

# Conformación y dinámica de macromoléculas y nanopartículas en disolución.

## Aplicaciones de supercomputación

*José García de la Torre*

Grupo de Química Física Macromolecular, Facultad de Química,  
Universidad de Murcia

### Sistema

- \*\* Macromoléculas o nanopartículas ( 1 – 100 nm), inmersas en un disolvente (biológicas -> acuoso)
- \*\* **Rígidas o flexibles**

### Propiedades

- \*\* conformacion (forma, flexibilidad)
- \*\* dinámica global (desplazamientos traslacional, rotacional,...)
- \*\* dinámica interna, efecto de agentes externos...

## Metodologías de supercomputación que empleamos:

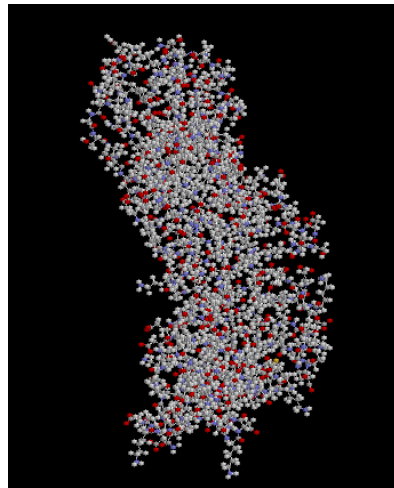
\*\* Librerías : Empleo de librerías de subrutinas de cálculo numérico ya paralelizadas (partículas rígidas)

\*\* “Pseudo-paralelización” : (1) fragmentación del cálculo en sub-cálculos, (2) ejecuciones en colas, (3) reagrupación de los sub-resultados. Desarrollo de herramientas para la automatización de estas tareas.

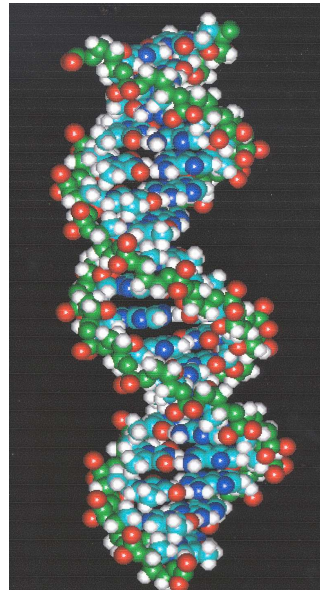
\*\* Paralelización de código

## Macromoléculas y partículas rígidas

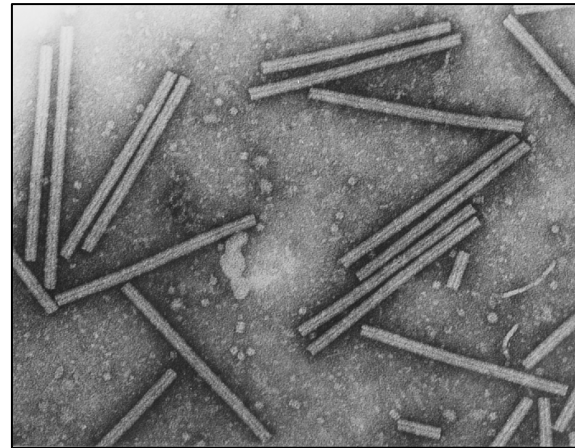
\*\* Propiedades de dinámica global dependientes de tamaño y forma globales, bien definidos



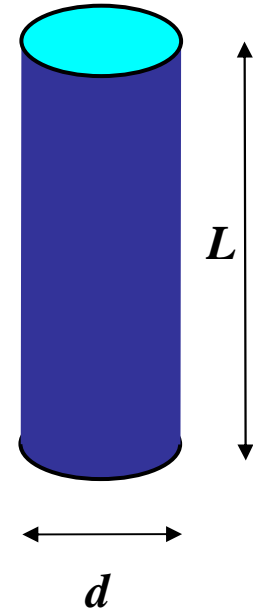
Proteína globular



ADN  
(fragmento)



