Project 2

Zihuan Qiao U65762086

2016/10/8

1.(a)

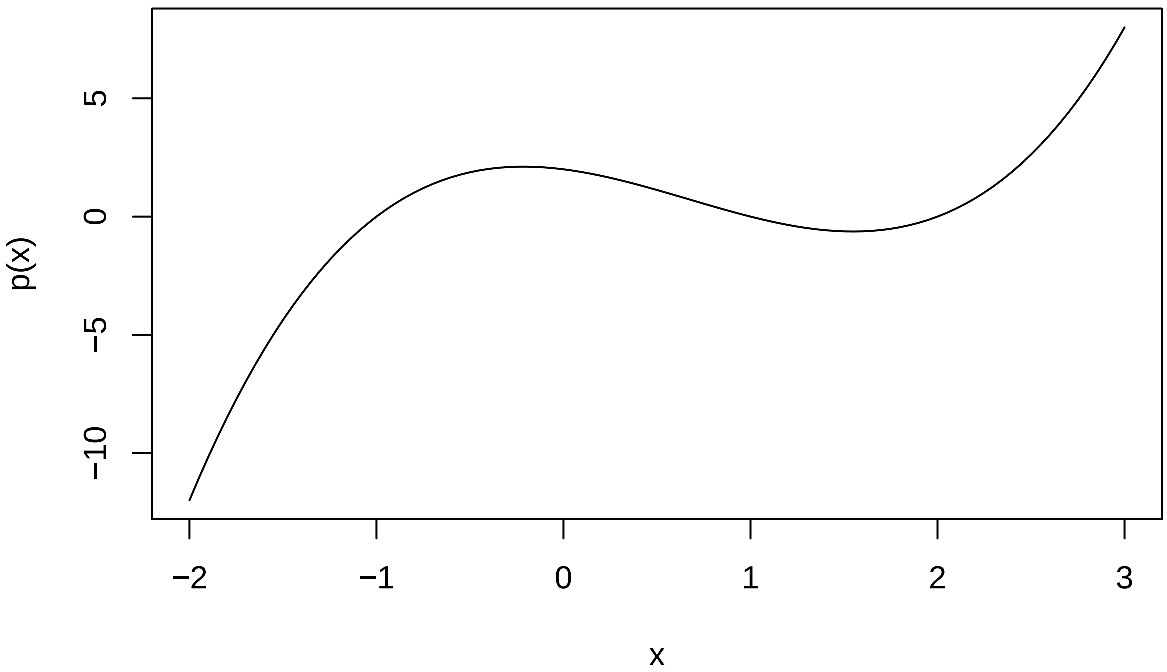
```
horner <- function (coef)
  return(function (x) {
    s <- 0
    for (i in seq(length(coef), 1, -1)) {
      s <- s * x + coef[i]
    }
    return(s) })
test <- c(1,-2,1)
p <- horner(test)
p
```

```
## function (x) {
##     s <- 0
##     for (i in seq(length(coef), 1, -1)) {
##       s <- s * x + coef[i]
##     }
##     return(s) }
## <environment: 0x7ff92410f240>
```

Explanation: Horner works because it starts by getting the coefficient for x^n, the coefficient for the term in the polynomial with the highest degree. Then every time in the loop, the result for the last loop times x and plus the coefficient for the term with the second highest degree. For each loop, all the terms entered times x, and the new term gets its coefficient. Finally, the term enters first times x for n times altogether, and gets its corresponding coefficient the time it enters. Similar cases are the other terms. The mathematical expression is: $p(x_i) = p(x_{i-1}) * x + coef[length(coef) - i + 1]$, $p(x_0) = 0$

1.(b)

```
horner <- function (coef)
  return(function (x) {
    s <- 0
    for (i in seq(length(coef), 1, -1)) {
      s <- s * x + coef[i]
    }
    return(s) })
p <- horner(c(2,-1,-2,1))
curve(p,-2,3)
```



1.(c)

```
horner <- function (coef)
  return(function (x) {
    s <- 0
    for (i in seq(length(coef), 1, -1)) {
      s <- s * x + coef[i]
    }
    return(s) })
hornerder <- function (coef)
  return(function (x) {
    s <- 0
    for (i in seq(length(coef), 2, -1)) {
      s <- s * x + (i-1)*coef[i]
    }
    return(s) })

test <- c(2,-1,-2,1)
p <- horner(test)
pder <- hornerder(test)

epsilon <- function()
{
  eps <- 1
  while((1+eps<=1)|(1+eps/2!=1))
  {
    eps <- eps/2
  }
  eps
}
```

```
root <- function(x)
{
  l=x;n=l-p(l)/pder(l) #l is last estimate root for p, n is next root for p
  while(abs(l-n)>=epsilon())
  {
    l=n
    n=l-p(l)/pder(l)
  }
  return(l)
}

root(-0.5)
```

```
## [1] -1
```

```
root(0)
```

```
## [1] 2
```

```
root(0.5)
```

```
## [1] 1
```

