
Question: Let $f(x) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$. At $x = (0, -1)$ draw the contour lines of the quadratic model Equation 1 assuming that B is the Hessian of f. Draw the family of solutions of Equation 2 as the trust region radius varies from $\Delta = 0$ to $\Delta = 2$. Repeat this at $x = (0, 0.5)$.

Quadratic Model: Taylor-series expansion of f around x_k is,

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla^2 f(x_k + tp) p \tag{1}$$

where $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$, and t is some scalar in the interval $(0, 1)$.
By using an approximation B_k to the Hessian in the second-order term, m_k is defined as follows:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$$

where B_k is some symmetric matrix. The difference between $m_k(p)$ and $f(x_k + p)$ is $\mathcal{O}(\|p\|^2)$, which is small when p is small.
To obtain each step, we seek a solution of the subproblem,

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \quad \text{such that } \|p\| \leq \Delta_k \tag{2}$$

where, $\Delta_k > 0$ is the trust-region radius.
Theorem 1: The vector p^* is a global solution of the trust-region problem,

$$\min_{p \in \mathbb{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p \quad \text{such that } \|p\| \leq \Delta_k$$

if and only if p^* is feasible and there is a scalar $\lambda \geq 0$ such that the following conditions are satisfied:

$$\begin{aligned}
&(B + \lambda I)p^* = -g && \text{(condition 1)}\\
&\lambda(\Delta - \|p^*\|) = 0 && \text{(condition 2)}\\
&(B + \lambda I) \quad \text{is positive semidefinite} && \text{(condition 3)}
\end{aligned}$$

Now, we have,

$$\begin{aligned}
f(x) &= 10(x_2 - x_1^2)^2 + (1 - x_1)^2\\
\Rightarrow \nabla f &= \begin{bmatrix} -40x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 20(x_2 - x_1^2) \end{bmatrix}\\
\Rightarrow \nabla^2 f &= \begin{bmatrix} -40x_2 + 120x_1^2 + 2 & -40x_1 \\ -40x_1 & 20 \end{bmatrix}
\end{aligned}$$

Now, by Theorem 1 p^* is a global solution of the trust-region for Equation 2, i.e.

$$p^* = arg\left(\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p\right)$$

if and only if p^* is feasible and $\exists \lambda \geq 0$ such that,

$$\begin{aligned}
&(B + \lambda I)p^* = -\nabla f && \text{(condition 1)}\\
&\lambda(\Delta - \|p^*\|) = 0 && \text{(condition 2)}\\
&(B + \lambda I) \geq 0 && \text{(condition 3)}
\end{aligned}$$

Part I: $x = (0, -1)$

At $x = (0, -1)$, we have the following,

$$f_k = 10(-1-0^2)^2 + (1-0)^2 = 11$$
$$g_k = \nabla f_k = \begin{bmatrix} -40 \times 0(-1-0^2) - 2(1-0) \\ 20(-1-0^2) \end{bmatrix} = \begin{bmatrix} -2 \\ -20 \end{bmatrix}$$
$$B_k = \nabla^2 f_k = \begin{bmatrix} -40(-1) + 120(0)^2 + 2 & -40(0) \\ -40(0) & 20 \end{bmatrix} = \begin{bmatrix} 42 & 0 \\ 0 & 20 \end{bmatrix}$$

Contour lines of the quadratic model Eq. 1: From equation 1, we have,

$$\begin{aligned} m_k(p) &= f_k + g_k^T p + \frac{1}{2} p^T B_k p \\ &= 11 + \begin{bmatrix} -2 \\ -20 \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}^T \begin{bmatrix} 42 & 0 \\ 0 & 20 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \\ &= 11 - (2p_1 + 20p_2) + \frac{1}{2}(42p_1^2 + 20p_2^2) \\ &= 11 - 2p_1 - 20p_2 + 21p_1^2 + 10p_2^2 \end{aligned}$$

