Chapter 8: Performance Surfaces and Optimum Points

Brandon Morgan

1/15/2021

E8.10

We are given the following vector function in simplified terms

$$F(x) = \frac{3}{2}x_1^2 + 2x_1x_2 + x_2^3 + 4x_1 + 4x_2$$

1

The quadtratic approximation is simply the second order Taylor series approximation for vectors:

$$F(x) = F(x^*) + \nabla F(x)|_{x=x^*}(x - x^*) + \frac{1}{2}\nabla^2 F(x)|_{x=x^*}(x - x^*)^2$$

Where $\nabla F(x)$ is known as the gradient, by $[\frac{\partial}{\partial x_1}F(x)|\frac{\partial}{\partial x_2}F(x)|\dots|\frac{\partial}{\partial x_n}F(x)]$, and where $\nabla^2 F(x)$ is known as the Hessian given by equation 8.11 in book.

The gradient of our function is calculated to be:

$$\nabla F(x) = [\ 3x_1 + 2x_2 + 4 \mid 2x_1 + 3x_2^2 + 4\ ]$$

A Hessian matrix is given by:

$$\nabla^2 F(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(x) & \frac{\partial^2}{\partial x_1 \partial x_2}F(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(x) & \frac{\partial^2}{\partial x_2^2}F(x) \end{bmatrix}$$

Where it can be shown from our example that

$$\frac{\partial^2}{\partial x_1^2}F(x) = 3$$

$$\frac{\partial^2}{\partial x_2^2}F(x) = 6x_2$$

$$\frac{\partial^2}{\partial x_1 \partial x_2}F(x) = 2$$

$$\frac{\partial^2}{\partial x_2 \partial x_1}F(x) = 2$$

which results in our Hessian matrix being:

$$\begin{bmatrix} 3 & 2 \\ 2 & 6x_2 \end{bmatrix}$$

From our working above we can find our second order Taylor Approximation about the point $x^* = [1, 0]$:

We get the following working:

Vector Function

$$F([1,0]) = \frac{11}{2}$$

Gradient

$$\frac{\partial}{\partial x_1} F([1,0]) = 7$$
$$\frac{\partial}{\partial x_2} F([1,0]) = 6$$

Hessian

$$\frac{\partial^2}{\partial x_1^2} F([1,0]) = 3$$

$$\frac{\partial^2}{\partial x_2^2} F([1,0]) = 0$$
$$\frac{\partial^2}{\partial x_1 \partial x_2} F([1,0]) = 2$$
$$\frac{\partial^2}{\partial x_2 \partial x_1} F([1,0]) = 2$$

Combined

$$F(x) = \frac{11}{2} + \begin{bmatrix} 7 & 6 \end{bmatrix} (x - \begin{bmatrix} 1 \\ 0 \end{bmatrix}) + \frac{1}{2}(x - \begin{bmatrix} 1 \\ 0 \end{bmatrix})^T \begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix} (x - \begin{bmatrix} 1 \\ 0 \end{bmatrix})$$

Which reduces down to:

$$F(x) = 1.5x_1^2 + 2x_1x_2 + 4x_1 + 4x_2$$

2

The stationary point of a function is found by any set of points that satisfy Eq. (8.27), $\nabla F(x)|_{x=x^*} = 0$. Thus any set of points that makes the gradient equal to zero is considered stationary.

For our second order taylor series approximation, we need to compute and find the gradient. It's gradient can be found by finding its partial derivates, as described earlier; however, this time we have our function in matrix form. The gradient of our vector function is the following:

$$\nabla F(X) = \begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}3 & 2\\2 & 0\end{bmatrix}(X - \begin{bmatrix}1\\0\end{bmatrix}) + \frac{1}{2}\begin{bmatrix}3 & 2\\2 & 0\end{bmatrix}(X - \begin{bmatrix}1\\0\end{bmatrix})^T$$

but because $\begin{bmatrix}3 & 2\\2 & 0\end{bmatrix} = \begin{bmatrix}3 & 2\\2 & 0\end{bmatrix}^T$:

$$\nabla F(X) = \begin{bmatrix}7\\6\end{bmatrix} + \begin{bmatrix}3 & 2\\2 & 0\end{bmatrix}(X - \begin{bmatrix}1\\0\end{bmatrix})$$

which reduces down to:

$$\nabla F(X) = \begin{bmatrix}3x_1 + 2x_2 + 4\\2x_1 + 4\end{bmatrix}$$

The reduced form was given earlier as well, one could have taken the partial derivatives with respect to each value of X and computed the same answer; however, matrix differentiation is a little bit more fun!

