ELECTRONIC STRUCTURE - PRACTICAL SESSION 1

Slater and Gaussian Type Orbitals
Calculation of integrals involving atomic orbitals

Gisela Martí Guerrero

January 2023

1 Problem 1

Consider the electronic Hamiltonian for a hydrogen atom (in a.u.)

$$\hat{H} = -\frac{1}{2}\cdot\nabla^2 - \frac{1}{r}$$

where

$$\nabla = \left[\frac{d^2}{dr^2} + \frac{2}{r}\frac{d}{dr} + \frac{1}{2r^2}\hat{L}^2\right]$$

and use a gaussian type 1s orbital to estimate the energy of its ground state by applying the variational method. What is the error in the ground state energy for this trial function?

```
import numpy as np
from scipy.optimize import minimize
from sympy import *

 # PROBLEM 1

def min(a):
    """
    Function to calculate the energy as a function of alpha

    """
    S = (np.pi/(2*a))**(3/2)
    T = 3*a**2*np.pi**(3/2) / (2*a)**(5/2)
    V = -2*np.pi / (2*a)
    E = (T+V)/S
    return E

# Provide an initial guess for the variational parameter a
a = 1
# Optimise the min function by varying a
ground_state = minimize(min, a)
print("Optimized alpha: ",ground_state.x,". Optimized ground state energy: ",ground_state.fun,"a.u.")

# Calculate the error
E_real = -0.5
error = abs(E_real - ground_state.fun)*100*2
print("The error is: ",error,"%")
```

The output of the code is:

```
Optimized alpha: [0.28294212] . Optimized ground state energy: -0.4244131815783876 a.u.
The error is: 15.117363684322482 %
```

2 Problem 2

Consider a 1s STO with $\zeta = 1$ and calculate the optimal exponent for a STO-1G function by maximizing their overlap integral:

$$S = \left(\frac{1}{\pi}\right)^{\frac{1}{2}} \cdot \left(\frac{2\alpha}{\pi}\right)^{\frac{3}{4}} \cdot \int e^{-r^2\alpha - r} d^3r$$

Use numerical integration to evaluate the overlap integral between the STO and the STO-1G functions for a fixed value of α and plot $S(\alpha)$ in the interval $0.1 < \alpha < 0.5$ to determine the value of α that leads to the best fit. What is the error in the energy for this function?

```
zeta, r, alpha, a1 = symbols("zeta r alpha a1", positive=True)

sto = (zeta **3 / pi) ** (1/2) * exp(-zeta * r)  # general expression for one STO
gto = (2 * alpha /pi) ** (3/4) * exp(-alpha * r**2)  # general expression for one GF

sto_1 = sto.subs(zeta, 1.0)  # zeta = 1.0
gto_1 = gto.subs(alpha, a1)  # alpha = a1

S = integrate(sto_1 * gto_1 * r**2, (r, 0, np.inf)) * 4 * pi  # the overlap between STO(1.0, r)
                                                                 and GF(alpha, r)

# We maximize this integral in terms of a1. We turn the maximization problem into minimization
  of the negative of the overlap S
def func(a):
    res = S.subs(a1, a[0]).evalf()
    return -res

res = minimize(func, x0=[0.2])
print("The optimal alpha exponent is: ", res.x[0])
print("The ground state energy for the 1s STO-1G function is: ", min(res.x[0]),"a.u.")
error = abs(E_real - min(res.x[0]))*100*2
print("The error is: ",error,"%")
```

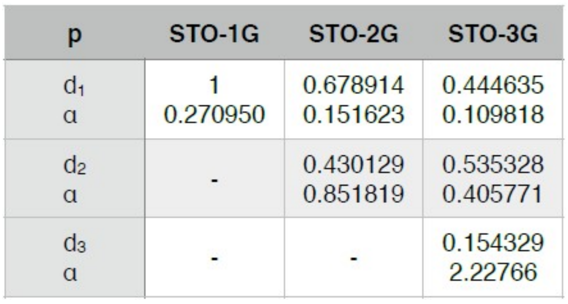
The output of the code is:

The optimal alpha exponent is: 0.27095000530375923 . The ground state energy for the 1s STO-1G function is: -0.4242184310709369 a.u. . The error is: 15.156313785812625 %

3 Problem 3

(a) Write a program to plot the wave function, its square and the radial distribution function for the 1s STO and for the 1s STO-1G, STO-2G, and STO-3G functions for which the optimal sets of coefficients and exponents are given in table 1.

