4. Dinâmica

Aprendemos na parte inicial do curso como é possível obter uma aproximação discreta para uma equação contínua obtendo uma versão que podemos tratar numericamente. As aplicações deste tipo de aproximação são inúmeras, contudo, vamos fixar nossa atenção em alguns problemas mais frequentes em física. O programa nesta próxima etapa será o seguinte:

1. Como resolver as equações de movimento de um sistema de partícula acopladas, dadas certas condições iniciais;
2. Como estabelecer condições de contorno adequadas ao problema;
3. Equilíbrio térmico e transição de fase;
4. Evolução temporal do sistema, e
5. Cálculo das Funções Correlação, $C(\vec{r},t)$ e $C(\vec{q},\omega)$;

As equações de movimento de um sistema são as equações que descrevem o movimento do sistema como função do tempo. Em geral são equações diferenciais acopladas obtidas via leis de Newton, ou via equivalentemente de uma formulação lagrangiana [**Lagrangian**]. Ao final, procuramos obter as posições e velocidades das partículas do sistema como função do tempo. Em uma formulação discreta obtemos a evolução do sistema em instantes de tempo múltiplos de algum pequeno intervalo Δt, que em princípio pode ser tão curto quanto necessário. A melhor forma de compreender o esquema é através de um exemplo. Abaixo discutimos o movimento de um conjunto de partículas interagindo via potencial de Lennard-Jones. As ideias gerais serão discutidas para a interação entre duas partículas, contudo, o código desenvolvido pode conter um número indefinido de partículas.

4.1 Partículas interagindo via potencial Lennard-Jones

Vamos considerar duas partículas que interagem através de um potencial central, do tipo Lennard-Jones 12-6. O potencial $V(r)$ é descrito pela fórmula Eq. 3.37, onde $r = |\vec{r}_i - \vec{r}_j|$ é a distância entre as partículas nas posições $\vec{r}_i$ e $\vec{r}_j$. A figura Fig. 3.3 é um desenho esquemático do potencial.

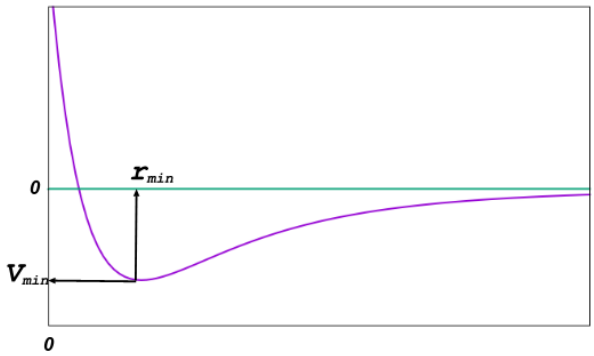

Fig. 3.3 Potencial Lennard-Jones (Eq. 3.37).

$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]. \qquad \textit{Eq. 3.37}$$

O mínimo do potencial é obtido fazendo $\frac{dV}{dr} = 0$. Resolvendo a equação resultante para r obtêm-se: $r_{min} = 2^{1/6}\sigma$ e $V(r_{min}) = -\varepsilon$. Este potencial é do tipo central, isto é, só depende da distância entre as partículas. Ele descreve bem interações entre átomos de gases nobres, mas pode ser uma aproximação razoável em outras circunstâncias. A parte atrativa deve-se a interações dipolares na molécula enquanto a parte repulsiva é uma aproximação ao princípio de exclusão de Pauli [**Lennard-Jones**]. A força entre duas partículas é obtida usando a definição $\vec{F} = -\frac{dV}{d\vec{r}}\,\hat{r}$.

Vamos resolver o problema de duas partículas interagindo via potencial de Lennard-Jones cuidadosamente, pois ele será um de nossos "toy model" ao longo do curso. O que aprendermos com ele será usado daqui em diante.

Suponha que o sistema é constituído por 2 partículas de massas iguais, m, localizadas segundo algum sistema de coordenadas nas posições $\{(x_1y_1z_1),(x_2y_2z_2)\}$ (Uma boa ideia é usar o sistema de coordenadas do centro de massas do sistema.). As equações de movimento são escritas como

$$\begin{aligned}
m\frac{d^2x_1}{dt^2} &= F^x{}_{21}\\
m\frac{d^2y_1}{dt^2} &= F^y{}_{21}\\
m\frac{d^2z_1}{dt^2} &= F^z{}_{21}\\
m\frac{d^2x_2}{dt^2} &= F^x{}_{12}\\
m\frac{d^2y_2}{dt^2} &= F^y{}_{12}\\
m\frac{d^2z_2}{dt^2} &= F^z{}_{12}
\end{aligned} \qquad \textit{Eq. 3.38}$$

