Homework 1 (part 1)

Michael Pena

2024-01-24

Problem 1

#i.

```
# build f(x1,x2) function
f <- function(x1,x2){
  cos(x1*x2)
}
```

#ii.

```
# build h(x1,x2) function
h <- function(x1,x2){
  1 - pi^2*(x1^2)/8
}
```

#iii.

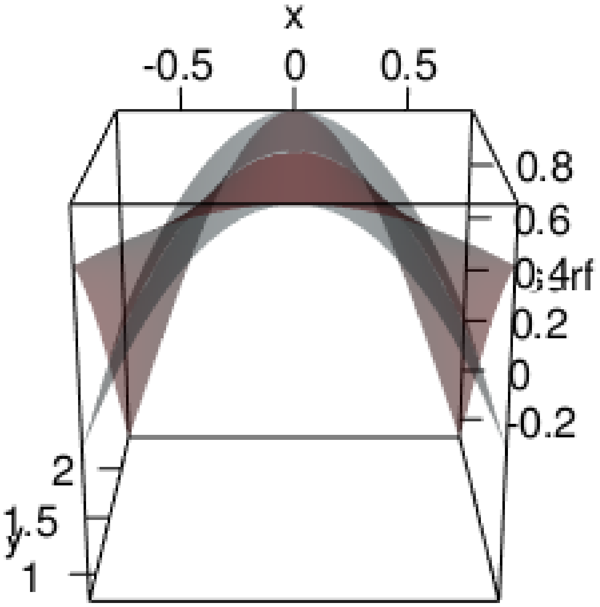
```
# build the sequences
x <- seq(-pi/4,pi/4,length = 30)
y <- seq(pi/4,3*pi/4,length = 30)
#grid <- expand.grid(x = x, y = y)

#fxy <- f(grid$x,grid$y)
#hxy <- h(grid$x,grid$y)
```

```
# plot the functions
fsurf <- outer(x,y,FUN = f)
hsurf <- outer(x,y,FUN = h)
persp3d(x, y, fsurf, col = "red",shade = 0.3, alpha = 0.5)
persp3d(x, y, hsurf, col = "lightblue", shade = 0.3, alpha = 0.5, add=T)
rglwidget(controllers = )
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead
```

```
## cleared error 1285
```



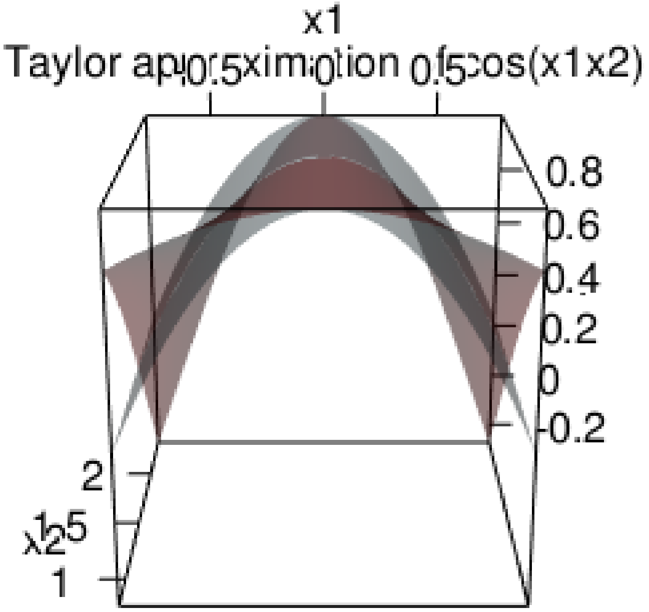
#iv.

now we need to add subscripts

```
#adding labels
persp3d(x, y, fsurf, col = "red",shade = 0.3, alpha = 0.5,
        main = "Taylor approximation of cos(x1x2)",
        xlab = "x1",
        ylab = "x2",
        zlab = "f")
persp3d(x, y, hsurf, col = "lightblue", shade = 0.3, alpha = 0.5, add=T)
rglwidget(controllers = )
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead
```

```
## cleared error 1285
```