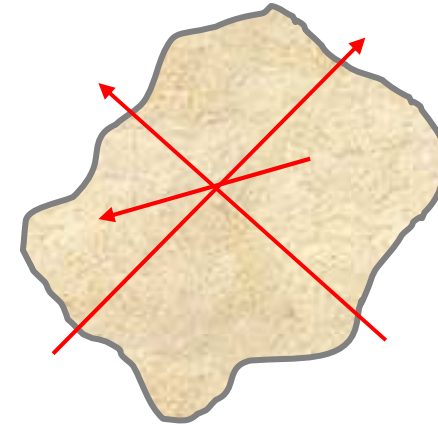
Virus mosaico del tabaco  
(filamentoso, cilindro)



## Macromoléculas y partículas rígidas

\*\* Propiedades de dinámica global en el seno de un líquido : dependientes de (un tensor de) **fricción**, **rozamiento**, dependiente de tamaño y forma.

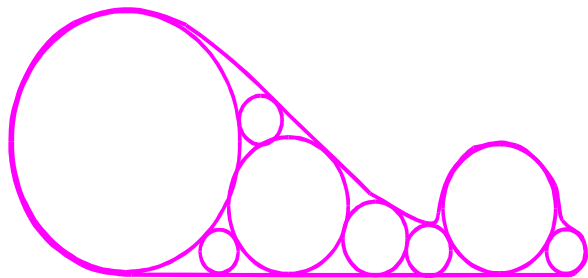
\*\* Analogías : momentos de inercia, campo eléctrico,...



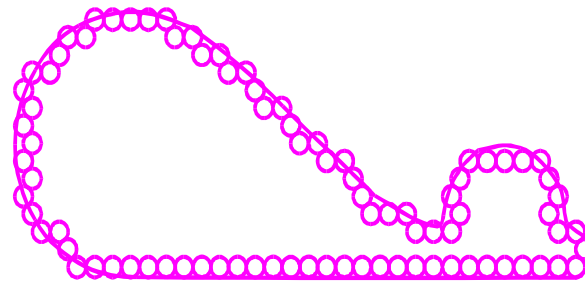
\*\* Enfoque: **elementos finitos (FE)**. En nuestro caso, elementos esféricos “**bead models**” (BM)

# Rigid-body hydrodynamics: Bead and shell models

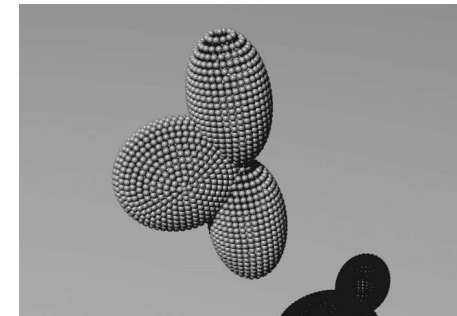
Particle modeled as an array of  $N$  spherical elements (beads).  
Size and shape of the particle is reproduced by the model.



Bead model (*in strict sense*)  
*Few beads, small  $N$*



Shell model  
 *$N \approx 2000$  minibeads*



Hydrodynamic calculation: Determination of the  $3N$  components of the frictional forces

→ Inversion of a  $3N \times 3N$  diffusion matrix,  $\underline{D}$ , ( 6000 x 6000 for shell model) **Use of parallelized HPC subroutines for linear algebra**

## Macromolecules and rigid particles

### Computational problem :

- \*\* **Inversión of matrix**  $3N \times 3N$  ( $N=2000$ ,  $6000 \times 6000$ ).
- \*\* Matrix : dense, real, square, simmetric, positive, definite.
- \*\* Via... decomposition LU, Choleski

### Library: LAPACK

(Linear Algebra PACKage) is a [software library](#) for [numerical linear algebra](#). It provides [routines](#) for solving [systems of linear equations](#) and [linear least squares](#), [eigenvalue problems](#), and [singular value decomposition](#). It also includes routines to implement the associated [matrix factorizations](#) such as [LU](#), [QR](#), [Cholesky](#) and [Schur decomposition](#) (*Wikipedia*)

**SxxTRF** - compute the factorization  $A = U^{**T}U$  or  $A = L*L^{**T}$  of the matrix A

**SxxTRI** - compute the inverse of matrix A using the factorization computed by **SxxTRF**

... Two options...

... options...

