Chapter 8: Performance Surfaces and Optimum Points

Brandon Morgan

1/15/2021

E8.7

We are given the following vector function

$$F(x) = \frac{1}{2}X^T \begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix} X + \begin{bmatrix} 4 - 4 \end{bmatrix} X + 2$$

1

Our function is in qudratic form from Eq. (8.35), $F(x) = \frac{1}{2}X^T AX + d^T X + c$. From Eq. (8.38), the gradient is given by $\nabla F(X) = AX + d$. The Hessian is given by Eq. (8.39), $\nabla^2 F(X) = A$.

Thus, our gradient is the following:

$$\nabla F(X) = \begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix} X + \begin{bmatrix} 4 \\ -4 \end{bmatrix}$$

and our Hessian is the following:

$$\nabla^2 F(X) = \begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix}$$

2

```r
library(purrr) # used for map2_dbl function
library(plotly) # used for 3D plotting
```

```
## Warning: package 'plotly' was built under R version 3.6.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(ggplot2) # used for plotting
```

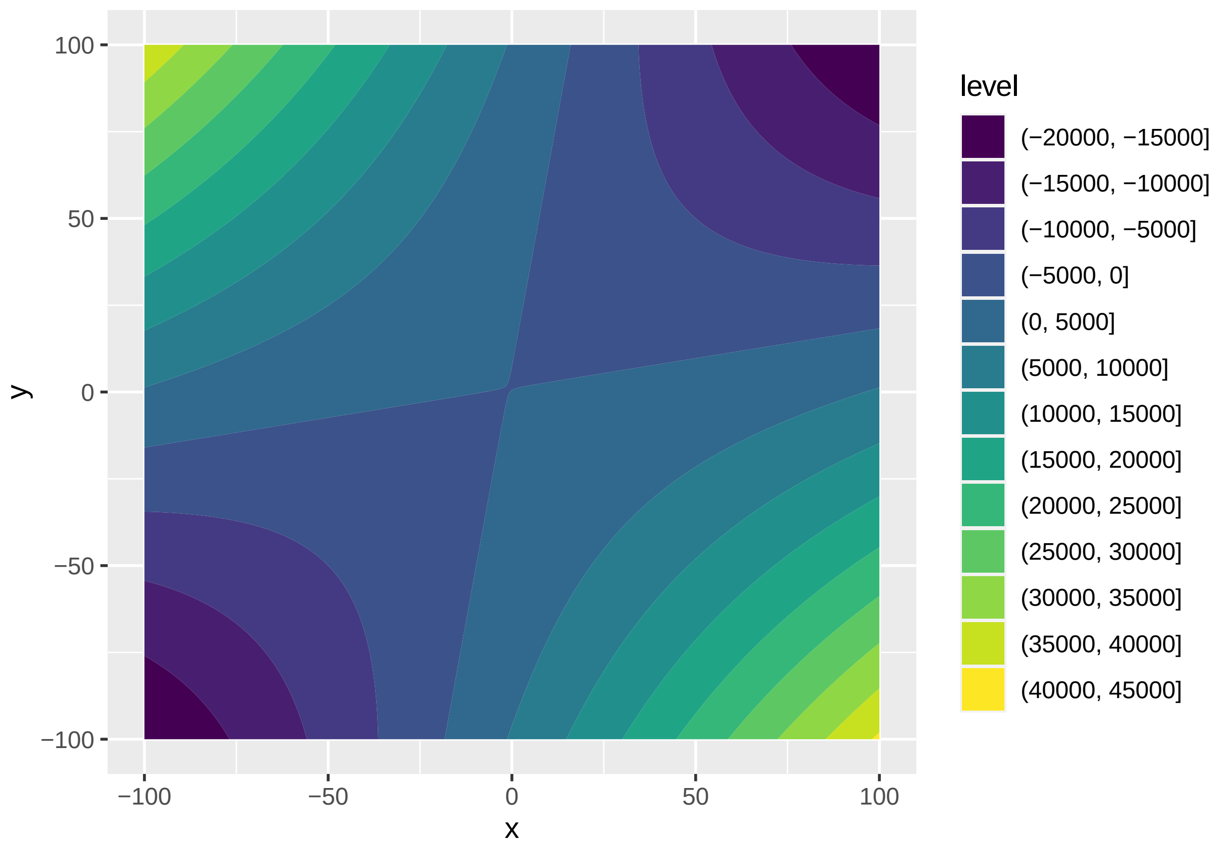
Here we have the contour plot of our function:

```r
fun = function(x, y) {
  X = matrix(c(x, y), ncol=1)
  A = matrix(c(1, -3, -3, 1),ncol=2)
  d = matrix(c(4, -4), ncol=1)
  c = 2

  0.5*t(X)%*%A%*%X+t(d)%*%X+2
}
```

