PHY 410
Homework Assignment 6

Han Wen

Person No. 50096432

October 26, 2014

Abstract

The goal of this assignment was to get familiar with the non-linear analysis, including finding minima and maxima for multiple dimension functions, and its application on models of NaCl.

Contents

1 Problem 1 **2**
- 1.1 Description . . . 2
- 1.2 Numerical Analysis . . . 2

2 Problem 2 **4**
- 2.1 Description . . . 4
- 2.2 Result . . . 4

3 Problem 3 **8**
- 3.1 Description . . . 8
- 3.2 Result . . . 8

A Appendix **11**
- A.1 python code . . . 11

1 Problem 1

1.1 Description

Now consider the 2-d function:

$$V(x, y) = -\frac{1}{2}r^2 + \frac{1}{4}r^4$$

where:

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

Try to minimize this function with the BFGS algorithm for various x,y values. What can you observe about the behavior? How many maxima are there? How many minima?

1.2 Numerical Analysis

To better evaluate its maxima and minima, I plotted the function with WolframAlpha interface [3] 1. We can see it has a "ring" of minima and one local maxima, when x,y approach infinite the function goes to infinite as well. Using the program to find the minima, with various initial guess of x and y (and the same accuracy 0.0000001)is shown in the table: 1

Table 1: trials

initial x	**initial y**	**iterations**	**final x**	**final y**
0.1	0.1	4	0.7071	0.7071
1.5	1	4	-0.8321	-0.5547
1	1.5	4	-0.5547	-0.8321
5	1	8	-0.9806	-0.1961
3	4	5	-0.6	-0.8

From the table we can find that for the x and y we input, the final out put will be proportional to the input with $x^2 + y^2 = 1$.