**xx = PP: SPTRF** + **SPTRI** - real symmetric positive definite matrix A stored in packed format, Cholesky factorization

**xx = GE: SGETRF** + **SGETRI** - general M-by-N matrix A using LU factorization and using partial pivoting with row interchanges

\*\* **LAPACK**, public domain <http://www.netlib.org/lapack/>

\*\* **IMKL** (Intel Math Kernel Library), academic price, implement LAPACK, comes along with (commercial, academic priced) Intel compilers Fortran 90/95, C++,... [Disponibile Ben Arabí](#)

Test: N approx. 1800, matrix 5400x5400. Hardware:

(A) 1x Intel Core Duo E6850, 1 chip, 2 cores/chip, 2 cores

(B) 1x Intel Core i7-860 1 chip, 4 cores/chip, 4 cores

(C) 2x Intel Xeon X5660 2 chips, 6 cores/chip, 12 cores

**PP** – LAPACK ó IMKL--(n threads) : approx. **80** seg (A, B, C)

**GE** – LAPACK ó IMK--(1 thread) : approx. 150 seg (A, B, C)

**GE** – IMKL--(n threads) : approx. 12 (A), 7 (B), **4** (C)

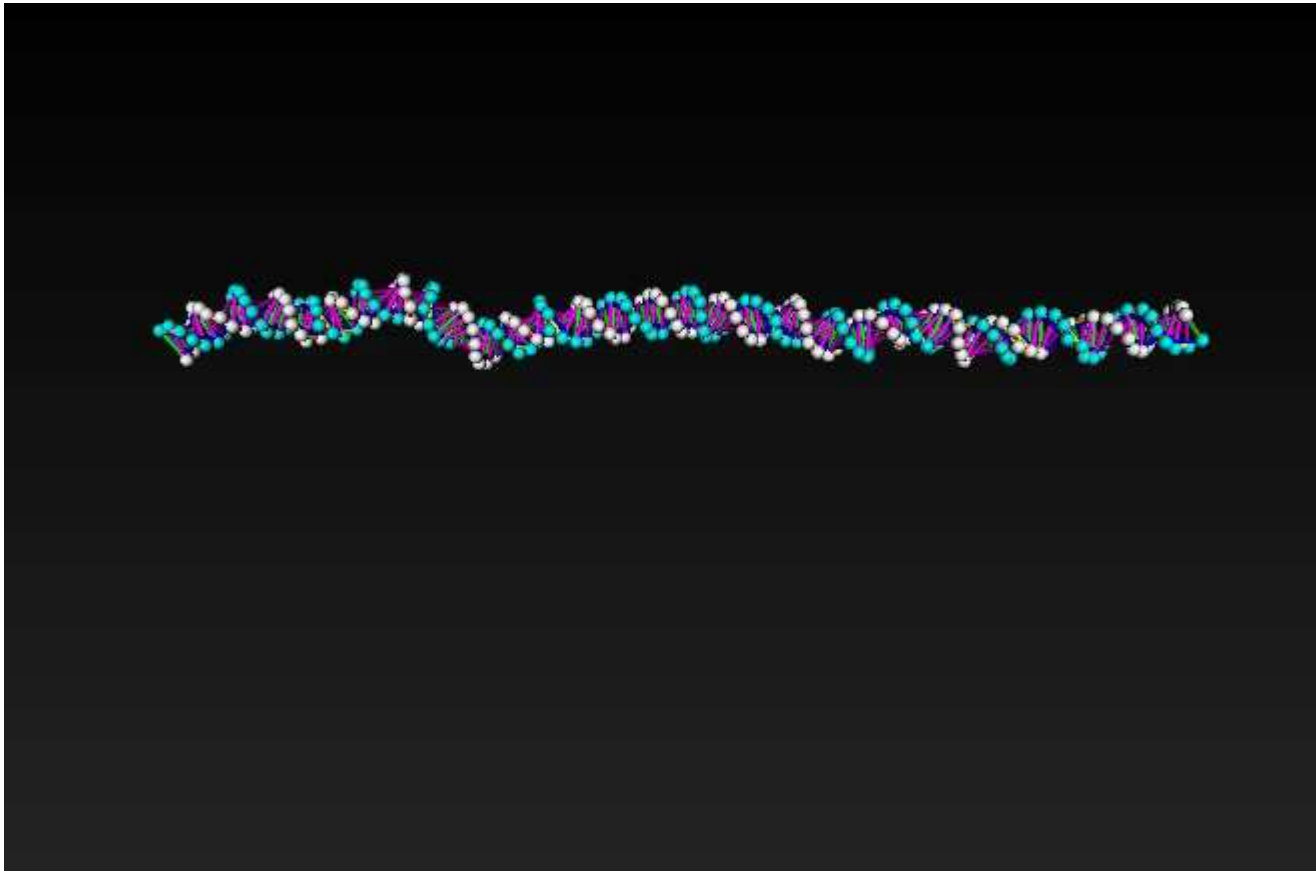
Flexibles

## Macromoléculas / nanopartículas flexibles

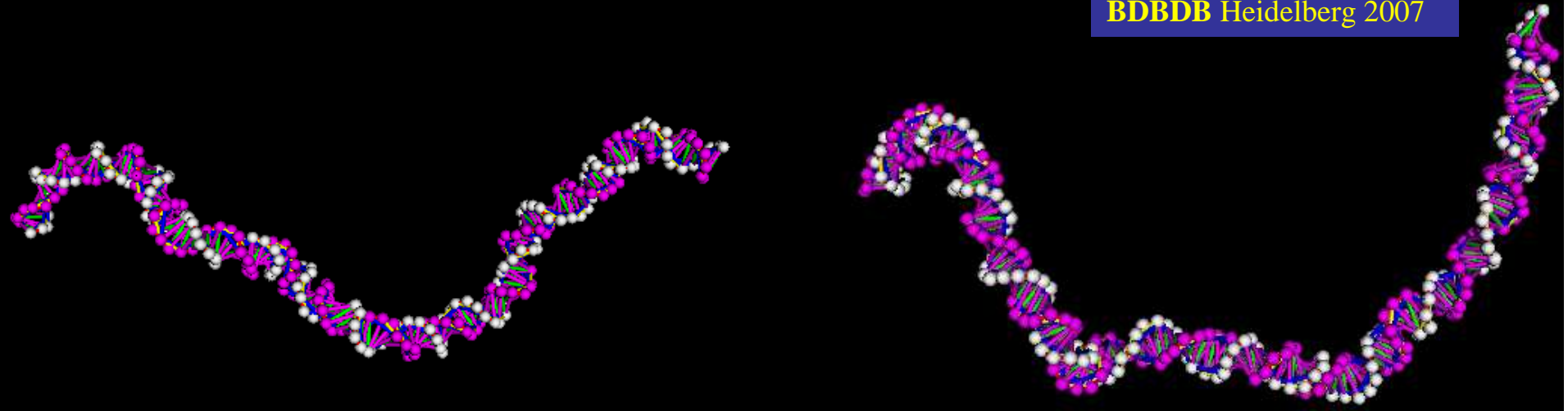
Ejemplo:

ADN 148 bp, moderadamente flexible

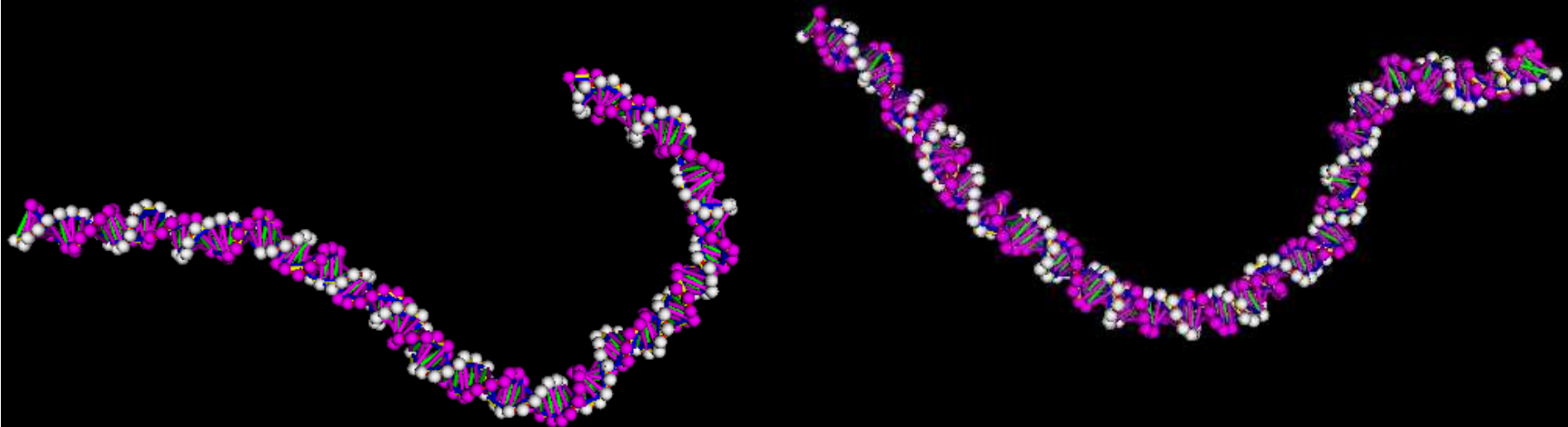
Dinámica interna. [Variabilidad conformacional](#)







