

Proyecto Final

Desarrollo Experimental II

Docente:

Dra. Laura Lorenia Yeomans Reyna

Simulación de Monte Carlo:

Exploración de la Ecuación de Van Der Waals

Alumno:

Martín Alejandro Paredes Sosa

Semestre: 2018-1

1. Introducción

La estructura de un líquido está fuertemente determinada por las interacciones repulsivas de corto alcance, y los estudios con simulaciones computacionales se han utilizado para mostrar que estas interacciones repulsivas de corto alcance pueden modelarse como de núcleo duro. Para modelar un líquido, sin embargo, se requiere una componente atractiva como parte del potencial de interacción entre las partículas del sistema.

Una de las ecuaciones de estado más importantes y que históricamente se han planteado para describir a los líquidos y la coexistencia líquido-vapor ha sido la ecuación de estado de Van der Waals [1]:

$$\left(p + \frac{N^2}{V^2}a\right)(V - Nb) = Nk_B T \quad (1)$$

Esta se puede derivarse como una aplicación de la teoría de perturbaciones de Zwanzig[2], que permite obtener las expresiones para los coeficientes a y b que coinciden con la propuesta de Van Der Waals [2]

$$a = -2\pi \int_{\sigma}^{\infty} u^{(1)}(r) r^2 dr \quad (2)$$

$$b = \frac{2\pi\sigma^3}{3} \quad (3)$$

donde σ es el diámetro de la esfera dura y $u^{(1)}(r)$ el potencial de interacción atractivo que se considera como la parte perturbativa del potencial de interacción entre las partículas del sistema.

$$u(r) = u^{(hs)}(r) + u^{(1)}(r) \quad (4)$$

2. Contexto

Para la obtención de (2), se asume que:

$$g_{hs} \approx \begin{cases} 0 & r \leq \sigma \\ 1 & r > \sigma, \end{cases} \quad (5)$$

Esta aproximación es para muy bajas concentraciones, donde la función de distribución radial de contacto $g_{hs}(\sigma^+) \approx 1$, que corresponde al modelo de Van Der Waals.

En el contexto del ensemble canónico en la Física Estadística, la ecuación de estado se obtiene a partir de:

$$p = \frac{1}{\beta} \left(\frac{\partial Z_N}{\partial V} \right)_T \quad (6)$$

Como $g_{hs}(r)$ depende de la concentración, la ecuación de estado es de la forma:

$$p = p_{hs} - \rho^2 \left[a(\rho) + r h o \left(\frac{\partial a(\rho)}{\partial \rho} \right)_T \right] \quad (7)$$

$$p_{hs} = \rho k_B T \left[1 + \frac{2}{3} \pi \sigma^3 \rho g_{hs}(\sigma^+) \right] \quad (8)$$

3. Metodología

- I. **Potencial Perturbativo:** La teoría de Zwanzig no impone ninguna restricción sobre el potencial perturbativo $u^{(1)}(r)$ con tal de que sea atractivo. Entonces, consideremos como potencial perturbativo la parte atractiva del modelo de pozo cuadrado:

$$u^{(1)}(r) = \begin{cases} -\varepsilon & \sigma < r < \lambda\sigma \\ 0 & r \geq \lambda\sigma \end{cases} \quad (9)$$

De modo que el potencial de interacción según la ecuación (4) queda:

$$u(r) = \begin{cases} \infty & r \leq \sigma \\ -\varepsilon & \sigma < r < \lambda\sigma \\ 0 & r \geq \lambda\sigma \end{cases} \quad (10)$$

- II. **Reducción de variables** Antes de empezar a realizar cálculos, se redujeron las ecuaciones, tomando como longitud característica el diámetro σ y como energía característica la energía térmica β^{-1}

Por lo tanto obtenemos las siguientes definiciones:

$$r^* \equiv \frac{r}{\sigma} \quad (11)$$

$$u^*(r) \equiv \frac{u(r)}{k_B T} = \beta u(r) \quad (12)$$

$$p^* \equiv \frac{p\sigma^3}{k_B T} = \beta p\sigma^3 \quad (13)$$

$$T^* \equiv \frac{k_B T}{\varepsilon} = \frac{1}{\beta\varepsilon} = \frac{1}{\varepsilon^*} \quad (14)$$

$$n^* \equiv \sigma^3 \rho \quad (15)$$

De tal forma que:

$$u^*(r) = \begin{cases} \infty & r^* \leq 1 \\ -\frac{1}{T^*} & 1 < r^* < \lambda \\ 0 & r^* \geq \lambda \end{cases} \quad (16)$$

Utilizando (11)-(15) en (8) obtenemos:

$$\begin{aligned} \frac{p_{hs}^*}{\beta\sigma^3} &= \frac{n^*}{\beta\sigma^3} \left[1 + \frac{2}{3}\pi\sigma^3 \frac{n^*}{\sigma^3} g_{hs}(1^+) \right] \\ p_{hs}^* &= n^* \left[1 + \frac{2}{3}\pi n^* g_{hs}(1^+) \right] \\ p_{hs}^* &= n^* [1 + n^* b^*] \end{aligned} \quad (17)$$

Donde:

$$b^* = \frac{2}{3}\pi g_{hs}(1^+) \quad (18)$$

Ahora en (7) obtenemos:

$$\begin{aligned} \frac{p^*}{\beta\sigma^3} &= \frac{p_{hs}^*}{\beta\sigma^3} - \left(\frac{n^*}{\sigma^3}\right)^2 \left[a(n^*) + \frac{n^*}{\sigma^3} \left(\frac{\partial a(n^*)}{\partial \frac{n^*}{\sigma^3}} \right)_{T^*} \right] \\ p^* &= p_{hs}^* - \frac{\beta n^{*2}}{\sigma^3} \left[a(n^*) + n^* \left(\frac{\partial a(n^*)}{\partial n^*} \right)_{T^*} \right] \\ p^* &= p_{hs}^* - n^{*2} \left[\frac{\beta}{\sigma^3} a(n^*) + n^* \left(\frac{\partial \frac{\beta}{\sigma^3} a(n^*)}{\partial n^*} \right)_{T^*} \right] \\ p^* &= p_{hs}^* - \left[a^* + n^* \left(\frac{\partial a^*}{\partial n^*} \right)_{T^*} \right] n^{*2} \end{aligned} \quad (19)$$

Donde:

$$a^* = \frac{\beta}{\sigma^3} a(n^*) \quad (20)$$

III. **Parámetro Críticos** La Ecuación de Van Der Waals (1) tiene una temperatura crítica T_c . Cuando $T > T_c$ la ecuación de estado es monovaluada y no hay transiciones al estado liquido. Cuando $T < T_c$, como la ecuación es cubica en V , tiene dos extremales que se juntan en $T = T_c$.

Pensando en la ecuación de Van Der Waals como una función del volumen a una temperatura dada $P = P(V; T)$, se tienen dos punto críticos donde se cumple $(\partial_V P)_T = 0$. Conforme $T \rightarrow T_c$ se obtiene un punto de inflexión con coordenadas (P_c, V_c, T_c) donde $(\partial_V^2 P)_T = 0$

Partiendo de la ecuación (1):

$$P = \frac{Nk_B T}{V - Nb} - \frac{N^2 a}{V^2} \quad (21)$$

$$\left(\frac{\partial P}{\partial V} \right)_T = -\frac{Nk_B T}{(V - Nb)^2} + \frac{2N^2 a}{V^3} \quad (22)$$

$$\left(\frac{\partial^2 P}{\partial V^2} \right)_T = \frac{2Nk_B T}{(V - Nb)^3} - \frac{6N^2 a}{V^4} \quad (23)$$

Aplicando la condición de punto crítico obtenemos el siguiente sistema de ecuaciones para los coeficientes a y b

$$\begin{aligned} -\frac{Nk_B T_c}{(V_c - Nb)^2} + \frac{2N^2 a}{V_c^3} &= 0 \\ \frac{2Nk_B T_c}{(V_c - Nb)^3} - \frac{6N^2 a}{V_c^4} &= 0 \end{aligned}$$

Despejamos a de la primera ecuación

$$\begin{aligned}\frac{2N^2a}{V_c^3} &= \frac{Nk_B T_c}{(V_c - Nb)^2} \\ a &= \frac{Nk_B T_c}{(V_c - Nb)^2} \frac{V_c^3}{2N^2}\end{aligned}\quad (24)$$

Ahora en la segunda ecuación hacemos lo mismo

$$\begin{aligned}\frac{2Nk_B T_c}{(V_c - Nb)^3} &= \frac{6N^2a}{V_c^4} \\ a &= \frac{2Nk_B T_c}{(V_c - Nb)^3} \frac{V_c^4}{6N^2}\end{aligned}$$

