ENGG1003 - Thursday Week 11

MATLAB vs. Python

Sarah Johnson and Steve Weller

University of Newcastle

20 May 2021

Last compiled: May 20, 2021 4:58pm +10:00

Lecture overview

1. Context
   - ENGG1003
   - what is MATLAB?
   - do we even need MATLAB?
2. MATLAB vs. Python
   - features and philosophy
   - key language details in MATLAB
3. Next steps
   - getting MATLAB, if you need it
   - Octave: free & mostly compatible with MATLAB

1) Context

- $\leq$ 2020, ENGG1003 used *MATLAB* and *C*
  - **from 2021, ENGG1003 uses Python only**
  - . . . yet some students will use MATLAB &/or C in later courses
- today's lecture: overview of MATLAB
- Monday week 12: overview of C

What is MATLAB?

- MATLAB is a computing environment and programming language for *matrix manipulation*
  - a matrix is a 2D array
  - MATLAB is an abbreviation for "matrix laboratory"

- MATLAB offers many additional "Toolboxes":
  - control design
  - image processing
  - machine learning
  - digital signal processing
  - computational fluid dynamics
  - etc.

Do we even need MATLAB?

- **MATLAB is not assessable in ENGG1003**
- BUT... MATLAB is currently used in a number of later courses in Engineering programs
- non-exhaustive list:

  CHEE4945, CHEE4975, ENGG2440, AERO3600, MCHA3400, MCHA3500, MCHA4100, ELEC2132, ELEC2430, ELEC3400, ELEC3410, ELEC3540, ELEC4100, many FYPs, ...

2) MATLAB vs. Python

Features and philosophy:

- **MATLAB** is proprietary, closed-source software
  - developed by MathWorks https://www.mathworks.com/
  - MATLAB license is free for students ...
    but very expensive otherwise

- **Python** is free and open-source software
  - you can keep programming in Python once you graduate!

Advantages of Python over MATLAB

- Python libraries offer similar functionality as MATLAB toolboxes
  - Python libraries growing much faster, are free, & supported by active online community
- Python is a very popular, in-demand language
  - many students are finding that employers are requesting Python coding skills for both positions and internships
  - MATLAB: 4 million users
  - Python: $>$ 8 million users (2 million new users in 2018)
  - Python ranked #1 most popular language in 2020 (IEEE)
- *MATLAB vs Python: Why and How to Make the Switch*
  https://realpython.com/matlab-vs-python/

MATLAB: key language details

- syntax
- arithmetic and relational operators
- flow control
- arrays
- plotting
- functions

If you're familiar with Python at level of ENGG1003, estimate transition to MATLAB in 1–2 weeks

MATLAB syntax

- MATLAB comments start with %. Python comments start with #
- white-space and indenting are very important in Python. MATLAB does not require the same (but it is highly recommended anyway for readability)
- MATLAB function `disp( )` replaces `print( )`
- help for a function via `help fname` rather than `help(fname)`

Arithmetic and relational operators

- addition, subtraction, multiplication and division are the same as Python. A difference is that
  - MATLAB uses `^` not ** for exponential
  - ie: `x**2` becomes `x^ 2`

- relational operators ==, >, <, >=, <= are the same, except
  - MATLAB uses ~= instead of != for 'not equal to'

- value of a MATLAB variable is *automatically* printed to the command window (console), unless ';' used to suppress
  - ie: `a = 3` prints value of `a`, whereas `a = 3;` does not

Flow control

- `if-else` statements and `for` / `while` loops work exactly the same way in both languages, with some small differences in syntax:
  - MATLAB does not need the ':' used in Python at the end of the loop definition / if condition
  - MATLAB designates the end of an If statement or loop by 'end' instead of by indenting
  - Python shortens `elseif` to `elif`. MATLAB does not

Flow control

Nested If-Else

```python
num = 10                                    # Python
if num == 10:
    print("num is equal to 10")
elif num == 20:
    print("num is equal to 20")
else:
    print("num is neither 10 nor 20")
```

```matlab
num = 10;                                   % MATLAB
if num == 10
    disp("num is equal to 10")
elseif num == 20
    disp("num is equal to 20")
else
    disp("num is neither 10 nor 20")
end
```

