Comentarios, Desarrollos u Observaciones

Desarrollo Experimental II

Docente: Dra. Laura Lorenia Yeomans Reyna

Portafolio III:
Simulación de Dinámica Browniana

Martín Alejandro Paredes Sosa

Semestre: 2018-1

Tarea VI: Resultados de la implementación de simulación de dinámica browniana

El sistema que se estudió fue el de un sistema de partículas coloidales cargadas de forma que el potencial de interacción entre ellas es tipo Yukawa. Como base se utilizaron los parametros de Gaylor.

Lo que se buscó fue conocer la propiedades estructurales del sistema, así como la propiedades que varian con el tiempo como lo son el desplazamiento cuadratico medio y el coeficiente de autodifusión.

Los resultados fueron los siguintes:

Termalización

A diferencia de Discos o Esferas Duras (HD o HS), la curva de termalización del potencial Yukawa no es siempre cero. La energía empieza muy alta, pero conforme avanzan las configuraciones esta disminuyte rapidamente hasta alcanzar un punto de equilibrio, como se muestra en la figura 1.

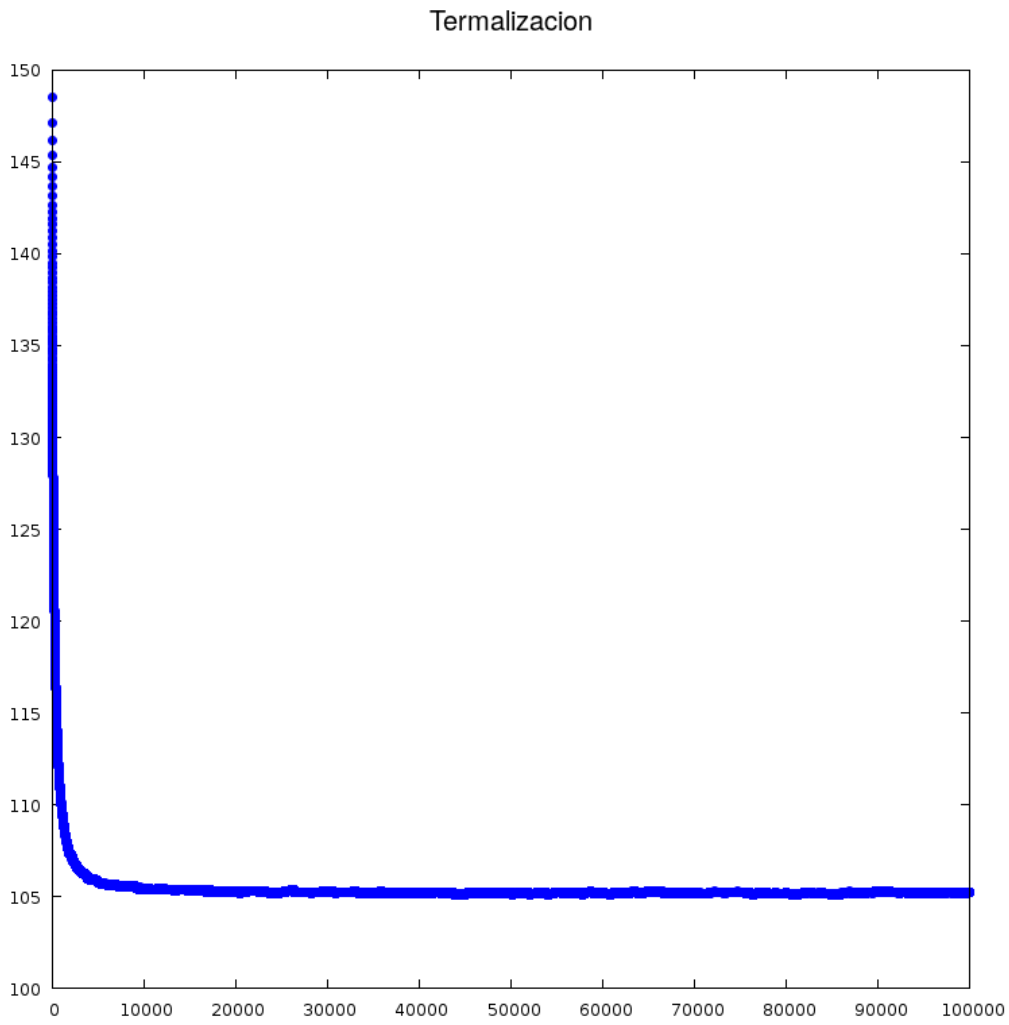

Figura 1: Curva de Termalizacion. Se utilizó un Potencial Yukawa

A partir de esta curva podemos decir cuando el sistema se encuntras en equlibrio, lo que nos permite decidir en que punto comenzar a guradar configuraciones.

