MATH 225 Linear Algebra and Differential Equations

Fall 2007 MATLAB Homework 2

Due date: Friday, November 16, 17:30

Scientific computing is the field of study concerned with constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific and engineering problems. As you may easily guess, currently engineers in industry or academia almost always work with computers to solve scientific problems. Compared to analytic treatments (i.e., finding exact solutions to problems, which is possible only in some exceptional situations), there are two difficulties involved in scientific computing. These are obtaining accurate enough solutions with finite-precision computers and obtaining these solutions fast enough (computational efficiency). To simultaneously satisfy these contradicting requirements is an exciting work! See for instance http://en.wikipedia.org/wiki/Scientific_computing if you want to learn more on this topic.

In the first two questions of this homework, we will touch the accuracy and efficiency issues for solving linear system of equations. Then, the third problem, even though it is simple, will help you to get an idea of mathematical modeling.

1. An $n \times n$ *Hilbert* matrix $\boldsymbol{H}$ has entries $h_{ij} = 1/(i+j-1)$, so it has the form

$$\begin{bmatrix} 1 & 1/2 & 1/3 & \cdots \\ 1/2 & 1/3 & 1/4 & \cdots \\ 1/3 & 1/4 & 1/5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

It is known that Hilbert matrices are nonsingular.

(a) Using Matlab command `rref`, find the reduced row echelon form of Hilbert matrices of order 8, 12, and 15. Do you obtain what you expect? Comment on the results.

(b) Now let's see whether Matlab produces accurate solutions with linear systems whose coefficient matrices are Hilbert matrices. For this purpose, for *n*=8,12, and 15, generate the *n*-vector $\boldsymbol{b} = \boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{x}$ is the *n*-vector with all of its components are equal to 1. Now solve these systems using the Matlab statement `xComputed=H\b,` where `xComputed` is the computed *x*-vector. Do you see a discrepancy between the correct solution $\boldsymbol{x}$ and the computed solution? Why do you think there is an inconsistency here?

Hints: You can generate Hilbert matrices using `hilb` command. Use the `ones` command to generate the vector of ones. The "/" operator solves a linear system using Gaussian elimination with partial pivoting. See Matlab documentation or http://www.netlib.org/lapack/ for more information.

2. The inverse of a matrix satisfies $\boldsymbol{A}\boldsymbol{A}^{-1} = I$. From the definition of matrix-matrix multiplication, we see that *j*th column of $\boldsymbol{A}^{-1}$ is the vector such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the *j*th column of the identity matrix (i.e., $\boldsymbol{e}_2 = [0 \quad 1 \quad 0 \quad \cdots \quad 0]^T$). By solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}_j$ for *j*=1,…,*n*, the columns of $\boldsymbol{A}^{-1}$ can be produced one at a time. A pseudo-code to accomplish this is given as follows:

$\boldsymbol{Ainv}$ =[]; % produce an 'empty' matrix
for *j* = 1 to *n*
  find solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}_j$
  $\boldsymbol{Ainv}$ =[$\boldsymbol{Ainv}$ $\boldsymbol{x}$] % append the newly generated column
end

Implement this pseudo-code in a function `invByCol` that takes only the matrix $\boldsymbol{A}$ as input and returns $\boldsymbol{A}^{-1}$. Test your function using the matrix

$$\boldsymbol{A} = \begin{bmatrix} 4 & 3 & 2 \\ 5 & 6 & 3 \\ 3 & 5 & 2 \end{bmatrix}.$$

To compare the performances of `invByCol` and Matlab's built-in function `inv`, write another function called `compareInvMethods(n)`, which generates a random matrix of order *n* (use `rand` function), then calls `invByCol` and `inv` functions to invert the matrix, and uses `tic` and `toc` functions of Matlab to compare the computational times of `invByCol` and `inv`. Try with random matrices of size 5, 10, and 100. Which function performs better? Comment on the results.

Hints: You may find the `length, eye,` and `disp` commands useful.

3. One application of linear systems is curve fitting as detailed in your text book. For this problem, write a `parabola` function to automatically setup and solve the system of equations for a parabola defined by $y = c_1x^2 + c_2x + c_3$. The function should take two input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, each of length three, that defines three points through which the parabola passes. It should return the vector of the coefficients c_i. Find the equation of the parabolas passing through each of the following set of points:

(a) (-2, -1), (1,-1), (2, 0)
(b) (-2, -2), (-1, 1), (2, -1)

Then, use c_i and the minimum and maximum of *x*-points to plot the parabola along with the original points (for example, with a red '*' on the plot) to verify that the equation of the parabola has been obtained correctly.

Hints: You will find the following Matlab commands useful:

`x.^2:` The square of the x vector
`plot(x,y):` Plot x versus y. Enter `help plot` for more information.
`hold on:` Holds the current plot so that subsequent graphing commands add to the existing graph.