CS 370 Fall 2008: Assignment 4

Instructor: Professor Keith Geddes

Lectures: MWF 3:30-4:20 MC 2017

Web Site: UW-ACE

Due: Tue Nov 18, 2008, 5:00 pm, in the Assignment Boxes, 3rd Floor MC

1. **(LU Factorization)**

   Consider the following linear system of equations:

$$\begin{aligned} 1.0\,x_1 + 1.0\,x_2 + 3.0\,x_3 &= 4.0 \\ 2.0\,x_1 + 1.0\,x_2 - 1.0\,x_3 + 1.0\,x_4 &= 1.0 \\ -1.0\,x_1 + 2.0\,x_2 + 3.0\,x_3 - 1.0\,x_4 &= 4.0 \\ 3.0\,x_1 - 1.0\,x_2 - 1.0\,x_3 + 2.0\,x_4 &= -3.0 \end{aligned}$$

   Use the floating point number system $F(10, 4, -10, 10)$ (i.e., base 10 carrying 4 significant digits) in this question.

   (a) What is the coefficient matrix A and the right-hand-side vector b so that the linear system is $A\,x = b$?

   (b) By hand calculation, carry out Gaussian elimination *with row pivoting* (sometimes called *partial pivoting*) on the coefficient matrix A to compute the LU factorization of A. In other words, find a permutation matrix P, unit lower triangular matrix L, and upper triangular matrix U such that $PA = L\,U$.

   (c) Solve the linear system (by hand) using the LU factorization determined in part (b).

   (d) Solve the new linear system $A\,x = c$ where the coefficient matrix A is the same as above but with the new right-hand-side vector

$$c = \begin{bmatrix} -1.8 \\ -5.1 \\ 6.3 \\ -8.4 \end{bmatrix}. \qquad (1)$$

2. **(Matlab function myInv)**

   Write a Matlab function `B = myInv(A)` which takes as input a matrix `A`. The output from `myInv` depends on the properties of the input matrix, as follows.

   - If `A` is square and not numerically singular then the output `B` is the inverse of `A` computed by Gaussian elimination with row pivoting.
   - If `A` is not square then the output `B` is a 1×2 matrix giving the number of rows and columns of `A`.
   - If `A` is square but numerically singular then the output `B` is a scalar 0. For deciding whether a square input matrix is numerically singular, use the Matlab command `cond` with any choice of norm.

   For the main computation (when the output `B` is the inverse of `A`), compute `B` by solving the matrix equation $AB = I$ for appropriately sized identity matrix I. To solve this system of equations, use the Matlab command `[L,U,P] = lu(A)` and then use Matlab matrix operations with matrices `P, I, L, U` to compute `B`. You are not allowed to use the Matlab command `inv` for this question.

   Testing your function:

   Demonstrate that your function works for the following matrices defined in Matlab: `ones(2,4)`, `zeros(3,3)`, and `hilb(k) for k=7:2:13` .

   To demonstrate each of these six cases, you should do the following steps in a Matlab session:

   - Declare a diary file (see `help diary`).
   - Initalize `A` to one of the matrices to be demonstrated.
   - Call `myInv`.
   - Set `diary off`.

   For your submission, print a listing of your function `myInv`, and print a copy of your diary file that shows the input and result from `myInv` for each of the six test cases.

3. **(Efficient matrix-vector operations)**

   Let A be an $n \times n$ nonsingular matrix, and let F and G be $n \times p$ matrices, where $1 \leq p \ll n$. You are asked to compute the matrix $W = FG^T A^{-1}$. One possible algorithm is the following:

   - Compute $V = FG^T$.
   - Noting that $W = VA^{-1}$, we can write $WA = V$, and by transposing both sides, we have $A^T W^T = V^T$.
   - Finally, compute W by the Matlab command: `W = (A'\V')'`.

   The Matlab command `A\b` (where `b` is a single vector) requires $\frac{2}{3}n^3 + O(n^2)$ flops.

   (a) Show that this algorithm requires $\frac{8}{3}n^3 + 2pn^2 + O(n^2)$ flops.

