ESPResSo una herramienta para la docencia e investigación

A. D. González-Martínez
Departamento de Física, Universidad Autónoma Metropolitana-Iztapalapa, 09340. México, CDMX.

S. Herrera Velarde
Subdirección de Postgrado e Investigación, Instituto Tecnológico Superior de Xalapa, Sección 5A Reserva Territorial s/n, 91096 Xalapa, Veracruz, México.

J. A. Moreno-Razo
Departamento de Física, Universidad Autónoma Metropolitana-Iztapalapa, 09340. México, CDMX.

Received 2 March 2020; accepted 30 May 2020

Este trabajo es una guía para la enseñanza de la dinámica molecular usando el software ESPResSo [1,2]. Las simulaciones de dinámica molecular involucran conocimientos multidisciplinarios; por ello, un software como ESPResSo es de gran utilidad, ya que antepone el aprendizaje de los estudiantes para comprender la física detrás de los algoritmos computacionales. Se muestra el uso básico del software con ayuda de la implementación de sistemas tipo Lennard-Jones en 2 y 3 dimensiones. Además de los sistemas puros, también se abordan conceptos de mezclas y sistemas confinados en cavidades. al involucrar la visualización y manipulación del sistema en 3 etapas (preparación, cálculo y análisis) se caracterizan los sistemas calculando los observables estándar, por ejemplo, la función de distribución radial y el desplazamiento cuadrático medio. Finalmente, se comparan los resultados con datos de la literatura y se demuestra que ESPResSo es una herramienta muy útil para la eseñanza, así como para la investigación.

Descriptores: Simulaciones de dinámica molecular; Potencial de Lennard-Jones; ESPResSo; Enseñanza para estudiantes de física no graduados.

This work is a guide to approach the teaching of Molecular Dynamics by using the software suite ESPResSo [1,2]. Molecular Dynamics simulations require multidisciplinary information. ESPResSo is well-suited because it prioritizes student learning to understand the physics behind computational algorithms. Basic use of the software is illustrated by implementing Lennard-Jones systems in two and three dimensions. In addition to pure systems, the concepts of mixtures and systems confined in cavities are also treated. Systems are visualized and manipulated in three stages: preparation, computation, and analysis. Systems are characterized by calculating standard observables, such as the radial distribution function and the mean square displacement. Finally, results are compared with data from the literature, thus establishing the usefulness of ESPResSo as a tool for teaching and research.

Keywords: Molecular dynamic simulations; Lennard-Jones potential; ESPResSo; physics simulations for undergraduates.

DOI: https://doi.org/10.31349/RevMexFisE.17.226

Introducción

ESPResSo es un software de dinámica molecular de grano grueso enfocado a sistemas que forman parte de la materia condensada blanda. La materia condensada blanda comprende una gran variedad de materiales, como polímeros, coloides, cristales líquidos y tensoactivos. La estructura y dinámica de estos se define en un amplio rango de escalas de longitud desde $\approx 10\mathring{A}$ hasta μm [3], pero tienen una característica en común. En todos estos sistemas, las unidades estructurales son mucho más grandes que los átomos que los constituyen; por ejemplo, las moléculas de polímero generalmente consisten en millones de átomos, los coloides ($\approx 0.1\mu$m) involucran miles de millones de átomos y las moléculas que constituyen los cristales líquidos no son muy grandes pero contienen decenas o cientos de átomos. Todos estos forman estructuras ordenadas que se mueven juntas en una gran unidad [4, 5].

El hecho de que la materia blanda consista en moléculas grandes o conjuntos de moléculas que se mueven colectivamente reduce el número de elementos necesarios para definir el sistema, por ello el modelo de grano grueso es uno de los más exitosos para describir las propiedades de estos materiales. En esta técnica, una molécula representa un grupo de átomos como sitios de interacción, lo que conlleva a la simplificación de las interacciones moleculares y por lo tanto reduce el tiempo de cómputo en las simulaciones [6].

La simulación computacional es una poderosa herramienta que se constituye como un tercer paradigma en la ciencia, al mismo nivel que la teoría y el experimento [7]. En particular, la metodología de Dinámica Molecular (**DM**) se ha convertido en las últimas décadas en una aliada muy poderosa en diversas áreas científicas, tales como la física, biología y química. Avances en campos como nanotecnología, nanomedicina, industria farmacéutica, materiales, entre otros, no hubieran sido posibles sin la contribución del denominado microscopio computacional, esto es, la combinación de dinámica molecular y visualización científica.

La técnica de **DM** tiene su origen en el trabajo para un sistema de esferas duras de Alder y Wainwrigth, publicado poco

después de finalizar la Segunda Guerra Mundial [8]. Desde entonces, la **DM** ha evolucionado al mismo paso que el hardware; las técnicas son cada vez más sofisticadas con modelos más precisos que permiten estudiar innumerables sistemas moleculares desde distintas perspectivas; no obstante la idea subyacente sigue siendo la misma, *resolver las ecuaciones de movimiento* de un conjunto de moléculas y extraer observables físicas del sistema.

El modelo más simple de un fluido consiste en definir el conjunto de **N** partículas, las cuales están contenidas en un volumen **V** e inmersas en un baño térmico que mantiene el sistema en conjunto a una temperatura constante **T**. Al definir las variables de control se define el ensamble termodinámico, y este dicta la naturaleza del sistema; por ejemplo, cuando se fija la temperatura, se establece el intervalo de velocidades para las partículas a través de la distribución de velocidades de Boltzmann. Al determinar la velocidad y la posición inicial del conjunto canónico se define el estado inicial del sistema; de estas condiciones parte el algoritmo de DM. Al calcular nuevas posiciones y velocidades en cada intervalo de tiempo δt [9], los modelos más complejos necesitarán establecer parámetros específicos como: masa, carga, orientación, momento angular, especie, entre otros.

El modelo de grano grueso más simple considera pares de partículas esféricas idénticas que interactúan ejerciendo fuerzas centrales entre sí, las fuerzas cumplen la tercera ley de Newton en su forma fuerte y no dependen de la velocidad de las partículas ni del tiempo explícitamente, de forma que la fuerza es conservativa y tiene asociado un potencial de interacción. La interacción pueden ser de corto o largo alcance y atractiva o repulsiva. Para modelar partículas cargadas se utilizan potenciales tipo Coulomb, para modelar fluidos simples se puede emplear el potencial tipo Lennard-Jones y para modelar cristales líquidos existe el potencial de Gay-Berne [10]. Cada modelo implica distintos detalles, los cuales se definen a través de parámetros; para ello, ESPResSo diferentes potenciales de interacción y de enlace, como en el caso de polímeros.

Hay dos aspectos principales que maximizan la eficiencia de la simulación, por un lado están los métodos numéricos y los algoritmos útlizados para resolver la dinámica de N cuerpos que interactúan por pares de partículas, es decir, el integrador y los algoritmos de implementación como listas de vecinos y radios de corte. Por otra parte, se tienen los conceptos específicos de hardware y la implementación de nodos en paralelo, ya sea a nivel de CPU [11] o GPU [12]. ESPResSo cuenta con metodologías bastante robustas, en particular para el cálculo de interacciones electrostáticas y para calcular de manera eficiente interacciones de largo alcance para sistemas con diferente periodicidad. ESPResSo es de los pocos softwares que permiten el cálculo de este tipo de interacción en una, dos y tres dimensiones [1, 2]. De esta manera, se puede crear un sistema de polímeros neutros o cargados o sistemas cargados confinados, por mencionar algunos ejemplos.

Características de ESPResSo

Afortunadamente, en la actualidad los científicos tienen a su disposición diversos paquetes de simulación de **DM** para el estudio de múltiples sistemas. En particular, ESPResSo posee ciertas características que lo hacen único y lo diferencian de otros paquetes de simulación computacional [2]; a diferencia de otros paquetes, el exito de ESPResSo viene de no utilizarlo como una caja negra [7], dado que es un paquete de simulación de uso gratuito que puede descargarse desde la página oficial[i]. Se recomienda descargar la última versión, puesto que las más antiguas quedarán obsoletas. En ESPResSo se pueden estudiar diversos sistemas, como esferas duras, coloides cargados, polímeros, sistemas confinados, cristales líquidos, entre muchos otros.

El usuario tiene acceso directo al código fuente, el cual está escrito en C, de manera que puede compilarse en una gran variedad de plataformas, desde computadoras personales hasta clusters con cientos de nodos. Para su uso y compilación ESPResSo requiere de otros paquetes de uso libre y puede instalarse únicamente en sistemas operativos Linux y Unix. Se recomienda compilar según el tipo de sistemas que se requiera simular. Es posible compilar utilizando todas las características de ESPResSo, sin embargo, esto puede resultar en un código más lento al correr las simulaciones, sobre todo para las versiones más antiguas. Las instrucciones para compilar ESPResSo se pueden consultar en la guía de usuario o bien en la página oficial. Aquí se asume que el software ya esta trabajando.

Para realizar una simulación de DM con ESPResSo, es necesario un archivo (*script*) que controla la simulación; este *script* se puede escribir en el lenguaje de programación Tcl o para las últimas versiones de Python [2]. En este archivo de control se definen los parámetros físicos del sistema, tales como el número y tipo de partículas, el tipo de interacciones entre partículas, la temperatura del sistema, el paso del tiempo, la duración de la simulación entre otros. ESPResSo fuerza al usuario a dar valores a todos los parámetros y a crear el sistema en unidades reducidas, por ello ESPResSo no puede utilizarse como una caja negra [7]. El hecho de que ESPResSo sea un software que requiere del usuario un cierto nivel de conocimiento de **DM**, lo convierte en una herramienta ideal de apoyo en la docencia para introducir a estudiantes de licenciatura a la simulación computacional, sin preocuparse, al menos en un primer acercamiento, por la implementación numérica de los algoritmos para calcular fuerzas, velocidades, posiciones o mantener constante la presión o temperatura. ESPResSo es una opción idónea para un primer acercamiento a la simulación computacional en carreras como física, química y bioquímica. Si se complementa con un software de visualización como Visual Molecular Dynamics [13], puede emplearse en el aula para mejorar la comprensión de los estudiantes en temas como la naturaleza atómica de la materia, la estructura molecular de los materiales, las transiciones de fase y los procesos de difusión, por mencionar algunos ejemplos.

En este trabajo se presenta el diseño de un archivo de control con el lenguaje de programación python e implementado a un sistema simple de partículas tipo Lennard-Jones (LJ) en 3 dimensiones, así como, la construcción del archivo de control en el lenguaje de programación TCL para un sistema de LJ 2D y confinado. De esta forma, el usuario puede comparar y ajustarse a cualquiera de los dos lenguajes. De nueva cuenta, se sugiere utilizar la última versión del software.

Diseño de un archivo para el sistema tipo Lennard-Jones

Para implementar una simulación en ESPResSo, primero se debe crear el archivo de control. Este se puede escribir en cualquier editor de textos sin formato utilizando el lenguaje Tcl o en Python. En el archivo se reconocen dos niveles de instrucciones, entre las propias del software ESPResSo llamando a funciones o rutinas programadas en C y las instrucciones del lenguaje utilizado (Tcl o Python) que manipulan variables a nivel local y no se comunican con el código fuente. El primer ejemplo a explorar es un sistema de partículas en 3D tipo LJ. Este sistema se ha estudiado ampliamente numeroso trabajos, por ello es posible comparar los resultados obtenidos con ESPResSo con aquellos ya reportados [9]. En primer lugar, se definen las variables del ensamble y algunas constantes de la simulación. Para el ensamble NVT se determina el número de partículas N, la temperatura T, y el volumen V. Este último se establece a través de la densidad ρ como, V = N/ρ:

Python script:
_____ LENNARD JONES _____ Parte I

```
import espressomd
import numpy as np
required_features = ["LENNARD_JONES"]
# Número de partículas
nparticulas = 1000
# temperatura
temperatura = 0.71
# densidad
rho = 0.8442
# dimensiones de caja cuadrada
volumen = nparticulas/rho
boxl = volumen**1/3
```

Las primeras lineas incluyen los paquetes y módulos que se utilizarán. Los requerimientos dependen del sistema, por ejemplo, se podría solicitar el uso de fuerzas externas **[EXTERNAL_FORCES]** o de un potencial tipo Coulomb **[ELECTROSTATICS]**. Las variables **número partículas (nparticulas), temperatura, densidad, volumen y longitud de la cada (boxl)** no están definidas en ESPResSo, únicamente son variables que ayudan a estructurar el sistema.

ESPResSo reconoce las variables cuando se escriben como argumento de funciones propias, por ejemplo: la variable temperatura se utiliza para definir el valor en unidades reducidas de la temperatura del sistema y el número de partículas `nparticulas` es necesario para definir la configuración inicial. Es decir, el código fuente de ESPResSo aún no conoce éstas variables pues son más bien de utilidad para el usuario. Para darle a conocer al código fuente (a ESPResSo) los valores de los parámetro de la simulación, se utilizan comandos. Por ejemplo, para enviarle a ESPResSo las condiciones periódicas de frontera se utiliza el comando **espressomd.System**. En este ejemplo se simula un sistema con periodicidad en las tres dimensiones, entonces la segunda parte del *script* es Python script:
_____ LENNARD JONES _____ Parte II

```
#dimensiones de la caja y periodicidad
system = espressomd.System(
box_l=[boxl,boxl,boxl],
 periodicity=[1,1,1])
```

A partir de esto es posible definir el potencial de interacción tipo Lennard-Jones U_{LJ} [14], el cual depende de dos parámetros: σ_0 establece la distancia mínima entre pares de partículas y ϵ_0 modula la profundidad de la interacción. La expresión matemática del potencial entre las partículas i,j está dada por la Ec. (1)

$$\Phi_{LJ}(r_{ij}) = 4\epsilon_0\left[\left(\frac{\sigma_0}{r_{ij}}\right)^{12} - \left(\frac{\sigma_0}{r_{ij}}\right)^{6}\right], \qquad (1)$$

donde $r_{ij} = |\vec{r}_i - \vec{r}_j|$ es la distancia entre las posiciones $\vec{r}_i$, $\vec{r}_j$, de las partículas i j, respectivamente y $\vec{F}_{ij} = -\nabla U_{LJ}(r_{ij})$ es la fuerza sobre la partícula i debido a su interacción con la partícula j cuando están separadas a una distancia r_{ij}. Como se presenta en la Fig. 1, es conveniente recorrer el potencial un factor ϵ_0, e ignorar la cola atractiva como

$$U_{LJ}(r_{ij}) = \begin{cases} \Phi(r_{ij}) - \Phi(r_c) & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c \end{cases}. \qquad (2)$$

El radio de corte r_c típico para Lennard-Jones es de $2.5\sigma_0$. En la Fig. 1 se muestra la diferencia entre el potencial de Lennard-Jones $\Phi_{LJ}(r_{ij})$ y el *potencial truncado y movido* $U_{LJ}(r_{ij})$. El radio de corte es útil para ahorrar tiempo de cómputo cuando los potenciales son de interacción de corto alcance.

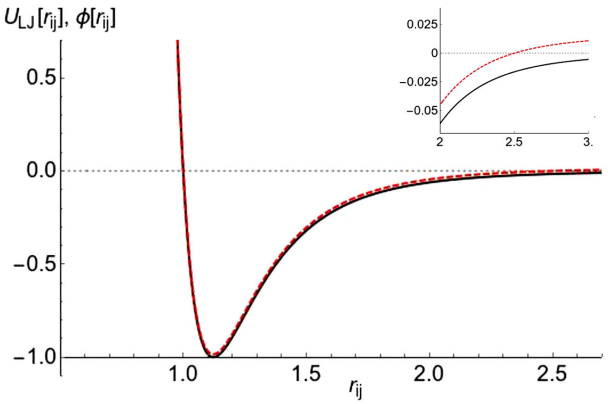

FIGURA 1. Potencial de interacción tipo Lennard-Jones $\Phi(r_{ij})$ (línea continua) y $U_{LJ}(r_{ij})$ (línea a trazos) para $\sigma_0 = 1.0$, $\epsilon_0 = 1.0$ y $r_c = 2.5\sigma_0$.

La fuerza total $\vec{F}_i(r_{ij})$ sobre la partícula i es la suma de las interacciones de todas las partículas restantes que componen el sistema, Ec. (3). Sin embargo, dado el alcance del potencial, solo interesa calcular la fuerza sobre la partícula i de aquel conjunto de partículas que cumplen la condición de radio de corte $r_{ij} \leq r_c$, En otras palabras cada partícula posee un conjunto de partículas vecinas con las cuales interactúa hasta una cierta distancia predefinida r_c, denominado lista de Verlet [9]. La lista de Verlet se calcula con el radio de Verlet o de lista $r_{list} > r_c$, que debe ser mayor que el radio de corte; esta lista no se calcula en cada iteración. ESPResSo permite establecer el radio de Verlet y con ello de manera interna se actualiza la lista de vecinos de forma automática. El valor de este parámetro depende del alcance del potencial de interacción, por esto se recomienda ejecutar corridas de prueba para determinar el valor mediante el cual se obtienen mejores resultados.

$$\vec{F}_i(r_{ij}) = \sum_{j=1, j\neq i}^{N} \vec{F}_{ij} = -\sum_{j=1, j\neq i}^{N} \nabla U_{LJ}(r_{ij}) \qquad (3)$$

Ahora es posible definir en el *script* el tipo de interacción como Lennard-Jones, utilizando los parámetros: σ_0, ϵ_0, el radio de corte y el radio de la lista de vecinos.

Python script:
LENNARD JONES Parte III

```
#parámetros del potencial LJ
epsilon0 = 1.0
sigma0   = 1.0
#radio de corte
rcut= 2.5*sigma0
#lista de vecino
system.cell_system.skin = 0.4
#interacción entre partículas tipo 0
system.non_bonded_inter[0, 0].
 lennard_jones.set_params(epsilon=epsilon0,
                          sigma=sigma0,
                          cutoff=rcut,
                          shift="auto")
```

El tipo de partícula es un número entero y adquiere mayor relevancia en la simulación de mezclas de partículas, en cuyo caso se deberán definir todas las interacciones posibles entre partículas del mismo tipo y entre partículas distintas; por ejemplo, se considera tener partículas de tipo "0" de diámetro σ_0 = 1.0 y partículas de tipo "1" con σ_0 = 2.0. Entonces, las partículas del tipo "0" son las definidas en la parte III, y para las partículas del tipo "1" se definen nuevos parámetros σ_1, ϵ_1, $rcut_1$ y $schift_1$ e incluyen las interacciones siguientes,

```
#interacción entre partículas tipo 1
system.non_bonded_inter[1, 1].
 lennard_jones.set_params(epsilon=epsilon1,
                          sigma=sigma1,
                          cutoff=rcut1,
                          shift="auto")
#interacción entre partículas tipo 0 y 1
system.non_bonded_inter[0, 1].
 lennard_jones.set_params(epsilon=epsilon01,
                          sigma=sigma01,
                          cutoff=rcut01,
                          shift="auto")
```

Para la interacción entre las especies (cruzada), se pueden utilizar las reglas de combinación de Lorentz-Berthelot [15] que definen los parámetros σ_{01}, ϵ_{01}, $rcut_{01}$ y $schift_{01}$.

