Colisión Elástica Hidrógeno-Kryptón

Darío Mitnik (Modificado del trabajo final de Germán Chiarelli)

Basado en Computational Physics, de J.M. Thijssen (Cap. II)

In [2]:

```
from numpy import diag,  linspace, array ,arange
from matplotlib.pyplot import axhline, xlabel, ylabel, plot, axis, \
                                figure, title, show
%matplotlib inline
import time
from __future__ import division
```

Potencial de Lennard-Jones

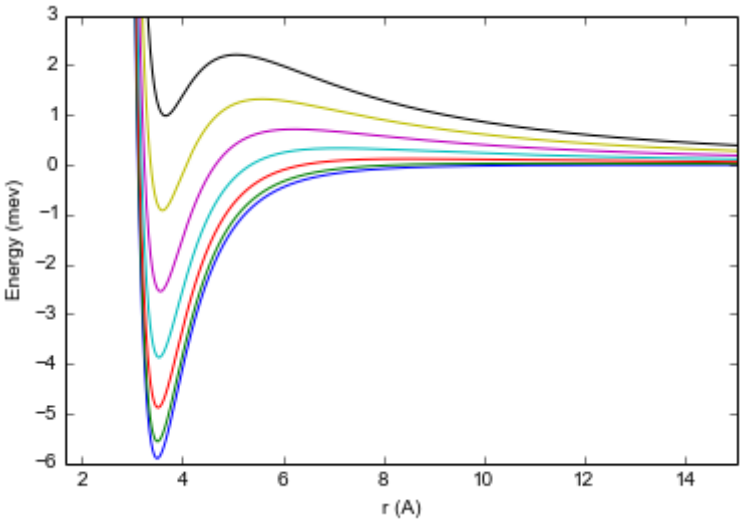
$$V_{LJ} = \epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - 2 \left(\frac{\sigma}{r} \right)^{6} \right]$$

In [1]:

```
# Unidades meV y A
Dae = 931.494061e6*1e3 #mev/c^2
m=(83.798)*Dae  #e o kg
hbc=0.19732697*1e7 #mev*A
h2m=(hbc**2)/(2*m)*83.798

eps=5.9 #mev
rho=3.5 #A ó 1e-10 m
```

In [3]:

```
# Ploteo de Potenciales

rmin=0.5*rho
step=0.01
xmin=rmin-step
xmax=30
x=arange(xmin,xmax,step)+step
nsize = len(x)


for l in range(7):
    Vl = eps*( (rho/x)**12 - 2*(rho/x)**6 ) + h2m*l*(l+1)/(x**2)
    plot(x,Vl)
axis([xmin-0.08,xmax-15,-6,3])
xlabel("r (A)")
ylabel("Energy (mev)")
show()
```



Soluciones Numéricas

Ecuación de Schrödinger:

$$-\frac{1}{2}\frac{\partial^2\varphi(x)}{\partial x^2}+V(x)\,\varphi(x)=E\,\varphi(x)\;\frac{\partial^2\varphi(x)}{\partial x^2}-2\left[V(x)-E\right]\,\varphi(x)\;=0\;\Rightarrow\frac{\partial^2\varphi(x)}{\partial x^2}\;=2$$

$$\mathbf{y}\equiv[y_0,y_1]=\left[\varphi(x),\frac{\partial\varphi(x)}{\partial x}\right]$$

$$\frac{\partial\mathbf{y}}{\partial x}=\frac{\partial}{\partial x}\left[\varphi(x),\frac{\partial\varphi(x)}{\partial x}\right]=\left[\frac{\partial\varphi(x)}{\partial x},\frac{\partial^2\varphi(x)}{\partial x^2}\right]=\quad=[y_1,2\,(V(x)-E)\,y_0]\equiv[y_1,g(x)y$$

In [4]:

```
import numpy as np
import scipy as sp
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import math as mt
```

In [5]:

```
def Vpot(x,l):
    return ( eps*( (rho/x)**12 - 2*(rho/x)**6 ) + h2m*l*(l+1)/(x**2) )
```

In [6]:

```
def g(y, x, E):
    return [y[1],(1/h2m)*(Vpot(x,l)-E)*y[0]]
```

In [7]:

```
#solución en el origen

a=6.12
C=(eps*a*(1./25.))**(0.5)
def u(r):
    u= np.exp(-C/(r**5))
    return u

def du(r):
    du= C * 5 / (r**6)* np.exp(-C / (r**5))
    return du

initialY = u(rmin), du(rmin)
```

In [8]:

```
# Solución en un rango de energías

nE = 200   # energías
nl= 9      # l quantum numbers

Uk=np.zeros((nl,nE,len(x)))
Emin=0.25
Emax=4.5
E = np.linspace(Emin,Emax,nE)
for l in range(nl):
    for i in range (nE):
        sol = odeint(g,initialY,x,(E[i],))
        Uk[l,i,:]=sol[:,0]
```

