Reporte de actividades - Métodos de Montecarlo

Carlos Manuel Rodríguez y Martínez
José Ramón Palacios Barreda
10 de junio de 2012

Índice

1. Resumen

Este documento resume las actividades realizadas durante la 2da parte del curso 'Métodos de Montecarlo', impartida por el Dr. Adrián Huerta como materia optativa (Feb-Jun 2012). El proyecto en el que se trabajó consistió en simular un fluido de Van der Waals mediante métodos de Montecarlo.

2. Fluido de Van der Waals

El fluido de Van der Waals es un modelo empírico nacido dentro de la Termodinámica, a mediados del siglo XIX. El físicos holandés J. Hendrik Van der Vaals propuso una modificación a la ecuación del gas ideal con la intención de mejorar la descripción de un fluido. Esta ecuación contiene 2 constantes ajustables asociadas a las fuerzas de atracción intermoleculares y el volumen que ocupan las partículas dentro del fluido en cuestión. La ecuación de Van der Waals:

$$P = \frac{RT}{v-b} - \frac{a}{v^2}$$

La característica más importante de esta ecuación (a parte de marcar el inicio del estudio teórico de los gases imperfectos en la Termodinámica) es la predicción de una zona de transición. La siguiente gráfica ilustra dicha región, que es una de las características que buscamos reproducir mediante métodos de Montecarlo:

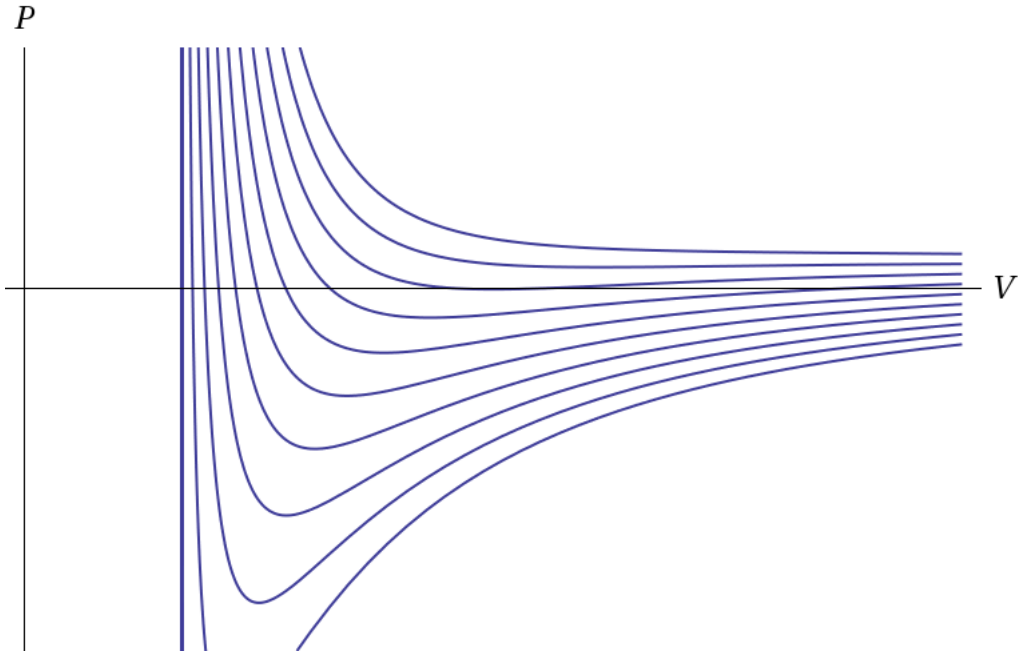


Figura 1: Fluido de Van der Waals

Cada curva se grafica a temperatura constante (isoterma), y como se ve en la figura, conforme la temperatura aumenta, la curva tiende la ecuación de gas ideal. Debido a que la ecuación de VdW posee 2 constantes que dependen del fluido analizado, nosotros comparamos nuestros resultados contra la ecuación reducia de VdW; escrita en forma adimensional y que no depende de dichas constantes:

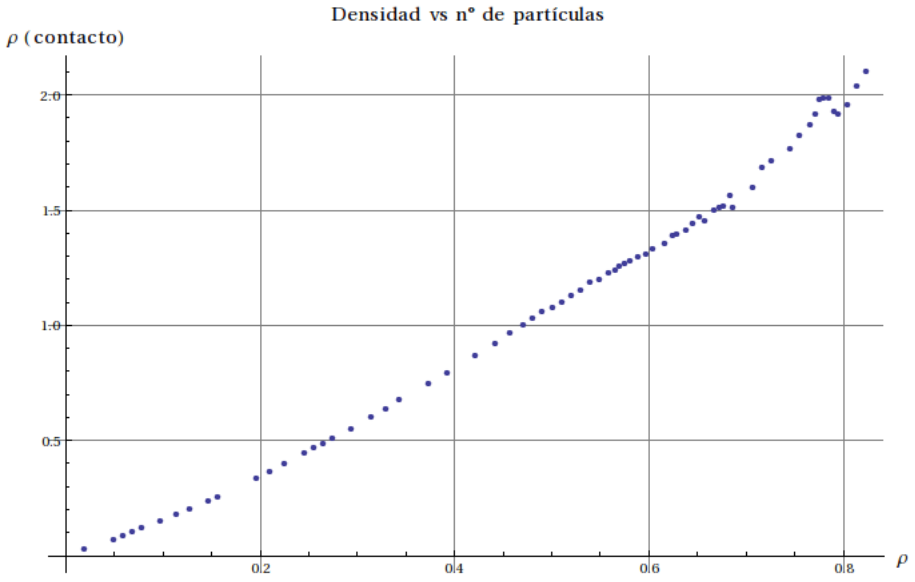
$$\pi = \frac{8\theta}{3\phi - 1} - \frac{3}{\phi^2}$$

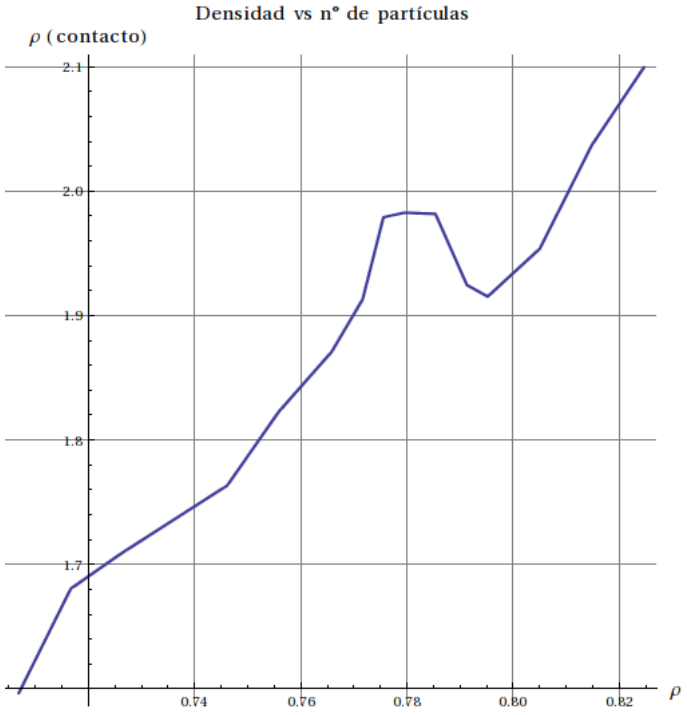
Donde π, θ, ϕ son la presión, temperatura y volumen reducidos, respectivamente.

3. Simulación

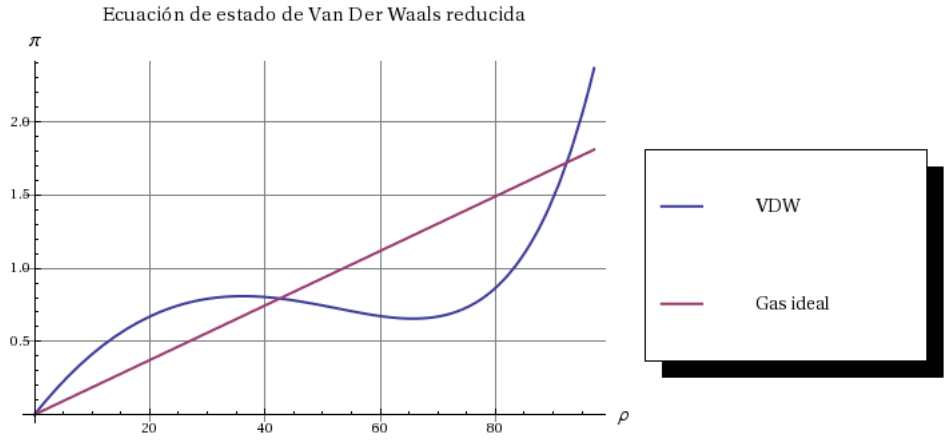
3.1. Resultados

En esta sección se presentan los resultados obtenidos usando métodos de Montecarlo. Se corrieron 70 simulaciones con un número fijo de iteraciones (8×10^4) y un número creciente de partículas. El número de partículas por corrida se varió de acuerdo a lo observado en las gráficas; es decir, tenemos una mayor densidad de puntos alrededor de la zona de transición de fase. Graficando la densidad ρ contra el valor de contacto (discutido en la sección de Post-procesamiento de datos) obtuvimos lo siguiente:





El *loop* de Van der Waals, o la zona de transición es claramente visible en el intervalo de densidad $[0.77, 0.81]$, que corresponde a un número de partículas de aproximadamente $N \approx 400$. Comparando, cualitativamente, con la ecuación reducida de Van der Waals, podemos notar la misma zona de transición; que como ya se mencionó, es una de las características relevantes de dicha ecuación de estado.

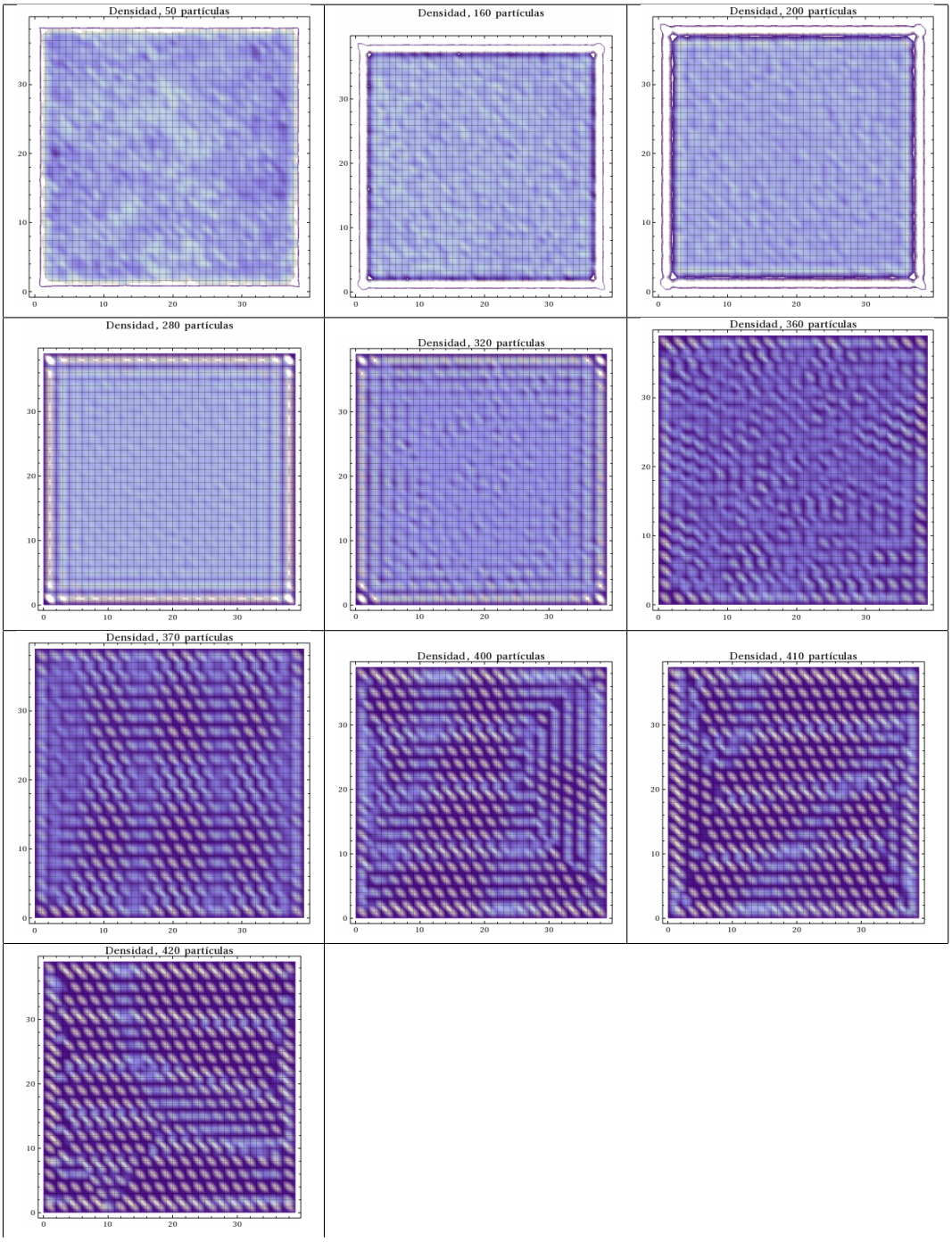


3.1.1. Cambio de fase

Durante el aumento progresivo del número de partículas que se le agregan al sistema se graficó la distribución de densidades. Esta distribución de densidades permite observar las estructuras que se forman en la zona interior del sistema de partículas. De aquí se aprecia que existe una correlación entre la zona de transición de fase que ocurre en el *loop* de Van der Waals y la estructura que se forma en el interior del sistema.

3.1.2. Patrones en la cristralización

Al saturar el sistema con un mayor número de partículas o disminuir el volumen de este las partículas acabarán adquiriendo una configuración estable de manera que al iterar el sistema este tendrá una configuración de densidades irregular, teniendo mayor densidad en las zonas donde exista una estructura cristalina. En las gráficas que se muestran a continuación se observa el cambio de estructura del sistema conforme se aproxima al punto de transición de fase que ocurre al agregarle $N \approx 400$ partículas al sistema.



4. Descripción del software

El script utlizado fue programado en Python, utilizando la librería VPython para el complemento visual de la simulación. Consta de dos módulos: *discos_duros.py* (simulador) & *shaker.py* (generar configuraciones). A continuación se describre la estructura general del programa, así co-

mo sus diferentes opciones.Las unidades utlizadas en el programa están en función del diámetro (*sigma*) de la partícula. El tamaño de la caja, el desplazamiento mínimo y áreas donde se calcula la densidad están todas en función de esta cantidad.

4.1. Shaker

El propósito de este script es generar una configuración inicial de discos; suficientemente desordenada para luego importarla desde *discos_duros.py*. Por defecto, la configuración se guarda al archivo *particles.csv*, en el mismo directorio donde se corre el script. Este archivo puede ser utilizado por el script principal (*discos_duros.py*) para simular sin alterar los resultados mientras desordena el sistema. Esta fue una consideración obviada al inicio del trabajo, pero que resultó importante para obtener resultados favorables. Ejemplo:

```
$ python shaker.py --particles=420 --makecrystal=triag --iterations=5000 --dsmax=0.08

Parámetros y opciones
-------------------------------------
* Partículas: 420
* Modo visual: True
* Iteraciones: 5000
* Sigma: 1
* L: 20*sigma


Se agregaron 420 particulas

Progreso: 48% Aceptacion: 0.0928571428571
```

Aquí se genera una configuración de alta densidad (420 partículas) con una configuración triangular. Es recomendable generar configuraciones cuyo factor de aceptación supere el 25 %. Este factor se refiere a la cantidad de movimientos aceptados por iteración. Con un valor de aceptación más o menos constante, aseguramos que el simulador generará datos confiables y no sesgados (por un desorden parcial).

4.1.1. Opciones

Es posible visualizar las opciones al llamar al script sin argumentos. El siguiente mensaje describe las opciones disponibles (cuya descripción ampliamos más adelante):

```
$ python shaker.py

Ejemplo de uso: python shaker.py [opciones]
--particles=100     | Necesaria           | Partículas para simulación.
--iterations=10000  | Necesaria           | Veces que se mueven las partículas.
--visual=False      | Opcional=True       | Modo visual (VPython).
--makecrystal=False | Opcional=cuad,triag | Crea arreglo cristalino.
--boxlength=20      | Opcional=20         | Tamaño de la caja.
--dsmax=1           | Opcional=0.5        | Movimiento máximo de partícula.
--import=NULL       | Opcional=coord.csv  | Importa archivo de coordenadas.
```

Las opciones necesarias para correr el script se marcan con *. El resto son opcionales y cuentan con valores predeterminados.