Igualando obtenemos

$$\begin{aligned}\frac{Nk_B T_c}{(V_c - Nb)^2} \frac{V_c^3}{2N^2} &= \frac{2Nk_B T_c}{(V_c - Nb)^3} \frac{V_c^4}{6N^2} \\ \frac{V_c^3}{2} &= \frac{2}{V_c - Nb} \frac{V_c^4}{6} \\ V_c - Nb &= \frac{V_c}{6} \\ b &= \frac{V_c}{3N}\end{aligned}\quad (25)$$

Sustituimos esto en (24)

$$\begin{aligned}a &= \frac{Nk_B T_c}{(V_c - N\frac{V_c}{3N})^2} \frac{V_c^3}{2N^2} \\ a &= \frac{k_B T_c}{(\frac{2V_c}{3})^2} \frac{V_c^3}{2N} \\ a &= \frac{9k_B T_c}{4V_c^2} \frac{V_c^3}{2N^2} \\ a &= \frac{9}{8} k_B T_c \frac{V_c}{N}\end{aligned}\quad (26)$$

A partir de las ecuaciones (25) y (26) obtenemos que

$$V_c = 3Nb \quad (27)$$

$$T_c = \frac{8a}{27k_B b} \quad (28)$$

$$P_c = \frac{a}{27b^2} \quad (29)$$

Ahora obtendremos las expresiones para las variables termodinámicas críticas de Van Der Waals para el caso de estudio con un potencial perturbativo de pozo cuadrado (9). Para

esto se utilizamos la ecuación (2) (pero a la Zwanzig) y el potencial reducido.

$$\begin{aligned}
 a^* &= -2\pi \int_1^\infty u^{*(1)}(r^*) g_{hs}(r^*) r^{*2} dr^* \\
 &= -2\pi \left[\int_1^\lambda u^{*(1)}(r^*) g_{hs}(r^*) r^{*2} dr^* + \int_\lambda^\infty u^{*(1)}(r^*) g_{hs}(r^*) r^{*2} dr^* \right] \\
 &= -2\pi \int_1^\lambda \frac{-1}{T^*} g_{hs}(r^*) r^{*2} dr^* \\
 a^* &= \frac{2\pi}{T^*} \int_1^\lambda g_{hs}(r^*) r^{*2} dr^*
 \end{aligned} \tag{30}$$

Para el caso Van Der Waals la integral queda:

$$a^* = \frac{2\pi}{3} \frac{\lambda^3 - 1}{T^*} \tag{31}$$

Para el coeficiente b^* se tiene la ecuación (18). Utilizando la suposición de Van Der Waals, entonces obtenemos (3), que en es forma reducida resulta:

$$b^* = \frac{2\pi}{3} \tag{32}$$

Por lo tanto, de (27),(28) y (29) se obtiene

$$V_c^* = 2\pi N \tag{33}$$

$$T_c^* = \frac{8}{27}(\lambda^3 - 1) \tag{34}$$

$$P_c^* = \frac{\lambda^3 - 1}{18\pi T^*} \tag{35}$$

- **Ejemplo:** Veamos de qué orden son los valores para las variables criticas para un sistema dado. Sea $\lambda = 1.25$, resulta:

$$V_c^* = 2\pi N$$

$$T_c^* = 0.2824$$

$$P_c^* = \frac{0.0169}{T^*}$$

IV. Función de Distribución Radial de HS Ahora utilizamos nuestro código de Monte Carlo, con el cual se realizaron corridas para diferentes concentraciones. A continuación se muestra un gráfico de $g(r) - n^*$ para el modelo de potencial HS solamente.

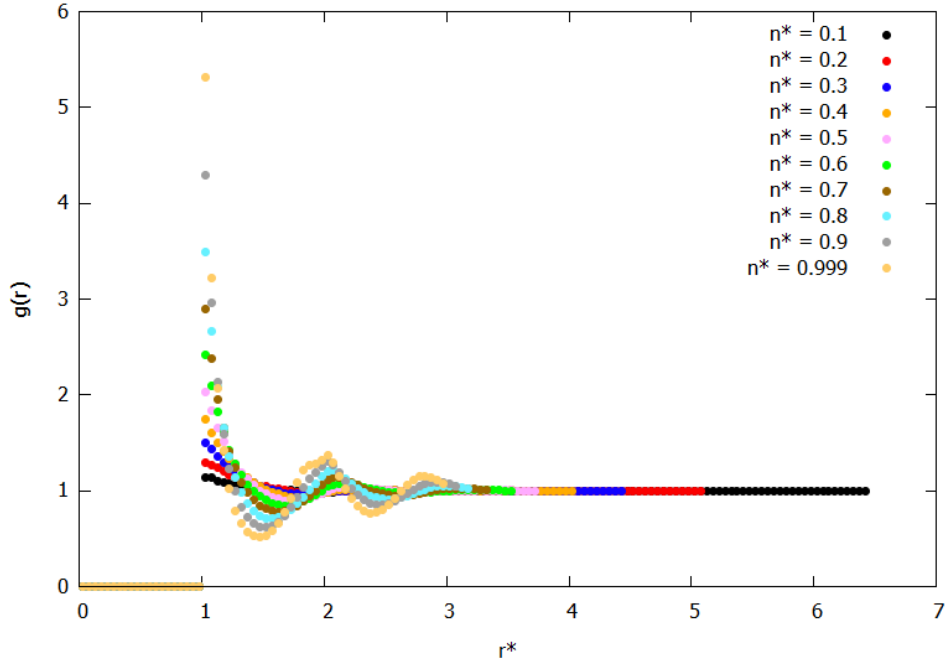


Figura 1: Función $g(r)$ para potencial de Esfera Dura

Se pueden apreciar que en el punto de contacto ($r^* = 1$) hay una mayor cantidad de partículas, y esta aumenta junto con la concentración. Los picos muestran la presencia de “vecinos” segundos, terceros, etc.

Estos datos se obtuvieron de la Simulación Monte Carlo para un sistema de 216 partículas, se realizaron 3×10^5 configuraciones y se consideró la configuración 1×10^5 como la de equilibrio, para poder olvidar la configuración inicial.

V Ecuación de Presión HS

En la termodinámica, suele usar la ecuación de estado de presión contra volumen. Dicho esto, nuestro análisis es $P^* - n^*$ dado la relación que hay entre la concentración y el volumen. Para el estudio de Esferas duras, ya conocemos la ecuación (17), junto con (18). Como n^* es un parámetro conocido de la simulación, basta con conocer el valor del punto de contacto $g_{hs}(1^+)$

La ecuación de estado de Carnahan-Starling para un sistema de esferas duras está dada por:

$$P^*(n^*) = n^* \frac{1 + \phi + \phi^2 - \phi^3}{(1 - \phi)^3}, \quad \phi = \frac{\pi}{6} n^* \quad (36)$$

En la figura 2, se muestra una comparación entre los datos simulados y las ecuaciones de estado de Gas Ideal y la de Carnahan-Starling.

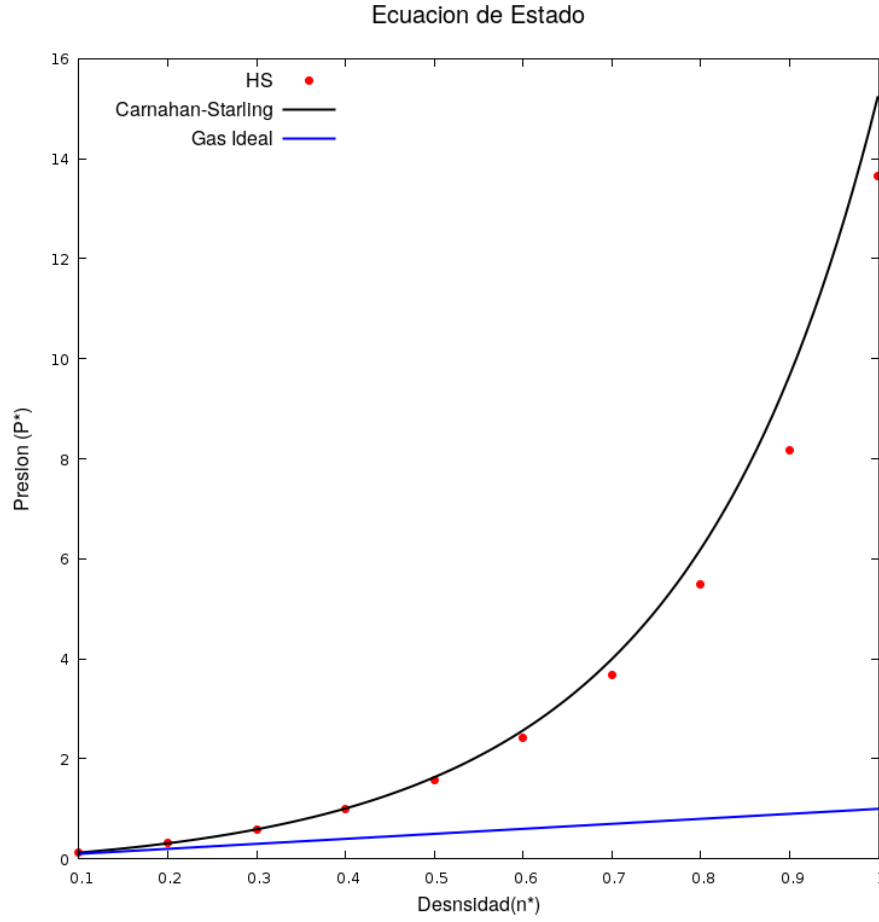


Figura 2: Ecuación de Estado de un sistema de esferas duras comparando con la de Gas Ideal y Carnahan Starling

VI. **Calculo de los coeficiente $a^*(n^*)$ y $b^*(n^*)$:** Con la información estructural que calculamos de las simulaciones (las $g(r^*)$) y p de las ecuaciones (18) y (30), se procedió a calcular los coeficientes.

