STAT 579 Homework 7

Yifan Zhu

November 3, 2016

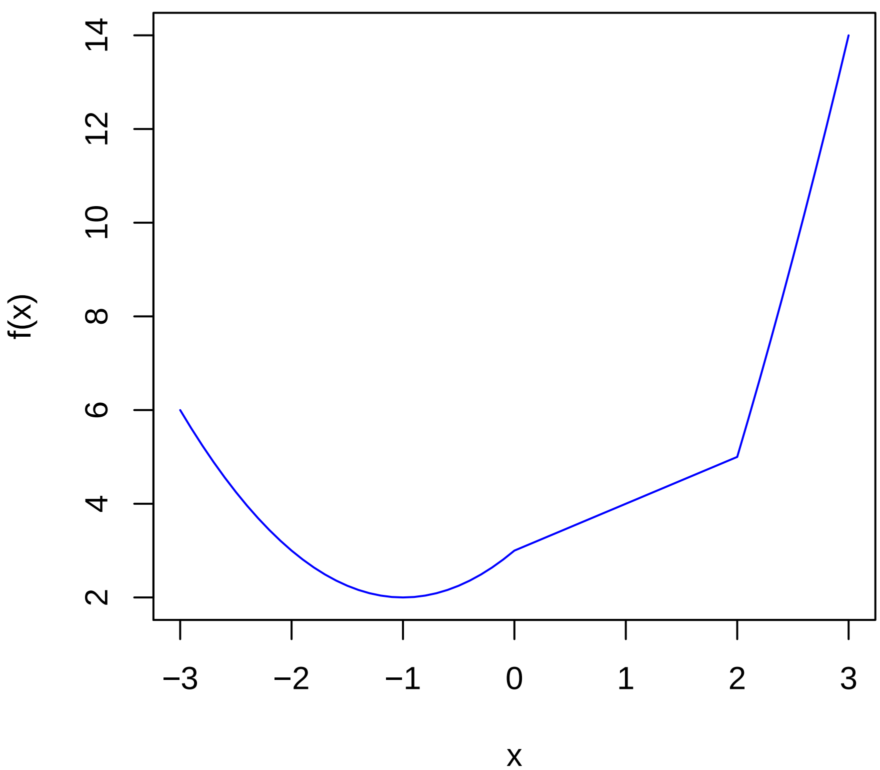
Problem 1

```
tridiag <- function(k, n) {
    A <- matrix(rep(0, n * n), ncol = n)
    diag(A) <- rep(k, n)
    diag(A[-1, -n]) <- rep(1, n - 1)
    diag(A[-n, -1]) <- rep(1, n - 1)
    return(A)
}

tridiag(n = 6, k = 5)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    1    0    0    0    0
## [2,]    1    5    1    0    0    0
## [3,]    0    1    5    1    0    0
## [4,]    0    0    1    5    1    0
## [5,]    0    0    0    1    5    1
## [6,]    0    0    0    0    1    5
```

Problem 2

```
tmpFn <- function(xVec) {
    x <- xVec
    ifelse(x < 0, x^2 + 2 * x + 3, ifelse(x < 2, x + 3, x^2 + 4 * x - 7))
}
x <- seq(-3, 3, 0.1)
fx <- tmpFn(x)

plot(x = x, y = fx, "l", col = "blue", xlab = "x", ylab = "f(x)")
```



Problem 3

```
gcd <- function(m, n) {
    re <- m%%n
    while (re != 0) {
        m <- n
        n <- re
        re <- m%%n
    }
    return(n)
}

gcd(60, 130)
```

```
## [1] 10
```

Problem 4

```r
order.matrix <- function(x) {
    dimx <- dim(x)
    ord <- order(x)
    n <- length(ord)
    dim(ord) <- dimx
    indices <- NULL
    for (i in 1:n) {
        indice <- which(ord == i, arr.ind = T)
        indices <- rbind(indices, indice)
    }
    indices <- cbind(1:n, indices)
    colnames(indices) <- c("order", "row", "column")
    return(indices)
}

x <- matrix(rchisq(n = 4 * 3, df = 1), ncol = 3)

order.matrix(x)
```

```
##       order row column
##  [1,]     1   2      2
##  [2,]     2   3      1
##  [3,]     3   3      3
##  [4,]     4   2      1
##  [5,]     5   3      2
##  [6,]     6   1      1
##  [7,]     7   4      3
##  [8,]     8   1      2
##  [9,]     9   4      1
## [10,]    10   1      3
## [11,]    11   4      2
## [12,]    12   2      3
```

Problem 5

(a)

```
polaroid <- function(x) {
    p <- length(x)
    q <- rep(0, p - 1)
    R <- sqrt(sum(x^2))
    q[p - 1] <- ifelse(atan(x[p]/x[p - 1]) >= 0, atan(x[p]/x[p - 1]), atan(x[p]/x[p -
        1]) + pi)
    for (i in (p - 2):2) {
        arctan <- atan((x[i + 1]/cos(q[i + 1]))/x[i])
        q[i] <- ifelse(arctan >= 0, arctan, arctan + pi)
    }
    q[1] <- ifelse(x[2]/cos(q[2]) >= 0, atan2(y = x[2]/cos(q[2]), x = x[1]),
        atan2(y = x[2]/cos(q[2]), x = x[1]) + 2 * pi)

    return(c(R, q))
}
```

(b)

```
normalize <- function(X) {
    norm <- sqrt(apply(X^2, MARGIN = 1, FUN = sum))
    sweep(X, MARGIN = 1, STAT = norm, FUN = "/")
}
```

(c)

```
y <- matrix(rnorm(n = 1000 * 5), ncol = 5)

z <- normalize(y)

kstest <- apply(z, MARGIN = 2, FUN = ks.test, "punif", min = -1, max = 1)

kstest
```

```
## [[1]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  newX[, i]
## D = 0.10213, p-value = 1.742e-09
## alternative hypothesis: two-sided
##
##
## [[2]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  newX[, i]
## D = 0.11491, p-value = 6.792e-12
```

```
## alternative hypothesis: two-sided
##
##
## [[3]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  newX[, i]
## D = 0.11309, p-value = 1.559e-11
## alternative hypothesis: two-sided
##
##
## [[4]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  newX[, i]
## D = 0.10606, p-value = 3.397e-10
## alternative hypothesis: two-sided
##
##
## [[5]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  newX[, i]
## D = 0.092739, p-value = 6.773e-08
## alternative hypothesis: two-sided
```

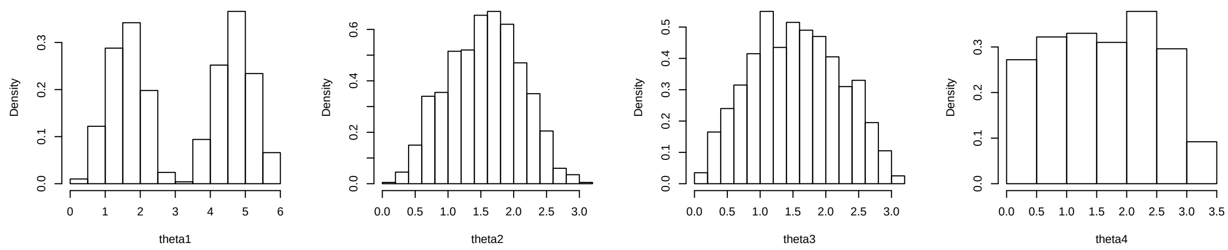
P-values are small, so they should not follow the uniform distribution on $(-1, 1)$.

(d)

```
polar <- apply(y, MARGIN = 1, FUN = polaroid)

R2 <- (polar[1, ])^2

ks.test(R2, "pchisq", df = 5)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  R2
## D = 0.025898, p-value = 0.5136
## alternative hypothesis: two-sided
```

```
theta <- polar[-1, ]

par(mfrow = c(1, 4))

hist(theta[1, ], xlab = "theta1", freq = F, main = "")
hist(theta[2, ], xlab = "theta2", freq = F, main = "")
hist(theta[3, ], xlab = "theta3", freq = F, main = "")
hist(theta[4, ], xlab = "theta4", freq = F, main = "")
```




```r
ks.test(theta[1, ], "punif", min = 0, max = 2 * pi)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  theta[1, ]
## D = 0.1087, p-value = 1.089e-10
## alternative hypothesis: two-sided
```

```r
ks.test(theta[2, ], "punif", min = 0, max = pi)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  theta[2, ]
## D = 0.18376, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
ks.test(theta[3, ], "punif", min = 0, max = pi)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  theta[3, ]
## D = 0.1122, p-value = 2.329e-11
## alternative hypothesis: two-sided
```

```r
ks.test(theta[4, ], "punif", min = 0, max = pi)
```

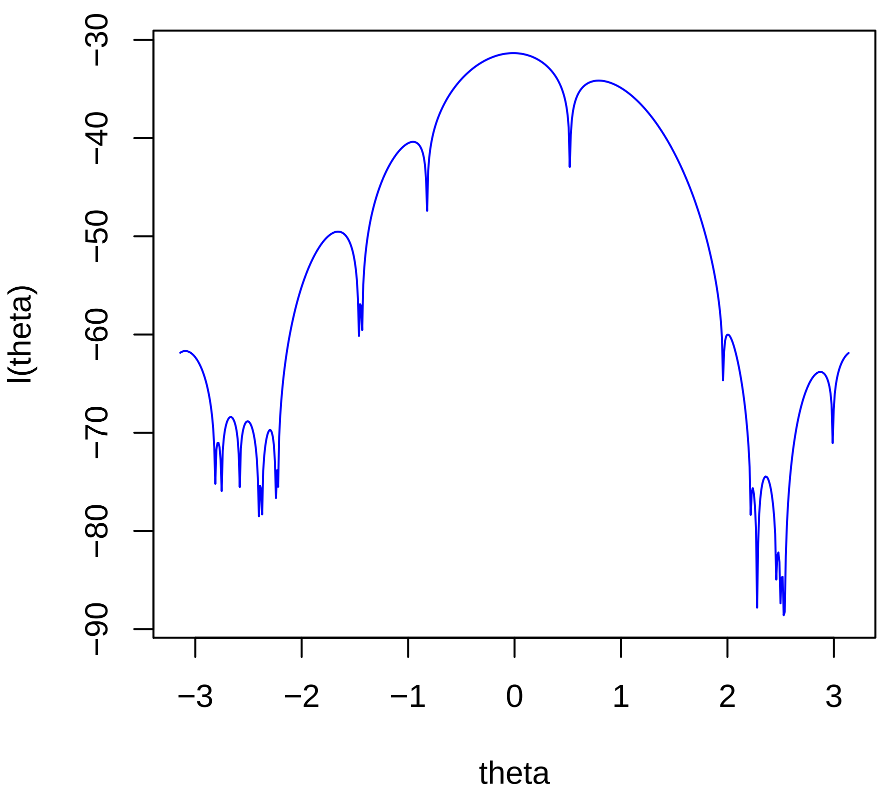
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  theta[4, ]
## D = 0.034434, p-value = 0.1865
## alternative hypothesis: two-sided
```

P-value for test for R^2 is large, so it shoould follow the χ^2_5. P-value for $\theta_1, \theta_2, \theta_3$ are small, so they should not follow uniform distribution. P-value for θ_4 is large, thus it should follow the uniform distribution on $[0, \pi)$.

Problem 6

(a)

```
x <- c(3.91, 4.85, 2.28, 4.06, 3.7, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88,
    2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.5)

l <- function(theta, x) {
    n <- length(x)
    -n * log(2 * pi) + sum(log(1 - cos(x - theta)))
}

theta <- seq(-pi, pi, 0.01)

ltheta <- apply(t(theta), MARGIN = 2, FUN = l, x = x)

plot(x = theta, y = ltheta, "l", col = "blue", xlab = "theta", ylab = "l(theta)",
    main = "")
```



(b)

```r
thetamin <- optimize(f = l, x = x, interval = c(-pi, pi), maximum = T)

thetamin
```

```
## $maximum
## [1] -0.0119724
##
## $objective
## [1] -31.34291
```

(c)

```r
options(digits = 20)

newton <- function(fun, derf, x0, eps) {
    iter <- 0
    repeat {
        iter <- iter + 1
        x1 <- x0 - fun(x0)/derf(x0)
        if (abs(x0 - x1) < eps || abs(fun(x1)) < 1e-10)
            break
        x0 <- x1
        cat("****** Iter. No: ", iter, " Current Iterate = ", x1, fill = T)
    }
    return(x1)
}

derl <- function(theta) {
    sum(sin(x - theta)/(1 - cos(x - theta)))
}

dderl <- function(theta) {
    -sum(1/(1 - cos(x - theta)))
}

newton(fun = derl, derf = dderl, x0 = 0, eps = 1e-04)
```

```
## ****** Iter. No:  1  Current Iterate =  0.011911932605691015
## ****** Iter. No:  2  Current Iterate =  0.035538787271073638
## ****** Iter. No:  3  Current Iterate =  0.081887140686978999
## ****** Iter. No:  4  Current Iterate =  0.17003691781937486
## ****** Iter. No:  5  Current Iterate =  0.32130252264058118
## ****** Iter. No:  6  Current Iterate =  0.49222040572661963
## ****** Iter. No:  7  Current Iterate =  0.52107506423276551
## ****** Iter. No:  8  Current Iterate =  0.52000207000084642
```

```
## [1] 0.52000000000761182
```

(d)

```r
newton(fun = derl, derf = dderl, x0 = -2, eps = 1e-04)
```

```
## ****** Iter. No:  1  Current Iterate =  -2.243846446394639
## ****** Iter. No:  2  Current Iterate =  -2.24317446162916

## [1] -2.2431853038615284
```

```r
newton(fun = derl, derf = dderl, x0 = -2.7, eps = 1e-04)
```

```
## ****** Iter. No:  1  Current Iterate =  -2.7258863441687975
## ****** Iter. No:  2  Current Iterate =  -2.7497175268270801
## ****** Iter. No:  3  Current Iterate =  -2.75325274417618

## [1] -2.753185343300327
```

The results are different with different initial point. It is because Newton method will end with it finds a point with zero value which is nearest to the starting point. And the function has many points when its derivative is 0.