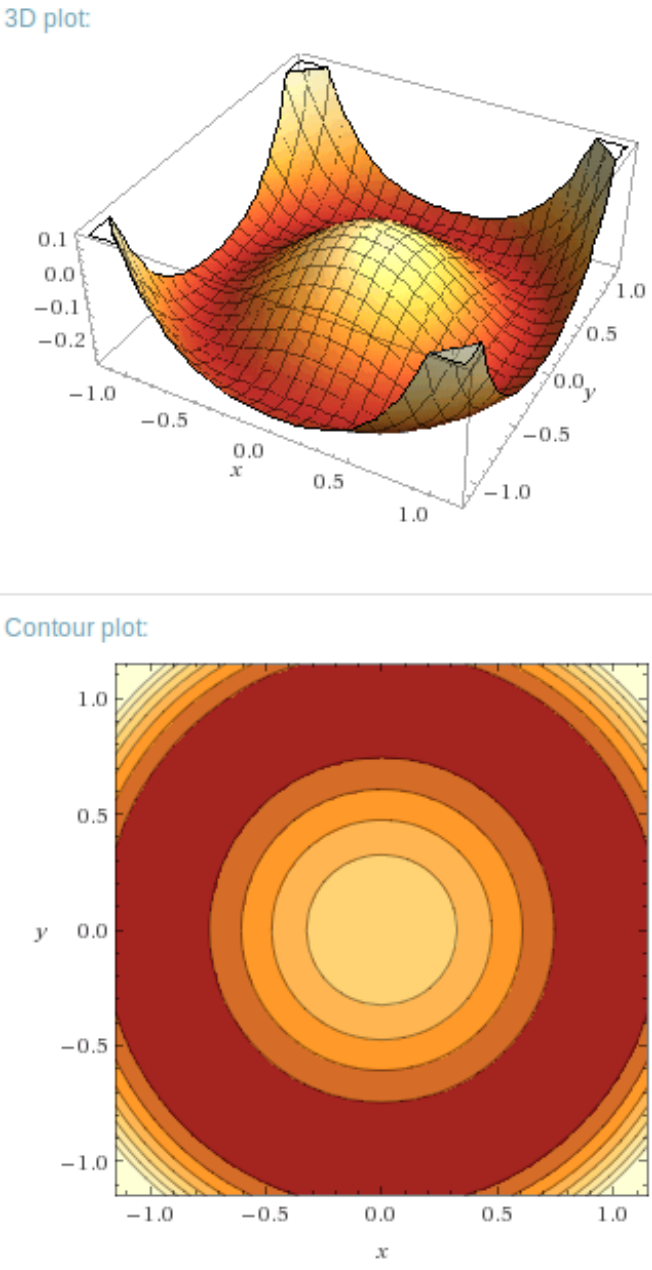

Figure 1: diagram of the function

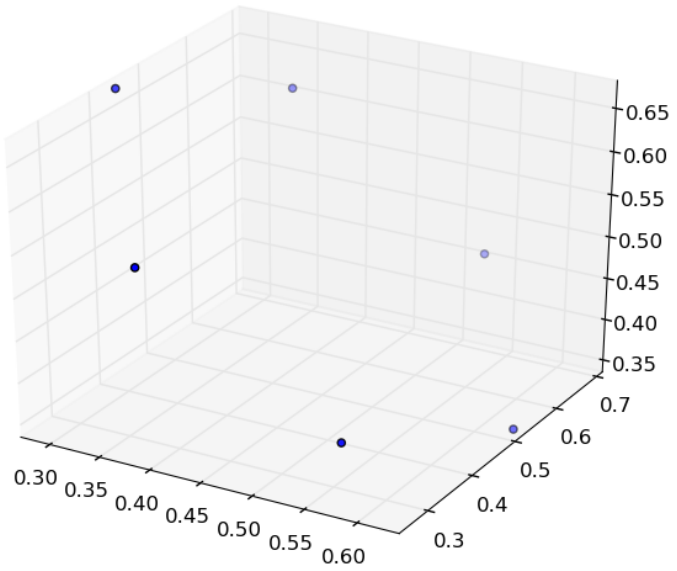

Figure 2: Na_3Cl_3 first configuration

2 Problem 2

2.1 Description

Determine the equilibrium configurations of NaCl clusters for n=3 and n=4. Plot them using the macros from class (x,y,z in scatter plots). To find all of the equilibrium configurations, use the strategy we talked about in class to determine all of the local minima. You can look at this paper for inspiration of how to find the correct configurations : K. Michaelian, "Evolving few-ion clusters of Na and Cl",Am. J. Phys. 66, 231 (1998). [2]

2.2 Result

To find the all the configurations, I used the random initial positions, yet due to the precision problem, I believe it's better I use this method to first generate lots of final minimized configurations, then pick the different configurations out of them, then set those configurations as input to better evaluate the minimized energy and coordinates.

For Na_3Cl_3, I found two different configurations:

First one 2, with binding energy -6.824eV

Second one 3, with binding energy -6.382eV

For Na_4Cl_4, I found two different configurations:

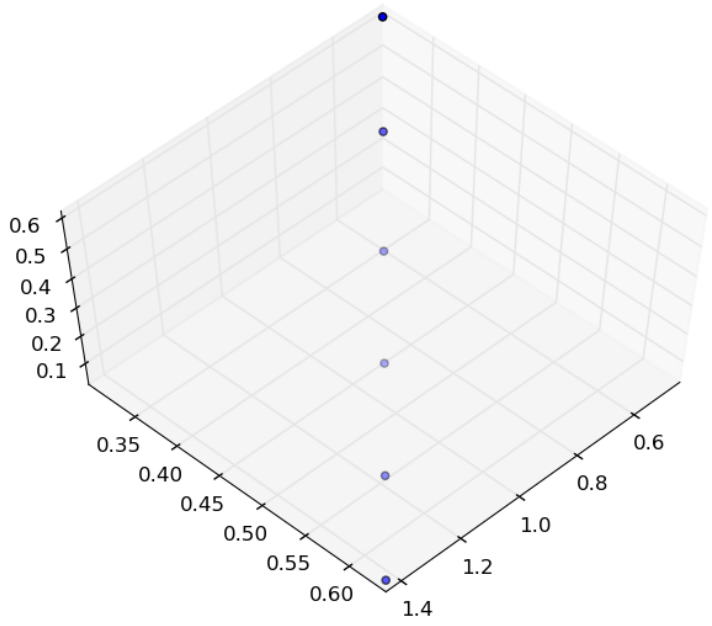

Figure 3: Na_3Cl_3 second configuration

First one 4, with binding energy -7.066eV

Second one 5, with binding energy -6.957eV

Third one 6, with binding energy -6.933eV

Fourth one 7, with binding energy -6.847eV

Fifth one 8, with binding energy -6.680eV

It took me a lot of time to find the right configuration. For in our program precision is a finite number, in fact, big enough to cause problems. Considering the whole procedure, when the difference is within the defined precision program assumes it's the right configuration, however, it is under lots of situations not the local minima, in these cases, the configurations are either distorted "right" configurations or meaningless. Despite I used random initial guess, still it's with some certain pattern, thus some configurations are quite difficult to reach. I managed to finally get all configurations in the reference by making guess with certain constrains and giving smaller precision.

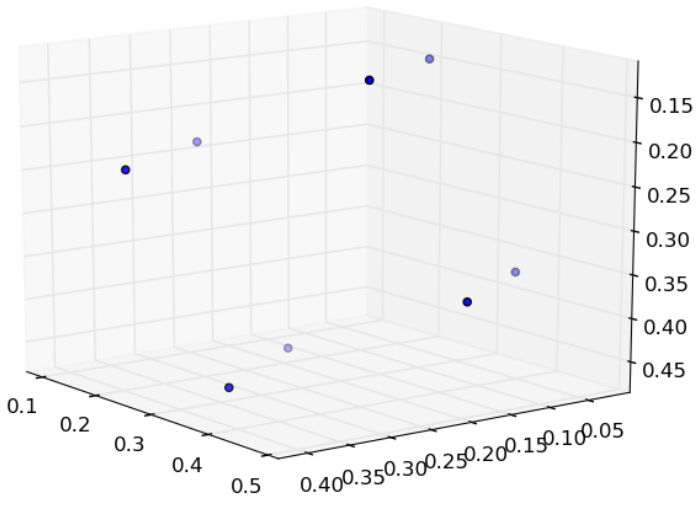

Figure 4: Na_4Cl_4 first configuration

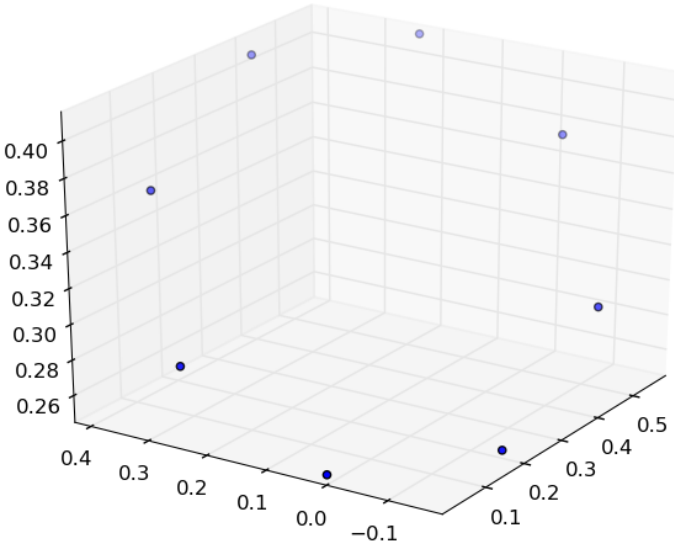

Figure 5: Na_4Cl_4 second configuration

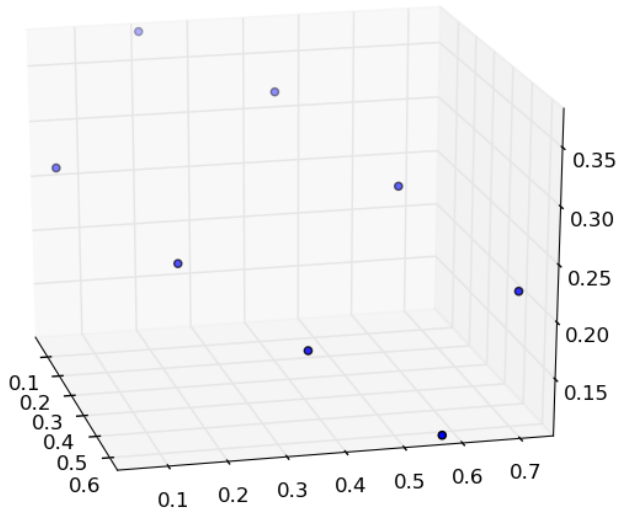

Figure 6: Na_4Cl_4 third configuration

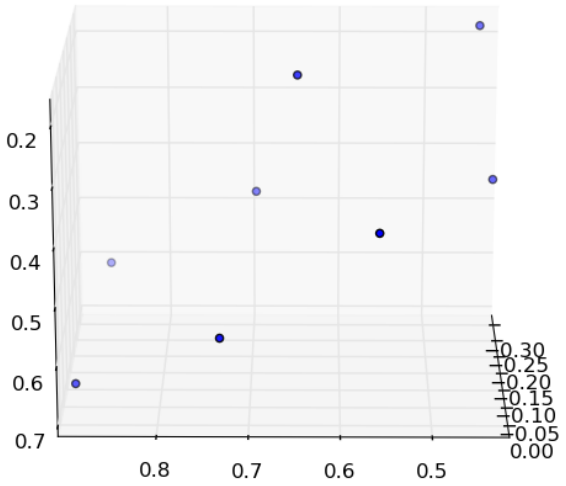

Figure 7: Na_4Cl_4 fourth configuration

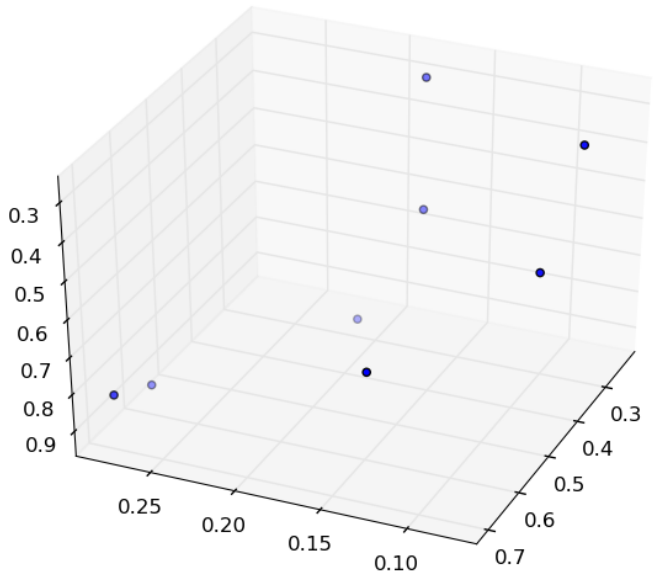

Figure 8: Na_4Cl_4 fifth configuration

3 Problem 3

3.1 Description

Download the NOAA Mauna Loa CO data set (included in the lecture 6 and lecture 7 example code, if the link is not working) and modify thecode to generate graphs using other columns in the data. Does the average rate of increase depend on which colums are used in a straight line fit? Explore linear models with 3 terms to decide whether the linear rate of increase is accelerating or decelerating. If the current rate of increase continues indefinitely, when will the concentration become toxic (50,000 ppm)?

