Exercise 08

Please submit your solution before **Thursday, 5 January, 8.00 a.m.** to
luca.donati@fu-berlin.de

1 Hydrogen wavefunctions

1.1 Introduction

The Hydrogen atom can be studied as a two-body system governed by the Hamiltonian:

$$\hat{H} = -\frac{\hbar}{2m_e}\nabla^2 - \frac{e^2}{4\pi\epsilon_0 r}$$

Exploiting the spherical geometry, we can use a spherical coordinate system with the proton at the origin and the electron moving in the three-dimensional space. Then, the time-independent Schrödinger equation becomes:

$$-\frac{\hbar}{2m_e}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right]\psi(r,\theta,\phi) - \frac{e^2}{4\pi\epsilon_0 r}\psi(r,\theta,\phi) = E\psi(r,\theta,\phi)$$

where $r \in [0, +\infty)$ is the radial distance of the electron from the origin (proton), $\theta \in [0, \pi]$ is the polar angle, and $\phi \in [0, 2\pi)$ is the azimuth angle. The wavefunction, solution of the equation, can be found separating the radial and the angular part (spherical harmonics):

$$\psi(r,\theta,\phi) = R_{nl}(r)Y_{lm}(\theta,\phi) = R_{nl}(r)f_l m(\theta)g_m(\phi)$$

where we have introduced respectively the principal, the azimuthal and the magnetic quantum numbers n, l and m that define an acceptable quantum state for the Schrödinger equation. The quantum numbers have respect the following conditions:

$$n = 1, 2, 3, ...$$

$$0 \leq l \leq n-1$$

$$m = 0, \pm 1, \pm 2, ..., \pm l$$

The normalized radial function is:

$$R_{nl}(r) = \sqrt{\left(\frac{2}{na_0}\right)^3 \frac{(n-l-1)!}{2n(n+l)!}} e^{-\frac{r}{na_0}} \left(\frac{2r}{na_0}\right)^l L_{n-l-1}^{2l+1}\left(\frac{2r}{na_0}\right) \tag{1}$$

where a_0 is the Bohr radius ($a_0 = 1$ for the exercise) and $L_{n-l-1}^{2l+1}\left(\frac{2r}{na_0}\right)$ are the associated Laguerre polynomials.
The normalized angular function is defined by the two functions:

$$f_{lm}(\theta) = (-1)^m \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}} P_{lm}(\cos\theta) \tag{2}$$

$$g_m(\phi) = e^{im\phi} \tag{3}$$

where $P_{lm}(\cos\theta)$ are the associated Legendre polynomials.

1.2 Exercise **(45 Points)**

As exercise, you have to write a Python script that, given a set of quantum numbers, implements the equations (1, 2, 3) and draws the radial function, the spherical harmonics and the orbitals.

1.2.1 Radial function (10 Points)

1. Define a vector for the coordinate r.
2. Implement the function (1) to compute $R_{nl}(r)$ and plot it.
3. Compute and plot the radial probability distribution.

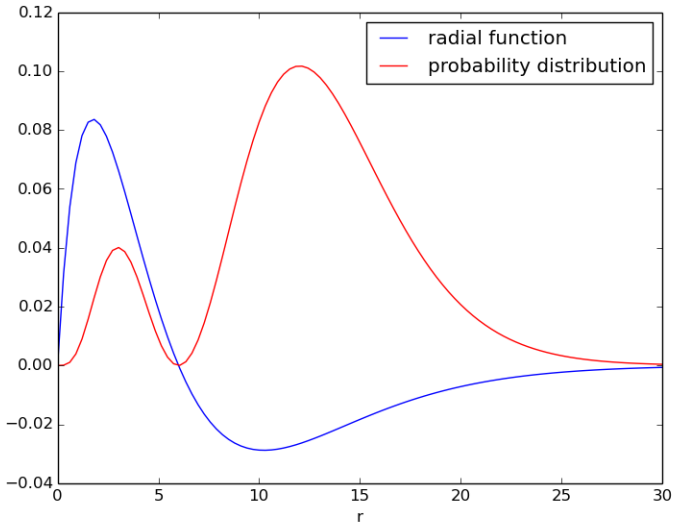

Figure 1: Radial function and radial prob. distribution for $n = 3$, $l = 1$ and $m = 0$.

Tips

1. To implement the associated Laguerre polynomials use the function:

```
from scipy.special import genlaguerre
...
genlaguerre(n-l-1,2*l+1)(2*r/(n*a0))
```

2. Be careful when you compute the radial probability distribution, your are working in spherical coordinates!

1.2.2 Angular function (20 Points)

1. Define two vectors ϕ and θ.
2. Implement the function (2) to compute $f_{lm}(\theta)$.
3. Implement the function (3) to compute $g_m(\phi)$.
4. Compute the spherical harmonics $Y_{lm}(\theta, \phi) = f_{lm}(\theta)g_m(\phi)$.
5. Compute and plot the angular probability distribution $|Y_{lm}|^2$. This quantity tells us the probability to find the electron in position (θ, ϕ). You can choose to plot the spherical harmonics in cartesian coordinates or projected on a unit sphere.