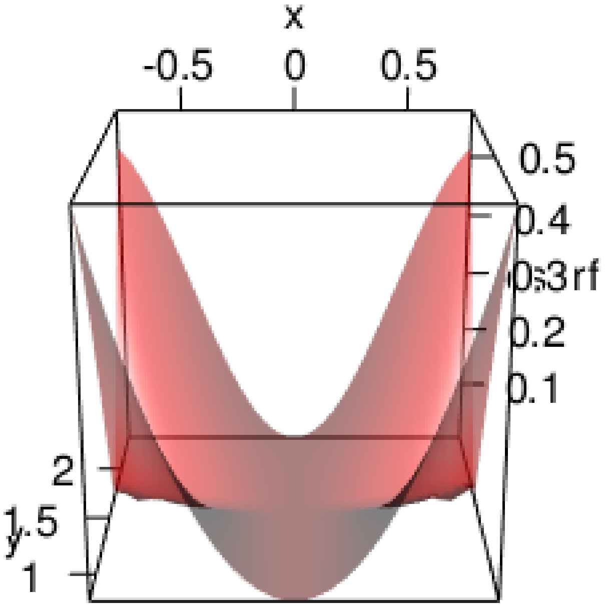
#v.

```
#build error function
e <- function(x1,x2) {
  abs(f(x1,x2) - h(x1,x2))
}
```

```
# render the plot of the function
esurf <- outer(x,y,FUN = e)
persp3d(x, y, esurf, col = "red",shade = 0.3, alpha = 0.5)
rglwidget(controllers = )
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead
```

```
## cleared error 1285
```

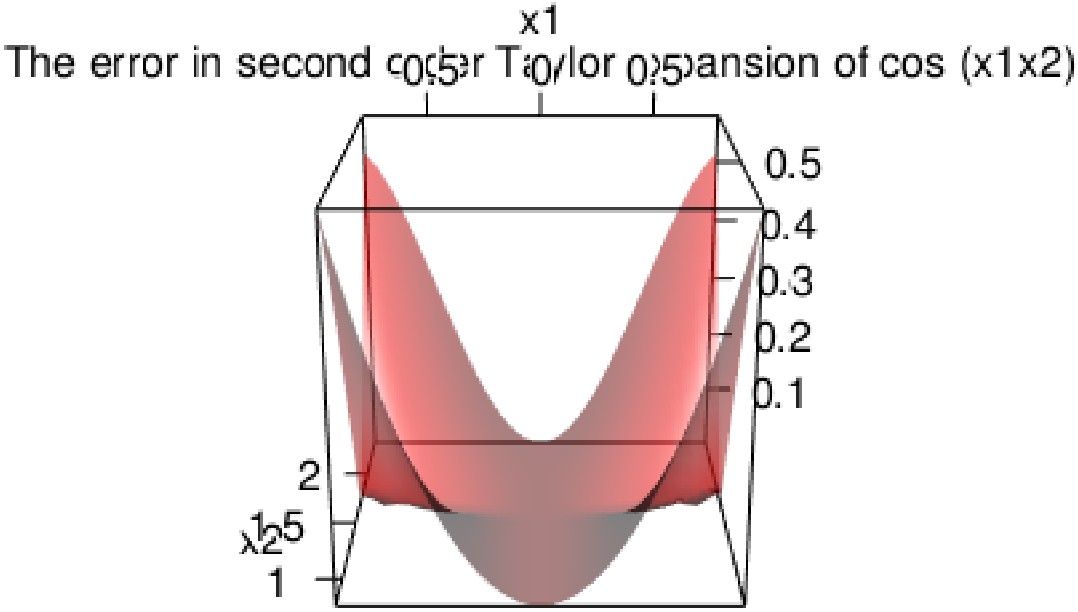


vi.

```
# add labelings
persp3d(x, y, esurf, col = "red",shade = 0.3, alpha = 0.5,
        main = "The error in second order Taylor expansion of cos (x1x2)",
        xlab = "x1",
        ylab = "x2",
        zlab = "e")
rglwidget(controllers = )
```

