TECHNISCHE UNIVERSITÄT DRESDEN

INTRODUCTION TO MATLAB

Cells, structs, strings and functions

Dario Cuevas and Vahid Rahmati

Dresden, November 19, 2014

01 Cells and structures

Cells:
They are similar to arrays, but each element can have a different size
Example:
To initialize a cell array:

```
A = cell(3,2)
```

To index, use curly brackets:

```
A{1,1} = magic(5);
A{3,2} = zeros(2,1);
```

To index a cell's element's elements: `A{1,1}(1,1)`

Structures:
Like Cells, but indexed with names:
Example:
For a structure named "subject",

```
subject.age = 30;
subject.country = 'Mexico';
subject.height = 1.83;
subject.results = [1, 0, 1, 1, 0];
```

To index the element's element, `subject.results(5)`

01 Cells and structures exercises

1. Create a vector-cell CellA whose first element is `[1]`, the second `[1, 2]`, then `[1,2,3]`, etc., until 5. The 6th element is `magic(7)`. The 7th one is empty.
2. Create a structure called MyStruct with elements: NoOfClassmates, CurrentYear, MyCell and Magia. The value of MyCell should be CellA from the previous exercise. The value of Magia should be the 6th element of CellA.
3. From MyStruct, change the 7th element of MyCell (that is, `MyStruct.MyCell{7}`) to `rand(2,10)`

02 Strings

Strings are arrays of letters.
`A = 'I am Jo';`
They are indexed like an array:
`A(1)` gives I , `A(2)` gives (empty space) ;
To create two-dimensional arrays of chars:
`B = char(A, 'Yes I am');`
Note: `C = '5';` is NOT a number but a string. `C+5` gives unexpected results.
Examples for indexing:
`A(8:end)` gives Jo
`B(2,1:3)` gives Yes

Exercise: Substitute Jo's name for your own in A. You might have to add or delete characters at the end.

03 Save and load commands

`save(filename, variable(s))`
For example, `A = 1; B = magic(5)`:
`save('myvariables.mat', 'A', 'B')`
When no variables are given, all are saved. For example:
`save('myworkspace.mat')` All variables are saved as a structure. Thus, myworkspace.mat is a structure with fields mystructure.A and mystructure.B
`load('mystructure.mat')` will load all the variables in mystructure.mat into your workspace. Matlab has to be in the folder where mystructure.mat is located.
`myStruct = load('mystructure.mat')` will load all the variables into myStruct. Then, to access them, use myStruct.A and myStruct.B

04 Scripts

- Scripts are successions of commands. Executed in the order found (from top to bottom).
- % at the beginning of a line means that it's a comment and won't be ran.
- Use ; at the end of each command to suppress the output of that command.
- To run the script, use F5.
- Use %% to divide the script in independent cells.
- To run a cell, press ctrl+Enter.
- Script names can have letters, underscores and numbers. Just like variables.
- Remember, all will be saved to the workspace (command window). Variables will be overwritten.

04 Functions

A function is defined as:

```
function [output1, output2, ...] = NameOfFunction(input1, input2,...)
end
```

- It's also a succession of commands
- All variables are stored in a temporary workspace and deleted afterwards.
- You can reuse names of variables that are in your main workspace without changing them.
- You cannot use variables from outside of the function unless passed as inputs.

To call a function:

```
[variable1, variable2,...] = NameOfFunction(input1, input2,...);
```

04 Remarks about functions

- Run the function with the name of the file, not the name of the function.
- Name the function the same as the file.
- An error is returned when a function is called with insufficient inputs.
- Use `~` to suppress an output.
- You can call a function without assigning its output. It will display the output on the command window.
- The variables given as input when calling the function need not have the same name as in the function definition.
- Use a ; when calling a function to not display its output.
- Matlab does not check the variable type of the inputs. This should be done “manually”.
- The comments at the beginning of a function are displayed when `help MyFunct` is called.

04 Exercises

1. Create a function called MyConcatenation, that takes as input two matrices. It should concatenate these two matrices one next to the other, in the order they are given. Additionally, these two lines of code should be included:

   ```
   figure
   imagesc(X)
   ```

   where X is the result of the concatenation. The function should also output X. The input matrices for the function should be: `A=ones(5,1)`, `B = magic(5)`.
2. Using MyConcatenation, concatenate the output of the previous exercise with a matrix `C = zeros(5,1)`.
3. Write a function MyElimination that removes the last 3 columns of the output of the last exercise.
4. Write a script that runs the last 3 exercises together.