MATH2221
Mathematics Laboratory II

Lecture 3: Functions, Logical Operators, and Conditional Statements

Gary Choi

January 21, 2025

Recall: Basic MATLAB functions

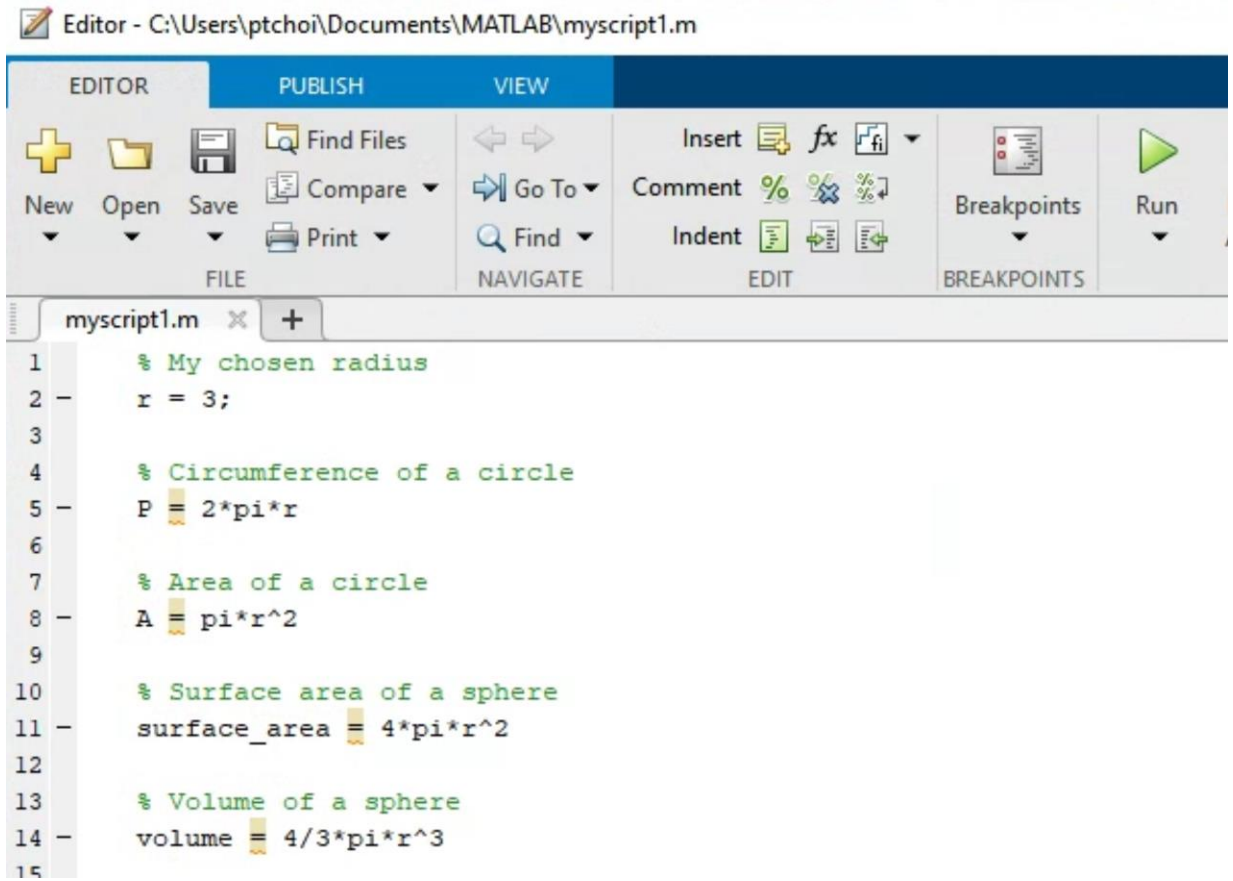
- **Vector and matrix commands**
  - 1:5, 2:2:10, (20:-2.1:1)', …
  - zeros(2,3), ones(1,4), eye(3), diag([1,2,3]), triu(A), tril(A), [A; B], …
  - Entrywise operations A.*B, A./B, A.^B, …

