For this homework, you will be using *Matlab* (or the free and open-source version of *Matlab* called *Octave*). Include a clearly-labeled printout of the work you did in Matlab for each problem.

You can read through

`http://web.gps.caltech.edu/classes/ge11d/doc/matlab_Resource_Seminar.pdf`

for a short Matlab introduction. For additional documentation, see the Matlab website

`http://www.mathworks.com/help/matlab/index.html`

Quick Tips:

- You can supress Matlab output by adding a '`;`' to the end of your Matlab command (this is useful for Matlab commands executed in loops).
- You can save plots/figures from the figure's dialog by selecting 'Save as' and then choosing `png` or `jpg` in the file-types dropdown.
- You can include Matlab code in LaTeX by copying and pasting it between `\begin{verbatim}` ... `\end{verbatim}`.

1. Find the general solution to the system of equations corresponding to each augmented matrix.

   (a)
   $$\left[\begin{array}{ccccc|c} 1 & 11 & -22 & 0 & 3 & 1 \\ 5244 & 1542 & -3084 & 576 & 16308 & 2 \\ 108 & 1188 & -2376 & 0 & 324 & 3 \\ 264 & 465 & -930 & 25 & 817 & 4 \\ 2166 & 3729 & -7458 & 206 & 6704 & 5 \end{array}\right]$$

   (b)
   $$\left[\begin{array}{ccccc|c} 7 & 3 & 2 & 3 & 7 & 1 \\ 9 & 3 & 2 & 1 & 10 & 2 \\ 7 & 5 & 2 & 2 & 8 & 3 \\ 4 & 10 & 3 & 1 & 10 & 4 \\ 4 & 6 & 1 & 3 & 9 & 5 \end{array}\right]$$

2. Although the `rref` command will row-reduce for you, Matlab makes it easy to row reduce "by hand." If `A` is a matrix, the row operation $R_2 \mapsto R_2 - 6R_1$ can be performed with the command `A(2,:)=A(2,:)-6*A(1,:)` and other row operations can be performed similarly.

   (a) Consider the 3×3 matrix $A = \begin{bmatrix} 1 & 4 & 0 \\ 1 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. You only need to perform two row operations to fully reduce A. Call these two operations $R^{(1)}$ and $R^{(2)}$. Use Matlab to perform those row operations on A; also, perform the same row operations on B (effectively, un-row reducing B) and call the result B'.

   (b) Let $B_1 = R^{(1)}(B)$ and $B_2 = R^{(2)}(B)$ be the matrices obtained from B by applying the two row operations in isolation (in Matlab, you can name variables `B1` and `B2` if you like). What happens when you perform the matrix product B_1A? How about B_2A? And B_2B_1A? Finally, compute $B'A$. Explain your results.

   (c) Let $X = \begin{bmatrix} 2 & 1 & 3 \\ 2 & 1 & 3 \\ 3 & 2 & 4 \end{bmatrix}$. Find a matrix Y so that $YX = \text{rref}(X)$.

3. A loop in Matlab/Octave is written with the

```
for i=1:n,
    [loop content]
end
```

syntax.

For example, if we want the variable x to be an accumulated sum of squares from 1^2 to 15^2 (that is $x = 1^2 + 2^2 + \cdots + 15^2$), we could write

```
x=0;
for i=1:15,
    x = x + i^2;
end
```

After executing this code x will be the value 1240. The code works as follows: first we assign the value 0 to x. Then we enter the loop. In the first iteration, when i is 1, the new value of x will be the current value, 0, plus 1^2. The next time through the loop, i is 2 and so we assign x to be the current value, 1^2, plus 2^2, etc.. Once we have looped through with the value of i being 15, we stop.

If statements are written with the

```
if [var]==[val],
    [if content]
end
```

syntax (note the double equals).

For example, if we wanted x to be the sum of the squares of only the even numbers between 1 and 15, we might write:

```
x=0;
for i=1:15,
    if round(i/2) == i/2,
        x = x + i^2;
    end
end
```

Here, `round(i/2)==i/2` is true when `i/2` has no decimal places and so it is even. Thus, when that happens (and only when that happens) we add the value i^2 to x.

Question:

(a) The command `rand(3,1)` will generate a random 3×1 column vector. Find out what percentage of random triples of vectors $\{\vec{a}, \vec{b}, \vec{c}\}$ are linearly independent. Sample at least 1000 random triples to gather a statistic.

Consider your result and come up with an explanation of why you got the percentage you did.

Tips: Build a loop to test this 1000 times, but start out with a much smaller loop; use '`;`' inside your loop so you don't get a bunch of unhelpful information printed to the screen; and use Matlab's built-in commands to help you like `rref` to row reduce, `rank` to find the rank, or `rand(3,3)` to make a 3×3 matrix of random numbers.

(b) The command `round(rand(3,1))` will generate a random 3×1 column vector of just zeros and ones. Find out what percentage of random triples $\{\vec{a}, \vec{b}, \vec{c}\}$ of zero-one vectors are linearly independent. Sample at least 1000 random triples to gather a statistic.

Explain your result. Why is it different or the same as the previous part? (Hint: it may be useful to think of random zero-one vectors as lying on the vertices of the unit cube.)