Implementación de las condiciones iniciales y VMD

10 de Junio de 2021

¿Qué es lo que queremos simular?

Queremos estudiar un sistema de partículas que interactúan entre si.

Las mismas se ubican en una caja cúbica de lado L.

Analizaremos como se comporta el sistema en función de la densidad y de la temperatura.

$$u(r) = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}]$$

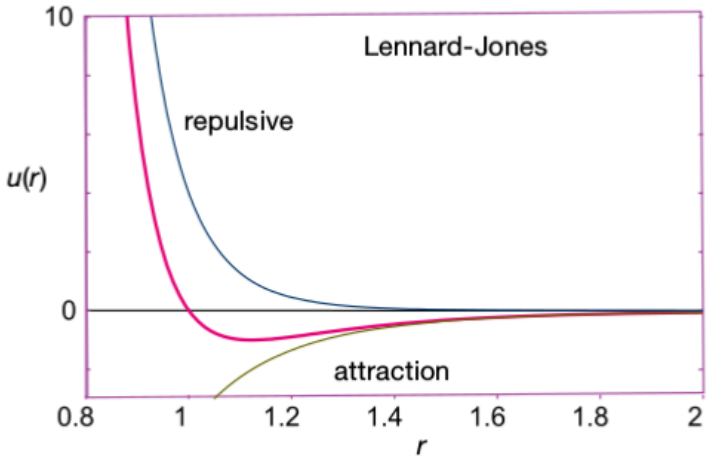
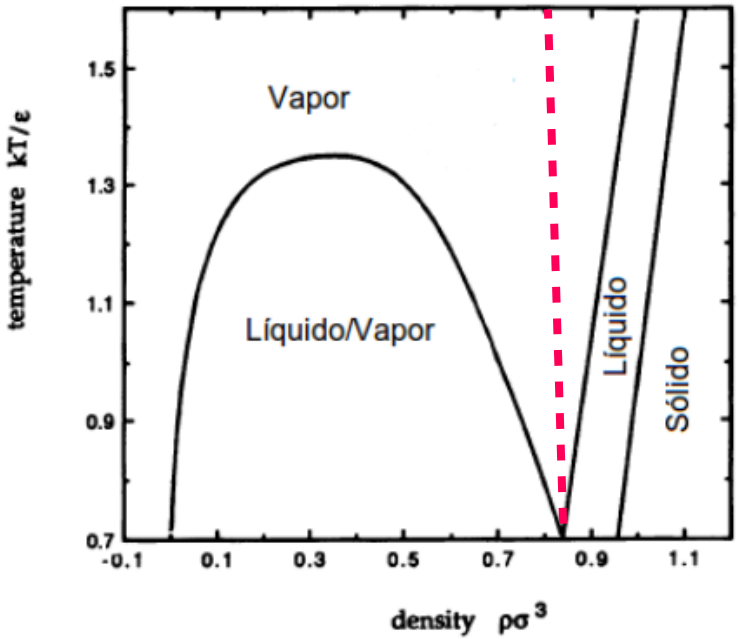


Diagrama de fases para un potencial de Lennard-Jones

Queremos estudiar un sistema de partículas que interactúan entre si.

Vamos a “movernos” sobre el diagrama de fases.

Caracterizaremos las distintas regiones mediante diferentes observables (ej. la presión y la energía).



Ingredientes

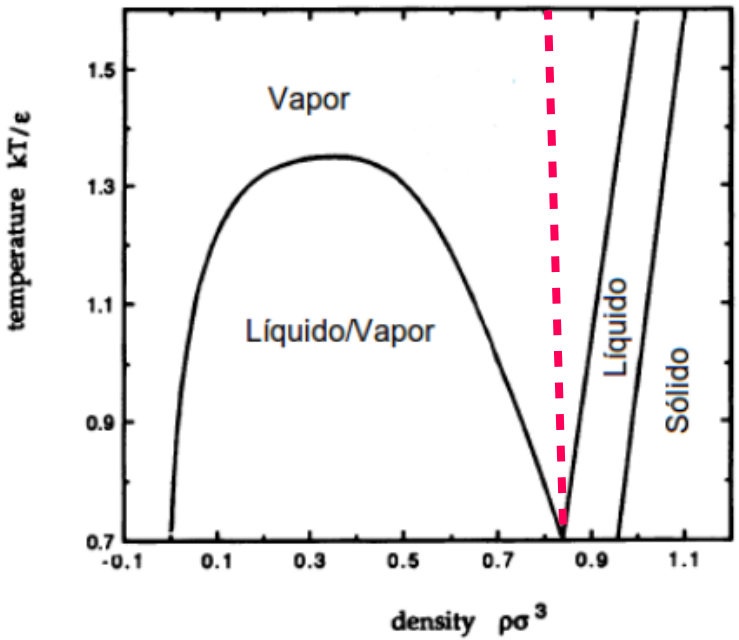
- Debemos indicar el número de partículas a simular para una dada densidad (fija) → volumen de la caja.

