MATLAB Tutorial - Week 4

Mathieu Dever

NOTE: at any point, typing `help` *`functionname`* in the Command window will give you a description and examples for the specified function

1 How to create a MATLAB function

A MATLAB function is a simple .m file (or MATLAB script) that follows a few extra syntax rules. You can get a MATLAB function template here:
`https://dl.dropbox.com/u/25195822/MATLAB%20workshop/Tuto-4/MATLAB_function_template.m`

- The script must start with

  `function [output(s)]=`*`functionname`*`(input(s))`

You can include as many inputs and outputs as you like. The variable names listed in `[output(s)]` will be the only variables returned when executing the function in MATLAB. Any other variables defined within the function will be deleted. For example, `rhoa=air_dens(Ta,RH,Pa)` returns the air density based on air temperature, humidity and pressure. If you look at the code (`open air_dens`), several variables are calculated (moisture correction, specific humidity,...). However, the density `rhoa` is the only output returned when executing the `air_dens` function.

- We usually include a comment section (using the % symbol) to provide information on the function, its inputs and outputs (please refer to the function template). This is very helpful if you want to remember how a function works!! This commented section will be displayed when typing

  `help` *`functionname`*

- The first part of the function is usually dedicated to checking if the number and dimension of inputs/outputs fit the function's need. The functions `nargin` and `nargout` help checking if the number of inputs (or 'arguments in') and outputs (or 'arguments out') are good.

For example, `rhoa=air_dens(Ta,RH,Pa)` needs 3 different inputs, so the code should start with an if-statements that checks the number of inputs/outputs:

```
if nargin ~= 3
   error('myApp:argChk', 'Wrong number of input arguments')
end
if nargout ~= 1
   error('myApp:argChk', 'Wrong number of output arguments')
end
```

Sometimes, inputs are only optional. For example the air pressure `Pa` in `rhoa=air_dens(Ta,RH,Pa)` is optional. For optional inputs, an if-statement at the beginning of the code usually defines the missing variable. In `air_dens`, the code assigns an average value to the air pressure if it is not specified in the inputs:

```
if nargin == 2
   Pa = 1020;
end
```

It is also important to note that the name of the I/O within the function do not have to match the name of I/O when calling the function. For example, I can call the `air_dens` function using `dens=air_dens(T,humid,press)`. MATLAB only uses the order of the inputs: *i.e.* the first input is the temperature, the second the humidity, etc... Therefore, the I/O's order is important!!

- Then, the main body of the function includes the code required for your calculations
- Finally, the function script **must** have the same name as the function. For `rhoa=air_dens(Ta,RH,Pa)`, the m-file has to be named 'air_dens.m'. Be careful to not use the same name for different m-files!!

NOTE: MATLAB functions must be carefully coded. You must make sure that it applies generally. The function must be reliable and should be usable by someone else with other inputs. MATLAB

functions are great and can be shared among MATLAB users. USGS put together a nice dataset of MATLAB functions that are useful in the oceanography world:
http://woodshole.er.usgs.gov/operations/sea-mat/. It is worth browsing through!

Another good reference is the file exchange forum, on MATLAB's website. A very, very large number of functions are available there. But remember to be critical when using those. I recommend reading comments, double-checking the code, using a simple example to verify the outputs, ...

http://www.mathworks.com/matlabcentral/fileexchange/

2 Example: Transform wind direction from meteorological coordinate to cartesian coordinates

We want to create a function called 'met2cart' that transforms a wind record (usually wind speed and wind direction FROM, relative to true north) into a wind vector with eastward (U) and northward (V) components.

- 2 inputs: WDIR (wind direction FROM) and WSPD (wind speed)
- 2 outputs: U (eastward component) and V (northward component)
- The first step is to change the wind direction from FROM to TO. This can be done by adding 180^o to the wind direction, using the *modulo* function `mod()`
  ```
  WDIR=mod(WDIR+180,360)
  ```
- Then, the wind direction must be transformed from clockwise degrees relative to True North to polar angles (anti-clockwise relative to east), once again using the `mod()` function:
  ```
  WDIR_polar=mod(90-WDIR,360);
  ```
- Finally, we can use the `pol2cart()` built-in MATLAB function to convert from polar coordinate to cartesian coordinate (in `pol2cart()`, the angle must be in radians):
  ```
  [U,V] = pol2cart(deg2rad(WDIR_polar),WSPD)
  ```

- Once the function is written, use simple values to double check the output:

```
[U,V]=met2cart(270,1);
%Eastward wind (coming FROM 270^o), should return U=1, V=0
```

Use the wind record from Sable Island from tutorial #2 and #3 to calculate wind vectors. You can use the `quiver()` function to plot it in a nice way.