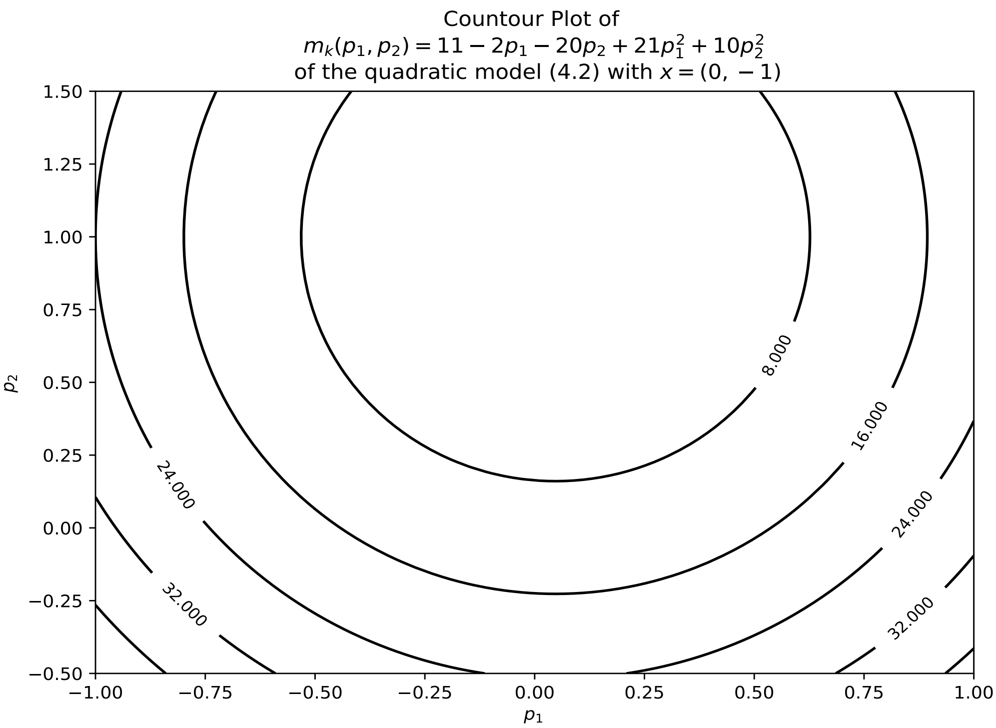

Figure 1: Countour Plot of $m_k(p_1, p_2)$ of the quadratic model Eq. 1 with $x = (0, -1)$

Python Code 1: Countour Plot of $m_k(p_1, p_2)$ - the quadratic model Eq. 1 with $x = (0, -1)$

```
import matplotlib
import numpy as np
```

```python
import matplotlib.pyplot as plt

#Initialization
xmin = -1
xmax = 1     # x interval [xmin, xmax]
ymin = -0.5
ymax = 1.5   # y interval [ymin, ymax]
nx = 100     # number of points in [xmin, xmax]
ny = 100     # number of points in [ymin, ymax]
#Create x, y axis
x = np.linspace(xmin, xmax, nx)
y = np.linspace(ymin, ymax, ny)
#Create mesh
X, Y = np.meshgrid(x, y)
#Define fuction
def f(x, y):
    return 11-2*x-20*y+21*x*x+10*y*y
#Evaluate functional values
Z = f(X, Y)
# Plot Contour
matplotlib.rcParams['contour.negative_linestyle'] = 'solid'
CS = plt.contour(X, Y, Z, 6, colors='k',)
plt.clabel(CS, fontsize=9, inline=1)
plt.title("Countour Plot of \n"
          r"$m_k(p_1,p_2)=11-2p_1-20p_2+21p_1^2+10p_2^2$"
          "\n of the quadratic model "
          r"$ (4.2) $" " with " r"$x=(0,-1)$")
plt.xlabel(r"$p_1$")
plt.ylabel(r"$p_2$")
plt.show()
```

The family of solutions of Eq. 2: Now, we have to draw the family of solutions of Eq. 2 as the trust region radius varies from $\Delta = 0$ to $\Delta = 2$. Let us recall, Equation 2 is,

$$\min_{p\in\mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2}p^T B_k p \quad \text{such that } \|p\| \leq \Delta_k$$

Now, using condition 1 we have,

$$\begin{aligned}
&(B_k + \lambda I)p^* = -\nabla f_k \\
\Rightarrow &\begin{bmatrix} 42+\lambda & 0 \\ 0 & 20+\lambda \end{bmatrix} p^* = \begin{bmatrix} 2 \\ 20 \end{bmatrix} \\
\Rightarrow &p^* = \begin{bmatrix} 42+\lambda & 0 \\ 0 & 20+\lambda \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 20 \end{bmatrix} \\
\Rightarrow &p^* = \begin{bmatrix} \frac{2}{42+\lambda} \\ \frac{20}{20+\lambda} \end{bmatrix} \\
\Rightarrow &\|p^*\| = \sqrt{\left(\frac{2}{42+\lambda}\right)^2 + \left(\frac{20}{20+\lambda}\right)^2}
\end{aligned}$$

