Problem 1

Suppose A is a lower triangular matrix. Let I be the identity matrix. We wish to show that A^{-1} is also lower triangular. Notice that $I = AA^{-1}$. Let a_{ij} and b_{ij} be the elements of A and A^{-1} respectively. Then we can establish the following relationship:

$$\begin{cases} a_{11}b_{1j} = \delta_{1j} \\ a_{21}b_{1j} + a_{22}b_{2j} = \delta_{2j} \\ \vdots \\ a_{n1}b_{1j} + \ldots + a_{nn}b_{nj} = \delta_{nj} \end{cases}$$

where δ_{ij} is the Kronecker Delta. This may be rewritten as follows:

$$\begin{cases} b_{1j} = \frac{\delta_{1j}}{a_{11}} \\ b_{2j} = \frac{\delta_{2j} - a_{21}b_{1j}}{a_{22}} \\ \vdots \\ b_{nj} = \frac{\delta_{nj} - \sum_{k=1}^{n-1} a_{nk}b_{kj}}{a_{nn}} \end{cases}$$

Thus it can be seen that whenever the column index exceeds the row index for b_{ij}, $b_{ij} = 0$, so A^{-1} is lower triangular. □

Problem 2

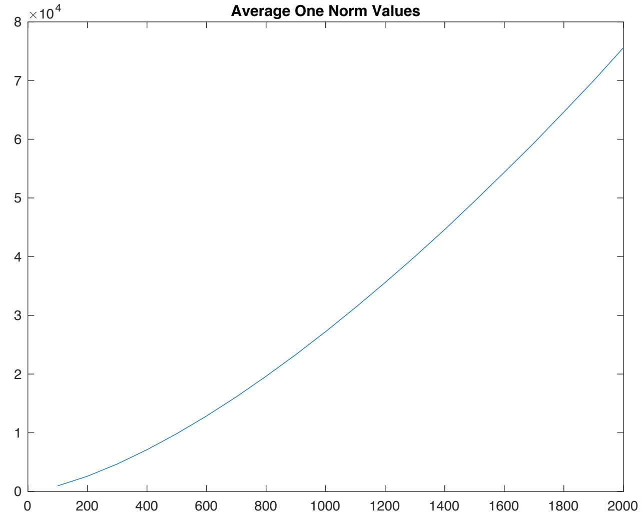
Let $||A||$ be an induced matrix on some $A \in \mathbb{C}^{m*m}$. By definition, $\rho(A) = \lambda_{\max}$. Consider the eigenvector for $\lambda_{\max}$, x. By definition, $Ax = \lambda_{\max}x$. Taking their norms gives $||Ax|| = ||\lambda_{\max}x|| = |\lambda_{\max}||x|$. This can be rewritten as $||A|||x| \geq |\lambda_{\max}|x|$, then removing $|x|$ gives $||A|| \geq |\lambda_{\max} = \rho(A)$ as desired. □

