Introduction to MATLAB

Ahmad Asadi

Amirkabir University of Technology
Department of Computer Engineering and Information Technology
ahmad.asadi@aut.ac.ir

Spring 2016

Outlines

Quick Look

What we will see

- What is MATLAB?
- Look around MATLAB
- Applications
- How to work with MATLAB
- Graphical User Interface of MATLAB

What is MATLAB?

- A high level programming language being used for technical sophisticated computations
- Everything is matrix
- Stands for: **MAT**rix **LAB**oratory
- Can be assumed as a powerful super calculator
- Matrix based structure $\rightarrow$ awesome to do linear algebra

Note

Matlab is extremely broader than what we will cover in this course. We just want to understand its basics.

Look around MATLAB

Pros

- Fast and easy prototyping
- A wide variety of provided libraries including wide diversity of applications
- Great easy graphical display facilities
- Providing facilities to quickly make a little tiny application
- Quick to learn & efficient to use

Cons

- It seems slow for some sort of programs (we will see them later)
- A program that is just for personal usages (not available on web, not designed for large scale applications, not designed in a multi-user fashion, etc.)

Applications

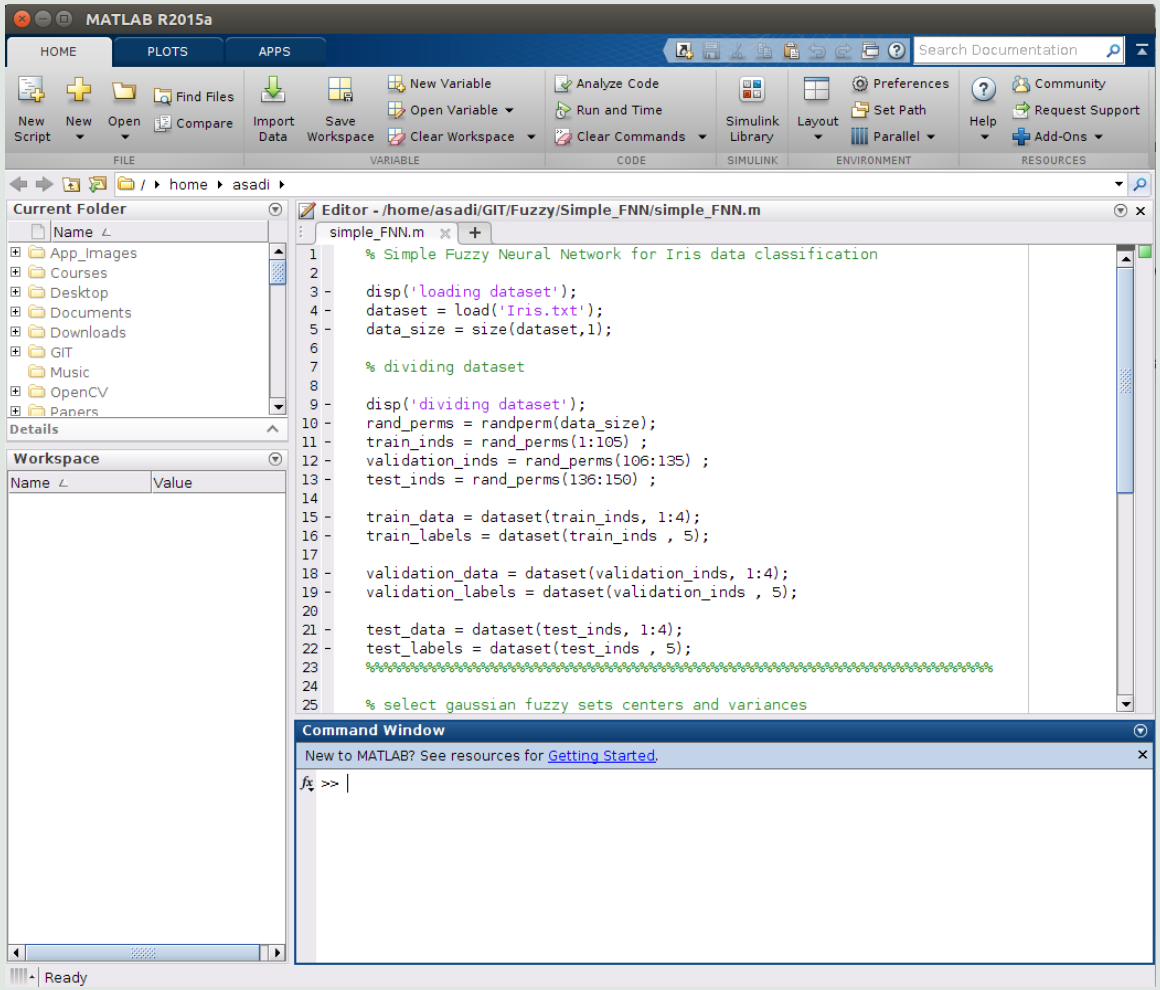
- Math and Computations
- Algorithm Development
- Modeling, Simulation and Prototyping
- Data Analysis, Exploration and Visualization
- Scientific and Engineering Graphics
- Optimized mining operations through modeling and simulation
- Automated data analysis, processing and reporting
- Forecast economical risk and profitability using financial predictive modeling
- Almost, one of the most useful handy applications for engineers and also scientists

How to work with MATLAB?