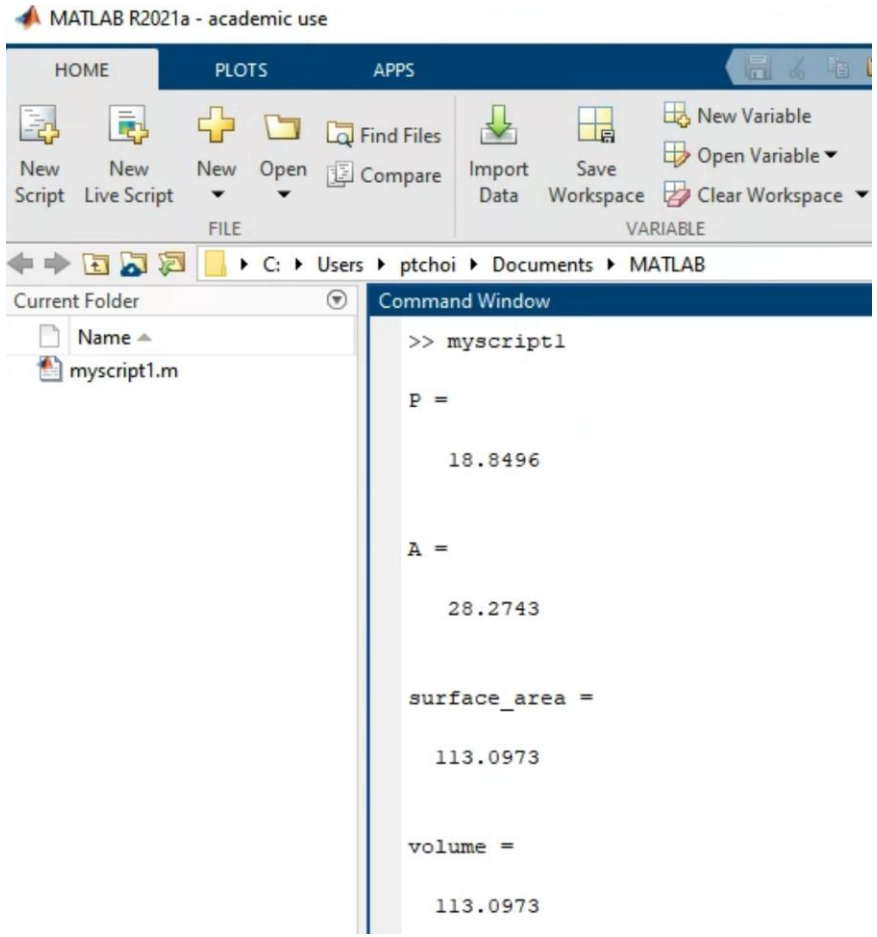
- **Matrix indexing**
  - A = [1, 2, 3, 4; 5, 6, 7, 8];
  - A(2,3), A(2,1:3), A(1,:), A(2, 2:end), A(end,1), …

- **Solving matrix equation:**
  - x = A\b

Recall: MATLAB scripts

- Writing MATLAB scripts
  - A file with a .m extension
  - Can execute a series of MATLAB statements





More built-in MATLAB commands for vectors and matrices

Description	MATLAB Command	Example
Generate linearly spaced vector	linspace(x1,x2,N)	linspace(1,8,10)
Reshape a vector or matrix	reshape(A,m,n)	A = 1:10; B = reshape(A,5,2)
Flip array left to right	fliplr	fliplr([1, 2, 3; 4, 5, 6])
Flip array up to down	flipud	flipud([1, 2, 3; 4, 5, 6])
Sort array elements	sort	sort([1, 3, 4, 2])
Average value of array	mean	mean(1:10)
Median value of array	median	median([1, 3, 4, 5, 9, 11])
Most frequent values in array	mode	mode([1, 1, 2, 3])
Standard deviation	std	std([1, 9, 7, 6, 3])

More built-in MATLAB commands for vectors and matrices

```
>> linspace(1,5,10)
ans =
   1.0000   1.4444   1.8889   2.3333   2.7778   3.2222   3.6667   4.1111   4.5556   5.0000
```

Generate 10 equally spaced points between 1 and 5

```
>> A = 1:12;
>> B = reshape(A, 3,4)
B =
    1     4     7    10
    2     5     8    11
    3     6     9    12
```

Note: For reshape(A,m,n), m*n must match the total number of elements in A.
The reshaped matrix is formed by taking entries from A in a columnwise manner.

```
>> C = reshape(A,4,3)'
C =
    1     2     3     4
    5     6     7     8
    9    10    11    12
```

If we want to have rearrange the elements in a row-wise manner, suitably apply the transpose operation

