Lab Session 1

MA423 : Matrix Computations Lab July-November 2025 S. Bora

Important instructions:

(i) **Switch to `format long e` for all experiments.**

(ii) **Submit a single livescript program that contains all comments, answers and codes necessary to produce the required outputs. Ensure that the answers are correctly numbered and the file does not include any irrelevant material. The livescript program should be saved as MA423YourrollnumberLab1.mlx**

1. This is an exercise on handling matrices in MATLAB.

   (a). Generate the following square matrix which is known as the Wilkinson matrix without using any *for loops*.

   $$W_{ij} = \begin{cases} -1 & \text{if } i > j \\ 1 & \text{if } i = j \text{ or } j = n \\ 0 & \text{otherwise} \end{cases}$$

   Here n is the size of the matrix. You may write a function program `W = Wilkinson(n)` which takes the size n of the matrix as input for this.
   Hint: Use the MATLAB commands `eyes`, `ones` and `tril`.

   (b). A real $2n \times 2n$ matrix $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & -H_{11}^T \end{bmatrix}$ is said to be Hamiltonian if H_{12} and H_{21} are $n \times n$ matrices such that $H_{12}^T = H_{12}$ and $H_{21}^T = H_{21}$. Here T denotes the transpose of a matrix. Use concatenation and the `randn` command to generate a random real Hamiltonian matrix.

2. 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.

3. 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.)

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

   (a) What do the following do?

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

   (b) Do the following commands

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

change the outputs in (a)?

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

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

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

5. 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`

6. Suppose `p` is a row vector consisting of coefficents of a polynomial. What does the following line do?

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

7. (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. The solution of a system of equations $Ax = b$ can be obtained in MATLAB by setting `x = A\b`. MATLAB uses GEPP (Gaussian Elimination with Partial Pivoting) to find x for this command. [Wait for a few more classes to know the details!]

   The same may also be found by setting `x = inv(A) * b`.

   Write an M-file which finds the time taken by both these commands (use `tic` and `toc` commands of MATLAB for this) for 20 matrices with sizes increasing from 200 to 1150 in steps of 50 and plots log of the time taken along the y-axis (use `semilogy`) and the matrix sizes along the x-axis. The plots for both the methods should be on the same graph. Use `legends` to distinguish between your curves.