Universidad Nacional de Córdoba
FACULTAD DE MATEMÁTICA ASTRONOMÍA Y FÍSICA

Proyecto final:
FISICA COMPUTACIONAL

El control de Temperatura en Dinámica Molecular

Paolo Sebastianelli

Introducción

Una buena termoestatización del sistema, en una simulación de Dinámica Molecular, es fundamental si se quiere estudiar un experimento a T constante.

Los experimentos computacionales a T constantes,o sea los experimentos que muestrean las configuraciones a partir de un ensamble canónico (NVT constantes), reflejan el comportamiento real del sistema de manera más precisa.

Se describen, en este trabajo final, dos métodos de regulación y control de la temperatura en MD con esquema canónico: el termostato de Andersen y el termostato de Nose-Hoover.

Fundamentos

Como nos enseña Frenkel [1], desde el punto de vista mecánico estadístico podemos imponer una temperatura T al sistema llevandolo a contacto con un baño térmico grande (un reservorio). La probabilidad de encontrar el sistema a una dada energía será entonces:

$$\mathcal{P}(p) = \left(\frac{\beta}{2\pi m}\right)^{3/2} \exp\left[-\beta p^2/(2m)\right]$$

y en particular obtendremos la relación entre la T impuesta y la energía cinética por partículas:

$$k_B T = m\,\langle v_\alpha^2 \rangle$$

Tenemos que tener en cuenta que en un ensamble canónico la Temperatura “cinética” instantánea, que tomamos como referencia Tk, fluctúa dado que su varianza es:

$$\frac{\sigma^2_{T_k}}{\langle T_k \rangle^2_{NVT}} = \frac{1}{N}\frac{\langle p^4 \rangle - \langle p^2 \rangle^2}{\langle p^2 \rangle^2} = \frac{2}{3N}$$

donde p es el momento y N es el numero de partículas.

Podemos ya diferenciar los distintos enfoques propuestos por Andersen y Nose-Hoover, por lo que concierne el acoplamiento sistema-termostato. En el primer caso, el acoplamiento se realiza por medio de fuerzas impulsivas estocásticas que actúan ocasionalmente sobre partículas seleccionadas de *manera aleatoria*. Se pueden considerar como movimientos de tipo MonteCarlo que llevan el sistema de una *shell* a energía constante a otra. La evolución temporal del sistema sigue entonces la descripción Newtoniana del problema, y se asegura que las *shells* sean visitadas según su peso estadístico de Boltzmann.

De otro lado Nosé ha demostrado como se pueda implementar una Dinámica Molecular a T constante, *de tipo determinista*. Se hace a través de un Lagrangiano extendido que contenga además de las coordenadas y velocidades de las partículas, aquellas “artificiales” del baño.

Termostato de Andersen

Pasos de la simulación a T constante con termostato de Andersen (con partes de codigo):

- Inizialización de posiciones y momentos

```
!++++++++++[Generatore di posizioni iniziali]++++++++++
!++++++++++[Caso di una struttura fcc]+++++++++++++++++

Subroutine Atom_Position(Lato_Cassa,x,y,z)

    integer(dp)  :: i,j
    real(dp) :: ox,oy,oz,a,a2,count_id
    real(dp), intent(IN) :: Lato_Cassa
    real(dp), dimension(N_particulas), intent(OUT) :: x,y,z

   ! open(20,file="struttura_fcc.dat",status="replace")

    write(*,*) 'numero di siti atomici', N_sitios
    ! write(*,*) 'L dopo di entrare', L
    ! write(*,*) 'N_sitios dentro atom position', N_sitios

    a=Lato_cassa/dfloat(N_sitios)
    a2=a*0.5_dp

    x=0._dp
    y=0._dp
    z=0._dp

    count_id=0._dp
        id_atom=0

      do k=1,N_sitios
        do j=1,N_sitios
          do i=1,N_sitios

            ox=a*dble(i-1)   !sottraggo L/2 cosi va da -L/2 a L/2
            oy=a*dble(j-1)
            oz=a*dble(k-1)
```

```
**********************************************
++++++++++++[Velocitá iniziali]+++++++++++++++++
**********************************************

routine Vel(Temp)

l(dp) :: sumx,sumy,sumz,sumxt,sumyt,sumzt,sumv2,sig
l(dp), intent(IN)  :: Temp
l(dp) :: Temp_inst_iniziale


open(31,file="velocitá_iniziali_gassdev.dat",status="replace")

 sumx=0._dp  !++++[azzero le componenti del centro di massa]++++
 sumy=0._dp
 sumz=0._dp

 idum=123456

 sig=dsqrt(Temp)  !in base alla temperatura che faccio entrare determ

 do i=1,N_particulas

  v_x(i)=Gasdev(idum)*sig    !assegno le velocitá con Gasdev
  v_y(i)=Gasdev(idum)*sig
  v_z(i)=Gasdev(idum)*sig

  sumx=sumx+v_x(i) !+++[somma delle velocitá:velocitá iniziali di tut
  sumy=sumy+v_y(i)
  sumz=sumz+v_z(i)

  sumv2=sumv2+(v_x(i)*v_x(i))+(v_y(i)*v_y(i))+(v_z(i)*v_z(i)) !++++[p
  !++ho calcolato il primo valore di  sumv2 che mi serve per calcolar
```

!**********[ver Inizialize.f90]**********

En termalización y simulación:

- Integración de las ecuaciones de Newton

```
!++++[muovo le particelle, perché ho le velocitá iniziali]+++++
!+++  Uso Velocity Verlet: Swope et al JCP 76 (1982) 637

call Nuove_Posizioni (r_x,r_y,r_z,v_x,v_y,v_z,F_x,F_y,F_z)   !in calcoli_nh

   !++++[Registro le vecchie forze]++++

   do i=1,N_particulas
   F_x_prima(i)=F_x(i)
   F_y_prima(i)=F_y(i)
   F_z_prima(i)=F_z(i)
   end do

!****[Primo step per il calcolo di V per un dt/2 in piú ****

!++++[Calcolo il primo step (dt/2) delle velocitá con l'algoritmo Velocity Verlet perché ho le forze]++++++++

call Vel_Newton_1 (id_tempo,v_x,v_y,v_z,F_x,F_y,F_z) !In Calcoli_nh

!++++++[Calcolo le nuove forze forze perché ho le nuove posizioni] +++++++++

call Forze (r_x,r_y,r_z,F_x,F_y,F_z,Energia,ecorr)  !in Calcoli

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

!++++[Calcolo il secondo step (dt/2) delle velocitá con l'algoritmo Velocity Verlet perché ho le forze]++++++++


 call Vel_Newton_2 (id_tempo,v_x,v_y,v_z,F_x,F_y,F_z,sum_x,sum_y,sum_z,Energia,sumv_2,Temp_inst,T_ref) !In Calcoli

 !write(1000,*) id_tempo,Temp_inst

 id_tempo=id_tempo+1
 tempo=dt*id_tempo
```

!**********[ver MD.f90 y Calcoli.f90]**********

– Asignación de una probabilidad **ν***dt de colisión estocástica

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!++++[ANDERSEN Termostato: collisioni con il bagno termico]++++++++
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   sig=dSQRT(Temp_richiesta)

   !write(*,*) sig

   DO i = 1, N_particulas

  !test per la collisione

    IF (ran2(idum).lt.10._dp*dt) THEN

  ! la particella collide

   v_x(i) = Gasdev(idum)*sig
   v_y(i) = Gasdev(idum)*sig
   v_z(i) = Gasdev(idum)*sig

    END IF
   END DO

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!+++++++++++++[FINE TERMOSTATO DI ANDERSEN]++++++++++++++++++++++++
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

– Si la partícula i ha sido elegida para colisionar con el baño su nueva velocidad se extrae de una distribución de Maxwell-Boltzmann que corresponde a la T deseada (Temp_richiesta). Las otras partículas no resultan afectada por la colisión.

En ambos casos, Andersen o Nosé-Hoover, lo que hay que elegir inicialmente es la fuerza de acoplamiento entre el sistema y el baño termostatico.

En este caso la fuerza de acoplamiento está dada por la frecuencia de colisiones estocásticas **ν.**

Termalización:

El estudio de la termalización es fundamental por cada valor de **ν**

Después de haber inizializado el sistema en una estructura fcc, la termalización llevada a cabo durante la simulación ha sido, como de lógica, función de la frecuencia de colisión:

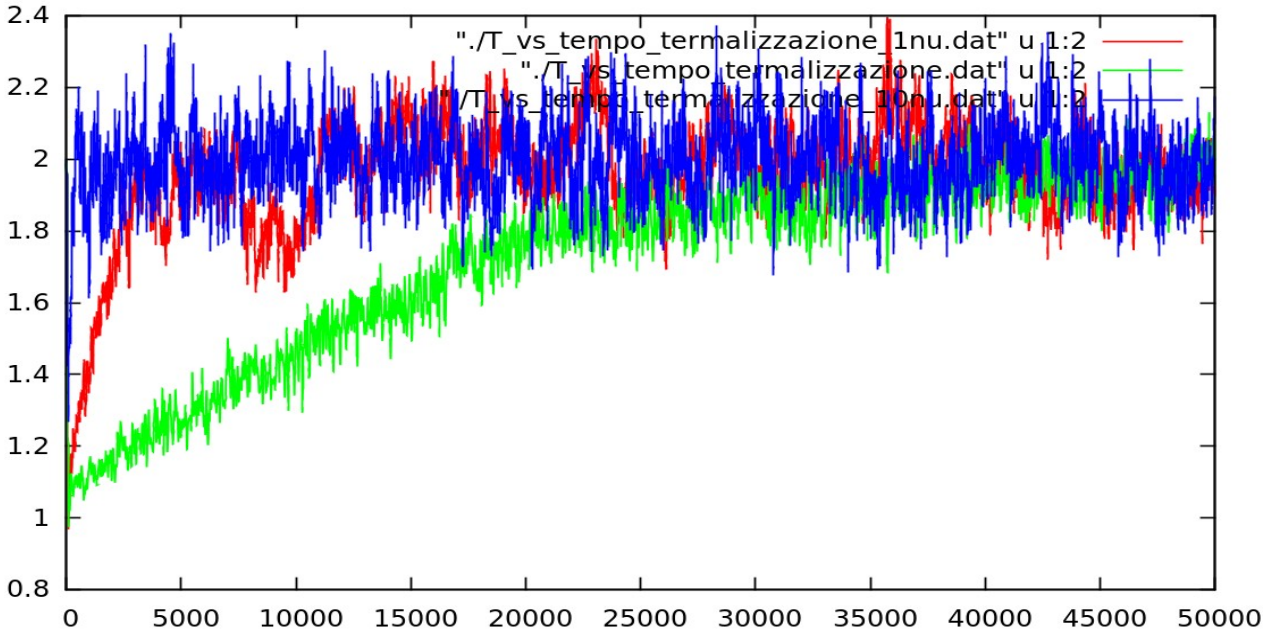

Fig.1 – Termalización como función de la frecuencia de colisión *nu*: 10(azul),1(rojo),0,1(verde). La Temperatura deseada es T=2.0

El sistema arranca de un valor de T=2.0 y decae alrededor de 1 casi instantáneamente. Necesita un tiempo, función de nu, para volver a estabilizarse al valor pedido.

