# Engineering 13300
# MATLAB 2

This assignment has three sections.

I) The first section has three Team Exercises (A,B and C) that are intended to be done with your team. Work the problems together and answer the questions. Type the solutions into MATLAB to check them AFTER you have worked them out by hand. After you find the correct answers, you can move to the second section

II) The second section has five team tasks (Tasks 1-5) that you may work together and develop code together as long as you document who contributed.

III) The third section has two individual tasks (Tasks 6-7). You can work together to conceptually help each other but each person should work on their own code.

## Section I

These are to be worked together as a team. There are no files to be uploaded to Brightspace. Rather, work the problems and come up with the answers and be prepared to present them to a member of the instructional team. After you get the right answers, move on to the next problem. The intent is to understand the concepts before we begin to write code ourselves.

### Team Exercise A (if-conditionals)

Recall the Python code on the right from the Python 2 assignment. Rewrite this code in MATLAB as a team. Use the values below to check your code replacing the *** in the code with the appropriate values

| X | Y | result |
|---|---|---|
| 40 | 40 | z=4 |
| 20 | 20 | z=400 |
| 55 | 20 | Error – z not defined |

```
x= ***
y = ***
if x<=50:
    z=4
    if y<30:
        z=x*y
    elif y>=100:
        z=x+y

elif y >= 60:
    if x > 80:
        z=x
    elif  y > 50:
        z=y
else:
        z=z*2

print('z=', z)
```

**Team Exercise B (repetition – loops - while)**
Recall the Python code below from the Python 3 assignment. Rewrite this code in MATLAB as a team. Verify that the MATLAB code is correct by checking for the expected output

Previous Python Code:
```
i=1
x=3
y=6
while i<10:
    for z in [x,y]:
        i+=z
    if x<3:
        x+=1
print(i)
```

Expected output: 10

**Team Exercise C (repetition – loops - while)**
Recall the Python code below from the Python 3 assignment. Rewrite this code in MATLAB as a team. Verify that the MATLAB code is correct by checking for the expected output

Previous Python Code:
```
x=[3,6,-2,1]
y=0

for index in x:
    if index <= 3:
        y=y+index
    print(f'y={y}')
```

Expected output:
```
y=3
y=3
y=1
y=2
```

**Recall the guidelines for team activities:**

1. You should work as a team; **all** team members will be held responsible for all material. You may work together and contribute to one program and submit similar codes as long as the contributors to the development of the solution are documented.
2. Each student is responsible for submitting their own assignment.

## Team Task 1 (of 7)

**Problem Steps**

1. Download the script `Ma2_matrix_magic_template.m` file.
2. Open template and complete the header information.
    a. Add the problem set number, your name, and your section-team number.
    b. Declare your partner in the paired programming area.
    c. List any additional contributors who work with you and your paired partner.
3. Save your script with the name format required by the deliverables list.
4. Use MATLAB to learn what these built-in functions do: `zeros` and `sum`.
5. In the `INITIALIZATION` section of your script file, create matrices `A` and `vals`, using the steps described below.
    a. Use the function **`zeros`** to create a 4x4 matrix named `A`.

```
A =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

    b. Use MATLAB matrix creation commands to create a matrix, `vals`, that matches the matrix below. **Note:** The numbers are not sequential.

```
vals =

     1     3     2     4
     5     6     7     8
     9    10    11    12
    13    15    14    16
```

6. In the `COPY & CONCATENATE` section of your script file, perform the following:
   **Note:** Do not hardcode assignments unless told to do so.
    a. Copy the center 2x2 matrix of `vals` and assign it to `M`.
    b. Copy from `vals` a 2-element row vector `[3 2]` and assign it to `C`.
    c. Copy from `vals` a 2-element row vector `[15 14]` and assign it to `D`.
    d. Create 1x4 row vector `E` that concatenates `D` between the first and fourth elements in the first row of `vals` to create the vector `[1 15 14 4]`, and uses square brackets to complete the concatenation in one line of code.
    e. Create 1x4 row vector `F` that concatenates `C` between the first and fourth elements in the fourth row of `vals` to create the vector `[13 3 2 16]`, and uses square brackets to complete the concatenation in one line of code.

7. In the `REPLACE MATRIX ELEMENTS` section of your script file, perform the following:
   a. Use only `M`, `E`, and `F` to replace the first row of `A`, the fourth row of `A`, and the center 2x2 matrix of `A`. Matrix `A` should look like the matrix below once these replacements are complete.

```
A =

     1    15    14     4
     0     6     7     0
     0    10    11     0
    13     3     2    16
```

   b. Complete the following replacements **without hardcoding values**:
      - Replace the 0 directly below the 1 in matrix `A` with the 12 from matrix `vals`.
      - Replace the 0 directly above the 13 in matrix `A` with the 8 from matrix `vals`.
      - Replace the 0 directly below the 4 in matrix `A` with the 9 from matrix `vals`.
      - Replace the 0 directly above the 16 in matrix `A` with the 5 from matrix `vals`.
8. In the `FINAL MATRIX` section of your script file, perform the following:
   a. Create a vector `X` that contains the sums of the columns of `A`.
   b. Concatenate vector `X` to the bottom of matrix `A` to create a new matrix, `G`. Concatenation requires the use of square brackets.
   c. Create a vector `Y` that contains the sums of the rows of `G`.
   d. Concatenate vector `Y` to the right of matrix `G` to create a new matrix, `H`. Concatenation requires the use of square brackets.
   e. Replace the lower right corner value of `H` with the sum of the first four values on the diagonal from the upper left corner and moving toward the lower right corner.
9. In the `FORMATTED TEXT DISPLAY` section of your script file, use three **`fprintf`** statements to display your results as shown:
   **Note:** Do not hardcode the numerical values within your **`fprintf`** statements; use array indexing of `H` to identify the appropriate values of `H` to display.
   After doing step 8.e, the value in the center of `H` is ___.
   After doing step 8.e, the value in the upper left of `H` is ___,
   and the value in the upper right of `H` is ___.
   After doing step 8.e, the value in the lower left of `H` is ___,
   and the value in lower right of `H` is ___.
10. **Publish** your script as `Ma2_matrix_magic_`*`username`*`_report.pdf` [refer the 'how to publish' demo video]

***Note: For Team task 2 & 3 – Open ENGR133_Fa20_Ma2_Answersheet.docx and save the file as `Ma2_team_teamnumber.pdf`. This is the only file you will be submitting for Team tasks 2 & 3***

## Team Task 2 (of 7):

**Learning Objectives:** The following exercise demonstrates scalar and row/column vector manipulations. These include matrix methods and element by element methods.

**Problem Steps:**
Create the following in a MATLAB script file:

- A scalar called `Ascalar` by typing `Ascalar = 3`
- A row vector called `Arowvector` and assign the values starting from 0 and ending at 3 in increments of 1 to it. Create this row vector by typing `Arowvector = [0 1 2 3]` into the `INPUTS` section of the script file. Be sure to use square brackets (you could also create the row vector by typing `Arowvector = [0,1,2,3]` where you use commas instead of spaces to separate elements)
- A second row vector called `Browvector` with four elements, 4, 2, 0, and $-2$. Do this by typing `Browvector = 4:-2:-2` (start value: increment: end value).
- A third row vector called `Crowvector` with three elements of your choice. The value of each element should be an integer in $[-5, +5]$.
- A column vector called `Acolvector`. Assign `Acolvector` the same values assigned to `Arowvector`. To do this, type `Acolvector = [0;1;2;3]`. (Note: To compute the transpose of an array, use the high comma ‘. For example, to assign the transpose of `Arowvector` to `Acolvector` type `Acolvector = Arowvector’`)
- A second column vector called `Bcolvector` with four elements. The value of each element should be an integer in $[-5, +5]$.

**Part A:**
Compute the mathematical operations in the table below. Record your results on the answer sheet:

1. Type the answer or response from MATLAB in the center column in the table.
2. In the right column, explain the results or why MATLAB cannot perform the operation.

| Mathematical Operation |
|---|
| `Arowvector + Arowvector` |
| `Arowvector + Browvector` |
| `Arowvector + Ascalar` |
| `Arowvector - Arowvector` |
| `Arowvector - Crowvector` |
| `Acolvector + Bcolvector` |
| `Arowvector + Bcolvector` |

**Part B:**
MATLAB was designed to perform matrix operations by default. Therefore, all mathematical operations involving *element-by-element* multiplication, division, and exponentiation of vectors require suppression of the default matrix math. This is done using **dot notation**, which refers to the use of a period before the following symbols:

- Multiplication: `*`
- Division: `/`
- Exponentiation: `^`

(Reminder: In Python you use `**` for exponentiation; `^` in Python is bitwise XOR)

Perform the operations listed in the table below. Record your results on the answer sheet:

1. Type the answer or response from MATLAB in the center column in the table.
2. In the right column, explain the results or why MATLAB cannot perform the operation.

| **Mathematical Operation** |
|---|
| `Arowvector *  Browvector` |
| `Arowvector .* Browvector` |
| `Arowvector *  Ascalar` |
| `Arowvector .* Ascalar` |
| `Arowvector ./ Browvector` |
| `Arowvector ^  Ascalar` |
| `Arowvector .^ Ascalar` |

**Task 2 Files:** `Ma2_team_teamnumber.pdf (answer sheet)`

## Team Task 3 (of 7):

**Learning Objectives:**

- Compare scalars and vectors using relational operators in MATLAB.
- Apply any, all, and find functions in MATLAB

**Problem Steps:**

**1.** In the MATLAB Command Window, type:

```
>> Cvector = [1, 2];
>> Aarray = [1, 2; 3, 4];
>> Barray = [1, 0; -1, 4];
>> Carray = [1, 2; 3, 4; 5, 6];
>> Darray = [0, 0, 1; 3, 5, -5; 1, 0, 1];
```

**2.** Perform the comparisons indicated in the table below (type the commands one at a time). Either explain the result or explain (in your own words) why MATLAB cannot perform the operation. Record the MATLAB answer or response in the answer sheet.

| | **Command** |
|---|---|
| a. | `Aarray >= Barray` |
| b. | `Answer_b = Aarray.*Barray ~= 1` |
| c. | `Barray < (Aarray - Answer_b) <= (Answer_b < 1) *3` |
| d. | `Barray > Carray` |
| e. | `[Barray; Cvector] == Carray` |

**3.** Perform the operations indicated in the table below. What is displayed in the MATLAB Command Window? Explain the result on the answer sheet. **(Hint:** Use MATLAB help to learn about **any**, **all**, and **find** functions.)

| | **Command** |
|---|---|
| a. | `any(Aarray) + any(Barray)` |
| b. | `all(Aarray) .* all(Barray)` |
| c. | `all(Carray > 1)` |
| d. | `all(any(Barray < -1 )) + any(all(Darray))` |
| e. | `find(Barray).^(find(Carray > 5))` |
| f. | `find(any(Darray == 1))` |

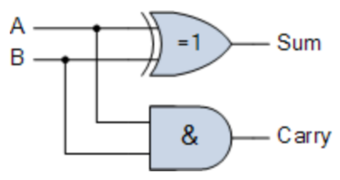
**Task 3 Files (keep working in the same file as task 2):** `Ma2_team_teamnumber.pdf`

## Team Task 4 (of 7):

**Learning Objectives:**
Use conditional statements including “if” in MATLAB;
Use logical operators in MATLAB;

Open the template file ***ENGR133_MATLAB_Template.m***. Modify the file to create a script file that solves the following problem:

Computers use binary for all calculations, and it is important that we can track binary operations to understand how the computer operates.  One way to represent a binary calculator is with an xor and AND gate as shown below.  When adding two single digit binary numbers, the sum represents the first digit and the carry represents the second digit.  So, if we added 1+1 in binary we would get 10, where zero is in the sum column and the carry is 1.

| Symbol | Truth Table | | | |
|---|---|---|---|---|
| | B | A | SUM | CARRY |
|  | 0 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 0 |
| | 1 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 1 |

Reference:  https://www.electronics-tutorials.ws/combination/comb_7.html

- Write a MATLAB program that uses the “input” command in MATLAB to prompt the user to enter a binary number of 1 or 0. The program should check if the number is anything other than a 1 or a 0 and if so print an error statement.  Remember that the “input” command in MATLAB yields a number rather than a string link Python.
- Next prompt the user for a second number of 1 or 0 and check for an error and display an error message if it is anything other than a 1 or 0.
- If the user enters a 1 or 0 for each input, the program calculates the result of the binary addition and displays the two digits with a print line that shows “Binary sum = ##” where the ##’s are the digits of the binary addition corresponding to the table above.

Note:  The “return” command in MATLAB can be used to terminate a program within an if statement

**Publish** your script as `Ma2_task4_Teamnumber_`*`username`*`_report.pdf`

**Task 4 Files:**

`Ma2_task4_Teamnumber_`*`username`*`_report.pdf`

## Team Task 5 (of 7):

**Learning Objectives:**
Use for loops to conduct repetitive operations in MATLAB;
use while loops to conduct repetitive operations in MATLAB;
use loops to evaluate corresponding data sets (i.e. arrays, and matrices) in MATLAB.

Open the template file ***ENGR133_MATLAB_Template.m***. Modify the file to create a script file that solves the following problem:

**Background:** Recall from Python 3 that a **Maclaurin series** is an *n*th degree polynomial that can be used to approximate a function around *x* = 0. The exponential function, $e^x$, can be approximated as follows:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \cdots$$

Where the infinite series approximation converges with the actual function over all real numbers. Practically, for a finite approximation, the fewer terms that are used, the less accurate the approximation will be, particularly as *x* gets further from 0.

**Part A**

Using you're the "`factorial`" function in MATLAB (e.g. z = factorial(3) yields z = 6) , write a program which will use the Maclaurin series defined above to approximate $e^x$ with specified values of *n* and *x*. The program should also print to the screen the percent error of the estimated value with respect to a calculation of the actual value. Revisit your flow diagram created in the Python 3 assignment. Make notes on changes needed for MATLAB. You do not have to turn in the flow diagram for this assignment. Note to ask user to input n, and x in this order.

**Example with an initialized value of *n* = 5 and *x* = 3:**

```
Approximate value:  18.40
Actual value:       20.09
Error:              -8.4%
```

**Publish** your script as `Ma2_task5A_Teamnumber_`*`username`*`_report.pdf`

**Part B**

This time, write a program that will generate as many terms as is necessary to approximate the function to a specified level of accuracy (i.e., below an percent error threshold) at a specified value of *x*. Additionally, print to the screen the number of terms the series must contain before the target accuracy is achieved. As in part A, revisit the flow diagram previously created for Python 3. As with Part A, you are not required to turn the flow diagram in for this assignment. Note that ask user to input x, and target error threshold in this order.

**Example with an initialized value of *x* = 3 and target error threshold of 5%:**

```
Target error threshold:  5%
Actual value:            20.09
Terms needed:            7
Approximate value:       19.41
```

**Publish** your script as `Ma2_task5B_Teamnumber_`*`username`*`_report.pdf`

**Task 5 Files:**

1. `Ma2_task5A_Teamnumber_`*`username`*`_report.pdf`
   `Ma2_task5B_Teamnumber_`*`username`*`_report.pdf`

# Individual Tasks

Guidelines for Tasks 6-7:
Tasks 6-7 are individual tasks. You may seek help from classmates, the instructional team or others but the work you submit should be your own. If you collaborate with others and use information developed together or by someone else, ALWAYS document and reference that material.

## Task 6 (of 7) Interest Compounding

Open the template file ***ENGR133_MATLAB_Template.m***. Modify the file to create a script file that solves the following problem:

Starting at age 25, Riley deposits $11,000 at the beginning of each year into a retirement savings account that pays 2% interest per year, compounded annually. In how many years would the savings account exceed $1 million?

**Problem Steps**

1. Draw a flowchart or diagram before you start to program and save it as `Ma2_Ind_username.pdf`
2. Initialize the balance of the account and the year (starting at zero) in the `INITIALIZATION` section.
3. In the `CALCULATIONS` section, create a while loop that will compound the balance of the account as long as the balance is less than $1 million.
4. Display the age of person when the balance finally exceeds $1 million using the `fprintf` function in the `OUTPUTS` section.

   ```
   >> The savings account would exceed $1 million after X years.
   ```
5. **Publish** your script as a PDF and name it `Ma2_Task6_`*`username`*`.pdf`. [refer the 'how to publish' demo video]

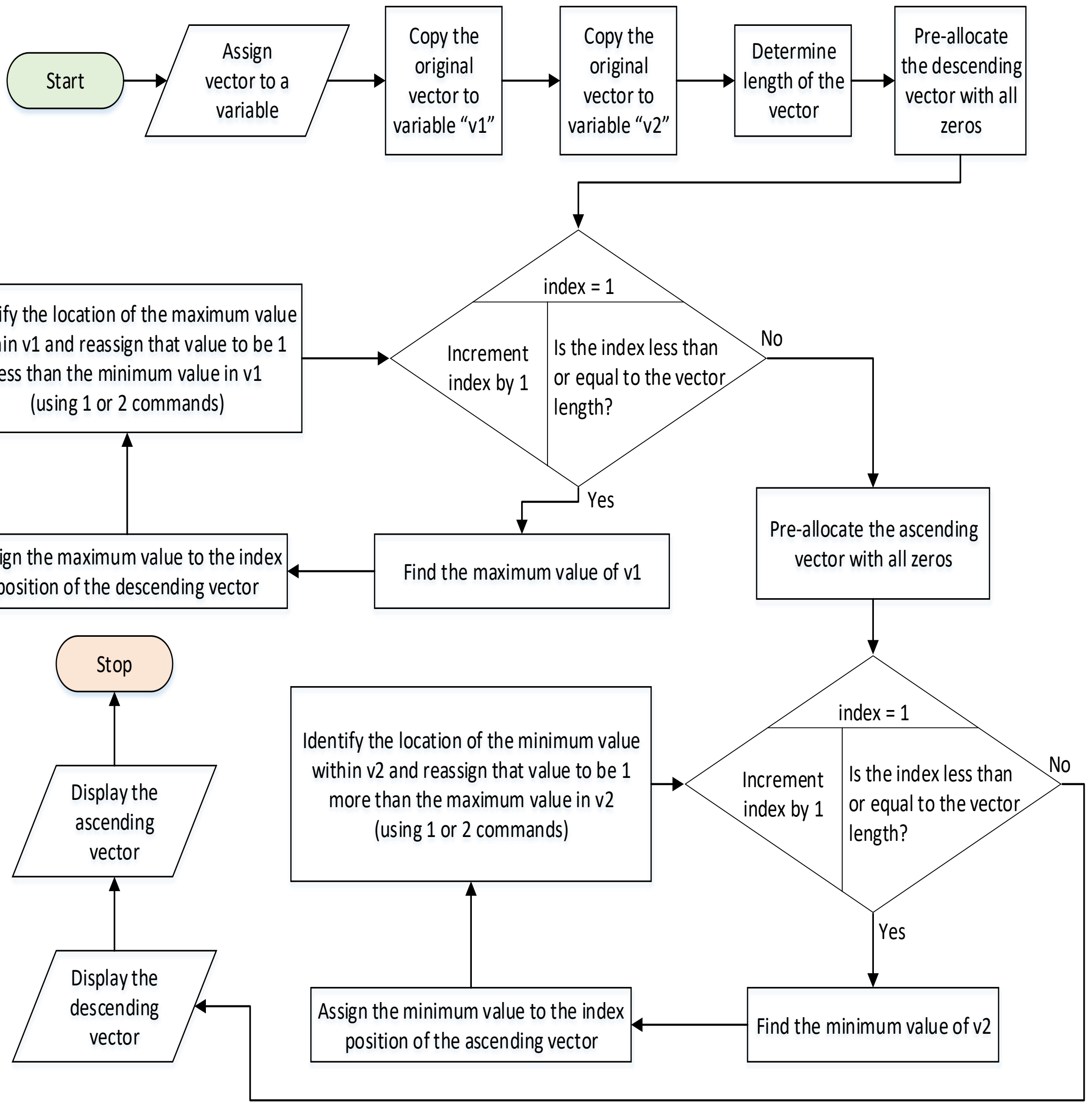
**Task 6 Files:**

- `Ma2_Ind_username.pdf`
- `Ma2_Task6_username.m`
- `Ma2_Task6_username.pdf` (published file)

## Task 7 (of 7) Vector Array Sorting using loops

Open the template file ***ENGR133_MATLAB_Template.m***. Modify the file to create a script file that solves the following problem:

Sorting algorithms are commonly used in computer programming. Below is a flowchart for a basic vector-sorting algorithm. It first sorts the vector elements into descending order. Then it sorts the same vector into ascending order. It will sort vectors of any size with unique elements (i.e., no repeated numbers).


Start
Assign vector to a variable
Copy the original vector to variable "v1"
Copy the original vector to variable "v2"
Determine length of the vector
Pre-allocate the descending vector with all zeros
index = 1
Increment index by 1
Is the index less than or equal to the vector length?
No
Yes
Find the maximum value of v1
Assign the maximum value to the index position of the descending vector
Identify the location of the maximum value within v1 and reassign that value to be 1 less than the minimum value in v1 (using 1 or 2 commands)
Pre-allocate the ascending vector with all zeros
index = 1
Increment index by 1
Is the index less than or equal to the vector length?
No
Yes
Find the minimum value of v2
Assign the minimum value to the index position of the ascending vector
Identify the location of the minimum value within v2 and reassign that value to be 1 more than the maximum value in v2 (using 1 or 2 commands)
Display the descending vector
Display the ascending vector
Stop

**Problem Steps**

1. In the `INITIALIZATION` section, set the initial vector to `v = [10 5 1 8 -9 0 2 3]`.
2. In the `CALCULATIONS` section, create two for loops, one for ascending order and one for descending order. Hint: lookup `zeros` and `length` inbuilt functions
3. Display the sorted array using `fprintf` and `disp` function in the `OUTPUTS` section.
4. **Publish** your script as a PDF and name it Ma2_Task7_*username*.pdf. [refer the 'how to publish' demo video]

Sample output:

```
v = [2 5 -10 45 -23 15]

Vector sorted in descending order:

    45 15 5 2 -10 -23


Vector sorted in ascending order:

    -23 -10 2 5 15 45
```

**Task 7 Files:**

- `Ma2_Task7_username.m`
- `Ma2_Task7_username.pdf` (published file)