In [9]:

```
k=range(nE)
for j in range(nE):
    k[j]= np.sqrt( E[j]/h2m )
```

In [10]:

```
# Funciones de Bessel

from numpy import sin, cos ,tan

jn=np.zeros((nl,nE,len(x)))
jv=np.zeros((nl,nE,len(x)))
for j in range(nE):
    for i in range (len(x)):
        jv[0,j,i]=sin(k[j]*x[i])/(k[j]*x[i])    # Bessel j0
        jn[0,j,i]=-cos(k[j]*x[i])/(k[j]*x[i])   # Neumann n0
        jv[1,j,i]=sin(k[j]*x[i])/((k[j]*x[i])**2)-cos(k[j]*x[i])/(k[j]*x[i])
        jn[1,j,i]=-cos(k[j]*x[i])/((k[j]*x[i])**2)-sin(k[j]*x[i])/(k[j]*x[i])

# Recursion: S_l+1 = 2l+1 / x  * S_l  - S_l-1
for l in range (1,nl-1):
    for j in range(nE):
        for i in range (len(x)):
            jv[l+1,j,i]=(2*l+1)/(k[j]*x[i]) * jv[l,j,i] - jv[l-1,j,i]
            jn[l+1,j,i]=(2*l+1)/(k[j]*x[i]) * jn[l,j,i] - jn[l-1,j,i]
```

In [11]:

```
# Se escogen r1 y r2 asintóticos

rmax=5*rho
i1  = int( len(x)*0.95  )
i2  = int( len(x)*0.98 )
x[i1],x[i2],np.pi/k[0] , rmax
```

Out[11]:

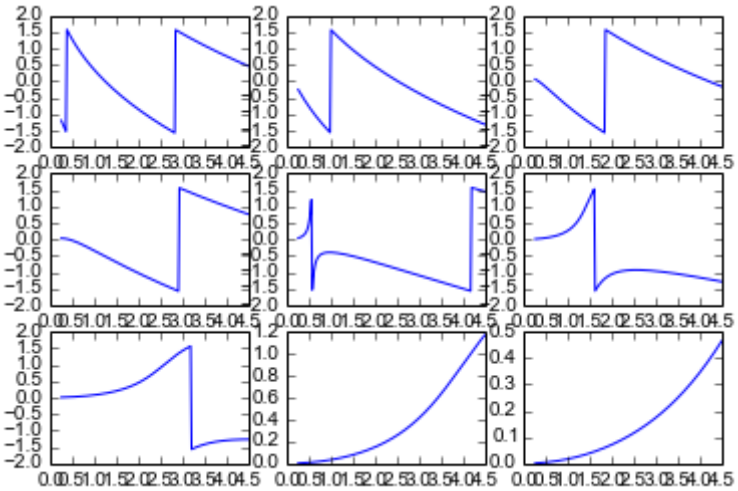
```
(28.590000000000025, 29.440000000000026, 9.0836684157226539, 17.5)
```

In [12]:

```
# Phase Shift

tandel=np.zeros((nl,nE))
nom=np.zeros((nl,nE))
den=np.zeros((nl,nE))
K=np.zeros((nl,nE))

for l in range(nl):
    for j in range(nE):
        K[l,j]=x[i1]*Uk[l,j,i2]/(x[i2]*Uk[l,j,i1])
        nom[l,j]=((K[l,j]* jv[l,j,i1]) - jv[l,j,i2])
        den[l,j]=((K[l,j]* jn[l,j,i1]) - jn[l,j,i2])
        tandel[l,j] = nom[l,j]/den[l,j]
```

In [13]:

```
delta=np.zeros((nl,nE))
delta2=np.zeros((nl,nE))
for l in range(nl):
    for j in range(nE):
        delta[l,j]=mt.atan(tandel[l,j])
    plt.subplot(3,3, l+1)
    plt.plot(E,delta[l,:],'-')
```



Sección Eficaz Total

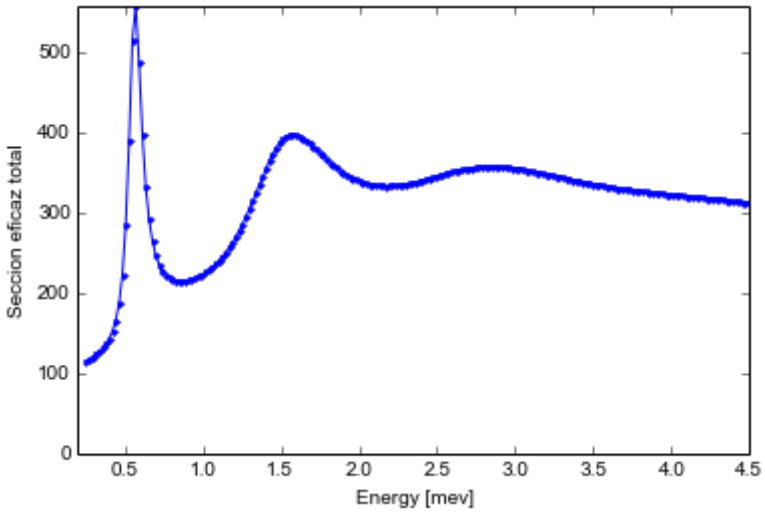
$$\sigma_{tot} = \frac{4\pi}{k^2} \sum (2l+1) \sin^2 \delta_l$$

In [ ]:

```
sindl=np.zeros((nl,nE))
suma=np.zeros(nE)
sigmatot=range(nE)
D=range(nE)

for j in range(nE):
    D[j]=4*np.pi/(k[j]**2)
    for l in range(nl):
        # Cálculo de Sección Eficaz parcial
        sindl[l,j]=sin(delta[l,j])
        suma[j]= suma[j] + (2*l+1) * (sindl[l,j]**2)
    # Sección Eficaz Total
    sigmatot[j]=D[j]*suma[j]
```

In [17]:

```
plt.figure(2)
plt.plot(E,sigmatot,'.-')
#plt.xscale('log')
ylabel("Seccion eficaz total ")
xlabel("Energy [mev]")
plt.xlim(0.2,max(E))
plt.ylim(0,max(sigmatot)+1)
show()
```



In [ ]:

```

```