Lab Session 1

MA-581: Numerical Computations Lab R. Alam August 2, 2021

---

1. Let A be a random matrix generated by `rand(8)`. Find the maximum values (a) in each column, (b) in each row, and (c) overall. Also use `find` to find the row and column indices of all elements that are larger than 0.25.

2. A *magic square* is an n-by-n matrix in which each integer $1, 2, \ldots, n^2$ appears once and for which all the row, column, and diagonal sums are identical. MATLAB has a command `magic` that returns magic squares. Check its output for a few values of n and use MATLAB to verify the summation property. (The antidiagonal sum will be the trickiest. Look for help on how to "flip" a matrix.)

3. Consider the magic square `A = magic(n)` for `n = 3, 4, or 5`. What does

   ```
   p = randperm(n); q = randperm(n); A = A(p,q);
   ```

   do to

   ```
   sum(A), sum(A')', sum(diag(A)), sum(diag(flipud(A))), rank(A)
   ```

   The magic square `A = magic(4)` is singular. What do

   ```
   null(A), null(A,'r'), null(sym(A)), and rref(A)
   ```

   tell you about linear dependence of columns of `A`?

4. Are the following true or false? Assume A is a generic n-by-n matrix.
   (a) `A^ (-1)` equals `1/A` (b) `A.^ (-1)` equals `1./A`

5. Suppose `p` is a row vector of polynomial coefficients. What does the following line do?

   ```
   (length(p)-1:-1:0) .* p
   ```

6. (a) Look up `diag` in the online help and use it (more than once) to build the 16-by-16 matrix

$$D = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

   (b) Now read about `toeplitz` and use it to build D.

(c) Use toeplitz and whatever else you need to build

$$\begin{bmatrix} 1 & 2 & 3 & \cdots & 8 \\ 0 & 1 & 2 & \cdots & 7 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 2 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{8} \\ \frac{1}{2} & 1 & \frac{1}{2} & \cdots & \frac{1}{7} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \frac{1}{7} & \frac{1}{6} & \ddots & 1 & \frac{1}{2} \\ \frac{1}{8} & \frac{1}{7} & \cdots & \frac{1}{2} & 1 \end{bmatrix}$$

The second case looks best in format `rat`.

8. A random Fibonacci sequence is generated by choosing x_1 and x_2 and setting $x_{n+1} := x_n \pm x_{n-1}$, $n \geq 2$. Here $+$ and $-$ must have equal probability of being chosen. It is known that, with probability 1, for large n the quantity $|x_n|$ is of order c^n, that is, $|x_n| = \mathcal{O}(c^n)$, where $c := 1.13198824...$ Your task is to test this assertion. Try the following script.

```
>> clear
>> rand('state', 1000)
>> x = [1, 2];
>> for n=2:999, x(n+1) = x(n)+sign( rand-0.5)*x(n-1); end
>> semilogy (1:1000, abs(x))
>> c =1.13198824;
>> hold on
>> semilogy(1:1000, c.^ [1:1000])
hold off
```

Try to understand what the above script does and why it does so. Use matlab command `help` to understand an in-built function/command whenever necessary.

9. Given a positive integer x_1, define $x_n := f(x_{n-1})$ if $x_{n-1} > 1$ else STOP, where

$$f(x) := \begin{cases} 3x+1, & \text{if } x \text{ is odd,} \\ x/2, & \text{if } x \text{ is even.} \end{cases}$$

The unanswered question is, does the process always terminate? Or is there some starting value that causes the process to go on forever, either because the numbers get larger and larger, or because some periodic cycle is generated? This unsolve problem is variously known as Collatz problem, the $3x+1$ problem, the Syracuse problem, Kakutani's problem, Hasse's algorithm, and Ulam's problem. There is ample computational evidence to support the conjecture that the iteration indeed terminates. Your task is to write a MATLAB script and test the validity of this claim. The following MATLAB code fragment generates the sequence starting with any specified `n`.

```
y = n;
while n > 1
if rem(n,2)==0
n = n/2;
else
n = 3*n+1;
end
y = [y n];
end
```

We don't know ahead of time how long the resulting vector y is going to be but the statement `y = [y n];` automatically increases `length(y)` each time it is executed.

*** End ***