Principles of Technical Computing

Starting with MATLAB

Matylda Jabłońska-Sabuka

Lappeenranta University of Technology

Overview of Week 1

- Matlab's working environment
- Basic math, scalar variables
- Character arrays
- Displaying results Command Window (disp, fprintf)
- Relational operators
- Conditional statements: if-else-end
- The for loop
- The while loop

Matlab environment

Working in the Command Window

- Place the cursor next to the command prompt (>>)
- Pressing Enter executes the command
- More than one command can be executed at once – separated with commas or semicolons
- Previously typed commands can be recalled one by one

Working in editor: Home → New Script

- Unlike command window history, scripts created in the Editor can be saved, and then opened and modified later on any other machine.
- Multiple commands can be typed, either in separate lines, or within one line separated with commas or semicolons.
- Command can be executed as follows:
    - by running the entire file (use mouse cursor to press run, or press F5 for Windows and Alt+Cmd+R for OS X) – when you try to run the entire file for the first time, you will have to save it,
    - by selecting a portion of commands and then pressing Evaluate Selection (F9 for Windows and Shift+F7 for OS X)

Rules for script file names

- must begin with a letter
- can contain letters, digits and the underscore character
- MATLAB is case sensitive, i.e. `a` and `A` are two different variables
- no spaces are allowed between the characters
- avoid using names of built-in functions (i.e. `sin`, `cos`, `exp`, etc.) or keywords (`break`, `case`, `catch`, `continue`, `else`, `elseif`, `end`, `for`, `function`, `global`, `if`, `otherwise`, `persistent`, `return`, `switch`, `try`, `while`)
- some predefined variables: `ans`, `pi`, `eps`, `inf`, `i`, `j`, `NaN`

'Tidying' and managing commands

- `clc` – clears the command window
- `clear x` – clears variable `x`
- `clear all` – clears all variables
- `close all` – closes all figure windows
- `who` – list of all used variables
- `whos` – list of all used variables together with their size and class

Variables and basic math in Matlab

Elementary arithmetic operations

Operation	Symbol	Precedence	Example
Addition	+	Fourth	5+3
Subtraction	-	Fourth	5-3
Multiplication	*	Third	5*3
Right division	/	Third	5/3
Left division	\	Third	5\3 = 3/5
Exponentiation	^	Second	5^3 (means $5^3 = 125$)
Parentheses	( )	First	3*(5-2)

Display formats

`short`, `long`, `short e`, `long e`, `bank`, *etc.*

Built-in functions

`sqrt(x)`	`nthroot(x,n)`	`exp(x)`	`abs(x)`
`log(x)`	`log10(x)`	`factorial(x)`	`perm(n)`
`sin(x)`	`sind(x)`	`cos(x)`	`cosd(x)`
`tan(x)`	`tand(x)`	`cot(x)`	`cotd(x)`
`round(x)`	`fix(x)`	`ceil(x)`	`floor(x)`

Use Help for more functions!

- By contents – list of Toolboxes
- By Index
- Search for key words
- Help has a lot of Demos

Rules for variable names

- must begin with a letter
- can contain letters, digits and the underscore character
- MATLAB is case sensitive, i.e. `a` and `A` are two different variables
- no spaces are allowed between the characters
- avoid using names of built-in functions (i.e. `sin`, `cos`, `exp`, etc.) or keywords (`break`, `case`, `catch`, `continue`, `else`, `elseif`, `end`, `for`, `function`, `global`, `if`, `otherwise`, `persistent`, `return`, `switch`, `try`, `while`)
- some predefined variables: `ans`, `pi`, `eps`, `inf`, `i`, `j`, `NaN`

Character arrays (text variables) and function `disp` in Matlab

Text strings

Variables in Matlab can also take textual values, as single characters or a string of characters. Value assignment using also the assignment operator =. However, the text has to be included in quotation.

```
x = 'Y';
y = 'ES';
```

Two strings can be connected using string concatenation `strcat`.

```
z = strcat('Y','ES');
z = strcat(x,y);
```

Elements of the string can be accessed one by one

```
y(2)
z(3)
```

Displaying results using command `disp`

All the numerical results from lines not ended with semi-colon operator are automatically displayed in Command Window.

There is a possibility to add explanatory text to the displayed results using the `disp` command.

```
disp('text')
```

Function `input` and more about displaying results in Matlab's Command Window

Argument input

Except for regular variable definition in the Command Window or the script file, one can avoid assigning specific value from the beginning and leave it up to the user.

Assume we want to calculate the average of points scored in three games. The points from each game are assigned to the variables by using the `input` command.

```
game1=input('Enter the points scored in game 1 ');
game2=input('Enter the points scored in game 2 ');
game3=input('Enter the points scored in game 3 ');
avg_points=(game1+game2+game3)/3
```

To run the code type the script file name in the Command Window and press Enter.

Displaying text and numerical results using `fprintf`

The output text can be displayed together with the numerical results using `fprintf` function.

```
fprintf('The average is :  %f',avg_points)
```

Additionally, `fprintf` creates a new line when seeing `\n` within the string.

Optionally, `%f` can be replaced by other arguments depending on the type of the variable whose value we display.