- Potencial de interacción.

- Algoritmo para integrar las ecuaciones de movimiento (Verlet en velocidades).

¿Condiciones de contorno?

¿Condiciones iniciales?



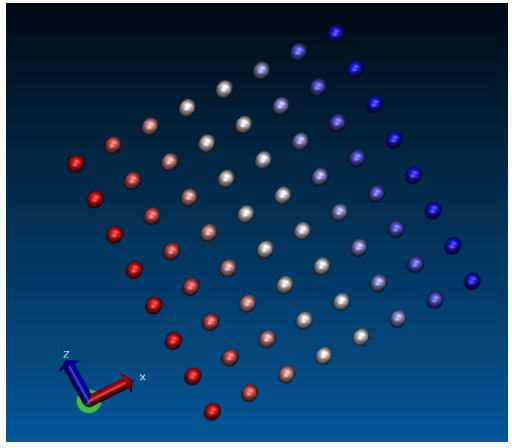
Condiciones iniciales: posiciones

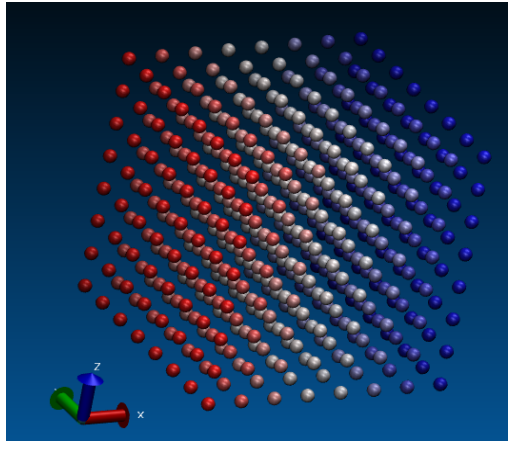
Las partículas empiezan posicionadas en un arreglo cúbico simple (sc), ¿por qué?

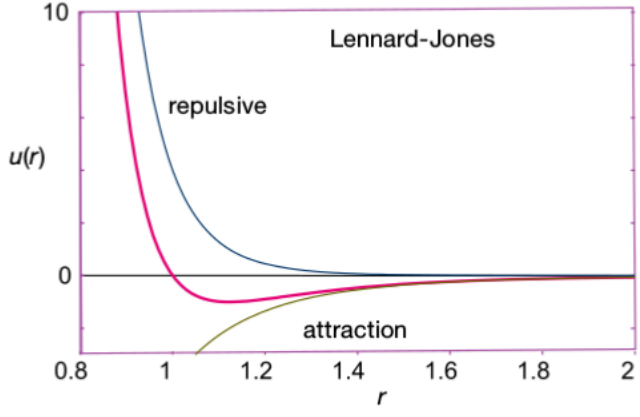
N = 512 → habrá 8 partículas por lado (*pow(N,1/3.)*).

$\rho = \frac{N}{L^3}$ → despejamos el lado de nuestro cubo.

Con estas dos cosas podemos obtener la distancia de separación entre las partículas.