Longer (148 bp) DNA. Snapshots (VisualBeads) during BD-noHI (SIMUFLEX) or MC (MONTEHYDRO) simulation



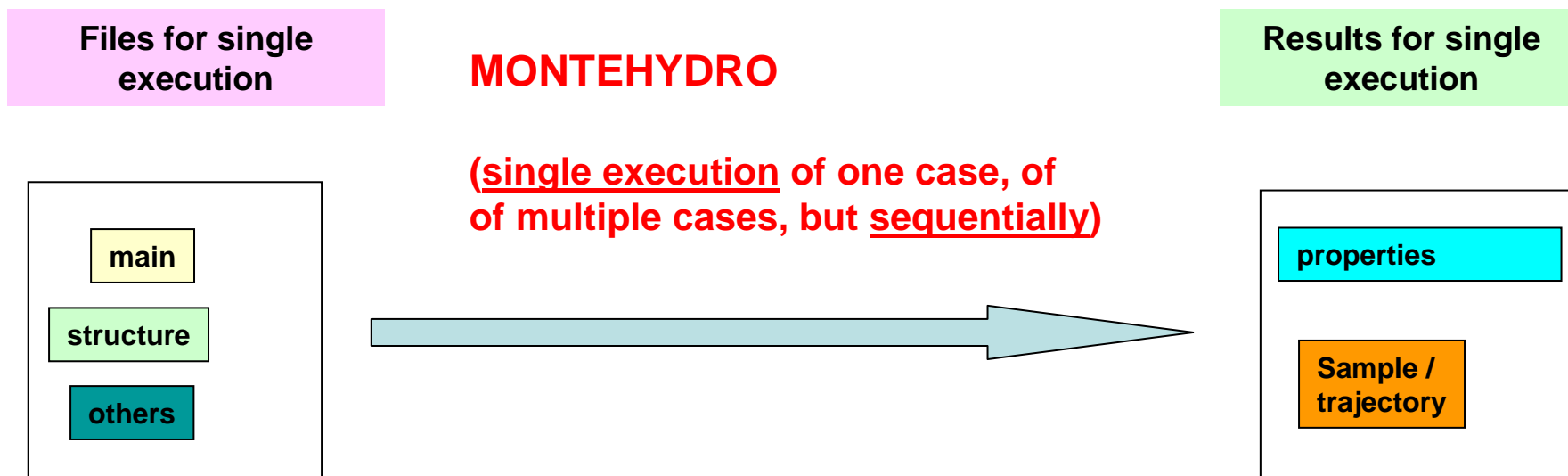
## Metodologías de supercomputación que empleamos:

\*\* Librerías : Empleo de librerías de subrutinas de cálculo numérico ya paralelizadas

\*\* “**Pseudo-paralelización**” : (1) fragmentación del cálculo en sub-cálculos, (2) ejecuciones en colas, (3) reagrupación de los sub-resultados. Desarrollo de herramientas para la automatización de estas tareas (**Partículas flexibles**)