Solving for when these two equations are set to zero will result in $x = [-2, 1]$. The process can be done algebraically because the second entry only contains x_1, thus $2x_1 + 4 = 0$ which only occurs at $x_1 = -2$; then, one can solve for x_2 by plugging x_1 back into the top equation.

3

A minimum is defined as a stationary point, a point that satisfies $\nabla F(x) = 0$, which has the smallest value when inputted back into $F(x)$.

Our gradient for the original function was calculated to be:

$$\nabla F(x) = [\ 3x_1 + 2x_2 + 4 \mid 2x_1 + 3x_2^2 + 4\ ]$$

setting the equations equal to each other and solving:

$$3x_1 + 2x_2 + 4 = 0 = 2x_1 + 3x_2^2 + 4$$

$$3x_1 + 2x_2 = 2x_1 + 3x_2^2$$

$$x_1 + 2x_2 = 3x_2^2$$

$$x_1 = 3x_2^2 - 2x_2$$
$$x_1 = x_2(3x_2 - 2)$$

Thus, we get a hyper plane of possible solutions for the stationary points:

$$\begin{bmatrix}x_2(3x_2 - 2)\\x_2\end{bmatrix}$$

Thus, plugging in the stationary point of our quadtratic Taylor series approximation about the point $x^* = [1, 0]^T$ will yield:

$$\begin{bmatrix} 1(3-2) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \neq \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

Thus our Taylor series approximation stationary point is not a stationary point for our function as it does not lie in the hyper plane of stationary points. Therefore, we can rule out the possibility of it being a minimum point for our function.

Original Function Plots

Here we have the plots of our function in 3 dimension with two different domains:

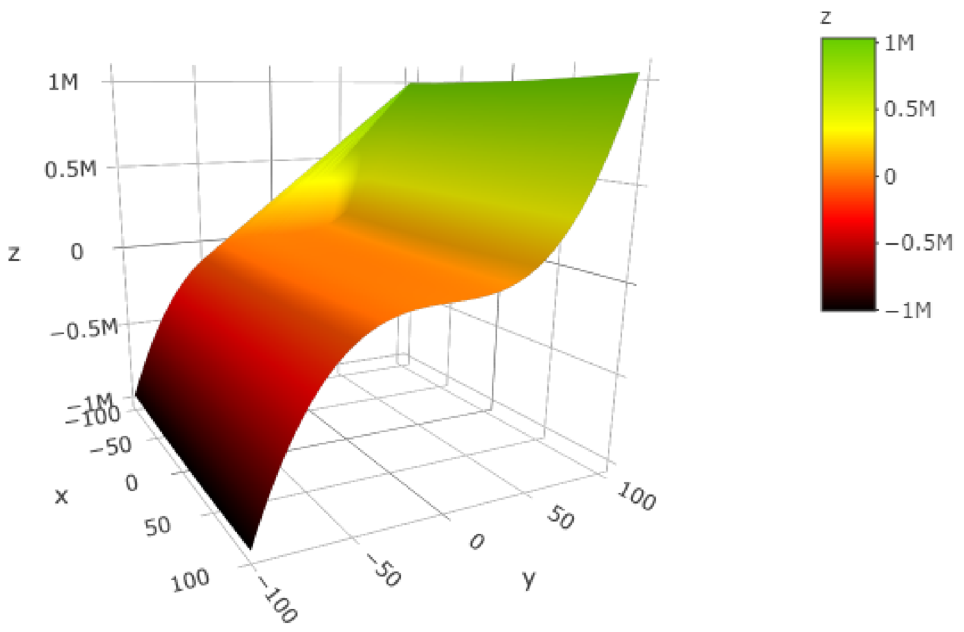


Figure 1: Original Function

Code for plots and Contour

```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
```

