THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

ENG 101 M#06
Matlab Loops

Dr. Emil Jovanov
Electrical and Computer Engineering
University of Alabama in Huntsville
emil.jovanov@uah.edu
http://www.ece.uah.edu/~jovanov

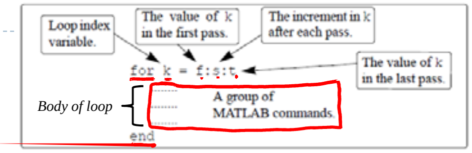

Loop index variable.
The value of k in the first pass.
The increment in k after each pass.
The value of k in the last pass.
for k = f:s:t
Body of loop
A group of MATLAB commands.
end

Figure 6-5: The structure of a for-end loop.

1. Loop sets k to f, and executes commands between for and the end commands, i.e., executes body of loop
2. Loop sets k to f+s, executes body
3. Process repeats itself until k > t
4. Program then continues with commands that follow end command

- f and t are usually integers
- s usually omitted. If so, loop uses increment of 1

```
s=0;
for i=2:2:10
  s=s+i;
end
s
```

i 2
2+2=4
4+2=6
6+2=8
8+2=10
~~10+2=12~~

2
4
6
8
10

```
for i=1:3

end
```

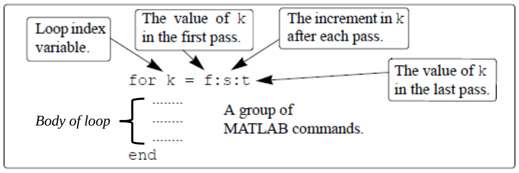

Loop index variable.
The value of k in the first pass.
The increment in k after each pass.
for k = f:s:t
The value of k in the last pass.
Body of loop
....
....
....
A group of MATLAB commands.
end

Figure 6-5: The structure of a `for-end` loop.

- Increment `s` can be negative
  - For example, `k = 25:-5:10` produces four passes with `k = 25, 20, 15, 10`
- If `f = t`, loop executes once
- If `f > t` and `s > 0`, or if `f < t and s < 0`, loop not executed

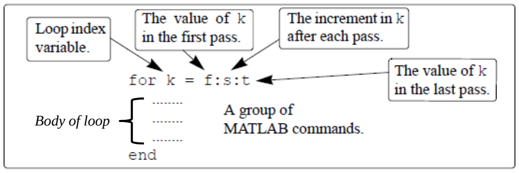

Loop index variable.
The value of k in the first pass.
The increment in k after each pass.
for k = f:s:t
The value of k in the last pass.
Body of loop
....
....
....
A group of MATLAB commands.
end

Figure 6-5: The structure of a `for-end` loop.

- If values of `k`, `s`, and `t` are such that `k` cannot be equal to `t`, then
  - If `s` positive, last pass is one where `k` has largest value smaller than `t`
    - For example, `k = 8:10:50` produces five passes with `k = 8, 18, 28, 38, 48`
  - If `s` is negative, last pass is one where `k` has smallest value larger than `t`

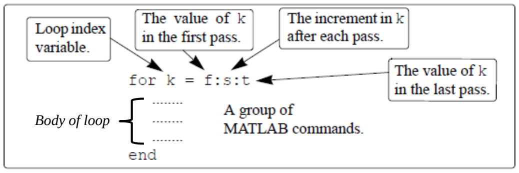

Loop index variable.
The value of k in the first pass.
The increment in k after each pass.
The value of k in the last pass.
for k = f:s:t
Body of loop
....
....
....
A group of MATLAB commands.
end

Figure 6-5: The structure of a `for-end` loop.

