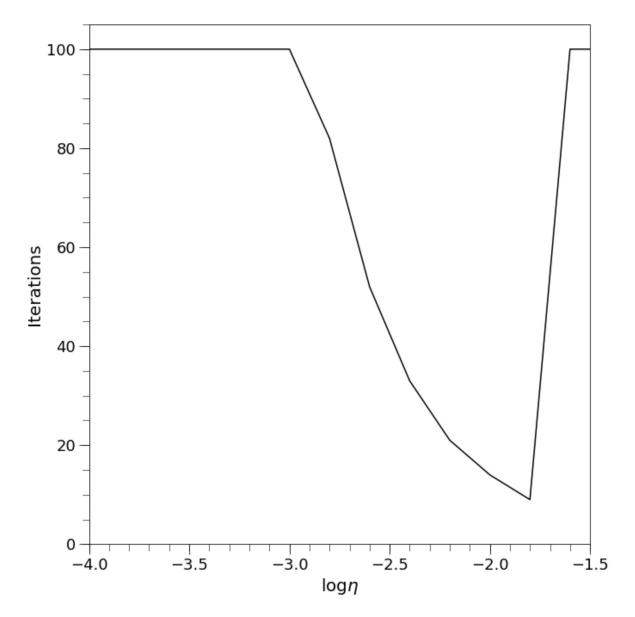
```
In [11]: | # -*- coding: utf-8 -*-
Created on Sun Oct 27 01:23:11 2019
@author: jorge
import sympy as sym
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
height = 10
width = 10
mpl.rcParams['figure.figsize'] = (width, height)
mpl.rcParams['font.size'] = 20
mpl.rcParams['figure.titlesize'] = 'small'
mpl.rcParams['legend.fontsize'] = 'small'
mpl.rcParams['xtick.major.size'] = 12
mpl.rcParams['xtick.minor.size'] = 8
mpl.rcParams['xtick.labelsize'] = 18
mpl.rcParams['ytick.major.size'] = 12
mpl.rcParams['ytick.minor.size'] = 8
mpl.rcParams['ytick.labelsize'] = 18
#get ipython().run line magic('matplotlib', 'inline')
#https://scipy-lectures.org/packages/sympy.html
# ^^ how to use sympy ^^
HX , HY = 50,50 #number of x,y points for countour
xmin, xmax = -15, 15
ymin, ymax = -12, 12
x1 = np.linspace(xmin, xmax, HX)
x2 = np.linspace(ymin,ymax,HY)
X1, X2 = np.meshgrid(x1, x2) # generate mesh grid
w1=sym.Symbol('w1') # define symbols
w2=sym.Symbol('w2')
j = (w1**2 + w2 - 11)**2 + (w1 + w2**2 - 7)**2 \# define equation
#compute gradient
j_grad1=sym.diff(j,w1)
j_grad2=sym.diff(j,w2)
#compute hessian
hess11=sym.diff(j grad1,w1)
hess12=sym.diff(j_grad1,w2)
hess21=sym.diff(j grad2,w1)
hess22=sym.diff(j grad2,w2)
# Routines for problems
jw thresh = 0.5
log etas = np.arange(-4, -1, 0.2, dtype=np.float) # learning rate
iters=[] # number of iterations until conveged
```

1 of 3

```
In [12]: fig, ax = plt.subplots()
for le in log_etas:
    eta = 10**le
    np.random.seed(seed=0)
    w=np.random.normal(0, 1, 2)
     j w=[]
    max iters=100 # number of iterations
    count= 1
    line=[]
    while(True):
         #compute gradient matrix and hessian matrix
         g= np.array([float(j grad1.subs({w1:w[0],w2:w[1]})),float(j grad2.subs
 (\{w1:w[0], w2:w[1]\})))
         H= np.array([[float(hess11.subs({w1:w[0],w2:w[1]})),float(hess12.subs({w...}))))
1:w[0],w2:w[1]}))],
                      [float (hess21.subs ({w1:w[0],w2:w[1]})), float (hess22.subs ({w...}))]
1:w[0],w2:w[1]))
         wnew = w-eta*q
         #loop check
         if( count>max_iters ):
             iters.append(max iters)
             break
         else:
             count=count + 1
             wprev=w.copy()
             w=wnew.copy()
             jw i = j.subs(\{w1:w[0], w2:w[1]\})
             jw.append(jw_i)
             try:
                 if(jw i <= jw thresh):</pre>
                     iters.append(count)
                     break
             except:
                 pass
print('Min Iterations @ log learning rate =', log etas[np.argmin(iters)])
ax.plot(log etas, iters, color='black')
ax.set_ylabel('Iterations')
ax.set xlabel(r'$\log\eta$')
ax.set ylim(0,105)
ax.set_xlim(-4, -1.5)
ax.xaxis.set_major_locator(mtick.MultipleLocator(0.5))
ax.xaxis.set_minor_locator(mtick.MultipleLocator(0.1))
ax.yaxis.set_major_locator(mtick.MultipleLocator(20))
ax.yaxis.set minor locator(mtick.MultipleLocator(5))
fig.tight layout()
 #plt.savefig('../prob6c.eps', dpi=500)
```

2 of 3 10/29/2019, 2:11 AM

Min Iterations @ log learning rate = -1.7999999999998



3 of 3