# Comentarios, Desarrollos u Observaciones

Desarrollo Experimental II

Docente: Dra. Laura Lorenia Yeomans Reyna

**Portafolio II**:
Simulación de Monte Carlo

Martín Alejandro Paredes Sosa

Semestre: 2018-1

# Tarea III: Ejercicios para movimientos arbitrarios de partículas, condiciones periódicas y energía de la configuración

A continuación se muestran los comentarios y avances relacionados con la tarea 3 del portafolio II.

## Actividad 1: Sin Condiciones Periódicas

La actividad consistió en realizar el movimiento de partículas de manera aleatoria, sin considerar condiciones periódicas, es decir la partículas no vuelven a entrar a la celda.

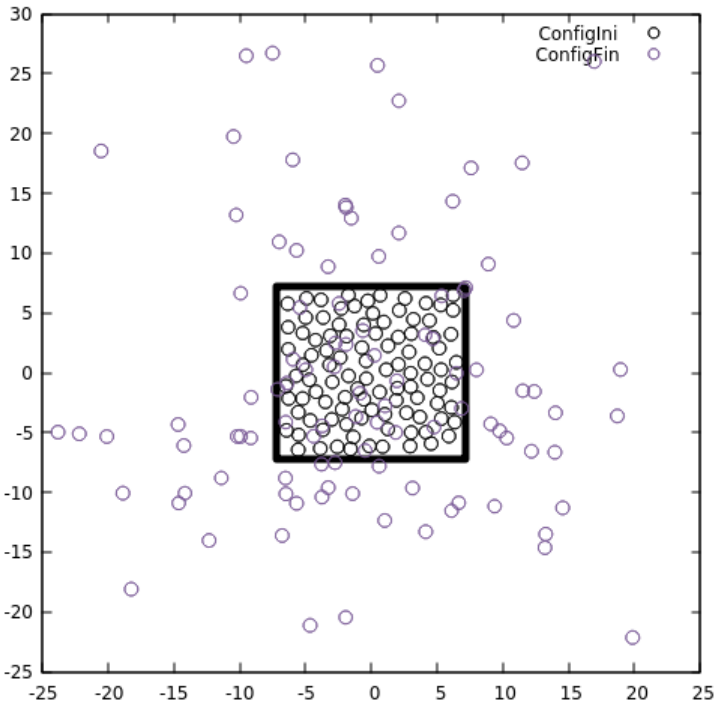


Figura 1: Configuración Inicial con $n^* = 0.5093$, con un $NStep = 1000$ y un $\delta_{MAX} = 0.5\sigma$

Se observa que las partículas se alejan consideradamente de las fronteras de la celda original.

## Actividad 2

La actividad consistió en realizar el movimiento de partículas de manera aleatoria, ahora considerando condiciones periódicas, es decir la partículas vuelven a entrar a la celda, cuando una sale, así asegurando que la concentración en la celda original se mantiene.

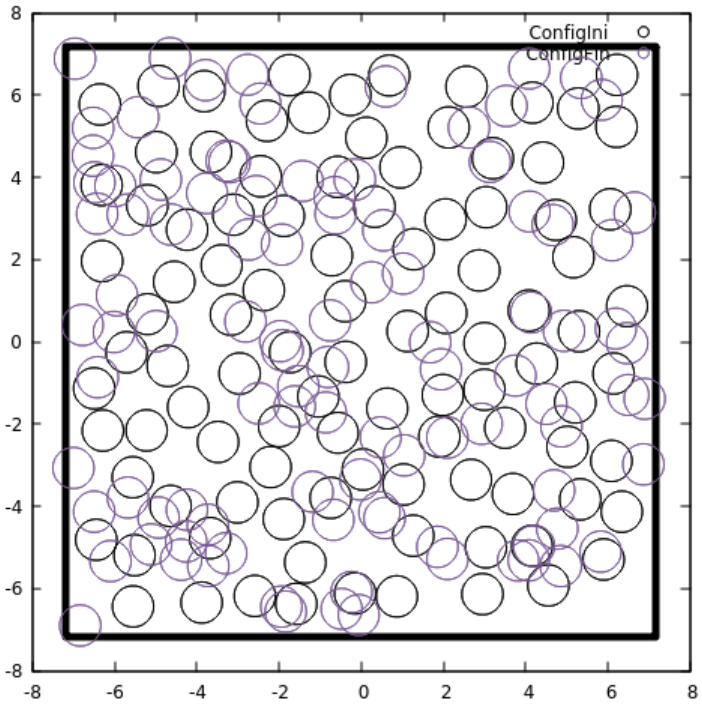


Figura 2: Configuración Inicial con $n^* = 0.5093$, con un $NStep = 1000$ y un $\delta_{MAX} = 0.5\sigma$

Se observa que todas las partículas están dentro de la celda original, ligeramente afuera pero sus centros siguen dentro de las fronteras.

Estas dos actividades permiten observar la forma en la que se mueven las partículas. La aplicación de condiciones periódicas es sencillo de implementar ya que consiste en volver a ingresar una partícula de lado opuesto de donde salio la otra.

## Actividad 3

La tercera actividad consistió en seguir el movimiento de dos partículas aleatorias. Esto permite observar las condiciones periódicas en acción.