We seek for the solution p^* such that $\|p^*\| \leq \Delta_k$. Clearly, $arg\left(\max_{\lambda\geq 0}\|p^*\|\right) = 0$ and

$$\max_{\lambda\geq 0}\|p^*\| = \sqrt{\left(\frac{2}{42}\right)^2 + \left(\frac{20}{20}\right)^2} = \frac{\sqrt{442}}{21}$$

Hence,

$$\|p^*\| = \begin{cases} \dfrac{\sqrt{442}}{21} & \text{if } \lambda = 0 \\ \sqrt{\left(\dfrac{2}{42+\lambda}\right)^2 + \left(\dfrac{20}{20+\lambda}\right)^2} < \dfrac{\sqrt{442}}{21} & \text{if } \lambda > 0 \end{cases}$$

Now, using condition 2 we have,

$$\lambda(\Delta_k - \|p^*\|) = 0$$

Which implies from the condition $\|p^*\| \leq \Delta_k$,

$$\Delta_k \in \begin{cases} \left[\dfrac{\sqrt{442}}{21}, \infty\right) & \text{if } \lambda = 0 \\ \left(0, \sqrt{\left(\dfrac{2}{42+\lambda}\right)^2 + \left(\dfrac{20}{20+\lambda}\right)^2}\right) \subseteq \left(0, \dfrac{\sqrt{442}}{21}\right) & \text{if } \lambda > 0 \end{cases}$$

Thus, finally we get the solution p^* as,

$$p^* = \begin{cases} \begin{bmatrix} \frac{1}{21} \\ 1 \end{bmatrix} & \text{if } \Delta_k \in \left[\frac{\sqrt{442}}{21}, \infty\right) \\ \begin{bmatrix} \frac{2}{42+\lambda} \\ \frac{20}{20+\lambda} \end{bmatrix} & \text{if } \Delta_k \in \left(0, \sqrt{\left(\frac{2}{42+\lambda}\right)^2 + \left(\frac{20}{20+\lambda}\right)^2}\right) \subseteq \left(0, \frac{\sqrt{442}}{21}\right) \end{cases}$$

Now, as the trust region radius varies from $\Delta = 0$ to $\Delta = 2$, we have,

$$p^* = \begin{cases} \begin{bmatrix} \frac{1}{21} \\ 1 \end{bmatrix} & \text{if } \Delta_k \in \left[\frac{\sqrt{442}}{21}, 2\right) \\ \begin{bmatrix} \frac{2}{42+\lambda} \\ \frac{20}{20+\lambda} \end{bmatrix} & \text{if } \Delta_k \in \left(0, \frac{\sqrt{442}}{21}\right) \end{cases}$$

Using Python, the trust region method has been implemented where the subproblem was handled by Newton's method.

Python Code 2: Trust region method - trust radius varies from $\Delta = 0$ to $\Delta = 2$ for $x = (0, 0.5)$