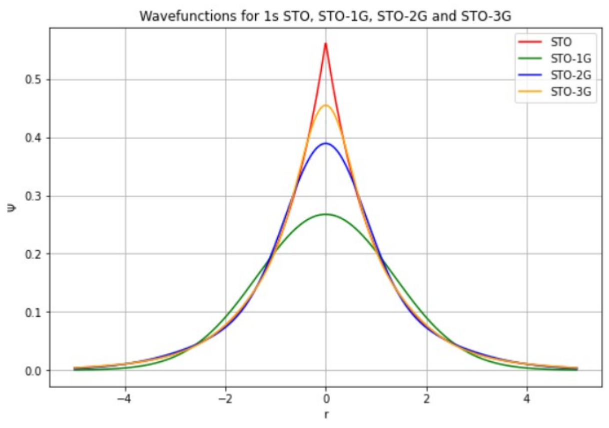
Table 1. Coefficients and exponents for the primitive GTOs in the 1s STO-3G orbital with $\zeta = 1$

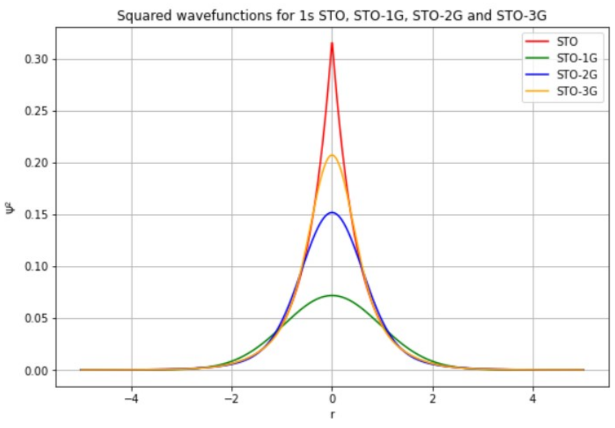


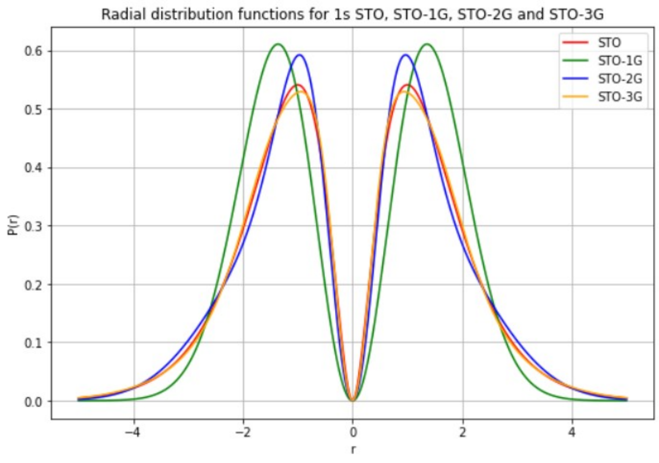
p	STO-1G	STO-2G	STO-3G
d_1 α	1 0.270950	0.678914 0.151623	0.444635 0.109818
d_2 α	-	0.430129 0.851819	0.535328 0.405771
d_3 α	-	-	0.154329 2.22766

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Coeff is the d parameter
coeff = np.array([[1.00000,0.0000000,0.000000],
                  [0.678914,0.430129,0.000000],
                  [0.444635,0.535328,0.154329]])