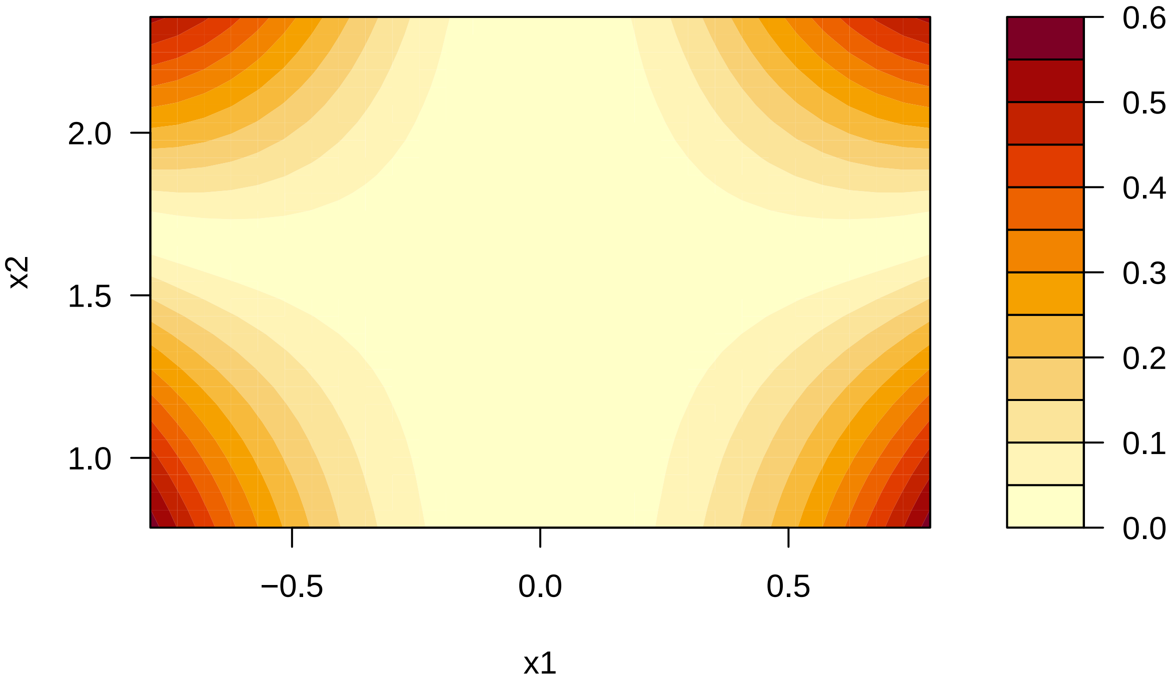
```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead

## cleared error 1285
```



vii.

```
# rendering contour for error function
filled.contour(x, y, esurf,
               xlab = "x1",
               ylab = "x2")
```



$h(x_1, x_2)$ estimates $f(x_1, x_2)$ closely where $x_1 = 0$. as we fall farther away from this region, h and f they diverge from eachother. This is clear by how the contour are gathered into the corners of the graphic. There is also a low contour in the around the line $x_2 = 1.7$; this is because the two surfaces cross each other at this curve. It would perhaps be best to use this Taylor approximation only around points that are around the line x_1.

Problem 2

(a).

$$[\mathbf{x}-\boldsymbol{\mu}]^T\boldsymbol{\Sigma}^{-1}[\mathbf{x}-\boldsymbol{\mu}] = \sigma_{22}(x_1-\mu_1)^2 - (\sigma_{12}+\sigma_{21})(x_1-\mu_1)(x_2-\mu_2) + \sigma_{11}(x_2-\mu_2)^2$$

$$f(\mathbf{x}_0) = f\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} = \frac{1}{2\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}}$$

$$(\mathbf{x}-\mathbf{x}_0)^T = \begin{pmatrix}x_1-\mu_1\\ x_2-\mu_2\end{pmatrix}$$

$$f_{x_1} = \frac{-e^{\frac{-1}{2}(\sigma_{22}(x_1-\mu_1)^2-(\sigma_{12}+\sigma_{21})(x_1-\mu_1)(x_2-\mu_2)+\sigma_{11}(x_2-\mu_2)^2)/(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})}}{4\pi(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})^{3/2}}\cdot(2\sigma_{22}(x_1-\mu_1)-(\sigma_{12}+\sigma_{21})(x_2-\mu_2))$$

$$f_{x_2} = \frac{-e^{\frac{-1}{2}(\sigma_{22}(x_1-\mu_1)^2-(\sigma_{12}+\sigma_{21})(x_1-\mu_1)(x_2-\mu_2)+\sigma_{11}(x_2-\mu_2)^2)/(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})}}{4\pi(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})^{3/2}}\cdot(2\sigma_{11}(x_2-\mu_2)-(\sigma_{12}+\sigma_{22})(x_1-\mu_1))$$

