Extra Credit Question 2

Accompanying document for the code written as a solution to Question 2

Graham Byrd

May 9, 2022

(This page intentionally left blank)

Contents

1 Structure **1**

2 Code Breakdown **2**

2.1 Getting the Legendre and Laguerre Polynomials 3

2.2 Getting the base case of our Spherical Harmonics 5

2.3 Generating our final Radial and Spherical Harmonics 6

2.4 Creating our Wavefunction and making our figure 8

3 Output **10**

1 Structure

The code for this question is split up into a main file, two main functions, and a few internal sub-functions.

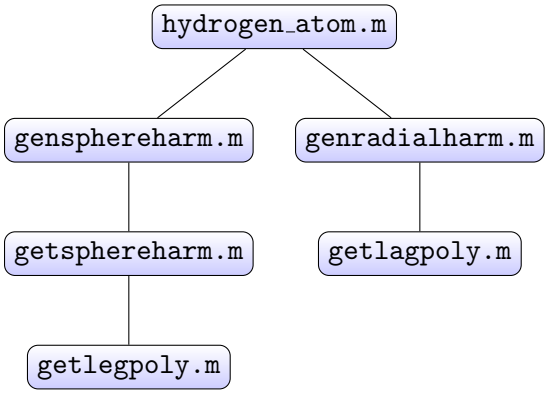


Figure 1.1: File dependence structure (all files below `hydrogen_atom.m` are functions).

This makes the main file more compact, and the whole system easier to read and understand. It also helped to reduce confusion during coding, and isolate bugs and errors. Essentially this figure is just a call chain, as we will soon see, we call `gensphereharm.m` first, which in turn calls `getsphereharm.m`, which then calls `getlegpoly.m`. These functions all do necessary calculations regarding parts of the formula for the Hydrogen orbitals.

2 Code Breakdown

In this chapter we will cover the various aspects, purposes, and structures behind the code that has been introduced. Our goal is to eventually generate the following formulae:

$$\psi_{n,\ell,m} = R_{n,l}(r)Y_{\ell,m}(\theta,\phi), \tag{2.1}$$

$$R_{n,\ell}(r) = -\sqrt{\left(\frac{2Z}{na_\mu}\right)^3 \frac{(n-\ell-1)!}{2n(n+\ell)!}} e^{-Zr/na_\mu} \left(\frac{2Zr}{na_\mu}\right)^\ell L_{n-\ell-1}^{(2\ell+1)}\left(\frac{2Zr}{na_\mu}\right), \tag{2.2}$$

$$Y_\ell^m(\theta,\varphi) = (-1)^m \sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}\, P_\ell^m(\cos\theta)\, e^{im\varphi}. \tag{2.3}$$

2.1 Getting the Legendre and Laguerre Polynomials

To start off with, we will take at the most basic functions we define, `getlegpoly.m` and `getlagpoly.m`. Looking at `getlegpoly.m`:

```
function legpoly = getlegpoly(m, l)
%{
        Gets the legendre polynomial for our given quantum number
        m and l (magnetic and angular quantum)

        https://en.wikipedia.org/wiki/Associated_Legendre_polynomials
%}

syms x;                 % set up symbolic variable x
rodrigues = (x^2 - 1)^l; % set up differentiated part of rodrigues formula

if (l+m == 0)
        differ = rodrigues;
else
        differ = diff(rodrigues,l+m); % see rodrigues formula on wikipedia
end

a_legpoly = (((-1)^m)/(2^l*factorial(l)))*((1-x^2)^(m/2))*differ;

legpoly = matlabFunction(a_legpoly);
end
```

This function generates the Associated Legendre polynomials[†] (P_ℓ^m in eq. (2.3)) in using the given m and l quantum number in the formula.

[†] Wikipedia

Now, looking at `getlagpoly.m`:

```
function lagpoly = getlagpoly(n,l)
% This function will get the laguerre polynomial for a given n and l

% setting up the alpha and n that the wikipedia page uses to calculate
alpha = 2*l + 1;
nn = n - l - 1;       % the legendre polynomial at the right degree

% once again, we have a derivative, so we need a symbolic x
syms x;
rodrigues = exp(-x) * x^(nn+alpha);   % differential term of rodrigues formula

if (nn == 0)
        dif = rodrigues;
else
        dif = diff(rodrigues,nn);
end

a_lagpoly = (x^(-alpha) * exp(x)) / (factorial(nn)) * dif;

lagpoly = matlabFunction(a_lagpoly);
end
```

As you have probably already guessed, this generates the Laguerre polynomial we see in the radial function ($L_{n-\ell-1}^{(2\ell+1)}$ in eq. (2.2)).