# Expon is the alpha parameter
expon = np.array([[0.270950,0.000000,0.000000],
                  [0.151623,0.851819,0.000000],
                  [0.109818,0.405771,2.227660]])

x = np.linspace(-5,5,1000)
r = abs(x)
zeta = 1.0

psi_STO = (zeta**3/np.pi)**(0.5)*np.exp(-zeta*r)
psi_STO_squared = psi_STO**2
radial_dist = 4*np.pi*r**2*psi_STO_squared

psi_CGF_STO1G = coeff[0,0]*(2*expon[0,0]/np.pi)**(0.75)*np.exp(-expon[0,0]*r**2)
psi_CGF_STO2G = coeff[1,0]*(2*expon[1,0]/np.pi)**(0.75)*np.exp(-expon[1,0]*r**2) +
                coeff[1,1]*(2*expon[1,1]/np.pi)**(0.75)*np.exp(-expon[1,1]*r**2) +
                coeff[1,2]*(2*expon[1,2]/np.pi)**(0.75)*np.exp(-expon[1,2]*r**2)
psi_CGF_STO3G = coeff[2,0]*(2*expon[2,0]/np.pi)**(0.75)*np.exp(-expon[2,0]*r**2) +
                coeff[2,1]*(2*expon[2,1]/np.pi)**(0.75)*np.exp(-expon[2,1]*r**2) +
                coeff[2,2]*(2*expon[2,2]/np.pi)**(0.75)*np.exp(-expon[2,2]*r**2)

psi_CGF_STO1G_squared = psi_CGF_STO1G**2
psi_CGF_STO2G_squared = psi_CGF_STO2G**2
psi_CGF_STO3G_squared = psi_CGF_STO3G**2

radial_dist_STO1G = 4*np.pi*r**2*psi_CGF_STO1G_squared
radial_dist_STO2G = 4*np.pi*r**2*psi_CGF_STO2G_squared
radial_dist_STO3G = 4*np.pi*r**2*psi_CGF_STO3G_squared

plt.figure(figsize=(9,6))
plt.title("Wavefunctions for 1s STO, STO-1G, STO-2G and STO-3G")
plt.xlabel("r (A)")
plt.ylabel("$\Psi$")
plt.plot(x, psi_STO, label="STO", color="red")
plt.plot(x, psi_CGF_STO1G, label="STO-1G", color="green")
plt.plot(x, psi_CGF_STO2G, label="STO-2G", color="blue")
plt.plot(x, psi_CGF_STO3G, label="STO-3G", color="orange")
plt.legend()
plt.grid()

plt.figure(figsize=(9,6))
plt.title("Squared wavefunctions for 1s STO, STO-1G, STO-2G and STO-3G")
plt.xlabel("r (A)")
plt.ylabel("$\Psi^2$")
plt.plot(x, psi_STO_squared, label="STO", color="red")
plt.plot(x, psi_CGF_STO1G_squared, label="STO-1G", color="green")
plt.plot(x, psi_CGF_STO2G_squared, label="STO-2G", color="blue")
plt.plot(x, psi_CGF_STO3G_squared, label="STO-3G", color="orange")
plt.legend()
plt.grid()