$$f_{x_1x_1} = \frac{e^{\frac{-1}{2}(\sigma_{22}(x_1-\mu_1)^2-(\sigma_{12}+\sigma_{22})(x_1-\mu_1)(x_2-\mu_2)+\sigma_{11}(x_2-\mu_2)^2)}}{4\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}}\cdot\left(\frac{1}{2}(2\sigma_{22}(x_1-\mu_1)-(\sigma_{12}+\sigma_{21})(x_2-\mu_2))^2-2\sigma_{22}\right)$$

$$f_{x_1x_2} = f_{x_2x_1} = \frac{e^{\frac{-1}{2}(\sigma_{22}(x_1-\mu_1)^2-(\sigma_{12}+\sigma_{22})(x_1-\mu_1)(x_2-\mu_2)+\sigma_{11}(x_2-\mu_2)^2)}}{4\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}} \cdot \left(\frac{1}{2}(2\sigma_{22}(x_1-\mu_1)-(\sigma_{12}+\sigma_{21})(x_2-\mu_2))((2\sigma_{11}(x_2-\mu_2)-(\sigma$$

$$f_{x_2x_2} = \frac{e^{\frac{-1}{2}(\sigma_{22}(x_1-\mu_1)^2-(\sigma_{12}+\sigma_{22})(x_1-\mu_1)(x_2-\mu_2)+\sigma_{11}(x_2-\mu_2)^2)}}{4\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}} \cdot \left(\frac{1}{2}(2\sigma_{11}(x_2-\mu_2)-(\sigma_{12}+\sigma_{21})(x_1-\mu_1))^2-2\sigma_{11}\right)$$

$$\nabla f \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} = \frac{-e^0}{4\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}} \begin{bmatrix}2\sigma_{22}(0)-(\sigma_{12}+\sigma_{22})(0)\\ 2\sigma_{11}(0)-(\sigma_{12}+\sigma_{22})(0)\end{bmatrix} = \frac{-1}{4\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}} \begin{bmatrix}0\\ 0\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix}$$

if

$$H \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} = \frac{-1}{4\pi(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})^{3/2}} \begin{bmatrix}2\sigma_{22} & -(\sigma_{12}+\sigma_{21})\\ -(\sigma_{12}+\sigma_{21}) & 2\sigma_{11}\end{bmatrix}$$

our second order Taylor approximation about point $\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}$

$$f(\mathbf{x}) \cong \frac{1}{2\pi\sqrt{\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21}}} - \frac{1}{8\pi(\sigma_{11}\sigma_{22}-\sigma_{12}\sigma_{21})^{3/2}}\left(2\sigma_{22}(x_1-\mu_1)^2-2(\sigma_{12}+\sigma_{21})(x_1-\mu_1)(x_2-\mu_2)+2\sigma_{11}(x_2-\mu_2)^2\right)+R$$

(b).

```
# building the function
f <- function(x1,x2,mu1,mu2,sig11,sig12,sig21,sig22){
  ((2*pi*sqrt(sig11*sig22 - sig21*sig12))^(-1))*exp((-0.5)*(2*sig22*(x1-mu1)^2 - (sig11*sig22 + sig12*s
}
```

```
#building the Taylor Series function
h <- function(x1,x2,mu1,mu2,sig11,sig12,sig21,sig22){
  ((2*pi*sqrt(sig11*sig22 - sig21*sig12))^(-1)) - ((8*pi*(sig11*sig22 - sig21*sig12)^(3/2))^(-1))*(2*si
}
```

```
# creating sequences
x <- seq(-1.5,1.5,length = 30)
y <- seq(-1.5,1.5,length = 30)
```

