**Numerical Computing Project**



Muhammad Mustafa Hameed  
CS-151020

Dated – 18th April 2018

Submitted to:

Sir Salman Zaffar

**Using Newton’s method to find the extremum of Equation:**

**F(x,y)=**

**Using starting vector =**

**Code it with using any programming language ad contour plot its values:**

**Explanation With code too:**

# Solution:

1. # -\*- coding: utf-8 -\*-
2. """
4. Created on Tue Apr 17 16:21:43 2018
6. @author: mmustafa
8. """
10. #Mini Project Numerical Computing
11. #Name: Muhammad Mustafa Hameed
12. #Roll NO: CS151020
13. #Section: CS-6A
14. #Subject: Numerical Computing



19. **from** numpy.linalg **import** inv
20. **import** numpy as np
21. **import** matplotlib.pyplot as plt

24. **print**("Multi-Dimensional Unconstrained Optimization Problem")
26. #complete list for h and k varying from 0.1 to 1.0
27. newlist=[0.1,0.5,1.0]

First loop to iterate the newlist to change h

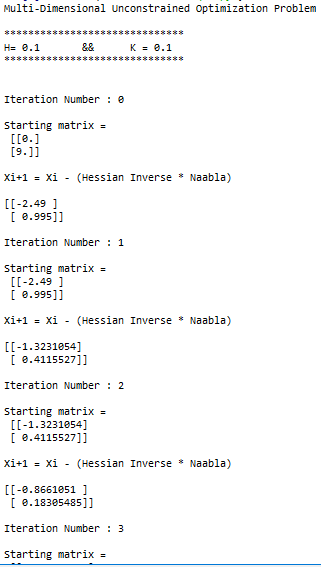
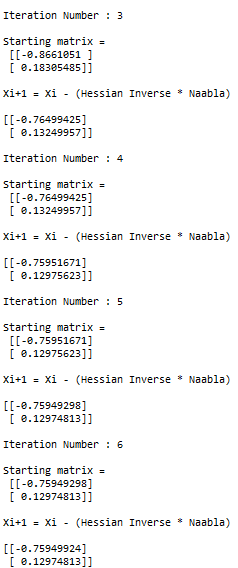
1. j=0
2. **while** True:
4. #declaring h and k
5. h=newlist[j]
6. k=newlist[j]
7. **print** ("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")
8. **print** ("H=",h,"      &&      K =",k)
9. **print** ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n")
10. #Declaring initial variables
11. x = 0.0
12. y = (1+5+1+0+2+0)
13. startMatrix= np.matrix([[x],[y]])
14. #Contour plot coding
15. xlist = np.linspace(-2.5, 1.0, 100)
16. ylist = np.linspace(-2.5, 1.0, 100)
17. X, Y = np.meshgrid(xlist, ylist)
18. Z = np.sqrt(X\*\*2 + Y\*\*2)
19. plt.figure()
20. cp = plt.contourf(X, Y, Z)
21. plt.colorbar(cp)
22. plt.title('Filled Contoured Plot of convergence')
23. myListX=[]
24. myListY=[]
25. plt.xlabel('x (cm)')
26. plt.ylabel('y (cm)')

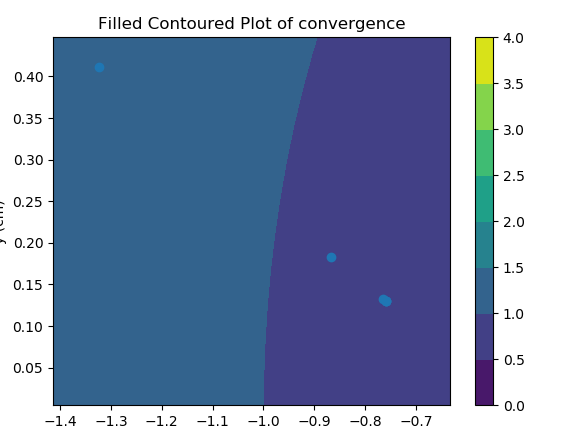
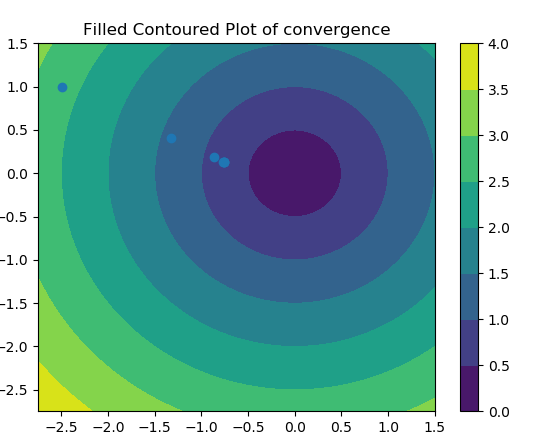
Second loop to iterate the function

