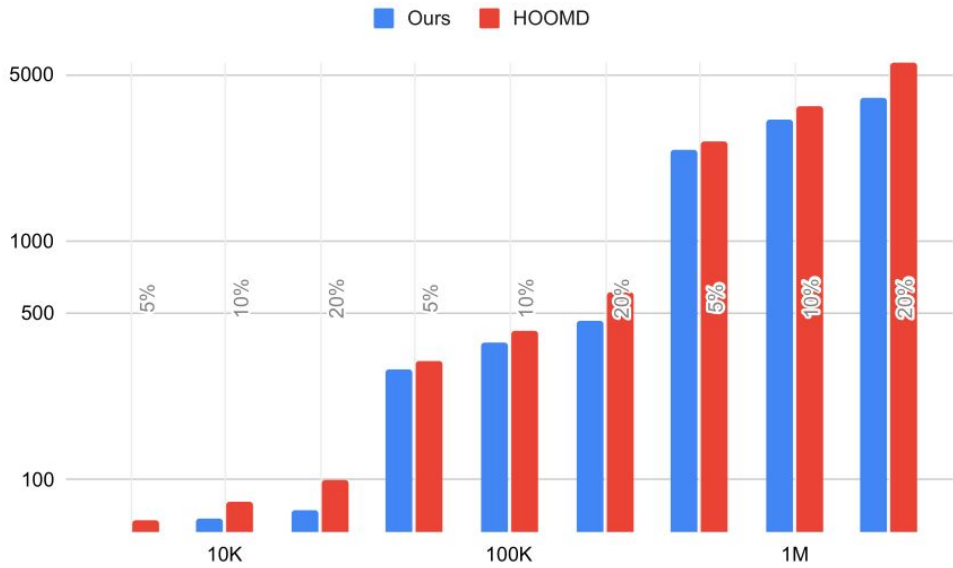
- Hoetzlein R (2014) Fast and Efficient Nearest Neighbor Search for Particle Simulations. In: GPU Technology Conference (GTC)
- Gross J, Köster M, Krüger A (2019) Fast and Efficient Nearest Neighbor Search for Particle Simulations. In: Computer Graphics and Visual Computing (CGVC)
- Chisholm R, Maddock S, Richmond P (2020) Improved GPU near neighbours performance for multi-agent simulations. Journal of Parallel and Distributed Computing 137:53–64
- https://github.com/InteractiveComputerGraphics/cuNSearch

Parallélisation sur GPU - Approches hiérarchiques

- Fernandez-Fernandez JA, Westhofen L, Löschner F, et al (2022) Fast octree neighborhood search for SPH simulations. ACM Trans Graph 41(6)
- Howard MP, Anderson JA, Nikoubashman A, et al (2016) Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units. Computer Physics Communications 203:45–52
- Ohno K, Nitta T, Nakai H (2017) SPH-based fluid simulation on GPU using Verlet list and subdivided cell-linked list. In: 2017 Fifth International Symposium on Computing and Networking (CANDAR), pp 132–138
- Howard MP, Statt A, Madutsa F, et al (2019) Quantized bounding volume hierarchies for neighbor search in molecular simulations on graphics processing units. Computational Materials Science 164:139–146
- Zhu Y (2022) RTNN: Accelerating neighbor search using **hardware ray tracing**. In: Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, ACM
- Exemple : Metric Trees

SOMA-BD : dynamique brownienne





Tests de performance sur un GPU A40 pour une simulation de 1M d'itérations (interactions Lennard-Jones)
PREPRINT : https://doi.org/10.21203/rs.3.rs-4592655/v1
Code C++ disponible

SOMA-BD : dynamique brownienne


```
{
  "PARTICLES": [
    {
      "type": 0,
      "radius": 3e-07,
      "maxRadius": 0.0,
      "minRadius": 0.0,
      "radius_std_dev": 0.0,
      "distribution": "lognormal",
      "number": 50000,
      "volumicMass": 2200.0,
      "psi":1
    },
    {
      "type": 1,
      "radius": 3e-07,
      "maxRadius": 0.0,
      "minRadius": 0.0,
      "radius_std_dev": 0.0,
      "distribution": "lognormal",
      "number": 50000,
      "volumicMass": 2200.0,
      "psi":-1
    }
  ],
  "FORCE":  [
    {
      "name": "Yukawa",
      "k":1e8,
      "U":10,
      "rcrep":1,
      "coeffdir":7.93e-13,
      "Rc": 3.2,
      "first": -1,
      "second": -1
    }
  ],
  "SOLVANT": {
    "dielectricConstant": 81.0,
    "eta":0.001
  },
…
```

