Computer projects for Numerical Linear Algebra, MA 660.

Computer projects are designed to be done in MATLAB, a matrix-oriented computer software installed in the computer lab, room 112 in Classroom Building (CB). MATLAB projects can be done with minimum coding and maximum convenience. However, you can use any other computer language or software, if you prefer.

The MATLAB package includes an on-line help. In addition, many MATLAB manuals are available on-line from various web sites. In particular, check out the official MathWorks (the producer of MATLAB) web page:

www.mathworks.com/access/helpdesk/help/techdoc/math_anal/math_anal.shtml

The projects must be submitted in a "presentable" form. Include a print-out of the MATLAB (or other) code, a print-out of the computer output (including clear readable graphics), and a one-page report (hand-written or typed on a computer).

Project 1. Do Experiment 2 on pages 65–66 of the textbook. The purpose is to compare the two Gram-Schmidt orthogonalization methods (the classical one and the modified one). First, you generate random matrices, then obtain random orthogonal matrices by using the **qr** procedure, then build a diagonal matrix whose entries are consecutive negative powers of two. Then you get a matrix A, which is badly ill-conditioned (by the way, find its condition number by **cond(A)**). Then you apply the classical and modified Gram-Schmidt methods to the matrix A by calling the procedures **clgs** and **mgs** (note: they are not included in MATLAB, but their codes **clgs.m** and **mgs.m** are available from the instructor's web page: you need to download and save them into your MATLAB working directory). Then you plot the resulting diagonal matrices **RC** and **RM** on a logarithmic scale – you get a graph similar to Figure 9.1 on page 66. Write a report, explain the results.

Project 2. Exercise 9.3 on page 68 of the textbook. The purpose is to see how a limited number of singular values and singular vectors of a large matrix can be used to approximately reconstruct the entire matrix.

Project 3. Explore the stability of the least squares algorithms (Lecture 19 in the textbook). First, generate a badly ill-conditioned matrix A of size $m \times n$ with some $50 \le m \le 100$ and $10 \le n \le 20$. Suggestions: generate two random orthogonal matrices, U of size $m \times m$ and V of size $n \times n$ (see Project 1), and a diagonal matrix D of size $m \times n$ whose n diagonal entries are consecutive negative powers of four or five (see Project 1). Then compute $A = UDV$. Next, generate a random n-vector y and compute $b = Ay$. Now solve the overdetermined linear system $Ax = b$ by several methods as in Lecture 19 (see below). After each solution compute and print the value of **norm(x-y)/norm(y)** as a measure of accuracy of the solution (the smaller the better). Make conclusions.

The methods you need to test are listed here:

1. Normal equations (the most popular but the least accurate)
2. Classical Gram-Schmidt
3. Modified Gram-Schmidt
4. QR factorization by Householder reflections
5. MATLAB **x=A\b** function (QR factorization with column pivoting)
6. Reduced SVD decomposition.

The last two are, generally, the most numerically accurate.

Note: for a better visualization of the results, you may use a fixed vector y (for example, set all its entries to $1/3 = 0.3333333333333\ldots$), then print the vector x found by each algorithm (use double-precision format, i.e. **format long**) and see which vectors look more like y.