Tara Ram Mohan
Homework 6
OPER 527

Question 1
(a) (10 points) Set up the problem using a penalty function. Clearly define your penalty functions.

```
# Given function: x^2 - 24*x + y^2 - 12*y + 200
f1 <- function(x, y){
  result1 <- x^2 - 24*x + y^2 - 12*y + 200
  return ( result1 )
}

# Function to give me the maximum or 0
max0 <- function( x ) {
  result1 <- max( x, 0 )
  return(result1)
}

# Function to give me the minimum  or 0
min0 <- function( x ) {
  result1 <- min( x, 0 )
  return(result1)
}

# Penalty function
# Constraint 1: 2*x + 4*y <= 12
# Constraint 2: x + 3*y <= 15
# Constraint 3: x >= 0
# Constraint 4: y >= 0
p1 <- function(x, y){
  c1 <- max0(-12 + 2*x + 4*y)
  c2 <- max0(-15 + x + 3*y)
  c3 <- min(x)
  c4 <- min(y)
  result1 <- mu1*(c1^2 + c2^2 + c3^2 + c4^2)
  return (result1)
}

# Function with penalty
p_theta1 <- function(x, y){
  result1 <- f1( x, y ) - p1( x, y )
  return (result1)
}
```

(b) (30 points) Use a gradient method to solve the penalty defined problem for μ = 1, 10, and 100. Do the solutions differ across μ?

```
mu1 <- 100
mu2 <- 10
mu3 <- 1

# D/dx
p_thetaprime1x <- function(x, y){
  result1 <- 2*x - 24
  c1 <- max0(-12 + 2*x + 4*y)
  c2 <- max0(-15 + x + 3*y)
  c3 <- min0(x)
  c4 <- min0(y)
  result1 <- result1 - mu1*(2*2*c1^4 + 1*2*c2^4 + 2*c3^5)
  return(result1)
}

# D/dy
p_thetaprime1y <- function(x, y){
  result1 <- 2*y - 12
  c1 <- max0(-12 + 2*x + 4*y)
  c2 <- max0(-15 + x + 3*y)
  c3 <- min0(x)
  c4 <- min0(y)
  result1 <- result1 - mu1*(4*2*c1^4 + 3*2*c2^4 + 2*c4^5)
  return (result1)
}

# Gradient function
p_g1 <- function(x, y){
  result1 <- c(p_thetaprime1x(x, y), p_thetaprime1y(x, y))
  return(result1)}

x0 <- c(0, 0) #starting value for steepest ascent
step1 <- 0.0001
count1 <- 1
error1 <-1

# Steepest ascent while loop
while(error1 > 0.001){
  x0hold <- x0
  x0 <- x0 + p_g1(x0[1], x0[2])*step1
  p_ghold <- p_g1(x0[1], x0[2])
```

```
  error1 <- t(p_ghold)%*%(p_ghold)
  count1 <- count1 + 1
}

#Penalty Solutions
# for mu3 = 1
p_sol1 <- c(-1.69, -1.5)
# for mu2 = 10
p_sol10 <- c(-1.055, -0.929)
# for mu1 = 100
p_sol100 <- c(-0.661, -0.58)
```

Yes, the solutions differ across μ.

(c) (10 points) Set up the problem using a barrier function. Clearly define your barrier functions.

```
mu1 <- 100
mu2 <- 10
mu3 <- 1

r = 1 # barrier parameter

# Barrier function
# Constraint 1: 2*x + 4*y <= 12
# Constraint 2: x + 3*y <= 15
# Constraint 3: x >= 0
# Constraint 4: y >= 0
b1 <- function(x, y){
  c1 <- r*ifelse(is.nan(log(-2*x - 4*y  + 12)),0,log(-2*x - 4*y  + 12))
  c2 <- r*ifelse(is.nan(log(-x - 3*y + 15)),0,log(-x - 3*y + 15))
  c3 <- r*ifelse(is.nan(log(x)),0,log(x))
  c4 <- r*ifelse(is.nan(log(y)),0,log(y))
  result1 <- mu1*(c1 + c2 + c3 + c4)
  return (result1)
}

# Function with penalty
b_theta1 <- function(x, y){
  result1 <- f1(x, y) + b1(x, y)
  return(result1)
}
```

