Anexo I. Código

En este anexo, vamos a mostrar parte del código desarrollado y comentar alguno de sus aspectos destacados. El lenguaje de programación escogido ha sido Python por su sencillez a la hora de escribir código pero también con un objetivo formativo para aprender un lenguaje que hoy en día es ampliamente utilizado en una gran variedad de ámbitos desde el análisis de datos hasta la simulación numérica. En cuanto a los detalles técnicos, la mayor parte de los programas fueron simulados en el Cluster de ordenadores del Departamento de Física de la Materia Condensada en la Universidad de Zaragoza.

A continuación, presentamos el código de cálculo de la tasa de escape Kramers íntegro, ya que, el resto de programas se consiguen variando segmentos de este.

```
import numpy as np ##importamos la librería matemática

## parámetros del sistema adimesional
gamma=10.   ##Viscosidad. Sistema sobreamortiguado
T=0.05  ##Temperatura
Fc=1.5  ##Fuerza crítica, desaparecen máximo y mínimo

#Parámetros para ir avanzando por la fuerza
N_pasos=20
Fmax=1.5
d_paso=Fmax/N_pasos

#Paso de tiempo y número de partículas
dt=0.01
N_particulas=100

##Posicion y velocidad inicial
x0=-0.5
v0=np.sqrt(T) ## La correspondiente al baño térmico

##parámetros para la introducción del ruido
D=2*gamma*T ##Coeficiente de difusión
eps=(dt*D)**0.5 ##Para el Runge-Kutta
##definimos listas para guardar resultados
time=[]
##definimos el nombre del archivo para guardar resultados
r1=str(T)
r2=str(N_pasos)
resultados=str("TE_"+r1+"_"+r2+".txt")

##Abrimos el fichero de reslultados
f=open(resultados,'w')
f.close()
f=open(resultados,'a')

## definimos G(energia libre)
AG=1

##G0 cúbico lineal:
def dG0(xp,Fp):
    return AG*(3/2-(6*xp**2))-Fp
```

```
"""
Definimos ahora las funciones del R-K considerando
dx/dt=f1(x,v) --> f1(x,v)=v
dv/dt=f2(x,v)+num_aleatorio
f2(x,v)= -gamma*v-dG(x)
para nuestro caso de un potencial de energía libre G y una fuerza F
del muelle lineal
"""
#Estas funciones son las que determinan la dinámica simulada con el
Runge-Kutta de segundo orden.

def f1(xp,vp):
    return vp

def f2(xp,vp,Fp):
    return (-gamma*vp-dG0(xp,Fp))

## programa principal

#comenzamos el R-K
xnw=x0
vnw=v0
p=0
F=0
##Bucle en fuerzas
for t in range(0,N_pasos):
    F=d_paso+F
    print(t,F)
    #Bucle en partículas
    for n in range(0,N_particulas):
        p=0 ##Contador de pasos que realiza cada partícula para salir
        xnw=x0
        vnw=v0
        #Bucle en tiempo para cada partícula
        while xnw < 2:
            num_rand=np.random.randn()
            g11=f1(xnw,vnw+eps*num_rand)
            g12=f2(xnw,vnw+eps*num_rand,F)
            g21=f1(xnw+dt*g11,vnw+dt*g12)
            g22=f2(xnw+g11*dt,vnw+g12*dt,F)
            xnw+=dt/2*(g11+g21)
            vnw+=dt/2*(g12+g22)+eps*num_rand
            p=p+1

        time.append(dt*p)

    ##Calculamos la media y añadimos a lista
    mean_t=0
    for k in range(t*N_particulas,N_particulas*(t+1)-1):
        mean_t+=time[k]

    mean_t=mean_t/N_particulas
```

```
    tescp=1./mean_t
        ##Calculamos la desviación
    dev_t=0
    for k in range(0,N_particulas):
        dev_t=(time[k]-mean_t)**2
    dev_t=np.sqrt(dev_t/N_particulas)
    desv=dev_t/mean_t**(2)
    s1=str(F)
    s2=str(tescp)
    s3=str(desv)
    s=str(s1 +"\t"+s2+"\t"+s3+"\n")
    f.write(s)
##cerramos fichero
f.close()
```

Las secciones más importantes del código están marcadas en negrita y son, la definición del potencial, el bucle en el que se desarrolla el R-K y la forma de guardar los resultados. Así pues, vamos a repasar estos puntos para los diferentes programas utilizados.

Programa para el cálculo de la fuerza de ruptura media en función de T en el *consta trate mode*.

En este caso, el potencial es el mismo así que no merece la pena repetirlo. El segmento del bucle, así como el código utilizado para recoger los datos lo presento a continuación.

```
#Bucle en a, a=dF/dt
for ia in range(0,N_a):
    a=a0*(af/a0)**(ia/N_a)    #af es la dF/dt final y a0 la inicial
    Fmed=0
    #Bucle en partículas
    for n in range(0,N_particulas):
        p=0 ##Contador de pasos que realiza cada partícula para salir
        xnw=x0
        vnw=v0
        #Bucle en tiempo para cada partícula
            while xnw < 2: #Consideramos que para x=2 la partícula ya
      ha salido
            F=a*p*dt
            num_rand=np.random.randn()
            g11=f1(xnw,vnw+eps*num_rand)
            g12=f2(xnw,vnw+eps*num_rand,F)
            g21=f1(xnw+dt*g11,vnw+dt*g12)
            g22=f2(xnw+g11*dt,vnw+g12*dt,F)
```

```
            xnw+=dt/2*(g11+g21)
            vnw+=dt/2*(g12+g22)+eps*num_rand
            p=p+1
        F=a*p*dt  #multiplicamos el número de pasos, p, por el tiempo
en cada paso dt
        Fmed+=F   #almacenamos el dato

    Fmedio=Fmed/N_particulas #calculamos la media
```

Programa para el cálculo de la fuerza de ruptura media en función de T en el *constant rate mode* cuando el dispositivo con el que tiramos es un muelle.

En este caso, la única parte que cambia del código es la de la definición del potencial. Además, el bucle de la sección anterior se incluye dentro de otro bucle que recorre diferentes valores de k pero no merece la pena repetir todo el código para un cambio tan pequeño.

```
##G0 cubico:
def dG0(xp,kp,ap,t):
    return AG*(3/2-(6*xp**2))+kp*(xp-ap/kp*t)
```

Además, calculamos la fuerza de salida como en el caso del *constant rate mode* y después lo corregimos dividiendo la fuerza media obtenida por χ recuperándose la fuerza de ruptura media correcta descrita en la memoria.

Programa para el cálculo de la fuerza de ruptura media en función de T en el potencial de Morse.

Para este problema, utilizamos el mismo código que en el *constat rate mode* pero modificando el potencial de la siguiente forma:

```
##G0 morse: 3(1-e^(-x-0.5))^2 D=k=1:
def dG0(xp,F):
    return 6*np.exp(-xp-0.5)*(1-np.exp(-xp-0.5))-F
```

Tanto el bucle, como la forma de calcular la fuerza de ruptura no varían.

Así, quedan comentados los segmentos más importantes del código. Aunque también se desarrolló el código para el caso en el que el dispositivo con el que se estira es un muelle no lineal que sigue el modelo de WLC, al haber un error en la parte de recuperar los resultados y ser estos erróneos no merece ser presentado. Sí que cabe destacar que este código era algo más complicado ya que es necesario resolver una ecuación implícita para resolver los parámetros dinámicos del sistema. Por otro lado, se ve como, en general, el código no plantea mayores complicaciones que la resolución de una ecuación diferencial estocástica.

Anexo II. Desarrollo de la teoría de Bell-Evans

En este anexo vamos a profundizar en la teoría fenomenológica de Bell-Evans y probar cómo se pasa de la ecuación de Bell [1] para la tasa de ruptura en función de la fuerza externa aplicada.

$$k(t) = k_0 e^{\frac{F(t)x^\dagger}{k_B T}} \tag{1}$$

Hasta la expresión para la fuerza de ruptura media. En (1), k_0 es la tasa de ruptura característica del sistema sin perturbar, $x^\dagger$ es la posición de la transición (distancia entre el mínimo y el máximo). Consideramos $S(t)$ la probabilidad de supervivencia, es decir, la probabilidad de que para cierto tiempo t la ruptura no haya ocurrido y asumimos que satisface:

$$\frac{dS(t)}{dt} = \dot{S}(t) = -k(t)S(t) \tag{2}$$

y, por tanto:

$$S(t) = exp\left[-\int_0^t k(t')dt'\right] \tag{3}$$

donde $k(t)$ viene dado por la expresión de Bell (1). La distribución de probabilidad de los tiempos de vida t^* es $-\dot{S}(t^*)dt^*$ lo que significa que el tiempo de vida medio tiene la forma:

$$\bar{t^*} = -\int_0^\infty t\dot{S}(t)dt = \int_0^\infty S(t)dt \tag{4}$$

La forma en la que se relaciona la distribución de probabilidad de las fuerzas de ruptura con la distribución de probabilidad de los tiempos de vida es:

$$p(F)dF = -\dot{S}(t^*)dt^* \tag{5}$$

Las limitaciones principales a la expresión de Bell son, que solo es válida para fuerzas pequeñas y que no considera la fluctuación de la coordenada molecular bajo la influencia de los potenciales moleculares y de la perturbación combinados. En este sentido, aunque se utiliza mucho, su rango de aplicación es bastante reducido.

Para el modo *constan rate* en el que la fuerza aplicada $F(t)$ aumenta linealmente con el tiempo $F(t) = \kappa V t$, siendo κ la constante del muelle del dispositivo con el que se estira y V la velocidad con la que se estira. Entonces:

$$S(t) = exp\left[-\int_0^t k_0 e^{\frac{\kappa t x^\dagger}{k_B T}}\right] = exp\left[-\frac{k_0}{\kappa V x^\dagger}(e^{\frac{\kappa V x^\dagger t}{k_B T}} - 1)\right] \tag{6}$$

Así según (5) tenemos:

$$p(F) = \frac{k_0}{\kappa V k_B T} exp\left[\frac{Fx^\dagger}{k_B T} - \frac{k_0}{\kappa V x^\dagger}(e^{\frac{Fx^\dagger}{k_B T}} - 1)\right] \tag{7}$$

y la fuerza de ruptura queda:

$$< F >= \kappa V k_B T \int_0^\infty S(t)dt = \frac{k_B T}{x^\dagger} e^{\frac{k_0}{\kappa V x^\dagger}} E_1\left(\frac{k_0}{\kappa V x^\dagger}\right) \tag{8}$$

Donde $E_1 = \int_x^\infty e^{-t}t^{-1}dt$. Para velocidades altas, esta integral se puede aproximar quedando, finalmente, la expresión del modelo de Bell-Evans:

$$< F > \approx \frac{k_B T}{x^\dagger} log(\frac{\kappa V x^\dagger e^{-\gamma}}{k_0}) \tag{9}$$

donde $\gamma = 0{,}5772...$ es la constante de Euler-Mascheroni de la que ya hemos hablado en el trabajo. Así queda vista la relación entre las dos expresiones.

Referencias

[1] G. I. Bell. Models for the specific adhesion of cells to cells. In: Science 200, 618 (1978).