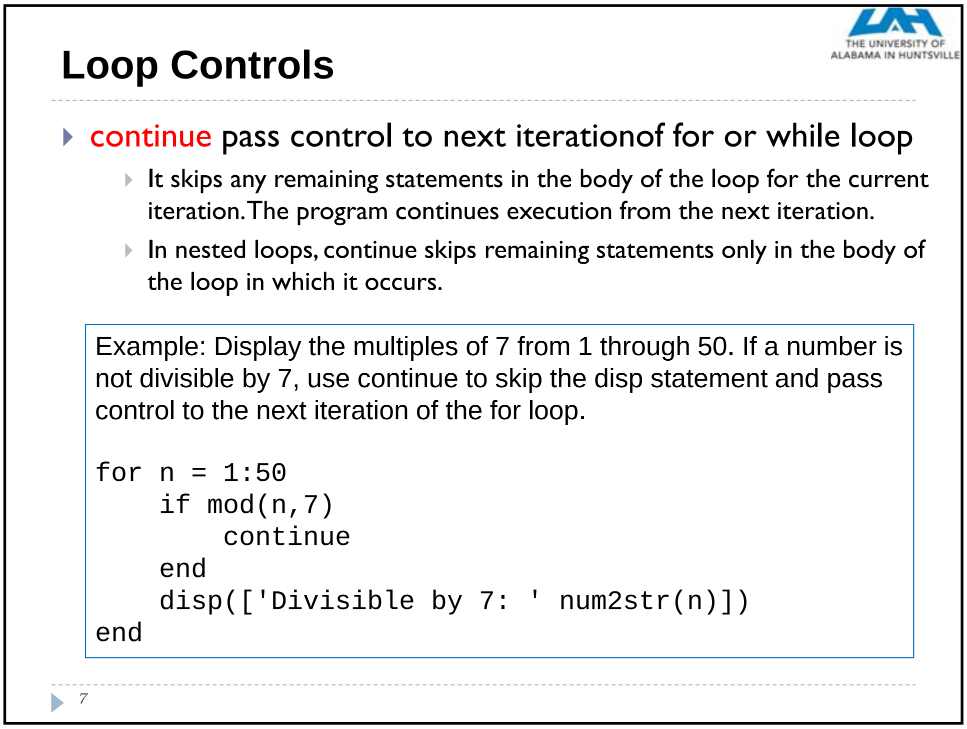
- Value of loop index variable (k) not displayed automatically
  - Can display value in each pass (sometimes useful for debugging) by typing k as one of commands in loop
- When loop ends, loop index variable (k) has value last assigned to it

Loop Controls

- Break terminates the execution of a for or while loop
  - Statements in the loop after the break statement do not execute.
  - In nested loops, break exits only from the loop in which it occurs. Control passes to the statement that follows the end of that loop.

Example: Sum a sequence of random numbers until the next random number is greater than an upper limit. Then, exit the loop using a *break* statement

```
limit = 0.8; s = 0;
while 1
  tmp = rand;
  if tmp > limit
    break
  end
  s = s + tmp;
end
```