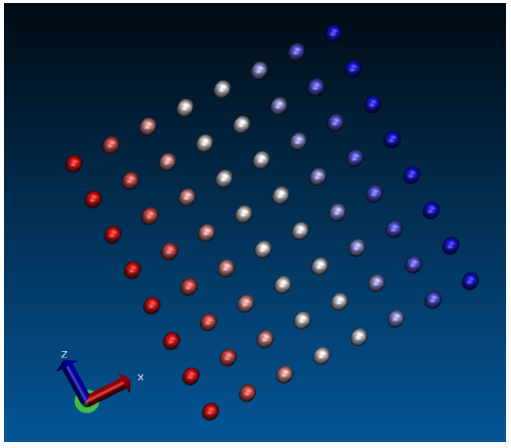


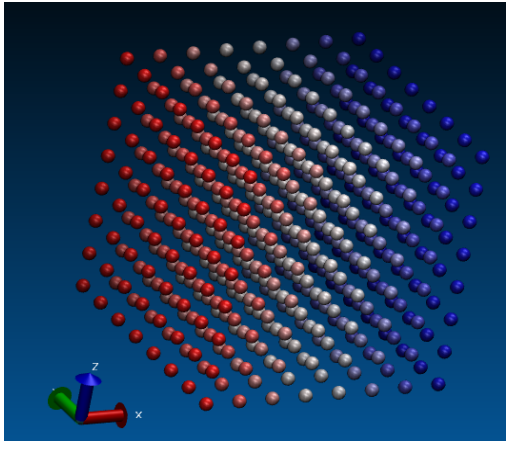


Condiciones iniciales: posiciones

Sea “a” la distancia entre las partículas (primeras vecinas).

```
for(k=0; k<partxlado;k++)
{
    for(j=0; j<partxlado;j++)
    {
        for(i=0; i<partxlado;i++)
        {
            indice = i+partxlado*j+partxlado*partxlado*k;
            vector_x[indice] = i*a+a/2.0;
            vector_y[indice] = j*a+a/2.0;
            vector_z[indice] = k*a+a/2.0;
        }
    }
}
```





“Barremos por completo cada piso y luego subimos otro”

Condiciones iniciales: velocidades

Distribución de Maxwell-Boltzmann (sistema en equilibrio a temperatura T)

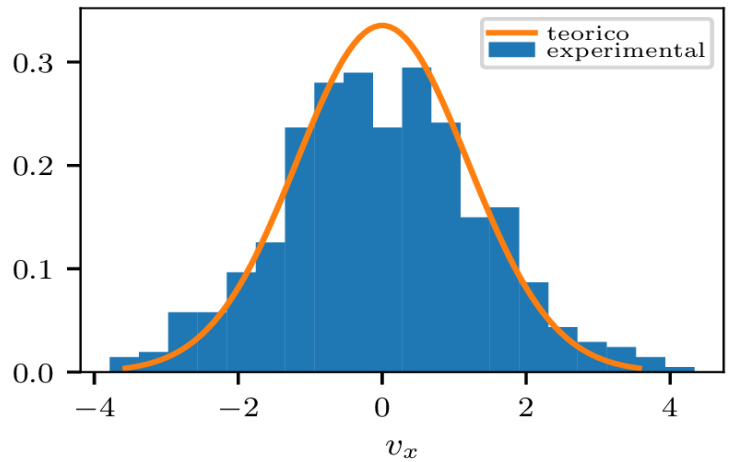
$$P(p) = \left(\frac{\beta}{2\pi m}\right)^{3/2} \exp\left[-\frac{1}{kT}\frac{p^2}{2m}\right]$$

alta

Distribución gaussiana centrada en cero y con dispersión √T

Como hacerlo:

1) Asignación de velocidades “gaussianas”.
2) Calculo valor medio (cada coordenada).
3) $v_x^{(i)} = v_x^{(i)} - \langle v_x \rangle$ (lo mismo con “y” y “z”).



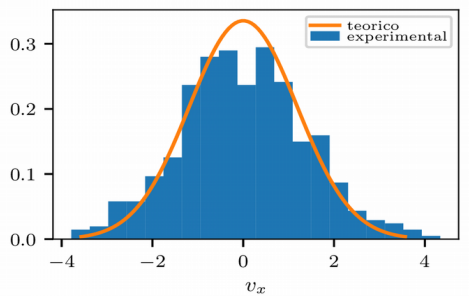
Condiciones iniciales: velocidades

Distribución de Maxwell-Boltzmann (sistema en equilibrio a temperatura T)