La pregunta inicial es: ¿la frecuencia de colisión modifica la distribución de velocidades?
El siguiente gráfico muestra que la distribución de velocidades, de tipo Maxwell-Boltzmann no es dependiente de **ν***:*

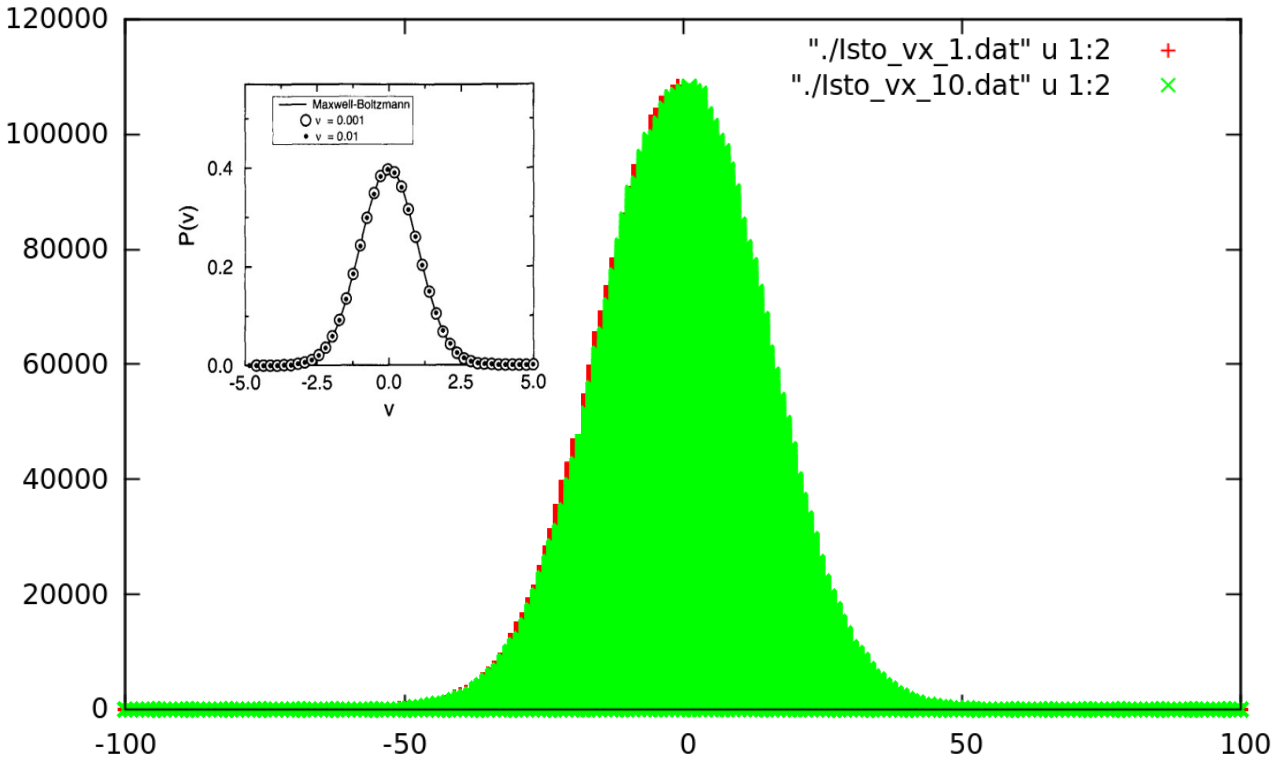

Fig.2 – Distribuciones de velocidades por diferentes nu -
Las distribuciones son indiependientes del valor de la frecuencia de colisión (T=2)

Cambiando los valores de la rho (densidad) se confirma el perfil de la curva relativa a la ecuación de estado de LJ. No es exactamente la misma de el articulo de referencia que propone el Frenkel, de Johnson et al. [2], pero sirve para asegurarnos que el algoritmo este trabajando bien.

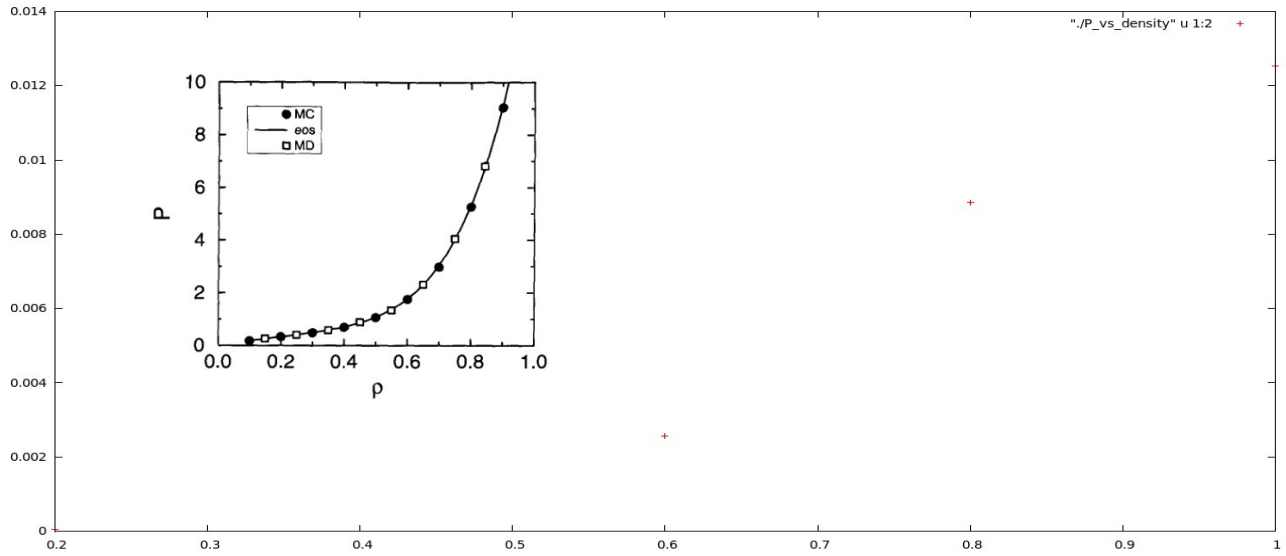

Fig.3 – P_vs_Densidad: en eje y los valores de P, en eje x los valores de densidad (0,2/0,6/0,8/1)
No sigue bien la figura del Frenkel porque no toma en cuenta las mejoras del articulo de Johnson et al.

