APMA 0160 Lecture 2

Control Flow & Programming Practice

Brown University, Summer 2017

Wednesday, June 28

Review

Consider the vector

```
v = [0 pi/4 pi/2 3*pi/4 pi 5*pi/4 3*pi/2 7*pi/4 2*pi]
```

How would you create this vector using `linspace` and using colon notation?

Review

Consider the vector

```
v = [0 pi/4 pi/2 3*pi/4 pi 5*pi/4 3*pi/2 7*pi/4 2*pi]
```

How would you create this vector using `linspace` and using colon notation?

```
v = 0:pi/4:2*pi %first:spacing:last
```

Review

Consider the vector

```
v = [0 pi/4 pi/2 3*pi/4 pi 5*pi/4 3*pi/2 7*pi/4 2*pi]
```

How would you create this vector using `linspace` and using colon notation?

```
v = 0:pi/4:2*pi %first:spacing:last
```

```
v = linspace(0,2*pi,9) %first, last, length
```

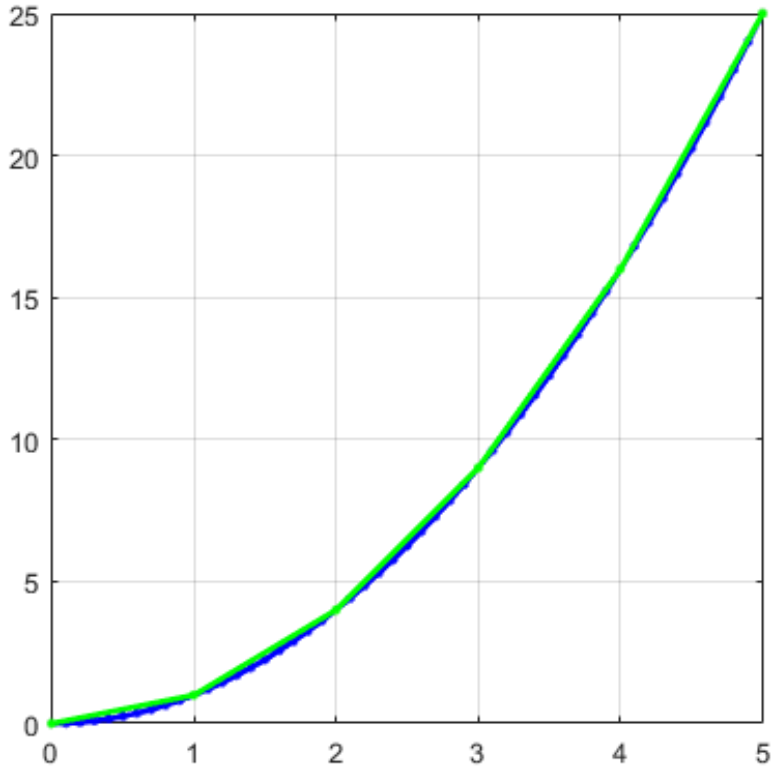
Plotting

A very useful application of vectors is to plot functions. You need two vectors of equal lengths:

MATLAB code

```
x = 0:5; y = x.^2;
plot(x,y,'g.-');

x = 0:0.1:5; y = x.^2;
plot(x,y,'b.-');
```



The only necessary part is `plot(x,y)`. The parameters that follow control the appearance of the plot.

Plotting

Ways to improve the appearance of your plots:

- Label your axes with `xlabel('name')` and `ylabel('name')`. Add a title wth `title('name')`.
- Change your marker width and/or color. More info on the MathWorks page.
- Control the line width, e.g. `'LineWidth',4`. The default is 1.
- Plot multiple functions on a single graph with `hold on`.
- Close all open plot windows with the `close all` command.
- Set the window with `axis([xmin xmax ymin ymax])`. Make the scale on the axes the same with `axis equal`.

Characters

- Characters are a way to represent text. In MATLAB, characters must be stored inside single quotes. Examples: `a = 'programming is fun'`, `b = 'hello world'`, `c = '12'`
- Later on, we will encounter functions that take in character arguments.
- If you want to print out a message, use the `disp` command.