alta

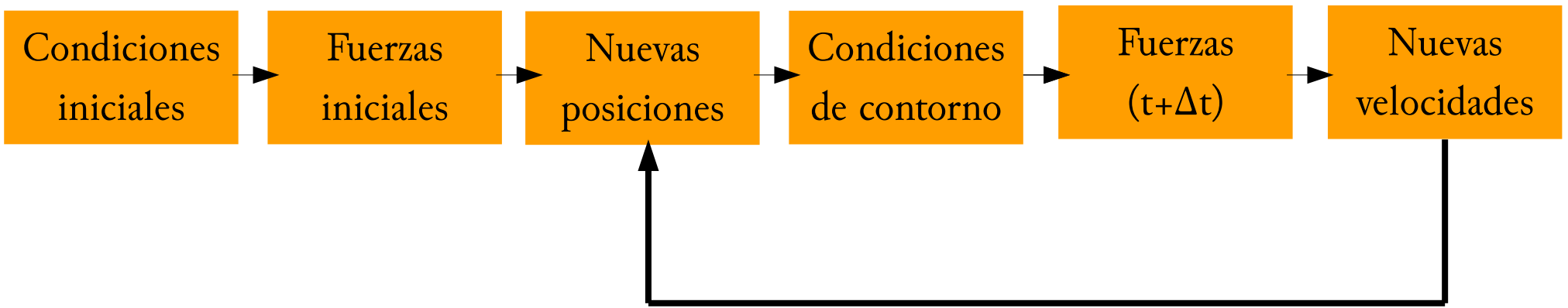
```
double ex = rand_gausiano(media,sigma);
double ey = rand_gausiano(media,sigma);
double ez = rand_gausiano(media,sigma);
```

```
//////////////////// Codigo que genera un random gausiano  ////////////////////
double drand(){
/* uniform distribution, (0..1] */
  return (rand()+1.0)/(RAND_MAX+1.0);
}
double random_normal(){
/* normal distribution, centered on 0, std dev 1 */
  return sqrt(-2*log(drand())) * cos(2*M_PI*drand());
}
double rand_gausiano(float media,float sigma){
  return media + sigma*random_normal();
}
// Fuente: https://stackoverflow.com/questions/7034930/how-to-generate-gaussian-
//pseudo-random-numbers-in-c-for-a-given-mean-and-varianc
///////////// ////////////////////////////  ///////////
```



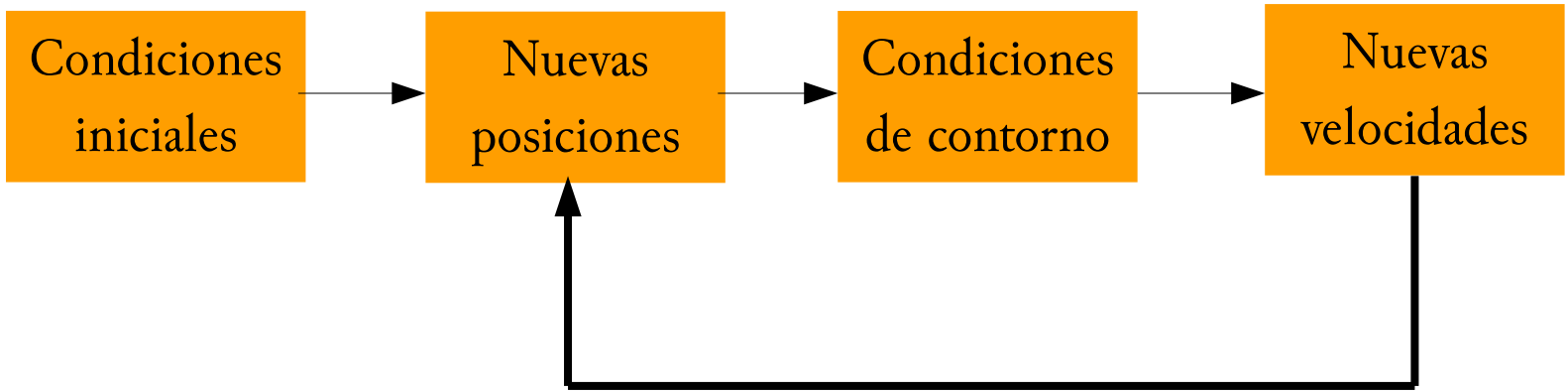
Algoritmo de Verlet (en velocidades)

$$\begin{cases} x^{(k)}(t+\Delta t) = x^{(k)}(t) + v^{(k)}(t)\ \Delta t + \frac{f^{(k)}(t)}{2m}\Delta t^2 \\ v^{(k)}(t+\Delta t) = v^{(k)}(t) + \frac{f^{(k)}(t+\Delta t)+f^{(k)}(t)}{2m}\Delta t \end{cases}$$