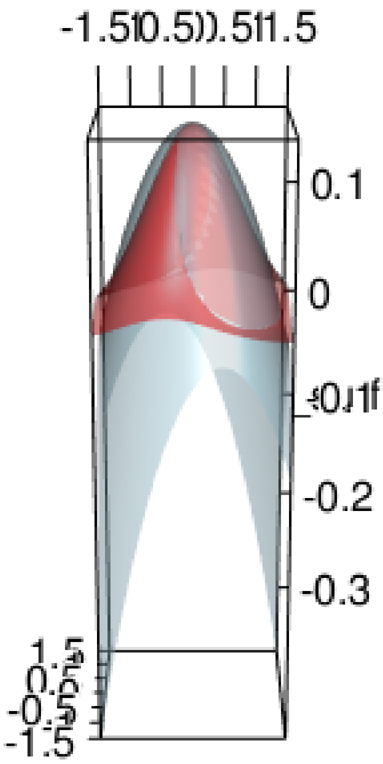
```
# render the surfaces
f_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = -0.3, sig21 = -0.3, sig22 = 1,FUN = f)
h_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = -0.3, sig21 = -0.3, sig22 = 1,FUN = h)
persp3d(x, y, f_surf, col = "red",shade = 0.7, alpha = 0.5)
persp3d(x, y, h_surf, col = "lightblue",shade = 0.7, alpha = 0.5, add = T)

rglwidget(controllers = )
```

(i.)

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead
```

```
## cleared error 1285
```

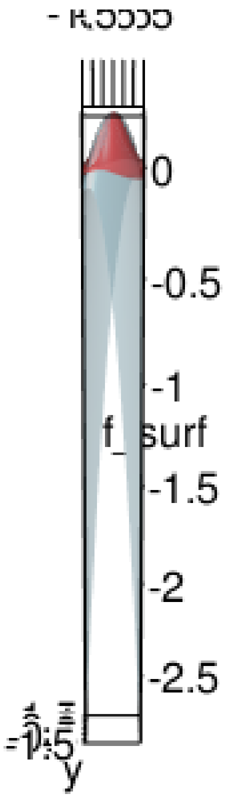


```
# render the surfaces
f_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = .8, sig21 = .8, sig22 = 1,FUN = f)
h_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = .8, sig21 = .8, sig22 = 1,FUN = h)
persp3d(x, y, f_surf, col = "red",shade = 0.7, alpha = 0.5)
persp3d(x, y, h_surf, col = "lightblue", shade = 0.7, alpha = 0.5, add=T)
rglwidget(controllers = )
```

(ii.)

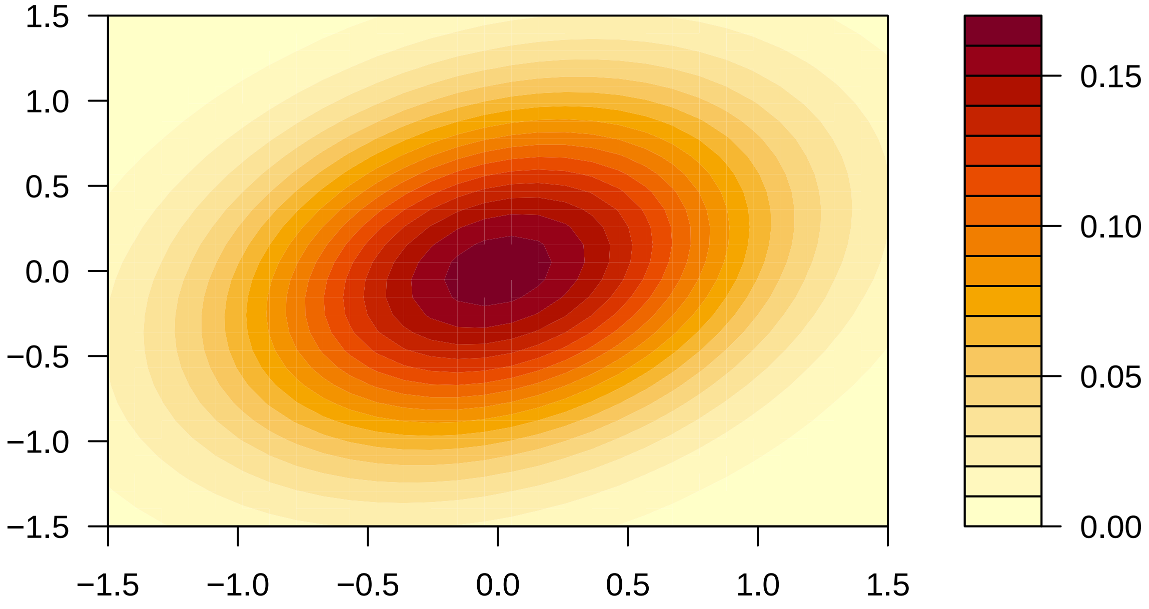
```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package; using rgl.snapshot() instead
```