Repeat this exercise, but instead of a linear fit, implement a quadratic fit to the data :

$$y = ax^2 + bx + c$$

Do this by minimizing the chi-squared function with BFGS or some other minimization algorithm. Compare to the simple matrix inversion fit of the linear problem as described in Lecture 14.

3.2 Result

I used the BFGS method to perform the minimization of chi-squared function when applying quadratic fit to the CO2 data. However, this case is a little complicated for the following reasons: First, if using the year directly, the numerical result for some steps will be too big, I consequently using the year-1953 as the x component Additionally, even if I applied the year transform, considering it is large amount of data we are dealing with, and the default setting of BFGS method

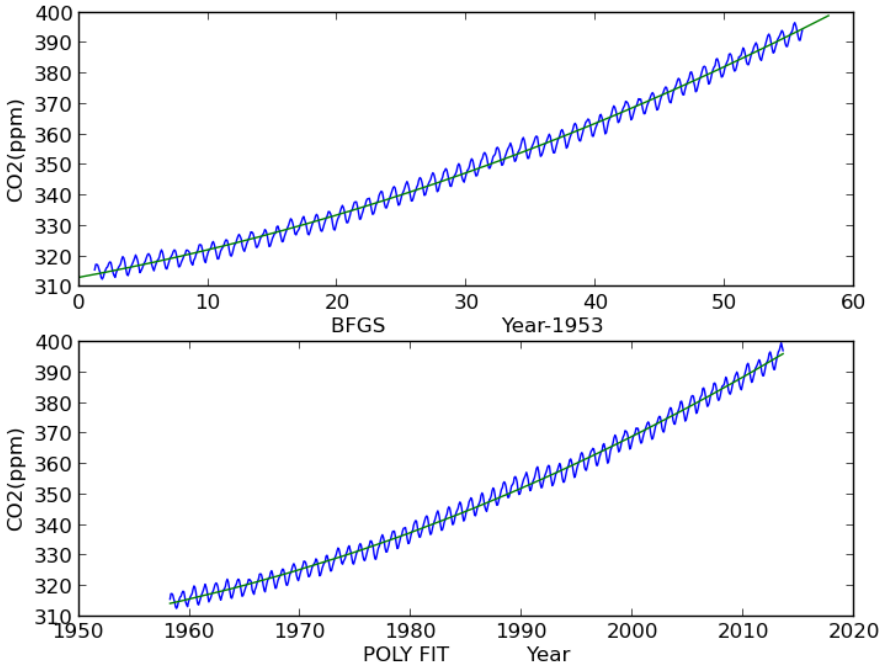

Figure 9: CO2 fit with both BFGS and poly fit

is to use an identity matrix for the first step, and our program in python do not support high enough precision, the minimization can fail under lots of circumstances. Therefore, I modified the program in two ways: one is before minimization, pick 3 points from the raw data and perform a Gauss-Jordan elimination to calculate a,b,c as the initial guess, so that the error can be small enough to continue iterations. Two, modified the σ in the chi-squared function so that the error can be small enough for further iterations.

Though the second modification is not quite justified, considering the specific case we are using, the limitation of python, and it is one out of three problems of one of the all homework assignments of one of the many courses, although my favourite one. I would say it is a rather elegant solution, combining the knowledge I've learned with new tools. And the result looks also quite optimal. Here is the plots of both BFGS and poly fit 9

Acknowledgements

I discussed this assignment with my classmates and used material from the cited references, but this write-up is my own.

References

[1] PHY 410-505 Webpage, `http://www.physics.buffalo.edu/phy410-505`.