MATLAB code

```
x = 200;
disp(x)
disp('x equals 200')
```

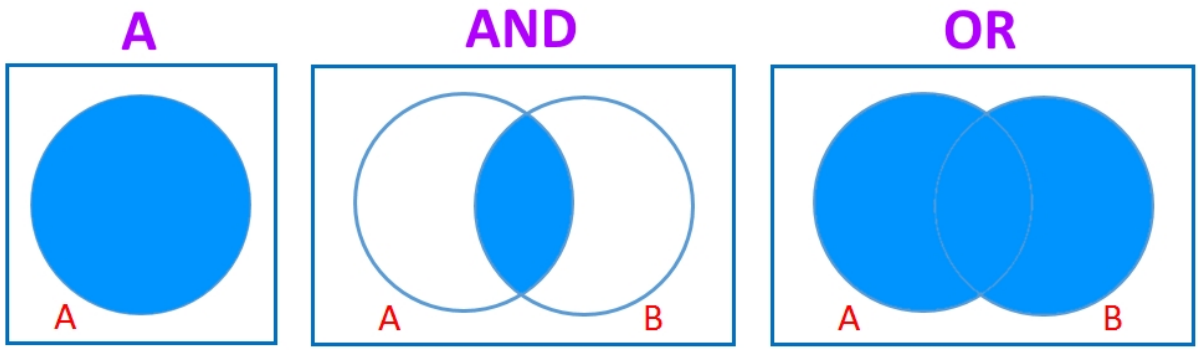
Booleans

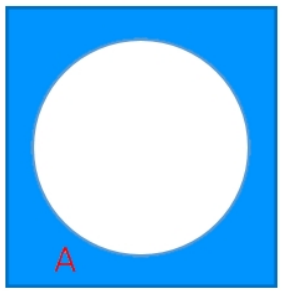
- Booleans are the result of a logical operation (e.g. `b > 5` or `isprime(3)`)
- Stored in MATLAB as either 1 (true) or 0 (false) rather than different types.
- Comparison operations in MATLAB include ==, ~=, <, <=, >, and >=. These all return booleans.
- Do not confuse the single equals with the double equals!

Boolean operations

What operations can be performed on a boolean?

- NOT (~): changes true into false and vice versa
- AND (&&): true if both statements are true, false otherwise
- OR (||) : true if at least one statement is true, false otherwise





a	b	not ~b	and a&&b	or a || b
0	0	1	0	0
0	1	0	0	1
1	0		0	1
1	1		1	1

Boolean operations

Booleans also have an order of operations: first NOT, then AND, then OR.

Simplify the following boolean expressions:

- `true && (2 > 4) || (6 < 7)`
- `false || ~ true && true`
- `5^2 == 25  || 4 == 4  && ~ 3 == 4`

if statements

As the name implies, this will execute a block of code only if the condition is determined to be true.

```
if condition
    %insert statements here
end
```

As an example, here is a script that warns the user if they are taking the squareroot of a negative number.

MATLAB code

```
if b < 0
    disp('Warning: b is negative')
end
if b >= 0
    c = sqrt(b);
end
```

&& is equivalent to nested ifs

Compare the following blocks of code

MATLAB code

```
if x > 0
    if y > 0
        z = sqrt(x) + sqrt(y);
    end
end
```

MATLAB code

```
if x > 0 && y > 0
    z = sqrt(x) + sqrt(y);
end
```

Convince yourself that these are functionally equivalent.

elseif and else

These are additions to an `if` statement that can be used to cover other cases.

```
if condition
elseif condition
        %insert statements here
elseif condition
        %insert statements here
...
else
    %insert statements here
end
```

`elseif` and `else` must follow an if statement but are not required in general.
Protip: Try to keep your indentation consistent.

elseif and else

Example:

MATLAB code

```
x = 4;

if x == 1
    disp('x is neither prime nor composite')
elseif isprime(x)
    disp('x is prime')
else
    disp('x is composite')
end
```

elseif and else

What's the issue with this code?

MATLAB code

```
x = 15;
if x > 0
    disp('x is positive')
elseif x > 10
    disp('x large')
else
    disp('x is nonpositive')
end
```

elseif and else

Compare the previous slide to:

MATLAB code

```
x = 15;
if x > 0
    disp('x is positive')
    if x > 10
        disp('x large')
    end
else
    disp('x is nonpositive')
end
```

for loops

- for loops are a way of iterating a calculation for each element in a vector

```
for index = vector
    %insert statements here
end
```

- The statements in the loop are repeated over and over again until the end of the vector is reached.

- for loops must be closed with end. Again, pay attention to indentation.

for loops

- An example that prints out each number in a vector:

MATLAB code

```
for v = 0:0.2:1
   disp(v)
end
```

- Once the end of the statement block is reached, control returns back to the top of the for loop. The commands in the loop is then repeated for the next element of the vector.

`for` loops

- In elementary school in the late 1700's, Gauss was asked to find the sum of the numbers from 1 to 100. The question was assigned as "busy work" by the teacher, but Gauss found the answer rather quickly by discovering a pattern.
- How would we use a loop to sum the first 100 positive integers?

for loops

- In elementary school in the late 1700's, Gauss was asked to find the sum of the numbers from 1 to 100. The question was assigned as "busy work" by the teacher, but Gauss found the answer rather quickly by discovering a pattern.
- How would we use a loop to sum the first 100 positive integers?

