Assignment 2.

Given February 25, due March 4.

Objectives: To explore the conditioning of linear systems; to solve a least squares problem in several different ways.

I *strongly recommend* MATLAB for this assignment. It can be done without MATLAB, but you will need software for generating a random matrix and orthogonalizing its columns, also software to do Gaussian elimination, matrix inversion, QR decomposition and singular value decomposition. Such software is available over netlib in LAPACK. To see what is available, you can type `mail netlib@ornl.gov` and give the message `send index for lapack`.

(1) Recall that the condition number of a matrix is the ratio of largest to smallest *singular value*, and that the singular value decomposition of an m by n matrix A is a factorization of the form $A = U\Sigma V^T$, where U is an m by m orthgonal matrix, V is an n by n orthogonal matrix, and Σ is an m by n matrix with the singular values along its main diagonal.

Write a routine to generate an n by n matrix with a given condition number,

```
function A = matgen(n, condno)
```

by generating two random orthogonal matrices U and V and a diagonal matrix Σ with $\sigma_{ii} = condno^{-(i-1)/(n-1)}$. Note that the largest singular value of $U\Sigma V^T$ is 1 and the smallest is $condno^{-1}$, hence the ratio is *condno*. You can generate a random orthogonal matrix in MATLAB by first generating a random matrix, `Mat = randn(n,n)`, and then computing its QR decomposition, `[Q,R] = qr(Mat)`. The matrix Q is then a random orthogonal matrix. You can check the condition number of the matrix you generate by using the function `cond` in MATLAB. Turn in a listing of your code.

(2) For $condno = (1, 10^4, 10^8, 10^{12}, 10^{16})$, use your routine to generate a random matrix A with condition number *condno*. Also generate a random vector `x_true` of length n, and compute the product `b = A*x_true`.

Solve $Ax = b$ using Gaussian elimination with partial pivoting. This can be done in MATLAB by typing $x = A\backslash b$. Determine the relative error in your computed solution $\|x - \texttt{x_true}\|/\|\texttt{x_true}\|$. Explain how this is related to the condition number of A. Compute the relative residual $\|b - Ax\|/(\|A\|\|x\|)$. Does the algorithm for solving $Ax = b$ appear to be *backward stable*; that is, is the computed solution the exact solution to a nearby problem?

Solve $Ax = b$ by inverting A: `Ainv = inv(A)` , and then multiplying b by A^{-1}: `x = Ainv*b`. Again look at relative errors and residuals. Does this algorithm appear to be *backward stable*?

Turn in a table showing the relative errors and residuals for each of the algorithms and each of the condition numbers tested, along with a brief explanation of the results.

(3) Find the polynomial of degree 11 that best fits the function $b(t) = cos(4t)$ at 50 equally spaced points t between 0 and 1. Set up the matrix A and right-hand side vector b, and determine the polynomial coefficients in three different ways:

(a) By using the MATLAB command `x = A\b` (which uses a Householder QR factorization routine).

(b) By using the singular value decomposition, `[U,Sigma,V]=svd(A)` .

(c) By solving the normal equations $A^T Ax = A^T b$. This can be done in MATLAB by typing `x = (A'*A)\(A'*b)`.

Print the results to 16 digits (`format long e`) and comment on the differences you see.