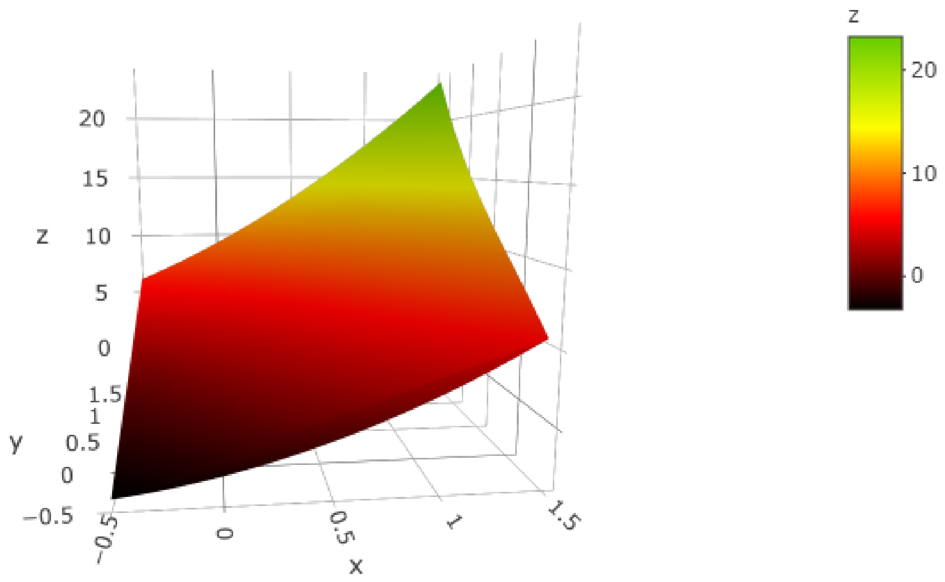


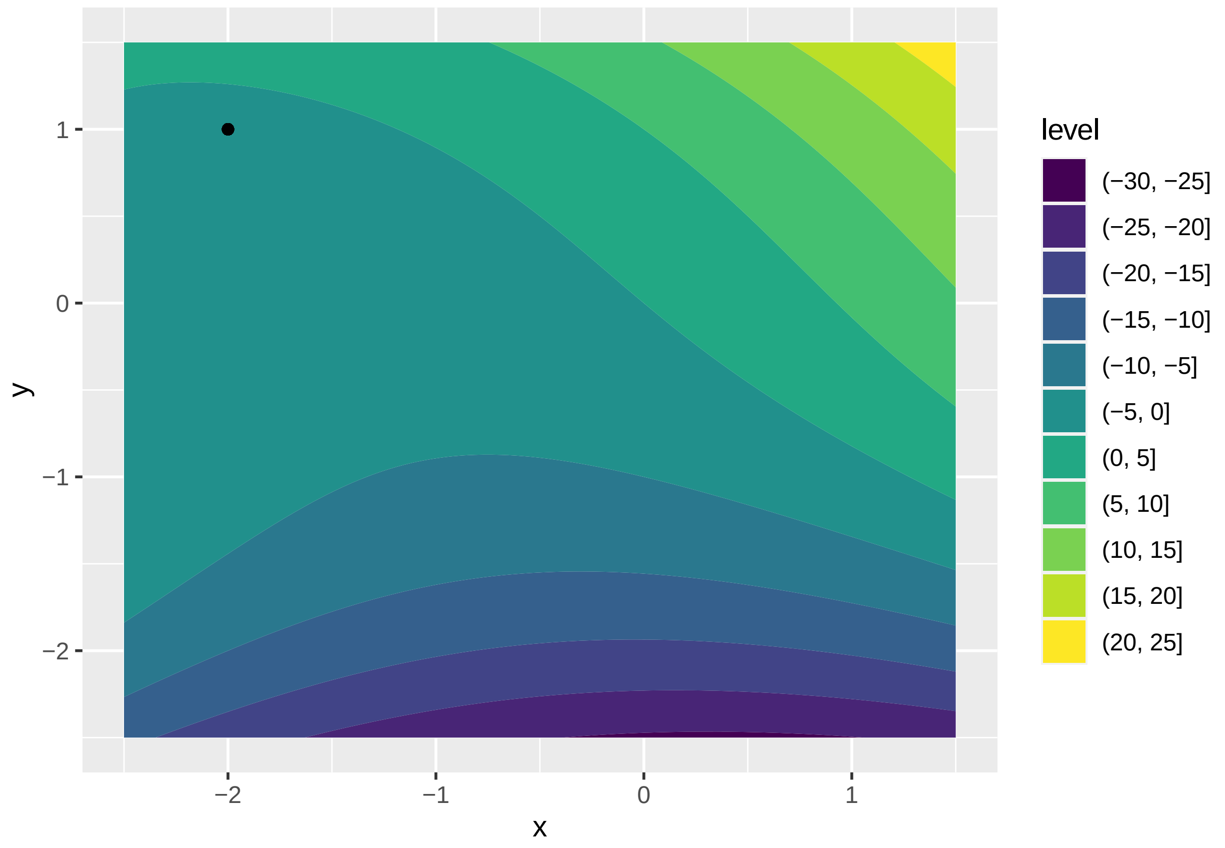
Figure 2: Original Function with zoomed in domain

```
##
## 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
```

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

```
fun = function(x, y) {
  1.5*x^2+2*x*y+y^3+4*x+4*y
}
```

```
# BOUNDS USED FOR CONTOUR
pointsX = seq(-2.5, 1.5, length=200)# create a 200x1 vector of values for x axis
pointsY = seq(-2.5, 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
# 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 + geom_point(aes(x=-2, y=1))
```



Here we have a zoomed in view of the contour plot on the bounds $x = [-2, 1.5]$ for our function with color scale, as well as our stationary point for our Taylor Series approximation.