\*\* Paralelización de código

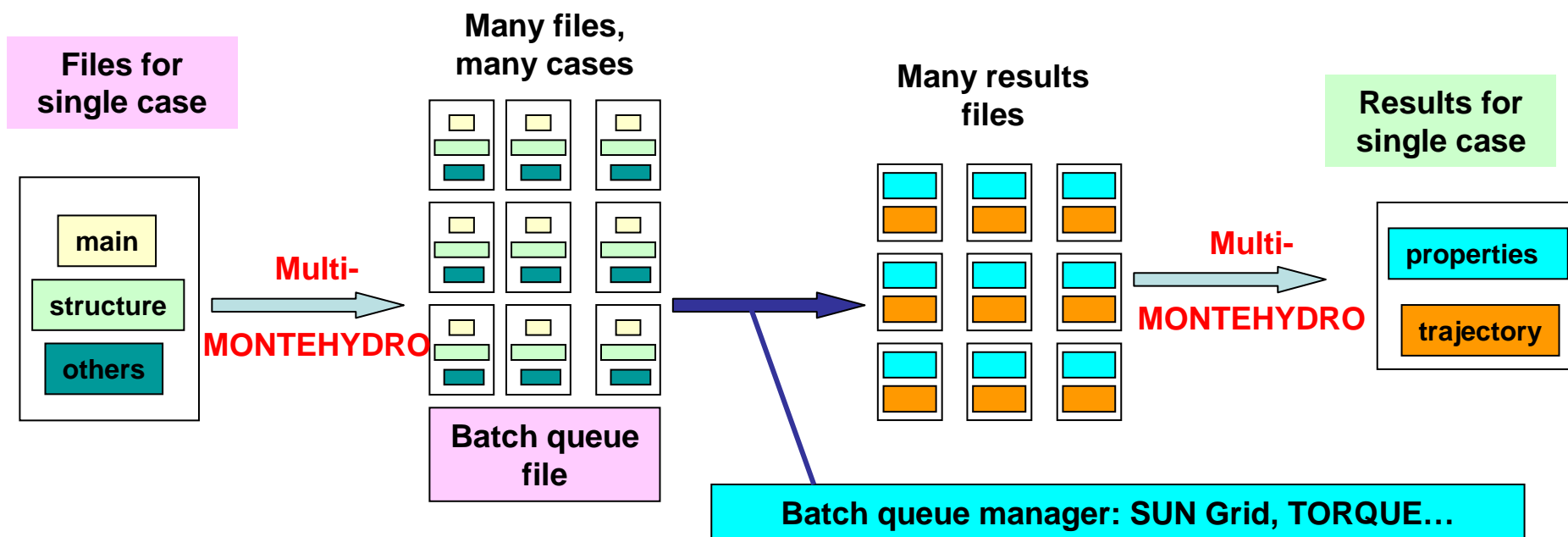
**Method: "Monte Carlo"**  
**Conventional computation; Example: MONTEHYDRO**



## Pre/Post-processing tools. Example: **Multi-MONTEHYDRO**

One Monte Carlo simulation of  $n$  steps is equivalent to  $s$  independent simulations of  $m$  steps with  $s*m=n$

“Independent” : \* Change seed of random number generators  
\* Change initial conformation



## Resumen

(A) **Monoproceso** Fichero de datos para muestra.

Ejecución HYDRO → Fichero de muestra y promedios estad.

(B) **Multiproceso, “pseudoparalelo”**

(1) Datos para muestra MULTIHYDRO → **Multiples ficheros** de sub-datos para submuestras

(2) MULTIHYDRO → **Script para gestor de colas de batch, multi-ejecuciones**  
HYDRO **paralelas** → multiples ficheros de sub-muestras y sub-resultados

(3) MULTIHYDRO → **Recolección y tratamiento de multi-subresultados** para obtener los resultados finales

Esquema idóneo para **clusters** como el de BenArabí, >800 cores, hilos, ejecuciones paralelas.