[2] Evolving few-ion clusters of Na and Cl `http://scitation.aip.org/content/aapt/journal/ajp/66/3/10.1119/1.18851`

[3] `http://www.wolframalpha.com/input/?i=plot+-0.5*%28x^2%2By^2%29%2B0.25*%28x^2%2By^2%29^2`

A Appendix

A.1 python code

The following python code was used to obtain the results in this report:

```
from cpt import *
from math import *
import numpy as np


def f (p) :
    x = p[0]
    y = p[1]
    return -0.5*(x*x+y*y) + 0.25 *(x*x+y*y)*(x*x+y*y)

def df(p) :
    x = p[0]
    y = p[1]
    x1 = -x+x**3+x*y**2
    y1 = -y+y**3+y*x**2
    return np.array( [x1,y1] )


print " Minimization using Broyden-Fletcher-Goldfarb-Shanno Algorithm"
print " Find minimum of f(x,y) given an initial guess for x, y"
p = input(" Enter starting point coordinates x y: ")
gtol = input( " Enter desired accuracy: ")
f_min = 0.0
iterations = 0
res = scipy.optimize.fmin_bfgs(f=f, fprime=df,x0=p, gtol=gtol)
print res

from nacl import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import random

#I am going to use the random initial coordinates to generate random models.
#comparing energy and save new configurations.
name="Na3Cl3"        # for output files
nNa = 3
nCl = 3
n = nNa + nCl
a = 0.23                        #The equilibrium distance


#file_name = name + ".data"
```

```
#outfile = open( file_name, 'w' )

Enmin = []                          #array to store new minimum energy
Enmin.append(10000.0)               #Giving a big number to start
#r_Na = [  [ 0, 0, 0 ], [ 1.5*a, (3**0.5/2)*a, 0] , [0,(3**0.5)*a,0] ]
#r_Cl = [  [ a, 0, 0 ], [ -0.5*a, (3**0.5/2)*a, 0 ] , [a,(3**0.5)*a,0]]

#r_Na = [  [ 0, 0, 0 ], [ a, a, 0] ]
#r_Cl = [  [ a, 0, 0 ], [ 0, a, 0 ] ]

r_Na = [  [ 0, 0, 0 ], [ 1.5*a, (3**0.5/2)*a, 0] , [0,(3**0.5)*a,0] ]
r_Cl = [  [ 0, 0, 0 ], [ -0.5*a, (3**0.5/2)*a, 0 ] , [a,(3**0.5)*a,0]]
for im in range(100):
    #define a random initial for both Nas and Cls
    for vec in r_Na:
        for iv in range(nNa):
            vec[iv]=3*a*random.random()
    for vec in r_Cl:
        for iv in range(nNa):
            vec[iv]=3*a*random.random()
    # Initialize the cluster, add guesses at the
    # minimum arrangement.
    cluster = Cluster()

    for i in xrange(nNa) :
        r = Vector(r_Na[i])
        cluster.add(Na(r))

    for i in xrange(nCl) :
        r = Vector(r_Cl[i])
        cluster.add(Cl(r))

#     print " " + name + " cluster"+str(random.random())
#     print " Initial potential energy = " + str( cluster.potential_energy() )

    # Minimize the function
    accuracy = 1e-6

    res = cluster.minimize( accuracy )

    pe = res[1]
    iterations = res[4]
    #determine if it's exist in array
#     for energy in Enmin:


    # Print out resulting files, and also
    # plot the values in matplotlib
```

```
    print " Minimized potential energy = " + str(pe) + " eV"
    #outfile.write( " Minimized potential energy = " + str(pe) + " eV")
    #print " Binding energy of cluster  = " + str( pe / 2.0 ) + " eV"
    #print " Number of function calls = " + str( iterations )

    for i in xrange( nNa + nCl - 1) :
        for j in range(i+1,nNa+nCl) :
            rij = cluster.ion(i).r - cluster.ion(j).r
            dr = sqrt( np.dot(rij,rij) )
            s =  "(" + cluster.ion(i).name + ")-(" + cluster.ion(j).name + ")"
