Lab 02: Variables, Arrays, and Scripts

Math 3341: Introduction to Scientific Computing Lab Spring 2018

In this assignment you will learn about proper ways to define variables, how to handle basic vector operations, create and run a script file, and produce and save a basic plot.

Variables

Just as in the rest of mathematics, variables help us represent quantities or expressions in order to make their use and re-use more convenient. When coding if a parameter (i.e. a variable) appears twice, invent a name for it and set its value once. This way if you need to change a value, you only need to change it once. Instances of the variable that you miss will cause problems (and be hard to find!).

Naming Variables

ML variable names must start with a letter. The first letter can then be followed by more letters, numbers or underscores. Using spaces is not permitted. Variable names are also case sensitive (i.e. $a \neq A$).

There are two naming conventions that should be used when defining variables. If using two "words" in your variable name either capitalize the first letter of the second word or separate them by an underscore _. Say we had a variable x. For example we could name this variable

`x_val` or `xVal`

This helps make variable names easy to read. Be as descriptive as possible with your variable names so that their use or value represented is very clear. However, you don't want them to be too long (or hard to spell) as it will make their use more difficult.

Avoid function names when naming variables. In general, a variable you define will take precedence over a ML function/variable name. This means it is possible to re-define some ML functions/variables which can create confusion in your code. When writing code it is a good idea to check if a name is already in use. This can be helpful when writing/running code to be sure you have no conflicts with ML functions/variables or variables you have defined yourself.

Default Variable Definitions

Command	Description
`pi`	variable defining π
`i` or `j`	imaginary number $i = \sqrt{-1}$

Arrays

In programming language, an array is the equivalent of a vector or matrix. There are several ways we can define row vectors, column vectors, or matrices. The general notation for a vector or matrix is a list of values enclosed in square brackets `[ ]` separated by either commas or semi-colons (or a combination of the two).

Values you want to appear in the same row are separated by commas (or spaces). Columns are separated by semi-colons.

Vectors

So to define a row vector, you can use spaces and commas interchangeably to separate elements. In other words,

`x = [1,2,3,4,5]` is the same as `x = [1 2 3 4 5]`

To define a column vector you can either define a row vector and use the transpose function `'` to produce a column vector or use the semi-colon ; after each element

`x=[1; 2; 3; 4; 5]` or `x = [1 2 3 4 5]'`

Matrices

A matrix is just a combination of row (or column) vectors. For instance,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$ in MATLAB becomes `A = [a, b ; c, d]`

Functions for Vectors & Matrices

Command	Description
`transpose(A)` or `A'`	gives the transpose of a (real valued) vector or matrix
`size(A)`	gives the size of an array or matrix
`zeros(m,n)`	creates an array with m rows and n columns of zeros. Note that `zeros(n)` creates an n by n array of zeros!
`ones(m,n)`	creates an array with m rows and n columns of ones. Note that `ones(n)` creates an n by n array of ones!
`linspace(a,b,N)`	creates a vector with N equally spaced entries ranging from a to b
`colon(j,i,k)` or `j:i:k`	creates a vector with entries from j to k with increment i

Script Files

Up to this point we have only used the command window to execute commands and functions. This can be tedious as our tasks become more complex. If edits need to be made we would need to re-enter each line of code. This is where a script file comes in handy.

A script file is simply a file that contains a chain of commands that you edit in a separate window, then execute with a single mouse click or command. This is where we can define variables, perform calculations and leave comments to remind us what the file calculates.

It is good practice to leave a comment at the top of any script file with a brief description of what the file does or what assignment it is for. It is also recommended to include the course and semester so future you doesn't go crazy trying to figure out when a file was written or when you may have needed it.

Before entering in any code or variable definitions it is also recommended that you place a `clear`. This clears all previous variable definitions from code for other problems or previous versions of your current code. This will help avoid conflicts with your variables. For example:

```
% MATH 3340, Spring 2018
% Homework 1 – Bisection Method

clear % clears previous variable definitions
< script contents here>
```

Additional Functions and Commands

The following is a list of commands used in this lab along with a brief description of what they do.

Commands that do not require inputs

Command	Description
`iskeyword`	prints a list of keywords used by ML. These cannot be used as variable names.
`who`	returns list of variables in use
`whos`	returns list of variables with their size and type
`clear`	clears all variables
`clc`	clears the command window

Commands that require inputs

Command	Description	Example
`clear <varName>`	clears specified variable(s)	`clear temp day x`
`exist <varName>`	returns 0 if variable name is not used	`exist x`
`disp`	displays text output in command window	`disp('hello world')` or `disp 'hello world'`
`%`	use before any comment in script file or command window	`% this is a comment`

Remember that to begin the lab you should do the following:

- In the command window enter the command `diary('lab02.txt')` this will create a `.txt` file. This will record all input and output in the command window until you type the command `diary off`.
- Type the command `beep off`. This will disable the sound that plays when there is an error in your code.

Lab Exercises

I. Defining Variables

1. Type `% Results for Part I` in the command window and hit enter.
2. Define the variable `pi = 1.25` and evaluate `cos(pi)`.
3. Now define `cos=2.1`. Use `who` and then `whos` to display the list of your currently used variables. Then, evaluate `cos(pi)`.
4. Enter `clear` to clear all variable definitions.

II. Vectors

1. Open the script file `lab02script.m`
2. Use the colon notation to create a vector z with entries that range from 1 to 20 with a step size of 2.
3. Use `linspace` to create a vector x with 40 entries ranging from 0 to 2π.
4. Evaluate $y = \cos(x)$.
5. You need to take care with element-wise vs. matrix operations when using MATLAB. Evaluate `f = sin(x)/(1+x^2)`. Run the script file.
6. Comment out your definition of `f`. To perform element-wise operations you need to include a '.' in front of the operation. Evaluate `g = sin(x)./(1+x.^2)`. This is one of the most common issues you will encounter in MATLAB!

III. Introduction to LaTeX

Now you'll create a lab report using LaTeX. This is similar to how you will format your homework assignments in the lecture course. Follow the instructions given in lab.

You will submit this lab as a single .pdf file using the provided LaTeX template.