Algoritmo de Verlet (en velocidades): caso sin interacción

$$\begin{cases} x^{(k)}(t+\Delta t) = x^{(k)}(t) + v^{(k)}(t)\ \Delta t \\ \\ v^{(k)}(t+\Delta t) = v^{(k)}(t) \end{cases}$$



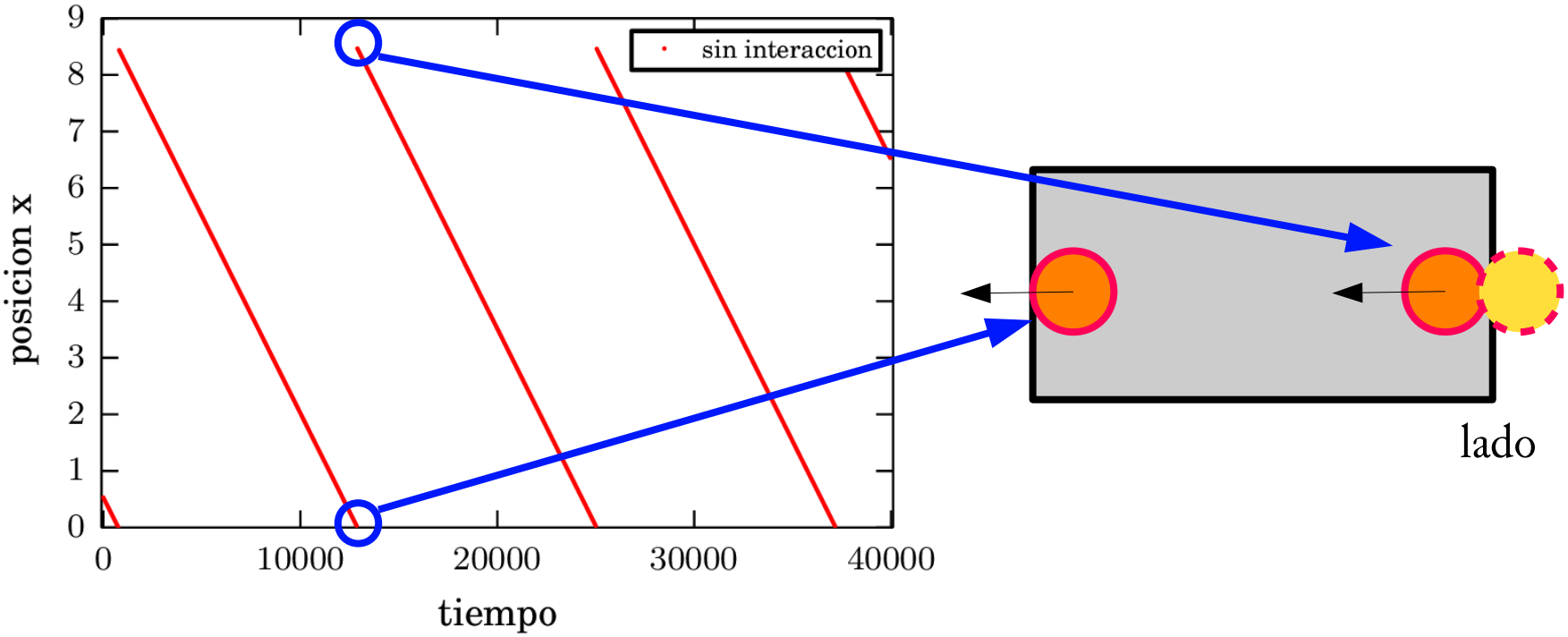
Condiciones de contorno: efecto Mario Bros

```
if (x<0) x = x + L;
if (x>L) x = x - L;
```

```
escape = ((int)((pos_x[i]+lado)/lado)-1)

(int)(pos_x[i]/lado) -> (int)(2.1/2.0) = 1
(int)(pos_x[i]/lado) -> (int)(0.1/2.0) = 0
(int)(pos_x[i]/lado) -> (int)(-0.1/2.0) = 0

(int)((pos_x[i]+lado)/lado) -> (int)(4.1/2.0) = 2
(int)((pos_x[i]+lado)/lado) -> (int)(2.1/2.0) = 1
(int)((pos_x[i]+lado)/lado) -> (int)(1.9/2.0) = 0

(int)((pos_x[i]+lado)/lado) - 1 -> (int)(4.1/2.0)-1 = 1
(int)((pos_x[i]+lado)/lado) - 1 -> (int)(2.1/2.0)-1 = 0
(int)((pos_x[i]+lado)/lado) - 1 -> (int)(1.9/2.0)-1 = -1

pos_x[i] = pos_x[i] + vel_x[i]*h;
pos_x[i] = pos_x[i] - lado*escape;
```