As componentes da força são:

$$F^x{}_{ij} = -\frac{d}{dx}V,\ F^y{}_{ij} = -\frac{d}{dy}V\ e\ F^z{}_{ij} = -\frac{d}{dz}V\ . \qquad \textit{Eq. 3.39}$$

Escrevendo o potencial explicitamente em termos das componentes x, y e z

$$V(|r_i - r_j|) = 4\varepsilon\left[\left(\frac{\sigma}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)^{12} - \left(\frac{\sigma}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)^{6}\right], \quad \textit{Eq. 3.40}$$

obtemos expressões para as componentes da força.

$$\begin{cases} F^x = -\dfrac{d}{dx}V(r) = -\dfrac{\partial r}{\partial x}\dfrac{d}{dr}V(r) \ , \\ F^y = -\dfrac{d}{dy}V(r) = -\dfrac{\partial r}{\partial y}\dfrac{d}{dr}V(r) \ , \\ F^z = -\dfrac{d}{dz}V(r) = -\dfrac{\partial r}{\partial z}\dfrac{d}{dr}V(r) \ , \end{cases} \quad \textit{Eq. 3.41}$$

As derivadas são facilmente obtidas

$$\begin{cases} \dfrac{\partial r}{\partial x} = \dfrac{\partial}{\partial x}\sqrt{x^2 + y^2 + z^2} = \dfrac{x}{r} \ , \\ \dfrac{\partial r}{\partial y} = \dfrac{\partial}{\partial y}\sqrt{x^2 + y^2 + z^2} = \dfrac{y}{r} \ , \\ \dfrac{\partial r}{\partial z} = \dfrac{\partial}{\partial z}\sqrt{x^2 + y^2 + z^2} = \dfrac{z}{r} \ , \\ \dfrac{d}{dr}V(r) = \dfrac{24\varepsilon}{r}\left[\left(\dfrac{\sigma}{r}\right)^{12} - \left(\dfrac{\sigma}{r}\right)^{6}\right] \ . \end{cases} \quad \textit{Eq. 3.42}$$

Lembrando que as derivadas são feitas em relação às coordenadas da partícula que estamos calculando a força, isto é, $F^x{}_{ij} = -F^x{}_{ji}$. Isto não introduz qualquer problema, uma vez que $x = x_i - x_j$ e quando trocamos x_i por x_j a força muda de sinal, como esperado. Para escrever o algoritmo a regra é sempre pensar uma parte de cada vez.

1. **Quantos passos no tempo (Npassos)?**
2. **Qual o tamanho do passo (dt)? (Tempo total =Npassos*dt)**
3. **Quais as condições iniciais ($\{(x_{01}y_{01}z_{01}),(x_{02}y_{02}z_{02})$) ?**
4. **Devemos sempre escrever uma rotina separada para calcular a força?**
5. **E outros detalhes que ficarão claros daqui a pouco.**

Antes de começar a escrever o código sempre estude cuidadosamente a física do problema. Desta forma é mais simples detectar erros no código e descobrir onde eles estão. Faça gráficos do potencial e da força, veja se eles são condizentes com o que você espera. Observe que se dermos condições iniciais com as partículas muito próximas a força será muito grande e haverá uma repulsão tão forte que as partículas adquirirão rapidamente uma velocidade muito

grande. Por outro lado, se começarmos muito longe a força será muito fraca e a evolução no tempo muito lenta. Finalmente, se começarmos com as partículas na posição de equilíbrio não haverá dinâmica!

4.2 Algoritmo

Comece criando um arquivo para entrada de dados. Use algum nome que faça sentido, por exemplo **Dados_Lennard-Jones.dat**. Em geral não é conveniente fazer um arquivo contendo as condições iniciais. Se o sistema que se procura estudar tem mais que umas poucas partículas ele crescerá rapidamente de tamanho. Outra desvantagem, que veremos em breve, é o fato de necessitarmos fazer médias sobre várias configurações independentes do sistema. Para diminuir o trabalho crie uma rotina específica para definir as condições iniciais, posições e velocidades, das partículas. Escreva uma função ou uma subrotina para calcular o valor da força em uma dada posição. Escrever sub-rotinas e funções separadamente, para cada tarefa, ajuda bastante se você quiser mudar o potencial ou o método de integração sem ter que mexer em todo o código. Escreva uma rotina para movimentar as partículas e, finalmente, escreva seus resultados em um arquivo de saída, por exemplo **Dinamica_Lennard-Jones.dat**. Uma possível estrutura para seu código deve ficar assim:

1. **Ler dados**
2. **Definir Condições iniciais**
3. **Cálculo das forças**
4. **Loop de Movimento**
5. **Fim do Loop**

Um último lembrete. Para as duas equações diferenciais descrevendo o movimento do sistema temos 6 equações de segunda ordem. Contudo, nossos métodos são aplicáveis a equações de primeira ordem. A maneira de reduzi-las a equações de primeira ordem é reescrevê-las como

$$m\frac{dv_1^x}{dt} = F^x{}_{21}$$

$$v_1^x = \frac{dx_1}{dt}$$

$$m\frac{dv_1^y}{dt} = F^y{}_{21}$$

$$v_1^y = \frac{dy_1}{dt}$$

$$m\frac{dv_1^z}{dt} = F^z{}_{21}$$

$$v_1^z = \frac{dz_1}{dt}$$

Com equações similares para a partícula 2.

Código

Abaixo vai um código para a solução numérica do problema contendo N_p partículas no sistema. O código é completamente funcional e foi escrito de forma que seja simples de entender, mas, certamente, não é um código otimizado. Como exercício você pode pensar em formas de melhorá-lo.

Código 4.1

Versão FORTRAN

```
!
!   Resolve numericamente as equacoes de movimento para duas
!   particulas interagindo via potencial de Lennard-Jone
!
!   O codigo foi escrito intencionalmente para ser claro e
!   nao eficiente. Muitas mudancas podem ser feitas que
!   aumentam sua eficiencia (speed up) em pelo menos 100x!!!
!
!   O metodo para solucao das equacoes discretas eh Euler,
!   sabidamente inadequado.
!
!
!=============================================================
!   Define parametros da simulacao
!   dt     : passo no tempo
!   eps    : parametro Lennard-Jones
!   sigma  : parametro Lennard-Jones
!   NtTotal: Numero total de passos no tempo
!   Np     : Numero de particulas
!=============================================================
!
Module Parametros
Implicit None
Real*8          :: dt
Real*8          :: eps,sigma
Integer         :: NtTotal,Np
End Module Parametros
!
!=============================================================
!   Vetores usados na simulacao
!   x,y,z   : Teem dimensao Np. x(i),y(i),z(i) contem as posicoes
!             da particula i=1...Np
!   vx,vy,vz: Sao as respectivas velocidades
!=============================================================
!
Module Vetores
Implicit None
Real*8,Dimension(:),Allocatable :: x,y,z
Real*8,Dimension(:),Allocatable :: vx,vy,vz

End Module Vetores
!
!=============================================================
!   Rotina para entrada de dados lidos no arquivo
!             Input_Lennard_Jones.dat
!   O comando "ALLOCATE" reserva espaço (dinamicamente)
!   para os respectivos vetores.
!
!=============================================================
!
Subroutine Data_Input()
Use Parametros
Use Vetores
Implicit none
```

```
Open(1,File="Input_Lennard_Jones.dat")
Read(1,*)NtTotal
Read(1,*)dt
Read(1,*)Np
Read(1,*)eps,sigma
Allocate(x(Np),y(Np),z(Np),vx(Np),vy(Np),vz(Np))
End Subroutine Data_input
!
!============================================================
!   Cria vetores de posicoes e velocidades iniciais para
!   a dinamica. Neste caso escolhi posicoes aleatorias para
!   a particula 1 e para a particula dois dei um incremento
!   2*sigma x aleatorio em cada direcao. Isto evita que as
!   posicoes sejam muito proximas da particula 1. Se isto
!   acontecer a forca de repulsao fica muito grande e o
!   algoritmo "explode"!
!
!   O comando Call Random_Number(r) aloca Np numeros
!   aleatorios em r, que serao usados para criar as posicoes
!   e velocidades iniciais
!
!============================================================
!
Subroutine Initial_Conditions()
Use Parametros
Use vetores
Implicit None
Real*8,Dimension(Np)  :: rx,ry,rz,rvx,rvy,rvz
Integer               :: ip
Call Random_Number(rx)
Call Random_Number(ry)
Call Random_Number(rz)
Call Random_Number(rvx)
Call Random_Number(rvy)
Call Random_Number(rvz)

    x(1) = sigma*rx(1)
        y(1) = sigma*ry(1)
            z(1) = sigma*rz(1)
            vx(1) = sigma*rvx(1)
        vy(1) = sigma*rvy(1)
    vz(1) = sigma*rvz(1)

    Loop_Condicoes_Iniciais: Do ip = 2 , Np

     x(ip) = x(ip-1) + 2.0D0*sigma*rx(ip)
        y(ip) = y(ip-1) + 2.0D0*sigma*ry(ip)
            z(ip) = z(ip-1) + 2.0D0*sigma*rz(ip)
            vx(ip) = sigma*rvx(ip)
        vy(ip) = sigma*rvy(ip)
    vz(ip) = sigma*rvz(ip)

    End Do Loop_Condicoes_Iniciais


End Subroutine Initial_Conditions
!
!============================================================
!   A subrotina Move, usa o algoritmo de Euler para dar
!   1 passo no tempo. Ela é chamada NtTotal de vezes.
!
!============================================================
!
Subroutine Move()
Use Parametros
Use Vetores
Implicit None
Real*8          :: Fx,Fy,Fz
```

```
Integer         :: ip



  Loop_Particulas: Do ip = 1 , Np

  vx(ip) = vx(ip) + dt*Fx(ip)
    vy(ip) = vy(ip) + dt*Fy(ip)
        vz(ip) = vz(ip) + dt*Fz(ip)

  x(ip) = x(ip) + dt*vx(ip)
    y(ip) = y(ip) + dt*vy(ip)
      z(ip) = z(ip) + dt*vz(ip)

  End Do Loop_Particulas

End Subroutine Move
!
!=============================================================
!   Fx, Fy e Fz, calculam as forcas nas direcoes x, y, e z
!
!=============================================================
!
Real*8 Function Fx(i)
Use Parametros
Use Vetores
Implicit None
Real*8          :: aux,r
Integer         :: i,j

aux = 0.0D0
Do j = 1 , Np
If(i .ne. j)Then
r = DSQRT((x(i)-x(j))**2 + (y(i)-y(j))**2 + (z(i)-z(j))**2)

aux = aux - 24.0D0*eps*(x(j)-x(i)) *                              &
                       ((sigma/r)**12 - (sigma/r)**6)/(r**2)
Endif
End Do
Fx=aux

End Function Fx
!
!=============================================================
!
Real*8 Function Fy(i)
Use Parametros
Use Vetores
Implicit None
Real*8          :: aux,r
Integer         :: i,j

aux = 0.0D0
Do j = 1 , Np
If(i .ne. j)Then
r = DSQRT( (x(i)-x(j))**2 + (y(i)-y(j))**2 +  (z(i)-z(j))**2)
aux = aux - 24.0D0*eps*(y(j)-y(i))*((sigma/r)**12 - (sigma/r)**6)/(r**2)
Endif
End Do
Fy = aux

End Function Fy
!
!=============================================================
!
Real*8 Function Fz(i)
Use Parametros
Use Vetores
```

```
Implicit None
Real*8          :: aux,r
Integer         :: i,j

aux = 0.0D0
Do j = 1 , Np
If(i .ne. j)Then
r = DSQRT((x(i)-x(j))**2 + (y(i)-y(j))**2 + (z(i)-z(j))**2)
aux = aux - 24.0D0*eps*(z(j)-z(i))*((sigma/r)**12 - (sigma/r)**6)/(r**2)
Endif
End Do
Fz = aux

End Function Fz
!
!============================================================
!   programa principal.
!
!   As chamadas de escrita (Write) escrevem em um arquivo
!   as posicoes das particulas em cada instante de tempo.
!   O arquivo esta pronto para ser usado em um programa de
!   visualizacao chamado OVITO.
!   (https://www.ovito.org/about/)
!
!============================================================
!
Program Lennard_Jones_1
Use Parametros
Use Vetores
Implicit None
Integer         :: itime,ip

Call Data_Input()
Write(*,*)NtTotal,dt,Np
Write(*,*)eps,sigma

Call Initial_Conditions()

Write(3,*)Np
Write(3,*)"Posicoes das particulas"
Write(3,*)'Cu',x(1),y(1),z(1)
Write(3,*)'Au',x(2),y(2),z(2)

Do itime = 1 , NtTotal
Call Move()
Write(3,*)Np
Write(3,*)"Posicoes das particulas"
Do ip = 1, Np
If(ip .eq. 1)Write(3,*)'Cu',x(ip),y(ip),z(ip)
If(ip .eq. 2)Write(3,*)'Au',x(ip),y(ip),z(ip)
End Do
End Do

End Program
```

Entrada de dados

Arquivo: Input_Lennard_Jones.dat

```
10000
1.0D-3
2
10.0D0   1.0D0
```

```
#   NtTotal
#   dt
#   Np          (Numero de Particulas)

#   Read(1,*)NtTotal
#   Read(1,*)dt
#   Read(1,*)Np
#   Read(1,*)eps,sigma
```