Big Picture

- Learn Rules (Syntax)
- Decompose interesting problem into simple steps
- Express each step according to MATLAB syntax
- Let MATLAB To do it!

Graphical User Interface (GUI)



Basics

What we will see

- Getting Started!
  - Hello World Again!
  - Simple Calculations
  - Hands on Variables
- Primitive Data Structures
  - Matrices and Vectors
  - Creating Special Vectors
  - Functions to create Matrices
- Operations
  - Matrix Operations
  - Array Operations
- Reading Values of Cells in Matrices and Vectors

Getting Started

Hello World Again

- Using command window
- There is commands for input/output that we will drill into later
- A handy one is
  - disp(ŕthis is a messagé) $\rightarrow$ prints the message string in command window
- The traditional first example!

```
1 >> disp('Hello World!')
2 Hello World!
```


Getting Started

Simple Calculations

- You can write any desired expression to be calculated and get its results simply in command window.

```
>> 2 + 3
ans =
5
>> sqrt((pi * 12)^2 / 3 - 57 * cos(pi/3))
ans =
21.007
```

- There exists a complete list of provided functions like *cos*() and *sqrt*() available here in MATLAB documentation: http://mathworks.com/help/matlab/functionlist.html

Hands on Variables

- Variables are named places on memory being used in order to keep a value
- Each variable has a specific data type
- There exists a list of defined data types in MATLAB documents

```
>> a = 2
a =
2
>> b = 3
b =
3
>> c = a + b
c =
5
>> (c * a * b)^2
ans =
900
```


Primitive data structures

Matrices & Vectors

- Almost the most primitive data structures in MATLAB $\rightarrow$ matrices
- Defined as bellow:

```
>> A = [1 2; 3 4]
A =   1 2
      3 4
```

- Separate rows by ';' and cols by ',' or ' '
- Vectors are special cases of matrices
  - **Row Vector** is an $N * 1$ matrix
  - **Column Vector** is a $1 * M$ matrix
- $size(A)$ returns dimensions of matrix A

Facilities in Creating Vectors

- Creating a vector with equally spaced intervals

```
>> A = 1:0.5:pi
A = 1.0000 1.5000 2.0000 2.5000 3.0000
```

- Creating a vector with *n* equally spaced intervals

```
>> A = linspace(0, pi, 7)
A = 0 0.5236 1.0472 1.5708 2.0944 2.6180 3.1416
```

Note

- MATLAB uses *pi* to represent π and *i* or *j* to represent imaginary unit

Matrices

There is still another useful slide!

There exist a list of useful functions being used to create matrices

- *zeros*(m, n) creates an $m * n$ matrix of all zeros
- *ones*(m, n) creates an $m * n$ matrix of all ones
- *eye*(m, n) creates an $m * n$ identity matrix
- *rand*(m, n) creates an $m * n$ uniformly distributed randoms
- *randn*(m, n) creates an $m * n$ normally distributed randoms
- *magic*(m) creates a square matrix with equal summation of rows, columns and diagonal
- *pascal*(m) creates a square pascal matrix

Operations

Operations on vectors and matrices are divided into two groups

1. Matrix Operations
   Operands of these kind of operations are matrices as whole.
2. Array Operations
   Operands of these kind of operations are elements of matrices. These kind of operations are being applied to matrices, element by element.

Operations

Matrix Operations

- $+ \rightarrow$ summation
- $- \rightarrow$ subtraction
- $* \rightarrow$ multiplication
- $/ \rightarrow$ division
- $\backslash \rightarrow$ left division($A \backslash B = INV(A) * B$)
- ^$\rightarrow$ exponentiation

Array Operations

- $.' \rightarrow$ array transpose
- .^$\rightarrow$ array power
- $.* \rightarrow$ array multiplication
- $./ \rightarrow$ array division

Reading values of a particle matrix

- get value of cell on row 1, col 3 of matrix A

```
28 >> A(1,2)
```

- get value of cells on row 2, from col 2 to col 5 of matrix A

```
29 >> A(3,2:5)
```

- get value of cells from row 3 to row 6 on col 3 of matrix A

```
30 >> A(3:6,3)
```

- get value of cells from row 1 to row 3, from col 2 to row 4 of matrix A

```
31 >> A(1:3,2:4)
```


Reading values of a particle matrix

- get value of all cells on row 3 of matrix A

```
32 >> A(3,:)
```

- get value of all cells on col 2 of matrix A

```
33 >> A(:,2)
```

- get value of all cells of matrix A

```
34 >> A(:,:)
```


Exercises

You are what you practice more *RichardCarlson*

1. Compute:
   $4[2, 32, 42, 55, 2]^T + (-2)[1, 3, 5, 2, -6]^T + [5, -10, 3, 5, 32]^T$
2. Compute determinant of matrix A without using any function:

$$A = \begin{bmatrix} 12 & 3323 & 411 \\ 30 & -331 & 345 \\ -12.323 & 34.653 & -34 \end{bmatrix}$$

3. Compute determinant of matrix A using $det()$ function for inner $2 * 2$ matrices
4. Do Gauss-Jordan to solve following equations:

$$\begin{cases} x + y + z = 5 \\ 2x + 3y + 5z = 8 \\ 4x + 5z = 2 \end{cases}$$