18.06 Problem Set 1

Due Wednesday, 12 September 2007 at 4pm in the undergrad. math office.

Problem 0: from the book

(a) problem set 1.2, problem 8.

(b) problem set 2.1, problem 4

(c) problem set 2.2, problem 19

(d) problem set 2.3, problem 11

(e) problem set 2.3, problem 19

Problem 1: elimination matrices

(a) Solve the following equations by elimination and backsubstitution:

$$\begin{aligned} x + 3y - 4z &= 7 \\ 2x - 4y + 2z &= 0 \\ 3x + 19y + z &= 2 \end{aligned}$$

(b) Let the above system be denoted by $Ax = b$. Find the three elimination matrices E_{21}, E_{31}, and E_{32} that put A into upper triangular form $U = E_{32}E_{31}E_{21}A$. Compute $M = E_{32}E_{31}E_{21}$.

(c) Change a single number in the above equations so as to get *no* solutions. Then change another number to get *infinitely many* solutions.

Problem 2: Matlab, timings, etc.

Warmup: Solve the equations in problem 1 using Matlab. You can do this by the commands:

```
A = [1 3 -4; 2 -4 2; 3 19 1]
b = [7; 0; 2]
x = A \ b
```

Verify that you get the same answer as in 1(a), of course! Now, *time* how long Matlab takes to solve the equations, using the Matlab commands *tic* and *toc* (like a clock: tic toc tic toc):

```
tic; x = A \ b; toc
```

This prints the time in seconds for the command(s) between tic and toc. It should be quite a small number...in fact, this matrix is so small that you are mostly measuring the overhead of the Matlab interpreter rather than the solution of the equations per se.

(a) Now, let's solve larger systems. Much larger systems. To save the trouble of coming up with equations by hand, we'll let Matlab choose them at random using the *rand(m,n)* command, which creates a random $m \times n$ matrix:

```
A = rand(100,100);
b = rand(100,1);
tic; x = A \ b; toc
```

Notice the semicolons (;) after the commands: this suppresses the output, which is useful if you don't want to print out the 100×100 matrix A.

The above code was for 100×100. Now try doubling this to 200. Then to 400. Then to 800. Then to 1600. By what factor, on average, does the computation time increase each time you double the number of rows and columns? (You may want to repeat each computation a few times to make sure you get consistent timing results.)

(b) Random linear equations like the ones above are very unlikely to have no solution. Construct a 2×2 or 3×3 set of equations by hand that (you think) have no solution. What does Matlab do when you give it `A \ b` with this A?