MM220: Computational Lab

Amrita Bhattacharya

October 7, 2020

1 Introduction to python -III: plotting, integration and optimzation

1.1 Locating, adding, and importing libraries

Python has strong in built libraries for numerical and scientific operations such as Numpy, Scipy, Sympy etc. Locate your libraries and add path of the libraries in your preamble (if required).

Listing 1: locate

```
locate numpy
locate scipy
```

Add the following in your preamble (if required), depending upon your search result to the path of the libraries;

Listing 2: path

```
#!/usr/bin/python
#!/usr/share/pyshared/
#!/usr/share/pyshared/scipy
```

Import the libraries, depending on their need;

Listing 3: Import

```
import matplotlib.pyplot as plt;
import numpy as np;
import scipy.optimize as opt;
```

1.2 Polynomial fitting

Listing 4: polyfit

```
#!/usr/bin/env python
import numpy as np
import matplotlib.pyplot as plt

v = np.array([13.72, 14.83, 16.0, 17.23, 18.52])
e = np.array([-56.29, -56.41, -56.46, -56.46, -56.42])

print (v.shape, e.shape)

### fit a parabola to the data
# y = cx^2 + bx + a
a = np.polyfit(v,e,2)

print a

plt.plot(v,e)
#make a vector to evaluate fits on with a lot of points so it looks smooth
vfit = np.linspace(min(v),max(v),100)
plt.plot(vfit, (a[0]*vfit**2 + a[1]*vfit + a[2]))
plt.legend(('original', 'parabolic fit'))
plt.xlabel("Volume ($\AA^3$)")
plt.ylabel("Energy (eV)")
plt.savefig('eos.eps')
plt.show()
```

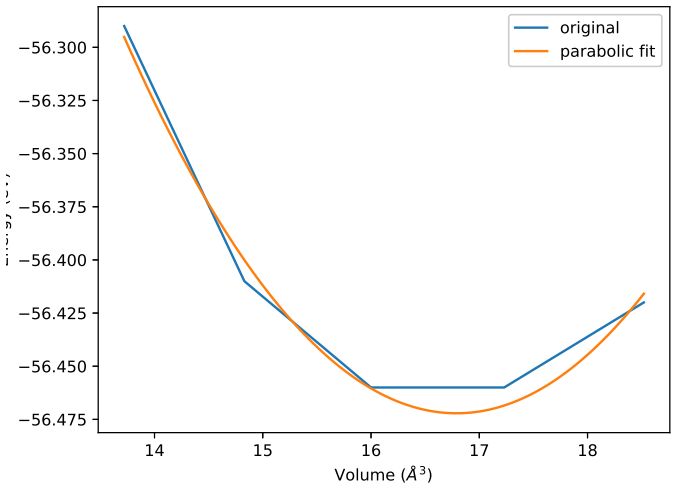

Figure 1: Plot showing the total energy plotted as a function of volume.

1.3 Integration

Listing 5: integration

```
import scipy
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad
import sympy

###### Define the function ########
def func(x):
    return 4*x**2 +2

###### Definite integration #######
def int_func(a, b):
    I= quad(func, a , b)
    return I[0]

print int_func(0, 2)

###### Symbolic integration #######
def int(x):
    x = sympy.symbols('x')
    val =sympy.integrate(func(x), x)
    return val

x = sympy.symbols("x")
value = int(x)
print (value)

######## Plotting done here ########
x = sympy.symbols('x')
intx=[]
for i in np.linspace(-3,3):
    intx.append(int(x).evalf(subs={x: i}))

x = np.linspace(-3,3)
plt.plot(x,func(x))
plt.plot(x,intx)
plt.xlabel("x")
plt.ylabel("y")
plt.legend(('f(x)', 'int(f(x))' ))
plt.savefig('int.eps')
plt.show()
```

The output of the given program should be as follows;

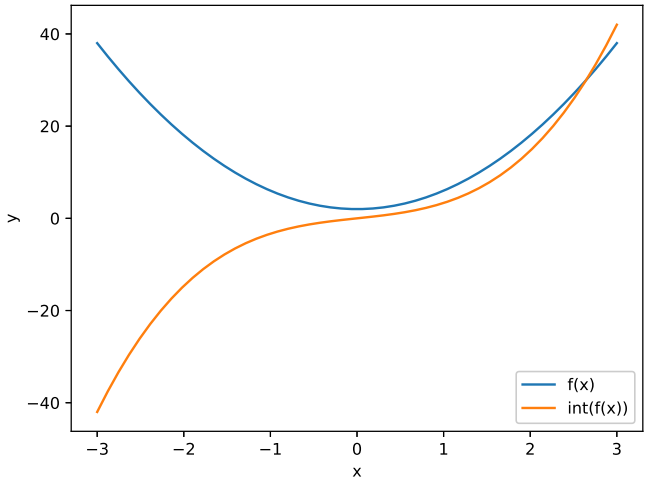

Figure 2: Plot of integration of a function.

1.4 Optimization

Listing 6: Optimize

```
import matplotlib.pyplot as plt;
import numpy as np;
import scipy.optimize as opt;

def func(x, a, b, c):
    return a * np.exp(-b * x) + c

xdata = np.linspace(0, 4, 50)
y = func(xdata, 2.5, 1.3, 0.5)
y_noise = 0.2 * np.random.normal(size=xdata.size)
ydata = y + y_noise

plt.plot(xdata, ydata, ".", label="Data");
popt, pcov = opt.curve_fit(func, xdata, ydata);

print (popt, pcov)

# Use the optimized parameters to plot the best fit
plt.plot(xdata, func(xdata, *popt), label="fit");

# Show the graph
plt.xlabel("x")
plt.ylabel("y")
plt.legend(('data', 'opt_plot' ))
plt.savefig('opt.eps')

plt.show();
```

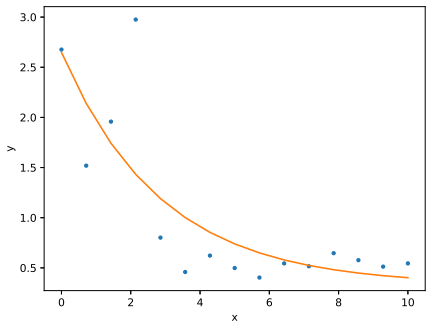

Figure 3: Plot showing the optimized fit to data points.

1.5 Reading writing file

Given a file "lat.dat". You can read, modify, and save it as "modlat.dat" using the following program.

Listing 7: R/W

```
import numpy as np;
########## Reading a file ############
file =open("lat.dat", "r")
mod_file = open("modlat.dat", "w")
data =[]
while True:
    line = file.readline()
    if line:
        data.append(list(map(float, line.split()[0:])))
    if not line:
        break
file.close()

print (data[0]), print (data[0][0])
############ Writing the file #############
for i in range(len(lat)):
  mod_file.write('%6.6f %6.6f \n' %(data[i][0], data[i][1] ))
mod_file.close()

############ Direct reading #################
x, y = np.loadtxt('lat.dat', delimiter=None, unpack=True)
print (x[0],y[0])
############ Plotting ####################
plt.xlabel("lattice_parameter (A)")
plt.ylabel("Energy (eV)")
plt.plot(x,y, label='Loaded from file!')
plt.savefig('latenergy.eps')
plt.show()
```

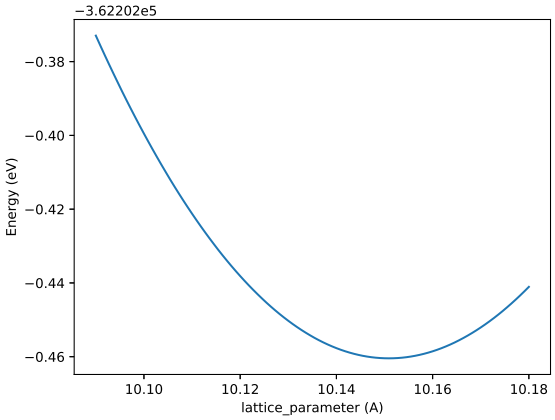

Figure 4: Plot showing the total energy plotted as a function of volume.

1.6 Main

In this last listing, we will discuss the concept behind __name__ == "__main__" in python. The main function (i.e. main.py) is the entry point for any python program (i.e. from where any program execution begins).

Listing 8: main

```
def cal_area(length, breadth):
    print("__name__:", __name__ )
    return length*breadth

if __name__=="__main__":
    print ("True")
    area = cal_area(10, 20)
    print(area)
```

The value of __name__ may be something other than "__main__", if you are importing it as a module from another file say (i.e. caller.py).

Listing 9: caller

```
import area
area.cal_area(10,20)
```

2 Assignments

1. $f(x) = xe^x$. Use symbolic python to write a program that integrates and plots $f(x)$ and $\int f(x)dx$ in the same graph for a range of $0 < x < 3$.

2. The Birch Murnaghan equation of state (BM EOS) is a relationship between the volume and energy of the solid;

$$E(V) = E_0 + \frac{9V_0B_0}{16}\left\{\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^3 B_0' + \left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^2 \left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]\right\} \quad (1)$$

The internal energy (E) is a function of the volume (V), where the parameters at the equilibrium i.e. the energy E_0, volume V_0, bulk modulus B_0 and it's derivative B_0' can be obtained from the fit of E and V.

In the following, we will use BM EOS fit to calculate the bulk modulus B_0 and it's derivative B_0' of a solid phase of Si clathrate.

(a) The file "lat.dat" gives the energy of a cubic bulk solid phase (Si clathrate) as a function of lattice parameter. Modify the file to create a new file called "latmod.dat", so that it gives energy as a function of volume of the solid. This relationship between volume and energy of the soild can be used for fitting the BM EOS.

(b) In order to fit the data to the BM EOS, you will have to make initial guesses of the parameters E_0, V_0, B_0 and B_0'. For this perform a polynomial fitting to the clathrate EOS and identify the coefficients a, b and c. So that the energy (E) is expanded as a function of volume (V) using the coefficients a, b and c and the parameters can be identified as follows;

Listing 10: Init

```
E =  a V^2 + b V + c

dE/dV = 2 a V + b

d^2E/dV^2 = 2a

V_0 = -b/2a (∵ dE/dV =0  at equillibrium)

E_0 = a V_0^2 +b V_0 + c

B_0 = V_0 d^2E/dV^2  = 2a V_0

B'_0 = 4 (∵ It is a small number usually)
```

Plot E as obtained in above with your original E-V data.

(c) Using the initial parameters as obtained from (b) set up a function that optimizes the energy using the BM EOS taking the initial parameters and volume as argument. Print the optimized equilibrium parameters i.e. the energy E_0, volume V_0 , bulk modulus B_0 and its derivative B'$_0$.