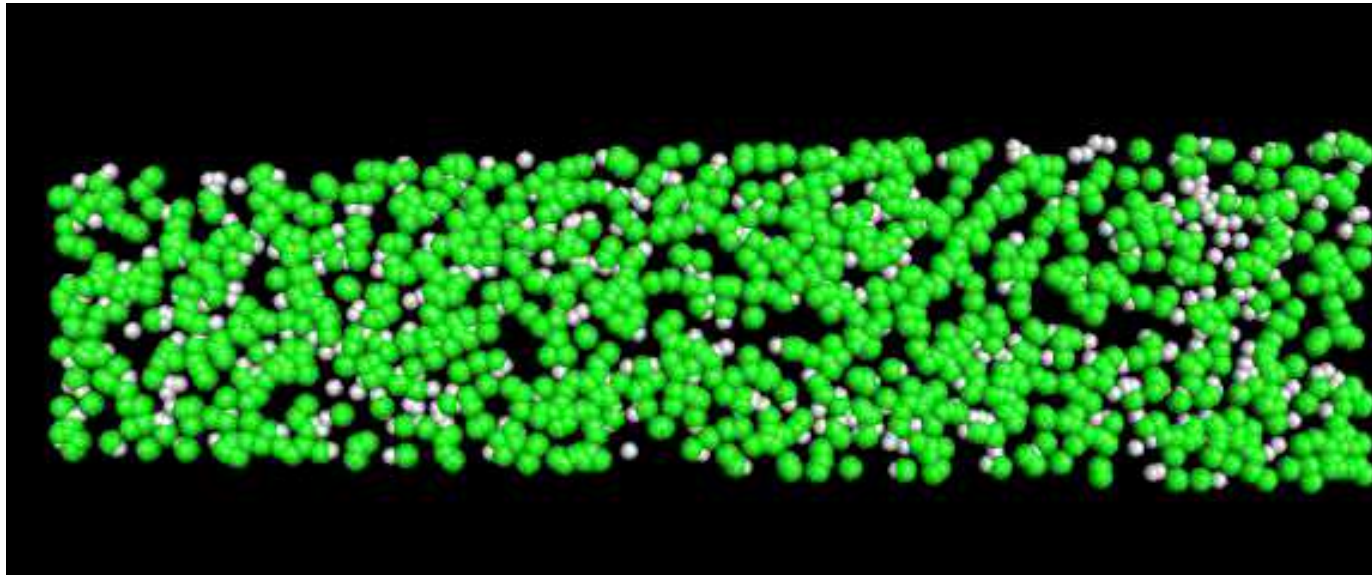
## Metodologías de supercomputación que empleamos:

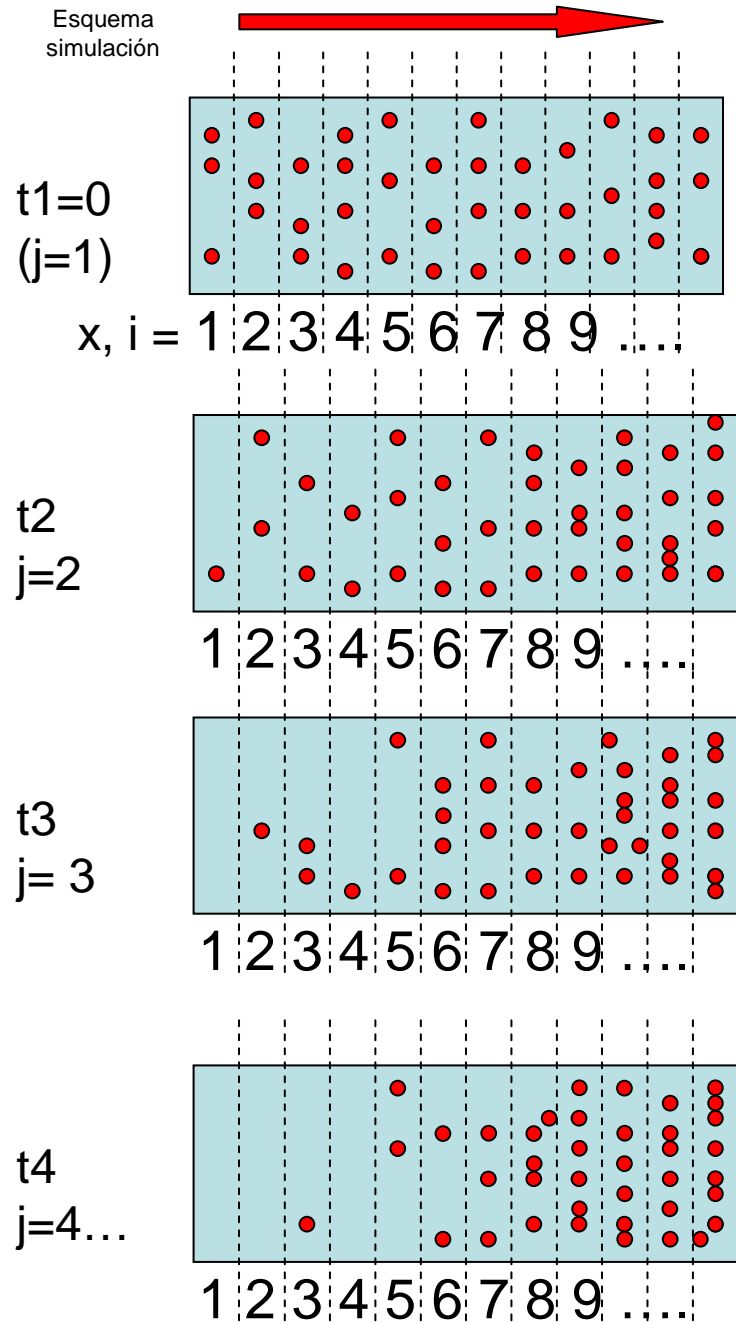
\*\* Librerías : Empleo de librerías de subrutinas de cálculo numérico ya paralelizadas (partículas rígidas)

