ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Μηχανική Μάθηση
Introduction to Matlab

Γιώργος Μπορμπουδάκης
Τμήμα Επιστήμης Υπολογιστών

About Matlab

- Matlab (MATrix LABoratory) is an interactive system for doing numerical computations.
  - It is easy to use
    - Easy and fast to write code
    - User-friendly interface
    - No memory management
  - Has many libraries (toolboxes)
    - E.g. for signal processing, bioinformatics, econometrics, **neural networks** and many more.
  - Great documentation
    - Type 'doc' in the Matlab prompt to open the documentation window.

Vectors and Matrices

- Every variable in Matlab is a vector or matrix.
- A normal variable is a 1x1 matrix.
- There are two types of vectors:
  - Row vectors (1xN)
    - E.g. [1 2 3 4 5] is a 1x5 row vector
  - Column vectors (Nx1)
    - E.g. [1;2;3;4;5] is a 5x1 column vector
- Matrices are of size NxM
  - E.g. [1 2 3; 4 5 6; 7 8 9] is a 3x3 matrix

Data Types

- Matlab is dynamically typed (types are determined at runtime)

- The default type for a number is **double**
  - Basic types: **single** (float), **logical** (boolean)
  - Other types: uint8, uint16, uint32, int8, int16, int32 and more
  - But be careful: some of them are **not supported** by some operations and may **not be compatible** to use with other data types.
  - Most of the times using the default type is OK (unless memory is an issue)

- Special values
  - Inf (Infinity)
  - NaN (Not a Number)

- Strings
  - They are also arrays (vectors), of characters.
  - Unlike most languages, strings begin with ‘ and not with “. E.g. ‘abc’ or [‘a’ ‘b’ ‘c’]
  - The characters are represented with ASCII codes

Accessing Elements of an Array

- Unlike most languages, the index of the first element of an array is 1
  - E.g. for x = [4 5 6]; x(1) will return 4.

- Arrays are declared with [..], but accessed with (..)

- To access elements of a multi-dimensional array use x(i1,i2,…,in)

Operating on Arrays

- Functions (usually) process whole arrays with a single call
  - E.g. x = [1 2 3 4]; y = [5 6 7 8]; z = x + y; (z = [6 8 10 12];)

- **Forget loops**: In most cases you will not have to use them.
  - Easier to make mistakes.
  - Using loops in general is a lot slower than using built-in functions.
    - E.g. A simple addition of two arrays will be at least 2 times slower (depending on the dimension of the arrays : more dimensions imply more nested for loops which again result in much slower code).

Operators

- Most basic operators are the same as in other languages (+,-,*,/)
  - Modulo operator: mod(x,y) (% is used for comments)
  - Power operator: ^
  - Transpose operator: ‘
- For some operators such as * / ^ there are two versions:
  - The first is for linear algebra operations
  - The second is for pair wise operations (.* ./ .^)

Operators

- Comparison operators (>, <, >=, <=, ==)
  - Compare a number, vector or matrix with another number or vector/matrix of same dimensions
  - E.g. x = [1,2,3,4];
  - x>=3 creates a vector of size 1x4, containing the values true wherever x contains a value >= 3 ([0,0,1,1] of type logical)
  - Can be used to index x: x(x >= 3) = [3,4]

- Logical Operators: && (and), || (or), ~ (not), &, |
  - &&, || are used between two values (short circuit evaluation)
  - &,| are used between a number, vector or matrix with another number of vector/matrix of the same dimensions
  - E.g. x((x <= 1) | (x >= 4)) gives [1,4]
  - ~ can be used with any expression or vector/matrix

Accessing Elements of an Array (colon operator)

- The Colon Operator (:) is perhaps the most important operator in Matlab.

- It is used to create a sequence of numbers.
  - E.g. 1:5 creates a row vector with the numbers [1 2 3 4 5]

- The spacing can be anything
  - E.g. 100:-5:80 creates [100 95 90 85 80]

- This operator allows to access portions of matrices
  - E.g. x(1:5, 2:3) to access the first 5 rows and the 2nd and 3rd columns

Accessing Elements of an Array

- We can use arrays of numbers to access elements of other arrays.
  - x([1,5:10,**end**])
    - End is a special keyword used to access the last element of an array.
  - x(x >= 3) returns an array of all elements in x greater or equal to 3

Conditional Statements

- If/Else/Elseif
  - E.g. if x > 2 … elseif x < 1 … else … end
  - After each opening ‘if’ an ‘end’ is needed to terminate the statement.
  - Be careful: elseif is not the same as else if.
    - The first elseif is connected to the previous if, the second uses another nested if statement and needs a separate end statement in the end.
    - Avoid the else if combination whenever possible.

