18.06 Problem Set 2

Due Wednesday, 19 September 2007 at 4 pm in 2-106.

Problem 1: (a) Do problem 12 from section 2.4 (P67) in your book.
(b) Do problem 36 from section 2.4 (P70) in your book.

Problem 2: Do problem 19 from section 2.5 (P80) in your book.

Problem 3: Do problem 9 from section 2.6 (P92) in your book.

Problem 4: Do problem 24 from section 2.6 (P94) in your book.

Problem 5: Show that the product of two lower triangular matrices is a lower triangular matrix. What are the diagonal entries of this product? (This tells us why the product of the elimination matrices, or their inverses, is lower triangular, since the elimination matrices are lower triangular.)

Problem 6: Let $A = \begin{pmatrix} 1 & 3 & -1 \\ 2 & 5 & 1 \\ 3 & 4 & 2 \end{pmatrix}$.
(a) Use Gauss-Jordan elimination to calculate A^{-1}.
(b) Find the LU decomposition of A.

Problem 7: True of false (give a reason if true; give a counterexample if false):
(a) The block matrix $\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}$ is always symmetric.
(b) If A and B are $n \times n$ symmetric matrices, so is AB.
(c) If A is symmetric and $I + A$ is invertible, then $(I + A)^{-1}$ is symmetric.
(d) If A is n by n symmetric matrix, then $\mathbf{x} \cdot A^2\mathbf{y} = (A\mathbf{x}) \cdot (A\mathbf{y})$ for any $n \times 1$ vectors $\mathbf{x}$ and $\mathbf{y}$.
(e) Any $n \times n$ matrix can be written as a sum of a symmetric matrix and an anti-symmetric matrix. (*anti-symmetric means* $A^T = -A$.)

Problem 8: (a) A matrix A is *idempotent* if $A^2 = A$. Show that $A = \begin{pmatrix} 2 & -2 & -4 \\ -1 & 3 & 4 \\ 1 & -2 & -3 \end{pmatrix}$ is idempotent.

(b) A matrix A is *nilpotent of class* p if $A^p = 0$ but $A^{p-1} \neq 0$. Show that $A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}$ is nilpotent.

Problem 9: Construct a random 1000×1000 matrix A in Matlab, and a random right hand side $\mathbf{b}$:

```
>> A = rand(1000,1000);
>> b = rand(1000,1);
```

We can solve $Ax = \mathbf{b}$ in two ways: by

```
>> x =A \ b ;
```

which (inside Matlab) uses elimination (via an LU decomposition) or by

```
>> x= inv(A) * b;
```

which explicitly computes the inverse of the matrix A.
(a) try both methods of computing $\mathbf{x}$, timing them with the tic and toc commands as in problem set 1. Which is faster? Is this consistent with how hard it is to apply the two methods by hand, using the techniques we learned in class?
(b) In a perfect world, these would give the same answer, and in both cases $A\mathbf{x}$ would equal $\mathbf{b}$, but this is not the case in reality. Computers store only about 15 decimal places of each number and (almost) every arithmetic operation involves some roundoff error. Compute the error $|A\mathbf{x} - \mathbf{b}|/|\mathbf{b}|$, via

```
>> norm(A*x - b) / norm(b)
```

for both methods of computing $\mathbf{x}$ above. Repeat for a few random matrices A. Which method seems to be more accurate in the face of roundoff error? Why do you think that might be?

Problem 10: Matlab has a command *inv*(A) to compute the inverse of a matrix A. It also has a command *pinv*(A) to compute something rather mysterious, a "pseudo-inverse" of A. Without poring over the Matlab documentation, let's try to figure out what *pinv* does.

(a) Verify that *inv* and *pinv* do the same thing (up to roundoff error, $14-15$ decimal places) if A is invertible. e.g. try a couple random 10×10 matrices.
(b) Consider a singular matrix

```
>> A = [1 2; 2 4]
```

The equation $A\mathbf{x} = \mathbf{b}$ has no solution for most right-hand sides $\mathbf{b}$; pick one such right-hand-side $\mathbf{b}$ (i.e. with no solution $\mathbf{x}$). However $pinv(A)$ gives something(!), although we don't know yet what it means, and we can compute

```
>> x= pinv(A) * b;
```

Let's try to figure out what this x could mean. If we compute $A * \mathbf{x}$, we won't get $\mathbf{b}$ (we can't!), but we get something else, call it bx:

```
>> bx = A * x;
```

For comparison, let's compute what we get when we try 100 random $\mathbf{x}$ values, call it br:

```
>> br = A * rand(2, 100);
```

To try to understand this, plot $\mathbf{b}$, bx, and br, using the command:

```
>>plot(b(1),b(2),'rx', bx(1),bx(2),'bo', br(1,:),br(2,:),'k.')
```

which puts a red "x" where $\mathbf{b}$ is, a blue "o" where bx is, and a black dot at each br. You will need to type:

```
>> axis equal
```

to force Matlab to plot the x and y axes with equal scaling. Now:
i) The br values have a simple pattern in the plot. Why?
ii) Infer the rule that Matlab used to pick $\mathbf{x} = pinv(A) * \mathbf{b}$. (You don't have to figure out how to compute $pinv(A)$ or x. Just describe what distinguishes bx.)
We will return to this kind of singular problem later in the term.

NOTE: For Matlab problems, you should print out and turn in sufficient information for the grader to follow what you did. Definitely any plots, and perhaps a transcript of the command window or an excerpt thereof.