Carson Slater

6375 Homework 6 Solutions

Question 1

(a)

```
forward_substitute <- function(L, b) {

  stopifnot(is.atomic(b))
  stopifnot(is.matrix(L))
  stopifnot(length(b) == nrow(L))
  stopifnot(is.zero(L[upper.tri(L)]))
  stopifnot(!is.zero(diag(L)))

  n <- length(b)
  x <- numeric(n)

  for (i in 1:n) {
    x[i] <- (b[i] - total(L[i, 1:(i - 1)] * x[1:(i - 1)])) / L[i, i]
  }
  x
}
```

```
# testing forward_substitute()
L <- matrix(c(
  5, 0, 0,
  7, 2, 0,
  4, 1, 9
), nrow = 3, byrow = TRUE)

b <- c(2, 1, 4)

forward_substitute(L, b)
# [1]  0.4000000 -0.9000000  0.3666667

base::forwardsolve(L, b)
# [1]  0.4000000 -0.9000000  0.3666667

solve(L, b)
```

```
# [1]  0.4000000 -0.9000000  0.3666667
```

(b)

```
back_substitute <- function(U, b) {

  stopifnot(is.atomic(b))
  stopifnot(is.matrix(U))
  stopifnot(length(b) == nrow(U))
  stopifnot(is.zero(U[lower.tri(U)]))
  stopifnot(!is.zero(diag(U)))

  n <- length(b)
  x <- numeric(n)

  x[n] <- b[n]/U[n,n]

  for (i in seq(n - 1, 1, -1)) {
     x[i] <- (b[i] - total(U[i, (i + 1):n] * x[(i + 1):n])) / U[i, i]
  }
  x
}
```

```
# testing back_substitute
n <- 5; n_elts <- (n*(n+1)/2)
U <- diag(n)
U[upper.tri(U, diag = TRUE)] <- 1:n_elts
U
#      [,1] [,2] [,3] [,4] [,5]
# [1,]    1    2    4    7   11
# [2,]    0    3    5    8   12
# [3,]    0    0    6    9   13
# [4,]    0    0    0   10   14
# [5,]    0    0    0    0   15

y <- 1:n

back_substitute(U, y)
# [1] -1.14074074 -0.28518519 -0.12222222 -0.06666667  0.33333333
```

```r
base::backsolve(U, y)
# [1] -1.14074074 -0.28518519 -0.12222222 -0.06666667  0.33333333

solve(U, y)
# [1] -1.14074074 -0.28518519 -0.12222222 -0.06666667  0.33333333
```

Question 2

I have read Trefethen and Bau Lecture 20.

Question 3

(a)

```r
row_reduce <- function(A) {

  stopifnot(nrow(A) == ncol(A))

  n <- nrow(A)

  # iterate through the columns
  for (i in 1:(n - 1)) {
    # iterate through the entries in each column below the diagonal
    for (j in (i + 1):n) {
      multiplier <- A[j, i] / A[i, i]
      A[j, ] <- A[j, ] - multiplier * A[i, ]
    }
  }
  A
}
```

```r
# testing row_reduce
A <- matrix(c(
  2, 1, 1, 0,
  4, 3, 3, 1,
  8, 7, 9, 5,
  6, 7, 9, 8
), nrow = 4, byrow = TRUE)
row_reduce(A)
```

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

(b)

```
LU <- function(A) {

  stopifnot(nrow(A) == ncol(A))

  n <- nrow(A)
  L <- diag(n)
  lower_vals <- numeric()

  # iterate through the columns
  for (i in 1:(n - 1)) {
    # iterate through the entries in each column below the diagonal
    for (j in (i + 1):n) {
      multiplier <- A[j, i] / A[i, i]
      # save the multipliers for the L matrix
      lower_vals <- c(lower_vals, multiplier)
      A[j, ] <- A[j, ] - multiplier * A[i, ]
    }
  }
  L[lower.tri(L)] <- lower_vals

  list("L" = L, "U" = A)
}
```

```
# testing LU()
LU(A)
# $L
#      [,1] [,2] [,3] [,4]
# [1,]    1    0    0    0
# [2,]    2    1    0    0
# [3,]    4    3    1    0
# [4,]    3    4    1    1
#
```

```
# $U
#      [,1] [,2] [,3] [,4]
# [1,]    2    1    1    0
# [2,]    0    1    1    1
# [3,]    0    0    2    2
# [4,]    0    0    0    2

with( LU(A), L %*% U )
#      [,1] [,2] [,3] [,4]
# [1,]    2    1    1    0
# [2,]    4    3    3    1
# [3,]    8    7    9    5
# [4,]    6    7    9    8
```

Question 4

(a)

```
solve_gaussian <- function(A, b) {
  LU <- LU(A)

  y <- forward_substitute(LU$L, b)

  back_substitute(LU$U, y)
}
```

```
# testing solve_gaussian()
b <- 1:4
solve_gaussian(A, b)
# [1]  1.0  0.5 -1.5  1.0

solve(A, b)
# [1]  1.0  0.5 -1.5  1.0
```

(b)

```
solve_qr_hh <- function(A, b) {
  QR <- hh(A)
```

```
  y <- t(QR$Q) %*% b

  back_substitute(QR$R, y)
}
```

```
solve_qr_hh(A, b)
# [1]  1.0  0.5 -1.5  1.0

solve(A, b)
# [1]  1.0  0.5 -1.5  1.0
```