   (b) Suggest a better method that requires only $\frac{2}{3}n^3 + 4pn^2 + O(n^2)$ flops.

4. **(Condition Numbers and Errors)**

   For an $n \times n$ linear system of equations $Ax = b$, the sensitivity of the computed solution $\hat{x}$ to changes in the input (or to roundoff errors during computation) is measured by the condition number $\kappa(A)$. Specifically, if x_{exact} denotes the exact solution and $\hat{x}$ denotes the solution computed by Gaussian elimination with partial pivoting then, as stated in the Course Notes, the relative error is bounded as follows:

   $$\frac{\|x_{exact} - \hat{x}\|}{\|\hat{x}\|} \leq \kappa(A)\, \epsilon_{machine}\,. \tag{2}$$

   Any valid norm may be used; for this question use the infinity norm.

   Write a Matlab script to test the accuracy of the solutions to the following linear systems:

   $$H_n\, x = b_n \tag{3}$$

   for `n = 2:2:12` , where H_n denotes the Hilbert matrix of order n (defined in Matlab by the command `hilb(n)`), and the order-n vector b_n is defined by

   $$b_n[i] = \frac{i}{i^2 + 1}\,, \quad i = 1, ..., n\,. \tag{4}$$

   For each n, compute $\hat{x}$ using the Matlab operator \ . Since the inverse matrix H_n^{-1} is known (see the Matlab command `invhilb(n)`), compute x_{exact} and the actual relative error:

   $$err_n = \frac{\|x_{exact} - \hat{x}\|}{\|\hat{x}\|}\,. \tag{5}$$

   Compare the actual relative error with the error bound stated above in terms of the condition number $\kappa(H_n)$ which can be obtained via the Matlab command `cond`. Present the results in a table showing, for each n, err_n and the corresponding theoretical error bound.

5. **(Google Page Rank)**

The objective of this question is to develop Matlab code that computes the Page Rank of a set of web pages based on the network adjacency graph. The adjacency graph is represented by the *connectivity matrix* which is a sparse matrix G where

$$G_{ij} = \begin{cases} 1 & \text{if } \exists \text{ a link from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

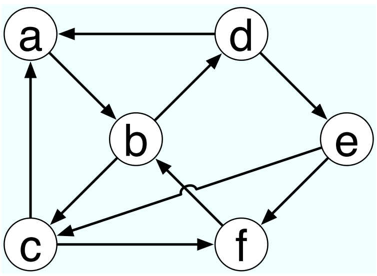
(a) Create a Matlab function with the calling prototype

```
function [p, iters] = PageRank(G, alpha)
```

which finds the steady-state solution (eigenvector for $\lambda = 1$) of the Page Rank problem using the iterative method discussed in class. The output `p` is a vector containing the node scores, and `iters` is the number of iterations the method took to converge.

Your method should take advantage of the sparsity of G. That is, at no time should your function create a full matrix of size $R \times R$, where R is the number of nodes. (See the discussion on "Practicalities" in Section 5.6 of the Course Notes.) Terminate the iteration once the solution is found to within a tolerance of 10^{-8} (i.e., none of the scores changes by more than the tolerance).

(b) Given the small web shown in the figure below, edit the supplied Matlab script `Google.m` to create the corresponding sparse matrix G as well as the node labels U. The node labels can be stored in a cell array, indexed using curly brackets. For example, you can set the second cell to "b" using `U{2} = 'b'`. Run your `PageRank` function on the network with an α-value of 0.85. Print out and hand in a bar plot showing the node scores (see the command `bar`). Which node has the highest score? Lowest score? Write the scores on your plot printout.



(c) An adjacency graph represented by connectivity matrix G and a list of URLs U is given in the file `cs.mat`. (Download `a4_skeleton_code.zip` and unzip to obtain the two supplied files `cs.mat` and `Google.m`.) Run the `Google.m` script on the network stored in `cs.mat`. Try different values of α in the range 0 to 1. Answer the following questions.

i. What do you notice about the relationship between α and the number of iterations?

ii. The list of the top-10 web pages changes dramatically with extreme values of α (e.g., 0 versus 1). What are the relative advantages and disadvantages of choosing α close to 0 or α close to 1?