Ajustement des tailles de noyaux CUDA

- A thread block is a programming abstraction that represents a group of threads that can be executed serially or in parallel. For better process and data mapping, threads are grouped into thread blocks. Threads in the same block can communicate with each other via shared memory, barrier synchronization or other synchronization primitives such as atomic operations (source : Thread block (CUDA programming) - Wikipedia)
- “The **cudaOccupancyMaxPotentialBlockSize( )** function makes it possible to compute a reasonably efficient execution configuration for a kernel without having to directly query the kernel's attributes or the device properties, regardless of what device is present or any compilation details” (source : CUDA Pro Tip: Occupancy API Simplifies Launch Configuration | NVIDIA Technical Blog)

- En réalité, pas toujours optimal…

Ajustement des tailles de noyaux CUDA

- Lancement de la simulation pendant 100 itérations avec des tailles prédéterminées allant de 32 à 2048 (multiples de 32) pour les kernels utilisés pour calculer les listes de voisins et appliquer les forces d’interaction
- Benchmark pour déterminer les tailles donnant les meilleures performances et qui seront utilisées pour toute la simulation
- Adapté à nos simulations qui restent “stables” (nombre de voisins), dans d’autres cas il serait probablement nécessaire de relancer un benchmark à intervalles réguliers pendant la simulation

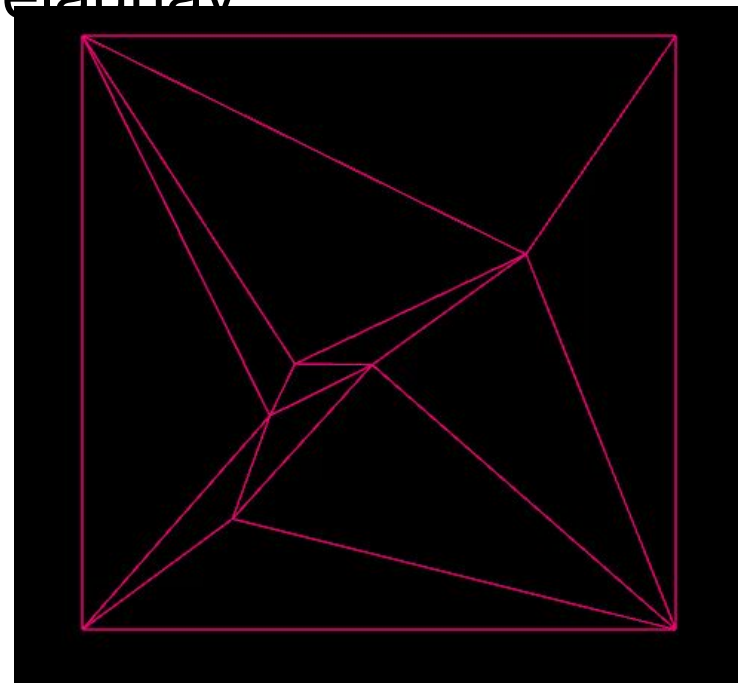
Ajustement des tailles de noyaux CUDA

GPU	Particles	LJ 5%	LJ 10%	LJ 20%	LJM 5%	LJM 10%	LJM 20%	Yukawa 5%	Yukawa 10%	Yukawa 20%	DLVO 5%	DLVO 10%	DLVO 20%
L40	10K	96	32	32	32	32	32	32	96	32	96	96	32
		2.34	2.20	2.94	2.60	2.96	3.05	2.25	2.65	3.10	2.73	3.28	3.43
	100K	64	64	32	64	64	32	64	64	64	64	64	64
		1.21	1.27	1.46	1.78	1.80	1.66	1.42	1.48	1.50	1.35	1.38	1.42
	1M	64	64	32	64	64	32	64	64	64	64	64	64
		0.85	0.86	1.06	1.23	1.22	1.48	1.05	1.04	1.38	1.04	1.04	1.04
H100	10K	128	96	64	128	96	128	96	96	256	128	256	128
		1.27	1.15	1.15	1.16	1.16	1.16	1.34	1.23	1.24	1.21	1.22	1.15
	100K	64	64	64	128	64	64	128	128	64	64	128	64
		1.18	1.09	1.13	1.12	1.10	1.11	1.25	1.21	1.28	1.10	1.12	1.10
	1M	512	512	512	512	512	512	512	512	512	512	512	512
		1.07	1.03	1.01	0.99	1.03	1.02	1.15	1.18	1.19	1.11	1.17	1.12

