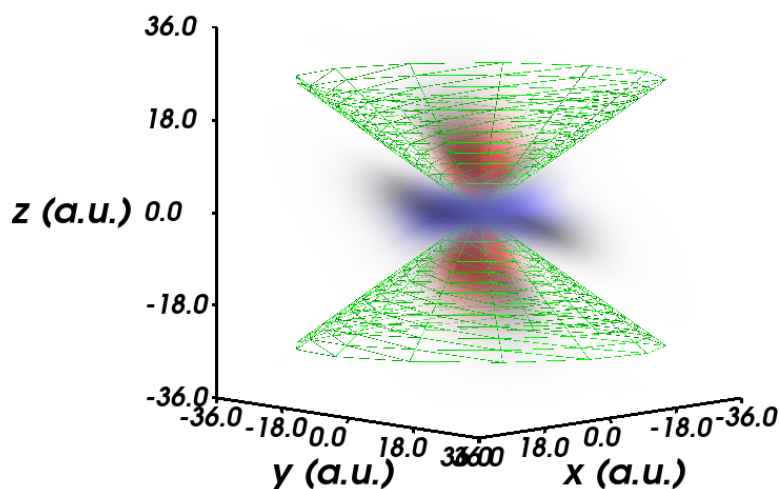
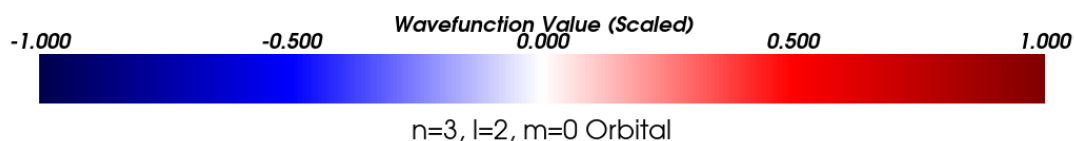


1 About this Software

The following are instructions for installing and using the GUI-based hydrogen atomic orbital visualizer contained in the Atomic.Orbitals.GUI.ipynb and Atomic.Orbitals.GUI.py script files. The atomic orbital visualizer and GUI are themselves simply an iPython Jupyter notebook (.ipynb) file or Python (.py) file that needs no separate installation. This program can be run in the Jupyter notebook application that is part of the installation instructions and can be used as described in a later section. The code is written exclusively in Python, and all installations are related to packages of Python or the Jupyter notebook software. An example of the GUI used to run the program and an (unrelated) output from the program are shown below.

Principal Quantum Number n:	2	Allowed Values: 1,2,3,...
Angular Momentum Quantum Number l:	1	Allowed Values: 0,1,...,n-1
Magnetic Quantum Number m:	0	Allowed Values: -l,-l+1,...,-1,l
Colormap:	seismic	
Colormap points:	21	Allowed Values: 1,3,5,...,255
Plot Mode:	orbital	
Fractional Cutoff Value:	0.0	Allowed Values in [0,1]
Plot All Nodes:	<input type="checkbox"/>	
Plot Radial Nodes:	<input type="checkbox"/>	
Plot θ Nodes:	<input type="checkbox"/>	
Plot ϕ Nodes:	<input type="checkbox"/>	
Opacity Scaling Exponent:	1.0	Must Be 0 or Above
Opacity Shift:	0.0	Allowed Values in [0,1]
Use White Background:	<input checked="" type="checkbox"/>	
Round Axes to Multiples of 10 a.u.:	<input type="checkbox"/>	
Maximum Radial Distance (a.u.) Per n:	12	
Reverse Colormap:	<input type="checkbox"/>	
Cut Quadrant for m = 0 Orbitals:	<input type="checkbox"/>	
Scale Wavefunction By Maximum Value:	<input checked="" type="checkbox"/>	
Nodal Surface Opacity:	0.8	Allowed Values in [0,1]
Radial Node RGB Values:	(0,1,0)	Allowed Values in [0,1]
θ Node RGB Values:	(0,1,0)	Allowed Values in [0,1]
ϕ Node RGB Values:	(0,1,0)	Allowed Values in [0,1]
RGBs Are Tuples		
Maximum Opacity:	0.6	Allowed Values in [0,1]
Outline Box:	<input type="checkbox"/>	
Number of Grid Points Per n:	20	
<input type="button" value="Defaults"/> <input type="button" value="Submit"/>		



2 Package Dependencies

2.1 “Quick” Installation Guide

The following steps should be followed to install the necessary packages:

1. Download and install Python 3. **Windows users should check the box that installs Python3 on the user’s PATH during installation!**
2. Install and/or open a terminal application and run the following command (all in one line) to install various necessary Python packages and mayavi dependencies,

```
python3 -m pip install numpy scipy sympy setuptools tk matplotlib  
vtk pyqt5 envisage traits traitsui apptools appinst pyface configobj
```

You may be prompted to confirm the downloads of various packages by typing “Y” and Enter/return into the terminal. If this does not run properly, try removing the “3” from the end of the python3 command. If one or more packages fails to install properly it is worth trying to install the package individually using the same installation command with only the offending package(s).

3. (Optional) If you wish to run the code through Jupyter, you will also need to run the following command to install Jupyter and additional packages for mayavi compatibility,

```
python3 -m pip install notebook ipywidgets ipyevents
```

4. Install git by downloading the installer from Sourceforge. You can confirm that it is properly installed by using the terminal command,

```
git --version
```

5. To download mayavi, use the command,

```
git clone https://github.com/enthought/mayavi.git
```

Then use the following commands in order,

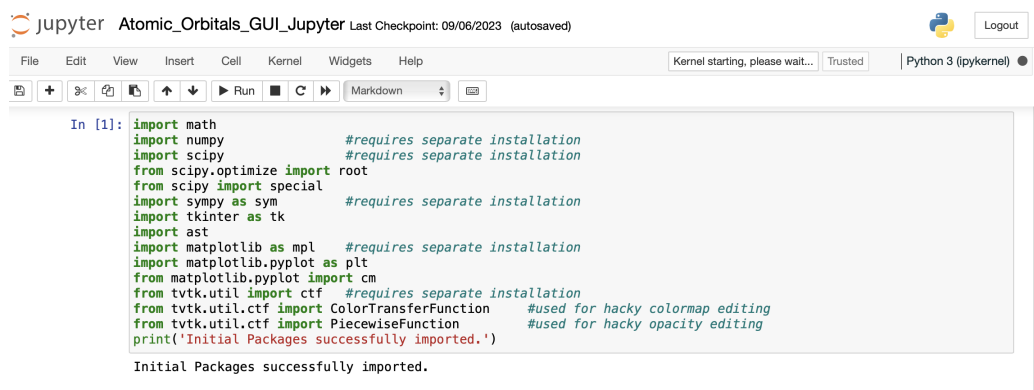
```
cd mayavi  
python3 setup.py install
```

The orbital visualizer code will now be fully usable since all requisite packages will be installed on your computer.

2.2 Additional Installation Explanations

The need for installation instructions comes from the various Python packages used in the Atomic_Orbitals_GUI.ipynb script and the large number of dependencies of the mayavi visualization suite in Python. Python is an extremely popular and relatively user-friendly coding language. For Windows users, be sure to choose the option to include Python on the user's PATH when first installing it, or else the executable command for Python in the terminal will not actually work! The PATH environment variable is where the computer looks to find executable files for various programs. You can also manually modify the PATH on Windows by searching for the environment variables and adding a new PATH variable corresponding to the locations of executable files on your computer. This can be used if Python is accidentally installed without adding the executable to the PATH, but will not be covered here.

The reason we need to install additional modules is that Python does not have all available features activated unless they are requested via the import command. Some packages with niche technical uses are not installed by default, which necessitates obtaining them from an external source. This allows programs to use fewer computer resources processing unnecessary computer code and saves hard drive space for those that do not need certain packages. In this program, there is a large block of code that imports the necessary Python packages, as shown in the code sample below.



The screenshot shows a Jupyter Notebook window titled "Atomic_Orbitals_GUI_Jupyter". The code cell contains the following imports:

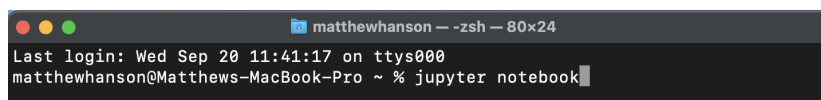
```
In [1]: import math
import numpy           #requires separate installation
import scipy           #requires separate installation
from scipy.optimize import root
from scipy import special
import sympy as sym     #requires separate installation
import tkinter as tk
import ast
import matplotlib as mpl #requires separate installation
import matplotlib.pyplot as plt
from matplotlib.pyplot import cm
from tvtk.util import ctf #requires separate installation
from tvtk.util.ctf import ColorTransferFunction #used for hacky colormap editing
from tvtk.util.ctf import PiecewiseFunction    #used for hacky opacity editing
print('Initial Packages successfully imported.')
```

Below the code cell, the output reads: "Initial Packages successfully imported."

To install most of the necessary packages on either Windows or MacOS computers, just run the following command in a terminal application once Python has been installed,

```
python3 -m pip install numpy scipy sympy setuptools tk matplotlib
vtk pyqt5 envisage traits traitsui apptools appinst pyface configobj
```

On a Mac terminal application, entering code in the terminal may look like the image shown below. In this example, the command opens the Jupyter application.



The screenshot shows a terminal window with the title "matthewhanson -- zsh -- 80x24". The output shows the last login time and the command executed:

```
Last login: Wed Sep 20 11:41:17 on ttys000
matthewhanson@Matthews-MacBook-Pro ~ % jupyter notebook
```

To find the terminal application, on MacOS simply open a Finder window and navigate to Applications/Utilities/Terminal. It is recommended to make the Terminal application a shortcut on the user's toolbar for ease of use. On Windows, simply type "PowerShell"

into the search bar near the Start menu and select the Windows PowerShell application. If PowerShell is not already installed, it can be downloaded for free from the Microsoft store. Once again, it is recommended that the user pin a terminal application to their toolbar for ease of access later. No knowledge of operating the terminal is required to utilize the orbital visualizer other than following the commands explicitly laid out in these instructions. While it is possible to execute the Jupyter notebook by searching for the application on Windows, it is much easier to open using command line.

The “pip install” command is a built-in command included with Python that is used to install additional packages to the Python distribution. The pip command for Python3 is also frequently used in the form,

```
pip3 install numpy
```

but the version of the command,

```
python3 -m pip install
```

ensures that executable files are installed on the user’s PATH for Windows machines. Note that some installations of Python will only recognize the execution command “python” in a terminal, so the above installation command can be modified by removing the “3” from the end of the python3 command. If the lengthier version of the pip install command is not used to install Jupyter, Windows users will typically receive an error that the execution command is not found when they try to run jupyter notebooks. Note that the above commands can be run for individual packages (e.g. “pip3 install numpy”) or for some small subset of the packages (e.g. “python3 -m pip install numpy scipy”), if desired, though it is easiest to install many at once. Note that when running pip on its own, the command pip3 must be run for installation of packages into Python3. Use of “pip install” rather than “pip3 install” will attempt to install packages for Python2 for MacOS users, which has now been deprecated despite being installed for MacOS by default.

Some of the necessary packages may already be installed depending on the user’s Python distribution. For instance, the numpy and scipy modules are typically installed in Anaconda distributions of Python, though we will not discuss using Anaconda or installing separate packages in Anaconda. Attempting to install packages that are already installed will not cause problems, but the user will see a large number of messages in the terminal indicating that the package dependencies are already satisfied.

3 Running the Program

3.1 Quick Running Instructions

The following steps can be used to run the Atomic_Orbitals_GUI.ipynb script through Jupyter notebooks:

1. Using a terminal application, enter the command,

```
jupyter notebook
```

If this fails, you may also try,

```
python -m notebook
```

2. Using Jupyter's browser window, navigate to the Atomic_Orbitals_GUI.ipynb file's location and click on the file to open it.
3. Run the script either cell by cell or by resetting the kernel and running all cells simultaneously (preferred).
4. Input the desired orbital quantum numbers and other desired settings and enjoy the interactive output.
5. When finished, simply close the Jupyter window and then hit "Quit" in the original Jupyter window before closing it. The program is now entirely terminated.

To run the pure Python script file Atomic_Orbitals_GUI.py when it is placed on the user's desktop, the following steps can be used:

1. Open the terminal application and type the following commands sequentially,

```
cd Desktop  
python3 Atomic_Orbitals_GUI.py
```

If the latter fails, try removing the "3" from the python3 command.

2. Input the desired orbital quantum numbers and other desired settings and enjoy the interactive output.
3. Run the command again to visualize another orbital, or simply close the terminal window once you are finished.

3.2 Additional Running Explanations

Once the requisite packages are installed, the Jupyter notebooks application can be run at any time by simply typing the following command into a command line terminal,

```
jupyter notebook
```

For reasons unknown, on certain Windows machines one of the following commands is needed instead,

```
python -m notebook
```

```
python3 -m notebook
```

Upon doing so, a new window will open up in your default web browser (though it is not actually a website) and you can interact with the jupyter notebook much like any file manager. Files can be opened by simply clicking on them, and directories/folders can be navigated likewise. An example of a directory in the Jupyter window is shown below.



Note that the jupyter notebook will default to using whatever directory/folder the “jupyter notebook” command was called in as its starting location. Typically, this will be one level above the “Desktop” directory/folder. Navigate to whatever directory/folder holds the Atomic_Orbitals_GUI.ipynb file and click on it to open the notebook. Note that for Windows users with OneDrive enabled, you may have to navigate to the OneDrive directory before your Desktop and other files are readily visible.

At this point, detailed descriptions of all of the code blocks and the utility of the required packages is explained within the Jupyter notebook itself. While the user can freely peruse the code and text blocks, running the code simply requires the user to execute the code either cell by cell or all at once while resetting the kernel. These commands are found in the upper toolbar on the Jupyter notebook, which is shown below. All the buttons are explained in the Jupyter users guide.



The “Run” button will execute whatever individual cell the user is utilizing in Jupyter and will proceed to the next cell. Using this button multiple times will allow a user to fully execute a program. The circular arrow resets the kernel for the program, resetting everything to a

state where the program has not run. Note that this means that later code cells may fail since the data they may require is no longer initialized by previous cells! To reset the kernel and run all cells, use the double right-facing arrow button next to the drop down box reading “Markdown” shown above. After several seconds of parsing the necessary package imports, the GUI will open so the user can choose settings. Upon closing the GUI, the program will perform the requested calculations and the final visualizations will be contained in separate Mayavi Scene windows. If invalid settings are requested, the program will re-open the GUI and request the user to try again. Note that these Mayavi Scene windows containing the orbital visualizations may be hidden behind the user’s browser window, especially on MacOS machines. Check for this by minimizing the main window if you do not see the output.

Once the Mayavi windows are closed, the program will also produce some simpler plots of the distinct portions of the orbitals and a plot of the user-defined values of the colormap and opacity scaling. Note that if the user does not have a distribution of \LaTeX installed on their computer, the titles and labels in these graphs will, by necessity, utilize an ugly typesetting format. Those with a \LaTeX distribution will have nicer typeset labels.

When the user is finished with the program, they should just close the Jupyter window that was running the script. In the original Jupyter window with the directory browser shown above, the user should then hit “Quit” and then close the window. The terminal will stop running Jupyter at this point and the user is finished with the program. The terminal window can be closed or utilized for other purposes at this point.

Alternatively, we have also included a standalone pure Python version of the code that does not run through Jupyter. The code is exactly identical to that within the Jupyter script. This code can be run directly from the command line using the `python3` command, which saves some steps, but lacks the additional pedagogical and explanatory blocks within the Jupyter environment. For everyday use, however, this is slightly simpler to use and requires fewer packages to be installed.