Writing your own MATLAB function

- You can create your own MATLAB function by writing a script in the form:

```
function [y1,...,yn] = myfun(x1,...,xm)
    % Statement1
    % Statement2
    %  …
    % Statementn
end
```

- The keyword "function" is necessary and cannot be changed
- The function name and the script file name will be myfun (can be changed)
- x1, …, xm are the function inputs (optional)
- y1, …, yn are the function outputs (optional)
- With one output, brackets are optional: function y = myfun(x1,...,xm)
- With no outputs, omit the equal sign: function myfun(x1,...,xm)
- With no inputs, parentheses are optional: function [y1,...,yn] = myfun

Writing your own MATLAB function

- Example:

```
function [x1,x2] = quadratic_formula(a,b,c)
    x1 = (-b + sqrt(b^2-4*a*c))/(2*a);
    x2 = (-b - sqrt(b^2-4*a*c))/(2*a);
end
```

First write a MATLAB script with these contents and then save the file

```
>> [x1,x2] = quadratic_formula(1,3,2)
x1 =
   -1
x2 =
   -2
```

We can then call the function in our code

Writing your own MATLAB function

- Example:

```
function A = areaCircle(R)
    rSquared = R.^2;
    A = pi.*rSquared;
end
```

First write a MATLAB script with these contents and then save the file

```
>> A = areaCircle(3)
A =
  28.2743
```

We can then call the function in our code

```
>> rSquared
Unrecognized function or variable
'rSquared'.
```

Note:
All variables defined inside the function will be local, i.e., they will not appear in the workspace outside the function

Writing your own MATLAB function

- You may also create a function handle using @

- Example:

```
>> f = @(x,y) 2*x+3*y;                    % the function name is f
                                            and the inputs are x,y
>> f(1,2)
ans =
    8
```

- Example:

```
>> g = @(x) (2*x.^2 – 3*x + 1./x);        % note the use of .^ and ./ here
>> g([1, 2, 3])                             to allow g to handle vectors
ans =
        0    2.5000    9.3333
```

Writing your own MATLAB function

- You may also create a function handle using @

- Example:

```
>> R = @(t) [cos(t), -sin(t); sin(t), cos(t)];      % rotation counterclockwise by t
>> Q = [1, 0; 0, -1];                               % reflection about x-axis
>> u = [1; 2];
>> v = R(-pi/6)*Q*R(pi/6)*u                         % first rotate u counterclockwise by pi/6
                                                    % then apply a reflection
                                                    % then rotate u clockwise by pi/6

v =
   -1.2321
   -1.8660
```

Writing your own MATLAB function

- Comparison between the two approaches:

  - Creating a function by writing a script with
    function output_parameters = myfun(input_parameters):
    - Easier to handle more complicated tasks inside the function (e.g. loops, conditional statements, calling other functions, …)
    - Easier to be reused in other functions and scripts

  - Creating a function handle using @:
    - Simpler procedure
    - Harder to solve more complicated tasks
    - Only defined and used within the current script

Relational operators

- Relational operators compare two numbers/arrays in a comparison statement and return a logical value or an array of logical values with
  - Logical value 1 (true)
  - Logical value 0 (false)

Description	MATLAB Command
Equal to (The = character is for assignment, whereas the == character is for comparing the elements in two arrays.)	==
Not equal to	~=
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=

Relational operators

```
>> 3>1
ans =
  logical
   1
```

Return a logical value 1 (true) since 3 > 1

```
>> a = (1+1==3)
a =
  logical
   0
```

Return a logical value 0 (false) since 1+1 is not equal to 3, store the logical value as a variable a

```
>> b = ([1,3,5] ~= 1)
b =
  1×3 logical array
   0   1   1
```

Return [0, 1, 1] because:
1 ~= 1 is false
3 ~= 1 is true
5 ~= 1 is true

Relational operators

```
>> c = (8<9)+(2>3)+(2*2==16/4)
c =
    2
```

8 < 9 is true (1)
2 > 3 is false (0)
(2*2==16/4) is true (1)
Therefore, we have 1+0+1 = 2

```
>> A = 'apple';
```

In MATLAB we can also define a string of characters as a variable (we need the single quote ‘ ’)

```
>> A == 'p'
ans =
  1×5 logical array
   0   1   1   0   0
```

In this case, the == operator will compare the character one by one

```
>>strcmp(A,'p')
ans =
  logical
   0
```

To compare the two strings as a whole, use strcmp

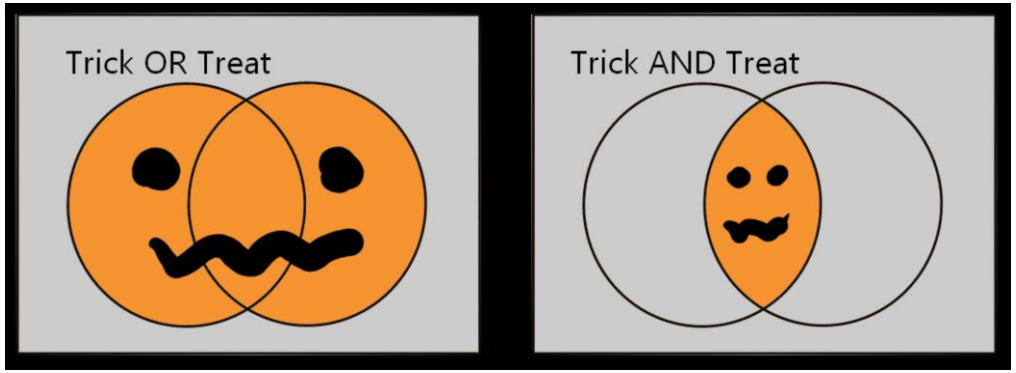
Logical operators

- The following logical (Boolean) operators return a logical value or an array of logical values with
  - Logical value 1 (true)
  - Logical value 0 (false)

Description	Command
AND	&
OR	|
NOT	~
Short-circuit AND For A && B, MATLAB does not evaluate condition B at all if condition A is false	&&
Short-circuit OR For A || B, MATLAB does not evaluate condition B at all if condition A is true	||

x	y	x && y	x || y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

x	~x
0	1
1	0



Logical operators

```
>> 3 & 9
ans =
 1
```

3 and 9 are both true (non-zero), so the result is 1.

```
>> a = 5 | 0
a =
 1
```

Assign “5 or 0” to variable a. Since at least one of the numbers is true (non-zero), we have a = 1

```
>> x = -2;
>> -7<x<-1
ans =
  0
```

Mathematically correct, but the answer is false since MATLAB executes from left to right. -7<x is true (=1) and then 1<-1 is false (0).

```
>> -7<x & x<-1
ans =
 1
```

-7<x is true (=1) and x<-1 is true (=1), so the result is 1.

Relational and logical operators for matrix indexing

- Example:

```
>> A = 1:100;
>> A(mod(A,3)==0 & mod(A,4) == 1)
ans =
  9   21   33   45   57   69   81   93
```

Select all entries of A divisible by 3 and with remainder 1 when divided by 4

- Example:

```
>> B = [2, 3, 5, 8; 2, 0, 2, 5; 0, 1, 2, 1];
>> B(B>=5) = 1;
>> B
B =
    2    3    1    1
    2    0    2    1
    0    1    2    1
```

Find all entries of B with value >= 5, then replace those entries with 1

The find operator

- The find function finds indices of nonzero elements.
  - When combined with relational/logical operators, it can easily return the indices of the elements satisfying certain conditions

- Example:

```
>> y = [2, 3, 5, 5, 4, 10, 5, 20];
>> a = find(y==5)
a =
    3    4    7
```

Return the indices of the entries in y with value = 5

- Example:

```
>> E = [7 3 9; 1 0 2; 5 8 4];
>> [a,b] = find(E>5 & E < 9);
>> [a,b]
ans =
    1    1
    3    2
```

Return the row and column indices of the entries in E with value > 5 and < 9

(row 1, column 1)
(row 3, column 2)

More on MATLAB data types

Data types	Description	Example
double	The default numeric type in MATLAB, array of double-precision numbers	1.2, pi, [1,2,3], …
single	Array of single-precision numbers, requires less storage space than double but has less precision and a smaller range	y = single(10)
int8, int16, int32, int64, uint8, uint16, uint32, uint64	Array of signed and unsigned integers, require less storage space	x = int16(325)
logical	Array of logical values of 1 or 0 to represent true or false	1 > 0, a == 2, …
Function handle	A data type that represents a function	f = @(x) 2*x+1;
Characters and strings	Text in character arrays and string arrays	'a', 'hello', '123'
Cell arrays (covered later)	Arrays that can contain data of varying types and sizes	c = { [1,3], 'abc'}
Structures (covered later)	Arrays with named fields that can contain data of varying types and sizes	A.age = 30; A.health = 'good';
… and many more!	https://www.mathworks.com/help/matlab/data-types.html	