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.lines as mlines

###### Initialization #########
# This is the code for Part I: x = (0,-1),
# i.e. x1 = 0, x2 = -1 Which is the starting point
x0 = 0
y0 = -1
gs = 3  # Grid Size
ng = 500     # Grid steps
Dmax = 2     # Maximum TR radius
nD = 200 # Number of steps for Trust radius
nth = 200 # Angle steps
```

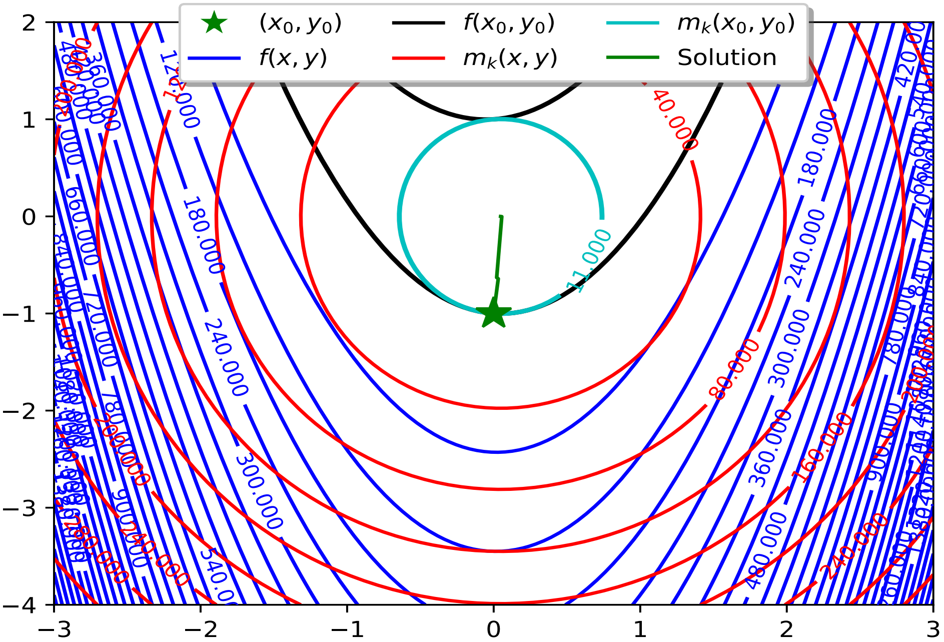

Figure 2: The family of solutions of Eq. 2 as the trust region radius varies from $\Delta = 0$ to $\Delta = 2$ for $x = (0, -1)$

```
# Plot the initial point

#Create x, y axis
xmin = x0-gs
xmax = x0+gs
ymin = y0-gs
ymax = y0+gs
x = np.linspace(xmin, xmax, ng)
y = np.linspace(ymin, ymax, ng)
#Create mesh
X, Y = np.meshgrid(x, y)
#Define fuction
def f(x, y):
    return 10*np.multiply(((y-np.multiply(x,x))),((y-np.multiply(x,x)))) +
    np.multiply(1-x,1-x)
#Evaluate functional values
Z = f(X, Y)
f0 = 10*(y0-x0**2)**2 + (1-x0)**2;    # initial function values
g0 = np.matrix([[-40*x0*(y0-x0**2)-2*(1-x0)], [20*(y0-x0**2)]])
B0 = np.matrix([[120*x0**2-40*y0+2, -40*x0], [-40*x0, 20]])
M = f0 + g0.item(0)*(X-x0)+g0.item(1)*(Y-y0) + 0.5*B0.item((0,0))*np.
    multiply((X-x0),(X-x0)) + B0.item((0,1))*np.multiply((X-x0),(Y-y0)) +
    0.5*B0.item((1,1))*np.multiply((Y-y0),(Y-y0))
# Plot Contour
plt.plot([x0], [y0], 'g*', markersize=15)
x0y0_legend = mlines.Line2D([], [], color='w', marker='*', markerfacecolor=
    'g',
                            markersize=15, label='$(x_0,y_0)$')
```

```python

matplotlib.rcParams['contour.negative_linestyle'] = 'solid'
CS = plt.contour(X, Y, Z, 30, colors='b', )
plt.clabel(CS, fontsize=9, inline=1)
f_legend = mlines.Line2D([], [], color='b', label='$f(x,y)$')

CS = plt.contour(X, Y, Z,levels = [f0],
                    colors=('k',),linestyles=('--',),linewidths=(2,))
plt.clabel(CS, fontsize=9, inline=1)
f_legend0 = mlines.Line2D([], [], color='k', label='$f(x_0,y_0)$')

CS = plt.contour(X, Y, M, 8, colors='r',)
plt.clabel(CS, fontsize=9, inline=1)
M_legend = mlines.Line2D([], [], color='r', label='$m_k(x,y)$')

