ME 444
MATLAB® FOR ENGINEERS

Lecturer:

Dr. Nurettin Furkan DOĞAN

Mechanical Engineering Department

4 th Floor Office No: 303

nfdogan@gantep.edu.tr

0342 317 1200- 2569

Office Hours:

Tuesday: 13.30- 15:00

Friday : 10.00- 11.30

CHAPTER 7

LOOP STRUCTURES-1

Loops

- Loops (repetitive structures) are one of the most used structures in programming.
- Loops are used if an operation is desired to be performed more than once (a finite number of course).
- The types of loops used in MATLAB are:
  - ✓ for loop
  - ✓ Nested For loop
  - ✓ While loop

Loops / `for` loop

- The **`for`** loop is used to perform an operation more than once. (For example, iteration solutions used in root finding problems).

```
for i=1:n
        operations
end
```

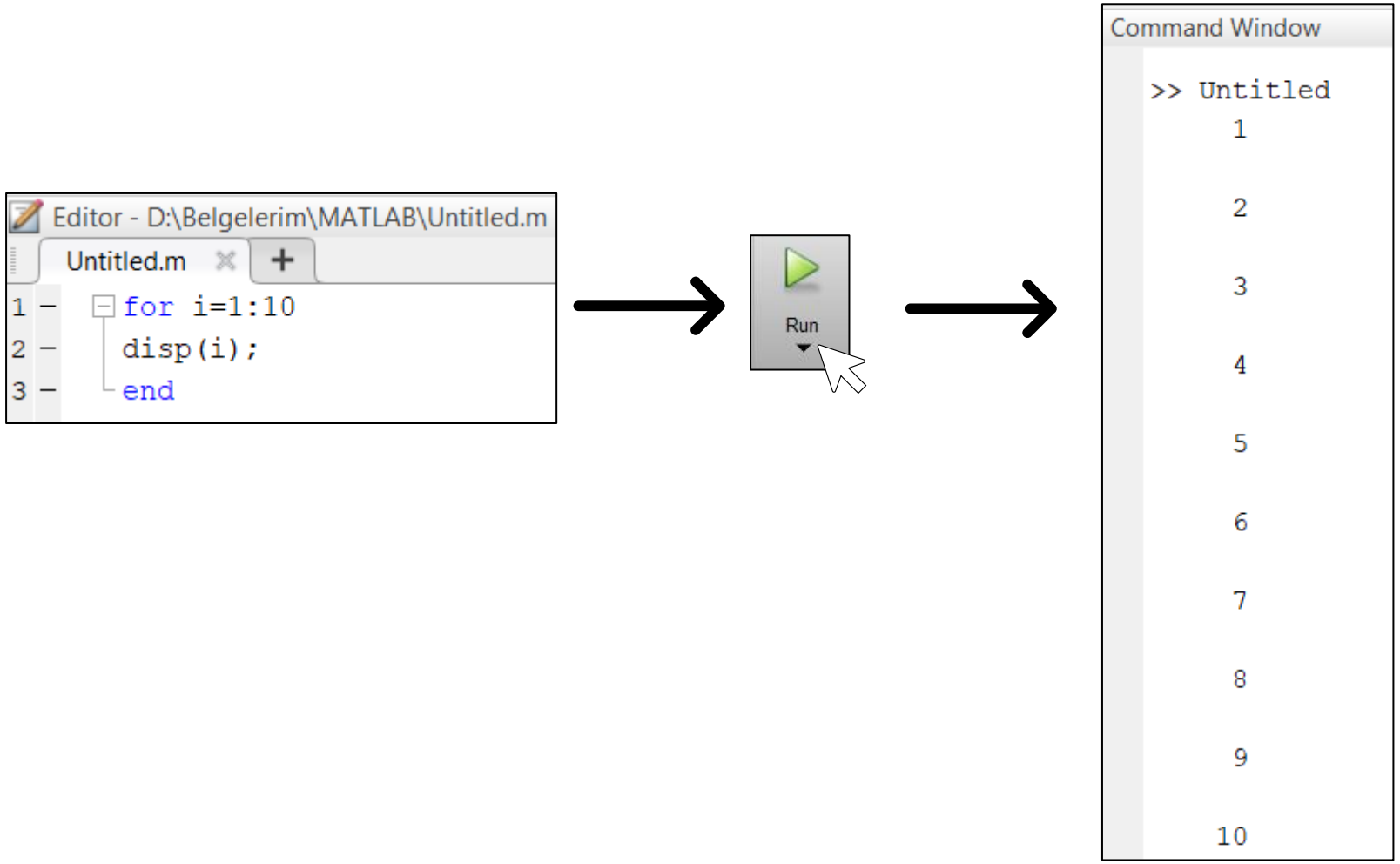
- ✓ `i:` It is a vector called an iterator and has an increment of 1 from 1 to n integers.
- ✓ The number of elements of i represents the number of operations to be performed. That is, the number of elements of our loop i returns.
- ✓ `operations:` They are operations that will run as many as the number of elements of the iterator.

- ✓ The `i` value does not have to start from 1.
- ✓ Also, the increment of i can be changed.

`i=3:2:11→ i=3,5,7,9,11`

Loops / `for` loop

- For example, let's create the program that prints the numbers 1 to 5 on the screen.



Loops / `for` loop

- For example, let's write a program that finds the sum of the numbers 1 to 10.

Editor - D:\Belgelerim\MATLAB\Untitled.m

Untitled.m

```
1 -    clear all; clc;
2 -    toplam=0;
3 -    for i=1:10
           toplam=toplam+i;
       end
       disp(toplam);
```

Run

Command Window

```
>> Untitled
    55
```

Command Window

```
1
3
6
10
15
21
28
36
45
55
```

✓ What do you think would happen if we put the expression disp(sum) inside the loop, before end???

✓ Each time the loop returns, disp runs and writes the total value to the screen at each stage.

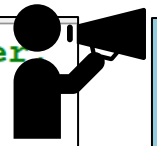
✓ Let's examine the loop step by step together:

i_{old}	i_{new}	sum_{old}	sum_{new}
-	1	0	1
1	2	1	3
2	3	3	6
3	4	6	10
4	5	10	15
5	6	15	21
6	7	21	28
7	8	28	36
8	9	36	45
9	10	45	55

Loops / `for` loop

- Let's write a program (using loops) to find the factorial of the entered number.

```
1   %This program calculates the factorial of entered number
2   num=input('Enter a number:');
3   f=1;
4   for i=1:num
5       f=f*i;
6   end
7   fprintf('%d! is equal to %d. \n',num,f);
8
```




```
Command Window
>> Untitled4
Enter a number:5
5! is equal to 120.
```

✓ Let's examine the loop step by step together: Suppose we entered 5.

i_{old}	i_{new}	f_{old}	f_{new}
-	1	1	1
1	2	1	2
2	3	2	6
3	4	6	24
4	5	24	120

```
Command Window
>> Untitled4
Enter a number:5
5! is equal to 1.
5! is equal to 2
5! is equal t
5! is equal t
5! is equal t
5! is equal to 120.
```

✓ What do you think would happen if we put the fprintf expression inside the loop, before the end???

✓ Every time the loop works, fprintf runs and prints the factorial value to the screen at each stage.

Loops / for loop

- Wouldn't it be wrong if the user entered a negative value in the previous example?
- Of course, it would be wrong. Negative numbers do not have a factorial. So, let's fix the codes.

```
%This program calculates the factorial of entered
%number.
num=input('Enter a number:');
f=1;
if num>0
for i=1:num
    f=f*i;
end
fprintf('%d! is equal to %d. \n',num,f);
elseif num==0
       fprintf('%d! is equal to 1. \n',num);
else
      fprintf('Please enter a positive number \n');
end
```


```
Command Window
>> Untitled10
Enter a number:7
7! is equal to 5040.
>> Untitled10
Enter a number:-5
Please enter a positive number
```

Loops / `for` loop

QUESTION: How do you set up a loop to create the following numbers?

```
0
50
100
150
200
```

✓ SOLUTION:

```
for i=0:50:200
    disp(i);
end
```

Loops / `for` loop

QUESTION: Let's ask the user to enter a number three times and print a message like 'The number you entered is x' after each number entry.

Editor - C:\Users\N.Furkan\Desktop\Matlab Uygulama\Untitled4.m

Untitled4.m

```
1 for i=1:3
2     num=input('Enter a number:');
3     fprintf('The entered number is %d. \n',num);
4 end
```

Command Window

```
>> Untitled4
Enter a number:3
The entered number is 3.
Enter a number:4
The entered number is 4.
Enter a number:7
The entered number is 7.
```

Loops / `for` loop

QUESTION: If we run the program below and enter 3, then 6, 8, 12 respectively, what would be the result?

Editor - C:\Users\N.Furkan\Desktop\Matlab Uygulama\Untitled4.m

Untitled4.m

```
1 —    n=input('How many loops do you want? ');
2 —    s=0;
3 —    for i=1:n
4 —        n=input('Enter a number:');
5 —        s=s+n;
6 —    end
7 —        fprintf('The result is %.2f \n',s);
```

Command Window

```
>> Untitled4
How many loops do you want? 3
Enter a number:6
Enter a number:8
Enter a number:12
The result is 26.00
```

Loops / `for` loop

QUESTION: In the previous factorial program, can the user enter a number again when a negative value is entered? How do you think we should proceed?

Next week

Chapter 7

LOOPS 2

CHAPTER 7

LOOP STRUCTURES-2

Loops / Nested `for` loop

- If there is another loop instead of the operations to be performed in a loop, it is called a nested loop.
- The general syntax for the nested for loop is:

✓ **Outer loop:** The outer loop includes the inner loop and operations1.

✓ The outer loop does not start a new loop until the inner loop has finished the entire loops.

✓ So for i=1, the inner loop runs from j=1 to m, completing the number of loops. Then i=2, the inner loop runs again from j=1 to m.

```
for i=1:n
        operations1
        for i=1:n
                operations2
        end
end
```

✓ **Inner loop:** The inner loop contains operations2.

✓ The outer loop does not proceed to the next round until the inner loop has completed the number of loops.

Just like in the for loop:

✓ The i and j value do not have to start from 1.

✓ Also, the increment of i and j can be changed.

✓ `i=3:2:11` → `i=3,5,7,9,11`

Loops / Nested `for` loop

- Let's make an example to understand how the nested for loop works.

```
for i=1:3
    fprintf('%d. loop for outer loop. \n',i);
    for j=1:5
        fprintf('%d. loop for inner loop. \n',j);
    end
end
```


```
>> Untitled
1. loop for outer loop.
1. loop for inner loop.
2. loop for inner loop.
3. loop for inner loop.
4. loop for inner loop.
5. loop for inner loop.
2. loop for outer loop.
1. loop for inner loop.
2. loop for inner loop.
3. loop for inner loop.
4. loop for inner loop.
5. loop for inner loop.
3. loop for outer loop.
1. loop for inner loop.
2. loop for inner loop.
3. loop for inner loop.
4. loop for inner loop.
5. loop for inner loop.
```

For outer loop that is i=1; inner loop i.e. worked from j=1 to 5. So, when the outer loop worked 1 time, the inner loop worked 5 times.

This happened every time the outer loop ran.

Loops / Nested `for` loop

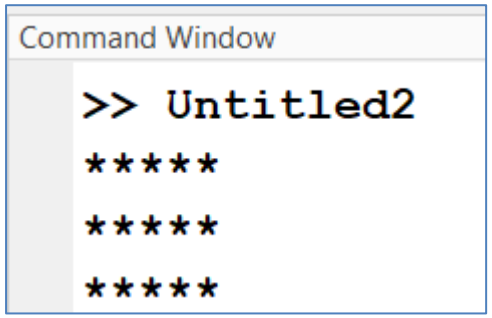
- Let's create the following pattern using a nested for loop. Let's not forget that we will print 1 * in each loop.

```
*****
*****
*****
```

✓ Let's examine the pattern first:

- There are 3 rows and each row has 5 *.
- We know that the inner loop is running first. Then let's print 5 * with the inner loop. Let's run the inner loop at 1:5.
- İçteki döngünün * yazma işi bittiğinde, yani döngü bittiğinde, alt satıra geçmesi gerekiyor ki tekrar 5 adet * yazabilsin.
- Then when the inner loop has finished writing *, let the outer loop do the work of moving to the bottom line. Let's run the outer loop at 1:3.

```
for i=1:3
    for j=1:5
        fprintf('*'); %The inner loop will
                      %run 5 times and write *
                      %side by side to the screen.
    end               %The inner loop is over
    fprintf('\n');    %The outer loop does the job
                      %of going to the bottom line
end
```



Loops / Nested `for` loop

- Let's create the following pattern using a nested for loop. Let's not forget that we will print 1 * in each loop.

```
*
**
***
****
*****
```

- ✓ Let's examine the pattern first:
- There are 5 lines and line 1 has 1, line 2 has 2, …, line 5 has 5 *.
- As in the previous example, let's make the * print job done by the inner loop, and the next line job by the outer loop.
- When i=1, let the inner loop run once, when i=2, let the inner loop run twice… etc.
- So let's check the inner loop's upper limit with the outer loop's current loop number.

```
for i=1:5
    for j=1:i
        fprintf('*'); %The inner loop will
                      %run 1 times and write * for i=1,
                      %run 2 times and writes ** for i=2
                      %...
    end               %The inner loop is over
    fprintf('\n');    % The outer loop does the job of
                      %going to the next line
end
```

Command Window

```
>> Untitled2
*
**
***
****
*****
```

Loops / Nested `for` loop

- Let's print the loop numbers of the previous example on the screen and examine them in more detail.

```
for i=1:5
     fprintf('i=%d ',i);
    for j=1:i
        fprintf('j=%d ',j);
    end
    fprintf('\n');
end
```

Command Window

```
>> Untitled2
i=1 j=1
i=2 j=1 j=2
i=3 j=1 j=2 j=3
i=4 j=1 j=2 j=3 j=4
i=5 j=1 j=2 j=3 j=4 j=5
```

Loops / Nested `for` loop

- Let's prepare a script that calculates the summation of the positive elements in each row in the matrix A and prints it on the screen.

```
A =

    33   -11     2
     4     5     9
    22     5    -7
     2    11     3
```

We need to make summation in this order:

A(1,1)+ A(1,2)+ A(1,3)

A(2,1)+ A(2,2)+ A(2,3)

A(3,1)+ A(3,2)+ A(3,3)

A(4,1)+ A(4,2)+ A(4,3)

Attention! While the row number is fixed, the column number is changing. Just as we did in the previous examples, while the outer loop count was kept constant, the inner loop was spinning fast.