plt.figure(figsize=(9,6))
plt.title("Radial distribution functions for 1s STO, STO-1G, STO-2G and STO-3G")
plt.xlabel("r (A)")
plt.ylabel("P(r)")
plt.plot(x, radial_dist, label="STO", color="red")
plt.plot(x, radial_dist_STO1G, label="STO-1G", color="green")
plt.plot(x, radial_dist_STO2G, label="STO-2G", color="blue")
plt.plot(x, radial_dist_STO3G, label="STO-3G", color="orange")
plt.legend()
plt.grid()
```







(b) Write a program to find the most probable electron-nucleus distance, the average electron-nucleus distance, and the radius of a sphere around the nucleus containing the electron with a 99% of probability for the 1s STO and the three STO-NG functions.

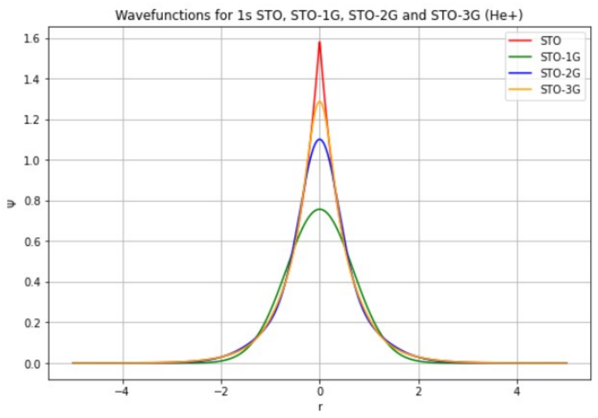
```
# The most probable electron-nucleus distance corresponds to the peak of each radial
  distribution function
# a_o (angstroms)
print("Most probable electron-nucleus distance for 1s STO: ", radial_dist.max())
print("Most probable electron-nucleus distance for 1s STO-1G: ", radial_dist_STO1G.max())
print("Most probable electron-nucleus distance for 1s STO-2G: ", radial_dist_STO2G.max())
print("Most probable electron-nucleus distance for 1s STO-3G: ", radial_dist_STO3G.max())

# The average electron-nucleus distance
# r = 3*a_o/2 (angstroms)
print("Average electron-nucleus distance for 1s STO: ", 3*radial_dist.max()/2)
print("Average electron-nucleus distance for 1s STO-1G: ", 3*radial_dist_STO1G.max()/2)
print("Average electron-nucleus distance for 1s STO-2G: ", 3*radial_dist_STO2G.max()/2)
print("Average electron-nucleus distance for 1s STO-3G: ", 3*radial_dist_STO3G.max()/2)

# radius of a sphere around the nucleus containing the electron with a 99% of probability
# r = 4.2*a_o (angstroms)
print("Radius of sphere with 99% probability for 1s STO: ", 4.2*radial_dist.max())
print("Radius of sphere with 99% probability for 1s STO-1G: ", 4.2*radial_dist_STO1G.max())
print("Radius of sphere with 99% probability for 1s STO-2G: ", 4.2*radial_dist_STO2G.max())
print("Radius of sphere with 99% probability for 1s STO-3G: ", 4.2*radial_dist_STO3G.max())
```

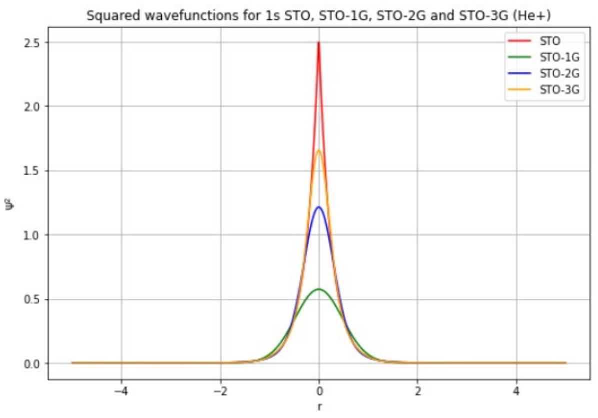
The output of the code is:

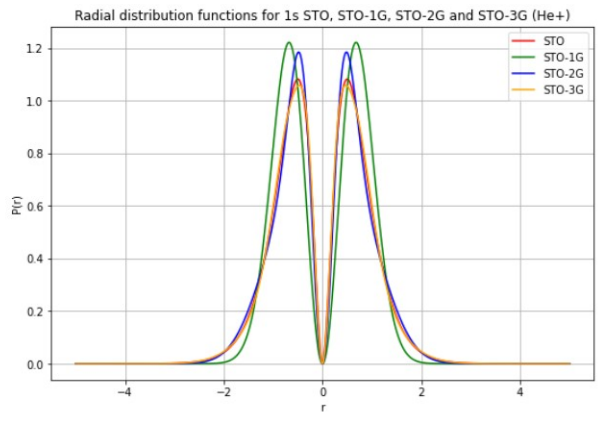
```
Most probable electron-nucleus distance for 1s STO: 0.5413324309728853
Most probable electron-nucleus distance for 1s STO-1G: 0.6111504054760712
Most probable electron-nucleus distance for 1s STO-2G: 0.5926938855047026
Most probable electron-nucleus distance for 1s STO-3G: 0.5296519395991914
Average electron-nucleus distance for 1s STO: 0.811998646459328
Average electron-nucleus distance for 1s STO-1G: 0.9167256082141069
Average electron-nucleus distance for 1s STO-2G: 0.8890408282570539
Average electron-nucleus distance for 1s STO-3G: 0.7944779093987872
Radius of sphere with 99% probability for 1s STO: 2.2735962100861182
Radius of sphere with 99% probability for 1s STO-1G: 2.566831702999499
Radius of sphere with 99% probability for 1s STO-2G: 2.489314319119751
Radius of sphere with 99% probability for 1s STO-3G: 2.224538146316604
```

(c) Use your programs to repeat a) and b) for a He+ ion for which $\zeta = 2$.

