Math 477/577 — Computer Assignment 4, due Oct.26, 2006

1. Take $m = 50$, $n = 12$. Using MATLAB's `linspace`, define t to be the m-vector corresponding to linearly spaced grid points from 0 to 1. using MATLAB's `vander` and `fliplr`, define A to be the $m \times n$ matrix associated with least squares fitting on this grid by a polynomial of degree $n - 1$. Take $\boldsymbol{b}$ to be the function $\cos(4t)$ evaluated on the grid. Now, calculate and print (to sixteen-digit precision) the least squares coefficient vector $\boldsymbol{x}$ by six methods:

   (a) Formation and solution of the normal equations, using MATLAB's \,

   (b) QR factorization computed using the `mgs` routine from Computer Assignment 3,

   (c) QR factorization computed via the `house` routine from Computer Assignment 3,

   (d) QR factorization computed by MATLAB's `qr`,

   (e) `x = A \ b` in MATLAB,

   (f) SVD, using MATLAB's `svd`.

   (g) The calculations above will produce six lists of twelve coefficients. In each list, shade with red pen the digits that appear to be wrong (affected by rounding error). Comment on what differences you observe. Do the normal equations exhibit instability? You do not have to explain your observations.

2. The usual Gaussian elimination algorithm involves a triply nested loop. The version given in the classnotes (as LU factorization) involves two explicit `for`-loops, and the third loop is implicit in the vectors $U(j, k : m)$ and $U(k, k : m)$. Rewrite this algorithm with just one explicit `for`-loop indexed by k. Inside this loop, U will be updated at each step by a certain rank-one outer product. This "outer product" form of Gaussian elimination may be a better starting point than the LU factorization algorithm from the classnotes if one wants to optimize computer performance. Implement both the LU factorization algorithm from the classnotes and the "outer product" version discussed here, and test and time them both with the matrices `A = gallery('lehmer',N)` for `N=100*[1:10]`.