Simulaciones de Dinámica Molecular
Detalles de implementación

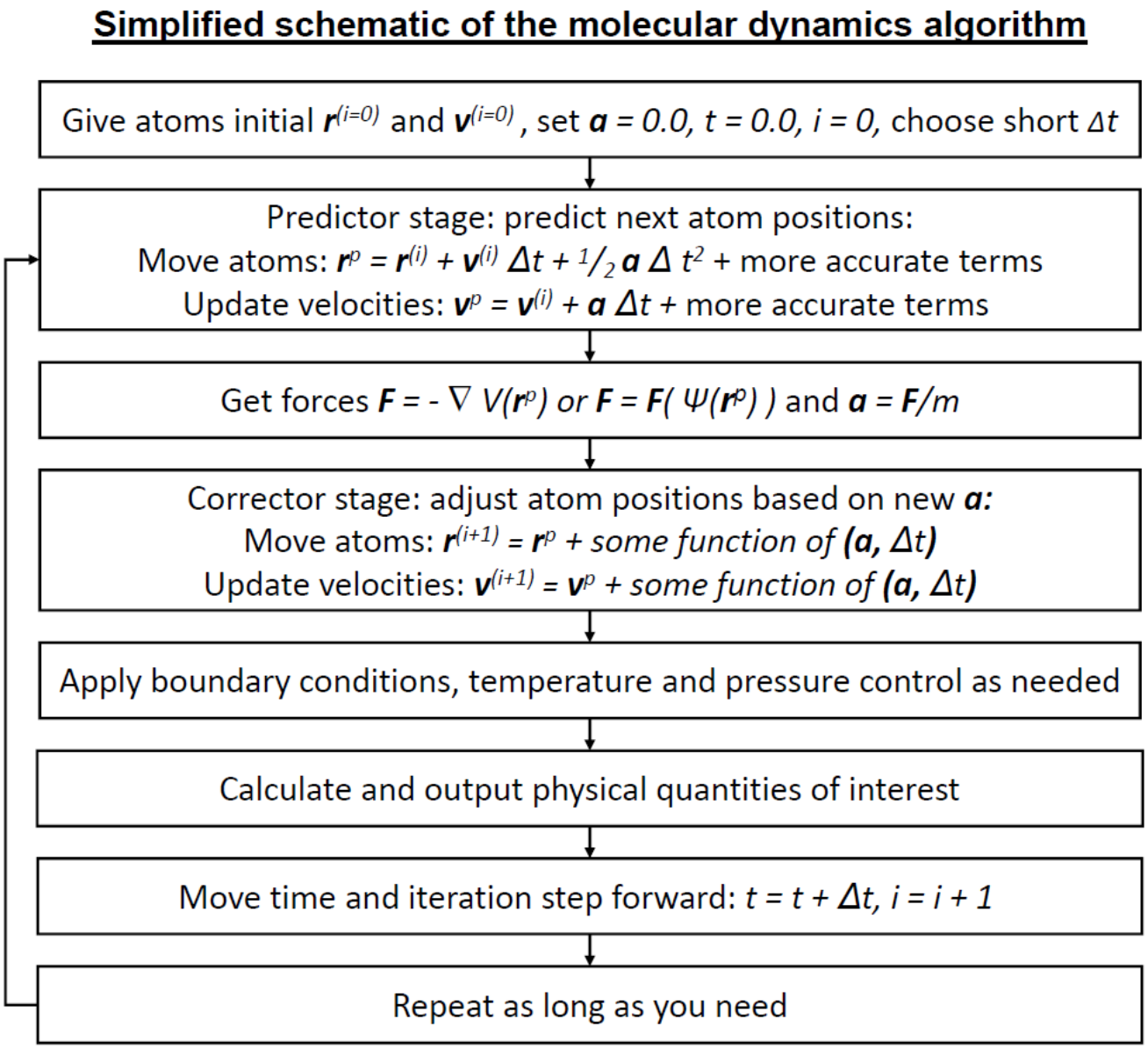
J. Manuel Solano Altamirano
Facultad de Ciencias Químicas
Benemérita Universidad Autónoma de Puebla

Pasos de tiempo sugeridos

Sistema	**Movimientos presentes**	δt(s) **sugerido**
Átomos	Trn	10^{-14}
Moléculas rígidas	Trn, Rot	5×10^{-15}
Mol. flexibles, enlaces rígidos	Trn, Rot, Tor	2×10^{-15}
Moléculas y enlaces flexibles	Trn, Rot, Tor, Vib	$5 \times 10^{-16} - 10^{-15}$
Granular matter	Trn, Rot	$5 \times 10^{-6} - \times 10^{-5}$

Cuadro: Trn≡Translacional, Rot≡Rotacional, Tor≡Torsional y Vib≡Vibracional.

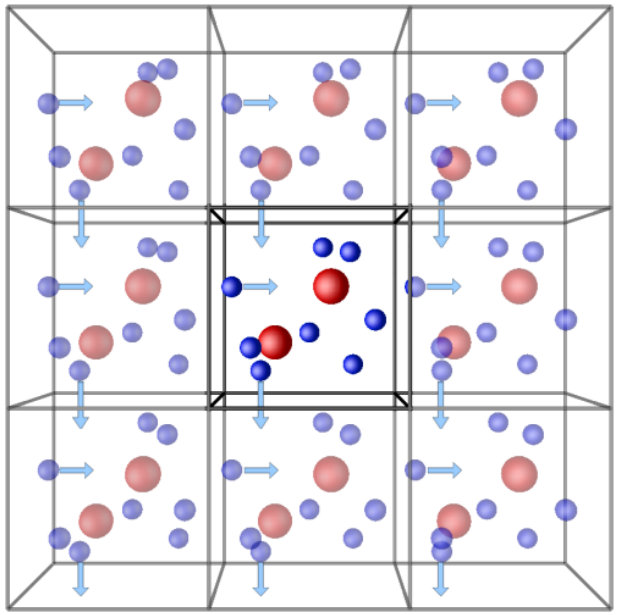
Diagrama básico de Dinámica Molecular



Programa maestro

```
1  PROGRAM LJMD
2      use MDSystem
3          ...
4      IMPLICIT NONE
5      CHARACTER(len=sln) :: inputFileName
6      INTEGER            :: i
7      CALL get_command_argument(1,inputFileName)
8      CALL LoadMDSysFromFile(inputFileName) !part of MDSystem
9      CALL LoadParticleDataFromFile(rstfile) !part of MDSystem
10     CALL InitIntegratorVerlet !part of IntegratorVerlet
11     CALL InitVanderWaalsForce !part o VanderWaalsForce
12     DO nfi=1,nsteps
13         IF (MOD(nfi,nprint) == 0) THEN
14             CALL AppendXYZData(trjfile)
15             CALL AppendProperties(ergfile)
16         END IF
17         CALL AddVanderWaalsForces !PBC should be implemented here as well
18         CALL AdvanceStepVerlet
19         CALL ApplyBoundaryConditions
20         CALL ApplyThermodynamicConditions
21     DONE
22     CALL SaveToRestFile(rstfile)
23     CALL CleanMDSys !part of MDSystem
24 END PROGRAM LJMD
```

Condiciones de frontera periódicas



- En la práctica no es necesario guardar la información de las posiciones de los átomos-imagen.*

Condiciones de frontera periódicas

- Posiciones: Si las partículas salen de la caja, ingresan nuevamente del lado contrario.

```
1  if (periodic_x) then
2    if (x >    x_size * 0.5) x = x - x_size
3    if (x <=  -x_size * 0.5) x = x + x_size
4  end if
```

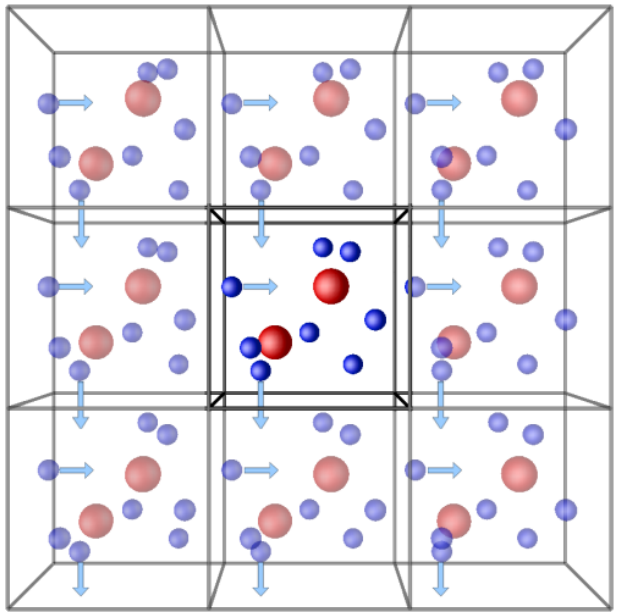
- Fuerzas: Deben considerarse las fuerzas entre las partículas imagen.

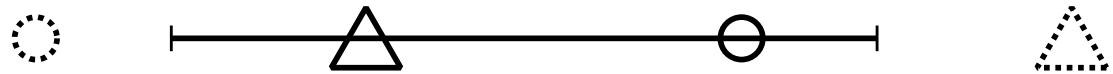
```
1  if (periodic_x) then
2    dx = x(j) - x(i)
3    if (dx >   x_size * 0.5) dx = dx - x_size
4    if (dx <= -x_size * 0.5) dx = dx + x_size
5  endif
```

Condiciones de frontera periódicas

```
1  #ifndef _BOUNDARYCONDITIONS_FNF_
2  #define _BOUNDARYCONDITIONS_FNF_
3  #include "mdsys.F95"
4  MODULE BoundaryConditions
5      use MDSystem
6      IMPLICIT NONE
7  CONTAINS
8      SUBROUTINE ApplyBoundaryConditions
9      IMPLICIT NONE
10     INTEGER    :: i,k
11     DO i=1,natoms
12         DO k=1,3
13             IF (x(i,k) > (0.5_dbl*boxlength)) THEN
14                 x(i,k)=x(i,k)-boxlength
15                 xp(i,k)=xp(i,k)-boxlength !needed because of Stormer-Verlet!
16             END IF
17             IF (x(i,k) <= (-0.5_dbl*boxlength)) THEN
18                 x(i,k)=x(i,k)+boxlength
19                 xp(i,k)=xp(i,k)+boxlength !needed because of Stormer-Verlet!
20             END IF
21         END DO
22     END DO
23     END SUBROUTINE ApplyBoundaryConditions
24 END MODULE BoundaryConditions
25 #endif
26 !#endif _BOUNDARYCONDITIONS_FNF_
```

Condiciones de frontera periódicas





Condiciones de frontera periódicas

```
1   SUBROUTINE AddVanderWaalsForces
2       use MDSystem
3       IMPLICIT NONE
4       REAL(kind=dbl) :: rsq, rcutsq, rinv, delta(3)
5       REAL(kind=dbl) :: r6, ffac
6       INTEGER :: i, j, k
7       epot=0.0_dbl
8       f=0.0_dbl
9       rcutsq=rcut*rcut
10      DO i=1,natoms-1
11          DO j=i+1,natoms
12              delta=x(i,:)-x(j,:)
13              DO k=1,3 !PBC
14                  IF (abs(delta(k)) > (0.5_dbl*boxlength)) THEN !PBC
15                      delta(k)=delta(k)-sign(boxlength,delta(k)) !PBC
16                  END IF
17              END DO !PBC
18              rsq = dot_product(delta,delta)
19              IF (rsq < rcutsq) THEN
20                  rinv = 1.0_dbl/rsq
21                  r6 = rinv*rinv*rinv
22                  ffac = (12.0_dbl*c12*r6 - 6.0_dbl*c6)*r6*rinv
23                  epot = epot + r6*(c12*r6 - c6)
24                  f(i,:) = f(i,:) + delta*ffac
25                  f(j,:) = f(j,:) - delta*ffac
26              END IF
27          END DO
28      END DO
29  END SUBROUTINE AddVanderWaalsForces
```