Logical statements and `if-else-end` construct in Matlab

Conditional structures

The relational operators for conditions are:

- == equal
- > grater
- < smaller
- ∼= not equal
- <= less or equal to
- >= greater or equal to

The logical operators for conditions are:

- & (sometimes &&) AND
- | (sometimes ||) OR

Recap of logic rules

S1	S2	AND
true	true	true
true	false	false
false	true	false
false	false	false

S1	S2	OR
true	true	true
true	false	true
false	true	true
false	false	false

S	NOT
true	false
false	true

In Matlab, any value other than zero is considered as `true` (logical 1), whereas exactly zero is treated as `false` (logical 0).

Simplest: `if - end`

```
   if <expression>
       <commands evaluated if True>
   end
```

An example follows; given a positive number of pens, a cost is computed and displayed.

```
if (pens >= 0)
   cost = pens*1.99;
   fprintf('Cost of %3.0f pens is $ %5.2f',pens,cost)
end
```

Intermediate: `if - else - end`

```
if <expression>
     <commands evaluated if True>
else <commands evaluated if False>
end
```

The example now prints a warning for negative numbers of pens.

```
if (pens >= 0)
     cost = pens*1.99;
     fprintf('Cost of %3.0f pens is $ %5.2f',pens,cost);
else disp('An invalid number of pens was specified.')
end
```

Most general: if - elseif - else - end

```
if <expression1>
     <commands evaluated if expression1 is True>
elseif <expression2>
     <commands evaluated if expression2 is True>
elseif <expression3>
     <commands evaluated if expression3 is True>
elseif <expression4>
     <commands evaluated if expression4 is True>
else <commands evaluated if all other expressions are False>
end
```

We aren't limited to 4 expressions here; we could add more. We could use this to price pens according to quantity, as follows.

```
sale = false;
if (pens >= 0) & (pens < 20)
      cost = pens;
      sale = true;
   elseif (pens >= 20) & (pens < 40)
      cost = pens*0.95;
      sale = true;
   elseif (pens >= 40) & (pens <= 100)
      cost = pens*0.90;
      sale = true;
   else
      disp('not a valid number of pens')
end
if sale
      fprintf('Cost of %3.0f pens is $ %5.2f \n',pens,cost)
end
```

The `for` loop in Matlab

for - end

The for loop repeats the commands that are inside of it. The basic structure is

```
for <list-of-values>
    <statements-to-be-done>
end
```

The list of values is usually numerical including vector of indices/steps for which the task should be done. Loop 'for' always has a finite number of evaluations. It is used to repeat output or do calculations.

Example

For each number from 5 to 31, calculate its multiplication by 2.

The general structure of loop 'for' is as follows

```
for n = 5:31
    disp(n*2)
end
```

The code will display each result one by one.

The `while` loop in Matlab

while - end

The while loop repeats the commands that are inside of it. The basic structure is

```
while <expression>
    <commands>
end
```

The <commands> inside the while loop will be repeated as long as the <expression> returns a value of True, provided that <expression> returns a scalar. If the <expression> evaluates to an array, then all of the values that result from it must be True for <commands> to be evaluated.

Example

How many times do I need to divide $x = 2$ by 3, to get a result equal less than 0.005?

```
clear all
x = 2;
d = 3;
n = 0;
thr = 0.005;
while (x>thr)
     x/d;
     n+1;
end
```

But why is Matlab forever busy?

Example

How many times do I need to divide $x = 2$ by 3, to get a result equal less than 0.005?

```
clear all
x = 2;
d = 3;
n = 0;
thr = 0.005;
while (x>thr)
     x = x/d;
     n = n+1;
end
```

The `switch - case` expression in Matlab

switch - case

```
switch switch_expr
   case case_expr
      statement, ..., statement
   case {case_expr1, case_expr2, case_expr3, ...}
      statement, ..., statement
   otherwise
      statement, ..., statement
end
```

The `switch_expr` can be a scalar or a string. A scalar `switch_expr` matches a `case_expr` if `switch_expr==case_expr`. A `case_expr` can include arithmetic or logical operators, but not relational operators such as < or >. To test for inequality, use `if - elseif` statements.

Example

```
switch w
    case 5
       disp('Go for a party')
    case {1,2,3,4}
       disp('Go to work')
    case {6,7}
       disp('Sleep late!')
    otherwise
       disp('There are only 7 days in a week...')
end
```

Example

Check whether the following formulas are trigonometric identities, that is, whether their left and right hand side values are equal for any arbitrarily chosen angles:

- $\sin^5\theta = \frac{10\sin\theta - 5\sin(3\theta) + \sin(5\theta)}{16}$
- $\cos\theta - \cos\phi = -2\sin\left(\frac{\theta+\phi}{2}\right)\sin\left(\frac{\theta-\phi}{2}\right)$

Example

Write some lines of code that will calculate the sums:

- $1 + 2 + 3 + ... + 20$
- $15 + 25 + 35 + ... + 4005$
- $2 \cdot 4 + 4 \cdot 6 + 6 \cdot 8 + ... + 98 \cdot 100$
- $1^2 + 2^3 + 3^4 + ... + 10^{11}$
- $\frac{1}{1+2} + \frac{2}{2+3} + \frac{3}{3+4} + ... + \frac{30}{30+31}$

Example

Write some lines of code that will figure out how many terms in the sum

$$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \dots$$

it requires for the sum to exceed one million.