Una diferencia importante entre los dos métodos de control de temperatura está en el

comportamiento de el Desplazamiento Cuadratico Medio (o RMS) en función del parámetro de acoplamiento con el baño, que hemos dicho, en este caso es la frecuencia de colisión, y después será la masa efectiva Q (o parámetro de inercia) asociado al baño termostatico.

Como podemos observar de la siguiente figura, obtenida por diferentes ν:

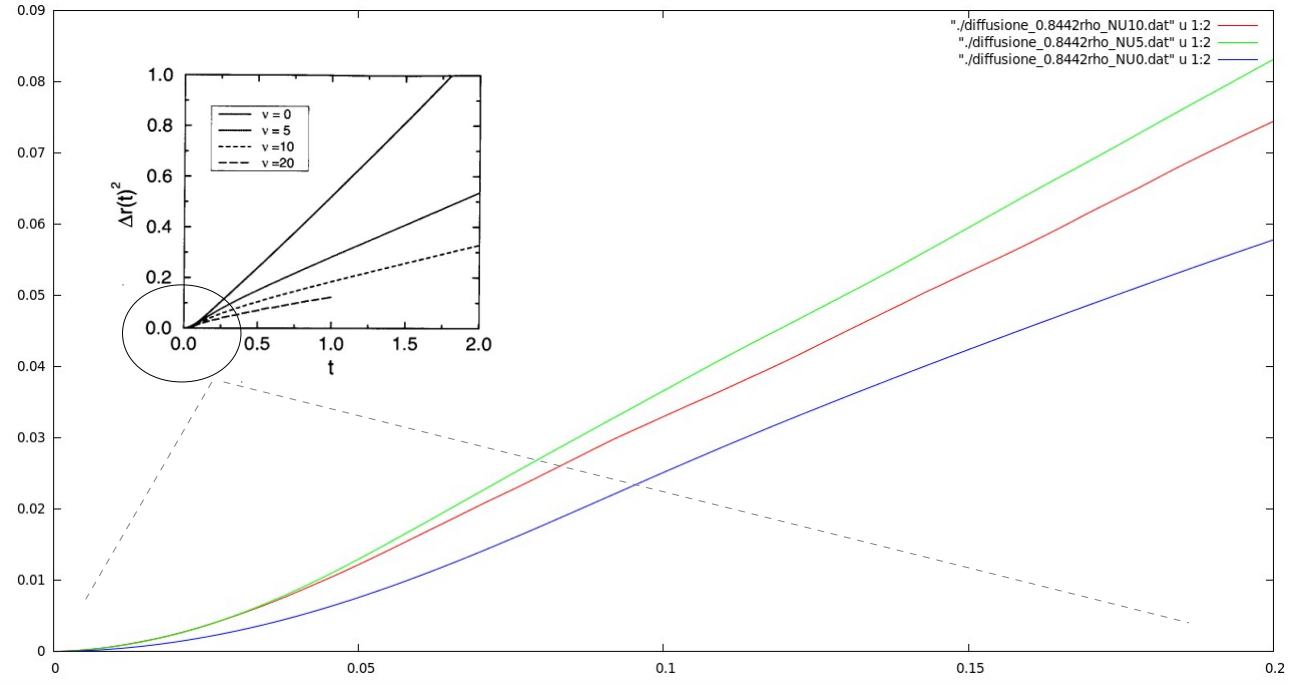

Fig.4 – Cambiando el valor de la frecuencia de colisión varía el DCM (o RMS)

Termostato de Nosé-Hoover

La naturaleza determinista del método de Nosé ha sido simplificada por intuición de Hoover.

Nosé formuló un Lagrangiano extendido, tomando en cuenta una coordenada **s** más, y un momento **p,** relativos al baño térmico que se utiliza.

La formulación del Lagrangiano, ya que uno quiere que la velocidad i-esima sea *s(dr/dt)(i)* es la siguiente:

$$\mathcal{L}_{\text{Nose}} = \sum_{i=1}^{N} \frac{m_i}{2} s^2 \dot{r}_i^2 - \mathcal{U}(\mathbf{r}^N) + \frac{Q}{2}\dot{s}^2 - \frac{L}{\beta} \ln s$$

donde

mi= masa de la partícula i
s = coordenada del termostato
Q = masa efectiva del termostato
L=3*N_particulas

y la parte logarítmica sirve para no perder el carácter canónico del ensamble.

