Computer Algebra and Technical Computing (MTH1006)

B. Vorselaars
bvorselaars@lincoln.ac.uk

School of Mathematics and Physics, University of Lincoln

Schedule today

- Recap
- Command-line basics
- Scripts
- Complex numbers
- Vectors

Matlab recap

- Finish the Matlab OnRamp course if you haven't done so yet. Good introduction in syntax and various commands.
- Continue *today* with logbook (include certificate of OnRamp and solutions of *all* regular exercises, including reflection)
- Matlab calculator demo

Matlab recap - variables

Variables in Matlab can be used to store things.

```
>> x=20^2
x =
    400
>> y=sqrt(x)
y =
     20
>> x=2*x
x =
    800
>> y
y =
     20
```

Other command line basics I

Output control

- Semicolon ; suppresses output to the screen and separates commands

```
>> 4*3; 3*2
ans =
      6
```

- Comma , only separates commands.

```
>> 4*3, 3*2
ans =
      12
ans =
      6
```

Other command line basics II

Long lines

- `...` (3 dots): continuation of command on next line

```
>> 3+  ...
3
ans  =
       6

>> x=  ...
5
```

implies that the variable x becomes equal to 5.

The 3 dots notation, `...`, is especially convenient for very long commands.

Other command line basics III

- `ans`: temporary variable where the *answer* is stored (if no variable is given)

```
>> 4*3; ans
ans =
    12
>> x=ans*2
x =
    24
```

- `whos` shows all variables in memory.

```
>> b=sin(5); x=3;
>> whos
  Name        Size                    Bytes  Class
              Attributes
  ans         1x1                         8  double
  x           1x1                         8  double
```

Other command line basics IIII

- %: comment sign. Any text behind it is discarded

```
>> x=3   %set x to 3
x =
      3

>> x=3   % x=4
x =
      3
```

Scripts

A script is a list of Matlab commands gathered together in a file. This is useful if we have a sequence of commands that we may wish to execute many times.
We can open such a file by typing at the Matlab prompt

```
>> edit myscript.m
```

Once we have saved a series of commands in the file we can run them by typing

```
>> myscript
```

An M-file of recently typed commands may easily be produced by highlighting them in the 'Command history' window and right-clicking on the mouse to choose 'Create M-file'.
demo

Live script

Matlab also has *live scripts*

- Similar to ipython/jupyter python notebooks.
- It is basically a script but the output is included in the file. It also allows one to add text.
- The extension (ending) of the filename is `.mlx` instead of `.m` as with a normal script.
- In this module I focus on normal scripts, but most of it also applies to live scripts.

Script filename

The filename of a *script* has similar restrictions to the name of a variable:

- Start with a letter
- Potentially continues with a letter, underscore, or digit (multiple times)
- No other characters, such as spaces, %, $, (, +, ., etc.
- Should not be one of the keywords, such as `end`
- Be aware when scripts have the same name but different capitalizations (`MYSCRIPT` vs `myscript`). This may lead to confusion. Best to use all lower case.

✓ `myscript`

✓ `Yes`

× `3script`: starts with a number

× `my script`: contains a space

× `exc3.1`: contains a dot

× `end`: is a reserved keyword

✓ `sin`: but not advised, since it is already a function

Complex numbers I

Imaginary part: use an `i`

```
>> (-10)^(1/2)
ans =
    0.0000 + 3.1623i

>> (1+2i)*(1-4i)
ans =
    9.0000 - 2.0000i
```

`i` is just shorthand for `1i`

```
>> i
ans =
          0 + 1.0000i
```

Complex numbers II

Real part

```
>> real(2+3i)
ans =
    2
```

Imaginary part

```
>> imag(2+3i)
ans =
    3
```

Complex numbers III

Absolute value of complex number:

```
>> abs(2+3i)
ans =
    3.6056
```

Angle in radians

```
>> angle(2+3i)
ans =
    0.9828
```

Euler formula: $z = |z|e^{i\phi}$

```
>> 3.6056*exp(0.9828*i)
ans =
    2.0000 + 3.0001i
```