CS = plt.contour(X, Y, M,levels = [f0],
                    colors=('c',),linestyles=('--',),linewidths=(2,))
plt.clabel(CS, fontsize=9, inline=1)
M_legend0 = mlines.Line2D([], [], color='c', label='$m_k(x_0,y_0)$')
# Solving subproblem
pN = -np.linalg.inv(B0)*g0  #Newton Step
XN = x0 + pN.item(0)
YN = y0 + pN.item(1)
MN = f0 + g0.item(0)*(XN-x0)+g0.item(1)*(YN-y0) + 0.5*B0.item((0,0))*np.multiply((XN-x0),(XN-x0)) + B0.item((0,1))*np.multiply((XN-x0),(YN-y0)) + 0.5*B0.item((1,1))*np.multiply((YN-y0),(YN-y0))
dD = Dmax/nD
xtc = np.zeros(nD)
ytc = np.zeros(nD)
th = np.linspace(0,2*np.pi,nth)
ct = np.cos(th)
st = np.sin(th)
for k in range(1, nD+1):
     Delta = k*dD;
     X = x0 + Delta*ct
     Y = y0 + Delta*st
     M = f0 + g0.item(0)*(X-x0)+g0.item(1)*(Y-y0) + 0.5*B0.item((0,0))*np.multiply((X-x0),(X-x0)) + B0.item((0,1))*np.multiply((X-x0),(Y-y0)) + 0.5*B0.item((1,1))*np.multiply((Y-y0),(Y-y0))
     lowpt = [i for i, v in enumerate(M) if v == min(M)] # minpos = [i for i, v in enumerate(array name here) if v == min(array name here)]
     lowpt = lowpt[0]
     nP = np.linalg.norm(np.subtract([XN, YN],[x0,y0]))
     if (nP>Delta) or (MN > M[lowpt]):
          xtc[k-1] = X[lowpt];
          ytc[k-1] = Y[lowpt];
     else :
          xtc[k-1] = XN;
          ytc[k-1] = YN;
Ax = np.insert(xtc,0,x0)
Ay = np.insert(ytc,0,y0)
plt.plot(Ax, Ay, 'g')
sol_legend = mlines.Line2D([], [], color='g', label='Solution')
plt.legend(handles=[x0y0_legend, f_legend, f_legend0, M_legend, M_legend0, sol_legend],loc='upper center', bbox_to_anchor=(0.5, 1.05),ncol=3, fancybox=True, shadow=True)
plt.show()
```

Part II: $x = (0, 0.5)$

Using the exact same process we have the following output.

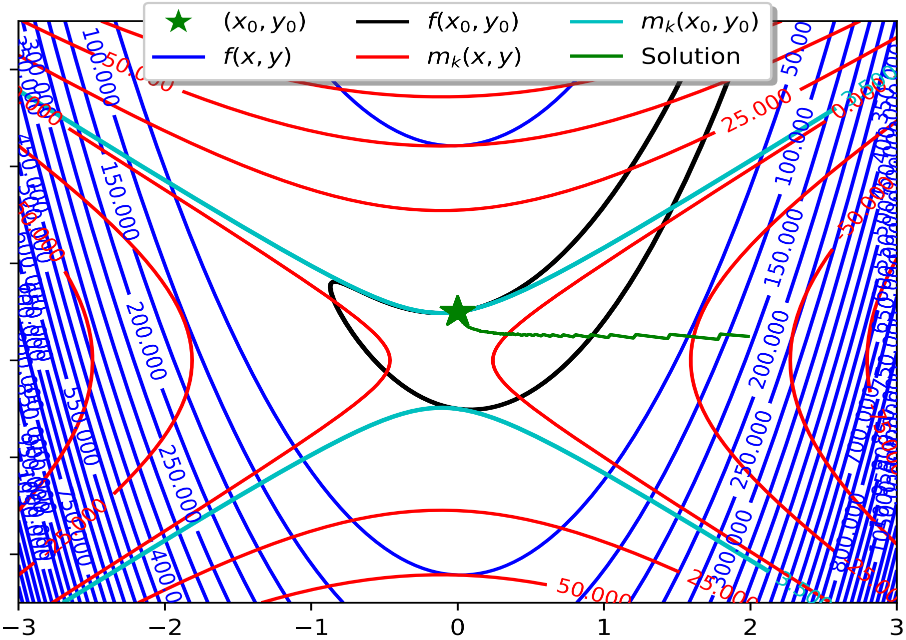

Figure 3: The family of solutions of Eq. 2 as the trust region radius varies from $\Delta = 0$ to $\Delta = 2$ for $x = (0, 0.5)$ and the Countour Plot of $m_k(p_1, p_2)$ of the quadratic model Eq. 1 with $x = (0, 0.5)$