Use the whos command for more information about the size, bytes, class etc. of all the variables in the current workspace

if, elseif, else statements

- **if-end statement:**

```
if expression
    statements                    % do this if the expression is true
end
```

- Example:

```
a = 3*4;
b = 2*6;
if a == b
  disp('a and b are equal');      % disp: display text/string/array
end
```

- Common mistake: Incorrectly using = (instead of ==) for testing equality

if, elseif, else statements

- **if-else-end statement:**

```
if expression
    statements1                % do this if the expression is true
else
    statements2                % do this if the expression is false
end
```

- Example:

```
a = 10;
b = 7;
if a > b
   disp('a is greater than b');
else
   disp('a is not greater than b');
end
```

if, elseif, else statements

- **if-elseif-else-end statement:**

```
if expression1
    statements1                % do this if expression1 is true

elseif expression2
    statements2                % do this if expression1 is false
                                   but expression2 is true
elseif expression 3
    statement3                 % do this if expression1 and expression 2 are false
                                   but expression3 is true
else
    statementsN                 % do this if all the above expressions are false

end
```

if, elseif, else statements

- Example:

```
num_goal_A = 5;                          % Number of goals by Team A
num_goal_B = 2;                          % Number of goals by Team B

if num_goal_A > num_goal_B               % Team A gets 3 points, Team B gets 0
   score_A = 3;
   score_B = 0;

elseif num_goal_A < num_goal_B           % Team A gets 0 points, Team B gets 3
   score_A = 0;
   score_B = 3;

else                                     % Each team gets 1 point
   score_A = 1;
   score_B = 1;
end
```

if, elseif, else statements

- Example: Comparing the values of A, B, C

```
A = 1; B = 4; C = 3;

if A <= B && B <= C
    disp('A <= B <= C');
elseif  A <= C && C <= B
    disp('A <= C <= B');
elseif B <= A && A <= C
    disp('B <= A <= C');
elseif B <= C && C <= A
    disp('B <= C <= A');
elseif C <= A && A <= B
    disp('C <= A <= B');
elseif C <= B && B <= A
    disp('C <= B <= A');
end
```

if, elseif, else statements

- Example: HK income tax

  Consider the following HK income tax scheme.

Net chargeable income	Progressive rate
on the first $50,000	2%
on the next $50,000	6%
on the next $50,000	10%
on the next $50,000	14%
on the remainder	17%

  Exercise: How to write a function income_tax(N) to compute the income tax for any given net chargeable income N?

if, elseif, else statements

- Solution:

```
function tax = income_tax(N)

if N <= 50000
    tax = N*0.02;
elseif N <= 50000*2
    tax = 50000*0.02 + (N-50000)*0.06;
elseif N <= 50000*3
    tax = 50000*0.02 + 50000*0.06 + (N-2*50000)*0.10;
elseif N <= 50000*4
    tax = 50000*0.02 + 50000*0.06 + 50000*0.10 + (N-3*50000)*0.14;
else
    tax = 50000*0.02 + 50000*0.06 + 50000*0.10 + 50000*0.14 + (N-4*50000)*0.17;
end

end
```

Switch case statement

- **Switch case statement:**

```
switch switch_expression

   case case_expression1
      statement1

   case case_expression2
      statement2
    ...
   otherwise
      statementN

end
```

Similar to if-elseif-else-end:

```
if switch_expression == case_expression1 (for scalar)
or
if strcmp(switch_expression,case_expression1) (for string)

elseif switch_expression == case_expression2 (for scalar)
or
elseif strcmp(switch_expression,case_expression2) (for string)

else
      statementN

end
```

Switch case statement

- Example:

```
day = 'Tue';

switch day
    case 'Tue'
        disp('MATH2221 Lecture');

    case 'Thu'
        disp('MATH2221 Lab');

    otherwise
        disp('Missing MATH2221');

end
```

```
if strcmp(day, 'Tue')

elseif strcmp(day, 'Thu')

else
```

Reminder: Lab 2 this week, no lecture/lab next week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1

Lab 2 onwards:

- Will involve both written part (lab assignment worksheet) and coding part (submit a zip file containing all the required code files to Blackboard)

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17		

Lecture 1- Lecture 13

Lab 1 - Lab 10 (40%)

Test 1 (30%)
Test 2 (30%)

Thank you!

Next time:

- For loop
- While loop
- Recursion