

DFT Helio con Potencial de Exchange (LDA) y Correlación (Perdew-Zunger)

Alejandra Mendez

In [1]:

```
%matplotlib inline
```

In [2]:

```
from numpy import sqrt, exp, linspace, zeros, identity, array, diag, c_, pi, log
from math import factorial
from numpy.linalg import eigh
from scipy.integrate import quad
from scipy.special import hyp1f1
from matplotlib.pyplot import plot, axis, title, xlabel, ylabel, show,\
                                legend, subplot
```

In []:

In [3]:

```
# Normalización de las funciones
def Normalize(U,x):
    h = x[1]-x[0] # assume uniformly spaced points
    n = len(x)
    for j in range(0,n-1):
        suma = 0.0
        for i in range(0,n-1):
            suma = suma + U[i,j]**2
        suma = suma*h
        rnorm = 1/sqrt(suma)
# Normalization
        rsign = 1
        if U[1,j] < 0:
            rsign = -1
        rnorm = rnorm * rsign
        for i in range(0,n):
            U[i,j] = U[i,j]*rnorm
    return U
```

In [4]:

```
def Laplacian(x):
    h = x[1]-x[0] # assume uniformly spaced points
    n = len(x)
    M = -2*identity(n,'d')
    for i in range(1,n):
        M[i,i-1] = M[i-1,i] = 1
    return M/h**2
```

In [5]:

```
# Defino las dimensiones de los arrays a utilizar
nsize = 1000
rmax = 10.0
h = rmax/nsize
rmin = h
```

In [6]:

```
# Defino las matrices y vectores necesarios
r = linspace(rmin, rmax, nsize)
T = array([nsize, nsize])
Vs = array([nsize, nsize])
U = zeros([nsize, nsize])
H = array([nsize, nsize])
E = array([nsize])
```

In [7]:

```
Upp = zeros(nsize)
Up = zeros(nsize)
Zh = zeros(nsize)
Vh = zeros(nsize)
den = zeros(nsize)
Vx = zeros(nsize)
Vc = zeros(nsize)
```

In [8]:

```
# Defino el sistema a resolver: Helio en el estado fundamental
Z = 2
l = 0
```

La ecuación de Kohn-Sham está dada por

$$\left[-\frac{1}{2} \nabla^2 + V_s(n|\mathbf{r}) \right] \phi_i = \epsilon_i \phi_i$$

donde

$$V_s(n|\mathbf{r}) = V_{\text{ext}}(r) + V_{\text{H}}(n|\mathbf{r}) + V_x(n|\mathbf{r}) + V_c(n|\mathbf{r})$$

El potencial de Hartree está dado por:

$$V_{\text{H}}(r) = \int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}$$

donde $n = \sum_i n_i = \sum_i |\phi_i|^2$ es la densidad total.

$$\Rightarrow \nabla^2 V_{\text{H}}(n|\mathbf{r}) = -4\pi n(\mathbf{r})$$

Debido a la simetría radial de la densidad,

$$Z_{\text{H}}(r) = rV_{\text{H}}(r) \quad \Rightarrow \quad \frac{d^2 Z_{\text{H}}(r)}{dr^2} = -4\pi r n(r)$$

Resuelvo usando el algoritmo de Verlet, e imponiendo dos condiciones de borde

$$Z_H(0) = 0 \quad y \quad Z_H(r_{\max}) = q = 2$$

El potencial de exchange está dado por la Local Density Approximation (LDA) formulada por Slater:

$$V_x(r) = A_x n(r)^{1/3} \quad : \quad A_x = -(3/\pi)^{1/3}$$

El potencial de correlación de Perdew-Zunger (parametrización de Ceperley-Alder's Quantum Monte Carlo calculations)

$$r_s = \left(\frac{3}{4\pi n(r)} \right)^{1/3}$$

$$\varepsilon_c(r) = \frac{\gamma}{1 + \beta_1 \sqrt{r_s} + \beta_2 r_s}$$

$$r_s \geq 1 \quad : \quad V_c(r) = \varepsilon_c(r) \frac{1 + 7/6\beta_1 \sqrt{r_s} + \beta_2 r_s}{1.0 + \beta_1 \sqrt{r_s} + \beta_2 r_s}$$

$$r_s < 1 \quad : \quad V_c(r) = A \log r_s + B - A/3 + \frac{2}{3} C r_s \log r_s + (2D - C)r_s/3$$