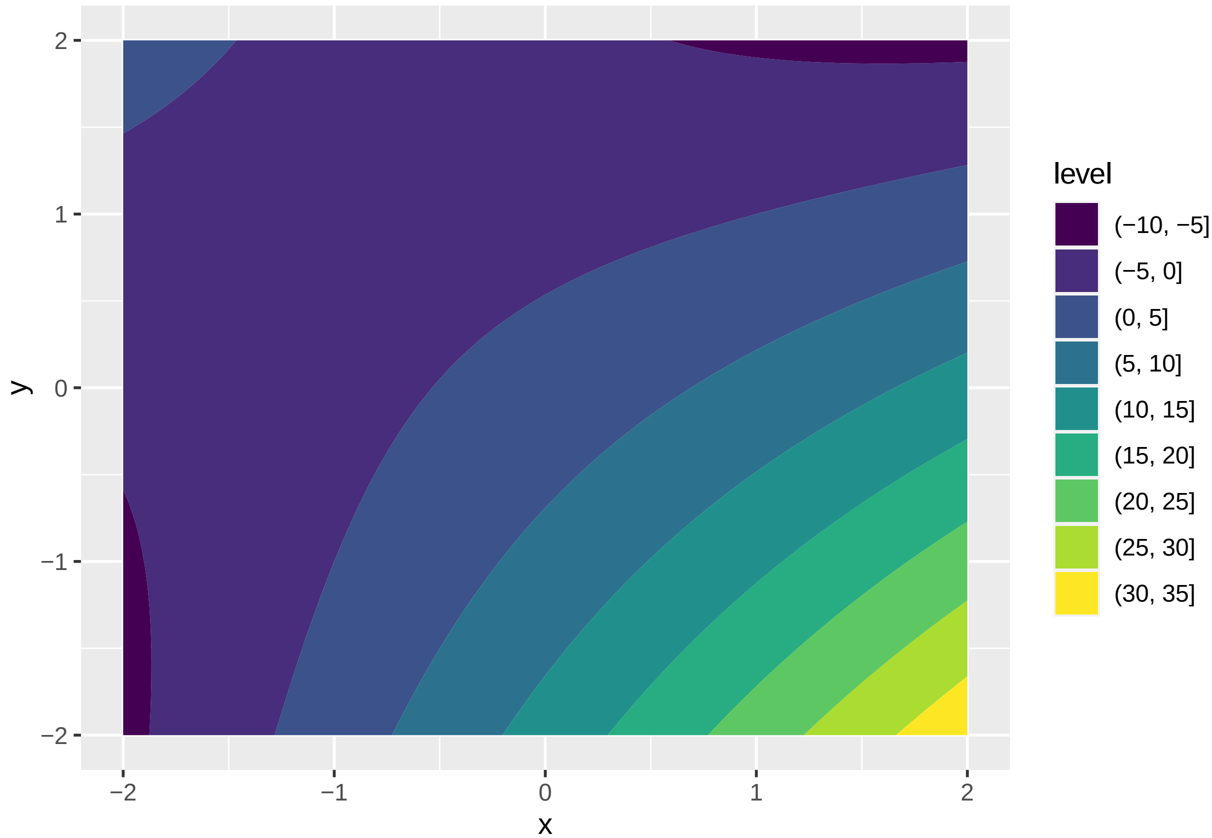
Here we have a global view of the contour plot on the bounds $x = [-100, 100]$ for our function with color scale.

```r
# BOUNDS USED FOR CONTOUR
pointsX = seq(-100, 100, length=200)# create a 200x1 vector of values for x axis
pointsY = seq(-100, 100, length=200) # create a 200x1 vector of values for y axis
myGrid = expand.grid(pointsX, pointsY) # create 200x200 grid of points
z = map2_dbl(myGrid$Var1, myGrid$Var2, ~fun(.x, .y)) # maps all possible combinations
# of the grid through the function
x = myGrid$Var1
y =  myGrid$Var2
myPlot = plot_ly(x=~x, y=~y, z=~z, type="mesh3d", intensity=~z,
        colors = colorRamp(c("black","red","yellow","chartreuse3")))
myGrid$z = z
#myPlot
v = ggplot( myGrid,aes(x, y, z=z)) +geom_contour_filled()
v
```