(d) (30 points) Use a gradient method to solve the barrier defined problem for μ = 1, 10, and 100. Do the solutions differ across μ.

```
# D/dx
b_thetaprime1x <- function(x, y){
  result1 <- 2*x - 24
  c1 <- r*ifelse(is.nan(log(-2*x - 4*y  + 12)),0,log(-2*x - 4*y  + 12))
  c2 <- r*ifelse(is.nan(log(-x - 3*y + 15)),0,log(-x - 3*y + 15))
  c3 <- r*ifelse(is.nan(log(x)),0,log(x))
  c4 <- r*ifelse(is.nan(log(y)),0,log(y))
  result1 <- result1 + mu1*(-2*c1 - c2 + c3)
  return(result1)
}

# D/dy
b_thetaprime1y <- function(x, y){
  result1 <- 2*y - 12
  c1 <- r*ifelse(is.nan(log(-2*x - 4*y  + 12)),0,log(-2*x - 4*y  + 12))
  c2 <- r*ifelse(is.nan(log(-x - 3*y + 15)),0,log(-x - 3*y + 15))
  c3 <- r*ifelse(is.nan(log(x)),0,log(x))
  c4 <- r*ifelse(is.nan(log(y)),0,log(y))
  result1 <- result1 + mu1*(-4*c1 - 3*c2 + c4)
  return(result1)
}

# Gradient function
b_g1 <- function(x, y){
  result1 <- c(b_thetaprime1x(x, y), b_thetaprime1y(x, y))
  return(result1)
}

x0 <- c(0.5, 0.5)
step1 <- 0.0001
count1 <- 1
error1 <- 1

# Steepest descent while loop
while(error1 > 0.0001) {
  x0hold <- x0
  x0 <- x0 - step1*b_g1(x0[1], x0[2])
  b_ghold <- b_g1(x0[1], x0[2])

  error1 <- t(b_ghold)%*%(b_ghold)
  count1 <- count1 + 1
}

#Barrier Solutions
```

```
# for mu3 = 1
b_sol1 <- c(5.99502, 0.00198)
# for mu2 = 10
b_sol10 <- c(2.28, 1.81)
# for mu1 = 100
b_sol100 <- c(0.752, 2.549)
```

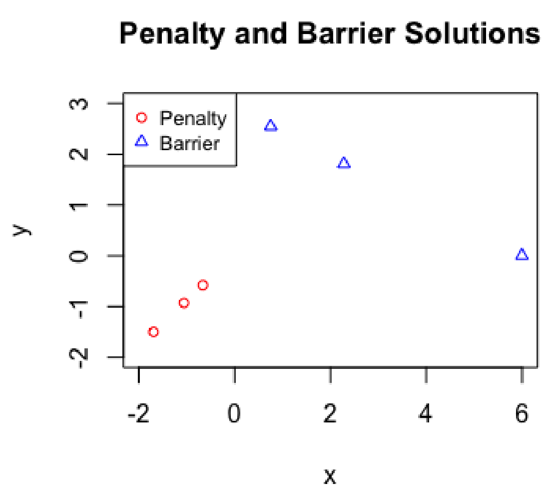
Yes, the solutions differ across μ.

(e) (10 points) Plot all of the solutions (x, y) on a single plot with colors that denote whether the Barrier Method or the Penalty method was used.

```
# Plot penalty solutions
x1 <- c(-1.69,-1.055,-0.661)
y1 <- c(-1.5,-0.929,-0.58)
plot(x1,y1,cex=.8,pch=1,
    xlab = "x", ylab = "y",
    main = "Penalty and Barrier Solutions",
    col="red",
    xlim = c(-2,6),ylim = c(-2,3))

# Plot barrier solutions
x2 <- c(5.99502,2.28,0.752)
y2 <- c(0.00198,1.81,2.549)
points(x2,y2,cex=.8,pch=2,col="blue")

# Add legend
legend("topleft", legend = c("Penalty","Barrier"),cex=.8,col=c("red","blue"),pch=c(1,2))
```



Question 2

(a) Write a simple Euler solver and program it.

```
lvsolve <- function(alpha, beta, S0, I0, R0, n) {
  Sout1 <- c(S0)
  Iout1 <- c(I0)
  Rout1 <- c(R0)

  for (i in 2:n) {
    S0h <- Sout1[i-1]
    I0h <- Iout1[i-1]
    R0h <- Rout1[i-1]
    Sout1[i] <- S0h - alpha*S0h*I0h
    Iout1[i] <- alpha*S0h*I0h - beta*I0h + I0h
    Rout1[i] <- beta*I0h + R0h
  }

  out1 <- data.frame(Sout1,Iout1,Rout1)
  names(out1) <- c("S","I","R")

  return(out1)
}
```

(b) Suppose we see the following data.

```
data1 <- matrix(c(
  0, 990, 10, 0,
  1, 986, 14, 0,
  2, 980, 19, 1,
  3, 973, 26, 1,
  4, 963, 36, 1,
  5, 949, 49, 2,
  6, 930, 67, 3,
  7, 905, 90, 5,
  8, 872, 120, 8,
  9, 830, 159, 11,
  10, 777, 207, 16,
  11, 713, 265, 22,
  12, 637, 333, 30,
  13, 552, 407, 41,
  14, 462, 483, 55,
  15, 373, 554, 73,
```

```
  16, 290, 614, 96
), nrow = 17, byrow = T)

data1 <- data.frame(data1)
names(data1) <- c('Day',"S","I","R")
```

