Math 3607: Homework 5

Due: 10:00PM, Tuesday, October 5, 2021

TOTAL: 30 points

- Problems marked with ✏ are to be done by hand; those marked with 🖳 are to be solved using a computer.

- **Important note.** Do not use *Symbolic Math Toolbox*. Any work done using `sym` or `syms` will receive NO credit.

- **Another important note.** When asked write a MATLAB function, write one at the end of your live script.

1. (Improved triangular substitutions; adapted from **FNC** 2.3.5) 🖳 If $B \in \mathbb{R}^{n \times p}$ has columns $\mathbf{b}_1, \ldots, \mathbf{b}_p$, then we can pose p linear systems at once by writing $AX = B$, where $X \in \mathbb{R}^{n \times p}$ whose jth column $\mathbf{x}_j$ solves $A\mathbf{x}_j = \mathbf{b}_j$ for $j = 1, \ldots, p$:

$$A \underbrace{\left[\begin{array}{c|c|c|c} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \end{array}\right]}_{=X} = \underbrace{\left[\begin{array}{c|c|c|c} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_p \end{array}\right]}_{=B}.$$

   (a) Modify `backsub.m` and `forelim.m` from lecture[1] so that they solve the case where the second input is an $n \times p$ matrix, for $p \geq 1$. Include the programs at the end of your live script.

   (b) If $AX = I$, then $X = A^{-1}$. Use this fact to write a MATLAB function `ltinverse` that uses your modified `forelim` to compute the inverse of a lower triangular matrix. Include the program at the end of your live script. Then test your function using the following matrices, that is, compare your numerical solutions against the given exact solutions.

$$L_1 = \begin{bmatrix} 2 & 0 & 0 \\ 8 & -7 & 0 \\ 4 & 9 & -27 \end{bmatrix}, \qquad L_1^{-1} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ \frac{4}{7} & -\frac{1}{7} & 0 \\ \frac{50}{189} & -\frac{1}{21} & -\frac{1}{27} \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 \\ 0 & \frac{1}{3} & 1 & 0 \\ 0 & 0 & \frac{1}{3} & 1 \end{bmatrix}, \qquad L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ \frac{1}{9} & -\frac{1}{3} & 1 & 0 \\ -\frac{1}{27} & \frac{1}{9} & -\frac{1}{3} & 1 \end{bmatrix}$$

[1]Lecture 12 (09/24/21, F): Square Linear Systems (Introduction)

2. (Triangular substitution and stability; **FNC** 2.3.6) Consider the following linear system $A\mathbf{x} = \mathbf{b}$:

$$\underbrace{\begin{bmatrix} 1 & -1 & 0 & \alpha - \beta & \beta \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{b}}.$$

(a) ✏️ Show that $\mathbf{x} = (1, 1, 1, 1, 1)^{\mathrm{T}}$ is the solution for any α and β.

(b) 🖥️ ~~Using MATLAB,~~ Using the `backsub` function written for the previous problem, solve the system with $\alpha = 0.1$ and $\beta = 10, 100, \ldots, 10^{12}$, making a table of the values of β and $|x_1 - 1|$. Write down your observation.

3. (Vectorizing `mylu.m`; **FNC** 2.4.7) Below is an instructional version of LU factorization code presented in lecture.

```
function [L,U] = mylu(A)
% MYLU    LU factorization (demo only--not stable!).
% Input:
%   A     square matrix
% Output:
%   L,U   unit lower triangular and upper triangular such that LU=A
  n = length(A);
  L = eye(n);     % ones on diagonal
  % Gaussian elimination
  for j = 1:n-1
    for i = j+1:n
      L(i,j) = A(i,j) / A(j,j);    % row multiplier
      A(i,j:n) = A(i,j:n) - L(i,j)*A(j,j:n);
    end
  end
  U = triu(A);
end
```

Consider the innermost loop. Since the different iterations in i are all independent, it is possible to *vectorize* this group of operations, that is, rewrite it without a loop. In fact, the necessary changes are to delete the keyword `for` in the inner loop, and delete the following `end` line. (You should also put a semicolon at the end of `i = j+1:n` to suppress extra output.)

(a) 🖥️ Make the changes as directed and verify that the function works properly.

(b) ✏️ Write out symbolically (*i.e.*, using ordinary elementwise vector and matrix notation) what the new version of the function does in the case $n = 5$ for the iteration with $j = 3$.

4. (Application of LU factorization: **FNC** 2.4.6) When computing the determinant of a matrix by hand, it is common to use cofactor expansion and apply the definition recursively. But this is terribly inefficient as a function of the matrix size.

(a) ✏️ Explain why, if $A = LU$ is an LU factorization,

$$\det(A) = u_{11}u_{22}\cdots u_{nn} = \prod_{i=1}^{n} u_{ii}.$$

This part is an analytical question. Do it by hand.

(b) 💻 Using the result of part (a), write a MATLAB function `determinant` that computes the determinant of a given matrix A using `mylu` from lecture. Include the function at the end of your live script. Use your function and the built-in `det` on the matrices `magic(n)` for $n = 3, 4, \ldots, 7$, and make a table (using `disp` or `fprintf`) showing n, the value from your function, and the relative error when compared to `det`.

5. (Proper usage of `lu`; **FNC** 2.6.1) ✏️ Suppose that $A \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$. On the left is correct MATLAB code to solve $A\mathbf{x} = \mathbf{b}$; on the right is similar but incorrect code. Explain using mathematical notation exactly what vector is found by the right-hand version.

```
[L,U]  = lu(A);
x = U \ ( L \ b );
```

```
[L,U]  = lu(A);
x = U \ L \ b;
```

6. (FLOP Counting) ✏️ Do **LM** 10.1–12(a,b,d). Justify your calculation of p and c for each part.

7. (Matrix norms; Sp20 midterm) Let

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}.$$

(a) ✏️ Calculate $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$, and $\|A\|_F$ all by hand.

(b) 💻 Imagine that MATLAB does not offer `norm` function and you are writing one for others to use, which begins with

```
function MatrixNorm(A, j)
% MatrixNorm   computes matrix norms
% Usage:
%    mat_norm(A, 1) returns the 1-norm of A
%    mat_norm(A, 2) is the same as mat_norm(A)
%    mat_norm(A, 'inf') returns the infinity-norm of A
%    mat_norm(A, 'fro') returns the Frobenius norm of A
```

Complete the program. (*Hint*: To handle the second input argument properly which can be a number or a character, use `ischaracter` and/or `strcmp`.)