HW 2
Due in class on Monday, Feb 7

Remember that you may (and should!) talk about the problems amongst yourselves, or discuss them with me or the TA, providing attribution for any good ideas you might get – but your final write-up should be your own.

Note that your codes should *not* use the MATLAB command `inv` to explicitly form matrix inverses. In general, you should use the backslash operator instead. You will not receive full credit for code that uses `inv`!

1: Costly computations. The following MATLAB code does far more work than necessary. For each of the four calculations, briefly explain why the existing code is inefficient and write an efficient alternative.

```
% -- Set up a large matrix and a couple vectors.
A = rand(2000);
x = rand(2000,1);
y = rand(2000,1);
z = rand(2000,1);

% -- Expressions to rewrite

% a)
u1 = x*y'*z;

% b)
u2 = x'*(A'*A)*x;

% c)  (hint: look up .* in MATLAB)
D = diag(x);
u3 = D*y;

% d)
u4 = (D*A)\y;
```

2: Setting up a linear system. You may recall from a discrete math class or an algorithms class that

$$\sum_{i=1}^{n} 1 = n$$
$$\sum_{i=1}^{n} i = \frac{1}{2}n^2 + \frac{1}{2}n$$
$$\sum_{i=1}^{n} i^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n.$$

In general, we have

$$\sum_{i=1}^{n} i^d = \sum_{j=1}^{d+1} a_j n^j,$$

and we can find the coefficients by writing a linear system[1]. For example, to find the formula for summing i, we could write

$$\sum_{i=1}^{1} i = 1^2 a_2 + 1 a_1$$
$$\sum_{i=1}^{2} i = 2^2 a_2 + 2 a_1.$$

Using this idea, write a short MATLAB program that solves a linear system to find the polynomials for $\sum_{i=1}^{n} i^d$ for $d \in \{0, 1, 2, 3, 4\}$. In your write-up, you should include the MATLAB code and the coefficients for $d = 3$ and $d = 4$.

[1] Actually, there are many different ways to write this problem, and they lead to many different types of linear systems. The matrix that I describe here is called a *Vandermonde* matrix, and it actually gets rather nasty for large values of d. We will see more about Vandermonde matrices (and "nicer" versions of the same problem) when we talk about interpolation later in the class.

3: A tiny Taylor expansion. Given fixed matrices A and E and a fixed vector b, define $x(t)$ as the solution to the linear system

$$(A + tE)x(t) = b.$$

Now define the first-order Taylor expansion[2]

$$\bar{x}(t) = x(0) + x'(0)t.$$

Complete the following MATLAB function that evaluates $x(t)$ and $\bar{x}(t)$

```
% function [x,xbar] = hw2taylor(t,A,E,b)
%
% Outputs:
% - x is the solution to (A+tE) x(t) = b
% - xbar is the first-order Taylor approximation to x
%   (expanding about t = 0).

function [x,xbar] = hw2taylor(t,A,E,b)
```

To show the correctness of your program, generate a 3-by-3 test matrices A and E and a right hand side b using MATLAB's `rand` command. For this fixed test problem, show a log-log plot of the absolute error $e(t) = \|\bar{x}(t) - x(t)\|_2$ versus t for $t = 2^{-1}, 2^{-2}, 2^{-3}, \ldots, 2^{-20}$. As before, make sure the plot is clearly labeled! Your plot should look linear for small values of t. Estimate the slope using the formula

$$\text{slope} \approx \frac{\log e(2^{-20}) - \log e(2^{-19})}{\log 2^{-20} - \log 2^{-19}} = \frac{\log(e(2^{-20})/e(2^{-19}))}{\log(1/2)}$$

What is the slope of the line? Why?

[2] This expansion appears in the derivation of sensitivity and error estimates for linear systems. So if you get confused about how to derive the formula, note that it actually appears in Heath.