(c) Set up a least squares function with barrier functions for the constraints and program it.

```
SSLV <- function(alpha,beta,S0,I0,R0,n){

  # Estimated output
  euler_out <- lvsolve(alpha,beta,S0,I0,R0,n)

  # Least squares calculation
  SS1_S <- sum(( euler_out$S - data1$S )^2)
  SS1_I <- sum(( euler_out$I - data1$I )^2)
  SS1_R <- sum(( euler_out$R - data1$R )^2)
  SS1 <- SS1_S + SS1_I + SS1_R

  # Add barrier functions for constaints
  SS1 <- SS1 + max( 0.0000001, 1/(alpha-0.00000001)^0.1 ) #changed
  SS1 <- SS1 + max( 0.0000001, 1/(beta-0.00000001)^0.1 ) #changed

  return (SS1) #changed
}
```

(d) Use Gauss-Newton method to solve for a and b. You will likely need to use numerical derivatives.

```
# Numerical derivatives
dSSLV <- function(param1,S0,I0,R0,n){
  h1 <- 0.0001

  alpha <- param1[1]
  beta <- param1[2]

  # alpha parameter
  g1 <- SSLV(alpha+h1,beta,S0,I0,R0,n) - SSLV(alpha-h1,beta,S0,I0,R0,n)
  g1 <- (g1) / (2*n*h1)

  # beta parameter
  g2 <- SSLV(alpha,beta+h1,S0,I0,R0,n) - SSLV(alpha,beta-h1,S0,I0,R0,n)
  g2 <- (g2) / (2*n*h1)

  return(c(g1,g2))
}
```

```
# Hessian Matrix
hessian1 <-function(param1,S0,I0,R0,n){

  h1 <- 0.000000001
  alpha <- param1[1]
  beta <- param1[2]

  matrix11 <- SSLV(alpha+h1,beta,S0,I0,R0,n) + SSLV(alpha-h1,beta,S0,I0,R0,n)
  matrix11 <- matrix11 - 2*SSLV(alpha,beta,S0,I0,R0,n)
  matrix11 <- matrix11/(n*h1^2)

  matrix22 <- SSLV(alpha,beta+h1,S0,I0,R0,n) + SSLV(alpha,beta-h1,S0,I0,R0,n)
  matrix22 <- matrix22-2*SSLV(alpha,beta,S0,I0,R0,n)
  matrix22 <- matrix22/(n*h1^2)

  matrix12 <- SSLV(alpha+h1,beta+h1,S0,I0,R0,n) + SSLV(alpha-h1,beta-h1,S0,I0,R0,n)
  matrix12 <- matrix12 - SSLV(alpha-h1,beta+h1,S0,I0,R0,n)- SSLV(alpha+h1,beta-h1,S0,I0,R0,n)
  matrix12 <- matrix12/(4*n*h1^2)

  result1 <- matrix(c(matrix11,matrix12,
                    matrix12,matrix22), nrow = 2, byrow = T)
  return(result1)

}

# Given starting values
S0 <- 990
I0 <- 10
R0 <- 0
n <- nrow(data1)
param1 <- c(0.00045,0.05) # Starting value

# Gausse-Newton Method
param1 <- param1 - solve( hessian1(param1,S0,I0,R0,n) )%*%dSSLV(param1,S0,I0,R0,n)

error1 = 1;
counter = 0;
while (error1 > 0.00001) {
  x1hold <- param1
  param1 <- param1 - solve( hessian1(param1,S0,I0,R0,n))%*%dSSLV(param1,S0,I0,R0,n)
  error1 <- sum( abs(x1hold - param1) )
  counter <- counter +1
}
```

```
# Question 2 TEST -------------------------------

# Plot given datapoints
plot(data1$Day,data1$S,
    xlab = "t", ylab = "",
    main = "Model of S, R, and I",
    xlim = c(0,18),ylim = c(0,1100),
    col = "magenta")
points(data1$Day,data1$I, col = "blue")
points(data1$Day,data1$R, col = "green")

# Plot estimated output
euler_out <- lvsolve(param1[1],param1[2],S0,I0,R0,n)
lines(euler_out$S, col = "magenta")
lines(euler_out$I, col = "blue")
lines(euler_out$R, col = "green")

# Legend
legend("left", legend=c("S(t)", "I(t)", "R(t)"),
      col=c("magenta", "blue", "green"), lty=1:2, cex=0.8)
```