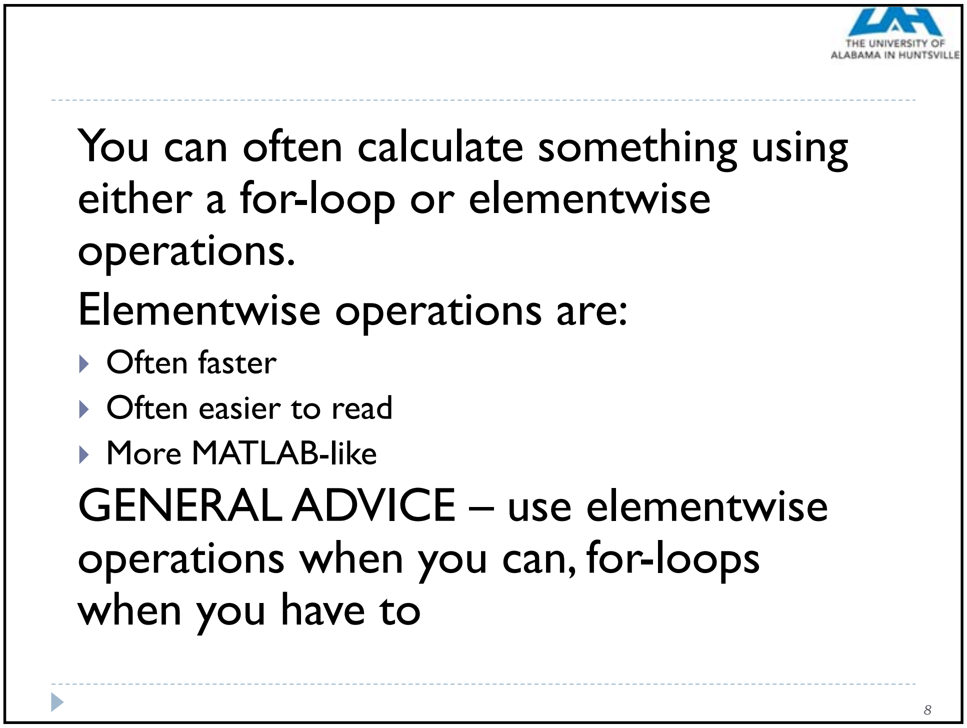


Loop Controls

- continue pass control to next iterationof for or while loop
  - It skips any remaining statements in the body of the loop for the current iteration.The program continues execution from the next iteration.
  - In nested loops, continue skips remaining statements only in the body of the loop in which it occurs.

Example: Display the multiples of 7 from 1 through 50. If a number is not divisible by 7, use continue to skip the disp statement and pass control to the next iteration of the for loop.

```
for n = 1:50
    if mod(n,7)
        continue
    end
    disp(['Divisible by 7: ' num2str(n)])
end
```

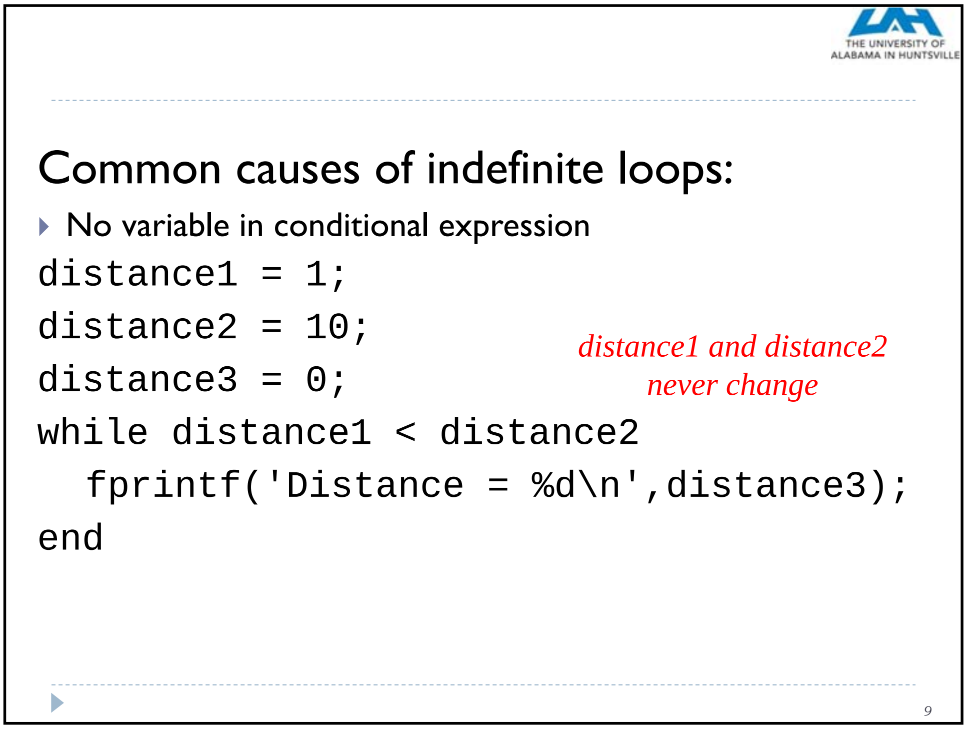


You can often calculate something using either a for-loop or elementwise operations.