De vuelta al sistema monodisperso en 3D, el siguiente paso es crear la configuración inicial, para ello existe la alternativa de crear una configuración al azar o partir de un arreglo cristalino. Por ejemplo, para una configuración sin traslapes y ordenada se puede utilizar:

Python script:
LENNARD JONES Parte IV

```
n = int(boxl/lj_sigma)
contador = 0
for k in range(0,n):
 rz = k*sigma0
 for j in range(0,n):
  ry = j*sigma0
  for i in range(0,n):
   rx = i*sigma0
   if contador $<$ nparticulas:
   posi = [rx, ry, rz]
#asignando posiciones en ESPResSo
   system.part.add(id=cont,pos=posi)
   contador += 1
```

mientras que para una configuración al azar donde podrían existir traslapes se sustituye la asignación de posiciones anterior por:

```
for i in range(0, nparticulas):
 system.part.add(id=i,
   pos=numpy.random.random(3) * boxl)
```

En este último caso, previo a la simulación, se eliminan los posibles traslapes entre las partículas. El procedimiento para generar una configuración físicamente aceptable se presenta en el siguiente ejemplo "Lennard-Jones en 2D confinado". La capacidad de ESPResSo para remover traslapes, es bastante útil al generar configuraciones iniciales de sistemas muy densos o con partículas de formas anisotrópas. En la Fig. 2 se muestra la diferencia entre las configuraciones iniciales ordenada y al azar.

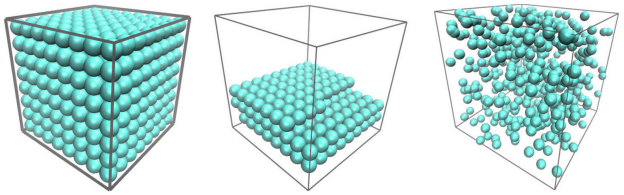

FIGURA 2. Configuraciones iniciales de partículas tipo Lennard-Jones para una configuración cuadrada (a) con $\rho = 1.0$, (b) $\rho = 0.3$ y (c) para una configuración al azar con $\rho = 0.7$.

En este ejercicio, tras crear la configuración inicial (parte IV) se itera el sistema utilizando el termostato de Langevin; durante un ciclo de *relajación*, para monitorear de la presión, energía y temperatura del sistema, como:

Python script:
_____ LENNARD JONES _____ Parte V

```
#termostato
system.thermostat.
 set_langevin(kT=temperatura,gamma=1.0)
#paso de integración
system.time_step = 0.001
#parámetros del ciclo
nciclos = 1000
nxciclo = 100
#comienzo de relajación
for paso in range(0, nciclos):
 #integrador
 system.integrator.run(nxciclo)
 #cálculo de temperatura y energía
 energias = system.analysis.energy()
 presion = system.analysis.
           pressure()["total"]
 etot = energias["total"]/nparticulas
 ek   = energias["kinetic"]/nparticulas
 #monitor de temperatura y energía
 print(paso,system.time,etot,ek,presion)
```

La instrucción **`system.time`** devuelve el tiempo de simulación e **`[integrator.run]`**(*nxciclo*) notifica a ESPResSo la necesidad de mover a las partículas (integrar las ecuaciones de movimiento) *nxciclo* veces en cada paso. Esto resulta útil para guardar configuraciones con un cierto espaciamiento y posteriormente crear una animación, o bien, para calcular observables. Podría ser necesario incrementar el ciclo de relajación, sin embargo esto depende de la configuración inicial, del grado de confinamiento o de la densidad del sistema. Para ello, se debe aumentar el valor de nxciclos, aunque también resulta equivalente modificar de forma apropiada el valor de nxciclo.

Una vez que la energía, temperatura o presión fluctúan alrededor de un valor aceptable, esto es, cuando se haya alcanzado una configuración de equilibrio, es posible medir los observables del iniciar el ciclo principal de DM. El ciclo DM es similar al ciclo de relajación, pero se guardan las posiciones de todas las partículas, generando la trayectoria que siguen todas las partículas en la ventana temporal que cubre el ciclo de producción. Para almacenar las configuraciones de las partículas se utiliza el comando de ESPResSo **`system.analysis.append()`**. Además se genera un archivo para guardar la trayectoria, para después visualizarla con algún software, por ejemplo, VMD. Por esta razón, se utiliza el formato *xyz*, que consiste simplemente en escribir los valores de las tres coordenadas. En particular, para el software VMD se tiene:

Python script:
_____ LENNARD JONES _____ Parte VI

```
#archivo de trayectoria
trayectoria = open("trayectoria.xyz",
                   mode="w")
#parámetros del ciclo
system.time_step = 0.001
nciclos= 50000
nxciclo= 100
#letrero de monitor
print("i time E_total E_cinética Presión")
#ciclo DM
for i in range(0, nciclos):
 #integrador
 system.integrator.run(nxciclo)
 #cálculo de energía y presión
 energias = system.analysis.energy()
 presion  = system.analysis.
            pressure()["total"]
 etot = energias["total"]/nparticulas
 ek   = energias['kinetic"]/nparticulas
 #monitor
 print(i,system.time,etot,ek,presion)
 #almacena configuraciones
 system.analysis.append()
 #Guardar película
 trayectoria.write("%s/n' %(nparticulas)")
 trayectoria.write("configuración%s/n"%(i))
 for p in system.part:
  posi=p.pos_folded
  trayectoria.write("H %s %s %s /n" %
             (posi[0],posi[1],posi[2]))
trayectoria.close()
```

En el archivo "trayectoria.xyz" se escribirá la trayectoria del sistema, guardando las posiciones de las partículas cada *nxciclo dt*. El software VMD [13] es muy cómodo para visualizar el sistema, pero también es posible en otros formatos y utilizar software como OVITO [16] o POV-Ray [17]. Se debe tener precaución para simulaciones muy largas, dado que pueden generarse archivos muy grandes ($\mathcal{O}$ (Gb)). El ciclo DM puede tardar desde algunos minutos hasta horas; esto depende del número de partículas, los tipos de interacciones entre partículas y/o del tiempo de simulación establecido en las variables nciclos y nxciclo.

Durante el ciclo de producción DM ESPResSo guarda las configuraciones en memoria. Al terminar, se pueden calcular algunas propiedades del sistema, por ejemplo: la función de distribución radial (Radial Distribution Function, RDF) y el desplazamiento cuadrático medio (MSD). Se debe considerar que, para el correcto cálculo de propiedades dinámicas, las posiciones almacenadas no deben tener la restricción de las condiciones periódicas folded. Entonces, para calcular la RDF, se debe modificar el *script* y añadir el cálculo.

Python script:
_____ LENNARD JONES _____ Parte VII

```
#cálculo de la fdr
r, fdr=system.analysis.rdf(
            rdf_type="<rdf>",
            type_list_a=[0],
            type_list_b=[0],
            r_min =0.0,
            r_max =boxl*0.5,
            r_bins=100)
```

```
fdr_file = open("fdr.dat", "w")
for i in range(100):
 fdr_file.write("%1.5e%1.5e/n"%(r[i],fdr[i]))
fdr_file.close()
```

La RDF es una cantidad muy útil y común en materia condensada blanda. Originalmente, se utiliza para caracterizar la estructura interna de un sistema molecular puro como un gas, líquido o sólido, ya que ellos exhiben características completamente diferentes. La RDF describe la distribución de distancias entre pares de partículas contenidas dentro de un volumen dado en 3D o de un área en 2D. Matemáticamente, si i y j son dos partículas en un fluido, la RDF de j con respecto a i, denotada por $g(r_{ij})$, es la probabilidad de encontrar la partícula j a la distancia r_{ij} de i, tomando a la partícula i como origen de coordenadas. En mecánica estadística, la RDF (o también conocida como función de correlación a pares) en un sistema de partículas (átomos, moléculas, coloides, etc.) describe cómo varía la densidad como función de la distancia desde una partícula de referencia. En términos más simples, es una medida de la probabilidad de encontrar una partícula i una distancia $|r|$ de una partícula de referencia dada, relativa a un gas ideal. La Fig. 3 muestra la RDF para un fluido tipo Lennard-Jones a distintas densidades y temperaturas.

En la Fig. 3 se observa que, para fases muy diluidas $(\rho = 0.20)$, la función $g(r)$ muestra que a distancias cercanas $r \approx \sigma_0$ es probable encontrar hasta dos partículas vecinas; en una fase menos diluida (líquida, $\rho = 0.60$) se pueden seguir encontrando vecinas a distancias de $r = 2.0\sigma_0$, en las fases solidas y cristalinas, la función de distribución radial tiene partículas vecinas hasta distancias de $r > 3.0\sigma_0$ [9].

En 1827, un botánico inglés llamado Robert Brown, llevó a cabo una serie de experimentos utilizando el polen de la planta *Clarkia Pulchella*; observó que esas partículas de polen en agua presentaban un movimiento muy errático y en zigzag. Este fenómeno intrigó a muchas de las más grandes

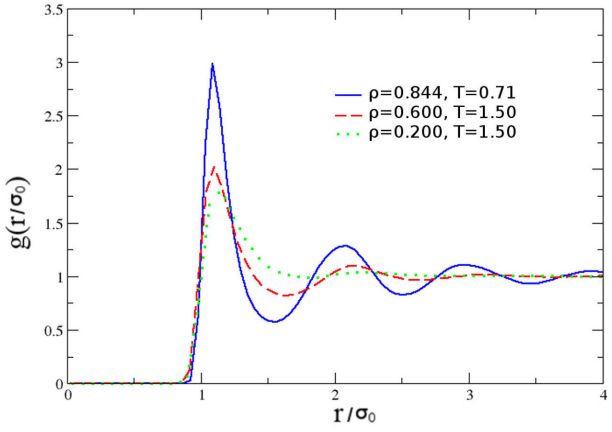

FIGURA 3. Función de correlación radial de un sistema Lennard-Jones modisperso en 3D para: (línea continua) $\rho = 0.844$ y $T = 0.71$, (línea a trazos) $\rho = 0.6$ y $T = 1.5$, (línea punteada) $\rho = 0.2$ y $T = 1.5$.

mentes de la Física tales como Michael Faraday, Albert Einstein y Jean Perrin. Robert Brown creyó que ese movimiento se debía a que el polen tenía vida, así que repitió el experimento con otros granos de polen provenientes de otras plantas y obtuvo el mismo fenómeno. Esa clase de movimiento no tuvo explicación durante casi medio siglo hasta el desarrollo la teoría cinética. En esa época, se pensaba que el movimiento (browniano) se debía a corrientes de convección o vibraciones que se transmitían a través del líquido. Fue hasta principios del siglo XX cuando se comprendió finalmente la verdadera causa del movimiento browniano: ese movimiento errático de los granos de polen se debía a las colisiones de las moléculas del fluido en donde estaba suspendido [18].

El movimiento browniano de una partícula suspendida en un fluido es el resultado de muchos movimientos irregulares pequeños, los cuales se deben a las colisiones moleculares con el fluido donde esta suspendida; entonces, el desplazamiento cuadrático medio correspondiente a esos movimientos es proporcional al número de colisiones de la partícula y, por ello es proporcional al tiempo. En 1905, Einstein aplicó la teoría cinética al cálculo del desplazamiento cuadrático medio de una gran esfera de radio R que experimenta un movimiento browniano en un gas de viscosidad dada.

En las simulaciones de MD, las trayectorias, velocidades y fuerzas se conocen para todo tiempo y para todas las partículas del sistema, entonces del análisis de estas trayectorias se obtiene una idea del comportamiento dinámico del sistema y permite el cálculo de coeficientes de transporte.

Es muy simple evaluar con ESPResSo el desplazamiento cuadrático medio MSD (Mean Square Displacement), para ello se modifica el *script* y se crea la observable, en este caso por medio de la instrucción **`ParticlePositions`** y **`Correlator`**, y se añade la siguiente instrucción antes de comenzar el ciclo **DM** en la parte VI del *script*;

```
pos_obs = ParticlePositions(ids=(0))
cpos=Correlator(
       obs1=pos_obs,
       tau_lin=16,
       tau_max=100.0,
       delta_N=10,
       corr_operation=
       "square_distance_componentwise",
       compress1="discard1")
```

En el cálculo del MSD con ESPResSo se añadió el algoritmo **tau** de múltiple correlación. La instrucción anterior incluye los valores predeterminados por el manual de usuario, que incluyen los detalles del algoritmo. Análogo a este proceso, se puede crear el observable **`ParticleVelocities`**, **`ParticleForces`**, entre otras opciones. Esto depende del análisis que se busque; con ESPResSo también se puede calcular el perfil de densidad **`DensityProfile`** y el perfil de densidad de flujo **`FluxDensityProfile`**, por mencionar

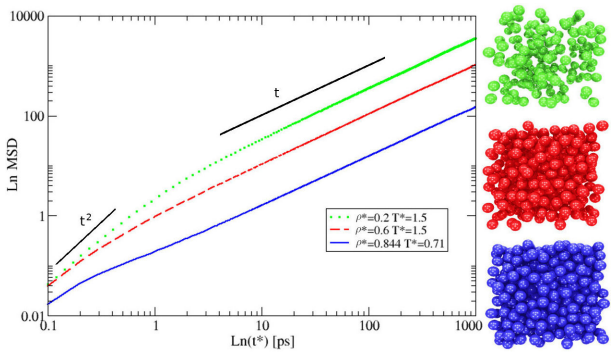

FIGURA 4. Desplazamiento cuadrático medio (MSD) *vs.* t. A tiempos cortos el MSD $\sim t^2$ (régimen balístico)), mientras que a tiempos largos MSD $\sim t$ (régimen difusivo, línea continua gruesa). También se muestran las configuraciones finales para tres sistemas del fluido tipo Lennard-Jones: (línea continua) $\rho = 0.844$, $D = 0.025$ a $T = 0.71$, (línea a trazos) $\rho = 0.6$, $D = 0.170$ a $T = 1.5$ y (línea punteada) $\rho = 0.2$, $D = 0.598$ a $T = 1.5$.

algunos. Por último, se debe finalizar el observable con la instrucción **`cpos.finalize()`** y se guarda el arreglo tiempo $vs.$ MSD, con la instrucción **`np.savetxt("msd.dat", cpos.result( ))`**.

En la Fig. 4 se muestra el cálculo del desplazamiento cuadrático medio MSD para tres diferentes densidades; también se muestra la configuración final de la trayectoria para cada uno de los escenarios. Este ejercicio, aunque es simple, podría servir para introducir de manera sencilla a los alumnos a la simulación computacional, y para fungir punto de partida para discutir y aterrizar los conceptos fundamentales de una simulación de DM. Al momento de construir el archivo de control, se pueden discutir ideas tales como: caja de simulación, densidad y fracción de empaque, radio de corte, condiciones periódicas a la frontera, configuración inicial, relajación el sistema y, por supuesto, uso correcto de las unidades reducidas, por mencionar algunos ejemplos. A partir de aquí, pueden crearse variantes sencillas y otras más complejas, como la siguiente, planteando un ejemplo para un sistema confinado en una cavidad circular en 2D.

ESPResSo-Tcl-script para un sistema Lennard-Jones en 2D confinado

En el siguiente ejemplo se construirá un sistema bidimensional confinado en una cavidad circular. Las partículas inómviles forman el *corral* de confinamiento e interactúan con partículas que se pueden mover en el interior. Por simplicidad, las partículas que forman la pared y las que difunden dentro de la cavidad tienen la misma forma y tamaño. El objetivo principal de este ejemplo es mostrar lo simple que es modificar el archivo control de Lennard-Jones para crear escenarios más complejos.

En este ejemplo solo interesa explorar cómo difunden y se acomodan las partículas al interior de la cavidad acorde al tipo de confinamiento [19]. La primer modificación al *script* se presenta al definir dos especies de partículas: inmóviles y móviles, lo cual se hace al fijar el radio de la cavidad y considerar que el diámetro de las partículas es la unidad, después se calcula el número de partículas inmóviles y con la densidad se calcula el número de partículas móviles:

TCL script:
LENNARD JONES 2D Parte I

```
#temperatura
set tem 1.0
#radio de la cavidad
set radio 15.0
#densidad
set rho 0.5
#número de partículas inmóviles
set inmoviles int([expr $radio*2.0*$pi])
#cálculo de partículas móviles
set pi 3.1416
set area [expr $pi * $radio * $radio ]
set moviles [expr int($area * $rho)]
#partículas totales
set nparticulas [expr $moviles+$inmoviles]
#periodicidad en 2D
setmd periodic 1 1 0
```

Como se pasa de un sistema en el bulto a uno en el plano, se debe cambiar la dimensión de la simulación de 3D a 2D por medio del comando **`setmd periodic`**. La modificación principal para el nuevo sistema consiste en crear las interacciones entre especies. A las partículas que conforman la pared se les denomina partículas inmóviles y se les asignará el tipo 1; aquellas que difunden dentro de la cavidad son las partículas móviles del tipo 0. Las interacciones entre partículas móviles están definidas de manera idéntica a las interacciones del ejemplo anterior (parte III). La interacción entre diferentes tipos de partículas se define de forma similar, de hecho, en el caso más simple, se determina el mismo tipo de interacción de Lennard-Jones con los mismos parámetros que para las partículas tipo 0. De requerirse, se puede delimitar otro tipo de interacciones entre partículas móviles-inmóviles. Finalmente, no es necesario precisar interacciones entre partículas del tipo 1, puesto que se mantendrán fijas durante la simulación.

TCL script:
LENNARD JONES 2D Parte II

```
#parámetros del potencial LJ
set lj_epsilon 1.0
set lj_sigma 1.0
#radio de corte
set lj_rcut [expr 2.5 * $lj_sigma]
#Lista de vecinos
setmd skin 0.4
#ajuste del potencial de LJ
set lj_shift [
  expr [calc_lj_shift $lj_sigma $lj_rcut ]]
#interacción entre partículas móviles
inter 0 0 lennard-jones $lj_epsilon
                        $lj_sigma
                        $lj_rcut
                        $lj_shift
#interacción entre partículas inmóviles
```

```
inter 1 1 lennard-jones $lj_epsilon
                        $lj_sigma
                        $lj_rcut
                        $lj_shift
#interacción entre partículas móvil-inmóvil
inter 0 1 lennard-jones $lj_epsilon
                        $lj_sigma
                        $lj_rcut
                        $lj_shift
```

Al momento de crear la configuración inicial se especifica cuáles partículas son móviles y cuáles permanecen estáticas. Para definir la movilidad de las partículas del tipo 0, se utiliza el comando **fix** 0 0 1, en este caso se le comunica a ESPResSo que este tipo de partícula tiene movimiento restringido en la dirección Z, esto es, las partículas se mueven en el plano XY. El conjunto de tres enteros en el comando **fix**, debe ser congruente con el comando **periodic** 0 0 1. Para las partículas inmóviles del tipo 1, se usa **fix** 1 1 1, asi es, el movimiento está restringido en las tres direcciones:

TCL script:
LENNARD JONES 2D Parte III

```
#posiciones al azar de partículas móviles
set rz 0.0
for {set i 0} {$i < $moviles} {incr i} {
 set rx [expr [t_random]*$radio*
        sin([t_random]*2.0*$pi)]
 set ry [expr [t_random]*$radio*
        cos([t_random]*2.0*$pi)]
 part $i pos $rx $ry $rz
         q 0.0
         type 0
         fix 0 0 1
}
#modelo de cavidad
set theta [expr 2.0*$pi/$inmoviles]
for {set i $moviles}{$i < $nparticulas}{incr i}{
  set s [expr $i*$theta]
  set rx [expr ($radio+$lj_sigma)*sin($s)]
  set ry [expr ($radio+$lj_sigma)*cos($s)]
  set rz 1.5
  #partículas inmóviles
  part $i pos $rx $ry $rz
          q 0.0
          type 1
          fix 1 1 1
}
```

A las partículas móviles se les asignan posiciones aleatorias dentro de la cavidad y a las partículas inmóviles se les otorgan posiciones dando saltos con la longitud de arco de círculo σ. La Fig. 5a) muestra la configuración inicial para un sistema con posiciones al azar. Es posible observar que existen traslapes entre las partículas; por ello, antes de comenzar el ciclo de producción, es necesario asegurarse de debemos asegurarnos que no se presenten traslapes, de lo contrario el sistema *explotaría* y es muy probable que ESPResSo detenga automáticamente la simulación con un mensaje de advertencia. Para quitar los traslapes se utiliza una atenuación del potencial. El comando

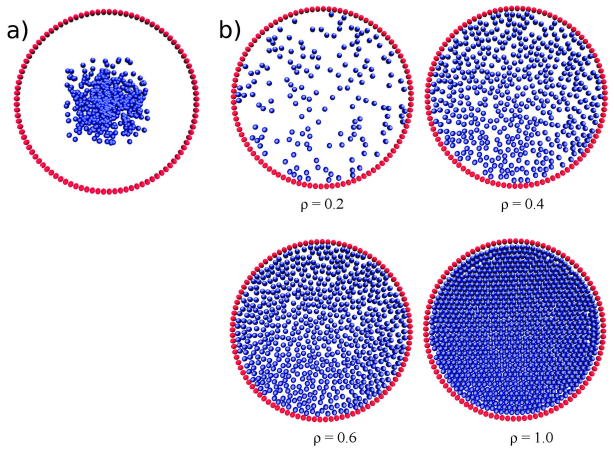

FIGURA 5. a) Configuración inicial con traslapes. b) Configuración final para un sistema Lennard-Jones 2D confinado en una cavidad circular a distintas densidades ρ.

inter forcecap *cap* sirve para incrementar paulatinamente la magnitud del potencial y con ello, la fuerza entre las partículas. Además, se utiliza el comando **analyze mindist** que calcula la distancia de separación mínima entre todos los pares de partículas móviles. Así, es posible detectar si todavía existen traslapes o identificar cuando el sistema alcance una configuración libre de traslape. En este ejemplo, por el tipo de interacción entre partículas, la distancia mínima entre un par de partículas móviles está definida por el parámetro *lj_sigma*; cuando las partículas estén a una distancia mayor o igual a $0.9{*}lj\_sigma$ significa que no existen traslapes y termina el ciclo para remover de los traslapes:

TCL script:
LENNARD JONES 2D Parte IV

```
#incremento de tiempo
setmd time_step 0.01
#termostato
thermostat langevin $temperatura 1.0
#distancia mínima entre partículas móviles
set dist_min [analyze mindist 0 0]
#ciclo para quitar traslapes
set min 0.0
set cap 10
inter forcecap $cap
while {$min < [expr 0.9*$lj_sigma ] }
{
    #integrador
    integrate 50
    #checa traslapes
    set min [analyze mindist [0 0]]
    incr cap 10
    inter forcecap $cap
}
```

La configuración que se obtiene después de remover los traslapes incluye las posiciones de las partículas al azar. Este tipo de configuración es representativa de un estado de equilibrio. Sin embargo, de ser necesario,

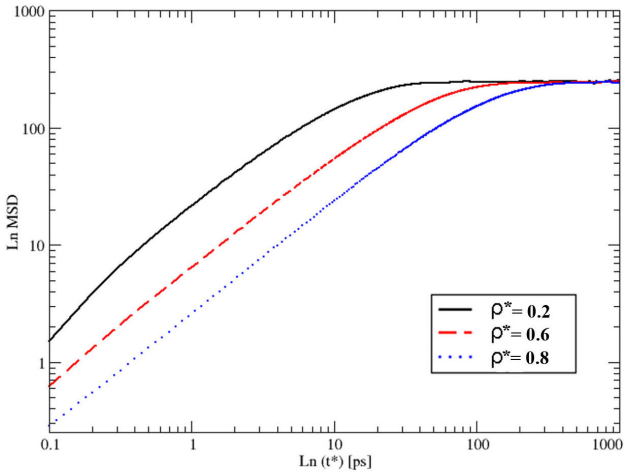

FIGURA 6. Desplazamiento cuadrático medio (MSD) para una cavidad circular 2D de radio 15σ, en la cuales difunden partículas tipo LJ a diferente temperatura y densidad: (línea punteada) $\rho = 0.844$ y $T = 0.71$, (línea a trazos) $\rho = 0.6$ y $T = 1.5$, (línea continua) $\rho = 0.2$ y $T = 1.5$.

se puede implementar un ciclo de relajación térmica y posteriormente el ciclo de producción; ambos ciclos son muy idénticos a los utilizados en el ejercicio anterior.

La Fig. 5b) muestra la configuración al final del ciclo de producción para sistemas con diferente densidad solo con modificar el tipo de cavidad. Por ejemplo, se podría comparar la estructura para las mismas condiciones de densidad y temperatura, bajo diferentes tipos de confinamiento: cuadrado, rectangular, triangular, cavidades del tipo sierra o sinusoidal. Para confinamiento en 3D, ESPResSo tiene constricciones predefinidas del tipo paralelepípedo, cilíndrico y esférico, solo por mencionar algunas. Finalmente, la Fig. 6 muestra el cálculo de MSD para diferentes densidad, que es congruente con los estudios previos [19].

Conclusiones

Actualmente, la DM es una herramienta consolidada dentro de diversas áreas científicas. El gran número de publicaciones científicas donde se utiliza la DM brinda una clara idea de la popularidad y madurez alcanzada por esta metodología a tan solo tres décadas de su aparición.

Recientemente, diversos autores han reportado el impacto positivo que se tiene en la enseñanza al emplear simulaciones de DM a nivel universitario. Desde el punto de vista didáctico, permite introducir a estudiantes de diferentes carreras en aspectos que van desde cómo montar una simulación, cómo calcular y analizar observables físicas, hasta la visualización de la evolución del sistema. Sin embargo, aunque los docentes puedan obtener fácilmente un software de dinámica molecular o incluso tener sus propio códigos, el aprendizaje de la metodología de DM podría resultar lento y difícil. Una de las problemáticas detectadas por aquellos autores que promueven el uso de simulaciones de DM es que los estudiantes no conocen las herramientas imprescindibles para realizar este tipo de simulaciones. En otras opalabras, muchos estudiantes no están familiarizados con algún lenguaje de programación, entorno linux, graficadores, software de visualización y mucho menos con la metodología de DM. en tornos académicos, esto conlleva invertir una gran cantidad de tiempo para proporcionar las nociones elementales para que los estudiantes logren construir una DM de un sistema básico.

En esta dirección, ESPResSo resulta una herramienta que puede utilizarse como un primer acercamiento a la simulación computacional con un doble propósito. Por un lado, para esconder los detalles de la implementación numérica de los algoritmos inherentes a una DM, y por otro lado, como una plataforma para montar de manera sencilla y rápida sistemas típicos, así como para calcular y analizar las propiedades de dichos sistemas. Las instrucciones mostradas en este trabajo otorgan las bases mínimas para generar y analizar sistemas relativamente simples, para posteriormente adaptarlos a sistemas más complejos.

i. `http://espressomd.org/wordpress`

1. H.J. Limbach, A. Arnold, B.A. Mann and C. Holm, *Computer Physics Communications* **174** (2006) 704-727. `https://doi.org/10.1016/j.cpc.2005.10.005`
2. A. Arnold *et al*. (2013) *E*SPResSo 3.1: Molecular Dynamics Software for Coarse-Grained Models. In: Griebel M., Schweitzer M. (eds) *M*eshfree Methods for Partial Differential Equations VI. Lecture Notes in Computational Science and Engineering, **89**. Springer, Berlin, Heidelberg.
3. P. G. De Gennes , *Soft Matter*, **64**, (1992), 645. `https://doi.org/10.1103/RevModPhys.64.645`
4. M. Doi , *Soft Matter Physics*, Oxford University Press (2013).
5. H.I. Ingólfsson *et al., Wiley Interdisciplinary Reviews: Computational Molecular Science*, **4**, (2014) 225-248. `https://doi.org/10.1002/wcms.1169`
6. E. Brini, E.A. Algaer, P. Ganguly, Li C., F. Rodríguez-Ropero and N.F.A. van der Vegt, *Soft Matter*, **9**, (2013) 2108-2119. `https://doi.org/10.1039/C2SM27201F`
7. J. Wong-Ekkabut and M. Karttunen, *Biochimica et Biophysica Acta (BBA)-Biomembranes*, **1858**, (2016) 2529-2538. `https://doi.org/10.1016/j.bbamem.2016.02.004`
8. B.J. Alder and T.E. Wainwright, *The Journal of Chemical Physics*, **27**, (1957) 1208-1209. `https://doi.org/10.1063/1.1743957`
9. D. Frenkel and B. Smit, *Ünderstanding molecular simulation: from algorithms to applications"*, **Vol. 1**. Elsevier (2001).

10. J.G. Gay and B.J. Berne,*The Journal of Chemical Physics*, **74**, (1981) 3316. https://doi.org/10.1063/1.441483

11. W. Smith and D. Fincham, *Informatio quartely for computer simulation of condensed phases*, (1988).

12. W. Liu, B. Schmidt, G. Voss and W. Müller-Wittig, *International Conference on High-Performance Computing, "Molecular dynamics simulations on commodity GPUs with CUDA"*, Springer (2007), 185-196. https://doi.org/10.1007/978-3-540-77220-0_20

13. W. Humphrey, A. Dalke and K. Schulten, *Journal of Molecular Graphics*, **14**, (1996) 33-38. https://doi.org/10.1016/0263-7855(96)00018-5

14. M . P. Allen and D. Tildesley, *"Computer simulation of liquids"*. Clarendon Press, Oxford (1987).

15. D. Boda and D. Henderson, *Molecular Physics*, **106**, (2008) 2367-2370. https://doi.org/10.1080/00268970802471137

16. A. Stukowski, *Ovito Open Visualization Tool* (2015). https://doi.org/10.1088/0965-0393/18/1/015012

17. T. Plachetka, *Proc. of Spring Conf. on Computer Graphics, Budmerice, Slovakia, "POV Ray: persistence of vision parallel raytracer,"* vol. 123 (1998).

18. P. Pearlea and B. Collett, *American Journal of Physics*, **78**, (2010) 1278. https://doi.org/10.1119/1.3475685

19. A. González, E. Díaz-Herrera, M. Sandoval, M. A. Chávez-Rojo and J. A. Moreno-Razo, *Selected Topics of Computational and Experimental Fluid Mechanics, "Confinement and Interaction Effects on the Diffusion of Passive Particles,"* Springer Cham. (2015) 385-394.