```
## cleared error 1285
```

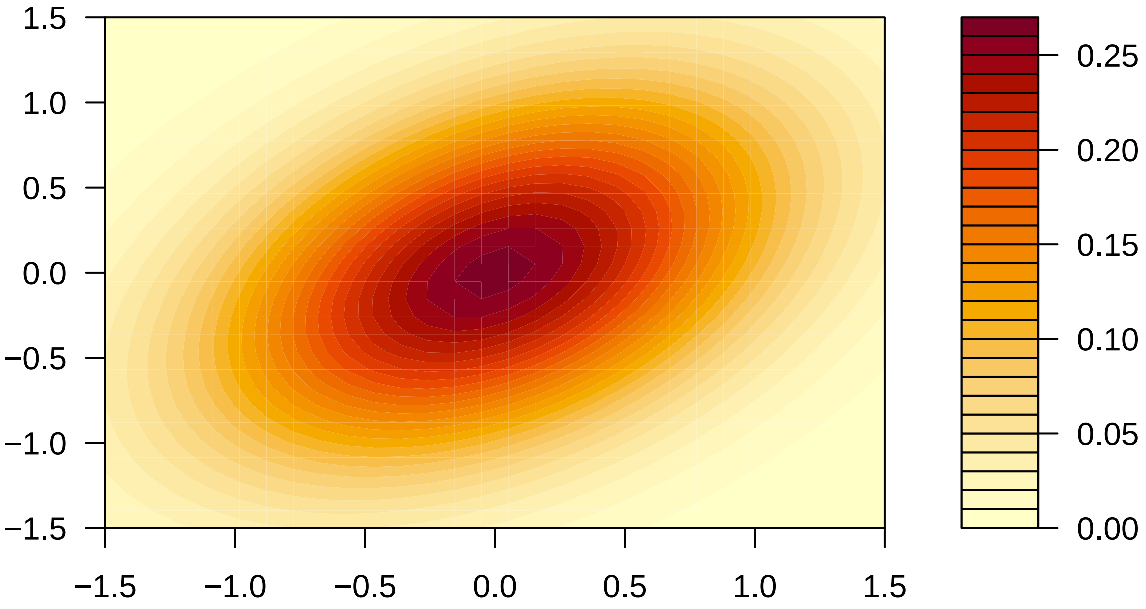


(c).

```
# case (i.)
# rendering contour for error function
f_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = -0.3, sig21 = -0.3, sig22 = 1,FUN = f)
filled.contour(x, y, f_surf)
```



```
# case (ii.)
# rendering contour for error function
f_surf <- outer(x, y, mu1 = 0, mu2 = 0 ,sig11 = 1, sig12 = .8, sig21 = .8, sig22 = 1,FUN = f)
filled.contour(x, y, f_surf)
```



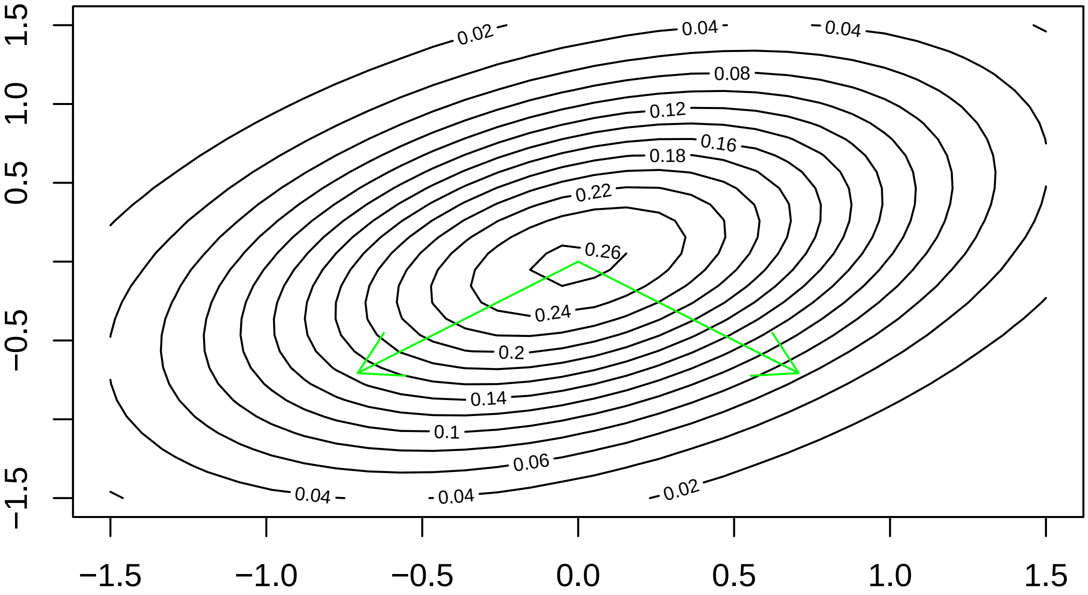
The shape of this contour is an ellipses; the center is about the origin of the x_1-x_2 plane. As we approach the center, we climb higher in our z-axis.

(d).

