I worked with John Pharo on this worksheet. It took about five hours.

Solving Large Linear Systems of Equations

(A)

The script I wrote to allow you to read in A and b is `load_data()` in `ws9.py`. These return lists of the $A_{(i)}$ and $b_{(i)}$, where $A_{(i)}$ and $b_{(i)}$ come from file pair i. You can then inspect $A_{(i)}$ and $b_{(i)}$ by printing them (though some of these have millions of elements) or printing some of their elements. You can also query the sizes of $A_{(i)}$ and $b_{(i)}$, to make sure they are the right shape (if $A_{(i)}$ is $n \times n$ then b should be length n). And for LSE i to be solvable, $A_{(i)}$ must not be singular; i.e., $\det(A_{(i)}) \neq 0$. This is because it must be that the solution $x = A^{-1}b$; if A^{-1} does not exist then there is no solution. My code in Part A of `ws9.py` print out the sizes of the $A_{(i)}$ and the $b_{(i)}$ and determines whether or not the $A_{(i)}$ are singular. The output follows.

```
A_(1) has shape (10, 10), while b_(1) has shape (10,).
A_(1) is nonsingular (has nonzero determinant). The LSE can be solved.
A_(2) has shape (100, 100), while b_(2) has shape (100,).
A_(2) is nonsingular (has nonzero determinant). The LSE can be solved.
A_(3) has shape (200, 200), while b_(3) has shape (200,).
A_(3) is nonsingular (has nonzero determinant). The LSE can be solved.
A_(4) has shape (1000, 1000), while b_(4) has shape (1000,).
A_(4) is nonsingular (has nonzero determinant). The LSE can be solved.
A_(5) has shape (2000, 2000), while b_(5) has shape (2000,).
A_(5) is nonsingular (has nonzero determinant). The LSE can be solved.
```

One can see that the shapes of the $A_{(i)}$ and the $b_{(i)}$ are compatible and none of the $A_{(i)}$ are singular, therefore these LSEs are all solvable.

(B)

I implemented my own Gaussian elimination LSE solver in `gauss_elim_solve()` of `ws9.py`. I tested it on the LSE $Ax = b$ with

$$A = \begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 8 \\ -11 \\ 3 \end{pmatrix},$$

which has solution

$$x = \begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}$$

The output of my driver code for this LSE is

```
Simple LSE:
A =
[[ 2  1 -1]
 [-3 -1  2]
 [-2  1  2]]
b =
[[  8]
 [-11]
 [ -3]]
```

```
Solution: x =
[[ 2.]
 [ 3.]
 [-1.]]
Time elapsed: 0.118 ms
```

My solver produces the correct output for this LSE.

I then ran my solver on the five provided LSEs; the timing information follows:

```
For the five given LSEs, the times required were:
#1 0.646 ms
#2 65.088 ms
#3 251.003 ms
#4 8214.902 ms
#5 40681.824 ms
```

(C)

The only straightforward NumPy LSE solver I could find was `numpy.linalg.solve()`. It works by LU decomposition. When using this solver on the five given LSEs, the following timing information results:

```
For the five given LSEs, the times required were:
#1 0.040 ms
#2 0.200 ms
#3 0.399 ms
#4 31.280 ms
#5 198.421 ms
```

One can see that for the larger LSEs (higher numbered), NumPy's solver works about two orders of magnitude faster than my solver.