Taylor Series Expansion

```
taylor = function(x, y) {
  1.5*x^2+2*x*y+4*x+4*y
}
```

```
# BOUNDS USED FOR CONTOUR
pointsX = seq(-2.5, -1.5, length=200)# create a 200x1 vector of values for x axis
pointsY = seq(0.5, 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, ~taylor(.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 = add_trace(p=myPlot, z = c(taylor(-2, 1)), x=c(-2), y=c(1), type="scatter3d")
#myPlot
v = ggplot( myGrid,aes(x, y, z=z)) +geom_contour_filled()
v + geom_point(aes(x=-2, y=1))
```

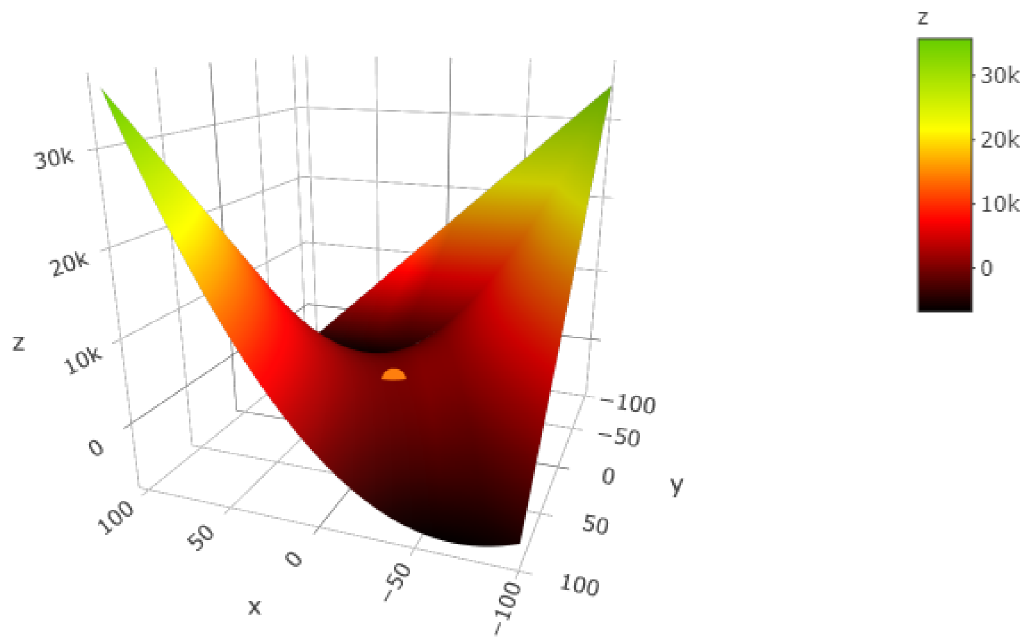


Figure 3: Quadtratic Approximation

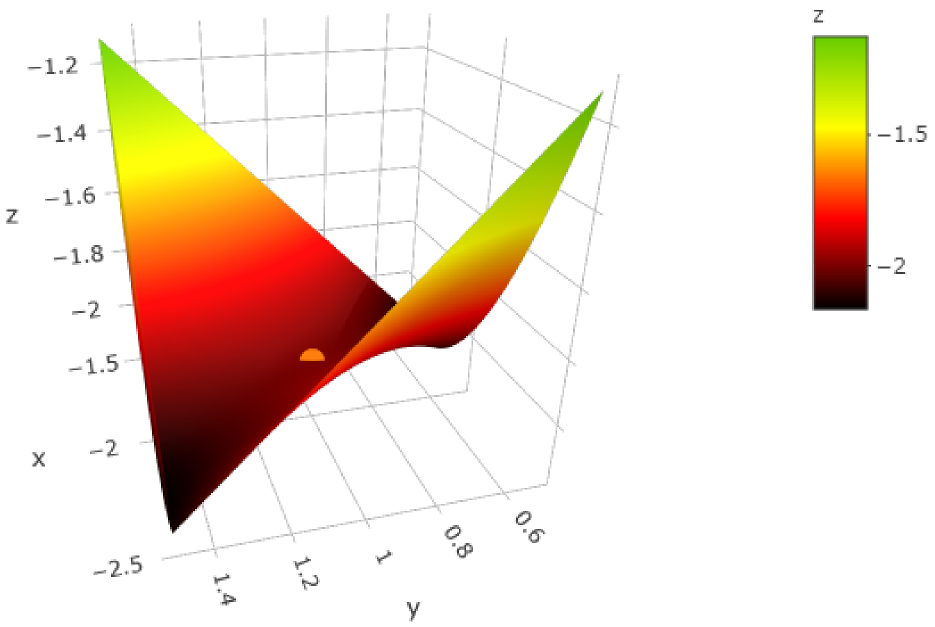
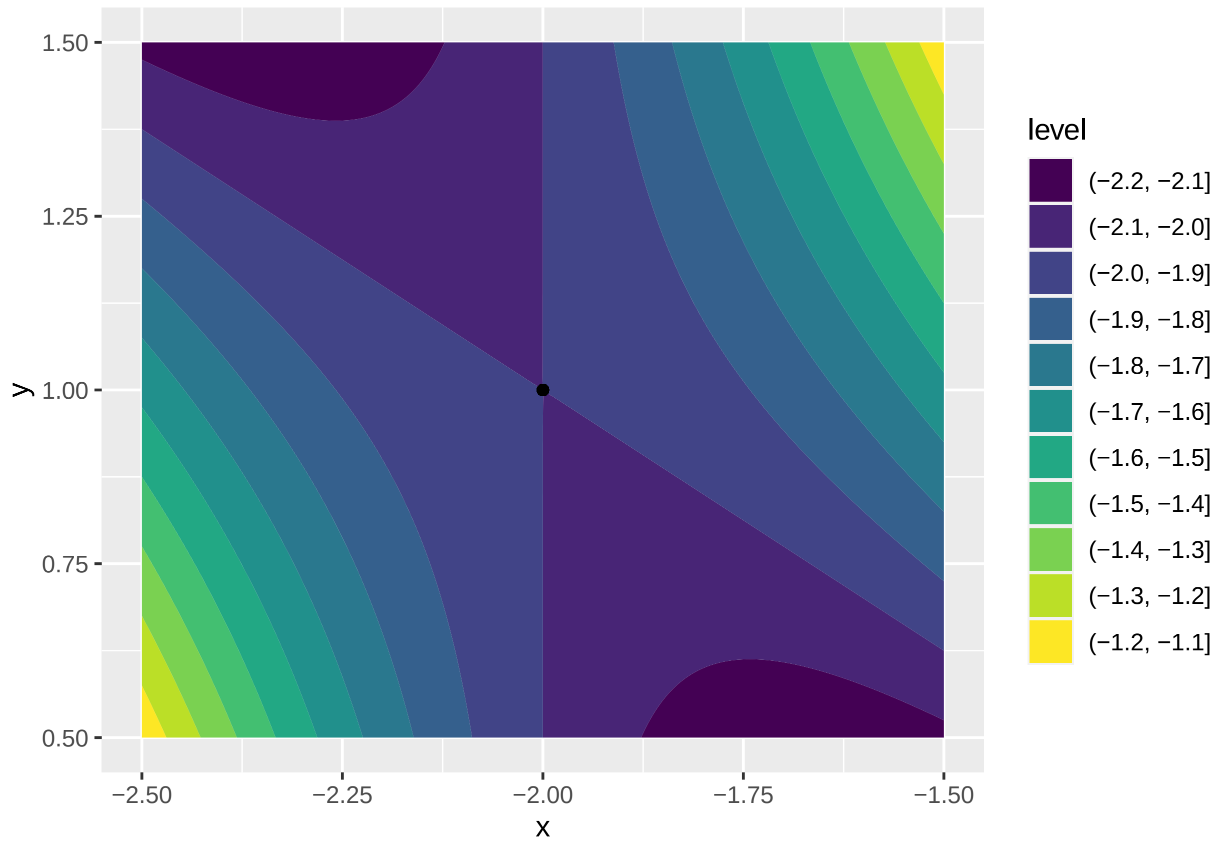


Figure 4: Quadtratic Approximation with zoomed in domain



Here we have a zoomed in view of the contour plot on the bounds $x = [-2.5, 1.5]$ for our function with color scale, as well as our stationary point for our Taylor Series approximation.