Las ecuaciones de movimiento de Nosé:

$$\begin{aligned}
\frac{d\mathbf{r}_i}{dt} &= \frac{\partial \mathcal{H}_{\text{Nose}}}{\partial \mathbf{p}_i} = \mathbf{p}_i/(m_i s^2) \\
\frac{d\mathbf{p}_i}{dt} &= -\frac{\partial \mathcal{H}_{\text{Nose}}}{\partial \mathbf{r}_i} = -\frac{\partial \mathcal{U}(\mathbf{r}^N)}{\partial \mathbf{r}_i} \\
\frac{ds}{dt} &= \frac{\partial \mathcal{H}_{\text{Nose}}}{\partial p_s} = p_s/Q \\
\frac{dp_s}{dt} &= -\frac{\partial \mathcal{H}_{\text{Nose}}}{\partial s} = \left(\sum_i p_i^2/(m_i s^2) - \frac{L}{\beta} \right) /s
\end{aligned}$$

se obtienen a partir del relativo hamiltoniano:

$$\mathcal{H}_{\text{Nose}} = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i s^2} + \mathcal{U}(\mathbf{r}^N) + \frac{p_s^2}{2Q} + L\frac{\ln s}{\beta}$$

Hoover simplificó está formulación considerando un termino, llamado coeficiente de fricción termodinámica :

$$\xi = s' p_s'/Q$$

en el cual s' y p' son variables que llamamos reales, y se relacionan con las virtuales de esta forma:

$$\begin{aligned}
r' &= r \\
p' &= p/s \\
s' &= s \\
\Delta t' &= \Delta t/s
\end{aligned}$$

Por medio de este coeficiente es posible re formular las ecuaciones de movimiento como:

$$\begin{aligned}
\dot{\mathbf{r}}_i &= \mathbf{p}_i/m_i \\
\dot{\mathbf{p}}_i &= -\frac{\partial \mathcal{U}(\mathbf{r}^N)}{\partial \mathbf{r}_i} - \xi \mathbf{p}_i \\
\dot{\xi} &= \left(\sum_i p_i^2/m_i - \frac{L}{\beta} \right) /Q \\
\dot{s}/s &= \frac{d \ln s}{dt} = \xi.
\end{aligned}$$

Igualmente, el esquema de Hoover no se puede implementar directamente en el algoritmo del Velocity Verlet que hemos encontrado en laboratorio 5, dado que si desarrolláramos el estudio de las ecuaciones, para calcular la velocidad i-esima al tiempo t+dt, nos daríamos cuenta que:

$$\begin{aligned} v_i(t+\Delta t) \;=\; & v_i(t) + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t) \\ & + f_i(t)/m_i - \xi(t)v_i(t)]\,\Delta t/2. \end{aligned}$$

vi(t+dt) se encuentra en ambos lados de la ecuación. Por esta razón el método viene implementado con un esquema predictor-corrector o solucionado iterativamente. En este caso se ha solucionado iterativamente con el método de Newton-Raphson *loop* o de las tangentes.

Pasos de la simulación a T constante con termostato de Nosé-Hoover:

- Inizialización de posiciones y velocidades y generación de S,PS (coordenada y momentos iniciales)

```
!**** [Inizializo] ***************************************************************
!*******************************INIZIALIZO*******************************************
!********************************************************************************
!********************************************************************************

    L=(dble(N_particulas)/rho)**(1._dp/3._dp)  ! lunghezza cassa
    a=L/(N_particulas*0.25_dp)**(1._dp/3._dp)     ! parametro reticolare
    delta_funz_rad=L/50._dp                    ! delta della funzione radiale

    call Atom_Position(L,r_x,r_y,r_z) !Posiziono gli atomi in una struttura fcc

    do i=1,N_particulas

        x_iniziale(i)=r_x(i)
        y_iniziale(i)=r_y(i)     !assegno le posizioni atomiche iniziali
        z_iniziale(i)=r_z(i)     !non so se mi serviranno, peró cosi ce le ho immagazzinate

    enddo

    !+++[a queste posizioni assegno le velocitá iniziali]++
    !+++[La prima temperatura assegnata é T_ref=1]+++++++++

    T_ref=1.0_dp

    !++++[Velocitá Iniziali]++++

    call Vel(T_ref,S,PS) !in Inizialize_nh

    !Esco con i valori di S e PS iniziali che mi servono pe


    !++++[Forze iniziali]+++avendo le posizioni assegnate p
    !++++ quindi ottengo le forze INIZIALI
    call Forze (r_x,r_y,r_z,F_x,F_y,F_z,Energia,ecorr) !in

    !+++[NOTA: velocitá e forze iniziali saranno quelle che

    !+++ [FINE INIZIALIZZAZIONE] +++++++*************************************************
    !*************************************************FINE INIZIALIZZAZIONE*************++++++++++
    !********************************************************************************************
    !********************************************************************************************
```

```
!*********************************************************
!++++++++++++++[CALCOLO INIZIALE DI S E PS]++++++++++++++++
!*********************************************************

   QQ=1._dp/Q
   Si = 0._dp                !il primo valore si S lo imposto a zero, poi si ricalcola
   PSi = (sumv2-G*Temp)*QQ     !Per calcolare il primo valore di PS ho bisogno di sumv2

   !G=3*N_particulas
   !Temp, é la richiesta
   !Q é la Hoover mass

!*********************************************************
!++++++++++++++[FINE S E PS]+++++++++++++++++++++++++++++++
!*********************************************************
```

!**********[ver Inizialize_NH.f90]**********

- Implementación del algoritmo Velocity Verlet (en termalización y simulación; se muestran solo las variaciones más significativas de código, respeto al termostato di Andersen):

  - Se mueven las partículas de un dt

  - Se hace la actualización de las velocidades de un dt/2

```
!*****************NOSE HOOVER INTEGRATION **********************
!*****************************************************************

Subroutine Vel_Newton_1 (Time,v_x,v_y,v_z,Fx,Fy,Fz,Fxo,Fyo,Fzo &
                         ,sumx,sumy,sumz,sumv2,Temp_richiesta,SSV,PSSV)
```

  - Se calculan los nuevos S y PS

```
QQ=1._dp/Q

SSV= SSV + PSSV*dt + (sumv2-G*Temp_richiesta)*((dt*dt)*0.5_dp)*QQ
PSSV = PSSV + (sumv2-G*Temp_richiesta)*(0.5_dp*dt)*QQ

!write(*,*) 'dentro vel_newton1 s,ps' , Time,SSV,PSSV
!write(*,*) 'dentro vel_newton1 G,Q,sumv2,Temp richiesta' , G,Q,sumv2,Temp_richiesta

!*****************************************************
!*****************************************************
!*****************************************************
```