Vectors

- Scalar: a single number. E.g., 5:

5

- Vector: collection of multiple numbers, either in a row or in a column. A row vector with elements 5,1,2 is

5	1	2

- A column vector with the same elements:

5
1
2

- A longer row vector of 5 elements

5	1	2	7	10

- Why difference in row and column? Important when Matrices will be discussed. Both are special types of matrices.
- Why vectors at all? Convenient if many numbers are considered. Easier to handle than one variable for each element.

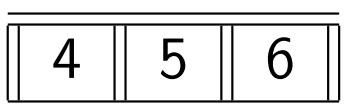
Vectors in Matlab

- Instead of variable a, b, c:

```
>> a=4, b=5, c=6
a =
     4
b =
     5
c =
     6
```

one can create a row vector $x = (4, 5, 6)$ in Matlab

```
>> x=[4, 5, 6]
x =
     4     5     6
```

Memory storage: 

- Matlab: elements separated by *commas* (or spaces), and vector elements enclosed by *square brackets*.

Vectors

```
>> x=[4, 5, 6]
x =
      4     5     6
```

- Accessing first element:

```
>> x(1)
ans =
      4
```

- Accessing second element:

```
>> x(2)
ans =
      5
```

- Accessing last (*end*) element:

```
>> x(end)
ans =
      6
```

Vectors

- Twice the third element?

```
>> 2*x(3)
ans =
      12
```

- Changing one element

4	5	6

```
>> x(2)=11
x =
      4     11      6
```

Memory storage:

4	11	6

Creating vectors

- Creating long vectors.

```
>> x=[2,3,4,5,6]
x =
     2     3     4     5     6
```

- Alternatively: use the colon operator: constant step of 1 between the elements:

```
>> x=2:6
x =
     2     3     4     5     6
```

- Step size can be specified as an extra number:

```
>> x=[3,5,7,9]
x =
     3     5     7     9
```

can be replaced by

```
>> x=3:2:9
```

Creating vectors

Vectors can be specified in three ways:

1. Specify all the elements of the vector, e.g.

   ```
   >> x=[0 2 4 6 8 10]
   ```

2. Specify the first and last elements and an increment.

   ```
   >> x=[0:2:10]
   ```

3. Generate a vector between two limits with a regular spacing interval and a specific number of elements: use the linspace command

   ```
   >> x=linspace(0,10,6)
   x =
        0     2     4     6     8    10
   ```

Vector operations

Convenience of a vector is that you can apply the same operation on all its elements

```
>> x=[1,2,3];
>> 4*x
ans =
     4     8    12

>> theta=[0, pi/2];
>> sin(theta)
ans =
     0     1
```

Vector operations – element-wise power

Maple applies matrix operations by default (explained in the linear algebra module next term). To use element-wise operations *instead*: use an extra dot '.'

```
>> x=[1, 2, 3]
>> x^2
Error using  ^
Inputs must be a scalar and a square matrix.
To compute element-wise POWER, use POWER
   (.^) instead.

>> x.^2
ans =
     1     4     9
```

Vector operations – element-wise multiplication

Matrix multiplication

```
>> x=[1, 2, 3]; y=[2, 2, 4];
>> x*y
Error using  *
Inner matrix dimensions must agree.
```

Element-wise multiplication

```
>> x.*y
ans =
     2     4    12
```

For scalars: the two multiplications are the same

```
>> 4*x
ans =
     4     8    12
>> 4.*x
ans =
     4     8    12
```

Vector operations – element-wise division

Matrix division

```
>> x=[1, 2, 3]; y=[2, 2, 4];
>> x/y
ans =
    0.7500
```

Elementwise division

```
>> x./y
ans =
    0.5000    1.0000    0.7500
```