So, the variable named **row** controls the outer loop, while the variable named **column** controls the inner loop.

Loops / Nested `for` loop

- Let's prepare a script that calculates the summation of the positive elements in each row in the matrix A and prints it on the screen.

```
A =

    33   -11     2
     4     5     9
    22     5    -7
     2    11     3
```

```
A=[33 -11 2; 4 5 9; 22 5 -7; 2 11 3];
[row, column]=size(A);
for i=1:row
    sum=0;
    for j=1:column
        if A(i,j)>=0
            sum=sum+A(i,j);
        end
    end
    fprintf('The sum of %d row is %d
\n',i,sum);
end
```

Command Window

```
>> Untitled3
The sum of 1 row is 35
The sum of 2 row is 18
The sum of 3 row is 27
The sum of 4 row is 16
```

Loops / `while` loop

- `while` statement is used as a conditional loop in MATLAB; It is used to repeat an action when it is not known how many times the action will be repeated.
- The general syntax for the nested for loop is:

```
while condition statement
      operations
      end
```

✓ The loop will run as long as the given condition expression is met.

Loops / while loop

- Let's write a function called **fact(n)** that calculates the factorial using the while statement. In this function, the value of *n* will be the input value of the function and the function will calculate the factorial of the *n*.

```
function f=fact(n)
i=1;
f=1;
while i<=n
    f=f*i;
    i=i+1;
end
end
```

Command Window

```
>> fact(3)

ans =

     6
```

- ✓ n!=1*2*3*…*n . Thus, the loop will run between 1 and n.
- ✓ In each loop, f variable will be multiplied with i and reassigned.

Loops / while loop

- To use multiple control statement in **while** loop, the || and && expression are used between the control statements.
- For example, if we want to run the loop when x variable is between 0 and 100. `while x >= 0 && x <= 100`
- Or, if we want to run the loop when x variable is less than 50 or y is less than 100. `while x < 50||y < 100`
- Let's ask for input by using **while** loop:
  - Let's make the program ask a new input until the user enters a negative number.

```
x=input('Enter x:');
y=input('Enter y:');

while x<50 || y<100

x=input('Enter x again:');
y=input('Enter y again:');

end
disp('End.');
```


```
Command Window
>> Untitled5
Enter x:55
Enter y:80
Enter x again:45
Enter y again:105
Enter x again:40
Enter y again:90
Enter x again:55
Enter y again:105
End.
```

Loops / while loop

- Let's ask for input by using **while** loop:
  - Make the program ask a new input until the user enters a negative number.

```
num=input('Enter a positive number:');
while num>0
num=input('Enter a positive number:');
end
disp('Alright.');
```


```
Command Window
>> Untitled5
Enter a positive number:12
Enter a positive number:20
Enter a positive number:15
Enter a positive number:-8
Alright.
```

Loops / while loop

- Let's store the numbers entered in the previous example in a vector A. This can be useful to be able to use the entered numbers later.

```
A=[ ]; %define an empty vector
index=1; %indicates the position of
element in the vector A
num=input('Enter a positive number:');
while num>=0
    A(index)=num;
    num=input('Enter a positive
number:');
    index=index+1;
end
disp('Alright. Here is the vector A:');
disp(A);
```


```
Command Window
>> Untitled5
Enter a positive number:5
Enter a positive number:8
Enter a positive number:15
Enter a positive number:20
Enter a positive number:-5
Alright. Here is the vector A:
     5     8    15    20
```

Loops / while loop

- The counter can be used in the while loop. That is, we can count how many times the loop is looped. In other words, we can determine how many times positive numbers were entered in the previous program.

```
counter=0;
num=input('Enter a positive number:');
while num>0
num=input('Enter a positive number:');
counter=counter+1;
end
disp('Alright.');
fprintf('Number of positive number:%d \n',counter);
```

Command Window

```
>> Untitled5
Enter a positive number:5
Enter a positive number:7
Enter a positive number:8
Enter a positive number:10
Enter a positive number:200
Enter a positive number:-8
Alright.
Number of positive number:5
```

Next week

Chapter 8

GRAPHIC OPERATIONS IN MATLAB