MATH 315, Fall 2020
Homework 3

James Nagy

Homework Team *(7)*: *(Diego Gonzalez, Yuting Hou, Ken Su)*

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. It should also reinforce some of the theory of vector norms.

1. Systems of linear equations have a unique solution when the matrix A is nonsingular, i.e. when $\det(A) \neq 0$. Consider the formula for the determinant of a 2×2 matrix.

   (a) What are two numerical issues introduced in Chapter 2 that may arise when computing this value?
   Answer: First, consider the matrix A, such that:

   $$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   Then, the determinant of such matrix is given by:

   $$\det(A) = ad - bc$$

   We recall from Chapter 2 that multiplications of numbers may lead to **overflows**, while the subtraction of two numbers may lead to **catastrophic cancellation**. In other words, calculating the determinant of this matrix may lead to `Inf` or cause loss of significance.

   (b) Develop two matrices that suffer from these issues. Assume a floating point system that represents the mantissa with 3 digits.
   Answer: The calculation of this first matrix's determinant will cause and overflow, as the result of the various multiplications will produce numbers that are greater than the largest number that can be represented using double precision:

   $$\begin{bmatrix} 1.00 * 10^{160} & 2.16 * 10^{160} \\ 3.00 * 10^{160} & 1.12 * 10^{15} \end{bmatrix}$$

Since we are considering a mantissa that is represented with only 3 digits, calculating the next matrix's determinant will lead to a loss of significance:

$$\begin{bmatrix} 0.632 & 0.527 \\ 6.00 & 5.00 \end{bmatrix}$$

2. (a) Write a MATLAB code that will implement row-oriented forward substitution (see Fig. 3.5). Your code should be written as a function m-file, with input A (an $n \times n$ matrix) and b (an $n \times 1$ vector), and output the solution x (the $n \times 1$ vector that solves $Ax = b$).

More specifically, the beginning of your code should contain the following:

```
function x = RowSolve(A,b)
% This function computes the solution of the linear system Ax=b using
% forward substitution, using a row-oriented approach
%
%   Input:  A - nxn matrix
%           b - nx1 vector
%
%   Output: x - solution of the linear system
%

n = length(b);
x = zeros(n,1);

for ...
```

(b) Write a MATLAB code that will implement column-oriented forward substitution (see Fig. 3.5). Your code should be written as a function m-file, with input A (an $n \times n$ matrix) and b (an $n \times 1$ vector), and output the solution x (the $n \times 1$ vector that solves $Ax = b$). More specifically, the beginning of your code should contain the following:

```
function x = ColSolve(A,b)
% This function computes the solution of the linear system Ax=b using
% forward substitution, using a column-oriented approach
%
%   Input:  A - nxn matrix
%           b - nx1 vector
%
%   Output: x - solution of the linear system
%

n = length(b);
x = zeros(n,1);

for ...
```

(c) The following Matlab statements can be used to explore the computational cost of each function and compare and assess both solutions:

```
n = 10;
A = tril(rand(n));
```

```
b = rand(n,1);

disp('row-oriented:')
tic
xR = RowSolve(A,b);
toc

disp('column-oriented:')
tic
xC = ColSolve(A,b);
toc

disp('infinity norm of difference:')
norm((xC - xR), Inf)

disp('infinity norm of residual:')
norm((b - A*xR), Inf)
```

Create a Matlab script file, called containing the above Matlab statements. Run your script multiple times for various n, e.g. $n = 10, 100, 1000, \dots$. Does one approach produce a solution faster than the other? Does n influence this? Do the two approaches produce the same results? Is the residual zero? Discuss your findings and rationale for them.
Answer: Using the column-oriented approach was definitely faster, which became specially noticeable when the size of the matrix A was increased. That is, while having $n = 10$ didn't produce a large difference between the executing times for both approaches, it became clear that the column-oriented approach performed better as n was increased to values of around 1000. However, the results from both functions were the same regardless of how computationally demanding they were. This makes sense, as both functions are implementations of the same algorithm. The reason that caused `ColSolve.m` to be faster is that MATLAB stores matrices in column-major order. Lastly, we should note that the residuals are not zero. On the contrary, we often saw residuals that had even different orders of magnitude. This may suggest that there are calculation errors produced during the multiple multiplications, divisions and subtractions.

(d) What is a line of code that serves the same function as `norm((xC - xR), Inf)`?
Answer: Since `norm((xC - xR), Inf)` returns the component with the largest absolute value in the vector `(xC - xR)`, an alternate line of code that would give us the same result is `max(abs(xC - xR))`.

3. Show that the ∞-norm satisfies the properties of a vector norm.
Answer: Let us remember the definition of ∞-norm:

$$\left\|\boldsymbol{x}\right\|_\infty = \max_{1\le i\le n}\{|x_i|\}$$

In words, the ∞-norm gives you the largest absolute value $|x_i|$ among the components of the vector $\boldsymbol{x}$. Now, let us see how it satisfies a vector norm:

Property 1:

$$\left\|\boldsymbol{x}\right\|_\infty \geq 0\,, \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^n$$

is satisfied, because the absolute value of any real number is always greater or equal to zero.

Property 2:

$$\left\|\boldsymbol{x}+\boldsymbol{y}\right\|_\infty \leq \left\|\boldsymbol{x}\right\|_\infty + \left\|\boldsymbol{y}\right\|_\infty \text{ (triangular inequality)}$$

is satisfied, because:

$$\left\|\boldsymbol{x}+\boldsymbol{y}\right\|_\infty = \max_{1\leq i\leq n}\{|x_i+y_i|\} \underset{1\leq m\leq n}{=} |x_m+y_m| \quad \leq \quad |x_m|+|y_m| \quad \leq \quad \left\|\boldsymbol{x}\right\|_\infty + \left\|\boldsymbol{y}\right\|_\infty$$

Where we have used the Triangle Inequality for the real numbers.
Property 3:

$$\left\|c\boldsymbol{x}\right\|_\infty = |c|\left\|\boldsymbol{x}\right\|_\infty$$

where $c \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^n$ is satisfied, because:

$$\left\|c\boldsymbol{x}\right\|_\infty = \max_{1\leq i\leq n}\{|cx_i|\} = \max_{1\leq i\leq n}\{|c||x_i|\} = |c|\max_{1\leq i\leq n}\{|x_i|\} = |c|\left\|\boldsymbol{x}\right\|_\infty$$

Thus, the ∞-norm satisfies the three properties of a vector norm.