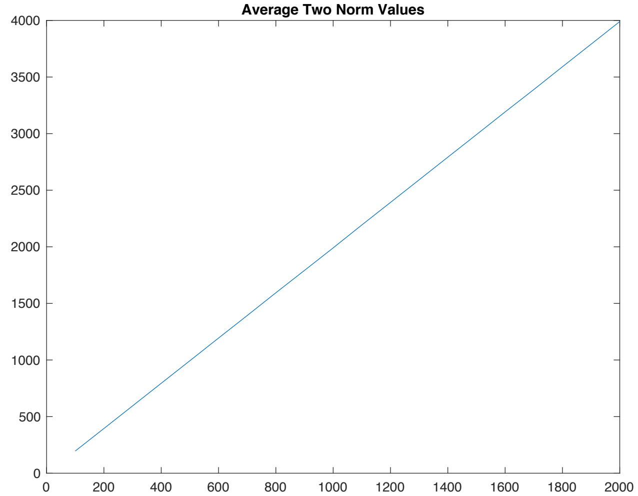
Problem 3

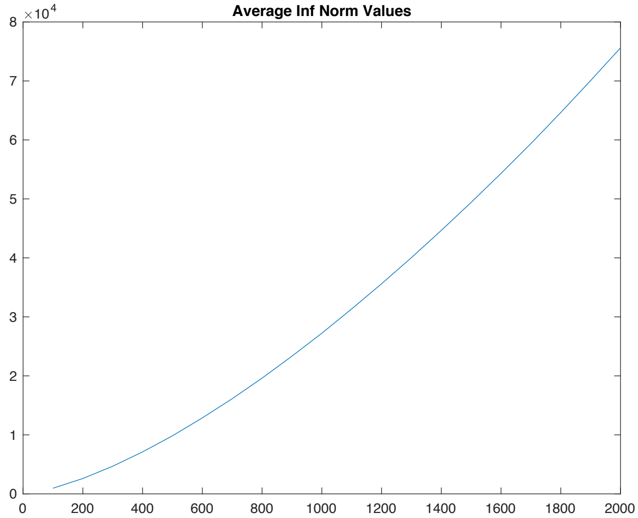
Suppose $A \in \mathbb{C}^{202*202}$ with $||A||_2 = 100$ and $||A||_F = 101$. From the text, we know $||A||_2 = \sigma_1$ and $||A||_F = \sqrt{\sigma_1^2 + \ldots + \sigma_n^2}$. Thus, $101 = \sqrt{10000 + \ldots + \sigma_{202}^2}$ after plugging in values. Rearranging, this becomes $201 = \sigma_2^2 + \ldots + \sigma_{202}^2$. Since the smallest value σ_{202} can obtain is 1, we arrive at $\sigma_1/\sigma_{202} \geq 100$.

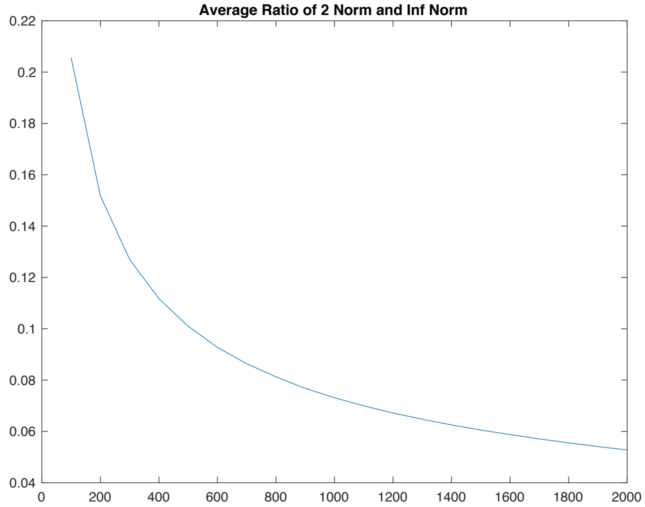
Problem 4

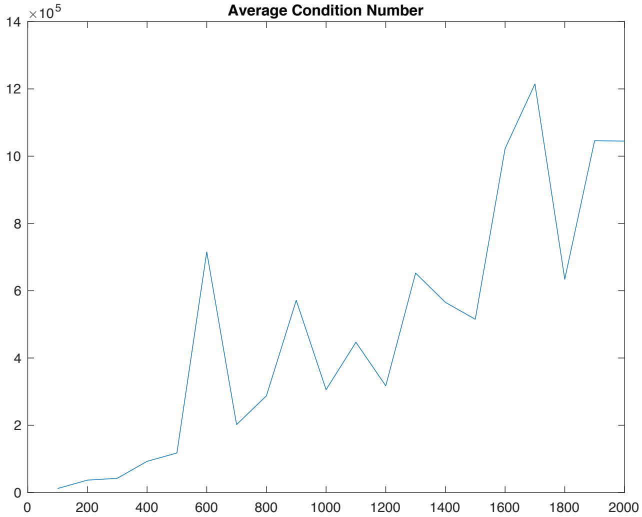
For this problem, I implemented a code "Problem4" in MATLAB that generated random matrices of size $m * m$ with a mean value of 0 and a standard deviation of $\sqrt{m}$. Using MATLAB's built-in norm functions, I obtained the p-norm for p$= 1, 2, \infty$ for the random matrices. As my GTID is odd, I computed the ratio between the 2-norm and infinity-norm as well. Additionally, I computed the average condition number for the matrices. My results are shown below:









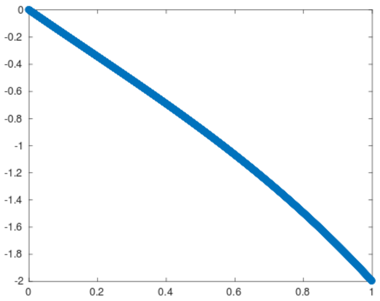


These results follow intuition, I believe. Both the one norm and inf norm can be defined as the maximum of the sum of columns and rows, respectively. So as $m \to \infty$, it follows that $||A||_1 \to \infty$ and $||A||_\infty \to \infty$. The $||A||_2$ results behave in a more linear manner, which is makes sense given $||A||_2\ |\lambda_{max}|$. Because the two norm behaves linearly and the inf norm behaves super linearly, the ratio between the two should cover to 0 as the inf norm increases at a faster rate. The condition number for the random matrix overall increases as the size of the matrix increases, though there are dips and spikes in the results. Overall, this still makes sense as a random matrix will tend to being ill-conditioned as it's size increases.

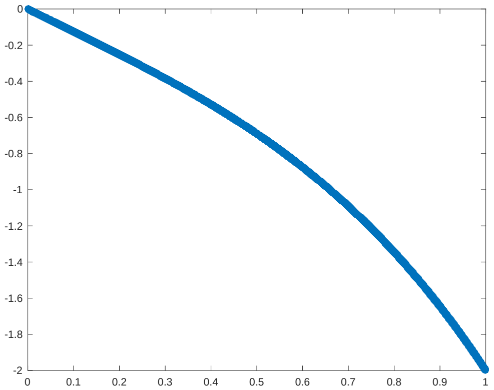
Problem 5

For this problem, we had to approximate solutions to $-u'' + \lambda u = 3x - \frac{1}{2}$ using Gaussian elimination with and without partial pivoting. My MATLAB code for this is called "Problem5". My results may be seen in the following graphs:

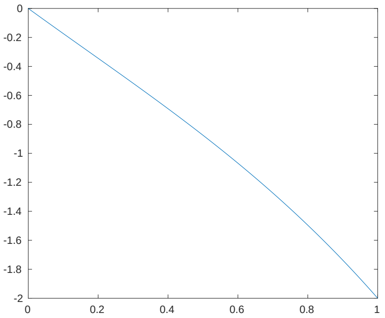
(a) Gaussian elimination for $\lambda = 0$



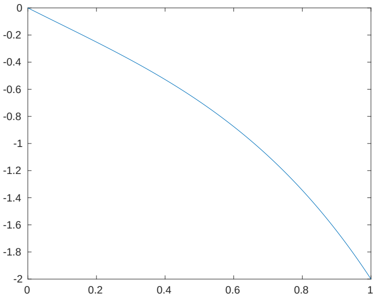
(b) Gaussian elimination for $\lambda = 2$



(c) True solution for $\lambda = 0$



(d) True solution for $\lambda = 2$



When comparing the true solutions to the differential equation with the approximation, for $\lambda = 0$ the average relative error was on the scale of 10^{-10} while for $\lambda = 2$ the average relative error was on the scale of 10^{-16}. These are extremely good results, but not surprising as small step sizes were used, which lead to refined answers. For the various computations, the CPU runtimes are presented below:

Table 1: CPU Runtimes in seconds

Size	$\lambda = 0$ without pivoting	$\lambda = 0$ with pivoting	$\lambda = 2$ without pivoting	$\lambda = 2$ with pivoting
200	0.096323	0.264187	0.142005	0.156812
400	0.300473	0.552547	0.527399	0.421511
800	2.041219	3.074913	4.295169	2.745189
1000	3.963348	5.379148	8.774305	6.098762
2000	34.324527	40.059037	107.276456	85.589778
4000	398.864609	313.789217	906.203220	859.576564

These CPU times make sense, as both algorithms theoretically $\sim \frac{2}{3}m^3$ flops.