```python
zeta_He = 2.0
coeff_He = coeff
expon_He = expon*zeta_He**2

psi_STO_He = (zeta_He**3/np.pi)**(0.5)*np.exp(-zeta_He*r)
psi_STO_squared_He = psi_STO_He**2
radial_dist_He = 4*np.pi*r**2*psi_STO_squared_He

psi_CGF_STO1G_He = coeff_He[0,0]*(2*expon_He[0,0]/np.pi)**(0.75)*np.exp(-expon_He[0,0]*r**2)
psi_CGF_STO2G_He = coeff_He[1,0]*(2*expon_He[1,0]/np.pi)**(0.75)*np.exp(-expon_He[1,0]*r**2) +
                   coeff_He[1,1]*(2*expon_He[1,1]/np.pi)**(0.75)*np.exp(-expon_He[1,1]*r**2) +
                   coeff_He[1,2]*(2*expon_He[1,2]/np.pi)**(0.75)*np.exp(-expon_He[1,2]*r**2)
psi_CGF_STO3G_He = coeff_He[2,0]*(2*expon_He[2,0]/np.pi)**(0.75)*np.exp(-expon_He[2,0]*r**2) +
                   coeff_He[2,1]*(2*expon_He[2,1]/np.pi)**(0.75)*np.exp(-expon_He[2,1]*r**2) +
                   coeff_He[2,2]*(2*expon_He[2,2]/np.pi)**(0.75)*np.exp(-expon_He[2,2]*r**2)

psi_CGF_STO1G_squared_He = psi_CGF_STO1G_He**2
psi_CGF_STO2G_squared_He = psi_CGF_STO2G_He**2
psi_CGF_STO3G_squared_He = psi_CGF_STO3G_He**2

radial_dist_STO1G_He = 4*np.pi*r**2*psi_CGF_STO1G_squared_He
radial_dist_STO2G_He = 4*np.pi*r**2*psi_CGF_STO2G_squared_He
radial_dist_STO3G_He = 4*np.pi*r**2*psi_CGF_STO3G_squared_He

plt.figure(figsize=(9,6))
plt.title("Wavefunctions for 1s STO, STO-1G, STO-2G and STO-3G (He+)")
plt.xlabel("r (A)")
plt.ylabel("$\Psi$")
plt.plot(x, psi_STO_He, label="STO", color="red")
plt.plot(x, psi_CGF_STO1G_He, label="STO-1G", color="green")
plt.plot(x, psi_CGF_STO2G_He, label="STO-2G", color="blue")
plt.plot(x, psi_CGF_STO3G_He, label="STO-3G", color="orange")
plt.legend()
plt.grid()