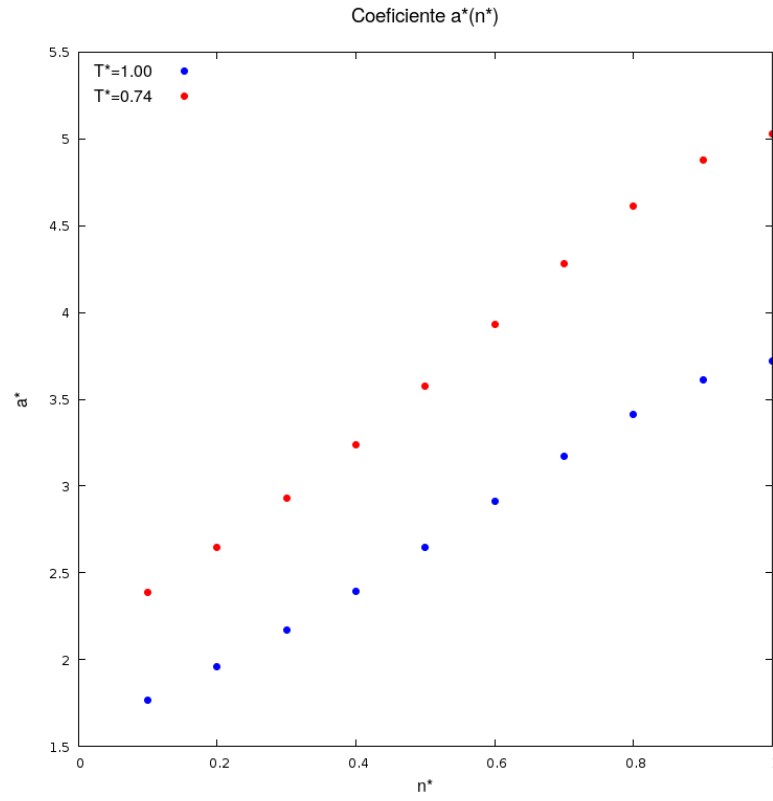


Figura 3: Coeficiente a^* a $T = 1.0$ y $T = 0.74$

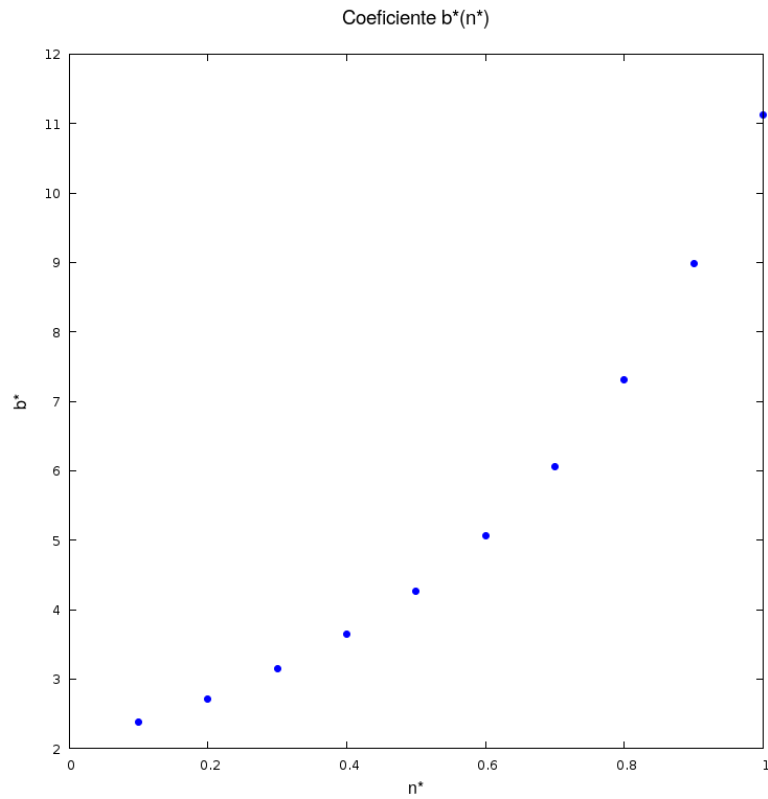


Figura 4: Coeficiente b^*

VII. **Ajuste de a^*** De los resultados que obtuvimos de a^* , podemos observar una ligera curvatura. Mediante el uso del programa Origin, procedimos a ajustar una curva, que en este caso resulto un polinomio de segundo orden. Para el caso de $T^* = 1.0$ se ajusto lo siguiente:

$$a^*(n^*) = 1.47 + 2.48n^* - 0.16n^{*2}$$

$$\partial_{n^*}a^* = 2.48 - 0.32n^*$$

Y para el caso de $T^* = 0.74$

$$a^*(n^*) = 1.99 + 3.35n^* - 0.21n^{*2}$$

$$\partial_{n^*}a^* = 3.35 - 0.42n^*$$

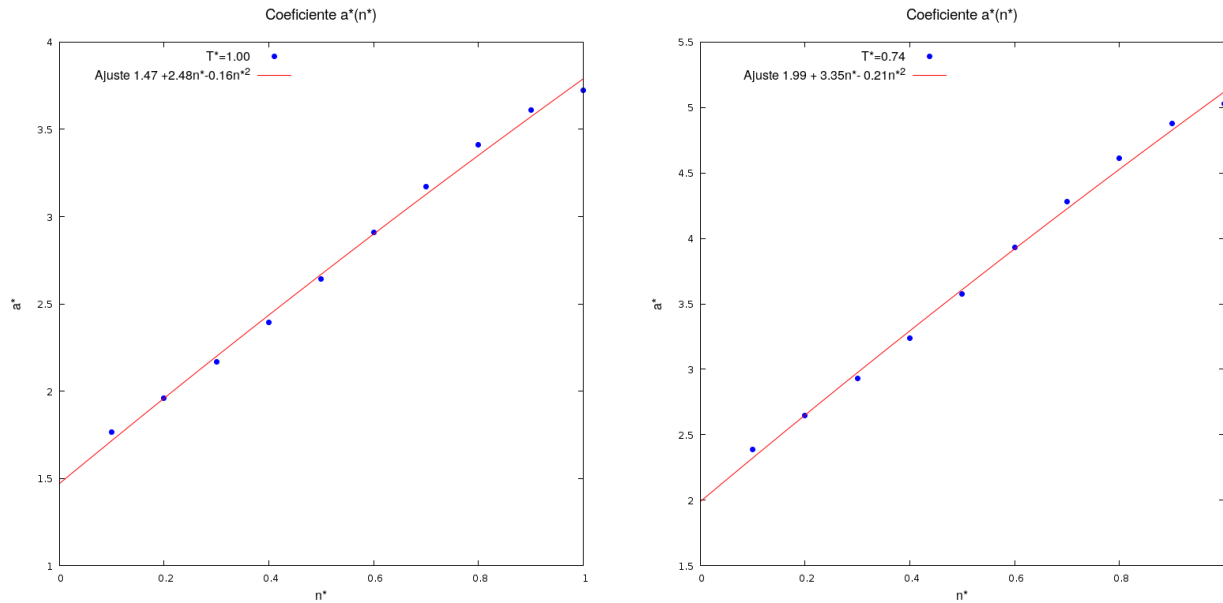


Figura 5: Ajuste del Coeficiente a^*

VIII. **Ecuación de Presión con Teoría de Perturbaciones de Zwanzig** Como ya conocemos la forma de las a^* , podemos calcular la ecuación de la presión a partir de la ecuación (19).

$$p^* = n^* [1 + n^*b^*] - \left[a^* + n^* \left(\frac{\partial a^*}{\partial n^*} \right)_{T^*} \right] n^{*2}$$

Se procedió a calcular la presión usando los dos casos de T^* . Lo que se obtuvo fue lo siguiente:

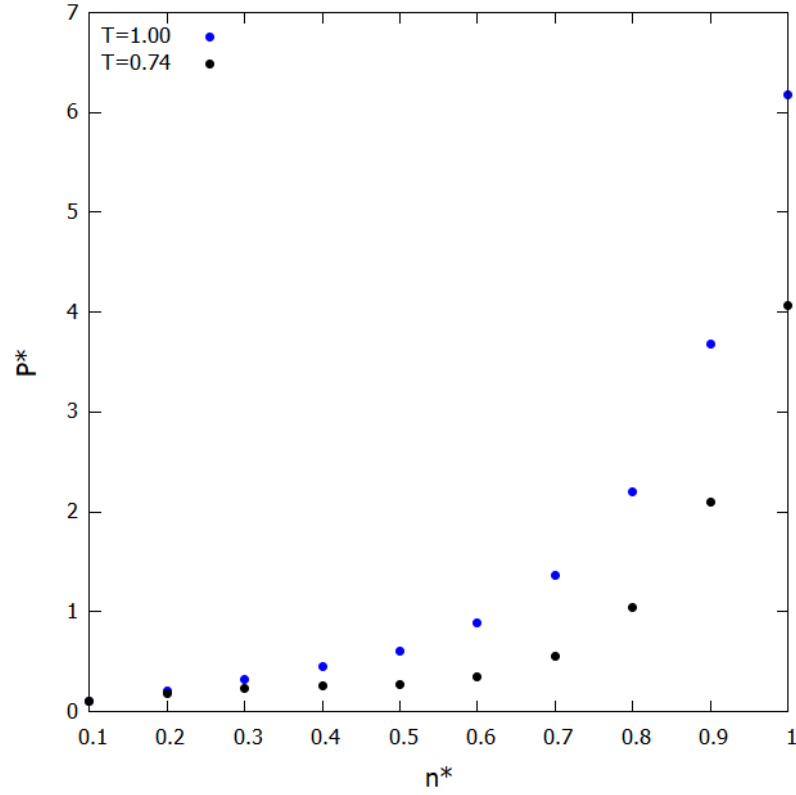


Figura 6: Isotermas de la presión de Zwanzig

Como escenario comparativo, se incluyeron las ecuaciones de Van Der Waals, Gas Ideal, Esferas Duras, junto a la de Teoría de Perturbaciones de Zwanzig. Se realizó para el caso de $T^* = 0.74$

Modelo	Ecuación de Presión
Gas Ideal	n^*
Esfera Dura (HS)	$n^* \left[1 + \frac{2\pi}{3} n^* g_{hs}(1^+) \right]$
Van Der Waals	$\frac{n^*}{1-n^*b^*} - n^{*2}a^*$
Zwanzig	$p_{hs}^* - \left[a^* + n^* \left(\frac{\partial a^*}{\partial n^*} \right)_{T^*} \right] n^{*2}$

Tabla 1: Ecuación de presión de diferentes modelos

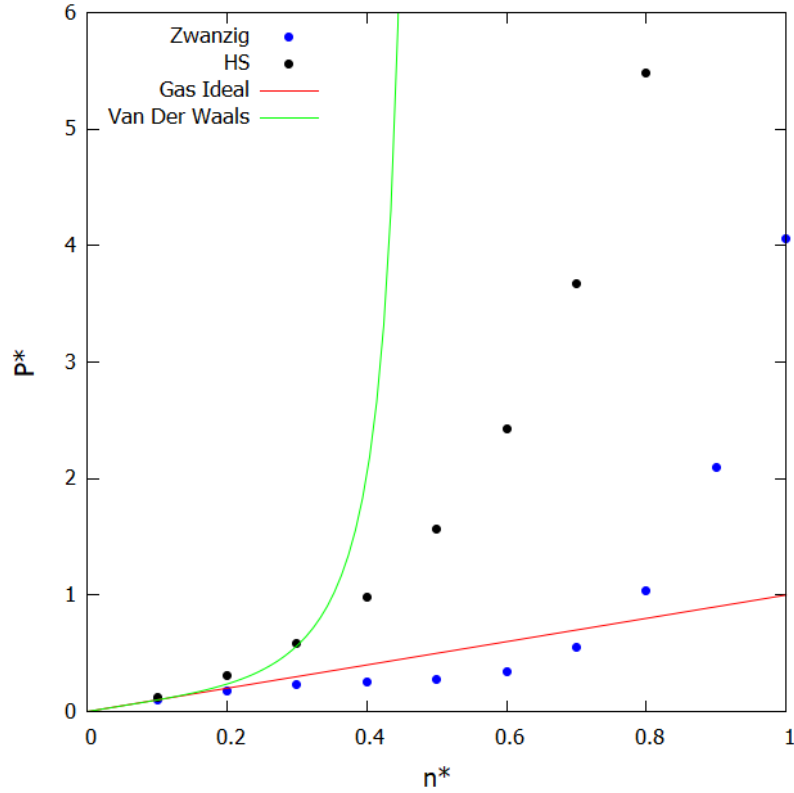


Figura 7: Ecuaciones de Estados para el caso de $T^* = 0.74$

IX. Energía potencial promedio

De la física estadística, con la teoría de Zwanzig para la función de partición es posible obtener una expresión para la energía media.

$$Z_N \approx (V - Nb)^N e^{-\beta N a^* n^*} \quad (37)$$

Siendo la energía promedio por partícula:

$$\begin{aligned} \bar{u} &= \frac{1}{N} \left(\frac{\partial \ln Z_n}{\partial \beta} \right) \\ &= \frac{1}{Z_N N} \left(\frac{\partial Z_n}{\partial \beta} \right) \\ &= \frac{(V - Nb)^N e^{-\beta N a^* n^*} (-N n^* a^*)}{(V - Nb)^N e^{-\beta N a^* n^*} N} \\ \bar{u} &= -a^* n^* \end{aligned} \quad (38)$$

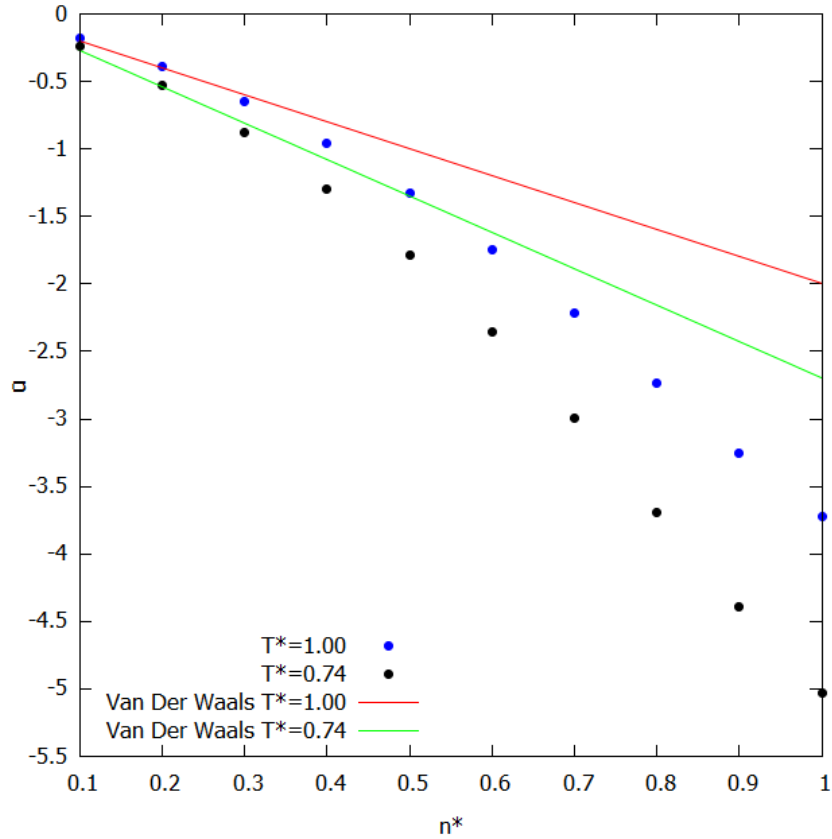
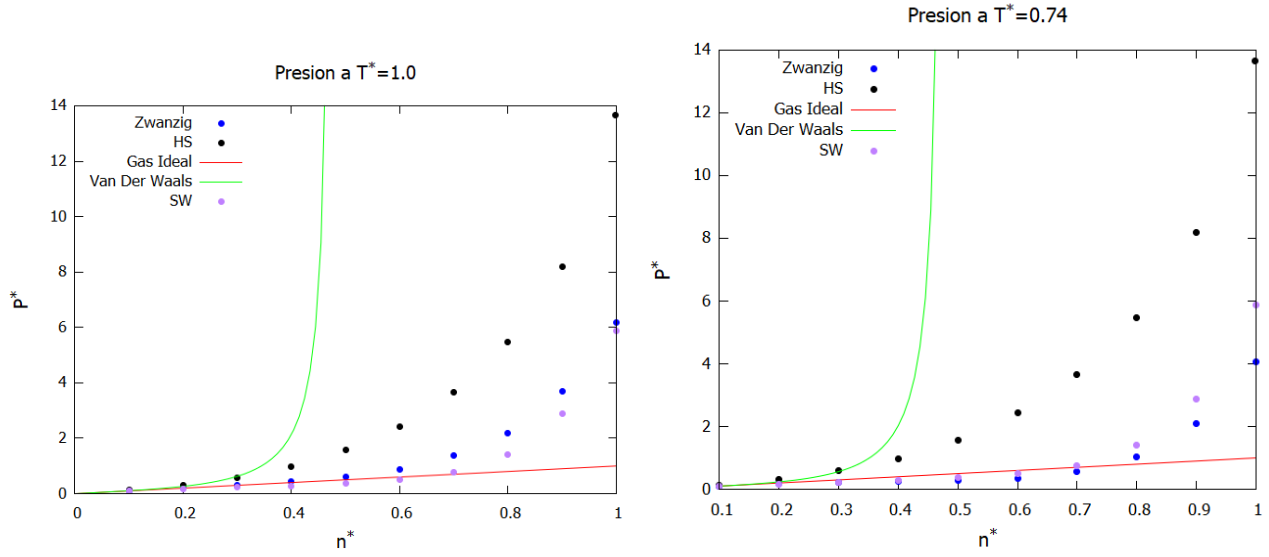
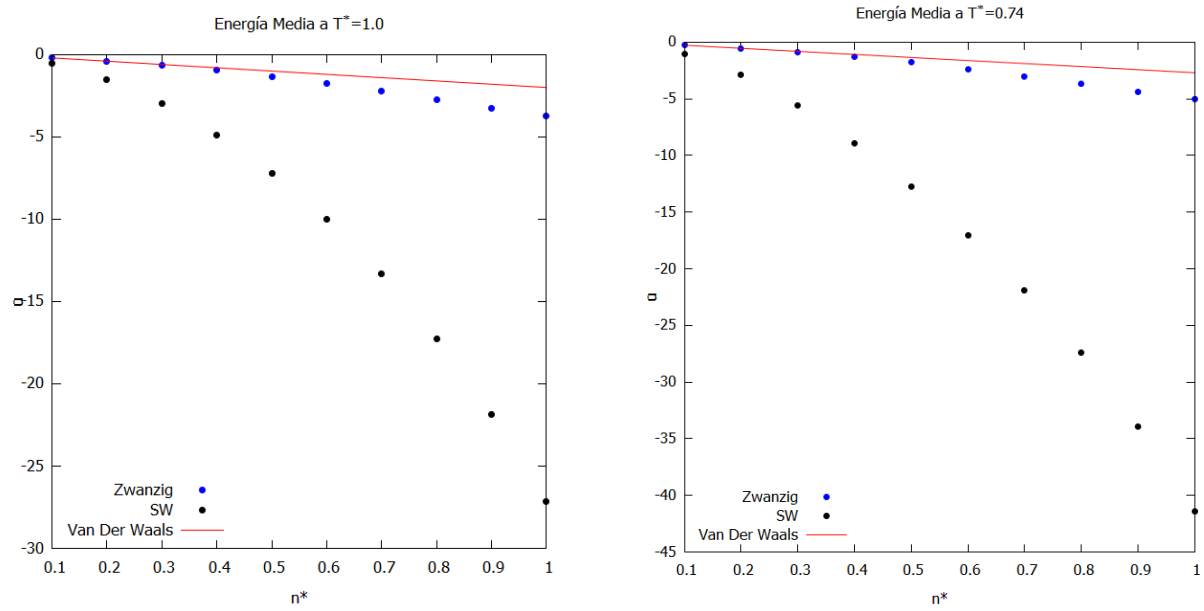


Figura 8: Energía Promedio para los casos de $T^* = 1.0$ y $T^* = 0.74$

X. **Validación de Modelos Teóricos** Ahora se implemento en el código de Monte Carlo el potencial de pozo cuadrado, con el cual se busca obtener la energía y la presión en función de la concentración para las Isothermas que se obtuvieron. Se utilizo una $\lambda = 1.25$. Los resultados fueron los siguientes:

Figura 9: Presión en función de la n^* de las IsothermasFigura 10: Energía en función de la n^* de las Isothermas

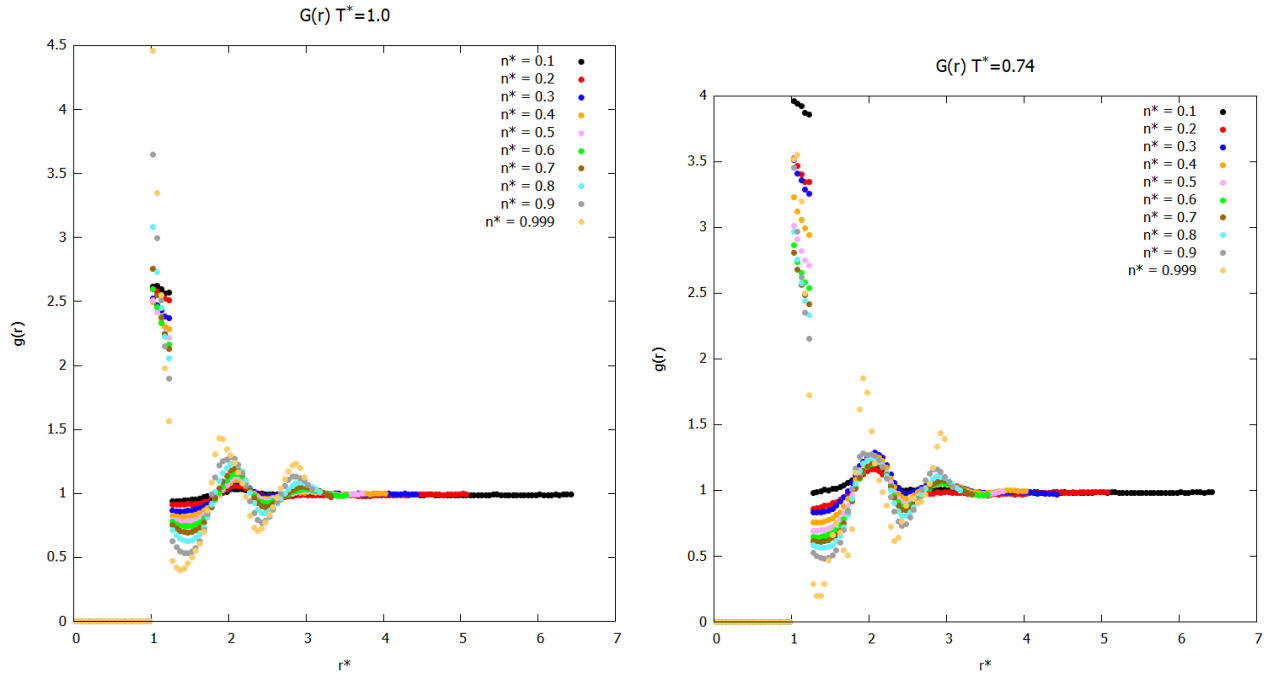


Figura 11: $G(r)$ de las Isotermas para varias concentraciones

4. Código Base

El siguiente código fue con lo que se corrió Monte Carlo

```

1 Module cte
2   Implicit None
3   Real, Parameter :: sigma = 1.0
4   Real, Parameter :: PI=4.0*atan(1.0)
5   Real, Parameter :: Lambda = 1.25                                !TAMANO DEL POZO
6   Real, Parameter :: TP = 0.74                                   !TEMPERATURA
7   REDUCIDA
8   Integer, Parameter :: CEq = 100000                             !CONFIG DE
9   EQUILIBRIO
10  Real :: BoxL, RCut, dRMax
11  Integer :: NN
12  Real, Parameter :: Dim = 1.0/3.0                                !DIMENSIONES (3D)
13  Real, Allocatable, Dimension(:) :: X, Y, Z                     !POSICIONES DE
14  LAS PARTICULAS
15  Real, Allocatable, Dimension(:,,:) :: CX, CY, CZ               !MATRICES DE
16  CONFIGURACION
17  !VARIABLES DE SIMULACION
18  Integer, Parameter :: N = 216
19  Integer, Parameter :: NStep = 300000                             !NSTEP - CEq
20  CONFIG DE EQUILIBRIO
21  Integer, Parameter :: IPrint = 1000

```



```

18 Integer, Parameter :: ISave = 100
19 Integer, Parameter :: IRatio = 100
20 Real, Parameter :: Dens = 0.999
21
22 End Module cte

```

Mod.f03

```

1  !=====
2  ! PROGRAMA PRINCIPAL DEL LA SIMULACION, LLAMA SUBROUTINAS PARA REALIZAR
3  ! LA SIMULACION
4  ! Autor: Martin Alejandro Paredes Sosa
5  !=====
6  Program Main
7  Use cte
8  Implicit None
9  Integer :: i, j, l, IStep , k, k2           !CONTADORES
10 Real :: VLRC, VI, V, VOld, VNew, DV, VN      !ENERGIAS
11 Real :: OldX, OldY, OldZ, NewX, NewY, NewZ    !VALORES TEMP DE POSC
12 Real :: RanX, RanY, RanZ, Dummy              !VALORES
13                                         !ALEATORIOS
14 Real :: MAcep, Ratio                         !VARIABLES DE CONTROL DE
15                                         !DRMAX
16 Logical :: Ctrl, Ctrl1, Ctrl1A, Ctrl2        !CONTROL LOGICO
17 Integer :: istat1, istat2
18 Character (len=80) :: err_msg1, err_msg2
19 Character (len = 12) :: Filename, chardens
20
21 !PEDIR DENSIDAD Y NUMERO DE PARTICULAS
22 Write(*,*) "
23
24 Write(*,*) "=====
25 Write(*,*) "NUMERO DE PARTICULAS"
26 Write(*,*) N
27 Write(*,*) "CONCENTRACION REDUCIDA"
28 Write(*,*) Dens
29 Write(*,*) "NUMERO DE CICLOS"
30 Write(*,*) NStep
31 Write(*,*) "MONITOREO EN PANTALLA (CADA CAUNTOS CICLOS)"
32 Write(*,*) IPrint
33 Write(*,*) "NUMERO DE PASOS PARA GUARDAR CONFIGURACION"
34 Write(*,*) ISave
35 Write(*,*) "FRECUENCIA DE CORRECCION EN DESPLAZAMIENTO"
36 Write(*,*) IRatio
37 Write(*,*) "
38
39 Write(*,*) "=====
40
41 4321 Format(I2.2)

```

```

37
38 !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLO DE POSICION DE PARTICULAS
39 Allocate( X(N), Y(N), Z(N))!, STAT= istat1 , ERRMSG=err_msg1 )
40
41
42 Open(1, File="ConfigIni.dat")
43
44 !GENERAR LA CONFIGURACION INICIAL
45 ConIni: If( Dens .LT. 0.7) Then
46
47     Call ConfigIni                                !CONFIG ALEATORIA
48
49 Else
50
51     Call ConfigIniReg                             !CONFIG REGULAR
52
53 End If ConIni
54 Write(*,*) "CONFIGURACION INICIAL LISTA"
55
56 !CALCULO/PARAMETROS PARA INICIALIZAR
57 RCut = BoxL / 2.0
58 dRMax = 0.1
59 MAcep = 0.0
60 k2 = 0
61 NN = ( NStep- CEq ) / ISave
62
63 !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLOS DE CONFIGURACION
64 Allocate( CX(N,NN), CY(N,NN), CZ(N,NN))!, STAT= istat2 , ERRMSG=err_msg2 )
65
66
67 !CORRECCION DE LARGO ALCANCE
68 VLRC = 0 !NO SE OCUPA LA CORRECCION POR SER DE CORTO ALCANCE
69
70 !CALCULAR LA ENERGIA DE LA CONFIGURACION
71 Call EnergyConfig(V)
72 VI = V + VLRC
73 Write(*,*) "ENERGIA DE LA CONFIGURACION INICIAL:", VI
74
75
76
77 !ABRIENDO ARCHIVOS PARA GUARDAR INFO DEL SISTEMA
78 Open(2, File="ConCeq.dat")
79
80 Write(chardens,4321) int( 10.0 * dens )
81 Filename = "Terma"//Trim(chardens)//".dat"
82 Open(3, File=Trim(Filename) )
83 Write(*,*) "Archivo ", trim(Filename)
84 Write(*,*) "
=====

```

```

85  Write(*,*) "|CONFIG||ENERGIA PARTICULA||RATIO||DR|"
86
87  !MOVIMIENTO DE PARTICULAS ALEATORIA
88
89  Configuracion: Do iStep = 1, NStep
90
91      Particula: Do i = 1, N
92
93          OldX = X(i)
94          OldY = Y(i)
95          OldZ = Z(i)
96
97          !CALCULAR LA ENERGIA DE LA i-PARTICULA
98          Call EnergyPart(OldX, OldY, OldZ, i, VOld)
99
100         !GENERAR VALORES ALEATORIOS PARA MOV TENTATIVOS
101         Call Random_Number(RanX)
102         Call Random_Number(RanY)
103         Call Random_Number(RanZ)
104
105         !MOVIMIENTO TENTATIVO
106         NewX = OldX + (2.0*RanX - 1.0)*dRMax
107         NewY = OldY + (2.0*RanY - 1.0)*dRMax
108         NewZ = OldZ + (2.0*RanZ - 1.0)*dRMax
109
110         !CONDICIONES PERIODICAS (MANTENER MISMA N EN TODA CONFIGURACION)
111         NewX = NewX - BoxL*Anint(NewX/BoxL)
112         NewY = NewY - BoxL*Anint(NewY/BoxL)
113         NewZ = NewZ - BoxL*Anint(NewZ/BoxL)
114
115         !CALCULAR LA ENERGIA DE LA PARTICULA EN LA NUEVA POSICION
116         Call EnergyPart(NewX, NewY, NewZ, i, VNew)
117
118         !MONTECARLO (CRITERIO DE ACEPTACION O RECHAZO DE MOV)
119         DV = VNew - VOld
120         Call Random_Number(Dummy) !PARA CRITERIO ENTRE 0.0 Y 75.0
121
122         !MONTECARLO (ACEPTANDO MOVIMIENTOS POR CRITERIOS)
123         MONTECARLO1: If(DV .LT. 75.0 ) Then
124
125             MONTECARLO2: If(DV .LE. 0.0 ) Then
126                 V = V + DV
127                 X(i) = NewX
128                 Y(i) = NewY
129                 Z(i) = NewZ
130                 MAcep = MAcep + 1.0 !MOVIMIENTO ACEPTADOS POR MONTECARLO
131
132             ElseIf( EXP(-DV) .GT. Dummy ) Then

```

```

133         V = V + DV
134         X(i) = NewX
135         Y(i) = NewY
136         Z(i) = NewZ
137         MAcep = MAcep + 1.0 !MOVIMIENTOS ACEPTADOS POR MONTECARLO
138
139     End If MONTECARLO2
140
141 End If MONTECARLO1
142
143 !ENERGIA POR PARTICULA
144 VN = (V+VLRC)/Real(N)
145
146
147 End Do Particula
148
149 !GUARDANDO LA THERMALIZACION DE CADA CONFIGURACION DEL SISTEMA
150 Write(3,*) IStep, VN
151
152
153
154 !AJUSTE DE DESPLAZAMIENTO DRMAX
155 Ctrl1 = Mod(IStep, IRatio) == 0
156 NdR : If (Ctrl1) Then
157
158     Ratio = MAcep / Real( N * IRatio ) !RAZON DE
159     ACEPTADOS
160     Ratios:If (Dens .LT. 0.3) Then
161         Ctrl1A = Ratio .GT. 0.95 !CRITERIO DE
162         ACEPTACION DE MOVIMIENTOS
163     Else
164         Ctrl1A = Ratio .GT. 0.5
165     End If Ratios
166
167     Criterio : If ( Ctrl1A ) Then
168         dRMax = dRMax * 1.05 !CRECER
169         DESPLAZAMIENTO
170     Else
171         dRMax = dRMax * 0.95 !DISMINUIR
172         DESPLAZAMIENTO
173     End If Criterio
174
175     MAcep = 0.0 !REINICIAR
176     CONTADOR DE MOV ACEPTADOS
177
178 End If NdR
179
180 !MONITOREO EN PANTALLA
181 Ctrl = Mod(IStep,IPrint) == 0 !CADA QUE TANTO

```

```
177 IMPRIMIR EN PANTALLA ENERGIA, DRMAX
178 MonitoreoEne: If(Ctrl) Then
179     Write(*,*) ISTEP, VN, Ratio , dRMax
180
181 End If MonitoreoEne
182
183
184 !GUARDANDO CONFIG DE EQUILIBRIO
185 CEQI: IF (IStep == CEq) Then
186     Do l = 1 , N
187         Write(2,*) X(1), Y(1), Z(1)
188     End Do
189     Write(*,*) "Configuracion CEq"
190     Close(2)
191 End IF CEQI
192
193
194
195 !GUARDANDO CONFIGURACION (EN EQUILIBRIO)
196 Ctrl2 = ( Mod(IStep,ISave) == 0 ) .AND. ( IStep .GT. CEq )
197 SAV: If (Ctrl2) Then
198
199     k2 = k2 + 1
200
201     SAV1:Do k = 1 , N
202
203         CX(k,k2) = X(k)
204         CY(k,k2) = Y(k)
205         CZ(k,K2) = Z(k)
206
207     End Do SAV1
208
209 End If SAV
210
211
212 End Do Configuracion
213
214 Write(*,*) "DONE ALL CONFIGURATIONS"
215
216 !GUARDAR CONFIG FINAL
217 !ConfigFin: Do i=1, N
218
219 !   Write(2,*) X(i), Y(i), Z(i)
220
221 !End Do ConfigFin
222 !Close (2)
223
224
```

```

225 WRITE(*,*) "DONE SAVING CONFIG FINAL"
226
227 Deallocate( X, Y, Z )
228 WRITE(*,*) "CLEAR MEMORY" !DEBUG
229
230
231 Write(chardens,4321) int( 10.0 * dens )
232 Filename = "pres"//Trim(chardens)//".dat"
233 Open(27, File=Trim(Filename) )
234
235
236 Call GdrCalc(1)
237 WRITE(*,*) "GDR DONE CALC" !DEBUG
238
239 Deallocate( CX, CY, CZ )!, STAT=istat1, ERRMSG = err_msg1 )
240 !Write(*,*) istat1, err_msg1 !DEBUG
241 Close(3)
242
243 WRITE(*,*) "DONE"
244 WRITE(*,*) "
=====
"
245
246
247
248
249 End Program Main

```

Main.f03

```

1 !=====
2 ! CONSTRUCCION DE UNA CONFIGURACION INICIAL ALEATORIA EN CELDA BIDIMENSIONAL
3 ! SIN TRASLAPES
4 ! Autor: Martin Paredes Sosa
5 !=====
6
7 Subroutine ConfigIni
8   Use cte
9   Implicit None
10  Real :: xRan, yRan, zRan, xij, yij, zij, dist      !POSC
11  Integer :: i, j                                  !CONTADOR
12
13  !CALCULANDO DIMENSIONES DE LA CAJA
14  BoxL = (1.0*N/Dens )**Dim
15  Write(*,*) "LONGITUD DE LA CELDA:", BoxL
16
17  !Open (1, File = "ConIni.dat" )
18
19  Colocar: Do i=1, N                                !BUSCAR LA POSICION ALEATORIA PARA LAS PARTICULAS
20    2 Call Random_Number(xRan) !VALOR ALEATORIO DE POSICION X \

```

```

21 Call Random_Number(yRan) !VALOR ALEATORIO DE POSICION Y | TENTATIVO
22 Call Random_Number(zRan) !VALOR ALEATORIO DE POSICION Z /
23
24 !COLOCAR DENTRO DE LA CELDA
25
26 X(i) = (xRan-0.5)*(BoxL-1) !\
27 Y(i) = (yRan-0.5)*(BoxL-1) !| [-(BoxL-1)/2 , (BoxL-1)/2]
28 Z(i) = (zRan-0.5)*(BoxL-1) !/
29
30 Traslape: Do j=1 , i-1
31
32 xij = X(i) - X(j) !CALCULANDO LA DISTANCIA ENTRE PARTICULAS
33 yij = Y(i) - Y(j)
34 zij = Z(i) - Z(j)
35
36 dist = xij*xij + yij*yij + zij*zij
37
38 DectTraslape: If(dist .LE. sigma ) Then
39
40 !Write(*,*) "TRASLAPE", i, j !DEBUG
41 GO TO 2
42
43 End If DectTraslape
44
45 End Do Traslape
46
47 Write(1,*) X(i), Y(i), Z(i) !GUARDANDO EN ARCHIVO LA POSICION
48
49 End Do Colocar
50
51 Close(1)
52
53
54 End Subroutine ConfigIni

```

ConfigIni.f03

```

1 !=====
2 ! CONSTRUCCION DE UNA CONFIGURACION INICIAL REGULAR EN CELDA BIDIMENSIONAL
3 ! SIN TRASLAPES
4 ! Autor: Martin Paredes Sosa
5 !=====
6
7 Subroutine ConfigIniReg
8 Use cte
9 Implicit None
10 !Real :: xRan, yRan,zRan, xij, yij, zij, dist
11 Real :: dBoxL
12 Integer :: i, j, k ,1 !CONTADOR
13 Integer :: N2, N3

```

```

14 Real, Dimension(:), Allocatable :: nX, nY, nZ      !GEN
15
16
17 !CALCULANDO DIMENSIONES DE LA CAJA
18 N2 = anint( N**(Dim) )
19
20 !BoxL = (1.0*N/Dens )**(Dim)
21
22 N3 = N2**(1.0/Dim)
23 !N = N3
24 BoxL = (1.0*N/Dens )**(Dim)
25
26 Write(*,*) "LONGITUD DE LA CELDA:", BoxL
27 Write(*,*) "TOTAL DE PARTICULAS COLOCADAS EN LA CELDA:", N3
28 dBoxL = BoxL/N2
29
30
31 Allocate( nX(N2), nY(N2), nZ(N2) )
32
33 !GENERANDO COORDENADAS PARA POSICIONES DE LAS PARTICULAS
34 GEN: Do i=1, N2
35
36     nx(i) = (-BoxL)/2.0 + dBoxL/2.0 + dBoxL*(i-1)
37     ny(i) = (-BoxL)/2.0 + dBoxL/2.0 + dBoxL*(i-1)
38     nz(i) = (-BoxL)/2.0 + dBoxL/2.0 + dBoxL*(i-1)
39
40 End Do GEN
41
42 !ESCRIBIENDO EN ARCHIVO
43 !Open (1, File = "ConIni.dat" )
44 l = 0
45 EscribirX: Do i = 1, N2
46
47     EscribirY: Do j = 1, N2
48
49         EscribirZ: Do k = 1, N2
50
51             l = l + 1
52             X(l) = nX(i)
53             Y(l) = nY(j)
54             Z(l) = nZ(k)
55
56         End Do EscribirZ
57
58     End Do EscribirY
59
60 End Do EscribirX
61
62 !Write(*,*) l !DEBUG

```



```

63
64 Do i=1, N3
65     Write(1,*) X(i), Y(i), Z(i)
66 End Do
67
68
69 Deallocate(nX, nY, nZ)
70
71 Close(1)
72
73
74 End Subroutine ConfigIniReg

```

ConfigIniReg.f03

```

1  !=====
2  ! CALCULO DE LA ENERGIA DE UNA DE LAS PARTICULAS DE LA CELDA
3  ! ESFERA DURA (HS)
4  !
5  ! Autor: Martin Alejandro Paredes Sosa
6  !=====
7
8 Subroutine EnergyPart(Rx1, Ry1, Rz1, i, V)
9     Use cte
10     Implicit None
11     Real :: VNew, Dist, Rxd, Rzd, Ryd
12     Real, Intent(In) :: Rx1, Ry1, Rz1
13     Real, Intent(Out) :: V
14     Integer, Intent(In) :: i
15     Integer :: j
16     !INICIAR ENERGIA EN 0
17     V = 0.0
18
19     BuscarPart: Do j=1, N
20
21         NoLaMisma: If(i .NE. j) Then
22
23             Rxd = Rx1 - X(j)
24             Ryd = Ry1 - Y(j)
25             Rzd = Rz1 - Z(j)
26
27             !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
28             Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
29             Ryd = Ryd - BoxL*Anint(Ryd/BoxL)
30             Rzd = Rzd - BoxL*Anint(Rzd/BoxL)
31
32             !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
33             Dist = sqrt( Rxd*Rxd + Ryd*Ryd + Rzd*Rzd )
34             !If(Dist .LE. 1.0) Write(*,*) Dist, i,j
35

```

```

36      ChecarInter: If(Dist .LT. RCut) Then
37
38          ChecarCercania: If (Dist .LE. 1.0) Then
39              VNew = 1.0E+10
40          Else
41              VNew = 0
42          End If ChecarCercania
43          V = V + VNew
44      End If ChecarInter
45
46
47      End If NoLaMisma
48
49      End Do BuscarPart
50
51      End Subroutine EnergyPart

```

EnergyPartHS.f03

```

1  !=====
2  ! CALCULO DE LA ENERGIA DE UNA DE LA CONFIGURACION DE LA CELDA
3  ! ESFERA DURA (HD)
4  !
5  ! Autor: Martin Alejandro Paredes Sosa
6  !=====
7
8  Subroutine EnergyConfig(V)
9      Use cte
10     Implicit None
11     Real :: V, Rx1, Rxd, Ry1, Ryd, Rz1, Rzd, Dist, VNew
12     Integer :: i, j
13     V = 0
14     IterPart: Do i=1, N-1
15
16         Rx1 = X(i)
17         Ry1 = Y(i)
18         Rz1 = Z(i)
19
20         IterPart2: Do j = i+1, N
21             Rxd = Rx1 - X(j)
22             Ryd = Ry1 - Y(j)
23             Rzd = Rz1 - Z(j)
24
25             !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
26             Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
27             Ryd = Ryd - BoxL*Anint(Ryd/BoxL)
28             Rzd = Rzd - BoxL*Anint(Rzd/BoxL)
29
30             !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
31             Dist = sqrt( Rxd*Rxd + Ryd*Ryd + Rzd*Rzd )

```

```

32
33      ChecarInter: If(Dist .LT. RCut) Then
34
35          ChecarCercania: If (Dist .LE. 1.0) Then
36
37              VNew = 1.0E+10
38
39          Else
40
41              VNew = 0
42
43          End If ChecarCercania
44
45          V = V + VNew
46
47      End If ChecarInter
48
49  End Do IterPart2
50 End Do IterPart
51
52 End Subroutine EnergyConfig

```

EnergyConfigHS.f03

```

1  !=====
2  ! CALCULO DE LA ENERGIA DE UNA DE LAS PARTICULAS DE LA CELDA
3  ! POZO CUADRADO (SW)
4  !
5  ! Autor: Martin Alejandro Paredes Sosa
6  !=====
7
8 Subroutine EnergyPart(Rx1, Ry1, Rz1, i, V)
9   Use cte
10  Implicit None
11  Real :: V, VNew, Dist, Rx1, Rxd, Ry1, Rz1, Rzd, Ryd
12  Integer :: i, j
13  !INICIAR ENERGIA EN 0
14  V = 0
15
16  BuscarPart: Do j=1, N
17
18      NoLaMisma: If(i .NE. j) Then
19
20          Rxd = Rx1 - X(j)
21          Ryd = Ry1 - Y(j)
22          Rzd = Rz1 - Z(j)
23
24          !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
25          Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
26          Ryd = Ryd - BoxL*Anint(Ryd/BoxL)

```

```

27 Rzd = Rzd - BoxL*Anint(Rzd/BoxL)
28
29 !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
30 Dist = sqrt( Rxd*Rxd + Ryd*Ryd + Rzd*Rzd )
31 !If(Dist .LE. 1.0) Write(*,*) Dist, i,j
32
33 ChecarInter: If(Dist .LT. RCut) Then
34
35     ChecarCercania: If (Dist .LE. 1.0) Then                ! DESPUES DEL
    POZO
36         VNew = 1.0E+10
37         Else If( (Dist .GT. 1.0) .AND. (Dist .LT. Lambda) ) Then ! EL POZO
38             VNew = -1.0 / TP
39         Else                                                ! ANTES DEL
    POZO
40             VNew = 0
41         End If ChecarCercania
42         V = V + VNew
43     End If ChecarInter
44
45
46     End If NoLaMisma
47
48 End Do BuscarPart
49
50 End Subroutine EnergyPart

```

EnergyPartSW.f03

```

1  !=====
2  ! CALCULO DE LA ENERGIA DE UNA DE LA CONFIGURACION DE LA CELDA
3  ! POZO CUADRADO (SW)
4  !
5  ! Autor: Martin Alejandro Paredes Sosa
6  !=====
7
8  Subroutine EnergyConfig(V)
9      Use cte
10     Implicit None
11     Real :: V, Rx1, Rxd, Ry1, Ryd, Rz1, Rzd, Dist, VNew
12     Integer :: i, j
13     V = 0
14     IterPart: Do i=1, N-1
15
16         Rx1 = X(i)
17         Ry1 = Y(i)
18         Rz1 = Z(i)
19
20         IterPart2: Do j = i+1, N
21             Rxd = Rx1 - X(j)

```

```

22      Ryd = Ry1 - Y(j)
23      Rzd = Rz1 - Z(j)
24
25      !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
26      Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
27      Ryd = Ryd - BoxL*Anint(Ryd/BoxL)
28      Rzd = Rzd - BoxL*Anint(Rzd/BoxL)
29
30      !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
31      Dist = sqrt( Rxd*Rxd + Ryd*Ryd + Rzd*Rzd )
32
33      ChecarInter: If(Dist .LT. RCut) Then
34
35          ChecarCercania: If (Dist .LE. 1.0) Then                ! DESPUES DEL
POZO
36              VNew = 1.0E+10
37          Else If( (Dist .GT. 1.0) .AND. (Dist .LT. Lambda) ) Then ! ZONA DEL POZO
38              VNew = -1.0 / TP
39          Else                                                    ! ANTES DEL
POZO
40              VNew = 0
41          End If ChecarCercania
42
43          V = V + VNew
44          End If ChecarInter
45
46      End Do IterPart2
47      End Do IterPart
48
49      End Subroutine EnergyConfig

```

EnergyConfigSW.f03

```

1  !=====
2  ! EL PROGRAMA REALIZA EL CALCULO DE LA GDR APARTIR DE LAS DIFERENTES
3  ! CONFIGURACIONES REALIZADAS EN EL PROGRAMA PRINCIPAL (3D)
4  !
5  ! AUTOR: MARTIN ALEJANDRO PAREDES SOSA
6  !=====
7
8  Subroutine GdrCalc(l)
9
10     Use cte
11     Implicit None
12
13     Integer,intent(in) :: l
14     Integer, Allocatable, Dimension(:) :: Histo
15     Real, Parameter :: delTar = 0.05
16     Integer :: MBin, iBin, NNN
17     Integer :: i, j, k                                ! CONTADORES

```

```

18 Real :: x0, y0, z0, xN, yN, zN, xON, yON, zON
19 Real :: rD, rU, rL, rM, c1, c2, gdr, gdrM, press, b
20 Integer :: istat1
21 Character (len=80) :: err_msg1
22 Character (len = 10) :: Filename, chardens
23 Logical :: Ctrl1, Ctrl2
24
25 1234 Format(I2.2)
26
27 MBin = Int( RCut / delTar )
28 NNN = Mbin + 1
29 Allocate( Histo(NNN) , STAT = istat1, ERRMSG = err_msg1)
30
31 Histo = 0
32
33 MBin = Int( RCut / delTar )
34
35 Parti0 : Do i = 1, N
36
37     NextParti : Do j = 1, N
38
39         NOTSAME : If (i /= j ) Then
40
41             StepCnfg : Do k = 1, NN
42
43                 !PARTICULA i ORIGEN
44                 x0 = CX( i , k )
45                 y0 = CY( i , k )
46                 z0 = CZ( i , k )
47
48                 !PARTICUAL j CERCANA
49                 xN = CX( j , k )
50                 yN = CY( j , k )
51                 zN = CZ( j , k )
52
53                 !DISTANCIA
54                 xON = xN - x0
55                 yON = yN - y0
56                 zON = zN - z0
57
58                 !CONDICION DE IMAGEN MINIMA
59                 xON = xON - BoxL*Anint( xON/BoxL )
60                 yON = yON - BoxL*Anint( yON/BoxL )
61                 zON = zON - BoxL*Anint( zON/BoxL )
62
63                 !DISTANCIA ENTRE LAS PARTICULAS
64                 rD = sqrt( (xON * xON) + (yON * yON) + (zON*zON) )
65
66                 !CERCANIA CINTA

```

```

67         iBin = Int( rD / delTar ) + 1
68
69         Guardar : If((iBin .LE. MBin) ) Then
70
71             Histo(iBin) = Histo(iBin) + 1
72
73             End If Guardar
74
75         End Do StepCnfg
76
77     End If NOTSAME
78
79 End Do NextParti
80
81 End Do Parti0
82
83
84
85 c1 = ( 4.0 / 3.0 ) * PI * Dens
86 Write(chardens,1234) int(dens*10)
87 Filename = "gdr"//Trim(chardens)//".dat"
88
89 !ABRIENDO ARCHIVO PARA GDR
90 Open( 5, file= Trim(filename) )
91
92 GdrCal: Do ibin = 1 , MBin
93
94     rL = Real(iBin - 1) * delTar
95
96     rU = rL + delTar
97     rM = rL + ( delTar/2.0 )
98
99     c2 = c1 * ( ( rU**3 ) - ( rL**3 ) )
100    gdrm = gdr
101    gdr = Real( Histo(iBin) )/ Real(NN) / Real(N) / c2
102
103    Write(5,*) rM , gdr
104
105
106    Ctrl1 = gdrm == 0                                     !CALCULO DE GDR ESFERA
107    DURA
108    Ctrl2 = gdr /= 0
109    PressCalc:If ( Ctrl1 .AND. Ctrl2 ) Then
110
111        b = (2.0/3.0)* PI * gdr                             !PARAMETRO DE VAN DER
112        WAALS
113        Press = dens*(1.0 + dens*b)                         !PRESION DISCO DURO
114        Write(27,*) dens, press
115        Write(*,*) dens, press

```

```
114
115     End If PressCalc
116
117 End Do GdrCal
118
119 Close(5)
120
121 Deallocate( Histo )
122
123 Write(*,*) "GDR DONE, SAVE"
124
125 End Subroutine GdrCalc
```

GDR.f03

Referencias

- [1] D. Kandepudi and I. Prigogine. *Modern Thermodynamics*, chapter 1.
- [2] D. McQuarrie. *Statistical Mechanics*, chapter 13-14.