\*\* “Pseudo-paralelización” : (1) fragmentación del cálculo en sub-cálculos, (2) ejecuciones en colas, (3) reagrupación de los sub-resultados.  
Desarrollo de herramientas para la automatización de estas tareas.

\*\* **Paralelización de código**

Simulación de un experimento de  
**sedimentación por centrifugación**  
partículas (verde) inmersas en un líquido (blanco)





Trayectorias rectilíneas, dos componentes  
 (1) Determinista, hacia el fondo ó derecha  
 (2) Aleatorio (dcha. o izq.)

Algoritmo de trayectoria : sencillo

Transporte neto hacia el fondo ó derecha

Problema : determinar  $n(x,t)$  ,ó  $n(i,j)$

\*\* Simular trayectorias (*independientes*)  
 para un número elevado de partículas

\*\* *Para cada partícula...*

\*\* ...ir dando los sucesivos pasos de  
 tiempo, t

\*\* En cada paso, t (j), determinar en que  
 intervalo x (i) se encuentra... y  
 $n(i,j) \leftarrow n(i,j)+1$



## PseudoFortran code for simulating sedimentation

```

n(:, :) = 0
DO i_part=1,n_particles
  ISEEDS(i_part)= int(RAN(iseed)*1.E7) !generates a seed for each particle
END DO

!***DO loop over particles
!-----
!$OMP PARALLEL DEFAULT(NONE) SHARED(n_particles, const1, const2, const3, const4,.....)
!$OMP DO PRIVATE(iseed, i, i_part, j, posn, random, drift)

DO i_part=1,n_particles
  iseed=ISEEDS(i_part) !initial seed for each particle
! Initial position, initial sector
  posn = ... Iseed, Const1, const2, ... ; i = ...posn ... Const1, const2, ... !Initial position

!$OMP ATOMIC
  n(i,1) = n(i,1)+1

! DO loop over successive times
!-----
  DO j = 2, n_t
    drift = ... posn, ... cons1, cons2, ... !deterministic component of displacement
    random = ... iseed, ... cons3, cons4, ... !random component of displacement
    posn = posn + drift + random ; i = ...posn ... Const1, const2, !New position

!$OMP ATOMIC
    n(i,j) = n(i,j)+1

  ENDDO
ENDDO
!$OMP END DO
!$OMP END PARALLEL

```

## CONCLUSIONES

\*\* **No inventemos la rueda...** → Tenemos **herramientas (librerías)** pero debemos saber algo de qué van... (álgebra, cálculo numérico...), e invertir esfuerzo en conocimiento y “benchmarking”. Ejemplo: IMKL inversión de matriz paralelizada

\*\* **Hay problemas paralelizables per se** → Paralelizemos / **subdividamos el problema**, multi-ejecutemos en cola de batch. Diseño de la subdivisión y de nuestras propias herramientas pre/post-procesadoras. Ejemplo: MONTEHYDRO

\*\* **Paralelización del código.** **Potente, pero compleja** en casos simples. No basta con saber informática (OMP,...) sino también un poquito de matemáticas, física,... Ejemplo: Simulación OMP sedimentación.

Finalmente, de parte de nuestro Grupo...

**“Help Wanted !!!”**

Agradecemos cualquier colaboración...

... voluntaria o, si llega el caso, remunerada... ;--)