COMS 3101
Programming Languages: MATLAB

Lecture 1

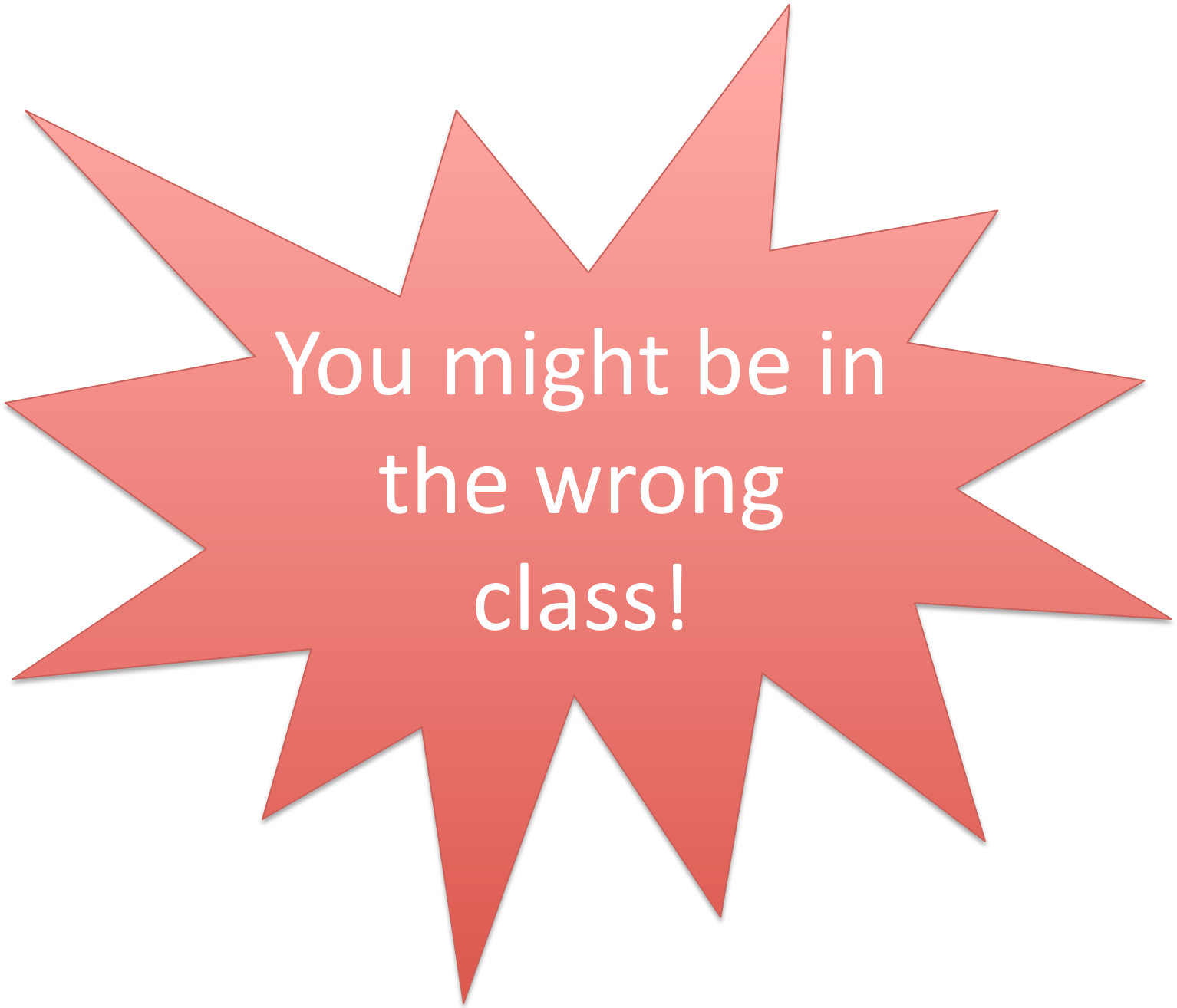
Fall 2013

Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/coms3101/matlab

What is MATLAB?

- MATLAB is a high-level language and interactive environment that allows one to solve science & engineering problems quickly using built-in functionality
- High-level language:
  - User-friendly, easy to use, built-in functions (+)
  - Slower, less control (-)
- Interactive environment:
  - Graphical User Interface (GUI)
  - Visualization


You might be in
the wrong
class!

What is MATLAB?