Here we have another contour plot of our function but this time on smaller bounds $x = [-2, 2]$ with color scale.

```
# BOUNDS USED FOR CONTOUR
pointsX = seq(-2, 2, length=200)# create a 200x1 vector of values for x axis
pointsY = seq(-2, 2, length=200) # create a 200x1 vector of values for y axis
myGrid = expand.grid(pointsX, pointsY) # create 200x200 grid of points
z = map2_dbl(myGrid$Var1, myGrid$Var2, ~fun(.x, .y)) # maps all possible combinations
# of the grid through the function
x = myGrid$Var1
y =  myGrid$Var2
myPlot = plot_ly(x=~x, y=~y, z=~z, type="mesh3d", intensity=~z,
        colors = colorRamp(c("black","red","yellow","chartreuse3")))
myGrid$z = z
#myPlot
v = ggplot( myGrid,aes(x, y, z=z)) +geom_contour_filled()
v
```



3

We want to find the function derivitve of $F(x)$ at the point $x^* = [0,0]^T$ in the direction of $p = [1,1]^T$.

The equation for a directional derivative is given by Eq. (8.12), $\frac{p^T \nabla F(X)}{||p||}$.

Thus, the derivative in the direction of P is computed to be:

$$\frac{\begin{bmatrix} 1 \\ -1 \end{bmatrix}^T (\begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 4 \\ -4 \end{bmatrix})}{|| \begin{bmatrix} 1 \\ -1 \end{bmatrix} ||} = \frac{8}{\sqrt{2}} = 4\sqrt{2}$$

In conclusion, the function has slope $4\sqrt{2}$ in the direction of p from the point x^*. The slope in this direction will only be zero if $p^T \nabla F(X) = 0$, which will only happen if the directional derivative p is orthogonal to the gradient at x^* (tangent to the contour).

Here we have another zoomed in contour plot of our function on the bounds $x = [-0.5, 1.5]$, along with the direction vector p from point x^*, which has slope $4\sqrt{2}$.

```r
# BOUNDS USED FOR CONTOUR
pointsX = seq(-.5, 1.5, length=200)# create a 200x1 vector of values for x axis
pointsY = seq(-.5, 1.5, length=200) # create a 200x1 vector of values for y axis
myGrid = expand.grid(pointsX, pointsY) # create 200x200 grid of points
z = map2_dbl(myGrid$Var1, myGrid$Var2, ~fun(.x, .y)) # maps all possible combinations
```

