MATH 315, Fall 2020
Homework 4

James Nagy

Homework Team *(team number)*: *(team name)*

This assignment is associated with Chapter 3, Solution of Linear Systems. It illustrates numerical methods for solving linear systems of equations, i.e. $Ax = b$, and also some complexities introduced when solving this problem on a computer.

NOTE: For hand calculations in the first problem, you do not need to LaTeX the details and steps – you can do this with pen or pencil, scan or save as a PDF file, and attach to the LaTeX parts.

1. Consider the following system of equations:

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 8 \\ -x_1 - 2x_2 + 3x_3 &= 1 \\ 3x_1 - 7x_2 + 4x_3 &= 10 \end{aligned}$$

   (a) Use Gaussian elimination to find the $A = LU$ factorization of A (by hand). Which built-in MATLAB function can be used to determine the LU factorization? Discuss the results you obtain when you implement this function.

   (b) Use Gaussian elimination to find the $PA = LU$ factorization of A (by hand). Which built-in MATLAB function can be used to determine this factorization?

   (c) Use your factorization from 1a to determine A^{-1}. (*Hint: Recall that* $A * A^{-1} = A * X = I$.) Discuss one benefit of using the factorization for this problem. (*Hint: The next problem might give you some ideas.*)

2. In this problem we consider solving $Ax = b_k$ for a given $n \times n$ matrix A, and multiple right hand side vectors, $b_1, b_2, \ldots, b_m$. We will assume these vectors are stored in an $n \times m$ matrix B; that is,

$$B = \begin{bmatrix} b_1 & b_2 & \cdots & b_m \end{bmatrix}.$$

   We could:

   - Use Gaussian elimination to solve each system $Ax = b_k$, or

- first compute the $PA = LU$ factorization, and then compute the solutions by using only backward and forward solves.

The purpose of this problem is to compare the time it takes for these two approaches. Here is code to illustrate how to do this comparison with 50 right hand side vectors:

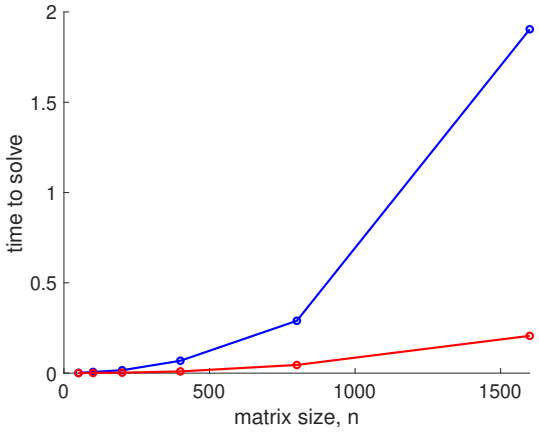
```
function [t1, t2] = CompareSpeed(n)
% (add comments here)
n_rhs = 50; n_runs = 10;
A  = rand(n); B = rand(n,n_rhs);
t1_sum = 0;
for j = 1:n_runs
    tic
    for k = 1:n_rhs
        x = A\B(:,k);
    end
    t1_sum = t1_sum + toc;
end
t1 = t1_sum/n_runs;

t2_sum = 0;
for j = 1:n_runs
    tic
    [L, U, P] = lu(A);
    for k = 1:n_rhs
        x = U\(L\(P*B(:,k)));
    end
    t2_sum = t2_sum + toc;
end
t2 = t2_sum/n_runs;
```

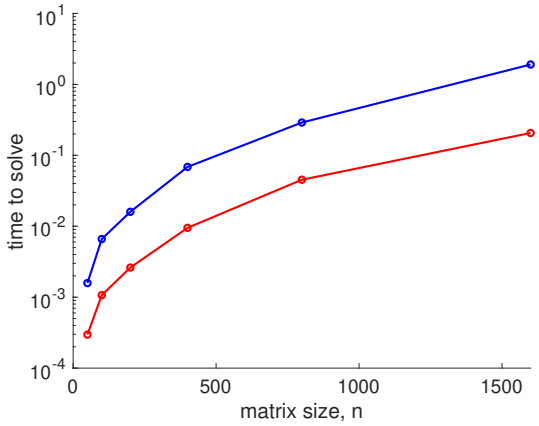
(a) Implement and run the above function, and test for $n = 50$.

(b) Add a few lines of comments (see the line `(add comments here)`) to explain the purpose of this function, inputs and outputs.

(c) What is the purpose of the loop `for j = 1:n_runs`?

(d) Write a script that will run the function for $n = 50, 100, 200, 400, 800, 1600$, and plot the timings using both `plot` and `semilogy`. Your plots should have matrix size, n, on the x-axis, and time to solve on the y-axis. That is, your plots should look similar to the following, but you should add a legend to each of the plots to indicate which approach is associated with the blue curve, and which is associated with the red.


time to solve
matrix size, n
0
0.5
1
1.5
2
0
500
1000
1500


time to solve
matrix size, n
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
500
1000
1500