Presión, temperatura y volumen

- Presión (fuerzas por pares):

$$P = \frac{1}{V}\left[\frac{1}{3}\sum_{i}^{N}\left(\sum_{j>i}^{N} F_{ij}|\vec{x}_j - \vec{x}_i| + \frac{|\vec{p}_i|}{m_i}\right)\right] \tag{1}$$

- Temperatura

$$T = \frac{1}{3Nk_B}\sum_{i}^{N} m_i|\vec{v}_i|^2 \tag{2}$$

Presión, temperatura y volumen

- Las implementaciones básicas de MD simulan sistemas con energía constante (ensamble microcanónico)
- Temperatura constante (el volumen y/o la presión varía(n)): nVT. Importante en reacciones biológicas.
- Presión constante (el volumen y/o la temperatura varía(n)): nPT. Importante en reacciones químicas.

Presión, temperatura y volumen

Controlar las variables del sistema para mantener T o P fijas:

- Métodos estocásticos: obligar a las variables del sistema a que mantengan una función de distribución específica.
- Métodos de acoplamiento fuerte: Escalar las variables del sistema para proporcionar un valor exacto.
- Métodos de acoplamiento débil: Escalar gradualmente la variable deseada en dirección al valor deseado

Termostatos y barostatos

- Termostato estocástico de Langevin: aplicar fricción y una fuerza aleatoria a los momentos:

$$\frac{d\vec{p}_i}{dt} = \sum_j^N (F_{ij}|\vec{x}_i - \vec{x}_j|) - \gamma \vec{p}_i + R_i(t) \tag{3}$$

donde $R_i(t)$ es la fuerza aleatoria, la cual presenta una dispersión relacionada con el coeficiente γ según:

$$\sigma_i^2 = 2m\gamma k_B T/\delta t \tag{4}$$

- Termostato de Andersen: Asigna velocidades a partículas aleatorias, y dichas velocidades se toman de una distrubución Maxwell-Boltzmann

Termostatos y barostatos

- Termostato gausiano/isocinético (acoplamiento fuerte): escala las velocidades según:

$$\frac{d\vec{p}_i}{dt} = \sum_j^N (F_{ij}|\vec{x}_i - \vec{x}_j|) - \alpha\vec{p}_i \tag{5}$$

- Baño de Berendsen (acoplamiento débil): la celda unitaria esta embebida en un baño térmico a temperatura T_0:

$$\frac{d\vec{p}_i}{dt} = \sum_j^N (F_{ij}|\vec{x}_i - \vec{x}_j|) - \frac{\vec{p}_i}{\tau_T}\left[\frac{T_0}{T} - 1\right] \tag{6}$$

Termostato de berendsen

```
1   SUBROUTINE ApplyBerendsenThermostat
2       use MDSystem !tau and Tb should be added here (read from input file)
3       IMPLICIT NONE
4       REAL(kind=dbl)    :: lambda
5       INTEGER           :: i
6       lambda = sqrt(1.0_dbl+(dt/tau)*(Tb/(2.0_dbl*ekin/3.0_dbl/natoms)-1.0_dbl))
7       ekin = 0.0
8       DO i=1,natoms
9         v(i,:) = lambda*v(i,:)
10        ekin= ekin + 0.5_dbl * mvsq2e * mass * dot_product(v(i,:),v(i,:))
11      END DO
12  END SUBROUTINE ApplyBerendsenThermostat
```