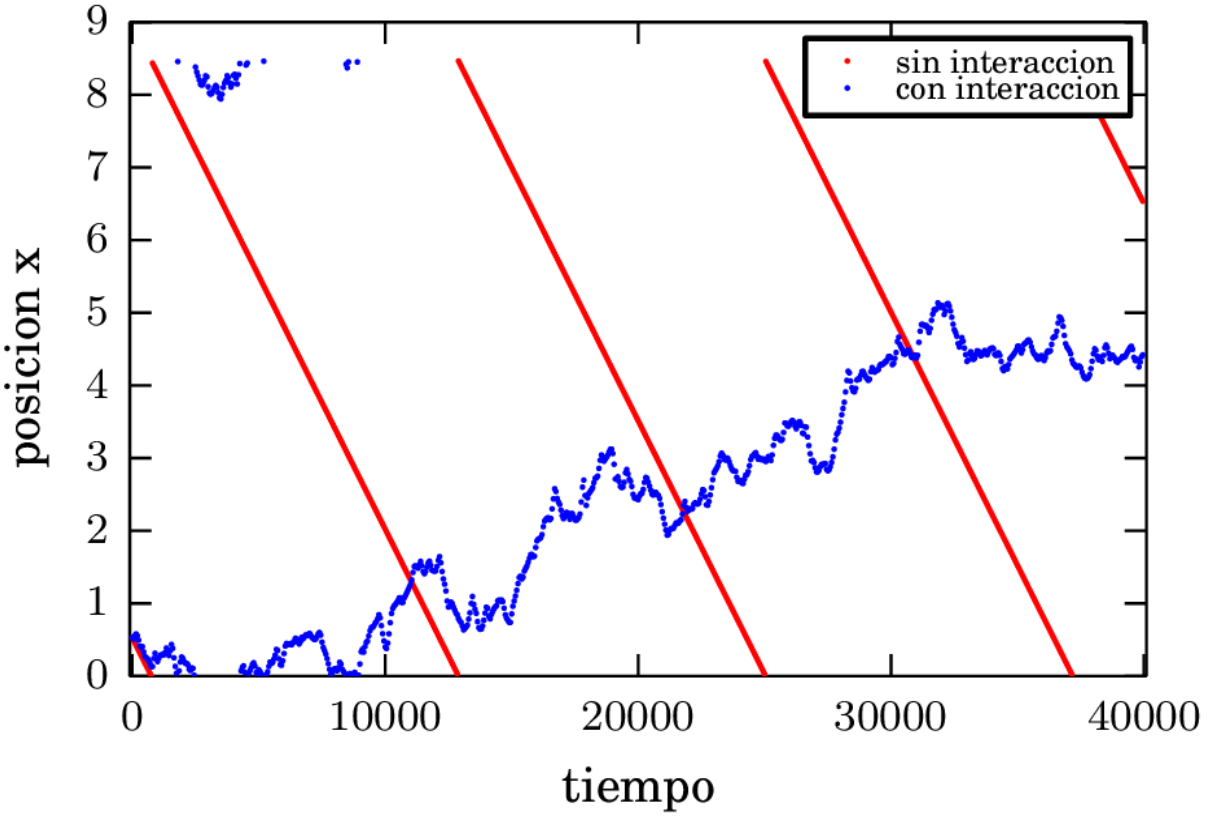
Condiciones de contorno: efecto Mario Bros

Coordenada x de una partícula en función del tiempo



Condiciones de contorno: efecto Mario Bros

Coordenada x de una partícula en función del tiempo

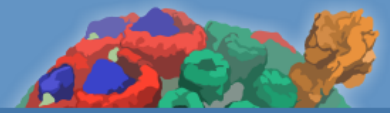


VMD (Visual Molecular Dynamics)

NIH CENTER FOR MACROMOLECULAR MODELING & BIOINFORMATICS | UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Type Keywords SEARCH

THEORETICAL and COMPUTATIONAL BIOPHYSICS GROUP



Home | Research | Publications | Software | Instruction | News | Galleries | Facilities | About Us

- Home
- Overview
- Publications
- Research
- Software
  - VMD Molecular Graphics Viewer
  - NAMD Molecular Dynamics Simulator
  - BioCoRE Collaboratory Environment
  - MD Service Suite
  - Structural Biology Software Database
  - Computational Facility
- Outreach

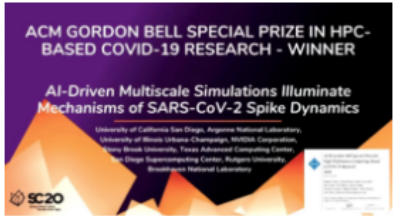
VMD Mailing List

Download VMD

VMD Visual Molecular Dynamics

VMD is a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting. VMD supports computers running MacOS X, Unix, or Windows, is distributed free of charge, and includes source code.
(more details...)

Breaking News



NAMD and VMD are part of the team winning the **2020 ACM Gordon Bell Special Prize** for high performance computing-based **COVID-19 research**, for the paper **AI-Driven Multiscale Simulations Illuminate Mechanisms of SARS-CoV-2 Spike Dynamics**, presented at Supercomputing 2020, Nov 18, 2020.

Spotlight

On April 20th, 2017, the **Journal of Physical Chemistry** published a **Memorial Issue in honor of Klaus Schulten** gathering more than 60 research articles.

VMD (Visual Molecular Dynamics)

Software Downloads

Download VMD:

VMD is a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting. Visit the **VMD website** for complete information and documentation.

Selecting an archive below will lead to a user registration and login page. Your download will continue after you have registered or logged in.

Version 1.9.4 LATEST ALPHA (2020-12-21) Platforms:

Latest pre-release ALPHA test version

- **Source Code**
- **LINUX_64 OpenGL, CUDA, OptiX, OSPRay** (Linux (RHEL 6.7 and later) 64-bit Intel/AMD x86_64 SSE, with CUDA 9.x, OptiX, OSPRay)
- **MacOS 11.x, ARM64 (64-bit "M1" Macs)** (Apple MacOS-X 11 or later)
- **MacOS 10.15, x86_64 (64-bit Intel x86_64)** (Apple MacOS-X 10.15 or later)
- **MacOS X OpenGL (32-bit Intel x86)** (Apple MacOS-X (10.10.x to 10.13.x) with hardware OpenGL (native bundle))
- **Windows 32-bit (32-bit Intel x86)** (Windows 10)
- **Windows 64-bit, CUDA, OptiX, OSPRay (64-bit Intel x86_64)** (Windows 10)

Version 1.9.3 (2016-11-30) Platforms:

We recommend that all users upgrade to VMD 1.9.3

- **Source Code**
- **LINUX_64 OpenGL, CUDA, OptiX, OSPRay** (Linux (RHEL 6.7 and later) 64-bit Intel/AMD x86_64 SSE, with CUDA 8.x, OptiX, OSPRay)
- **LINUX_64 Text-mode w/ EGL** (Linux (RHEL 6.7 and later) 64-bit Intel/AMD x86_64 w/ SSE, Text-mode w/ EGL)
- **LINUX_64 Text-mode** (Linux (RHEL 6.7 and later) 64-bit Intel/AMD x86_64 w/ SSE, Text-mode)
- **LINUX MIC-AVX512 Text-mode** (Linux (RHEL 6.7 and later) 64-bit Intel Xeon Phi MIC w/ AVX-512, Text-mode, OSPRay)
- **LINUX MIC-AVX512, OpenGL, CUDA, OptiX, OSPRay** (Linux (RHEL 6.7 and later) 64-bit Intel Xeon Phi MIC w/ AVX-512, OpenGL, CUDA7.5, OptiX, OSPRay)
- **LINUX OpenPOWER Text-mode** (Linux 64-bit IBM OpenPOWER w/ VSX, Text-mode)
- **MacOS X OpenGL (32-bit Intel x86)** (Apple MacOS-X (10.4.7 to 10.13.x) with hardware OpenGL (native bundle))
- **Windows OpenGL, CUDA** (Windows XP/Vista/7/8/10 (32-bit) with OpenGL and CUDA)
- **Windows OpenGL (32-bit Intel x86)** (Microsoft Windows XP/Vista/7/8/10 (32-bit) using OpenGL)
- **NCSA Blue Waters (Cray XK7 w/ OpenGL)** (NCSA Blue Waters (Cray XK7) MPI, CUDA, OpenGL Pbuffers, TachyonL-OptiX)
- **ORNL Titan (Cray XK7)** (ORNL Titan (Cray XK7) MPI, CUDA, TachyonL-OptiX)
- **CSCS Piz Daint (Cray XC50 w/ EGL)** (CSCS Piz Daint (Cray XC50) MPI, CUDA, EGL Pbuffers, TachyonL-OptiX)

VMD: guía de instalación (versión criolla)

http://gebi.df.uba.ar/

GEBI

GEBI @ Buenos Aires University

Home | Divulgation | Former members | Forum | Forum | Forum | Outreach | People | Posters | Publications | Research | Support

Simulations of pedestrian dynamics

Posted on 13 November, 2017 by Guillermo Frank

In Divulgation can be find simulations of pedestrian dynamics. *(Diferentes simulaciones de dinámica peatonal pueden hallarse en la solapa Divulgación.)*

Posted in Sin categoría | Leave a comment

Posters at the DF Abierto 2017

Posted on 13 November, 2017 by Guillermo Frank

Search

Archives

- November 2017
- April 2016

Meta

- Log in

VMD: guía de instalación (versión criolla)

http://gebi.df.uba.ar/

Support

Howto's by F. Cornes, I. Sticco and G. Frank

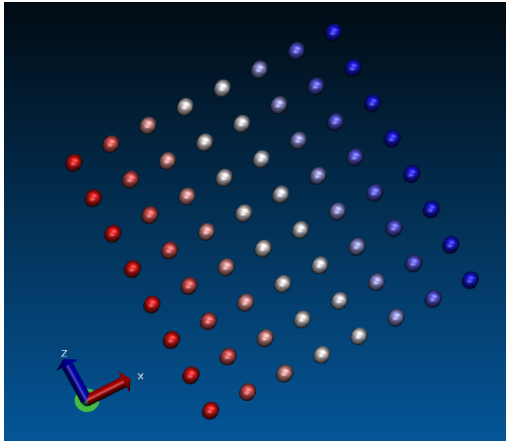
System configuration (from scratch)	.pdf
Lammps for dummies (step 1)	.pdf
Lammps amenities (step 2)	.pdf
Lammps with CUDA (step 3)	.pdf
Trick for e-mail notifications	.pdf
VMD installation (step 1)	.pdf

Pasemos a la acción...

VMD: lectura y almacenamiento de los datos

```
1   512
2
3   1     0.528343     0.529371     0.529147
4   1     1.587318     0.526940     0.527764
5   1     2.643157     0.529457     0.528675
6   1     3.701050     0.529256     0.529891
7   1     4.759105     0.531290     0.528963
```

```
513   1     6.877462     7.934325     7.935531
514   1     7.933778     7.936970     7.933520
515   512
516
517   1     0.527645     0.529702     0.529254
518   1     1.587518     0.524840     0.526488
519   1     2.641116     0.529874     0.528311
520   1     3.698822     0.529472     0.530742
```



Type x y z

```
fprintf(fp, "%i\t %.6f \t %.6f \t %.6f \n",1,*(pos_x+i),*(pos_y+i),*(pos_z+i));
```

Importante: guardar un único archivo de la siguiente manera: nombre.XYZ