Configuraciones Inicial y Final

Cuando se empieza, se genera una configuracion inicial aleatoria sin traslapes, el sistema conto con 800 particulas y una fración en volomen de 4.4×10^{-4} que aproxima a una concentracion de $n^* = 8.40338122 \times 10^{-4}$. Este ultimo es parametro de Gaylor. La celda era una caja cuadrada de lado $l = 98.3736267$

Las configuraciones inicial y final fueron:

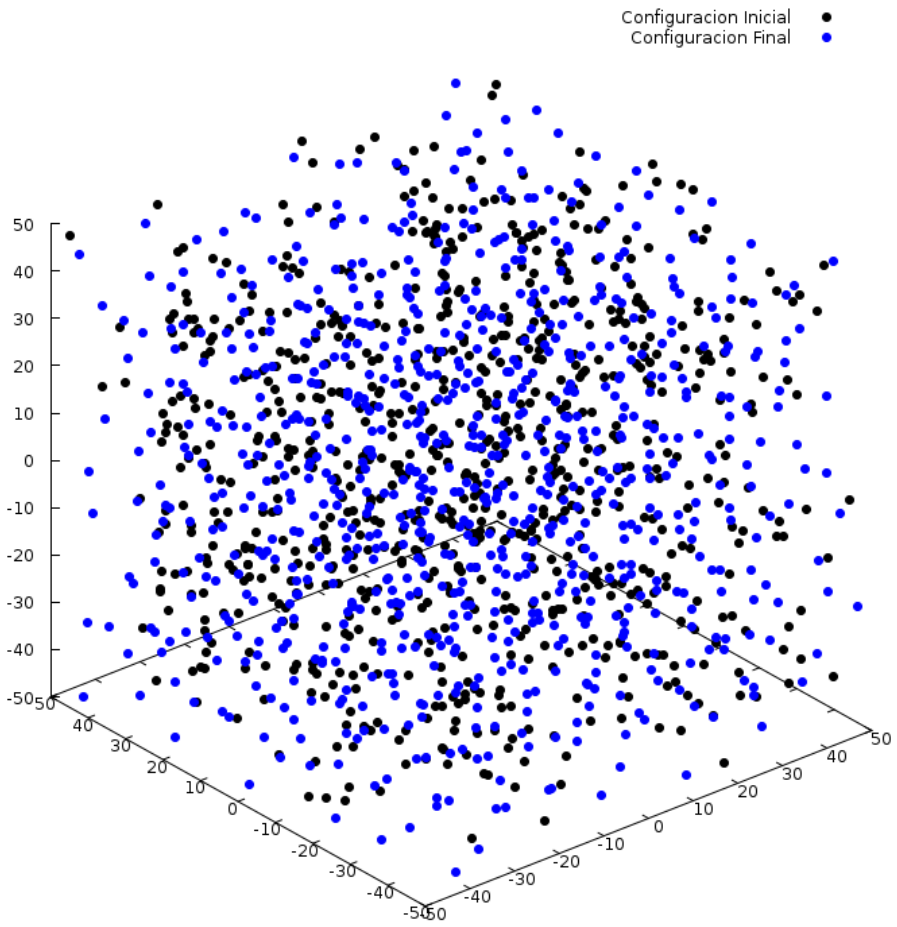

Figura 2: Configuración Inicial (Negro) y configuración Final (Azul)

Función de distribución radial G(r)

La simulación contó con $100,000$ configuraciones y guardo cada 100 a partir de la configuración $20,000$ que fue caundo se considero que ya alcanzó el equlibrio. Se observa en la figura 3 que aproximadamente

cuando se alcanza $r \approx 7.5$ se empiezan a encontrar las partículas. Para cuando $r \approx 10.0$ este alcanza su máximo. En la figura 4 se compara la G(r) de Gabriela con la propia. Se observa que ambas se parecen, solo que la de gabriela presenta mas ruido. Esto se puede deber a que se uso un tamaño de cinta mas fino.