1. i = 0
2. **while** True:
4. **print** ("Iteration Number :",i,"\n")
5. **print**("Starting matrix = \n",startMatrix,"\n")
7. #First Derivative i.e function'x
8. (Here I am using the central difference formula for first order differentiation that is f((x+h,y)-f(x-h,y))/2h)
10. var1 = ( ((x+h)\*\*3) + (2\*(x+h)\*y) + (2\*(y)\*\*2) - (2\*(x+h)) + (y) + 8 )
11. var2 = ( ((x-h)\*\*3) + (2\*(x-h)\*y) + (2\*(y)\*\*2) - (2\*(x-h)) + (y) + 8 )
13. fx = (var1 - var2)/(2 \* h)
15. (Here I am using the central difference formula for first order differentiation that is f((x,y+k)-f(x,y-k))/2h)
16. #First Derivative i.e function'y
17. var1 = ( ((x)\*\*3) + (2\*x\*(y+k)) + (2\*(y+k)\*\*2)  - (2\*(x)) + (y+k) + 8)
18. var2 = ( ((x)\*\*3) + (2\*x\*(y-k)) + (2\*(y-k)\*\*2)  - (2\*(x)) + (y-k) + 8)
20. fy = (var1 - var2)/(2 \* k)
22. (Here I am using the central difference formula for second order differentiation that is f((x+h,y)-2F(x,y)-f(x-h,y))/h^2)
23. #Second Derivative i.e function"x
24. var1 = ( ((x+h)\*\*3) + (2\*(x+h)\*y) + (2\*(y)\*\*2) - (2\*(x+h)) + (y) + 8)
25. var2 = ( ((x)\*\*3)   + (2\*(x)\*y)   + (2\*(y)\*\*2) - (2\*(x))   + (y) + 8)
26. var3 = ( ((x-h)\*\*3) + (2\*(x-h)\*y) + (2\*(y)\*\*2) - (2\*(x-h)) + (y) + 8)
28. fxx = (var1 - (2 \* var2) + var3)/((h)\*\*2)
29. (Here I am using the central difference formula for second order differentiation that is f((x,y+k)-2F(x,y)-f(x,y-k))/h^2)
31. #Second Derivative i.e function"y
32. var1 = ( ((x)\*\*3) + (2\*x\*(y+k)) + (2\*(y+k)\*\*2) - (2\*(x)) + (y+k) + 8)
33. var2 = ( ((x)\*\*3) + (2\*(x)\*y)   + (2\*(y)\*\*2)   - (2\*(x)) + (y)   + 8)
34. var3 = ( ((x)\*\*3) + (2\*x\*(y-k)) + (2\*(y-k)\*\*2) - (2\*(x)) + (y-k) + 8)
36. fyy = (var1 - (2 \* var2) + var3)/((k)\*\*2)
38. Here I am using the central difference formula for first order differentiation that is f((x+h,y+k)-F(x+h,y-k)-f(x-h,y+k)- f(x-h,y-k))/4hk)
40. #Second Derivative i.e function"xy
41. var1 = ( ((x+h)\*\*3)  + (2\*(x+h)\*(y+k)) + (2\*(y+k)\*\*2) - (2\*(x+h)) + (y+k) + 8)
42. var2 = ( ((x+h)\*\*3)  + (2\*(x+h)\*(y-k)) + (2\*(y-k)\*\*2) - (2\*(x+h)) + (y-k) + 8)
43. var3 = ( ((x-h)\*\*3)  + (2\*(x-h)\*(y+k)) + (2\*(y+k)\*\*2) - (2\*(x-h)) + (y+k) + 8)
44. var4 = ( ((x-h)\*\*3)  + (2\*(x-h)\*(y-k)) + (2\*(y-k)\*\*2) - (2\*(x-h)) + (y-k) + 8)
46. fxy = (var1 - var2 - var3 + var4)/(4\*h\*k)
48. var1 = fxx
49. var2 = fxy
50. var3 = fxy
51. var4 = fyy
52. vara = fx
53. varb = fy
55. #
57. naabla= np.array([[vara],[ varb]])
59. #            print("delta F Nable = \n",naabla,"\n")
61. Hessian= np.array([[var1,var2 ], [var3, var4]])
62. #            print("Hessian = \n",Hessian,"\n")
64. Hinv = inv(Hessian)
65. #            print("Hessian inverse = \n",Hinv,"\n")
67. startMatrix=startMatrix-(np.matmul(Hinv,naabla))
69. **print**("Xi+1 = Xi - (Hessian Inverse \* Naabla) \n")
70. **print**(startMatrix,"\n")
72. x=startMatrix[0,0]
73. x=float(format(x,".5f"))
75. y=startMatrix[1,0]
76. myListX.append([x])
77. myListY.append([y])
79. #breaking 2nd Loop
80. **if** i >10:
81. **break**
82. i = i + 1
83. plt.scatter(myListX,myListY)
84. plt.show()
85. #breaking 1st loop
86. **if** j >1:
87. **break**
88. j = j + 1