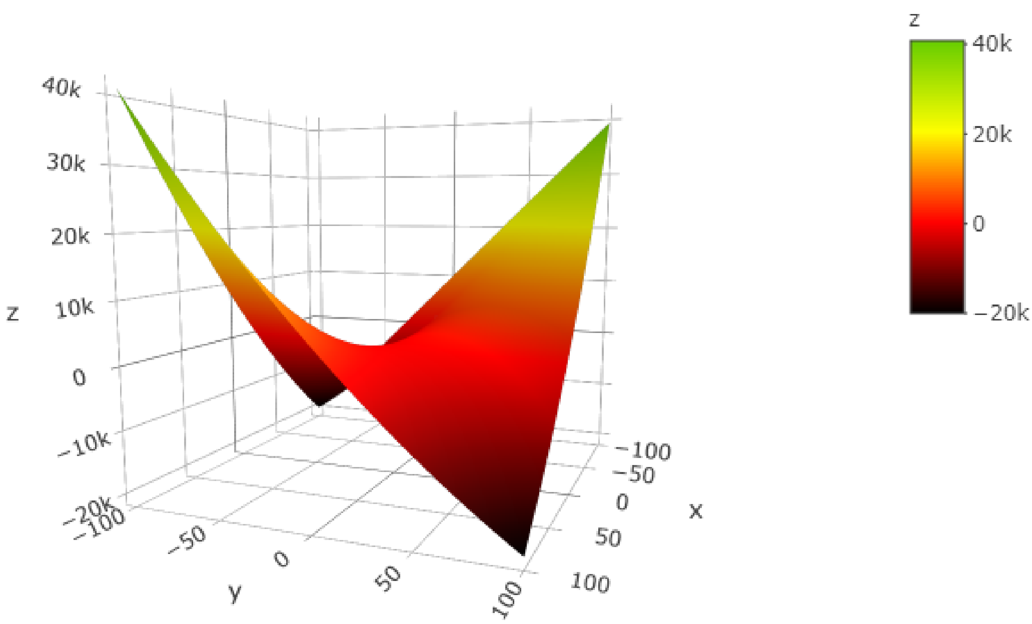


Figure 1: Original Function

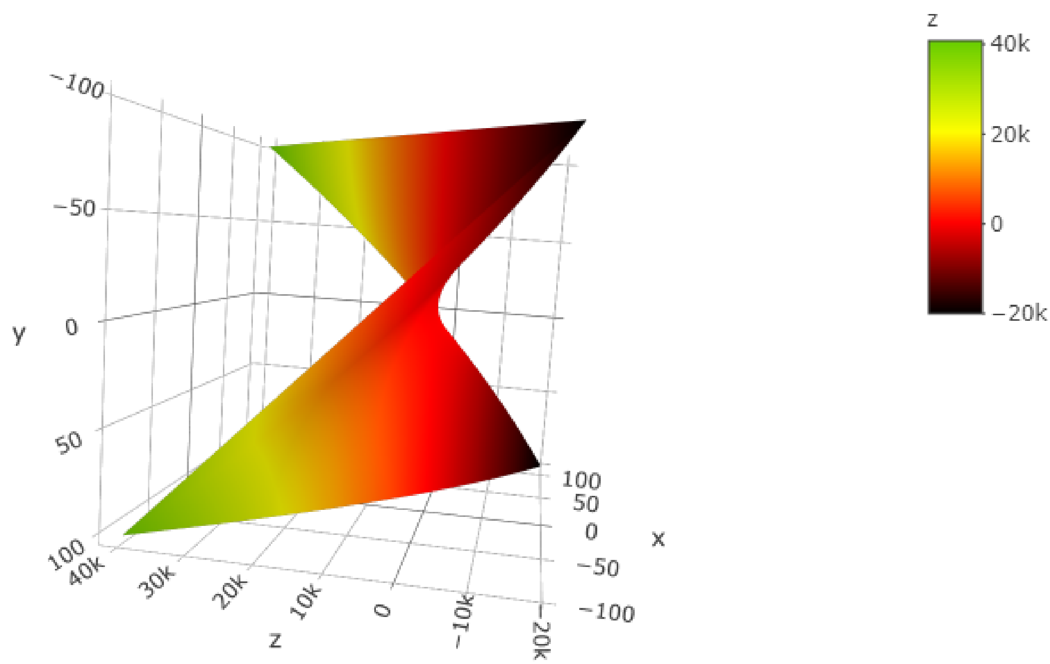


Figure 2: Original Function with inverted view

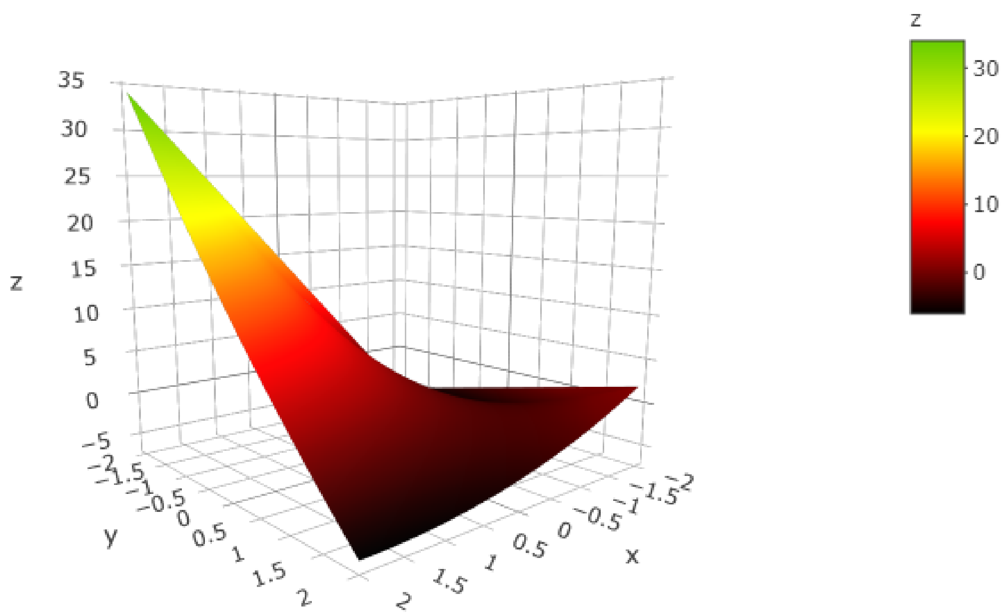


Figure 3: Original Function with zoomed in domain

```r
# of the grid through the function
x = myGrid$Var1
y =  myGrid$Var2
myPlot = plot_ly(x=~x, y=~y, z=~z, type="mesh3d", intensity=~z,
        colors = colorRamp(c("black","red","yellow","chartreuse3")))
myGrid$z = z
#myPlot
v = ggplot( myGrid,aes(x, y, z=z)) +geom_contour_filled()+
  geom_segment(aes(x=0,y=0, xend=1, yend=1, colour="segment"), arrow=arrow(length=unit(0.25, "cm")))+
  geom_point(aes(x=0,y=0), color="red")
v
```