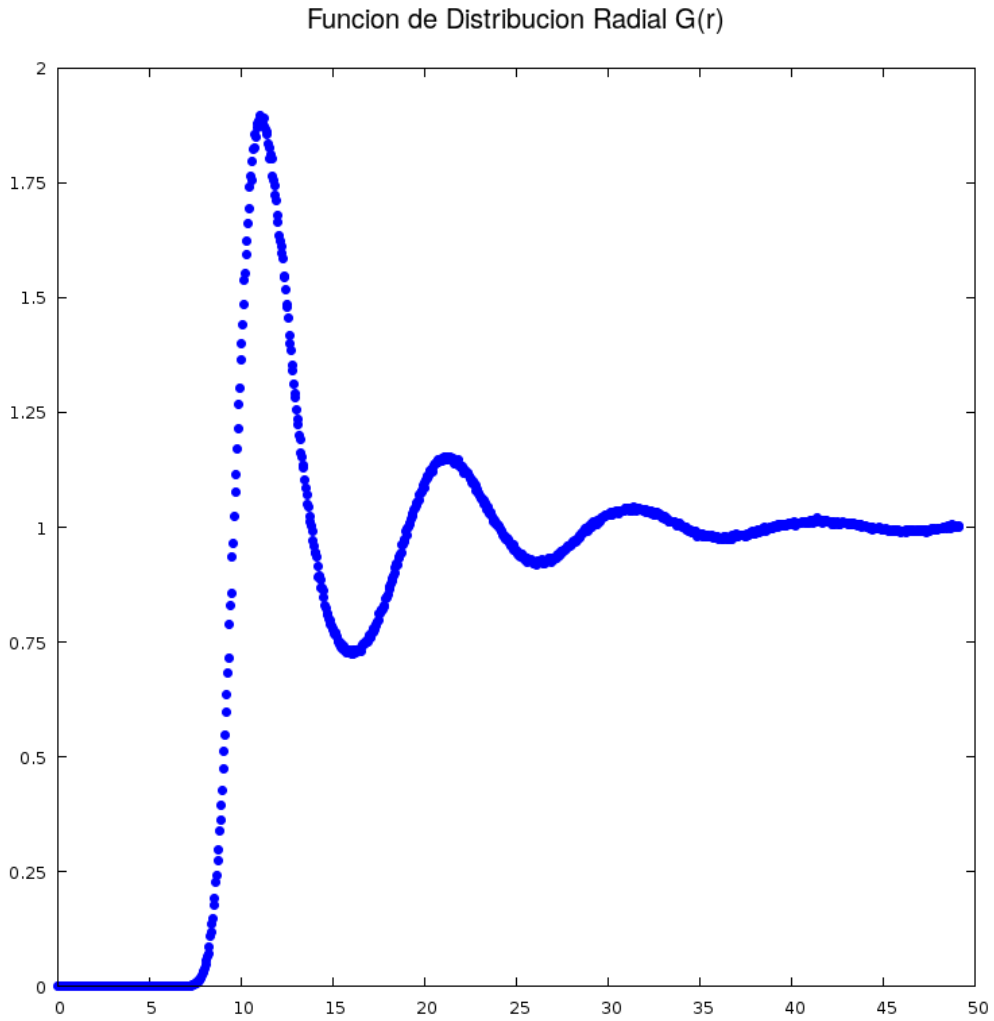

Figura 3: Funcion de Distribucion Radial de 800 particulas y $n^* = 8.4033 \times 10^{-4}$

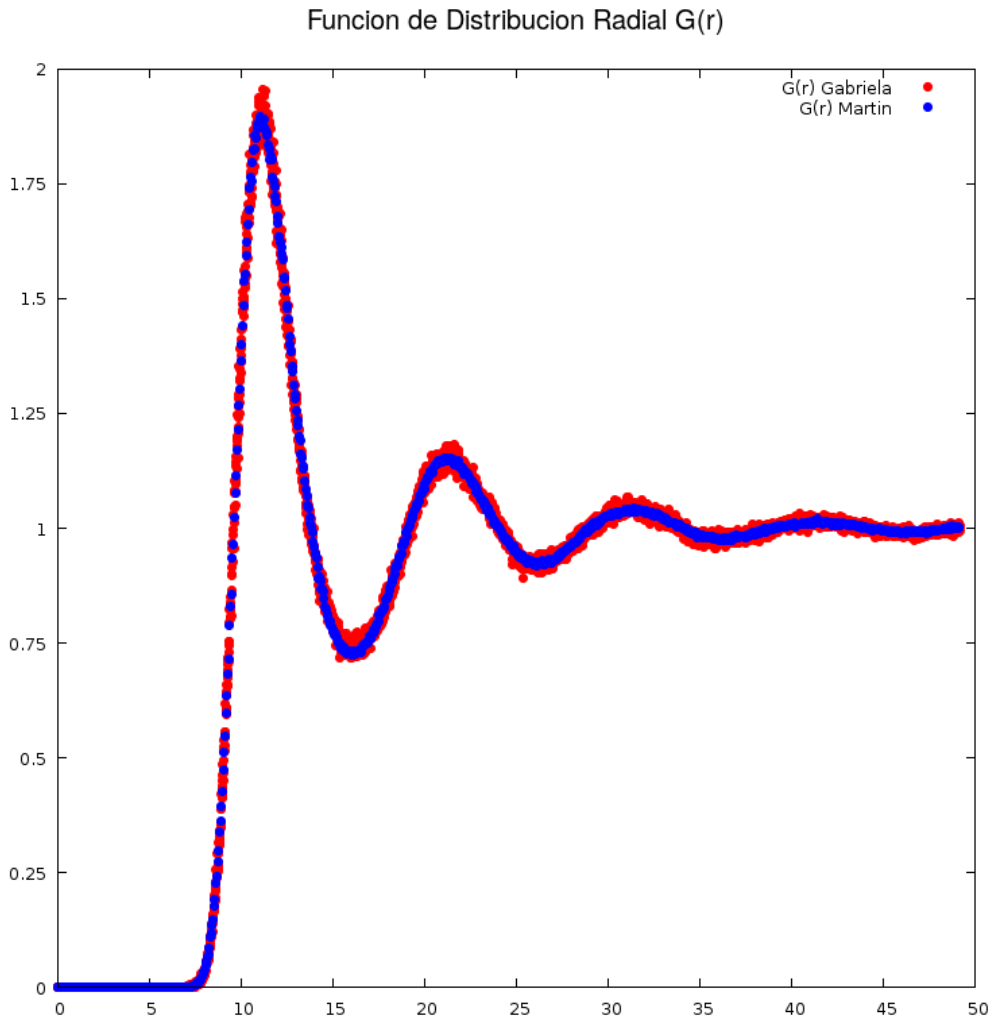

Figura 4: Funcion de Distribucion Radial de 800 particulas y $n^* = 8.4033 \times 10^{-4}$, Comparando con Gabriela

Presión

En este caso se pudo monitorear la presión del sistema conforme este evoluciono. Se obtuvo lo siguiente:

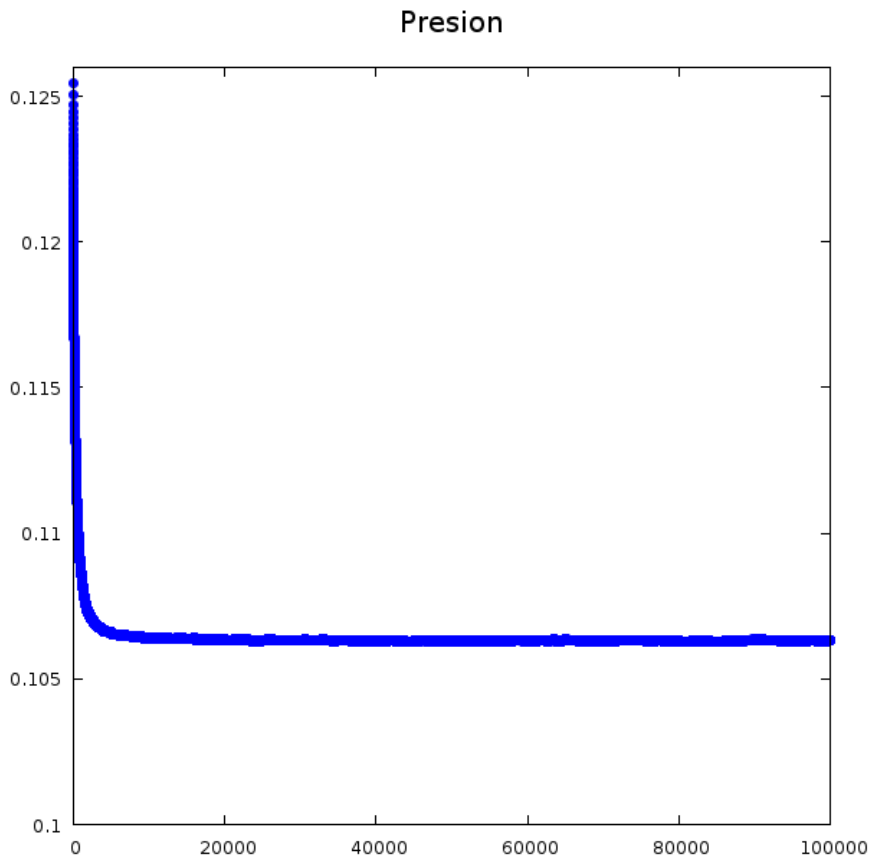

Figura 5: Presión del sistema conforme este evoluciona

Código

El codigo que se utilizo el fue le siguiente

```
Module cte
  Implicit None
  Real, Parameter :: sigma = 1.0                                    !DIAMETRO DE LAS PARTICULAS
  Real, Parameter :: PI=4.0*atan(1.0)                               !PI
  Real :: YukA, YukZk                                               !POTENCIAL YUKAWA
  Integer, Parameter :: CEq = 20000                                 !CONFIGURACION DE
    EQUILIBRIO
  Real :: Dens, BoxL, RCut, dT                                      !PARAMETROS DE LA
    SIMULACION (REALES)
  Integer :: N, NStep, ISave, ISave2, IPrint, IRatio, NN            !PARAMETROS DE LA
    SIMULACION (ENTEROS)
  Real, Parameter :: Dim = 1.0/3.0                                  !Dimensiones (2D o 3D)
  Real, Allocatable, Dimension(:) :: X, Y, Z, XR, YR, ZR            !POSICIONES DE LAS
    PARTICULAS X Y Z
  Real, Allocatable, Dimension(:,:) :: CX, CY, CZ                   !MATRICES DE CONFIGURACION
    GR
  Real, Allocatable, Dimension(:,:) :: CXD, CYD, CZD                !MATRICES DE CONFIGURACION
    WDT
```

```
  Real, Allocatable, Dimension(:) :: FX, FY, FZ                    !FUERZAS DE INTERACCION
End Module cte
```

Listing 1: Modulo de Variables Globales

```
!=====================================================================
! PROGRAMA PRINCIPAL DEL LA SIMULACION, LLAMA SUBRUTINAS PARA REALIZAR
! LA SIMULACION. SE UTILIZA DINAMICA BROWNIANA
! AUTOR: MARTIN ALEJANDRO PAREDES SOSA
!=====================================================================
Program Main
  Use cte
  Implicit None
  Integer :: i, j, IStep , k, k2,  ki                          !CONTADORES
  Real :: RanX, RanY, RanZ                                     !VALORES ALEATORIOS PARA POSICION
  Real :: phi                                                  !VARIABLE TEMP FRACCION EN VOLUMEN
  Real :: Var
  Logical :: Ctrl, Ctrl1, Ctrl2                                !CONTROL LOGICO

  Integer :: istat1, istat2
  Character (len=80) :: err_msg1, err_msg2


  !PARAMETROS DE SIMULADOR
  N = 800
  NStep =100000
  ISave = 100                                                 !G(r)
  ISave2 = 100                                                !W(t), D(t)
  iPrint = 1000
  dt = 0.0004

  phi = 4.4D-4
  Dens = 6*phi/pi
  YukA = 556.0
  YukZk = 0.149
  YukA = YukA * Exp( YukZk )
  NN = ( NStep- CEq ) / ISave
  !Write(*,*) phi !DEBUG

  !PEDIR DENSIDAD Y NUMERO DE PARTICULAS
  Write(*,*) "NUMERO DE PARTICULAS"
  Write(*,*) N
  Write(*,*) "CONCENTRACION REDUCIDA"
  Write(*,*) Dens
  Write(*,*) "NUMERO DE CICLOS"
  Write(*,*) NStep
  Write(*,*) "MONITOREO EN PANTALLA (CADA CAUNTOS CICLOS)"
  write(*,*) IPrint
  Write(*,*) "NUMERO DE PASOS PARA GUARDAR CONFIGURACION ( G(r) )"
  write(*,*) ISave
```

```
46  Write(*,*) "NUMERO DE PASOS PARA GUARDAR CONFIGURACION ( W(t),D(t)  )"
47  write(*,*) ISave2
48  Write(*,*) "PASO DE TIEMPO"
49  write(*,*) DT
50  Write(*,*) "
      ==========================================================================
      "
51
52  !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLO DE POSICION DE PARTICULAS
53  Allocate( X(N), Y(N), Z(N), STAT= istat1 , ERRMSG=err_msg1  )
54  Allocate( XR(N), YR(N), ZR(N) )
55
56  !ALOJAR ESPACIO EN MEMORIA PARA LAS FUERZAS DE INTERACCION
57  Allocate( FX(N), FY(N), FZ(N) )
58
59  !ALOJAR ESPACIO EN MEOMORIA PARA LOS ARREGLOS DE CONFIGURACION
60  Allocate( CX(N,NN), CY(N,NN), CZ(N,NN), STAT= istat2 , ERRMSG=err_msg2 )
61  Allocate( CXD(N,NN), CYD(N,NN), CZD(N,NN) )
62
63
64  !GENERAR LA CONFIGURACION INICIAL
65  Call ConfigIni
66  Write(*,*) "CONFIGURACION INICIAL LISTA"
67
68  !CALCULO/PARAMETROS PARA INICIALIZAR
69  RCut = BoxL / 2.0
70  Var = sqrt(2.0*dt)
71  k2 = 0                                                          !G(r)
72  ki = 0                                                          !W(t), D(t)
73
74
75  !CALCULO DE FUERZAS DE LA CONFIGURACION INICIAL
76  Open(3, File="Terma.dat" )
77  Call Fuerza(iStep)
78
79
80  !ABRIENDO ARCHIVOS PARA GUARDAR INFO DEL SISTEMA
81  Open(2, File="ConFin.dat")
82
83
84  !MOVIMIENTO DE PARTICULAS
85
86
87  Configuracion: Do iStep = 1, NStep
88
89     Particula: Do i = 1, N
90
91        !GENERAR VALORES ALEATORIO CON DISTRIBUCION GAUSSIANA
92        Call RanGauss(RanX)
```

```
93          Call RanGauss(RanY)
94          Call RanGauss(RanZ)
95
96          !MOVIMINETO DE PARTICULAS EN BASE A LA ECUACION DE LANGEVIN
97          !SOBREAMORTIGUADA, O REGIMEN DIFUSIVO
98          !ALGORITMO DE EMARK PARA EL DESPLAZAMIENTO
99
100         X(i) = X(i) + FX(i) * dt + Var * RanX
101         Y(i) = Y(i) + FY(i) * dt + Var * RanY
102         Z(i) = Z(i) + FZ(i) * dt + Var * RanZ
103
104         XR(i) = XR(i) + FX(i) * dt + Var * RanX
105         YR(i) = YR(i) + FY(i) * dt + Var * RanY
106         ZR(i) = ZR(i) + FZ(i) * dt + Var * RanZ
107
108         !CONDICIONES PERIODICAS (MANTENER MISMA N EN TODA CONFIGURACION)
109         X(i) = X(i) - BoxL*Anint( X(i) / BoxL )
110         Y(i) = Y(i) - BoxL*Anint( Y(i) / BoxL )
111         Z(i) = Z(i) - BoxL*Anint( Z(i) / BoxL )
112
113         !FIN DE MOVIMIENTO DE PARTICULAS DE LA CONFIGURACION ISTEP
114
115     End Do Particula
116
117     !ALMACENANDO CONFIGURACIONES DE EQUILIBRIO CX, CY, CZ PARA LA G(R)
118     Ctrl1 =  Mod( iStep, iSave) == 0
119     Ctrl2 = iStep .GT. CEq
120     Ctrl = Ctrl1 .AND. Ctrl2
121
122     ConfigGR: If (Ctrl) Then
123        k2 = k2 + 1
124
125        SAV:Do k = 1 , N
126
127           CX(k,k2) = X(k)
128           CY(k,k2) = Y(k)
129           CZ(k,k2) = Z(k)
130           !Write(*,*) "ENTRO"
131
132        End Do SAV
133
134     End If ConfigGR
135
136     !ALMACENANDO CONFIGURACIONES DE EQUILIBRIO CXD, CYD, CZD PARA W(t) Y D(t)
137     Ctrl1 =  Mod( iStep, iSave2) == 0
138     Ctrl2 = iStep .GT. CEq
139     Ctrl = Ctrl1 .AND. Ctrl2
140
141     ConfigWt: If (Ctrl) Then
```

```
142
143         ki = ki + 1
144
145         SAV1:Do k = 1 , N
146
147            CXD(k,ki) = XR(k)
148            CYD(k,ki) = YR(k)
149            CZD(k,ki) = ZR(k)
150
151         End Do SAV1
152
153      End If ConfigWt
154
155      !CALCULO DE FUERZAS DE LAS CONFIGURACIONES
156      Call Fuerza(iStep)
157
158   End Do Configuracion
159
160   Write(*,*) "DONE CONFIGURACIONES"
161   Close(3)
162
163   !GUARDAR CONFIG FINAL
164   ConfigFin: Do i=1, N
165
166      Write(2,*) X(i), Y(i), Z(i)
167
168   End Do ConfigFin
169   Close (2)
170   WRITE(*,*) "DONE SAVING CONFIG FINAL"
171   Deallocate( X, Y, Z, XR, YR, ZR )
172   WRITE(*,*) "CLEAR MEMORY POSICIONES" !DEBUG
173
174   !CALCULO DE PROPIEDADES
175   WRITE(*,*) "CALCULANDO GDR" !DEBUG
176   Call GdrCalc
177   WRITE(*,*) "GDR DONE CALC" !DEBUG
178   Deallocate( CX, CY, CZ )
179   WRITE(*,*) "CLEAR MEMORY CONFIGURACIONES G(r)" !DEBUG
180
181   !CALCULO DE PROPIEDADES DE WDT
182   Write(*,*) "CALCULADO WDT" !DEBUG
183   Call WDT(ki)
184   WRITE(*,*) "WDT DONE CALC" !DEBUG
185   Deallocate( CXD, CYD, CZD )
186   WRITE(*,*) "CLEAR MEMORY CONFIGURACIONES W(t)" !DEBUG
187
188
189   WRITE(*,*) "DONE"
190
```

```
191
192 End Program Main
```

Listing 2: Código Principal

```
1  !================================================================================
2  ! CONSTRUCCION DE UNA CONFIGURACION INICIAL ALEATORIA EN CELDA TRIDIMENSIONAL
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
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      !COLOCAR DENTRO DE LA CELDA
24
25      X(i) = (xRan-0.5)*(BoxL-1)                  !\
26      Y(i) = (yRan-0.5)*(BoxL-1)                  !|   [-(BoxL-1)/2 , (BoxL-1)/2]
27      Z(i) = (zRan-0.5)*(BoxL-1)                  !/
28
29      !Write(*,*) X(i), Y(i), Z(i)               !DEBUG
30
31      Traslape: Do j=1 , i-1
32
33         xij = X(i) - X(j)              !CALCULANDO LA DISTANCIA ENTRE PARTICULAS
34         yij = Y(i) - Y(j)
35         zij = Z(i) - Z(j)
36
37         dist = xij*xij + yij*yij + zij*zij
38
39         DectTraslape: If(dist .LE. sigma ) Then        !DETECTANDO TRASLAPES
40
41            !Write(*,*) "TRASLAPE", i, j    !DEBUG
42            GO TO 2
43
44         End If DectTraslape
45
```

```
46      End Do Traslape
47
48      Write(1,*) X(i), Y(i), Z(i) !GUARDANDO EN ARCHIVO LA POSICION
49
50    End Do Colocar
51
52    Close(1)
53
54
55  End Subroutine ConfigIni
```

Listing 3: Código para generar la configuración Inicial Aleatoria

```
1  !=====================================================================
2  ! SUBRUTINA PARA EL CALCULO DE FUERZAS DE INTERACCION Y ENERGIA DE LA
3  ! CONFIGURACION PARA LA SIMULACION DE DINAMICA BROWNIANA
4  !
5  ! AUTOR : MARTIN ALEJANDRO PAREDES SOSA
6  !=====================================================================
7
8  Subroutine Fuerza(L)
9    Use cte
10   Implicit None
11   Real :: EnePot, U, U2, U3                                    !ENERGIA
12   Real :: FXI, FYI, FZI, fxij, fyij, fzij                      !FUERZAS TEMP
13   Real :: xij, yij, zij, rij                                   !POSICIONES
14   Real :: Pres, Pres1
15
16   Integer :: i, j, L                                           !CONTADORES ("L" CONTADOR
       DE LA CONFIGURACION)
17   Logical :: Ctrl1, Ctrl2
18
19   !INICIALIZANDO
20   EnePot = 0.0
21   FX = 0.0
22   FY = 0.0
23   FZ = 0.0
24
25   Pres1 = 0.0
26
27   Parti1: Do i = 1, N - 1
28
29      FXI = FX(i)
30      FYI = FY(i)
31      FZI = FZ(i)
32
33      Parti2: Do j = i + 1, N
34
35         !SEPARACION
36         xij = X(i) - X(j)
```

```
37          yij = Y(i) - Y(j)
38          zij = Z(i) - Z(j)
39
40          !CONDICION DE IMAGEN MINIMA
41          xij = xij - BoxL * Anint( xij / BoxL )
42          yij = yij - BoxL * Anint( yij / BoxL )
43          zij = zij - BoxL * Anint( zij / BoxL )
44
45          !DISTANCIA
46          rij = sqrt( xij*xij + yij*yij + zij*zij )
47
48          !TRASLAPES
49          Ctrl1 = rij .LE. 1.0
50          Traslape : If(Ctrl1) Then
51
52             Write(*,*) "TRASLAPE", i, j
53
54          End If Traslape
55
56          !IMPLEMENTACION DEL POTENCIAL
57          Ctrl2 = rij .LT. RCut
58          Potencial: If(Ctrl2) Then
59
60             U = Exp( -YukZk * rij )
61             U2 = YukA * U  * (YukZk * rij + 1.0 ) / (rij**3)
62             U3 = U2 * rij * rij
63             EnePot = (YukA * U) / rij + EnePot
64
65             fxij = xij * U2
66             fyij = yij * U2
67             fzij = zij * U2
68
69             FXI = FXI + fxij
70             FYI = FYI + fyij
71             FZI = FZI + fzij
72
73             FX(j) = FX(j) - fxij
74             FY(j) = FY(j) - fyij
75             FZ(j) = FZ(j) - fzij
76
77             !PRECALCULO DE PRESION
78             Pres1 = Pres1 + U3
79
80          End If Potencial
81
82       End Do Parti2
83
84       !GUARDANDO FUERZA
85       FX(i) = FXI
```

```
86      FY(i) = FYI
87      FZ(i) = FZI
88
89    End Do Parti1
90
91    !CALCULO DE PRESION
92    Pres = dens + (dens / (3.0 * real(N) ) ) * Pres1
93
94    !GUADANDO TERMALIZACION (ENERGIA POR PARTICULA)
95    Write(3,*) L , EnePot / Real(N), Pres
96
97    If( mod(L , iPrint) == 0 ) Then
98       Write(*,*) L , EnePot / Real(N) , Pres                    !MONITOREO EN PANTALLA
99    End If
100
101 End Subroutine Fuerza
```

Listing 4: Código para calculo de Fuerzas Potencial Yukawa

```
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
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 :: x0, y0, z0, xN, yN, zN, x0N, y0N, z0N
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
23   MBin = Int( RCut / delTar )
24
25   Allocate( Histo(MBin+1) , STAT = istat1, ERRMSG = err_msg1)
26
27   Histo = 0 !INICIALIZANDO EN 0 HITOGRAMA
28
29   Parti0 : Do i = 1, N
30
31      NextParti : Do j = 1, N
```

```
32
33        NOTSAME : If (i /= j ) Then
34
35           StepCnfg : Do k = 1, NN
36
37              !PARTICULA i ORIGEN
38              xO = CX( i , k )
39              yO = CY( i , k )
40              zO = CZ( i , k )
41
42              !PARTICUAL j CERCANA
43              xN = CX( j , k )
44              yN = CY( j , k )
45              zN = CZ( j , k )
46
47              !DISTANCIA
48              xON = xN - xO
49              yON = yN - yO
50              zON = zN - zO
51
52              !CONDICION DE IMAGEN MINIMA
53              xON = xON - BoxL*Anint( xON/BoxL )
54              yON = yON - BoxL*Anint( yON/BoxL )
55              zON = zON - BoxL*Anint( zON/BoxL )
56
57              !DISTANCIA ENTRE LAS PARTICULAS
58              rD = sqrt( (xON * xON) + (yON * yON) + (zON*zON) )
59
60              !CERCANIA CINTA
61              iBin =  Int( rD / delTar ) + 1
62
63              Guardar : If((iBin .LE. MBin)  ) Then
64
65                 Histo(iBin) = Histo(iBin) + 1
66
67              End If Guardar
68
69           End Do StepCnfg
70
71        End If NOTSAME
72
73     End Do NextParti
74
75  End Do PartiO
76
77
78
79  c1 = ( 4.0 / 3.0 ) * PI * Dens
80
```

```
  !ABRIENDO ARCHIVO PARA GDR
  Open( 5, file= "gdr.dat" )

  GdrCal: Do ibin = 1 , MBin

     rL = Real(iBin - 1) * delTar

     rU = rL + delTar
     rM = rL + ( delTar/2.0 )

     c2 = c1 * ( ( rU**3 ) - ( rL**3 ) )
     gdr = Real( Histo(iBin) )/ Real(NN) / Real(N) / c2
     gdrm = gdr

     Write(5,*) rM , gdr

  End Do GdrCal

  Close(5)

  Deallocate( Histo )

  Write(*,*) "GDR DONE, SAVE"

End Subroutine GdrCalc
```