  - Se calculan las nuevas fuerzas
  - Se termina la actualización de las velocidades de otro dt/2
  - Se aplica el *loop* Newton-Raphson y el criterio de convergencia

```
!***************************NEWTON_R LOOP

 psn = PSSV2          !un nuovo ps
 ready = .FALSE.    ! variabile logica in false
 iter = 0

        DO WHILE (.NOT.ready.AND.iter.LT.100)    ! cicla 100 volte?

           iter = iter + 1 ! aumenta di uno iter
           pso = psn       !definisce vecchio il ps appena ridefinito ..come vecchio
           delps = 0._dp

           DO i = 1, N_particulas

              vxo(i) = vxn(i)  !definisce le velocitá come vecchie
              vyo(i) = vyn(i) ! definisce le velocitá iniziali come quelle calcolate nello switch 2
              vzo(i) = vzn(i)

              bx(i) = -(0.5_dp*dt)*(Fx(i)-pso*vxo(i)) - (v_x(i)-vxo(i))
              ri = vxo(i)*dt*QQ

              delps = delps + ri*bx(i) !qui calcola la variazione di ps nel tempo e lo fa tramite bx

              by(i) = -(0.5_dp*dt)*(Fy(i)-pso*vyo(i)) - (v_y(i)-vyo(i))
              ri = vyo(i)*dt*QQ

              delps = delps + ri*by(i) !qui ricalcola la variazione di ps nel tempo e lo fa tramite by

              bz(i) = -(0.5_dp*dt)*(Fz(i)-pso*vzo(i)) - (v_z(i)-vzo(i))
              ri = vzo(i)*dt*QQ

              delps = delps + ri*bz(i) !qui ricalcola la variazione di ps nel tempo e lo fa tramite bz

           END DO

           di = -(pso*(0.5_dp*dt)+1._dp)

           delps = delps - di*((-sumv2+G*Temp_richiesta)*(0.5_dp*dt)*QQ-(PSSV2-pso))

           delps = delps/(-dt*(dt*0.5_dp)*sumv2/Q+di)

           sumv2 = 0._dp

           DO i = 1, N_particulas

              vxn(i) = vxn(i) + (bx(i)+(0.5_dp*dt)*vxo(i)*delps)/di
              vyn(i) = vyn(i) + (by(i)+(0.5_dp*dt)*vyo(i)*delps)/di
              vzn(i) = vzn(i) + (bz(i)+(0.5_dp*dt)*vzo(i)*delps)/di

              sumv2 = sumv2 + vxn(i)*vxn(i) + vyn(i)*vyn(i) + vzn(i)*vzn(i)

           END DO
```


!**********[ver calcoli_NH.f90]**********

Termalización:

Es necesario tener en cuenta que la termalización, cuando se aplica el método de NH, es función de la masa efectiva Q del baño térmico.

La siguiente figura mostra como varía la termalización en función de Q:

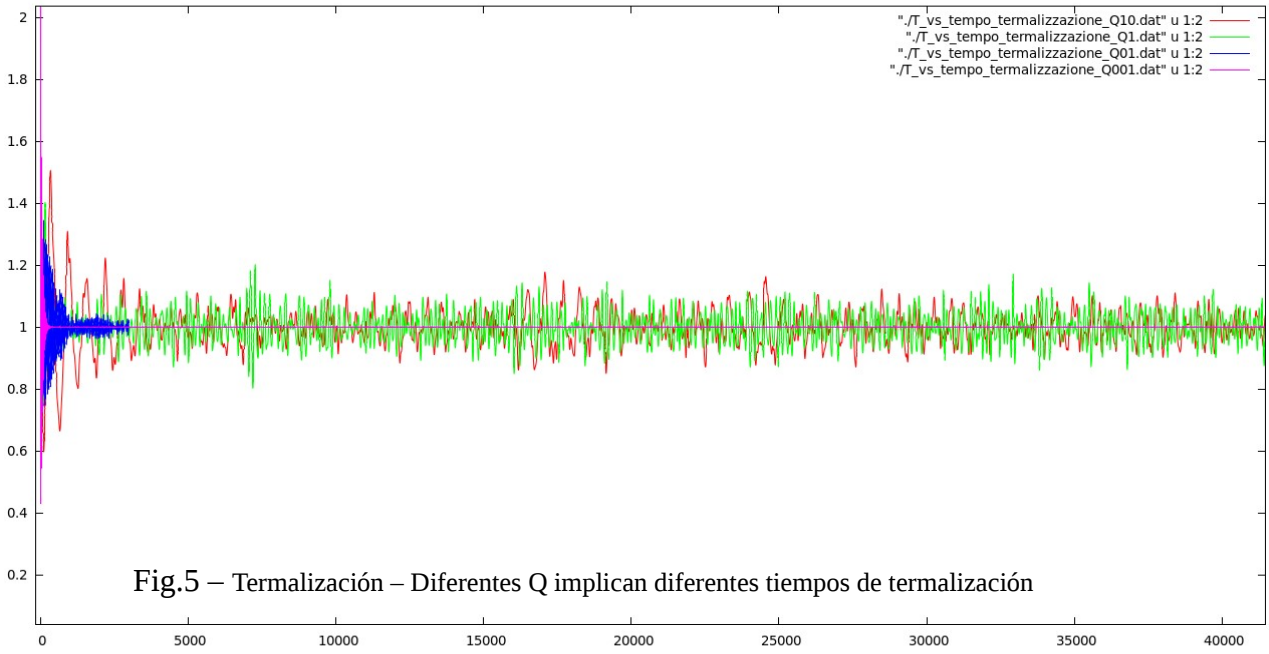

Fig.5 – Termalización – Diferentes Q implican diferentes tiempos de termalización

A una masa efectiva mayor le toca un tiempo de termalización mayor. Además como se puede ver, es importante elegir un Q oportuno. Un Q demasiado grande lleva a perder el control de la T y un Q demasiado chico está caracterizado por demasiada oscilaciones en el dt elegido.