In [9]:

```
# Defino la densidad TOTAL
def densidad():
    noc = 2.0      # nro de ocupacion = 2
    ang = 1.0/(4.0*pi)
    for i in range(nsize):
        den[i] = noc*(U[i,0]/r[i])**2*ang
    return den
```

In [10]:

```
# Defino el potencial de Hartree
# Uso el índice j (nro de iteracion) para no incluir el potencial Vh
# en la primera iteración
def VHartree(j):
    if (j>0):
        # 1ra condicion de borde
        Up[0] = 0.0
        Zh[0] = 0.0
        Zh[1] = h
        for i in range(nsize):
            Upp[i] = -4.0*pi*den[i]*r[i]
        for i in range(nsize-1):
            Up[i+1] = Up[i] + 0.5*(Upp[i] + Upp[i+1])*h
        # 2da condicion de borde
        alfa = -Up[nsize-1]
        for i in range(nsize-1):
            Zh[i+1] = Zh[i] + Up[i]*h + 0.5*Upp[i]*h**2
        for i in range(nsize):
            Zh[i] = Zh[i] + alfa*r[i]
        for i in range(nsize):
            Vh[i] = Zh[i]/r[i]
    return Vh
```

In [11]:

```
# Defino el potencial de Exchange: Local Density Approximation
# Nuevamente uso j para no incluir el potencial en la primera iteración
def Vexchange(j):
    if (j>0):
        Ax = -(3./pi)**(1./3.)
        for i in range(nsize):
            Vx[i] = Ax*(den[i])** (1./3.)
    return Vx
```

In [12]:

```
# Defino el potencial de Correlación:
# Perdew-Zunger parametrization of Ceperley-Alder Approximation (Quantum Monte Carl
# Nuevamente uso j para no incluir el potencial en la primera iteración
def Vcorrelation(j):
    rs = zeros(nsize)
    ec = zeros(nsize)
    gamma = -0.1423
    beta1 = 1.0529
    beta2 = 0.3334
    A = 0.0311
    B = -0.048
    C = 0.0020
    D = -0.0116
    if (j>0):
        for i in range(nsize):
            rs[i] = (3.0/(4.0*pi*den[i]))**(1./3.)
            if (rs[i]>=1.0):
                deno = 1.0 + beta1*sqrt(rs[i]) + beta2*rs[i]
                ec[i] = gamma/deno
                num = 1.0 + 7.0/6.0*beta1*sqrt(rs[i]) + beta2*rs[i]
                Vc[i] = ec[i]*num/deno
            if (rs[i]<1.0):
                Vc[i] = A*log(rs[i]) + B - A/3.0 + (2.0/3.0)*C*rs[i]*log(rs[i]) + (
    return Vc
```

In [13]:

```
# Inicializo algunas variables utiles en la iteración
# j : nro de iteraciones
j = 0
Eorb = []
Etotal = []
u0 = zeros((nsize,1))
```

Empieza loop

In [114]:

```
# Calculo la densidad y el potencial de Hartree
densidad()
VHartree(j)
Vexchange(j)
Vcorrelation(j)

# Construyo el operador T y el operador Vs (potencial efectivo)
T = (-0.5)*Laplacian(r)
Vs = -Z/r + l*(l+1)/(2*r**2) + Vh + Vx + Vc
```

In [115]:

```
# Construyo el Hamiltoniano del sistema
H = T + diag(Vs)
```

In [116]:

```
# Calculo autovalores (E) y autovectores (U)
E,U = eigh(H)
j = j + 1

# Normalization
U = Normalizate(U,r)
```

In [117]:

```
# Adjunto los resultados para ver los resultados de cada iteración
# Eorb : energías
# u0 : funciones radiales reducidas
Eorb.append(E[0])
Eorb
```

Out[117]:

```
[-1.9998000399911799,
 -0.37786578797247494,
 -0.66356678291625482,
 -0.52999650040237312,
 -0.58459683623164993,
 -0.56059873665716498,
 -0.57084596046395664,
 -0.56641400498086891,
 -0.56832039868635931,
 -0.56749842591616506]
```