Listing 5: Código para el calculo de la G(r)

```
!=============================================================================================
! SUBRUTINA DE CALCULO DE PROPIEDADES DE AUTO DIFUSION. DESPLAZAMIENTO CUADRATICO MEDIO Y
! COEFICIENTE DE DIFUSION DEPENDIENTE DEL TIEMPO
!
! AUTOR: MARTIN ALEJANDRO PAREDES SOSA
!=============================================================================================

Subroutine WDT(k)

  Use cte
  Implicit None

  Integer :: k, i, j, p, nmax                                  !CONTADORES
  Real :: Ti, Time                                             !CONTROL DE TIEMPO
  Real :: Wtx, Wty, Wtz, Wt                                    !DESPLAZAMIENTO CUADRATICO MEDIO
  Real :: Dif                                                  !DIFUSION

  Open(96, File="wdt.dat")

  !TIEMPO ENTRE CONFIGURACIONES
  Ti = Real(iSave2) * dt

```

```
23   !BARRIDO TEMPORAL
24   TEMPO: Do i = 1 , k-1
25
26      nmax = k - i
27
28      Wtx = 0.0
29      Wty = 0.0
30      Wtz = 0.0
31      Wt = 0.0
32
33      !BARRIMIENTO DE PARTICULAS
34      BPart: Do p = 1, N
35
36         !BARRIDO EN TIEMPO
37         Tiempo: Do j = 1, nmax
38
39            Wtx = Wtx + ( CXD(p,i+j) - CXD(p,j)  ) * ( CXD(p,i+j) - CXD(p,j)  )
40            wty = Wty + ( CYD(p,i+j) - CYD(p,j)  ) * ( CYD(p,i+j) - CYD(p,j)  )
41            Wtz = Wtz + ( CZD(p,i+j) - CZD(p,j)  ) * ( CZD(p,i+j) - CZD(p,j)  )
42
43         End Do Tiempo
44
45      End Do BPart
46
47      Time = Ti * Real(i)
48      Wt = (Wtx + Wty + Wtz ) / ( Real(nmax) * Real(N) * 6.0 )
49      Dif = Wt / Time
50
51      Write(96,*) Time, Wt, Dif
52
53   End Do TEMPO
54   Close(96)
55
56 End Subroutine WDT
```

Listing 6: Código del calculo de la W(t) y D(t)

```
1  !=====================================================================
2  ! SUBRUTINA GENERADORA DE VALORES ALEATORIOS CON UNA DISTRIBUCION
3  ! GAUSSIANA.
4  !
5  ! AUTOR: MARTIN ALEJANDRO PAREDES SOSA
6  ! MODIFICACION DEL DE GABY
7  !=====================================================================
8
9  Subroutine RanGauss(Gauss)
10   Use cte
11   Implicit None
12
13   Real :: Rand1, Rand2 !VALORES ALEATORIOS CON DISTRIBUCION UNIFORME
```

```
14   Real :: Gauss        !VALORES ALEATORIOS CON DISTRIBUCION GAUSSIANA
15
16   !GENERANDO NUMEROS ALEATORIOS UNIFORMES
17 12 Call random_number(Rand1)
18   Call random_number(Rand2)
19
20   !PREVENIR QUE DIVERJA EL LOGARITMO AL EVALUARLO EN Rand1
21   Diver:If (rand1.le.1E-8) Then
22      go to 12
23   End If Diver
24
25   !GENERANDO UN NUMERO ALEATORIOS CON DISTRIBUCION GAUSSIANA
26   Gauss = Sqrt ( -2.0 * log(Rand1) ) * cos(2.0 * pi * Rand2)
27
28
29 End Subroutine RanGauss
```

Listing 7: Código para generar valores aleatorios con distribución gaussiana