1.(c) annotation2:return the coefficient of p'

```
coep <- function(coef)
{
  a <- coef[-1]
  for(i in 1:length(a))
  {
    a[i] <- a[i]*i
  }
  return(a)
}

test <- c(2,-1,-2,1)
coep(test)
```

```
## [1] -1 -4  3
```

2.

```
#(a)
x <- seq(-2,3,0.5)
y <- c(-5.22,-2.21,0.11,0.94,0.71,-0.17,0.56,-0.94,-0.79,-1.15,-0.25)
X <- matrix(c(rep(1,11),x,x^2,x^3),nrow=11,byrow=FALSE)
XtX <- crossprod(X)
Xty <- crossprod(X,y)
T <- rbind(cbind(XtX,Xty),cbind(crossprod(y,X),crossprod(y)))
T
```

```
##         [,1]     [,2]     [,3]       [,4]      [,5]
## [1,] 11.000   5.5000  30.2500   42.62500  -8.41000
## [2,]  5.500  30.2500  42.6250  164.31250   7.03500
## [3,] 30.250  42.6250 164.3125  340.65625 -39.70250
## [4,] 42.625 164.3125 340.6562 1125.95312  15.31875
## [5,] -8.410   7.0350 -39.7025   15.31875  36.76750
```

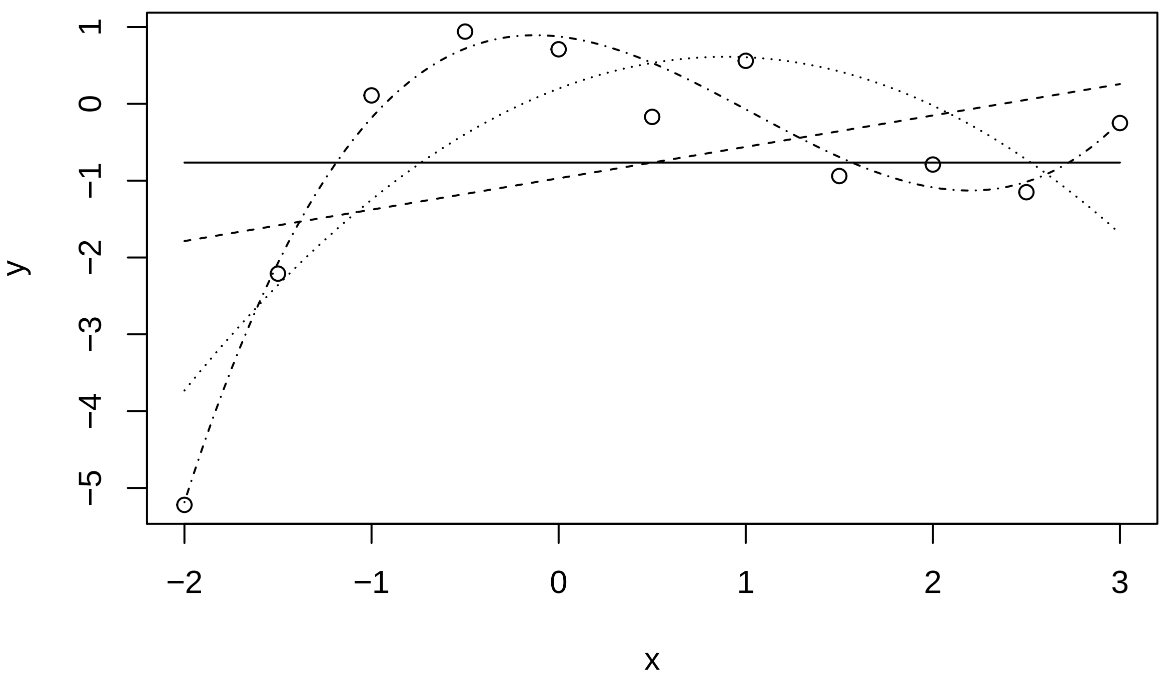
```
#(b)
SWEEP <- function(T,k)
{
  D <- T[k,k]
  T[k,] <- T[k,]/D
  for(i in 1:nrow(T))
  {
    if(i!=k){
    B <- T[i,k]
    T[i,] <- T[i,]-B*T[k,]
    T[i,k] <- -B/D
    }
  }
  T[k,k] <- 1/D
  return(T)
}

for(k in 1:4)
{
  print(SWEEP(SWEEP(T,k),k))
}
```

```
##         [,1]     [,2]     [,3]       [,4]      [,5]
## [1,] 11.000   5.5000  30.2500   42.62500  -8.41000
## [2,]  5.500  30.2500  42.6250  164.31250   7.03500
## [3,] 30.250  42.6250 164.3125  340.65625 -39.70250
## [4,] 42.625 164.3125 340.6562 1125.95312  15.31875
## [5,] -8.410   7.0350 -39.7025   15.31875  36.76750
##         [,1]     [,2]     [,3]       [,4]      [,5]
## [1,] 11.000   5.5000  30.2500   42.62500  -8.41000
## [2,]  5.500  30.2500  42.6250  164.31250   7.03500
## [3,] 30.250  42.6250 164.3125  340.65625 -39.70250
## [4,] 42.625 164.3125 340.6562 1125.95312  15.31875
## [5,] -8.410   7.0350 -39.7025   15.31875  36.76750
##         [,1]     [,2]     [,3]       [,4]      [,5]
## [1,] 11.000   5.5000  30.2500   42.62500  -8.41000
## [2,]  5.500  30.2500  42.6250  164.31250   7.03500
## [3,] 30.250  42.6250 164.3125  340.65625 -39.70250
## [4,] 42.625 164.3125 340.6562 1125.95312  15.31875
## [5,] -8.410   7.0350 -39.7025   15.31875  36.76750
##         [,1]     [,2]     [,3]       [,4]      [,5]
## [1,] 11.000   5.5000  30.2500   42.62500  -8.41000
## [2,]  5.500  30.2500  42.6250  164.31250   7.03500
## [3,] 30.250  42.6250 164.3125  340.65625 -39.70250
## [4,] 42.625 164.3125 340.6562 1125.95312  15.31875
## [5,] -8.410   7.0350 -39.7025   15.31875  36.76750
```

```r
#(c)
horner <- function (coef)
  return(function (x) {
    s <- 0
    for (i in seq(length(coef), 1, -1)) {
      s <- s * x + coef[i]
    }
    return(s) })

plot(x, y)
a <- seq(-2, 3, length.out = 100)
p <- ncol(T) - 1
for (k in 1:p) {
  T <- SWEEP(T, k)
  lines(a, horner(T[1:k, p + 1])(a), lty = k)
  print(c(k, T[p + 1, p + 1]))
}
```




```
## [1]  1.00000 30.33767
## [1]  2.00000 25.74358
## [1]  3.00000 11.31609
## [1] 4.000000 1.242871
```

A regression model with the degree that is k is plotted at each k. Each point on the lines or curves represents the value of the regression function, $y_i = \sum_{j=0}^{k-1} \beta_j x_i^j$ corresponding to every one in 100 a values. For example, when k is 2, the function plots the linear regression function $y_i = \beta_0 + \beta_1 x_i$ for x_i being 100 arithmetic progression numbers from -2 to 3. So when k is 2, a linear line is plotted. Similarly, when k is 1, it's a horizontal straight line. When k is 2, it's a linear line. When k is 3, it's a parabola. And when k is 4, it's a curve. Also we can see from the plot that when k is 4, the regression model fits the points the best.

The RSS value for the regression model with the degree that is k is printed at each k. For example, when k is 2, the RSS for the linear regression function $y_i = \beta_0 + \beta_1 x_i$ is printed as 25.74358. The smaller the value of RSS, the better the corresponding model fits the data.

2.(e)*

```
epsilon <- function()
{
  eps <- 1
  while((1+eps<=1)|(1+eps/2!=1))
  {
    eps <- eps/2
  }
  eps
}

eye <- function(matrix)
{
  diag(ncol(matrix))
}

x <- seq(-2,3,0.5)
y <- c(-5.22,-2.21,0.11,0.94,0.71,-0.17,0.56,-0.94,-0.79,-1.15,-0.25)
X <- matrix(c(rep(1,11),x,x^2,x^3),nrow=11,byrow=FALSE)
y <- as.matrix(y)

XtX <- crossprod(X)
Xty <- crossprod(X,y)

XtX[lower.tri(XtX,diag=FALSE)]=0
UD <- XtX
XtX <- crossprod(X)
XtX[upper.tri(XtX,diag=TRUE)]=0
L <- XtX

mod <- function(x)
{
  sum(x^2)/length(x)
}

GS <- function(beta0)
{
  l <- beta0
  n <- backsolve(UD,eye(UD))%*%(Xty-L%*%l)
  while(mod(abs(l-n))/mod(abs(l))>epsilon())
    {
      l <- n
      n <- backsolve(UD,eye(UD))%*%(Xty-L%*%l)
    }
  n
}

GS(rep(1,4))
```

```
##             [,1]
## [1,]  0.8765501
## [2,] -0.2678476
```

```
## [3,] -1.0032634
## [4,]  0.3230458
```