Elementwise operations are:

- Often faster
- Often easier to read
- More MATLAB-like

GENERAL ADVICE – use elementwise operations when you can, for-loops when you have to

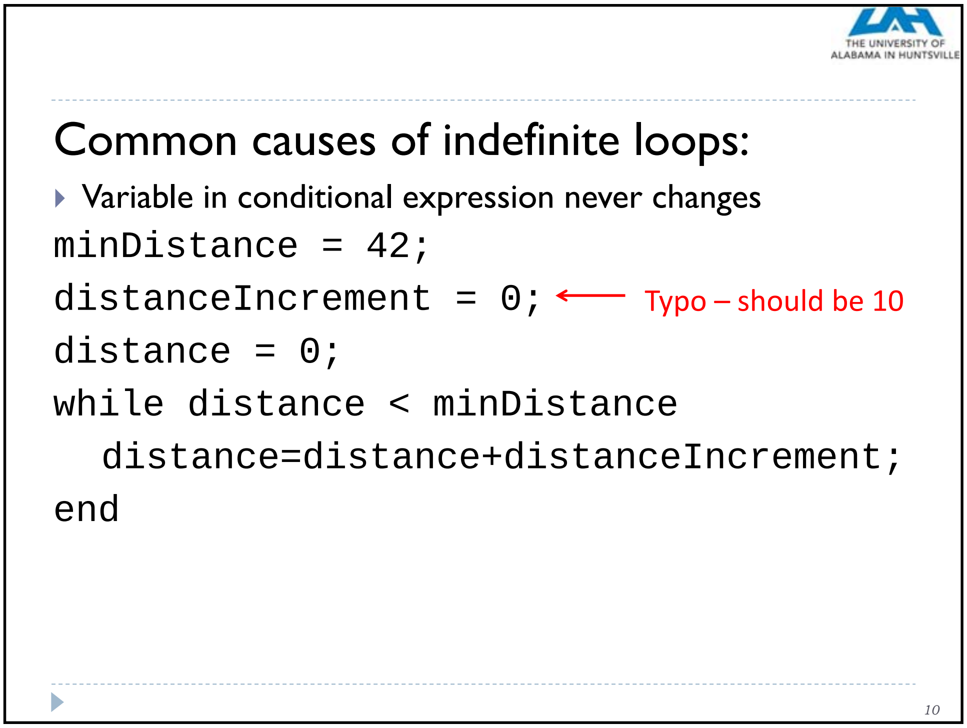


Common causes of indefinite loops:

- No variable in conditional expression

```
distance1 = 1;
distance2 = 10;
distance3 = 0;
while distance1 < distance2
  fprintf('Distance = %d\n',distance3);
end
```