- Scripting language designed for “gluing together” computations
- Object Oriented Programming (OOP) takes a backseat
- Documentation is sufficient:
  - http://www.mathworks.com/help/techdoc/index.html
- Ideal for developing a prototype or a model, suitable for quick and dirty computation
- Poor choice for a major commercial package

Course Info – Instructor

- Ilia (Eli) Vovsha
  - Email: iv2121@columbia.edu
  - Office Hours:
    - Friday (1:00 – 2:00pm), TA room (Mudd 122A)
    - Monday (3:00 – 4:00pm), notify by email
- PhD student in Machine Learning (final year)
  - ML is a field in which we develop algorithms/systems with a learning component
  - Math + Data + Experiments ➔ MATLAB

Course Info – Goals

- Learn MATLAB to:
  - Perform mathematical operations
  - Manipulate and visualize data
  - Generate curves and plots
  - Implement prototypes to solve toy problems
- We will focus on the formulation and implementation of (simple) mathematical problems

Course Info – Syllabus

- Basic functionality:
  - Workspace, variables, errors, generic commands.
- Arrays:
  - The “building blocks” of MATLAB i.e, vectors, matrices.
- Scripts, simple functions:
  - Some basic user-defined & built-in functions.
- Control flow, operators:
  - Loops (if, else, while), mathematical operations.

Course Info – Syllabus

- File input/output:
  - Loading & saving data, handling different formats.
- Basic plotting:
  - Generating simple curves, plots, and figures.
- Useful data structures:
  - Cell arrays, character strings, 'structs'.
- Advanced functions and plotting:
  - More interesting functions and non-trivial plots.

Course Info – Syllabus

- Debugging, advanced functionality:
  - Recursion, time functions, profiler.
- Practical mathematics:
  - Solving equations (linear algebra), basic statistics.
- Optimization toolbox:
  - Formulating a data-driven mathematical problem and solving it using a toolbox routine.

Course Info – Grading

- 4 Homeworks (4 x 15%)
  - Posted on Monday morning. Due the following Mon, by start of class
  - See course website for submission instructions
  - Two parts: 1st part is required. 2nd part is optional
- Final Project (40%)
  - Solve a toy problem of your choice. Submit a write-up and your code
  - Project can be waived if you complete the 2nd part on every HW

Technical Details

- You may use any platform you wish to run MATLAB.
- Download a windows version or use your CUNIX/CS account to log in
- If you decide to run MATLAB on Linux/Unix, you'll need to install an X-Server for plotting and visualization

Technical Details

- X-Server for plotting and visualization:
  - http://sourceforge.net/projects/xming/
- Putty (for windows):
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
- Instructions:
  - Launch Xming
  - Open a session in putty with host name: 'cunix.cc.columbia.edu'
  - Make sure the X11 option of the SSH category is enabled
  - Enter account username/password
  - Type 'matlab &' to start the engine

Basic Functionality – Workspace

- You type commands in the *command window*, MATLAB tracks your variables in the *workspace* and your commands in the *command history*
- Can manipulate directories from the command line:
  - cd, mkdir, rmdir, ls
- Help window
  - 'help' command is about to become your new best friend

Basic Functionality – Help

- Can search using the window
- Typically easier to use:
  - doc < ***function_name***>
  - help < ***function_name***>
- Documentation for a particular function often lists related functions as well

Basic Functionality – Commands

- Clean up:
  - clear < ***variable*** >
  - clear all
  - clc (clear command window)
- Status check:
  - what (returns the MATLAB files (.m , .mat) in the current directory)
  - who (returns the variables in your workspace)
  - whos (variables with additional info, e.g. size)

Data Structures – Variables

- MATLAB does not use explicit type initialization:
  - int a (WRONG!)
  - a = 2 (Good)
  - a = 'hello' (also good)
- Add operator ';' to avoid printing the variable:
  - a = 2;
- Use 'disp' to improve display of variable:
  - disp(a)

Data Structures – Variables

- Can assign and reassign a variable directly:
  - a = (1+1) /4;
  - a = ‘hello’
- Case sensitive:
  - a = 2 & A = 4 are two different variables
- Avoid using reserved words or functions as variable names:
  - sort = 2 (DANGEROUS!)
  - Double check with ‘help < variable >’ to be safe

Data Structures – Variables

- Some reserved words (keywords):
  - for, if, else, while, end, return…
  - Use 'iskeyword' to check
- Some built-in variables:
  - Inf & -Inf +/- infinity
  - pi 3.141…
  - NaN Not a number (0.0/0.0)

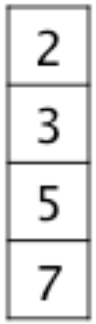
Data Structures – Arrays

- The secret behind MATLAB's popularity (?)
- Started out as a 'matrix laboratory'
- It turns out we do a lot of computation with vectors and matrices
- MATLAB makes it easy to manipulate these data structures

Data Structures – Vectors

- Define a row vector:
  - r = [2 3 5 7]
  - r = [2,3,5,7];