plt.figure(figsize=(9,6))
plt.title("Squared wavefunctions for 1s STO, STO-1G, STO-2G and STO-3G (He+)")
plt.xlabel("r (A)")
plt.ylabel("$\Psi^2$")
plt.plot(x, psi_STO_squared_He, label="STO", color="red")
plt.plot(x, psi_CGF_STO1G_squared_He, label="STO-1G", color="green")
plt.plot(x, psi_CGF_STO2G_squared_He, label="STO-2G", color="blue")
plt.plot(x, psi_CGF_STO3G_squared_He, label="STO-3G", color="orange")
plt.legend()
plt.grid()

plt.figure(figsize=(9,6))
plt.title("Radial distribution functions for 1s STO, STO-1G, STO-2G and STO-3G (He+)")
plt.xlabel("r (A)")
plt.ylabel("P(r)")
plt.plot(x, radial_dist_He, label="STO", color="red")
plt.plot(x, radial_dist_STO1G_He, label="STO-1G", color="green")
plt.plot(x, radial_dist_STO2G_He, label="STO-2G", color="blue")
plt.plot(x, radial_dist_STO3G_He, label="STO-3G", color="orange")
plt.legend()
plt.grid()

# The most probable electron-nucleus distance for He+
print("Most probable electron-nucleus distance for 1s STO (He+): ", radial_dist_He.max())
print("Most probable electron-nucleus distance for 1s STO-1G (He+): ", radial_dist_STO1G_He.max())
print("Most probable electron-nucleus distance for 1s STO-2G (He+): ", radial_dist_STO2G_He.max())
print("Most probable electron-nucleus distance for 1s STO-3G (He+): ", radial_dist_STO3G_He.max())

# The average electron-nucleus distance is for He+
print("Average electron-nucleus distance for 1s STO (He+): ", 3*radial_dist_He.max()/2)
print("Average electron-nucleus distance for 1s STO-1G (He+): ", 3*radial_dist_STO1G_He.max()/2)
print("Average electron-nucleus distance for 1s STO-2G (He+): ", 3*radial_dist_STO2G_He.max()/2)
print("Average electron-nucleus distance for 1s STO-3G (He+): ", 3*radial_dist_STO3G_He.max()/2)

# radius of a sphere around the nucleus containing the electron with a 99% of probability for He+
print("Radius of sphere with 99% probability for 1s STO (He+): ", 4.2*radial_dist_He.max())
print("Radius of sphere with 99% probability for 1s STO-1G (He+): ", 4.2*radial_dist_STO1G_He.max(
print("Radius of sphere with 99% probability for 1s STO-2G (He+): ", 4.2*radial_dist_STO2G_He.max(
print("Radius of sphere with 99% probability for 1s STO-3G (He+): ", 4.2*radial_dist_STO3G_He.max(
```







The output of the code is:

```
Most probable electron-nucleus distance for 1s STO (He+): 1.082593865223363
Most probable electron-nucleus distance for 1s STO-1G (He+): 1.2222398858468455
Most probable electron-nucleus distance for 1s STO-2G (He+): 1.1853811713716156
Most probable electron-nucleus distance for 1s STO-3G (He+): 1.0592510597092695
Average electron-nucleus distance for 1s STO (He+): 1.6238907978350445
Average electron-nucleus distance for 1s STO-1G (He+): 1.8333598287702682
Average electron-nucleus distance for 1s STO-2G (He+): 1.7780717570574234
Average electron-nucleus distance for 1s STO-3G (He+): 1.5888765895639043
Radius of sphere with 99% probability for 1s STO (He+): 4.546894233938125
Radius of sphere with 99% probability for 1s STO-1G (He+): 5.1334075205567515
Radius of sphere with 99% probability for 1s STO-2G (He+): 4.978600919760786
Radius of sphere with 99% probability for 1s STO-3G (He+): 4.448854450778932
```