For Loops 2, While Loops

ENGR 1181
MATLAB 9/10
Today's Learning Objectives

- After today’s class, students will be able to:
  - Use more complex ways of setting the loop index.
  - Construct nested loops in the following situations:
    - For use with two dimensional arrays
    - For repeating a calculation numerous times
  - Use loops to repeat a code with conditional statements.
Today's Learning Objectives

- After today’s class, students will be able to:
  - Explain the proper application of while loops.
  - Use external inputs (e.g., tic, toc) for real time programming.
What is a Loop?

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

- Each repetition of the loop is called a pass.

- *Either* the number of passes can be fixed - typical for loop

- *Or* the loop can be terminated after some condition is satisfied - typical While loop
Review: For Loops

```matlab
for k=1:3:10
    x=k^2
end
```

- Every for must have an end
- Loops can be used in the command window or in script files
- Loops can be nested (loops inside of other loops!)
- if statements can be used inside of loops... and loops can be used inside if statements
Summing with For Loops

- If you use a for loop to calculate a sum, 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.

- Remember to avoid using MATLAB function names as variable names!

```matlab
sum1=0;
for k=2:2:10
    sum1=sum1+k;
end
disp(sum1)
```

```
sum1= 30
```
For Loops With If Statements

- Conditional statements can be used within for loops
- Just like nesting for loops, if statements and for loops can also be nested
- Useful in many applications such as sorting data or displaying statements to the screen
The Continue Command

- The ‘continue’ command will stop the remainder of the current loop pass.
- The program continues with the next iteration of the loop.
- Often used with if statements: If a certain condition is satisfied, the pass can be stopped.
- See prep material for an example of ‘continue’.
- Type ‘help continue’ in command window for more information.
The Break Command

- The ‘break’ command will terminate a current loop.
- MATLAB will jump to the end and carry on with the program.
- Often used with if statements: If a certain condition is satisfied, the loop can be terminated.
- See prep material for an example with ‘break’.
- Type ‘help break’ in command window for more information.
Function: ‘tic’ and ‘toc’

- tic and toc are timing functions that monitor elapsed time in your program.
- tic starts a stopwatch that runs in the background.
- toc returns the elapsed time since the last tic command.
- The value of toc can be assigned to a variable.
  - Ex: time1 = toc;
If-elseif-end with Strings

- Use `strcmp` for comparing strings in conditional statements

- The command `a=='text'` will only work if `a` and `text` are of same length!
About While Loops

- **While** loops run as long as a conditional expression is true.
- Conditional variables must be "initialized" before the loop.
- The conditional variable must advance with each pass. (Otherwise the loop will continue indefinitely!)
- ‘Ctrl + C’ will terminate execution of an indefinite loop.
- Like for loops, must have an end.
Example: Simple While Loop

```
a=0
while a<10
    a=a+1
    if (a==5)
        continue
    end
    disp(a)
end
```

Note:
- conditional variable is initialized
- variable advances each time
- one end corresponds to if
- one end corresponds to while
Activity- For loops with if-else-end

- Given the vector \( v = [2 \ 4 \ 6 \ 8 \ 10] \), use a loop to create a new array with the square of each element.

- Check for the following conditions and display message:
  - If square <30, display ‘The square is ... and is less than 30’
  - If 30<square<60, use continue command to skip display
  - If square >60, use break to exit the loop

- Display the new array outside the loop
Activity Solution

\[ v = [2 \ 4 \ 6 \ 8 \ 10]; \]

\[ \text{for } k = 1:4 \]

\[ \quad \text{vs}(k) = v(k)^2; \]

\[ \quad \text{if } \text{vs}(k) < 30 \]

\[ \quad \quad \text{fprintf('The square is } \%i, \text{ which is less than 30.'}, \text{ vs}(k)) \]

\[ \quad \text{elseif } (\text{vs}(k) > 30 \ & \ \text{vs}(k) < 60) \]

\[ \quad \quad \text{continue} \]

\[ \quad \text{else} \]

\[ \quad \quad \text{break} \]

\[ \quad \text{end} \]

\[ \text{end} \]

\[ \text{disp(vs)} \]
Activity: Complex While Loop

- **Task**
  
  Write a program that will use a while loop to find the smallest number divisible by both 3 and 5, with the square of the number greater than 3325.

  **Hint:** Use `rem` command

- **Steps**
  
  1. Initialize variables: `sq` & `n`
  2. Loop while `sq < 3325`
  3. Advance `n` by 3 in each pass
  4. Check if `n` is divisible by 5
  5. If yes, square the `n`
  6. Repeat loop until `sq > 3325`
  7. Loop stops & `n` value will give us the smallest integer to meet conditions
Activity Solution: Complex While Loops

n=0; sq=0;    %initialize variables: # & square
while sq<=3325  %loop until square >3325
    n=n+3  %advance # by 3 each pass
    if rem(n,5)==0 %check if # is divisible by 5
        sq=n^2; %if yes, square #
    end
end  %end while loop

disp(n)  %display the # that meets criteria
Important Takeaways

- For loops and if statements can be used in combination by nesting.
- The ‘break’ command has MATLAB jump to the ‘end’ of the loop (breaks the loop).
- The ‘continue’ command has MATLAB jump to the end of the iteration (doesn’t break the loop).
- While loops need a conditional statement and conditional variables must be initialized.
- While loop will run indefinitely until condition is met.
- ‘rem()’, ’tic’, and ‘toc’ are very useful tools.