First Assignment: Preparatory Tasks
Computational Atomic Physics

Julia Lindohf

Summer 2022

The task is to create a plot, see figure 1.

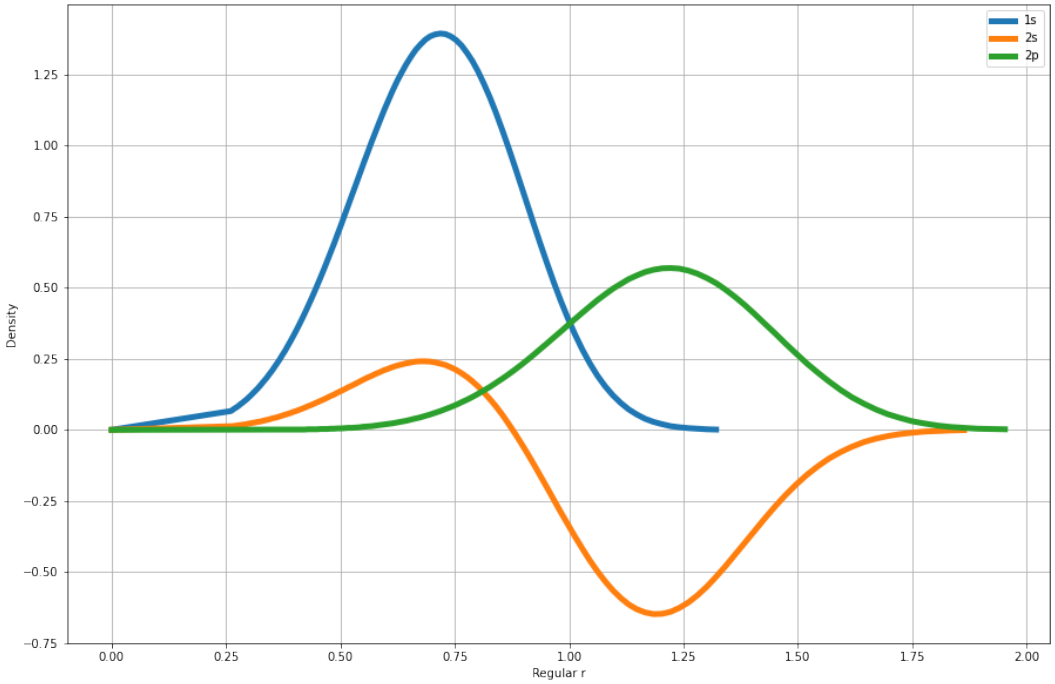

Figure 1: The density plot, based on the given *.dot* file is displayed in this figure.

The python codes are displayed here. To begin with, an object class has been created to process the *.dot* file. The final product is a data frame, a hash table. It is a fast data type. If the number of spectra is plenty, there is an advantage of using a hash table to store the results.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

class wavefunctionplot:
  def __init__(self, sqrtlist, wavefunction, str_elem):
    # to store the instance variable
    self.sqrtlist = sqrtlist
    self.wavefkt = wavefunction
    self.listnames = str_elem

  def breakframe(self):
    # the number of elements
    L = len(self.sqrtlist)
    # to store all usable indice
    indexlist = []
    newlist = [ ]
    for i in range(L):
      try:
        elem = self.sqrtlist[i]
        newelem = float(elem)
        newlist.append(newelem)
      except:
        newelem = 0
        indexlist.append(i)
        newlist.append(newelem)
    indexlist.append(L)
    # the products are two new instance lists
    self.indexlist = indexlist

  def newdataframe(self):
    # to store the data in a new dictionary
    self.newdict = {}

    def newfram(list1, list2):
      # the first list contains the sqrt r
      # to transform the sqrt r to regular r
      regular_r = [np.sqrt(float(r)) for r in list1]
      wave = [float(elem) for elem in list2]
      data = { 'Regular_r': regular_r, 'wavedensity': wave}
      # the output is a new dataframe
      #print( data )
      return pd.DataFrame(data)

    nr_entrance = len(self.listnames)
    L = nr_entrance
    # to create new entrance to the dictionary
    for i in range(L):
      first_index = self.indexlist[i]+1
      if i != L:
        last_index =self.indexlist[i+1]-1
      else:
        last_index = self.indexlist[i+1]
```

```
        list1 = self.sqrtlist[first_index:last_index]
        list2 = self.wavefkt[first_index:last_index]
        # print( len(list1), len(list2))
        self.newdict[self.listnames[i]] = newfram(list1 , list2)

    def plot_lineplots(self):
      plt.figure(figsize=(15, 10))
      for ent in self.listnames:
        newdataframe = self.newdict[ent]
        plt.plot(newdataframe['Regular_r'].tolist(),
        newdataframe['wavedensity'].tolist(), label=ent,  linewidth=5)
      plt.legend()
      plt.grid(True)
      plt.xlabel('Regular r')
      plt.ylabel('Density')
      plt.show()
```

It is easy to use this object class.

```
inputdata = pd.read_fwf("Be_2s2p3P.dat")

inputdata.rename(columns = {'sqrt(r)':'sqrt_r',
'P(nl;r)':'wavefunctions'}, inplace = True)
inputdata.columns

# to divide the dataframe into three chunks
list1 = inputdata['sqrt_r'].tolist()
list2 = inputdata['wavefunctions'].tolist()

model1 =  wavefunctionplot(list1 , list2 , ['1s', '2s', '2p'])
model1.breakframe()

model1.newdataframe()
model1.plot_lineplots()
```