- particles*: especifica la cantidad de partículas a agregar para generar la configuración. Dada una configuración (tipo de arreglo y tamaño de caja), el script siempre intentará agregar cuantas partículas sea posible. En un arreglo cristalino (cuadrado), la cantidad máxima de partículas será igual a L^2, donde L es el lado de la caja (*boxsize*).
- iterations*: especifica la cantidad de iteraciones para la corrida.

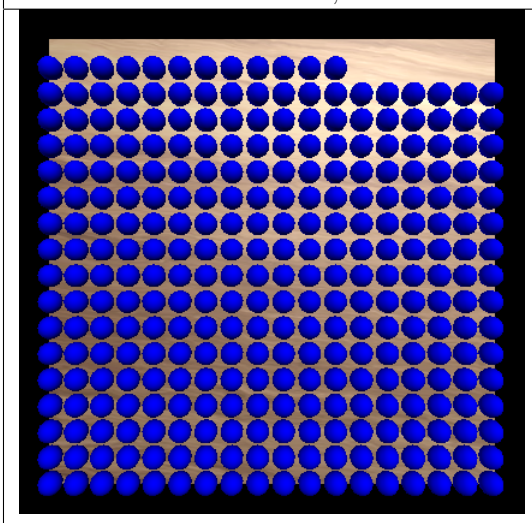
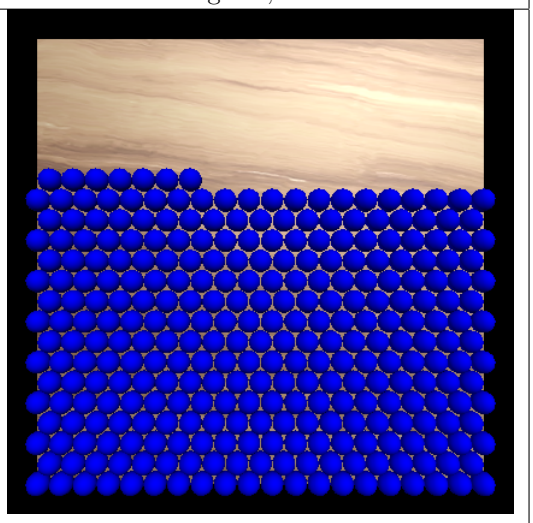
- visual: activa/desactiva el modo visual. El script corre de igual manera sin este extra.
- makecrystal

  cuad: genera un arreglo cristalino (cuadrado).

  triag: genera un arreglo triangular (permite una cantidad mayor de partículas).
- boxlength: tamaño de la caja en múltiplos de sigma.
- dsmax: tamaño del paso para cada movimiento, en múltiplos de sigma. Si el factor de aceptación es bajo, se puede reducir este número para mejorar el valor. Es importante mantener la consistencia entre corridas respecto a este valor. Es decir, un set de datos deberá ser generado con un mismo dsmax.
- import: nombre de archivo a importar para continuar configruación. En caso de requerir un tiempo excesivo de cómputo, la configuración puede ser guardada/cargada en cualquier momento.

4.1.2. Configuraciones

Las configuraciones iniciales no tienen relevancia en cuanto a los datos que el simulador generará una vez desordenado el sistema. La razón: el sistema no posee memoria. Sin embargo, una selección apropiada de arreglo para las partículas permitirá insertar una mayor cantidad de partículas o mejorar el tiempo que debe correr el script para desordenar el sistema hasta un valor de aceptación cercano al 25 %. No especificar esta opción resulta en una configuración aleatoria (útil para un número bajo de partículas). Los siguientes ejemplos ilustran la diferencia de espacio libre para la misma cantidad de partículas en las 2 configuraciones disponibles.

Cristalino cuadrado, n = 300	Triangular, n = 300
	

4.1.3. Procedimiento: Agregar partículas al sistema

En general, ambas configuraciones realizan lo siguiente antes de agregar una partículas:

- Calcular un factor para optimizar el espacio libre.
- Calcular nuevas posiciones para cada partículas

- Verificar que la partícula esté dentro las fronteras de la caja
- Verificar que la partícula no traslape con alguna dentro de la configuración

En el caso de arreglo cristalino, el factor de espaciamiento está dado por:

$$a = (L - k * \sigma)/k$$

Donde k es la cantidad máxima de partículas permisible por fila/columna. Para el caso triangular, el factor de espaciamiento está dado por:

$$a = \frac{\sqrt{3}}{2}$$

Asociado a la geometría de la configuración. En ambos casos se recorre la caja horizontalmente en incrementos de una partícula (utilizando el factor calculado en el caso cuadrado y σ en el caso triangular). El recorrido vertical se hace en múltiplos del factor.

4.1.4. Procedimiento: Mover partículas

El movimiento de una partícula es simple: aleatorio. Se genera un par ordenado tomado de una distribución uniforme de números pseudo-aleatorios en el rango $(-ds, ds)$. Donde, como antes, ds representa el paso en múltiplos de σ.

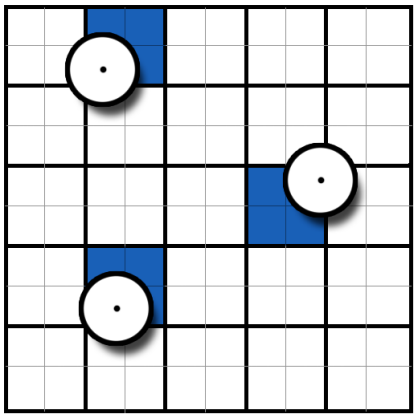
4.2. Discos Duros - Simulador

4.2.1. Opciones

Todas las opciones son idénticas a *shaker.py*, así, aplican los mismos comentarios que para dicho script. Como antes, se recomienda preparar la configuración en *shaker.py* con un factor de aceptación no menor a 25 % para luego correr la simulación en *discos_duros.py*.

4.2.2. Procedimiento: Cálculo de densidad

Para calcular la densidad (numérica) del fluido, dividimos la caja en una retícula cuadrada a la que asignamos un contador por casilla. El lado de cada casilla se especifica en múltiplos de σ. Tras cada iteración se calcula la posición de cada partícula y se ajusta a la retícula con un criterio ilustrado por el siguiente diagrama:



Para el caso donde el lado de la casilla sea menor al radio de cada partícula, omitimos los bordes en este cálculo. De otro modo, tendríamos una franja con densidad nula.

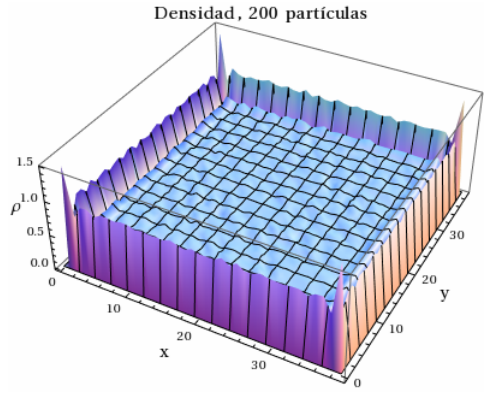
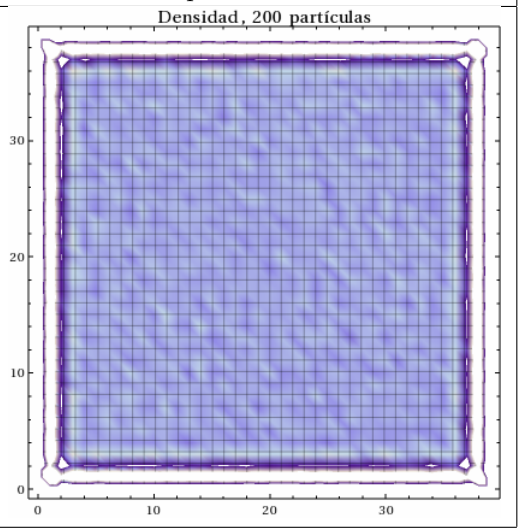
4.2.3. Datos

A lo largo de la simulación, el script genera una serie de archivos para documentar la evolución del sistema. Cada que se completa el 10 % de la iteraciones solicitadas, el programa genera 2 archivos con el conteo en curso: la retícula con el conteo de aceptación y el mismo archivo dividido entre en el número de partículas del sistema (densidad numérica). Este se puede modificar con la opción *dumpall*, que controla si deseamos que se genere la documentación completa o solo el resultado final. Los archivos generados:

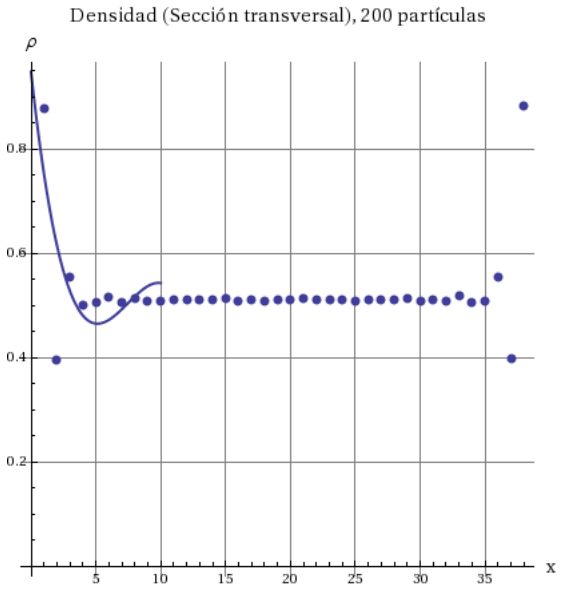
- *dump.csv*: conteo de partículas por casilla.
- *dump_densidad.csv*: conteo de partículas dividido entre el número total de partículas (densidad numérica).
- *flatGrid.csv*: promedio de la retícula en una dimensión. Útil para estimar el llamado *valor de contacto*, discutido en la siguiente sección.

4.3. Post-procesamiento de datos

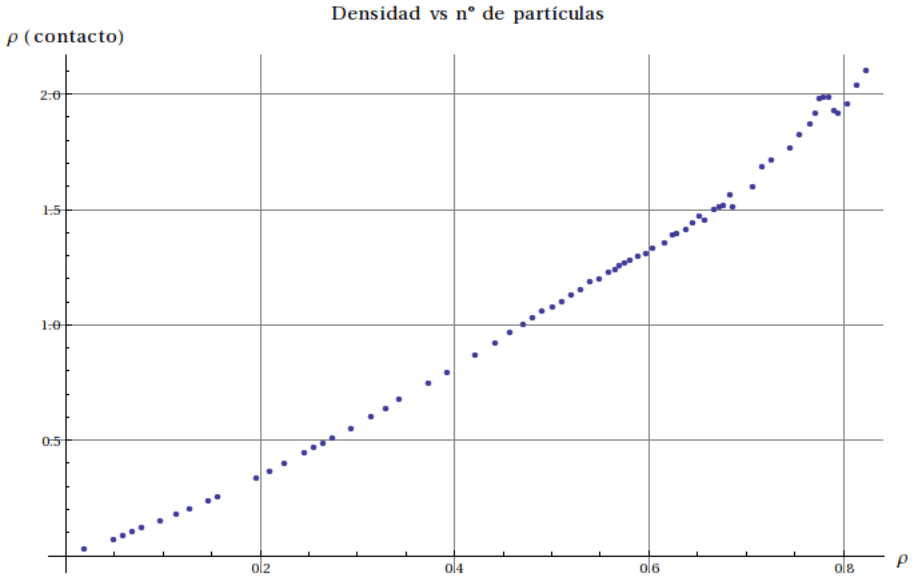
Las gráficas fueron generadas con los datos extraídos del script *discos_duros.py* y *Mathematica*. Utilizando los archivos con la densidad numérica y el conteo de partículas por casillas se generaron gráficas como las siguientes, que ilustran la distribución de las partículas a lo largo de la simulación:

Distribución de partículas	Mapa de densidad
	

El tercer archivo generado por la simulación nos fue útil para calcular el valor de contacto, el cual nos proporciona una estimación de la presión del fluido simulado mediante una regresión cúbica sobre la colección de puntos obtenida. La siguiente gráfica ilustra dicho proceso en el cual nos interesa la intersección de la gráfica de regresión con el eje vertical (valor de contacto):



Los valores de contacto $\rho(N)$ se graficaron para obtener la presión del fluido en función de la cantidad de partículas presentes. La siguiente gráfica muestra un ejemplo de dicho proceso:



5. Referencias

[1] Krauth, W. *Statistical Mechanics: Algorithms and Computations*. Oxford University Press, New York, 2006. cap. 1.