MATLAB code

```
s = 0
for v = 1:100
   s = s + v;
end
```

Double `for` loops

- It is possible to have a loop inside of another loop. This occurs often when dealing with matrices.
- What does the following code do?

MATLAB code

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
s = 0;
for j = 1:3
    for k = 1:3
        s = s + A(j,k);
    end
end
```

while loops

- A more general loop that runs as long as a condition is evaluated to be true.

```
while condition
    %insert statements here
end
```

- At each loop, the computer checks whether `condition` is true or false. If it is true, the commands inside the loop get executed. If not, the contents of the loop are skipped.
- This is useful for getting the computer to iterate something UNTIL a condition is met (or not met).
- Must be closed with `end`.

while loops

Here is an example that calculates a familiar operation

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Do you recognize what `f` is?

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f
0	10	10

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f
0	10	10
1	9	10*9

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f
0	10	10
1	9	10*9
2	8	10*9*8

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f
0	10	10
1	9	10*9
2	8	10*9*8
3	7	10*9*8*7

while loops

MATLAB code

```
n = 10;
f = n;
while n > 1
    n = n - 1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Iteration	n	f
0	10	10
1	9	10*9
2	8	10*9*8
3	7	10*9*8*7
⋮	⋮	⋮
9	1	10!

Infinite loops

- What happens if the termination condition for a while loop is never false? Example:

```
a = 0;
b = 0;
while b < 5
    a = a + 1
end
```

- A common beginner's mistake is to forget updating the counter, thus creating an infinite loop.
- Protip: Ctrl + C will kill a running MATLAB program.

while loops

Find the first number that is divisible by 36 and 28

while loops

Find the first number that is divisible by 36 and 28
Ask yourself: what condition should be fulfilled?

while loops

Find the first number that is divisible by 36 and 28
Ask yourself: what condition should be fulfilled?
The loop should run as long as the condition is NOT fulfilled

while loops

Find the first number that is divisible by 36 and 28
Ask yourself: what condition should be fulfilled?
The loop should run as long as the condition is NOT fulfilled
Are we update variables? Do we have a way for the loop to end?

while loops

Find the first number that is divisible by 36 and 28
Ask yourself: what condition should be fulfilled?
The loop should run as long as the condition is NOT fulfilled
Are we update variables? Do we have a way for the loop to end?

MATLAB code

```
p = 1;
while mod(p,36) ~= 0 || mod(p,28) ~= 0
%equivalent: while ~(mod(p,36) == 0 && mod(p,28) == 0)
    p = p + 1;
end
disp(p)
```

`for` loops vs. `while` loops

What is the difference between the two kinds of loops? When is one better than the other?

- `while` loops are more general (any for loop can be rewritten as a while loop).
- `for` loops are useful when you want to loop over items in a vector.

Other loop keywords

- `break` will exit out of a loop. It usually goes inside of an `if` statement.

MATLAB code

```
for n = 1:10000
    if abs(1/n) < 0.001
        break
    end
end
disp(n)
```

- `continue` exits the current iteration of a loop and proceeds to the next one.

MATLAB code

```
for n = [-3 -2 -1 0 1 2 3]
    if n == 0
        continue
    else
        disp(1/n);
    end
end
```

Functions in MATLAB

It is often useful to be able to define your own functions.
The signature of a function tells the user what the input, output, and name are.

For example

```
function [area,volume] = cylinder(height,radius)
```

Each function must have

- The keyword `function`
- The inputs (on the left)
- The outputs (on the right)
- An `end`

Function example

MATLAB code

```
%input: a vector x
%output: the mean of the elements of x
function y = average(x)
if ~isvector(x)
    error('Input must be a vector')
end
y = sum(x)/length(x);
end
```

Once you declare that y is being returned, you must define y somewhere! Don't forget to close a function with end.

Good coding suggestions

- Comment code (using the percent sign). At the top of the file, write your name and a brief description of what the code does.
- Keep code indented. MATLAB does not care about spacing/alignment, so use this to your advantage.
- Give variables obvious, descriptive names. Try to be consistent when naming variables.
- Avoid extraneous variables, extraneous printing, and other slow processes.
- Avoid hardcoding whenever possible. Use helper functions to keep things neat.

A ball is launched straight up into the air with a velocity v_0 from a starting height of s_0. Assuming there is no air resistance and that gravitational acceleration is 9.8 $\mathrm{m/s^2}$ downwards, write a MATLAB function `t = freefall(v0,s0)` that determines how much time `t` the ball spends in air before it hits the ground (reaches a height of zero). Assume v_0 and s_0 are both nonnegative.

Now let's try some exercises!

In groups of 2, work through the practice problems on Canvas.