- Define a column vector:
  - c = [2; 3; 5; 7];
  - c = [2,3,5,7]';
  - We can use a transpose operator (`)



Data Structures – Matrices

- Define a matrix:
  - M = [2,4; 3,6; 8,12];
- Concatenate two vectors:
  - v1 = [2,4]; v2 = [3,6];
  - M = [v1; v2];
- Can concatenate vectors and matrices. Dimensions and types must agree

2	4
3	6
8	12

Data Structures – Matrices

- Special constructor:
  - r = 1:5; ➔ [1,2,3,4,5]
  - r = 1:2:5; ➔ [1,3,5]
  - M = [1:5; 1:5]
  - M = [1:5; 1:2:5] (ERROR!)

Data Structures – Matrices

- Some predefined matrix creation functions:
  - M = zeros(2,3); Matrix of zeros
  - M = ones(2,3); Matrix of ones
  - M = eye(2); Identity matrix (2 by 2)
  - M = rand(2,3); Random numbers (uniformly distr.)
  - 1st argument = # of rows, 2nd argument = # of columns.
- We will consider 2D matrices almost exclusively, but may just as well use 3D cubes e.g. M = zeros(5,5,5)

Data Structures – Matrices

- Matrix dimensionality:
  - M = rand(2,3);
  - [r,c] = size(M);
  - r = size(M,1); # rows
  - c = size(M,2); # columns
  - 1st argument = # of rows, 2nd argument = # of columns.

Data Structures – Matrices

- Indexing & Accessing array elements:
  - Index starts from 1
  - e1 = M(1,2); Explicit access
  - e1 = v(4); Explicit access
  - v1 = M(1,1:2); Return 1st two columns from row 1
  - v1 = M(:,2); Return 2nd column
  - e1 = M(1,end) Return element from last column in row 1

Basic Operators

- The usual math operators:
  - + - * / ^
- Logical operators:
  - & | ~
- Relational operators:
  - > < >= <= == ~=
- Matrix operators:
  - Ambiguous, element wise or matrix operation?
  - .* ./ .^ use dot to disambiguate

Matrix Operators

- X = [2 3 4; 5 4 6]; Y = [1 2 3; 3 3 3];
- Operations:
  - Z = X + Y ~~(X .+ Y)~~
  - Z = X – Y ~~(X .- Y)~~
  - Z = X .* Y ~~(X * Y)~~
  - Z = (X') * Y
  - Z = X .^ 2 ~~(X ^ 2)~~

2	3	4
5	4	6

1	2	3
3	3	3

Array Manipulation

- Special functions for arrays:
  - sum(), prod(), mean(), max(), min()
  - First argument we pass to each function is the array
  - What if array is a matrix, what does sum(X) mean?
- Useful routines:
  - sort(), find(), sortrows(), any()
  - Same issue as above, what does sort a matrix mean?

Array Manipulation

- Special functions for arrays:
  - sum(), prod(), mean(), max(), min()
- Suppose M is a matrix, then:
  - TScol = sum(M); TScol is a row vector, each element is the column sum
  - TSrow = sum(M,2); TSrow is a column vector, each element is the row sum

  - TS = sum(sum(M)); TS is the total sum of all elements in the matrix
  - Can also obtain the location of the max/min value
    (if argument is a vector): [val, loc] = min(M);

Array Manipulation

- Useful routines:
  - sort(), find(), sortrows(), any()
- Function sort() sorts elements in *ascending* (default) order. Can specify the order explicitly:
  - sort_list = sort(M, 'descend');
  - If M is a matrix, sorts each column independently
  - As above, can sort across columns: sort(M,2,'ascend')
  - If consistency between rows is required, use sortrows()

Array Manipulation

- Function find() returns indices of elements which satisfy some condition:
  - find(M) Return indices of nonzero elements of M (default)
  - find(M == 1) Return indices of elements of M equal to one
  - find(M >= 2 & M < 6) Return indices that satisfy both conditions
- If M is a matrix, the returned index is a single value which is a bit annoying. Solution: use ind2sub(), sub2ind() to convert between indices
- [r,c] = ind2sub(size(M),index)

Scripts

- MATLAB specific script files (M-files)
- Files have the extension .m (example: test.m)
- To run your script just type ‘test’ in the command line
- A script is a list of commands that should be executed
- To pass arguments to the script we will use functions (more on that later)

Comments

- Comments make the code more manageable
- Can add a one-line comment by using the '%' symbol:
  - **% this is a comment**
- Anything that follows a % is ignored by the compiler (not executed)
- When you type help <func_name> you get the comments at the top of the <func_name> script
- Can also use block comments: %{ …. %}