Homework 5 Solutions.
Authors: Luther Lu and Nima Leclerc
Question 1

a. Given an $n\ x\ n$ matrix A, prove that the time complexity for computing A^{-1} with gaussian elimination is $O(n^3)$.
b. Assume that we are given the QR factorization of A. What is the optimal complexity for computing $e_n{}^T A^{-1} e_i$?
c. Devise an algorithm that achieves this and demonstrate that your code scales correctly.

Solutions

a. We would like to find the $n\ x\ n$ matrix B that satisfies the equation below.

$$\begin{bmatrix} x & \cdots & \cdots & \cdots & x \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ x & \cdots & \cdots & \cdots & x \end{bmatrix} B = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

Therefore, we will perform gaussian elimination on the matrix A.

$$\begin{bmatrix} x & \cdots & \cdots & \cdots & x \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ x & \cdots & \cdots & \cdots & x \end{bmatrix}$$

And reduce it to the identity matrix. Note that x may represent a different value each time. The first step is zeroing all entries below the first row in the first column. By doing so, we need to modify the $(n-1)\ x\ (n-1)$ matrix block at the bottom right corner. This action therefore costs $(n-1)^2$ operations and the resultant matrix is

$$\begin{bmatrix} x & \cdots & \cdots & \cdots & x \\ 0 & x & \cdots & \cdots & \vdots \\ 0 & x & \ddots & \cdots & \vdots \\ 0 & x & \cdots & \ddots & \vdots \\ 0 & x & \cdots & \cdots & x \end{bmatrix}$$

We would like to subsequently perform these gaussian elimination steps for the next columns. We will see that this next step costs $(n-2)^2$ operations. When we have reached the last column, we will have performed a total of

$$(n-1)^2 + (n-2)^2 + ... + 1^2$$

operations. We can express this as the sum of the first $n-1$ square numbers. If one performs the above operation over every *n* iterations (hence perform a sum over *n* terms), the expected complexity for the independent sum is of order *O(n)*. However, each step requires a quadratic operation. From an intuitive standpoint, this would ultimately yield a complexity of order $O(n^3)$ *.*

b. If we are given $A = QR$, we know that $A^{-1} = R^{-1}Q^T$. Now we would like to find $e_n{}^T R^{-1} Q^T e_i$. This is equivalent to extracting the i^{th} element from the last row in $R^{-1}Q^T$. We expect that R^{-1} is also upper triangular and that we won't need to explicitly take the transpose of Q. To get the last row of $R^{-1}Q^T$, we will need the last row of R^{-1}. Since we know that R^{-1} is upper triangular, there is only one nonzero entry and that

$$R_{n,n}^{-1} = \frac{1}{R_{n,n}}$$

Therefore, the last row of $R^{-1}Q^T$ is essentially

$$\begin{bmatrix} \frac{Q_{n,1}^T}{R_{n,n}} & \frac{Q_{n,2}^T}{R_{n,n}} & \cdots & \frac{Q_{n,n}^T}{R_{n,n}} \end{bmatrix}$$

Now we would like to extract the i^{th} element of this array. Computing

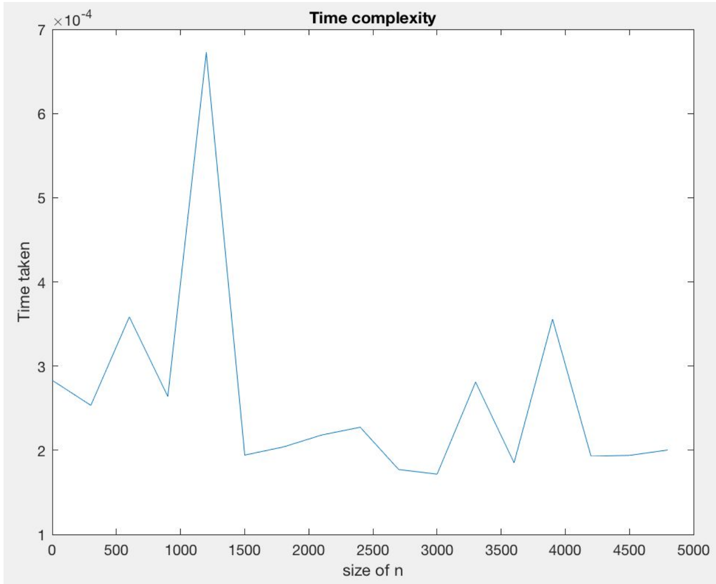
$$\frac{Q_{n,i}^T}{R_{n,n}}$$

will take constant time.

c. First I wrote a script to test that my formulation of

$$\frac{Q_{n,i}^T}{R_{n,n}}$$

is correct. Indeed this is correct.



We then analyze the time complexity of performing this calculation. The size of n ranges from 1 to 5000. We see that the time for the operation stays relatively the same. Therefore, it is a constant time operation.

Code:

```
tim = [];
for n = 1 : 300: 5000
    A = rand(n);
    [Q, R] = qr(A);
    tic;
    Q(ceil(n/2), n)/R(n,n)
    tim(end+1) = toc;
end
```