Conditional Statements

- Switch/Case
  - Can also be used with strings.
    - E.g.

```
switch x
        case ‘abc’
                …
        case ‘def’
                …
        otherwise
                …
end
```

  - Notice that you don’t have to use any break statement.
  - (doc *switch*)

Loops / Break-Continue

- For/While
  - Examples:
    - For
      - □ for i = 1:10 … end
      - □ for i = 1:10

```
            for j = 10:-1:1
            …
            end
    end
```

    - While
      - □ while x > 10 … end

- Break/Continue
  - Same as in other languages like C and Java.

Other

- Semicolon
  - Used to end statements. If it is not used the result of the expression is printed to the prompt.
  - Also used to change rows when creating an array
    - [1;2;3] creates a column vector (3x1)

- Comma
  - The comma operator can be used to separate statements/expressions.

- Comments start with % (% is the same as // in C)

- To continue code in a new line use the '…' operator (three dots).

Input / Output

- Input
  - Read from standard input:
    - Function *input* (see doc/help input)
  - Read from file:
    - Function *fscanf* (see doc/help fscanf)

- Output
  - Print to file
    - *fprintf*(FileID, format,…) (see doc/help fprintf)
  - Print to standard output
    - *fprintf*(format,…) (see doc/help fprintf)
  - Save results to file
    - *save* filename variable1 variable2 … variableN

Data Structures

- Structs
  - Structs are easy to use. You do not have to define some struct (like in C), but you can "build it" at runtime.

  - To access a field of a struct the '.' operator is used.
    - x.a = 1;
    - If x.a does not exist it is added at runtime.
      - □ E.g. x.a = 1; x.b = [1,2,3]; x.c = 'abc';

  - The fields of structs can be of any type.

Data Structures

- Cell Arrays
  - Cell arrays can hold any type of data in each cell.
  - Instead of '[' and ']' use '{' and '}' to create a cell array.
    - x = { 'abc', 1, [5;6;7], [], {'def',2} } (1x5 cell array)
  - Instead of '(' and ')' use '{' and '}' to access an element of a cell array.
    - x{1} to access 'abc'
    - x{3} = [1 2] to replace the [5;6;7] with [1 2]

Scripts

- In Matlab, you do not need any main function to run the program.
- Programs are scripts which are running through an interpreter (code can also be compiled).
- It is not good practice to run everything through scripts for many reasons.
- That does not mean that scripts are not used; they should be avoided if a function can be used instead (similar to avoiding having everything in the main function in C).

Functions

- Functions are organized in a single file.
  - The filename is also the function name.

- In a file there can be several functions, but only the function with the same name as the filename is accessible from outside (like classes in Java).

- The syntax is simple:

```
function [r1,r2,…,rn] = function_name(a1,a2,…,an)
…
end
```

- It is not necessary to have an end after each function definition but it is good practice to do so (especially if there are other functions in the file).

- In Matlab a function can return more than one value.

Functions

- It is possible to call functions with fewer arguments, but not with more.

- Also, it is possible to get any number of results you want.
  - E.g. function [a, b, c] = f (x, y, z) … end

    [~, k] = f(l, m)

    This will ignore the first return value (~), store b into k after calling f with two arguments.

    Of course it is not always possible to call with fewer arguments! Most functions use all of their arguments.

Functions: Pre-Allocation

- x = zeros(s1,s2,…,sn): create a s1xs2x…xsn matrix initialized with zeros.
- x = ones(s1,s2,…,sn): create a s1xs2x…xsn matrix initialized with ones.
- s = struct('field1', values1, 'field2', values2, ...): create a struct with some fields and values.
- s = struct('field1', {}, 'field2', {}, ...): create a struct with some fields and empty values.
- c = cell(s1,s2,…,sn): create a s1xs2x…xsn empty cell array.
- (see *doc* cell, *doc* struct for more constructors)

Functions: Arrays

- size, length
- reshape, squeeze, permute, repmat
- sort, sortrows
- union, intersect, setdiff, setxor, unique
- ismember, issorted
- all, any, find
- full, sparse

Functions: Statistics / Distributions / Operations

- min, max
- mean, median, mode
- std, var, corr, cov

- normcdf, normpdf, normrnd
- chi2cdf, chi2pdf
- rand, randi

- sum, prod, cumsum, cumprod

Functions: Plots

- figure, plot, plot3, ezplot, subplot
- hist, bar
- scatter, scatter3

- hold on, hold off (used to plot multiple graphs)
- title, legend, xlabel, ylabel
- axis, xlim, ylim

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης (1)

Σημείωμα αδειοδότησης (2)

- Ως **Μη Εμπορική** ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου
    και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.
- .

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Ιωάννης Τσαμαρδίνος 2015. Γιώργος Μπορμπουδάκης «Μηχανική Μάθηση. Introduction to Matlab». Έκδοση: 1.0. Ηράκλειο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: https://opencourses.uoc.gr/courses/course/view.php?id=362.

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.