Homework 7: QR Factorization and Least Square Problems
Due: Friday, October 29, 2010

1. Read Health 3.1–3.4, 3.5.3, 3.5.4

2. New Routines. Write routines for

(a) The classic Gram-Schmidt (CGS) construction of the QR factorization of a matrix $A_{m \times n} = Q_{m \times n}R_{n \times n}$

```c
int QRfactorCGS(double **A, int m, int n, double **Q, double **R);
```

Return 1 if successful, return 0 if the factorization fails because the columns are not linearly independent.

(b) The modified Gram-Schmidt (MGS) construction of the QR factorization of a matrix $A_{m \times n} = Q_{m \times n}R_{n \times n}$

```c
int QRfactorMGS(double **A, int m, int n, double **Q, double **R);
```

Same return value as in (a). You may do column pivoting if you wish, but it’s not required.

(c) The solution of $A\vec{x} = \vec{b}$ given by the QR factorization, where $\vec{x} \in \mathbb{R}^n$ and $\vec{b} \in \mathbb{R}^m$

```c
int QRsolve(double **Q, double **R, int m, int n, double *b, double *x);
```

Store the solution in $x$.

(d) Solution of the Normal equation $A^T A \vec{x} = A^T \vec{b}$ using LU factorization

```c
int Normalsolve(double **A, int m, int n, double *b, double *x);
```

You may either write a Cholesky factorization routine or use your earlier LU routines.

3. Testing the routines, $\tau$-matrix returns. Let $A$ be an $n \times n$ symmetric matrix structured as follows:

- First we put 1 at each diagonal position and a random number from the uniform distribution on [-1,1] at each off-diagonal position, while maintaining the symmetry $A^T = A$.
- Then we replace each off-diagonal entry with $|a_{i,j}| > \tau$ by zero, where $\tau$ is a parameter.

For $n = 100, 105, 110, \ldots, 1000$. Set $\tau = 0.1, 0.5, 0.9$.

(a) Calculate the solution of $A\vec{x} = \vec{b}$, where the entries in $b$ are all one’s, using your earlier LU routines. Consider this solution to be the “exact” answer, $\vec{x}^*$.

(b) Find the max-norm error, $e_n = \|\vec{x} - \vec{x}^*\|_\infty$ between the exact solution and the solution calculated using each of the following methods:

- QRfactorCGS()/QRsolve()
- QRfactorMGS()/QRsolve()
- Normalsolve()

Plot the error curves for each vs $n$ together on one graph and conclude which is the best/worst methods for square systems.
4. A least squares problem: polynomial fitting. We will fit a polynomial of degree \( n - 1 \)

\[ p_{n-1}(t) = x_1 + x_2 t + x_3 t^3 + \cdots + x_n t^{n-1} \]

to \( m \) data points \((t_i, y_i)\), \( m > n \). We choose \( t_i = (i - 1)/(m - 1) \), \( i = 1, \ldots, m \), so that the data points are equally spaced on the interval \([0,1]\). Warning: This problem is ill-conditioned.

We will generate the corresponding values \( y_i \) by choosing values for the \( x_j \), say \( x_j = 1 \), \( j = 1, \ldots, n \), and evaluating the resulting polynomial to obtain \( y_i = p_{n-1}(t_i) \), \( i = 1, \ldots, m \). We could see whether we can recover the \( x_j \) that we used to generate the \( y_i \), but to make it more interesting, we first randomly perturb the \( y_i \) values to simulate the data error typical of least squares problems. Specifically, we take

\[ y_i = y_i + (2u_i - 1)\epsilon, \quad i = 1, \ldots, m \]

where each \( u_i \) is a random number uniformly distributed on the interval \([0, 1]\), and \( \epsilon = 10^{-10} \). Set \( m = 21 \) and \( n = 12 \).

Having generated the data set \((t_i, y_i)\) as just outlined, we will now compare the three methods used in problem 3(b). For which method is the solution more sensitive to the perturbations we introduced into the data? Which method comes closer to recovering \( x \) that we used to generate the data?