In [118]:

```
u0 = c_[u0,U[:,0]]
u0
```

Out[118]:

```
array([[ 0.00000000e+00,  5.54429408e-02,  4.17356874e-02, ...,
         4.61269756e-02,  4.61662087e-02,  4.61493068e-02],
       [ 0.00000000e+00,  1.08690339e-01,  8.18237253e-02, ...,
         9.04301963e-02,  9.05070881e-02,  9.04739626e-02],
       [ 0.00000000e+00,  1.59807402e-01,  1.20318328e-01, ...,
         1.32966960e-01,  1.33079962e-01,  1.33031280e-01],
       ...,
       [ 0.00000000e+00,  1.30455973e-08,  2.16163472e-05, ...,
         5.28186334e-06,  5.21206345e-06,  5.24203264e-06],
       [ 0.00000000e+00,  8.69445713e-09,  1.44099907e-05, ...,
         3.52090987e-06,  3.47437990e-06,  3.49435795e-06],
       [ 0.00000000e+00,  4.34644629e-09,  7.20472309e-06, ...,
         1.76035523e-06,  1.73709124e-06,  1.74707983e-06]])
```

La energía total en este esquema está dada por:

$$E = \sum_i \epsilon_i - E_H - E_x + E_c$$

siendo

$$E_H = \frac{1}{2} \int dr V_H(r) n(r) \quad , \quad E_x = \frac{1}{4} \int dr V_x(r) n(r) \quad \text{y} \quad E_c = \frac{1}{2} \int dr V_c(r) n(r)$$

In [119]:

```
# Calculo la integral del potencial Vh*
suma = 0.0
for i in range(nsize):
    suma = suma + Vh[i]*U[i,0]**2
Eh = suma*h
print('Eh = ',Eh)
```

('Eh = ', 1.9968131611693958)

In [120]:

```
# Calculo la integral del potencial Vx
suma = 0.0
for i in range(nsize):
    suma = suma + Vx[i]*(U[i,0]**2)
Ex = 0.5*suma*h
print('Ex = ',Ex)
```

('Ex = ', -0.28735320557579874)

In [121]:

```
### Exchange LDA : -0.268 a.u.
### Exchange HF: -0.313
```

In [122]:

```
# Calculo la integral del potencial Vh*
suma = 0.0
for i in range(nsize):
    suma = suma + Vc[i]*U[i,0]**2
Ec = suma*h
print('Ec = ',Ec)
```

```
('Ec = ', -0.060863453009681934)
```

In [123]:

```
### Energia correlacion Expt : -0.042 a.u.
```

In [124]:

```
# Calculo la energia total del sistema

Etot = 2.0*(E[0]) - Eh - Ex + Ec
Etotal.append(Etot)

print('Energía Total: ')
Etotal
```

Energía Total:

Out[124]:

```
[-3.9996000799823599,
-2.6233502182290622,
-3.0257182698931011,
-2.8545905243657108,
-2.9277290205942199,
-2.8961497378034822,
-2.9097426879316632,
-2.9038838888694993,
-2.9064077827586474,
-2.9053202604356092]
```

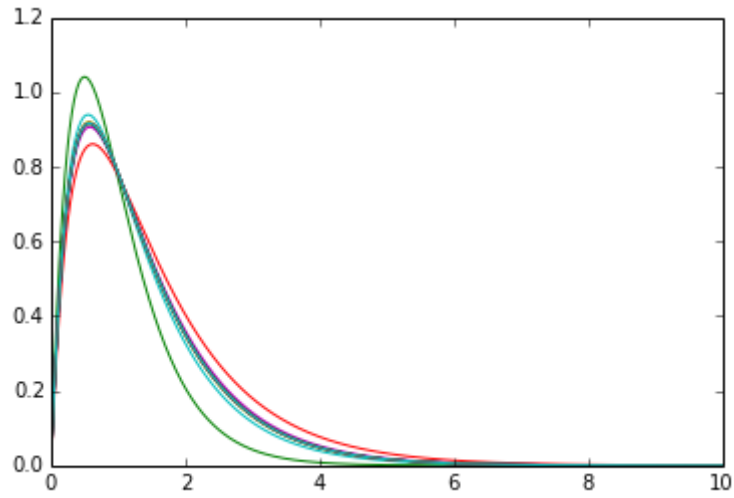
Energia total : -2.90368 a.u.

Fin loop ¶

In [125]:

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(r,u0);
```



In []: