Problem 1

Suppose $\epsilon \to 0$.

(a) Consider $(1 + O(\epsilon))(1 + O(\epsilon))$. Multiplying this out, this becomes $1 + 2O(\epsilon) + O(\epsilon)^2$. We can rewrite these terms as $1 + c * \epsilon + d * \epsilon + e * f * \epsilon^2$, where c, d, e, f are arbitrary constants. Because $\epsilon \to 0$, ϵ^2 is significantly smaller than the other terms and thus becomes 0. Likewise, we can rewrite $c * \epsilon + d * \epsilon$ as $h * \epsilon$, which can then be wrote as $O(\epsilon)$. Thus our result becomes $1 + O(\epsilon)$ as desired.

(b) Consider $(1 + O(\epsilon))^{-1}$. We know that $(1 + O(\epsilon))^{-1} * (1 + O(\epsilon)) = 1$ by definition. From part (a), we know $(1 + O(\epsilon))(1 + O(\epsilon)) = (1 + O(\epsilon))$. Thus we may write $(1 + O(\epsilon))^{-1} * (1 + O(\epsilon)) * (1 + O(\epsilon)) = 1$. Applying this a second time give us $(1 + O(\epsilon))^{-1} * (1 + O(\epsilon)) * (1 + O(\epsilon)) * (1 + O(\epsilon)) = 1$. Cancelling terms gives us $(1 + O(\epsilon)) * (1 + O(\epsilon)) = 1$. Dividing by $(1 + O(\epsilon))$, we find that $(1 + O(\epsilon))^{-1} = (1 + O(\epsilon))$ as desired.

Problem 2

Consider the matrix

$$A = \begin{bmatrix} 375 & 374 \\ 752 & 750 \end{bmatrix}$$

(a) Recall that $A^{-1} = \frac{1}{\det A}[\mathrm{tr}\,(A)I - A]$ Thus,

$$A^{-1} = \begin{bmatrix} 375 & -187 \\ -376 & \frac{375}{2} \end{bmatrix}$$

Then, $\kappa_\infty = ||A||_\infty ||A^{-1}||_\infty = 1502 * 562 = 844124$.

(b) For this part, we will chose b using A^{-1} to find an appropriate x, then perturb b by δb to find δx. Consider $b = [1, 2]^T$ and $\delta b = [0.1, 0.7]^T$. Then we find $x = [1, -1]^T$ and $\delta x = [-92.4, 92.65]^T$. Thus we find that

$$\frac{||\delta b||_\infty}{||b||_\infty} = \frac{0.7}{2} = 0.35, \quad \frac{||\delta x||_\infty}{||x||_\infty} = \frac{92.65}{1} = 92.65$$

Problem 3

Let $x \in \mathbb{C}^2$ and consider

$$Q = \begin{bmatrix} c & \bar{s} \\ -s & c \end{bmatrix}$$

where $c \in \mathbb{R}$ and $c^2 + |s|^2 = 1$. Write x as $x = [x_1, x_2]^T$. We seek c and s such that $Q^* x$ has 0 as its second element. We have

$$\begin{bmatrix} c & -\bar{s} \\ s & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

This can be written as

$$\begin{cases} c * x_1 - \bar{s} * x_2 = r \\ s * x_1 + c * x_2 = 0 \end{cases}$$

Looking at the first row, we see that by dividing by r we obtain $c * \frac{x_1}{r} - \bar{s} * \frac{x_2}{r} = 1$. Since $c^2 + |s|^2 = 1$, we speculate that $c = \frac{x_1}{r}$ and $s = -\frac{x_2}{r}$. Notice that this satisfies $-\frac{x_2}{r} * x_1 + \frac{x_1}{r} x_2 = 0$, so

$$Q = \begin{bmatrix} x_1/r & -x_2/r \\ x_2/r & x_1/r \end{bmatrix}$$

Problem 4

For this problem, I was not sure what would be a good method to alter Householder, so I instead implemented a Givens rotation QR factorization. I did this because the Toeplitz matrix is sparse, which is not optimal for Householder but is optimal for Givens. My code for this can be seen in the function "givensQR" in the file "QR.py". I believe the cost for this is $O(m^4)$ because there is a for loop nested inside another one, and within this for loop there is a matrix multiplication, which takes approximately m^2 time.

Problem 5

(a) To see my code for this, please reference the function "svdmatrixgen" in the file "QR.py". The idea behind this function is to create a diagonal matrix with the required singular values, then to use one of the QR factorization methods on a random matrix to generate a Q. Then the matrix is computed as $A = QDQ^*$, where D is the diagonal matrix.

(b) To see my code for this, please refer to the functions "classicGR" and modifiedGR" in the file "QR.py". I ran a numerical experiment for matrices of size $10 * 10$ to size $490 * 490$. These results can be seen in the file "results.txt"; as the size of the matrices increase it is noted that the difference between the modified GS and the classic GS increases substantially. This can be explained by the instability of the classical GS, thus it makes sense that as the sizes of the matrices increase, the error increases too.

(c) To see my code for this, please refer to the function "householderQR" in the file "QR.py". I performed numerical experiments comparing the Q matrices of the householder decomposition and modified GS, whose results can be seen in "results2.txt". From this, something clearly is amiss[1]. I can imagine one of two problems: either one of the methods is poorly optimized for the matrix we are using, or one of my functions is poorly written. If it is the first issue, I believe that since the householder method is good for dense matrices, the modified GS is the issue. However, if it is the second issue, then it is because my householderQR function is not particularly good.

[1] I was sick this weekend, so I was not able to fix this unfortunately.