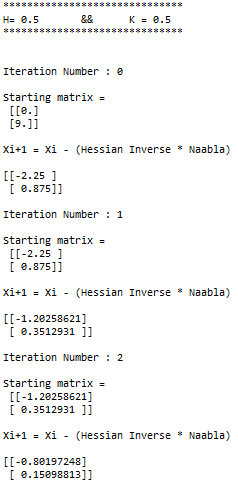
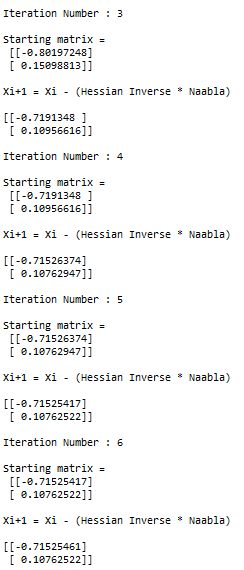
# OUTPUT:

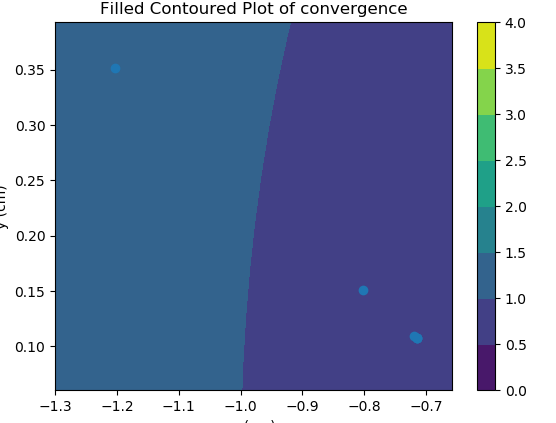
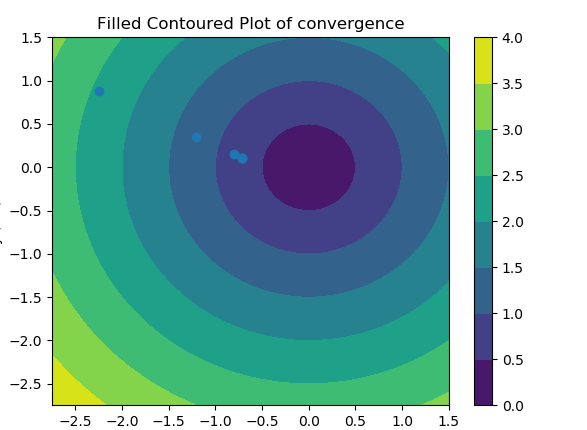
When H= 0.1 && K=0.1

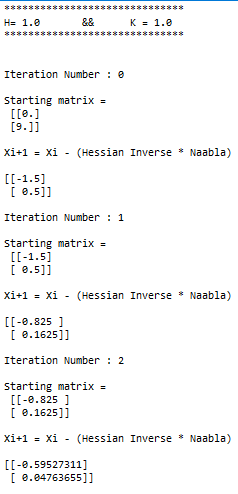
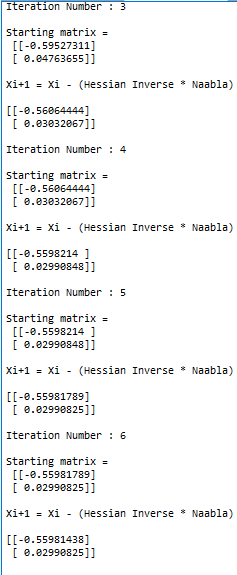