A partir de los 20000 pasos ya todos los sistemas, con todo tipo de Q se encuentran termalizados.

Para ver como responde el sistema a un ajuste de T durante la simulación se muestra un gráfico de T vs tiempo, caracterizado por un cambio de T1=1 a T2=1.5 al paso 12000 de la corrida.

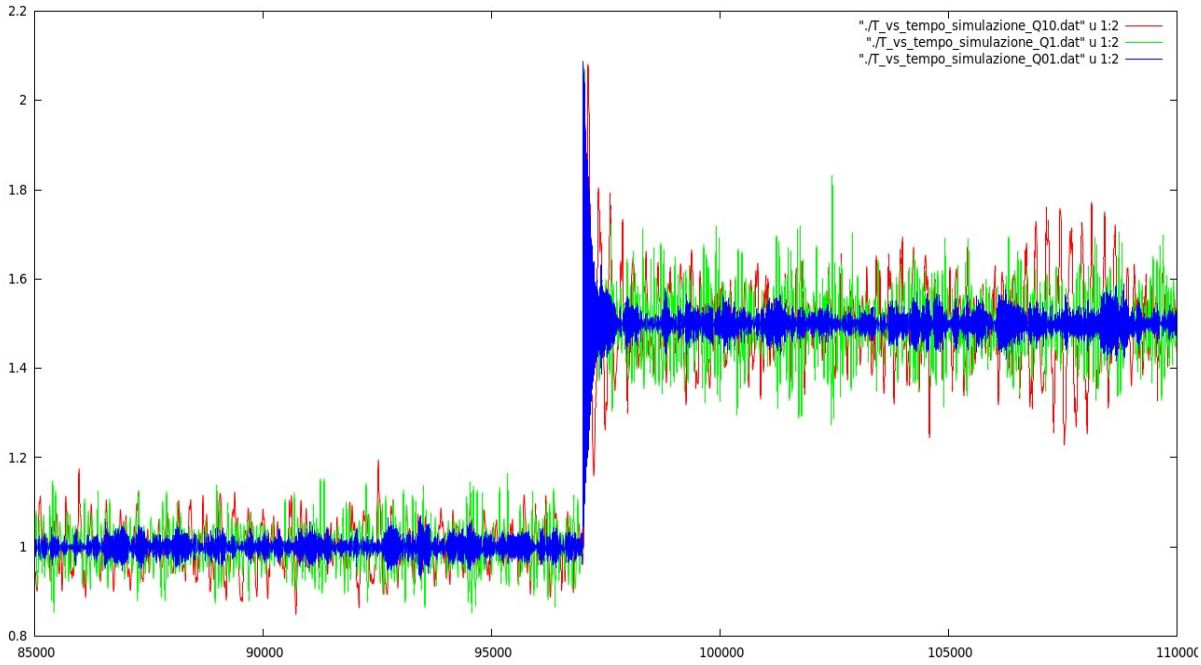

Fig.6 – Cambio de Temperatura – Respuesta del sistema al cambio de T1=1 → T2=1.5
DIferentes Q implican diferentes respuestas.

Hemos subrayado al principio como una de las diferencias entre los dos métodos reside en el comportamiento del Desplazamiento Cuadrático Medio respeto al cambio de la contaste de acoplamiento que en este caso es Q.

Se muestra como, aparentemente (ya que no se puede afirmar con certeza, como dice también Frenkel [1]), el RMS sea afectado de manera menor por parte de Q.

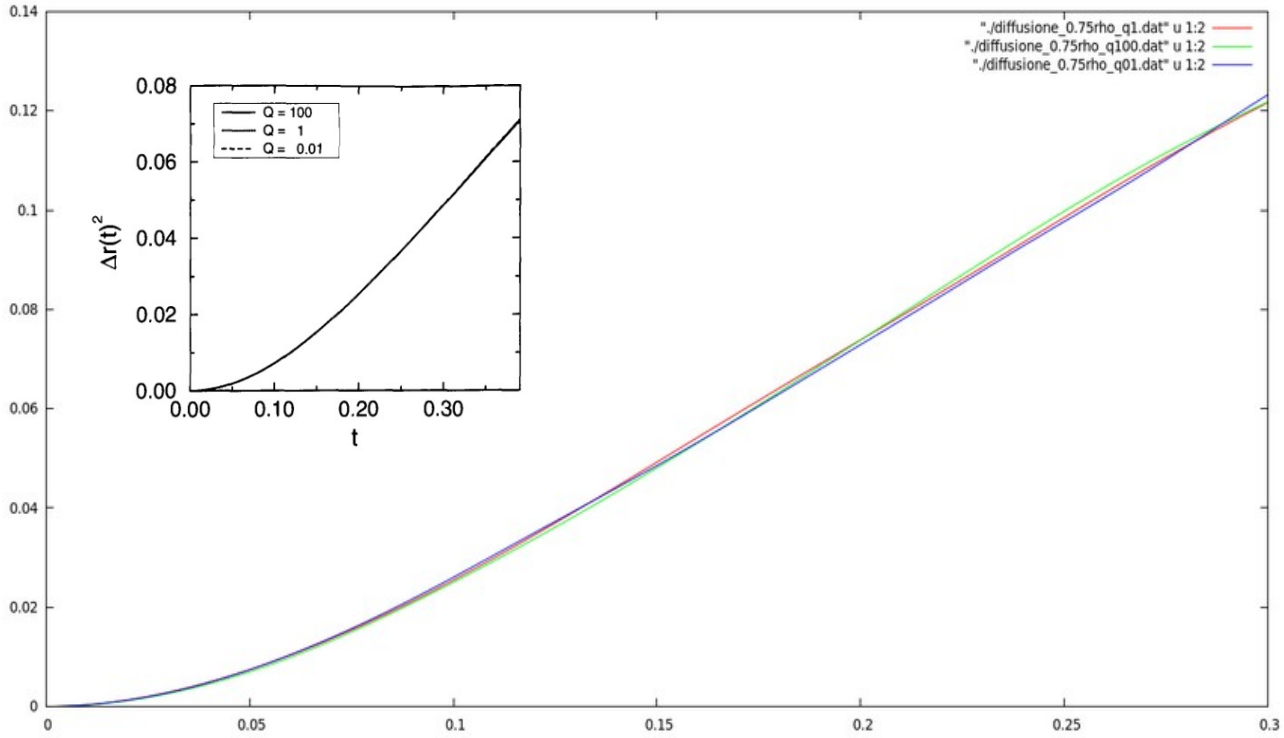

Fig.7 – RMS – Indipendencia del Desplazamiento cuadratico medio del valor de Q

Conclusiones

El método determinista de Nosé-Hoover, a pesar de la dificultad de implementación, se demuestra el más representativo, y garantiza un control más efectivo de la temperatura.

Se han podido reproducir las figuras extraída del capitulo 6 del Frenkel & Smit, pero además, se puede agregar a titulo complementario la siguiente figura, que muestra como la función de distribución radial, con ambos los termostatos, se mantiene igual a la calculada, por rho=0,8, en el Laboratorio 5, obtenida con el solo *Rescaling* de las velocidades.

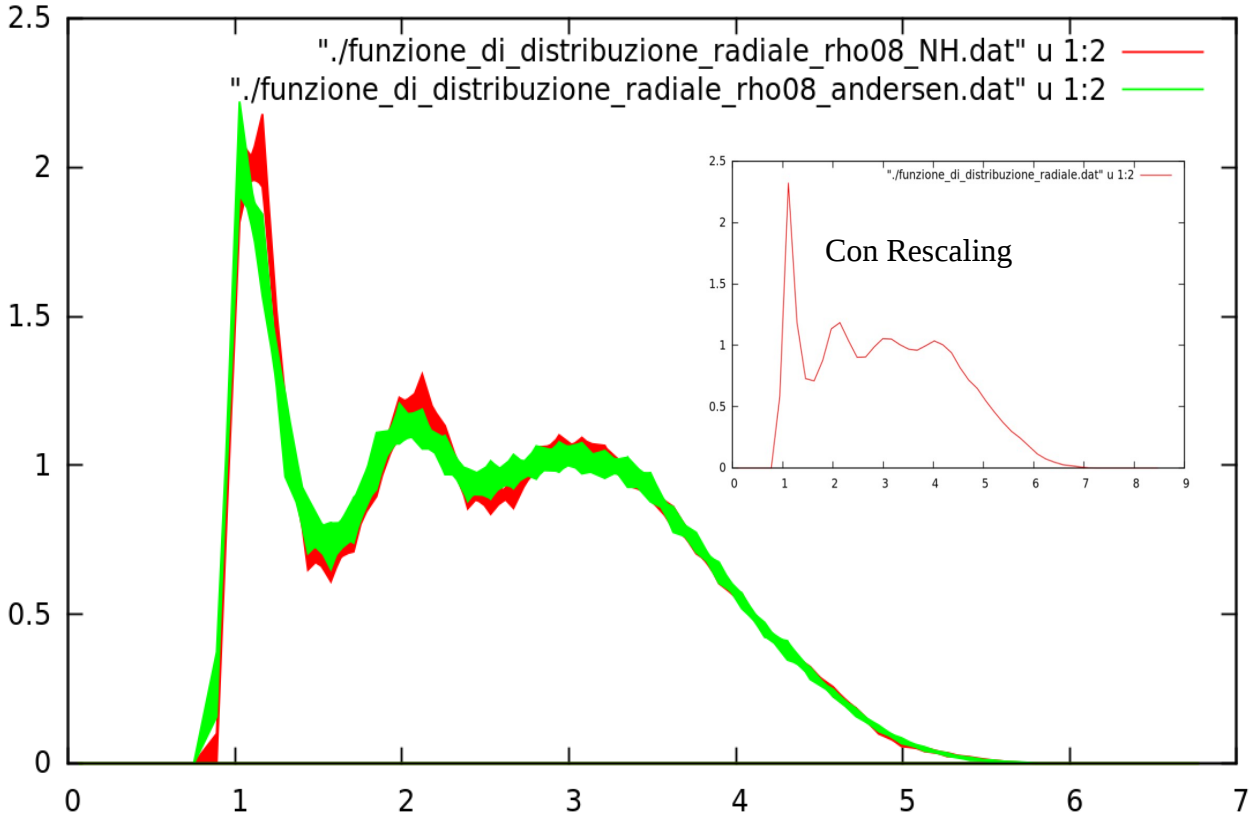

Fig. – Función de distribución radial– Rho=0.8 N=256
Comparación NH_Andersen_Labo5

Bibliografía

[1] Understanding Molecular Simulation – D. Frenkel B.Smit – Academic Press
[2] The LJ equation of state revisited – J.K Johnson et al. - Mol. Phys. 78:591-618,1993
[3] Computer simulation of liquids – M.P. Allen – D.J. Tildesley – Clarendon Press