```
# case i. eigenvectors
S1 <- matrix(c(1,-0.3,-0.3,1), nrow=2, ncol = 2)
# compute e-vectors and e-values
eigen(S1)
```

```
## eigen() decomposition
## $values
## [1] 1.3 0.7
##
## $vectors
##            [,1]       [,2]
## [1,] -0.7071068 -0.7071068
## [2,]  0.7071068 -0.7071068
```

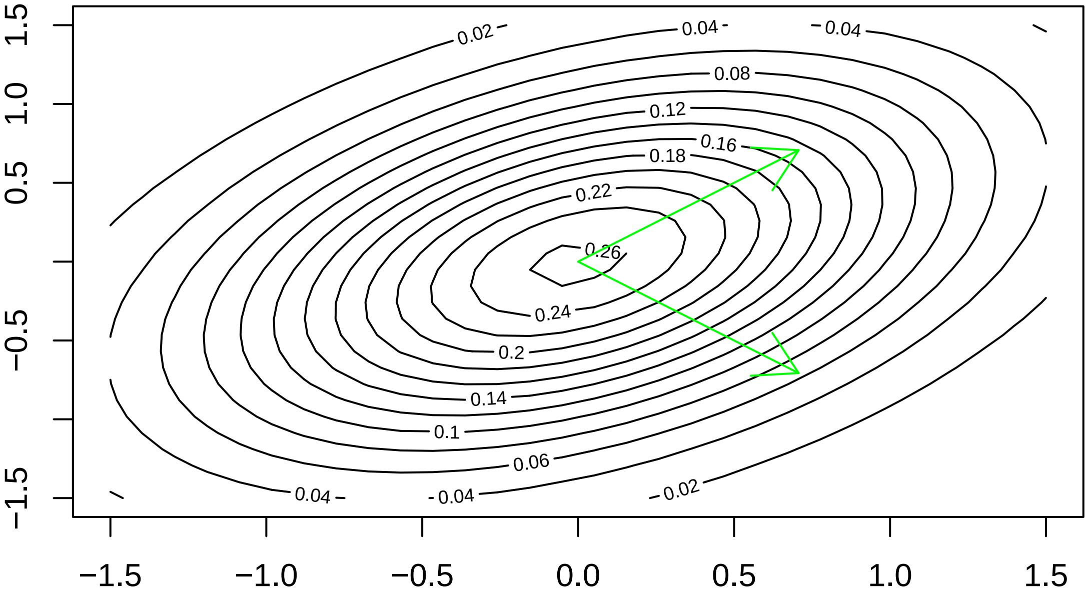
```
# superimpose the eigenvectors and eigenvalues over the contour
contour(x, y, f_surf)
arrows(0, 0, eigen(S1)$vectors[1, 1], eigen(S1)$vectors[2, 2], col = "green")
arrows(0, 0, eigen(S1)$vectors[2, 1], eigen(S1)$vectors[1, 2], col = "green")
```



```
# case ii. eigenvectors
S2 <- matrix(c(1,0.8,0.8,1), nrow=2, ncol = 2)
# compute e-vectors and e-values
eigen(S2)
```

```
## eigen() decomposition
## $values
## [1] 1.8 0.2
##
## $vectors
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

```
# superimpose the eigenvectors and eigenvalues over the contour
contour(x, y, f_surf)
arrows(0, 0, eigen(S2)$vectors[1, 1], eigen(S2)$vectors[2, 2], col = "green")
arrows(0, 0, eigen(S2)$vectors[2, 1], eigen(S2)$vectors[1, 2], col = "green")
```



eigenvectors and eigenvalues will typically correspond with the shape of the ellipses that are formed in the contour graphs.