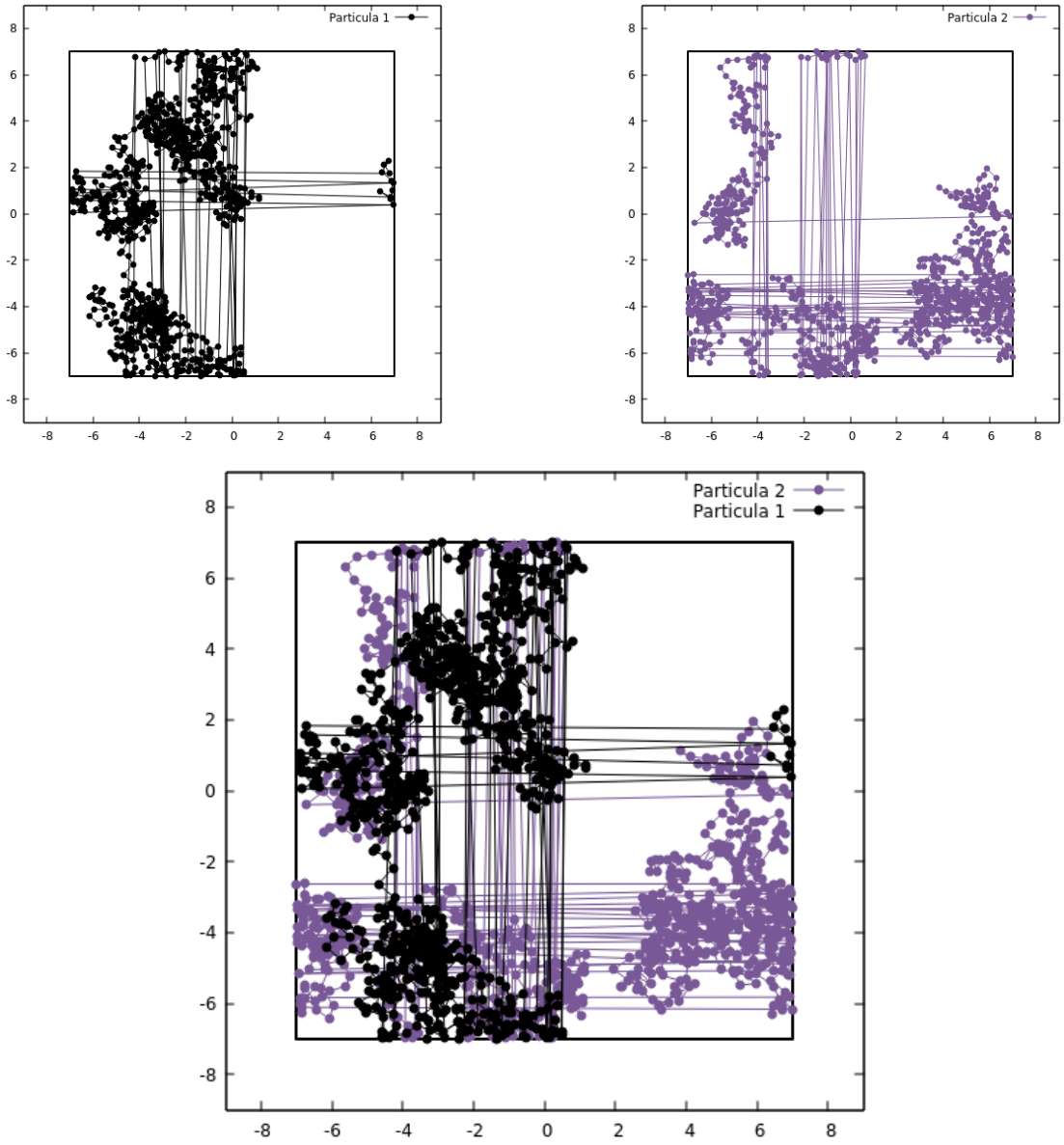


Figura 3: Con $n^* = 0.5093$, con un $NStep = 1000$ y un $\delta_{MAX} = 0.5\sigma$. Se realizo la traza de las partículas 84 y 57.

Se aprecia como cuando una partícula sale de las fronteras de la celda esta entra de lado opuesto cuando aparecen las lineas largas que atraviesan la celda.

## Actividad 4

La cuarta actividad consistió en realizar lo mismo que las actividades anteriores solo que adaptando a tres dimensiones.

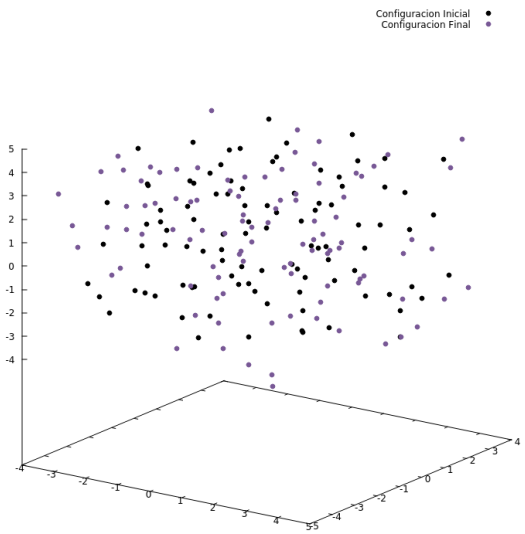


Figura 4: Con $n^* = 0.1910$, con un $NStep = 100000$ y un $\delta_{MAX} = 0.5\sigma$.

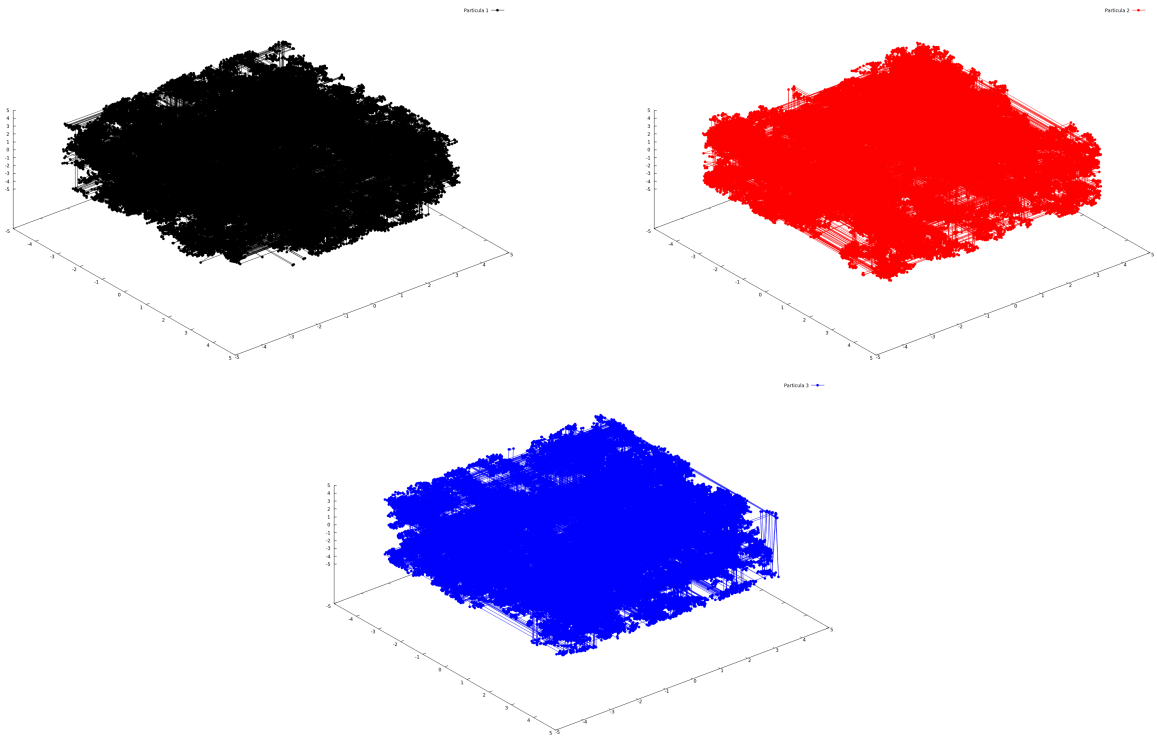

Figura 5: Con $n^* = 0.1910$, con un $NStep = 100000$ y un $\delta_{MAX} = 0.5\sigma$. Se realizo la traza de las partículas 84, 57 y 4.

Se observa las configuraciones inicial y final de un arreglo en 3D. Las trazas de las partículas se aprecia algo parecido a la de la actividad 3 por las condiciones periódicas.

## Actividad 5

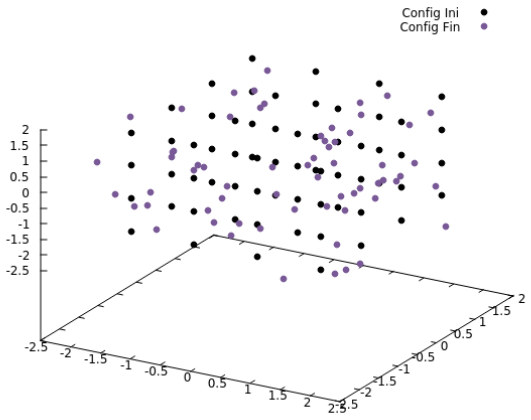


Figura 6: Con $n^* = 1.0$, con un $NStep = 1000$ y un $\delta_{MAX} = 0.5\sigma$.

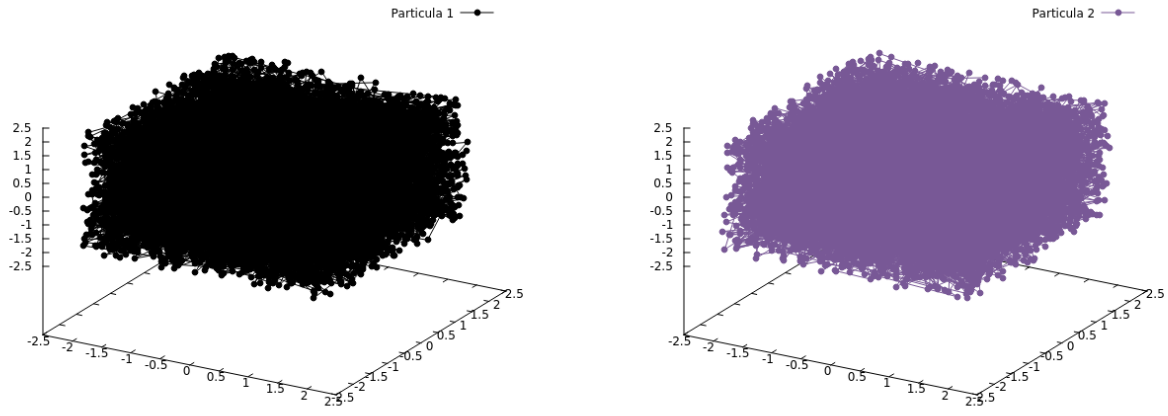


Figura 7: Con $n^* = 1.0$, con un $NStep = 1000$ y un $\delta_{MAX} = 0.5\sigma$. Se realizo la traza de las partículas 64 y 36.

En la actividad se utilizó una configuración regular cúbica en lugar de una aleatoria. y se observan las mismas cosas.

## Actividad 6

En esta actividad se realizo el cálculo de la energía de un sistema de disco duros (HD). Se obtuvo lo siguiente.

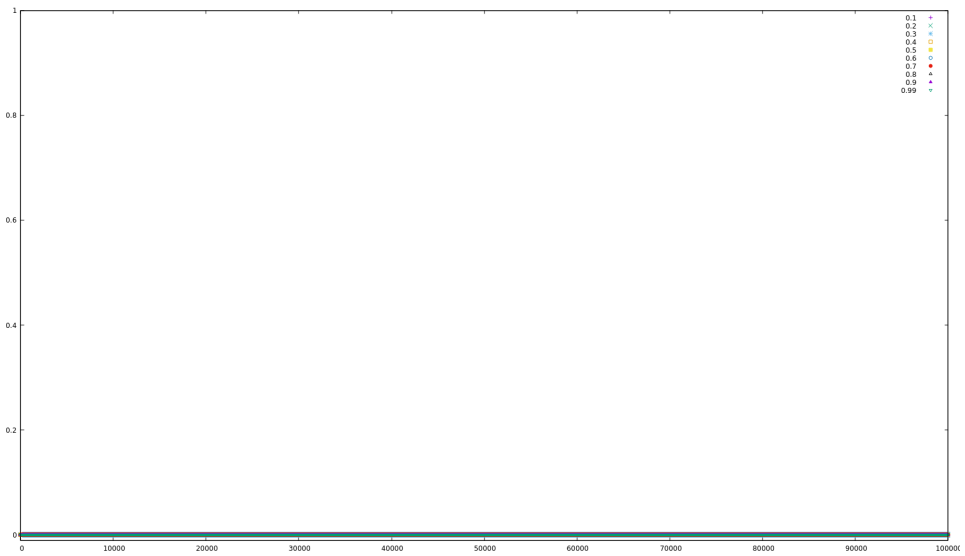


Figura 8: Termalización de HD para concentraciones de 0.1 a 0.99

Como es el potencial de discos duro, todas las configuraciones tuvieron energía 0 como era de esperarse.

# Tarea IV: Ejecución de Código Monte Carlo

La tarea 4 consiste de una actividad, en la cual se implementó el código de Montecarlo.

## Actividad 7

Lo que se realizó en la actividad fue generar lo archivos que nos permiten observar como se va comportando la energía conforme pasan las configuraciones. El objetivo de este observar cuando un sistema se termaliza, es decir, cuando este alcanza un estado de equilibrio.

Se realizaron corridas de la simulación de Monte Carlo para las concentraciones de $n^* = 0.1$ y para la de $n^* = 0.5$, con $10,000$ configuraciones y 100 partículas. Se escogieron estas ya que se nos pide realizar estas corridas para configuraciones iniciales diferentes, pero para altas concentraciones la configuración aleatoria sin traslapes, suele tomar mucho tiempo o no se puede llegar a una que cumpla con las condiciones.

Empezando para la configuración aleatoria sin traslapes, llegamos a lo siguiente:

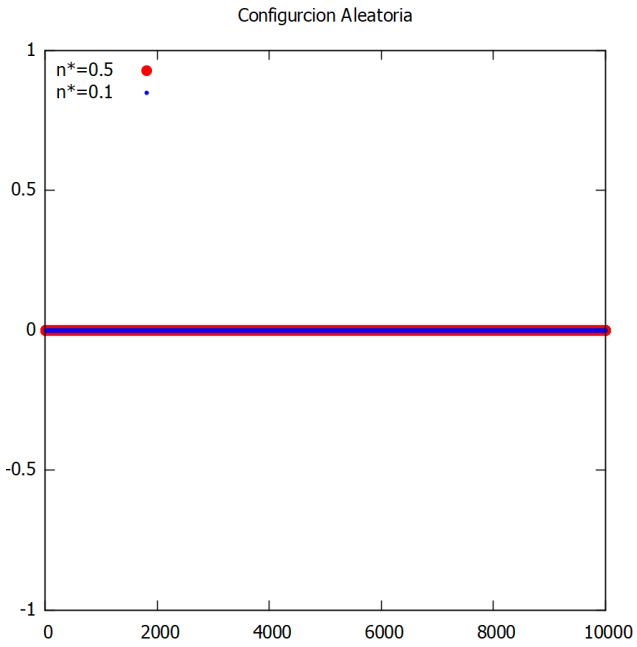


Figura 9: Termalización de simulación Monte Carlo con configuración Inicial Aleatoria para concentraciones de 0.1 y 0.5

Para la configuración regular, llegamos a lo siguiente:

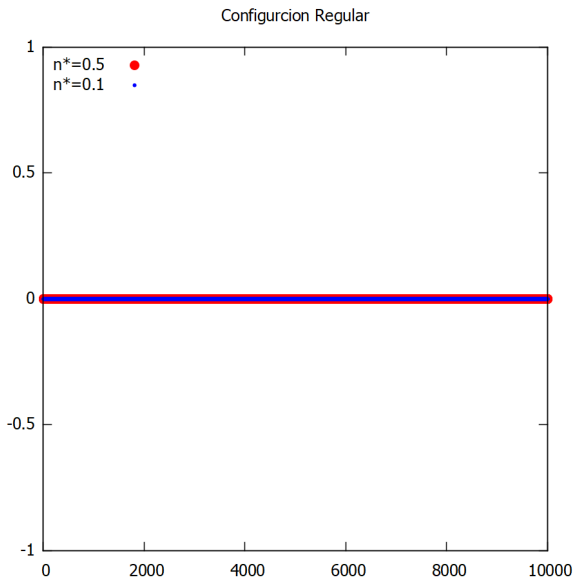


Figura 10: Termalización de simulación Monte Carlo con configuración Inicial Regular para concentraciones de 0.1 y 0.5

Podemos apreciar que la energía en ambas siempre es cero como es de esperarse, ya que para que la energía aumente en el sistema, la partículas se deberían encontrar traslapadas, es decir que estas al colisionar no fueran duras.

Para la configuración aleatoria con traslapes es otra historia, ya que al permitir que estas se encuentre encimadas, el sistema comienza con mucha energía. Lo obtenido fue lo siguiente.

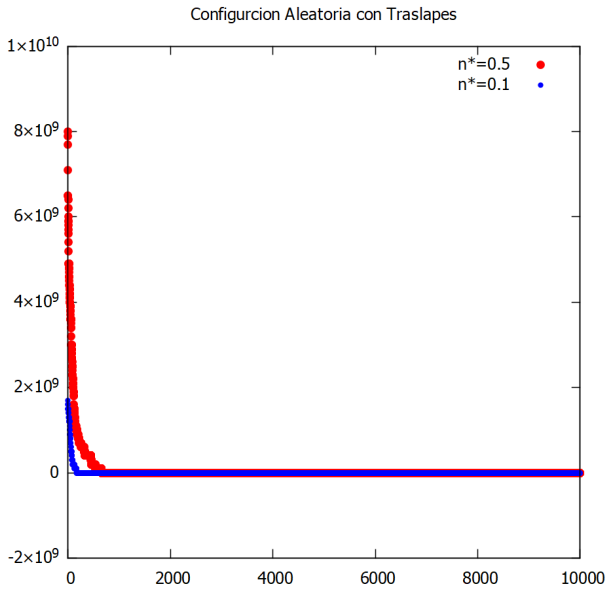


Figura 11: Termalización de simulación Monte Carlo con configuración Inicial Aleatoria con Traslapes para concentraciones de 0.1 y 0.5

Se aprecia que gracias al Monte Carlo la energía disminuye, pero se observo que ambos casos la energía nunca que llego a cero que para el caso de $n^* = 0.5$ la energía por partícula se estabilizó en $-184.320007$ , mientras que para $n^* = 0.1$ se estabilizó en 143.360001.

Esto me permite decir que las configuraciones que inician con traslapes no son tan confiables para este caso de estudió.

# Tarea V: Propiedades Estructurales de un Sistema de Discos Duros

Esta tarea consistió en observar las gráficas de las G(r) y como se altera con los diferente parámetros en la simulación.

## Actividad 8: G(r) bajo diferentes parámetros

Lo primero que se realizó fue realizar una simulación con diferentes configuraciones iniciales. Estas se realizaron para concentraciones de $n^* = 0.5$ y con $30,000$ configuraciones y 100 partículas.

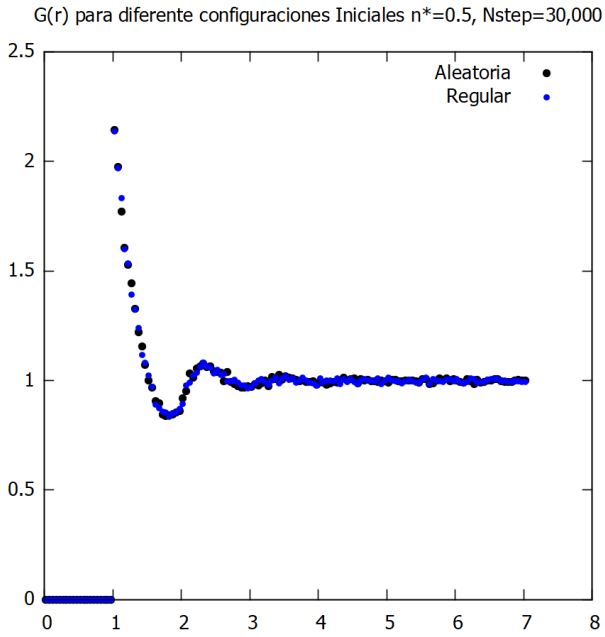


Figura 12: Distribución Radial para diferentes configuraciones iniciales

Luego se realizó variando el numero de partículas bajo las misma condiciones.

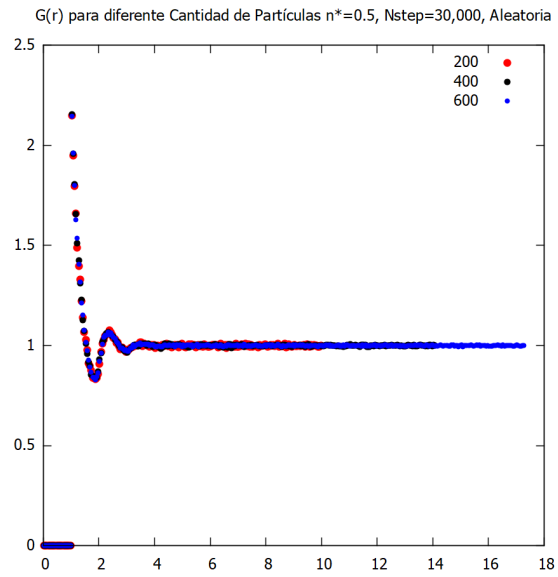


Figura 13: Distribución Radial para diferente numero de partículas

Finalmente se realizó para un deltar diferente y mismas condiciones que al principio.

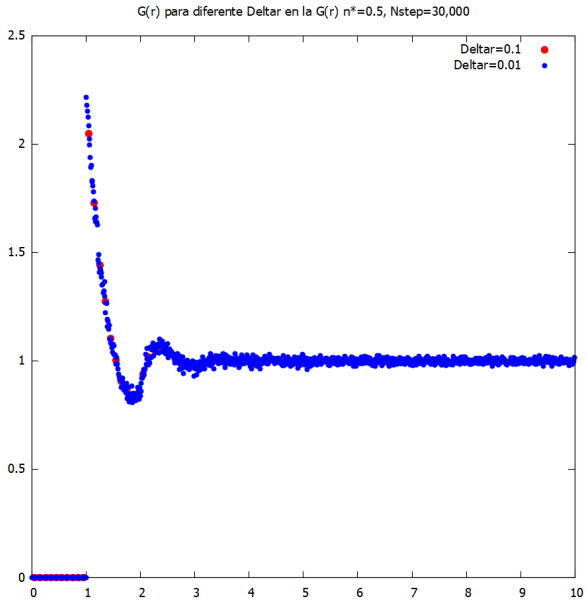


Figura 14: Distribución Radial para deltar diferentes

En figura 12 nos muestra que la configuración aleatoria tiene mejor comportamiento, esto se debe a que la regular no ha olvidado la configuración inicial. La figura 13 se observa que entre más partículas, existen mas datos para realizar el calculo de la distribución radial. La figura 14 nos muestra que un deltar pequeño puede llegar a generar mas ruido y es mas recomendable que no se ni muy pequeño ni muy grande.

## Actividad 9: Densidad local de partículas

La función de distribución nos permite obtener un estimado de cuantas partículas consistió la simulación. Lo que se realizó fue tomar los archivos de la gdr de la actividad anterior para diferente numero de partículas.

Nuestros resultados fueron los siguientes:

| N (Entrada) | N ( g(r) ) |
|---|---|
| 200 | 198.77 |
| 400 | 396.39 |
| 600 | 597.36 |

Tabla 1: Número de partículas.
El que se dio de entrada y el que se calcula.

## Actividad 10: Función de correlación radial g(r) para las concentraciones reducidas

Lo que se busca es observar el comportamiento que tiene la g(r) con la variación de la concentración. Se realizaron corridas para las concentraciones de $n^* = 0.01$, $n^* = 0.1$, $n^* = 0.3$, $n^* = 0.5$, $n^* = 0.7$, $n^* = 0.9$ y $n^* = 0.999$

Los resultados fueron lo siguientes:

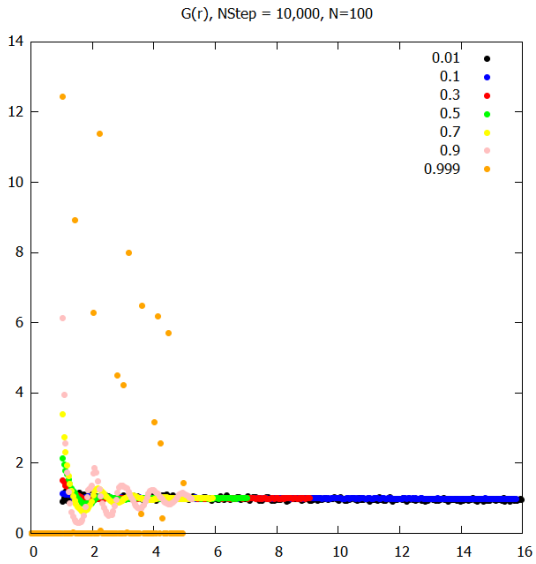


Figura 15: Distribución Radial para diferente concentraciones

Se observa que a bajas concentraciones no alcanza a tomar forma ya que las partículas se encuentran alejadas. Para la concentración de $n^* = 0.999$ no alcanzo a olvidar la configuración inicial regular, lo que explica su forma.

La termalización fue la siguiente:

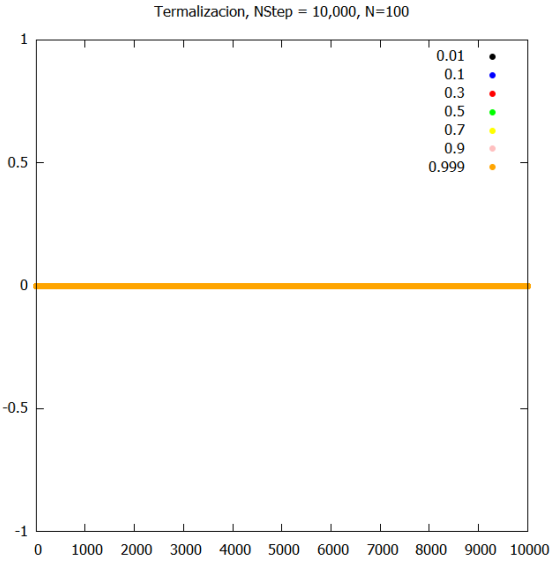


Figura 16: Termalización para diferente concentraciones

Como era de esperarse la energía siempre fue 0.

## Actividad 11: Ecuación de Estado

Lo que se realizó fue calcular la presión del sistema a partir de los archivos de la GDR. Se tomaron los archivos de la actividad anterior y se procedió a calcular la presión.

Los resultados fueron los siguientes

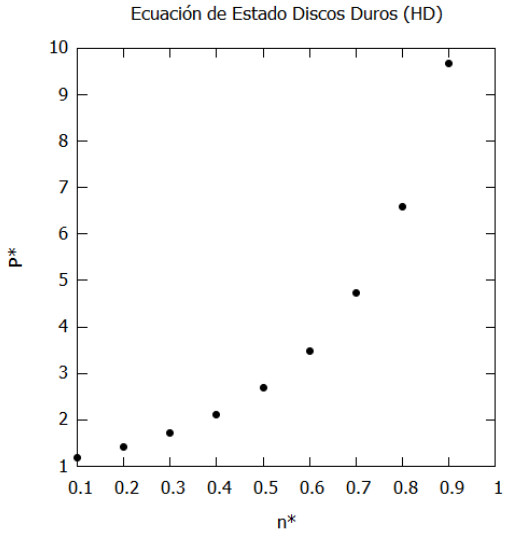


Figura 17: Ecuación de Estado para discos duros (HD)

# Código Utilizado

A continuación se muestra código que se utilizo. Estos no se utilizaron tal cual en todas las actividades, sino que son la base , es decir, este se adapto según lo que se pedía.

```
Module cte
  Implicit None
  Real, Parameter :: sigma = 1.0
  Real, Parameter :: PI=4*atan(1.0)
  Integer, Parameter :: CEq = 1000         !CONFIGURACION DE EQUILIBRIO (SEGUN USER)
  Real ::  BoxL, RCut, dRMax!,  Dens
  Integer :: NN                            !, N, NStep, ISave, IPrint, IRatio,
  Real, Parameter :: Dim = 1.0/2.0         !Dimensiones (2D o 3D)
  Real, Allocatable, Dimension(:) :: X, Y       !POSICIONES
  Real, Allocatable, Dimension(:,:) :: CX, CY   !MATRICES DE LAS CONFIGURACIONES

  ! PARA CORRIDAS SIN INTERACCION DE USER
  Integer, Parameter :: N = 100           !NUMERO DE PARTICULAS
  Integer, Parameter :: NStep = 10000     !NUMERO DE CONFIGURACIONES (PASOS)
  Integer, Parameter :: Iprint = 1000     !IMPRESION EN PANTALLA
  Real, Parameter :: Dens=0.999           !VARIAR SEGUN CORRIDA(CONCENTRACION)
  Integer, Parameter :: ISave = 10        !CUANDO GUARDAR UNA CONFIGURACION
  Integer, Parameter :: IRatio = 10       !CUANDO CAMBIAR EL TAMANO DE PASO
End Module cte
```

Listing 1: Código del Modulo de Variables Globales

```
!========================================================================
! PROGRAMA PRINCIPAL DEL LA SIMULACION, LLAMA SUBRUTINAS PARA REALIZAR
! LA SIMULACION
! Autor: Martin Alejandro Paredes Sosa
!========================================================================
Program Main
  Use cte
  Implicit None
  Integer :: i, j, IStep , k, k2                              !CONTADORES
  Real :: VLRC, VI, V, VOld, VNew, DV, VN                     !ENERGIAS
  Real :: OldX, OldY, OldZ, NewX, NewY, NewZ                  !VALORES TEMP DE POSC
  Real :: RanX, RanY, RanZ, Dummy                             !VALORES ALEATORIOS
  Real :: MAcep, Ratio                                        !VARIABLES DE CONTROL DE
     DRMAX
  Logical :: Ctrl, Ctrl1, Ctrl1A, Ctrl2                       !CONTROL LOGICO
  Integer :: istat1, istat2
  Character (len=80) :: err_msg1, err_msg2
  Character (len=10):: Filename, cons             ! NOMBRE DE ARCHIVO

  !Dens = 0.1
  !CONCE: Do While(Dens .LT. 1.0)

  !PEDIR DENSIDAD Y NUMERO DE PARTICULAS
```

```
23  Write(*,*) "NUMERO DE PARTICULAS"
24  Write(*,*) N
25  Write(*,*) "CONCENTRACION REDUCIDA"
26  Write(*,*) Dens
27  Write(*,*) "NUMERO DE CICLOS"
28  Write(*,*) NStep
29  Write(*,*) "MONITOREO EN PANTALLA (CADA CAUNTOS CICLOS)"
30  Write(*,*) IPrint
31  Write(*,*) "NUMERO DE PASOS PARA GUARDAR CONFIGURACION"
32  Write(*,*) ISave
33  Write(*,*) "FRECUENCIA DE CORRECCION EN DESPLAZAMIENTO"
34  Write(*,*) IRatio
35  Write(*,*) "
     ==========================================================================
     "
36
37  !CONC:Do while (Dens .LT.1.0)
38  !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLO DE POSICION DE PARTICULAS
39  Allocate( X(N), Y(N), STAT= istat1 , ERRMSG=err_msg1  )
40
41
42  !GENERAR LA CONFIGURACION INICIAL
43  Cnfg: If (Dens .LE. 0.65) Then
44     Call ConfigIni
45     Write(*,*) "CONFIGURACION ALEATORIA INICIAL LISTA"
46  Else
47     Call ConfigIniReg
48     Write(*,*) "CONFIGURACION REGULAR INICIAL LISTA"
49  End If Cnfg
50  !CALCULO/PARAMETROS PARA INICIALIZAR
51  RCut = BoxL / 2.0
52  dRMax = 0.1
53  MAcep = 0.0
54  k2 = 0
55  NN = ( NStep- CEq ) / ISave
56
57  !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLOS DE CONFIGURACION
58  Allocate( CX(N,NN), CY(N,NN), STAT= istat2 , ERRMSG=err_msg2 )
59
60
61  !CORRECCION DE LARGO ALCANCE
62  VLRC = 0 !NO SE OCUPA LA CORRECCION POR SER DE CORTO ALCANCE
63
64  !CALCULAR LA ENERGIA DE LA CONFIGURACION
65  Call EnergyConfig(V)
66  VI = V + VLRC
67  Write(*,*) "ENERGIA DE LA CONFIGURACION INICIAL:",  VI
68
69  !ABRIENDO ARCHIVOS PARA GUARDAR INFO DEL SISTEMA
```

```
70  Open(2, File="ConFin.dat")
71
72  Write(Cons,256) Int(100.0*Dens)
73  Filename = "Terma"//trim(Cons)//".dat"
74
75  Open(3, File=Trim(Filename) )
76  !MOVIMIENTO DE PARTICULAS ALEATORIA
77
78
79  Configuracion: Do iStep = 1, NStep
80
81     Particula: Do i = 1, N
82
83        !TOMANDO POSICION DE PARTICULA i
84        OldX = X(i)
85        OldY = Y(i)
86
87        !CALCULAR LA ENERGIA DE LA i-PARTICULA
88        Call EnergyPart(OldX, OldY, i, VOld)
89
90        !GENERAR VALORES ALEATORIOS PARA MOV TENTATIVOS
91        Call Random_Number(RanX)
92        Call Random_Number(RanY)
93
94        !MOVIMIENTO TENTATIVO
95        NewX = OldX + (2.0*RanX - 1.0)*dRMax
96        NewY = OldY + (2.0*RanY - 1.0)*dRMax
97
98        !CONDICIONES PERIODICAS (MANTENER MISMA N EN TODA CONFIGURACION)
99        NewX = NewX - BoxL*Anint(NewX/BoxL)
100       NewY = NewY - BoxL*Anint(NewY/BoxL)
101
102       !CALCULAR LA ENERGIA DE LA PARTICULA EN LA NUEVA POSICION
103       Call EnergyPart(NewX, NewY, i, VNew)
104
105       !MONTECARLO (CRITERIO DE ACEPTACION O RECHAZO DE MOV)
106       DV = VNew - VOld          !CAMBIO DE ENERGIA ENTRE CONFIG
107       Call Random_Number(Dummy) !PARA CRITERIO ENTRE 0.0 Y 75.0
108
109       !MONTECARLO (ACEPTANDO MOVIMIENTOS POR CRITERIOS)
110       MONTECARLO1: If(DV .LT. 75.0 ) Then    !CRITERIO 1 DV MENOR A 75
111
112          MONTECARLO2: If(DV .LE. 0.0 ) Then  !CRITERIO 2 ACEPTANDO MOVIMIENTOS
113
114             V = V + DV
115             X(i) = NewX
116             Y(i) = NewY
117
118             MAcep = MAcep + 1.0 !MOVIMIENTO ACEPTADOS POR MONTECARLO
```

```
119
120           ElseIf( EXP(-DV) .GT. Dummy ) Then  !CRITERIO 3 OTRA FORMA DE ACEPTAR
    MOVIMIENTO
121              V = V + DV
122              X(i) = NewX
123              Y(i) = NewY
124
125              MAcep = MAcep + 1.0 !MOVIMIENTOS ACEPTADOS POR MONTECARLO
126
127           End If MONTECARLO2
128
129        End If MONTECARLO1
130
131        !ENERGIA POR PARTICULA
132        VN = (V+VLRC)/Real(N)
133
134     End Do Particula
135
136     !GUARDANDO LA TERMALIZACION DE CADA CONFIGURACION DEL SISTEMA
137     Write(3,*) IStep, VN
138
139     !AJUSTE DE DESPLAZAMIENTO DRMAX
140     Ctrl1 = Mod(IStep, IRatio) == 0        !CHECA SI TOCA AJUSTAR EL DESPLAZAMIENTO
141     NdR : If (Ctrl1) Then
142
143        Ratio =  MAcep / Real( N * IRatio  )!RAZON DE ACEPTADOS
144
145        LDensG:If (Dens .GT. 0.2) Then
146
147           Ctrl1A = Ratio .GT. 0.5          !CRITERIO DE ACEPTACION GDens
148
149        Else
150
151           Ctrl1A = Ratio .GT. 0.99         !CRITERIO DE ACEPTACION LDens
152
153        End If LDensG
154
155        CriterioDes : If ( Ctrl1A ) Then
156
157           dRMax = dRMax * 1.05             !CRECER DESPLAZAMIENTO
158
159        Else
160
161           dRMax = dRMax * 0.95             !DISMINUIR DESPLAZAMIENTO
162
163        End If CriterioDes
164
165        MAcep = 0.0                         !REINICIAR CONTADOR DE MOV ACEPTADOS
166
```

```
167     End If NdR
168
169     !MONITOREO EN PANTALLA
170     Ctrl = Mod(IStep,IPrint) == 0          !CADA QUE TANTO IMPRIMIR EN PANTALLA
    ENERGIA, DRMAX
171     MonitoreoEne: If(Ctrl) Then
172
173        Write(*,*)  ISTEP, VN, Ratio , dRMax !CONFIG - ENERGIA/PARTICULA - RAZON
    ACEPTADOS - DESPLAZAMIENTO
174
175     End If MonitoreoEne
176
177     !GUARDANDO CONFIGURACION (EN EQUILIBRIO)
178     Ctrl2 = ( Mod(IStep,ISave) == 0 ) .AND. ( IStep .GT. CEq ) !CONTROL DE GUARDADO
179     SAV: If (Ctrl2) Then
180
181        k2 = k2 + 1             !CONFIG GUARDADA (CONTADOR)
182
183        SAV1:Do k = 1 , N       !CORRER LA PARTICULA
184
185           CX(k,k2) = X(k)      !GUARDANDO LA PARTICULA Y SU CONFIG
186           CY(k,k2) = Y(k)
187
188        End Do SAV1
189
190     End If SAV
191
192  End Do Configuracion
193
194  Write(*,*) "DONE ALL CONFIGURATIONS"
195
196  !GUARDAR CONFIG FINAL
197  ConfigFin: Do i=1, N
198
199     Write(2,*) X(i), Y(i)
200
201  End Do ConfigFin
202  Close (2)
203  WRITE(*,*) "DONE SAVING CONFIG FINAL"
204
205  Deallocate( X, Y )
206  WRITE(*,*) "CLEAR MEMORY" !DEBUG
207
208  Call GdrCalc
209  WRITE(*,*) "GDR DONE CALC" !DEBUG
210
211  Deallocate( CX, CY )
212
213  Close(3)
```

```
214   !Dens = Dens + 0.1
215   ! End Do CONCE
216
217   WRITE(*,*) "DONE"
218
219 256 Format (I3.3)
220
221 End Program
```

Listing 2: Código Principal

```
 1 !=============================================================================
 2 ! CONSTRUCCION DE UNA CONFIGURACION INICIAL ALEATORIA EN CELDA BIDIMENSIONAL
 3 ! SIN TRASLAPES
 4 ! Autor: Martin Alejandro Paredes Sosa
 5 !=============================================================================
 6
 7 Subroutine ConfigIni
 8   Use cte
 9   Implicit None
10   Real :: xRan, yRan, xij, yij, 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
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
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
35
36         dist = xij*xij + yij*yij
37
38         DectTraslape: If(dist .LE. sigma ) Then
39
```

```
40            GO TO 2
41
42          End If DectTraslape
43
44       End Do Traslape
45
46       Write(1,*) X(i), Y(i)                !GUARDANDO EN ARCHIVO LA POSICION
47
48    End Do Colocar
49
50    Close(1)
51
52
53 End Subroutine ConfigIni
```

Listing 3: Código para generar la configuración Inicial Aleatoria

```
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, xij, yij,  dist
11   Real :: dBoxl
12   Integer :: i, j, k ,l                          !CONTADOR
13   Integer :: N2, N3
14   Real, Dimension(:),Allocatable :: nX, nY       !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) )
32
33   !GENERANDO COORDENADAS PARA POSICIONES DE LAS PARTICULAS
```

```
  GEN: Do i=1, N2

     nx(i) = (-BoxL)/2.0 + dBoxL/2.0 + dBoxL*(i-1)
     ny(i) = (-BoxL)/2.0 + dBoxL/2.0 + dBoxL*(i-1)

  End Do GEN

  !ESCRIBIENDO EN ARCHIVO
  Open (1, File = "ConIni.dat" )
  l = 0
  EscribirX: Do i = 1, N2

     EscribirY: Do j = 1, N2

        l = l + 1
        X(l) = nX(i)
        Y(l) = nY(j)

     End Do EscribirY

  End Do EscribirX

  !Write(*,*) l !DEBUG

  Do i=1, N3
     Write(1,*) X(i), Y(i)
  End Do


  Deallocate(nX, nY)

  Close(1)


End Subroutine ConfigIniReg
```

Listing 4: Código para generar la configuración Inicial Regular

```
!=============================================================================
! CALCULO DE LA ENERGIA DE UNA DE LA CONFIGURACION DE LA CELDA
!
! Autor: Martin Alejandro Paredes Sosa
!=============================================================================

Subroutine EnergyConfig(V)
  Use cte
  Implicit None
  Real ::  V, Rx1, Rxd, Ry1, Ryd, Dist, VNew
  Integer :: i, j
  V = 0
```

```
13   IterPart: Do i=1, N-1
14
15      Rx1 = X(i)
16      Ry1 = Y(i)
17
18      IterPart2: Do j = i+1, N
19         Rxd = Rx1 - X(j)
20         Ryd = Ry1 - Y(j)
21
22         !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
23         Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
24         Ryd = Ryd - BoxL*Anint(Ryd/BoxL)
25
26         !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
27         Dist = sqrt( Rxd*Rxd + Ryd*Ryd  )
28
29         ChecarInter: If(Dist .LT. RCut)  Then
30
31            ChecarCercania: If (Dist .LE. 1.0) Then
32               VNew = 1.0E+10
33            Else
34               VNew = 0
35            End If ChecarCercania
36
37            V = V + VNew
38         End If ChecarInter
39
40      End Do IterPart2
41   End Do IterPart
42
43 End Subroutine EnergyConfig
```

Listing 5: Código para calculo de Energía de la Configuración de HD

```
1  !=========================================================================
2  ! CALCULO DE LA ENERGIA DE UNA DE LAS PARTICULAS DE LA CELDA
3  !
4  ! Autor: Martin Alejandro Paredes Sosa
5  !=========================================================================
6
7  Subroutine EnergyPart(Rx1, Ry1, i, V)
8    Use cte
9    Implicit None
10   Real :: V, VNew, Dist, Rx1, Rxd, Ry1, Ryd
11   Integer :: i, j
12   !INICIAR ENERGIA EN 0
13   V = 0
14
15   BuscarPart: Do j=1, N
16
```

```
17      NoLaMisma: If(i .NE. j) Then
18
19         Rxd = Rx1 - X(j)
20         Ryd = Ry1 - Y(j)
21
22         !CONDICION DE IMAGEN MINIMA (LOCALIZAR PARTICULAS EN CELDAS CERCANAS)
23         Rxd = Rxd - BoxL*Anint(Rxd/BoxL)
24         Ryd = Ryd - BoxL*Anint(Ryd/BoxL)
25
26         !INGRESANDO MODELO DE INTERACCON (DISCOS DUROS)
27         Dist = sqrt( Rxd*Rxd + Ryd*Ryd  )
28         !If(Dist .LE. 1.0) Write(*,*) Dist, i,j
29
30         ChecarInter: If(Dist .LT. RCut)  Then
31
32            ChecarCercania: If (Dist .LE. 1.0) Then
33               VNew = 1.0E+10
34               !Write(*,*) "Ohh"
35            Else
36               VNew = 0
37            End If ChecarCercania
38
39            V = V + VNew
40         End If ChecarInter
41
42
43      End If NoLaMisma
44
45   End Do BuscarPart
46
47 End Subroutine EnergyPart
```

Listing 6: Código para calculo de Energía por partícula de HD

```
1  !=========================================================================
2  ! EL PROGRAMA REALIZA EL CALCULO DE LA GDR APARTIR DE LAS DIFERENTES
3  ! CONFIGURACIONES REALIZADAS EN EL PROGRAMA PRINCIPAL
4  !
5  ! AUTOR: Martin Alejandro Paredes Sosa
6  !=========================================================================
7
8  Subroutine GdrCalc
9
10   Use cte
11   Implicit None
12
13   Integer, Allocatable, Dimension(:) :: Histo
14   Real, Parameter :: delTar = 0.05
15   Integer :: MBin, iBin
16   Integer :: i, j, k                                          !CONTADORES
```

```
17  Real :: xO, yO, xN, yN, xON, yON
18  Real :: rD, rU, rL, rM, c1, c2, gdr, gdrm, press
19  Integer :: istat1
20  Character (len=80) :: err_msg1
21  logical :: Ctrl1, Ctrl2
22  Character (len=12):: Filename, cons                              ! NOMBRE DE ARCHIVO
23
24  MBin = Int( RCut / delTar ) ! CINTA MAXIMA
25
26  Allocate( Histo(MBin+1) , STAT = istat1, ERRMSG = err_msg1)
27  Histo = 0 ! ESTABLECER TODO EL ARREGLO EN 0
28
29  PartiO : Do i = 1, N
30     NextParti : Do j = 1, N
31        NOTSAME : If (i /= j ) Then
32           StepCnfg : Do k = 1, NN
33
34              !PARTICULA i ORIGEN
35              xO = CX( i , k )
36              yO = CY( i , k )
37
38              !PARTICUAL j CERCANA
39              xN = CX( j , k )
40              yN = CY( j , k )
41
42              !DISTANCIA
43              xON = xN - xO
44              yON = yN - yO
45
46              !CONDICION DE IMAGEN MINIMA
47              xON = xON - BoxL*Anint( xON/BoxL )
48              yON = yON - BoxL*Anint( yON/BoxL )
49              rD = sqrt( (xON * xON) + (yON * yON) )
50              If (rd .LE. 1.0 ) write(100,*) rD, i, j , k
51
52              !CERCANIA CINTA
53              iBin =  Int( rD / delTar ) + 1
54
55              Guardar : If((iBin .LE. MBin)  ) Then
56
57                 Histo(iBin) = Histo(iBin) + 1 !ACUMULANDO  PARTICULAS EN CINTAS
58
59              End If Guardar
60
61           End Do StepCnfg
62
63        End If NOTSAME
64     End Do NextParti
65  End Do PartiO
```

```
66
67
68    c1 = PI * Dens
69
70    !ABRIENDO ARCHIVO PARA GDR
71    Write(Cons,256) Int(100.0 * Dens)
72       Filename = "gdr"//trim(Cons)//".dat"
73    Open( 5, file= Filename )
74
75    GdrCal: Do ibin = 1 , MBin
76
77       rL = Real(iBin - 1) * delTar       !CINTA INFERIOR
78
79       rU = rL + delTar                   !CINTA SUPERIOR
80       rM = rL + ( delTar/2.0 )           !CINTA INTERMEDIA
81
82       c2 = c1 * ((rU*rU) - (rL*rL))      !PRECALCULO G(r)
83       gdr = Real( Histo(iBin) )/ Real(NN) / Real(N) / c2 !CALCULANDO G(r)
84       Write(5,*) rM , gdr
85
86    End Do GdrCal
87
88    Close(5)
89
90    Deallocate( Histo )
91
92    Write(*,*) "GDR DONE, SAVE"
93
94 256 Format (I2.2)
95
96 End Subroutine GdrCalc
```

Listing 7: Código para el calculo de la G(r)

```
1  !=========================================================================
2  ! CALCULO DE LA PRESION PARA EL CASO DE DE DISCOS DUROS CON LOS ARCHIVOS DE
3  ! DE LA G(r) OBTENIDOS DE LA SIMULACION
4  !
5  ! AUTOR : MARTIN ALEJANDRO PAREDES SOSA
6  !=========================================================================
7
8  !DECLARACION DE VARIABLES
9  Program Calc
10   !Use Basic
11   Implicit None
12   Integer :: DENS                                  ! PARA NOMBRE DE ARCHIVO
13   Integer :: State                                 ! ESTADO DE LECTURA
14   Integer :: k, i, j                               ! CONTADOR
15   Character (len=3), Parameter :: Start = "gdr"    ! NOMBRE DE ARCHIVO DE ENTRADA
16   Character (len=4), Parameter :: En = ".dat"      ! EXTENSION ARCHIVO DE ENTRADA
```

```
17  Character (len=12):: Filename, cons              ! NOMBRE DE ARCHIVO
18  Real, Parameter :: PI = 4.0 * ATAN(1.0)          ! VALOR DE PI
19  Real :: Des                                      ! CONCENTRACION
20  Real, Dimension(:),Allocatable ::  R , G         ! RADIO | DISTRIBUCION RADIAL
21  Real :: gr1                                      ! PARAMETROS PARA CALCULO DE a Y b
      VAN DER WAALS
22  Real :: Press                                    ! ACUMULADOR PARA INTEGRACION
23
24  Write(*,*) "=========================="
25  !Open(8, File = "a_starT1.dat", Action= "write")
26  Open(9, File = "Pres.dat", Action= "write") !ARCHIVO DE SALIDA
27
28  Archivo:  Do Dens = 1, 9
29
30     !TAMANO DEL ARCHIVO POR LEER
31     Write(Cons,256) Dens*10             ! SELECCION DEL ARCHIVO PARA LEER
32     Filename = start//trim(Cons)//En    ! VARIAR SEGUN  NOMBRO EL ARCHIVO; Start Y
    En TAMBIEN CAMBIAR
33     Write(*,*) "Archivo: ",Filename     ! CHECAR QUE ARCHIVO VA A LEER (SI NO ES EL
    MISMO NOMBRE FALLA LA CORRIDA)
34
35     Open( 1, File = Trim(Filename), action= "read", Status ="old" ) !ARCHIVO DE
    ENTRADA
36
37     k = 0  !REINICIA CONTADOR PARA NUEVO ARCHIVO (RECUENTO DE FILAS)
38
39     Sizes: Do                          !BUSCANDO TAMANO DE ARCHIVO (RENGLONES QUE
    QUE TIENE)
40
41        Read( 1,*, iostat = state  )
42        k = k + 1
43        If ( state .LT. 0 ) Exit        !CONDICION DE SALIDA (YA NO HAY MAS REGLONES
    EN EL ARCHIVO)
44
45     End Do Sizes
46
47     Write(*,*) "Tiene", k, "Renglones" !DEBUG LINE (SIZE OF FILE)
48
49     Rewind 1 !REINICIAR ARCHIVO DE ENTRADA
50     Allocate ( R(k), G(k) ) !ALOJAR ESPACIO EN MEMORIA
51
52     !SAVING FILE DATA
53     Saves : Do i = 1, k+1
54
55        Read( 1,*, iostat = state  ) R(i), G(i)   ! CONSIDERAR EL NUMERO DE COLUMNAS
56        If ( state .LT. 0 ) Exit                  ! CONDICION DE SALIDA (YA NO HAY
    MAS REGLONES EN EL ARCHIVO) POR SEGURIDAD
57
58     End Do Saves
```

```
59
60      Write(*,*) "DATOS GUARDADOS EN MEMORIA"
61
62      !CALC DE PRESION HD
63
64      Locate: Do i = 1, k            ! BUSCANDO EL PRIMER DATO .GE. 0
65
66         If (G(i) .GT. 0.0) Exit
67
68      End Do Locate
69
70      gr1 = G(i)                     ! Ghd(1+)
71      Des = Real(Dens)*0.1           ! CONCENTRACION A PARTIR DEL NOMBRE DEL ARCHIVO
72      Press = 1.0 + 0.5*Pi*Des*gr1   ! CALCULO PRESION HD
73
74      Write(9,*) Des , press         ! ESCRITURA EN ARCHIVO DE SALIDA
75
76      Deallocate(R,G)                ! LIBERANDO MEMORIA PARA SIGUIENTE ARCHIVO O
    FINAL
77      Write(*,*) "=========================="
78    End Do Archivo
79
80  512 Format (I5.5)
81  256 Format (I2.2)
82  End Program Calc
```

Listing 8: Código del calculo de presión para HD