            print " " + s + " r_" + str(i) + str(j) + " = " + str( dr ) + " nm"
    print str(cluster)
#     outfile.write( str(cluster) )
#outfile.close()

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

#[x,y,z] = cluster.convert()
#ax.scatter( x,y,z )
#plt.show()

#This program is going to minimize chi-square for Co2, and compared to that of pol
#A quadratic fit will be used, AKA, y=ax^2+bx+c
#Han Wen
from cpt import *
import math
import matplotlib.pyplot as plt
import numpy as np


#Read data and store in x and y list, x for year y for ppm
file_name = "co2_mm_mlo.txt"
file = open(file_name, "r")
lines = file.readlines()
file.close()

x=[]
y=[]
for line in lines:
    if len(line) > 4:
        try:
            year = int(line[0:4])
            if year > 1957 and year < 2013:
                words = str.split(line)
                ppm = float(words[4])
                y.append(ppm)
                date=float(words[2])-1957.0
```

```
                x.append(float(date))                                   #to avoid prec
        except ValueError:
            pass


#Using thress points to have a resonable guess
print x[1],y[1],x[10],y[10],x[100],y[100]
m = Matrix(3,3)
m[0][0]=x[1]**2
m[0][1]=x[1]
m[0][2]=1
m[1][0]=x[10]**2
m[1][1]=x[10]
m[1][2]=1
m[2][0]=x[100]**2
m[2][1]=x[100]
m[2][2]=1

w=Matrix(3,1)
w[0][0]=y[1]
w[1][0]=y[10]
w[2][0]=y[100]
msave=Matrix_copy(m)
abc=Matrix_copy(w)
solve_Gauss_Jordan(m,abc)
print abc



def chisquare(p):
    a=p[0]
    b=p[1]
    c=p[2]
    sum=0.0
    for i in range(len(y)):
        sum += (y[i]-a*x[i]**2-b*x[i]-c)**2
    return 1.0*0.0001*sum                                               #sigma=1 for all i

def df(p):
    a=p[0]
    b=p[1]
    c=p[2]
    dfda=0.0
    dfdb=0.0
    dfdc=0.0
    for i in range(len(y)):
        dwhole=(y[i]-a*x[i]**2-b*x[i]-c)
        dfda += dwhole*(-x[i]**2)
```

```
        dfdb += dwhole*(-x[i])
        dfdc += dwhole*(-1.0)
    return np.array([dfda*0.01*2.0,dfdb*0.01*2.0,dfdc*0.01*2.0])

p=[abc[0][0],abc[1][0],abc[2][0]]                    #a,b,c
print abc[0][0],abc[1][0],abc[2][0]
#p=[0.001197,0.785956,313.268]
print df(p),chisquare(p)
iterations = 0
gtol=0.000001                              #accuracy
res = scipy.optimize.fmin_bfgs(f=chisquare, fprime=df,x0=p, gtol=gtol,norm=1000)
print res
print res[0]
print res[1]
print res[2]

#plot poly fit
file_name = "co2_polyfit_output_python.txt"
file = open(file_name, "r")
lines = file.readlines()
file.close()
pz=[]
px=[]
py=[]
for line in lines:
    if len(line) > 4:
        try:
            year = int(line[0:4])
            if year > 1957 and year < 2014:
                words = str.split(line)
                ppm = float(words[1])
                py.append(ppm)
                pz.append(float(words[2]))
                px.append(float(words[0]))
#to avoid precission loss, choose years-1957 as x
        except ValueError:
            pass

plt.subplot(2, 1, 1)
plt.plot( x, y )
plt.xlabel('BFGS                    Year-1953')
plt.ylabel('CO2(ppm)')
t=np.arange(0,58,0.01)
plt.plot(t,res[0]*t**2+res[1]*t+res[2])
#plt.plot(t,0.0119696*t**2+0.786*t+313.269)

plt.subplot(2, 1, 2)
plt.plot( px, py )
```

```
plt.xlabel('POLY FIT            Year')
plt.ylabel('CO2(ppm)')
plt.plot(px,pz)
plt.show()
```