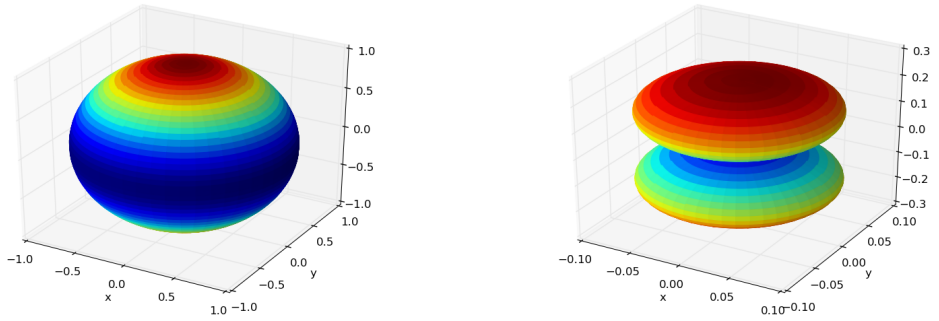

Figure 2: Angular probability distribution for $n = 3$, $l = 1$ and $m = 0$, projected on a unit sphere (left) and in cartesian coordinates (right)

Tips

1. To implement the associated Legendre polynomials use the function:

```
from scipy.special import lpmv
...
lpmv(m,l,np.cos(theta))
```

2. To plot $|Y_{lm}|^2$, it is necessary that Y_{lm} is stored in a matrix. To accomplish this, we compute the outer product between the vectors $f_{lm}(\theta)$ and $g_m(\phi)$ with the function outer(). If u and v are two vectors, then outer(u,v) returns a matrix $m_{ij} = u_i v_j$. To implement in python:

```
Y_lm = np.outer( g_m, f_lm)
rho = np.abs(Y_lm)**2
```

3. To plot a 3d sphere, you need its cartesian coordinates, so you need to implement its analytical representation:

$$x = r\sin(\theta)\cos(\phi)$$
$$y = r\sin(\theta)\sin(\phi)$$
$$z = r\cos(\theta)$$

Also in this case, you need x, y and z as matrices, so you need to use outer():

```
xs = 1 * np.outer(np.cos(phi), np.sin(theta)) #radius=1
ys = ...
zs = 1 * np.outer(np.ones(np.size(theta)), np.cos(theta))
```

4. Finally to plot a 3d surface, follow the script:

```
from mpl_toolkits.mplot3d import Axes3D
...
color_map = cm.jet
scalarMap = cm.ScalarMappable(norm=plt.Normalize(vmin=np.min(rho),...
                                ... vmax=np.max(rho)), cmap=color_map)

C = scalarMap.to_rgba(rho) #scalarmap according to the
                           #probability distribution stored in rho

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.plot_surface(xs, ys, zs, rstride=2, cstride=2, color='b', facecolors=C)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

5. In cartesian coordinates $\rho = |Y_{lm}|^2$ is the distance of the probability distribution from the origin. In other words, the points of the surface (fig.2 right) indicate the probability to find the electron in position (θ, ϕ). Greater is the distance between the surface and the origin (red areas), greater is the probability. Finally, to represent the spherical harmonics in cartesian coordinates, you just need to replace the unit radius with ρ.

1.2.3 Orbitals (15 Points)

The orbitals combine the radial function with the spherical harmonics and tell us in which area of the 3d space we have more chance to find the electron. In other words, the orbital is a function that maps the position of the space to a certain probability. You should write a script that represents the section of the orbitals when $z = 0$.

1. Define two vectors $x \in [-R, R]$ and $y \in [-R, R]$ where R is an enough big number.
2. Use x and y and the function meshgrid() to define two full grids X and Y of coordinates (look tips for better explanation).
3. Convert your cartesian space (defined by the matrices X and Y) in spherical coordinates.
4. Recompute $R_{nl}(r)$, $f_{lm}(\theta)$ and $g_m(\phi)$. Notes that now, r, θ and ϕ are matrices, because we want to assign a s certain probability to each point of the plane $z = 0$.
5. Compute and plot the probability distribution.

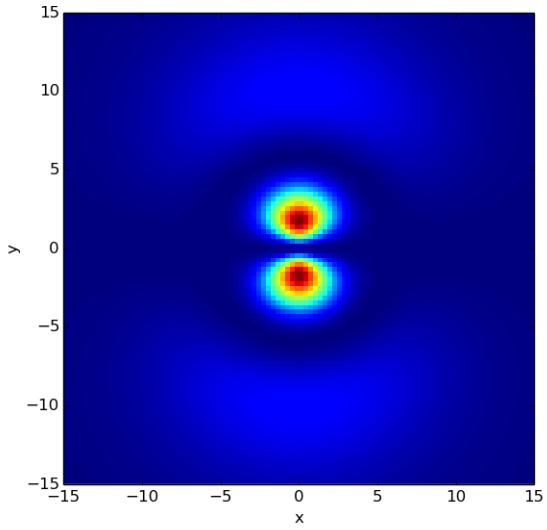

Figure 3: Section $z = 0$ of the orbital for $n = 3$, $l = 1$ and $m = 0$.

Tips

1. The function meshgrid() replicates the grid vectors x and y to produce the coordinates of a rectangular grid (X, Y). We need this because, we want assign a certain probability to each point (x, y) of the plane. If we define our space on matrices, we can compute all the possible combinations in one step. The following example, should clarify how meshgrid() works.

```
u=np.array([0,1,2,3])
v=u
U, V = np.meshgrid(u,v)

print U
array([[0, 1, 2, 3],
```

```
        [0, 1, 2, 3],
        [0, 1, 2, 3],
        [0, 1, 2, 3]])

print V
array([[0, 0, 0, 0],
        [1, 1, 1, 1],
        [2, 2, 2, 2],
        [3, 3, 3, 3]])
```

2. You need to define a function to convert cartesian to spherical coordinates:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arctan \frac{\sqrt{x^2 + y^2}}{z}$$

$$\phi = \arctan \frac{y}{x}$$

   In python the function arctan() is defined by atan2():

```
np.atan2(y,x) # = arctan(y/x)
```

3. Once you have a matrix p containing the probability distribution of the plane, you can plot it with the function imshow():

```
fig = plt.figure()
plt.imshow(p.T,extent=[-rmax, rmax, -rmax, rmax], interpolation='none',origin='lower')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```