distance1 and distance2 never change

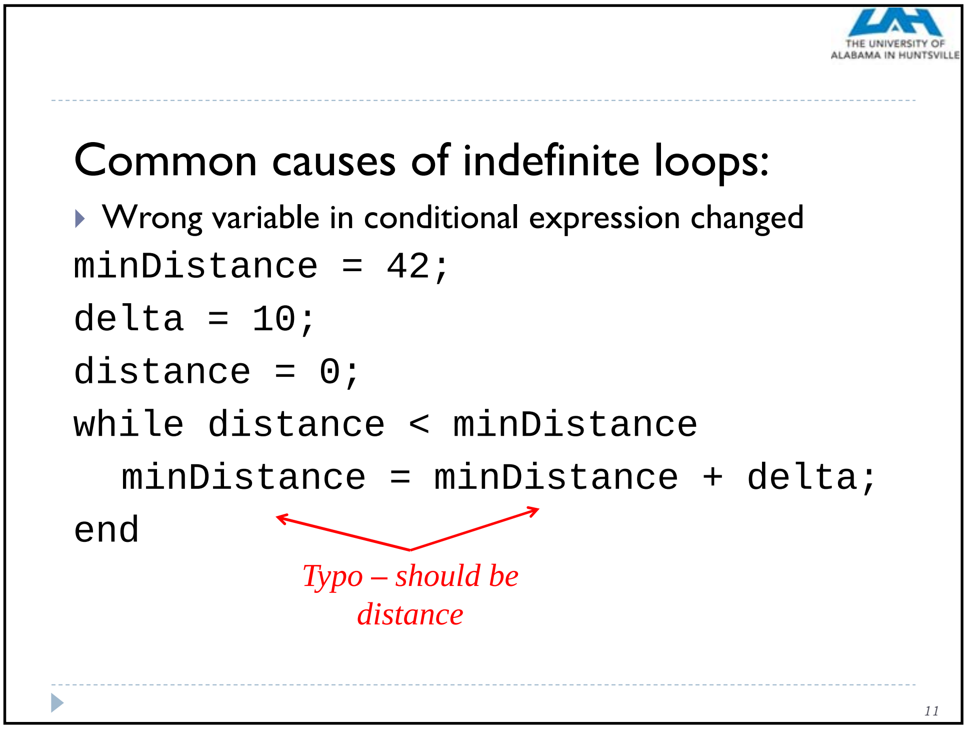


Common causes of indefinite loops:

- Variable in conditional expression never changes

```
minDistance = 42;
distanceIncrement = 0;
distance = 0;
while distance < minDistance
  distance=distance+distanceIncrement;
end
```

← Typo – should be 10

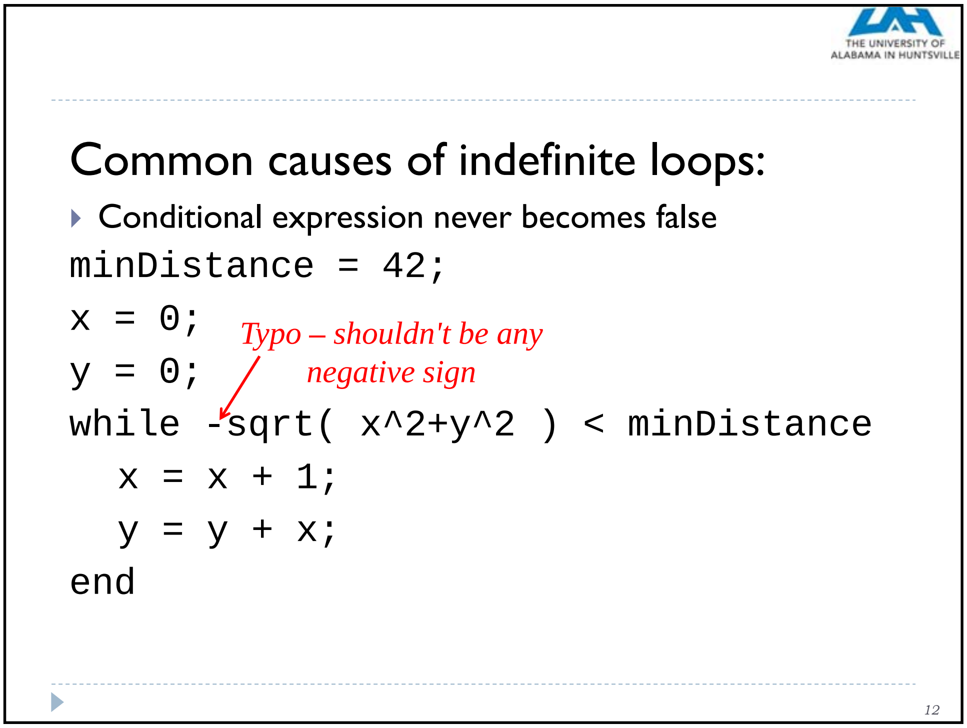


Common causes of indefinite loops:

- Wrong variable in conditional expression changed

```
minDistance = 42;
delta = 10;
distance = 0;
while distance < minDistance
  minDistance = minDistance + delta;
end
```

Typo – should be distance



Common causes of indefinite loops:

- Conditional expression never becomes false

```
minDistance = 42;
x = 0;
y = 0;
while -sqrt( x^2+y^2 ) < minDistance
  x = x + 1;
  y = y + x;
end
```

Typo – shouldn't be any negative sign

If your program gets caught in an indefinite loop,

- Put the cursor in the Command Window
- Press CTRL+C

If a loop or conditional statement is placed inside another loop or conditional statement, the former are said to be *nested* in the latter.

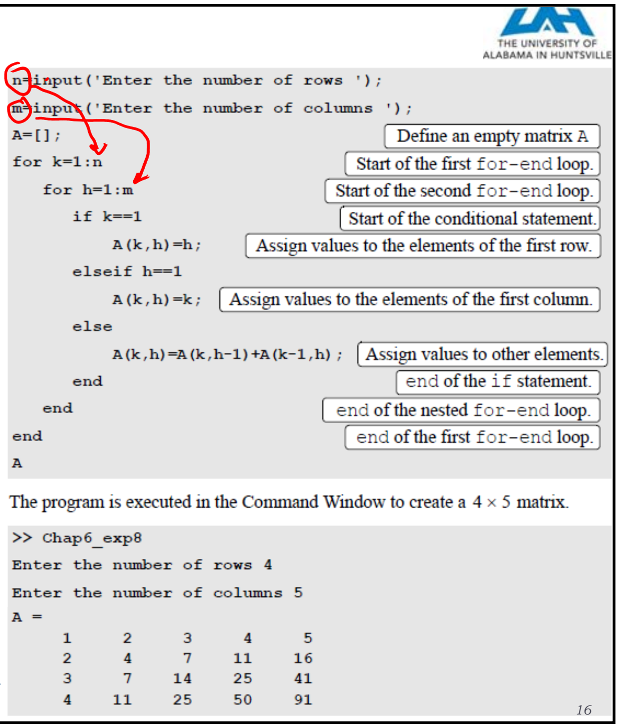
- Most common to hear of a *nested loop*, i.e., a loop within a loop
  - Often occur when working with two-dimensional problems
- Each loop and conditional statement must have an `end` statement

Loop example

- Create a 2 dimensional matrix with **n** rows and **m** columns
- Ask user for the number of rows and columns
- Elements in the first row will represent number of the column
- First elements in every row will represent row number
- All other elements will be equal to the sum of elements in
  - the same row and previous column, and
  - the same column and previous row

15

EXAMPLE



```
n=input('Enter the number of rows ');
m=input('Enter the number of columns ');
A=[];
for k=1:n
    for h=1:m
        if k==1
             A(k,h)=h;
        elseif h==1
             A(k,h)=k;
        else
             A(k,h)=A(k,h-1)+A(k-1,h);
        end
    end
end
A
```

Define an empty matrix A

Start of the first for-end loop.

Start of the second for-end loop.

Start of the conditional statement.

Assign values to the elements of the first row.

Assign values to the elements of the first column.

Assign values to other elements.

end of the if statement.

end of the nested for-end loop.

end of the first for-end loop.

The program is executed in the Command Window to create a 4 × 5 matrix.

```
>> Chap6_exp8
Enter the number of rows 4
Enter the number of columns 5
A =
      1     2     3     4     5
      2     4     7    11    16
      3     7    14    25    41
      4    11    25    50    91
```

- Elements in the first row will represent # of the column
- First elements in every row will represent row #
- All other elements will be equal to the sum of elements in
  - the same row & previous col
  - the same col & previous row

16