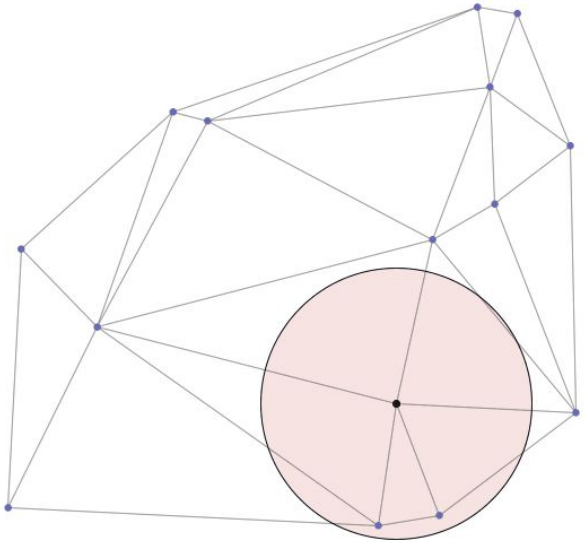
Taille déterminée après la phase d'autotuning

Facteur d'amélioration des perfs

Recherche de voisinage sur des triangulations de Delaunay

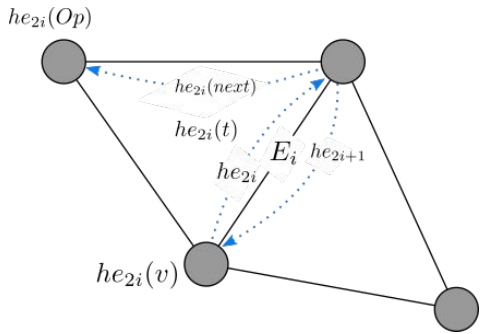


Mise à jour dynamique

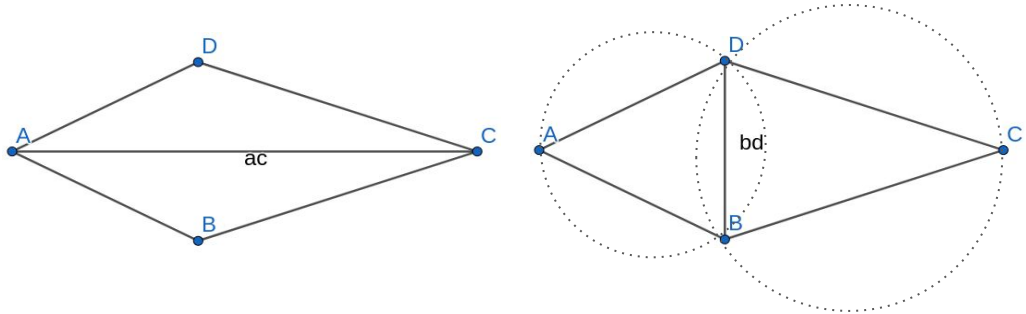


Calcul voisinage

Mise à jour dynamique



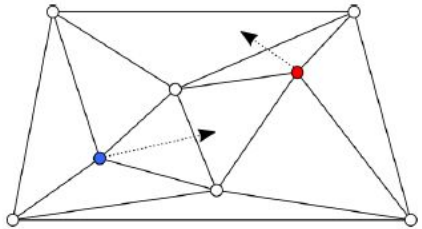
Structure de type “demi-arête”



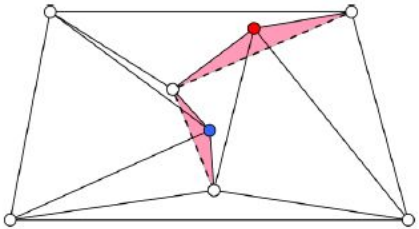
Opération de base : **flip** (GPU friendly) permettant de rendre toute triangulation “Delaunay”