2.2 Getting the base case of our Spherical Harmonics

For this function our goal is simple, construct the actual Y function. We can do this via multiplication and a little bit of thinking. If our $l = 0$, then our Associated Legendre polynomial is 1. If it is anything else, i.e. $l > 0$, we have the expected Legendre polynomial that we calculate in `getlegpoly.m`. This is clear looking at the code in `getsphereharm.m`:

```
function sphericalharm = getsphereharm(theta,phi,m,l)
%{
        Constructs the spherical harmonics given the
        magnetic quantum number and the angular momentum
        quantum number.
%}

P = getlegpoly(m,l);
if (l == 0)
        sphericalharm = sqrt(((2*l+1)/(4*pi)).*(factorial(l-abs(m))...
                /factorial(l+abs(m)))).*exp(1i*m*phi);
else
        sphericalharm = sqrt(((2*l+1)/(4*pi)).*(factorial(l-abs(m))...
                /factorial(l+abs(m)))).*exp(1i*m*phi).*P(cos(theta));
end
end
```

Once we have the Spherical Harmonics we can now look at its various cases, covered in the next section.

2.3 Generating our final Radial and Spherical Harmonics

Looking now towards our two functions that we call directly in our main code, they are very similar, the main idea is dealing with specific cases of quantum numbers. First looking at the `gensphericalharm.m` function,

```
function Y = gensphereharm(theta, phi, m, l)
%{
        This does our final calculation, and generates the spherical
        harmonic values for our input variables.

        m can be = -l to l, so we have a few cases which are
        covered in the below linked part of the wikipedia article:
        https://en.wikipedia.org/wiki/Spherical_harmonics#Real_form

        The third listed case seems much simpler to code, so we are using that.
%}

if (m < 0)
        Y = sqrt(2) * (-1)^m * imag(getsphereharm(theta,phi,abs(m),l));
elseif (m == 0)
        Y = getsphereharm(theta,phi,m,l);
else
        Y = sqrt(2) * (-1)^m * real(getsphereharm(theta,phi,m,l));
end
end
```

For ease of reference, the calculations done in that if-loop are characterized by the Wikipedia link here and included in the code, and the formula is shown below:

$$Y_{\ell,m} = \begin{cases} \sqrt{2}(-1)^m \mathfrak{Im}[Y_\ell^{|m|}] & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \sqrt{2}(-1)^m \mathfrak{Re}[Y_\ell^m] & \text{if } m > 0. \end{cases} \tag{2.4}$$

A similar concept manifests but in a different way in the case of `genradialharm.m`. We will see that the if-loop is again accounting for specific cases of our quantum numbers, but it happens a step earlier, when we are calculating our Laguerre polynomial. There is a case such that when $n-l-1=0$ our Laguerre polynomial goes to 1, similarly to how our Legendre polynomial goes to 1 if $l=0$. This is important to include in our code, as it gives us a way to minimize errors *and* speed up computation time.

Below is the function `genradialharm.m`;

```
function R = genradialharm(r,n,l)
% We now generate the Radial Harmonic

% We have a universal constant for an atom of constant mass,
% so we set it equal to 1
a = 1;
% for a hydrogen atom, we have Z=1 so we can just omit it from our calculation,
lagpoly = getlagpoly(n,l);
if (n-l-1 == 0)
        lagpoly = 1;
else
        lagpoly = lagpoly(2*r/(n*a));
end
R = sqrt((2/n*a).^3*factorial(n-l-1)/(2*n*factorial(n+l)))...
        .*exp(-r/(n*a)).*(2*r/(n*a)).^l.*lagpoly;
end
```

This is the last of our defined functions, and all that is left is our script that generates our wavefunction and figure.

2.4 Creating our Wavefunction and making our figure

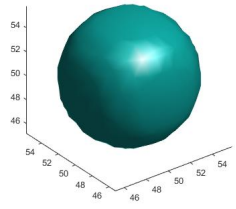
Recalling eq. (2.1), we know all we have to do to create our wave function is multiply the Spherical and Radial harmonics. We also know that ψ is a probability amplitude, so to get our actual orbitals, we need to plot $|\psi|^2$. Looking at our script, `hydrogen_atom.m`:

```
%% Variable initialization

% The 50's can be changed to fit the plot better,
% but anything beyond 100 steps between them uses
% too much processing power in calculation
grid = linspace(-50,50,1e2);

% Mapping vars. to x,y,z
[x, y, z] = meshgrid(grid,grid,grid);

% Spherical Transform
r = sqrt(x.^2 + y.^2 + z.^2);
theta = acos(z./r);
phi = atan2(y, x);

% Init. vars.
n = 4;
l = 3;
m = 0;

% Make the harmonic and radial part
Y = gensphereharm(theta,phi,m,l);
R = genradialharm(r,n,l);

% Combine the parts
psi = R .* Y;

%% Plotting

figure()
isosurface(psi.^2,1e-5,sign(psi))
% Isovalue set to 1e-5, just works the best, and I dont get any
% improvement by lowering it.
axis equal
% Ensures axis have same unit length
axis vis3d
% Freezes aspect ratio, allows the plot to look like the actual orbitals
```

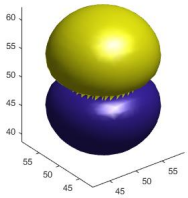
This script is fairly straightforward, we assign our x, y, z values, translate them to spherical, assign some quantum numbers, call our functions, get our ψ, then plot it. The majority of testing and thought went into plotting it the best. `isosurface()` gives a beautiful surface rendering at the right `isovalue`, and `vis3d` keeps the native aspect ratio, generating plots that are very accurate to real orbitals.

3 Output

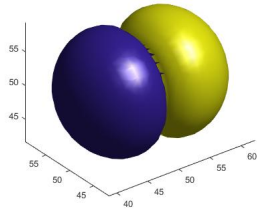
This chapter will just be a showcase of the figures this script generates, from the $n = 1$ orbitals to the $n = 4$ orbitals. For the sake of space and time, I wont show any duplicate or inverted ($m = \pm 1$) figures, these can be obtained via the code if so desired.



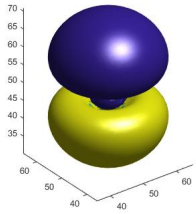
(a) $n = 1, \ell = 0, m = 0$



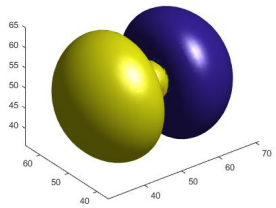
(b) $n = 2, \ell = 1, m = 0$



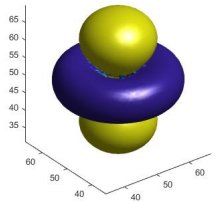
(c) $n = 2, \ell = 1, m = 1$



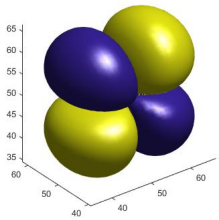
(d) $n = 3, \ell = 1, m = 0$



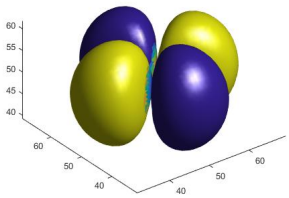
(e) $n = 3, \ell = 1, m = 1$



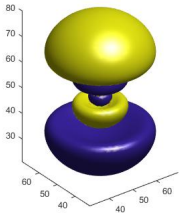
(f) $n = 3, \ell = 2, m = 0$



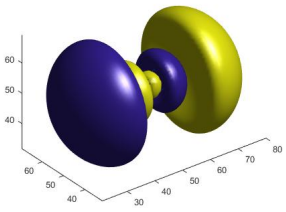
(g) $n = 3, \ell = 2, m = 1$



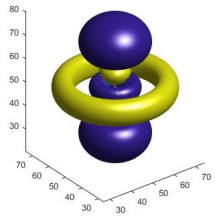
(h) $n = 3, \ell = 2, m = 2$



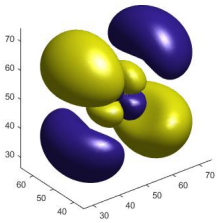
(i) $n = 4, \ell = 1, m = 0$



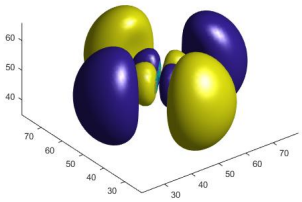
(j) $n = 4, \ell = 1, m = 1$



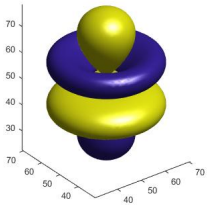
(a) $n = 4, \ell = 2, m = 0$



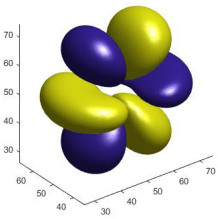
(b) $n = 4, \ell = 2, m = 1$



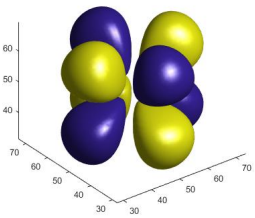
(c) $n = 4, \ell = 2, m = 2$



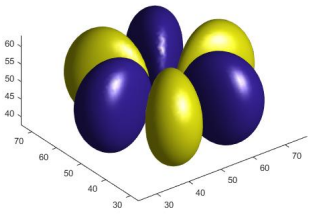
(d) $n = 4, \ell = 3, m = 0$



(e) $n = 4, \ell = 3, m = 1$



(f) $n = 4, \ell = 3, m = 2$



(g) $n = 4, \ell = 3, m = 3$