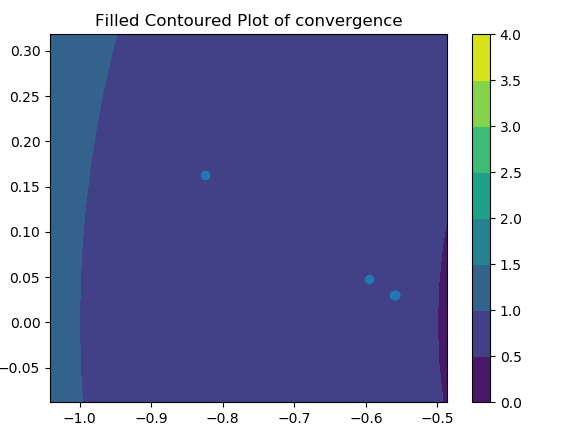
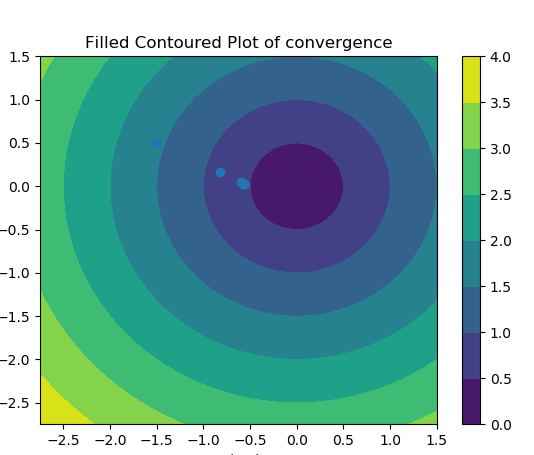
When H= 0.5 && K=0.5:



When H= 1.0 && K=1.0:



**Using Newton’s method to find the extremum of Equation:**

**F(x,y)=**

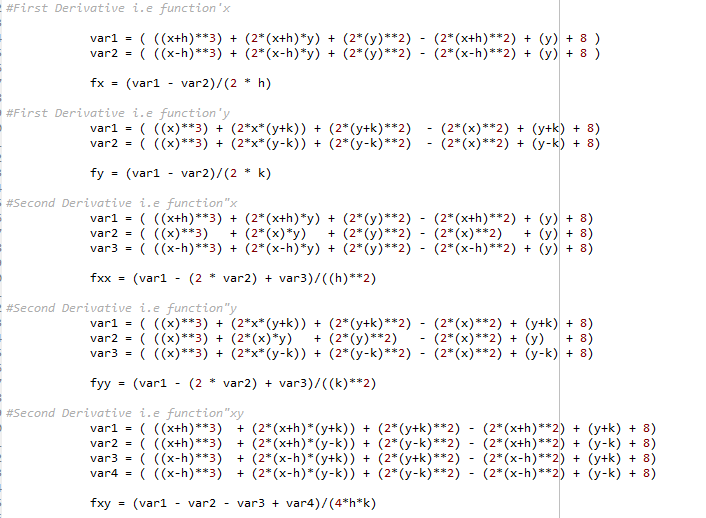
**Using starting vector =**

**Code it with using any programming language ad contour plot its values:**

**Explanation With code too:**

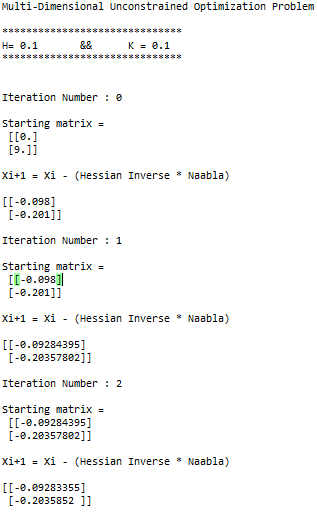
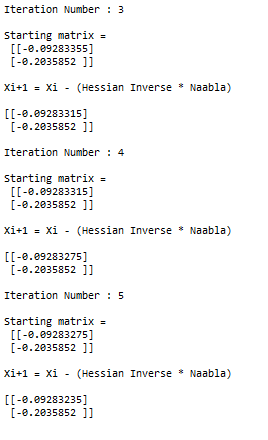
## Solution:

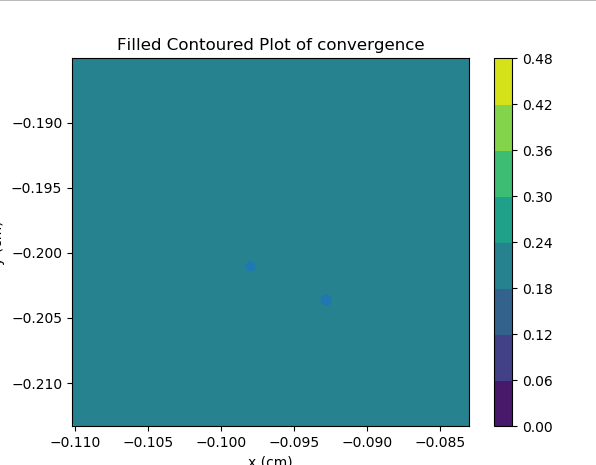
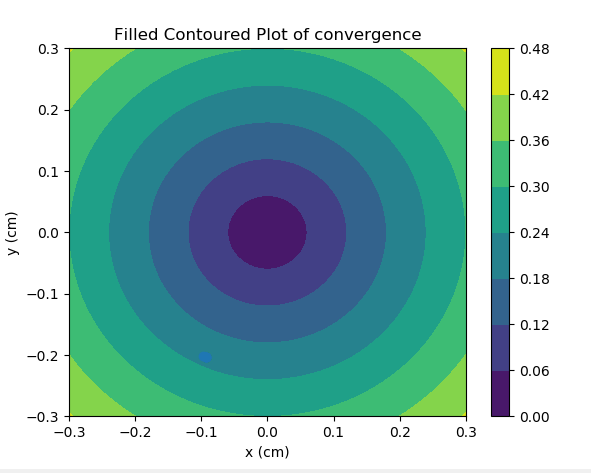
Same code used as before but changes are as follows:



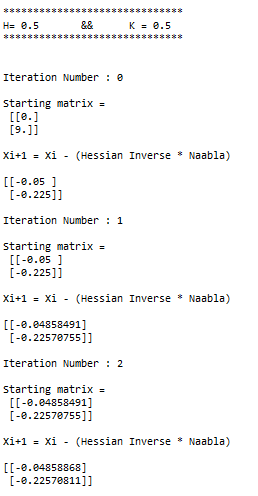
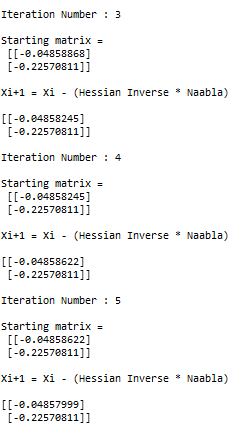
# OUTPUT:

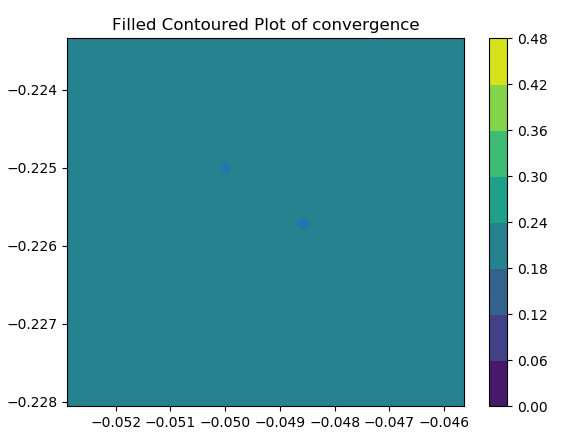
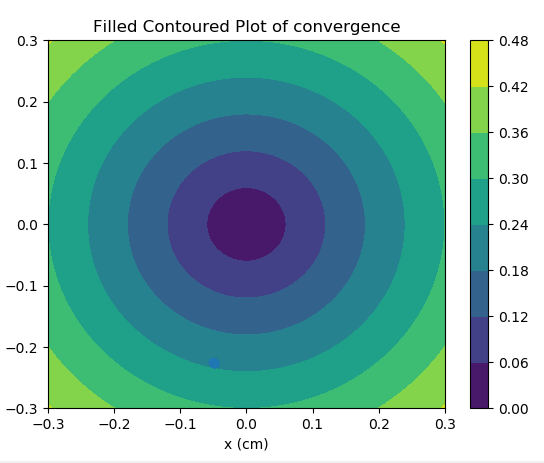
When H= 0.1 && K=0.1:



When H= 0.5 && K=0.5:



When H= 1.0 && K=1.0:

