



THE OHIO STATE UNIVERSITY

COLLEGE OF ENGINEERING

Class 20: For Loops



Today's Learning Objectives

Students will be able to:

1. The basic structure of a for-end loop.
2. Use loops for array access and creation.
3. Conditional statements in for loops.



Today's Topics

1. `for` loops
2. Using loops to manipulate arrays
3. Conditionals with `for` loops



Loops – Overview

A loop allows a group of commands in a program to be repeated.

- Each repetition of the loop is called a pass
- The loop terminates:
 - After a fixed number of passes (for loop) OR
 - After some condition is satisfied (while loop)



for Loop – Syntax

Loop index
variable

Each pass, **k** takes on the
next value in this vector

```
for k = values
```

```
    . . . . .
```

```
    . . . . .
```

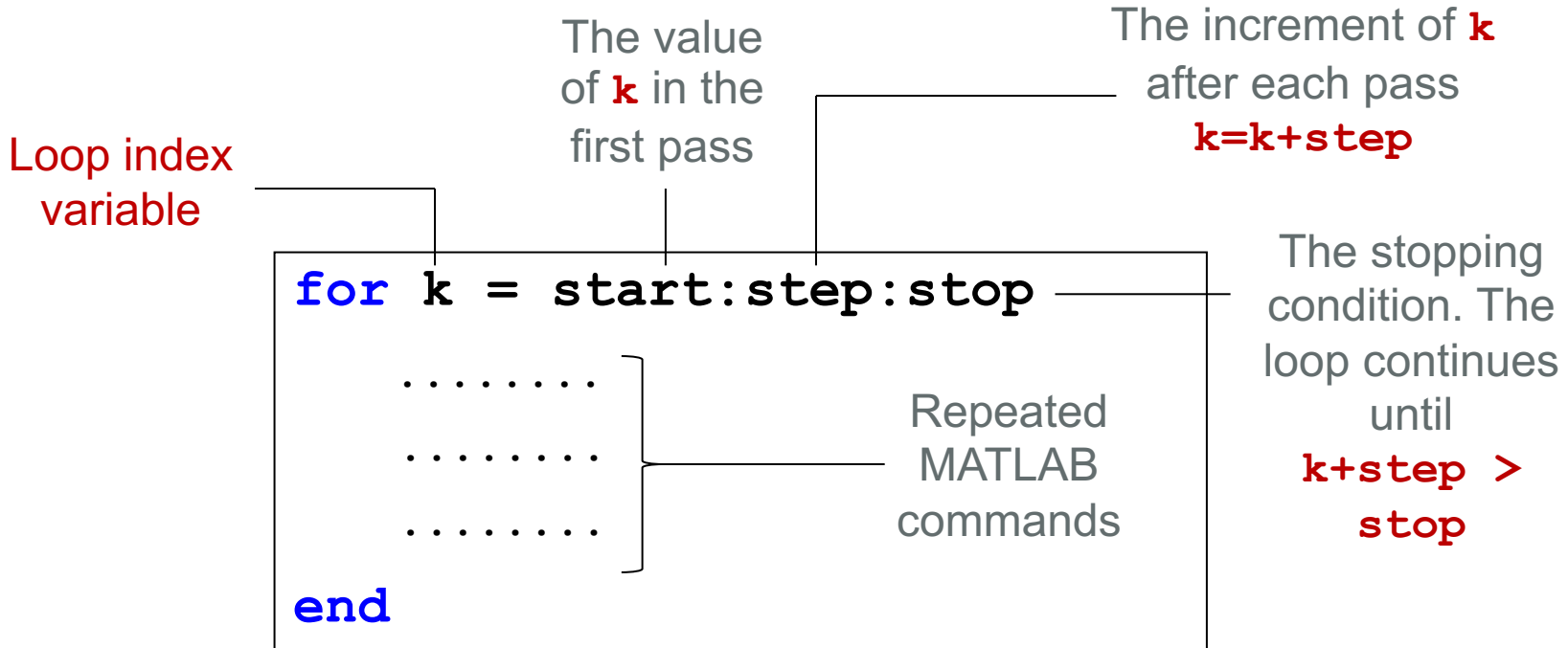
```
    . . . . .
```

```
end
```

Repeated
MATLAB
commands



for Loop – Typical Structure



Read: “loop the following `for` `k` equals `start` incrementing by `step` until it's `stop`”



for Loop – Example

```
for k = 1 : 1 : 5  
    k  
end
```

The loop will repeat itself until the loop index variable, **k**, equals the last value, 5, for a total of 5 passes. **k** equals 5 after the loop ends.

Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
k=1	k=2 (=1+1)	k=3 (=2+1)	k=4 (=3+1)	k=5 (=4+1)



for Loop – Indexing elements of a vector

The loop variable is often used for indexing vectors.

Example: Display each element of a vector.

```
v = [2 4 6 8];  
for index = 1:length(v)  
    v(index)  
end
```

Output:

```
ans =  
    2  
ans =  
    4  
ans =  
    6  
ans =  
    8
```



for Loop – Summing Example

Indexing a vector lets us do calculations on the elements one-by-one. Example: summing

- The sum variable must be initialized to zero *before* the loop.
- MATLAB needs to be reminded to start counting at 0 when it calculates a sum.

```
v = [2 4 6 8];  
sumv = 0;  
for index = 1:length(v)  
    sumv = sumv + v(index)  
end
```



for Loop – Summing Example

Indexing a vector lets us do calculations on the elements one-by-one. Example: summing

```
v = [2 4 6 8];  
sumv = 0;  
for index = 1:length(v)  
    sumv = sumv + v(index)  
end
```

Output:

```
sumv =  
      2  
sumv =  
      6  
sumv =  
     12  
sumv =  
     20
```



for Loop – Vector Creation

for loops can be used to create vectors element-by-element.

Example:

- Loop variable **n** as the index for vector **vec**
- Each time the loop **n** increases by 1, it creates the next element of vector **vec**

```
for n=1:4  
    vec(n)=3*n  
end
```

The final pass will return:

```
vec =  
     3     6     9    12
```



Break – Exit a Loop structure

- Break command exits a loop.
- Any statements after the break will not be executed.

```
...  
dog = zeros(4,1);  
for bone = 1 : 1 : 4  
    disp(bone);  
    if bone == 3  
        break;  
    end  
end  
...
```

```
1  
2  
3  
>>
```



Making a Vector: Six Ways

Square Brackets

```
V=[2 4 6 8]
```

load from a file

```
V=load('V.txt')
```

Grow with indexing

```
V(1)=2
```

```
V(2)=4
```

```
V(3)=6
```

```
V(4)=8
```

Colon notation
or `linspace()`

```
V=2:2:8
```

OR

```
V=linspace(2,8,4)
```

Array Math

```
x=1:4
```

```
V=2*x
```

for loop

```
for k=1:4
```

```
    V(k)=2*k
```

```
end
```



Demo 2

Create the following vector using a **for** loop

$$V = [0 \ 0.5 \ 1 \ 1.5 \ 2]$$

Recall:

Solution:

```
for k = 1:5
    V(k) = 0.5*(k-1)
end
```

Square Brackets

$V = [2 \ 4 \ 6 \ 8]$

```
for loop
for k=1:4
    V(k)=2*k
end
```



Conditionals with `for` Loops

Conditional statements can be used within `for` loops

- Any MATLAB code can be repeated with `for` loops, including `if` statements
- Useful in many applications such as sorting through data or displaying statements to the screen
- There is no limit to the number of conditional statements that can be used this way



Conditionals with for Loops

Example Problem:

- Given the vector $\mathbf{v} = [2 \ 4 \ 6 \ 8]$, use a loop to calculate the square of each element.
- If the square is less than 40, display a message saying:
“The square is __, which is < 40.”
- If the square is greater than 40, display a message saying:
“The square is __, which is > 40.”



Conditionals with for Loops

Example Program:

```
v=[2 4 6 8];  
for k=1:4  
    vs=v(k)^2;  
    if vs<40  
        fprintf('\n The square is %i, which is < 40.', vs)  
    elseif vs>40  
        fprintf('\n The square is %i, which is > 40.', vs)  
    end  
end
```



Conditionals with for Loops

Example Program Output:

```
The square is 4, which is < 40.  
The square is 16, which is < 40.  
The square is 36, which is < 40.  
The square is 64, which is > 40.
```



Demo 3: Ask user to input a vector and count the number of positive numbers in it.

Solution:

```
V = input('Input a vector:')  
count = 0;  
for i = 1:length(V)  
    if V(i) > 0  
        count = count+1;  
    end  
end
```

Hint: you will need to create a variable to keep track: each time a positive number is found, the “counter” goes up by one

```
fprintf('There are %i positive numbers in the vector\n',count)
```



Important Takeaways

- The index is the position of each element inside the array or matrix.
- Starts at 1 (and not 0) for arrays and (1,1) for matrices.
- **for** loops repeat the command for the determined number of iterations.
- Counter for the **for** loop iterations is typically associated with the index of an array.
- **if** statements can be used inside **for** loops



Preview of Next Classes

- While Loops



Appendix: Additional For Loops

Content



Nested for Loops

A **for** loop can be nested with another loop:

- Useful when creating and/or analyzing matrices
- The loop index variable for the first loop addresses the row, and the loop index variable for the inner loop addresses each column (or *vice versa*)
- No limit to the number of loops that can be nested



Nested for Loops

Example:

```
for k = 1 : 3
    for n = 1 : 4
        .....
        commands
        .....
    end
end
```

Every time **k** is increased by 1, the nested loop loops 4 times with the value of **n** ranging from 1-4.

```
k = 1 n = 1
k = 1 n = 2
k = 1 n = 3
k = 1 n = 4

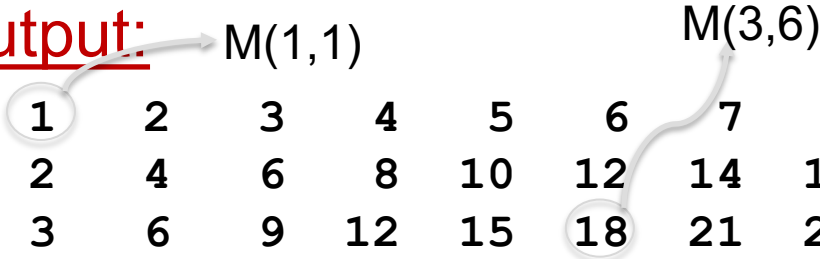
k = 2 n = 1
k = 2 n = 2
k = 2 n = 3
k = 2 n = 4

k = 3 n = 1
k = 3 n = 2
k = 3 n = 3
k = 3 n = 4
```



Demo 4: Make a 10 by 10 multiplication table matrix using two nested for loops

Example output:



M =	1	2	3	4	5	6	7	8	9	10
	2	4	6	8	10	12	14	16	18	20
	3	6	9	12	15	18	21	24	27	30
	4	8	12	16	20	24	28	32	36	40
	5	10	15	20	25	30	35	40	45	50
	6	12	18	24	30	36	42	48	54	60
	7	14	21	28	35	42	49	56	63	70
	8	16	24	32	40	48	56	64	72	80
	9	18	27	36	45	54	63	72	81	90
	10	20	30	40	50	60	70	80	90	100



Demo 4: Make a 10 by 10 multiplication table matrix using two nested for loops

Solution:

```
M = [];  
for r=1:10  
    for c=1:10  
        M(r,c) = r*c;  
    end  
end  
M
```