Mise à jour dynamique

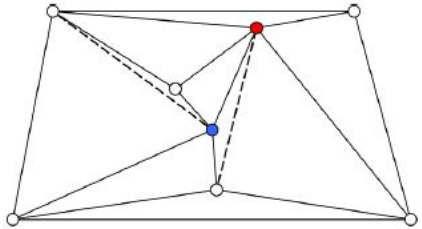
a) Mise à jour de la position des sommets
b) Recherche parallèle des arêtes problématiques
c) Flip puis retour à l'étape (b)
d) L'algorithme s'arrête quand la triangulation est entièrement "légalisée"



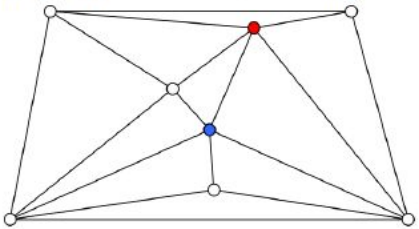
(a) Initial positions and triangulation



(b) Points displaced implying inverted triangles

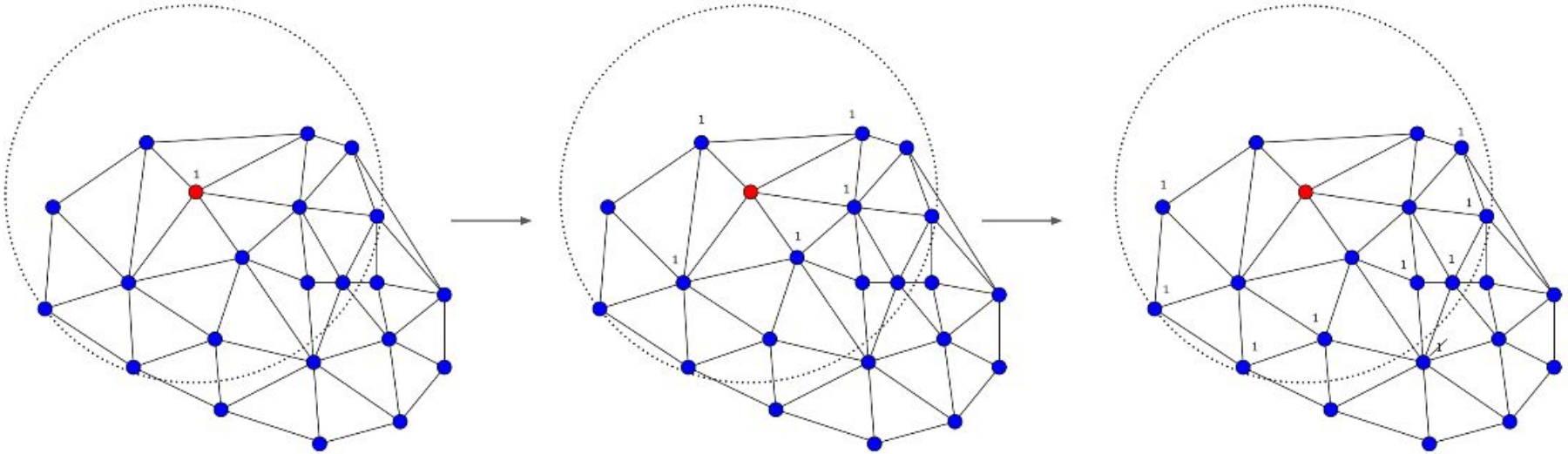


(c) Fixed valid triangulation



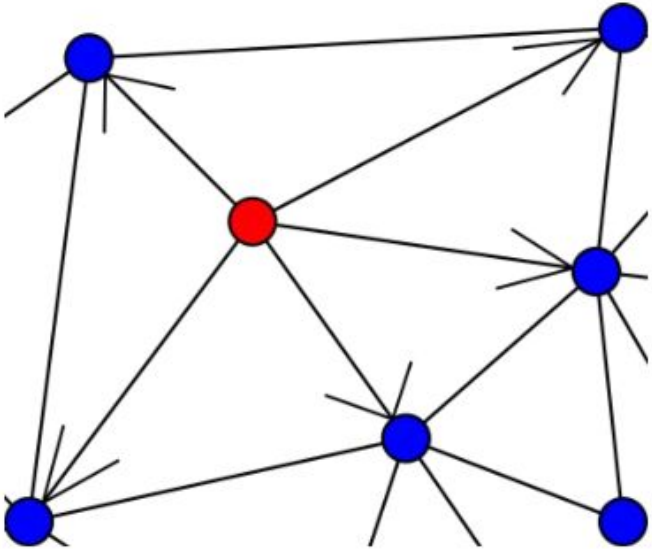
(d) Legalized triangulation

“Pulses” pour le calcul de voisinage

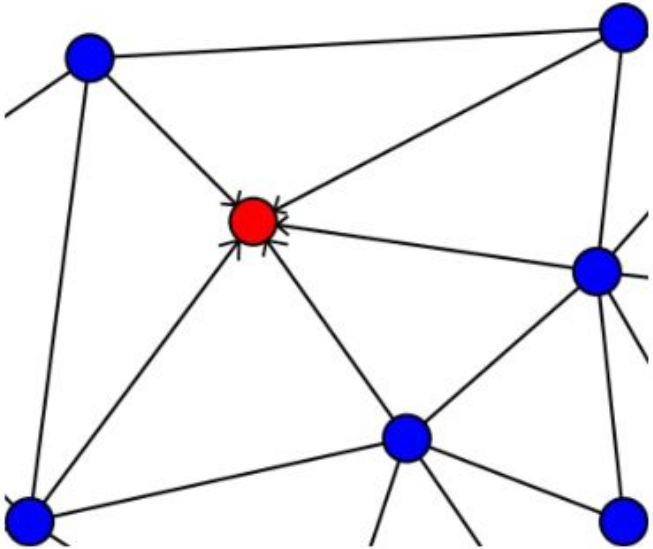


“Pulses” pour le calcul de voisinage

Pulse sharing:



Pulse receiving:



“Pulses” pour le calcul de voisinage

Pulse receiving:

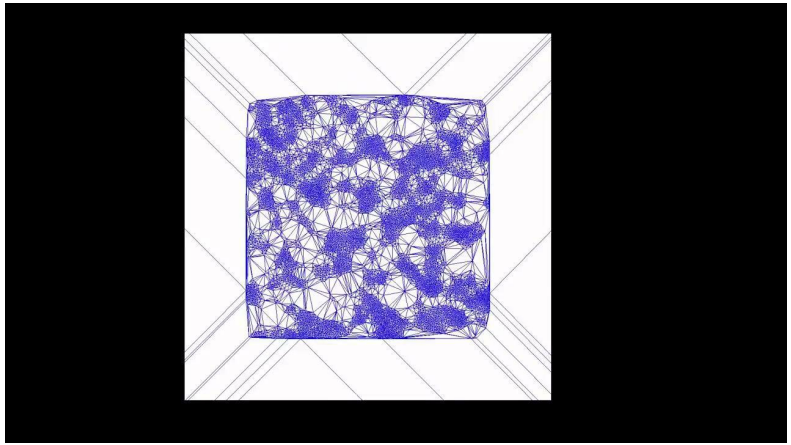
Idea: Parallelize those for using more than 1 thread per vertex

```
ParallelFor vertex v in graph G, mapped to thread t                ▷ Gather step
    for neighbor u of vertex v do
        for pulse p in to_share[u] do
            if v has not reached p before, it's not in the list received and g(v, p) then
                reached[v].add(p)
                received[v].add(p)
                theres_change ← true
            end if
        end for
    end for
end ParallelFor
```

Pulse sharing:

```
ParallelFor vertex v in graph G, mapped to thread t                ▷ Share list step
    Empty to_share[v]
    for pulse p in received[v] do        ▷ Every vertex shared the pulses it just received
        to_share[v].add(p)
    end for
end ParallelFor
```

Recherche de voisinage sur des triangulations de Delaunay



Particles	Scale	GPU	Delaunay			Grid		
			Mesh update	NN	Total (ms)	Construction	NN	Total (ms)
100K	10K	RTX 3090	0.65	0.06	0.71	0.56	0.01	0.57
		RTX A3000	0.67	0.06	0.73	0.64	0.01	0.65
1M	10K	RTX 3090	1.71	0.13	1.84	0.69	0.04	0.73
		RTX A3000	3.59	0.26	3.85	1.03	0.02	1.05