Math 551 Fall 2018
Lab 1

Goals: To learn and practice different forms of matrix input and basic operations with matrices in Matlab. The matrix operations to be studied include matrix addition and subtraction, scalar product, matrix product and elementwise matrix product in Matlab, matrix concatenation, and selecting submatrices.

To get started: Create a new Matlab script file and save it as "lab01.m".

Matlab commands to practice: `length, linspace, size, max, min, mean, sum, randi (rand), *, .*, eye, zeros, ones, diag, spdiags`.

What you have to submit: The file `lab01.m`, which you will create during the lab session.

Reminders About the Grading Software

Remember, the labs are graded with the help of software tools. If you do not follow the instructions, you will lose points. If the instructions ask you to assign a value to a variable, you must **use the variable name given in the instructions**. If the instructions ask you to make a variable named `x1` and instead you make a variable named `x` or `X` or `X1` or any name other than `x1`, the grading software will not find your variable and you *will* lose points. Required variables are listed in the lab instructions in a gray box like this:

Variables: `x1`, `A`, `q`

At times you will also be asked to answer questions in a comment in your M-file. You must **format your text answers correctly**. Questions that you must answer are shown in gray boxes in the lab. For example, if you see the instruction

Q7: What does the Matlab command `lu` do?

your file must contain a line similar to the following somewhere

```
% Q7: It computes the LU decomposition of a matrix.
```

The important points about the formatting are

- The answer is on a single line.
- The first characters on the line is the comment character `%`.
- The answer is labeled in the form `Q`n`:`, where n stands for the question number from the gray box.

If you do not format your answers correctly, the grading software will not find them and you *will* lose points.

INTRODUCTION

It is assumed that the readers are familiar with basic matrix operations and their properties, so the definitions and facts here are given mostly for the sake of a review.

We can think of a matrix as a table of numbers:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \dots & \dots & \ddots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \ddots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix}$$

The matrix $\mathbf{A}$ above contains m rows and n columns. We will say that $\mathbf{A}$ is an $m \times n$ (m-by-n) matrix, and call m, n the dimensions of the matrix A. The matrix A above can also be compactly written as $\mathbf{A} = (a_{ij})$, where a_{ij} is called (i, j) element of the matrix $\mathbf{A}$.

Matlab (originally named "MATrix LABoratory") has an extensive list of functions which work with matrices. The operations discussed in this project consist of matrix addition/subtraction, multiplication of a matrix by a constant (scalar multiplication), transposition of a matrix, selecting submatrices, and concatenating several smaller matrices into a larger one, matrix multiplication and componentwise matrix multiplication, and operations of finding maximal/minimal element in the matrix, and finding the sum/mean in the row/column of the matrix.

Scalar multiplication of a matrix $\mathbf{A}$ by a constant c consists of multiplying every element of the matrix by this constant:

$$c\mathbf{A} = (ca_{ij}) .$$

Matrix addition and subtraction are possible if the matrices have the same dimensions and is done componentwise:

$$\mathbf{A} \pm \mathbf{B} = (a_{ij} \pm b_{ij}) .$$

If n is a positive integer then the $n \times n$ *identity matrix*, denoted by $\mathbf{I}_n$, has 1 in every diagonal entry (i, i) for $1 \le i \le n$, and 0 in every other entry. There is one identity matrix for every dimension n. For instance,

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

are the 2×2 and 3×3 identity matrices.

The identity matrices are special examples of *diagonal matrices*. A matrix $\mathbf{A}$ is diagonal if $A(i, j)$ is 0 whenever $i \neq j$ (it is not required that $a_{ii} \neq 0$ for any i).

A *matrix product* $\mathbf{C} = \mathbf{AB}$ of an $m \times n$ matrix $\mathbf{A}$ and an $l \times p$ matrix $\mathbf{B}$ exists if and only if $n = l$. If this condition is satisfied, then $\mathbf{C}$ is an $m \times p$ matrix with the elements

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Observe that matrix multiplication is non-commutative, $\mathbf{AB} \neq \mathbf{BA}$.

For example if $\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ -1 & -1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 0 & 2 \\ 3 & -1 & 1 \end{bmatrix}$ then the product is given by

$$\mathbf{AB} = \begin{bmatrix} (2 \cdot 4 + 3 \cdot 3) & (2 \cdot 0 + 3 \cdot -1) & (2 \cdot 2 + 3 \cdot 1) \\ (1 \cdot 4 + 0 \cdot 3) & (1 \cdot 0 + 0 \cdot -1) & (1 \cdot 2 + 0 \cdot 1) \\ (-1 \cdot 4 + -1 \cdot 3) & (-1 \cdot 0 + -1 \cdot -1) & (-1 \cdot 2 + -1 \cdot 1) \end{bmatrix} = \begin{bmatrix} 17 & -3 & 7 \\ 4 & 0 & 2 \\ -7 & 1 & -3 \end{bmatrix}.$$

If $\mathbf{C} = \begin{bmatrix} 2 & 1 \\ 1 & 7 \end{bmatrix}$ and $\mathbf{D} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$ then the product CD is not defined because the number of columns of $\mathbf{C}$ does not equal the number of rows of $\mathbf{D}$.

Please, perform the tasks below.

TASKS

1. Open the "lab01.m" Matlab script you have created and type in the following commands:

   ```
   %% basic operations with matrices
   A=[1 2 -10 4; 3 4 5 -6; 3 3 -2 5];
   B=[3 3 4 2];
   ```

   This will create a new cell and define two new variables `A` and `B`. The variable `A` stores a 3×4 matrix, while the variable `B` contains a 1×4 row vector. Use the Workspace window to look at the variables `A` and `B`.

   Variables: `A`, `B`

2. Type in the following commands:

   ```
   lengthA = length(A)
   lengthB = length(B)
   ```

   Variables: `lengthA`, `lengthB`

   Q1: What does the command `length(A)` do?

3. Add the vector `B` as the fourth row of the matrix `A` and save the result as a new matrix `C` using the following code:

   ```
   C=[A; B];
   ```

   Variables: `C`

4. Create a new matrix `D` whose entries are located in the rows $2, 3, 4$ and columns $3, 4$ of the matrix `C`:

   ```
   D=C(2:4,3:4);
   ```

   Variables: `D`

5. Matlab allows to create equally spaced vectors which can be useful in many situations. For instance, run the following code:

```
EqualSpaced=0:pi/10:2*pi
EqualSpaced1=linspace(0,2*pi,21)
```

Take a look at the output produced by Matlab. Observe that both of these commands created the vector with equally spaced entries ranging from 0 to 2π and the distances between the elements equal to $\pi/10$.

Variables: EqualSpaced, EqualSpaced1

6. Create a new cell and find the maximal and minimal elements in each of the columns of the matrix `A`. Store the result respectively in row vectors named `maxcolA` and `mincolA`. You can use Matlab `max` and `min` functions:

```
%% Max, min, sum and mean
maxcolA=max(A)
mincolA=min(A)
```

Variables: `maxcolA` and `mincolA`

7. Now repeat the previous step but find maximal and minimal elements in each row of the matrix `A`. One way is to use `max` and `min` with an additional argument 2 to show that you are interested in the maximum and minimum along the second dimension (i.e. `max(A,[],2)`). Save the results as `maxrowA` and `minrowA`. Finally find the maximal and minimal elements in the whole matrix `A`. One way to do this is to find the maximal (or minimal) element in the maximal (or minimal) elements of each column of `A` (e.g. `max(maxcolA)` or `max(max(A))`). Save the results as `maxA` and `minA`.

Variables: `maxrowA`, `minrowA`, `maxA`, `minA`

8. Repeat the last two steps using the commands `mean` and `sum` instead of `max` and `min`. Type `help mean` and `help sum` in the command line to learn more about `mean` and `sum`, respectively. You should create six new variables, corresponding to the column/row mean/sum, and mean/sum of the elements of the entire matrix.

Variables: `meancolA`, `meanrowA`, `sumcolA`, `sumrowA`, `meanA`, `sumA`

Q2: What do the commands `mean` and `sum` do?

9. Create a new cell and use the command `randi([-4 4],5,3)` to create two matrices `F` and `G` with 5 rows and 3 columns and random integer entries from -4 to 4:

```
%% Matrix addition/subtraction, and multiplication
F=randi([-4 4],5,3)
G=randi([-4 4],5,3)
```

Variables: `F`, `G`

10. Perform the operations `0.4*F`, `F+G`, `F-G`, `F.*G`, storing the results in `ScMultF`, `SumFG`, `DiffFG`, and `ElProdFG` respectively.

Variables: `ScMultF`, `SumFG`, `DiffFG`, `ElProdFG`

Q3: What does the last operation do?

11. Check the size of `F` and the size of `A`, storing the results in `sizeF` and `sizeA`, respectively.

Variables: `sizeF`, `sizeA`

Q4: Can we matrix-multiply `F` and `A`? Explain.

12. Compute $H = F * A$.
    Variables: `H`
    Q5: What are the dimensions of `H`?

13. Generate the identity matrix `eye33` with 3 rows and 3 columns using the Matlab `eye` command.
    Variables: `eye33`

14. Run the commands:

```
zeros53=zeros(5,3);
ones42=ones(4,2);
```

    Variables: `zeros53`, `ones42`
    Q6: What do the functions `zeros` and `ones` do?

15. Generate the diagonal 3×3 matrix S with the diagonal elements $\{1, 2, 7\}$ using the Matlab `diag` function. To learn about `diag` type `help diag` in the command line.
    Variables: `S`

16. Now let us do the opposite: extract the diagonal elements from the matrix and save them in the separate vector. The same function `diag` accomplishes that as well:

```
R=rand(6,6)
diagR=diag(R)
```

    This creates a matrix `R` with random entries from the interval $(0, 1)$, extracts the diagonal entries from it, and saves them in the vector `diagR`. Observe that the function `diag` has other interesting possibilities (use `help diag`).
    Variables: `R`,`diagR`

17. Another function which allows to create diagonal matrices is `spdiags`. Technically, this function creates a *sparse* matrix (a matrix with a large number of zeros). These matrices are stored in a special form in Matlab (only non-zero elements are stored), and operations with them are also done in a special way. To convert sparse form of the matrix to the regular one, the command `full` can be used. Run the following code:

```
diag121=spdiags([-ones(10,1) 2*ones(10,1) -ones(10,1)],[-1 0 1],10,10);
full(diag121)
```

    Variables: `diag121`
    Q7: What does this code do?