Flow control

Example: use a `for` loop to add integers from 1 to 10

```python
sum = 0                          # Python
for i in range(1,11):
    sum = sum+i
```

```matlab
sum = 0;                          % MATLAB
for i = 1:10
    sum = sum+i;
end
```

Flow control

Example: use a `while` loop to find the first (smallest) integer which, when squared, is greater than 100

```python
n = 1                                   # Python
n_squared = 1
while n_squared < 100:
    n = n+1
    n_squared = n**2
```

```matlab
n = 1                                   % MATLAB
n_squared = 1
while n_squared < 100
    n = n+1
    n_squared = n^2
end
```

Arrays

- MATLAB arrays work a lot like `numpy` arrays. If you can work with Numpy arrays you will find MATLAB arrays are easy. Both do vectorisation in the same way
- There are a couple of syntax differences to note:
  - In MATLAB, when you want to index an array, you use round brackets `'()'`. Square brackets `'[]'` are used to create arrays

```python
arr = np.array([10, 20, 30])    # Python
s = arr[i]
```

```
arr = [10, 20, 30]              % MATLAB
s = arr(i)
```

Arrays

- 2D arrays use ';' to designate the next row

```
a = np.array([[2,3],[4,5]])     # Python
s = arr[i,j]
```

```
a = [2 3; 4 5]                  % MATLAB
s = a(i,j)
```

- All zero / all one matrices are very similar

```
m = np.zeros([5,10,3])  # Python 5x10x3 array
```

```
m = zeros(5,10,3)       % MATLAB 5x10x3 array
```

- In MATLAB a new copy of an array is created by default

```
b = a.copy()  # Python
```

```
b = a         % MATLAB
```

Arrays

- **in Python, the index of the first element in an array is '0', in MATLAB it is '1'**
- in Python, the index of the last element in an array is '-1', in MATLAB it is '`end`'
- length of an array is `length` rather than `len`
- array slicing works similarly in MATLAB to Python
- no need to call `linspace()` to create an array in MATLAB, `start:step:end` will do

```
1 a = 5:1:100                    % MATLAB
```

Plotting

- plotting in MATLAB is very similar to `Matplotlib.pyplot`
- including the functions `plot()` `imshow()` `subplot()` `scatter()` `title()` `axis()` `xlabel()` `tick()` `figure()` and many more
- two tips:
  - in Python plotting a new curve will add it to the figure. In MATLAB it will replace the first curve unless you use a `hold on` command first
  - MATLAB does not need a `show()` command, the figure is shown automatically and does not need to be closed for the remainder of the program to be run

Plotting

```python
import numpy as np                                  # Python
import matplotlib.pyplot as plt
v0 = 5
g = 9.81
t = np.linspace(0,1,1001)
y = v0*t - 0.5*g*t**2
plt.plot(t, y)
plt.xlabel('t (s)')
plt.title('Velocity over time')
plt.show()
```

```matlab
v0 = 5;                                             % MATLAB
g = 9.81;
t = 0:.001:1;
y = v0*t - 0.5*g*t.^2;
plot(t, y)
xlabel('t (s)')
title('Velocity over time')
```

Functions

- MATLAB functions work similarly to Python
  - declared by defining the function name, inputs, outputs and operation in the function declaration, then call the function by name as needed
  - functions start with the keyword `function` not `def`
  - function output is defined at the *start* of the function, not the end
  - end of function is indicated with the keyword `end`

```python
def addition(num_1, num_2):                    # Python
    total = num_1 + num_2
    return total
```

```
function [total] = addition(num_1, num_2)  % MATLAB
    total = num_1 + num_2;
end
```

Next steps

- getting MATLAB, if you need it *for later courses*
  - **MATLAB is not assessable in ENGG1003**
  - https://www.newcastle.edu.au/current-students/support/it/software-and-tools
- Octave: free & mostly compatible with MATLAB
  - https://www.gnu.org/software/octave/index